Bayesian Policy Gradient and Actor-Critic Algorithms

Mohammad Ghavamzadeh, Yaakov Engel, Michal Valko

Bayesian Policy Gradient and Actor-Critic Algorithms

Mohammad Ghavamzadeh* mohammad.ghavamzadeh@inria.fr
Adobe Research & INRIA Lille - Team SequeL
Yaakov Engel yakiengel@gmail.com
Rafael Advanced Defense System, Israel.
Michal Valko michal.valko@inria.fr
INRIA Lille - Team SequeL

Editor:

Abstract

Policy gradient methods are reinforcement learning algorithms that adapt a parameterized policy by following a performance gradient estimate. Many conventional policy gradient methods use Monte-Carlo techniques to estimate this gradient. The policy is improved by adjusting the parameters in the direction of the gradient estimate. Since Monte-Carlo methods tend to have high variance, a large number of samples is required to attain accurate estimates, resulting in slow convergence. In this paper, we first propose a Bayesian framework for policy gradient, based on modeling the policy gradient as a Gaussian process. This reduces the number of samples needed to obtain accurate gradient estimates. Moreover, estimates of the natural gradient as well as a measure of the uncertainty in the gradient estimates, namely, the gradient covariance, are provided at little extra cost. Since the proposed Bayesian framework considers system trajectories as its basic observable unit, it does not require the dynamics within trajectories to be of any particular form, and thus, can be easily extended to partially observable problems. On the downside, it cannot take advantage of the Markov property when the system is Markovian.

To address this issue, we proceed to supplement our Bayesian policy gradient framework with a new actor-critic learning model in which a Bayesian class of non-parametric critics, based on Gaussian process temporal difference learning, is used. Such critics model the action-value function as a Gaussian process, allowing Bayes' rule to be used in computing the posterior distribution over action-value functions, conditioned on the observed data. Appropriate choices of the policy parameterization and of the prior covariance (kernel) between action-values allow us to obtain closed-form expressions for the posterior distribution of the gradient of the expected return with respect to the policy parameters. We perform detailed experimental comparisons of the proposed Bayesian policy gradient and actor-critic algorithms with classic Monte-Carlo based policy gradient methods, as well as with each other, on a number of reinforcement learning problems.

Keywords: reinforcement learning, policy gradient methods, actor-critic algorithms, Bayesian inference, Gaussian processes

∗. Mohammad Ghavamzadeh is at Adobe Research, on leave of absence from INRIA Lille - Team SequeL.

1. Introduction

Policy gradient (PG) methods[1] are reinforcement learning (RL) algorithms that maintain a parameterized action-selection policy and update the policy parameters by moving them in the direction of an estimate of the gradient of a performance measure. Early examples of PG algorithms are the class of REINFORCE algorithms (Williams, 1992), which are suitable for solving problems in which the goal is to optimize the average reward. Subsequent work (e.g., Kimura et al., 1995, Marbach, 1998, Baxter and Bartlett, 2001) extended these algorithms to the cases of infinite-horizon Markov decision processes (MDPs) and partially observable MDPs (POMDPs), while also providing much needed theoretical analysis. However, both the theoretical results and empirical evaluations have highlighted a major shortcoming of these algorithms, namely, the high variance of the gradient estimates. This problem may be traced to the fact that in most interesting cases, the time-average of the observed rewards is a high-variance (although unbiased) estimator of the true average reward, resulting in the sample-inefficiency of these algorithms.

One solution proposed for this problem was to use an artificial *discount factor* in these algorithms (Marbach, 1998, Baxter and Bartlett, 2001), however, this creates another problem by introducing bias into the gradient estimates. Another solution, which does not involve biasing the gradient estimate, is to subtract a *reinforcement baseline* from the average reward estimate in the updates of PG algorithms (e.g., Williams, 1992, Marbach, 1998, Sutton et al., 2000). In Williams (1992) an average reward baseline was used, and in Sutton et al. (2000) it was conjectured that an approximate value function would be a good choice for a state-dependent baseline. However, it was shown in Weaver and Tao (2001) and Greensmith et al. (2004), perhaps surprisingly, that the mean reward is in general *not* the optimal constant baseline, and that the true value function is generally *not* the optimal state-dependent baseline.

A different approach for speeding-up PG algorithms was proposed by Kakade (2002) and refined and extended by Bagnell and Schneider (2003) and Peters et al. (2003). The idea is to replace the policy gradient estimate with an estimate of the so-called *natural* policy gradient. This is motivated by the requirement that the policy updates should be invariant to bijective transformations of the parametrization. Put more simply, a change in the way the policy is parametrized should not influence the result of the policy update. In terms of the policy update rule, the move to natural-gradient amounts to linearly transforming the gradient using the inverse Fisher information matrix of the policy. In empirical evaluations, natural PG has been shown to significantly outperform conventional PG (e.g., Kakade 2002, Bagnell and Schneider 2003, Peters et al. 2003, Peters and Schaal 2008).

Another approach for reducing the variance of policy gradient estimates, and as a result making the search in the policy-space more efficient and reliable, is to use an explicit representation for the value function of the policy. This class of PG algorithms are called actor-critic algorithms. Actor-critic (AC) algorithms comprise a family of RL methods that maintain two distinct algorithmic components: An *actor*, whose role is to maintain and update an action-selection policy; and a *critic*, whose role is to estimate the value function associated with the actor's policy. Thus, the critic addresses the problem of *prediction*,

1. The term has been coined in Sutton et al. (2000), but here we use it more liberally to refer to a whole class of reinforcement learning algorithms.

whereas the actor is concerned with *control*. Actor-critic methods were among the earliest to be investigated in RL (Barto et al., 1983, Sutton, 1984). They were largely supplanted in the 1990's by methods, such as SARSA (Rummery and Niranjan, 1994), that estimate action-value functions and use them directly to select actions without maintaining an explicit representation of the policy. This approach was appealing because of its simplicity, but when combined with function approximation was found to be unreliable, often failing to converge. These problems led to renewed interest in PG methods.

Actor-critic algorithms (e.g., Sutton et al. 2000, Konda and Tsitsiklis 2000, Peters et al. 2005, Bhatnagar et al. 2007) borrow elements from these two families of RL algorithms. Like value-function based methods, a critic maintains a value function estimate, while an actor maintains a separately parameterized stochastic action-selection policy, as in policy based methods. While the role of the actor is to select actions, the role of the critic is to evaluate the performance of the actor. This evaluation is used to provide the actor with a feedback signal that allows it to improve its performance. The actor typically updates its policy along an estimate of the gradient (or natural gradient) of some measure of performance with respect to the policy parameters. When the representations used for the actor and the critic are *compatible*, in the sense explained in Sutton et al. (2000) and Konda and Tsitsiklis (2000), the resulting AC algorithm is simple, elegant, and provably convergent (under appropriate conditions) to a local maximum of the performance measure used by the critic, plus a measure of the temporal difference (TD) error inherent in the function approximation scheme (Konda and Tsitsiklis, 2000, Bhatnagar et al., 2009).

Existing AC algorithms are based on parametric critics that are updated to optimize frequentist fitness criteria. By "frequentist" we mean algorithms that return a point estimate of the value function, rather than a complete posterior distribution computed using Bayes' rule. A Bayesian class of critics based on Gaussian processes (GPs) has been proposed by Engel et al. (2003, 2005), called Gaussian process temporal difference (GPTD). By their Bayesian nature, these algorithms return a full posterior distribution over value functions. Moreover, while these algorithms may be used to learn a parametric representation for the posterior, they are generally capable of searching for value functions in an infinite-dimensional Hilbert space of functions, resulting in a non-parametric posterior.

Both conventional and natural policy gradient and actor-critic methods rely on Monte-Carlo (MC) techniques in estimating the gradient of the performance measure. MC estimation is a frequentist procedure, and as such violates the *likelihood principle* (Berger and Wolpert, 1984).[2] Moreover, although MC estimates are unbiased, they tend to suffer from high variance, or alternatively, require excessive sample sizes (see O'Hagan, 1987 for a discussion). In the case of policy gradient estimation this is exacerbated by the fact that consistent policy improvement requires multiple gradient estimation steps.

In O'Hagan (1991) a Bayesian alternative to MC estimation is proposed.[3] The idea is to model integrals of the form $\int f(x)g(x)dx$ as GPs. This is done by treating the first term f in the integrand as a *random function*, the randomness of which reflects our subjective uncertainty concerning its true identity. This allows us to incorporate our prior knowledge

2. The likelihood principle states that in a parametric statistical model, all the information about a data sample that is required for inferring the model parameters is contained in the likelihood function of that sample.
3. O'Hagan (1991) mentions that this approach may be traced even as far back as Poincaré (1896).

of f into its prior distribution. Observing (possibly noisy) samples of f at a set of points $\{x_1, \ldots, x_M\}$ allows us to employ Bayes' rule to compute a posterior distribution of f conditioned on these samples. This, in turn, induces a posterior distribution over the value of the integral.

In this paper, we first propose a Bayesian framework for policy gradient estimation by modeling the gradient as a GP. Our Bayesian policy gradient (BPG) algorithms use GPs to define a prior distribution over the gradient of the expected return, and compute its posterior conditioned on the observed data. This reduces the number of samples needed to obtain accurate gradient estimates. Moreover, estimates of the natural gradient as well as a measure of the uncertainty in the gradient estimates, namely the gradient covariance, are provided at little extra cost. Additional gains may be attained by learning a transition model of the environment, allowing knowledge transfer between policies. Since our BPG models and algorithms consider complete system trajectories as their basic observable unit, they do not require the dynamics within each trajectory to be of any special form. In particular, it is not necessary for the dynamics to have the Markov property, allowing the resulting algorithms to handle partially observable MDPs, Markov games, and other non-Markovian systems. On the downside, BPG algorithms cannot take advantage of the Markov property when this property is satisfied. To address this issue, we supplement our BPG framework with actor-critic methods and propose an AC algorithm that incorporates GPTD as its critic. However, rather than merely plugging-in our critic into an existing AC algorithm, we show how the posterior moments returned by the GPTD critic allow us to obtain closed-form expressions for the posterior moments of the policy gradient. This is made possible by utilizing the Fisher kernel (Shawe-Taylor and Cristianini, 2004) as our prior covariance kernel for the GPTD state-action *advantage* values. Unlike the BPG methods, the Bayesian actor-critic (BAC) algorithm takes advantage of the Markov property of the system trajectories and uses individual state-action-reward transitions as its basic observable unit. This helps reduce variance in the gradient estimates, resulting in steeper learning curves when compared to BPG and the classic MC approach.

It is important to note that a short version of the two main parts of this paper, *Bayesian policy gradient* and *Bayesian actor-critic*, appeared in Ghavamzadeh and Engel (2006) and Ghavamzadeh and Engel (2007), respectively. This paper extends these conference papers in the following ways:

- We have included a discussion on using Bayesian Quadrature (BQ) for estimating vector-valued integrals to the paper. This is totally relevant to this work because the gradient of a policy (the quantity that we are interested in estimating using BQ) is a vector-valued integral when the size of the policy parameter vector is more than 1, which is usually the case. This also helps to better see the difference between the two models we propose for BPG. In Model 1, we place a vector-valued Gaussian process (GP) over a component of the gradient integrant, while in Model 2, we put a scalar-valued GP over a different component of the gradient integrant.

- We describe the BPG and BAC algorithms in more details and show the details of using online sparsification in these algorithms. Moreover, we show how BPG can be extended to partially observable Markov decision processes (POMDPs) along the same lines that the standard PG algorithms can be used in such problems.

- In comparison to Ghavamzadeh and Engel (2006), we report more details of the experiments and more experimental results, especially in using the posterior variance (covariance) of the gradient to select the step size for updating the policy parameters.
- We include all the proofs in this paper (almost none was reported in the two conference papers), in particular, the proofs of Propositions 3, 4, 5, and 6. These proofs are important and the proof techniques are novel and definitely useful for the community. The importance of these proofs come from the fact that they show how with the right choice of GP prior (the one that uses the family of Fisher information kernels), we are able to use BQ and have a Bayesian estimate of the gradient integral, while initially everything indicates that BQ cannot be used for the estimation of this integral.
- We apply the BAC algorithm to two new domains: "Mountain Car", a 2-dimensional continuous state and 1-dimensional discrete action problem, and "Ship Steering", a 4-dimensional continuous state and 1-dimensional continuous action problem.

2. Reinforcement Learning, Policy Gradient, and Actor-Critic Methods

Reinforcement learning (RL) (Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 1998) is term describing a class of learning problems in which an agent (or controller) interacts with a dynamic, stochastic, and incompletely known environment (or plant), with the goal of finding an action-selection strategy, or *policy*, to optimize some measure of its long-term performance. This interaction is conventionally modeled as a Markov decision process (MDP), or if the environmental state is not completely observable, as a partially observable MDP (POMDP) (Puterman, 1994). In this work we restrict our attention to the discrete-time MDP setting.

Let $\mathcal{P}(\mathcal{X})$, $\mathcal{P}(\mathcal{A})$, and $\mathcal{P}(\mathbb{R})$ denote the set of probability distributions on (Borel) subsets of the sets $\mathcal{X}$, $\mathcal{A}$, and $\mathbb{R}$, respectively. A MDP is a tuple $(\mathcal{X}, \mathcal{A}, q, P, P_0)$ where $\mathcal{X}$ and $\mathcal{A}$ are the state and action spaces; $q(\cdot|x,a) \in \mathcal{P}(\mathbb{R})$ and $P(\cdot|x,a) \in \mathcal{P}(\mathcal{X})$ are the probability distributions over rewards and next states when action a is taken at state x (we assume that q and P are stationary); and $P_0(\cdot) \in \mathcal{P}(\mathcal{X})$ is the probability distribution according to which the initial state is selected. We denote the random variable distributed according to $q(\cdot|x,a)$ as $r(x,a)$ and its mean as $\bar{r}(x,a)$.

In addition, we need to specify the rule according to which the agent selects an action at each possible state. We assume that this rule is *stationary*, i.e., does not depend explicitly on time. A stationary policy $\mu(\cdot|x) \in \mathcal{P}(\mathcal{A})$ is a probability distribution over actions, conditioned on the current state. A MDP controlled by a policy μ induces a Markov chain over state-action pairs $\boldsymbol{z}_t = (x_t, a_t) \in \mathcal{Z} = \mathcal{X} \times \mathcal{A}$, with a transition probability density $P^\mu(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = P(x_t|x_{t-1}, a_{t-1})\mu(a_t|x_t)$, and an initial state density $P_0^\mu(\boldsymbol{z}_0) = P_0(x_0)\mu(a_0|x_0)$. We generically denote by $\xi = (\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T) \in \Xi$, $T \in \{0, 1, \ldots, \infty\}$ a path generated by this Markov chain. Note that Ξ is the set of all possible trajectories that can be generated by the Markov chain induced by the current policy μ. The probability (density) of such a path is given by

$$\Pr(\xi;\mu) = P_0^\mu(\boldsymbol{z}_0)\prod_{t=1}^{T} P^\mu(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = P_0(x_0)\prod_{t=0}^{T-1}\mu(a_t|x_t)P(x_{t+1}|x_t,a_t). \tag{1}$$

We denote by $R(\xi) = \sum_{t=0}^{T-1} \gamma^t r(x_t, a_t)$ the (possibly discounted, $\gamma \in [0,1]$) *cumulative return* of the path ξ. $R(\xi)$ is a random variable both because the path ξ itself is a random variable, and because, even for a given path, each of the rewards sampled in it may be stochastic. The expected value of $R(\xi)$ for a given path ξ is denoted by $\bar{R}(\xi)$. Finally, we define the *expected return* of policy μ as

$$\eta(\mu) = \mathbf{E}\big[R(\xi)\big] = \int_{\Xi} \bar{R}(\xi) \Pr(\xi;\mu) d\xi. \tag{2}$$

The t-step state-action occupancy density of policy μ is given by

$$P_t^\mu(\boldsymbol{z}_t) = \int_{\mathcal{Z}^t} d\boldsymbol{z}_0 \dots d\boldsymbol{z}_{t-1} P_0^\mu(\boldsymbol{z}_0) \prod_{i=1}^{t} P^\mu(\boldsymbol{z}_i|\boldsymbol{z}_{i-1}).$$

It can be shown that under certain regularity conditions (Sutton et al., 2000), the expected return of policy μ may be written in terms of state-action pairs (rather than in terms of trajectories as in Eq. 2) as

$$\eta(\mu) = \int_{\mathcal{Z}} d\boldsymbol{z} \pi^\mu(\boldsymbol{z}) \bar{r}(\boldsymbol{z}), \tag{3}$$

where $\pi^\mu(\boldsymbol{z}) = \sum_{t=0}^{\infty} \gamma^t P_t^\mu(\boldsymbol{z})$ is a discounted weighting of state-action pairs encountered while following policy μ. Integrating a out of $\pi^\mu(\boldsymbol{z}) = \pi^\mu(x,a)$ results in the corresponding discounted weighting of states encountered by following policy μ: $\nu^\mu(x) = \int_{\mathcal{A}} da \pi^\mu(x,a)$. Unlike ν^μ and π^μ, $(1-\gamma)\nu^\mu$ and $(1-\gamma)\pi^\mu$ are distributions. They are analogous to the stationary distributions over states and state-action pairs of policy μ in the undiscounted setting, respectively, since as $\gamma \to 1$, they tend to these distributions, if they exist.

Our aim is to find a policy μ^* that maximizes the expected return, i.e., $\mu^* = \arg\max_\mu \eta(\mu)$. A policy μ is assessed according to the expected cumulative rewards associated with states x or state-action pairs $\boldsymbol{z}$. For all states $x \in \mathcal{X}$ and actions $a \in \mathcal{A}$, the action-value function and the value function of policy μ are defined as

$$Q^\mu(\boldsymbol{z}) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{z}_t)|\boldsymbol{z}_0 = \boldsymbol{z}\right] \qquad \text{and} \qquad V^\mu(x) = \int_{\mathcal{A}} da \mu(a|x) Q^\mu(x,a).$$

In policy gradient (PG) methods, we define a class of smoothly parameterized stochastic policies $\big\{\mu(\cdot|x;\boldsymbol{\theta}), x \in \mathcal{X}, \boldsymbol{\theta} \in \Theta\big\}$. We estimate the gradient of the expected return, defined by Eq. 2 (or Eq. 3), with respect to the policy parameters $\boldsymbol{\theta}$, from the observed system trajectories. We then improve the policy by adjusting the parameters in the direction of the gradient (e.g., Williams 1992, Marbach 1998, Baxter and Bartlett 2001). Since in this setting a policy μ is represented by its parameters $\boldsymbol{\theta}$, policy dependent functions such as $\eta(\mu)$, $\Pr\big(\xi;\mu\big)$, $\pi^\mu(\boldsymbol{z})$, $\nu^\mu(x)$, $V^\mu(x)$, and $Q^\mu(\boldsymbol{z})$ may be written as $\eta(\boldsymbol{\theta})$, $\Pr(\xi;\boldsymbol{\theta})$, $\pi(\boldsymbol{z};\boldsymbol{\theta})$, $\nu(x;\boldsymbol{\theta})$, $V(x;\boldsymbol{\theta})$, and $Q(\boldsymbol{z};\boldsymbol{\theta})$, respectively. We assume

Assumption 1 (Differentiability) *For any state-action pair (x,a) and any policy parameter $\boldsymbol{\theta} \in \Theta$, the policy $\mu(a|x;\boldsymbol{\theta})$ is continuously differentiable in the parameters $\boldsymbol{\theta}$.*

The *score function* or *likelihood ratio* method has become the most prominent technique for gradient estimation from simulation. It has been first proposed in the 1960's (Aleksandrov

et al., 1968, Rubinstein, 1969) for computing performance gradients in i.i.d. (independently and identically distributed) processes, and was then extended to *regenerative* processes including MDPs by Glynn (1986, 1990), Reiman and Weiss (1986, 1989), Glynn and L'Ecuyer (1995), and to episodic MDPs by Williams (1992). This method estimates the gradient of the expected return with respect to the policy parameters $\boldsymbol{\theta}$, defined by Eq. 2, using the following equation:[4]

$$\nabla\eta(\boldsymbol{\theta}) = \int \bar{R}(\xi)\frac{\nabla \Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})}\Pr(\xi;\boldsymbol{\theta})d\xi. \tag{4}$$

In Eq. 4, the quantity $\frac{\nabla \Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})} = \nabla \log \Pr(\xi;\boldsymbol{\theta})$ is called the (Fisher) score function or likelihood ratio. Since the initial-state distribution P_0 and the state-transition distribution P are independent of the policy parameters $\boldsymbol{\theta}$, we may write the score function for a path ξ using Eq. 1 as[5]

$$\boldsymbol{u}(\xi;\boldsymbol{\theta}) = \nabla \log \Pr(\xi;\boldsymbol{\theta}) = \frac{\nabla \Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})} = \sum_{t=0}^{T-1}\frac{\nabla\mu(a_t|x_t;\boldsymbol{\theta})}{\mu(a_t|x_t;\boldsymbol{\theta})} = \sum_{t=0}^{T-1}\nabla \log \mu(a_t|x_t;\boldsymbol{\theta}). \tag{5}$$

Previous work on policy gradient used classical MC to estimate the gradient in Eq. 4. These methods generate i.i.d. sample paths $\xi_1,\ldots,\xi_M$ according to $\Pr(\xi;\boldsymbol{\theta})$, and estimate the gradient $\nabla\eta(\boldsymbol{\theta})$ using the MC estimator

$$\widehat{\nabla\eta}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} R(\xi_i)\nabla \log \Pr(\xi_i;\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} R(\xi_i)\sum_{t=0}^{T_i-1}\nabla \log \mu(a_{t,i}|x_{t,i};\boldsymbol{\theta}). \tag{6}$$

This is an unbiased estimate, and therefore, by the law of large numbers, $\widehat{\nabla\eta}(\boldsymbol{\theta}) \to \nabla\eta(\boldsymbol{\theta})$ as M goes to infinity, with probability one.

The policy gradient theorem (Marbach, 1998, Proposition 1; Sutton et al., 2000, Theorem 1; Konda and Tsitsiklis, 2000, Theorem 1) states that the gradient of the expected return, defined by Eq. 3, for parameterized policies satisfying Assumption 1 is given by

$$\nabla\eta(\boldsymbol{\theta}) = \int dxda\ \nu(x;\boldsymbol{\theta})\nabla\mu(a|x;\boldsymbol{\theta})Q(x,a;\boldsymbol{\theta}). \tag{7}$$

Observe that if $b : \mathcal{X} \to \mathbb{R}$ is an arbitrary function of x (also called a *baseline*), then

$$\int_{\mathcal{Z}} dxda\ \nu(x;\boldsymbol{\theta})\nabla\mu(a|x;\boldsymbol{\theta})b(x) = \int_{\mathcal{X}} dx\ \nu(x;\boldsymbol{\theta})b(x)\nabla\Big(\int_{\mathcal{A}} da\mu(a|x;\boldsymbol{\theta})\Big) = \int_{\mathcal{X}} dx\ \nu(x;\boldsymbol{\theta})b(x)\nabla(1) = 0,$$

and thus, for any baseline $b(x)$, the gradient of the expected return can be written as

$$\nabla\eta(\boldsymbol{\theta}) = \int dxda\ \nu(x;\boldsymbol{\theta})\nabla\mu(a|x;\boldsymbol{\theta})\big(Q(x,a;\boldsymbol{\theta})+b(x)\big) = \int_{\mathcal{Z}} d\boldsymbol{z}\pi(\boldsymbol{z};\boldsymbol{\theta})\nabla \log \mu(a|x;\boldsymbol{\theta})\big(Q(\boldsymbol{z};\boldsymbol{\theta})+b(x)\big). \tag{8}$$

4. Throughout the paper, we use the notation ∇ to denote $\nabla_{\boldsymbol{\theta}}$ – the gradient w.r.t. the policy parameters.
5. To simplify notation, we omit $\boldsymbol{u}$'s dependence on the policy parameters $\boldsymbol{\theta}$, and denote $\boldsymbol{u}(\xi;\boldsymbol{\theta})$ as $\boldsymbol{u}(\xi)$ in the sequel.

The baseline may be chosen in such a way so as to minimize the variance of the gradient estimates (Greensmith et al., 2004).

Now consider the actor-critic (AC) framework in which the action-value function for a fixed policy μ, Q^μ, is approximated by a learned function approximator. If the approximation is sufficiently good, we may hope to use it in place of Q^μ in Eqs. 7 and 8, and still point roughly in the direction of the true gradient. Sutton et al. (2000) and Konda and Tsitsiklis (2000) showed that if the approximation $\hat{Q}^\mu(\cdot;\boldsymbol{w})$ with parameter $\boldsymbol{w}$ is *compatible*, i.e., $\nabla_{\boldsymbol{w}}\hat{Q}^\mu(x,a;\boldsymbol{w}) = \nabla \log \mu(a|x;\boldsymbol{\theta})$, and if it minimizes the mean squared error

$$\mathcal{E}^\mu(\boldsymbol{w}) = \int_{\mathcal{Z}} d\boldsymbol{z}\pi^\mu(\boldsymbol{z})\big[Q^\mu(\boldsymbol{z}) - \hat{Q}^\mu(\boldsymbol{z};\boldsymbol{w})\big]^2 \tag{9}$$

for parameter value $\boldsymbol{w}^*$, then we may replace Q^μ with $\hat{Q}^\mu(\cdot;\boldsymbol{w}^*)$ in Eqs. 7 and 8. The second condition means that $\hat{Q}^\mu(\cdot;\boldsymbol{w}^*)$ is the projection of Q^μ onto the space $\{\hat{Q}^\mu(\cdot;\boldsymbol{w})|\boldsymbol{w}\in\mathbb{R}^n\}$, with respect to a ℓ_2-norm weighted by π^μ.

An approximation for the action-value function, in terms of a linear combination of basis functions, may be written as $\hat{Q}^\mu(\boldsymbol{z};\boldsymbol{w}) = \boldsymbol{w}^\top\boldsymbol{\psi}(\boldsymbol{z})$. This approximation is compatible if the $\boldsymbol{\psi}$'s are compatible with the policy, i.e., $\boldsymbol{\psi}(\boldsymbol{z};\boldsymbol{\theta}) = \nabla \log \mu(a|x;\boldsymbol{\theta})$. Note that compatibility is well defined under Assumption 1. Let $\mathcal{E}^\mu(\boldsymbol{w})$ denote the mean squared error

$$\mathcal{E}^\mu(\boldsymbol{w}) = \int_{\mathcal{Z}} d\boldsymbol{z}\pi^\mu(\boldsymbol{z})\big[Q^\mu(\boldsymbol{z}) - \boldsymbol{w}^\top\boldsymbol{\psi}(\boldsymbol{z}) - b(x)\big]^2 \tag{10}$$

of our compatible linear approximation $\boldsymbol{w}^\top\boldsymbol{\psi}(\boldsymbol{z})$ and an arbitrary baseline $b(x)$. Let $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathcal{E}^\mu(\boldsymbol{w})$ denote the optimal parameter. It can be shown that the value of $\boldsymbol{w}^*$ does not depend on the baseline $b(x)$. As a result, the mean squared-error problems of Eqs. 9 and 10 have the same solutions (see e.g., Bhatnagar et al. 2007, 2009). It can also be shown that if the parameter $\boldsymbol{w}$ is set to be equal to $\boldsymbol{w}^*$, then the resulting mean squared error $\mathcal{E}^\mu(\boldsymbol{w}^*)$, now treated as a function of the baseline $b(x)$, is further minimized by setting $b(x) = V^\mu(x)$ (Bhatnagar et al., 2007, 2009). In other words, the variance in the action-value function estimator is minimized if the baseline is chosen to be the value function itself.

A convenient and rather flexible choice for a space of policies that ensures compatibility between the policy and the action-value representation is a parametric exponential family

$$\mu(a|x;\boldsymbol{\theta}) = \frac{1}{Z_{\boldsymbol{\theta}}(x)}\exp\Big(\boldsymbol{\theta}^\top\boldsymbol{\phi}(x,a)\Big),$$

where $Z_{\boldsymbol{\theta}}(x) = \int_{\mathcal{A}} da \exp\big(\boldsymbol{\theta}^\top\boldsymbol{\phi}(x,a)\big)$ is a normalizing factor, referred to as the *partition function*. It is easy to show that $\boldsymbol{\psi}(\boldsymbol{z}) = \boldsymbol{\phi}(\boldsymbol{z}) - \mathbf{E}_{a|x}[\boldsymbol{\phi}(\boldsymbol{z})]$, where $\mathbf{E}_{a|x}[\cdot] = \int_{\mathcal{A}} da\mu(a|x;\boldsymbol{\theta})[\cdot]$, and as a result, $\hat{Q}^\mu(\boldsymbol{z};\boldsymbol{w}^*) = \boldsymbol{w}^{*\top}\big(\boldsymbol{\phi}(\boldsymbol{z}) - \mathbf{E}_{a|x}[\boldsymbol{\phi}(\boldsymbol{z})]\big) + b(x)$ is a compatible action-value function for this family of policies. Note that $\mathbf{E}_{a|x}[\hat{Q}(\boldsymbol{z};\boldsymbol{w}^*)] = b(x)$, since $\mathbf{E}_{a|x}\big[\boldsymbol{\phi}(\boldsymbol{z}) - \mathbf{E}_{a|x}[\boldsymbol{\phi}(\boldsymbol{z})]\big] = 0$. This means that if $\hat{Q}^\mu(\boldsymbol{z};\boldsymbol{w}^*)$ approximates $Q^\mu(\boldsymbol{z})$, then $b(x)$ must approximate the value function $V^\mu(x)$. The term $\hat{A}^\mu(\boldsymbol{z};\boldsymbol{w}^*) = \hat{Q}^\mu(\boldsymbol{z};\boldsymbol{w}^*) - b(x) = \boldsymbol{w}^{*\top}\big(\boldsymbol{\phi}(\boldsymbol{z}) - \mathbf{E}_{a|x}[\boldsymbol{\phi}(\boldsymbol{z})]\big)$ approximates the *advantage function* $A^\mu(\boldsymbol{z}) = Q^\mu(\boldsymbol{z}) - V^\mu(x)$ (Baird, 1993).

3. Bayesian Quadrature

Bayesian quadrature (BQ) (O'Hagan, 1991) is, as its name suggests, a Bayesian method for evaluating an integral using samples of its integrand. We consider the problem of evaluating the integral

$$\rho = \int f(x)g(x)dx. \tag{11}$$

If $g(x)$ is a probability density function, i.e., $g(x) = p(x)$, this becomes the problem of evaluating the expected value of $f(x)$. A well known frequentist approach to evaluating such expectations is the Monte-Carlo (MC) method. For MC estimation of such expectations, it is typically required that samples $x_1, x_2, \ldots, x_M$ are drawn from $p(x)$.[6] The integral in Eq. 11 is then estimated as

$$\hat{\rho}_{MC} = \frac{1}{M}\sum_{i=1}^{M} f(x_i). \tag{12}$$

It is easy to show that $\hat{\rho}_{MC}$ is an unbiased estimate of ρ, with variance that diminishes to zero as $M \to \infty$. However, as O'Hagan (1987) points out, MC estimation is fundamentally unsound, as it violates the likelihood principle, and moreover, does not make full use of the data at hand. The alternative proposed in O'Hagan (1991) is based on the following reasoning: In the Bayesian approach, $f(\cdot)$ is random simply because it is unknown. We are therefore uncertain about the value of $f(x)$ until we actually evaluate it. In fact, even then, our uncertainty is not always completely removed, since measured samples of $f(x)$ may be corrupted by noise. Modeling f as a Gaussian process (GP) means that our uncertainty is completely accounted for by specifying a Normal prior distribution over functions. This prior distribution is specified by its mean and covariance, and is denoted by $f(\cdot) \sim \mathcal{N}\big(\bar{f}(\cdot), k(\cdot,\cdot)\big)$. This is shorthand for the statement that f is a GP with prior mean and covariance

$$\mathbf{E}\big[f(x)\big] = \bar{f}(x) \quad \text{and} \quad \mathbf{Cov}\big[f(x), f(x')\big] = k(x,x'), \quad \forall x, x' \in \mathcal{X}, \tag{13}$$

respectively. The choice of kernel function k allows us to incorporate prior knowledge on the smoothness properties of the integrand into the estimation procedure. When we are provided with a set of samples $\mathcal{D}_M = \big\{\big(x_i, y(x_i)\big)\big\}_{i=1}^{M}$, where $y(x_i)$ is a (possibly noisy) sample of $f(x_i)$, we apply Bayes' rule to condition the prior on these sampled values. If the measurement noise is normally distributed, the result is a Normal posterior distribution of $f|\mathcal{D}_M$. The expressions for the posterior mean and covariance are standard:

$$\begin{aligned} \mathbf{E}\big[f(x)|\mathcal{D}_M\big] &= \bar{f}(x) + \boldsymbol{k}(x)^\top \boldsymbol{C}(\boldsymbol{y} - \bar{\boldsymbol{f}}), \\ \mathbf{Cov}\big[f(x), f(x')|\mathcal{D}_M\big] &= k(x,x') - \boldsymbol{k}(x)^\top \boldsymbol{C}\boldsymbol{k}(x'). \end{aligned} \tag{14}$$

Here and in the sequel, we make use of the definitions:

$$\begin{aligned} &\bar{\boldsymbol{f}} = \big(\bar{f}(x_1), \ldots, \bar{f}(x_M)\big)^\top, && && \boldsymbol{k}(x) = \big(k(x_1,x), \ldots, k(x_M,x)\big)^\top, \\ &\boldsymbol{y} = \big(y(x_1), \ldots, y(x_M)\big)^\top, && [\boldsymbol{K}]_{i,j} = k(x_i,x_j), && \boldsymbol{C} = (\boldsymbol{K} + \boldsymbol{\Sigma})^{-1}, \end{aligned}$$

where $\boldsymbol{K}$ is the kernel (or Gram) matrix, and $[\boldsymbol{\Sigma}]_{i,j}$ is the measurement noise covariance between the ith and jth samples. It is typically assumed that the measurement noise is

6. If samples are drawn from some other distribution, importance sampling variants of MC may be used.

i.i.d., in which case $\mathbf{\Sigma} = \sigma^2 \boldsymbol{I}$, where σ^2 is the noise variance and $\boldsymbol{I}$ is the (appropriately sized - here $M \times M$) identity matrix.

Since integration is a linear operation, the posterior distribution of the integral in Eq. 11 is also Gaussian, and the posterior moments are given by (O'Hagan, 1991)

$$\begin{aligned} \mathbf{E}[\rho|\mathcal{D}_M] &= \int \mathbf{E}\big[f(x)|\mathcal{D}_M\big] g(x) dx, \\ \mathbf{Var}[\rho|\mathcal{D}_M] &= \iint \mathbf{Cov}\big[f(x), f(x')|\mathcal{D}_M\big] g(x) g(x') dx dx'. \end{aligned} \tag{15}$$

Substituting Eq. 14 into Eq. 15, we obtain

$$\mathbf{E}[\rho|\mathcal{D}_M] = \rho_0 + \boldsymbol{b}^\top \boldsymbol{C}(\boldsymbol{y} - \bar{\boldsymbol{f}}) \qquad \text{and} \qquad \mathbf{Var}[\rho|\mathcal{D}_M] = b_0 - \boldsymbol{b}^\top \boldsymbol{C} \boldsymbol{b},$$

where we made use of the definitions:

$$\rho_0 = \int \bar{f}(x) g(x) dx \;, \qquad \boldsymbol{b} = \int \boldsymbol{k}(x) g(x) dx \;, \qquad b_0 = \iint k(x, x') g(x) g(x') dx dx'. \tag{16}$$

Note that ρ_0 and b_0 are the prior mean and variance of ρ, respectively.

Rasmussen and Ghahramani (2003) experimentally demonstrated how this approach, when applied to the evaluation of an expectation, can outperform MC estimation by orders of magnitude in terms of the mean-squared error.

In order to prevent the problem from "degenerating into infinite regress", as phrased by O'Hagan (1991),[7] we should choose the functions g, k, and $\bar{f}$ so as to allow us to solve the integrals in Eq. 16 analytically. For example, O'Hagan provides the analysis required for the case where the integrands in Eq. 16 are products of multivariate Gaussians and polynomials, referred to as Bayes-Hermite quadrature. One of the contributions of our work is in providing analogous analysis for kernel functions that are based on the *Fisher kernel* (Jaakkola and Haussler, 1999, Shawe-Taylor and Cristianini, 2004).

It is important to note that in MC estimation, samples must be drawn from the distribution $p(x) = g(x)$, whereas in the Bayesian approach, samples may be drawn from arbitrary distributions. This affords us with flexibility in the choice of sample points, allowing us, for instance, to actively design the samples $x_1, \ldots, x_M$ so as to maximize information gain.

3.1 Vector-Valued Integrals

O'Hagan (1991) treated the case where the integral to be estimated is a scalar-valued integral. However, in the context of our PG method, it is useful to consider vector-valued integrals, such as the gradient of the expected return with respect to the policy parameters, which we shall study in Section 4. In the BQ framework, an integral of the form in Eq. 11 may be vector-valued for one of two possible reasons: either f is a vector-valued GP and

7. What O'Hagan means by "degenerating into infinite regress" is simply that if we cannot compute the posterior integrals of Eq. 16 analytically, then we have started with estimating one integral (Eq. 11) and ended up with three (Eq. 16), and if we repeat this process, this can go forever and leave us with infinite integrals to evaluate. Therefore, for Bayesian MC to work, it is crucial to be able to analytically calculate the posterior integrals, and this can be achieved through the way we divide the integrant into two parts and the way we select the kernel function.

g is a scalar-valued function, or f is a scalar-valued GP and g is a vector-valued function. These two possibilities correspond to two very different data-generation models. In the first of these, an n-valued function $f(\cdot) = \big(f_1(\cdot), \ldots, f_n(\cdot)\big)^\top$ is sampled from the GP distribution of f. This distribution may include correlations between different components of f. Hence, in general, to specify the GP prior distribution, one needs to specify not only the covariance kernel of each component j of f, $k_{j,j}(x, x') = \mathbf{Cov}\big[f_j(x), f_j(x')\big]$, but also cross-covariance kernels for pairs of different components, $k_{j,\ell}(x, x') = \mathbf{Cov}\big[f_j(x), f_\ell(x')\big]$. Thus, instead of a single kernel function, we now need to specify a matrix of kernel functions.[8] Similarly, we also need to specify a vector of prior means, consisting of a function for each component: $\bar{f}_j(x) = \mathbf{E}\big[f_j(x)\big]$. The distribution of the measurement noise added to $f(x)$ to produce $y(x)$ may also include correlations, requiring us to specify an array of noise covariance matrices $\mathbf{\Sigma}_{j,\ell}$. As we show below, the GP posterior distribution is also specified in similar terms.

In the second model, a scalar-valued function is sampled from the GP prior distribution, which is specified by a single prior mean function and a single prior covariance-kernel function. Gaussian noise may be added, and the result is then multiplied by each of the components of the n-valued function g to produce the integrand. This model is significantly simpler, both conceptually and in terms of the number of parameters required to specify it. To see how a model of the first kind may arise, consider the following example.

Example 1 *Let $\rho(\boldsymbol{\theta}) = \int f(x; \boldsymbol{\theta}) g(x) dx$, where f is a scalar GP, parameterized by a vector of parameters $\boldsymbol{\theta}$. Its prior mean and covariance functions must therefore depend on $\boldsymbol{\theta}$. We denote these dependencies by writing:*

$$\mathbf{E}\big[f(x; \boldsymbol{\theta})\big] = \bar{f}(x; \boldsymbol{\theta}), \quad \mathbf{Cov}\big[f(x; \boldsymbol{\theta}), f(x'; \boldsymbol{\theta})\big] = k(x, x'; \boldsymbol{\theta}), \quad \forall x, x' \in \mathcal{X}.$$

We choose $\bar{f}(x; \boldsymbol{\theta})$ and $k(x, x'; \boldsymbol{\theta})$ so as to be once and twice differentiable in $\boldsymbol{\theta}$, respectively. Suppose now that we are not interested in estimating $\rho(\boldsymbol{\theta})$, but rather in its gradient with respect to the parameters $\boldsymbol{\theta}$: $\nabla_{\boldsymbol{\theta}} \rho(\boldsymbol{\theta}) = \int \nabla_{\boldsymbol{\theta}} f(x; \boldsymbol{\theta}) g(x) dx$. It may be easily verified that the mean functions and covariance kernels of the vector-valued GP $\nabla_{\boldsymbol{\theta}} f(x; \boldsymbol{\theta})$ are given by

$$\mathbf{E}\big[\nabla_{\boldsymbol{\theta}} f(x; \boldsymbol{\theta})\big] = \nabla_{\boldsymbol{\theta}} \bar{f}(x; \boldsymbol{\theta}) \quad \text{and} \quad \mathbf{Cov}\big[\partial_{\theta_j} f(x; \boldsymbol{\theta}), \partial_{\theta_\ell} f(x'; \boldsymbol{\theta})\big] = \partial_{\theta_j} \partial_{\theta_\ell} k(x, x'; \boldsymbol{\theta}),$$

where ∂_{θ_j} denotes the jth component of $\nabla_{\boldsymbol{\theta}}$. □

Propositions 1 and 2 specify the form taken by the mean and covariance functions of the integral GP under the two models discussed above.

Proposition 1 (Vector-valued GP) *Let f be an n-valued GP with mean functions $\bar{f}_j(x) = \mathbf{E}\big[f_j(x)\big]$ and covariance functions $k_{j,\ell}(x, x') = \mathbf{Cov}\big[f_j(x), f_\ell(x')\big]$, $\forall j, \ell \in \{1, \ldots, n\}$, and let g be a scalar-valued function. Then, the mean and covariance of ρ defined by Eq. 11 are of the following form:*

$$\mathbf{E}\big[\rho_j\big] = \int \bar{f}_j(x) g(x) dx, \quad \mathbf{Cov}\big[\rho_j, \rho_\ell\big] = \iint k_{j,\ell}(x, x') g(x) g(x') dx dx', \quad \forall j, \ell \in \{1, \ldots, n\}.$$

8. Note that to satisfy the symmetry property of the covariance operator, we require that $k_{j,\ell}(x, x') = k_{\ell,j}(x', x) = k_{\ell,j}(x, x')$, for all $x, x' \in \mathcal{X}$ and $j, \ell \in \{1, \ldots, n\}$.

Proposition 2 (Scalar-valued GP) *Let f be a scalar-valued GP with mean function $\bar{f}(x) = \mathbf{E}\big[f(x)\big]$ and covariance function $k(x,x') = \mathbf{Cov}\big[f(x), f(x')\big]$, and let g be an n-valued function. Then, the mean and covariance of ρ defined by Eq. 11 are of the following form:*

$$\mathbf{E}\big[\rho_j\big] = \int \bar{f}(x) g_j(x) dx, \quad \mathbf{Cov}\big[\rho_j, \rho_\ell\big] = \iint k(x,x') g_j(x) g_\ell(x') dx dx', \quad \forall j, \ell \in \{1, \ldots, n\}.$$

The proofs of these two propositions follow straightforwardly from the definition of the covariance operator in terms of expectations, and the order-exchangeability of GP expectations and integration with respect to x.

To wrap things up, we need to describe the form taken by the posterior moments of f in the vector-valued GP case. Using the standard Gaussian conditioning formulas, it is straightforward to show that

$$\begin{aligned}
\mathbf{E}\big[f_j(x)|\mathcal{D}_M\big] &= \bar{f}_j(x) + \sum_{m,m'} \boldsymbol{k}_{j,m}(x)^\top \boldsymbol{C}_{m,m'} (\boldsymbol{y}_{m'} - \bar{\boldsymbol{f}}_{m'}), \\
\mathbf{Cov}\big[f_j(x), f_\ell(x')|\mathcal{D}_M\big] &= k_{j,\ell}(x,x') - \sum_{m,m'} \boldsymbol{k}_{j,m}(x)^\top \boldsymbol{C}_{m,m'} \boldsymbol{k}_{m',\ell}(x'),
\end{aligned} \tag{17}$$

where

$$\begin{aligned}
\bar{\boldsymbol{f}}_j &= \big(\bar{f}_j(x_1), \ldots, \bar{f}_j(x_M)\big)^\top, & \boldsymbol{k}_{j,\ell}(x) &= \big(k_{j,\ell}(x_1, x), \ldots, k_{j,\ell}(x_M, x)\big)^\top, \\
\boldsymbol{y}_\ell &= \big(y_\ell(x_1), \ldots, y_\ell(x_M)\big)^\top, & [\boldsymbol{K}_{j,\ell}]_{i,i'} &= k_{j,\ell}(x_i, x_{i'}),
\end{aligned}$$

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_{1,1} & \ldots & \boldsymbol{K}_{1,n} \\ \vdots & & \vdots \\ \boldsymbol{K}_{n,1} & \ldots & \boldsymbol{K}_{n,n} \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{1,1} & \ldots & \boldsymbol{\Sigma}_{1,n} \\ \vdots & & \vdots \\ \boldsymbol{\Sigma}_{n,1} & \ldots & \boldsymbol{\Sigma}_{n,n} \end{bmatrix}, \quad \boldsymbol{C} = (\boldsymbol{K} + \boldsymbol{\Sigma})^{-1},$$

and $\boldsymbol{C}_{j,\ell}$ is the (j,ℓ)th $M \times M$ block of $\boldsymbol{C}$. The posterior moments of f, given in Eq. 17, may now be substituted into the expressions for the moments of the integral ρ, given in Proposition 1, to produce the posterior moments of ρ in the vector-valued GP case.

Clearly, models of the first type (vector-valued GP) are potentially richer and more complex than models of the second type (scalar-valued GP), as the latter only requires us to define a single prior mean function, a single prior covariance kernel function, and a single noise covariance matrix; while the former requires us to define n prior mean functions, and $n(n+1)/2$ prior covariance kernel functions and noise covariance matrices. One way to simplify the first type of models is to define a single prior mean function $\bar{f}$, a single prior covariance kernel function k, and to postulate that $\bar{f}_j(x) = \bar{f}(x)$, $k_{j,\ell}(x,x') = \delta_{j,\ell}\, k(x,x')$, and $\boldsymbol{\Sigma}_{j,\ell} = \delta_{j,\ell}\, \boldsymbol{\Sigma}$, $\forall j, \ell \in \{1, \ldots, n\}$, where δ denotes the Kronecker delta function. Applying these simplifying assumptions to the expressions for the posterior moments (Eq. 17) results in a complete decoupling between the posterior moments for the different components of f, and consequently a decoupling between the components of the integral ρ as well, since Eq. 17 becomes

$$\begin{aligned}
\mathbf{E}\big[f_j(x)|\mathcal{D}_M\big] &= \bar{f}_j(x) + \boldsymbol{k}_{j,j}(x)^\top \boldsymbol{C}_{j,j} (\boldsymbol{y}_j - \bar{\boldsymbol{f}}_j), \\
\mathbf{Cov}\big[f_j(x), f_\ell(x')|\mathcal{D}_M\big] &= \delta_{j,\ell} \left(k_{j,j}(x,x') - \boldsymbol{k}_{j,j}(x)^\top \boldsymbol{C}_{j,\ell} \boldsymbol{k}_{\ell,\ell}(x') \right),
\end{aligned} \tag{18}$$

where $\boldsymbol{C}_{j,\ell} = \delta_{j,\ell}\left(\boldsymbol{K}_{j,\ell} + \boldsymbol{\Sigma}_{j,\ell}\right)^{-1}$. Note that all the terms in Eq. 18, except $\boldsymbol{y}_j$, do not depend on the indices j and ℓ. In other words, these simplifying assumptions amount to assuming that ρ is a vector of n independent integrals, each of which may be estimated individually as

$$\begin{aligned}\mathbf{E}\big[f_j(x)|\mathcal{D}_M\big] &= \bar{f}(x) + \boldsymbol{k}(x)^\top \boldsymbol{C}(\boldsymbol{y}_j - \bar{\boldsymbol{f}}),\\ \mathbf{Cov}\big[f_j(x), f_\ell(x')|\mathcal{D}_M\big] &= \delta_{j,\ell}\left(k(x,x') - \boldsymbol{k}(x)^\top \boldsymbol{C}\boldsymbol{k}(x')\right),\end{aligned}$$

where $\boldsymbol{C} = (\boldsymbol{K} + \boldsymbol{\Sigma})^{-1}$. It should, however, be kept in mind that ignoring correlations between the components of f, when such correlations exist, may result in suboptimal use of the available data (see Rasmussen and Williams, 2006, Chapter 9 for references on GP regression with multiple outputs).

4. Bayesian Policy Gradient

In this section, we use vector-valued Bayesian quadrature to estimate the gradient of the expected return with respect to the policy parameters, allowing us to propose new *Bayesian policy gradient* (BPG) algorithms. In the frequentist approach to policy gradient, the performance measure used is $\eta(\boldsymbol{\theta}) = \int \bar{R}(\xi)\Pr(\xi;\boldsymbol{\theta})d\xi$ (Eq. 2). In order to serve as a useful performance measure, it has to be a deterministic function of the policy parameters $\boldsymbol{\theta}$. This is achieved by averaging the cumulative return $R(\xi)$ over all possible paths ξ and all possible returns accumulated in each path. In the Bayesian approach we have an additional source of randomness, namely, our subjective Bayesian uncertainty concerning the process generating the cumulative return. Let us denote

$$\eta_B(\boldsymbol{\theta}) = \int \bar{R}(\xi)\Pr(\xi;\boldsymbol{\theta})d\xi, \tag{19}$$

where $\eta_B(\boldsymbol{\theta})$ is a random variable because of the Bayesian uncertainty. Under the quadratic loss, the optimal Bayesian performance measure is the posterior expected value of $\eta_B(\boldsymbol{\theta})$, $\mathbf{E}\big[\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big]$. However, since we are interested in optimizing the performance rather than evaluating it,[9] we would rather evaluate the posterior distribution of the *gradient* of $\eta_B(\boldsymbol{\theta})$ with respect to the policy parameters $\boldsymbol{\theta}$. The posterior mean of the gradient is [10]

$$\nabla\mathbf{E}\big[\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big] = \mathbf{E}\big[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big] = \mathbf{E}\left[\int R(\xi)\frac{\nabla\Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})}\Pr(\xi;\boldsymbol{\theta})d\xi\Big|\mathcal{D}_M\right]. \tag{20}$$

Consequently, in BPG we cast the problem of estimating the gradient of the expected return (Eq. 20) in the form of Eq. 11. As described in Section 3, we need to partition the integrand into two parts, $f(\xi;\boldsymbol{\theta})$ and $g(\xi;\boldsymbol{\theta})$. We will model f as a GP and assume that g is a function known to us. We will then proceed by calculating the posterior moments of the gradient $\nabla\eta_B(\boldsymbol{\theta})$ conditioned on the observed data. Because in general, $R(\xi)$ cannot be known exactly, even for a given ξ (due to the stochasticity of the rewards), $R(\xi)$ should

9. Although evaluating the posterior distribution of performance is an interesting question in its own right.
10. We may interchange the order of the gradient and the expectation operators for the mean, $\nabla\mathbf{E}\big[\eta_B(\boldsymbol{\theta})\big] = \mathbf{E}\big[\nabla\eta_B(\boldsymbol{\theta})\big]$, but the same is not true for the variance, namely, $\nabla\mathbf{Var}\big[\eta_B(\boldsymbol{\theta})\big] \neq \mathbf{Cov}\big[\nabla\eta_B(\boldsymbol{\theta})\big]$.

always belong to the GP part of the model, i.e., $f(\xi;\boldsymbol{\theta})$. Interestingly, in certain cases it is sufficient to know the Fisher information matrix corresponding to $\Pr(\xi;\boldsymbol{\theta})$, rather than having exact knowledge of $\Pr(\xi;\boldsymbol{\theta})$ itself. We make use of this fact in the sequel. In the next two sections, we investigate two different ways of partitioning the integrand in Eq. 20, resulting in two distinct Bayesian policy gradient models.

4.1 Model 1 – Vector-Valued GP

In our first model, we define g and f as follows:

$$g(\xi;\boldsymbol{\theta}) = \Pr(\xi;\boldsymbol{\theta}) \quad , \quad f(\xi;\boldsymbol{\theta}) = \bar{R}(\xi)\frac{\nabla \Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})} = \bar{R}(\xi)\nabla \log \Pr(\xi;\boldsymbol{\theta}).$$

We place a vector-valued GP prior over $f(\xi;\boldsymbol{\theta})$ which induces a GP prior over the corresponding noisy measurement $y(\xi;\boldsymbol{\theta}) = R(\xi)\nabla \log \Pr(\xi;\boldsymbol{\theta})$. We adopt the simplifying assumptions discussed at the end of Section 3.1: We assume that each component of $f(\xi;\boldsymbol{\theta})$ may be evaluated independently of all other components, and use the same kernel function and noise covariance for all components of $f(\xi;\boldsymbol{\theta})$. We therefore omit the component index j from $\boldsymbol{K}_{j,j}$, $\boldsymbol{\Sigma}_{j,j}$ and $\boldsymbol{C}_{j,j}$, denoting them simply as $\boldsymbol{K}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{C}$, respectively. Hence, for the jth component of f and y we have, a priori

$$\begin{aligned}\boldsymbol{f}_j &= \big(f_j(\xi_1;\boldsymbol{\theta}),\ldots,f_j(\xi_M;\boldsymbol{\theta})\big)^\top \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{K}),\\ \boldsymbol{y}_j &= \big(y_j(\xi_1;\boldsymbol{\theta}),\ldots,y_j(\xi_M;\boldsymbol{\theta})\big)^\top \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{K}+\boldsymbol{\Sigma}).\end{aligned}$$

In this vector-valued GP model, the posterior mean and covariance of $\nabla\eta_B(\boldsymbol{\theta})$ are

$$\mathbf{E}\big[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big] = \boldsymbol{Y}\boldsymbol{C}\boldsymbol{b} \qquad \text{and} \qquad \mathbf{Cov}\big[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big] = \Big(b_0 - \boldsymbol{b}^\top\boldsymbol{C}\boldsymbol{b}\Big)\boldsymbol{I}, \tag{21}$$

respectively, where

$$\boldsymbol{Y} = \begin{bmatrix}\boldsymbol{y}_1^\top\\ \vdots\\ \boldsymbol{y}_n^\top\end{bmatrix}, \quad \boldsymbol{b} = \int \boldsymbol{k}(\xi)\Pr(\xi;\boldsymbol{\theta})d\xi\ , \quad \text{and} \quad b_0 = \iint k(\xi,\xi')\Pr(\xi;\boldsymbol{\theta})\Pr(\xi';\boldsymbol{\theta})d\xi d\xi'. \tag{22}$$

Our choice of kernel, which allows us to derive closed-form expressions for $\boldsymbol{b}$ and b_0, and as a result for the posterior moments of the gradient, is the quadratic Fisher kernel (Jaakkola and Haussler, 1999, Shawe-Taylor and Cristianini, 2004)

$$k(\xi_i,\xi_j) = \Big(1 + \boldsymbol{u}(\xi_i)^\top\boldsymbol{G}(\boldsymbol{\theta})^{-1}\boldsymbol{u}(\xi_j)\Big)^2, \tag{23}$$

where $\boldsymbol{u}(\xi) = \nabla \log \Pr(\xi;\boldsymbol{\theta})$ is the Fisher score function of the path ξ defined by Eq. 5, and $\boldsymbol{G}(\boldsymbol{\theta})$ is the corresponding Fisher information matrix defined as[11]

$$\boldsymbol{G}(\boldsymbol{\theta}) = \mathbf{E}\big[\boldsymbol{u}(\xi)\boldsymbol{u}(\xi)^\top\big] = \int \boldsymbol{u}(\xi)\boldsymbol{u}(\xi)^\top \Pr(\xi;\boldsymbol{\theta})d\xi. \tag{24}$$

11. To simplify notation, we omit $\boldsymbol{G}$'s dependence on the policy parameters $\boldsymbol{\theta}$, and denote $\boldsymbol{G}(\boldsymbol{\theta})$ as $\boldsymbol{G}$ in the sequel.

Proposition 3 *Using the quadratic Fisher kernel from Eq. 23, the integrals* $\boldsymbol{b}$ *and* b_0 *in Eq. 22 have the following closed form expressions*

$$(\boldsymbol{b})_i = 1 + \boldsymbol{u}(\xi_i)^\top \boldsymbol{G}^{-1}\boldsymbol{u}(\xi_i) \qquad \textit{and} \qquad b_0 = 1 + n.$$

Proof See Appendix A. ■

4.2 Model 2 – Scalar-Valued GP

In our second model, we define g and f as follows:

$$g(\xi;\boldsymbol{\theta}) = \nabla \Pr(\xi;\boldsymbol{\theta}) \quad , \quad f(\xi) = \bar{R}(\xi).$$

Now g is a vector-valued function, while f is a scalar valued GP representing the expected return of the path given as its argument. The noisy measurement corresponding to $f(\xi_i)$ is $y(\xi_i) = R(\xi_i)$, namely, the *actual* return accrued while following the path ξ_i. In this model, the posterior mean and covariance of the gradient $\nabla\eta_B(\boldsymbol{\theta})$ are

$$\mathbf{E}\big[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big] = \boldsymbol{BCy} \qquad \text{and} \qquad \mathbf{Cov}\big[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big] = \boldsymbol{B}_0 - \boldsymbol{BCB}^\top, \tag{25}$$

respectively, where

$$\boldsymbol{y} = \big(R(\xi_1),\ldots,R(\xi_M)\big)^\top, \qquad \boldsymbol{B} = \int \nabla \Pr(\xi;\boldsymbol{\theta})\boldsymbol{k}(\xi)^\top d\xi\ ,$$
$$\boldsymbol{B}_0 = \iint k(\xi,\xi')\nabla \Pr(\xi;\boldsymbol{\theta})\nabla \Pr(\xi';\boldsymbol{\theta})^\top d\xi d\xi'. \tag{26}$$

Here, our choice of kernel function, which again allows us to derive closed-form expressions for $\boldsymbol{B}$ and $\boldsymbol{B}_0$, is the Fisher kernel (Jaakkola and Haussler, 1999, Shawe-Taylor and Cristianini, 2004)

$$k(\xi,\xi') = \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1}\boldsymbol{u}(\xi'). \tag{27}$$

Proposition 4 *Using the Fisher kernel from Eq. 27, the integrals* $\boldsymbol{B}$ *and* $\boldsymbol{B}_0$ *in Eq. 26 have the following closed-form expressions*

$$\boldsymbol{B} = \boldsymbol{U} \qquad \textit{and} \qquad \boldsymbol{B}_0 = \boldsymbol{G},$$

where $\boldsymbol{U} = \big[\boldsymbol{u}(\xi_1),\ldots,\boldsymbol{u}(\xi_M)\big]$.

Proof See Appendix B. ■

Table 1 summarizes the two BPG models presented in Sections 4.1 and 4.2. Our choice of Fisher-type kernels was motivated by the notion that a good representation should depend on the process generating the data (see Jaakkola and Haussler, 1999, Shawe-Taylor and Cristianini, 2004, for a thorough discussion). Our particular selection of linear and quadratic Fisher kernels were guided by the desideratum that the posterior moments of the gradient be analytically tractable as discussed in Section 3.

	Model 1	**Model 2**
Deter. factor (g)	$g(\xi;\boldsymbol{\theta}) = \Pr(\xi;\boldsymbol{\theta})$	$g(\xi;\boldsymbol{\theta}) = \nabla \Pr(\xi;\boldsymbol{\theta})$
GP factor (f)	$f(\xi;\boldsymbol{\theta}) = \bar{R}(\xi)\nabla \log \Pr(\xi;\boldsymbol{\theta})$	$f(\xi) = \bar{R}(\xi)$
Measurement (y)	$y(\xi;\boldsymbol{\theta}) = R(\xi)\nabla \log \Pr(\xi;\boldsymbol{\theta})$	$y(\xi) = R(\xi)$
Prior mean of f	$\mathbf{E}\big[f_j(\xi;\boldsymbol{\theta})\big] = 0$	$\mathbf{E}\big[f(\xi)\big] = 0$
Prior Cov. of f	$\mathbf{Cov}\big[f_j(\xi;\boldsymbol{\theta}), f_\ell(\xi';\boldsymbol{\theta})\big] = \delta_{j,\ell}\, k(\xi,\xi')$	$\mathbf{Cov}\big[f(\xi), f(\xi')\big] = k(\xi,\xi')$
Kernel function	$k(\xi,\xi') = \big(1 + \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1}\boldsymbol{u}(\xi')\big)^2$	$k(\xi,\xi') = \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1}\boldsymbol{u}(\xi')$
$\mathbf{E}\big[\nabla\eta_B(\boldsymbol{\theta})\vert\mathcal{D}_M\big] =$	$\boldsymbol{YCb}$	$\boldsymbol{BCy}$
$\mathbf{Cov}\big[\nabla\eta_B(\boldsymbol{\theta})\vert\mathcal{D}_M\big] =$	$(b_0 - \boldsymbol{b}^\top \boldsymbol{Cb})\boldsymbol{I}$	$\boldsymbol{B}_0 - \boldsymbol{BCB}^\top$
$\boldsymbol{b}$ or $\boldsymbol{B}$	$(\boldsymbol{b})_i = 1 + \boldsymbol{u}(\xi_i)^\top \boldsymbol{G}^{-1}\boldsymbol{u}(\xi_i)$	$\boldsymbol{B} = \boldsymbol{U}$
b_0 or $\boldsymbol{B}_0$	$b_0 = 1 + n$	$\boldsymbol{B}_0 = \boldsymbol{G}$

Table 1: Summary of the Bayesian policy gradient Models 1 and 2.

As described above, in either model, we are restricted in the choice of kernel (quadratic Fisher kernel in Model 1 and Fisher kernel in Model 2) in order to be able to derive closed-form expressions for the posterior mean and covariance of the gradient integral. The loss due to this restriction depends on the problem at hand and is hard to quantify. This loss is exactly the loss of selecting an inappropriate prior in any Bayesian algorithm or, more generally, of choosing a wrong representation (function space) in a machine learning algorithm (referred to as *approximation error* in approximation theory). However, the experimental results of Section 6 indicate that this restriction did not cause a significant error (especially for Model 1) in our gradient estimates, as those estimated by BPG were more accurate than the ones estimated by the MC-based method, given the same number of samples.

4.3 A Bayesian Policy Gradient Evaluation Algorithm

We can now use our two BPG models to define corresponding algorithms for evaluating the gradient of the expected return with respect to the policy parameters. Pseudo-code for these algorithms is shown in Algorithm 1. The generic algorithm (for either model) takes a set of policy parameters $\boldsymbol{\theta}$ and a sample size M as input, and returns an estimate of the posterior moments of the gradient of the expected return with respect to the policy parameters. This algorithm generates M sample paths to evaluate the gradient. For each path ξ_i, the algorithm first computes its score function $\boldsymbol{u}(\xi_i)$ (Line 6). The score function is needed for computing the kernel function k, the measurement $\boldsymbol{y}$ in Model 1, and $\boldsymbol{b}$ or $\boldsymbol{B}$. The algorithm then computes the return R and the measurement $y(\xi_i)$ for the observed path ξ_i (Lines 7 and 9), and updates the kernel matrix $\boldsymbol{K}$ (Line 8) using

$$\boldsymbol{K} := \begin{bmatrix} \boldsymbol{K} & \boldsymbol{k}(\xi_i) \\ \boldsymbol{k}^\top(\xi_i) & k(\xi_i,\xi_i) \end{bmatrix}. \tag{28}$$

Finally, the algorithm adds the measurement error $\boldsymbol{\Sigma}$ to the covariance matrix $\boldsymbol{K}$ (Line 12) and computes the posterior moments of the policy gradient (Line 14). $\boldsymbol{B}(:,i)$ on Line 10 denotes the ith column of the matrix $\boldsymbol{B}$.

Algorithm 1 A Bayesian Policy Gradient Evaluation Algorithm

BPG_Eval$(\boldsymbol{\theta}, M)$
- sample size $M > 0$
- a vector of policy parameters $\boldsymbol{\theta} \in \mathbb{R}^n$

Set $\boldsymbol{G} = \boldsymbol{G}(\boldsymbol{\theta})$, $\mathcal{D} = \emptyset$
for $i = 1$ to M **do**
  Sample a path ξ_i using the policy $\mu(\boldsymbol{\theta})$
  $\mathcal{D} := \mathcal{D} \bigcup \{\xi_i\}$
  $\boldsymbol{u}(\xi_i) = \sum_{t=0}^{T_i-1} \nabla \log \mu(a_{t,i}|x_{t,i};\boldsymbol{\theta})$
  $R(\xi_i) = \sum_{t=0}^{T_i-1} r(x_{t,i}, a_{t,i})$
  Update $\boldsymbol{K}$ using Eq. 28
  $y(\xi_i) = R(\xi_i)\boldsymbol{u}(\xi_i)$ (Model 1) or $y(\xi_i) = R(\xi_i)$ (Model 2)
  $(\boldsymbol{b})_i = 1 + \boldsymbol{u}(\xi_i)^\top \boldsymbol{G}^{-1}\boldsymbol{u}(\xi_i)$ (Model 1) or $\boldsymbol{B}(:,i) = \boldsymbol{u}(\xi_i)$ (Model 2)
end for
$\boldsymbol{C} = (\boldsymbol{K} + \boldsymbol{\Sigma})^{-1}$
$b_0 = 1 + n$ (Model 1) or $\boldsymbol{B}_0 = \boldsymbol{G}$ (Model 2)
Compute the posterior mean and covariance of the policy gradient
$\mathbf{E}\big(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}\big) = \boldsymbol{YCb}$, $\mathbf{Cov}\big(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}\big) = (b_0 - \boldsymbol{b}^\top\boldsymbol{Cb})\boldsymbol{I}$ (Model 1)
or
$\mathbf{E}\big(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}\big) = \boldsymbol{BCy}$, $\mathbf{Cov}\big(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}\big) = \boldsymbol{B}_0 - \boldsymbol{BCB}^\top$ (Model 2)
return $\mathbf{E}\big(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}\big)$ and $\mathbf{Cov}\big(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}\big)$

The kernel functions used in Models 1 and 2 (Eqs. 23 and 27) are both based on the Fisher kernel. Computing the Fisher kernel requires calculating the Fisher information matrix $\boldsymbol{G}(\boldsymbol{\theta})$ (Eq. 24). Consequently, every time we update the policy parameters, we need to recompute $\boldsymbol{G}$. In Algorithm 1 we assume that the Fisher information matrix is known. However, in most practical situations this will not be the case, and consequently the Fisher information matrix must be estimated. Let us briefly outline two possible approaches for estimating the Fisher information matrix in an online manner.

1) Monte-Carlo Estimation: The BPG algorithm generates a number of sample paths using the current policy parameterized by $\boldsymbol{\theta}$ in order to estimate the gradient $\nabla\eta_B(\boldsymbol{\theta})$. We can use these generated sample paths to estimate the Fisher information matrix $\boldsymbol{G}(\boldsymbol{\theta})$ in an online manner, by replacing the expectation in $\boldsymbol{G}$ with empirical averaging as $\hat{\boldsymbol{G}}_M(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{u}(\xi_i)\boldsymbol{u}(\xi_i)^\top$. It can be shown that $\hat{\boldsymbol{G}}_M$ is an unbiased estimator of $\boldsymbol{G}$. One may obtain this estimate recursively $\hat{\boldsymbol{G}}_{i+1} = (1-\frac{1}{i})\hat{\boldsymbol{G}}_i + \frac{1}{i}\boldsymbol{u}(\xi_i)\boldsymbol{u}(\xi_i)^\top$, or more generally $\hat{\boldsymbol{G}}_{i+1} = (1-\zeta_i)\hat{\boldsymbol{G}}_i + \zeta_i\boldsymbol{u}(\xi_i)\boldsymbol{u}(\xi_i)^\top$, where ζ_i is a step-size with $\sum_i \zeta_i = \infty$ and $\sum_i \zeta_i^2 < \infty$. Using the Sherman-Morrison matrix inversion lemma, it is possible to directly estimate the inverse of the Fisher information matrix as

$$\hat{\boldsymbol{G}}_{i+1}^{-1} = \frac{1}{1-\zeta_i}\left[\hat{\boldsymbol{G}}_i^{-1} - \zeta_i \frac{\hat{\boldsymbol{G}}_i^{-1}\boldsymbol{u}(\xi_i)\big(\hat{\boldsymbol{G}}_i^{-1}\boldsymbol{u}(\xi_i)\big)^\top}{1-\zeta_i+\zeta_i\boldsymbol{u}(\xi_i)^\top\hat{\boldsymbol{G}}_i^{-1}\boldsymbol{u}(\xi_i)}\right].$$

2) Maximum Likelihood Estimation: The Fisher information matrix defined by Eq. 24 depends on the probability distribution over paths. This distribution is a product of two factors, one corresponding to the current policy and the other corresponding to the MDP's state-transition probability P (see Eq. 1). Thus if P is known, the Fisher information matrix may be evaluated offline. We can model P using a parameterized model and then estimate the maximum likelihood (ML) model parameters. This approach may lead to a model-based treatment of policy gradients, which could allow us to transfer information between different policies. Current policy gradient algorithms, including the algorithms described in this paper, are extremely wasteful of training data, since they do not have any *disciplined* way to use data collected for previous policy updates in computing the update of the current policy. Model-based policy gradient may help solve this problem.

4.4 BPG Online Sparsification

Algorithm 1 can be made more efficient, both in time and memory, by sparsifying the solution. Such sparsification may be performed incrementally and helps to numerically stabilize the algorithm when the kernel matrix is singular, or nearly so. Sparsification may, in some cases, reduce the accuracy of the solution (the posterior moments of the policy gradient), but it often makes the algorithms significantly faster, especially for large sample sizes. Here we use an online sparsification method proposed by Engel et al. (2002) (see also Csató and Opper, 2002) to selectively add a new observed path to a set of *dictionary* paths $\tilde{\mathcal{D}}$ used as a basis for representing or approximating the full solution. We only add a new path ξ_i to $\tilde{\mathcal{D}}$, if $k(\xi_i,\xi_i) - \tilde{\boldsymbol{k}}(\xi_i)^\top \tilde{\boldsymbol{K}}^{-1} \tilde{\boldsymbol{k}}(\xi_i) > \tau$, where $\tilde{\boldsymbol{k}}$ and $\tilde{\boldsymbol{K}}$ are the dictionary kernel vector and kernel matrix before observing ξ_i, respectively, and τ is a positive threshold parameter that determines the level of accuracy in the approximation as well as the level of sparsity attained. If the new path is added to $\tilde{\mathcal{D}}$ the dictionary kernel matrix $\tilde{\boldsymbol{K}}$ is expanded as shown in Eq. 28.

Proposition 5 *Let $\tilde{\boldsymbol{K}}$ be the $m \times m$ sparse kernel matrix, where $m \leq M$ is the cardinality of $\tilde{\mathcal{D}}_M$. Let $\boldsymbol{A}$ be the $M \times m$ matrix, whose ith row is $[\boldsymbol{A}]_{i,|\tilde{\mathcal{D}}_i|} = 1$ and $[\boldsymbol{A}]_{i,j} = 0$; $\forall j \neq |\tilde{\mathcal{D}}_i|$, if we add the sample path ξ_i to the set of sample paths, and be $\tilde{\boldsymbol{k}}(\xi_i)^\top \tilde{\boldsymbol{K}}^{-1}$ followed by zeros, otherwise. Finally, let $(\tilde{\boldsymbol{b}})_i = 1 + \boldsymbol{u}(\xi_i)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi_i)$ and $\tilde{\boldsymbol{B}} = [\boldsymbol{u}(\xi_1), \ldots, \boldsymbol{u}(\xi_m)]$ with $\xi_i \in \tilde{\mathcal{D}}$. Then, using the sparsification method described above, the posterior moments of the gradient are given by*

Proof See Appendix C. ■

4.5 A Bayesian Policy Gradient Algorithm

So far we were concerned with estimating the gradient of the expected return with respect to the policy parameters. In this section, we present a Bayesian policy gradient (BPG) algorithm that employs the Bayesian gradient estimation methods proposed in Section 4.3 to update the policy parameters. The pseudo-code of this algorithm is shown in Algorithm 2. The algorithm starts with an initial vector of policy parameters $\boldsymbol{\theta}_0$, and updates the parameters in the direction of the posterior mean of the gradient of the expected return estimated

$$\mathbf{E}\big[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big] = \boldsymbol{Y}\boldsymbol{\Sigma}^{-1}\boldsymbol{A}\big(\tilde{\boldsymbol{K}}\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A}+\boldsymbol{I}\big)^{-1}\tilde{\boldsymbol{b}}$$

for Model 1

$$\mathbf{Cov}\big[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big] = \Big[(1+n)-\tilde{\boldsymbol{b}}^\top\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A}\big(\tilde{\boldsymbol{K}}\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A}+\boldsymbol{I}\big)^{-1}\tilde{\boldsymbol{b}}\Big]\boldsymbol{I}$$

and

$$\mathbf{E}\big[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big] = \tilde{\boldsymbol{B}}\big(\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A}\tilde{\boldsymbol{K}}+\boldsymbol{I}\big)^{-1}\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{y}$$

for Model 2

$$\mathbf{Cov}\big[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big] = \boldsymbol{G}-\tilde{\boldsymbol{B}}\big(\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A}\tilde{\boldsymbol{K}}+\boldsymbol{I}\big)^{-1}\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A}\tilde{\boldsymbol{B}}^\top\ .$$

by Algorithm 1. This is repeated N times, or alternatively, until the gradient estimate is sufficiently close to zero.

Algorithm 2 A Bayesian Policy Gradient Algorithm

BPG$(\boldsymbol{\theta}_0, \boldsymbol{\beta}, N, M)$
- initial policy parameters $\boldsymbol{\theta}_0 \in \mathbb{R}^n$
- learning rates β_j , $j = 0, \ldots, N-1$
- number of policy updates $N > 0$
- sample size $M > 0$ for the gradient evaluation algorithm (**BPG_Eval**)

for $j = 0$ to $N-1$ **do**
  $\Delta\boldsymbol{\theta}_j = \mathbf{E}\big(\nabla\eta_B(\boldsymbol{\theta}_j)|\mathcal{D}_M\big)$ from **BPG_Eval**$(\boldsymbol{\theta}_j, M)$
  $\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j + \beta_j\Delta\boldsymbol{\theta}_j$ (Conventional Gradient)
  or
  $\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j + \beta_j\boldsymbol{G}(\boldsymbol{\theta}_j)^{-1}\Delta\boldsymbol{\theta}_j$ (Natural Gradient)
end for
return $\boldsymbol{\theta}_N$

5. Extension to Partially Observable Markov Decision Processes

The Bayesian policy gradient models and algorithms of Section 4 can be extended to partially observable Markov decision processes (POMDPs) along the same lines as in Section 6 of Baxter and Bartlett (2001). In the partially observable case, the stochastic parameterized policy $\mu(\cdot|\cdot;\boldsymbol{\theta})$ controls a POMDP, i.e., the policy has access to an observation process that depends on the state, but it may not observe the state itself directly.

Specifically, for each state $x \in \mathcal{X}$, an observation $o \in \mathcal{O}$ is generated independently according to a probability distribution P_o over observations in $\mathcal{O}$. We denote the probability of observation o at state x by $P_o(o|x)$. A stationary stochastic parameterized policy $\mu(\cdot|\cdot;\boldsymbol{\theta})$ is a function mapping observations $o \in \mathcal{O}$ into probability distributions over the actions $\mu(\cdot|o;\boldsymbol{\theta}) \in \mathcal{P}(\mathcal{A})$. A path generated by the Markov chain induced by policy $\mu(\cdot|\cdot;\boldsymbol{\theta})$ is a sequence of actions and observations $\xi = (o_0, a_0, o_1, a_1, \ldots, o_{T-1}, a_{T-1}, o_T)$, $T \in \{0, 1, \ldots, \infty\}$.

The probability of such a path is given by

$$\Pr(\xi|\mu) = \Pr(\xi|\boldsymbol{\theta}) = P_0(x_0)P_o(o_0|x_0)\prod_{t=0}^{T-1}\mu(a_t|o_t;\boldsymbol{\theta})P(x_{t+1}|x_t,a_t)P_o(o_{t+1}|x_{t+1}).$$

The Fisher score of this path may be written as $\boldsymbol{u}(\xi;\boldsymbol{\theta}) = \frac{\nabla \Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})} = \sum_{t=0}^{T-1}\nabla \log \mu(a_t|o_t;\boldsymbol{\theta})$, which is the same as in the observable case (Eq. 5), except here the policy is defined over observations instead of states. As a result, the models and algorithms of Section 4 may be used in the partially observable case with no change, substituting observations for states.

Moreover, similarly to the gradient estimated by the GPOMDP algorithm in Baxter and Bartlett (2001), the gradient estimated by Algorithm 1, $\nabla\eta_B(\boldsymbol{\theta})$, may be employed with the conjugate-gradients and line-search methods of Baxter et al. (2001) for making better use of gradient information. This allows us to exploit the information contained in the gradient estimate more aggressively than by simply adjusting the parameters by a small amount in the direction of $\nabla\eta_B(\boldsymbol{\theta})$. Conjugate-gradients and line-search are two widely used techniques in non-stochastic optimization that allow us to find better gradient directions than the pure gradient direction, and to obtain better step sizes, respectively.

Note that in this section, we followed Baxter and Bartlett (2001) (the GPOMDP algorithm) and considered stochastic policies that map observations to actions. However, as mentioned by Baxter and Bartlett (2001), it is immediate that the same algorithm works for any finite history of observations. Moreover, along the same way that Aberdeen and Baxter (2001) showed that GPOMDP can be extended to apply to policies with internal state, our BPG POMDP algorithm can also be extended to handle such policies.

6. BPG Experimental Results

In this section, we compare the Bayesian quadrature (BQ) and the plain MC gradient estimates on a simple bandit problem as well as on a continuous state and action linear quadratic regulator (LQR). We also evaluate the performance of the Bayesian policy gradient (BPG) algorithm described in Algorithm 2 on the LQR, and compare it with a Monte-Carlo based policy gradient (MCPG) algorithm.

6.1 A Simple Bandit Problem

The goal of this example is to compare the BQ and MC estimates of the gradient (for some fixed set of policy parameters) using the same sample. Our bandit problem has a single state and a continuous action space $\mathcal{A} = \mathbb{R}$, thus, each path ξ_i consists of a single action a_i. The policy, and therefore also the distribution over the paths is given by $a \sim \mathcal{N}(\theta_1 = 0, \theta_2^2 = 1)$. The parameters θ_1 and θ_2 are the mean and the standard deviation of this distribution. The score function of the path $\xi = a$ and the Fisher information matrix for the policy are $\boldsymbol{u}(\xi) = [a, a^2 - 1]^\top$ and $\boldsymbol{G} = \mathrm{diag}(1, 2)$, respectively.

Table 2 shows the exact gradient of the expected return and its MC and BQ estimates using 10 and 100 samples for two instances of the bandit problem corresponding to two different deterministic reward functions $r(a) = a$ and $r(a) = a^2$. The average over 10^4 runs of the MC and BQ estimates and their standard deviations are reported in Table 2. The true gradient is analytically tractable and is reported as "Exact" in Table 2 for reference.

	Exact	MC (10)	BQ (10)	MC (100)	BQ (100)
$r(a) = a$	$\begin{pmatrix}1\\0\end{pmatrix}$	$\begin{pmatrix}0.9950 \pm 0.438\\-0.0011 \pm 0.977\end{pmatrix}$	$\begin{pmatrix}0.9856 \pm 0.050\\0.0006 \pm 0.060\end{pmatrix}$	$\begin{pmatrix}1.0004 \pm 0.140\\0.0040 \pm 0.317\end{pmatrix}$	$\begin{pmatrix}1.0000 \pm 0.000001\\0.0000 \pm 0.000004\end{pmatrix}$
$r(a) = a^2$	$\begin{pmatrix}0\\2\end{pmatrix}$	$\begin{pmatrix}0.0136 \pm 1.246\\2.0336 \pm 2.831\end{pmatrix}$	$\begin{pmatrix}0.0010 \pm 0.082\\1.9250 \pm 0.226\end{pmatrix}$	$\begin{pmatrix}0.0051 \pm 0.390\\1.9869 \pm 0.857\end{pmatrix}$	$\begin{pmatrix}0.0000 \pm 0.000003\\2.0000 \pm 0.000011\end{pmatrix}$

Table 2: The true gradient of the expected return and its MC and BQ estimates for two instances of the bandit problem corresponding to two different reward functions.

As shown in Table 2, the variance of the BQ estimates are lower than the variance of the MC estimates by an order of magnitude for the small sample size ($M = 10$), and by 6 orders of magnitude for the large sample size ($M = 100$). The BQ estimate is also more accurate than the MC estimate for the large sample size, and is roughly the same for the small sample size.

6.2 Linear Quadratic Regulator

In this section, we consider the following linear system in which the goal is to minimize the expected return over 20 steps.[12] Thus, it is an episodic problem with paths of length 20.

System:
Initial State: $x_0 \sim \mathcal{N}(0.3, 0.001)$
Reward: $r_t = x_t^2 + 0.1a_t^2$
Transition: $x_{t+1} = x_t + a_t + n_x \; ; \; n_x \sim \mathcal{N}(0, 0.01)$

Policy:
Actions: $a_t \sim \mu(\cdot|x_t; \boldsymbol{\theta}) = \mathcal{N}(\lambda x_t, \sigma^2)$
Parameters: $\boldsymbol{\theta} = (\lambda \; , \; \sigma)^\top$

We run two sets of experiments on this system. We first fix the set of policy parameters and compare the BQ and MC estimates of the gradient of the expected return using the same sample. We then proceed to solving the complete policy gradient problem and compare the performance of the BPG algorithm (with both conventional and natural gradients) with a Monte-Carlo based policy gradient (MCPG) algorithm.

6.2.1 Gradient Estimation

In this section, we compare the BQ and MC estimates of the gradient of the expected return for the policy induced by parameters $\lambda = -0.2$ and $\sigma = 1$. We use several different sample sizes (number of paths used for gradient estimation) $M = 5j \; , \; j = 1, \ldots, 20$ for the BQ and MC estimates. For each sample size, we compute the MC and BQ estimators using the same sample, repeat this process 10^4 times, and then compute the average. The true gradient is analytically tractable and is used for comparison purposes.

Figure 1 shows the mean squared error (MSE) (left column) and the mean absolute angular error (right column) of the MC and BQ estimates of the gradient for several different sample sizes. The absolute angular error is the absolute value of the angle between the true and estimated gradients. In this figure, the BQ gradient estimates were calculated using

12. What we mean by reward and return in this section is in fact cost and loss, and this is why we are dealing with a minimization, and not a maximization, problem here. The reason for this is to maintain consistency in notations and definitions throughout the paper.

Model 1 (top row) and Model 2 (bottom row) with sparsification. The error bars in the figures on the right column are the standard errors of the mean absolute angular errors.

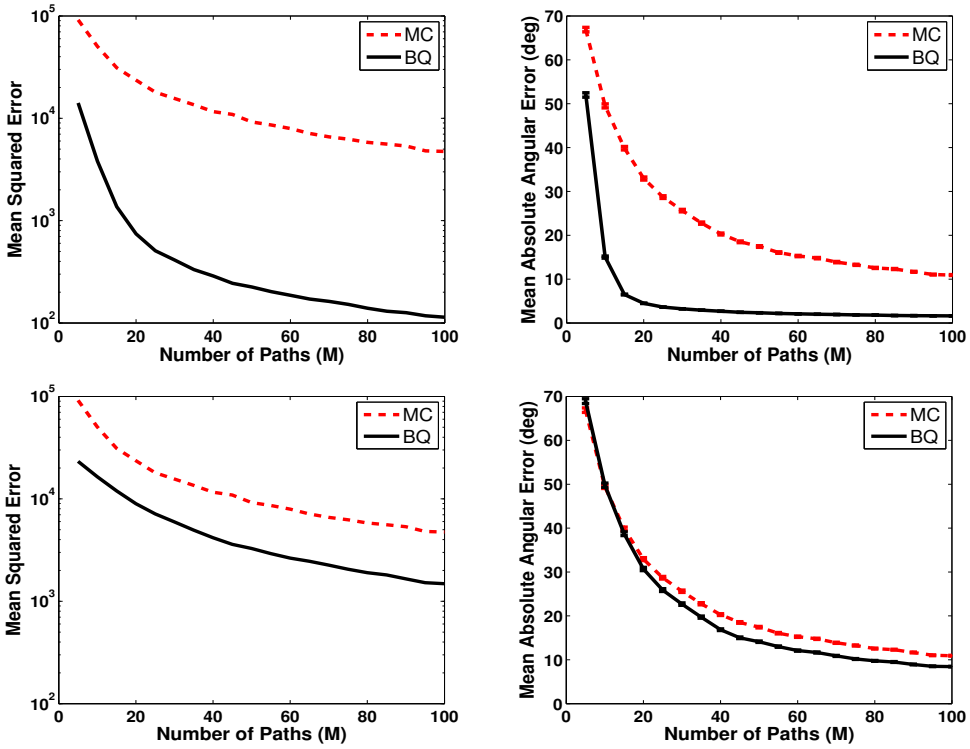

Figure 1: Results for the LQR problem using Model 1 (top row) and Model 2 (bottom row) with sparsification. Shown are the MSE (left column) and the mean absolute angular error (right column) of the MC and BQ estimates as a function of the number of sample paths M. All results are averaged over 10^4 runs.

We ran another set of experiments in which we added i.i.d. Gaussian noise to the rewards: $r_t = x_t^2 + 0.1a_t^2 + n_r \ ; \ n_r \sim \mathcal{N}(0, \sigma_r^2)$. Note that in Models 1 and 2, $y(\xi)$, the noisy sample of $f(\xi)$, is of the form $R(\xi)\nabla \log \Pr(\xi;\boldsymbol{\theta})$ and $R(\xi)$, respectively (see Sections 4.1 and 4.2). Moreover, since each reward r_t is a Gaussian random variable with variance σ_r^2, the return $R(\xi) = \sum_{t=0}^{T-1} r_t$ is also a Gaussian random variable with variance $T\sigma_r^2$. Therefore in this case, the measurement noise covariance matrices for Models 1 and 2 may be written as $\boldsymbol{\Sigma} = T\sigma_r^2 \operatorname{diag}\left(\left(\frac{\partial}{\partial\theta_i}\log p(\xi_1;\boldsymbol{\theta})\right)^2, \ldots, \left(\frac{\partial}{\partial\theta_i}\log p(\xi_M;\boldsymbol{\theta})\right)^2\right)$ and $\boldsymbol{\Sigma} = T\sigma_r^2 \boldsymbol{I}$, respectively, where $T = 20$ is the path length.[13] We tried two different Gaussian reward noise standard deviations: $\sigma_r = 0.1$ and 1 in our experiments. Adding noise to the rewards slightly increased the error of the BQ and MC estimates of the gradient. However, the graphs comparing these estimates remained quite similar to those shown in Figure 1. Hence in Figure 2, we compare the MSE

13. In Model 1, $\boldsymbol{\Sigma}$ is the measurement noise covariance matrix for the ith component of the gradient $\frac{\partial}{\partial\theta_i}\eta_B(\boldsymbol{\theta})$. Note that $\frac{\partial}{\partial\theta_i}\log p(\xi_j;\boldsymbol{\theta})$ depends only on the policy and can be calculated using Eq. 5.

(left column) and the mean absolute angular error (right column) of the BQ estimates with and without noise in the rewards as a function of the number of sample paths M. In this figure, the noise in the rewards has variance $\sigma_r^2 = 1$, and the BQ gradient estimates were calculated using Model 1 (top row) and Model 2 (bottom row) with sparsification.

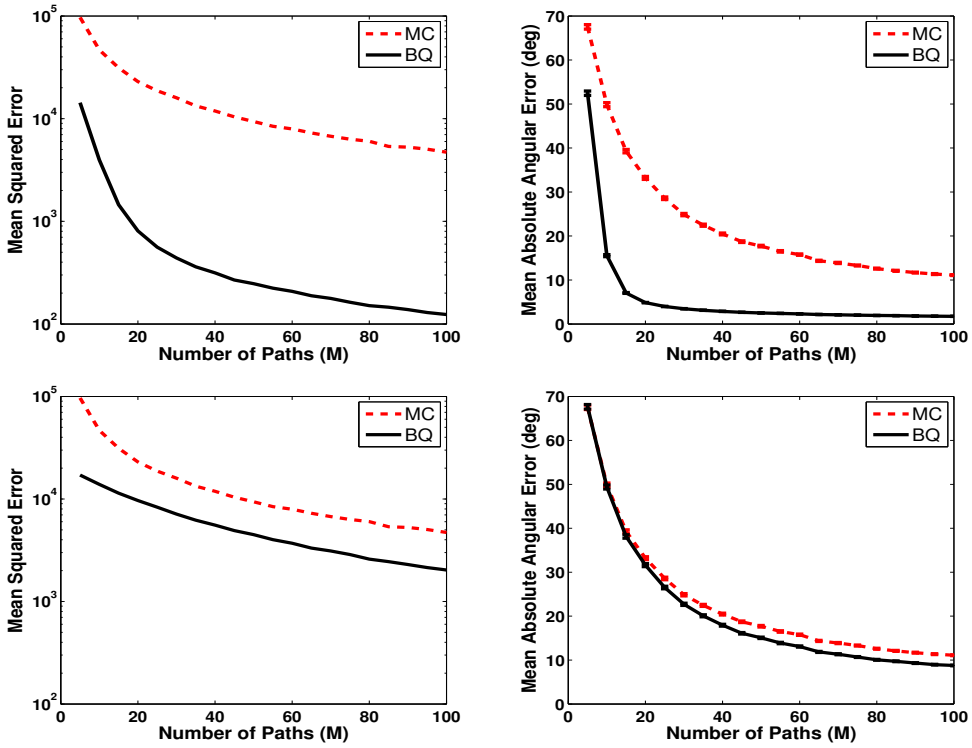

Figure 2: Results for the LQR problem in which the rewards are corrupted by i.i.d. Gaussian noise with $\sigma_r^2 = 1$. Shown are the MSE (left column) and the mean absolute angular error (right column) of the BQ estimates with and without noise in the rewards as a function of the number of sample paths M. The BQ gradient estimates were calculated using Model 1 (top row) and Model 2 (bottom row) with sparsification. All results are averaged over 10^4 runs.

6.2.2 Policy Optimization

In this section, we use Bayesian policy gradient (BPG) to optimize the policy parameters in the LQR problem. Figure 3 shows the performance of the BPG algorithm with the conventional (BPG) and natural (BPNG) gradient estimates, versus a MC-based policy gradient (MCPG) algorithm, for sample sizes (number of sample paths used to estimate the gradient of each policy) $M = 5$, 10, 20, and 40. We use Algorithm 2 with the number of updates set to $N = 100$, and Model 1 with sparsification for the BPG and BPNG methods. Since Algorithm 2 computes the Fisher information matrix for each set of policy parameters,

the estimate of the natural gradient is provided at little extra cost at each step. The returns obtained by these methods are averaged over 10^4 runs. The policy parameters are initialized randomly at each run. In order to ensure that the learned parameters do not exceed an acceptable range, the policy parameters are defined as $\lambda = -1.999 + 1.998/(1 + e^{\kappa_1})$ and $\sigma = 0.001 + 1/(1 + e^{\kappa_2})$. The optimal solution is $\lambda^* \approx -0.92$, $\sigma^* = 0.001$, $\eta_B(\lambda^*, \sigma^*) = 0.3067$, corresponding to $\kappa_1^* \approx -0.16$ and $\kappa_2^* \to \infty$.

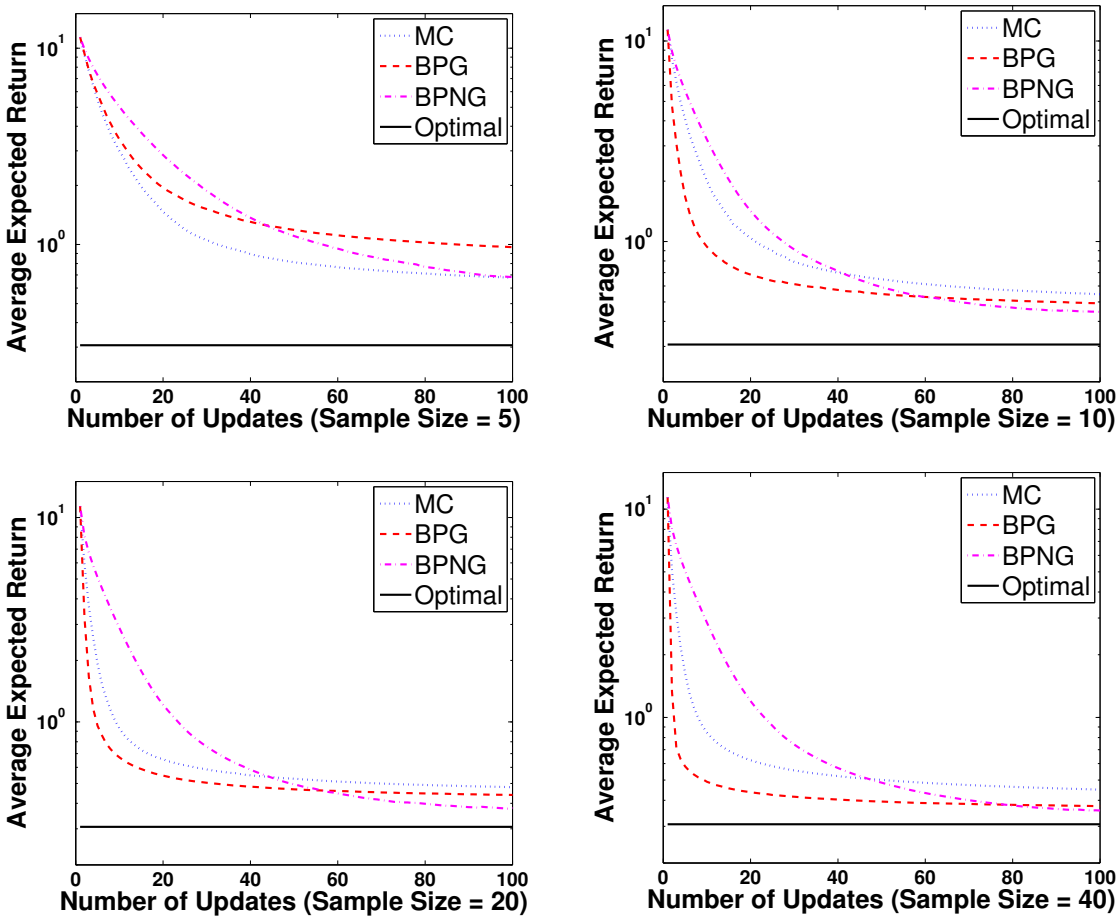

Figure 3: A comparison of the average expected returns of the Bayesian policy gradient algorithm using conventional (BPG) and natural (BPNG) gradient estimates, with the average expected return of a MC-based policy gradient algorithm (MCPG) for sample sizes $M = 5$, 10, 20, and 40. All results are averaged over 10^4 runs.

Figure 3 shows that the MCPG algorithm performs better than BPG and BPNG only for the smallest sample size ($M = 5$), whereas for larger samples BPG and BPNG dominate MCPG. The better performance of MCPG for very small sample size is due to the fact that in this case, the Bayesian estimators, BPG and BPNG, like any other Bayesian estimator or

posterior in such case, rely more on the prior, and thus, are not accurate if the prior is not very informative. A similar phenomenon was also reported by Rasmussen and Ghahramani (2003). We used two different learning rates for the two components of the gradient. For a fixed sample size, BPG and MCPG methods start with an initial learning rate and decrease it according to the schedule $\beta_j = \beta_0\big(20/(20+j)\big)$. The BPNG algorithm uses a fixed learning rate multiplied by the determinant of the Fisher information matrix. We tried many values for the initial learning rates used by these algorithms and those in Table 3 yielded the best performance of those we tried.

β_0	$M=5$	$M=10$	$M=20$	$M=40$
MCPG	$0.01, 0.05$	$0.05, 0.05$	$0.05, 0.10$	$0.05, 0.10$
BPG	$0.01, 0.05$	$0.07, 0.10$	$0.15, 0.15$	$0.10, 0.30$
BPNG	$0.010, 0.005$	$0.010, 0.005$	$0.015, 0.005$	$0.015, 0.005$
BPG-var	$0.05, 0.05$	$0.10, 0.10$	$0.10, 0.15$	$0.15, 0.30$

Table 3: Initial learning rates β_0 used by the policy gradient algorithms for the two components of the gradient.

So far we have assumed that the Fisher information matrix is known. In the next experiment, we estimate it using both MC and a model-based maximum likelihood (ML) method, as discussed in Section 4.3. In the ML approach, we model the transition probability function as $P(x_{t+1}|x_t, a_t) = \mathcal{N}(c_1 x_t + c_2 a_t + c_3, c_4^2)$, and then estimate its parameters (c_1, c_2, c_3, c_4) using observing state transitions. Figure 4 shows that the BPG algorithm, when the Fisher information matrix is estimated using ML and MC, still performs better than MCPG. Top and bottom rows contain the results for the BPG algorithm with conventional (BPG-ML and BPG-MC) and natural (BPNG-ML and BPNG-MC) gradient estimates, respectively. Although the BPG-ML (BPNG-ML) outperforms BPG-MC (BPNG-MC) for small sample sizes, the difference in their performance disappears as we increase the sample size. One reason for the good performance of BPG-ML is that the form of the state transition function $P(x_{t+1}|x_t, a_t)$ has been selected correctly. Here we used the same initial learning rates and learning rate schedules as in the experiments of Figure 3 (see Table 3).

Although the proposed Bayesian policy gradient algorithm (Algorithm 2) uses only the posterior mean of the gradient in its updates, it can be extended to make judicious use of the second moment information provided by the Bayesian policy gradient estimation algorithm (Algorithm 1). In the last experiment of this section, we use the posterior covariance of the gradient, provided by Algorithm 1, to select the learning rate and the direction of the updates in Algorithm 2. The idea is to use a small learning rate when the variance of the gradient estimate is large, and to have a large update when it is small. We refer to the resulting algorithm by the name BPG-var. This algorithm uses a fixed learning rate parameter (see Table 3) multiplied by $\Big[\big(1+n\big)\boldsymbol{I} - \mathbf{Cov}\big(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M\big)\Big]/(1+n)$ in its updates. Note that $n+1$ is b_0 in the calculation of the posterior covariance of the gradient in Model 1 (see Proposition 3), and is used here as an upper bound for the posterior covariance of the gradient. Figure 5 compares the average expected return of BPG-var with BPG and

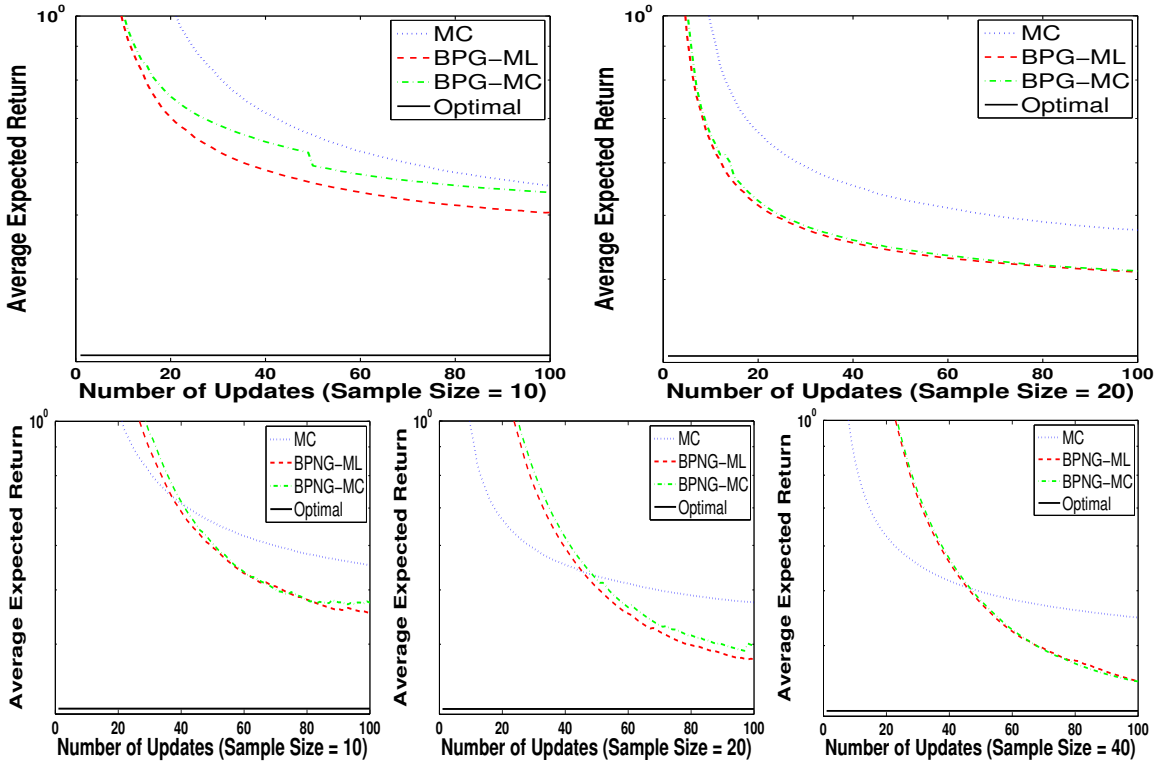

Figure 4: A comparison of the average expected returns of the BPG algorithm, when the Fisher information matrix is estimated using ML and MC, with the average expected return of a MC-based policy gradient algorithm (MCPG). The top and bottom rows contain the results for the BPG algorithm with conventional (BPG-ML and BPG-MC) and natural (BPNG-ML and BPNG-MC) gradient estimates, respectively. All results are averaged over 10^4 runs.

MCPG for sample sizes $M = 5$, 10, 20, and 40. The figure shows that BPG-var performs better than BPG and MCPG for all the sample sizes. It even has a better performance than MCPG for the smallest sample size ($M = 5$). Comparing Figures 3 and 5 shows that BPG-var converges faster than BPNG and has similar final performance. As we expected, BPG-var and BPG perform more and more alike as we increase the sample size. This is because by increasing the sample size the estimated gradient (the posterior mean of the gradient), and as a result, the update direction used by BPG becomes more reliable.

In an approach similar to the one used in the experiments of Figure 5, Vien et al. (2011) used BQ to estimate the Hessian matrix distribution, and then used its mean as learning rate schedule to improve the performance of BPG. They empirically showed that their method performs better than BPG and BPNG in terms of convergence speed.

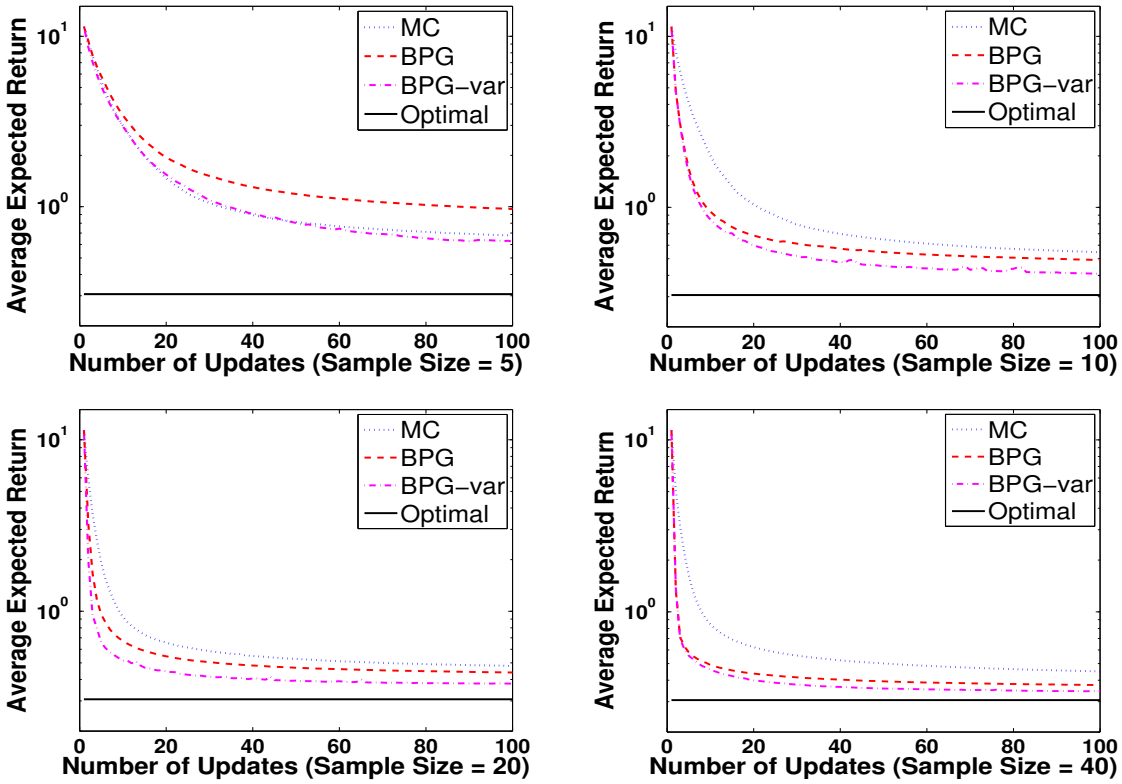

Figure 5: A comparison of the average expected returns of the BPG algorithm that uses the posterior covariance in its updates (BPG-var), with the average expected return of the BPG and a MC-based policy gradient algorithm (MCPG) for sample sizes $M = 5$, 10, 20, and 40. All results are averaged over 10^4 runs.

7. Bayesian Actor-Critic

The models and algorithms of Section 4 consider complete trajectories as the basic observable unit, and thus, do not require the dynamics within each trajectory to be of any special form. In particular, it is not necessary for the dynamics to have the Markov property, allowing the resulting algorithms to handle partially observable MDPs, Markov games, and other non-Markovian systems. On the down side, these algorithms cannot take advantage of the Markov property when operating in Markovian systems. Moreover, since the unit of observation of these algorithms is the entire trajectory, their gradient estimates have larger variance than the algorithms that will be discussed in this section, whose unit of observation is *(current state, action, next state)*, since they take advantage of the Markov property, especially when the size of the trajectories is large.

In this section, we apply the Bayesian quadrature idea to the policy gradient expression given by Eq. 7, i.e.,

$$\nabla\eta(\boldsymbol{\theta}) = \int dxda\ \nu(x;\boldsymbol{\theta})\nabla\mu(a|x;\boldsymbol{\theta})Q(x,a;\boldsymbol{\theta}),$$

and derive a family of Bayesian actor-critic (BAC) algorithms. In this approach, we place a Gaussian process (GP) prior over action-value functions using a prior covariance kernel defined on state-action pairs: $k(\boldsymbol{z},\boldsymbol{z}') = \mathbf{Cov}\big[Q(\boldsymbol{z}),Q(\boldsymbol{z}')\big]$. We then compute the GP posterior conditioned on the sequence of individual observed transitions. In the same vein as Section 4, by an appropriate choice of a prior on action-value functions, we are able to derive closed-form expressions for the posterior moments of $\nabla\eta(\boldsymbol{\theta})$. The main questions here are: **1)** how to compute the GP posterior of the action-value function given a sequence of observed transitions? and **2)** how to choose a prior for the action-value function that allows us to derive closed-form expressions for the posterior moments of $\nabla\eta(\boldsymbol{\theta})$? Fortunately, well developed machinery for computing the posterior moments of $Q(\boldsymbol{z})$ is provided in a series of papers by Engel et al. (2003, 2005) (for a thorough treatment see Engel, 2005). In the next two sections, we will first briefly review some of the main results pertaining to the Gaussian process temporal difference (GPTD) model proposed in Engel et al. (2005), and then will show how they may be combined with the Bayesian quadrature idea in developing a family of Bayesian actor-critic algorithms.

7.1 Gaussian Process Temporal Difference Learning

The Gaussian process temporal difference (GPTD) learning (Engel et al., 2003, 2005) model is based on a statistical generative model relating the observed reward signal r to the unobserved action-value function Q

$$r(\boldsymbol{z}_i) = Q(\boldsymbol{z}_i) - \gamma Q(\boldsymbol{z}_{i+1}) + N(\boldsymbol{z}_i,\boldsymbol{z}_{i+1}), \tag{29}$$

where $N(\boldsymbol{z}_i,\boldsymbol{z}_{i+1})$ is a zero-mean noise signal that accounts for the discrepancy between $r(\boldsymbol{z}_i)$ and $Q(\boldsymbol{z}_i) - \gamma Q(\boldsymbol{z}_{i+1})$. Let us define the finite-dimensional processes $\boldsymbol{r}_t$, Q_t, N_t, and the $t\times(t+1)$ matrix $\boldsymbol{H}_t$ as follows:

$$\begin{aligned} \boldsymbol{r}_t &= \big(r(\boldsymbol{z}_0),\ldots,r(\boldsymbol{z}_t)\big)^\top, & Q_t &= \big(Q(\boldsymbol{z}_0),\ldots,Q(\boldsymbol{z}_t)\big)^\top, \\ N_t &= \big(N(\boldsymbol{z}_0,\boldsymbol{z}_1),\ldots,N(\boldsymbol{z}_{t-1},\boldsymbol{z}_t)\big)^\top, \end{aligned} \tag{30}$$

$$\boldsymbol{H}_t = \begin{bmatrix} 1 & -\gamma & 0 & \ldots & 0 \\ 0 & 1 & -\gamma & \ldots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \ldots & 1 & -\gamma \end{bmatrix}. \tag{31}$$

The set of Eqs. 29 for $i = 0,\ldots,t-1$ may be written as $\boldsymbol{r}_{t-1} = \boldsymbol{H}_t Q_t + N_t$. Under certain assumptions on the distribution of the discounted return random process (Engel et al., 2005), the covariance of the noise vector N_t is given by

$$\boldsymbol{\Sigma}_t = \sigma^2\boldsymbol{H}_t\boldsymbol{H}_t^\top = \sigma^2\begin{bmatrix} 1+\gamma^2 & -\gamma & 0 & \ldots & 0 \\ -\gamma & 1+\gamma^2 & -\gamma & \ldots & 0 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \ldots & -\gamma & 1+\gamma^2 \end{bmatrix}. \tag{32}$$

In episodic tasks, if $\boldsymbol{z}_{t-1}$ is the last state-action pair in the episode (i.e., $\boldsymbol{x}_t$ is a zero-reward absorbing terminal state), $\boldsymbol{H}_t$ becomes a square $t \times t$ invertible matrix of the form shown in Eq. 31 with its last column removed. The effect on the noise covariance matrix $\boldsymbol{\Sigma}_t$ is that the bottom-right element becomes 1 instead of $1 + \gamma^2$.

Placing a GP prior on Q and assuming that N_t is also normally distributed, we may use Bayes' rule to obtain the posterior moments of Q:

$$\begin{aligned} \hat{Q}_t(\boldsymbol{z}) &= \mathbf{E}\left[Q(\boldsymbol{z})|\mathcal{D}_t\right] = \boldsymbol{k}_t(\boldsymbol{z})^\top \boldsymbol{\alpha}_t, \\ \hat{S}_t(\boldsymbol{z}, \boldsymbol{z}') &= \mathbf{Cov}\left[Q(\boldsymbol{z}), Q(\boldsymbol{z}')|\mathcal{D}_t\right] = k(\boldsymbol{z}, \boldsymbol{z}') - \boldsymbol{k}_t(\boldsymbol{z})^\top \boldsymbol{C}_t \boldsymbol{k}_t(\boldsymbol{z}'), \end{aligned} \tag{33}$$

where $\mathcal{D}_t$ denotes the observed data up to and including time step t. We used here the following definitions:

$$\begin{aligned} \boldsymbol{k}_t(\boldsymbol{z}) &= \big(k(\boldsymbol{z}_0, \boldsymbol{z}), \ldots, k(\boldsymbol{z}_t, \boldsymbol{z})\big)^\top, & \boldsymbol{K}_t &= \big[\boldsymbol{k}_t(\boldsymbol{z}_0), \boldsymbol{k}_t(\boldsymbol{z}_1), \ldots, \boldsymbol{k}_t(\boldsymbol{z}_t)\big], \\ \boldsymbol{\alpha}_t &= \boldsymbol{H}_t^\top \left(\boldsymbol{H}_t \boldsymbol{K}_t \boldsymbol{H}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1} \boldsymbol{r}_{t-1}\,, & \boldsymbol{C}_t &= \boldsymbol{H}_t^\top \left(\boldsymbol{H}_t \boldsymbol{K}_t \boldsymbol{H}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1} \boldsymbol{H}_t\,. \end{aligned} \tag{34}$$

Note that $\hat{Q}_t(\boldsymbol{z})$ and $\hat{S}_t(\boldsymbol{z}, \boldsymbol{z}')$ are the posterior mean and covariance functions of the posterior GP, respectively. As more samples are observed, the posterior covariance decreases, reflecting a growing confidence in the Q-function estimate $\hat{Q}_t$.

7.2 A Family of Bayesian Actor-Critic Algorithms

We are now in a position to describe the main idea behind our BAC approach. Making use of the linearity of Eq. 7 in Q and denoting $\boldsymbol{g}(\boldsymbol{z}; \boldsymbol{\theta}) = \pi^\mu(\boldsymbol{z}) \nabla \log \mu(\boldsymbol{a}|\boldsymbol{x}; \boldsymbol{\theta})$, we obtain the following expressions for the posterior moments of the policy gradient (O'Hagan, 1991):

$$\begin{aligned} \mathbf{E}[\nabla \eta(\boldsymbol{\theta})|\mathcal{D}_t] &= \int_{\mathcal{Z}} d\boldsymbol{z} \boldsymbol{g}(\boldsymbol{z}; \boldsymbol{\theta}) \hat{Q}_t(\boldsymbol{z}; \boldsymbol{\theta}), \\ \mathbf{Cov}\left[\nabla \eta(\boldsymbol{\theta})|\mathcal{D}_t\right] &= \int_{\mathcal{Z}^2} d\boldsymbol{z} d\boldsymbol{z}' \boldsymbol{g}(\boldsymbol{z}; \boldsymbol{\theta}) \hat{S}_t(\boldsymbol{z}, \boldsymbol{z}') \boldsymbol{g}(\boldsymbol{z}'; \boldsymbol{\theta})^\top. \end{aligned} \tag{35}$$

Substituting the expressions for the posterior moments of Q from Eq. 33 into Eq. 35, we obtain

$$\begin{aligned} \mathbf{E}[\nabla \eta(\boldsymbol{\theta})|\mathcal{D}_t] &= \int_{\mathcal{Z}} d\boldsymbol{z} \boldsymbol{g}(\boldsymbol{z}; \boldsymbol{\theta}) \boldsymbol{k}_t(\boldsymbol{z})^\top \boldsymbol{\alpha}_t, \\ \mathbf{Cov}\left[\nabla \eta(\boldsymbol{\theta})|\mathcal{D}_t\right] &= \int_{\mathcal{Z}^2} d\boldsymbol{z} d\boldsymbol{z}' \boldsymbol{g}(\boldsymbol{z}; \boldsymbol{\theta}) \left(k(\boldsymbol{z}, \boldsymbol{z}') - \boldsymbol{k}_t(\boldsymbol{z})^\top \boldsymbol{C}_t \boldsymbol{k}_t(\boldsymbol{z}')\right) \boldsymbol{g}(\boldsymbol{z}'; \boldsymbol{\theta})^\top. \end{aligned}$$

These equations provide us with the general form of the posterior policy gradient moments. We are now left with a computational issue, namely, how to compute the integrals appearing in these expressions? We need to be able to evaluate the following integrals:

$$\boldsymbol{B}_t = \int_{\mathcal{Z}} d\boldsymbol{z} \boldsymbol{g}(\boldsymbol{z}; \boldsymbol{\theta}) \boldsymbol{k}_t(\boldsymbol{z})^\top, \qquad \boldsymbol{B}_0 = \int_{\mathcal{Z}^2} d\boldsymbol{z} d\boldsymbol{z}' \boldsymbol{g}(\boldsymbol{z}; \boldsymbol{\theta}) k(\boldsymbol{z}, \boldsymbol{z}') \boldsymbol{g}(\boldsymbol{z}'; \boldsymbol{\theta})^\top. \tag{36}$$

Using these definitions, we may write the gradient posterior moments compactly as

$$\mathbf{E}[\nabla \eta(\boldsymbol{\theta})|\mathcal{D}_t] = \boldsymbol{B}_t \boldsymbol{\alpha}_t\,, \qquad \mathbf{Cov}\left[\nabla \eta(\boldsymbol{\theta})|\mathcal{D}_t\right] = \boldsymbol{B}_0 - \boldsymbol{B}_t \boldsymbol{C}_t \boldsymbol{B}_t^\top. \tag{37}$$

In order to render these integrals analytically tractable, we choose our prior covariance kernel to be the sum of an arbitrary state-kernel k_x and the (invariant) Fisher kernel k_F between state-action pairs (see e.g., Shawe-Taylor and Cristianini, 2004, Chapter 12). The (policy dependent) Fisher information kernel and our overall state-action kernel are then given by

$$k_F(\boldsymbol{z}, \boldsymbol{z}') = \boldsymbol{u}(\boldsymbol{z}; \boldsymbol{\theta})^\top \boldsymbol{G}(\boldsymbol{\theta})^{-1} \boldsymbol{u}(\boldsymbol{z}') \;, \qquad k(\boldsymbol{z}, \boldsymbol{z}') = k_x(\boldsymbol{x}, \boldsymbol{x}') + k_F(\boldsymbol{z}, \boldsymbol{z}') \;, \tag{38}$$

where $\boldsymbol{u}(\boldsymbol{z}; \boldsymbol{\theta})$ and $\boldsymbol{G}(\boldsymbol{\theta})$ are the score function and Fisher information matrix defined as[14]

$$\boldsymbol{u}(\boldsymbol{z}; \boldsymbol{\theta}) = \nabla \log \mu(a|x; \boldsymbol{\theta}), \tag{39}$$

$$\boldsymbol{G}(\boldsymbol{\theta}) = \mathbf{E}_{x \sim \nu^\mu, a \sim \mu} \left[\nabla \log \mu(a|x; \boldsymbol{\theta}) \nabla \log \mu(a|x; \boldsymbol{\theta})^\top \right] = \mathbf{E}_{\boldsymbol{z} \sim \pi^\mu} \left[\boldsymbol{u}(\boldsymbol{z}; \boldsymbol{\theta}) \boldsymbol{u}(\boldsymbol{z}; \boldsymbol{\theta})^\top \right]. \tag{40}$$

Although here we have total flexibility in selecting the state kernel, we are restricted to the Fisher kernel for state-action pairs. This restriction may cause an error in approximating some action-value functions Q. This error depends on the problem at hand and is hard to quantify. This is exactly the same as selecting an inaccurate prior in any Bayesian algorithm or choosing a wrong representation (function space) in any machine learning algorithm (referred to as *approximation error* in the approximation theory). However, this restriction did not cause a significant error in our experiments (see Section 8), as in almost all of them, the gradients estimated by BAC were more accurate than those estimated by the MC-based method, given the same number of samples.

Note that in Sections 4 to 6 we used a formulation in which the observable unit is a system trajectory, and thus, the expected return and its gradient are defined by Eqs. 2 and 4. In this formulation, the score function and Fisher information matrix are defined by Eqs. 5 and 24. However, in the formulation used in this section and in the rest of the paper, where the observable unit is an individual state-action-reward transition, the expected return and its gradient are defined by Eqs. 3 and 7. In this formulation, the score function and Fisher information matrix are defined by Eqs. 39 and 40, respectively.

A nice property of the Fisher kernel is that while $k_F(\boldsymbol{z}, \boldsymbol{z}')$ depends on the policy, it is invariant to policy reparameterization. In other words, it only depends on the actual probability mass assigned to each action and not on its explicit dependence on the policy parameters. As mentioned above, another attractive property of this particular choice of kernel is that it renders the integrals in Eq. 36 analytically tractable, as made explicit in the following proposition

Proposition 6 *Let $k(\boldsymbol{z}, \boldsymbol{z}') = k_x(x, x') + k_F(\boldsymbol{z}, \boldsymbol{z}')$ for all $(\boldsymbol{z}, \boldsymbol{z}') \in \mathcal{Z}^2$, where $k_x : \mathcal{X}^2 \to \mathbb{R}$ is an arbitrary positive definite state-kernel and $k_F : \mathcal{Z}^2 \to \mathbb{R}$ is the Fisher information kernel. Then $\boldsymbol{B}_t$ and $\boldsymbol{B}_0$ from Eq. 36 satisfy*

$$\boldsymbol{B}_t = \boldsymbol{U}_t \;, \qquad \boldsymbol{B}_0 = \boldsymbol{G}, \tag{41}$$

where $\boldsymbol{U}_t = \big[\boldsymbol{u}(\boldsymbol{z}_0), \boldsymbol{u}(\boldsymbol{z}_1), \ldots, \boldsymbol{u}(\boldsymbol{z}_t)\big]$.

14. Similar to $\boldsymbol{u}(\xi)$ and $\boldsymbol{G}$ defined by Eqs. 5 and 24, to simplify the notation, we omit the dependence of $\boldsymbol{u}$ and $\boldsymbol{G}$ to the policy parameters $\boldsymbol{\theta}$, and replace $\boldsymbol{u}(\boldsymbol{z}; \boldsymbol{\theta})$ and $\boldsymbol{G}(\boldsymbol{\theta})$ with $\boldsymbol{u}(\boldsymbol{z})$ and $\boldsymbol{G}$ in the sequel.

Proof See Appendix D. ■

An immediate consequence of Proposition 6 is that, in order to compute the posterior moments of the policy gradient, we only need to be able to evaluate (or estimate) the score vectors $\boldsymbol{u}(\boldsymbol{z}_i),\ i = 0, \ldots, t$ and the Fisher information matrix $\boldsymbol{G}$ of our policy. Evaluating the Fisher information matrix $\boldsymbol{G}$ is somewhat more challenging, since on top of taking the expectation with respect to the policy $\mu(a|x;\boldsymbol{\theta})$, computing $\boldsymbol{G}$ involves an additional expectation over the state-occupancy density $\nu^\mu(x)$, which is not generally known. In most practical situations we therefore have to resort to estimating $\boldsymbol{G}$ from data. When ν^μ in the definition of the Fisher information matrix (Eq. 40) is the stationary distribution over states under policy μ, one straightforward method to estimate $\boldsymbol{G}$ from a trajectory $\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_t$ is to use the (unbiased) estimator (see Proposition 6 for the definition of $\boldsymbol{U}_t$):

$$\hat{\boldsymbol{G}}_t = \frac{1}{t+1} \sum_{i=0}^{t} \boldsymbol{u}(\boldsymbol{z}_i)\boldsymbol{u}(\boldsymbol{z}_i)^\top = \frac{1}{t+1} \boldsymbol{U}_t \boldsymbol{U}_t^\top. \tag{42}$$

In case ν^μ in Eq. 40 is a discounted weighting of states encountered by following policy μ (as it is considered in this paper), a method for estimating $\boldsymbol{G}$ from a number of trajectories is shown in Algorithm 3. Note that $(1-\gamma)\nu^\mu$ corresponds to the distribution of a Markov chain that starts from a state sampled according to P_0 and at each step either follows the policy μ with probability γ or restarts from a new initial state drawn from P_0 with probability $1-\gamma$. It is easy to show that the average number of steps between two successive restarts of this distribution is $1/(1-\gamma)$.

Algorithm 3 Fisher Information Matrix Estimation Algorithm

G-EST$(\boldsymbol{\theta}, M)$
- $\boldsymbol{\theta}$ policy parameters
- $M > 0$ number of episodes used to estimate the Fisher information matrix

$\hat{\boldsymbol{G}}(\boldsymbol{\theta}) = \boldsymbol{0}$
for $i = 1$ to M **do**
  done = false, term = false, $t = -1$
  Draw $x_0^i \sim P_0(\cdot)$
  while not done **do**
    $t = t + 1$
    Draw $a_t^i \sim \mu(\cdot|x_t^i;\boldsymbol{\theta})$ and $x_{t+1}^i \sim P(\cdot|x_t^i, a_t^i)$
    if $x_{t+1}^i = x_{\text{term}}$ **then** done = true
    if (done = false $\wedge$ term = false) **then**
      $\hat{\boldsymbol{G}}(\boldsymbol{\theta}) := \hat{\boldsymbol{G}}(\boldsymbol{\theta}) + \boldsymbol{u}(\boldsymbol{z}_t^i;\boldsymbol{\theta})\boldsymbol{u}(\boldsymbol{z}_t^i;\boldsymbol{\theta})^\top$ and w.p. $1-\gamma$ term = true
    end if
  end while
  if term = false **then** $\hat{\boldsymbol{G}}(\boldsymbol{\theta}) = \hat{\boldsymbol{G}}(\boldsymbol{\theta}) + \left(\boldsymbol{u}(\boldsymbol{z}_t^i;\boldsymbol{\theta})\boldsymbol{u}(\boldsymbol{z}_t^i;\boldsymbol{\theta})^\top\right)/(1-\gamma)$
end for
return $\hat{\boldsymbol{G}}(\boldsymbol{\theta}) := \hat{\boldsymbol{G}}(\boldsymbol{\theta})/M$

Algorithm 4 is a pseudocode sketch of the Bayesian actor-critic algorithm, using either the conventional gradient or the natural gradient in the policy update, and with $\boldsymbol{G}$ estimated using either $\hat{\boldsymbol{G}}_t$ in Eq. 42 or $\hat{\boldsymbol{G}}(\boldsymbol{\theta})$ in Algorithm 3.

Algorithm 4 A Bayesian Actor-Critic Algorithm

BAC$(\boldsymbol{\theta}, M, \epsilon)$
- $\boldsymbol{\theta}$ initial policy parameters
- $M > 0$ episodes for gradient evaluation
- $\epsilon > 0$ termination threshold

done = false
while not done **do**
  Run **GPTD** for M episodes. **GPTD** returns $\boldsymbol{\alpha}_t$ and $\boldsymbol{C}_t$ (Eq. 34)
  Compute an estimate of the Fisher information matrix $\hat{\boldsymbol{G}}_t$ (Eq. 42) or $\hat{\boldsymbol{G}}(\boldsymbol{\theta})$ (Algorithm 3)
  Compute $\boldsymbol{U}_t$ (Proposition 6)
  $\Delta\boldsymbol{\theta} = \boldsymbol{U}_t\boldsymbol{\alpha}_t$ **(conventional gradient)** or
  $\Delta\boldsymbol{\theta} = \hat{\boldsymbol{G}}_t^{-1}\boldsymbol{U}_t\boldsymbol{\alpha}_t$ or $\Delta\boldsymbol{\theta} = \hat{\boldsymbol{G}}(\boldsymbol{\theta})^{-1}\boldsymbol{U}_t\boldsymbol{\alpha}_t$ **(natural gradient)**
  $\boldsymbol{\theta} := \boldsymbol{\theta} + \beta\Delta\boldsymbol{\theta}$
  if $|\Delta\boldsymbol{\theta}| < \epsilon$ **then** done = true
end while
return $\boldsymbol{\theta}$

7.3 BAC Online Sparsification

As was done for the BPG algorithms in Section 4.4, Algorithm 4 may be made more efficient, both in time and memory, by sparsifying the solution. Engel et al. (2005) presented a sparse approximation of the GPTD algorithm by using an online sparsification method from Engel et al. (2002). This sparsification method incrementally constructs a dictionary $\tilde{\mathcal{D}}$ of representative state-action pairs. Upon observing a new state-action pair $\boldsymbol{z}_i$, the distance between the feature-space image of $\boldsymbol{z}_i$ and the span of the images of current dictionary members is computed. If the squared distance $\delta_i = k(\boldsymbol{z}_i, \boldsymbol{z}_i) - \tilde{\boldsymbol{k}}_{i-1}^\top(\boldsymbol{z}_i)\tilde{\boldsymbol{K}}_{i-1}^{-1}\tilde{\boldsymbol{k}}_{i-1}(\boldsymbol{z}_i)$ exceeds some positive threshold τ, $\boldsymbol{z}_i$ is added to the dictionary, otherwise, it is left out. In calculating δ_i, $\tilde{\boldsymbol{k}}_{i-1}$ and $\tilde{\boldsymbol{K}}_{i-1}$ are the dictionary kernel vector and kernel matrix before observing $\boldsymbol{z}_i$, respectively. Engel et al. (2005) showed that using this sparsification procedure, the posteriors moments of Q may be compactly approximated as $\hat{Q}_t(\boldsymbol{z}) = \tilde{\boldsymbol{k}}_t^\top(\boldsymbol{z})\tilde{\boldsymbol{\alpha}}_t$ and $\hat{S}_t(\boldsymbol{z}, \boldsymbol{z}') = k(\boldsymbol{z}, \boldsymbol{z}') - \tilde{\boldsymbol{k}}_t^\top(\boldsymbol{z})\tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{k}}_t(\boldsymbol{z}')$, where

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\boldsymbol{H}}_t^\top\left(\tilde{\boldsymbol{H}}_t\tilde{\boldsymbol{K}}_t\tilde{\boldsymbol{H}}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1}\boldsymbol{r}_{t-1}\,, \qquad \tilde{\boldsymbol{C}}_t = \tilde{\boldsymbol{H}}_t^\top\left(\tilde{\boldsymbol{H}}_t\tilde{\boldsymbol{K}}_t\tilde{\boldsymbol{H}}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1}\tilde{\boldsymbol{H}}_t\,. \quad (43)$$

In Eq. 43, $\tilde{\boldsymbol{H}}_t = \boldsymbol{H}_t\boldsymbol{A}_t$, where $\boldsymbol{A}_t$ is a $|\mathcal{D}_t| \times |\tilde{\mathcal{D}}_t|$ matrix whose i'th row is $[\boldsymbol{A}]_{i,|\tilde{\mathcal{D}}_i|} = 1$ and $[\boldsymbol{A}]_{i,j} = 0\ ;\ \forall j \neq |\tilde{\mathcal{D}}_i|$, if we add the state-action pair $\boldsymbol{z}_i$ to the dictionary, and is $\tilde{\boldsymbol{k}}_{i-1}^\top(\boldsymbol{z}_i)\tilde{\boldsymbol{K}}_{i-1}^{-1}$ followed by zeros otherwise.

Proposition 7 *Using the sparsification method described above, the posterior moments of the gradient are approximated as*

$$\mathbf{E}\big[\nabla\eta(\boldsymbol{\theta})|\mathcal{D}_t\big] = \tilde{\boldsymbol{U}}_t\tilde{\boldsymbol{\alpha}}_t\ , \qquad \mathbf{Cov}\big[\nabla\eta(\boldsymbol{\theta})|\mathcal{D}_t\big] = \boldsymbol{G} - \tilde{\boldsymbol{U}}_t\tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{U}}_t^\top\ ,$$

where $\tilde{\boldsymbol{\alpha}}_t$ *and* $\tilde{\boldsymbol{C}}_t$ *are given by Eq. 43 and* $\tilde{\boldsymbol{U}}_t = \big[\boldsymbol{u}(\boldsymbol{z}_1), \ldots, \boldsymbol{u}(\boldsymbol{z}_{|\tilde{\mathcal{D}}_t|})\big]$ *with* $\boldsymbol{z}_i \in \tilde{\mathcal{D}}_t$.

Proof The proof is straightforward by plugging the sparsified posterior mean and covariance of Q with $\tilde{\boldsymbol{\alpha}}_t$ and $\tilde{\boldsymbol{C}}_t$ from Eq. 43 in Eq. 35 and following the steps until the end of Proposition 6. ■

8. BAC Experimental Results

In this section, we empirically evaluate the performance of the Bayesian actor-critic method presented in this paper in a 10-state random walk problem as well as in the widely used continuous-state-space mountain car problem (Sutton and Barto, 1998) and ship steering problem (Miller et al., 1990). In Section 8.1, we first compare BAC, Bayesian quadrature (BQ), and Monte Carlo (MC) gradient estimates in the 10-state random walk problem. We then evaluate the performance of the BAC algorithm on the same problem, and compare it with a Bayesian policy gradient (BPG) algorithm and a MC-based policy gradient (MCPG) algorithm. In Section 8.2, we compare the performance of the BAC algorithm with a MCPG algorithm on the mountain car problem. The BPG, BAC, and MCPG algorithms used in our experiments are Algorithms 2 and 4 presented in this paper, and Algorithm 1 in Baxter and Bartlett (2001), respectively. In Section 8.3, we compare the performance of the BAC algorithm with a MCPG algorithm on a problem in the ship steering domain. Similar to Section 8.2, the BAC, and MCPG algorithms used in our experiments are Algorithm 4 presented in this paper and Algorithm 1 in Baxter and Bartlett (2001), respectively.

8.1 A Random Walk Problem

In this section, we consider a 10-state random walk problem, $\mathcal{X} = \{1, 2, \ldots, 10\}$, with states arranged linearly from state 1 on the left to state 10 on the right. The agent has two actions to choose from: $\mathcal{A} = \{left, right\}$. The left wall is a retaining barrier, meaning that if the $left$ action is taken at state 1, in the next time-step the state will be 1 again. State 10 is a zero reward absorbing state. The only stochasticity in the transitions is induced by the policy, which is defined as $\mu(right|x) = 1/1 + \exp(-\theta_x)$ and $\mu(left|x) = 1 - \mu(right|x)$, for all $x \in \mathcal{X}$. Note that each state x has an independent parameter θ_x. Each episode begins at state 1 and ends when the agent reaches state 10. The mean reward is 1 for states 1–9 and is 0 for state 10. The observed rewards for states 1–9 are obtained by corrupting the mean rewards with a 0.1 standard deviation i.i.d. Gaussian noise. The discount factor is $\gamma = 0.99$. In the BAC experiments, we use the Gaussian state kernel $k_x(x, x') = \exp(-||x-x'||^2/(2\sigma_k^2))$ with $\sigma_k = 3$ and the state-action kernel $0.01k_F(\boldsymbol{z}, \boldsymbol{z}')$.

We first compare the MC, BQ, and BAC estimates of $\nabla\eta(\boldsymbol{\theta})$ for the policy induced by the parameters $\theta_x = \log(41/9)$ for all $x \in \mathcal{X}$, which is equivalent to $\mu(right|x) = 0.82$. We use several different sample sizes: $M = 5j,\ j = 1, \ldots, 20$. Here, by sample size we mean

the number of episodes used to estimate the gradient. For each value of M, we compute the gradient estimates 10^3 times. The true gradient is calculated analytically for reference. Figure 6 shows the mean squared error and the mean absolute angular error of MC, BQ, and BAC estimates of the gradient for different sample sizes M. The error bars in the right figure are the standard errors of the mean absolute angular errors. The results depicted in Figure 6 indicate that the BAC gradient estimates are more accurate and have lower variance than their MC and BQ counterparts.

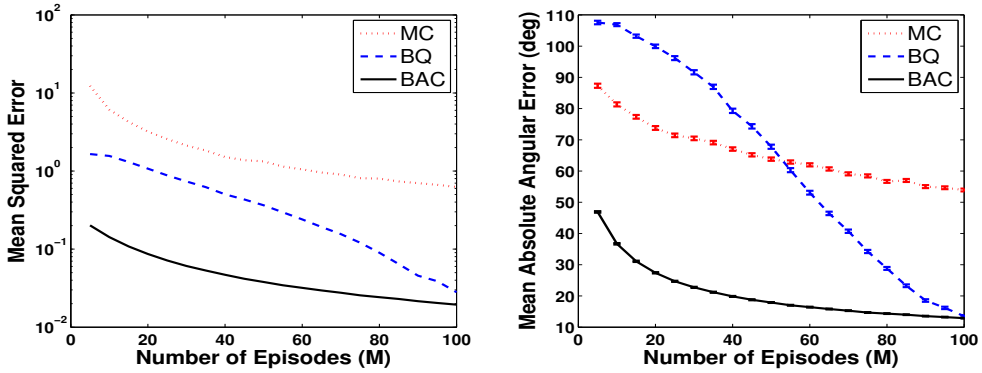

Figure 6: The mean squared error and the mean absolute angular error of MC, BQ, and BAC gradient estimations as a function of the number of sample episodes M. All results are averaged over 10^3 runs.

Next, we use BAC to optimize the policy parameters and compare its performance with a BPG algorithm and a MCPG algorithm for $M = 1$, 25, 50, and 75. The BPG algorithm uses Model 1 of Section 4.1. We use Algorithm 4 with the number of policy updates set to 500 and the same kernels as in the previous experiment. The Fisher information matrix is estimated using Algorithm 3. The returns obtained by these methods are averaged over 10^3 runs. For a fixed sample size M, we tried many values of the learning rate, β, for MCPG, BPG, and BAC, and those in Table 4 yielded the best performance. Note that the learning rate used for each algorithm in each experiment is fixed and does not converge to zero. BAC showed a very robust performance when we changed the learning rate. By robust we mean that it never generated a policy for which an episode does not end after 10^6 steps. This seems to be due to the fact that BAC gradient estimates are more accurate and have less variance than their MC and BPG counterparts. The performance of BPG improves as we increase the sample size M. It performs worse than MCPG for $M = 1$ and 25, but achieves a performance similar to BAC for $M = 100$.

Figure 7 depicts the results of these experiments. From left to right and top to bottom the sub-figures correspond to the experiment in which all the algorithms used $M =$ 1, 25, 50, and 75 trajectories per policy update, respectively. Each curve depicts the difference between the exact average discounted return for the 500 policies that follow each policy update and η^* – the optimal average discounted return. All curves are averaged over

β	$M = 1$	$M = 25$	$M = 50$	$M = 75$
MCPG	0.005	0.075	0.1	0.75
BPG	0.0035	0.015	0.09	0.5
BAC	5	5	5	5

Table 4: Learning rates used by the algorithms in the experiments of Figure 7.

10^3 repetitions of the experiment. The BAC algorithm clearly learns significantly faster than the other algorithms (note that the vertical scale is logarithmic).

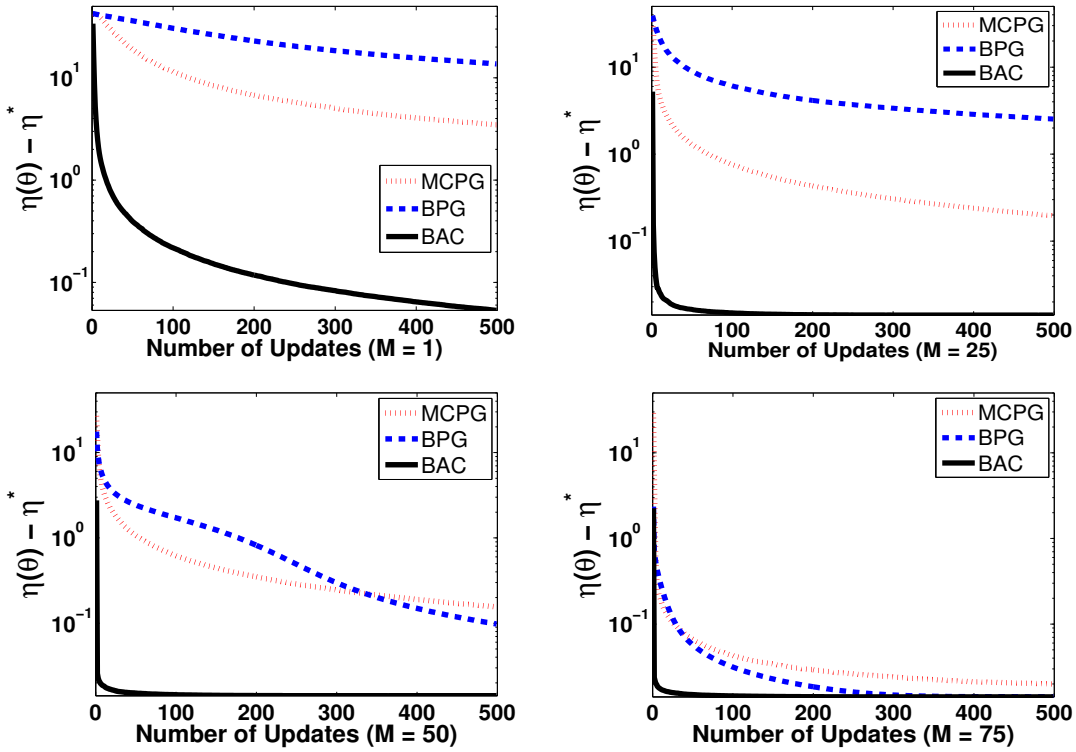

Figure 7: Results for the policy learning experiment. The graphs depict the performance of the policies learned by each algorithm during 500 policy updates. From left to right and top to bottom the number of episodes used to estimate the gradient is $M = 1,\ 25,\ 50$ and 75. All results are averaged over 10^3 independent runs.

Remark: Since BQ (and as a result BPG) is based on defining a kernel over system trajectories (quadratic Fisher kernel in Model 1 and Fisher kernel in Model 2), its performance degrades when the system generates trajectories of different size. This effect can be observed by most kernels that have been used in the literature for the trajectories generated by dynamical systems. This can be also observed in our experiments: BQ performs much

better than MC in the "Linear Quadratic Regulator" problem (Section 6.2), in which all the system trajectories are of size 20, while its superiority over MC is less apparent in the "Random Walk" problem (Section 8.1). This is why we are not going to use BQ and BPG in the "Mountain Car" (Section 8.2) and "Ship Steering" (Section 8.3) problems, in which the system trajectories have different lengths.

8.2 Mountain Car

In this section, we consider the mountain car problem as formulated in Sutton and Barto (1998), and report the results of applying the BAC and MCPG algorithms to optimize the policy parameters in this task. The state $\boldsymbol{x}$ consists of the position x and the velocity $\dot{x}$ of the car: $\boldsymbol{x} = (x, \dot{x})$. The reward is -1 on all time steps until the car reaches its goal at the top of the hill, which ends the episode. There are three possible actions: *forward*, *reverse*, and *zero*. The car moves according to the following simplified dynamics:

$$\begin{array}{lll} -1.2 \leq x_{t+1} \leq 0.5 & , & -0.07 \leq \dot{x}_{t+1} \leq 0.07 \ , \\ x_{t+1} = \text{bound}[x_t + \dot{x}_{t+1}] & , & \dot{x}_{t+1} = \text{bound}\big[\dot{x}_t + 0.001 a_t - 0.0025 \cos(3x_t)\big] \ . \end{array}$$

When x_{t+1} reaches the left boundary, $\dot{x}_{t+1}$ is set to zero and when it reaches the right boundary, the goal is reached and the episode ends. Each episode starts from a random position and velocity uniformly sampled from their domains. We use the discount factor $\gamma = 0.99$.

In order to define the policy, we first map the states $\boldsymbol{x} = (x, \dot{x})$ to the unit square $[0,1] \times [0,1]$. The policy used in our experiments has the following form:

$$\mu(a_i|\boldsymbol{x}) = \frac{\exp\big(\phi(\boldsymbol{x}, a_i)^\top \boldsymbol{\theta}\big)}{\sum_{j=1}^{3} \exp\big(\phi(\boldsymbol{x}, a_j)^\top \boldsymbol{\theta}\big)} \ , \qquad i = 1, 2, 3.$$

The policy feature vector is defined as $\phi(\boldsymbol{x}, a_i) = \big(\phi(\boldsymbol{x})^\top \delta_{a_1 a_i}, \phi(\boldsymbol{x})^\top \delta_{a_2 a_i}, \phi(\boldsymbol{x})^\top \delta_{a_3 a_i}\big)^\top$, where $\delta_{a_j a_i}$ is 1 if $a_j = a_i$, and is 0 otherwise. The state feature vector $\phi(\boldsymbol{x})$ is composed of 16 Gaussian functions arranged in a 4×4 grid over the unit square as follows:

$$\phi(\boldsymbol{x}) = \Big(\exp\big(-||\boldsymbol{x} - \bar{\boldsymbol{x}}_1||^2/(2\kappa^2)\big), \ldots, \exp\big(-||\boldsymbol{x} - \bar{\boldsymbol{x}}_{16}||^2/(2\kappa^2)\big)\Big)^\top,$$

where the $\bar{\boldsymbol{x}}_i$'s are the 16 points of the grid $\{0, 0.25, 0.5, 1\} \times \{0, 0.25, 0.5, 1\}$ and $\kappa = 1.3 \times 0.25$.

In Figure 8, we compare the performance of BAC with a MCPG algorithm for $M =$ 5, 10, 20, and 40 episodes used to estimate each gradient. For BAC, we use Algorithm 4 with the number of policy updates set to 500, a Gaussian state kernel $k_x(\boldsymbol{x}, \boldsymbol{x}') = \exp\big(-||\boldsymbol{x} - \boldsymbol{x}'||^2/(2\sigma_k^2)\big)$, with $\sigma_k = 1.3 \times 0.25$, and the state-action kernel $k_F(\boldsymbol{z}, \boldsymbol{z}')$. The Fisher information matrix is estimated using Algorithm 3. After every 50 policy updates the learned policy is evaluated for 10^3 episodes to estimate accurately the average number of steps to goal. Each evaluation episode starts from a random position and velocity uniformly chosen from their ranges, and continues until the car either reaches the goal or a limit of 200 time-steps is exceeded. The experiment is repeated 100 times for the entire horizontal

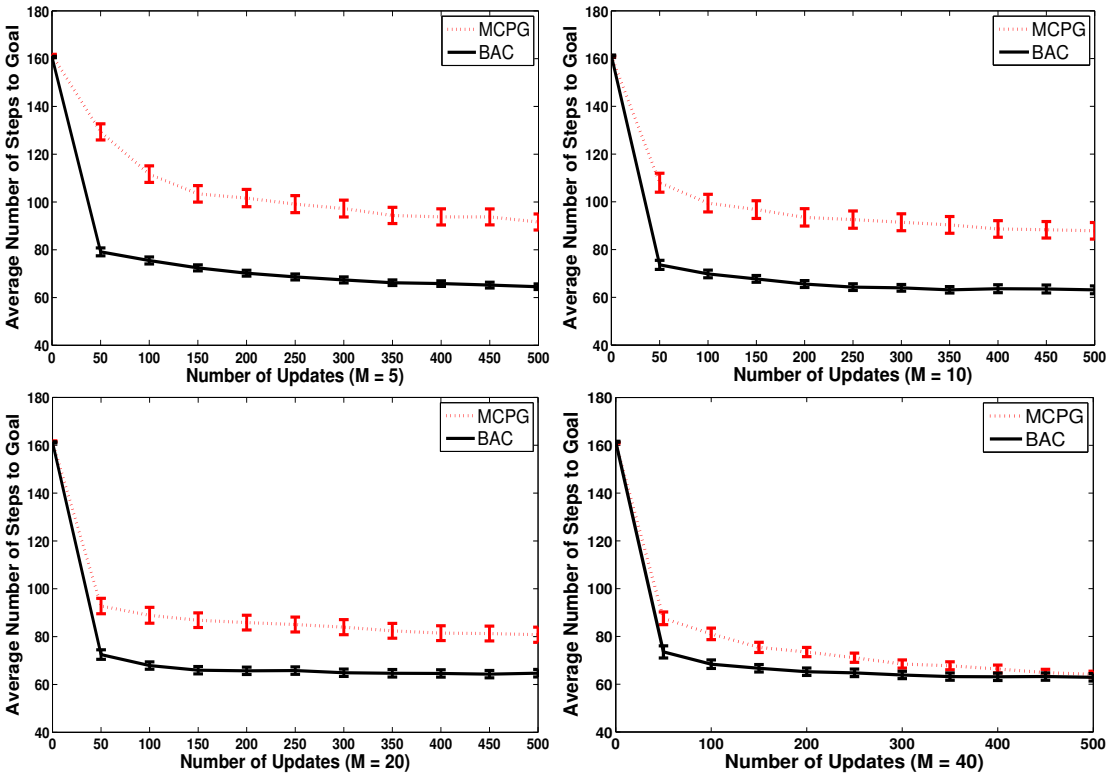

Figure 8: The graphs depict the performance of the policies learned by BAC and a MCPG algorithm during 500 policy updates in the mountain car problem. From left to right and top to bottom the number of episodes used to estimate the gradient is $M = 5,\ 10,\ 20,$ and 40. All results are averaged over 100 independent runs.

axis to obtain average results and confidence intervals. The error bars in this figure are the standard errors of the performance of the algorithms.

For a fixed sample size M, each method starts with an initial learning rate and decreases it according to the schedule $\beta_t = \beta_0 \beta_c / (\beta_c + t)$. We tried many values of the learning rate parameters (β_0, β_c) for MCPG and BAC, and those in Table 5 yielded the best performance. Note that $\beta_c = \infty$ means that we used a fixed learning rate β_0 for that experiment. The graphs indicate that BAC performs better and has lower variance than MCPG. It is able to find a good policy with only $M = 5$ sample size and its performance does not change much as the sample size is increased. On the other hand, the performance of MCPG improves and its variance is reduced as we increase the sample size. Note that for $M = 40$, MCPG finally achieves a similar performance (still with slower rate) as BAC.

β	$M = 5$	$M = 10$	$M = 20$	$M = 40$
MCPG	0.025 , ∞	0.1 , 100	0.2 , 100	0.25 , ∞
BAC	0.025 , ∞	0.05 , ∞	0.1 , ∞	0.1 , 250

Table 5: Learning rates used by the algorithms in the experiments of Figure 8.

8.3 Ship Steering

In this section, we perform comparative experiments between BAC and MCPG on a more challenging problem in the continuous state continuous action *ship steering* domain (Miller et al., 1990).

Domain Description In this domain, a ship is located in a 150×150 meter square water surface. At any point in time t, the state of the ship is described by four continuous variables that are defined below along with their range of values

$$\mathbf{x}_t = (x_t, y_t, \theta_t, \dot{\theta}_t) \in [0\text{m}, 150\text{m}] \times [0\text{m}, 150\text{m}] \times [-180^\circ, 180^\circ] \times [-15^\circ/\text{s}, 15^\circ/\text{s}],$$

where x_t and y_t represent the position of the ship, θ_t the angle between the vertical axis and the ship orientation, and $\dot{\theta}_t$ the actual turning rate (see the upper-left panel in Figure 9). At the beginning of each episode, the ship starts at $(x_1, y_1) = (40\text{m}, 40\text{m})$, with θ_1 and $\dot{\theta}_1$ sampled uniformly at random from their ranges. The only available action variable is $a_t \in [-15^\circ, 15^\circ]$, which is the *desired* turning rate. To model the ship inertia and water resistance, there is a $T = 5$ time steps lag for the desired turning rate to become the actual turning rate. Moreover, the ship moves with the constant speed of $V = 3\text{m/s}$ and $\Delta = 0.2\text{s}$ is the sampling interval. The following set of equations summarizes the ship's dynamics:

$$x_{t+1} = x_t + \Delta\, V \sin\theta_t \qquad y_{t+1} = y_t + \Delta\, V \cos\theta_t$$
$$\theta_{t+1} = \theta_t + \Delta\, \dot{\theta}_t \qquad \dot{\theta}_{t+1} = \dot{\theta}_t + \frac{\Delta}{T}(a_t - \dot{\theta}_t)$$

The goal of the ship is to navigate to $(x_*, y_*) = (100\text{m}, 100\text{m})$ within 500 times steps. If this does not happen or the ship moves out of the boundary, the episode terminates as a failure. The goal of the policy is to maximize the probability of the ship successfully reaching (x_*, y_*). Thus, we set the discount factor to $\gamma = 1$ in this problem.

Learning For both MCPG and BAC, we used a CMAC function approximator with 9 four dimensional tilings, each of them discretizing the state space into $5 \times 5 \times 36 \times 5 = 4500$ tiles. Therefore, each policy parameter $\boldsymbol{w}$ is of size $N = 9 \times 4500 = 40500$. Each state $\mathbf{x}$ is represented by a binary vector $\phi(\mathbf{x})$, where $\phi_i(\mathbf{x}) = 1$ if and only if the state $\mathbf{x}$ falls in the ith tile, and thus, $\sum_{i=1}^{N} \phi_i(\mathbf{x}) \leq 9$. To define a precise mapping from states to actions, $\boldsymbol{w}_t : \mathbf{x_t} = (x_t, y_t, \theta_t, \dot{\theta}_t) \to a_t$, we first sample $\tilde{a}_t$ from the Gaussian

$$\tilde{a}_t \sim \mathcal{N}\left(\frac{\sum_{i=1}^{N} \boldsymbol{w}_t^{(i)} \phi_i(\mathbf{x}_t)}{\sum_{i=1}^{N} \phi_i(\mathbf{x}_t)}, 1\right),$$

and then map it to the allowed range $[-15^\circ, 15^\circ]$ using the sigmoid transformation

$$a_t = 15^\circ \cdot \frac{2}{\pi} \cdot \arctan\left(\frac{\pi}{2} \cdot \tilde{a}_t\right).$$

For the BAC experiments, we used the Gaussian state kernel $k_x(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||^2/(2\sigma_k^2))$, with $\sigma_k = 1$ and the state-action kernel $k_F(\boldsymbol{z}, \boldsymbol{z}')$, i.e., the Fisher kernel.

Setup In order to improve the computational efficiency, we use several numerical approximations. First, to calculate the score function for a trajectory (Eq. 5) in both MCPG and BAC, we approximate the gradient of the action distribution in a_t with the one in $\tilde{a}_t$, i.e.,

$$\nabla \log \mu(a_t|\mathbf{x}_t; \boldsymbol{w}_t) \approx \nabla \log \mu(\tilde{a}_t|\mathbf{x}_t; \boldsymbol{w}_t).$$

Second, we calculate the gradient using the online sparsification procedure described in Section 4.4. Finaly, we never explicitly calculate the inverse of the Fisher information matrix $\hat{\boldsymbol{G}}$ and instead calculate the product of $\hat{\boldsymbol{G}}^{-1}$ with the score. For the numerical stability we also add 10^{-6} to the diagonal of $\hat{\boldsymbol{G}}$.

Similar to the other experiments in the paper, we varied the number of trajectories used to estimate the gradient of a policy as $M = 5$, 10, and 20. Table 6 shows the best values of the learning rate β for both MCPG and BAC for different values of M. To evaluate each method, we ran 100 independent learning trials. At each trial, we evaluate the performance of the policy every 100 iterations by executing it 100 (independent) times with θ_1 and $\dot{\theta}_1$ randomly sampled. For each of these execution, we observe if the ship reached (x_*, y_*) within 500 steps and estimate the policy success ratio. We set the total number of gradient updates to $T = 3000$ for $M = 5$ and 10 and to $T = 1000$ for $M = 20$.

β	$M = 5$	$M = 10$	$M = 20$
MCPG	0.01	0.01	0.01
BAC	0.5	0.4	0.5

Table 6: Learning rates used by the algorithms in the experiments of Figure 9.

Results The results for all the experiments are presented in Figure 9 along with their standard deviations. Naturally, using more trajectories for the gradient update improves both methods. However, this improvement is bigger for the BAC method. In the case of $M = 5$, MCPG produces slightly better policies at the beginning of learning, but is soon outperformed by BAC. For $M = 10$ and 20, BAC produces better policies from the beginning, especially for $M = 20$. This is consistent with the results of the other experimental domains reported in the paper. For all values of M, BAC converges to a policy with a better success ratio than MCPG. Finally, as expected, BAC has usually less variance in its performance than MCPG.

9. Discussion

In this paper, we first proposed an alternative approach to the conventional frequentist (Monte-Carlo based) policy gradient estimation procedure. Our approach is based on *Bayesian quadrature* (O'Hagan, 1991), a Bayesian method for integral evaluation. The idea is to model the gradient of the expected return with respect to the policy parameters, which is of the form of an integral, as Gaussian processes (GPs). This is done by dividing

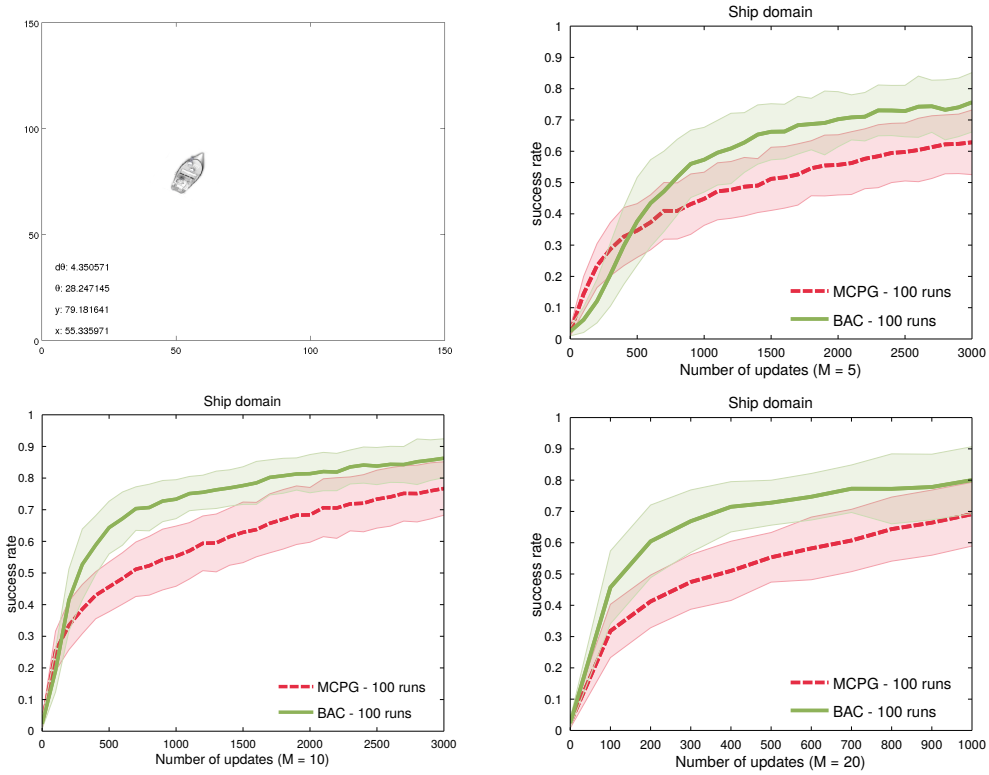

Figure 9: Success rate of the policies learned by BAC and MCPG in the ship steering problem.

the integrand into two parts, treating one as a random function (or random field), whose random nature reflects our subjective uncertainty concerning its true identity. This allows us to incorporate our prior knowledge of this term (part) into its prior distribution. Observing (possibly noisy) samples of this term allows us to employ Bayes' rule to compute a posterior distribution of it conditioned on these samples. This in turn induces a posterior distribution over the value of the integral, which is the gradient of the expected return. By properly partitioning the integrand and by appropriately selecting a prior distribution, a closed-form expression for the posterior moments of the gradient of the expected return is obtained. We proposed two different ways of partitioning the integrand resulting in two distinct Bayesian models. For each model, we showed how the posterior moments of the gradient conditioned on the observed data are calculated. In line with previous work on Bayesian quadrature, our Bayesian approach tends to significantly reduce the number of samples needed to obtain accurate gradient estimates. Moreover, estimates of the natural gradient and the gradient covariance are provided at little extra cost. We performed detailed experimental comparisons of the Bayesian policy gradient (BPG) algorithms presented in the paper with classic Monte-Carlo based algorithms on a bandit problem as well as on a linear quadratic regulator problem. The experimental results are encouraging, but we

conjecture that even better gains may be attained using this approach. This calls for additional theoretical and empirical work. It is important to note that the gradient estimated by Algorithm 1 may be employed in conjunction with conjugate-gradients and line-search methods for making better use of the gradient information. We also showed that the models and algorithms presented in this paper can be extended to partially observable problems without any change along the same lines as Baxter and Bartlett (2001). This is due to the fact that our BPG framework considers complete system trajectories as its basic observable unit, and thus, does not require the dynamic within each trajectory to be of any special form. This generality has the downside that our proposed framework cannot take advantage of the Markov property when the system is Markovian.

To address this issue, we then extended our BPG framework to actor-critic algorithms and presented a new Bayesian take on the actor-critic architecture. By using GPs and choosing their prior distributions to make them compatible with a parametric family of policies, we were able to derive closed-form expressions for the posterior distribution of the policy gradient updates. The posterior mean is used to update the policy and the posterior covariance to gauge the reliability of this update. Our Bayesian actor-critic (BAC) framework uses individual state-action-reward transitions as its basic observable unit, and thus, is able to take advantage of the Markov property of the system trajectories (when the system is indeed Markovian). This improvement seems to be borne out in our experiments, where BAC provides more accurate estimates of the policy gradient than either of the two BPG models for the same amount of data. Similar to BPG, another feature of BAC is that its natural-gradient variant is obtained at little extra cost. For both BPG and BAC, we derived the sparse form of the algorithms, which would make them significantly more time and memory efficient. Finally, we performed an experimental evaluation of the BAC algorithm, comparing it with classic Monte-Carlo based policy gradient algorithms, as well as our BPG algorithms, on a random walk problem, the widely used mountain car problem (Sutton and Barto, 1998), and the continuous state and continuous action ship steering domain (Miller et al., 1990).

Additional experimental work is required to investigate the behavior of BPG and BAC algorithms in larger and more realistic domains, involving continuous and high-dimensional state and action spaces. The BPG and BAC algorithms proposed in the paper use only the posterior mean of the gradient in their updates. We conjecture that the second-order statistics obtained from BPG and BAC (both in the actor and critic) may be used to devise more efficient algorithms. In one of the experiments in Section 6, we employed the covariance information provided by Algorithm 1 for risk-aware selection of the step size in the gradient updates, which showed promising performance. Other interesting directions for future work include **1)** investigating other possible partitions of the integrand in the expression for $\nabla\eta_B(\boldsymbol{\theta})$ into a GP term and a deterministic term, **2)** using other types of kernel functions such as sequence kernels, **3)** combining our approach with MDP model estimation to allow transfer of learning between different policies (model-based Bayesian policy gradient), and **4)** investigating more efficient methods for estimating the Fisher information matrix. Another direction is to derive a fully non-parametric actor-critic algorithm. In BAC, the critic is based on Gaussian process temporal difference learning, which is a non-parametric method, while the actor uses a family of parameterized policies. The idea here would be to

replace the actor in the BAC algorithm with a non-parametric actor that performs gradient search in a function space (e.g., a reproducing kernel Hilbert space) of policies.

Acknowledgments

Part of the computational experiments was conducted using the Grid'5000 experimental testbed (https://www.grid5000.fr). Yaakov Engel was supported by an Alberta Ingenuity fellowship.

Appendix A. Proof of Proposition 3

We start the proof with the $M \times 1$ vector $\boldsymbol{b}$, whose ith element can be written as

$$\begin{aligned}
(\boldsymbol{b})_i &= \int k(\xi, \xi_i) \Pr(\xi; \boldsymbol{\theta}) d\xi \\
&\overset{\text{(a)}}{=} \int \left(1 + \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi_i)\right)^2 \Pr(\xi; \boldsymbol{\theta}) d\xi \\
&\overset{\text{(b)}}{=} \int \Pr(\xi; \boldsymbol{\theta}) d\xi + 2 \left(\int \boldsymbol{u}(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi \right)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi_i) + \int \boldsymbol{u}(\xi_i)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi) \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi_i) \Pr(\xi; \boldsymbol{\theta}) d\xi \\
&\overset{\text{(c)}}{=} 1 + \left(\boldsymbol{u}(\xi_i)^\top \boldsymbol{G}^{-1}\right) \left(\int \boldsymbol{u}(\xi) \boldsymbol{u}(\xi)^\top \Pr(\xi; \boldsymbol{\theta}) d\xi \right) \left(\boldsymbol{G}^{-1} \boldsymbol{u}(\xi_i)\right) \\
&\overset{\text{(d)}}{=} 1 + \left(\boldsymbol{u}(\xi_i)^\top \boldsymbol{G}^{-1}\right) \boldsymbol{G} \left(\boldsymbol{G}^{-1} \boldsymbol{u}(\xi_i)\right) \overset{\text{(e)}}{=} 1 + \boldsymbol{u}(\xi_i)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi_i)
\end{aligned}$$

(a) substitutes $k(\xi, \xi_i)$ with the quadratic Fisher kernel from Eq. 23, **(b)** is algebra, **(c)** follows from **(i)** $\int \Pr(\xi; \boldsymbol{\theta}) d\xi = 1$, and **(ii)** $\int \boldsymbol{u}(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi = \int \nabla \log \Pr(\xi; \boldsymbol{\theta}) \Pr(\xi; \boldsymbol{\theta}) d\xi = \int \nabla \Pr(\xi; \boldsymbol{\theta}) d\xi = \nabla \int \Pr(\xi; \boldsymbol{\theta}) d\xi = \nabla(1) = 0$, **(d)** is the result of replacing the integral with the Fisher information matrix $\boldsymbol{G}$, **(e)** is algebra, and thus, the claim follows.

Now the proof for the scalar b_0

$$\begin{aligned}
b_0 &= \iint k(\xi, \xi') \Pr(\xi; \boldsymbol{\theta}) \Pr(\xi'; \boldsymbol{\theta}) d\xi d\xi' \\
&\overset{\text{(a)}}{=} \iint \left(1 + \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi')\right)^2 \Pr(\xi; \boldsymbol{\theta}) \Pr(\xi'; \boldsymbol{\theta}) d\xi d\xi' \\
&\overset{\text{(b)}}{=} \iint \Pr(\xi; \boldsymbol{\theta}) \Pr(\xi'; \boldsymbol{\theta}) d\xi d\xi' + 2 \iint \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi') \Pr(\xi; \boldsymbol{\theta}) \Pr(\xi'; \boldsymbol{\theta}) d\xi d\xi' \\
&+ \iint \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi') \boldsymbol{u}(\xi')^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi) \Pr(\xi; \boldsymbol{\theta}) \Pr(\xi'; \boldsymbol{\theta}) d\xi d\xi' \\
&\overset{\text{(c)}}{=} 1 + 2 \left(\int \boldsymbol{u}(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi \right)^\top \boldsymbol{G}^{-1} \left(\int \boldsymbol{u}(\xi') \Pr(\xi'; \boldsymbol{\theta}) d\xi' \right) \\
&+ \int \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1} \left(\int \boldsymbol{u}(\xi') \boldsymbol{u}(\xi')^\top \Pr(\xi'; \boldsymbol{\theta}) d\xi' \right) \boldsymbol{G}^{-1} \boldsymbol{u}(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi \\
&\overset{\text{(d)}}{=} 1 + \int \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi
\end{aligned} \tag{44}$$

(a) substitutes $k(\xi, \xi')$ with the quadratic Fisher kernel from Eq. 23, **(b)** is algebra, **(c)** follows from **(i)** $\iint \Pr(\xi; \boldsymbol{\theta}) \Pr(\xi'; \boldsymbol{\theta}) d\xi d\xi' = 1$, and **(ii)** $\int \boldsymbol{u}(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi = 0$, and finally **(d)** is the result of replacing the integral within the parentheses with the Fisher information matrix $\boldsymbol{G}$.

The Fisher information matrix $\boldsymbol{G}$ is positive definite and symmetric. Thus, it can be written as $\boldsymbol{G} = \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^\top$, where $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$ and $\boldsymbol{\Lambda} = \text{diag}[\lambda_1, \ldots, \lambda_n]$ are the matrix of orthonormal eigenvectors and the diagonal matrix of eigenvalues of matrix $\boldsymbol{G}$, respectively.

By replacing $\boldsymbol{G}^{-1}$ with $\boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^\top$ in Eq. 44 we obtain

$$
\begin{aligned}
b_0 &= 1 + \int \boldsymbol{u}^\top(\xi)\boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^\top\boldsymbol{u}(\xi)\Pr(\xi;\boldsymbol{\theta})d\xi \\
&\overset{\text{(a)}}{=} 1 + \int \left(\boldsymbol{V}^\top\boldsymbol{u}(\xi)\right)^\top \boldsymbol{\Lambda}^{-1}\left(\boldsymbol{V}^\top\boldsymbol{u}(\xi)\right)\Pr(\xi;\boldsymbol{\theta})d\xi \overset{\text{(b)}}{=} 1 + \int \left(\sum_{i=1}^n \lambda_i^{-1}\left(\boldsymbol{V}^\top\boldsymbol{u}(\xi)\right)_i^2\right)\Pr(\xi;\boldsymbol{\theta})d\xi \\
&\overset{\text{(c)}}{=} 1 + \sum_{i=1}^n \lambda_i^{-1}\left(\int \left(\boldsymbol{v}_i^\top\boldsymbol{u}(\xi)\right)^2\Pr(\xi;\boldsymbol{\theta})d\xi\right) \overset{\text{(d)}}{=} 1 + \sum_{i=1}^n \lambda_i^{-1}\left(\int \boldsymbol{v}_i^\top\boldsymbol{u}(\xi)\boldsymbol{v}_i^\top\boldsymbol{u}(\xi)\Pr(\xi;\boldsymbol{\theta})d\xi\right) \\
&\overset{\text{(e)}}{=} 1 + \sum_{i=1}^n \lambda_i^{-1}\left(\int \boldsymbol{v}_i^\top\boldsymbol{u}(\xi)\boldsymbol{u}(\xi)^\top\boldsymbol{v}_i\Pr(\xi;\boldsymbol{\theta})d\xi\right) \overset{\text{(f)}}{=} 1 + \sum_{i=1}^n \lambda_i^{-1}\boldsymbol{v}_i^\top\left(\int \boldsymbol{u}(\xi)\boldsymbol{u}(\xi)^\top\Pr(\xi;\boldsymbol{\theta})d\xi\right)\boldsymbol{v}_i \\
&\overset{\text{(g)}}{=} 1 + \sum_{i=1}^n \lambda_i^{-1}\boldsymbol{v}_i^\top\boldsymbol{G}\boldsymbol{v}_i \overset{\text{(h)}}{=} 1 + \sum_{i=1}^n \lambda_i^{-1}\boldsymbol{v}_i^\top\lambda_i\boldsymbol{v}_i = 1 + \sum_{i=1}^n \boldsymbol{v}_i^\top\boldsymbol{v}_i = 1 + \sum_{i=1}^n ||\boldsymbol{v}_i||^2 \overset{\text{(i)}}{=} 1 + n
\end{aligned}
$$

(a) and **(b)** are algebra, **(c)** is the result of switching the sum and the integral, **(d)** is algebra, **(e)** follows from the fact that $\boldsymbol{v}_i^\top\boldsymbol{u}(\xi)$ is a scalar, and thus, can be replaced by its transpose, **(f)** is algebra, **(g)** substitutes the integral within the parentheses with the Fisher information matrix $\boldsymbol{G}$, **(h)** replaces $\boldsymbol{G}\boldsymbol{v}_i$ with $\lambda_i\boldsymbol{v}_i$, **(i)** follows from the orthonormality of $\boldsymbol{v}_i$'s, and thus, the claim follows.

Appendix B. Proof of Proposition 4

We start with the proof of $\boldsymbol{B}$. This $n \times M$ matrix may be written as

$$
\begin{aligned}
\boldsymbol{B} &= \int \nabla\Pr(\xi;\boldsymbol{\theta})\boldsymbol{k}(\xi)^\top d\xi = \int \nabla\Pr(\xi;\boldsymbol{\theta})\left[k(\xi,\xi_1),\ldots,k(\xi,\xi_M)\right]d\xi \\
&\overset{\text{(a)}}{=} \int \nabla\Pr(\xi;\boldsymbol{\theta})\left[\boldsymbol{u}(\xi)^\top\boldsymbol{G}^{-1}\boldsymbol{u}(\xi_1),\ldots,\boldsymbol{u}(\xi)^\top\boldsymbol{G}^{-1}\boldsymbol{u}(\xi_M)\right]d\xi \\
&\overset{\text{(b)}}{=} \left(\int \nabla\Pr(\xi;\boldsymbol{\theta})\boldsymbol{u}(\xi)^\top d\xi\right)\boldsymbol{G}^{-1}\left[\boldsymbol{u}(\xi_1),\ldots,\boldsymbol{u}(\xi_M)\right] \\
&\overset{\text{(c)}}{=} \left(\int \boldsymbol{u}(\xi)\boldsymbol{u}(\xi)^\top\Pr(\xi;\boldsymbol{\theta})d\xi\right)\boldsymbol{G}^{-1}\left[\boldsymbol{u}(\xi_1),\ldots,\boldsymbol{u}(\xi_M)\right] \\
&\overset{\text{(d)}}{=} \boldsymbol{G}\boldsymbol{G}^{-1}\left[\boldsymbol{u}(\xi_1),\ldots,\boldsymbol{u}(\xi_M)\right] \overset{\text{(e)}}{=} \left[\boldsymbol{u}(\xi_1),\ldots,\boldsymbol{u}(\xi_M)\right] = \boldsymbol{U}
\end{aligned}
$$

(a) substitutes $k(\xi,\xi_i)$ with the Fisher kernel from Eq. 27, **(b)** is algebra, **(c)** follows from $\nabla\Pr(\xi;\boldsymbol{\theta}) = \boldsymbol{u}(\xi)\Pr(\xi;\boldsymbol{\theta})$, **(d)** substitutes the integral within the parentheses with the Fisher information matrix $\boldsymbol{G}$, **(e)** is algebra, and thus, the claim follows.

Now the proof for the $n \times n$ matrix $\boldsymbol{B}_0$

$$
\begin{aligned}
\boldsymbol{B}_0 &= \iint k(\xi,\xi')\nabla\Pr(\xi;\boldsymbol{\theta})\nabla\Pr(\xi';\boldsymbol{\theta})^\top d\xi d\xi' \\
&\overset{\text{(a)}}{=} \iint \nabla\Pr(\xi;\boldsymbol{\theta})k(\xi,\xi')\nabla\Pr(\xi';\boldsymbol{\theta})^\top d\xi d\xi' \\
&\overset{\text{(b)}}{=} \iint \left(\boldsymbol{u}(\xi)\Pr(\xi;\boldsymbol{\theta})\right)\boldsymbol{u}(\xi)^\top\boldsymbol{G}^{-1}\boldsymbol{u}(\xi')\left(\boldsymbol{u}(\xi')\Pr(\xi';\boldsymbol{\theta})\right)^\top d\xi d\xi' \\
&\overset{\text{(c)}}{=} \left(\int \boldsymbol{u}(\xi)\boldsymbol{u}(\xi)^\top\Pr(\xi;\boldsymbol{\theta})d\xi\right)\boldsymbol{G}^{-1}\left(\int \boldsymbol{u}(\xi')\boldsymbol{u}(\xi')^\top\Pr(\xi';\boldsymbol{\theta})d\xi'\right) \overset{\text{(d)}}{=} \boldsymbol{G}\boldsymbol{G}^{-1}\boldsymbol{G} = \boldsymbol{G}
\end{aligned}
$$

(a) follows from the fact that $k(\xi, \xi')$ is scalar, **(b)** substitutes $k(\xi, \xi')$ with the Fisher information kernel from Eq. 27 and $\nabla \Pr(\xi; \boldsymbol{\theta})$ with $\boldsymbol{u}(\xi) \Pr(\xi; \boldsymbol{\theta})$, **(c)** is algebra, **(d)** is the result of substituting the integrals within the parentheses with the Fisher information matrix $\boldsymbol{G}$, and thus, the claim follows.

Appendix C. Proof of Proposition 5

Here we only show the proof for Model 1, the proof for Model 2 is straightforward following the same arguments. The sparse approximations of the kernel matrix $\boldsymbol{K}$ and kernel vector $\boldsymbol{k}(\cdot)$ may be written as $\boldsymbol{K} \approx \boldsymbol{A}\tilde{\boldsymbol{K}}\boldsymbol{A}^\top$ and $\boldsymbol{k}(\cdot) \approx \boldsymbol{A}\tilde{\boldsymbol{k}}(\cdot)$, respectively (Eqs. 2.2.8 and 2.2.9 in Engel, 2005). If we replace $\boldsymbol{K}$ and $\boldsymbol{k}(\cdot)$ in Eq. 21 with their sparse approximations, we obtain

$$\begin{aligned}
\mathbf{E}\big[\nabla \eta_B(\boldsymbol{\theta}) | \mathcal{D}_M\big] &= \boldsymbol{Y}\big(\boldsymbol{A}\tilde{\boldsymbol{K}}\boldsymbol{A}^\top + \boldsymbol{\Sigma}\big)^{-1}\boldsymbol{b}, \\
\mathbf{Cov}\big[\nabla \eta_B(\boldsymbol{\theta}) | \mathcal{D}_M\big] &= \Big[b_0 - \boldsymbol{b}^\top\big(\boldsymbol{A}\tilde{\boldsymbol{K}}\boldsymbol{A}^\top + \boldsymbol{\Sigma}\big)^{-1}\boldsymbol{b}\Big]\boldsymbol{I}. \qquad (45)
\end{aligned}$$

Sparsification does not change b_0 and it remains equal to $n+1$ (see Proposition 3), however it modifies $\boldsymbol{b}$ to

$$\boldsymbol{b} = \int \boldsymbol{k}(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi = \int \boldsymbol{A}\tilde{\boldsymbol{k}}(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi = \boldsymbol{A} \int \tilde{\boldsymbol{k}}(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi = \boldsymbol{A}\tilde{\boldsymbol{b}},$$

where $\tilde{\boldsymbol{b}} = \int \tilde{\boldsymbol{k}}(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi$ is exactly $\boldsymbol{b}$, only the kernel vector $\boldsymbol{k}(\cdot)$ has been replaced by the sparse kernel vector $\tilde{\boldsymbol{k}}(\cdot)$. Thus using Proposition 3, we have $(\tilde{\boldsymbol{b}})_i = 1 + \boldsymbol{u}(\xi_i)^\top \boldsymbol{G}^{-1}\boldsymbol{u}(\xi_i)$, with $\xi_i \in \tilde{\mathcal{D}}$. By replacing $\boldsymbol{b}$ with $\boldsymbol{A}\tilde{\boldsymbol{b}}$ in Eq. 45, we obtain

$$\begin{aligned}
\mathbf{E}\big[\nabla \eta_B(\boldsymbol{\theta}) | \mathcal{D}_M\big] &= \boldsymbol{Y}\big(\boldsymbol{A}\tilde{\boldsymbol{K}}\boldsymbol{A}^\top + \boldsymbol{\Sigma}\big)^{-1}\boldsymbol{A}\tilde{\boldsymbol{b}} = \boldsymbol{Y}\boldsymbol{\Sigma}^{-1}\big(\boldsymbol{A}\tilde{\boldsymbol{K}}\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1} + \boldsymbol{I}\big)^{-1}\boldsymbol{A}\tilde{\boldsymbol{b}}, \\
\mathbf{Cov}\big[\nabla \eta_B(\boldsymbol{\theta}) | \mathcal{D}_M\big] &= \Big(b_0 - \tilde{\boldsymbol{b}}^\top\boldsymbol{A}^\top\big(\boldsymbol{A}\tilde{\boldsymbol{K}}\boldsymbol{A}^\top + \boldsymbol{\Sigma}\big)^{-1}\boldsymbol{A}\tilde{\boldsymbol{b}}\Big)\boldsymbol{I} = \Big(b_0 - \tilde{\boldsymbol{b}}^\top\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\big(\boldsymbol{A}\tilde{\boldsymbol{K}}\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1} + \boldsymbol{I}\big)^{-1}\boldsymbol{A}\tilde{\boldsymbol{b}}\Big)\boldsymbol{I}.
\end{aligned}$$

The claim follows using Lemma 1.3.2 in Engel (2005).

Appendix D. Proof of Proposition 6

We start the proof with the $n \times (t+1)$ matrix $\boldsymbol{B}_t$, whose ith column may be written as

$$\begin{aligned}
(\boldsymbol{B}_t)_i &= \int_{\mathcal{Z}} d\boldsymbol{z}\boldsymbol{g}(\boldsymbol{z}; \boldsymbol{\theta}) k(\boldsymbol{z}, \boldsymbol{z}_i) = \int_{\mathcal{Z}} d\boldsymbol{z}\pi^\mu(\boldsymbol{z}) \nabla \log \mu(a|x; \boldsymbol{\theta}) \Big(k_x(x, x_i) + k_F(\boldsymbol{z}, \boldsymbol{z}_i)\Big) \\
&= \int_{\mathcal{Z}} d\boldsymbol{z}\pi^\mu(\boldsymbol{z}) \nabla \log \mu(a|x; \boldsymbol{\theta}) k_x(x, x_i) + \int_{\mathcal{Z}} d\boldsymbol{z}\pi^\mu(\boldsymbol{z}) \nabla \log \mu(a|x; \boldsymbol{\theta}) k_F(\boldsymbol{z}, \boldsymbol{z}_i) \\
&= \int_{\mathcal{X}} dx\nu^\mu(x) k_x(x, x_i) \int_{\mathcal{A}} da\mu(a|x; \boldsymbol{\theta}) \nabla \log \mu(a|x; \boldsymbol{\theta}) + \int_{\mathcal{Z}} d\boldsymbol{z}\pi^\mu(\boldsymbol{z}) \boldsymbol{u}(\boldsymbol{z}) \boldsymbol{u}(\boldsymbol{z})^\top \boldsymbol{G}^{-1}\boldsymbol{u}(\boldsymbol{z}_i) \\
&= \int_{\mathcal{X}} dx\nu^\mu(x) k_x(x, x_i) \int_{\mathcal{A}} da \nabla \mu(a|x; \boldsymbol{\theta}) + \left(\int_{\mathcal{Z}} d\boldsymbol{z}\pi^\mu(\boldsymbol{z}) \boldsymbol{u}(\boldsymbol{z}) \boldsymbol{u}(\boldsymbol{z})^\top\right) \boldsymbol{G}^{-1}\boldsymbol{u}(\boldsymbol{z}_i) \\
&= \int_{\mathcal{X}} dx\nu^\mu(x) k_x(x, x_i) \nabla \left(\int_{\mathcal{A}} da\mu(a|x; \boldsymbol{\theta})\right) + \boldsymbol{G}\boldsymbol{G}^{-1}\boldsymbol{u}(\boldsymbol{z}_i) \\
&= \int_{\mathcal{X}} dx\nu^\mu(x) k_x(x, x_i) \nabla(1) + \boldsymbol{u}(\boldsymbol{z}_i) = \boldsymbol{u}(\boldsymbol{z}_i)
\end{aligned}$$

The 1st line follows from the definition of matrix $\boldsymbol{B}_t$, function $\boldsymbol{g}$, and kernel k, the 2nd line is algebra, the 3rd line follows from the definition of π^μ and the Fisher kernel k_F, the 4th line is algebra, the 5th line is the result of replacing the integral in the parentheses with the Fisher information matrix $\boldsymbol{G}$, finally the 6th line is algebra, and the claim follows.

Now the proof for the $n \times n$ matrix $\boldsymbol{B}_0$

$$
\begin{aligned}
\boldsymbol{B}_0 &= \int_{\mathcal{Z}^2} d\boldsymbol{z} d\boldsymbol{z}' \boldsymbol{g}(\boldsymbol{z};\boldsymbol{\theta}) k(\boldsymbol{z},\boldsymbol{z}') \boldsymbol{g}(\boldsymbol{z}';\boldsymbol{\theta})^\top \\
&\overset{\text{(a)}}{=} \int_{\mathcal{Z}^2} d\boldsymbol{z} d\boldsymbol{z}' \pi^\mu(\boldsymbol{z}) \nabla \log \mu(a|x;\boldsymbol{\theta}) \Big(k_x(x,x') + k_F(\boldsymbol{z},\boldsymbol{z}') \Big) \nabla \log \mu(a'|x';\boldsymbol{\theta})^\top \pi^\mu(\boldsymbol{z}') \\
&\overset{\text{(b)}}{=} \int_{\mathcal{Z}^2} d\boldsymbol{z} d\boldsymbol{z}' \pi^\mu(\boldsymbol{z}) \nabla \log \mu(a|x;\boldsymbol{\theta}) k_x(x,x') \nabla \log \mu(a'|x';\boldsymbol{\theta})^\top \pi^\mu(\boldsymbol{z}') \\
&+ \int_{\mathcal{Z}^2} d\boldsymbol{z} d\boldsymbol{z}' \pi^\mu(\boldsymbol{z}) \nabla \log \mu(a|x;\boldsymbol{\theta}) k_F(\boldsymbol{z},\boldsymbol{z}') \nabla \log \mu(a'|x';\boldsymbol{\theta})^\top \pi^\mu(\boldsymbol{z}') \\
&\overset{\text{(c)}}{=} \int_{\mathcal{X}^2} dx dx' \nu^\mu(x) \nu^\mu(x') k_x(x,x') \int_{\mathcal{A}^2} da da' \mu(a|x;\boldsymbol{\theta}) \nabla \log \mu(a|x;\boldsymbol{\theta}) \nabla \log \mu(a'|x';\boldsymbol{\theta})^\top \mu(a'|x';\boldsymbol{\theta}) \\
&+ \int_{\mathcal{Z}^2} d\boldsymbol{z} d\boldsymbol{z}' \pi^\mu(\boldsymbol{z}) \boldsymbol{u}(\boldsymbol{z}) \boldsymbol{u}(\boldsymbol{z})^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\boldsymbol{z}') \boldsymbol{u}(\boldsymbol{z}')^\top \pi^\mu(\boldsymbol{z}') \\
&\overset{\text{(d)}}{=} \int_{\mathcal{X}^2} dx dx' \nu^\mu(x) \nu^\mu(x') k_x(x,x') \int_{\mathcal{A}} da \nabla \mu(a|x;\boldsymbol{\theta}) \int_{\mathcal{A}} da' \nabla \mu(a'|x';\boldsymbol{\theta})^\top \\
&+ \left(\int_{\mathcal{Z}} d\boldsymbol{z} \pi^\mu(\boldsymbol{z}) \boldsymbol{u}(\boldsymbol{z}) \boldsymbol{u}(\boldsymbol{z})^\top \right) \boldsymbol{G}^{-1} \left(\int_{\mathcal{Z}} d\boldsymbol{z}' \pi^\mu(\boldsymbol{z}') \boldsymbol{u}(\boldsymbol{z}') \boldsymbol{u}(\boldsymbol{z}')^\top \right) = \boldsymbol{G} \boldsymbol{G}^{-1} \boldsymbol{G} = \boldsymbol{G}
\end{aligned}
$$

(a) follows from the definition of function $\boldsymbol{g}$ and kernel k, **(b)** is algebra, **(c)** follows from the definition of π^μ and the Fisher kernel k_F, **(c)** is algebra, finally **(d)** follows from $\int_{\mathcal{A}} da \nabla \mu(a|x;\boldsymbol{\theta}) = 0$ and $\boldsymbol{G} = \int_{\mathcal{Z}} dz \pi^\mu(\boldsymbol{z}) \boldsymbol{u}(\boldsymbol{z}) \boldsymbol{u}(\boldsymbol{z})^\top$, and the claim follows.

References

D. Aberdeen and J. Baxter. Policy-gradient learning of controllers with internal state. Technical report, Australian National University, 2001.

V. Aleksandrov, V. Sysoyev, and V. Shemeneva. Stochastic optimization. *Engineering Cybernetics*, 5:11–16, 1968.

J. Bagnell and J. Schneider. Covariant policy search. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

L. Baird. Advantage updating. Technical Report WL-TR-93-1146, Wright Laboratory, 1993.

A. Barto, R. Sutton, and C. Anderson. Neuron-like elements that can solve difficult learning control problems. *IEEE Transaction on Systems, Man and Cybernetics*, 13:835–846, 1983.

J. Baxter and P. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

J. Baxter, P. Bartlett, and L. Weaver. Experiments with infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:351–381, 2001.

J. Berger and R. Wolpert. *The Likelihood Principle*. Institute of Mathematical Statistics, Hayward, CA, 1984.

D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee. Incremental natural actor-Critic algorithms. In *Proceedings of Advances in Neural Information Processing Systems 20*, pages 105–112, 2007.

S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor-critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14:641–668, 2002.

Y. Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, The Hebrew University of Jerusalem, Israel, 2005.

Y. Engel, S. Mannor, and R. Meir. Sparse online greedy support vector regression. In *Proceedings of the Thirteenth European Conference on Machine Learning*, pages 84–96, 2002.

Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 154–161, 2003.

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *Proceedings of the Twenty Second International Conference on Machine Learning*, pages 201–208, 2005.

M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. In *Proceedings of Advances in Neural Information Processing Systems 19*, pages 457–464, 2006.

M. Ghavamzadeh and Y. Engel. Bayesian Actor-Critic algorithms. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pages 297–304, 2007.

P. Glynn. Stochastic approximation for Monte Carlo optimization. In *Proceedings of the Winter Simulation Conference*, pages 356–365, 1986.

P. Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33:75–84, 1990.

P. Glynn and P. L'Ecuyer. Likelihood ratio gradient estimation for regenerative stochastic recursions. *Advances in Applied Probability*, 27(4):1019–1053, 1995.

E. Greensmith, P. Bartlett, and J. Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2004.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings of Advances in Neural Information Processing Systems 11*, 1999.

S. Kakade. A natural policy gradient. In *Proceedings of Advances in Neural Information Processing Systems 14*, 2002.

H. Kimura, M. Yamamura, and S. Kobayashi. Reinforcement learning by stochastic hill-climbing on discounted reward. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 295–303, 1995.

V. Konda and J. Tsitsiklis. Actor-Critic algorithms. In *Proceedings of Advances in Neural Information Processing Systems 12*, pages 1008–1014, 2000.

P. Marbach. *Simulated-Based Methods for Markov Decision Processes*. PhD thesis, Massachusetts Institute of Technology, 1998.

W. Miller, R. Sutton, and P. Werbos. *Neural Networks for Control*. MIT Press, 1990.

A. O'Hagan. Monte-Carlo is fundamentally unsound. *The Statistician*, 36:247–249, 1987.

A. O'Hagan. Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, 29: 245–260, 1991.

J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

J. Peters, S. Vijayakumar, and S. Schaal. Reinforcement learning for humanoid robotics. In *Proceedings of the Third IEEE-RAS International Conference on Humanoid Robots*, 2003.

J. Peters, S. Vijayakumar, and S. Schaal. Natural actor-critic. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 280–291, 2005.

H. Poincaré. *Calcul des Probabilités*. Georges Carré, Paris, 1896.

M. Puterman. *Markov Decision Processes*. Wiley Interscience, 1994.

C. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. In *Proceedings of Advances in Neural Information Processing Systems 15*, pages 489–496, 2003.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

M. Reiman and A. Weiss. Sensitivity analysis via likelihood ratios. In *Proceedings of the Winter Simulation Conference*, 1986.

M. Reiman and A. Weiss. Sensitivity analysis for simulations via likelihood ratios. *Operations Research*, 37, 1989.

R. Rubinstein. *Some Problems in Monte Carlo Optimization*. PhD thesis, Polytechnic Institute, Riga, Latvia, 1969.

G. Rummery and M. Niranjan. *On-line Q-learning using Connectionist Systems*. Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University, 1994.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

R. Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

R. Sutton and A. Barto. *An Introduction to Reinforcement Learning*. MIT Press, 1998.

R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of Advances in Neural Information Processing Systems 12*, pages 1057–1063, 2000.

N. Vien, H. Yu, and T. Chung. Hessian matrix distribution for Bayesian policy gradient reinforcement learning. *Information Sciences*, 181(9):1671–1685, 2011.

L. Weaver and N. Tao. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth International Conference on Uncertainty in Artificial Intelligence*, pages 538–545, 2001.

R. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.