Policy Search for Path Integral Control

Vicenç Gómez[1,2], Hilbert J Kappen[2], Jan Peters[3,4], and Gerhard Neumann[3]

[1] Universitat Pompeu Fabra, Barcelona
Department of Information and Communication Technologies,
E-08018 Barcelona, Spain
vicen.gomez@upf.edu

[2] Radboud University, Nijmegen,
Donders Institute for Brain, Cognition and Behaviour
6525 AJ Nijmegen, The Netherlands
b.kappen@science.ru.nl

[3] Technische Universität Darmstadt,
Intelligent Autonomous Systems,
Hochschulstr. 10, 64289 Darmstadt, Germany
{neumann,peters}@ias.tu-darmstadt.de
[4] Max Planck Institute for Intelligent Systems,
Spemannstr. 38, 72076 Tübingen, Germany

Abstract. Path integral (PI) control defines a general class of control problems for which the optimal control computation is equivalent to an inference problem that can be solved by evaluation of a path integral over state trajectories. However, this potential is mostly unused in real-world problems because of two main limitations: first, current approaches can typically only be applied to learn open-loop controllers and second, current sampling procedures are inefficient and not scalable to high dimensional systems. We introduce the efficient Path Integral Relative-Entropy Policy Search (PI-REPS) algorithm for learning feedback policies with PI control. Our algorithm is inspired by information theoretic policy updates that are often used in policy search. We use these updates to approximate the state trajectory distribution that is known to be optimal from the PI control theory. Our approach allows for a principled treatment of different sampling distributions and can be used to estimate many types of parametric or non-parametric feedback controllers. We show that PI-REPS significantly outperforms current methods and is able to solve tasks that are out of reach for current methods.

Keywords: Path Integrals, Stochastic Optimal Control, Policy Search

1 Introduction

Stochastic Optimal Control is a powerful framework for computing optimal controllers in noisy systems with continuous states and actions. Optimal control computation usually involves estimation of the value function (or optimal cost-to-go) which, except for the simplest case of a linear system with quadratic rewards and Gaussian noise, is hard to perform exactly. In all other cases, we either have to rely on approximations of the system dynamics, e.g. by linearizations [22] or the value function [12][5]. However, such approximations can significantly degenerate the quality of the estimated controls and hinder the application for complex, non-linear tasks.

Path integral (PI) control theory [7,20] defines a general class of stochastic optimal control problems for which the optimal cost-to-go (and the optimal control) is given explicitly in terms of a path integral. Its computation only involves the path costs of sample roll-outs or (state) trajectories, which are given by the reward along the state trajectory plus the log-probability of the trajectory under the uncontrolled dynamics. The optimal trajectory distribution of the system corresponds to a soft-max probability distribution that uses the path costs in its exponent. This fact allows for using probabilistic inference methods for the computation of the optimal controls, which is one of the main reasons why PI control theory has recently gained a lot of popularity.

However, PI control theory suffers from limitations that reduce its direct application in real-world problems. First, to compute the optimal control, one has to sample many trajectories starting from a certain (initial) state $\mathbf{x}_0$. Such procedure is clearly infeasible for real stochastic environments, as the re-generation of a large number of sample trajectories would be required for each time-step. Hence, current algorithms based on PI control theory are so far limited to optimize state-independent controllers, such as open-loop torque control [19] or parametrized movement primitives such as Dynamic Movement Primitives [18,5].

Second, PI control theory requires sampling from the uncontrolled process. Such procedure requires a huge amount of samples in order to reach areas with low path costs. While open-loop iterative approaches [19] address this problem by importance sampling using a mean control trajectory, they do not provide a principled treatment for adjusting also the variance of the sampling policy. As the uncontrolled process might have small variance, such procedure still takes a large amount of samples to converge to the optimal policy. While some approaches that are used in practice relax these theoretical conditions and also change the sampling variance heuristically [16], they disregard the theoretical basis of PI control and are also restricted to open-loop controllers.

In this paper we introduce Path Integral Relative-Entropy Policy Search (PI-REPS), a new policy search approach that learns to sample from the optimal state trajectory distribution. We reuse insights from the policy search community and require that the information loss of the trajectory distribution update is bounded [11]. Such strategy ensures a stable and smooth learning process. However, instead of explicitly maximizing the expected reward as it is typically done in policy search, our aim is now to approximate the optimal state trajectory distribution obtained by PI control. This computation involves minimizing the Kullback-Leibler (KL) divergence between the trajectory distribution obtained after the policy update and the desired distribution under additional constraints. PI-REPS includes the probability distribution of the initial state $\mathbf{x}_0$ in the KL optimization. This allows direct applicability of the method for learning state feedback controllers and leads to an improvement in terms of sampling efficiency.

In the next section we review current control methods based on path integral theory. In section 3, we describe in detail PI-REPS. In section 4, we show empirically that

[5] In [12], the function that is approximated is called desirability function which corresponds to the exp-transformed value function.

PI-REPS outperforms current PI-based and policy search methods on a double-link swing-up as well as on a quad-link swing-up problem.

2 Path Integral Control

We now briefly review the concepts of PI control that are relevant for the present paper. We consider the following stochastic dynamics of the state vector $\mathbf{x}_t \in \mathbb{R}^n$ under controls $\mathbf{u}_t \in \mathbb{R}^m$

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t)dt + \mathbf{G}(\mathbf{x}_t)(\mathbf{u}_t dt + d\boldsymbol{\xi}_t) \tag{1}$$

where $\boldsymbol{\xi}_t$ is $m-$dimensional Wiener noise with covariance $\boldsymbol{\Sigma}_\mathbf{u} \in \mathbb{R}^{m\times m}$ and $\mathbf{f}$ and $\mathbf{G}$ are arbitrary functions. For zero control, the system is driven uniquely by the deterministic drift $\mathbf{f}(\mathbf{x}_t)dt = \mathbf{f}_t dt$ and the local diffusion $\mathbf{G}(\mathbf{x}_t)d\boldsymbol{\xi}_t = \mathbf{G}_t d\boldsymbol{\xi}_t$. The cost-to-go is defined as an expectation over all trajectories starting at $\mathbf{x}_0$ with control path $\mathbf{u}_{0:T-dt}$

$$J(\mathbf{x}_0, \mathbf{u}_{0:T-dt}) = \left\langle r_T(\mathbf{x}_T) + \sum_{t=0}^{T-dt} C_t(\mathbf{x}_t, \mathbf{u}_t)dt \right\rangle \tag{2}$$

The terms $r_T(\mathbf{x}_T)$ and $C_t(\mathbf{x}_t, \mathbf{u}_t)$ denote the cost at end-time T and the immediate (running) cost respectively. $C_t(\mathbf{x}_t, \mathbf{u}_t)$ is expressed as a sum of an arbitrary state-dependent term $r_t(\mathbf{x}_t)$ and a quadratic control term $\mathbf{u}_t^\intercal \mathbf{R} \mathbf{u}_t$

$$C_t(\mathbf{x}_t, \mathbf{u}_t) = r_t(\mathbf{x}_t) + \frac{1}{2}\mathbf{u}_t^\intercal \mathbf{R} \mathbf{u}_t.$$

Minimization of (2) leads to the Hamilton-Jacobi-Bellman (HJB) equations, which in the general case are non-linear, second order partial differential equations. However, if the cost matrix and noise covariance are such that $\mathbf{R} = \lambda {\boldsymbol{\Sigma}_\mathbf{u}}^{-1}$ the resulting equation is *linear* in the exponentially transformed cost-to-go function $\Psi(\mathbf{x}_0)$, where $J(\mathbf{x}_0) = -\lambda \log \Psi(\mathbf{x}_0)$. The function $\Psi(\mathbf{x}_0)$ is called desirability function. The solution for $\Psi(\mathbf{x}_0)$ using the optimal controls is given by the Feynman-Kac formula as a path integral [7]

$$\Psi(\mathbf{x}_0) = \int p_{\text{uc}}(\boldsymbol{\tau}|\mathbf{x}_0) \exp\left(-\frac{\sum_{t=0}^{T} r_t(\mathbf{x}_t)}{\lambda}\right) d\boldsymbol{\tau} \tag{3}$$

where $p_{\text{uc}}(\boldsymbol{\tau}|\mathbf{x}_0)$ is the conditional probability of a state trajectory $\boldsymbol{\tau} = \mathbf{x}_{dt:T}$ starting at $\mathbf{x}_0$ and following the uncontrolled process.

The relation $\mathbf{R} = \lambda {\boldsymbol{\Sigma}_\mathbf{u}}^{-1}$ forces control and noise to act in the same dimensions, but in an inverse relation. Thus, for fixed λ, the larger the noise, the cheaper the control and vice-versa. Parameter λ can be seen as a temperature: higher values of λ result in optimal solutions that are closer to the uncontrolled process.

Define the path value of a trajectory $\boldsymbol{\tau}$ as

$$S(\boldsymbol{\tau}|\mathbf{x}_0) = \sum_{t=0}^{T} r_t(\mathbf{x}_t) - \lambda \log p_{\text{uc}}(\boldsymbol{\tau}|\mathbf{x}_0). \tag{4}$$

The optimal path distribution can be obtained from (3) and is given by

$$p^*(\boldsymbol{\tau}|\mathbf{x}_0) = \frac{\exp(-S(\boldsymbol{\tau}|\mathbf{x}_0)/\lambda)}{\int \exp(-S(\boldsymbol{\tau}|\mathbf{x}_0)/\lambda) d\boldsymbol{\tau}}. \tag{5}$$

The optimal control is given as an expectation of the first direction of the noise $d\boldsymbol{\xi}_0$ over the optimal trajectory distribution (5). This is an inference problem that can be solved using Monte Carlo methods, e.g, by forward sampling from the uncontrolled process, as proposed in [7]. However, as the optimal trajectory distribution depends on the initial state $\mathbf{x}_0$, the sampling process has to be repeated at each state which limits the application of PI control in practice. This restriction can be ignored as in [3], at the cost of losing theoretical guarantees of optimality.

2.1 Iterative Path Integral Control

Sampling from uncontrolled process will often result in a poor estimate of the optimal trajectory distribution as the uncontrolled process typically leads to areas of high state costs, i.e., most generated trajectories will have very small probability under the optimal trajectory distribution. Formally, the main problem is the evaluation of the integral in the normalization of equation (5), as this integral is performed over the whole trajectory space. To alleviate this problem, importance sampling schemes that use a (baseline) controlled process to improve the sampling efficiency has been proposed [7]. In this case, the path cost (4) has to be corrected for the extra drift term introduced by the baseline control. An iterative version of this approach was formally derived in [19] and has resulted in several applications [14,2].

There are two main problems with this approach: first, it only considers the mean control trajectory. Since it neglects the state-dependence of the control beyond the initial state, the result is an open-loop controller that may perform poorly when applied to a stochastic system. Second, this approach does not provide a principled treatment for adapting the sampling variance, and hence, might need a large amount of samples if the variance of the uncontrolled process is low.

2.2 Policy Improvement with Path Integrals (PI^2)

Inspired by the PI theory, [18] introduced the PI^2 algorithm in the reinforcement learning community, which has been successfully applied to a variety of robotic systems for tasks such as planning, gain scheduling and variable stiffness control [3,15,17].

PI^2 uses parametrized policies to represent trajectories in the state space. Typically, PI^2 uses open-loop controllers such as Dynamic Movement Primitives (DMPs) [6]. PI^2 identifies the parameters $\boldsymbol{\theta}_t$ of the DMP with the control commands $\mathbf{u}_t$ in eq. (1). Such strategy, however, renders the constraint $\boldsymbol{R} = \lambda \boldsymbol{\Sigma}_{\mathbf{u}}{}^{-1}$ meaningless. This constraint is also often neglected which might even lead to better performance [16]. The method is model-free in the sense that no model needs to be learned. However, it is implicitly assumed that all the noise of the system is generated by the exploration in the DMP parameters, which is an unrealistic assumption. The noise $\boldsymbol{\xi}_t$ in PI^2 is interpreted as user controlled exploration noise that acts on $\boldsymbol{\theta}_t$.

2.3 Kullback Leibler divergence minimization

The PI class of control problems is included in a larger class of (discrete) problems, also known as linearly solvable Markov Decision Processes (MDP) or KL-control [20,21,8] for which the control cost can be expressed as a KL divergence between a controlled process $p(\boldsymbol{\tau}|\mathbf{x}_0)$ and $p_{\text{uc}}(\boldsymbol{\tau}|\mathbf{x}_0)$.

Unlike the continuous case where the controls act as a drift on the uncontrolled process (1), the controls in the discrete case can fully reshape the state-transition probabilities $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$, with the only restriction of being compatible with the uncontrolled process, i.e. $p(\mathbf{x}_{t+1}|\mathbf{x}_t) = 0, \forall \mathbf{x}_{t+1}$ such that $p_{\text{uc}}(\mathbf{x}_{t+1}|\mathbf{x}_t) = 0$. Policy iteration algorithms for that broader class of problems also consider KL minimization have been proposed recently in [1,12]. However, in continuous state spaces, these approaches typically rely on an iterative approximation of the desirability function. Similar to value function approximation, the errors of such approximation can accumulate and damage the policy update. Moreover, these methods do not provide a principled treatment for setting the variance of the sampling policy. Another extension of the PI control theory can be found in [13], where the path integrals are embedded in a reproducing Kernel Hilbert Space (RKHS). While this is also a promising approach, it again relies on approximation of the desirability function $\Psi(\mathbf{x})$ which we typically want to avoid.

In the area of policy search, a common approach is to bound the KL-divergence between the old and the new policy. A bound on the KL-divergence can be more efficient as penalizing the KL for determining the policy update as we obtain a pre-specified step-size of the policy update in the space of probability distributions. This step-size enables to control the exploration that is performed by the policy update in a more principled way. This insight has led to the development of several successful policy search algorithms, such as the relative entropy policy search (REPS) algorithm [11], contextual REPS [10] or a hierarchical extension of REPS [4]. Bounding the KL-divergence between the old and the new policy is almost equivalent to penalizing it, however, it qualifies us to let the temperature of the soft-max distribution be set by the KL-bound instead of hand-tuning the temperature or using heuristics.

REPS divides the policy updates into two steps. It first computes a probability for each observed state action pair by solving an optimization problem with the objective of maximizing the expected rewards while bounding the KL-divergence between the new and the old distributions. This probability corresponds to the desired probability that this sample is used by the new policy. Subsequently, these probabilities are used as weights to estimate a new parametric policy by performing a weighted maximum likelihood update. While our approach is inspired by REPS-based algorithms, there are significant differences: REPS is used to either directly learn in the parameter space of low-level controllers [4,10], which is restricted to controllers with a small number of parameters, such as DMPs [5] or it used to estimate the probability of state action samples [11].

3 Path Integral - Relative Entropy Policy Search (PI-REPS)

PI-REPS considers problems of the PI class but uses an explicit representation of a time-dependent stochastic policy $\pi_t(\mathbf{u}_t|\mathbf{x}_t), \forall t < T$, that maps a state $\mathbf{x}_t$ into a probability distribution over control actions $\mathbf{u}_t$. The objective of PI-REPS is to find the optimal policy π^* that generates the optimal distribution of state trajectories given by the PI theory, Eq. (5), under some additional constraints. For that, it alternates two steps. In the first step, the target distribution (5) is approximated from samples generated by the current policy. At the same time, the information loss to the old policy is bounded to avoid overly greedy policy updates [11,4]. The result of this optimization problem is specified by a weight for each of the seen sample trajectories.

This weight is used in the second step, where the current policy $\tilde{\pi}_t(\mathbf{u}_t|\mathbf{x}_t)$ is updated in a way that can reproduce the desired weighted trajectory distribution. This policy update is computed in a (weighted) maximum likelihood sense. These two steps are iterated until convergence. We describe the details of PI-REPS in the following subsections.

3.1 Learning the Optimal Trajectory Distribution

In first step of PI-REPS, the current control policy is used to generate data in the form of sample trajectories $\mathcal{D} = \{\mathbf{x}_{0:T}^{[i]}\}_{i=1\ldots N}$. Based on these data, we obtain a new trajectory distribution that minimizes the expected KL-divergence to the optimal distribution, i.e.,

$$\text{argmin}_p \int \mu(\mathbf{x}_0)\text{KL}\left(p(\boldsymbol{\tau}|\mathbf{x}_0) \;\|\; p^*(\boldsymbol{\tau}|\mathbf{x}_0)\right) d\mathbf{x}_0. \tag{6}$$

As we want to learn a trajectory distribution, we can not directly use the average reward as optimization criterion as this is done in REPS. REPS would choose a trajectory distribution that might be infeasible, while for PI-REPS we know that the target distribution $p^*(\boldsymbol{\tau}|\mathbf{x}_0)$ is optimal, and, hence, feasible.

In addition to this objective, we also want to stay close to the old trajectory distribution $q(\boldsymbol{\tau}|\mathbf{x}_0)$, i.e., we bound

$$\int \mu(\mathbf{x}_0)\text{KL}\left(p(\boldsymbol{\tau}|\mathbf{x}_0) \;\|\; q(\boldsymbol{\tau}|\mathbf{x}_0)\right) d\mathbf{x}_0 \leq \epsilon. \tag{7}$$

As in REPS, the parameter ϵ can be used as trade-off between exploration and exploitation. As additional constraint, we require that $p(\boldsymbol{\tau}|\mathbf{x}_0)$ defines a proper probability distribution, i.e.,

$$\forall \mathbf{x}_0 : \int p(\boldsymbol{\tau}|\mathbf{x}_0) d\boldsymbol{\tau} = 1.$$

However, this optimization problem requires that we obtain many trajectory samples for each initial state, which is infeasible in many situations. We want to be able to deal with situations where only one trajectory per initial state $\mathbf{x}_0$ can be obtained. For this reason, we extend our objective to optimize also over the initial state distribution, i.e., we optimize over the joint distribution $p(\boldsymbol{\tau}, \mathbf{x}_0)$. The resulting objective is given by

$$\begin{aligned}
&\text{argmin}_p \int p(\boldsymbol{\tau}, \mathbf{x}_0) \log \frac{p(\boldsymbol{\tau}, \mathbf{x}_0)}{p^*(\boldsymbol{\tau}, \mathbf{x}_0)} d\boldsymbol{\tau} d\mathbf{x}_0 \\
=&\text{argmax}_p \int p(\boldsymbol{\tau}, \mathbf{x}_0) \left(\frac{1}{\lambda} S(\boldsymbol{\tau}|\mathbf{x}_0) + \log \mu(\mathbf{x}_0) - \log p(\boldsymbol{\tau}, \mathbf{x}_0)\right) d\boldsymbol{\tau} d\mathbf{x}_0,
\end{aligned} \tag{8}$$

where $p^*(\boldsymbol{\tau}, \mathbf{x}_0) = p^*(\boldsymbol{\tau}|\mathbf{x}_0)\mu(\mathbf{x}_0)$. However, the initial state distribution $\mu(\mathbf{x}_0)$ can not be freely chosen as it is given by the task. Hence, we need to ensure that the marginal distribution $p(\mathbf{x}_0) = \int p(\boldsymbol{\tau}, \mathbf{x}_0) d\boldsymbol{\tau}$ matches the given state distribution $\mu(\mathbf{x}_0)$ for all states $\mathbf{x}_0$. Note that by introducing these constraints we would end up in the original optimization problem (6), but with an infinite number of constraints. However, a common approach to relax this condition is to only match state-feature averages of the marginals [10,4], i.e.

$$\int p(\mathbf{x}_0)\phi(\mathbf{x}_0) d\mathbf{x}_0 = \int \mu(\mathbf{x}_0)\phi(\mathbf{x}_0) d\mathbf{x}_0 = \hat{\phi}_0,$$

where $\hat{\phi}_0$ is the mean feature vector of the samples corresponding to the initial state. The feature vector $\phi(\cdot)$ can be, for example, all linear and quadratic terms of the initial state. In this case, we would match mean and covariance of both distributions. The complete optimization problem reads[6]

$$\begin{aligned} &\text{argmax}_p \int p(\boldsymbol{\tau}, \mathbf{x}_0)\Big(\frac{S(\boldsymbol{\tau}|\mathbf{x}_0)}{\lambda} - \log p(\boldsymbol{\tau}, \mathbf{x}_0)\Big) d\boldsymbol{\tau} d\mathbf{x}_0, \\ &\text{s.t.:} \int p(\mathbf{x}_0)\phi(\mathbf{x}_0) d\mathbf{x}_0 = \hat{\phi}_0, \\ &\qquad \int p(\boldsymbol{\tau}, \mathbf{x}_0) \log \frac{p(\boldsymbol{\tau}, \mathbf{x}_0)}{q(\boldsymbol{\tau}, \mathbf{x}_0)} d\boldsymbol{\tau} d\mathbf{x}_0 \le \epsilon, \\ &\qquad \int p(\boldsymbol{\tau}, \mathbf{x}_0) d\boldsymbol{\tau} d\mathbf{x}_0 = 1. \end{aligned} \tag{9}$$

We solve the above optimization problem using the method of Lagrange multipliers. The solution for $p(\boldsymbol{\tau}, \mathbf{x}_0)$ can be obtained in closed form (see supplement for the derivation)

$$p(\boldsymbol{\tau}, \mathbf{x}_0) \propto q(\boldsymbol{\tau}, \mathbf{x}_0)^{\frac{\eta}{\eta+1}} \exp\left(\frac{S(\boldsymbol{\tau}|\mathbf{x}_0) - \phi(\mathbf{x}_0)^{\intercal}\boldsymbol{\theta}}{\eta + 1}\right) \tag{10}$$

where η and $\boldsymbol{\theta}$ are the Lagrange multipliers corresponding to the KL-divergence and the feature constraints respectively. Their optimal values can be found by optimizing the corresponding dual function $g(\boldsymbol{\theta}, \eta)$

$$[\boldsymbol{\theta}^*, \eta^*] = \text{argmin}_{\boldsymbol{\theta},\eta} g(\boldsymbol{\theta}, \eta), \text{ s.t: } \eta > 0, \tag{11}$$

which is also given in the supplement. Note that the solution $p(\boldsymbol{\tau}, \mathbf{x}_0)$ represents a geometric average between the old distribution and the optimal distribution. The parameter η, which specifies how much we want to interpolate, is chosen by the optimization.

3.2 Weighted Maximum Likelihood Policy Updates

From estimates of the probability $p(\boldsymbol{\tau}, \mathbf{x}_0)$ of the sample trajectories, we can fit a parametrized policy $\tilde{\pi}_t(\mathbf{u}_t|\mathbf{x}_t; \boldsymbol{\omega}_t)$ for each time-step $t < T$ that can reproduce the trajectory distribution $p(\boldsymbol{\tau}, \mathbf{x}_0)$. For each time-step, we want to find the policy $\tilde{\pi}_t$ such that

[6] Note that the $\log \mu(\mathbf{x}_0)$ term can be neglected. Due to the initial state constraints, the path-cost component which is only dependent on the initial state has no influence.

the resulting transition probabilities $p^{\hat{\pi}}(\mathbf{x}_{t+1}|\mathbf{x}_t) = \int P(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)\hat{\pi}_t(\mathbf{u}_t|\mathbf{x}_t; \boldsymbol{\omega}_t)d\mathbf{u}_t$ match the estimated transition distribution from $p(\boldsymbol{\tau}, \mathbf{x}_0)$, where $P(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ corresponds to the model dynamics, assumed to be known. This is an inference problem with latent variables $\mathbf{u}_t$. To solve it, we first compute, for each transition, the action $\mathbf{u}_t^*$ that is most likely to have generated the transition. This controls $\mathbf{u}_t^*$ can be computed from the given control affine system dynamics with $\mathbf{u}_t^* = (\mathbf{G}_t^T\mathbf{G}_t)^{-1}\mathbf{G}_t^T(d\mathbf{x}_t - \mathbf{f}(\mathbf{x}_t)dt)/dt$[7]. Subsequently we extract a parametric policy out of the trajectory distribution $p(\boldsymbol{\tau}, \mathbf{x}_0)$ computed from PI-REPS by minimizing

$$\begin{aligned}
\boldsymbol{\omega}_t^* &= \operatorname{argmin}_{\boldsymbol{\omega}_t} \mathrm{KL}\left(p(\boldsymbol{\tau}, \mathbf{x}_0) \,\|\, p^{\hat{\pi}}(\boldsymbol{\tau}, \mathbf{x}_0)\right) \\
&= \operatorname{argmin}_{\boldsymbol{\omega}_t} \int p(\boldsymbol{\tau}, \mathbf{x}_0) \log\left(\frac{p(\mathbf{x}_{t+1}|\mathbf{x}_t)}{p^{\hat{\pi}}(\mathbf{x}_{t+1}|\mathbf{x}_t)}\right) d\boldsymbol{\tau} d\mathbf{x}_0 + \text{const} \\
&\approx \operatorname{argmax}_{\boldsymbol{\omega}_t} \int p(\boldsymbol{\tau}, \mathbf{x}_0) \log \hat{\pi}_t(\mathbf{u}_t^*|\mathbf{x}_t; \boldsymbol{\omega}_t) d\boldsymbol{\tau} d\mathbf{x}_0 + \text{const} \\
&= \operatorname{argmax}_{\boldsymbol{\omega}_t} \sum_i \frac{p(\boldsymbol{\tau}^{[i]}, \mathbf{x}_0^{[i]})}{q(\boldsymbol{\tau}^{[i]}, \mathbf{x}_0^{[i]})} \log \hat{\pi}_t(\mathbf{u}_t^{*[i]}|\mathbf{x}_t^{[i]}; \boldsymbol{\omega}_t).
\end{aligned}$$

The division by $q(\boldsymbol{\tau}^{[i]}, \mathbf{x}_0^{[i]})$ in the fourth row of the equation results from using samples from $q(\boldsymbol{\tau}^{[i]}, \mathbf{x}_0^{[i]})$ to approximate the integral. This minimization problem can be seen as a weighted maximum likelihood problem with weights $d_i, i = 1 \dots N$ given by

$$d_i = q(\boldsymbol{\tau}^{[i]}, \mathbf{x}_0^{[i]})^{\frac{-1}{\eta+1}} \exp\left(\frac{S(\boldsymbol{\tau}^{[i]}|\mathbf{x}_0^{[i]}) - \boldsymbol{\theta}^\intercal \boldsymbol{\phi}_{\mathbf{x}_0}^{[i]}}{\eta + 1}\right).$$

In the presented approach we use time-dependent Gaussian policies that are linear in the states, i.e. $\hat{\pi}_t(\mathbf{u}_t|\mathbf{x}_t) \sim \mathcal{N}(\mathbf{u}_t|\mathbf{k}_t + \mathbf{K}_t\mathbf{x}_t, \boldsymbol{\Sigma}_t)$. The resulting update equations for $\mathbf{k}_t$, $\mathbf{K}_t$ and $\boldsymbol{\Sigma}_t$ are given by a weighted linear regression and the weighted sample-covariance matrix, respectively [10]. The estimate of $\boldsymbol{\Sigma}_t$ will also contain the variance of the control noise. As this noise is automatically added by the system dynamics, we do not need to add this noise as exploration noise of the policy. Hence, we subtract the control noise from the estimated variance of the policy while ensuring that $\boldsymbol{\Sigma}_t$ stays positive (semi-)definite.

3.3 Step-Based versus Episode-Based Weight Computation

So far, we computed a single weight per trajectory and used this weight to update the policy for all time-steps. However, we can use the simple observation that, if a trajectory distribution is optimal for the time-steps $t = 1 \dots T$, it also has to be optimal for all time segments that also end in T but start at $t' > 1$. Hence we can perform the optimization for each time-step separately and use the obtained weights to fit the policy for the corresponding time step. We path value to come $S_t(\boldsymbol{\tau}'|\mathbf{x}_t) = \sum_{h=t}^{T} r_h(\mathbf{x}_h) + \lambda \log p_{\text{uc}}(\boldsymbol{\tau}'|\mathbf{x}_t)$, where $\boldsymbol{\tau}' = \boldsymbol{\tau}_{t+dt:T}$ in the exponent of the optimal trajectory distribution. For the initial feature constraints, we use the observed average

[7] If the controls $\mathbf{u}_t$ and the noise ϵ_t can be observed, $\mathbf{u}_t^*$ can be computed by $\mathbf{u}_t^* = \mathbf{u}_t + \epsilon_t$.

state features from the old policy at this time step. Such an approach has the advantage that it can considerably reduce the variance of the weights computed for later time steps, and, hence, render PI-REPS more sample efficient. However, as the optimization has now to be done for each time step, the step-based variant is also computationally more demanding.

3.4 Relation to existing approaches

To point out the contributions of this paper, we summarize the novel aspects of PI-REPS with respect to the previously mentioned approaches. In comparison to the REPS algorithm, we use our algorithm to generate a weighting of trajectories, not state-action pairs. As we learn trajectory distributions, we can not freely choose the desired trajectory distribution as certain distributions might not be feasible. In PI-REPS we circumvent this problem by minimizing the Kullback-Leibler divergence to the optimal trajectory distribution instead of maximizing the reward. Due to the optimization over trajectory distributions, the weighted maximum likelihood update is different as we need to obtain $\mathbf{u}_t^*$ from the system dynamics instead of using the executed action $\mathbf{u}_t$.

The constrained optimization problem also leads to a very different solution: while PI-REPS interpolates between the old (initial) and the optimal trajectory distribution (eq. 13), REPS is always affected by the influence of the initial distribution. PI-REPS also considers the initial state in the optimization. Although a similar constraint has been used in REPS for contextual policy search [10], our use is novel since it allows a time step version of the algorithm that, as we show in section 3.3, improves significantly the sample efficiency.

4 Experiments

We evaluated PI-REPS on two simulated benchmark tasks, a double-link swing-up and a quad-link swing-up task. We compared it against variants of previous approaches such as iterative PI control (open loop) [14] and a closed loop extension by fitting a policy with weighted maximum likelihood as performed by our approach. Moreover, we compare the episode-based and the step-based version of PI-REPS and also present the first experiments for model-based reinforcement learning, where in addition to learning a controller, we also learn the forward model of the robot. Finally, we evaluated the influence of the control noise, the KL-bound ϵ as well as the influence of the initial policy and the number of samples used for the policy update. Our experiments show that PI-REPS is a promising approach for stochastic optimal control and model-based reinforcement learning that can find high-quality policies for tasks that are beyond the reach of current methods.

4.1 Double Link Swing-Up

In this task, we used a two link pendulum that needs to swing-up from the bottom position and balance at the top position. Each link had a length of 1m and a weight of 1kg. The torque of each motor was limited to $|u_i| < 10\text{Nm}$. We used a viscous friction model to damp the joint velocities. One episode was composed of 70 time steps with $dt = 66\text{ms}$. The state rewards were $r_t(\mathbf{q}, \dot{\mathbf{q}}) = -10^4\mathbf{q}^T\mathbf{q}$, which punishes the squared distance of the joint angles to the upright position. The reward was given for the last 20 time steps only. The default standard deviation of the control noise was set to $\boldsymbol{\Sigma}_u = 0.5/dt\mathbf{I}$. We used the double link swing-up task for exhaustive parameter evaluation and comparison for being a challenging task but still feasible for running a large number of experiments.

Comparison of different path integral algorithms. We compared our method to different versions of current PI algorithms. We used the step-based variants of all algorithms in this comparison. The episode variants basically show the same results with a slower convergence rate. In the first approach, we applied the iterative path integral method with open loop control to our problem with control noise, as described in section 2.1. Here we simply used a constant action for each time step. In order to estimate this action from samples we used the weighting $d_i = \exp(S(\boldsymbol{\tau}^{[i]}|\mathbf{x}_0^{[i]})/\lambda)$ for each sample. As in the original PI^2 approach [18], we scaled the λ parameter by the range of the path integral values $S(\boldsymbol{\tau}^{[i]}|\mathbf{x}_0^{[i]})$, i.e. $\lambda = \lambda_{\text{PI}^2}/(\max_i S(\boldsymbol{\tau}^{[i]}|\mathbf{x}_0^{[i]}) - \min_i S(\boldsymbol{\tau}^{[i]}|\mathbf{x}_0^{[i]}))$. The value for λ_{PI^2} was empirically optimized. Subsequently, we extended this approach to use the time-dependent linear feedback policy and maximum likelihood updates for the policy as introduced by our approach. Note that this approach is also equivalent to the state of the art policy search approach PoWER [9]. However, in contrast to our method, PoWER as well as PI^2 do not use a principled approach to set the temperature of the soft-max distribution. Moreover, they do not account for the state-dependent part of the path integral as it is done by the use of our baseline. We also evaluated the effect of the baseline by using PI-REPS without state features. In each iteration of the algorithm, we sampled 800 new trajectories. Although this number of trajectories is too large for a real robot application, PI-REPS is model-based and therefore we can first learn a model of the robot using the real robot interactions and, subsequently, use the model to generate an arbitrary number of trajectories. The results of such a model-based reinforcement learning approach are presented at the end of this subsection.

Fig. 1(a) shows a global comparison of the methods. As expected, it can be seen that the open-loop control policy, as used in our version of PI^2, can not deal with the stochastic setup. If we extend PI^2 to learn a linear feedback controller, we can learn successfully the task, but convergence is very slow. As a next step, we introduce the information theoretic policy update to obtain a more principled treatment of the temperature parameter, but we still disable the features used for the baseline in our approach. This method is denoted as *No Features* in Fig. 1(a). While the convergence rate is now significantly improved, ignoring the initial state-distribution constraint results in a bias of the resulting solution and the resulting policy can not reach the quality of the proposed approach with a state-dependent base line. We also compared our approach to state of the art optimal control methods that are based on linearization of the system dynamics, such as the AICO approach [22], but we were not able to find good policies due to the high non-linearities in the task. Clearly, we can see that PI-REPS with a state-dependent base line produces policies of the highest quality. An illustration of the learned movement can be seen in Fig. 2.

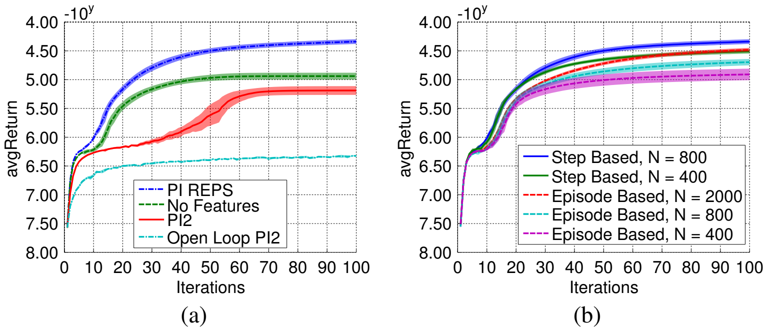


Fig. 1. (a) Comparison of iterative PI with open loop control with our extension of learning closed-loop controller by weighted maximum likelihood, PI-REPS without the feature constraints and the step-based PI-REPS algorithm. PI-REPS outperforms all other methods. (b) Comparison of the step-based variant of PI-REPS with the episode-based variant.

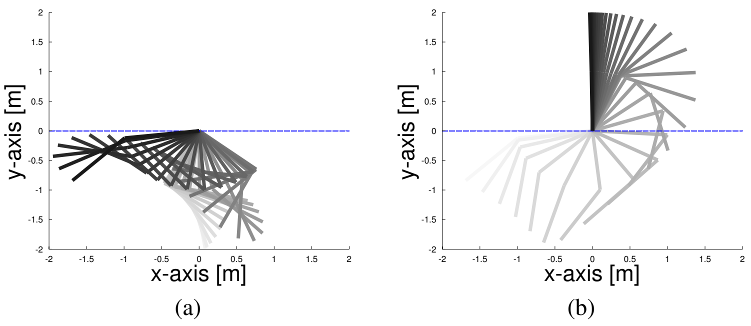


Fig. 2. Illustration of the estimated swing-up movement with the double link. (a) time steps 1 to 25. (b) time steps 36 to 70. Lighter colors indicate an earlier time step.

Step-based versus Episode-based Weighting Computation. We now compare the step-based and the episode-based versions of PI-REPS. From Fig. 1(b), we observe that the step-based version is clearly more sample-efficient, as it reduces the variance of the estimates of the weights for later time steps. However, it is also computationally more demanding, since we need to compute the weights for every time step. The episode-based version with 2000 samples reaches the performance of the step-based version with 400 samples. Hence, if generating samples from the model is cheap, the episode-based version can also be used efficiently.

Exploration in PI-REPS. Exploration is determined by two parameters, the exploration rate Σ_0 of the initial policy and the KL-bound ϵ. For large values of ϵ, PI-REPS converges quickly to the target distribution and stops exploring too soon. In contrast, too small values of ϵ result in too conservative policy updates. This behavior can be clearly

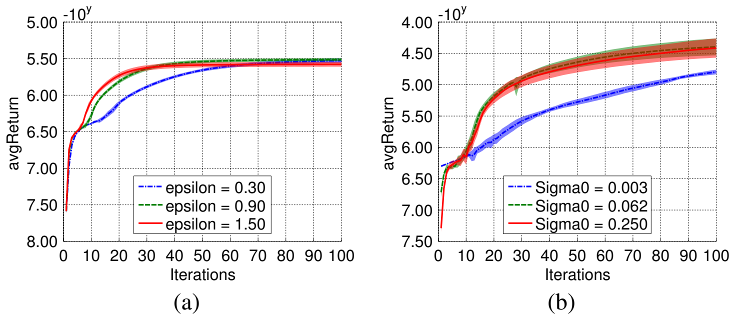


Fig. 3. Exploration in PI-REPS. (a) The value of ϵ determines the convergence and the quality of the obtained policy. For large ϵ, changes in the policy are large, resulting in faster convergence, but too little exploration. For too small ϵ, convergence is slow. (b) Evaluation of the initial exploration rate. If we just use the variance of the uncontrolled process for exploration from the beginning, we get very slow convergence. However, PI-REPS allows for using different sampling policies which are updated by policy search.

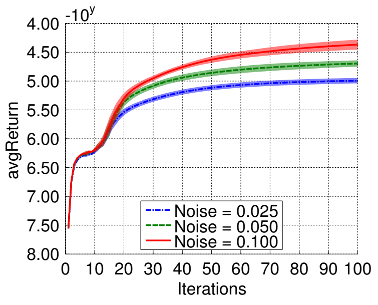
seen in Fig. 3(a). We identified an optimal value of ϵ to be 0.9 this value was used in all other experiments.

A second factor that determines exploration is $\varSigma_0$. If we would fully rely on the noise of the uncontrolled process for exploration, the policy search procedure would take a long time. Therefore, we start with a highly stochastic policy and slowly move to the target distribution by the information theoretic policy updates. From Fig. 3(b), we can clearly see that only using the noise of the system is very inefficient, but higher values of the initial variance lead to a compelling performance.

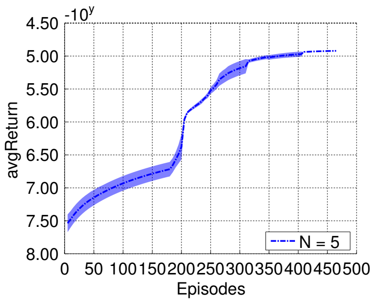
Influence of the Control Noise. In this experiment we evaluated our approach with respect to the control noise $\mathbf{\Sigma}_u$ of the system. Note that, by changing the control noise, we also inherently change the reward function in the path integral framework. Fig. 4 (a) shows the performance for different control noise values. As we can see, good policies can be found for all noise levels, while the costs are decreased with higher noise levels due to the smaller control costs.

Model-Based Reinforcement Learning. While the focus on this paper is to derive an efficient stochastic optimal control method that is based on path integral, we can also directly apply our method to model-based reinforcement learning if we combine the PI-REPS policy updates with a probabilistic model learning technique. In this case, the trajectories generated by the real robot are only used to update the model. From the model, a large number of virtual samples are generated to perform the policy updates. As a proof of concept, we used a simple time-dependent linear model with Gaussian noise, i.e., $P_t(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(\mathbf{x}_{t+1}|\mathbf{A}_t\mathbf{x}_t + \mathbf{B}_t\mathbf{u}_t + \mathbf{a}_t, \mathbf{\Sigma}_t)$.

We estimated such a model for each time step by performing maximum likelihood on the transition samples at each time step. As the model is time-varying, it can also capture non-linear dynamics. We started the algorithm with 25 initial trajectories and



(a)



(b)

Fig. 4. (a) Evaluation for different values of the control noise. PI-REPS could learn high-quality policies even in the existence of a large amount of noise. The difference in the obtained reward is because the reward depends on the noise variance. (b) Experiment with model-based reinforcement learning. We learned time-varying linear models at each time step. A good swing-up policy could be learned already after 300 episodes.

subsequently collected 5 trajectories in each iteration. From the learned models, we generated 1000 trajectories for the policy updates. Fig. 4(b) shows these results. We observe that a high-quality policy can be learned after 300 episodes, which is remarkable if compared to state of the art policy search approaches [4,18]. Yet, the performance of the final policy is affected by the simplicity of the learned model in comparison to the policy found on the real model of the robot.

4.2 Quad-Link Swing-Up

To conclude this experiment section, we used a quad-link pendulum swing-up task. We used the same physical properties, i.e., link length of 1m and a mass of 1kg, the same reward function as well as the same number of time steps as in the double link experiment. Given the increased complexity and increased weight of the whole robot, we increased the maximum torques to 20Nm. We evaluated the episode-based version of our algorithm with a different number of samples.

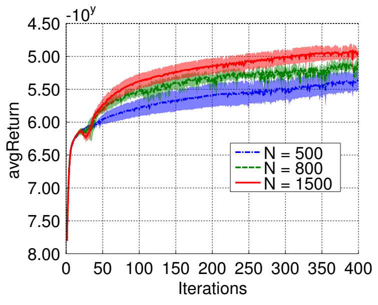


Fig. 5. Learning curve for the quad-link. An increasing number of samples always increases the performance.

The results can be seen in Figure 5. We can see that, due to the increased dimensionality of the problem, more samples are needed to solve the task. However, in contrast to competing methods, PI-REPS is still able to learn high quality policies for this complex task. An illustration of the swing-up movement can be seen in Fig. 6.

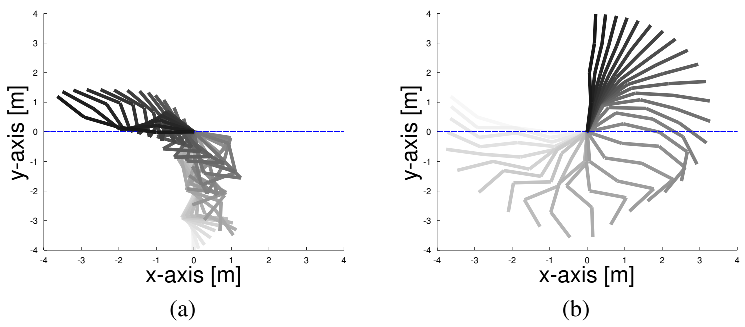


Fig. 6. Illustration of the estimated swing-up movement with the quad link. (a) time steps 1 to 35. (b) time steps 36 to 70. Lighter colors indicate an earlier time step.

5 Conclusions

In this paper we presented PI-REPS, the first approach for PI control that can be used to learn state-feedback policies in an efficient manner. PI-REPS has several benefits to previous PI methods. It allows for a principled treatment of the adaptation of the sampling policy by the information theoretic policy updates. This type of update specifies the temperature of the soft-max distribution. In previous approaches, this temperature had to be chosen heuristically, resulting, as our experiments show, in a poor quality of the estimated policy.

The PI-REPS policy update is based on a weighted maximum likelihood estimate. This is a general approach, not limited to the time varying linear policies that we considered in this paper. Using more complex models such as mixture models, Gaussian processes or neural networks seems to be a promising research direction. We will also investigate the use of more sophisticated model-learning techniques to improve the sample-efficiency in terms of real robot interactions.

Acknowledgements This work was supported by the European Community Seventh Framework Programme (FP7/2007-2013) under grant agreement 270327 (CompLACS).

References

1. M. G. Azar, V. Gómez, and H. J. Kappen. Dynamic policy programming. *Journal of Machine Learning Research*, 13(Nov):3207–3245, 2012.
2. B. Berret, I. Yung, and F. Nori. Open-loop stochastic optimal control of a passive noise-rejection variable stiffness actuator: Application to unstable tasks. In *Intelligent Robots and Systems*, pages 3029–3034. IEEE, 2013.
3. J. Buchli, F. Stulp, E. Theodorou, and S. Schaal. Learning variable impedance control. *The International Journal of Robotics Research*, 30(7):820–833, 2011.
4. C. Daniel, G. Neumann, and J. Peters. Hierarchical Relative Entropy Policy Search. In *International Conference on Artificial Intelligence and Statistics*, 2012.

5. A. Ijspeert and S. Schaal. Learning Attractor Landscapes for Learning Motor Primitives. In *Advances in Neural Information Processing Systems 15*. MIT Press, Cambridge, MA, 2003.
6. A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in neural information processing systems*, pages 1523–1530, 2002.
7. H. J. Kappen. Linear theory for control of nonlinear stochastic systems. *Physical Review Letters*, 95(20):200201, 2005.
8. H. J. Kappen, V. Gómez, and M. Opper. Optimal control as a graphical model inference problem. *Machine Learning*, 87:159–182, 2012.
9. J. Kober and J. Peters. Policy search for motor primitives in robotics. *Mach. Learn.*, 84(1-2):171–203, July 2011.
10. A. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann. Data-Efficient Contextual Policy Search for Robot Movement Skills. In *Proceedings of the National Conference on Artificial Intelligence*, 2013.
11. J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1607–1612, 2010.
12. K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *International Conference on Robotics Science and Systems*, 2012.
13. K. Rawlik, M. Toussaint, and S. Vijayakumar. Path integral control by reproducing kernel Hilbert space embedding. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 1628–1634. AAAI Press, 2013.
14. E. Rombokas, E. Theodorou, M. Malhotra, E. Todorov, and Y. Matsuoka. Tendon-driven control of biomechanical and robotic systems: A path integral reinforcement learning approach. In *International Conference on Robotics and Automation*, pages 208–214, 2012.
15. F. Stulp and S. Schaal. Hierarchical reinforcement learning with movement primitives. In *Humanoid Robots, 11th IEEE-RAS International Conference on*, pages 231–238, 2011.
16. F. Stulp and O. Sigaud. Path Integral Policy Improvement with Covariance Matrix Adaptation. In *International Conference Machine Learning*, 2012.
17. F. Stulp, E. Theodorou, J. Buchli, and S. Schaal. Learning to grasp under uncertainty. In *International Conference on Robotics and Automation*, pages 5703–5708. IEEE, 2011.
18. E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11:3137–3181, 2010.
19. E. Theodorou and E. Todorov. Relative entropy and free energy dualities: connections to path integral and KL control. In *IEEE 51st Annual Conference on Decision and Control*, pages 1466–1473, 2012.
20. E. Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems 19*, pages 1369–1376. MIT Press, Cambridge, MA, 2006.
21. E. Todorov. Policy gradients in linearly-solvable MDPs. In *Advances in Neural Information Processing Systems*, pages 2298–2306, 2010.
22. M. Toussaint. Robot Trajectory Optimization using Approximate Inference. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

Appendix: Dual function for PI-REPS

We derive the dual function. For notation simplicity, we use $p_{\boldsymbol{\tau}\mathbf{x}_0}$ for $p(\boldsymbol{\tau}, \mathbf{x}_0)$, $\phi_{\mathbf{x}_0}$ for $\phi(\mathbf{x}_0)$, $q_{\boldsymbol{\tau}\mathbf{x}_0}$ for $q(\boldsymbol{\tau}, \mathbf{x}_0)$ and $S_{\boldsymbol{\tau}}$ for $S(\boldsymbol{\tau}, \mathbf{x}_0)$. The Lagrangian is

$$\begin{aligned}\mathcal{L} &= \int_{\boldsymbol{\tau},\mathbf{x}_0} p_{\boldsymbol{\tau}\mathbf{x}_0}\left(S_{\boldsymbol{\tau}} - \log p_{\boldsymbol{\tau}\mathbf{x}_0} - \lambda - \phi_{\mathbf{x}_0}^{\intercal}\boldsymbol{\theta} - \eta \log \frac{p_{\boldsymbol{\tau}\mathbf{x}_0}}{q_{\boldsymbol{\tau}\mathbf{x}_0}}\right) d\boldsymbol{\tau} d\mathbf{x}_0 + \lambda + \hat{\phi}_0^{\intercal}\boldsymbol{\theta} + \eta\epsilon \\ &= \int_{\boldsymbol{\tau},\mathbf{x}_0} p_{\boldsymbol{\tau}\mathbf{x}_0}\left(S_{\boldsymbol{\tau}} - (\eta+1)\log p_{\boldsymbol{\tau}\mathbf{x}_0} + \eta \log q_{\boldsymbol{\tau}\mathbf{x}_0} - \lambda - \phi_{\mathbf{x}_0}^{\intercal}\boldsymbol{\theta}\right) d\boldsymbol{\tau} d\mathbf{x}_0 \\ &\quad + \lambda + \hat{\phi}_0^{\intercal}\boldsymbol{\theta} + \eta\epsilon\ , \end{aligned} \tag{12}$$

where $\boldsymbol{\theta}, \eta$ and λ appear due to the constraints of the features, the KL-bound and the normalization, respectively. Taking derivative and solving for $p_{\boldsymbol{\tau}\mathbf{x}_0}$ gives

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial p_{\boldsymbol{\tau}\mathbf{x}_0}} &= S_{\boldsymbol{\tau}} - (\eta+1)(\log p_{\boldsymbol{\tau}\mathbf{x}_0} + 1) - \lambda - \phi_{\mathbf{x}_0}^{\intercal}\boldsymbol{\theta} + \eta \log q_{\boldsymbol{\tau}\mathbf{x}_0} = 0 \\ \log p_{\boldsymbol{\tau}\mathbf{x}_0} &= \frac{\eta \log q_{\boldsymbol{\tau}\mathbf{x}_0}}{\eta+1} + \frac{S_{\boldsymbol{\tau}} - \phi_{\mathbf{x}_0}^{\intercal}\boldsymbol{\theta}}{\eta+1} + \frac{-\lambda - (\eta+1)}{\eta+1} \\ p_{\boldsymbol{\tau}\mathbf{x}_0} &= Z^{-1} q_{\boldsymbol{\tau}\mathbf{x}_0}{}^{\frac{\eta}{\eta+1}} \exp\left(\frac{S_{\boldsymbol{\tau}} - \phi_{\mathbf{x}_0}^{\intercal}\boldsymbol{\theta}}{\eta+1}\right), \qquad Z = \exp\left(\frac{\lambda + (\eta+1)}{\eta+1}\right). \end{aligned} \tag{13}$$

From the normalization constraint

$$\exp\left(\frac{\lambda + (\eta+1)}{\eta+1}\right) = \int_{\boldsymbol{\tau},\mathbf{x}_0} q_{\boldsymbol{\tau}\mathbf{x}_0}{}^{\frac{\eta}{\eta+1}} \exp\left(\frac{S_{\boldsymbol{\tau}} - \phi_{\mathbf{x}_0}^{\intercal}\boldsymbol{\theta}}{\eta+1}\right) d\boldsymbol{\tau} d\mathbf{x}_0. \tag{14}$$

Plugging (13) into (12) and simplifying we arrive to the following dual function

$$g(\boldsymbol{\theta}, \eta) = (\eta+1) + \lambda + \hat{\phi}_0^{\intercal}\boldsymbol{\theta} + \eta\epsilon. \tag{15}$$

Reinserting (14) in (15)

$$\begin{aligned} g(\boldsymbol{\theta}, \eta) &= (\eta+1) + \lambda + \hat{\phi}_0^{\intercal}\boldsymbol{\theta} + \eta\epsilon = (\eta+1)\left[\frac{(\eta+1)+\lambda}{\eta+1}\right] + \hat{\phi}_0^{\intercal}\boldsymbol{\theta} + \eta\epsilon \\ &= (\eta+1)\log\left(\int_{\boldsymbol{\tau},\mathbf{x}_0} q_{\boldsymbol{\tau}\mathbf{x}_0}{}^{\frac{\eta}{\eta+1}} \exp\left(\frac{S_{\boldsymbol{\tau}} - \phi_{\mathbf{x}_0}^{\intercal}\boldsymbol{\theta}}{\eta+1}\right) d\boldsymbol{\tau} d\mathbf{x}_0\right) + \hat{\phi}_0^{\intercal}\boldsymbol{\theta} + \eta\epsilon. \end{aligned}$$

Replacing the integral by a sum over sample trajectories generated by q yields

$$g(\boldsymbol{\theta}, \eta) = (\eta+1)\log\left(\frac{1}{N}\sum_i q_{\boldsymbol{\tau}\mathbf{x}_0}^{[i]}{}^{\frac{-1}{\eta+1}} \exp\left(\frac{S_{\boldsymbol{\tau}}^{[i]} - \boldsymbol{\theta}^{\intercal}\boldsymbol{\phi}_{\mathbf{x}_0}^{[i]}}{\eta+1}\right)\right) + \hat{\phi}_0^{\intercal}\boldsymbol{\theta} + \eta\epsilon.$$

The dual function can be evaluated from the state trajectory samples. The distribution $q_{\boldsymbol{\tau}\mathbf{x}_0}^{[i]}$ can be computed using the current policy and the model, i.e. [8],

$$q_{\boldsymbol{\tau}\mathbf{x}_0}^{[i]} = \mu(\mathbf{x}_0) \prod_{t=0}^{T-1} \int_{\mathbf{u}_t} P_t(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)\pi_t(\mathbf{u}_t|\mathbf{x}_t) d\mathbf{u}_t. \tag{16}$$

[8] In the case of Gaussian policies such as we consider here, the integral in (16) can be computed analytically.