Learning time-dependent feeback policies with model-based policy search

Master-Thesis von Rudolf Lioutikov aus Darmstadt

Oktober 2013

TECHNISCHE UNIVERSITÄT DARMSTADT

Computerscience

Intelligent Autonomous Systems

Learning time-dependent feeback policies with model-based policy search

Vorgelegte Master-Thesis von Rudolf Lioutikov aus Darmstadt

1. Gutachten: Prof. Dr. Ing. Jan Peters

2. Gutachten: Dr. Gerhard Neumann

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den October 22, 2013

_____

(R. Lioutikov)

Abstract

Stochastic Optimal Control (SOC) is typically used to plan a movement for a specific situation. However, most SOC methods are not able to generalize the movement plan to a new situation, and, hence, replanning is required. In this paper we present a SOC method that allows us for reusing the controller in a new situation as it is more robust to deviations from the initial movement plan. In order to improve the robustness of the controller, we employ an information-theoretic policy update which explicitly operates on trajectory distributions instead of single trajectories. Our information theoretic policy update limits the 'distance' between the trajectory distributions of the old and the new control policy and ensures a stable and smooth policy update. The introduced bound offers a closed form solution for the resulting policy and extends results from recent developments in SOC. In difference to many standard SOC algorithms, our approach can directly infer the system dynamics from data points, and, hence, can also be used for model-based reinforcement learning. This paper represents an extension of [13]. In addition to revisiting the content we provide an extensive theoretical comparison our approach with related work, discuss additional aspects of the implementation and introduce further evaluations.

1 Introduction

The goal of Stochastic Optimal Control (SOC), see [22, 11, 27, 19], is to find the optimal control for a finite horizon of time steps. A common approach to solve the SOC problem is dynamic programming, i.e. the value function is computed iteratively backwards in time, where the value function estimates the expected future reward for a given state and time step. Unfortunately only few cases offer analytical solutions, e.g. discrete systems and linear system with quadratic cost functions and Gaussian noise (LQG). For more complex systems, many approaches rely on a linearisation of the underlying system [24, 27]. However, while these approaches work well for planning a single movement, the resulting controller is hard to generalize to a new situation, e.g., a movement task starting from a slightly different initial position. In this case, replanning is often required. A second short coming of state of the art SOC methods is that the policy update might by unstable due to the used approximations of either the system dynamics [24, 27], or of the value function by approximate dynamic programming approaches [9, 16]. Due to this unstable policy update, the resulting state distribution of the new policy might jump, leading the agent into areas of the state space where the estimated policy might have poor quality.

In this paper, we tackle both problems by introducing an information theoretic policy update for SOC that directly operates on trajectory distributions instead of single nominal trajectories. We will use an information theoretic constraint in our optimization to ensure that the new policy will stay close to our previously generated data. Such 'closeness to the data' constraint can be formalized by limiting the Kullback-Leibler (KL) $\mathrm{KL}(p||q)$ divergence between the distribution q that has generated the data and the resulting distribution p that is generated by applying the new policy. Such constraint avoids jumps in the resulting state distribution of the new policy. Our method is inspired by the policy search community, where similar constraints have already been used to find good parameters for parametric policies [17, 6, 7].

As for most SOC methods, we focus on estimating locally optimal control policies for single movements. The controller should be robust to deviations from the nominal trajectory, however, we do not aim to estimate a globally optimal controller. For example, for estimating a back-hand swing in tennis, we want to estimate a controller which can be applied to many configurations of the

incoming ball without re-estimating the controller for a slightly different situation. However, we only want to use the same controller for the same type of movement, i.e., it does not need to generalize from backhand to forehand movements. As in the LQG case, our goal is to estimate a time-dependent linear feedback controller. As many SOC methods, our approach is model based. We learn a time dependent linear model representation, which can be easily obtained from samples. Hence, our approach is easily applicable to model-based reinforcement learning, where the system dynamics are not known beforehand.

This paper is an extension of our results presented in [13]. We provide an extensive theoretical comparison our approach with related work, discuss additional aspects of the implementation and introduce further evaluations. In the next section we will discuss the related work, subsequently we will present an alternative formulation of SOC based on constrained optimization that is the basis for information theoretic SOC (ITSOC). In this formulation, we can straightforwardly add the information-theoretic bound to stay close to the data. After introducing the information-theoretic bounds for the constrained optimization, we will state our specific algorithm including the representation of the policy and the value function. Finally, we will discuss the relation to existing approaches and present experiments of our method on a simulated 4-link non-linear planar arm.

2 Preliminaries and Related Work

Current SOC methods either use an approximate system model, e.g., by linearizing the system, or approximate the value function also called approximate dynamic programming (ADP). We will first review these two approaches and and point out important differences to our approach. Subsequently, we will discuss SOC control algorithms that also use Kullback-Leibler divergence terms to determine the policy, i.e., dynamic policy programming [1], SOC by approximate inference [19] and path integral approaches [23, 20]. We will also discuss the relation to approximate dynamic programming with soft-max operators [16] and existing policy search algorithms [7].

2.0.1 Stochastic Optimal Control

We consider systems with continuous state and action spaces where we put our focus on robotics. We will denote the state of the robot as $\boldsymbol{x}$ and the control action as $\boldsymbol{u}$. The robot follows its stochastic system dynamics and transfers to state $\boldsymbol{x}_{t+1}$ from state $\boldsymbol{x}_t$ with probability $p_t(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$ when taking action $\boldsymbol{u}_t$. The goal of the robot is to find the optimal policy $\pi_t^*(\boldsymbol{u}|\boldsymbol{x})$ which maximizes the expected accumulative reward for a given time horizon H

$$\pi^* = \arg\max_{\pi} J = \arg\max_{\pi} \mathbb{E}_{p^{\pi}(\boldsymbol{\tau})} \left[\sum_{t=1}^{H-1} r_t(\boldsymbol{x}_t, \boldsymbol{u}_t) + r_H(\boldsymbol{x}_H) \right], \tag{1}$$

where the expectation is done with respect to the trajectories $\boldsymbol{\tau} = (\boldsymbol{x}_{1:H}, \boldsymbol{u}_{1:H-1})$, $r_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$ denotes the immediate reward signal for time step t and $r_H(\boldsymbol{x}_H)$ the final reward for reaching state $\boldsymbol{x}_H$. Note that the optimal policy is typically not stationary but depends

on the time step t as we deal with a finite time horizon. The actions $\boldsymbol{u}_t$ of the agent are chosen by a policy $\pi_t(\boldsymbol{u}_t|\boldsymbol{x}_t)$ that is in the finite horizon formulation also time-dependent. The trajectory distribution of policy π is given by

$$p^{\pi}(\boldsymbol{\tau}) = p_1(\boldsymbol{x}_1)\prod_{t=1}^{H-1} p_t(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)\pi_t(\boldsymbol{u}_t|\boldsymbol{x}_t),$$

where $p(\boldsymbol{x}_1)$ is the initial state distribution. We will further denote $\mu_t^{\pi}(\boldsymbol{x}_t)$ as the state distribution of policy π. It represents the probability of being in state $\boldsymbol{x}_t$ at time step t when following policy π. Formally, the state distribution $\mu_t^{\pi}(\boldsymbol{x}_t)$ can be defined as

$$\mu_t^{\pi}(\boldsymbol{x}_t) = \int \dots \int p^{\pi}(\boldsymbol{\tau}) d\boldsymbol{x}_1 \dots d\boldsymbol{x}_{t-1} d\boldsymbol{x}_{t+1} \dots d\boldsymbol{x}_H d\boldsymbol{u}_1 \dots d\boldsymbol{u}_{H-1} \tag{2}$$

as marginalization over the state-action space of all remaining time steps. Under the assumption that the system dynamics are known and do not need to be approximated, the optimal policy π^* is deterministic which we will write as $\boldsymbol{u}^* = \pi^*(\boldsymbol{x})$.

2.0.2 Dynamic Programming

Dynamic Programming (DP) is a common approach to solve a SOC problem by iteratively estimating the value function [15, 4, 9, 3]. The value function $V_t^{\pi}(\boldsymbol{x}_t)$ computes the expected future reward when being in state $\boldsymbol{x}_t$ at time step t and following policy π. In its recursive form, the definition of the value function is given as

$$V_t^{\pi}(\boldsymbol{x}) = \int \pi_t(\boldsymbol{u}_t|\boldsymbol{x}_t) \left(r_t(\boldsymbol{x}_t, \boldsymbol{u}_t) + \int p_t(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) V_{t+1}^{\pi}(\boldsymbol{x}_{t+1}) d\boldsymbol{x}_{t+1} \right) d\boldsymbol{u}_t, \tag{3}$$

for $t \leq H$ and $V_H^{\pi}(\boldsymbol{x}) = r_H(\boldsymbol{x})$. The optimal value function $V_t^*(\boldsymbol{x}_t)$ can be obtained by iterating

$$V_t^*(\boldsymbol{x}_t) = \max_{\boldsymbol{u}} Q^*(\boldsymbol{x}_t, \boldsymbol{u}), \tag{4}$$

$$Q^*(\boldsymbol{x}_t, \boldsymbol{u}) = r_t(\boldsymbol{x}_t, \boldsymbol{u}) + \int p_t(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}) V_{t+1}^*(\boldsymbol{x}_{t+1}) d\boldsymbol{x}_{t+1} \tag{5}$$

backwards in time. Similarly, the optimal policy can be obtained by

$$\boldsymbol{u}_t^* = \pi_t^*(\boldsymbol{x}_t) = \operatorname{argmax}_{\boldsymbol{u}} \left(Q^*(\boldsymbol{x}_t, \boldsymbol{u}) \right) \tag{6}$$

Unfortunately, analytic solutions only exists in the case of discrete state and action spaces or for linear systems with a quadratic reward function with Gaussian noise, also known as LQG case. It is well known that the optimal value function is in the LQG case given as quadratic function, i.e., $V_t^*(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{V}_t \boldsymbol{x} + \boldsymbol{x}^T \nu_t + \nu_t$ and the optimal policy is given as time varying linear feedback controller, i.e., $\pi^*(\boldsymbol{x}) = \boldsymbol{s}_t + \boldsymbol{S}_t \boldsymbol{x}$. The parameters of the reward function as well as for the controller can be obtained analytically [22].

For non-LQG systems we have to resort to some type of approximations. Here, we can distinguish between approaches based on approximating the system dynamics and approaches approximating the value function.

Approximating the System Dynamics.

Approaches based on approximating of the system dynamics, such as the iLQG [24] or the AICO [27] algorithm, linearise the system at the current estimate of the trajectory, where for each time step a different linearisation is used. The optimal controller is computed analytically by using the LQG solution and, subsequently, a new trajectory is estimated by simulating this controller. This trajectory is used as a new linearisation point. Due the used LQG solution, the obtained policy is greedy with respect to the linearised system dynamics. However, the linearization of the model might change heavily in each iteration causing jumps in the resulting update of the trajectory. These problems can been alleviated by introducing heuristics such as regularizing the LQG solution [24] or using a learning rate on the estimated trajectory [27]. The advantage of linearisation approaches is their computational efficiency. However, in the case of small deviations from the nominal trajectory, the linearisation of the model becomes less accurate. Hence, the robustness of the resulting policy typically degrades and replanning is required if, for example, we slightly deviate from the planned initial state.

Approximating the Value Function.

Other approximate dynamic programming approaches approximate the value function, such as fitted V-iteration [10] or fitted Q-iteration [9, 16]. These approaches are sample-based and iteratively collect data with the currently estimated policy. The data is subsequently used to improve the estimate of the value function that is approximated by supervised learning methods, such as linear function approximators [4, 19], locally-weighted regression [16] and regression trees [9]. Some of these methods are model-free and originate from the reinforcement learning domain. Value function approximation is typically more general than linearisation approaches, however, these algorithms also come with an increased computational complexity. Value-function approximation suffers from a similar issue as the approaches based on linearization of the system dynamics — the approximation of the value function may ‘damage’ the resulting trajectory distribution of the policy, causing jumps and oscillations between subsequent iterations. Such behavior is in general not desirable as uncontrolled jumps in the trajectory distribution might lead the agent in areas of the state space not seen before, and, hence, our policy might be arbitrarily bad in this area.

Advantage Weighted Regression.

One problem of value function approximation approaches in continuous action spaces is the $\max_{\boldsymbol{u}}$-operator. Advantage Weighted Regression [16] is a method to efficiently compute this max-operator by replacing it with a soft-max operator [16]. The soft-max operator is implemented by using a stochastic policy

$$\pi_t(\boldsymbol{u}_t|\boldsymbol{x}_t) \propto \exp\left(\frac{Q_t^\pi(\boldsymbol{x}_t, \boldsymbol{u}_t) - V_t^\pi(\boldsymbol{x}_t)}{\eta}\right),$$

where $Q_t^\pi(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is the state-action value function of the current policy. These probabilities are only evaluated on a finite set of samples and are subsequently used to determine the new value function as weighting term in a weighted regression algorithm. As the term inside the exponential function denotes the advantage function, the weighted regression is denoted as *advantage weighted regression* (AWR). From the weighted samples, we can also obtain a new parametric policy with a similar advantage weighted regression.

2.0.3 SOC based on Approximate Inference

An alternative view on SOC has been presented in [19] which is based on approximate inference. The main idea is to transform the reward into probabilities by introducing a binary reward event with probability $p(R_t = 1|\boldsymbol{x}_t, \boldsymbol{u}_t) \propto \exp(r_t(\boldsymbol{x}, \boldsymbol{u})/\eta)$. Given a prior policy π_0, the goal is to compute the a policy π that produces a trajectory distribution $p^\pi(\boldsymbol{\tau})$ which is most similar to posterior trajectory distribution $p^{\pi_0}(\boldsymbol{\tau}|R_{1:H} = 1)$ after conditioning on seeing the reward event in every time step. More formally, we need to find a policy π which minimizes the KL-divergence between the posterior $p^{\pi_0}(\boldsymbol{\tau}|r_{1:H} = 1)$ and the new trajectory distribution $p^\pi(\boldsymbol{\tau})$, i.e.,

$$\pi^* = \text{argmin}_\pi \text{KL}(p^\pi(\boldsymbol{\tau})||p^{\pi_0}(\boldsymbol{\tau}|r_{1:H} = 1)).$$

This minimization is performed iteratively. The policy exhibits a similar soft-max structure as the one presented in this paper. However, the exponential transformation of the reward is an assumption we need to put in this approach, and the parameter η_t has to be chosen by the user. As we will see, the most important difference to our approach is that the KL-divergence is directly used on the trajectory distributions $p^\pi(\boldsymbol{\tau})$. Such KL-formulation can be decomposed into the sum of the Kullback Leibler divergences $\text{KL}(\pi_t(\boldsymbol{u}|\boldsymbol{x})||\pi_{0,t}(\boldsymbol{u}|\boldsymbol{x}))$ of the policies for each time step. These KL-terms can be seen as additional cost terms in the immediate reward function which punish deviations of the estimated policy from the prior policy π_0 [19]. Such punishment term is also used in the dynamic policy programming algorithm that is a special case of the approximate inference framework introduced in [19]. Similar ideas of using the KL between the estimated policy and a prior policy have also been introduced within the field of linear solvable MDPs [25, 26] To solve this modified SOC problem, typically value function approximation methods are used. Consequently, SOC based on approximate inference suffers from the same deficits as ADP methods as the approximation of the value function might damage the resulting policy update. In contrast, our KL-divergence acts on the marginals $p_t^\pi(\boldsymbol{x}, \boldsymbol{u})$ that also includes the state

distribution $\pi^{\mu}(\boldsymbol{x})$ at each time step. Consequently, we can directly control the damage of the state distributions $\pi^{\mu}(\boldsymbol{x})$, which results in a more stable policy update.

2.0.4 SOC based on Path Integrals

The path integral (PI) formulation of SOC [23] has recently gained a lot of attention as it allows for computing the optimal policy without the use of dynamic programming. The PI approach exponentially transforms the continuous-time value function and computes the optimal value function and policy using the Feynman-Kac theorem [23]. The optimal value function for time step t is then given by

$$V_t^{\pi}(\boldsymbol{x}) = \eta \log \int p^{\pi_0}(\boldsymbol{\tau}|\boldsymbol{x}_t = \boldsymbol{x}) \exp\left(\frac{\sum_{t=1}^{H} r_t(\boldsymbol{x}_t)}{\eta}\right) d\boldsymbol{\tau}, \tag{7}$$

where $p_t^{\pi_0}(\boldsymbol{\tau}|\boldsymbol{x}_t = \boldsymbol{x})$ is the trajectory distribution of the *uncontrolled* process[1] starting at time step t in state $\boldsymbol{x}$. The parameter η defines the temperature of the exponential transformation of the value function and is chosen heuristically. The original PI approach is based on Monte-Carlo roll-outs and therefore requires a lot of samples, however, recently more efficient approaches based on value function approximation [20] have been introduced that are essentially similar to the SOC by approximate inference approaches. The PI approach suffers from two severe limitations. Firstly, it assumes that the control costs are quadratic and the quadratic control cost matrix coincides with the covariance matrix of the system noise. Moreover, it makes the assumption that all the noise in the system only acts on state variables which can be controlled by the agent. Both assumptions are a quite limiting.

2.0.5 Policy Search Methods

Our algorithm was inspired by the policy search community, where related information-theoretic bounds have already been introduced to learn to select and improve movement primitives for playing robot table-tennis [17], robot tether-ball [6] or robot hockey [7, 12]. The KL-bound has been originally introduced by the Relative Entropy Policy Search (REPS) algorithm [17], which has been extended in [6] to hierarchical policies and in [7] to learning sequential motor tasks. While a similar time-indexed formulation of REPS has been used in [7], the algorithm was designed for learning with movement primitives and can not be directly applied to the SOC formulation. We will show in our experiments that this algorithm quickly degrades in the presence of noise in the system dynamics. There has also been theoretical evidence that the information theoretic policy update has beneficial properties. In the case of an adversarial MDP with known transition dynamics, the information theoretic policy update used in this paper achieves optimal regret bounds [28].

Another policy search approach that is highly related to our approach is the model-based PILCO algorithm [8]. PILCO learns a gaussian process (GP) model [18] of the system dynamics and uses deterministic approximate inference methods for predicting the trajectory distribution of the current policy with the learned models. The predicted trajectory distribution is used to obtain the policy gradient. PILCO performs a greedy policy update with the currently learned forward model, i.e., it searches for a local optimum of the policy parameters. As the currently estimated model is likely to be inaccurate, such greedy update might be dangerous and might again cause jumps in the trajectory distributions. PILCO also neglects the problem of exploring the state space as it relies on a deterministic policy. It is therefore likely to get stuck in a local minimum of the policy parameters. In difference to PILCO,

[1] Or a process using a prior policy π_0

we learn much a simpler model of the system dynamics, where we learn an individual linear model for each time step. While such models are less data efficient as time-independent GP models, they require considerably less computation time and offer a sufficient accuracy. As we will show, PILCO is not suitable for such models due to the problems connected with the greedy policy update.

3 Information-Theoretic Stochastic Optimal Control

We will start our discussion by reformulating the SOC problem as constrained optimization problem. This reformulation has the advantage that we can easily add more constraints such as the information theoretic KL-bounds. We want to maximize the expected reward, which we now write in terms of the state distribution $\mu_t^\pi(\boldsymbol{x})$ as

$$J_{\pi\mu} = \sum_{t=1}^{H-1} \iint \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})r_t(\boldsymbol{x},\boldsymbol{u})d\boldsymbol{x}\,d\boldsymbol{u} + \int \mu_H^\pi(\boldsymbol{x})r_H(\boldsymbol{x})d\boldsymbol{x}. \tag{8}$$

The advantage of such an approach is that we can easily add new constraints which, for example, ensure that we stay close to the generated data. As first constraint, we require that π and μ jointly define a distribution, i.e., $\int \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})d\boldsymbol{x}\,d\boldsymbol{u} = 1, \forall t$. Moreover, we can not freely choose our state distribution $\mu_t^\pi(\boldsymbol{x})$, but $\mu_t^\pi(\boldsymbol{x})$ has to comply with the policy and the system dynamics, i.e.,

$$\mu_t^\pi(\boldsymbol{x}') = \iint \mu_{t-1}(\boldsymbol{x})\pi_{t-1}(\boldsymbol{u}|\boldsymbol{x})p_{t-1}(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u})d\boldsymbol{x}\,d\boldsymbol{u}, \forall t > 1 \wedge \forall \boldsymbol{x}'. \tag{9}$$

Additionally, the state distribution $\mu_1^\pi(\boldsymbol{x})$ for the first time step has to reproduce the given initial state distribution of the problem, i.e., $\mu_1^\pi(\boldsymbol{x}) = p_1(\boldsymbol{x}), \forall \boldsymbol{x}$. The optimization problem defined in Equation (8), under the constraints of Equations (9) and the intial state constraint is equivalent to the original SOC problem. However, the state distribution constraints are infeasible in continuous action spaces as we would have an infinite amount of constraints.

3.1 Approximate Constraints by Expected Feature Matching

Therefore, we resort to matching expected feature averages [17], i.e.,

$$\int \mu_t^\pi(\boldsymbol{x}')\boldsymbol{\phi}(\boldsymbol{x}')d\boldsymbol{x}' = \iiint \mu_{t-1}(\boldsymbol{x})\pi_{t-1}(\boldsymbol{u}|\boldsymbol{x})p_{t-1}(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u})\boldsymbol{\phi}(\boldsymbol{x}')d\boldsymbol{x}\,d\boldsymbol{u}\,d\boldsymbol{x}' \tag{10}$$

for $t > 1$ and

$$\int \mu_1^\pi(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})d\boldsymbol{x} = \int p_1(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})d\boldsymbol{x} = \hat{\boldsymbol{\phi}}_1 \tag{11}$$

for $t = 1$, where $\boldsymbol{\phi}$ is a feature function and $\hat{\boldsymbol{\phi}}_1$ is the mean observed feature vector of the initial state. In this paper, we will use all linear and squared terms of the state $\boldsymbol{x}$ as the feature vector. Hence, we match the mean and the covariance matrices of both distributions. Note that such formulation introduces a new type of approximation for SOC problems. Instead of approximating

the system dynamics or the value function, we approximate the state distribution constraints. As we will see, a value function like term emerges out of this constraints in our policy update equation. However, this value function like term is not obtained by approximating the future return, but its purpose is to guarantee that the distribution constraints are satisfied.

3.2 Staying Close to the Data with Information-Theoretic Bounds

As we already argued, it is crucial for SOC methods to stay close to the data in order to achieve a stable policy update. The data is given as state-action pairs $(\boldsymbol{x}_t, \boldsymbol{u}_t)$ generated from the old state-action distribution $q_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$. As we want our new trajectory distribution $p^\pi(\tau)$ to stay close to this old data distribution, we bound the Kullback-Leibler divergence between the marginal distributions of the single time steps, i.e., $p_t^\pi(\boldsymbol{x}_t, \boldsymbol{u}_t) = \mu_t^\pi(\boldsymbol{x}_t)\pi_t(\boldsymbol{u}_t|\boldsymbol{x}_t)$ and $q_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$. For $t < H$ the bound is given by

$$\epsilon \geq \iint \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x}) \log \frac{\mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})}{q_t(\boldsymbol{x}, \boldsymbol{u})} d\boldsymbol{x}\, d\boldsymbol{u}, \tag{12}$$

and for $t = H$ by

$$\epsilon \geq \int \mu_H^\pi(\boldsymbol{x}) \log \frac{\mu_H^\pi(\boldsymbol{x})}{q_H(\boldsymbol{x})} d\boldsymbol{x}. \tag{13}$$

Thus, we have all ingredients to state the full optimization problem which defines the information theoretic SOC algorithm

$$\begin{aligned}
\text{argmax}_{\pi,\mu^\pi} \quad & \sum_{t=1}^{H-1} \iint \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x}) r_t(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{x}\, d\boldsymbol{u} + \int \mu_H^\pi(\boldsymbol{x}) r_H(\boldsymbol{x}) d\boldsymbol{x}, \\
\text{s.t:} \quad & \iint \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x}) d\boldsymbol{x}\, d\boldsymbol{u} = 1 \text{ for } t < H \text{ and } \int \mu_H^\pi(\boldsymbol{x}) d\boldsymbol{x} = 1, \\
1 < t < H: \quad & \int \mu_{t+1}^\pi(\boldsymbol{x}')\boldsymbol{\phi}(\boldsymbol{x}') d\boldsymbol{x}' = \iiint \mu_t(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x}) p(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{u})\boldsymbol{\phi}(\boldsymbol{x}') d\boldsymbol{x}\, d\boldsymbol{u}\, d\boldsymbol{x}', \\
t = 1: \quad & \int \mu_1^\pi(\boldsymbol{x}')\boldsymbol{\phi}(\boldsymbol{x}') d\boldsymbol{x}' = \hat{\boldsymbol{\phi}}_1, \\
1 \leq t < H: \quad & \int \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x}) \log \frac{\mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})}{q_t(\boldsymbol{x}, \boldsymbol{u})} d\boldsymbol{x}\, d\boldsymbol{u} \leq \epsilon, \\
t = H: \quad & \int \mu_H^\pi(\boldsymbol{x}_H) \log \frac{\mu_H^\pi(\boldsymbol{x}_t)}{q_H(\boldsymbol{x}_H)} d\boldsymbol{x} \leq \epsilon,
\end{aligned} \tag{14}$$

Note that the constraints are always satisfiable as $q_t(\boldsymbol{x}, \boldsymbol{u})$ has been generated from roll-outs on our real system, and, hence, $\mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x}) = q_t(\boldsymbol{x}, \boldsymbol{u})$ is always a valid solution, although not the optimal one.

The stated optimization problem can be solved by the method of Lagrangian multipliers. The state-action probability $p_t^\pi(\boldsymbol{x}, \boldsymbol{u}) = \mu_t^\pi(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})$ can be obtained for each sample in closed form and is given by

$$p_t^\pi(\boldsymbol{x}, \boldsymbol{u}) \propto q_t(\boldsymbol{x}, \boldsymbol{u}) \exp\left(\frac{A_t(\boldsymbol{x}, \boldsymbol{u})}{\eta_t}\right), \tag{15}$$

$$A_t(\boldsymbol{x}, \boldsymbol{u}) = r_t(\boldsymbol{x}, \boldsymbol{u}) + \mathbb{E}_{p_t(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{u})}[v_{t+1}(\boldsymbol{x}')] - v_t(\boldsymbol{x}) \tag{16}$$

where $v_t(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\theta}_t$. The parameters η_t and $\boldsymbol{\theta}_t$ denote the Lagrangian multipliers for the KL-constraints and the constraints for $\mu_t^\pi(\boldsymbol{x})$, respectively. The Lagrangian multipliers can be found by minimizing the dual function $g(\eta_{1:H}, \boldsymbol{\theta}_{1:H})$, i.e.,

$$\{\eta_{1:H}^*, \boldsymbol{\theta}_{1:H}^*\} = \text{argmin}_{\eta_{1:H}, \boldsymbol{\theta}_{1:H}} g(\eta_{1:H}, \boldsymbol{\theta}_{1:H}), \quad s.t: \eta_t > 0, \forall t. \tag{17}$$

The derivation of the dual function is given in the appendix. We can see that our old data distribution $q_t(\boldsymbol{x}, \boldsymbol{u})$ is weighted by an exponential function. The term $A_t(\boldsymbol{x}, \boldsymbol{u})$ in the nominator strongly resembles the structure of an advantage function, if we would assume that $v_{t+1}(\boldsymbol{x}')$ denotes a value function. However, such a relationship can not be established. To improve our understanding of the meaning of the function v_t, we examine the structure of the dual function in more detail. The dual function $g(\eta_{1:H}, \boldsymbol{\theta}_{1:H})$ is given by

$$g(\eta_{1:H}, \boldsymbol{\theta}_{1:H}) = \hat{\boldsymbol{\phi}}_1^T \boldsymbol{\theta} + \sum_{t=1}^{H} \eta_t \epsilon + \sum_{t=1}^{H} \eta_t \log Z_t, \text{ with} \tag{18}$$

$$Z_{t<H} = \iint q_t(\boldsymbol{x}, \boldsymbol{u}) \exp\left(\frac{A_t(\boldsymbol{x}, \boldsymbol{u})}{\eta_t}\right) d\boldsymbol{x}\, d\boldsymbol{u}, \tag{19}$$

$$Z_H = \int q_H(\boldsymbol{x}) \exp\left(\frac{r_H(\boldsymbol{x}) - v_H(\boldsymbol{x})}{\eta_H}\right) d\boldsymbol{x}. \tag{20}$$

We can see that, due to the function $A_t(\boldsymbol{x}, \boldsymbol{u})$, each $v_t(\boldsymbol{x})$ is coupled with the function $v_{t-1}(\boldsymbol{x})$ from the previous as well as with $v_{t+1}(\boldsymbol{x})$ from the next time step. Hence, all functions $v_t(\boldsymbol{x})$ are connected and $v_t(\boldsymbol{x})$ can not be determined in a dynamic programming fashion by a backwards iteration. While the optimization of the dual function is more expensive than the traditional backwards iteration, it is also the biggest strength of the information theoretic approach, as only a combined optimization of the $\boldsymbol{\theta}_t$ vectors can control the 'damage' imposed on the state distributions $\mu_t^\pi(\boldsymbol{x})$. In contrast to approximate dynamic programming, the computation of the value function as intermediate step is not necessary. In the ITSOC formulation, the function $v_t(\boldsymbol{x})$ and the policy are always connected and we cannot compute one term without the other.

3.3 Sample-based ITSOC

While ITSOC offers an analytical solution for the policy, the dual function can in general not be evaluated analytically due to the integrals over the state action space. However, the dual function can be straightforwardly approximated by samples obtained from the distributions $q_t(\boldsymbol{x}, \boldsymbol{u})$. The sample-based dual function is given in the appendix. As a consequence, we also only obtain the probabilities $\mu_t^{\pi}(\boldsymbol{x})\pi_t(\boldsymbol{u}|\boldsymbol{x})$ for a discrete set of samples.

In order to create roll-outs with our policy, we need to resort to a parametric policy $\tilde{\pi}_t(\boldsymbol{u}|\boldsymbol{x};\boldsymbol{\omega}_t)$, where $\boldsymbol{\omega}_t$ denotes the parameters of the policy. The parameters can be found by approximating the distribution $\pi_t(\boldsymbol{u}|\boldsymbol{x})$, which we implement by minimizing the Kullback-Leibler divergence $\mathrm{KL}(\pi_t(\boldsymbol{u}|\boldsymbol{x})||\tilde{\pi}_t(\boldsymbol{u}|\boldsymbol{x};\boldsymbol{\omega}_t))$ between $\pi_t(\boldsymbol{u}|\boldsymbol{x})$ and $\tilde{\pi}_t(\boldsymbol{u}|\boldsymbol{x};\boldsymbol{\omega}_t)$ on the given set of samples [7], i.e.,

$$\begin{aligned} \boldsymbol{\omega}_t^* &= \mathrm{argmin}_{\boldsymbol{\omega}_t} \mathrm{KL}\left(\pi_t(\boldsymbol{u}|\boldsymbol{x})||\tilde{\pi}_t(\boldsymbol{u}|\boldsymbol{x};\boldsymbol{\omega}_t)\right), & (21) \\ &= \mathrm{argmax}_{\boldsymbol{\omega}_t} \sum_{\boldsymbol{x},\boldsymbol{u}} \exp\left(\frac{A_t(\boldsymbol{x},\boldsymbol{u})}{\eta_t}\right) \log \tilde{\pi}_t(\boldsymbol{u}|\boldsymbol{x};\boldsymbol{\omega}_t) + \mathrm{const.} & (22) \end{aligned}$$

This optimization yields a weighted maximum likelihood estimate of $\boldsymbol{\omega}_t$ with the weightings

$$d_{i,t} = \exp(A_t(\boldsymbol{x}_i, \boldsymbol{u}_i)/\eta_t). \tag{23}$$

Note that we dropped the distribution $q_t(\boldsymbol{x}, \boldsymbol{u})$ as the samples have already been generated by q_t. With the same trick, the distribution $q_t(\boldsymbol{x}, \boldsymbol{u})$ can also be dropped from the dual function $g(\eta_{1:H}, \boldsymbol{\theta}_{1:H})$, and, hence, the distribution $q_t(\boldsymbol{x}, \boldsymbol{u})$ does not need to be known in its analytic form.

4 Finding Locally Optimal Policies with Information Theoretic Stochastic Optimal Control

In this section, we present our resulting algorithm for estimating robust, locally optimal policies with ITSOC. We will also extend our algorithm with a local model-learning algorithm. Using the learned models, we can use our approach for model-based reinforcement learning where the system dynamics are unknown. Additionally, we can use the learned models as efficient simulator to create virtual roll-outs.

4.1 Representation of the Policy and the Feature Function

Inspired by optimal controllers in the LQG case [22, 24], we will use a linear representation for the policy and a quadratic representation for the functions $v_t(\boldsymbol{x})$, i.e.,

$$\pi_t(\boldsymbol{u}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{k}_t + \boldsymbol{K}_t\boldsymbol{x}, \boldsymbol{\Sigma}_t), \quad v_t(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{V}_t\boldsymbol{x} + v_t^T\boldsymbol{x},$$

where $\boldsymbol{V}_t$ is an upper triangular matrix. The parameters $\boldsymbol{k}_t$, $\boldsymbol{K}_t$ and $\boldsymbol{\Sigma}_t$ of the policy can be efficiently estimated by a weighted linear regression while the parameters $\boldsymbol{\theta}_t = \{\upsilon_t, \boldsymbol{V}_t\}$ of υ_t need to be obtained by optimizing the dual function. Assuming a set of samples $\mathscr{D} = \{\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t}\}_{i=1...M,t=1...H}$ and the weightings $d_{i,t}$, the parameters $\boldsymbol{k}_t$, $\boldsymbol{K}_t$ and $\boldsymbol{\Sigma}_t$ can be computed by the following weighted linear regression.

$$\begin{bmatrix} \boldsymbol{k}_t^T \\ \boldsymbol{K}_t^T \end{bmatrix} = (\boldsymbol{X}_t^T \boldsymbol{D}_t \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t^T \boldsymbol{D}_t \boldsymbol{U}_t, \tag{24}$$

$$\boldsymbol{\Sigma}_t = \frac{\sum_i d_{i,t} (\boldsymbol{\mu}_{i,t} - \tilde{\boldsymbol{u}}_{i,t})(\boldsymbol{\mu}_{i,t} - \tilde{\boldsymbol{u}}_{i,t})^T}{\sum_i d_{i,t}}, \tag{25}$$

with

$$\boldsymbol{X}_t = \begin{bmatrix} 1 & \boldsymbol{x}_{i,t} \\ \vdots & \\ 1 & \boldsymbol{x}_{M,t} \end{bmatrix}, \quad \boldsymbol{U}_t = \begin{bmatrix} \boldsymbol{u}_{i,t} \\ \vdots \\ \boldsymbol{u}_{M,t} \end{bmatrix}, \quad \boldsymbol{D}_t = \text{diag}([d_{i,t}]_{i=1...M}) \tag{26}$$

and

$$\boldsymbol{\mu}_{i,t} = \boldsymbol{k}_t + \boldsymbol{K}_t \boldsymbol{x}_{i,t}. \tag{27}$$

Note that in difference to SOC approaches based on linearisations of the model, we use stochastic policies and explicitly control the exploration of the policy by estimating the covariance matrix $\boldsymbol{\Sigma}_t$ used for exploration. Hence, we do not work with a single trajectory as linearisation point, but with a distribution of trajectories. The covariance matrix $\boldsymbol{\Sigma}_t$ is based on our weighted samples, and, hence, the 'exploration direction' of our policy also adapts to our weighted samples.

4.2 Learning Local Models for Reinforcement Learning

Since we also want to use our approach for RL, we will learn the system dynamics model $p_t(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{u})$ from our generated samples. Following up our assumption of locality, we will use simple linear Gaussian models for each time step, i.e., $p_t(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{u}) = \mathcal{N}(\boldsymbol{x}'|\boldsymbol{a}_t + \boldsymbol{A}_t \boldsymbol{x} + \boldsymbol{B}_t \boldsymbol{u}, \boldsymbol{C}_t)$. We obtain the parameters $\boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{B}_t$ and $\boldsymbol{C}_t$ of the models from our sampled data points $(\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t})$ by a standard maximum likelihood estimate. As our experiments show in Section 5, the representation as time-dependent linear models also allows for a more efficient optimization of the policies, resulting in policies of higher quality. In the RL formulation of our algorithm, we assume that the reward function r_t is known as prior knowledge and only the system dynamics need to be learned.

4.3 Estimating the Expected Feature Vector

The information-theoretic formulation requires estimating $\mathbb{E}_{p_t(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u})}[v_t(\boldsymbol{x}')]$ for each sample. Due to the system dynamics representation as linear time-varying Gaussian models and the quadratic feature representation, this operation can be performed analytically by

$$\begin{aligned}\mathbb{E}_{p_t(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u})}[v_{t+1}(\boldsymbol{x}')] &= \int \mathcal{N}(\boldsymbol{x}'|\boldsymbol{a}_t + \boldsymbol{A}_t\boldsymbol{x} + \boldsymbol{B}_t\boldsymbol{u}, \boldsymbol{C}_t)\left(\boldsymbol{x}^T\boldsymbol{V}_t\boldsymbol{x} + v_t^T\boldsymbol{x}\right) d\boldsymbol{x}' \\ &= \boldsymbol{\mu}_t(\boldsymbol{x},\boldsymbol{u})^T\boldsymbol{V}_t\boldsymbol{\mu}_t(\boldsymbol{x},\boldsymbol{u}) + \text{trace}(\boldsymbol{C}_t\boldsymbol{V}_t) + v_t^T\boldsymbol{\mu}_t(\boldsymbol{x},\boldsymbol{u}) \qquad (28)\end{aligned}$$

with $\boldsymbol{\mu}_t(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{a}_t + \boldsymbol{A}_t\boldsymbol{x} + \boldsymbol{B}_t\boldsymbol{u}$. Equation (28) can be rewritten in the feature vector representation $\mathbb{E}_{p_t(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u})}[\boldsymbol{\phi}(\boldsymbol{x}')^T]\boldsymbol{\theta}_{t+1}$ which we omit due to space constraints.

4.4 Optimizing the Dual

In our ITSOC setup, we want to cope with time horizons as large as $H = 50$ or even $H = 100$. Additionally the quadratic feature representation easily results in 40 to 60 parameters per time step. Hence, we easily end up with up to several thousands of parameters for the minimization of the dual function. In comparison to the formulation in [7], which has been used to optimize high-level policies, the number of parameters is increased by a factor of 10 to 100. With such a high number of parameters, the constraint optimization problem of the dual function can also run into numerical problems. Hence, an efficient and numerical stable optimization procedure is required. In this section we will elaborate on the different techniques that we approached. In all cases we provided the algorithms with the analytic gradients and Hessians.

Constrained optimization

Since the dual is constrained by the lower bound for $\eta 0$, a constrained optimizer is the most intuitive approach. In our case we used a trust-region-reflective algorithm [5, 14]. Unfortunately the optimizer was not able to find reliable solutions in a suitable time, due to numerical instabilities.

Unconstrained optimization

A common approach to circumvent lower bound constraints is to apply the exp-trick [21], in our case resulting in an unconstrained problem. The nature of the trick is to substitute η by a non-negative function $\eta = f(\zeta) = \exp(\zeta)$ and optimize for ζ instead of η. Again, the optimizer did not achieve satisfactory results. We suspect the cause to be the introduction of additional non-linearities in combination with numerical instability. We also tried different sigmoidal substitution functions, e.g. arctangent, logistic or algebraic function. Non showed any significant improvement.

Input: Initial estimate for all $\boldsymbol{\theta}$, initial estimates for all η **Compute initial** MAE(ϵ) and $f_{\max}$
while (MAE$(\epsilon) > \Delta\epsilon$) ‖ (MAE$(\boldsymbol{\theta}) > \Delta\boldsymbol{\theta}$) **Optimize** g **for each** η_t: $\eta_t = \arg\min_{\eta_t} g(\eta_{1:H}; \boldsymbol{\theta}_{1:H}), \forall t$ **Optimize** g **for all** $\boldsymbol{\theta}_t$: $\boldsymbol{\theta}_{1:H} = \arg\min_{\boldsymbol{\theta}_{1:H}} g(\boldsymbol{\theta}_{1:H}; \eta_{1:H})$ **Compute** MAE(ϵ) and MAE$(\boldsymbol{\theta})$
Output: optimized $\boldsymbol{\theta}_{1:H}$ and $\eta_{1:H}$

Table 1: Iterative optimization of the dual function. We decompose the optimization problem in finding the single η_t values while keeping the $\boldsymbol{\theta}$ value fixed. Subsequently, we optimize for the $\boldsymbol{\theta}_{1:H}$ values, which is an unconstrained optimization problem. Both algorithms are only run for a small number of internal optimization steps to avoid oscillations in the parameter updates. As termination criteria for our iterative optimization procedure, we check wether our constraints are approximately met. If this is the case, the optimization is stopped.

Iterative optimization

A closer look at the dual reveals, that the Lagrangian parameters η and $\boldsymbol{\theta}$ can be optimized separately using a coordinate descent [2] like method. First, we optimize for each η_t individually while fixing the $\boldsymbol{\theta}_t$ parameters. Next, we fix the η parameters and optimize for all $\boldsymbol{\theta}_t$. We iterate over these two steps until we reach a satisfactory solution. The optimization for $\boldsymbol{\theta}_t$ is unconstrained, therefore we used a large-scale method [14] here, whereas we applied a trust-region-reflective optimization [14] for each of the η_t. Both algorithms only run for a small number of iterations, e.g, 10 iterations. Using this approach we were able to find good solutions in a suitable amount of time.

We experienced, that using the progress of the dual value as a termination criteria for the optimization is rather ineffective as both constraints can be already satisfied with quite high accuracy while the dual value is still far from converging. Therefore, we introduced a new termination criteria which relaxes the KL and the feature constraints. At each iteration, we compute the maximum absolute error MAE_ϵ between the current KL divergence of each time step and the desired KL divergence ϵ. If the error is smaller then a certain threshold $\Delta\epsilon$, we assume the constraint as approximately satisfied. Additionally we compute the maximum absolute error $\text{MAE}_{\boldsymbol{\theta}}$ for the average state features, where we normalize with respect to the standard deviation of the corresponding state feature. We again regard the feature constraint as sufficiently satisfied if $\text{MAE}_{\boldsymbol{\theta}}$ is beneath the threshold $\Delta\boldsymbol{\theta}$. If both constraints are satisfied, we stop the iterations. Applying these termination criteria, we experienced only small if any penalties in the performance of the algorithm while benefiting greatly in terms of computation time. The resulting algorithm for optimizing the dual is given in Algorithm 1.

4.5 Algorithm

In each iteration we use the currently estimated policy $\tilde{\pi}_t(\boldsymbol{u}|\boldsymbol{x}; \boldsymbol{\omega}_t)$ to create N_{real} roll-outs. Since we assume the sampling to be an expensive process, we construct the current set of samples $\mathscr{D}$ from the last L iterations. The samples from the real system are only used to train the linear time-varying models, which again are used to generate an arbitrary amount of virtual samples $\tilde{\mathscr{D}}$.
We then use the virtual samples $\tilde{\mathscr{D}}$ with model learning to replace the the integrals in the dual function of ITSOC. The resulting algorithm is summarized in Algorithm 2. We start the algorithm with a rather large variance for the initial policy π_t^0. At each iteration, the policy improves towards the (locally) optimal policy. Subsequently, the ITSOC algorithm decreases the exploration automatically in each iteration until it finally collapses to a deterministic policy.

Input: KL-bound ϵ, number of iterations K, samples N and virtual samples M, L last iterations to reuse
Initialize $\tilde{\pi}_t^0$ using Gaussians with zero mean and high variance.

for $k = 1$ to K ... # iterations

- **Collect data on the real system following** $\tilde{\pi}_t^{k-1}$:
  $\mathscr{D}_k = \{\boldsymbol{x}_{i,t}, \boldsymbol{u}_{i,t}\}_{i=1...N, t=1...H}$
- **Re-use last** L **iterations**: $\mathscr{D} = \{\mathscr{D}_l\}_{l=\max(1,k-L)...k}$
- **Estimate time-varying linear models using** $\mathscr{D}$
- **Collect data on the learned system following** $\tilde{\pi}_t^{k-1}$:
  $\tilde{\mathscr{D}}_k = \{\tilde{\boldsymbol{x}}_{i,t}, \tilde{\boldsymbol{u}}_{i,t}\}_{i=1...M, t=1...H}$
- **Minimize dual function on** $\tilde{\mathscr{D}}_k$:
  $[\eta_{1:H}, \boldsymbol{\theta}_{1:H}] = \text{argmin}_{\eta'_{1:H}, \boldsymbol{\theta}'_{1:H}} g\left(\eta'_{1:H}, \boldsymbol{\theta}'_{1:H}; \tilde{\mathscr{D}}_k\right)$
- **Estimate new policy** $\tilde{\pi}_t^k$ **for each** t:
  Compute weighting $d_{t,i}$, see Eq. (23)
  Compute policy parameters $\boldsymbol{k}_t, \boldsymbol{K}_t, \boldsymbol{\Sigma}_t$ using $\tilde{\mathscr{D}}_k$ and weightings $d_{t,i}$, see Eq. (25)

Output: Policies $\tilde{\pi}_t^K(\boldsymbol{u}|\boldsymbol{x})$ for all $t = 1, \ldots, H$

Table 2: The information-theoretic SOC algorithm. We collect data on the real system which we use to estimate individual linear models for each time step. These models are used to generate virtual samples by simulating whole roll-outs. We minimize the dual-function defined on these virtual samples and, subsequently, compute a weighting for each data point. This weighting is used to obtain new policy parameters by using a weighted maximum likelihood estimate.

4.6 Relation to Existing Approaches

There are tight connections but also important differences to SOC algorithms that use KL-terms [19, 1]. The KL-terms in these approaches act on the policy instead of directly acting on the state action distribution. As a consequence, these approaches have to rely on value function approximation and suffer from the drawbacks of an unstable policy update that comes with it. Moreover, [19, 1] use the KL as punishment term that is traded-off with the traditional cost function. As a consequence, the temperature parameter η has to be chosen by the user while in ITSOC it is given from the optimization problem. In ITSOC, the user has to choose the ϵ parameter, that is typically much easier to choose and can stay constant during the iterations.

The path integral (PI) formulation of SOC [23] also shares a lot of similarities with the ITSOC formulation. We can clearly see that the value function given for the path integrals in Eq. 7 and the dual function for ITSOC share the same structure. However, in ITSOC, the integral is performed over the state-action space while in the PI formulation, the integral is only performed for the action space. Such formulation would, in theory, require that we restart the process for many times at state $\boldsymbol{x}_t$ to obtain the optimal controls for state $\boldsymbol{x}_t$. This requirement is typically neglected in common PI implementations [23], which is, however, only a heuristics. Furthermore, the PI approach also does not control the damage on the state distributions by the policy update. It is based on Monte-Carlo roll-outs and therefore requires a lot of samples. The temperature parameter η_t are also chosen by heuristics in the PI approach.

Our algorithm was inspired by the policy search community, where related information-theoretic bounds have already been introduced to learn high-level policies [7]. The algorithm given [7] did not use a model for estimating the expected next state features and, therefore, produced biased solutions. Moreover, the optimization problem in [7] could be solved straightforwardly, as the number of parameters was much lower. In order to apply the information-theoretic policy updates for SOC, we needed to develop our more efficient and numerically stable optimization strategy.

AWR with information-theoretic bounds

For comparisons, we will use a variant of the advantage weighted regression (AWR) algorithm [16] to compare the effects of value function approximation and matching the expected state features. In our variant of AWR, we also use a time-dependent quadratic value function and time-dependent linear feedback controllers as policy representation. We reformulate AWR such that we can optimize the temperature of the soft-max distribution with a similar information-theoretic bound. Therefore, we formulate a similar optimization problem as for ITSOC, which can, however, be solved independently for each time step. In time step t, we want to maximize the expected advantage-function while we keep the KL between the new and old policy bounded, i.e.,

$$\max_{\pi_t} \int_{\boldsymbol{s},\boldsymbol{\omega}} q_t(\boldsymbol{s})\pi_t(\boldsymbol{\omega}|\boldsymbol{s})A_t(\boldsymbol{s},\boldsymbol{\omega}), \quad \text{s.t:} \int_{\boldsymbol{s}} q_t(\boldsymbol{s})\mathrm{KL}(\pi_t(\boldsymbol{\omega}|\boldsymbol{s})||q_t(\boldsymbol{\omega}|\boldsymbol{s}))d\boldsymbol{s} \leq \epsilon. \tag{29}$$

The solution for $\pi_t(\boldsymbol{\omega}|\boldsymbol{s})$ can be obtained similarly by the method of Lagrangian multipliers and is given by

$$\pi_t(\boldsymbol{\omega}|\boldsymbol{s}) = q_t(\boldsymbol{u}|\boldsymbol{s}) \exp\left(\frac{A_t(\boldsymbol{s},\boldsymbol{\omega})}{\eta_t}\right). \tag{30}$$

We compute the advantage of the given samples $\boldsymbol{x}^{[i]}$ and $\boldsymbol{u}^{[i]}$ by

$$A_t(\boldsymbol{s},\boldsymbol{\omega}) = r_t(\boldsymbol{x},\boldsymbol{u}) + \mathbb{E}_{\boldsymbol{s}'}[V_{t+1}(\boldsymbol{s}')] - V_{t,\text{old}}(\boldsymbol{s})), \tag{31}$$

where $V_{t+1}(\boldsymbol{s}')$ is obtained by an advantage weighted regression where we use the states $\boldsymbol{s}_{t+1}$ as inputs and the Q-values $Q_{t+1}(\boldsymbol{x},\boldsymbol{u}) = r_{t+1}(\boldsymbol{x},\boldsymbol{u}) + \mathbb{E}_{\boldsymbol{s}'}(V_{t+2}(\boldsymbol{s}')$ as target values. The value function $V_{t,\text{old}}(\boldsymbol{s})$ for the current time step is approximated by using the samples without weighting, i.e., we compute the value function when using the old policy for the current time step while following the new policy for the future time steps. Because we do not have a constrained on the state distribution, the optimization problem becomes much simpler but the policy update also becomes less stable. We will use the same strategy to learn the linear time dependent models as well as to generate virtual samples as for the ITSOC method to allow

5 Experiments

We evaluate our information-theoretic SOC algorithm on a 2-link planar arm and on a 4-link non-linear planar arm in different scenarios. The links of the arm have a length of one meter and a mass of one kilogram. The maximum torque is set to $25N$ and we also implement a simple friction model for the joints. We use 50 time steps and a time step of dt $= 0.066$s. In addition to the control noise used in some experiments, we always use a Gaussian distribution for the intial state distribution $p_1(\boldsymbol{x})$. Hence, we do not want to estimate a single nominal trajectory but a controller which works well in a broader area for the initial state.

We compare our algorithm against our variant of advantage weighted regression (AWR), see [16], a variant of PILCO [8] where we also learn linear forward models and linear feedback controllers as policy instead of Gaussian Process models and RBF policies

as in the original PILCO algorithm and against the AICO algorithm [27] that uses linearisation of the underlying system dynamics. We evaluate the robustness of these algorithms to system noise and non-linearities in the dynamics. Moreover, we evaluate our approach in a model-based RL setup where we want to minimize the amount of real-robot interactions. Here, we compare our time-varying linear model to a constant linear model for which we use the data from all time steps for estimating the parameters.

5.1 Scenarios

2-Link Scenarios.

We use the first scenario of the 2-Link planar arm as illustration to show how the state distribution evolves with increasing number of iterations in the different approaches. The reward function is given as squared punishment term for the torques at each time step and a via-point reward $r_v(\boldsymbol{x})$ at time step $t_1 = 25$ and $t_2 = 50$. The via-point reward $r_v(\boldsymbol{x})$ is proportional to the negative squared distance in task space $r_v(\boldsymbol{x}) = -(\boldsymbol{y} - v)^T \boldsymbol{H}(\boldsymbol{y} - v)$, where $\boldsymbol{y} = f(\boldsymbol{x})$ is the end-effector position for state $\boldsymbol{x}$ and $\boldsymbol{H}$ is set to 10^4 for all state dimensions which denote a position. We will refer to this task as 2-link reaching task. The state vector of the robot is 4 dimensional, resulting in a 14 dimensional feature vector.

In a second scenario, we learn a swing-up movement to evaluate the approaches on highly non-linear tasks. The goal is to swing-up and balance the pendulum such that it reaches the upright position with zero velocity at $t = 50$. In order to simplify the learning problem we also punish the agent if the joints leave a pre-defined area of $q_1 \in [2/3\pi, 3\pi]$ and $q_2 \in [-\pi, \pi]$. This punishment term allows us to punish overturning of the pendulum and to pre-select one of the two possible swing-up solutions. Hence, we avoid the multi-modalities that are inherent in the solution space, and, hence, we simplify the learning problem.

4-Link Scenarios.

In the first scenario, the arm has to reach two via-points in task space while we disable gravity. The state vector $\boldsymbol{x}$ of the robot is 8 dimensional containing all joint positions and velocities. A $d = 8$ dimensional state vector results in a 44 dimensional feature vector $\boldsymbol{\phi}(\boldsymbol{x})$, consisting of 8 linear and $d(d+1)/2 = 36$ squared terms.

In the second scenario, we test the robustness against non-linearities by enabling gravity. However, in order to simplify the search problem, we additionally add a high punishment term if the joint angles leave a pre-specified area. Hence, we limit the search space and avoid multi-modal solutions.

Finally, we extend the 4-link arm scenario to the task of playing robot tennis. We add the position and velocity of a ball into our state space. The ball starts with random initial x-position and velocity. The ball moves with constant velocity, however, at time step 25 it bounces against the floor, introducing a perturbation. At the bounce, the velocities are perturbed by a significant amount of multiplicative noise, changing the incoming position of the ball. The agent has to hit the ball at time step 50. To do so, it has to reach a vicinity of 20cm of the ball position at $t = 50$. If it succeeds, it gets a positive reward proportional to the velocity of the end-effector in y-direction. Otherwise, the reward is proportional to the negative squared distance to the ball location. We add the position and the velocity of the ball in our state space, resulting in a 12 dimensional state vector and 90 dimensional feature vector.

Evolution of the State-Distributions

To illustrate the behavior of different algorithms, we show subsequent trajectory distributions during the iterations of the algorithms in Figure 1 for the two-link reaching task. We show the distributions of our ITSOC algorithm, the approximate dynamic program-

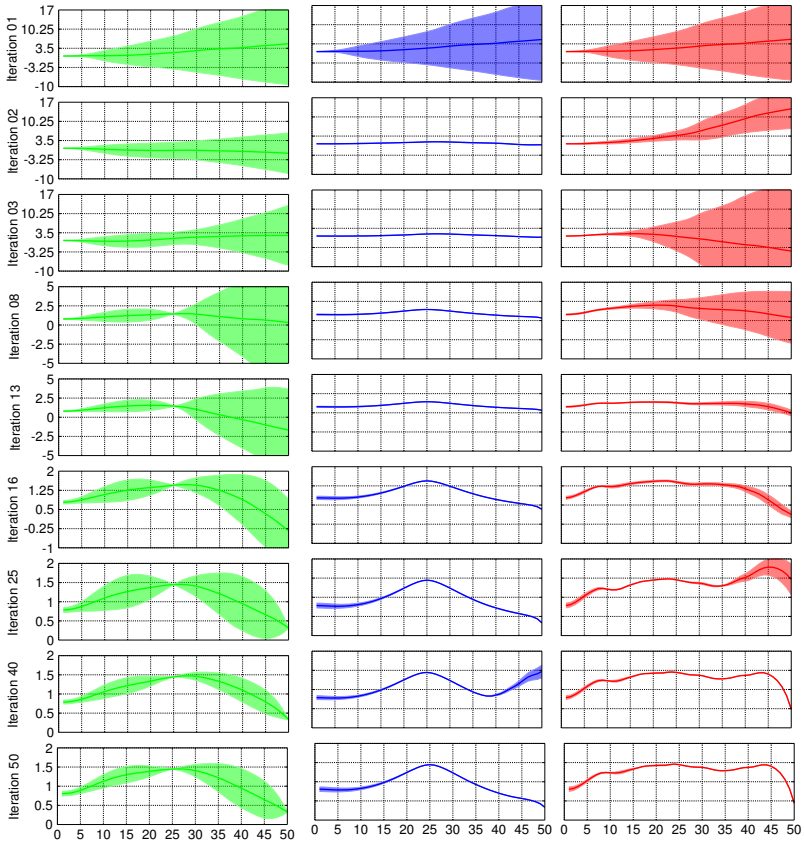
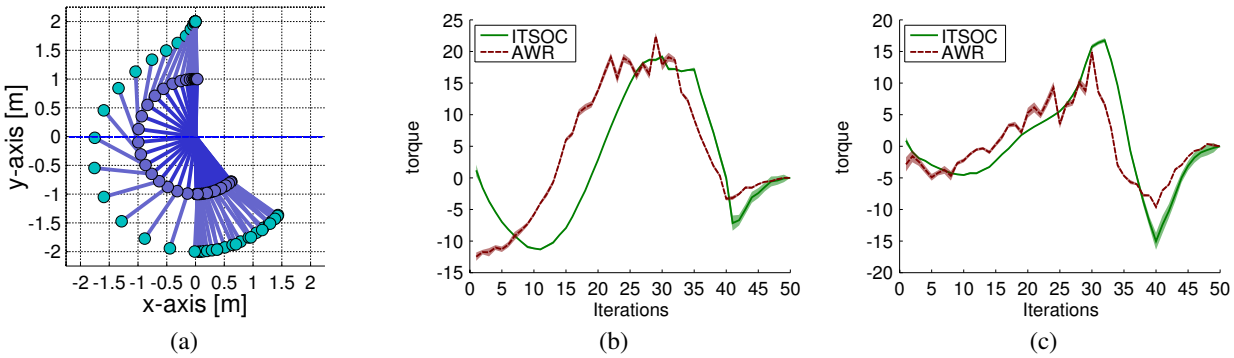


Figure 1: The plot shows the trajectory distributions for the last joint q_2 of a two link planar arm with ITSOC (left), linear PILCO (middle) and AWR (right). The x-axis depicts the time steps while in the rows we can see the distribution for subsequent iterations. While the AWR approach shows a jumping behavior and a resulting limited learning progress, the information-theoretic approach smoothly transforms the initial exploratory distribution into a goal-directed movement. PILCO quickly jumps to a good solution, but fails to further improve the policy. The average reward of ITSOC is -11.2, while for AWR it is -52.4 and PILCO oscillates between -3 and -2000.

ming based AWR algorithm and the linear PILCO method. We can see that the distributions change smoothly for our approach, allowing the algorithm to efficiently find an optimal solution. In contrast, the distributions jump for AWR and it quickly converges to a deterministic, but sub-optimal policy. Linear PILCO jumps to good solutions already after a few iterations, however, as it exploits the models greedily, already small inaccuracies in the model can cause jumps in the trajectory distribution.

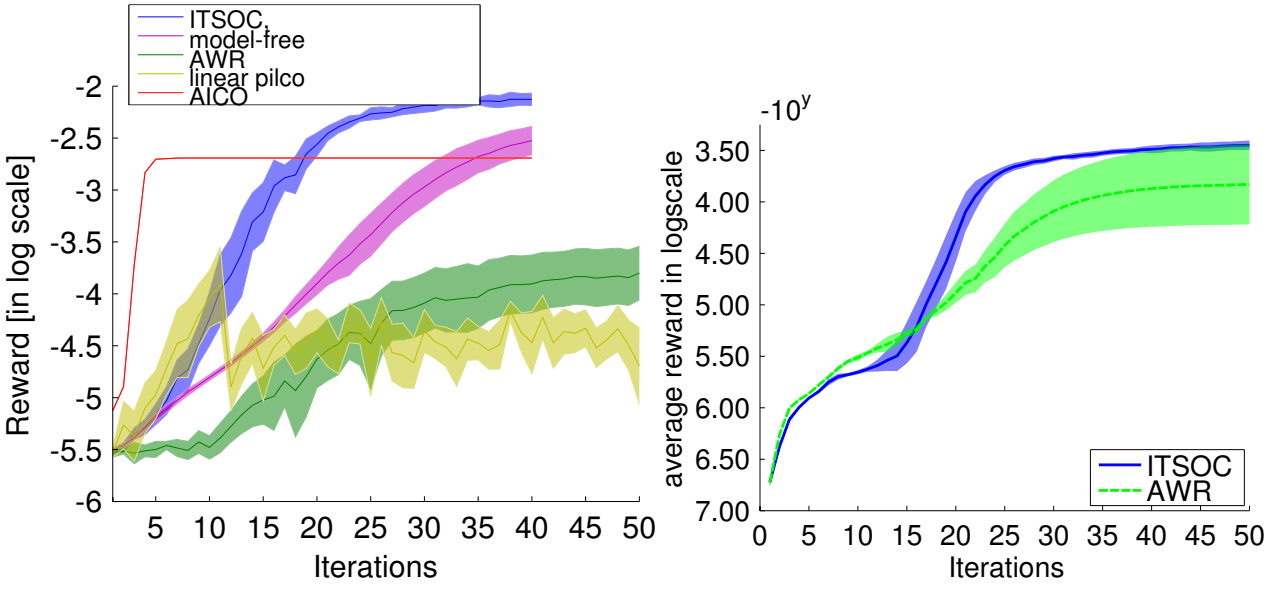
5.2 Evaluation on a highly Non-linear Task

As a proof of concept, to show that learning time-dependent linear models also works for highly non-linear tasks, we learn to swing-up a two-link pendulum. As reward function, we punish the squared distance of the end-effector to the up-right position in the last 10 time steps. The two-link pendulum is able to directly go to the upright position, however, due to the torque punishment factor, such behavior is suboptimal. The optimal reward can only be reached by performing a swinging movement. An illustration of the learned movement can be seen in Figure 2. The agent was able to swing up the pendulum while smoothly controlling its torques. The resulting torque trajectories are shown for the ITSOC algorithm as well as for the value-based AWR method. AWR only found a suboptimal solution that exhibits considerably more jerk in the torque profile.



(a) (b) (c)

Figure 2: (a)Two-Link Pendulum Swing-Up solution found by ITSOC. (b,c)



(a) Task-Space Reward (b) Uni-Modal Reward

Figure 3: (a)We compare the ITSOC algorithm to advantage weighted regression (AWR) and AICO. The evaluations are done for different noise levels of additive control noise. Model-based ITSOC clearly shows the best performance while the biased version of ITSOC quickly degrades with an increasing noise level. Note that all plots are in log-scale for the y-axis. (b) Comparison of ITSOC and AWR on the 4-link task with gravity. We simplified the task by punishing the agent when the joints leave a pre-specified area. Hence, the solution space of the task becomes uni-modal. In this setup, the value function can be efficiently approximated by a squared function, and, hence, AWR performs almost as good as ITSOC.

Comparison of Algorithms

We first compare our approach to the AWR approach, the linear PILCO and the AICO approach on this scenario. The results are shown in Figure 3(a). The ITSOC approach significantly outperforms all other methods in terms of learning speed and/or quality of the final policy. Linear PILCO quickly jumps to a good solution, but fails to find a solution of the same quality due to the lack of exploration. A similar behavior could be observed for the AICO algorithm that uses the known system dynamics.

Robustness to System Noise

In this evaluation we use the 4-link task with the via-points in end-effector space. We evaluate our algorithm with learned time-varying models, and without the usage of models. In addition, we compare to the AWR approach and the AICO approach on this scenario. For the model-free approach we do not estimate the expected next features but use just a single-sample estimate. Hence,

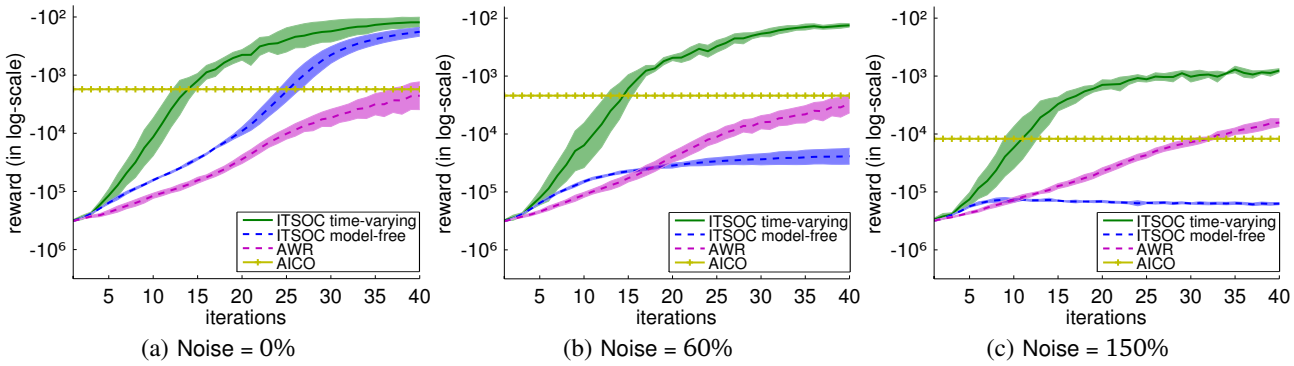


Figure 4: Comparison of information-theoretic (IT) SOC with learned models and without learned models where we use a single sample estimate for the expectation of the next features, which results in a bias in our optimization. In addition we compare to advantage weighted regression (AWR) and AICO. The evaluations are done for different noise levels of additive control noise. Model-based ITSOC clearly shows the best performance while the biased version of ITSOC quickly degrades with an increasing noise level. Note that all plots are in log-scale for the y-axis.

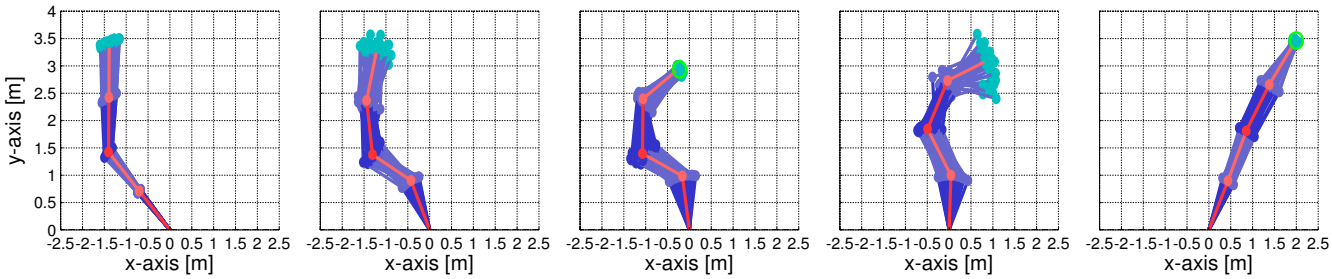


Figure 5: Illustration of the resulting postures for the via-point task in task space. The non-linear planar arm has to reach the via-points at $t = 25$ and $t = 50$ illustrated in green. The plot shows for each time step sample from the resulting distribution of postures. The mean of the postures is shown in red. The robot manages to reach the via-point while exhibiting a significant amount of variance in joint space.

the model free optimization is biased [7]. We use additive control noise with a standard deviation of $0\%, 60\%$ and 150% of the maximum torques which can be applied by the robot. We use $N = 500$ samples per iteration and do not keep samples from old iterations to avoid effects from the sampling process. The results are shown in Figure 4(a). For the evaluation without noise, both the model-free and model-based version of our algorithm estimate good policies where the model-based approach can still outperform the model-free method. As the model-based algorithm calculates its policy solely on the data generated by the learned time-varying models, we believe that the reason for the increased performance is that the linear model representation is favoured by our squared representation of the value function. However, both approaches significantly outperform the AWR approach in terms of learning speed and quality of the final policy. We also compute the optimal control law with AICO which results in a policy with an average reward of -1700 compared to -150 of the model-based approach. We believe the main reason for the lower performance of AICO is that the linearisations of the model are computed at a nominal trajectory whereas our approach uses the samples in a broader area to estimate more robust models. With an increasing level of the control noise, the performance of the model-free method quickly degrades due to the bias. The model-based information theoretic SOC method can still reliably outperform the AWR and the AICO approach.

We also illustrate the resulting postures for different time points in Figure 5 for the setting with 60% noise. The robot manages to reach the via-points (illustrated in green) in task space while it still exhibits a large variance in joint space. In between the via-points, also the variance in task space grows.

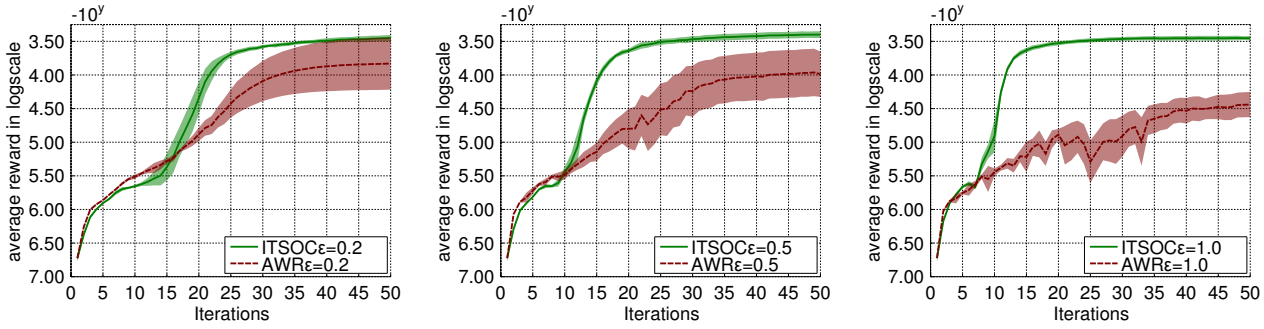


Figure 6: Evaluation of different KL-bounds ϵ for ITSOC and the modified AWR approach. We can see that while AWR only works well for a very small ϵ, ITSOC allows for the use of much higher ϵ values as it also ensures that the distance in the state-distributions stays bounded. As a consequence, ITSOC needs less samples to converge to the optimal solution.

Influence of Information-Theoretic Bounds

We evaluated the influence of the information theoretic bound ϵ for ITSOC as well as for the AWR algorithm. We use the uni-modal 4-link reaching task as both, AWR and ITSOC work well in this setup. As we can see from Figure 6, AWR needs a very small bound on the Kullback-Leibler divergence to learn the optimal solution and, hence, shows a limited learning speed. In contrast, ITSOC works well for much higher values of the KL-divergence as the KL-divergence also contains the state distributions. The general behavior is that, if we set the KL-bound too low, we result in a slow learning progress. However, if ϵ is too high, we get fast learning but we also get premature convergence. Yet, the algorithm works in a wide range of values for the parameter ϵ.

Robustness of the Feature Constraints

Next, we demonstrate the influence of the threshold $\mathrm{MAE}(\boldsymbol{\theta})$ for the state feature constraints as described in Algorithm 1. Figure 7(a) shows how the average reward decreases with a higher threshold, since the state feature constraint is further relaxed. Interestingly, the average reward is hardly affected until $\mathrm{MAE}(\boldsymbol{\theta}) = 1.5$, but then quickly drops with an increasing value of $\mathrm{MAE}(\boldsymbol{\theta})$. As we can see in Figure 7(b), the time needed to compute the optimization decreases exponentially with the increasing relaxation of the feature constraints. We conclude that a value of $\mathrm{MAE}(\boldsymbol{\theta})$ between 0.5 and 1.0 is a reasonable choice, at least for this example.

Robot Tennis

In the robot tennis example, we used 100 samples per iteration and learned the time-varying models with the last 500 samples. The robot could reliably hit the incoming ball at different positions with a high velocity in the y-direction. An illustration of the resulting posture time-series for two different hitting movement is shown in Figure (8). The range of the incoming balls is approximately $1.5m$ in the x-direction and $0.5m$ in the y-direction. We can see that the robot already adapts his movement in the beginning to the predicted incoming position of the ball, but he quickly adapts his movement when the ball changes his velocity when bouncing at the floor at time step $t = 40$. The performance of the policy with an increasing number of iterations is shown in Figure (8)(c). We also compared our approach to AWR on this more complex task. The results show that AWR again can not cope with the more complex reward function and converges slowly.

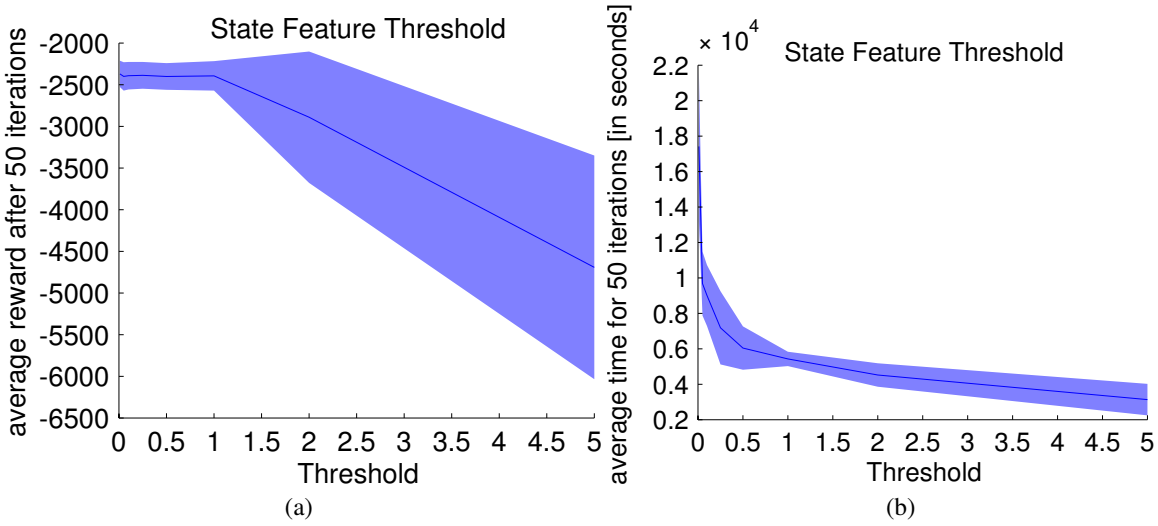


Figure 7: (a) The effect of increasing relaxation of the feature constraints MAE($\boldsymbol{\theta}$) on the average reward. (b) Computation time for different MAE($\boldsymbol{\theta}$). While the performance of ITSOC is almost unaffected for MAE($\boldsymbol{\theta}$) ≤ 1.0, the computation time depends exponentially on MAE($\boldsymbol{\theta}$). Hence, a reasonable choice for MAE($\boldsymbol{\theta}$) ≤ 1.0 in this example is 1.0.

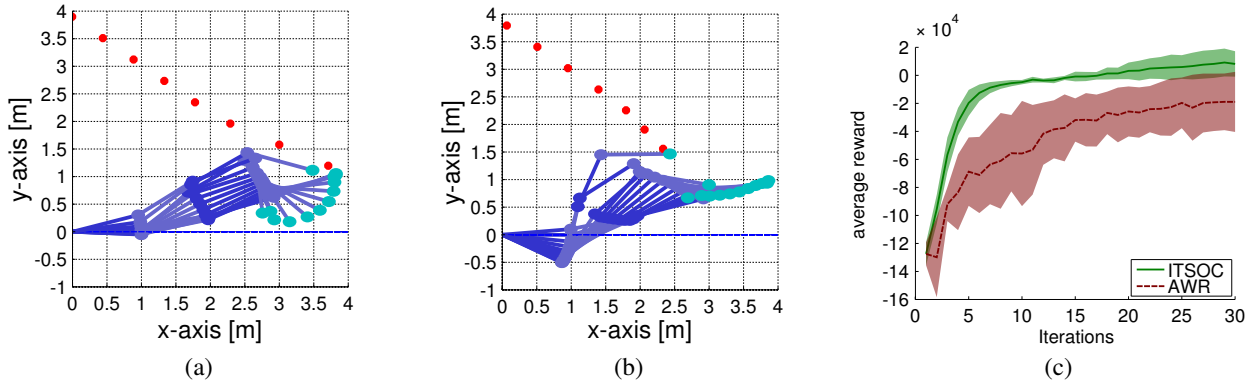
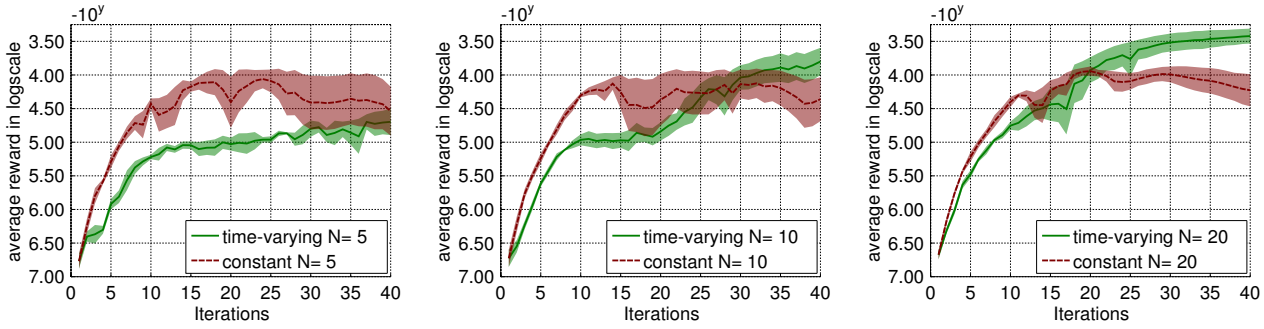


Figure 8: (a,b) Illustration of the robot playing tennis for two different configurations of the incoming ball. The ball has been simulated with constant velocity but bounces off the floor in a stochastic way, such that the robot needs to sensory feedback to adapt his movement. The robot can react to this perturbation and reliably hits the ball. (c) Comparison of ITSOC and AWR on the tennis task. While ITSOC learns shows a smooth learning process and converges to good solutions, the value function approximation of AWR causes jumps in the state distributions which results in a worse performance.

Model-Based Reinforcement Learning

In this experiment, we evaluate the ability of our approach for model-based reinforcement leaning. We use the 4-link pendulum task with gravity and two via-points in joint space. For our evaluation, we use a different number of new samples $N = 5$ and $N = 10$ and we always keep the data of the last $L = 10$ iterations. With the collected data we learn either a time-varying or a constant linear model. The constant linear model has the advantage that it can use more data points, however, it can not capture the system dynamics as well as the time-varying linear models. The comparison is shown in Figure 9. We can see that for a small number of samples, the time-varying models have too little data-points and consequently the constant model outperforms the time-varying model. However, for an increasing number of new samples, the time-varying model is still slower in the beginning, but outperforms the constant model in the end. It converges to a summed reward of -300 in comparison to a summed reward of -10000 for the constant model. In comparison to other reinforcement learning approaches the results are promising and the movement could be learned within 200 to 300 episodes, however, we can not keep up with state of the art model-based RL approaches [8] as it learns



(a) $N = 5$ (b) $N = 10$ (c) $N = 20$

Figure 9: Comparison of learning constant linear models against learning time-varying linear models with a different number of collected roll-outs $N = 5, 10, 20$. For a small number of roll-outs, the constant model works better as it can use much more data-points. However, the constant model can not capture the non-linearity in the system and therefore converges to a worse solution (-10000) than the solution found by the time-varying model (-300). This evaluation clearly shows that our approach can be used for data-efficient reinforcement learning.

time-independent GP models, which is more data-efficient. Yet, the performance of our approach could be drastically improved by using more sophisticated model learning methods, for example, learning an hierarchical prior which connects the models of the single time steps.

6 Conclusion and Future Work

In this paper, we presented a novel information theoretic stochastic optimal control algorithm. The key idea of our approach is that we want to stay close to the generated data such that we can ensure a stable learning progress. To our knowledge, this notion of closeness to the data is missing in all other stochastic optimal control algorithms. However, we believe it is a key ingredient for save approximation of the value function or the corresponding function $v_t(\boldsymbol{x})$, respectively. We show that our method can significantly outperform traditional approximate dynamic programming methods in terms of sample efficiency as well as quality of the final policy.

The information theoretic formulation provides several advantages over traditional approaches. We can get a closed-form solution for the estimated policy, at least on a finite set of samples. Furthermore, we can control the exploration of the policy in a principled manner without heuristics or fine-tuning. Moreover, we can use the roll-outs to learn simple local models which allow us to also use our algorithm for reinforcement learning. We use time-varying linear models to approximate the real system dynamics, however, the linear models are not computed at a single point of linearisation but on our current distribution of samples. Consequently, the estimated models are more robust then linear models obtained from linearisation.

The biggest disadvantage of our approach is the computation time. While methods based on linearisation can be computed in several seconds, and approximate dynamic programming methods need several minutes, our approach needs several hours to optimize the dual-function for 40 iterations. The main problem is the dimensionality of the dual function, as we have one $\boldsymbol{\theta}_t$ per time step and $\boldsymbol{\theta}_t$ can have easily up to 80 parameters. For future work, we will investigate dimensionality reduction techniques to reduce the dimensionality of the feature space. We will also investigate the combination of our locally linear policies with learned inverse dynamics controllers and evaluate our approach on a tendon-driven real robot arm.

References

[1] M. Azar, V. Gómez, and H. J. Kappen. Dynamic Policy Programming. *Journal of Machine Learning Research*, 13(Nov):3207–3245, 2012.

[2] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

[3] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro Dynamic Programming*. Athena Scientific, 1998.

[4] J. Boyan. Least-Squares Temporal Difference Learning. In *16th International Conference on Machine Learning*, pages 49–56, 1999.

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[6] C. Daniel, G. Neumann, and J. Peters. Hierarchical Relative Entropy Policy Search. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[7] C. Daniel, G. Neumann, and J. Peters. Learning Sequential Motor Tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013. submitted.

[8] M. Deisenroth and C. Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *28th International Conference on Machine Learning (ICML)*, 2011.

[9] D. Ernst, P. Geurts, and L. Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Resource*, 6:503–556, 2005.

[10] Geoffrey J. Gordon. Stable Function Approximation in Dynamic Programming. In *12th International Conference on Machine Learning (ICML)*, 1995.

[11] H. Kappen. An Introduction to Stochastic Control Theory, Path Integrals and Reinforcement Learning. In *Cooperative Behavior in Neural Systems*, volume 887, 2007.

[12] A. Kupcsik, M. Deisenroth, J. Peters, and G. Neumann. Data-Efficient Generalization of Robot Skills with Contextual Policy Search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2013.

[13] R Lioutikov, A. Paraschos, J. Peters, and G. Neumann. Sample-Based Information Theoretic Stochastic Optimal Control. subbmitted to ICRA 2014, see Attachment, 2013.

[14] The Mathworks. Matlab Optimization Toolbox User's Guide.

[15] J. Morimoto and C.Atkeson. Minimax differential dynamic programming: An application to robust bipedwalking. *Neural Information Processing Systems 2002*, 2002.

[16] G. Neumann and J. Peters. Fitted Q-Iteration by Advantage Weighted Regression. In *Neural Information Processing Systems (NIPS)*, 2009.

[17] J. Peters, K. Mülling, and Y. Altun. Relative Entropy Policy Search. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*, 2010.

[18] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[19] K. Rawlik, M. Toussaint, and S. Vijayakumar. On Stochastic Optimal Control and Reinforcement Learning by Approximate Inference. In *Proceedings of Robotics: Science and Systems*, 2012.

[20] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. Path Integral Control by Reproducing Kernel Hilbert Space Embedding. In *IJCAI*, 2013.

[21] FernSassower Sisser. Elimination of bounds in Optimization Problems by Transforming Variables. *Mathematical Programming*, 20(1):110–121, 1981.

[22] R. Stengel. *Stochastic Optimal Control: Theory and Application*. John Wiley & Sons, Inc., 1986.

[23] E. Theodorou, J. Buchli, and S. Schaal. Reinforcement Learning of Motor Skills in High Dimensions: a Path Integral Approach. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[24] E. Todorov and Weiwei L. A Generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems. In *24th American Control Conference (ACC)*, 2005.

[25] Emanuel Todorov. Linearly-solvable markov decision problems. In *NIPS*, pages 1369–1376, 2006.

[26] Emanuel Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478–11483, 2009.

[27] M. Toussaint. Robot Trajectory Optimization using Approximate Inference. In *26th International Conference on Machine Learning*, (ICML), 2009.

[28] A. Zimin and G. Neu. Online learning in episodic Markovian decision processes by relative entropy policy search. In *Neural Information Processing Systems (NIPS)*, 2013.

A Derivation of the Dual-Function

We start the derivation of the dual with the Lagrangian of the optimization problem. For the sake of clarity, we will treat all time steps the same and neglect the initial state distribution constraint as well as the final KL-bound. The Lagrangian is given as

$$L = \sum_{t=1}^{T} \iint p^t_{\boldsymbol{xu}} \left(r^t_{\boldsymbol{xu}} - \eta_t \log \frac{p^t_{\boldsymbol{xu}}}{q^t_{\boldsymbol{xu}}} + \mathbb{E}[\boldsymbol{\phi}^T_{\boldsymbol{x}'}]\boldsymbol{\theta}_{t+1} - \boldsymbol{\phi}^T_{\boldsymbol{x}}\boldsymbol{\theta}_t - \lambda_t \right) d\boldsymbol{x}\, d\boldsymbol{u} + \eta_t \epsilon + \lambda_t,$$

where $p^t_{\boldsymbol{xu}} = p^\pi_t(\boldsymbol{x}, \boldsymbol{u})$ and we chose a similar subscript notation for q_t and $\boldsymbol{\phi}$. Differentiating the Lagrangian w.r.t. $p^t_{\boldsymbol{xu}}$

$$\frac{\partial L}{\partial p^t_{\boldsymbol{xu}}} = r^t_{\boldsymbol{xu}} - \eta_t \left(\log \frac{p^t_{\boldsymbol{xu}}}{q^t_{\boldsymbol{xu}}} + 1 \right) + \mathbb{E}[\boldsymbol{\phi}^T_{\boldsymbol{x}'}]\boldsymbol{\theta}_{t+1} - \boldsymbol{\phi}^T_{\boldsymbol{x}}\boldsymbol{\theta}_t - \lambda_t$$

and setting the result to zero yields the closed form solution for $p^t_{\boldsymbol{xu}}$,

$$p^t_{\boldsymbol{xu}} = q^t_{\boldsymbol{xu}} \exp\left(\frac{r^t_{\boldsymbol{xu}} + \mathbb{E}[\boldsymbol{\phi}^T_{\boldsymbol{x}'}]\boldsymbol{\theta}_{t+1} - \boldsymbol{\phi}^T_{\boldsymbol{x}}\boldsymbol{\theta}_t}{\eta_t} \right) \exp\left(1 - \frac{\lambda_t}{\eta_t}. \right) \tag{32}$$

Out of our normalization constraint we follow that

$$\exp\left(1 - \frac{\lambda_t}{\eta_t} \right) = \left(\int\int q^t_{\boldsymbol{xu}} \exp\left(\frac{r^t_{\boldsymbol{xu}} + \mathbb{E}[\boldsymbol{\phi}^T_{\boldsymbol{x}'}]\boldsymbol{\theta}_{t+1} - \boldsymbol{\phi}^T_{\boldsymbol{x}}\boldsymbol{\theta}_t}{\eta_t} \right) d\boldsymbol{x}\, d\boldsymbol{u} \right)^{-1}. \tag{33}$$

Setting Equation (32) back into the Lagrangian, results after some transformations in the following equation

$$g(\eta_{1:H}, \lambda_{1:H}) = \sum_{t=1}^{T} \eta_t \epsilon - \eta_t + \lambda_t = \sum_{t=1}^{T} \eta_t \epsilon - \eta_t \log\left(\exp(1 - \lambda_t/\eta_t) \right).$$

We can now use Equation (33) to determine the dual function in terms of $\eta_{1:H}$ and $\boldsymbol{\theta}_{1:H}$,

$$g(\eta_{1:H}, \boldsymbol{\theta}_{1:H}) = \sum_{t=1}^{T} \eta_t \epsilon + \eta_t \log\left(\int q^t_{\boldsymbol{xu}} \exp\left(\frac{r^t_{\boldsymbol{xu}} + \mathbb{E}[\boldsymbol{\phi}^T_{\boldsymbol{x}'}]\boldsymbol{\theta}_{t+1} - \boldsymbol{\phi}^T_{\boldsymbol{x}}\boldsymbol{\theta}_t}{\eta_t} \right) d\boldsymbol{x}\, d\boldsymbol{u} \right). \tag{34}$$

Including the KL-bound of the last time step as well as the initial state constraint, the dual function is given as

$$\begin{aligned} g(\eta_{1:H}, \boldsymbol{\theta}_{1:H}; \mathscr{D}) &= \sum_{t=1}^{H-1} \eta_t \log\left(\frac{1}{N} \sum_{\boldsymbol{x},\boldsymbol{u} \in \mathscr{D}_t} \exp\left(\frac{r^t_{\boldsymbol{xu}} + \mathbb{E}[\boldsymbol{\phi}^T_{\boldsymbol{x}'}]\boldsymbol{\theta}_{t+1} - \boldsymbol{\phi}^T_{\boldsymbol{x}}\boldsymbol{\theta}_t}{\eta_t} \right) \right) \\ &\quad + \sum_{t=1}^{T} \eta_t \epsilon + \eta_H \log\left(\frac{1}{N} \sum_{\boldsymbol{x} \in \mathscr{D}_H} \exp\left(\frac{r^T_{\boldsymbol{x}} - \boldsymbol{\phi}^T_{\boldsymbol{x}}\boldsymbol{\theta}_H}{\eta_H} \right) \right) + \hat{\boldsymbol{\phi}}^T_1 \boldsymbol{\theta}_1, \end{aligned} \tag{35}$$

where we replaced the distribution q_t by $1/N$ as we now use samples $\mathscr{D}$ which have been generated from q_t. The set $\mathscr{D}_t$ denotes all samples for time step t.

B Derivation for the closed-form solution of the dual

In this section we give an in-depth derivation for the closed-form solution of the dual. For the sake of readability we change the format to landscape.

1 Constraint Optimization Problem

Starting with the Constraint Optimization Problem.

$$\arg \max_{\mu^{\pi}(\boldsymbol{x}),\pi(\boldsymbol{u}|\boldsymbol{x})} J(\boldsymbol{x},\boldsymbol{u}) = \arg \max_{\mu^{\pi}(\boldsymbol{x}),\pi(\boldsymbol{u}|\boldsymbol{x})} \sum_{k=1}^{K-1} \left[\int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x}) \, r_k(\boldsymbol{x},\boldsymbol{u}) \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \right] + \int_{\boldsymbol{x}} \mu_K^{\pi}(\boldsymbol{x}) \, r_K(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \tag{1}$$

$$\text{s.t. } \forall k < K : \int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x}) \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} = 1 \, , \qquad k = K : \int_{\boldsymbol{x}} \mu_T^{\pi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = 1 \, , \tag{2}$$

$$\forall k > 1 : \int_{\boldsymbol{x}} \mu_k^{\pi}(\boldsymbol{x}) \, \boldsymbol{\phi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int_{\boldsymbol{x}'} \int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_{k-1}^{\pi}(\boldsymbol{x}) \, \pi_{k-1}(\boldsymbol{u}|\boldsymbol{x}) \, p_{k-1}(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}) \, \boldsymbol{\phi}(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{x}' \, , \qquad k = 1 \, : \int_{\boldsymbol{x}} \mu_1^{\pi}(\boldsymbol{x}) \, \boldsymbol{\phi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int_{\boldsymbol{x}} p_1(\boldsymbol{x}) \, \boldsymbol{\phi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \, , \tag{3}$$

$$\forall k < K : \int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x}) \log \frac{\mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x})}{q_k(\boldsymbol{x},\boldsymbol{u})} \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \leq \epsilon \, , \qquad k = K : \int_{\boldsymbol{x}} \mu_K^{\pi}(\boldsymbol{x}) \log \frac{\mu_K^{\pi}(\boldsymbol{x})}{q_K(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x} \leq \epsilon \, . \tag{4}$$

Where (1) is the objective function.
(2) are the normalization constraints.
(3) are the state constraints.
(4) are the contraints to ensure the KL bound between two successive distributions.

2 Lagrangian

Transformed into the Lagrangian

$$\begin{aligned}
\mathcal{L}(\mu^{\pi}(\boldsymbol{x}), \pi(\boldsymbol{u}|\boldsymbol{x}), \lambda_{1:K}, \boldsymbol{\theta}_{1:K}, \eta_{1:K}) = & \sum_{k=1}^{K-1} \left[\int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x}) \, r_k(\boldsymbol{x},\boldsymbol{u}) \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \right] + \int_{\boldsymbol{x}} \mu_K^{\pi}(\boldsymbol{x}) \, r_K(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\
& + \sum_{k=1}^{K-1} \left[\lambda_k - \lambda_k \int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x}) \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \right] + \lambda_K - \lambda_K \int_{\boldsymbol{x}} \mu_T^{\pi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\
& + \sum_{k=2}^{K} \left[\boldsymbol{\theta}_k \int_{\boldsymbol{x}'} \int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_{k-1}^{\pi}(\boldsymbol{x}) \, \pi_{k-1}(\boldsymbol{u}|\boldsymbol{x}) \, p_{k-1}(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}) \, \boldsymbol{\phi}(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{x}' - \boldsymbol{\theta}_k \int_{\boldsymbol{x}} \mu_k^{\pi}(\boldsymbol{x}) \, \boldsymbol{\phi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \right] + \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1(\boldsymbol{x}) \, \boldsymbol{\phi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} - \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} \mu_1^{\pi}(\boldsymbol{x}) \, \boldsymbol{\phi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\
& + \sum_{k=1}^{K-1} \left[\eta_k \epsilon - \eta_k \int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x}) \log \frac{\mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x})}{q_k(\boldsymbol{x},\boldsymbol{u})} \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \right] + \eta_K \epsilon - \eta_K \int_{\boldsymbol{x}} \mu_K^{\pi}(\boldsymbol{x}) \log \frac{\mu_K^{\pi}(\boldsymbol{x})}{q_K(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x}
\end{aligned}$$

$$\begin{aligned}
= & \sum_{k=1}^{K-1} \left[\int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x}) \, r_k(\boldsymbol{x},\boldsymbol{u}) \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \right] + \int_{\boldsymbol{x}} \mu_K^{\pi}(\boldsymbol{x}) \, r_K(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\
& + \sum_{k=1}^{K-1} \left[-\lambda_k \int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x}) \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \right] + \sum_{k=1}^{K} [\lambda_k] - \lambda_K \int_{\boldsymbol{x}} \mu_T^{\pi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\
& + \sum_{k=1}^{K-1} \left[\boldsymbol{\theta}_{k+1} \int_{\boldsymbol{x}'} \int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x}) \, p_k(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}) \, \boldsymbol{\phi}(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{x}' - \boldsymbol{\theta}_k \int_{\boldsymbol{x}} \mu_k^{\pi}(\boldsymbol{x}) \, \boldsymbol{\phi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \right] - \boldsymbol{\theta}_K \int_{\boldsymbol{x}} \mu_K^{\pi}(\boldsymbol{x}) \, \boldsymbol{\phi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} + \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1(\boldsymbol{x}) \, \boldsymbol{\phi}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\
& + \sum_{k=1}^{K-1} \left[-\eta_k \int_{\boldsymbol{x}} \int_{\boldsymbol{u}} \mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x}) \log \frac{\mu_k^{\pi}(\boldsymbol{x}) \, \pi_k(\boldsymbol{u}|\boldsymbol{x})}{q_k(\boldsymbol{x},\boldsymbol{u})} \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \right] + \sum_{k=1}^{K} [\eta_k \epsilon] - \eta_K \int_{\boldsymbol{x}} \mu_K^{\pi}(\boldsymbol{x}) \log \frac{\mu_K^{\pi}(\boldsymbol{x})}{q_K(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x}
\end{aligned}$$

Combining the sums

$$\mathcal{L}\left(\mu^{\pi}\left(\boldsymbol{x}\right),\pi\left(\boldsymbol{u}|\boldsymbol{x}\right),\lambda_{1:K},\boldsymbol{\theta}_{1:K},\eta_{1:K}\right)=\sum_{k=1}^{K-1}\left[\int_{\boldsymbol{x}}\int_{\boldsymbol{u}}\mu_{k}^{\pi}\left(\boldsymbol{x}\right)\pi_{k}\left(\boldsymbol{u}|\boldsymbol{x}\right)\left(r_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)-\lambda_{k}+\boldsymbol{\theta}_{k+1}\int_{\boldsymbol{x}'}p_{k}\left(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}\right)\boldsymbol{\phi}\left(\boldsymbol{x}'\right)\,\mathrm{d}\boldsymbol{x}'-\boldsymbol{\theta}_{k}\boldsymbol{\phi}\left(\boldsymbol{x}\right)-\eta_{k}\log\frac{\mu_{k}^{\pi}\left(\boldsymbol{x}\right)\pi_{k}\left(\boldsymbol{u}|\boldsymbol{x}\right)}{q_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)}\right)\,\mathrm{d}\boldsymbol{u}\,\mathrm{d}\boldsymbol{x}\right]$$
$$+\boldsymbol{\theta}_{1}\int_{\boldsymbol{x}}p_{1}\left(\boldsymbol{x}\right)\boldsymbol{\phi}\left(\boldsymbol{x}\right)\,\mathrm{d}\boldsymbol{x}+\sum_{k=1}^{K}\left[\lambda_{k}+\eta_{k}\epsilon\right]+\int_{\boldsymbol{x}}\mu_{K}^{\pi}\left(\boldsymbol{x}\right)\left(r_{K}\left(\boldsymbol{x}\right)-\lambda_{K}-\boldsymbol{\theta}_{K}\boldsymbol{\phi}\left(\boldsymbol{x}\right)-\eta_{K}\log\frac{\mu_{K}^{\pi}\left(\boldsymbol{x}\right)}{q_{K}\left(\boldsymbol{x}\right)}\right)\,\mathrm{d}\boldsymbol{x} \tag{5}$$

3 Derivatives

Taking the first derivative of $\mathcal{L}\left(\mu^{\pi}\left(\boldsymbol{x}\right),\pi\left(\boldsymbol{u}|\boldsymbol{x}\right),\lambda_{1:K},\boldsymbol{\theta}_{1:K},\eta_{1:K}\right)$ with respect to $\mu_{k}^{\pi}\left(\boldsymbol{x}\right)\pi_{k}\left(\boldsymbol{u}|\boldsymbol{x}\right)$,

$$\frac{\partial\mathcal{L}\left(\mu^{\pi}\left(\boldsymbol{x}\right),\pi\left(\boldsymbol{u}|\boldsymbol{x}\right),\lambda_{1:K},\boldsymbol{\theta}_{1:K},\eta_{1:K}\right)}{\partial\left(\mu_{k}^{\pi}\left(\boldsymbol{x}\right)\pi_{k}\left(\boldsymbol{u}|\boldsymbol{x}\right)\right)}=r_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)-\lambda_{k}+\boldsymbol{\theta}_{k+1}\int_{\boldsymbol{x}'}p_{k}\left(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}\right)\boldsymbol{\phi}\left(\boldsymbol{x}'\right)\,\mathrm{d}\boldsymbol{x}'-\boldsymbol{\theta}_{k}\boldsymbol{\phi}\left(\boldsymbol{x}\right)-\eta_{k}\log\frac{\mu_{k}^{\pi}\left(\boldsymbol{x}\right)\pi_{k}\left(\boldsymbol{u}|\boldsymbol{x}\right)}{q_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)}-\eta_{k}$$

and setting it to zero.

$$\begin{aligned}
0&=\frac{\partial\mathcal{L}\left(\mu^{\pi}\left(\boldsymbol{x}\right),\pi\left(\boldsymbol{u}|\boldsymbol{x}\right),\lambda_{1:K},\boldsymbol{\theta}_{1:K},\eta_{1:K}\right)}{\partial\left(\mu_{k}^{\pi}\left(\boldsymbol{x}\right)\pi_{k}\left(\boldsymbol{u}|\boldsymbol{x}\right)\right)}\\
0&=r_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)-\lambda_{k}+\boldsymbol{\theta}_{k+1}\int_{\boldsymbol{x}'}p_{k}\left(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}\right)\boldsymbol{\phi}\left(\boldsymbol{x}'\right)\,\mathrm{d}\boldsymbol{x}'-\boldsymbol{\theta}_{k}\boldsymbol{\phi}\left(\boldsymbol{x}\right)-\eta_{k}\log\frac{\mu_{k}^{\pi}\left(\boldsymbol{x}\right)\pi_{k}\left(\boldsymbol{u}|\boldsymbol{x}\right)}{q_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)}-\eta_{k}\\
\log\frac{\mu_{k}^{\pi}\left(\boldsymbol{x}\right)\pi_{k}\left(\boldsymbol{u}|\boldsymbol{x}\right)}{q_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)}&=\frac{r_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)+\boldsymbol{\theta}_{k+1}\int_{\boldsymbol{x}'}p_{k}\left(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}\right)\boldsymbol{\phi}\left(\boldsymbol{x}'\right)\,\mathrm{d}\boldsymbol{x}'-\boldsymbol{\theta}_{k}\boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_{k}}-\frac{\lambda_{k}}{\eta_{k}}-1\\
\mu_{k}^{\pi}\left(\boldsymbol{x}\right)\pi_{k}\left(\boldsymbol{u}|\boldsymbol{x}\right)&=q_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)\exp\left(\frac{r_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)+\boldsymbol{\theta}_{k+1}\int_{\boldsymbol{x}'}p_{k}\left(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}\right)\boldsymbol{\phi}\left(\boldsymbol{x}'\right)\,\mathrm{d}\boldsymbol{x}'-\boldsymbol{\theta}_{k}\boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_{k}}\right)\exp\left(-\frac{\lambda_{k}}{\eta_{k}}-1\right)
\end{aligned} \tag{6}$$

From the normalization constraint we know

$$\exp\left(\frac{\lambda_{k}}{\eta_{k}}+1\right)=\int_{\boldsymbol{x}}\int_{\boldsymbol{u}}q_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)\exp\left(\frac{r_{k}\left(\boldsymbol{x},\boldsymbol{u}\right)+\boldsymbol{\theta}_{k+1}\int_{\boldsymbol{x}'}p_{k}\left(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}\right)\boldsymbol{\phi}\left(\boldsymbol{x}'\right)\,\mathrm{d}\boldsymbol{x}'-\boldsymbol{\theta}_{k}\boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_{k}}\right)\,\mathrm{d}\boldsymbol{u}\,\mathrm{d}\boldsymbol{x} \tag{7}$$

Since there are not action at the last time step, we also take the derivative of $\mathcal{L}\left(\mu^{\pi}\left(\boldsymbol{x}\right),\pi\left(\boldsymbol{u}|\boldsymbol{x}\right),\lambda_{1:K},\boldsymbol{\theta}_{1:K},\eta_{1:K}\right)$ with respect to $\mu_{K}^{\pi}\left(\boldsymbol{x}\right)$,

$$\frac{\partial\mathcal{L}\left(\mu^{\pi}\left(\boldsymbol{x}\right),\pi\left(\boldsymbol{u}|\boldsymbol{x}\right),\lambda_{1:K},\boldsymbol{\theta}_{1:K},\eta_{1:K}\right)}{\partial\mu_{K}^{\pi}\left(\boldsymbol{x}\right)}=r_{K}\left(\boldsymbol{x}\right)-\lambda_{K}-\boldsymbol{\theta}_{K}\boldsymbol{\phi}\left(\boldsymbol{x}\right)-\eta_{K}\log\frac{\mu_{K}^{\pi}\left(\boldsymbol{x}\right)}{q_{K}\left(\boldsymbol{x}\right)}-\eta_{K}$$

and set it to zero.

$$\begin{aligned}
0&=\frac{\partial\mathcal{L}\left(\mu^{\pi}\left(\boldsymbol{x}\right),\pi\left(\boldsymbol{u}|\boldsymbol{x}\right),\lambda_{1:K},\boldsymbol{\theta}_{1:K},\eta_{1:K}\right)}{\partial\mu_{K}^{\pi}\left(\boldsymbol{x}\right)}\\
0&=r_{K}\left(\boldsymbol{x}\right)-\lambda_{K}-\boldsymbol{\theta}_{K}\boldsymbol{\phi}\left(\boldsymbol{x}\right)-\eta_{K}\log\frac{\mu_{K}^{\pi}\left(\boldsymbol{x}\right)}{q_{K}\left(\boldsymbol{x}\right)}-\eta_{K}\\
\log\frac{\mu_{K}^{\pi}\left(\boldsymbol{x}\right)}{q_{K}\left(\boldsymbol{x}\right)}&=\frac{r_{K}\left(\boldsymbol{x}\right)-\boldsymbol{\theta}_{K}\boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_{K}}-\frac{\lambda_{K}}{\eta_{K}}-1\\
\mu_{K}^{\pi}\left(\boldsymbol{x}\right)&=q_{K}\left(\boldsymbol{x}\right)\exp\left(\frac{r_{K}\left(\boldsymbol{x}\right)-\boldsymbol{\theta}_{K}\boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_{K}}\right)\exp\left(-\frac{\lambda_{K}}{\eta_{K}}-1\right)
\end{aligned} \tag{8}$$

From the normalization constraint we know

$$\exp\left(\frac{\lambda_{K}}{\eta_{K}}+1\right)=\int_{\boldsymbol{x}}q_{K}\left(\boldsymbol{x}\right)\exp\left(\frac{r_{K}\left(\boldsymbol{x}\right)-\boldsymbol{\theta}_{K}\boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_{K}}\right)\,\mathrm{d}\boldsymbol{x} \tag{9}$$

4 Dual

Inserting (6) and (8) into (5) results in the following dual

$$
\begin{aligned}
g\left(\lambda_{1:K}, \boldsymbol{\theta}_1, \eta_{1:K}\right) = \sum_{k=1}^{K-1} \Bigg[& \int_{\boldsymbol{x}} \int_{\boldsymbol{u}} q_k\left(\boldsymbol{x}, \boldsymbol{u}\right) \exp\left(\frac{r_k\left(\boldsymbol{x}, \boldsymbol{u}\right) + \boldsymbol{\theta}_{k+1} \int_{\boldsymbol{x}'} p_k\left(\boldsymbol{x}' | \boldsymbol{x}, \boldsymbol{u}\right) \boldsymbol{\phi}\left(\boldsymbol{x}'\right) \, \mathrm{d}\boldsymbol{x}' - \boldsymbol{\theta}_k \boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_k}\right) \exp\left(-\frac{\lambda_k}{\eta_k} - 1\right) \\
& \left(r_k\left(\boldsymbol{x}, \boldsymbol{u}\right) - \lambda_k + \boldsymbol{\theta}_{k+1} \int_{\boldsymbol{x}'} p_k\left(\boldsymbol{x}' | \boldsymbol{x}, \boldsymbol{u}\right) \boldsymbol{\phi}\left(\boldsymbol{x}'\right) \, \mathrm{d}\boldsymbol{x}' - \boldsymbol{\theta}_k \boldsymbol{\phi}\left(\boldsymbol{x}\right) - \eta_k \log \frac{q_k\left(\boldsymbol{x}, \boldsymbol{u}\right) \exp\left(\frac{r_k\left(\boldsymbol{x}, \boldsymbol{u}\right) + \boldsymbol{\theta}_{k+1} \int_{\boldsymbol{x}'} p_k\left(\boldsymbol{x}' | \boldsymbol{x}, \boldsymbol{u}\right) \boldsymbol{\phi}\left(\boldsymbol{x}'\right) \, \mathrm{d}\boldsymbol{x}' - \boldsymbol{\theta}_k \boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_k}\right) \exp\left(-\frac{\lambda_k}{\eta_k} - 1\right)}{q_k\left(\boldsymbol{x}, \boldsymbol{u}\right)} \right) \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \Bigg] \\
& + \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1\left(\boldsymbol{x}\right) \boldsymbol{\phi}\left(\boldsymbol{x}\right) \, \mathrm{d}\boldsymbol{x} + \sum_{k=1}^{K} \left[\lambda_k + \eta_k \epsilon\right] + \int_{\boldsymbol{x}} q_K\left(\boldsymbol{x}\right) \exp\left(\frac{r_K\left(\boldsymbol{x}\right) - \boldsymbol{\theta}_K \boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_K}\right) \exp\left(-\frac{\lambda_K}{\eta_K} - 1\right) \left(r_K\left(\boldsymbol{x}\right) - \lambda_K - \boldsymbol{\theta}_K \boldsymbol{\phi}\left(\boldsymbol{x}\right) - \eta_K \log \frac{q_K\left(\boldsymbol{x}\right) \exp\left(\frac{r_K\left(\boldsymbol{x}\right) - \boldsymbol{\theta}_K \boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_K}\right) \exp\left(-\frac{\lambda_K}{\eta_K} - 1\right)}{q_K\left(\boldsymbol{x}\right)} \right) \mathrm{d}
\end{aligned}
$$

$$
\begin{aligned}
= & \sum_{k=1}^{K-1} \Bigg[\int_{\boldsymbol{x}} \int_{\boldsymbol{u}} q_k\left(\boldsymbol{x}, \boldsymbol{u}\right) \exp\left(\frac{r_k\left(\boldsymbol{x}, \boldsymbol{u}\right) + \boldsymbol{\theta}_{k+1} \int_{\boldsymbol{x}'} p_k\left(\boldsymbol{x}' | \boldsymbol{x}, \boldsymbol{u}\right) \boldsymbol{\phi}\left(\boldsymbol{x}'\right) \, \mathrm{d}\boldsymbol{x}' - \boldsymbol{\theta}_k \boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_k}\right) \exp\left(-\frac{\lambda_k}{\eta_k} - 1\right) \\
& \left(r_k\left(\boldsymbol{x}, \boldsymbol{u}\right) - \lambda_k + \boldsymbol{\theta}_{k+1} \int_{\boldsymbol{x}'} p_k\left(\boldsymbol{x}' | \boldsymbol{x}, \boldsymbol{u}\right) \boldsymbol{\phi}\left(\boldsymbol{x}'\right) \, \mathrm{d}\boldsymbol{x}' - \boldsymbol{\theta}_k \boldsymbol{\phi}\left(\boldsymbol{x}\right) - r_k\left(\boldsymbol{x}, \boldsymbol{u}\right) - \boldsymbol{\theta}_{k+1} \int_{\boldsymbol{x}'} p_k\left(\boldsymbol{x}' | \boldsymbol{x}, \boldsymbol{u}\right) \boldsymbol{\phi}\left(\boldsymbol{x}'\right) \, \mathrm{d}\boldsymbol{x}' + \boldsymbol{\theta}_k \boldsymbol{\phi}\left(\boldsymbol{x}\right) + \lambda_k + \eta_k \right) \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \Bigg] \\
& + \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1\left(\boldsymbol{x}\right) \boldsymbol{\phi}\left(\boldsymbol{x}\right) \, \mathrm{d}\boldsymbol{x} + \sum_{k=1}^{K} \left[\lambda_k + \eta_k \epsilon\right] + \int_{\boldsymbol{x}} q_K\left(\boldsymbol{x}\right) \exp\left(\frac{r_K\left(\boldsymbol{x}\right) - \boldsymbol{\theta}_K \boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_K}\right) \exp\left(-\frac{\lambda_K}{\eta_K} - 1\right) \left(r_K\left(\boldsymbol{x}\right) - \lambda_K - \boldsymbol{\theta}_K \boldsymbol{\phi}\left(\boldsymbol{x}\right) - r_K\left(\boldsymbol{x}\right) + \boldsymbol{\theta}_K \boldsymbol{\phi}\left(\boldsymbol{x}\right) + \lambda_K + \eta_K \right) \, \mathrm{d}\boldsymbol{x}
\end{aligned}
$$

$$
\begin{aligned}
= & \sum_{k=1}^{K-1} \left[\int_{\boldsymbol{x}} \int_{\boldsymbol{u}} q_k\left(\boldsymbol{x}, \boldsymbol{u}\right) \exp\left(\frac{r_k\left(\boldsymbol{x}, \boldsymbol{u}\right) + \boldsymbol{\theta}_{k+1} \int_{\boldsymbol{x}'} p_k\left(\boldsymbol{x}' | \boldsymbol{x}, \boldsymbol{u}\right) \boldsymbol{\phi}\left(\boldsymbol{x}'\right) \, \mathrm{d}\boldsymbol{x}' - \boldsymbol{\theta}_k \boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_k}\right) \exp\left(-\frac{\lambda_k}{\eta_k} - 1\right) \eta_k \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x} \right] \\
& + \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1\left(\boldsymbol{x}\right) \boldsymbol{\phi}\left(\boldsymbol{x}\right) \, \mathrm{d}\boldsymbol{x} + \sum_{k=1}^{K} \left[\lambda_k + \eta_k \epsilon\right] + \int_{\boldsymbol{x}} q_K\left(\boldsymbol{x}\right) \exp\left(\frac{r_K\left(\boldsymbol{x}\right) - \boldsymbol{\theta}_K \boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_K}\right) \exp\left(-\frac{\lambda_K}{\eta_K} - 1\right) \eta_K \, \mathrm{d}\boldsymbol{x} \\
= & \sum_{k=1}^{K-1} \left[\eta_k\right] + \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1\left(\boldsymbol{x}\right) \boldsymbol{\phi}\left(\boldsymbol{x}\right) \, \mathrm{d}\boldsymbol{x} + \sum_{k=1}^{K} \left[\lambda_k + \eta_k \epsilon\right] + \eta_K \\
= & \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1\left(\boldsymbol{x}\right) \boldsymbol{\phi}\left(\boldsymbol{x}\right) \, \mathrm{d}\boldsymbol{x} + \sum_{k=1}^{K} \left[\lambda_k + \eta_k \epsilon + \eta_k\right] \qquad (10)
\end{aligned}
$$

Inserting (7) and (9) into (10) leads to a more convenient representation of the dual

$$
\begin{aligned}
g\left(\boldsymbol{\theta}_{1:K}, \eta_{1:K}\right) &= \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1\left(\boldsymbol{x}\right) \boldsymbol{\phi}\left(\boldsymbol{x}\right) \, \mathrm{d}\boldsymbol{x} + \sum_{k=1}^{K} \left[\lambda_k + \eta_k \epsilon + \eta_k\right] \\
&= \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1\left(\boldsymbol{x}\right) \boldsymbol{\phi}\left(\boldsymbol{x}\right) \, \mathrm{d}\boldsymbol{x} + \sum_{k=1}^{K} \left[\eta_k \epsilon + \eta_k \left(\frac{\lambda_k}{\eta_k} + 1\right)\right] \\
&= \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1\left(\boldsymbol{x}\right) \boldsymbol{\phi}\left(\boldsymbol{x}\right) \, \mathrm{d}\boldsymbol{x} + \sum_{k=1}^{K} \left[\eta_k \epsilon\right] + \sum_{k=1}^{K-1} \left[\eta_k \log \exp\left(\frac{\lambda_k}{\eta_k} + 1\right)\right] + \eta_K \log \exp\left(\frac{\lambda_K}{\eta_K} + 1\right) \\
&= \boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1\left(\boldsymbol{x}\right) \boldsymbol{\phi}\left(\boldsymbol{x}\right) \, \mathrm{d}\boldsymbol{x} + \sum_{k=1}^{K} \left[\eta_k \epsilon\right] + \sum_{k=1}^{K-1} \left[\eta_k \log \int_{\boldsymbol{x}} \int_{\boldsymbol{u}} q_k\left(\boldsymbol{x}, \boldsymbol{u}\right) \exp\left(\frac{r_k\left(\boldsymbol{x}, \boldsymbol{u}\right) + \boldsymbol{\theta}_{k+1} \int_{\boldsymbol{x}'} p_k\left(\boldsymbol{x}' | \boldsymbol{x}, \boldsymbol{u}\right) \boldsymbol{\phi}\left(\boldsymbol{x}'\right) \, \mathrm{d}\boldsymbol{x}' - \boldsymbol{\theta}_k \boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_k}\right) \, \mathrm{d}\boldsymbol{u} \, \mathrm{d}\boldsymbol{x}\right] + \eta_K \log \int_{\boldsymbol{x}} q_K\left(\boldsymbol{x}\right) \exp\left(\frac{r_K\left(\boldsymbol{x}\right) - \boldsymbol{\theta}_K \boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_K}\right) \, \mathrm{d}\boldsymbol{x}
\end{aligned}
\tag{11}
$$

5 Closed Form

In order to find a closed form solution we make the following assumptions:
All random variables are gaussian distributed.

$$
\begin{aligned}
&\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{x} | \boldsymbol{\mu}_{\boldsymbol{x},k}, \boldsymbol{\Sigma}_{\boldsymbol{x},k}\right) \\
&\boldsymbol{u} \sim \mathcal{N}\left(\boldsymbol{u} | \boldsymbol{\mu}_{\boldsymbol{u},k}, \boldsymbol{\Sigma}_{\boldsymbol{u},k}\right) \\
&\boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{y} | \boldsymbol{\mu}_{\boldsymbol{y},k}, \boldsymbol{\Sigma}_{\boldsymbol{y},k}\right), \quad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{u} \end{bmatrix}, \; \boldsymbol{\mu}_{\boldsymbol{y},k} = \begin{bmatrix} \boldsymbol{\mu}_{\boldsymbol{x},k} \\ \boldsymbol{\mu}_{\boldsymbol{u},k} \end{bmatrix}, \; \boldsymbol{\Sigma}_{\boldsymbol{y},k} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{x},k} & \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{u},k}^T \\ \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{u},k} & \boldsymbol{\Sigma}_{\boldsymbol{u},k} \end{bmatrix}
\end{aligned}
$$

The state-action distributions are Gaussian distributions. For later convenience we use the canonical representation.

$$
q_k\left(\boldsymbol{x}, \boldsymbol{u}\right) = q_k\left(\boldsymbol{y}\right) = \mathcal{N}\left(\boldsymbol{y} | \boldsymbol{\mu}_{\boldsymbol{y},k}, \boldsymbol{\Sigma}_{\boldsymbol{y},k}\right) = \frac{\exp\left(-\frac{1}{2} \boldsymbol{\mu}_{\boldsymbol{y},k}^T \boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} \boldsymbol{\mu}_{\boldsymbol{y},k}\right)}{\left|2\pi \boldsymbol{\Sigma}_{\boldsymbol{y},k}\right|^{1/2}} \exp\left(-\frac{1}{2} \boldsymbol{y}^T \boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} \boldsymbol{\mu}_{\boldsymbol{y},k}\right) \tag{12}
$$

$$
q_K\left(\boldsymbol{x}\right) = \mathcal{N}\left(\boldsymbol{x} | \boldsymbol{\mu}_{\boldsymbol{x},K}, \boldsymbol{\Sigma}_{\boldsymbol{x},K}\right) = \frac{\exp\left(-\frac{1}{2} \boldsymbol{\mu}_{\boldsymbol{x},K}^T \boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} \boldsymbol{\mu}_{\boldsymbol{x},K}\right)}{\left|2\pi \boldsymbol{\Sigma}_{\boldsymbol{x},K}\right|^{1/2}} \exp\left(-\frac{1}{2} \boldsymbol{x}^T \boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} \boldsymbol{\mu}_{\boldsymbol{x},K}\right) \tag{13}
$$

The reward function is quadratic and has the follwoing form.

$$
r_k\left(\boldsymbol{y}\right) = -\frac{1}{2} \boldsymbol{y}^T \boldsymbol{R}_k \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{r}_k \tag{14}
$$

$$
r_K\left(\boldsymbol{x}\right) = -\frac{1}{2} \boldsymbol{x}^T \boldsymbol{R}_K \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{r}_K \tag{15}
$$

The state transition is approximated by linear models.

$$
p_k\left(\boldsymbol{x}' | \boldsymbol{x}, \boldsymbol{u}\right) = p_k\left(\boldsymbol{x}' | \boldsymbol{y}\right) = \mathcal{N}\left(\boldsymbol{x}' | \boldsymbol{\mu}_{\boldsymbol{x}'|\boldsymbol{y},k}, \boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k}\right), \quad \boldsymbol{\mu}_{\boldsymbol{x}'|\boldsymbol{y},k} = \boldsymbol{A}_k \boldsymbol{y} + \boldsymbol{a}_k
$$

The state constraint results in moment matching. For later convenience we choose the following form.

$$\boldsymbol{\theta}_1 \int_{\boldsymbol{x}} p_1(\boldsymbol{x})\,\boldsymbol{\phi}(\boldsymbol{x})\ \mathrm{d}\boldsymbol{x} = -\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{V}_1 \boldsymbol{\mu}_{\boldsymbol{x},1} + \boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{v}_1 - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_1 \boldsymbol{\Sigma}_{\boldsymbol{x},1}\right) \tag{16}$$

$$\begin{aligned}
\boldsymbol{\theta}_{k+1} \int_{\boldsymbol{x}'} p_k\left(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}\right)\boldsymbol{\phi}\left(\boldsymbol{x}'\right)\ \mathrm{d}\boldsymbol{x}' &= -\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x}'|\boldsymbol{y},k}^T \boldsymbol{V}_{k+1}\boldsymbol{\mu}_{\boldsymbol{x}'|\boldsymbol{y},k} + \boldsymbol{\mu}_{\boldsymbol{x}'|\boldsymbol{y},k}^T \boldsymbol{v}_{k+1} - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_{k+1}\boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k}\right) \\
&= -\frac{1}{2}\left(\boldsymbol{A}_k\boldsymbol{y}+\boldsymbol{a}_k\right)^T \boldsymbol{V}_{k+1}\left(\boldsymbol{A}_k\boldsymbol{y}+\boldsymbol{a}_k\right) + \left(\boldsymbol{A}_k\boldsymbol{y}+\boldsymbol{a}_k\right)^T \boldsymbol{v}_{k+1} - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_{k+1}\boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k}\right) \\
&= -\frac{1}{2}\left(\boldsymbol{y}^T\boldsymbol{A}_k^T\boldsymbol{V}_{k+1}\boldsymbol{A}_k\boldsymbol{y} + 2\boldsymbol{y}^T\boldsymbol{A}_k^T\boldsymbol{V}_{k+1}\boldsymbol{a}_k + \boldsymbol{a}_k^T\boldsymbol{V}_{k+1}\boldsymbol{a}_k\right) + \boldsymbol{y}^T\boldsymbol{A}_k^T\boldsymbol{v}_{k+1} + \boldsymbol{a}_k^T\boldsymbol{v}_{k+1} - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_{k+1}\boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k}\right) \\
&= -\frac{1}{2}\boldsymbol{y}^T\boldsymbol{A}_k^T\boldsymbol{V}_{k+1}\boldsymbol{A}_k\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{A}_k^T\boldsymbol{V}_{k+1}\boldsymbol{a}_k - \frac{1}{2}\boldsymbol{a}_k^T\boldsymbol{V}_{k+1}\boldsymbol{a}_k + \boldsymbol{y}^T\boldsymbol{A}_k^T\boldsymbol{v}_{k+1} + \boldsymbol{a}_k^T\boldsymbol{v}_{k+1} - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_{k+1}\boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k}\right) \\
&= -\frac{1}{2}\boldsymbol{y}^T\boldsymbol{A}_k^T\boldsymbol{V}_{k+1}\boldsymbol{A}_k\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{A}_k^T\left(\boldsymbol{v}_{k+1} - \boldsymbol{V}_{k+1}\boldsymbol{a}_k\right) + \boldsymbol{a}_k^T\left(\boldsymbol{v}_{k+1} - \frac{1}{2}\boldsymbol{V}_{k+1}\boldsymbol{a}_k\right) - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_{k+1}\boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k}\right)
\end{aligned} \tag{17}$$

Again for later convenience, we define an extension to $\boldsymbol{V}_k, \boldsymbol{v}_k$ such that they have the same dimensionalities as $\boldsymbol{R}_k, \boldsymbol{r}_k$

$$\boldsymbol{\theta}_k\boldsymbol{\phi}(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{x}^T\boldsymbol{V}_k\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{v}_k = -\frac{1}{2}\boldsymbol{y}^T\hat{\boldsymbol{V}}_k\boldsymbol{y} + \boldsymbol{y}^T\hat{\boldsymbol{v}}_k, \quad \hat{\boldsymbol{V}}_k = \begin{bmatrix}\boldsymbol{V}_k & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}, \quad \hat{\boldsymbol{v}}_k = \begin{bmatrix}\boldsymbol{v}_k \\ \boldsymbol{0}\end{bmatrix} \tag{18}$$

We know take the dual (11)

$$g\left(\boldsymbol{\theta}_{1:K},\eta_{1:K}\right)=\boldsymbol{\theta}_1\int_{\boldsymbol{x}}p_1\left(\boldsymbol{x}\right)\boldsymbol{\phi}\left(\boldsymbol{x}\right)\ \mathrm{d}\boldsymbol{x}+\sum_{k=1}^{K}\left[\eta_k\epsilon\right]+\sum_{k=1}^{K-1}\left[\eta_k\log\int_{\boldsymbol{x}}\int_{\boldsymbol{u}}q_k\left(\boldsymbol{x},\boldsymbol{u}\right)\exp\left(\frac{r_k\left(\boldsymbol{x},\boldsymbol{u}\right)+\boldsymbol{\theta}_{k+1}\int_{\boldsymbol{x}'}p_k\left(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{u}\right)\boldsymbol{\phi}\left(\boldsymbol{x}'\right)\ \mathrm{d}\boldsymbol{x}'-\boldsymbol{\theta}_k\boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_k}\right)\ \mathrm{d}\boldsymbol{u}\ \mathrm{d}\boldsymbol{x}\right]+\eta_K\log\int_{\boldsymbol{x}}q_K\left(\boldsymbol{x}\right)\exp\left(\frac{r_K\left(\boldsymbol{x}\right)-\boldsymbol{\theta}_K\boldsymbol{\phi}\left(\boldsymbol{x}\right)}{\eta_K}\right)\ \mathrm{d}\boldsymbol{x}$$

and insert (12), (13), (14), (15), (16), (17), (18).

$$\begin{aligned}
&g\left(\boldsymbol{V}_{1:K},\boldsymbol{v}_{1:K},\eta_{1:K}\right)=\left(-\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x},1}^T\boldsymbol{V}_1\boldsymbol{\mu}_{\boldsymbol{x},1}+\boldsymbol{\mu}_{\boldsymbol{x},1}^T\boldsymbol{v}_1-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_1\boldsymbol{\Sigma}_{\boldsymbol{x},1}\right)\right)+\sum_{k=1}^{K}\left[\eta_k\epsilon\right]\\
&+\sum_{k=1}^{K-1}\left[\eta_k\log\int_{\boldsymbol{y}}\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_{\boldsymbol{y},k},\boldsymbol{\Sigma}_{\boldsymbol{y},k}\right)\exp\left(\frac{\left(-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{R}_k\boldsymbol{y}+\boldsymbol{y}^T\boldsymbol{r}_k\right)+\left(-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{A}_k^T\boldsymbol{V}_{k+1}\boldsymbol{A}_k\boldsymbol{y}+\boldsymbol{y}^T\boldsymbol{A}_k^T\left(\boldsymbol{v}_{k+1}-\boldsymbol{V}_{k+1}\boldsymbol{a}_k\right)+\boldsymbol{a}_k^T\left(\boldsymbol{v}_{k+1}-\frac{1}{2}\boldsymbol{V}_{k+1}\boldsymbol{a}_k\right)-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_{k+1}\boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k}\right)\right)-\left(-\frac{1}{2}\boldsymbol{y}^T\hat{\boldsymbol{V}}_k\boldsymbol{y}+\boldsymbol{y}^T\hat{\boldsymbol{v}}_k\right)}{\eta_k}\right)\ \mathrm{d}\boldsymbol{y}\right]\\
&+\eta_K\log\int_{\boldsymbol{x}}\mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{x},K},\boldsymbol{\Sigma}_{\boldsymbol{x},K}\right)\exp\left(\frac{\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{R}_K\boldsymbol{x}+\boldsymbol{x}^T\boldsymbol{r}_K\right)-\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{V}_K\boldsymbol{x}+\boldsymbol{x}^T\boldsymbol{v}_K\right)}{\eta_K}\right)\ \mathrm{d}\boldsymbol{x}
\end{aligned}$$

$$\begin{aligned}
&=\left(-\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x},1}^T\boldsymbol{V}_1\boldsymbol{\mu}_{\boldsymbol{x},1}+\boldsymbol{\mu}_{\boldsymbol{x},1}^T\boldsymbol{v}_1-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_1\boldsymbol{\Sigma}_{\boldsymbol{x},1}\right)\right)+\sum_{k=1}^{K}\left[\eta_k\epsilon\right]\\
&+\sum_{k=1}^{K-1}\left[\eta_k\log\int_{\boldsymbol{y}}\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_{\boldsymbol{y},k},\boldsymbol{\Sigma}_{\boldsymbol{y},k}\right)\exp\left(\frac{-\frac{1}{2}\boldsymbol{y}^T\left(\boldsymbol{R}_k+\boldsymbol{A}_k^T\boldsymbol{V}_{k+1}\boldsymbol{A}_k-\hat{\boldsymbol{V}}_k\right)\boldsymbol{y}+\boldsymbol{y}^T\left(\boldsymbol{r}_k+\boldsymbol{A}_k^T\left(\boldsymbol{v}_{k+1}-\boldsymbol{V}_{k+1}\boldsymbol{a}_k\right)-\hat{\boldsymbol{v}}_k\right)+\boldsymbol{a}_k^T\left(\boldsymbol{v}_{k+1}-\frac{1}{2}\boldsymbol{V}_{k+1}\boldsymbol{a}_k\right)-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_{k+1}\boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k}\right)}{\eta_k}\right)\ \mathrm{d}\boldsymbol{y}\right]\\
&+\eta_K\log\int_{\boldsymbol{x}}\mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{x},K},\boldsymbol{\Sigma}_{\boldsymbol{x},K}\right)\exp\left(\frac{-\frac{1}{2}\boldsymbol{x}^T\left(\boldsymbol{R}_K-\boldsymbol{V}_K\right)\boldsymbol{x}+\boldsymbol{x}^T\left(\boldsymbol{r}_K-\boldsymbol{v}_K\right)}{\eta_K}\right)\ \mathrm{d}\boldsymbol{x}
\end{aligned}$$

$$\begin{aligned}
&=\left(-\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x},1}^T\boldsymbol{V}_1\boldsymbol{\mu}_{\boldsymbol{x},1}+\boldsymbol{\mu}_{\boldsymbol{x},1}^T\boldsymbol{v}_1-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_1\boldsymbol{\Sigma}_{\boldsymbol{x},1}\right)\right)+\sum_{k=1}^{K}\left[\eta_k\epsilon\right]\\
&+\sum_{k=1}^{K-1}\left[\left(\boldsymbol{a}_k^T\left(\boldsymbol{v}_{k+1}-\frac{1}{2}\boldsymbol{V}_{k+1}\boldsymbol{a}_k\right)-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_{k+1}\boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k}\right)\right)+\eta_k\log\int_{\boldsymbol{y}}\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{\mu}_{\boldsymbol{y},k},\boldsymbol{\Sigma}_{\boldsymbol{y},k}\right)\exp\left(-\frac{1}{2}\boldsymbol{y}^T\frac{\boldsymbol{R}_k+\boldsymbol{A}_k^T\boldsymbol{V}_{k+1}\boldsymbol{A}_k-\hat{\boldsymbol{V}}_k}{\eta_k}\boldsymbol{y}+\boldsymbol{y}^T\frac{\boldsymbol{r}_k+\boldsymbol{A}_k^T\left(\boldsymbol{v}_{k+1}-\boldsymbol{V}_{k+1}\boldsymbol{a}_k\right)-\hat{\boldsymbol{v}}_k}{\eta_k}\right)\ \mathrm{d}\boldsymbol{y}\right]\\
&+\eta_K\log\int_{\boldsymbol{x}}\mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{x},K},\boldsymbol{\Sigma}_{\boldsymbol{x},K}\right)\exp\left(-\frac{1}{2}\boldsymbol{x}^T\frac{\boldsymbol{R}_K-\boldsymbol{V}_K}{\eta_K}\boldsymbol{x}+\boldsymbol{x}^T\frac{\boldsymbol{r}_K-\boldsymbol{v}_K}{\eta_K}\right)\ \mathrm{d}\boldsymbol{x}
\end{aligned}$$

Introducing the abbrevations

$$
\begin{aligned}
\boldsymbol{\Omega}_k &= \boldsymbol{R}_k + \boldsymbol{A}_k^T \boldsymbol{V}_{k+1} \boldsymbol{A}_k - \hat{\boldsymbol{V}}_k \\
\boldsymbol{\omega}_k &= \boldsymbol{r}_k + \boldsymbol{A}_k^T \left(\boldsymbol{v}_{k+1} - \boldsymbol{V}_{k+1} \boldsymbol{a}_k \right) - \hat{\boldsymbol{v}}_k \\
\boldsymbol{\Omega}_K &= \boldsymbol{R}_K - \boldsymbol{V}_K \\
\boldsymbol{\omega}_K &= \boldsymbol{r}_K - \boldsymbol{v}_K
\end{aligned}
$$

we can rewrite the dual as

$$
\begin{aligned}
g\left(\boldsymbol{V}_{1:K}, \boldsymbol{v}_{1:K}, \eta_{1:K}\right) &= \left(-\frac{1}{2} \boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{V}_1 \boldsymbol{\mu}_{\boldsymbol{x},1} + \boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{v}_1 - \frac{1}{2} \mathrm{tr}\left(\boldsymbol{V}_1 \boldsymbol{\Sigma}_{\boldsymbol{x},1} \right) \right) + \sum_{k=1}^{K} \left[\eta_k \epsilon \right] \\
&+ \sum_{k=1}^{K-1} \left[\left(\boldsymbol{a}_k^T \left(\boldsymbol{v}_{k+1} - \frac{1}{2} \boldsymbol{V}_{k+1} \boldsymbol{a}_k \right) - \frac{1}{2} \mathrm{tr}\left(\boldsymbol{V}_{k+1} \boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k} \right) \right) + \eta_k \log \int_{\boldsymbol{y}} \mathcal{N}\left(\boldsymbol{y} | \boldsymbol{\mu}_{\boldsymbol{y},k}, \boldsymbol{\Sigma}_{\boldsymbol{y},k} \right) \exp\left(-\frac{1}{2} \boldsymbol{y}^T \frac{\boldsymbol{\Omega}_k}{\eta_k} \boldsymbol{y} + \boldsymbol{y}^T \frac{\boldsymbol{\omega}_k}{\eta_k} \right) \, \mathrm{d}\boldsymbol{y} \right] \\
&+ \eta_K \log \int_{\boldsymbol{x}} \mathcal{N}\left(\boldsymbol{x} | \boldsymbol{\mu}_{\boldsymbol{x},K}, \boldsymbol{\Sigma}_{\boldsymbol{x},K} \right) \exp\left(-\frac{1}{2} \boldsymbol{x}^T \frac{\boldsymbol{\Omega}_K}{\eta_K} \boldsymbol{x} + \boldsymbol{x}^T \frac{\boldsymbol{\omega}_K}{\eta_K} \right) \, \mathrm{d}\boldsymbol{x} \\
&= \left(-\frac{1}{2} \boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{V}_1 \boldsymbol{\mu}_{\boldsymbol{x},1} + \boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{v}_1 - \frac{1}{2} \mathrm{tr}\left(\boldsymbol{V}_1 \boldsymbol{\Sigma}_{\boldsymbol{x},1} \right) \right) + \sum_{k=1}^{K} \left[\eta_k \epsilon \right] \\
&+ \sum_{k=1}^{K-1} \left[\left(\boldsymbol{a}_k^T \left(\boldsymbol{v}_{k+1} - \frac{1}{2} \boldsymbol{V}_{k+1} \boldsymbol{a}_k \right) - \frac{1}{2} \mathrm{tr}\left(\boldsymbol{V}_{k+1} \boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k} \right) \right) + \eta_k \log \int_{\boldsymbol{y}} \frac{\exp\left(-\frac{1}{2} \boldsymbol{\mu}_{\boldsymbol{y},k}^T \boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} \boldsymbol{\mu}_{\boldsymbol{y},k} \right)}{\left| 2\pi \boldsymbol{\Sigma}_{\boldsymbol{y},k} \right|^{1/2}} \exp\left(-\frac{1}{2} \boldsymbol{y}^T \boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} \boldsymbol{\mu}_{\boldsymbol{y},k} \right) \exp\left(-\frac{1}{2} \boldsymbol{y}^T \frac{\boldsymbol{\Omega}_k}{\eta_k} \boldsymbol{y} + \boldsymbol{y}^T \frac{\boldsymbol{\omega}_k}{\eta_k} \right) \, \mathrm{d}\boldsymbol{y} \right] \\
&+ \eta_K \log \int_{\boldsymbol{x}} \frac{\exp\left(-\frac{1}{2} \boldsymbol{\mu}_{\boldsymbol{x},K}^T \boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} \boldsymbol{\mu}_{\boldsymbol{x},K} \right)}{\left| 2\pi \boldsymbol{\Sigma}_{\boldsymbol{x},K} \right|^{1/2}} \exp\left(-\frac{1}{2} \boldsymbol{x}^T \boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} \boldsymbol{\mu}_{\boldsymbol{x},K} \right) \exp\left(-\frac{1}{2} \boldsymbol{x}^T \frac{\boldsymbol{\Omega}_K}{\eta_K} \boldsymbol{x} + \boldsymbol{x}^T \frac{\boldsymbol{\omega}_K}{\eta_K} \right) \, \mathrm{d}\boldsymbol{x} \\
&= \left(-\frac{1}{2} \boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{V}_1 \boldsymbol{\mu}_{\boldsymbol{x},1} + \boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{v}_1 - \frac{1}{2} \mathrm{tr}\left(\boldsymbol{V}_1 \boldsymbol{\Sigma}_{\boldsymbol{x},1} \right) \right) + \sum_{k=1}^{K} \left[\eta_k \epsilon \right] \\
&+ \sum_{k=1}^{K-1} \left[\left(\boldsymbol{a}_k^T \left(\boldsymbol{v}_{k+1} - \frac{1}{2} \boldsymbol{V}_{k+1} \boldsymbol{a}_k \right) - \frac{1}{2} \mathrm{tr}\left(\boldsymbol{V}_{k+1} \boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k} \right) \right) + \eta_k \log \int_{\boldsymbol{y}} \frac{\exp\left(-\frac{1}{2} \boldsymbol{\mu}_{\boldsymbol{y},k}^T \boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} \boldsymbol{\mu}_{\boldsymbol{y},k} \right)}{\left| 2\pi \boldsymbol{\Sigma}_{\boldsymbol{y},k} \right|^{1/2}} \exp\left(-\frac{1}{2} \boldsymbol{y}^T \left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} + \frac{\boldsymbol{\Omega}_k}{\eta_k} \right) \boldsymbol{y} + \boldsymbol{y}^T \left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} \boldsymbol{\mu}_{\boldsymbol{y},k} + \frac{\boldsymbol{\omega}_k}{\eta_k} \right) \right) \, \mathrm{d}\boldsymbol{y} \right] \\
&+ \eta_K \log \int_{\boldsymbol{x}} \frac{\exp\left(-\frac{1}{2} \boldsymbol{\mu}_{\boldsymbol{x},K}^T \boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} \boldsymbol{\mu}_{\boldsymbol{x},K} \right)}{\left| 2\pi \boldsymbol{\Sigma}_{\boldsymbol{x},K} \right|^{1/2}} \exp\left(-\frac{1}{2} \boldsymbol{x}^T \left(\boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} + \frac{\boldsymbol{\omega}_K}{\eta_K} \right) \boldsymbol{x} + \boldsymbol{x}^T \left(\boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} \boldsymbol{\mu}_{\boldsymbol{x},K} + \frac{\boldsymbol{\omega}_K}{\eta_K} \right) \right) \, \mathrm{d}\boldsymbol{x}
\end{aligned}
$$

The insides of both integrals are proportinal to Gaussian distributions. So by multiplying them with a respective factor we can remove the integrals from the equation, and, hence, find the closed form of the dual.
The goal is to find the factor which leads to the following partition function

$$\frac{\exp\left(-\frac{1}{2}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}+\frac{\boldsymbol{\omega}_k}{\eta_k}\right)^T\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}+\frac{\boldsymbol{\omega}_k}{\eta_k}\right)\right)}{\left|2\pi\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\right|^{1/2}}$$

$$=\frac{\exp\left(-\frac{1}{2}\left(\boldsymbol{\mu}_{\boldsymbol{y},k}^T\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}+\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}+\boldsymbol{\mu}_{\boldsymbol{y},k}^T\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\frac{\boldsymbol{\omega}_k}{\eta_k}+\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\frac{\boldsymbol{\omega}_k}{\eta_k}\right)\right)}{\left|2\pi\boldsymbol{\Sigma}_{\boldsymbol{y},k}\left(\frac{\boldsymbol{\Omega}_k}{\eta_k}\boldsymbol{\Sigma}_{\boldsymbol{y},k}+\mathrm{I}\right)^{-1}\right|^{1/2}}$$

$$=\frac{\exp\left(-\frac{1}{2}\left(\boldsymbol{\mu}_{\boldsymbol{y},k}^T\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}-\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}+\left(\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\right)^{-1}\right)\boldsymbol{\mu}_{\boldsymbol{y},k}+2\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}+\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\frac{\boldsymbol{\omega}_k}{\eta_k}\right)\right)}{\left|2\pi\boldsymbol{\Sigma}_{\boldsymbol{y},k}\left(\frac{\boldsymbol{\Omega}_k}{\eta_k}\boldsymbol{\Sigma}_{\boldsymbol{y},k}+\mathrm{I}\right)^{-1}\right|^{1/2}}$$

$$=\frac{\exp\left(-\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{y},k}^T\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}+\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{y},k}^T\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}+\left(\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\right)^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}-\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}-\frac{1}{2}\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\frac{\boldsymbol{\omega}_k}{\eta_k}\right)}{\left|2\pi\boldsymbol{\Sigma}_{\boldsymbol{y},k}\right|^{1/2}\left|\left(\frac{\boldsymbol{\Omega}_k}{\eta_k}\boldsymbol{\Sigma}_{\boldsymbol{y},k}+\mathrm{I}\right)^{-1}\right|^{1/2}}$$

$$=\frac{\exp\left(-\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{y},k}^T\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}\right)}{\left|2\pi\boldsymbol{\Sigma}_{\boldsymbol{y},k}\right|^{1/2}}\frac{\exp\left(\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{y},k}^T\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}+\left(\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\right)^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}-\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}-\frac{1}{2}\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\frac{\boldsymbol{\omega}_k}{\eta_k}\right)}{\left|\left(\frac{\boldsymbol{\Omega}_k}{\eta_k}\boldsymbol{\Sigma}_{\boldsymbol{y},k}+\mathrm{I}\right)^{-1}\right|^{1/2}}$$

$$=\frac{\exp\left(-\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{y},k}^T\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}\right)}{\left|2\pi\boldsymbol{\Sigma}_{\boldsymbol{y},k}\right|^{1/2}}\exp\left(\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{y},k}^T\frac{\boldsymbol{\Omega}_k}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}-\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}-\frac{1}{2}\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\frac{\boldsymbol{\omega}_k}{\eta_k}\right)\left|\frac{\boldsymbol{\Omega}_k}{\eta_k}\boldsymbol{\Sigma}_{\boldsymbol{y},k}+\mathrm{I}\right|^{1/2}$$

$$=\frac{\exp\left(-\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{y},k}^T\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}\right)}{\left|2\pi\boldsymbol{\Sigma}_{\boldsymbol{y},k}\right|^{1/2}}\exp\left(\left(\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{y},k}^T\frac{\boldsymbol{\Omega}_k}{\eta_k}-\frac{\boldsymbol{\omega}_k^T}{\eta_k}\right)\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k}-\frac{1}{2}\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}+\frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\frac{\boldsymbol{\omega}_k}{\eta_k}\right)\left|\frac{\boldsymbol{\Omega}_k}{\eta_k}\boldsymbol{\Sigma}_{\boldsymbol{y},k}+\mathrm{I}\right|^{1/2}$$

We see that the first factor corresponds to the current partition function, and therefore the second fator is the one we were looking for.
The derivation of the factor for $k = K$ is analog.

We multiply the factor to the inside of the integral, which afterwards evaluates to one, and its inverse outside the integral.

$$
\begin{aligned}
g\left(\boldsymbol{V}_{1:K}, \boldsymbol{v}_{1:K}, \eta_{1:K}\right) &= \left(-\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{V}_1 \boldsymbol{\mu}_{\boldsymbol{x},1} + \boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{v}_1 - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_1 \boldsymbol{\Sigma}_{\boldsymbol{x},1}\right)\right) + \sum_{k=1}^{K} \left[\eta_k \epsilon\right] \\
&+ \sum_{k=1}^{K-1} \left[\left(\boldsymbol{a}_k^T \left(\boldsymbol{v}_{k+1} - \frac{1}{2}\boldsymbol{V}_{k+1}\boldsymbol{a}_k\right) - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_{k+1}\boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k}\right)\right) + \eta_k \log\left(\exp\left(\left(\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{y},k}^T \frac{\boldsymbol{\Omega}_k}{\eta_k} - \frac{\boldsymbol{\omega}_k^T}{\eta_k}\right)\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} + \frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1} \boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k} - \frac{1}{2}\frac{\boldsymbol{\omega}_k^T}{\eta_k}\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} + \frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\frac{\boldsymbol{\omega}_k}{\eta_k}\right)\left|\frac{\boldsymbol{\Omega}_k}{\eta_k}\boldsymbol{\Sigma}_{\boldsymbol{y},k} + \mathrm{I}\right|^{1/2}\right)^{-1}\right] \\
&+ \eta_K \log\left(\exp\left(\left(\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x},K}^T \frac{\boldsymbol{\Omega}_K}{\eta_K} - \frac{\boldsymbol{\omega}_K^T}{\eta_K}\right)\left(\boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} + \frac{\boldsymbol{\Omega}_K}{\eta_K}\right)^{-1} \boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1}\boldsymbol{\mu}_{\boldsymbol{x},K} - \frac{1}{2}\frac{\boldsymbol{\omega}_K^T}{\eta_K}\left(\boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} + \frac{\boldsymbol{\Omega}_K}{\eta_K}\right)^{-1}\frac{\boldsymbol{\omega}_K}{\eta_K}\right)\left|\frac{\boldsymbol{\Omega}_K}{\eta_K}\boldsymbol{\Sigma}_{\boldsymbol{x},K} + \mathrm{I}\right|^{1/2}\right)^{-1} \\
&= \left(-\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{V}_1 \boldsymbol{\mu}_{\boldsymbol{x},1} + \boldsymbol{\mu}_{\boldsymbol{x},1}^T \boldsymbol{v}_1 - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_1 \boldsymbol{\Sigma}_{\boldsymbol{x},1}\right)\right) + \sum_{k=1}^{K} \left[\eta_k \epsilon\right] \\
&+ \sum_{k=1}^{K-1} \left[\left(\boldsymbol{a}_k^T \left(\boldsymbol{v}_{k+1} - \frac{1}{2}\boldsymbol{V}_{k+1}\boldsymbol{a}_k\right) - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{V}_{k+1}\boldsymbol{\Sigma}_{\boldsymbol{x}'|\boldsymbol{y},k}\right)\right) + \left(-\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{y},k}^T\boldsymbol{\Omega}_k + \boldsymbol{\omega}_k^T\right)\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} + \frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1}\boldsymbol{\mu}_{\boldsymbol{y},k} + \frac{1}{2}\boldsymbol{\omega}_k^T\left(\boldsymbol{\Sigma}_{\boldsymbol{y},k}^{-1} + \frac{\boldsymbol{\Omega}_k}{\eta_k}\right)^{-1}\frac{\boldsymbol{\omega}_k}{\eta_k} - \frac{1}{2}\log\left|\frac{\boldsymbol{\Omega}_k}{\eta_k}\boldsymbol{\Sigma}_{\boldsymbol{y},k} + \mathrm{I}\right|\right] \\
&+ \eta_K \log\left(-\frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x},K}^T\boldsymbol{\Omega}_K + \boldsymbol{\omega}_K^T\right)\left(\boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} + \frac{\boldsymbol{\Omega}_K}{\eta_K}\right)^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1}\boldsymbol{\mu}_{\boldsymbol{x},K} + \frac{1}{2}\boldsymbol{\omega}_K^T\left(\boldsymbol{\Sigma}_{\boldsymbol{x},K}^{-1} + \frac{\boldsymbol{\Omega}_K}{\eta_K}\right)^{-1}\frac{\boldsymbol{\omega}_K}{\eta_K} - \frac{1}{2}\log\left|\frac{\boldsymbol{\Omega}_K}{\eta_K}\boldsymbol{\Sigma}_{\boldsymbol{x},K} + \mathrm{I}\right|
\end{aligned}
\tag{19}
$$