Journal of Machine Learning Research 11 (2010) 3137-3181 Submitted 1/10; Revised 7/10; Published 11/10

A Generalized Path Integral Control Approach to Reinforcement Learning

Evangelos A.Theodorou ETHEODOR@USC.EDU
Jonas Buchli JONAS@BUCHLI.ORG
Stefan Schaal* SSCHAAL@USC.EDU
Department of Computer Science
University of Southern California
Los Angeles, CA 90089-2905, USA

Editor: Daniel Lee

Abstract

With the goal to generate more scalable algorithms with higher efficiency and fewer open parameters, reinforcement learning (RL) has recently moved towards combining classical techniques from optimal control and dynamic programming with modern learning techniques from statistical estimation theory. In this vein, this paper suggests to use the framework of stochastic optimal control with path integrals to derive a novel approach to RL with parameterized policies. While solidly grounded in value function estimation and optimal control based on the stochastic Hamilton-Jacobi-Bellman (HJB) equations, policy improvements can be transformed into an approximation problem of a path integral which has no open algorithmic parameters other than the exploration noise. The resulting algorithm can be conceived of as model-based, semi-model-based, or even model free, depending on how the learning problem is structured. The update equations have no danger of numerical instabilities as neither matrix inversions nor gradient learning rates are required. Our new algorithm demonstrates interesting similarities with previous RL research in the framework of probability matching and provides intuition why the slightly heuristically motivated probability matching approach can actually perform well. Empirical evaluations demonstrate significant performance improvements over gradient-based policy learning and scalability to high-dimensional control problems. Finally, a learning experiment on a simulated 12 degree-of-freedom robot dog illustrates the functionality of our algorithm in a complex robot learning scenario. We believe that **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}) offers currently one of the most efficient, numerically robust, and easy to implement algorithms for RL based on trajectory roll-outs.

Keywords: stochastic optimal control, reinforcement learning, parameterized policies

1. Introduction

While reinforcement learning (RL) is among the most general frameworks of learning control to create truly autonomous learning systems, its scalability to high-dimensional continuous state-action systems, for example, humanoid robots, remains problematic. Classical value-function based methods with function approximation offer one possible approach, but function approximation under the non-stationary iterative learning process of the value-function remains difficult when one exceeds about 5-10 dimensions. Alternatively, direct policy learning from trajectory roll-outs has recently made significant progress (Peters, 2007), but can still become numerically brittle and full of open

∗. Also at ATR Computational Neuroscience Laboratories, Kyoto 619-0288, Japan.

tuning parameters in complex learning problems. In new developments, RL researchers have started to combine the well-developed methods from statistical learning and empirical inference with classical RL approaches in order to minimize tuning parameters and numerical problems, such that ultimately more efficient algorithms can be developed that scale to significantly more complex learning system (Dayan and Hinton, 1997; Koeber and Peters, 2008; Peters and Schaal, 2008c; Toussaint and Storkey, 2006; Ghavamzadeh and Yaakov, 2007; Deisenroth et al., 2009; Vlassis et al., 2009; Jetchev and Toussaint, 2009).

In the spirit of these latter ideas, this paper addresses a new method of probabilistic reinforcement learning derived from the framework of stochastic optimal control and path integrals, based on the original work of Kappen (2007) and Broek et al. (2008). As will be detailed in the sections below, this approach makes an appealing theoretical connection between value function approximation using the stochastic HJB equations and direct policy learning by approximating a path integral, that is, by solving a statistical inference problem from sample roll-outs. The resulting algorithm, called **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}), takes on a surprisingly simple form, has no open algorithmic tuning parameters besides the exploration noise, and it has numerically robust performance in high dimensional learning problems. It also makes an interesting connection to previous work on RL based on probability matching (Dayan and Hinton, 1997; Peters and Schaal, 2008c; Koeber and Peters, 2008) and motivates why probability matching algorithms can be successful.

This paper is structured into several major sections:

- Section 2 addresses the theoretical development of stochastic optimal control with path integrals. This is a fairly theoretical section. For a quick reading, we would recommend Section 2.1 for our basic notation, and Table 1 for the final results. Exposing the reader to a sketch of the details of the derivations opens the possibility to derive path integral optimal control solutions for other dynamical systems than the one we address in Section 2.1.

  The main steps of the theoretical development include:

  - Problem formulation of stochastic optimal control with the stochastic Hamilton-Jacobi-Bellman (HJB) equation
  - The transformation of the HJB into a linear PDE
  - The generalized path integral formulation for control systems with controlled and uncontrolled differential equations
  - General derivation of optimal controls for the path integral formalism
  - Path integral optimal control applied to special cases of control systems

- Section 3 relates path integral optimal control to reinforcement learning. Several main issues are addressed:

  - Reinforcement learning with parameterized policies
  - Dynamic Movement Primitives (DMP) as a special case of parameterized policies, which matches the problem formulation of path integral optimal control.
  - Derivation of **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}), which is an application of path integral optimal control to DMPs.

- Section 4 discusses related work.

- Section 5 illustrates several applications of $\mathbf{PI}^2$ to control problems in robotics.
- Section 6 addresses several important issues and characteristics of RL with $\mathbf{PI}^2$.

2. Stochastic Optimal Control with Path Integrals

The goal in stochastic optimal control framework is to control a stochastic dynamical system while minimizing a performance criterion. Therefore, stochastic optimal control can be thought as a constrained optimization problem in which the constrains corresponds to stochastic dynamical systems. The analysis and derivations of stochastic optimal control and path integrals in the next sections rely on the Bellman Principle of optimality (Bellman and Kalaba, 1964) and the HJB equation.

2.1 Stochastic Optimal Control Definition and Notation

For our technical developments, we will use largely a control theoretic notation from trajectory-based optimal control, however, with an attempt to have as much overlap as possible with the standard RL notation (Sutton and Barto, 1998). Let us define a finite horizon cost function for a trajectory τ_i (which can also be a piece of a trajectory) starting at time t_i in state $\mathbf{x}_{t_i}$ and ending at time[1] t_N

$$R(\tau_i) = \phi_{t_N} + \int_{t_i}^{t_N} r_t \, dt, \tag{1}$$

with $\phi_{t_N} = \phi(x_{t_N})$ denoting a terminal reward at time t_N and r_t denoting the immediate cost at time t. In stochastic optimal control (Stengel, 1994), the goal is to find the controls $\mathbf{u}_t$ that minimize the value function:

$$V(\mathbf{x}_{t_i}) = V_{t_i} = \min_{\mathbf{u}_{t_i:t_N}} E_{\tau_i}\left[R(\tau_i)\right], \tag{2}$$

where the expectation $E_{\tau_i}[.]$ is taken over all trajectories starting at $\mathbf{x}_{t_i}$. We consider the rather general class of control systems:

$$\dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t,t) + \mathbf{G}(\mathbf{x}_t)\left(\mathbf{u}_t + \varepsilon_t\right) = \mathbf{f}_t + \mathbf{G}_t\left(\mathbf{u}_t + \varepsilon_t\right), \tag{3}$$

with $\mathbf{x}_t \in \Re^{n\times 1}$ denoting the state of the system, $\mathbf{G}_t = \mathbf{G}(\mathbf{x}_t) \in \Re^{n\times p}$ the control matrix, $\mathbf{f}_t = \mathbf{f}(\mathbf{x}_t) \in \Re^{n\times 1}$ the passive dynamics, $\mathbf{u}_t \in \Re^{p\times 1}$ the control vector and $\varepsilon_t \in \Re^{p\times 1}$ Gaussian noise with variance Σ_{ε}. As immediate cost we consider

$$r_t = r(\mathbf{x}_t,\mathbf{u}_t,t) = q_t + \frac{1}{2}\mathbf{u}_t^T\mathbf{R}\mathbf{u}_t, \tag{4}$$

where $q_t = q(\mathbf{x}_t,t)$ is an arbitrary state-dependent cost function, and $\mathbf{R}$ is the positive semi-definite weight matrix of the quadratic control cost. The stochastic HJB equation (Stengel, 1994; Fleming and Soner, 2006) associated with this stochastic optimal control problem is expressed as follows:

$$-\partial_t V_t = \min_{\mathbf{u}}\left(r_t + (\nabla_{\mathbf{x}}V_t)^T\mathbf{F}_t + \frac{1}{2}trace\left((\nabla_{\mathbf{xx}}V_t)\mathbf{G}_t\Sigma_{\varepsilon}\mathbf{G}_t^T\right)\right), \tag{5}$$

1. If we need to emphasize a particular time, we denote it by t_i, which also simplifies a transition to discrete time notation later. We use t without subscript when no emphasis is needed when this "time slice" occurs, t_0 for the start of a trajectory, and t_N for the end of a trajectory.

where $\mathbf{F}_t$ is defined as $\mathbf{F}_t = \mathbf{f}(\mathbf{x}_t,t) + \mathbf{G}(\mathbf{x}_t)\mathbf{u}_t$. To find the minimum, the cost function (4) is inserted into (5) and the gradient of the expression inside the parenthesis is taken with respect to controls $\mathbf{u}$ and set to zero. The corresponding optimal control is given by the equation:

$$\mathbf{u}(\mathbf{x}_t) = \mathbf{u}_t = -\mathbf{R}^{-1}\mathbf{G}_t^T(\nabla_{x_t}V_t).$$

Substitution of the optimal control above, into the stochastic HJB (5), results in the following nonlinear and second order Partial Differential Equation (PDE):

$$-\partial_t V_t = q_t + (\nabla_{\mathbf{x}}V_t)^T\mathbf{f}_t - \frac{1}{2}(\nabla_{\mathbf{x}}V_t)^T\mathbf{G}_t\mathbf{R}^{-1}\mathbf{G}_t^T(\nabla_{\mathbf{x}}V_t) + \frac{1}{2}trace\left((\nabla_{\mathbf{xx}}V_t)\mathbf{G}_t\Sigma_{\boldsymbol{\varepsilon}}\mathbf{G}_t^T\right).$$

The $\nabla_{\mathbf{x}}$ and $\nabla_{\mathbf{xx}}$ symbols refer to the Jacobian and Hessian, respectively, of the value function with respect to the state $\mathbf{x}$, while ∂_t is the partial derivative with respect to time. For notational compactness, we will mostly use subscripted symbols to denote time and state dependencies, as introduced in the equations above.

2.2 Transformation of HJB into a Linear PDE

In order to find a solution to the PDE above, we use a exponential transformation of the value function:

$$V_t = -\lambda\log\Psi_t.$$

Given this logarithmic transformation, the partial derivatives of the value function with respect to time and state are expressed as follows:

$$\partial_t V_t = -\lambda\frac{1}{\Psi_t}\partial_t\Psi_t,$$

$$\nabla_{\mathbf{x}}V_t = -\lambda\frac{1}{\Psi_t}\nabla_{\mathbf{x}}\Psi_t,$$

$$\nabla_{\mathbf{xx}}V_t = \lambda\frac{1}{\Psi_t^2}\nabla_{\mathbf{x}}\Psi_t\,\nabla_{\mathbf{x}}\Psi_t^T - \lambda\frac{1}{\Psi_t}\nabla_{\mathbf{xx}}\Psi_t.$$

Inserting the logarithmic transformation and the derivatives of the value function we obtain:

$$\frac{\lambda}{\Psi_t}\partial_t\Psi_t = q_t - \frac{\lambda}{\Psi_t}(\nabla_{\mathbf{x}}\Psi_t)^T\mathbf{f}_t - \underline{\frac{\lambda^2}{2\Psi_t^2}(\nabla_{\mathbf{x}}\Psi_t)^T\mathbf{G}_t\mathbf{R}^{-1}\mathbf{G}_t^T(\nabla_{\mathbf{x}}\Psi_t)} + \frac{1}{2}trace(\Gamma), \tag{6}$$

where the term Γ is expressed as:

$$\Gamma = \left(\lambda\frac{1}{\Psi_t^2}\nabla_{\mathbf{x}}\Psi_t\,\nabla_{\mathbf{x}}\Psi_t^T - \lambda\frac{1}{\Psi_t}\nabla_{\mathbf{xx}}\Psi_t\right)\mathbf{G}_t\Sigma_{\boldsymbol{\varepsilon}}\mathbf{G}_t^T.$$

The trace of Γ is therefore:

$$trace(\Gamma) = \underline{\lambda\frac{1}{\Psi^2}trace\left(\nabla_{\mathbf{x}}\Psi_t^T\mathbf{G}_t\Sigma_{\boldsymbol{\varepsilon}}\mathbf{G}_t\nabla_{\mathbf{x}}\Psi_t\right)} - \lambda\frac{1}{\Psi_t}trace\left(\nabla_{\mathbf{xx}}\Psi_t\mathbf{G}_t\Sigma_{\boldsymbol{\varepsilon}}\mathbf{G}_t^T\right). \tag{7}$$

Comparing the underlined terms in (6) and (7), one can recognize that these terms will cancel under the assumption of $\lambda \mathbf{R}^{-1} = \Sigma_{\varepsilon}$, which implies the simplification:

$$\lambda \mathbf{G}_t \mathbf{R}^{-1} \mathbf{G}_t^T = \mathbf{G}_t \Sigma_{\varepsilon} \mathbf{G}_t^T = \Sigma(\mathbf{x}_t) = \Sigma_t. \tag{8}$$

The intuition behind this assumption (cf. also Kappen, 2007; Broek et al., 2008) is that, since the weight control matrix $\mathbf{R}$ is inverse proportional to the variance of the noise, a high variance control input implies cheap control cost, while small variance control inputs have high control cost. From a control theoretic stand point such a relationship makes sense due to the fact that under a large disturbance (= high variance) significant control authority is required to bring the system back to a desirable state. This control authority can be achieved with corresponding low control cost in $\mathbf{R}$.

With this simplification, (6) reduces to the following form

$$-\partial_t \Psi_t = -\frac{1}{\lambda} q_t \Psi_t + \mathbf{f}_t^T (\nabla_{\mathbf{x}} \Psi_t) + \frac{1}{2} trace \left((\nabla_{\mathbf{xx}} \Psi_t) \mathbf{G}_t \Sigma_{\varepsilon} \mathbf{G}_t^T \right), \tag{9}$$

with boundary condition: $\Psi_{t_N} = \exp\left(-\frac{1}{\lambda}\phi_{t_N}\right)$. The partial differential equation (PDE) in (9) corresponds to the so called Chapman Kolmogorov PDE, which is of second order and linear. Analytical solutions of (9) cannot be found in general for general nonlinear systems and cost functions. However, there is a connection between solutions of PDEs and their representation as stochastic differential equation (SDEs), that is mathematically expressed by the Feynman-Kac formula (Øksendal, 2003; Yong, 1997). The Feynman-Kac formula (see appendix B) can be used to find distributions of random processes which solve certain SDEs as well as to propose numerical methods for solving certain PDEs. Applying the Feynman-Kac theorem, the solution of (9) is:

$$\Psi_{t_i} = E_{\tau_i}\left(\Psi_{t_N} e^{-\int_{t_i}^{t_N} \frac{1}{\lambda} q_t dt} \right) = E_{\tau_i}\left[\exp\left(-\frac{1}{\lambda}\phi_{t_N} - \frac{1}{\lambda} \int_{t_i}^{t_N} q_t \, dt \right) \right]. \tag{10}$$

Thus, we have transformed our stochastic optimal control problem into the approximation problem of a path integral. With a view towards a discrete time approximation, which will be needed for numerical implementations, the solution (10) can be formulated as:

$$\Psi_{t_i} = \lim_{dt \to 0} \int p(\tau_i | \mathbf{x}_i) \exp\left[-\frac{1}{\lambda} \left(\phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt \right) \right] d\tau_i, \tag{11}$$

where $\tau_i = (\mathbf{x}_{t_i}, ..., \mathbf{x}_{t_N})$ is a sample path (or trajectory piece) starting at state $\mathbf{x}_{t_i}$ and the term $p(\tau_i | \mathbf{x}_i)$ is the probability of sample path τ_i conditioned on the start state $\mathbf{x}_{t_i}$. Since Equation (11) provides the exponential cost to go Ψ_{t_i} in state $\mathbf{x}_{t_i}$, the integration above is taken with respect to sample paths $\tau_i = (\mathbf{x}_{t_i}, \mathbf{x}_{t_{i+1}}, ..., \mathbf{x}_{t_N})$. The differential term $d\tau_i$ is defined as $d\tau_i = (d\mathbf{x}_{t_i}, ..., d\mathbf{x}_{t_N})$. Evaluation of the stochastic integral in (11) requires the specification of $p(\tau_i | \mathbf{x}_i)$, which is the topic of our analysis in the next section.

2.3 Generalized Path Integral Formulation

To develop our algorithms, we will need to consider a more general development of the path integral approach to stochastic optimal control than presented in Kappen (2007) and Broek et al. (2008). In particular, we have to address that in many stochastic dynamical systems, the control transition matrix $\mathbf{G}_t$ is state dependent and its structure depends on the partition of the state in directly and

non-directly actuated parts. Since only some of the states are directly controlled, the state vector is partitioned into $\mathbf{x} = [\mathbf{x}^{(m)^T} \;\; \mathbf{x}^{(c)^T}]^T$ with $\mathbf{x}^{(m)} \in \Re^{k\times 1}$ the non-directly actuated part and $\mathbf{x}^{(c)} \in \Re^{l\times 1}$the directly actuated part. Subsequently, the passive dynamics term and the control transition matrix can be partitioned as $\mathbf{f}_t = [\mathbf{f}_t^{(m)^T} \;\; \mathbf{f}_t^{(c)^T}]^T$ with $\mathbf{f}_m \in \Re^{k\times 1}$, $\mathbf{f}_c \in \Re^{l\times 1}$ and $\mathbf{G}_t = [0_{k\times p} \;\; \mathbf{G}_t^{(c)^T}]^T$ with $\mathbf{G}_t^{(c)} \in \Re^{l\times p}$. The discretized state space representation of such systems is given as:

$$\mathbf{x}_{t_{i+1}} = \mathbf{x}_{t_i} + \mathbf{f}_{t_i} dt + \mathbf{G}_{t_i} \left(\mathbf{u}_{t_i} dt + \sqrt{dt}\varepsilon_{t_i} \right),$$

or, in partitioned vector form:

$$\begin{pmatrix} \mathbf{x}_{t_{i+1}}^{(m)} \\ \mathbf{x}_{t_{i+1}}^{(c)} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{t_i}^{(m)} \\ \mathbf{x}_{t_i}^{(c)} \end{pmatrix} + \begin{pmatrix} \mathbf{f}_{t_i}^{(m)} \\ \mathbf{f}_{t_i}^{(c)} \end{pmatrix} dt + \begin{pmatrix} 0_{k\times p} \\ \mathbf{G}_{t_i}^{(c)} \end{pmatrix} \left(\mathbf{u}_{t_i} dt + \sqrt{dt}\varepsilon_{t_i} \right). \tag{12}$$

Essentially the stochastic dynamics are partitioned into controlled equations in which the state $\mathbf{x}_{t_{i+1}}^{(c)}$ is directly actuated and the uncontrolled equations in which the state $\mathbf{x}_{t_{i+1}}^{(m)}$ is not directly actuated. Since stochasticity is only added in the directly actuated terms (c) of (12), we can develop $p(\tau_i|\mathbf{x}_i)$ as follows.

$$\begin{aligned} p(\tau_i|\mathbf{x}_{t_i}) &= p(\tau_{i+1}|\mathbf{x}_{t_i}) \\ &= p(\mathbf{x}_{t_N}, ..., \mathbf{x}_{t_{i+1}}|\mathbf{x}_{t_i}) \\ &= \Pi_{j=i}^{N-1} p\left(\mathbf{x}_{t_{j+1}}|\mathbf{x}_{t_j}\right), \end{aligned}$$

where we exploited the fact that the start state $\mathbf{x}_{t_i}$ of a trajectory is given and does not contribute to its probability. For systems where the control has lower dimensionality than the state (12), the transition probabilities $p\left(\mathbf{x}_{t_{j+1}}|\mathbf{x}_{t_j}\right)$ are factorized as follows:

$$\begin{aligned} p\left(\mathbf{x}_{t_{j+1}}|\mathbf{x}_{t_j}\right) &= p\left(\mathbf{x}_{t_{j+1}}^{(m)}|\mathbf{x}_{t_j}\right) \cdot p\left(\mathbf{x}_{t_{j+1}}^{(c)}|\mathbf{x}_{t_j}\right) \\ &= p\left(\mathbf{x}_{t_{j+1}}^{(m)}|\mathbf{x}_{t_j}^{(m)}, \mathbf{x}_{t_j}^{(c)}\right) \cdot p\left(\mathbf{x}_{t_{j+1}}^{(c)}|\mathbf{x}_{t_j}^{(m)}, \mathbf{x}_{t_j}^{(c)}\right) \\ &\propto p\left(\mathbf{x}_{t_{j+1}}^{(c)}|\mathbf{x}_{t_j}\right), \end{aligned} \tag{13}$$

where we have used the fact that $p\left(\mathbf{x}_{t_{i+1}}^{(m)}|\mathbf{x}_{t_i}^{(m)}, \mathbf{x}_{t_i}^{(c)}\right)$ is the Dirac delta function, since $\mathbf{x}_{t_{j+1}}^{(m)}$ can be computed deterministically from $\mathbf{x}_{t_j}^{(m)}, \mathbf{x}_{t_j}^{(c)}$. For all practical purposes,[2] the transition probability of the stochastic dynamics is reduced to the transition probability of the directly actuated part of the state:

$$p(\tau_i|\mathbf{x}_{t_i}) = \Pi_{j=i}^{N-1} p\left(\mathbf{x}_{t_{j+1}}|\mathbf{x}_{t_j}\right) \propto \Pi_{j=i}^{N-1} p\left(\mathbf{x}_{t_{j+1}}^{(c)}|\mathbf{x}_{t_j}\right). \tag{14}$$

Since we assume that the noise ε is zero mean Gaussian distributed with variance Σ_ε, where $\Sigma_\varepsilon \in \Re^{l\times l}$, the transition probability of the directly actuated part of the state is defined as:[3]

$$p\left(\mathbf{x}_{t_{j+1}}^{(c)}|\mathbf{x}_{t_j}\right) = \frac{1}{\left((2\pi)^l \cdot |\Sigma_{t_j}|\right)^{1/2}} \exp\left(-\frac{1}{2} \left\| \mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)} - \mathbf{f}_{t_j}^{(c)} dt \right\|_{\Sigma_{t_j}^{-1}}^2 \right), \tag{15}$$

2. The delta functions will all integrate to 1 in the path integral.

3. For notational simplicity, we write weighted square norms (or Mahalanobis distances) as $\mathbf{v}^T\mathbf{M}\mathbf{v} = \|\mathbf{v}\|_{\mathbf{M}}^2$.

where the covariance $\Sigma_{t_j} \in \Re^{l \times l}$ is expressed as $\Sigma_{t_j} = \mathbf{G}_{t_j}^{(c)} \Sigma_{\boldsymbol{\varepsilon}} \mathbf{G}_{t_j}^{(c)^T} dt$. Combining (15) and (14) results in the probability of a path expressed as:

$$p(\tau_i|\mathbf{x}_{t_i}) \propto \frac{1}{\Pi_{j=i}^{N-1} \left((2\pi)^l \|\Sigma_{t_j}|\right)^{1/2}} \exp\left(-\frac{1}{2}\sum_{j=1}^{N-1} \left\| \mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)} - \mathbf{f}_{t_j}^{(c)} dt \right\|_{\Sigma_{t_j}^{-1}}^2\right).$$

Finally, we incorporate the assumption (8) about the relation between the control cost and the variance of the noise, which needs to be adjusted to the controlled space as $\Sigma_{t_j} = \mathbf{G}_{t_j}^{(c)} \Sigma_{\boldsymbol{\varepsilon}} \mathbf{G}_{t_j}^{(c)^T} dt = \lambda \mathbf{G}_{t_j}^{(c)} \mathbf{R}^{-1} \mathbf{G}_{t_j}^{(c)^T} dt = \lambda \mathbf{H}_{t_j} dt$ with $\mathbf{H}_{t_j} = \mathbf{G}_{t_j}^{(c)} \mathbf{R}^{-1} \mathbf{G}_{t_j}^{(c)^T}$. Thus, we obtain:

$$p(\tau_i|\mathbf{x}_{t_i}) \propto \frac{1}{\Pi_{j=i}^{N-1} \left((2\pi)^l |\Sigma_{t_j}|\right)^{1/2}} \exp\left(-\frac{1}{2\lambda}\sum_{j=i}^{N-1} \left\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \right\|_{\mathbf{H}_{t_j}^{-1}}^2 dt\right).$$

With this formulation of the probability of a trajectory, we can rewrite the the path integral (11) as:

$$\begin{aligned}
\Psi_{t_i} &= \lim_{dt\to 0} \int \frac{\exp\left(-\frac{1}{\lambda}\left(\phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2}\sum_{j=i}^{N-1} \left\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \right\|_{\mathbf{H}_{t_j}^{-1}}^2 dt\right)\right)}{\Pi_{j=i}^{N-1} \left((2\pi)^{l/2} |\Sigma_{t_j}|^{1/2}\right)} d\tau_i^{(c)} \\
&= \lim_{dt\to 0} \int \frac{1}{D(\tau_i)} \exp\left(-\frac{1}{\lambda} S(\tau_i)\right) d\tau_i^{(c)}, \qquad (16)
\end{aligned}$$

where, we defined

$$S(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2}\sum_{j=i}^{N-1} \left\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \right\|_{\mathbf{H}_{t_j}^{-1}}^2 dt,$$

and

$$D(\tau_i) = \Pi_{j=i}^{N-1} \left((2\pi)^{l/2} |\Sigma_{t_j}|^{1/2}\right).$$

Note that the integration is over $d\tau_i^{(c)} = \left(d\mathbf{x}_{t_i}^{(c)}, ..., d\mathbf{x}_{t_N}^{(c)}\right)$, as the non-directly actuated states can be integrated out due to the fact that the state transition of the non-directly actuated states is deterministic, and just added Dirac delta functions in the integral (cf. Equation (13)). Equation (16) is written in a more compact form as:

$$\begin{aligned}
\Psi_{t_i} &= \lim_{dt\to 0} \int \exp\left(-\frac{1}{\lambda} S(\tau_i) - \log D(\tau_i)\right) d\tau_i^{(c)} \\
&= \lim_{dt\to 0} \int \exp\left(-\frac{1}{\lambda} Z(\tau_i)\right) d\tau_i^{(c)}, \qquad (17)
\end{aligned}$$

where $Z(\tau_i) = S(\tau_i) + \lambda \log D(\tau_i)$. It can be shown that this term is factorized in path dependent and path independent terms of the form:

$$Z(\tau_i) = \tilde{S}(\tau_i) + \frac{\lambda(N-i)l}{2}\log(2\pi dt\lambda),$$

where $\tilde{S}(\tau_i) = S(\tau_i) + \frac{\lambda}{2}\sum_{j=i}^{N-1}\log|\mathbf{H}_{t_j}|$. This formula is a required step for the derivation of optimal controls in the next section. The constant term $\frac{\lambda(N-i)l}{2}\log(2\pi dt\lambda)$ can be the source of numerical instabilities especially in cases where fine discretization dt of stochastic dynamics is required. However, in the next section, and in a great detail in Appendix A, lemma 1, we show how this term drops out of the equations.

2.4 Optimal Controls

For every moment of time, the optimal controls are given as $\mathbf{u}_{t_i} = -\mathbf{R}^{-1}\mathbf{G}_{t_i}^T(\nabla_{x_{t_i}}V_{t_i})$. Due to the exponential transformation of the value function, the equation of the optimal controls can be written as

$$\mathbf{u}_{t_i} = \lambda\mathbf{R}^{-1}\mathbf{G}_{t_i}\frac{\nabla_{\mathbf{x}_{t_i}}\Psi_{t_i}}{\Psi_{t_i}}.$$

After substituting Ψ_{t_i} with (17) and canceling the state independent terms of the cost we have:

$$\mathbf{u}_{t_i} = \lim_{dt\to 0}\left(\lambda\mathbf{R}^{-1}\mathbf{G}_{t_i}^T\frac{\nabla_{\mathbf{x}_{t_i}^{(c)}}\left(\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}d\tau_i^{(c)}\right)}{\int\ e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}d\tau_i^{(c)}}\right),$$

Further analysis of the equation above leads to a simplified version for the optimal controls as

$$\boxed{\mathbf{u}_{t_i} = \int P(\tau_i)\,\mathbf{u}_L(\tau_i)\,d\tau_i^{(c)}}, \tag{18}$$

with the probability $P(\tau_i)$ and local controls $\mathbf{u}_L(\tau_i)$ defined as

$$\boxed{P(\tau_i) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}d\tau_i}} \tag{19}$$

$$\mathbf{u}_L(\tau_i) = -\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\lim_{dt\to 0}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right).$$

The path cost $\tilde{S}(\tau_i)$ is a generalized version of the path cost in Kappen (2005a) and Kappen (2007), which only considered systems with state independent control transition[4] $\mathbf{G}_{t_i}$. To find the local controls $\mathbf{u}_L(\tau_i)$ we have to calculate the $\lim_{dt\to 0}\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)$. Appendix A and more precisely lemma 2 shows in detail the derivation of the final result:

$$\lim_{dt\to 0}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right) = -\mathbf{H}_{t_i}^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right),$$

where the new term $\mathbf{b}_{t_i}$ is expressed as $\mathbf{b}_{t_i} = \lambda\mathbf{H}_{t_i}\Phi_{t_i}$ and $\Phi_{t_i} \in \Re^{l\times 1}$ is a vector with the j^{th} element defined as:

$$(\Phi_{t_i})_j = \frac{1}{2}trace\left(\mathbf{H}_{t_i}^{-1}\left(\partial_{[\mathbf{x}_{t_i}^{(c)}]_j}\mathbf{H}_{t_i}\right)\right).$$

4. More precisely if $\mathbf{G}_{t_i}^{(c)} = \mathbf{G}^{(c)}$ then the term $\frac{\lambda}{2}\sum_{j=i}^{N-1}\log|\mathbf{H}_{t_j}|$ disappears since it is state independent and it appears in both nominator and denominator in (19). In this case, the path cost is reduced to $\tilde{S}(\tau_i) = S(\tau_i)$.

The local control can now be expressed as:

$$\mathbf{u}_L(\tau_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\mathbf{H}_{t_i}^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right),$$

By substituting $\mathbf{H}_{t_i} = \mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}$ in the equation above, we get our main result for the local controls of the sampled path for the generalized path integral formulation:

$$\boxed{\mathbf{u}_L(\tau_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\left(\mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\right)^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right)}. \tag{20}$$

The equations in boxes (18), (19) and (20) form the solution for the generalized path integral stochastic optimal control problem. Given that this result is of general value and constitutes the foundation to derive our reinforcement learning algorithm in the next section, but also since many other special cases can be derived from it, we summarized all relevant equations in Table 1.

The **Given** components of Table 1 include a model of the system dynamics, the cost function, knowledge of the system's noise process, and a mechanism to generate trajectories τ_i. It is important to realize that this is a *model-based* approach, as the computations of the optimal controls requires knowledge of ε_i. ε_i can be obtained in two ways. First, the trajectories τ_i can be generated purely in simulation, where the noise is generated from a random number generator. Second, trajectories could be generated by a real system, and the noise ε_i would be computed from the difference between the actual and the predicted system behavior, that is, $\mathbf{G}_{t_i}^{(c)}\varepsilon_i = \dot{\mathbf{x}}_{t_i} - \hat{\dot{\mathbf{x}}}_{t_i} = \dot{\mathbf{x}}_{t_i} - (\mathbf{f}_{t_i} + \mathbf{G}_{t_i}\mathbf{u}_{t_i})$. Computing the prediction $\hat{\dot{\mathbf{x}}}_{t_i}$ also requires a model of the system dynamics.

Previous results in Kappen (2005a), Kappen (2007), Kappen (2005b) and Broek et al. (2008) are special cases of our generalized formulation. In the next section we show how our generalized formulation is specialized to different classes of stochastic dynamical systems and we provide the corresponding formula of local controls for each class.

2.5 Special Cases

The purpose of this section is twofold. First, it demonstrates how to apply the path integral approach to specialized forms of dynamical systems, and how the local controls in (20) simplify for these cases. Second, this section prepares the special case which we will need for our reinforcement learning algorithm in Section 3.

2.5.1 SYSTEMS WITH ONE DIMENSIONAL DIRECTLY ACTUATED STATE

The generalized formulation of stochastic optimal control with path integrals in Table 1 can be applied to a variety of stochastic dynamical systems with different types of control transition matrices. One case of particular interest is where the dimensionality of the directly actuated part of the state is 1D, while the dimensionality of the control vector is 1D or higher dimensional. As will be seen below, this situation arises when the controls are generated by a linearly parameterized function approximator. The control transition matrix thus becomes a row vector $\mathbf{G}_{t_i}^{(c)} = \mathbf{g}_{t_i}^{(c)T} \in \Re^{1\times p}$. According to (20), the local controls for such systems are expressed as follows:

$$\mathbf{u}_L(\tau_i) = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}^{(c)}}{\mathbf{g}_{t_i}^{(c)T}\mathbf{R}^{-1}\mathbf{g}_{t_i}^{(c)}}\left(\mathbf{g}_{t_i}^{(c)T}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right).$$

- **Given**:
  - The system dynamics $\dot{\mathbf{x}}_t = \mathbf{f}_t + \mathbf{G}_t\left(\mathbf{u}_t + \boldsymbol{\varepsilon}_t\right)$ (cf. 3)
  - The immediate cost $r_t = q_t + \frac{1}{2}\mathbf{u}_t^T \mathbf{R}\mathbf{u}_t$ (cf. 4)
  - A terminal cost term ϕ_{t_N} (cf. 1)
  - The variance $\Sigma_{\boldsymbol{\varepsilon}}$ of the mean-zero noise $\boldsymbol{\varepsilon}_t$
  - Trajectory starting at t_i and ending at t_N: $\boldsymbol{\tau}_i = (\mathbf{x}_{t_i}, ..., \mathbf{x}_{t_N})$
  - A partitioning of the system dynamics into (c) controlled and (m) uncontrolled equations, where $n = c + m$ is the dimensionality of the state $\mathbf{x}_t$ (cf. Section 2.3)
- **Optimal Controls**:
  - Optimal controls at every time step t_i: $\mathbf{u}_{t_i} = \int P\left(\boldsymbol{\tau}_i\right)\mathbf{u}\left(\boldsymbol{\tau}_i\right) d\boldsymbol{\tau}_i^{(c)}$
  - Probability of a trajectory: $P\left(\boldsymbol{\tau}_i\right) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)} d\boldsymbol{\tau}_i}$
  - Generalized trajectory cost: $\tilde{S}(\boldsymbol{\tau}_i) = S(\boldsymbol{\tau}_i) + \frac{\lambda}{2}\sum_{j=i}^{N-1} \log|\mathbf{H}_{t_j}|$ where
    - $S(\boldsymbol{\tau}_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2}\sum_{j=i}^{N-1} \left\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \right\|_{\mathbf{H}_{t_j}^{-1}}^2 dt$
    - $\mathbf{H}_{t_j} = \mathbf{G}_{t_j}^{(c)} \mathbf{R}^{-1} \mathbf{G}_{t_j}^{(c)T}$
  - Local Controls: $\mathbf{u}_L(\boldsymbol{\tau}_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\left(\mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\right)^{-1}\left(\mathbf{G}_{t_i}^{(c)}\boldsymbol{\varepsilon}_{t_i} - \mathbf{b}_{t_i}\right)$ where
    - $\mathbf{b}_{t_i} = \lambda \mathbf{H}_{t_i}\Phi_{t_i}$
    - $[\Phi_{t_i}]_j = \frac{1}{2} trace\left(\mathbf{H}_{t_i}^{-1}\left(\partial_{[\mathbf{x}_{t_i}^{(c)}]_j}\mathbf{H}_{t_i}\right)\right)$

Table 1: Summary of optimal control derived from the path integral formalizm.

Since the directly actuated part of the state is 1D, the vector $\mathbf{x}_{t_i}^{(c)}$ collapses into the scalar $x_{t_i}^{(c)}$ which appears in the partial differentiation above. In the case that $\mathbf{g}_{t_i}^{(c)}$ does not depend on $x_{t_i}^{(c)}$, the differentiation with respect to $x_{t_i}^{(c)}$ results to zero and the the local controls simplify to:

$$\mathbf{u}_L(\boldsymbol{\tau}_i) = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}^{(c)}\mathbf{g}_{t_i}^{(c)T}}{\mathbf{g}_{t_i}^{(c)T}\mathbf{R}^{-1}\mathbf{g}_{t_i}^{(c)}}\boldsymbol{\varepsilon}_{t_i}.$$

2.5.2 SYSTEMS WITH PARTIALLY ACTUATED STATE

The generalized formula of the local controls (20) was derived for the case where the control transition matrix is state dependent and its dimensionality is $\mathbf{G}_t^{(c)} \in \Re^{l \times p}$ with $l < n$ and p the dimensionality of the control. There are many special cases of stochastic dynamical systems in optimal control

and robotic applications that belong into this general class. More precisely, for systems having a state dependent control transition matrix that is square ($\mathbf{G}_{t_i}^{(c)} \in \Re^{l \times l}$ with $l = p$) the local controls based on (20) are reformulated as:

$$\mathbf{u}_L(\tau_i) = \varepsilon_{t_i} - \mathbf{G}_{t_i}^{(c)^{-1}} \mathbf{b}_{t_i}. \tag{21}$$

Interestingly, a rather general class of mechanical systems such as rigid-body and multi-body dynamics falls into this category. When these mechanical systems are expressed in state space formulation, the control transition matrix is equal to rigid body inertia matrix $\mathbf{G}_{t_i}^{(c)} = \mathbf{M}(\theta_{t_i})$ (Sciavicco and Siciliano, 2000). Future work will address this special topic of path integral control for multi-body dynamics.

Another special case of systems with partially actuated state is when the control transition matrix is state independent and has dimensionality $\mathbf{G}_t^{(c)} = \mathbf{G}^{(c)} \in \Re^{l \times p}$. The local controls, according to (20), become:

$$\mathbf{u}_L(\tau_i) = \mathbf{R}^{-1}\mathbf{G}^{(c)^T} \left(\mathbf{G}^{(c)}\mathbf{R}^{-1}\mathbf{G}^{(c)^T}\right)^{-1} \mathbf{G}^{(c)}\varepsilon_{t_i}. \tag{22}$$

If $\mathbf{G}_{t_i}^{(c)}$ is square and state independent, $\mathbf{G}_{t_i}^{(c)} = \mathbf{G}^{(c)} \in \Re^{l \times l}$, we will have:

$$\mathbf{u}_L(\tau_i) = \varepsilon_{t_i}. \tag{23}$$

This special case was explored in Kappen (2005a), Kappen (2007), Kappen (2005b) and Broek et al. (2008). Our generalized formulation allows a broader application of path integral control in areas like robotics and other control systems, where the control transition matrix is typically partitioned into directly and non-directly actuated states, and typically also state dependent.

2.5.3 SYSTEMS WITH FULLY ACTUATED STATE SPACE

In this class of stochastic systems, the control transition matrix is not partitioned and, therefore, the control $\mathbf{u}$ directly affects all the states. The local controls for such systems are provided by simply substituting $\mathbf{G}_{t_i}^{(c)} \in \Re^{n \times p}$ in (20) with $\mathbf{G}_{t_i} \in \Re^{n \times n}$. Since $\mathbf{G}_{t_i}$ is a square matrix we obtain:

$$\mathbf{u}_L(\tau_i) = \varepsilon_{t_i} - \mathbf{G}_{t_i}^{-1}\mathbf{b}_{t_i},$$

with $\mathbf{b}_{t_i} = \lambda \mathbf{H}_{t_i}\Phi_{t_i}$ and

$$(\Phi_{t_i})_j = \frac{1}{2} trace\left(\mathbf{H}_{t_i}^{-1}\left(\partial_{(\mathbf{x}_{t_i})_j}\mathbf{H}_{t_i}\right)\right),$$

where the differentiation is not taken with respect to $(\mathbf{x}_{t_i}^{(c)})_j$ but with respect to the full state $(\mathbf{x}_{t_i})_j$. For this fully actuated state space, there are subclasses of dynamical systems with square and/or state independent control transition matrix. The local controls for these cases are found by just substituting $\mathbf{G}_{t_i}^{(c)}$ with $\mathbf{G}_{t_i}$ in (21), (22) and (23).

3. Reinforcement Learning with Parameterized Policies

Equipped with the theoretical framework of stochastic optimal control with path integrals, we can now turn to its application to reinforcement learning with parameterized policies. Since the beginning of actor-critic algorithms (Barto et al., 1983), one goal of reinforcement learning has been

to learn compact policy representations, for example, with neural networks as in the early days of machine learning (Miller et al., 1990), or with general parameterizations (Peters, 2007; Deisenroth et al., 2009). Parameterized policies have much fewer parameters than the classical time-indexed approach of optimal control, where every time step has it own set of parameters, that is, the optimal controls at this time step. Usually, function approximation techniques are used to represent the optimal controls and the open parameters of the function approximator become the policy parameters. Function approximators use a state representation as input and not an explicit time dependent representation. This representation allows generalization across states and promises to achieve better generalization of the control policy to a larger state space, such that policies become re-usable and do not have to be recomputed in every new situation.

The path integral approach from the previous sections also follows the classical time-based optimal control strategy, as can be seen from the time dependent solution for optimal controls in (33). However, a minor re-interpretation of the approach and some small mathematical adjustments allow us to carry it over to parameterized policies and reinforcement learning, which results in a new algorithm called **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}).

3.1 Parameterized Policies

We are focusing on direct policy learning, where the parameters of the policy are adjusted by a learning rule directly, and not indirectly as in value function approaches of classical reinforcement learning (Sutton and Barto, 1998)—see Peters (2007) for a discussion of pros and cons of direct vs. indirect policy learning. Direct policy learning usually assumes a general cost function (Sutton et al., 2000; Peters, 2007) in the form of

$$J(\mathbf{x}_0) = \int p(\tau_0) R(\tau_0) d\tau_0, \tag{24}$$

which is optimized over states-action trajectories[5] $\tau_0 = (\mathbf{x}_{t_0}, \mathbf{a}_{t_0}, ..., \mathbf{x}_{t_N})$. Under the first order Markov property, the probability of a trajectory is

$$p(\tau_i) = p(\mathbf{x}_{t_i}) \Pi_{j=i}^{N-1} p(\mathbf{x}_{t_{j+1}} | \mathbf{x}_{t_j}, \mathbf{a}_{t_j}) p(\mathbf{a}_{t_j} | \mathbf{x}_{t_j}).$$

Both the state transition and the policy are assumed to be stochastic. The particular formulation of the stochastic policy is a design parameter, motivated by the application domain, analytical convenience, and the need to inject exploration during learning. For continuous state action domains, Gaussian distributions are most commonly chosen (Gullapalli, 1990; Williams, 1992; Peters, 2007). An interesting generalized stochastic policy was suggested in Rueckstiess et al. (2008) and applied in Koeber and Peters (2008), where the stochastic policy $p(\mathbf{a}_{t_i} | \mathbf{x}_{t_i})$ is linearly parameterized as:

$$\mathbf{a}_{t_i} = \mathbf{g}_{t_i}^T (\theta + \varepsilon_{t_i}), \tag{25}$$

with $\mathbf{g}_{t_i}$ denoting a vector of basis functions and θ the parameter vector. This policy has state dependent noise, which can contribute to faster learning as the signal-to-noise ratio becomes adaptive since it is a function of $\mathbf{g}_{t_i}$. It should be noted that a standard additive-noise policy can be expressed in this formulation, too, by choosing one basis function $(\mathbf{g}_{t_i})_j = 0$. For Gaussian noise ε the probability of an action is $p(\mathbf{a}_{t_i} | \mathbf{x}_{t_i}) = N\left(\theta^T \mathbf{g}_{t_i}, \Sigma_{t_i}\right)$ with $\Sigma_{t_i} = \mathbf{g}_{t_i}^T \Sigma_\varepsilon \mathbf{g}_{t_i}$. Comparing the policy formulation

5. We use $\mathbf{a}_t$ to denote actions here in order to avoid using the symbol $\mathbf{u}$ in a conflicting way in the equations below, and to emphasize that an action does not necessarily coincide with the control command to a physical system.

in (25) with the control term in (3), one recognizes that the control policy formulation (25) should fit into the framework of path integral optimal control.

3.2 Generalized Parameterized Policies

Before going into more detail of our proposed reinforcement learning algorithm, it is worthwhile contemplating what the action $\mathbf{a}_t$ actually represents. In many applications of stochastic optimal control there are three main problems to be considered: i) trajectory planning, ii) feedforward control, and iii) feedback control. The results of optimization could thus be an optimal kinematic trajectory, the corresponding feedforward commands to track the desired trajectory accurately in face of the system's nonlinearities, and/or time varying linear feedback gains (gain scheduling) for a negative feedback controller that compensates for perturbations from accurate trajectory tracking.

There are very few optimal control algorithms which compute all three issues simultaneously, such as Differential Dynamic Programming(DDP) (Jacobson and Mayne, 1970), or its simpler version the Iterative Linear Quadratic Regulator(iLQR) (Todorov, 2005). However, these are model based methods which require rather accurate knowledge of the dynamics and make restrictive assumptions concerning differentiability of the system dynamics and the cost function.

Path integral optimal control allows more flexibility than these related methods. The concept of an "action" can be viewed in a broader sense. Essentially, we consider any "input" to the control system as an action, not unlike the inputs to a transfer function in classical linear control theory. The input can be a motor command, but it can also be anything else, for instance, a desired state, that is subsequently converted to a motor command by some tracking controller, or a control gain (Buchli et al., 2010) . As an example, consider a robotic system with rigid body dynamics (RBD) equations (Sciavicco and Siciliano, 2000) using a parameterized policy:

$$\ddot{\mathbf{q}} = \mathbf{M}(\mathbf{q})^{-1}\left(-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})-\mathbf{v}(\mathbf{q})\right)+\mathbf{M}(\mathbf{q})^{-1}\mathbf{u}, \tag{26}$$

$$\mathbf{u} = \mathbf{G}(\mathbf{q})(\theta+\varepsilon_{t_i}), \tag{27}$$

where $\mathbf{M}$ is the RBD inertia matrix, $\mathbf{C}$ are Coriolis and centripetal forces, and $\mathbf{v}$ denotes gravity forces. The state of the robot is described by the joint angles $\mathbf{q}$ and joint velocities $\dot{\mathbf{q}}$. The policy (27) is linearly parameterized by θ, with basis function matrix $\mathbf{G}$—one would assume that the dimensionality of θ is significantly larger than that of $\mathbf{q}$ to assure sufficient expressive power of this parameterized policy. Inserting (27) into (26) results in a differential equation that is compatible with the system equations (3) for path integral optimal control:

$$\begin{aligned} \ddot{\mathbf{q}} &= \mathbf{f}(\mathbf{q},\dot{\mathbf{q}})+\tilde{\mathbf{G}}(\mathbf{q})(\theta+\varepsilon_{t_i}) \\ &\textit{where} \\ \mathbf{f}(\mathbf{q},\dot{\mathbf{q}}) &= \mathbf{M}(\mathbf{q})^{-1}\left(-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})-\mathbf{v}(\mathbf{q})\right), \\ \tilde{\mathbf{G}}(\mathbf{q}) &= \mathbf{M}(\mathbf{q})^{-1}\mathbf{G}(\mathbf{q}). \end{aligned} \tag{28}$$

This example is a typical example where the policy directly represents motor commands.

Alternatively, we could create another form of control structure for the RBD system:

$$\begin{aligned} \ddot{\mathbf{q}} &= \mathbf{M}(\mathbf{q})^{-1}\left(-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})-\mathbf{v}(\mathbf{q})\right)+\mathbf{M}(\mathbf{q})^{-1}\mathbf{u}, \\ \mathbf{u} &= \mathbf{K}_P(\mathbf{q}_d-\mathbf{q})+\mathbf{K}_D(\dot{\mathbf{q}}_d-\dot{\mathbf{q}}), \\ \ddot{\mathbf{q}}_d &= \mathbf{G}(\mathbf{q}_d,\dot{\mathbf{q}}_d)(\theta+\varepsilon_{t_i}). \end{aligned} \tag{29}$$

Here, a Proportional-Derivative (PD) controller with positive definite gain matrices $\mathbf{K}_P$ and $\mathbf{K}_D$ converts a desired trajectory $\mathbf{q}_d, \dot{\mathbf{q}}_d$ into a motor command $\mathbf{u}$. In contrast to the previous example, the parameterized policy generates the desired trajectory in (29), and the differential equation for the desired trajectory is compatible with the path integral formalism.

What we would like to emphasize is that the control system's structure is left to the creativity of its designer, and that path integral optimal control can be applied on various levels. Importantly, as developed in Section 2.3, only the *controlled* differential equations of the entire control system contribute to the path integral formalism, that is, (28) in the first example, or (29) in the second example. And *only these controlled differential equations* need to be known for applying path integral optimal control—none of the variables of the uncontrolled equations is ever used.

At this point, we make a very important transition from model-based to model-free learning. In the example of (28), the dynamics model of the control system needs to be known to apply path integral optimal control, as this is a *controlled* differential equation. In contrast, in (29), the system dynamics are in an *uncontrolled* differential equation, and are thus irrelevant for applying path integral optimal control. In this case, only knowledge of the desired trajectory dynamics is needed, which is usually created by the system designer. Thus, we obtained a model-free learning system.

3.3 Dynamic Movement Primitives as Generalized Policies

As we are interested in model-free learning, we follow the control structure of the 2^{nd} example of the previous section, that is, we optimize control policies which represent desired trajectories. We use Dynamic Movement Primitives (DMPs) (Ijspeert et al., 2003) as a special case of parameterized policies, which are expressed by the differential equations:

$$\begin{aligned} \frac{1}{\tau}\dot{z}_t &= f_t + \mathbf{g}_t^T(\theta+\varepsilon_t), \\ \frac{1}{\tau}\dot{y}_t &= z_t, \\ \frac{1}{\tau}\dot{x}_t &= -\alpha x_t, \\ f_t &= \alpha_z(\beta_z(g-y_t)-z_t). \end{aligned} \tag{30}$$

Essentially, these policies code a learnable point attractor for a movement from y_{t_0} to the goal g, where θ determines the shape of the attractor. $y_t, \dot{y}_t$ denote the position and velocity of the trajectory, while z_t, x_t are internal states. α_z, β_z, τ are time constants. The basis functions $\mathbf{g}_t \in \Re^{p\times 1}$are defined by a piecewise linear function approximator with Gaussian weighting kernels, as suggested in Schaal and Atkeson (1998):

$$\begin{aligned} [\mathbf{g}_t]_j &= \frac{w_j\, x_t}{\sum_{k=1}^{p} w_k}(g-y_0), \\ w_j &= \exp\left(-0.5h_j(x_t-c_j)^2\right), \end{aligned} \tag{31}$$

with bandwith h_j and center c_j of the Gaussian kernels—for more details see Ijspeert et al. (2003). The DMP representation is advantageous as it guarantees attractor properties towards the goal while remaining linear in the parameters θ of the function approximator. By varying the parameter θ the shape of the trajectory changes while the goal state g and initial state y_{t_0} remain fixed. These properties facilitate learning (Peters and Schaal, 2008a).

3.4 Policy Improvements with Path Integrals: The (PI^2) Algorithm

As can be easily recognized, the DMP equations are of the form of our control system (3), with only one controlled equation and a one dimensional actuated state. This case has been treated in Section 2.5.1. The motor commands are replaced with the parameters θ—the issue of time dependent vs. constant parameters will be addressed below. More precisely, the DMP equations can be written as:

$$\begin{pmatrix} \dot{x}_t \\ \dot{z}_t \\ \dot{y}_t \end{pmatrix} = \begin{pmatrix} -\alpha x_t \\ y_t \\ \alpha_z(\beta_z(g-y_t)-z_t) \end{pmatrix} + \begin{pmatrix} 0_{1\times p} \\ 0_{1\times p} \\ \mathbf{g}_t^{(c)T} \end{pmatrix} (\theta_t + \varepsilon_t).$$

The state of the DMP is partitioned into the controlled part $\mathbf{x}_t^{(c)} = y_t$ and uncontrolled part $\mathbf{x}_t^{(m)} = (x_t \; z_t)^T$. The control transition matrix depends on the state, however, it depends only on one of the state variables of the uncontrolled part of the state, that is, x_t. The path cost for the stochastic dynamics of the DMPs is given by:

$$\begin{aligned}
\tilde{S}(\tau_i) &= \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2}\sum_{j=i}^{N-1} \left\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \right\|^2_{\mathbf{H}_{t_j}^{-1}} dt + \frac{\lambda}{2}\sum_{j=i}^{N-1} \log|\mathbf{H}_{t_j}| \\
&\propto \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} \left\| \mathbf{g}_{t_j}^{(c)T}(\theta_{t_j} + \varepsilon_{t_j}) \right\|^2_{\mathbf{H}_{t_j}^{-1}} \\
&= \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} \frac{1}{2}(\theta_{t_j} + \varepsilon_{t_j})^T \mathbf{g}_{t_j}^{(c)} \mathbf{H}_{t_j}^{-1} \mathbf{g}_{t_j}^{(c)T}(\theta_{t_j} + \varepsilon_{t_j}) \\
&= \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} \frac{1}{2}(\theta_{t_j} + \varepsilon_{t_j})^T \frac{\mathbf{g}_{t_j}^{(c)} \mathbf{g}_{t_j}^{(c)T}}{\mathbf{g}_t^{(c)T} \mathbf{R}^{-1} \mathbf{g}_t^{(c)}}(\theta_{t_j} + \varepsilon_{t_j}) \\
&= \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} \frac{1}{2}(\theta_{t_j} + \varepsilon_{t_j})^T \mathbf{M}_{t_j}^T \mathbf{R} \mathbf{M}_{t_j}(\theta_{t_j} + \varepsilon_{t_j}).
\end{aligned} \tag{32}$$

with $\mathbf{M}_{t_j} = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_j}\mathbf{g}_{t_j}^T}{\mathbf{g}_{t_j}^T \mathbf{R}^{-1} \mathbf{g}_{t_j}}$. $\mathbf{H}_t$ becomes a scalar given by $\mathbf{H}_t = \mathbf{g}_t^{(c)T} \mathbf{R}^{-1} \mathbf{g}_t^{(c)}$. Interestingly, the term $\frac{\lambda}{2}\sum_{j=i}^{N-1} \log|\mathbf{H}_{t_j}|$ for the case of DMPs depends only on x_t, which is a deterministic variable and therefore can be ignored since it is the same for all sampled paths. We also absorbed, without loss of generality, the time step dt in cost terms. Consequently, the fundamental result of the path integral stochastic optimal problem for the case of DMPs is expressed as:

$$\boxed{\mathbf{u}_{t_i} = \int P(\tau_i)\, \mathbf{u}_L(\tau_i)\, d\tau_i^{(c)}}, \tag{33}$$

where the probability $P(\tau_i)$ and local controls $\mathbf{u}(\tau_i)$ are defined as

$$P(\tau_i) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)} d\tau_i}, \quad \mathbf{u}_L(\tau_i) = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}^{(c)} \mathbf{g}_{t_i}^{(c)T}}{\mathbf{g}_{t_i}^{(c)T} \mathbf{R}^{-1} \mathbf{g}_{t_i}^{(c)}} \varepsilon_{t_i},$$

and the path cost given as

$$\tilde{S}(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} \varepsilon_{t_j}^T \mathbf{M}_{t_j}^T \mathbf{R} \mathbf{M}_{t_j} \varepsilon_{t_j}.$$

Note that $\theta = 0$ in these equations, that is, the parameters are initialized to zero. These equations correspond to the case where the stochastic optimal control problem is solved with one evaluation of the optimal controls (33) using dense sampling of the whole state space under the "passive dynamics" (i.e., $\theta = 0$), which requires a significant amount of exploration noise. Such an approach was pursued in the original work by Kappen (2007) and Broek et al. (2008), where a potentially large number of sample trajectories was needed to achieve good results. Extending this sampling approach to high dimensional spaces, however, is daunting, as with very high probability, we would sample primarily rather useless trajectories. Thus, biasing sampling towards good initial conditions seems to be mandatory for high dimensional applications.

Thus, we consider only local sampling and an iterative update procedure. Given a current guess of θ, we generate sample roll-outs using stochastic parameters $\theta + \varepsilon_t$ at every time step. To see how the generalized path integral formulation is modified for the case of iterative updating, we start with the equations of the update of the parameter vector θ, which can be written as:

$$\begin{aligned}
\theta_{t_i}^{(new)} &= \int P(\tau_i) \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T(\theta + \varepsilon_{t_i})}{\mathbf{g}_{t_i}^T\mathbf{R}^{-1}\mathbf{g}_{t_i}} d\tau_i \\
&= \int P(\tau_i) \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T\varepsilon_{t_i}}{\mathbf{g}_{t_i}^T\mathbf{R}^{-1}\mathbf{g}_{t_i}} d\tau_i + \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T\theta}{\mathbf{g}_{t_i}^T\mathbf{R}^{-1}\mathbf{g}_{t_i}} \\
&= \delta\theta_{t_i} + \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T}{trace(\mathbf{R}^{-1}\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T)}\theta \\
&= \delta\theta_{t_i} + \mathbf{M}_{t_i}\theta. \qquad (34)
\end{aligned}$$

The correction parameter vector $\delta\theta_{t_i}$ is defined as $\delta\theta_{t_i} = \int P(\tau_i) \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T\varepsilon_{t_i}}{\mathbf{g}_{t_i}^T\mathbf{R}^{-1}\mathbf{g}_{t_i}} d\tau_i$. It is important to note that $\theta_{t_i}^{(new)}$ is now time dependent, that is, for every time step t_i, a different optimal parameter vector is computed. In order to return to one single time independent parameter vector $\theta^{(new)}$, the vectors $\theta_{t_i}^{(new)}$ need to be averaged over time t_i.

We start with a first tentative suggestion of averaging over time, and then explain why it is inappropriate, and what the correct way of time averaging has to look like. The tentative and most intuitive time average is:

$$\theta^{(new)} = \frac{1}{N}\sum_{i=0}^{N-1} \theta_{t_i}^{(new)} = \frac{1}{N}\sum_{i=0}^{N-1} \delta\theta_{t_i} + \frac{1}{N}\sum_{i=0}^{N-1} \mathbf{M}_{t_i}\theta.$$

Thus, we would update θ based on two terms. The first term is the average of $\delta\theta_{t_i}$, which is reasonable as it reflects the knowledge we gained from the exploration noise. However, there would be a second update term due to the average over projected mean parameters θ from every time step—it should be noted that $\mathbf{M}_{t_i}$ is a projection matrix onto the range space of $\mathbf{g}_{t_i}$ under the metric $\mathbf{R}^{-1}$, such that a multiplication with $\mathbf{M}_{t_i}$ can only shrink the norm of θ. From the viewpoint of having optimal parameters for *every time step*, this update component is reasonable as it trivially eliminates the part of the parameter vector that lies in the null space of $\mathbf{g}_{t_i}$ and which contributes to the command cost

of a trajectory in a useless way. From the view point of a parameter vector that is *constant and time independent* and that is updated *iteratively*, this second update is undesirable, as the multiplication of the parameter vector θ with $\mathbf{M}_{t_i}$ in (34) and the averaging operation over the time horizon reduces the L_2 norm of the parameters at every iteration, potentially in an uncontrolled way.[6] What we rather want is to achieve convergence when the average of $\delta\theta_{t_i}$ becomes zero, and we do not want to continue updating due to the second term.

The problem is avoided by eliminating the projection matrix in the second term of averaging, such that it become:

$$\theta^{(new)} = \frac{1}{N}\sum_{i=0}^{N-1}\delta\theta_{t_i} + \frac{1}{N}\sum_{i=0}^{N-1}\theta = \frac{1}{N}\sum_{i=0}^{N-1}\delta\theta_{t_i} + \theta.$$

The meaning of this reduced update is simply that we keep a component in θ that is irrelevant and contributes to our trajectory cost in a useless way. However, this irrelevant component will not prevent us from reaching the optimal effective solution, that is, the solution that lies in the range space of $\mathbf{g}_{t_i}$. Given this modified update, it is, however, also necessary to derive a compatible cost function. As mentioned before, in the unmodified scenario, the last term of (32) is:

$$\frac{1}{2}\sum_{j=i}^{N-1}(\theta+\varepsilon_{t_j})^T\mathbf{M}_{t_j}^T\mathbf{R}\mathbf{M}_{t_j}(\theta+\varepsilon_{t_j})$$

To avoid a projection of θ, we modify this cost term to be:

$$\frac{1}{2}\sum_{j=i}^{N-1}(\theta+\mathbf{M}_{t_j}\varepsilon_{t_j})^T\mathbf{R}(\theta+\mathbf{M}_{t_j}\varepsilon_{t_j}).$$

With this modified cost term, the path integral formalism results in the desired $\theta_{t_i}^{(new)}$ without the $\mathbf{M}_{t_i}$ projection of θ.

The main equations of the iterative version of the generalized path integral formulation, called **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}), can be summarized as:

$$P(\tau_i) = \frac{e^{-\frac{1}{\lambda}S(\tau_i)}}{\int e^{-\frac{1}{\lambda}S(\tau_i)}d\tau_i}, \tag{35}$$

$$S(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j}dt + \frac{1}{2}\sum_{j=i}^{N-1}(\theta+\mathbf{M}_{t_j}\varepsilon_{t_j})^T\mathbf{R}(\theta+\mathbf{M}_{t_j}\varepsilon_{t_j})dt, \tag{36}$$

$$\delta\theta_{t_i} = \int P(\tau_i)\,\mathbf{M}_{t_i}\varepsilon_{t_i}d\tau_i, \tag{37}$$

$$[\delta\theta]_j = \frac{\sum_{i=0}^{N-1}(N-i)\;w_{j,t_i}\;[\delta\theta_{t_i}]_j}{\sum_{i=0}^{N-1}w_{j,t_i}(N-i)}, \tag{38}$$

$$\theta^{(new)} = \theta^{(old)} + \delta\theta.$$

Essentially, (35) computes a discrete probability at time t_i of each trajectory roll-out with the help of the cost (36). For every time step of the trajectory, a parameter update is computed in (37) based

6. To be precise, θ would be projected and continue shrinking until it lies in the intersection of all null spaces of the $\mathbf{g}_{t_i}$ basis function—this null space can easily be of measure zero.

on a probability weighted average over trajectories. The parameter updates at every time step are finally averaged in (38). Note that we chose a weighted average by giving every parameter update a weight[7] according to the time steps left in the trajectory and the activation of the kernel in (31). This average can be interpreted as using a function approximator with only a constant (offset) parameter vector to approximate the time dependent parameters. Giving early points in the trajectory a higher weight is useful since their parameters affect a large time horizon and thus higher trajectory costs. Other function approximation (or averaging) schemes could be used to arrive at a final parameter update—we preferred this simple approach as it gave very good learning results. The final parameter update is $\theta^{(new)} = \theta^{(old)} + \delta\theta$.

The parameter λ regulates the sensitivity of the exponentiated cost and can automatically be optimized for every time step i to maximally discriminate between the experienced trajectories. More precisely, a constant term can be subtracted from (36) as long as all $S(\tau_i)$ remain positive—this constant term[8] cancels in (35). Thus, for a given number of roll-outs, we compute the exponential term in (35) as

$$\exp\left(-\frac{1}{\lambda}S(\tau_i)\right) = \exp\left(-h\frac{S(\tau_i)-\min S(\tau_i)}{\max S(\tau_i)-\min S(\tau_i)}\right),$$

with h set to a constant, which we chose to be $h = 10$ in all our evaluations. The max and min operators are over all sample roll-outs. This procedure eliminates λ and leaves the variance of the exploration noise ε as the only open algorithmic parameter for **PI**^{2}. It should be noted that the equations for **PI**^{2} have no numerical pitfalls: no matrix inversions and no learning rates,[9] rendering **PI**^{2} to be very easy to use in practice.

The pseudocode for the final **PI**^{2} algorithm for a one dimensional control system with function approximation is given in Table 2. A tutorial Matlab example of applying **PI**^{2} can be found at http://www-clmc.usc.edu/software .

4. Related Work

In the next sections we discuss related work in the areas of stochastic optimal control and reinforcement learning and analyze the connections and differences with the **PI**^{2} algorithm and the generalized path integral control formulation.

4.1 Stochastic Optimal Control and Path Integrals

The path integral formalism for optimal control was introduced in Kappen (2005a,b). In this work, the role of noise in symmetry breaking phenomena was investigated in the context of stochastic optimal control. In Kappen et al. (2007), Wiegerinck et al. (2006), and Broek et al. (2008), the path integral formalism is extended for the stochastic optimal control of multi-agent systems.

Recent work on stochastic optimal control by Todorov (2008), Todorov (2007) and Todorov (2009b) shows that for a class of discrete stochastic optimal control problems, the Bellman equa-

7. The use of the kernel weights in the basis functions (31) for the purpose of time averaging has shown better performance with respect to other weighting approaches, across all of our experiments. Therefore this is the weighting that we suggest. Users may develop other weighting schemes as more suitable to their needs.

8. In fact, the term inside the exponent results by adding $\frac{h\min S(\tau_i)}{\max S(\tau_i)-\min S(\tau_i)}$, which cancels in (35), to the term $-\frac{hS(\tau_i)}{\max S(\tau_i)-\min S(\tau_i)}$ which is equal to $-\frac{1}{\lambda}S(\tau_i)$.

9. **R** is a user design parameter and usually chosen to be diagonal and invertible.

- **Given**:
  - An immediate cost function $r_t = q_t + \theta_t^T \mathbf{R} \theta_t$ (cf. 1)
  - A terminal cost term ϕ_{t_N} (cf. 1)
  - A stochastic parameterized policy $\mathbf{a}_t = \mathbf{g}_t^T (\theta + \varepsilon_t)$ (cf. 25)
  - The basis function $\mathbf{g}_{t_i}$ from the system dynamics (cf. 3 and Section 2.5.1)
  - The variance Σ_ε of the mean-zero noise ε_t
  - The initial parameter vector θ
- **Repeat** until convergence of the trajectory cost R:
  - Create K roll-outs of the system from the same start state $\mathbf{x}_0$ using stochstic parameters $\theta + \varepsilon_t$ at every time step
  - **For** $k = 1...K$, compute:
    - $P(\tau_{i,k}) = \frac{e^{-\frac{1}{\lambda} S(\tau_{i,k})}}{\sum_{k=1}^{K} [e^{-\frac{1}{\lambda} S(\tau_{i,k})}]}$
    - $S(\tau_{i,k}) = \phi_{t_N,k} + \sum_{j=i}^{N-1} q_{t_j,k} + \frac{1}{2} \sum_{j=i+1}^{N-1} (\theta + \mathbf{M}_{t_j,k} \varepsilon_{t_j,k})^T \mathbf{R} (\theta + \mathbf{M}_{t_j,k} \varepsilon_{t_j,k})$
    - $\mathbf{M}_{t_j,k} = \frac{\mathbf{R}^{-1} \mathbf{g}_{t_j,k} \mathbf{g}_{t_j,k}^T}{\mathbf{g}_{t_j,k}^T \mathbf{R}^{-1} \mathbf{g}_{t_j,k}}$
  - **For** $i = 1...(N-1)$, compute:
    - $\delta\theta_{t_i} = \sum_{k=1}^{K} [P(\tau_{i,k}) \mathbf{M}_{t_i,k} \varepsilon_{t_i,k}]$
  - Compute $[\delta\theta]_j = \frac{\sum_{i=0}^{N-1} (N-i) w_{j,t_i} [\delta\theta_{t_i}]_j}{\sum_{i=0}^{N-1} w_{j,t_i} (N-i)}$
  - Update $\theta \leftarrow \theta + \delta\theta$
  - Create one noiseless roll-out to check the trajectory cost $R = \phi_{t_N} + \sum_{i=0}^{N-1} r_{t_i}$. In case the noise cannot be turned off, that is, a stochastic system, multiple roll-outs need be averaged.

Table 2: Pseudocode of the **PI**^{2} algorithm for a 1D Parameterized Policy (Note that the discrete time step dt was absorbed as a constant multiplier in the cost terms).

tion can be written as the KL divergence between the probability distribution of the controlled and uncontrolled dynamics. Furthermore it is shown that the class of discrete KL divergence control problem is equivalent to the continuous stochastic optimal control formalism with quadratic cost control function and under the presence of Gaussian noise. In Kappen et al. (2009), the KL divergence control formalism is considered and it is transformed to a probabilistic inference problem. In all this aforementioned work, both in the path integral formalism as well as in KL divergence control, the class of stochastic dynamical systems under consideration is rather restrictive since the

control transition matrix is state independent. Moreover, the connection to direct policy learning in RL and model-free learning was not made in any of the previous projects.

Our **PI**^{2} algorithm differs with respect to the aforementioned work in the following points.

- In Todorov (2009b) the stochastic optimal control problem is investigated for discrete action - state spaces and therefore it is treated as Markov Decision Process (MDP). To apply our **PI**^{2} algorithm, we do not discretize the state space and we do not treat the problem as an MDP. Instead we work in continuous state - action spaces which are suitable for performing RL in high dimensional robotic systems. To the best of our knowledge, our results present RL in one of the most high dimensional continuous state action spaces.

- In our derivations, the probabilistic interpretation of control comes directly from the Feynman-Kac Lemma. Thus we do not have to impose any artificial "pseudo-probability" treatment of the cost as in Todorov (2009b). In addition, for the continuous state - action spaces we do not have to learn the value function as it is suggested in Todorov (2009b) via Z-learning. Instead we directly find the controls based on our generalization of optimal controls.

- In the previous work, the problem of how to sample trajectories is not addressed. Sampling is performed at once with the hope to cover the all state space. We follow a rather different approach that allows to attack robotic learning problems of the complexity and dimensionality of the little dog robot.

- The work in Todorov (2009a) considers stochastic dynamics with state dependent control matrix. However, the way of how the stochastic optimal control problem is solved is by imposing strong assumptions on the structure of the cost function and, therefore, restrictions of the proposed solution to special cases of optimal control problems. The use of this specific cost function allows transforming the stochastic optimal control problem to a deterministic optimal control problem. Under this transformation, the stochastic optimal control problem can be solved by using deterministic algorithms.

- With respect to the work in Broek et al. (2008), Wiegerinck et al. (2006) and Kappen et al. (2009) our **PI**^{2} algorithm has been derived for a rather general class of systems with control transition matrix that is state dependent. In this general class, Rigid body and multi-body dynamics as well as the DMPs are included. Furthermore we have shown how our results generalize previous work.

4.2 Reinforcement Learning of Parameterized Policies

There are two main classes of related algorithms: Policy Gradient algorithms and probabilistic algorithms.

Policy Gradient algorithms (Peters and Schaal, 2006a,b) compute the gradient of the cost function (24) at every iteration and the policy parameters are updated according to $\theta^{(new)} = \theta^{(old)} + \alpha\nabla_\theta J$. Some well-established algorithms, which we will also use for comparisons, are as follows (see also Peters and Schaal, 2006a,b).

4.2.1 REINFORCE

Williams (1992) introduced the episodic REINFORCE algorithm, which is derived from taking the derivative of (24) with respect to the policy parameters. This algorithm has rather slow convergence

due to a very noisy estimate of the policy gradient. It is also very sensitive to a reward baseline parameter b_k (see below). Recent work derived the optimal baseline for REINFORCE (cf. Peters and Schaal, 2008a), which improved the performance significantly. The episodic REINFORCE update equations are:

$$\begin{aligned} \nabla_{\theta_k} J &= E_{\tau_0}\left[(R(\tau_0) - b_k) \sum_{i=0}^{N-1} \nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i}) \right], \\ b_k &= \frac{E_{\tau_0}\left[\left(\sum_{i=0}^{N-1} \nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i})\right)^2 R(\tau_0) \right]}{E_{\tau_0}\left[\left(\sum_{i=0}^{N-1} \nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i})\right)^2 \right]}, \end{aligned}$$

where k denotes the k-th coefficient of the parameter vector and $R(\tau_0) = \frac{1}{N}\sum_{i=0}^{N-1} r_{t_i}$.

4.2.2 GPOMDP AND THE POLICY GRADIENT THEOREM ALGORITHM

In their GPOMDP algorithm, Baxter and Bartlett (2001) introduced several improvements over REINFORCE that made the gradient estimates more efficient. GPOMDP can also be derived from the policy gradient theorem (Sutton et al., 2000; Peters and Schaal, 2008a), and an optimal reward baseline can be added (cf. Peters and Schaal, 2008a). In our context, the GPOMDP learning algorithm can be written as:

$$\begin{aligned} \nabla_{\theta_k} J &= E_{\tau_0}\left[\sum_{j=0}^{N-1} (r_{tj} - b_{t_j}^{(k)}) \sum_{i=0}^{j} (\nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i})) \right], \\ b_{t_i}^{(k)} &= \frac{E_{\tau_0}\left[(\nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i}))^2 r_{ti} \right]}{E_{\tau_0}\left[(\nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i}))^2 \right]}. \end{aligned}$$

4.2.3 THE EPISODIC NATURAL ACTOR CRITIC

One of the most efficient policy gradient algorithm was introduced in Peters and Schaal (2008b), called the Episodic Natural Actor Critic. In essence, the method uses the Fisher Information Matrix to project the REINFORCE gradient onto a more effective update direction, which is motivated by the theory of natural gradients by Amari (1999). The eNAC algorithm takes the form of:

$$\begin{aligned} \xi_{t_i,k} &= \begin{bmatrix} \nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i}) \\ 1 \end{bmatrix}, \\ \begin{bmatrix} \nabla_\theta J \\ J_0 \end{bmatrix} &= E_{\tau_0}\left[\sum_{i=0}^{N-1} \xi_{t_i,k} \xi_{t_i,k}^T \right]^{-1} E_{\tau_0}\left[R(\tau_0) \sum_{i=0}^{N-1} \xi_{t_i,k} \right], \end{aligned}$$

where J_0 is a constant offset term.

4.2.4 POWER

The PoWER algorithm (Koeber and Peters, 2008) is a probabilistic policy improvement method, not a gradient algorithm. It is derived from an Expectation-Maximization framework using probability

matching (Dayan and Hinton, 1997; Peters and Schaal, 2008c). Using the notation of this paper, the parameter update of PoWER becomes:

$$\delta\theta = E_{\tau_0}\left[\sum_{i=0}^{N-1} R_{t_i}\frac{\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T}{\mathbf{g}_{t_i}^T\mathbf{g}_{t_i}}\right]^{-1} E_{\tau_0}\left[\sum_{t_i=t_o}^{t_N} R_{t_i}\frac{\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T\varepsilon_t}{\mathbf{g}_{t_i}^T\mathbf{g}_{t_i}}\right],$$

where $R_{t_i} = \sum_{j=i}^{N-1} r_{t_j}$. If we set $\mathbf{R}^{-1} = c\ \mathbf{I}$ in the update (37) of $\mathbf{PI}^2$, and set $\frac{\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T}{\mathbf{g}_{t_i}^T\mathbf{g}_{t_i}} = \mathbf{I}$ in the matrix inversion term of (39), the two algorithms look essentially identical. But it should be noted that the rewards r_{t_i} in PoWER need to behave like an improper probability, that is, be strictly positive and integrate to a constant number—this property can make the design of suitable cost functions more complicated. $\mathbf{PI}^2$, in contrast, uses exponentiated sum of reward terms, where the immediate reward can be arbitrary, and only the cost on the motor commands needs be quadratic. Our empirical evaluations revealed that, for cost functions that share the same optimum in the PoWER pseudo-probability formulation and the $\mathbf{PI}^2$ notation, both algorithms perform essentially identical, indicating that the matrix inversion term in PoWER may be unimportant for many systems. It should be noted that in Vlassis et al. (2009), PoWER was extended to the discounted infinite horizon case, where PoWER is the special case of a non-discounted finite horizon problem.

5. Evaluations

We evaluated $\mathbf{PI}^2$ in several synthetic examples in comparison with REINFORCE, GPOMDP, eNAC, and, when possible, PoWER. Except for PoWER, all algorithms are suitable for optimizing immediate reward functions of the kind $r_t = q_t + \mathbf{u}_t\mathbf{R}\mathbf{u}_t$. As mentioned above, PoWER requires that the immediate reward behaves like an improper probability. This property is incompatible with $r_t = q_t + \mathbf{u}_t\mathbf{R}\mathbf{u}_t$ and requires some special nonlinear transformations, which usually change the nature of the optimization problem, such that PoWER optimizes a different cost function. Thus, only one of the examples below has a compatible a cost function for all algorithms, including PoWER. In all examples below, exploration noise and, when applicable, learning rates, were tuned for every individual algorithms to achieve the best possible numerically stable performance. Exploration noise was only added to the maximally activated basis function in a motor primitive,[10] and the noise was kept constant for the entire time that this basis function had the highest activation—empirically, this tick helped improves the learning speed of all algorithms.

5.1 Learning Optimal Performance of a 1 DOF Reaching Task

The first evaluation considers learning optimal parameters for a 1 DOF DMP (cf. Equation 30). The immediate cost and terminal cost are, respectively:

$$r_t = 0.5 f_t^2 + 5000\ \theta^T\theta, \qquad \phi_{t_N} = 10000(\dot{y}_{t_N}^2 + 10(g - y_{t_N})^2)$$

with $y_{t_0} = 0$ and $g = 1$—we use *radians* as units motivated by our interest in robotics application, but we could also avoid units entirely. The interpretation of this cost is that we would like to reach the goal g with high accuracy while minimizing the acceleration of the movement and while keeping the parameter vector short. Each algorithm was run for 15 trials to compute a parameter update, and

10. That is, the noise vector in (25) has only one non-zero component.

a total of 1000 updates were performed. Note that 15 trials per update were chosen as the DMP had 10 basis functions, and the eNAC requires at least 11 trials to perform a numerically stable update due to its matrix inversion. The motor primitives were initialized to approximate a 5-th order polynomial as point-to-point movement (cf. Figure 1a,b), called a minimum-jerk trajectory in the motor control literature; the movement duration was 0.5 seconds, which is similar to normal human reaching movements. Gaussian noise of $N(0,0.1)$ was added to the initial parameters of the movement primitives in order to have different initial conditions for every run of the algorithms. The results are given in Figure 1. Figure 1a,b show the initial (before learning) trajectory generated by the DMP together with the learning results of the four different algorithms after learning—essentially, all algorithms achieve the same result such that all trajectories lie on top of each other. In Figure 1c, however, it can be seen that **PI**^{2} outperforms the gradient algorithms by an order of magnitude. Figure 1d illustrates learning curves for the same task as in Figure 1c, just that parameter updates are computed already after two roll-outs—the eNAC was excluded from this evaluation as it would be too heuristic to stabilize its ill-conditioned matrix inversion that results from such few roll-outs. **PI**^{2} continues to converge much faster than the other algorithms even in this special scenario. However, there are some noticeable fluctuation after convergence. This noise around the convergence baseline is caused by using only two noisy roll-outs to continue updating the parameters, which causes continuous parameter fluctuations around the optimal parameters. Annealing the exploration noise, or just adding the optimal trajectory from the previous parameter update as one of the roll-outs for the next parameter update can alleviate this issue—we do not illustrate such little "tricks" in this paper as they really only affect fine tuning of the algorithm.

5.2 Learning Optimal Performance of a 1 DOF Via-Point Task

The second evaluation was identical to the first evaluation, just that the cost function now forced the movement to pass through an intermediate via-point at $t = 300ms$. This evaluation is an abstract approximation of hitting a target, for example, as in playing tennis, and requires a significant change in how the movement is performed relative to the initial trajectory (Figure 2a). The cost function was

$$r_{300ms} = 100000000(G - y_{t_{300ms}})^2, \qquad \phi_{t_N} = 0$$

with $G = 0.25$. Only this single reward was given. For this cost function, the PoWER algorithm can be applied, too, with cost function $\tilde{r}_{300ms} = \exp(-1/\lambda \quad r_{300ms})$ and $\tilde{r}_{t_i} = 0$ otherwise. This transformed cost function has the same optimum as r_{300ms}. The resulting learning curves are given in Figure 2 and resemble the previous evaluation: **PI**^{2} outperforms the gradient algorithms by roughly an order of magnitude, while all the gradient algorithms have almost identical learning curves. As was expected from the similarity of the update equations, PoWER and **PI**^{2} have in this special case the same performance and are hardly distinguishable in Figure 2. Figure 2a demonstrates that all algorithms pass through the desired target G, but that there are remaining differences between the algorithms in how they approach the target G—these difference have a small numerical effect in the final cost (where **PI**^{2} and PoWER have the lowest cost), but these difference are hardly task relevant.

5.3 Learning Optimal Performance of a Multi-DOF Via-Point Task

A third evaluation examined the scalability of our algorithms to a high-dimensional and highly redundant learning problem. Again, the learning task was to pass through an intermediate target G,

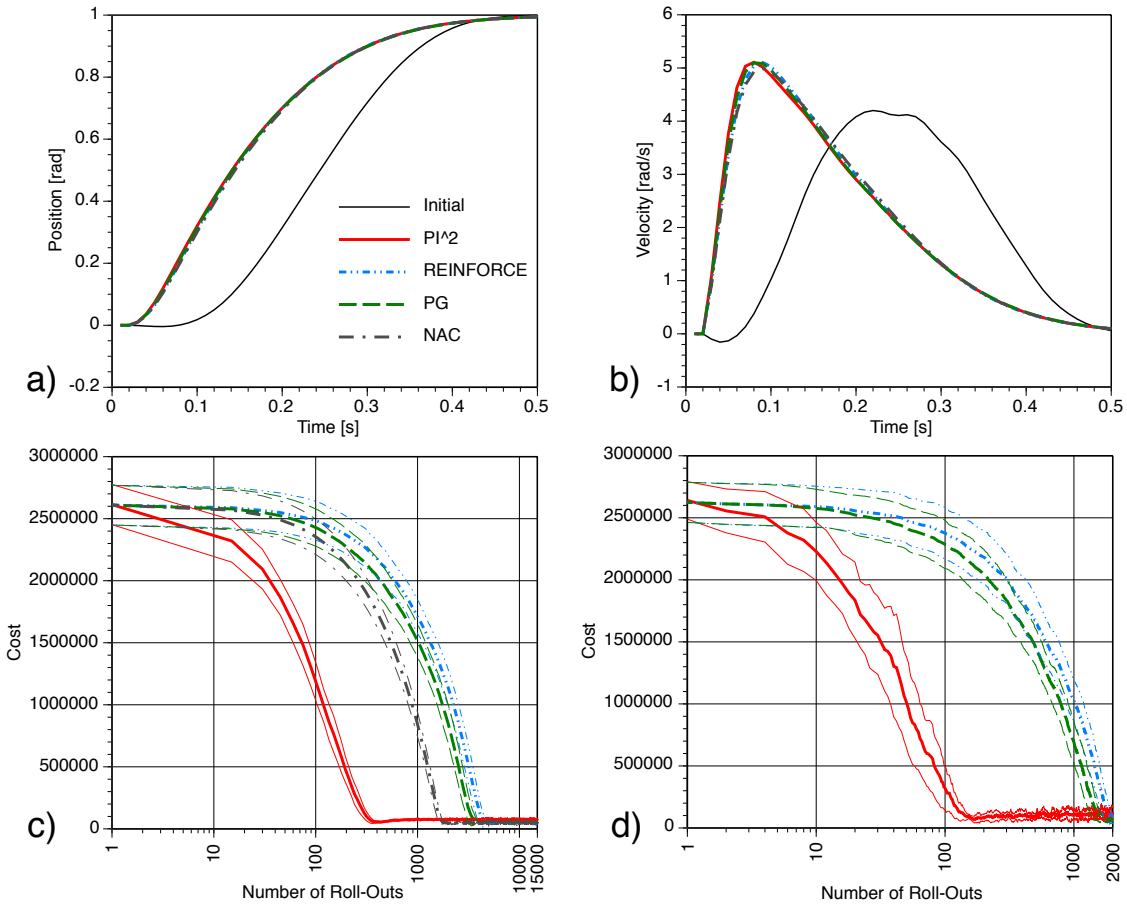


Figure 1: Comparison of reinforcement learning of an optimized movement with motor primitives. a) Position trajectories of the initial trajectory (before learning) and the results of all algorithms after learning—the different algorithms are essentially indistighuishable. b) The same as a), just using the velocity trajectories. c) Average learning curves for the different algorithms with 1 std error bars from averaging 10 runs for each of the algorithms. d) Learning curves for the different algorithms when only two roll-outs are used per update (note that the eNAC cannot work in this case and is omitted).

just that a $d = 2, 10$, or 50 dimensional motor primitive was employed. We assume that the multi-DOF systems model planar robot arms, where d links of equal length $l = 1/d$ are connected in an open chain with revolute joints. Essentially, these robots look like a multi-segment snake in a plane, where the tail of the snake is fixed at the origin of the 2D coordinate system, and the head of the snake can be moved in the 2D plane by changing the joint angles between all the links. Figure 3b,d,f illustrate the movement over time of these robots: the initial position of the robots is when all joint angles are zero and the robot arm completely coincides with the x-axis of the coordinate frame. The goal states of the motor primitives command each DOF to move to a joint angle, such that the entire robot configuration afterwards looks like a semi-circle where the most distal link of the robot

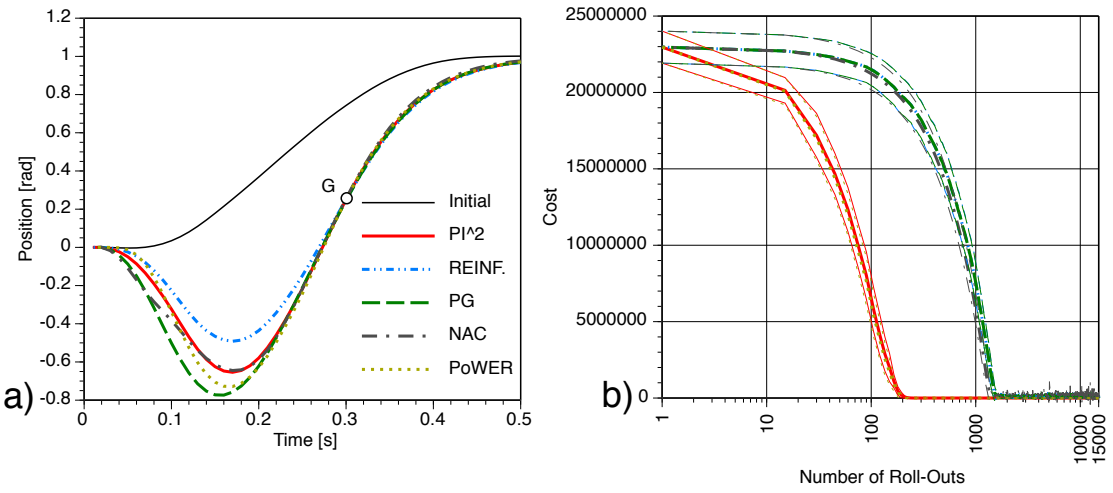


Figure 2: Comparison of reinforcement learning of an optimized movement with motor primitives for passing through an intermediate target G. a) Position trajectories of the initial trajectory (before learning) and the results of all algorithms after learning. b) Average learning curves for the different algorithms with 1 std error bars from averaging 10 runs for each of the algorithms.

(the end-effector) touches the y-axis. The higher priority task, however, is to move the end-effector through a via-point $G = (0.5, 0.5)$. To formalize this task as a reinforcement learning problem, we denote the joint angles of the robots as ξ_i, with $i = 1, 2, ..., d$, such that the first line of (30) reads now as $\ddot{\xi}_{i,t} = f_{i,t} + \mathbf{g}_{i,t}^T(\theta_i + \varepsilon_{i,t})$—this small change of notation is to avoid a clash of variables with the (x, y) task space of the robot. The end-effector position is computed as:

$$x_t = \frac{1}{d}\sum_{i=1}^{d} cos(\sum_{j=1}^{i} \xi_{j,t}), \quad y_t = \frac{1}{d}\sum_{i=1}^{d} sin(\sum_{j=1}^{i} \xi_{j,t}).$$

The immediate reward function for this problem is defined as

$$\begin{aligned} r_t &= \frac{\sum_{i=1}^{d}(d+1-i)\left(0.1 f_{i,t}^2 + 0.5\ \theta_i^T \theta_i\right)}{\sum_{i=1}^{d}(d+1-i)}, \\ \Delta r_{300ms} &= 100000000\left((0.5 - x_{t_{300ms}})^2 + (0.5 - y_{t_{300ms}})^2\right), \\ \phi_{t_N} &= 0, \end{aligned} \tag{39}$$

where Δr_{300ms} is added to r_t at time $t = 300ms$, that is, we would like to pass through the via-point at this time. The individual DOFs of the motor primitive were initialized as in the 1 DOF examples above. The cost term in (39) penalizes each DOF for using high accelerations and large parameter vectors, which is a critical component to achieve a good resolution of redundancy in the arm. Equation (39) also has a weighting term $d + 1 - i$ that penalizes DOFs proximal to the orgin more than those that are distal to the origin—intuitively, applied to human arm movements, this would mean that wrist movements are cheaper than shoulder movements, which is motivated by the

fact that the wrist has much lower mass and inertia and is thus energetically more efficient to move.

The results of this experiment are summarized in Figure 3. The learning curves in the left column demonstrate again that **PI**^{2} has an order of magnitude faster learning performance than the other algorithms, irrespective of the dimensionality. **PI**^{2} also converges to the lowest cost in all examples:

Algorithm	2-DOFs	10-DOFs	50-DOFs
PI^2	98000 ± 5000	15700 ± 1300	2800 ± 150
REINFORCE	125000 ± 2000	22000 ± 700	19500 ± 24000
PG	128000 ± 2000	28000 ± 23000	27000 ± 40000
NAC	113000 ± 10000	48000 ± 8000	22000 ± 2000

Figure 3 also illustrates the path taken by the end-effector before and after learning. All algorithms manage to pass through the via-point G appropriately, although the path particularly before reaching the via-point can be quite different across the algorithms. Given that **PI**^{2} reached the lowest cost with low variance in all examples, it appears to have found the best solution. We also added a "stroboscopic" sketch of the robot arm for the **PI**^{2} solution, which proceeds from the very right to the left as a function of time. It should be emphasized that there were absolutely no parameter tuning needed to achieve the **PI**^{2} results, while all gradient algorithms required readjusting of learning rates for every example to achieve best performance.

5.4 Application to Robot Learning

Figure 4 illustrates our application to a robot learning problem. The robot dog is to jump across as gap. The jump should make forward progress as much as possible, as it is a maneuver in a legged locomotion competition which scores the speed of the robot—note that we only used a physical simulator of the robot for this experiment, as the actual robot was not available. The robot has three DOFs per leg, and thus a total of $d = 12$ DOFs. Each DOF was represented as a DMP with 50 basis functions. An initial seed behavior (Figure 5-top) was taught by learning from demonstration, which allowed the robot barely to reach the other side of the gap without falling into the gap—the demonstration was generated from a manual adjustment of spline nodes in a spline-based trajectory plan for each leg.

PI^2 learning used primarily the forward progress as a reward, and slightly penalized the squared acceleration of each DOF, and the length of the parameter vector. Additionally, a penalty was incurred if the yaw or the roll exceeded a threshold value—these penalties encouraged the robot to

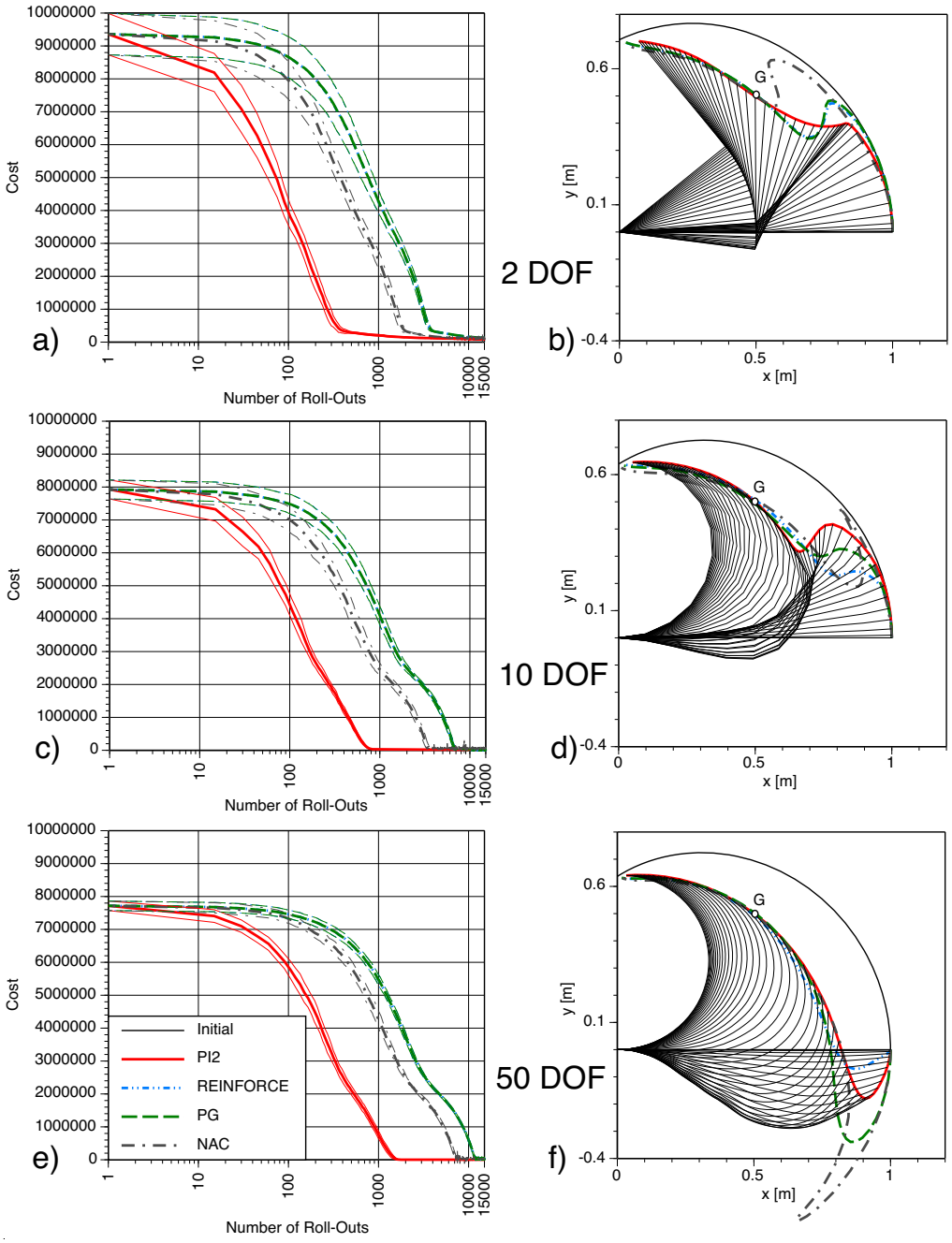
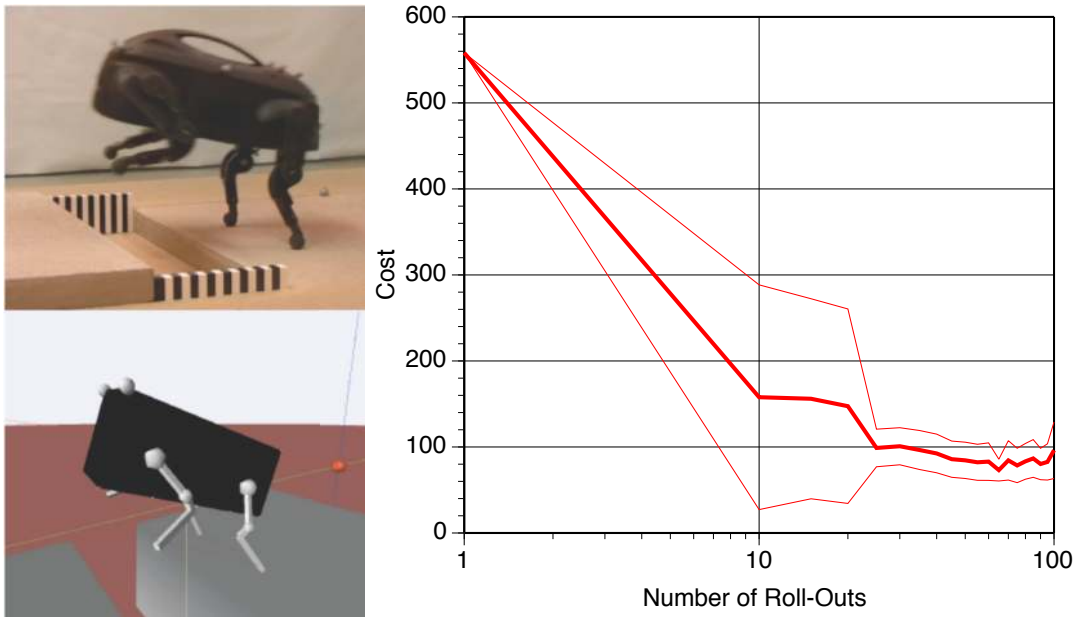


Figure 3: Comparison of learning multi-DOF movements (2,10, and 50 DOFs) with planar robot arms passing through a via-point G. a,c,e) illustrate the learning curves for different RL algorithms, while b,d,f) illustrate the end-effector movement after learning for all algorithms. Additionally, b,d,f) also show the initial end-effector movement, before learning to pass through G, and a "stroboscopic" visualization of the arm movement for the final result of $\mathbf{PI}^2$ (the movements proceed in time starting at the very right and ending by (almost) touching the y axis).



(a) Real & Simulated Robot Dog

(b) Learning curve for Dog Jump with $\mathbf{PI}^2$ $\pm 1 std$

Figure 4: Reinforcement learning of optimizing to jump over a gap with a robot dog. The improvement in cost corresponds to about 15 cm improvement in jump distance, which changed the robot's behavior from an initial barely successful jump to jump that completely traversed the gap with entire body. This learned behavior allowed the robot to traverse a gap at much higher speed in a competition on learning locomotion. The experiments for this paper were conducted only on the robot simulator.

jump straight forward and not to the side, and not to fall over. The exact cost function is:

$$
\begin{aligned}
r_t &= r_{roll} + r_{yaw} + \sum_{i=1}^{d} \left(a_1 f_{i,t}^2 + 0.5 a_2\, \theta_i^T \theta\right) (a_1 = 1.e-6, a_2 = 1.e-8), \\
r_{roll} &= \begin{cases} 100 * (|roll_t| - 0.3)^2, & \text{if } (|roll_t| > 0.3) \\ 0, & otherwise, \end{cases} \\
r_{yaw} &= \begin{cases} 100 * (|yaw_t| - 0.1)^2, & \text{if } (|yaw_t| > 0.1) \\ 0, & otherwise, \end{cases} \\
\phi_{t_N} &= 50000 (goal - x_{nose})^2,
\end{aligned}
$$

where $roll, yaw$ are the roll and yaw angles of the robot's body, and x_{nose} is the position of the front tip (the "nose") of the robot in the forward direction, which is the direction towards the $goal$. The multipliers for each reward component were tuned to have a balanced influence of all terms. Ten learning trials were performed initially for the first parameter update. The best 5 trials were kept, and five additional new trials were performed for the second and all subsequent updates. Essentially, this method performs importance sampling, as the rewards for the 5 trials in memory were re-computed

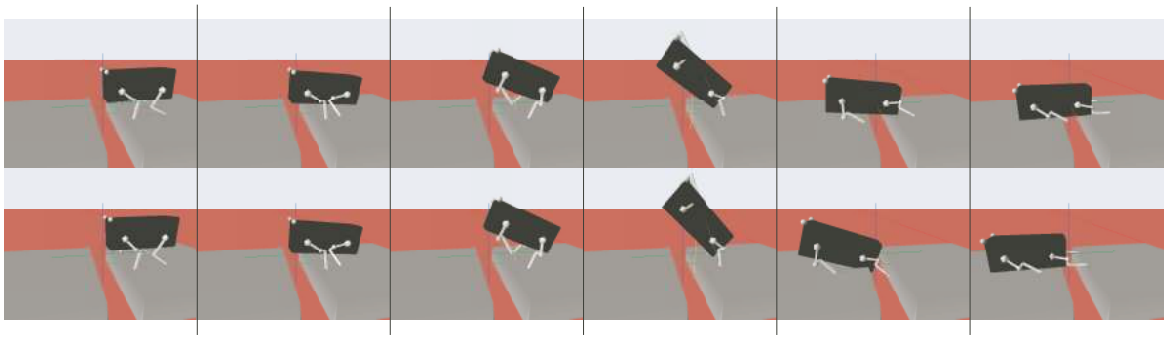


Figure 5: Sequence of images from the simulated robot dog jumping over a 14cm gap. Top: before learning. Bottom: After learning. While the two sequences look quite similar at the first glance, it is apparent that in the 4th frame, the robot's body is significantly higher in the air, such that after landing, the body of the dog made about 15cm more forward progress as before. In particular, the entire robot's body comes to rest on the other side of the gap, which allows for an easy transition to walking. In contrast, before learning, the robot's body (and its hind legs) are still on the right side of the gap, which does not allow for a successful continuation of walking.

with the latest parameter vectors. A total of 100 trials was performed per run, and ten runs were collected for computing mean and standard deviations of learning curves.

Figure 4 illustrates that after about 30 trials (i.e., 5 updates), the performance of the robot was converged and significantly improved, such that after the jump, almost the entire body was lying on the other side of the gap. Figure 4 captures the temporal performance in a sequence of snapshots of the robot. It should be noted that applying $\mathbf{PI}^2$ was algorithmically very simple, and manual tuning only focused on generated a good cost function, which is a different research topic beyond the scope of this paper.

6. Discussion

This paper derived a more general version of stochastic optimal control with path integrals, based on the original work by Kappen (2007) and Broek et al. (2008). The key results were presented in Table 1 and Section 2.5, which considered how to compute the optimal controls for a general class of stochastic control systems with state-dependent control transition matrix. One important class of these systems can be interpreted in the framework of reinforcement learning with parameterized policies. For this class, we derived Policy Improvement with Path Integrals ($\mathbf{PI}^2$) as a novel algorithm for learning a parameterized policy. $\mathbf{PI}^2$ inherits its sound foundation in first order principles of stochastic optimal control from the path integral formalism. It is a probabilistic learning method without open algorithmic tuning parameters, except for the exploration noise. In our evaluations, $\mathbf{PI}^2$ outperformed gradient algorithms significantly. It is also numerically simpler and has easier cost function design than previous probabilistic RL methods that require that immediate rewards are pseudo-probabilities. The similarity of $\mathbf{PI}^2$ with algorithms based on probability matching indicates that the principle of probability matching seems to approximate a stochastic optimal control framework. Our evaluations demonstrated that $\mathbf{PI}^2$ can scale to high dimensional control systems, unlike many other reinforcement learning systems.

Some issues, however, deserve more detailed discussions in the following paragraphs.

6.1 The Simplification $\lambda \mathbf{R}^{-1} = \Sigma_{\varepsilon}$

In order to obtain linear 2^{nd} order differential equations for the exponentially transformed HJB equations, the simplification $\lambda \mathbf{R}^{-1} = \Sigma_{\varepsilon}$ was applied. Essentially, this assumption couples the control cost to the stochasticity of the system dynamics, that is, a control with high variance will have relatively small cost, while a control with low variance will have relatively high cost. This assumption makes intuitively sense as it would be mostly unreasonable to attribute a lot of cost to an unreliable control component. Algorithmically, this assumption transforms the Gaussian probability for state transitions into a quadratic command cost, which is exactly what our immediate reward function postulated. Future work may allow removing this simplification by applying generalized versions of the Feynman-Kac Lemma.

6.2 Model-based, Hybrid, and Model-free Learning

Stochastic optimal control with path integrals makes a strong link to the dynamic system to be optimized—indeed, originally, it was derived solely as model-based method. As this paper demonstrated, however, this view can be relaxed. The roll-outs, needed for computing the optimal controls, can be generated either from simulating a model, or by gathering experience from an actual system. In the latter case, only the control transition matrix of the model needs be known, such that we obtain a hybrid model-based/model-free method. In this paper, we even went further and interpreted the stochastic dynamic system as a parameterized control policy, such that no knowledge of the model of the control system was needed anymore—that is, we entered a model-free learning domain. It seems that there is a rich variety of ways how the path integral formalism can be used in different applications.

6.3 Rules of Cost Function Design

The cost functions allowed in our formulations can have arbitrary state cost, but need quadratic command cost. This is somewhat restrictive, although the user can be flexible in what is defined as a command. For instance, the dynamic movement primitives (30) used in this paper can be written in two alternative ways:

$$\begin{aligned} \frac{1}{\tau}\dot{z}_t &= f_t + \mathbf{g}_t^T(\theta + \varepsilon_t), \\ &\text{or} \\ \frac{1}{\tau}\dot{z}_t &= \left[\mathbf{g}_t^T f_t\right]\left(\begin{bmatrix} \theta \\ 1 \end{bmatrix} + \tilde{\varepsilon}_t\right), \end{aligned}$$

where the new noise vector $\tilde{\varepsilon}_t$ has one additional coefficient. The second equation treats f_t as another basis function whose parameter is constant and is thus simply not updated. Thus, we added f_t to the command cost instead of treating it as a state cost.

We also numerically experimented with violations of the clean distinction between state and command cost. Equation (36) could be replaced by a cost term, which is an arbitrary function of state and command. In the end, this cost term is just used to differentiate the different roll-outs in a reward weighted average, similarly as in Peters and Schaal (2008c) and Koeber and Peters (2008). We noticed in several instances that $\mathbf{PI}^2$ continued to work just fine with this improper cost formulation.

Again, it appears that the path integral formalism and the **PI**^{2} algorithm allow the user to exploit creativity in designing cost functions, without absolute need to adhere perfectly to the theoretical framework.

6.4 Dealing with Hidden State

Finally, it is interesting to consider in how far **PI**^{2} would be affected by hidden state. Hidden state can either be of stochastic or deterministic nature, and we consider hidden state as adding additional equations to the system dynamics (3). Section 2.3 already derived that deterministic hidden states drop out of the **PI**^{2} update equations—these states of the system dynamics were termed as " non-directly actuated" states.

More interesting are hidden state variables that have stochastic differential equations, that is, these equations are uncontrolled but do have a noise term and a non-zero corresponding coefficient in $\mathbf{G}_t$ in Equation (3), and these equations are coupled to the other equations through their passive dynamics. The noise term of these equations would, in theory, contribute terms in Equation (36), but given that neither the noise nor the state of these equations are observable, we will not have the knowledge to add these terms. However, as long as the magnitude of these terms is small relative to the other terms in Equation (36), **PI**^{2} will continue to work fine, just a bit sub-optimally. This issue would affect other reinforcement learning methods for parameterized policies in the same way, and is not specific to **PI**^{2}.

6.5 Arbitrary States in the Cost Function

As a last point, we would like to consider which variables can actually enter the cost functions for **PI**^{2}. The path integral approach prescribes that the cost function needs to be a function of the state and command variables of the system equations (3). It should be emphasized that the state cost q_t can be any deterministic function of the state, that is, anything that is predictable from knowing the state, even if we do not know the predictive function. There is a lot of flexibility in this formulation, but it is also more restrictive than other approaches, for example, like policy gradients or the PoWER algorithm, where arbitrary variables can be used in the cost, no matter whether they are states or not.

We can think of any variable that we would like to use in the cost as having a corresponding differential equation in the system dynamics (3), that is, we simply add these variables as state variables, just that we do not know the analytical form of these equations. As in the previous section, it is useful to distinguish whether these states have deterministic or stochastic differential equations.

If the differential equation is deterministic, we can cover the case with the derivations from Section 2.3, that is, we consider such an equation as uncontrolled deterministic differential equation in the system dynamics, and we already know that we can use its state in the cost without any problems as it does not contribute to the probability of a roll-out.

If the differential equation is stochastic, the same argument as in the previous section applies, that is, the (unknown) contribution of the noise term of this equation to the exponentiated cost (36) needs to be small enough for **PI**^{2} to work effectively. Future work and empirical evaluations will have to demonstrate when these issues really matter—so far, we have not encountered problems in this regard.

7. Conclusions

The path integral formalism for stochastic optimal control has a very interesting potential to discover new learning algorithms for reinforcement learning. The $\mathbf{PI}^2$ algorithm derived in this paper for learning with parameterized policies demonstrated a surprisingly good performance, literally without any need for manual tuning of the parameters of the algorithm. We also demonstrated that the algorithm scales well into very high dimensional domains that were previously hardly approachable for reinforcement learning. Future work will thus allow us to focus much more on machine learning algorithms for cost function design, as the algorithmic components of the learning algorithm seem to be able to move towards a "black box" character.

Acknowledgments

This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, IIS-9988642, ECS-0326095, ANI-0224419, the DARPA program on Learning Locomotion, the Multidisciplinary Research Program of the Department of Defense (MURI N00014-00-1-0637), and the ATR Computational Neuroscience Laboratories. J.B. was supported by a prospective researcher fellowship from the Swiss National Science Foundation. E.T. was supported by a Myronis Fellowship.

Appendix A.

Appendix **A** contains the lemmas **A1** and **A2** and one theorem. The theorem provides the main result of the generalized path integral control formalism expressed by (18), (19), (20). Its proof is based on results proven in the lemmas **A1** and **A2**. In appendix **B** we provide the Feynman-Kac formula and we sketch the corresponding proof.

Lemma 1 *: The optimal control solution to the stochastic optimal control problem expressed by (1),(2),(3) and (4) is formulated as:*

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[-\mathbf{R}^{-1} \mathbf{G}_{t_i}^{(c)T} \int \tilde{p}(\tau_i) \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\tau_i) d\tau_i \right]$$

where $\tilde{p}(\tau_i) = \frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_i)\right)}{\int \exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_i)\right) d\tau_i}$ *is a path dependent probability distribution. The term* $\tilde{S}(\tau_i)$ *is a path function defined as* $\tilde{S}(\tau_i) = S(\tau_i) + \frac{\lambda}{2}\sum_{j=i}^{N-1} \log|\mathbf{H}_{t_j}|$ *that satisfies the following condition* $\lim_{dt\to 0} \int \exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_i)\right) d\tau_i \in \mathcal{C}^{(1)}$ *for any sampled trajectory starting from state* $\mathbf{x}_{t_i}$*. Moreover the term* $\mathbf{H}_{t_j}$ *is given by* $\mathbf{H}_{t_j} = \mathbf{G}_{t_j}^{(c)} \mathbf{R}^{-1} \mathbf{G}_{t_j}^{(c)T}$ *while the term* $S(\tau_i)$ *is defined according to*

$$S(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2} \sum_{j=i}^{N-1} \| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \|_{\mathbf{H}_{t_j}}^2 dt.$$

Proof The optimal controls at the state $\mathbf{x}_{t_i}$ is expressed by the equation $\mathbf{u}_{t_i} = -\mathbf{R}^{-1}\mathbf{G}_{t_i} \nabla_{\mathbf{x}_{t_i}} V_{t_i}$. Due to the exponential transformation of the value function $\Psi_{t_i} = -\lambda \log V_{t_i}$ the equation of the optimal controls is written as:

$$\mathbf{u}_{t_i} = \lambda \mathbf{R}^{-1} \mathbf{G}_{t_i} \frac{\nabla_{\mathbf{x}_{t_i}} \Psi_{t_i}}{\Psi_{t_i}}.$$

In discrete time the optimal control is expressed as follows:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\nabla_{\mathbf{x}_{t_i}} \Psi_{t_i}^{(dt)}}{\Psi_{t_i}^{(dt)}} \right).$$

By using equation (17) and substituting $\Psi^{(dt)}(\mathbf{x}_{t_i}, t)$ we have:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\nabla_{\mathbf{x}_{t_i}} \int \exp\left(-\frac{1}{\lambda} Z(\tau_i)\right) d\tau_i}{\int \exp\left(-\frac{1}{\lambda} Z(\tau_i)\right) d\tau_i} \right).$$

Substitution of the term $Z(\tau_i)$ results in the equation:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\nabla_{\mathbf{x}_{t_i}} \int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i) - \frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) d\tau_i}{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i) - \frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) d\tau_i} \right).$$

Next we are using standard properties of the exponential function that lead to:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\nabla_{\mathbf{x}_{t_i}} \left[\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) \exp\left(-\frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) d\tau_i \right]}{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) \exp\left(-\frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) d\tau_i} \right).$$

The term $\exp\left(-\frac{\lambda N l}{2} \log\left(2\pi dt \lambda\right)\right)$ does not depend on the trajectory τ_i, therefore it can be taken outside the integral as well as outside the gradient. Thus we will have that:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\exp\left(-\frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) \nabla_{\mathbf{x}_{t_i}} \left[\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i \right]}{\exp\left(-\frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) \int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i} \right).$$

The constant term drops from the nominator and denominator and thus we can write:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \left[\frac{\nabla_{\mathbf{x}_{t_i}} \int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i}{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i} \right] \right).$$

Under the assumption that term $\exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i$ is continuously differentiable in $\mathbf{x}_{t_i}$ and dt we can change order of the integral with the differentiation operations. In general for $\nabla_x \int f(x,y) dy = \int \nabla_x f(x,y) dy$ to be true, $f(x,t)$ should be continuous in y and differentiable in x. Under this assumption, the optimal controls can be further formulated as:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\int \nabla_{\mathbf{x}_{t_i}} \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i}{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i} \right].$$

Application of the differentiation rule of the exponent results in:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) \nabla_{\mathbf{x}_{t_i}} \left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i}{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i} \right].$$

The denominator is a function of $\mathbf{x}_{t_i}$ the current state and thus it can be pushed inside the integral of the nominator:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \int \frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_i)\right)}{\int \exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_i)\right) d\tau_i} \nabla_{\mathbf{x}_{t_i}} \left(-\frac{1}{\lambda}\tilde{S}(\tau_i) \right) d\tau_i \right].$$

By defining the probability $\tilde{p}(\tau_i) = \frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_i)\right)}{\int \exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_i)\right) d\tau_i}$ the expression above can be written as:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \int \tilde{p}(\tau_i) \nabla_{\mathbf{x}_{t_i}} \left(-\frac{1}{\lambda}\tilde{S}(\tau_i) \right) d\tau_i \right].$$

Further simplification will result in:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[-\mathbf{R}^{-1} \mathbf{G}_{t_i}^T \int \tilde{p}(\tau_i) \nabla_{\mathbf{x}_{t_i}} \tilde{S}(\tau_i) d\tau_i \right].$$

We know that the control transition matrix has the form $\mathbf{G}(\mathbf{x}_{t_i})^T = [0^T \quad \mathbf{G}_c(\mathbf{x}_{x_{t_i}})^T]$. In addition the partial derivative $\nabla_{\mathbf{x}_{t_i}} \tilde{S}(\tau_i)$ can be written as $\nabla_{\mathbf{x}_{t_i}} \tilde{S}(\tau_i)^T = [\nabla_{\mathbf{x}_{t_i}^{(m)}} \tilde{S}(\tau_i)^T \quad \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\tau_i)^T]$. By using these equations we will have that:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(-\mathbf{R}^{-1} [0^T \quad \mathbf{G}_{t_i}^{(c)T}] \int \tilde{p}(\tau_o) \begin{bmatrix} \nabla_{\mathbf{x}_{t_i}^{(m)}} \tilde{S}(\tau_i) \\ \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\tau_i) \end{bmatrix} d\tau_i \right).$$

The equation above can be written in the form:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(-[0^T \quad \mathbf{R}^{-1} \mathbf{G}_{t_i}^{(c)T}] \int \tilde{p}(\tau_i) \begin{bmatrix} \nabla_{\mathbf{x}_{t_i}^{(m)}} \tilde{S}(\tau_i) \\ \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\tau_i) \end{bmatrix} d\tau_i \right).$$

or

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(-[0^T \quad \mathbf{R}^{-1} \mathbf{G}_{t_i}^{(c)T}] \begin{bmatrix} \int \tilde{p}(\tau_i) \cdot \nabla_{\mathbf{x}_{t_i}^{(m)}} \tilde{S}(\tau_i) d\tau_i \\ \int \tilde{p}(\tau_i) \cdot \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\tau_i) d\tau_i \end{bmatrix} \right).$$

Therefore we will have the result

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[-\mathbf{R}^{-1} \mathbf{G}_{t_i}^{(c)T} \int \tilde{p}(\tau_i) \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\tau_i) d\tau_i \right].$$

■

Lemma 2 *: Given the stochastic dynamics and the cost in (1),(2),(3) and(4) the gradient of the path function $\tilde{S}(\tau_i)$ with respect to the directly actuated part of the state $\mathbf{x}_{t_i}^{(c)}$ is formulated as:*

$$\lim_{dt \to 0} \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\tau_i) \right) = -\mathbf{H}_{t_i}^{-1} \left(\mathbf{G}_{t_i}^{(c)} \varepsilon_{t_i} - \mathbf{b}_{t_i} \right)$$

where the function $b(\mathbf{x}_{t_i})$ *defined as* $\lambda \mathbf{H}(\mathbf{x}_{t_i})\Phi_{t_i}$ *with* $\mathbf{H}_{t_i} = \mathbf{G}_{t_i}^{(c)} \mathbf{R}^{-1} \mathbf{G}_{t_i}^{(c)^T}$ *and the quantity* $\Phi_{t_i} \in \Re^{l \times 1}$ *is expressed as:*

$$\Phi_{t_i} = \frac{1}{2} \begin{bmatrix} trace\left(\mathbf{H}_{t_i}^{-1} \ \partial_{\mathbf{x}_{t_i}^{(c1)}} \mathbf{H}_{t_i}\right) \\ trace\left(\mathbf{H}_{t_i}^{-1} \ \partial_{\mathbf{x}_{t_i}^{(c2)}} \mathbf{H}_{t_i}\right) \\ \cdot \\ \cdot \\ \cdot \\ trace\left(\mathbf{H}_{t_i}^{-1} \ \partial_{\mathbf{x}_{t_i}^{(cl)}} \mathbf{H}_{t_i}\right) \end{bmatrix}.$$

Proof

We are calculating the term $\nabla_{\mathbf{x}_{t_o}^{(c)}} \tilde{S}(\tau_o)$. More precisely we have shown that

$$\tilde{S}(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2} \sum_{j=i}^{N-1} \| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \|_{\mathbf{H}_{t_j}}^2 dt + \frac{\lambda}{2} \sum_{j=i}^{N-1} \log |\mathbf{H}_{t_j}|.$$

To limit the length of our derivation we introduce the notation $\gamma_{t_j} = \alpha_{t_j}^T \ \mathbf{h}_{t_j}^{-1} \ \alpha_{t_j}$ and $\alpha_{t_j} = \left(\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)} - \mathbf{f}_{t_j}^{(c)} dt\right)$ and it is easy to show that $\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \|_{\mathbf{H}_{t_j}}^2 dt = \frac{1}{dt} \gamma_{t_j}$ and therefore we will have:

$$\tilde{S}(\tau_i) = \phi_{t_N} + \frac{1}{2dt} \sum_{j=i}^{N-1} \gamma_{t_j} + \sum_{t_o}^{t_N} Q_{t_j} dt + \frac{\lambda}{2} \sum_{j=i}^{N-1} \log |\mathbf{H}_{t_j}|.$$

■

In the analysis that follows we provide the derivative of the 1th, 2th and 4th term of the cost function. We assume that the cost of the state during the time horizon $Q_{t_i} = 0$. In cases that this is not true then the derivative $\nabla_{\mathbf{x}_{t_i}^{(c)}} \sum_{t_i}^{t_N} Q_{t_i} dt$ needs to be found as well. By calculating the term $\nabla_{\mathbf{x}_{t_o}^{(c)}} \tilde{S}(\tau_o)$ we can find the local controls $\mathbf{u}(\tau_i)$. It is important to mention that the derivative of the path cost $S(\tau_i)$ is taken only with respect to the current state $\mathbf{x}_{t_o}$.

The first term is:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}} (\phi_{t_N}) = 0.$$

DERIVATIVE OF THE 2TH TERM $\nabla_{\mathbf{x}_{t_i}^{(c)}} \left[\frac{1}{2dt} \sum_{i=1}^{N-1} \gamma_{t_i}\right]$ OF THE COST $S(\tau_i)$.

The second term can be found as follows:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}} \left[\frac{1}{2dt} \sum_{j=i}^{N-1} \gamma_{t_j}\right].$$

The operator $\nabla_{\mathbf{x}_{t_o}^{(c)}}$ is linear and it can massaged inside the sum:

$$\frac{1}{2dt}\sum_{j=i}^{N-1}\nabla_{\mathbf{x}_{t_j}^{(c)}}\left(\gamma_{t_j}\right).$$

Terms that do not depend on $\mathbf{x}_{t_i}^{(c)}$ drop and thus we will have:

$$\frac{1}{2dt}\nabla_{\mathbf{x}_{t_i}^{(c)}}\gamma_{t_i}.$$

Substitution of the parameter $\gamma_{t_i}=\alpha_{t_i}^T\ \mathbf{H}_{t_i}^{-1}\ \alpha_{t_i}$ will result in:

$$\frac{1}{2dt}\nabla_{\mathbf{x}_{t_i}^{(c)}}\left[\alpha_{t_i}^T\ \mathbf{H}_{t_i}^{-1}\ \alpha_{t_i}\right].$$

By making the substitution $\beta_{t_i}=\mathbf{H}_{t_i}^{-1}\ \alpha_{t_i}$ and applying the rule $\nabla\left(\mathbf{u}(\mathbf{x})^T\mathbf{v}(\mathbf{x})\right)=\nabla\left(\mathbf{u}(\mathbf{x})\right)\mathbf{v}(\mathbf{x})+\nabla\left(\mathbf{v}(\mathbf{x})\right)\mathbf{u}(\mathbf{x})$ we will have that:

$$\frac{1}{2dt}\left[\nabla_{\mathbf{x}_{t_i}^{(c)}}\alpha_{t_i}\ \beta_{t_i}+\nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \alpha_{t_i}\right]. \tag{40}$$

Next we find the derivative of α_{t_o}:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}}\alpha_{t_i}=\nabla_{\mathbf{x}_{t_i}^{(c)}}\left[\mathbf{x}_{t_{i+1}}^{(c)}-\mathbf{x}_{t_i}^{(c)}-\mathbf{f}_c(\mathbf{x}_{t_i})dt\right].$$

and the result is

$$\nabla_{\mathbf{x}_{t_i}^{(c)}}\alpha_{t_i}=-I_{l\times l}-\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}dt.$$

We substitute back to (40) and we will have:

$$\frac{1}{2dt}\left[-\left(I_{l\times l}+\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}dt\right)\ \beta_{t_i}+\nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \alpha_{t_i}\right].$$

$$-\frac{1}{2dt}\left(I_{l\times l}+\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}dt\right)\ \beta_{t_i}+\frac{1}{2dt}\ \nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \alpha_{t_i}.$$

After some algebra the result of $\nabla_{\mathbf{x}_{t_i}^{(c)}}\left(\frac{1}{2dt}\sum_{i=1}^{N-1}\gamma_{t_i}\right)$ is expressed as:

$$-\frac{1}{2dt}\ \beta_{t_i}-\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}\ \beta_{t_i}+\frac{1}{2dt}\ \nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \alpha_{t_i}.$$

The next step now is to find the limit of the expression above as $dt\to 0$. More precisely we will have that:

$$\lim_{dt\to 0}\left[-\frac{1}{2dt}\ \beta_{t_i}-\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}\ \beta_{t_i}+\frac{1}{2dt}\ \nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \alpha_{t_i}\right].$$

LIMIT OF THE FIRST SUBTERM: $-\frac{1}{2dt}\ \beta_{t_i}$

We will continue our analysis by finding the limit for each one of the 3 terms above. The limit of the first term is calculated as follows:

$$\begin{aligned}\lim_{dt\to 0}\left(-\frac{1}{2dt}\ \ \beta_{t_i}\right) &= -\lim_{dt\to 0}\left(\frac{1}{2dt}\ \ \mathbf{H}_{t_i}^{-1}\ \ \alpha_{t_i}\right)\\ &= -\frac{1}{2}\ \ \mathbf{H}_{t_i}^{-1}\ \ \lim_{dt\to 0}\alpha_{t_i}\\ &= -\frac{1}{2}\ \ \mathbf{H}_{t_i}^{-1}\ \ \lim_{dt\to 0}\left((\mathbf{x}_{t_{i+1}}^{(c)}-\mathbf{x}_{t_i}^{(c)})\frac{1}{dt}-\mathbf{f}_{t_i}^{(c)}\right).\end{aligned}$$

LIMIT OF THE SECOND SUBTERM: $-\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}\ \ \beta_{t_i}$

The limit of the second term is calculated as follows:

$$\begin{aligned}-\lim_{dt\to 0}\left(\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}\ \ \beta_{t_i}\right) &= -\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_c(\mathbf{x}_{t_i})\ \ \lim_{dt\to 0}\beta_{t_i}\\ &= -\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}\ \ \lim_{dt\to 0}\left(\mathbf{H}_{t_i}^{-1}\ \ \alpha_{t_i}\right)\\ &= -\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_c(\mathbf{x}_{t_i})\ \ \mathbf{H}_{t_i}^{-1}\ \ \lim_{dt\to 0}\alpha_{t_i}\\ &= 0.\end{aligned}$$

The limit of the term $\lim_{dt\to 0}\alpha_{t_i}$ is derived as:

$$\lim_{dt\to 0}\left(\mathbf{x}_{t_{i+1}}^{(c)}-\mathbf{x}_{t_i}^{(c)}-\mathbf{f}_c(\mathbf{x}_{t_i})dt\right)=\lim_{dt\to 0}\left(\mathbf{x}_{t_{i+dt}}^{(c)}-\mathbf{x}_{t_i}^{(c)}\right)-\lim_{dt\to 0}\mathbf{f}_c(\mathbf{x}_{t_i})dt=0-0=0.$$

LIMIT OF THE THIRD SUBTERM: $\frac{1}{2dt}\ \ \nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \ \alpha_{t_i}$

Finally the limit of the third term can be found as:

$$\begin{aligned}&\lim_{dt\to 0}\left(\frac{1}{2dt}\ \ \nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \ \alpha_{t_i}\right)=\\ &=\lim_{dt\to 0}\nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \ \lim_{dt\to 0}\left(\frac{1}{2dt}\ \ \alpha_{t_i}\right)=\\ &=\lim_{dt\to 0}\nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \ \frac{1}{2}\lim_{dt\to 0}\left((\mathbf{x}_{t_{i+1}}^{(c)}-\mathbf{x}_{t_i}^{(c)})\frac{1}{dt}-\mathbf{f}_{t_i}^{(c)}\right).\end{aligned}$$

We substitute $\beta_{t_i}=\mathbf{H}_{t_i}^{-1}\ \ \alpha_{t_i}$ and write the matrix $\mathbf{H}_{t_i}^{-1}$ in row form:

$$= \lim_{dt\to 0} \nabla_{\mathbf{x}_{t_i}^{(c)}} \left(\mathbf{H}_{t_i}^{-1} \;\; \alpha_{t_i}\right) \;\; \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_i}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right) =$$

$$= \lim_{dt\to 0} \nabla_{\mathbf{x}_{t_i}^{(c)}} \left(\begin{bmatrix} \mathbf{H}_{t_i}^{(1)^{-T}} \\ \mathbf{H}_{t_i}^{(2)^{-T}} \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{H}_{t_i}^{(l)^{-T}} \end{bmatrix} \;\; \alpha_{t_i} \right) \;\; \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right)$$

$$= \lim_{dt\to 0} \nabla_{\mathbf{x}_{t_i}^{(c)}} \begin{bmatrix} \mathbf{H}_{t_i}^{(1)^{-T}} & \alpha_{t_i} \\ \mathbf{H}_{t_i}^{(2)^{-T}} & \alpha_{t_i} \\ \cdot & \\ \cdot & \\ \cdot & \\ \mathbf{H}_{t_i}^{(l)^{-T}} & \alpha_{t_i} \end{bmatrix} \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right).$$

We can push the operator $\nabla_{\mathbf{x}_{t_i}^{(c)}}$ insight the matrix and apply it to each element.

$$= \lim_{dt\to 0} \begin{bmatrix} \nabla^T_{\mathbf{x}_{t_i}^{(c)}} \left(\mathbf{H}_{t_i}^{(1)^{-T}} \;\; \alpha_{t_i} \right) \\ \nabla^T_{\mathbf{x}_{t_i}^{(c)}} \left(\mathbf{H}_{t_i}^{(2)^{-T}} \;\; \alpha_{t_i} \right) \\ \cdot \\ \cdot \\ \cdot \\ \nabla^T_{\mathbf{x}_{t_i}^{(c)}} \left(\mathbf{H}_{t_i}^{(l)^{-T}} \;\; \alpha_{t_i} \right) \end{bmatrix} \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right).$$

We again use the rule $\nabla \left(\mathbf{u}(\mathbf{x})^T \mathbf{v}(\mathbf{x}) \right) = \nabla \left(\mathbf{u}(\mathbf{x}) \right) \mathbf{v}(\mathbf{x}) + \nabla \left(\mathbf{v}(\mathbf{x}) \right) \mathbf{u}(\mathbf{x})$ and thus we will have:

$$= \lim_{dt\to 0} \begin{bmatrix} \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(1)^{-T}} \;\; \alpha_{t_i} + \nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i} \;\; \mathbf{H}_{t_i}^{(1)^{-T}} \right)^T \\ \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(2)^{-T}} \;\; \alpha_{t_i} + \nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i} \;\; \mathbf{H}_{t_i}^{(2)^{-T}} \right)^T \\ \cdot \\ \cdot \\ \cdot \\ \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(l)^{-T}} \;\; \alpha_{t_i} + \nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i} \;\; \mathbf{H}_{t_i}^{(l)^{-T}} \right)^T \end{bmatrix} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right).$$

We can split the matrix above into two terms and then we pull out the terms α_{t_i} and $\nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i}$ respectively :

$$= \lim_{dt\to 0} \left(\alpha_{t_i}^T \begin{bmatrix} \nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(1)^{-T}} \\ \nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(2)^{-T}} \\ \cdot \\ \cdot \\ \cdot \\ \nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(l)^{-T}} \end{bmatrix} + \begin{bmatrix} \mathbf{H}_{t_i}^{(1)^{-T}} \\ \mathbf{H}_{t_i}^{(2)^{-T}} \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{H}_{t_i}^{(l)^{-T}} \end{bmatrix} \nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i}^T \right) \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right)$$

$$= \lim_{dt\to 0} \left(\alpha_{t_i}^T \ \nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{-1} + \mathbf{H}_{t_i}^{-1} \ \nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i}^T \right) \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right)$$

$$= \left(\lim_{dt\to 0} (\alpha_{t_i}^T) \ \nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{-1} + \mathbf{H}_{t_i}^{-1} \ \lim_{dt\to 0} \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i}^T \right) \right) \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right).$$

Since $\lim_{dt\to 0} \left(\alpha_{t_i}^T \right) = 0_{1\times l}$ and $\lim_{dt\to 0} \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i}^T \right) = -I_{l\times l}$ the final result is expressed as follows

$$\lim_{dt\to 0} \left(\frac{1}{2dt} \nabla_{\mathbf{x}_{t_i}^{(c)}} \beta_{t_i} \ \alpha_{t_i} \right) = -\mathbf{H}_{t_i}^{-1} \ \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_o}^{(c)} \right).$$

After we have calculated the 3 sub-terms, the 2th term of the of the derivative of path cost $S(\tau_o)$ can be expressed in the following form:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}} \left(\frac{1}{2\lambda dt} \sum_{j=i}^{N-1} \gamma_{t_j} \right) = -\mathbf{H}_{t_i}^{-1} \ \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)}) \right).$$

DERIVATIVE OF THE FOURTH TERM $\nabla_{\mathbf{x}_{t_i}^{(c)}} \left(\frac{\lambda}{2} \sum_{j=i}^{N-1} \log |\mathbf{H}_{t_j}| \right)$ OF THE COST $S(\tau_i)$.

The analysis for the 4th term is given below:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}} \left(\frac{\lambda}{2} \sum_{j=i}^{N-1} \log |\mathbf{H}_{t_j}| \right) = \frac{\lambda}{2} \nabla_{\mathbf{x}_{t_i}^{(c)}} \log |\mathbf{H}_{t_i}|.$$

If we assume that $\mathbf{x}_{t_o}^{(c)} = [x_{t_o}^{(c1)}, x_{t_o}^{(c2)} .... x_{t_o}^{(cl)}]$ and take the derivative with respect to each element we will have

$$\partial_{\mathbf{x}_{t_i}^{(ci)}} \left(\frac{\lambda}{2} \log |\mathbf{H}_{t_i}| \right) = \frac{\lambda}{2} \frac{1}{|\mathbf{H}_{t_i}|} \ \partial_{\mathbf{x}_{t_i}^{(ci)}} |\mathbf{H}_{t_i}|.$$

$$\partial_{\mathbf{x}_{t_i}^{(ci)}} \left(\frac{\lambda}{2} \log |\mathbf{H}_{t_i}| \right) = \frac{\lambda}{2} \frac{1}{|\mathbf{H}(\mathbf{x}_{t_i})|} \ |\mathbf{H}_{t_i}| \ trace \left(\mathbf{H}_{t_i}^{-1} \cdot \partial_{\mathbf{x}_{t_i}^{(ci)}} \mathbf{H}_{t_i} \right).$$

$$\partial_{\mathbf{x}_{t_i}^{(ci)}} \left(\frac{\lambda}{2} \log |\mathbf{H}_{t_i}| \right) = \frac{\lambda}{2} trace \left(\mathbf{H}_{t_i}^{-1} \ \partial_{\mathbf{x}_{t_i}^{(ci)}} \mathbf{H}_{t_i} \right).$$

Where we make used of the identity $\partial det(A) = det(A)Tr\left(A^{-1}\partial A\right)$. The result is expressed as:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}}\left(\frac{\lambda}{2}\log|\mathbf{H}_{t_i}|\right) = \frac{\lambda}{2}\begin{bmatrix} trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(c1)}}\mathbf{H}_{t_i}\right) \\ trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(c2)}}\mathbf{H}_{t_i}\right) \\ \cdot \\ \cdot \\ \cdot \\ trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(cl)}}\mathbf{H}_{t_i}\right) \end{bmatrix}.$$

or in a more compact form:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}}\left(\frac{\lambda}{2}\log|\mathbf{H}_{t_i}|\right) = \mathbf{H}_{t_i}^{-1}\mathbf{b}_{t_i}.$$

where $\mathbf{b}(\mathbf{x}_{t_i}) = \lambda\mathbf{H}(\mathbf{x}_{t_i})\Phi_{t_i}$ and the quantity $\Phi_{t_i} \in \Re^{l\times 1}$ is defined as:

$$\Phi_{t_i} = \frac{1}{2}\begin{bmatrix} trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(c1)}}\mathbf{H}_{t_i}\right) \\ trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(c2)}}\mathbf{H}_{t_i}\right) \\ \cdot \\ \cdot \\ \cdot \\ trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(cl)}}\mathbf{H}_{t_i}\right) \end{bmatrix}. \tag{41}$$

Since we computed all the terms of the derivative of the path cost $\tilde{S}(\tau_o)$ and after putting all the terms together we have the result expressed as follows:

$$\lim_{dt\to 0}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right) = -\mathbf{H}_{t_i}^{-1}\left(\lim_{dt\to 0}\left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)})\frac{1}{dt} - \mathbf{f}_{t_i}^{(c)}\right) - \mathbf{b}_{t_i}\right).$$

By taking into account the fact that $\lim_{dt\to 0}\left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)})\frac{1}{dt} - \mathbf{f}_{t_i}^{(c)}\right) = \mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i}$ we get the following final expression:

$$\lim_{dt\to 0}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right) = -\mathbf{H}_{t_i}^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right).$$

Theorem 3 *: The optimal control solution to the stochastic optimal control problem expressed by (1),(2),(3),(4) is formulated by the equation that follows:*

$$\mathbf{u}_{t_i} = \lim_{dt\to 0}\int \tilde{p}(\tau_i)\ \mathbf{u}_L(\tau_i)\ d\tau_i,$$

where $\tilde{p}(\tau_i) = \frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_i)\right)}{\int \exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_i)\right) d\tau_i}$ ***is a path depended probability distribution and the term*** $\mathbf{u}(\tau_i)$ ***defined as*** $\mathbf{u}_L(\tau_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\left(\mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\right)^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right)$ ***are the local controls of each sampled trajectory starting from state*** $\mathbf{x}_{t_i}$***. The terms*** ε_{t_i} ***and*** $\mathbf{b}_{t_i}$ ***are defined as*** $\varepsilon_{t_i} = \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)})\frac{1}{dt} - \mathbf{f}_{t_i}^{(o)}\right)$ ***and*** $\mathbf{b}(\mathbf{x}_{t_i}) = \lambda\mathbf{H}(\mathbf{x}_{t_i})\Phi_{t_i}$ ***with*** $\mathbf{H}_{t_i} = \mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)^T}$ ***and*** Φ_{t_i} ***given in (41).***

Proof

To prove the theorem we make use of the Lemma 2 and we substitute $\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)$ in the main result of Lemma 1. More precisely from lemma A1 we have that:

$$\mathbf{u}_{t_i} = \lim_{dt\to 0}\left(\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\int \tilde{p}(\tau_i)\,\mathbf{H}_{t_i}^{-1}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right)d\tau_i\right).$$

The terms $\mathbf{R}^{-1}$ and $\mathbf{G}_{t_i}$ can be pushed insight the integral since they are independent of $\tau_i = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$. Thus we have the expression:

$$\mathbf{u}_{t_i} = \lim_{dt\to 0}\left(\int \tilde{p}(\tau_i)\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\mathbf{H}_{t_i}^{-1}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right)d\tau_i\right),$$

$$\mathbf{u}_{t_i} = \lim_{dt\to 0}\int \tilde{p}(\tau_i)\ \mathbf{u}_L^{(dt)}(\tau_i)\,d\tau_i,$$

where the local controls $\mathbf{u}_L^{(dt)}(\tau_i)$ are given as follows:

$$\mathbf{u}_L^{(dt)}(\tau_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\mathbf{H}_{t_i}^{-1}\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i).$$

After applying the limit, and making use of the result in Lemma 2 the equation above is expressed as:

$$\mathbf{u}_{t_i} = \int \tilde{p}(\tau_i)\,\mathbf{u}_L(\mathbf{x}_{t_{i+1}}, \mathbf{x}_{t_i})d\tau_i,$$

where the local controls $\mathbf{u}_L(\mathbf{x}_{t_{i+1}}, \mathbf{x}_{t_i})$ are given as follows:

$$\mathbf{u}_L(\tau_i) = \mathbf{u}_L(\mathbf{x}_{t_{i+1}}, \mathbf{x}_{t_i}) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\mathbf{H}_{t_i}^{-1}\left(\lim_{dt\to 0}\left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)})\frac{1}{dt} - \mathbf{f}_{t_i}^{(c)}\right) - \mathbf{b}_{t_i}\right),$$

or in a simpler form:

$$\mathbf{u}_L(\mathbf{x}_{t_{i+1}}, \mathbf{x}_{t_i}) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\mathbf{H}_{t_i}^{-1}\left(\mathbf{G}_c\varepsilon_{t_i} - \mathbf{b}_{t_i}\right).$$

By substituting with $\mathbf{H}(\mathbf{x}_{t_i}) = \mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}$ we have the final result:

$$\mathbf{u}_L(\tau_i) = \mathbf{u}_L(\mathbf{x}_{t_{i+1}}, \mathbf{x}_{t_i}) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\left(\mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\right)^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right).$$

■

Appendix B.

Theorem 4 *: Let us assume that* $\mathbf{x}$ *satisfies the SDE* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},t) + \mathbf{G}(\mathbf{x})\varepsilon(t)$. *Then* $\Psi(\mathbf{x},t_o) = \Psi(\mathbf{x},t_o,t_N) = E\left(\Psi(\mathbf{x},t_N)\, e^{\int_{t_o}^{t_N} -\frac{1}{\lambda} q(\mathbf{x}) d\tau}\right)$ *if and only if* $\Psi(\mathbf{x},t)$ *satisfies the backward Kolmogorov PDE:*

$$-\partial_t \Psi_t = -\frac{1}{\lambda} q_t \Psi_t + \mathbf{f}_t^T (\nabla_{\mathbf{x}} \Psi_t) + \frac{1}{2} trace\left((\nabla_{\mathbf{xx}} \Psi_t) \mathbf{G}_t \Sigma_{\varepsilon} \mathbf{G}_t^T\right),$$

with boundary condition:

$$\Psi(\mathbf{x},t_N) = \exp\left(-\frac{1}{\lambda}\phi(\mathbf{x}(t_N))\right).$$

Proof Given that $\mathbf{x}$ satisfies the SDE $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},t) + \mathbf{G}(\mathbf{x})\varepsilon(t)$ and $\Psi(\mathbf{x},t)$ satisfies the PDE above, application of Ito lemma (Øksendal, 2003) to function $Y(t) = \Psi(\mathbf{x}_t,t)\, e^{\int_{t_o}^{t} -\frac{1}{\lambda} q(\mathbf{x}) d\tau}$ leads to the final result $\Psi(\mathbf{x},t_o) = E\left(\Psi(\mathbf{x},t_N)\, e^{\int_{t_o}^{t_N} -\frac{1}{\lambda} q(\mathbf{x}) d\tau}\right)$. This result is the solution of the linear PDE. ■

References

S. Amari. Natural gradient learning for over- and under-complete bases in ica. *Neural Computation*, 11(8):1875–83, 1999.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5): 115–133, 1983.

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

R. Bellman and R. Kalaba. *Selected Papers On mathematical trends in Control Theory*. Dover Publications, 1964.

B. Van Den Broek, W. Wiegerinck, and H. J. Kappen. Graphical model inference in optimal control of stochastic multi-agent systems. *Journal of Artificial Intelligence Research*, 32(1):95–122, 2008.

J. Buchli, E. Theodorou, F. Stulp, and S. Schaal. Variable impedance control - a reinforcement learning approach. In *Robotics Science and Systems*, 2010.

P. Dayan and G. Hinton. Using em for reinforcement learning. *Neural Computation*, 9, 1997.

M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian Process Dynamic Programming. *Neurocomputing*, 72(7–9):1508–1524, March 2009.

W. H. Fleming and H. M. Soner. *Controlled Markov Processes and Viscosity Solutions*. Applications of aathematics. Springer, New York, 2nd edition, 2006.

M. Ghavamzadeh and E. Yaakov. Bayesian actor-critic algorithms. In *ICML '07: Proceedings of The 24th International Conference on Machine Learning*, pages 297–304, 2007.

V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3:671–692, 1990.

A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1547–1554. Cambridge, MA: MIT Press, 2003.

D. H. Jacobson and D. Q. Mayne. *Differential dynamic programming*. American Elsevier Pub. Co., New York,, 1970.

N. Jetchev and M Toussaint. Trajectory prediction: learning to map situations to robot trajectories. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 449–456, 2009.

H. J. Kappen. Linear theory for control of nonlinear stochastic systems. *Phys. Rev. Lett.*, 95:200201, Nov 2005a.

H. J. Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics: Theory and Experiment*, (11):P11011, 2005b.

H. J. Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. In J. Marro, P. L. Garrido, and J. J. Torres, editors, *Cooperative Behavior in Neural Systems*, volume 887 of *American Institute of Physics Conference Series*, pages 149–181, February 2007.

H. J. Kappen, W. Wiegerinck, and B. van den Broek. A path integral approach to agent planning. In *AAMAS*, 2007.

H. J. Kappen, Gmez V., and Opper M. Optimal control as a graphical model inference problem. *Journal for Machine Learning Research (JMLR)*, arXiv:0901.0633v, 2009. Submitted.

J. Koeber and J. Peters. Policy search for motor primitives. In D. Schuurmans, J. Benigio, and D. Koller, editors, *Advances in Neural Information Processing Systems 21 (NIPS 2008)*, pages 297–304, Vancouver, BC, Dec. 8-11, 2008. Cambridge, MA: MIT Press.

W. Thomas Miller, Richard S., and Paul J. Werbos. *Neural Networks for Control*. Neural network modeling and connectionism. MIT Press, Cambridge, Mass., 1990.

B. K. Øksendal. *Stochastic Differential Equations : An Introduction with Applications*. Springer, Berlin; New York, 6th edition, 2003.

J. Peters. *Machine Learning of Motor Skills for Robotics.* PhD thesis, University of Southern California, 2007.

J. Peters and S. Schaal. Policy gradient methods for robotics. In *Proceedings of the IEEE International Conference on Intelligent Robotics Systems (IROS 2006)*, 2006a.

J. Peters and S. Schaal. Reinforcement learning for parameterized motor primitives. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN 2006)*, 2006b.

J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–97, 2008a.

J. Peters and S. Schaal. Natural actor critic. *Neurocomputing*, 71(7-9):1180–1190, 2008b.

J. Peters and S. Schaal. Learning to control in operational space. *International Journal of Robotics Research*, 27:197–212, 2008c.

T. Rueckstiess, M. Felder, and J. Schmidhuber. State-dependent exploration for policy gradient methods. In *ECML PKDD '08: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Part II*, pages 234–249, 2008.

S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998.

L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Advanced textbooks in control and signal processing. Springer, London ; New York, 2000.

R. F. Stengel. *Optimal Control and Estimation*. Dover books on advanced mathematics. Dover Publications, New York, 1994.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive computation and machine learning. MIT Press, Cambridge, 1998.

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

E. Todorov. Stochastic optimal control and estimation methods adapted to the noise characteristics of the sensorimotor system. *Neural Computation*, 17(5):1084, 2005.

E. Todorov. Linearly-solvable markov decision problems. In B. Scholkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS 2007)*, Vancouver, BC, 2007. Cambridge, MA: MIT Press.

E. Todorov. General duality between optimal control and estimation. In *In Proceedings of the 47th IEEE Conference on Decision and Control*, 2008.

E. Todorov. Classic maximum principles and estimation-control dualities for nonlinear stochastic systems. 2009a. (Submitted).

E. Todorov. Efficient computation of optimal actions. *Proceedings National Academy of Science USA*, 106(28):11478–83, 2009b.

M. Toussaint and A. Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes, 2006.

N. Vlassis, M. Toussaint, G. Kontes, and Piperidis. S. Learning model-free control by a monte-carlo em algorithm. *Autonomous Robots*, 27(2):123–130, 2009.

W. Wiegerinck, B. van den Broek, and H. J. Kappen. Stochastic optimal control in continuous space-time multi-agent system. In *UAI*, 2006.

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

J. Yong. Relations among odes, pdes, fsdes, bsdes, and fbsdes. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 3, pages 2779–2784, Dec 1997.