Reinforcement Learning of Motor Skills in High Dimensions: A Path Integral Approach

Evangelos Theodorou, Jonas Buchli, and Stefan Schaal

***Abstract*— Reinforcement learning (RL) is one of the most general approaches to learning control. Its applicability to complex motor systems, however, has been largely impossible so far due to the computational difficulties that reinforcement learning encounters in high dimensionsal continuous state-action spaces. In this paper, we derive a novel approach to RL for parameterized control policies based on the framework of stochastic optimal control with path integrals. While solidly grounded in optimal control theory and estimation theory, the update equations for learning are surprisingly simple and have no danger of numerical instabilites as neither matrix inversions nor gradient learning rates are required. Empirical evaluations demonstrate significant performance improvements over gradient-based policy learning and scalability to high-dimensional control problems. Finally, a learning experiment on a robot dog illustrates the functionality of our algorithm in a real-world scenario. We believe that our new algorithm, Policy Improvement with Path Integrals ($\mathbf{PI}^2$), offers currently one of the most efficient, numerically robust, and easy to implement algorithms for RL in robotics.**

I. INTRODUCTION

While reinforcement learning (RL) is among the most general frameworks of learning control to create truely autonomous learning systems, its scalability to high-dimensional continuous state-action system, e.g., 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 [19], but can still become numerically brittle and full of open 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 approches 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 systems [5], [16], [22], [33], [6], [8].

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 [14], [4]. As will be detailed in the sections below, this approach make an appealing theoretical connection between value function approximation using the stochastic HJB equations and direct policy learning by approximating a path integral, i.e., by solving a statistical inference problem from sample roll-outs. The resulting algorithm, called **P**olicy **I**mprovement with **P**ath **I**ntegrals ($\mathbf{PI}^2$), takes on a surprisingly simple form, has no open tuning parameters besides the exploration noise, and performs numerically robustly in high dimensional learning problems. It also makes an interesting connection to previous work on RL based on probablity matching [5], [22], [16] and explains why probability matching algorithms can be successful.

In the next section, we first develop a generalized form of stochastic optimal control with path integrals. Second, from this formulation, we can derive the $\mathbf{PI}^2$ algorithm for probabilistic direct policy learning. Third, we discuss $\mathbf{PI}^2$ in the context of previous work on direct policy learning in the literature. In the evaluations, we will provide comparisons of different algorithms on learning control with parameterized dynamic systems policies [10], and we will demonstrate the application of $\mathbf{PI}^2$ to learning a complex jumping behavior on an actual robot dog.

II. STOCHASTIC OPTIMAL CONTROL WITH PATH INTEGRALS

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 [29]. Let us define a finite horizon reward function for a trajectory $\boldsymbol{\tau}_i$ starting at time t_i in state $\mathbf{x}_{t_i}$ and ending at time t_N

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

with $\phi_{t_N} = \phi(x_{t_N})$ denoting a terminal reward at time t_N and r_t denoting the immediate reward at time t. In stochastic optimal control [27], the goal is to find the controls $\mathbf{u}_t$ that minimize the value function:

$$V(\mathbf{x}_{t_i}) = V_t = \min_{\mathbf{u}_{t_i:t_N}} E_{\boldsymbol{\tau}_i}\left[R(\boldsymbol{\tau}_i)\right] \tag{2}$$

where the expection $E[.]$ is taken over all trajectories starting at $\mathbf{x}_{t_i}$. We consider the rather general control system

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

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 $\epsilon_t \in \Re^{p\times 1}$ Gaussian noise with variance $\boldsymbol{\Sigma}_\epsilon$. As immediate reward we consider

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

where $q_t = q(\mathbf{x}_t, t)$ is an arbitrary state-dependent reward function, and $\mathbf{R}$ is the positive semi-definite weight matrix of the quadratic control cost. The HJB equation [27],[7] associated with this stochastic optimal control problem is expressed as follows:

$$\partial_t V_t = q_t + (\partial_{\mathbf{x}} V_t)^T \mathbf{f}_t - \frac{1}{2}(\partial_{\mathbf{x}} V_t)^T \mathbf{G}_t \mathbf{R}^{-1} \mathbf{G}_t^T (\partial_{\mathbf{x}} V_t) + \frac{1}{2} trace\left((\partial_{\mathbf{xx}} V_t)\mathbf{G}_t \mathbf{\Sigma}_\epsilon \mathbf{G}_t^T\right) \quad (5)$$

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

Linearization of the HJB

The use of the logarithmic transformation of the value function $V_t = -\lambda \log \Psi_t$ as well as the assumption $\lambda \mathbf{G}_t \mathbf{R}^{-1} \mathbf{G}_t^T = \mathbf{G}_t \mathbf{\Sigma}_\epsilon \mathbf{G}_t^T = \mathbf{\Sigma}(\mathbf{x}_t) = \mathbf{\Sigma}_t$ results in:

$$-\partial_t \Psi_t = -\frac{1}{\lambda} q_t \Psi_t + \mathbf{f}_t^T (\partial_x \Psi_t) + \frac{1}{2} trace\left((\partial_{\mathbf{xx}} \Psi_t \mathbf{G}_t \mathbf{\Sigma}_\epsilon \mathbf{G}_t^T\right) \quad (6)$$

with boundary condition:$\Psi_{t_N} = \exp\left(-\frac{1}{\lambda}\phi_{t_N}\right)$. The PDE in (7) corresponds to the so called Kolmogorov backward PDE which is second order and linear. Analytical solutions of (7) cannot be found for the general case of a nonlinear system under any cost function. However, there is a connection between solutions of PDEs and their representation as stochastic differential equation (SDEs) which goes back to the Feynman Kac formula [17],[36]. The Feynman Kac formula can be used to find distributions of random processes which solve certain SDEs as well as to propose numerical methods for solving certain PDEs. According to Feyman Kac theorem the solution of (7) is:

$$\Psi_t = E\left(\Psi_{t_N} e^{-\int_{t_o}^{t_N} \frac{1}{\lambda} q_t d\tau}\right) = E\left[\exp\left(-\frac{1}{\lambda}\phi_{t_N}\right)\exp\left(-\frac{1}{\lambda}\int_{t_o}^{t_N} q_t \; dt\right)\right] \quad (7)$$

Generalized Path Integral Formulation

In many stochastic dynamical systems such as rigid body dynamics or DMPs the state $\mathbf{x} \in \Re^{n\times 1}$ can be partitioned into $\mathbf{x} = [\mathbf{x}^{(m)} \;\; \mathbf{x}^{(c)}]$ with $\mathbf{x}^{(m)} \in \Re^{k\times 1}$ and $\mathbf{x}^{(c)} \in \Re^{l\times 1}$ the part of the state that is directly controllable. The control variable $\mathbf{u} \in \Re^{p\times 1}$ has dimensionality smaller than the state $p < n$. Moreover the term $\mathbf{f}_{t_i}$ and the control transition matrix can be partitioned as $\mathbf{f}_{t_i}^T = [\mathbf{f}_{t_i}^{(m)T} \;\; \mathbf{f}_{t_i}^{(c)T}]^T$ with $\mathbf{f}_m \in \Re^{k\times 1}$, $\mathbf{f}_c \in \Re^{l\times 1}$ and $\mathbf{G}_{t_i} = [\mathbf{0}_{k\times p} \;\; \mathbf{G}_{t_i}^{(c)T}]^T$ with $\mathbf{G}_{t_i}^c \in \Re^{l\times p}$. For such systems it can been shown that the solution of (7) is

$$\Psi_{t_i} = \int p(\boldsymbol{\tau}_i) \exp\left(-\frac{1}{\lambda}\left(\phi_{t_N} + \int_{t_i}^{t_N} q_t dt\right)\right) d\boldsymbol{\tau}_i^{(c)} \quad (8)$$

where the transition probability of a trajectory $p(\boldsymbol{\tau}_i)$, under a Gaussian noise ϵ assumption, can be written with only the controlled states $\mathbf{x}^{(c)}$:

$$\begin{aligned} p(\boldsymbol{\tau}_i) &= \Pi_{j=i}^{N-1} p\left(\mathbf{x}_{t_{j+1}}^{(c)} | \mathbf{x}_{t_j}^{(m)}, \mathbf{x}_{t_j}^{(c)}\right) \\ &= \Pi_{j=i}^{N} \left(2\pi |\mathbf{\Sigma}_{t_j}|\right)^{-l/2} \exp\left(-\frac{1}{2\lambda}\sum_{j=i}^{N-1} \gamma_{t_j} dt\right) \end{aligned} \quad (9)$$

Since $\boldsymbol{\tau}_i = \left(\mathbf{x}_{t_i}, \mathbf{x}_{t_{i+1}}, ..., \mathbf{x}_{t_N}\right)$ are sample paths starting at state $\mathbf{x}_{t_i}$ the integration above is taken with respect to $\boldsymbol{d\tau}_i^{(c)} = \left(\mathbf{dx}_{t_1}^{(c)}, ..., \mathbf{dx}_{t_N}^{(c)}\right)$. In the previous equation, the quantity γ is defined as $\gamma_{t_j} = \boldsymbol{\alpha}_{t_j}^T \mathbf{h}_{t_j}^{-1} \boldsymbol{\alpha}_{t_j}$ and the terms $\mathbf{h}_{t_j} \in \Re^{l\times l}$ and $\boldsymbol{\alpha}_{t_j} \in \Re^{l\times 1}$ are expressed as $\mathbf{h}_{t_j} = \mathbf{h}(\mathbf{x}_{t_j}) = \mathbf{G}_{t_j}^{(c)} \mathbf{R}^{-1} \mathbf{G}_{t_j}^{(c)T}$ and $\boldsymbol{\alpha}_{t_j} = \boldsymbol{\alpha}(\mathbf{x}_{t_{j+1}}^{(c)}, \mathbf{x}_{t_j}) = \mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)} - \mathbf{f}_{t_j}^{(c)} dt$. Substitution of the transition probability into (8) results in:

$$\Psi_{t_i} = \lim_{dt\to 0} \int \frac{1}{D(\boldsymbol{\tau}_i)} e^{-\frac{1}{\lambda} S(\boldsymbol{\tau}_i)} d\boldsymbol{\tau}_i = \lim_{dt\to 0} \int e^{-\frac{1}{\lambda} Z(\boldsymbol{\tau}_i)} d\boldsymbol{\tau}_i \quad (10)$$

where $Z(\boldsymbol{\tau}_i)$ is defined as $Z(\boldsymbol{\tau}_i) = S(\boldsymbol{\tau}_i) + \lambda \log D(\boldsymbol{\tau}_i)$. The terms $D(\boldsymbol{\tau}_i)$ and $S(\boldsymbol{\tau}_i)$ are defined as $D(\boldsymbol{\tau}_i) = \Pi_{j=i}^{N} \left(2\pi |\mathbf{\Sigma}_{t_j}|\right)^{l/2}$ and $S(\boldsymbol{\tau}_i)$ is expressed as

$$S(\boldsymbol{\tau}_i) = -\phi_{t_N} - \frac{1}{2}\sum_{j=i}^{N-1} \gamma_{t_j} dt - \sum_{j=i}^{t_N} q_{t_j} dt \quad (11)$$

By taking the limit as $dt \to 0$ we can calculate the logarithmic value function Ψ_{t_i} which is the solution of (7). The term $Z(\boldsymbol{\tau}_i)$ is the total cost of the sample path $\boldsymbol{\tau}_i$. A factorization of the cost $Z(\boldsymbol{\tau}_i)$ in path depended and constant terms will simplify the derivation of optimal control.

$$Z(\boldsymbol{\tau}_i) = S(\boldsymbol{\tau}_i) + \frac{\lambda}{2}\sum_{j=i}^{N-1} \log |\mathbf{h}_{t_j}| + \frac{\lambda N l}{2} \log\left(2\pi dt \lambda\right) \quad (12)$$

The constant term in the equation is problematic since $\lim_{dt\to 0} \frac{\lambda N l}{2} \log\left(2\pi dt\lambda\right) = \infty$. However, as it is shown in the next section, the constant term drops and a new quantity $\tilde{S}(\boldsymbol{\tau}_i)$ which depends only on the state dependent terms of $Z(\boldsymbol{\tau}_i)$ will play the role of cost per sampled path.

Optimal Controls

The optimal controls are given as $\mathbf{u}_{t_i} = -\mathbf{R}^{-1} \mathbf{G}_{t_i}^T (\partial_{x_{t_i}} V_{t_i})$. Due to the logarithmic transformation of the value function, the equation of the optimal controls can be written as $\mathbf{u}_{t_i} = \lambda \mathbf{R}^{-1} \mathbf{G}_{t_i} (\partial_{\mathbf{x}_{t_i}} \Psi_{t_i}) / \Psi_{t_i}$. After substituting Ψ_{t_i} with (10) and dropping the state independent terms of the cost we have:

$$\mathbf{u}_{t_i} = \lim_{dt\to 0}\left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\partial_{\mathbf{x}_{t_i}}\left(\int e^{-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)} \boldsymbol{d\tau}_i\right)}{\int \; e^{-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)} \boldsymbol{d\tau}_i}\right) \quad (13)$$

with $\tilde{S}(\boldsymbol{\tau}_i) = S(\boldsymbol{\tau}_i) + \frac{\lambda}{2}\sum_{i=0}^{N-1} \log |\mathbf{h}_{t_i}|$. Further analysis of the equation above leads to a simplified version of the equation for optimal controls formulated as $\mathbf{u}_{t_i} =$

$\int P(\boldsymbol{\tau}_i)\,\mathbf{u}(\boldsymbol{\tau}_i)\,d\boldsymbol{\tau}_i$ with the probability $P(\boldsymbol{\tau}_i)$ and local controls $\mathbf{u}(\boldsymbol{\tau}_i)$ defined as

$$P(\boldsymbol{\tau}_i) = \frac{e^{\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)}}{\int e^{\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)} d\boldsymbol{\tau}_i} \tag{14}$$

$$\mathbf{u}(\boldsymbol{\tau}_i) = -\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T} \lim_{dt\to 0}\left(\partial_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\boldsymbol{\tau}_i)\right) \tag{15}$$

The path cost $\tilde{S}(\boldsymbol{\tau}_i)$ is a generalized version of the path cost in [11], [12], which only considered systems with state independent control transition. [1] To find the local controls $\mathbf{u}(\boldsymbol{\tau}_i)$ we have to calculate the $\lim_{dt\to 0}\partial_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\boldsymbol{\tau}_i)$. Due to space limitations, we do not provide the detailed derivations. The final result is:

$$\lim_{dt\to 0}\left(\partial_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\boldsymbol{\tau}_i)\right) = -\mathbf{h}_{t_i}^{-1}\left(\mathbf{G}_{t_i}^{(c)}\boldsymbol{\epsilon}_{t_0} - \mathbf{b}_{t_i}\right) \tag{16}$$

where the new term $\mathbf{b}_{t_i}$ is expressed as $\mathbf{b}_{t_i} = \lambda \mathbf{h}_{t_i}\boldsymbol{\Phi}_{t_i}$ and $\boldsymbol{\Phi}_{t_i} \in \Re^{l\times 1}$ a vector with the jth element defined as:

$$(\boldsymbol{\Phi}_{t_i})_j = trace\left(\mathbf{h}_{t_i}^{-1}\cdot \partial_{\mathbf{x}_{t_i}^{(cj)}}\mathbf{h}_{t_i}\right) \tag{17}$$

The local control can now be expressed as:

$$\mathbf{u}(\boldsymbol{\tau}_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\mathbf{h}_{t_i}^{-1}\left(\mathbf{G}_{t_i}^{(c)}\boldsymbol{\epsilon}_{t_0} - \mathbf{b}_{t_i}\right) \tag{18}$$

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:

$$\begin{aligned}\mathbf{u}(\boldsymbol{\tau}_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}&\left(\mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\right)^{-1} \\ &\times\left(\mathbf{G}_{t_i}^{(c)}\boldsymbol{\epsilon}_{t_0} - \mathbf{b}_{t_i}\right)\end{aligned} \tag{19}$$

Previous work in [15], [11], [12], [4] are special cases of our generalized formulation. [2]

III. Parametrized Policies

Equipped with the theoretical framework of stochastic optimal control with path integrals, we can now turn to its application to reinforcement learning with parametrized policies. For this kind of direct policy learning, a general cost function $J = \int_{\boldsymbol{\tau}} p(\boldsymbol{\tau})R(\boldsymbol{\tau})d\boldsymbol{\tau}$ is usually assumed [19] and optimized over state $\mathbf{x}_t$ and action $\mathbf{a}_t$ trajectories $\boldsymbol{\tau} = (\mathbf{x}_{t_0}, \mathbf{a}_{t_o}, ..., \mathbf{x}_{t_N},)$. Under the Markov property, the probability of a trajectory is $p(\boldsymbol{\tau}) = p(\mathbf{x}_{t_o})\Pi_{i=1}^{N-1}p(\mathbf{x}_{t_{i+1}}|\mathbf{x}_{t_i},\mathbf{a}_{t_i})p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i})$. As suggested in [16], the mean of the stochastic policy $p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i})$ is linearly parametrized as:

$$\mathbf{a}_{t_i} = \mathbf{g}_{t_i}^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{t_i}) \tag{20}$$

[1] More precisely if $\mathbf{G}_{t_i}^{(c)} = \mathbf{G}^c$ then the term $\frac{\lambda}{2}\sum_{i=0}^{N-1}\log|\mathbf{h}_{t_i}|$ drops since it is state independent and it appears in both nominator and denominator in (14). In this case, the path cost is reduced to $\tilde{S}(\boldsymbol{\tau}_i) = S(\boldsymbol{\tau}_i)$.

[2] In fact, for stochastic systems with state independent control transition matrix $\mathbf{G}_{t_i}^{(c)} = \mathbf{G}^{(c)}$ the term $\mathbf{b}_{t_i} = 0_{l\times 1}$ since $\mathbf{h}_{t_i}$ becomes state independent and therefore $\partial_{\mathbf{x}_{t_i}^{(cj)}}\mathbf{h}_{t_i} = 0$. In such case the local controls are reduced to $\mathbf{u}(\boldsymbol{\tau}_i) = \boldsymbol{\epsilon}_{t_0}$.

TABLE I

Pseudocode of the $\mathbf{PI}^2$ algorithm for a 1D Parameterized Policy (Note that the discrete time step dt was absobed as a constant multiplier in the cost terms).

- **Given**:
  - An immediate cost function $r_t = q_t + \boldsymbol{\theta}_t^T\mathbf{R}\boldsymbol{\theta}_t$
  - A terminal cost term ϕ_{t_N}
  - A stochastic parameterized policy $\mathbf{a}_t = \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_t)$ (cf. 20)
  - The basis function $\mathbf{g}_{t_i}$ from the system dynamics
  - The variance Σ_ϵ of the mean-zero noise $\boldsymbol{\epsilon}_t$
  - The initial parameter vector $\boldsymbol{\theta}$
- **Repeat** until convergence of the trajectory cost R:
  - **step 1:** Create K roll-outs of the system from the same start state $\mathbf{x}_0$ using stochstic parameters $\boldsymbol{\theta} + \boldsymbol{\epsilon}_t$ at every time step
  - **step 2: For** all K roll-outs, compute:
    - **step 2.1:** $\mathbf{M}_{t_j,k} = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_j,k}\ \mathbf{g}_{t_j,k}^T}{\mathbf{g}_{t_j,k}^T\mathbf{R}^{-1}\mathbf{g}_{t_j,k}}$
    - **step 2.2:** Compute the cost for each sampled trajectory: $S(\boldsymbol{\tau}_{i,k}) = \phi_{t_N,k} + \sum_{j=i}^{N-1} q_{t_j,k} + \frac{1}{2}\sum_{j=i+1}^{N-1}(\boldsymbol{\theta} + \mathbf{M}_{t_j,k}\boldsymbol{\epsilon}_{t_j,k})^T\mathbf{R}(\boldsymbol{\theta} + \mathbf{M}_{t_j,k}\boldsymbol{\epsilon}_{t_j,k})$
    - **step 2.3:** $P(\boldsymbol{\tau}_{i,k}) = \frac{e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,k})}}{\sum_{k=1}^{K}[e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,k})}]}$
  - **step 3: For** all i time steps, compute:
    - **step 3.1:** $\delta\boldsymbol{\theta}_{t_i} = \sum_{k=1}^{K}\left[P(\boldsymbol{\tau}_{i,k})\,\mathbf{M}_{t_i,k}\ \boldsymbol{\epsilon}_{t_i,k}\right]$
  - **step 4:** Compute $\delta\boldsymbol{\theta} = \frac{\sum_{i=0}^{N-1}(N-i)\quad\delta\boldsymbol{\theta}_{t_i}}{\sum_{i=0}^{N-1}(N-i)}$
  - **step 5:** Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$
  - **step 6:** Create one noiseless roll-out to check the trajectory cost $R = \phi_{t_N} + \sum_{i=0}^{N-1} r_{t_i}$

where $\mathbf{g}_{t_i}$ is a vector of basis functions and $\boldsymbol{\theta}$ is a parameter vector. For Gaussian noise $\boldsymbol{\epsilon}$ the policy distribution is $p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i}) = N\left(\boldsymbol{\theta}^T\mathbf{g}_{t_i}, \boldsymbol{\Sigma}_{t_i}\right)$. In our work, we use a special case of parametrized policies in form of Dynamic Movement Primitives (DMPs) [10], which are expressed as:

$$\begin{aligned}\frac{1}{\tau}\dot{z}_t &= f_t + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_t), \\ \frac{1}{\tau}\dot{y}_t &= z_t, \quad \frac{1}{\tau}\dot{x}_t = -\alpha x_t\end{aligned} \tag{21}$$

with $f_t = \alpha_z(\beta_z(g - y_t) - z_t)$. 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 – for more details and the definition of the basis functions $\mathbf{g}_t$ see [10]. The DMP equations are obviously of the form of our control system (3), just with a row vector as control transition matrix $\mathbf{G}_t^{(c)} = \mathbf{g}_t{}^T \in \Re^{1\times p}$. Thus, we can treat the parameters θ as if they were control commands, and, after some algebra and simplifications, we derive the **P**olicy **I**mprovement with **P**ath **I**ntegrals ($\mathbf{PI}^2$) which is summarized in table I.

Essentially, in step (2.3), the term $P(\boldsymbol{\tau}_{i,k})$ is discrete probability at time t_i of each trajectory roll-out that is computed with the help of the cost $S(\boldsymbol{\tau}_{i,k})$ step (2.2). For every time step of the trajectory, a parameter update is computed $\delta\boldsymbol{\theta}_{t_i}$ in step (3.1) based on a probability weighted average cost over the sampled trajectories. The parameter updates at every time step are finally averaged in step

(4) by giving every parameter update a weight according to the time steps left in the trajectory, and the parameter update $\boldsymbol{\theta}^{(new)} = \boldsymbol{\theta}^{(old)} + \delta\boldsymbol{\theta}$ takes place in step (5). The entire formulation allows an interactive updating of θ. 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. Moreover, a constant term can be subtracted from the cost $S(\boldsymbol{\tau}_i)$ as long as all $S(\boldsymbol{\tau}_i)$ is positive. Thus, for a given number of roll-outs, we compute the exponential term in (14) as

$$e^{\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)} = \exp\left(-c\frac{\tilde{S}(\boldsymbol{\tau}_i) - \min\tilde{S}(\boldsymbol{\tau}_i)}{\max\tilde{S}(\boldsymbol{\tau}_i) - \min\tilde{S}(\boldsymbol{\tau}_i)}\right) \quad (22)$$

with $c = 10$ in all our evaluations – this procedure eliminates λ and leaves the variance of the exploration noise ϵ as the only open parameter for $\mathbf{PI}^2$. It should be noted that the equations for $\mathbf{PI}^2$ have no numerical pitfalls: no matrix inversions and no learning rates [3], rendering $\mathbf{PI}^2$ to be very easy to use.

IV. Evaluations

We evaluated $\mathbf{PI}^2$ in several synthetic examples in comparison with REINFORCE, GPOMDP, eNAC, [20], [21]. Another interesting algorithm is the PoWER algorithm [16], which is a probabilistic policy improvement method, not a gradient algorithm. It is derived from an Expectation-Maximization framework using probability matching [5], [22] Except for PoWER, all algorithms are suitable for optimizing immediate reward functions of the kind $r_t = q_t + \mathbf{u}_t\mathbf{R}\mathbf{u}_t$. PoWER requires that the immediate reward behaves like an improper probability, i.e., the rewards integrate to a finite number. This property is incompatible with $r_t = q_t + \mathbf{u}_t\mathbf{R}\mathbf{u}_t$ and requires some special non-linear transformations, which usually change the nature of the optimization problem, such that PoWER optimizes a different cost function. Thus, we excluded PoWER from a direct comparison in our evaluations. 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[4], and the noise was kept constant for the entire time that this basis function had the highest activation – empirically, this trick helped improving the learning speed of all algorithms – we do not illustrate such little "tricks" in this paper as they really only affect fine tuning of the algorithm.

A. Learning Optimal Performance of a Multi-DOF Via-Point Task

In our first evaluation we examine the scalability of our algorithms to a high-dimensional and highly redundant learning problem. We assume that the multi-DOF systems are models of planar robot arms, where $d = 2, 10$ or $d = 50$ 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 1b,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 (the endeffector) touches the y-axis. The higher priority task, however, is to move the endeffector 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 (22) reads now as $\ddot{\xi}_{i,t} = f_{i,t} + \mathbf{g}_{i,t}^T(\boldsymbol{\theta}_i + \boldsymbol{\epsilon}_{i,t})$ – this small change of notation is to avoid a clash of variables with the $(x.y)$ task space of the robot. The endeffector position is computed as:

$$x_t = \frac{1}{d}\sum_{i=1}^{d} cos(\sum_{j=1}^{i}\xi_{j,t}), \quad y_t = \frac{1}{d}\sum_{i=1}^{d} sin(\sum_{j=1}^{i}\xi_{j,t}) \quad (23)$$

The immediate reward function for this problem is defined as

$$\begin{aligned} r_t &= \frac{\sum_{i=1}^{d}(d+1-i)\left(0.1f_{i,t}^2 + 0.5\ \boldsymbol{\theta}_i^T\boldsymbol{\theta}\right)}{\sum_{i=1}^{d}(d+1-i)} \quad (24)\\ \Delta r_{300ms} &= 10^8\left((0.5 - x_{t_{300ms}})^2 + (0.5 - y_{t_{300ms}})^2\right)\\ \phi_{t_N} &= 0 \end{aligned}$$

where Δr_{300ms} is added to r_t at time $t = 300ms$, i.e., we would like to pass through the via-point at this time. The individual DOFs of the motor primitive were initialized to code a 5th order spline from the start position to the end position. The cost term in (24) 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 (24) 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 1. The learning curves in the left column demonstrate that $\mathbf{PI}^2$ has an order of magnitude faster learning performance than the other algorithms, irrespective of the dimensionality. $\mathbf{PI}^2$ also converges to the lowest cost in all examples:

[3] $\mathbf{R}$ is a user design parameter and usually chosen to be diagonal and invertible.

[4] I.e., the noise vector in (20) has only one non-zero component.

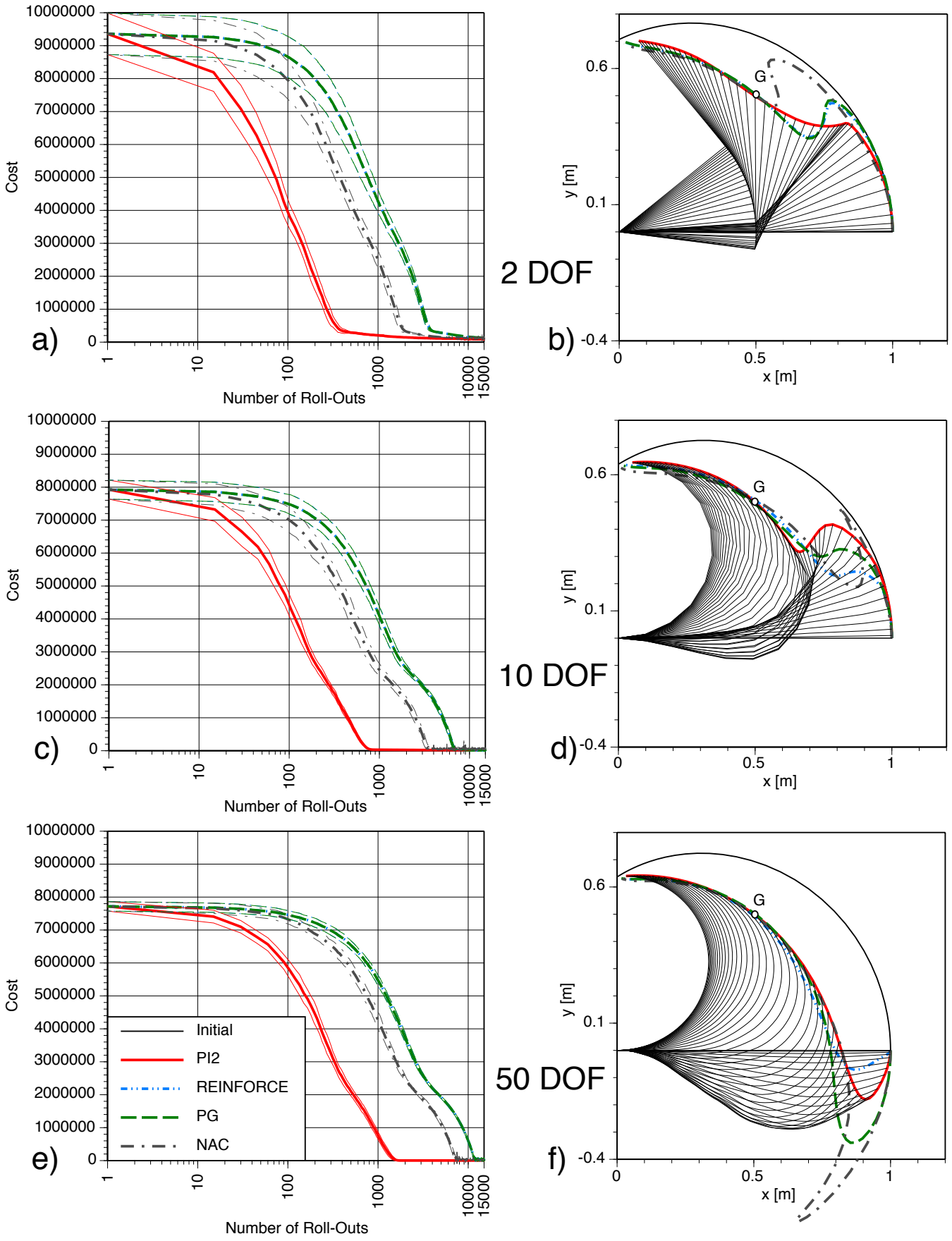


Fig. 1. Comparision of learning multi-DOF movements (2,10, and 50 DOFs) with planar robot arms passing through a via-point G. a,c,e) illustrate the learning curves for different RL algorithms, while b,d,f) illustrate the endeffector movement after learning for all algorithms. Additionally, b,d,f) also show the initial endeffector 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).

Algorithm	2-DOFs	10-DOFs	50-DOFs
$\mathbf{PI}^2$	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 1 also illustrates the path taken by the endeffector 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 $\mathbf{PI}^2$ 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 $\mathbf{PI}^2$ solution, which proceeds from the very right to the left as a function of time. It should be emaphasized that there was absolutely no parameter tuning needed to achieve the $\mathbf{PI}^2$ results, while all gradient algorithms required readjusting of learning rates for every example to achieve best performance.

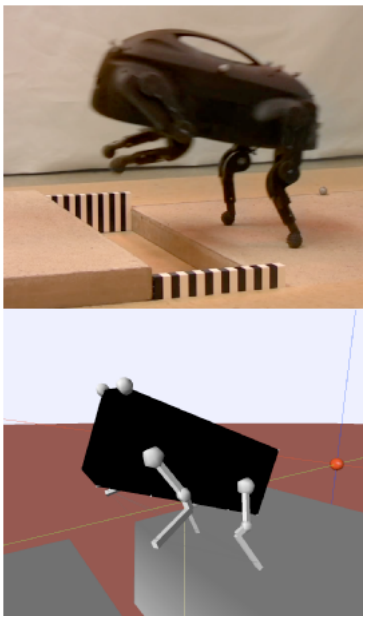
B. Application to Robot Learning

Figure 2 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. 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 3-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.

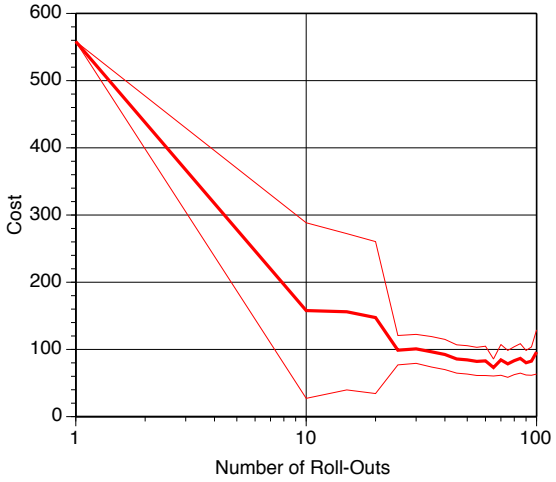
$\mathbf{PI}^2$ learning used primarily the forward progress as a reward, and slightly penalized the squared acceleration of each DOF, and the length of the parameter vector. Additionaly, a penalty was incurred if the yaw or the roll exceeded a threshold value – these penalties encouraged the robot to jump straight forward and not to the side, and not to fall over. The exact cost function was:

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

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 parameters a_1 and a_2 are tuned as $(a_1 = 1.e-6, a_2 = 1.e-8)$ 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 peforms importance sampling, as the rewards for the 5 trials in memory were re-computed 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. Learning was performed on a physical simulator of the robot dog, as the real robot dog was not available for this experiment.



(a) Real & Simulated Robot Dog



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

Fig. 2. 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.

Figure 2 illustrates that after about 30 trials (i.e., 5 updates), the performance of the robot was significantly improved, such that after the jump, almost the entire body

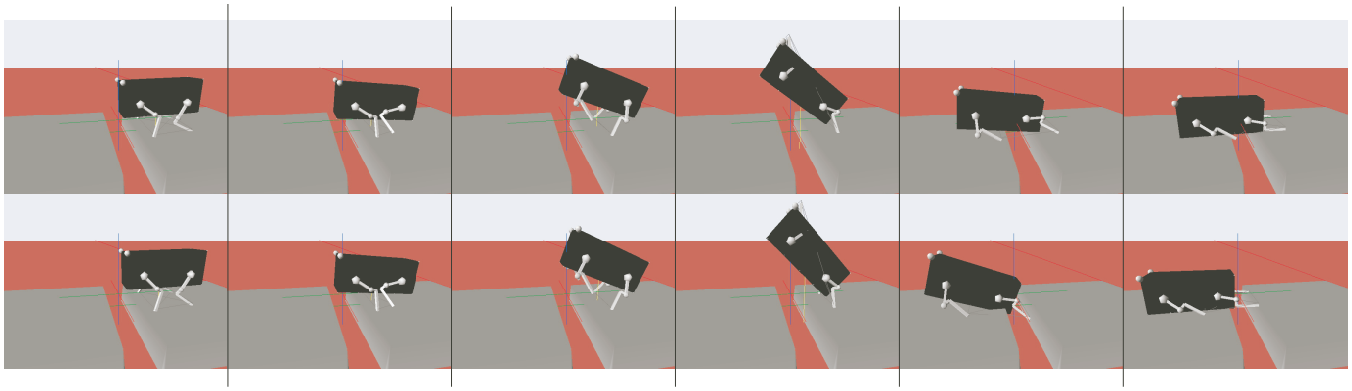


Fig. 3. Sequence of images from the simulated robot dog jumping over a 14cm gap. Top: before learning. Bottom: After learning. While the two sequences look quite similar at the first glance, it is apparent that in the 4th frame, the robot's body is significantly heigher in the air, such that after landing, 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 continution of walking.

was lying on the other side of the gap. Figure 3 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.

V. Conclusions

This paper derived a slightly more general version of stochastic optimal control with path integrals, based on the original work by [12], [4]. The key results were presented in Section II, 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 parametrized policy. $\mathbf{PI}^2$ inherits its sound foundation in first order principles of stochastic optimal control from the path integral formalizm. It is a probabilistic learning method without open 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. Our evaluations demonstrated that $\mathbf{PI}^2$ can scale to high dimensional control systems, unlike many other reinforcement learning systems.

VI. 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.

References

[1] S. Amari. Natural gradient learning for over- and under-complete bases in ica. *Neural Comput*, 11(8):1875–83, 1999. 20047135 0899-7667 Journal Article.

[2] 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.

[3] Jonathan Baxter and Peter L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[4] B. van den Broek, W. Wiegerinck, and B. Kappen. Graphical model inference in optimal control of stochastic multi-agent systems. *Journal of Artificial Intelligence Research*, 32(1):95–122, 2008.

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

[6] Marc P. Deisenroth, Carl E. Rasmussen, and Jan Peters. Gaussian Process Dynamic Programming. *Neurocomputing*, 72(7–9):1508–1524, March 2009.

[7] Wendell Helms Fleming and H. Mete Soner. *Controlled Markov processes and viscosity solutions*. Applications of mathematics. Springer, New York, 2nd edition, 2006.

[8] Mohammad Ghavamzadeh and Yaakov Engel. Bayesian actor-critic algorithms. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 297–304, New York, NY, USA, 2007. ACM.

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

[10] 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. clmc.

[11] H J Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(11):P11011, 2005a. URL `http://stacks.iop.org/1742-5468/2005/P11011`.

[12] 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.

[13] H J Kappen, Gomez V., and M. Opper. Optimal control as graphical model inference problem. *(Submitted)*, 2009.

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

[15] Hilbert J. Kappen. Linear theory for control of nonlinear stochastic systems. *Phys. Rev. Lett.*, 95(20):200201, Nov 2005b.

[16] J. Koeber and J. Peters. Learning motor primitives in robotics. In D. Schuurmans, J. Benigio, and D. Koller, editors, *Advances in Neural*

Information Processing Systems 21 (NIPS 2008), Vancouver, BC, Dec. 8-11, 2009. Cambridge, MA: MIT Press. clmc.

[17] B. K. ksendal. *Stochastic differential equations : an introduction with applications*. Universitext. Springer, Berlin ; New York, 6th edition, 2003.

[18] W. Thomas Miller, Richard S. Sutton, and Paul J. Werbos. *Neural networks for control*. Neural network modeling and connectionism. MIT Press, Cambridge, Mass., 1990. r

[19] J. Peters. *Machine learning of motor skills for robotics.* PhD thesis, 2007.

[20] J. Peters and S. Schaal. Policy gradient methods for robotics. In *Proceedings of the IEEE International Conference on Intelligent Robotics Systems (IROS 2006)*, Beijing, Oct. 9-15, 2006a. clmc.

[21] 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. clmc.

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

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

[24] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Netw*, 21(4):682–97, 2008c. clmc Journal Article United States the official journal of the International Neural Network Society.

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

[26] L. Sciavicco and Bruno Siciliano. *Modelling and control of robot manipulators*. Advanced textbooks in control and signal processing. Springer, New York, 2000.

[27] Robert F. Stengel. *Optimal control and estimation*. Dover books on advanced mathematics. Dover Publications, New York, 1994.

[28] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, and K.-R. Mller, editors, *Advances in Neural Processing Systems 12*, Denver, CO, 2000. MIT Press.

[29] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning : An introduction*. Adaptive computation and machine learning. MIT Press, Cambridge, 1998.

[30] 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.

[31] E. Todorov. General duality between optimal control and estimation. In *proceedings of the 47th ieee conf. on decision and control*, 2008.

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

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

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

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

[36] Jiongmin Yong. Relations among odes, pdes, fsdes, bsdes, and fbsdes. volume 3, pages 2779–2784 vol.3, Dec 1997.