

# An Iterative Path Integral Reinforcement Learning Approach

Evangelos Theodorou, Jonas Buchli, Freck Stulp, and Stefan Schaal

Although significant progress has been made in learning control from trajectory roll-outs, performing Reinforcement Learning in high dimensional and continuous state action spaces remains a challenging problem. Especially in the domain of humanoid robot control, the high dimensionality, the lack of accurate dynamical models, and the hybrid and unstable nature of the underlying dynamics due to contact with the environment make the application of traditional model based and model free Reinforcement Learning methods rather difficult.

In this short paper we present a new probabilistic method [1] for reinforcement learning that is derived based from the Bellman principle for *continuous state-action spaces*. Under the constraint of nonlinear stochastic dynamics, the result of the Bellman principle is the stochastic Hamilton Jacobi Bellman (HJB) equation, which is a nonlinear and second order Partial Differential Equation (PDE). Exponentiation of the value function and the use of mild assumptions on the exploration noise and control cost transforms the nonlinear PDE into a linear and second order PDE. The resulting PDE is the so called Backward Chapman Kolmogorov PDE. The use of the first of the nonlinear Feynman Kac Lemmas provides a bridge between Forward Stochastic Differential Equations (FSDEs) and Partial Differential Equations (PDEs) [2] of the type of a Backward Chapman Kolmogorov PDE. The FSDEs are used for sampling parts of the state space which are relevant to the task goal. Moreover sampling is performed in an iterative fashion to localize exploration and avoid the curse of dimensionality. The resulting algorithm, called **Policy Improvement with Path Integrals (PI<sup>2</sup>)**, takes on a surprisingly simple form, has no algorithmic 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 probability matching [3], [4], [5] and explains why probability matching algorithms can be successful.

## Policy Improvement with Path Integrals

Let us define a finite-horizon reward function for a trajectory  $\tau_i$  starting at time  $t_i$  in state  $\mathbf{x}_{t_i}$  and ending at time  $t_N$  as  $R(\tau_i) = \phi_{t_N} + \int_{t_i}^{t_N} r_t dt$  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  $t$ . The goal is to find the controls  $\theta_t$  that minimize the cost function  $V(\mathbf{x}_{t_i}) = V_t = \min_{\theta_{t_i:t_N}} E_{\tau_i} [R(\tau_i)]$  where the expectation  $E[\cdot]$  is taken over all trajectories starting at  $\mathbf{x}_{t_i}$ . We consider the rather general control system  $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}_t, t) + \mathbf{G}(\mathbf{x}_t)(\theta_t + \epsilon_t) = \mathbf{f}_t + \mathbf{G}_t(\theta_t + \epsilon_t)$  with  $\mathbf{x}_t \in \mathbb{R}^{n \times 1}$  denoting the state of the system,  $\mathbf{G}_t = \mathbf{G}(\mathbf{x}_t) \in \mathbb{R}^{n \times p}$  the control matrix,  $\mathbf{f}_t =$

TABLE I  
PSEUDOCODE OF THE **PI<sup>2</sup>** ALGORITHM FOR A 1D PARAMETERIZED POLICY (NOTE THAT THE DISCRETE TIME STEP  $dt$  WAS ABSORBED AS A CONSTANT MULTIPLIER IN THE COST TERMS).

- 
- **Given:**
    - An immediate cost function  $r_t = q_t + \theta_t^T \mathbf{R} \theta_t$
    - A terminal cost term  $\phi_{t_N}$
    - A stochastic parameterized policy  $\mathbf{a}_t = \mathbf{g}_t^T(\theta + \epsilon_t)$
    - The basis function  $\mathbf{g}_t$  from the system dynamics
    - The variance  $\Sigma_\epsilon$  of the mean-zero noise  $\epsilon_t$
    - The initial parameter vector  $\theta$
  - **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 stochastic parameters  $\theta + \epsilon_t$  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:  $S(\tau_{i, k}) = \phi_{t_N, k} + \sum_{j=i}^{N-1} q_{t_j, k} + \frac{1}{2} \sum_{j=i+1}^{N-1} (\theta + \mathbf{M}_{t_j, k} \epsilon_{t_j, k})^T \mathbf{R} (\theta + \mathbf{M}_{t_j, k} \epsilon_{t_j, k})$
      - \* **step 2.3:**  $P(\tau_{i, k}) = \frac{e^{-\frac{1}{\lambda} S(\tau_{i, k})}}{\sum_{k=1}^K [e^{-\frac{1}{\lambda} S(\tau_{i, k})}]}$
    - **step 3:** For all  $i$  time steps, compute:
      - \* **step 3.1:**  $\delta \theta_{t_i} = \sum_{k=1}^K [P(\tau_{i, k}) \mathbf{M}_{t_i, k} \epsilon_{t_i, k}]$
    - **step 4:** Compute  $\delta \theta^m = \frac{\sum_{i=0}^{N-1} (N-i) w_{t_i}^m \delta \theta_{t_i}^m}{\sum_{i=0}^{N-1} (N-i) w_{t_i}^m}$
    - **step 5:** Update  $\theta \leftarrow \theta + \delta \theta$
    - **step 6:** Create one noiseless roll-out to check the trajectory cost  $R = \phi_{t_N} + \sum_{i=0}^{N-1} r_{t_i}$
- 

$\mathbf{f}(\mathbf{x}_t) \in \mathbb{R}^{n \times 1}$  the passive dynamics,  $\theta_t \in \mathbb{R}^{p \times 1}$  the control vector and  $\epsilon_t \in \mathbb{R}^{p \times 1}$  Gaussian noise with variance  $\Sigma_\epsilon$ . As immediate reward we consider  $r_t = r(\mathbf{x}_t, \theta_t, t) = q_t + \theta_t^T \mathbf{R} \theta_t$  where  $q_t = q(\mathbf{x}_t, t)$  is an arbitrary state-dependent reward function, and  $\mathbf{R}$  the positive definite weight matrix of the quadratic control cost.

The **PI<sup>2</sup>** algorithm is given in I. In step (2.3), the term  $P(\tau_{i, k})$  is the discrete probability at time  $t_i$  of each trajectory roll-out that is computed with the help of the cost  $S(\tau_{i, k})$  step (2.2). For every time step of the trajectory, a parameter update  $\delta \theta_{t_i}$  is computed 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) and the parameter update  $\theta^{(new)} = \theta^{(old)} + \delta \theta$  takes place in step (5).

## Evaluations

We have evaluated **PI<sup>2</sup>** in three different tasks and experimental platforms. In the first evaluation in Figure 1 we are comparing **PI<sup>2</sup>** against Policy Gradient Methods in the task of reaching a goal state by passing via an intermediate target

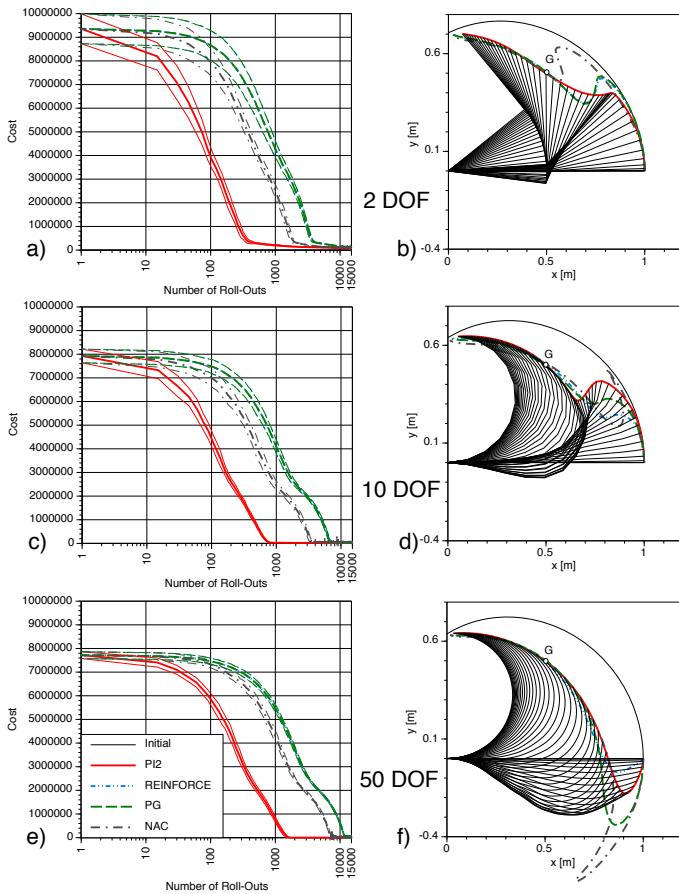


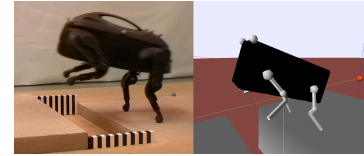
Fig. 1. Comparison of learning multi-DOF movements (2,10, and 50 DOFs) with planar robot arms passing through a via-point  $G$ . a,c,e) illustrate the learning curves for different RL algorithms, while b,d,f) illustrate the end-effector movement after learning for all algorithms.  $PI^2$  performs an order of magnitude better than other methods.

$G$  for a planar robot manipulator of 2,10 and 50 DOF. In the second evaluation in Figure 2, the Little Dog robot learns the task of how to jump over a large gap. In the third evaluation we use a Phantom robot (Figure 3) and we are applying  $PI^2$  not only for learning optimal state trajectories but also for learning the optimal gain schedule. The task for the phantom robot is to reach a goal state by accurately passing via an intermediate target, while maximizing compliance during control. The resulting behavior is illustrated in Figure 4.

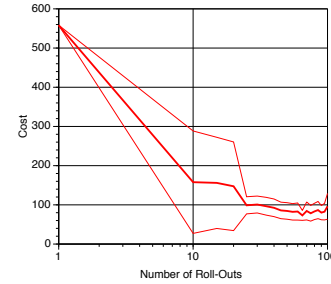
#### REFERENCES

- [1] E.A. Theodorou, J. Buchli, and S. Schaal. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *International Conference Of Robotics and Automation (ICRA2010)*, 2010. To Appear.
- [2] Jongmin Yong. Relations among ODEs, PDEs, FSDEs, BSDEs, and FBSDEs. In *Proceedings of the 36th IEEE Conference on Decision and Control, 1997*, volume 3, pages 2779–2784, Dec 1997.
- [3] P. Dayan and G. Hinton. Using EM for reinforcement learning. *Neural Computation*, 9, 1997.
- [4] J. Peters and S. Schaal. Learning to control in operational space. *International Journal of Robotics Research*, 27:197–212, 2008.
- [5] 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. MIT Press, Cambridge, MA.

**Topics: Learning Algorithms; Preference:Oral**



(a) Real & Simulated Robot Dog



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

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 a 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.

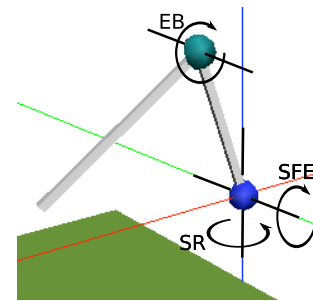


Fig. 3. Illustration of the three DOF kinematics of the Phantom robot.

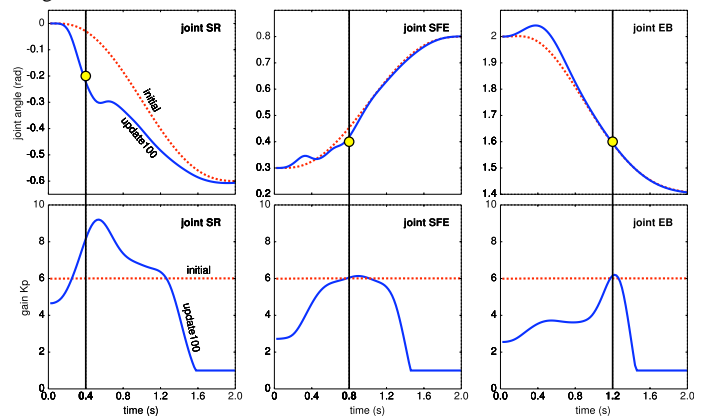


Fig. 4. The plots in the first row illustrate the joint movements of the robot. In the second row the time varying gains schedule is illustrated. The dashed line is the initial trajectory and the blue line corresponds to the trajectory after learning. At the intermediate target states the gains increase so that the task of passing through the intermediate goal can be accomplished, while the gains are kept low otherwise.