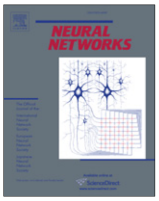
Neural Networks 21 (2008) 682–697

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



2008 Special Issue

Reinforcement learning of motor skills with policy gradients

Jan Peters[a,b,*], Stefan Schaal[b,c]

[a] *Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany*
[b] *University of Southern California, 3710 S. McClintoch Ave – RTH401, Los Angeles, CA 90089-2905, USA*
[c] *ATR Computational Neuroscience Laboratory, 2-2-2 Hikaridai, Seika-cho, Soraku-gun Kyoto 619-0288, Japan*

ARTICLE INFO

Article history:
Received 11 September 2007
Received in revised form
24 February 2008
Accepted 24 February 2008

Keywords:
Reinforcement learning
Policy gradient methods
Natural gradients
Natural Actor-Critic
Motor skills
Motor primitives

ABSTRACT

Autonomous learning is one of the hallmarks of human and animal behavior, and understanding the principles of learning will be crucial in order to achieve true autonomy in advanced machines like humanoid robots. In this paper, we examine learning of complex motor skills with human-like limbs. While supervised learning can offer useful tools for bootstrapping behavior, e.g., by learning from demonstration, it is only reinforcement learning that offers a general approach to the final trial-and-error improvement that is needed by each individual acquiring a skill. Neither neurobiological nor machine learning studies have, so far, offered compelling results on how reinforcement learning can be scaled to the high-dimensional continuous state and action spaces of humans or humanoids. Here, we combine two recent research developments on learning motor control in order to achieve this scaling. First, we interpret the idea of modular motor control by means of motor primitives as a suitable way to generate parameterized control policies for reinforcement learning. Second, we combine motor primitives with the theory of stochastic policy gradient learning, which currently seems to be the only feasible framework for reinforcement learning for humanoids. We evaluate different policy gradient methods with a focus on their applicability to parameterized motor primitives. We compare these algorithms in the context of motor primitive learning, and show that our most modern algorithm, the Episodic Natural Actor-Critic outperforms previous algorithms by at least an order of magnitude. We demonstrate the efficiency of this reinforcement learning method in the application of learning to hit a baseball with an anthropomorphic robot arm.

1. Introduction

In order to ever leave the well-structured environments of factory floors and research labs, future robots will require the ability to acquire novel behaviors and motor skills as well as to improve existing ones based on rewards and costs. Similarly, the understanding of human motor control would benefit significantly if we can synthesize simulated human behavior and its underlying cost functions based on insight from machine learning and biological inspirations. Reinforcement learning is probably the most general framework in which such learning problems of computational motor control can be phrased. However, in order to bring reinforcement learning into the domain of human movement learning, two deciding components need to be added to the standard framework of reinforcement learning: first, we need a domain-specific policy representation for motor skills, and, second, we need reinforcement learning algorithms which work efficiently with this representation while scaling into the domain of high-dimensional mechanical systems such as humanoid robots.

Traditional representations of motor behaviors in robotics are mostly based on desired trajectories generated from spline interpolations between points, i.e., spline nodes, which are part of a longer sequence of intermediate target points on the way to a final movement goal. While such a representation is easy to understand, the resulting control policies, generated from a tracking controller of the spline trajectories, have a variety of significant disadvantages, including that they are time indexed and thus not robust towards unforeseen disturbances, that they do not easily generalize to new behavioral situations without complete recomputation of the spline, and that they cannot easily be coordinated with other events in the environment, e.g., synchronized with other sensory variables like visual perception during catching a ball. In the literature, a variety of other approaches for parameterizing movement have been suggested to overcome these problems, see Ijspeert, Nakanishi, and Schaal (2002, 2003) for more information. One of these approaches proposed using parameterized nonlinear dynamical systems as motor primitives, where the attractor properties of these dynamical systems defined the

* Corresponding author at: Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany.
E-mail address: jan.peters@tuebingen.mpg.de (J. Peters).

0893-6080/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2008.02.003

desired behavior (Ijspeert et al., 2002, 2003). The resulting framework was particularly well suited for supervised imitation learning in robotics, exemplified by examples from humanoid robotics where a full-body humanoid learned tennis swings or complex polyrhythmic drumming patterns. One goal of this paper is the application of reinforcement learning to both traditional spline-based representations as well as the more novel dynamic system based approach.

However, despite the fact that reinforcement learning is the most general framework for discussing the learning of movement in general, and motor primitives for robotics in particular, most of the methods proposed in the reinforcement learning community are not applicable to high-dimensional systems such as humanoid robots. Among the main problems are that these methods do not scale beyond systems with more than three or four degrees of freedom and/or cannot deal with parameterized policies. Policy gradient methods are a notable exception to this statement. Starting with the pioneering work[1] of Gullapali and colleagues (Benbrahim & Franklin, 1997; Gullapalli, Franklin, & Benbrahim, 1994) in the early 1990s, these methods have been applied to a variety of robot learning problems ranging from simple control tasks (e.g., balancing a ball on a beam (Benbrahim, Doleac, Franklin, & Selfridge, 1992), and pole balancing (Kimura & Kobayashi, 1998)) to complex learning tasks involving many degrees of freedom such as learning of complex motor skills (Gullapalli et al., 1994; Mitsunaga, Smith, Kanda, Ishiguro, & Hagita, 2005; Miyamoto et al., 1995, 1996; Peters & Schaal, 2006; Peters, Vijayakumar, & Schaal, 2005a) and locomotion (Endo, Morimoto, Matsubara, Nakanishi, & Cheng, 2005; Kimura & Kobayashi, 1997; Kohl & Stone, 2004; Mori, Nakamura, aki Sato, & Ishii, 2004; Nakamura, Mori, & Ishii, 2004; Sato, Nakamura, & Ishii, 2002; Tedrake, Zhang, & Seung, 2005).

The advantages of policy gradient methods for parameterized motor primitives are numerous. Among the most important ones are that the policy representation can be chosen such that it is meaningful for the task, i.e., we can use a suitable motor primitive representation, and that domain knowledge can be incorporated, which often leads to fewer parameters in the learning process in comparison to traditional value function based approaches. Moreover, there exist a variety of different algorithms for policy gradient estimation in the literature, most with rather strong theoretical foundations. Additionally, policy gradient methods can be used model-free and therefore also be applied to problems without analytically known task and reward models.

Nevertheless, many recent publications on applications of policy gradient methods in robotics overlooked the newest developments in policy gradient theory and their original roots in the literature. Thus, a large number of heuristic applications of policy gradients can be found, where the success of the projects mainly relied on ingenious initializations and manual parameter tuning of algorithms. A closer inspection often reveals that the chosen methods might be statistically biased, or even generate infeasible policies under less fortunate parameter settings, which could lead to unsafe operation of a robot. The main goal of this paper is to discuss which policy gradient methods are applicable to robotics and which issues matter, while also introducing some new policy gradient learning algorithms that seem to have superior performance over previously suggested methods. The remainder of this paper will proceed as follows: firstly, we will introduce the general assumptions of reinforcement learning, discuss motor primitives in this framework and pose the problem statement of this paper. Secondly, we will analyze the different approaches to policy gradient estimation and discuss their applicability to reinforcement learning of motor primitives. We focus on the most useful methods and examine several algorithms in depth. The presented algorithms in this paper are highly optimized versions of both novel and previously published policy gradient algorithms. Thirdly, we show how these methods can be applied to motor skill learning in humanoid robotics and show learning results with a seven degree of freedom, anthropomorphic SARCOS Master Arm.

1.1. *General assumptions and problem statement*

Most robotics domains require the state-space and the action spaces to be continuous and high dimensional such that learning methods based on discretizations are not applicable for higher-dimensional systems. However, as the policy is usually implemented on a digital computer, we assume that we can model the control system in a discrete-time manner and we will denote the current time step[2] by k. In order to take possible stochasticity of the plant into account, we denote it using a probability distribution

$$\boldsymbol{x}_{k+1} \sim p\left(\boldsymbol{x}_{k+1} \,|\boldsymbol{x}_k, \boldsymbol{u}_k\right) \tag{1}$$

where $\boldsymbol{u}_k \in \mathbb{R}^M$ denotes the current action, and $\boldsymbol{x}_k$, $\boldsymbol{x}_{k+1} \in \mathbb{R}^N$ denote the current and the next state respectively. We furthermore assume that actions are generated by a policy

$$\boldsymbol{u}_k \sim \pi_{\boldsymbol{\theta}}\left(\boldsymbol{u}_k \,|\boldsymbol{x}_k\right) \tag{2}$$

which is modeled as a probability distribution in order to incorporate exploratory actions; for some special problems, the optimal solution to a control problem is actually a stochastic controller, see e.g., Sutton, McAllester, Singh, and Mansour (2000). The policy is parameterized by some policy parameters $\boldsymbol{\theta} \in \mathbb{R}^K$ and assumed to be continuously differentiable with respect to its parameters $\boldsymbol{\theta}$. The sequence of states and actions forms a trajectory (also called history or roll-out) denoted by $\boldsymbol{\tau} = [\boldsymbol{x}_{0:H}, \boldsymbol{u}_{0:H}]$ where H denotes the horizon, which can be infinite. At each instant of time, the learning system receives a reward denoted by $r(\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathbb{R}$.

The general goal of policy gradient reinforcement learning is to optimize the policy parameters $\boldsymbol{\theta} \in \mathbb{R}^K$ so that the expected return

$$J(\boldsymbol{\theta}) = \frac{1}{a_\Sigma} E\left\{\sum_{k=0}^{H} a_k r_k\right\} \tag{3}$$

is optimized where a_k denote time-step-dependent weighting factors and a_Σ is a normalization factor in order to ensure that the normalized weights a_k/a_Σ sum up to one. We require that the weighting factors fulfill $a_{l+k} = a_l a_k$ in order to be able to connect to the previous policy gradient literature; examples are the weights $a_k = \gamma^k$ for discounted reinforcement learning (where γ is in $[0, 1]$) where $a_\Sigma = 1/(1-\gamma)$; alternatively, they are set to $a_k = 1$ for the average reward case where $a_\Sigma = H$. In these cases, we can rewrite *a normalized expected return* in the form

$$J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^\pi(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) r(\boldsymbol{x}, \boldsymbol{u}) \mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{u} \tag{4}$$

[1] Note that there has been earlier work by the control community, see e.g., Dyer and McReynolds (1970), Hasdorff (1976) and Jacobson and Mayne (1970), which is based on exact analytical models. Extensions based on learned, approximate models originated in the literature on optimizing government decision policies, see Werbos (1979), and have also been applied in control (Atkeson, 1994; Morimoto & Atkeson, 2003). In this paper, we limit ourselves to model-free approaches as the most general framework, while future work will address specialized extensions to model-based learning.

[2] Note, that throughout this paper, we will use k and l for denoting discrete steps, m for update steps and h for the current vector element, e.g., $\boldsymbol{\theta}_h$ denotes the hth element of $\boldsymbol{\theta}$.

as used in Sutton et al. (2000), where $d^{\pi}(\boldsymbol{x}) = a_{\Sigma}^{-1}\sum_{k=0}^{\infty} a_k p(\boldsymbol{x}_k = \boldsymbol{x})$ is the weighted state distribution.[3]

In general, we assume that for each considered policy $\pi_{\boldsymbol{\theta}}$, a state-value function $V^{\pi}(\boldsymbol{x}, k)$, and the state-action value function $Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}, k)$ exist and are given by

$$V^{\pi}(\boldsymbol{x}, k) = E\left\{ \sum_{l=k}^{H} a_l r_l \middle| \boldsymbol{x}_k = \boldsymbol{x} \right\}, \tag{5}$$

$$Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}, k) = E\left\{ \sum_{l=k}^{H} a_l r_l \middle| \boldsymbol{x}_k = \boldsymbol{x}, \boldsymbol{u}_k = \boldsymbol{u} \right\}. \tag{6}$$

In the infinite horizon case, i.e., for $H \to \infty$, we write $V^{\pi}(\boldsymbol{x})$ and $Q^{\pi}(\boldsymbol{x}, \boldsymbol{u})$ as these functions have become time-invariant. Note, that we can define the expected return in terms of the state-value function by

$$J(\boldsymbol{\theta}) = \int_{\mathbb{X}} p(\boldsymbol{x}_0) V^{\pi}(\boldsymbol{x}_0, 0) d\boldsymbol{x}_0, \tag{7}$$

where $p(\boldsymbol{x}_0)$ is the probability of $\boldsymbol{x}_0$ being the start-state. Whenever we make practical use of the value function, we assume that we are given "good" basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ so that the state-value function can be approximated with linear function approximation $V^{\pi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{v}$ with parameters $\boldsymbol{v} \in \mathbb{R}^c$ in an approximately unbiased way.

1.2. Motor primitive policies

In this section, we first discuss how motor plans can be represented and then how we can bring these into the standard reinforcement learning framework. For this purpose, we consider two forms of motor plans, i.e., (1) *spline-based trajectory plans* and (2) *nonlinear dynamic motor primitives* introduced in Ijspeert et al. (2002). Spline-based trajectory planning is well known in the robotics literature, see e.g., Miyamoto et al. (1996) and Sciavicco and Siciliano (2007). A desired trajectory is represented as connected pieces of simple polynomials, e.g., for third-order splines, we have

$$q_{d,n}(t) = \theta_{0n} + \theta_{1n}t + \theta_{2n}t^2 + \theta_{3n}t^3 \tag{8}$$

in $t \in [t_{n-1}, t_n]$ under the boundary conditions of

$$q_{d,n-1}(t_n) = q_{d,n}(t_n) \quad \text{and} \quad \dot{q}_{d,n-1}(t_n) = \dot{q}_{d,n}(t_n).$$

A given tracking controller, e.g., a PD control law or an inverse dynamics controller, ensures that the trajectory is realized accurately. Thus, a desired movement is parameterized by its spline nodes and the duration of each spline node. These parameters can be learned from fitting a given trajectory with a spline approximation algorithm (Wada & Kawato, 1994), or by means of optimization or reinforcement learning (Miyamoto et al., 1996). We call such parameterized movement plans motor primitives.

For nonlinear dynamic motor primitives, we use the approach developed in Ijspeert et al. (2002). These dynamic motor primitives can be seen as a type of central pattern generator which is particularly well suited for learning as it is linear in the parameters and are invariant under rescaling. In this approach, movement plans $(\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d)$ for each degree of freedom (DOF) of the robot are represented in terms of the time evolution of the nonlinear dynamical systems

$$\ddot{q}_d = f(q_d, \boldsymbol{z}, g, \tau, \theta) \tag{9}$$

where $(q_d, \dot{q}_d)$ denote the desired position and velocity of a joint, z the internal state of the dynamic system which evolves in accordance to a canonical system $\ddot{z} = f_c(z, \tau)$, g the goal (or point attractor) state of each DOF, τ the movement duration shared by all DOFs, and θ the open parameters of the function f. In contrast to splines, formulating movement plans as dynamic systems offers useful invariance properties of a movement plan under temporal and spatial scaling, as well as natural stability properties — see Ijspeert et al. (2002) for a discussion. Adjustment of the primitives using sensory input can be incorporated by modifying the internal state z of the system as shown in the context of drumming (Pongas, Billard, & Schaal, 2005) and biped locomotion (Nakanishi et al., 2004; Schaal, Peters, Nakanishi, & Ijspeert, 2004). The equations used in order to create Eq. (9) are given in the Appendix. The original work in Ijspeert et al. (2002) demonstrated how the parameters θ_h can be learned to match a template trajectory by means of supervised learning — this scenario is, for instance, useful as the first step of an imitation learning system. Here, we will add the ability of self-improvement of the movement primitives in Eq. (9) by means of reinforcement learning, which is the crucial second step in imitation learning.

The systems in Eqs. (8) and (9) are point-to-point movements, i.e., such tasks are rather well suited for the introduced episodic reinforcement learning methods. In both systems, we have access to at least 2nd derivatives in time, i.e., desired accelerations, which are needed for model-based feedforward controllers. In order to make the reinforcement framework feasible for learning with motor primitives, we need to add exploration to the respective motor primitive framework, i.e., we need to add a small perturbation $\epsilon \sim \mathcal{N}(0, \sigma^2)$ to the desired accelerations, such that the nominal target output $\ddot{q}_d$ becomes the perturbed target output $\ddot{\hat{q}}_d = \ddot{q}_d + \epsilon$. By doing so, we obtain a stochastic policy

$$\pi(\ddot{\hat{q}}_d|\ddot{q}_d) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\ddot{\hat{q}}_d - \ddot{q}_d)^2}{2\sigma^2}\right). \tag{10}$$

This policy will be used throughout the paper. It is particularly practical as the exploration can be easily controlled through only one variable σ.

2. Policy gradient approaches for parameterized motor primitives

The general goal of policy optimization in reinforcement learning is to optimize the policy parameters $\boldsymbol{\theta} \in R^K$ so that the expected return $J(\boldsymbol{\theta})$ is maximal. For motor primitive learning in robotics, we require that any change to the policy parameterization has to be smooth as drastic changes can be hazardous for the robot and its environment. Also, it would render initializations of the policy based on domain knowledge or imitation learning useless, as these would otherwise vanish after a single update step (Schaal, 1997). Additionally, we need to guarantee that the policy is improved in the update steps at least on average which rules out greedy value function based methods with approximated value functions, as these methods are frequently problematic in regard of this property (Kakade, 2003). For these reasons, policy gradient methods, which follow the steepest descent on the expected return, are currently the most suitable method for motor learning. These methods update the policy parameterization at time step m according to the gradient update rule

$$\boldsymbol{\theta}_{m+1} = \boldsymbol{\theta}_m + \alpha_m \nabla_{\boldsymbol{\theta}} J|_{\boldsymbol{\theta}=\boldsymbol{\theta}_m}, \tag{11}$$

where $\alpha_m \in \mathbb{R}^+$ denotes a learning rate. If the gradient estimate is unbiased and learning rates fulfill

$$\sum_{m=0}^{\infty} \alpha_m > 0 \quad \text{and} \quad \sum_{m=0}^{\infty} \alpha_m^2 = 0, \tag{12}$$

[3] In most cases, e.g., for $a_k = \gamma^k$, this distribution is a multi-modal mixture distribution even if the distribution $p(x_k = x)$ is unimodal. Only for $a_k = 1$, the state weighted distribution $d^{\pi}(x)$ will converge to the stationary distribution.

Table 1
General setup for policy gradient reinforcement learning

```
input: initial policy parameterization $\theta_0$.
1    repeat
2        obtain policy gradient $g$ from estimator (see Tables 2–6)
3        update policy $\theta_{m+1} = \theta_m + \alpha_m g$.
4    until policy parameterization $\theta_m \approx \theta_{m+1}$ converges
return: optimal policy parameters $\theta^* = \theta_{m+1}$.
```

the learning process is guaranteed to converge to at least a local minimum. The general setup is shown in Table 1.

The main problem in policy gradient methods is obtaining a good estimator of the policy gradient $\nabla_\theta J|_{\theta=\theta_m}$. Traditionally, people have used deterministic model-based methods for obtaining the gradient (Dyer & McReynolds, 1970; Hasdorff, 1976; Jacobson & Mayne, 1970). However, in order to become autonomous we cannot expect to be able to model every detail of the robot and environment appropriately. Therefore, we need to estimate the policy gradient only from data generated during the execution of a task, i.e., without the need for a model. In this section, we will study different approaches and discuss which of these are useful in robotics. The literature on policy gradient methods has yielded a variety of estimation methods over the last years. The most prominent approaches, which have been applied to robotics are finite-difference and likelihood ratio methods, more well known as REINFORCE methods in reinforcement learning.

2.1. *Finite-difference methods*

Finite-difference methods are among the oldest policy gradient approaches dating back to the 1950s; they originated from the stochastic simulation community and are quite straightforward to understand. The policy parameterization is varied by small increments $\Delta\theta_i$ and for each policy parameter variation $\theta_m + \Delta\theta_i$ roll-outs are performed which generate estimates $\hat{J}_i = J(\theta_m + \Delta\theta_i)$ and $\Delta\hat{J}_i \approx J(\theta_m + \Delta\theta_i) - J_{\text{ref}}$ of the expected return. There are different ways of choosing the reference value J_{ref}, e.g. forward-difference estimators with $J_{\text{ref}} = J(\theta_m)$ and central-difference estimators with $J_{\text{ref}} = J(\theta_m - \Delta\theta_i)$. The most general way is to formulate the determination of the reference value J_{ref} and the policy gradient estimate $\mathbf{g}_{\text{FD}} \approx \nabla_\theta J|_{\theta=\theta_m}$ as a regression problem which can be solved by

$$\left[\mathbf{g}_{\text{FD}}^{\mathrm{T}}, J_{\text{ref}}\right]^{\mathrm{T}} = \left(\Delta\Theta^{\mathrm{T}}\Delta\Theta\right)^{-1}\Delta\Theta^{\mathrm{T}}\hat{\mathbf{J}}, \tag{13}$$

where

$$\Delta\Theta = \begin{bmatrix}\Delta\theta_1, & \dots, & \Delta\theta_I\\ 1, & \dots, & 1\end{bmatrix}^{\mathrm{T}}, \quad \text{and} \tag{14}$$

$$\hat{\mathbf{J}} = [\hat{J}_1, \dots, \hat{J}_I]^{\mathrm{T}}, \tag{15}$$

denote the I samples. If single parameters are perturbed, this method is known as the Kiefer–Wolfowitz procedure and if multiple parameters are perturbed simultaneously, it is known as Simultaneous Perturbation Stochastic gradient Approximation (SPSA), see Sadegh and Spall (1997) and Spall (2003) for in-depth treatment. This approach can be highly efficient in simulation optimization of deterministic systems (Spall, 2003) or when a common history of random numbers (Glynn, 1987; Kleinman, Spall, & Naiman, 1999) is being used (the later trick is known as the PEGASUS method in reinforcement learning, see Ng and Jordan (2000)), and can get close to a convergence rate of $O(I^{-1/2})$ (Glynn, 1987). However, when used on a real system, the uncertainties degrade the performance resulting in convergence rates ranging between $O(I^{-1/4})$ and $O(I^{-2/5})$ depending on the chosen reference value (Glynn, 1987). An implementation of this algorithm is shown in Table 2.

Table 2
Finite-difference gradient estimator

```
input: policy parameterization $\theta$.
1    repeat
2        generate policy variation $\Delta\theta_i$.
3        estimate $J(\theta + \Delta\theta_i) \approx \hat{J}_i = \sum_{k=0}^{H} a_k r_k$ from roll-outs.
4        compute gradient $\left[\mathbf{g}_{\text{FD}}^{\mathrm{T}}, J_{\text{ref}}\right]^{\mathrm{T}} = \left(\Delta\Theta^{\mathrm{T}}\Delta\Theta\right)^{-1}\Delta\Theta^{\mathrm{T}}\hat{\mathbf{J}}$.
             with $\Delta\Theta^{\mathrm{T}} = \begin{bmatrix}\Delta\theta_1, \dots, \Delta\theta_I\\ 1, \dots, 1\end{bmatrix}$,
             and $\hat{\mathbf{J}}^{\mathrm{T}} = [\hat{J}_1, \dots, \hat{J}_I]$.
5    until gradient estimate $\mathbf{g}_{\text{FD}}$ converged.
return: gradient estimate $\mathbf{g}_{\text{FD}}$.
```

Finite-difference methods are widely applicable as they do not require knowledge of the policy and they do not depend on the differentiability of the policy with respect to the policy parameters. While these facts do not matter for the case of motor primitive learning, both can be important in other setups, e.g., a setup where we only have access to a few selected parameters of the policy in a complex, unknown system or when optimizing policy parameters which can only take discrete values.

Due to the simplicity of this approach, finite-difference methods have been successfully applied to robot motor skill learning in numerous applications (Kohl & Stone, 2004; Mitsunaga et al., 2005; Miyamoto et al., 1995, 1996; Tedrake et al., 2005). However, the straightforward application is not without peril as the generation of $\Delta\theta_i$ requires proper knowledge of the system, as badly chosen $\Delta\theta_i$ can destabilize the policy so that the system becomes instable and the gradient estimation process is prone to fail. Even in the field of simulation optimization where the destabilization of the system is not such a dangerous issue, the careful generation of the parameter perturbation is a topic of debate with strong requirements on the generating process (Sadegh & Spall, 1997). Practical problems often require that each element of the vector $\Delta\theta_i$ has a different order of magnitude, making the generation particularly difficult. Therefore, this approach can be applied only under strict human supervision.

2.2. *Likelihood ratio methods and REINFORCE*

Likelihood ratio methods are driven by an different important insight. Assume that trajectories τ are generated from a system by roll-outs, i.e., $\tau \sim p_\theta(\tau) = p(\tau|\theta)$ with rewards $r(\tau) = \sum_{k=0}^{H} a_k r_k$. In this case, the policy gradient can be estimated using the likelihood ratio trick, see e.g. Aleksandrov, Sysoyev, and Shemeneva (1968) and Glynn (1987), or REINFORCE trick (Williams, 1992). First, from the view of trajectories, the expected return of a policy can be written as an expectation over all possible trajectories $\mathbb{T}$:

$$J(\theta) = \int_{\mathbb{T}} p_\theta(\tau)\, r(\tau)\mathrm{d}\tau.$$

Subsequently, we can rewrite the gradient by

$$\begin{aligned}\nabla_\theta J(\theta) &= \int_{\mathbb{T}} \nabla_\theta p_\theta(\tau)\, r(\tau)\mathrm{d}\tau\\ &= \int_{\mathbb{T}} p_\theta(\tau)\, \nabla_\theta \log p_\theta(\tau)\, r(\tau)\mathrm{d}\tau,\\ &= E\left\{\nabla_\theta \log p_\theta(\tau)\, r(\tau)\right\}.\end{aligned} \tag{16}$$

Importantly, the derivative $\nabla_\theta \log p_\theta(\tau)$ can be computed without knowledge of the generating distribution $p_\theta(\tau)$ as

$$p_\theta(\tau) = p(\mathbf{x}_0)\prod_{k=0}^{H} p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k)\, \pi_\theta(\mathbf{u}_k|\mathbf{x}_k) \tag{17}$$

Table 3
General likelihood ratio policy gradient estimator "Episodic REINFORCE" with an optimal baseline

```
input: policy parameterization $\boldsymbol{\theta}$.
1  repeat
2      perform a trial and obtain $\boldsymbol{x}_{0:H}, \boldsymbol{u}_{0:H}, r_{0:H}$
3      for each gradient element $g_h$
4          estimate optimal baseline
           $b^h = \frac{\left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k)\right)^2 \sum_{l=0}^{H} a_l r_l \right\rangle}{\left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k)\right)^2 \right\rangle}$
5          estimate the gradient element
           $g_h = \left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k)\right) \left(\sum_{l=0}^{H} a_l r_l - b^h\right) \right\rangle.$
6      end for.
7  until gradient estimate $\mathbf{g}_{\mathrm{RF}}$ converged.
return: gradient estimate $\mathbf{g}_{\mathrm{RF}}$.
```

implies that

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = \sum_{k=0}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k), \tag{18}$$

i.e., the derivatives with respect to the control system do not have to be computed.[4] As, in general, the following kind of integral always yields zero:

$$\int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \mathrm{d}\boldsymbol{\tau} = \int_{\mathbb{T}} \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \mathrm{d}\boldsymbol{\tau} = \nabla_{\boldsymbol{\theta}} 1 = 0, \tag{19}$$

a constant baseline can be inserted into the policy gradient estimate, resulting in the final gradient estimator

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = E\left\{\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})(r(\boldsymbol{\tau}) - b)\right\}, \tag{20}$$

where $b \in \mathbb{R}$ can be chosen arbitrarily (Williams, 1992) but usually with the goal to minimize the variance of the gradient estimator. Note that the baseline was most likely first suggested in Williams (1992) and is unique to reinforcement learning as it requires a separation of the policy from the state-transition probability densities. Therefore, the general path likelihood ratio estimator or episodic REINFORCE gradient estimator (Williams, 1992) is given by

$$\mathbf{g}_{\mathrm{RF}} = \left\langle \left(\sum_{k=0}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k)\right) \left(\sum_{l=0}^{H} a_l r_l - b\right) \right\rangle, \tag{21}$$

where $\langle f(\boldsymbol{\tau}) \rangle = \int_{\mathbb{T}} f(\boldsymbol{\tau}) \mathrm{d}\boldsymbol{\tau}$ denotes the average over trajectories. This type of method is guaranteed to converge to the true gradient at the fastest theoretically possible pace of $O(I^{-1/2})$ where I denotes the number of roll-outs (Glynn, 1987) even if the data are generated from a highly stochastic system. An implementation of this algorithm is shown in Table 3 together with an optimal estimator for the baseline.

Besides the theoretically faster convergence rate, likelihood ratio gradient methods have a variety of advantages in comparison to finite-difference methods. As the generation of policy parameter variations is no longer needed, the complicated control of these variables can no longer endanger the gradient estimation process. Furthermore, in practice, already a single roll-out can suffice for an unbiased gradient estimate (Baxter & Bartlett, 2001; Spall, 2003) viable for a good policy update step, thus reducing the number of roll-outs needed. Finally, this approach has yielded the most real-world robot motor learning results (Benbrahim & Franklin, 1997; Endo et al., 2005; Gullapalli et al., 1994; Kimura & Kobayashi, 1997; Mori et al., 2004; Nakamura et al., 2004; Peters et al., 2005a). In the subsequent two sections, we will strive to explain and improve this type of gradient estimator.

In the following two sections, we will strive to give an extensive overview on the two classes of likelihood ratio policy gradient methods, 'vanilla' policy gradient methods in Section 3, and natural policy gradient methods in Section 4. The most important part here is to present the best methods of both classes and, subsequently, compare them in Section 5.1. The succeeding method will be used in Section 5.2 for a real robot evaluation.

3. 'Vanilla' policy gradient approaches

Despite the fast asymptotic convergence speed of the gradient estimate, the variance of the likelihood-ratio gradient estimator can be problematic in theory as well as in practice. This can be illustrated straightforwardly with an example.

Example 1. When using a REINFORCE estimator with a baseline $b = 0$ in a scenario where there is only a single reward of always the same magnitude, e.g., $r(\boldsymbol{x}, \boldsymbol{u}) = c \in R$ for all $\boldsymbol{x}, \boldsymbol{u}$, then the variance of the gradient estimate will grow at least cubically with the length of the planning horizon H as

$$\mathrm{Var}\{g_{\mathrm{RF}}\} = H^2 c^2 \sum_{k=0}^{H} \mathrm{Var}\{\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k)\}, \tag{22}$$

if $\mathrm{Var}\{\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k)\} > 0$ for all k. Furthermore, it will also grow quadratically with the magnitude of the reward c. The mean gradient remains unaltered by the reward magnitude c in this example as the reward is constant.[5]

For this reason, we need to address this issue and we will discuss several advances in likelihood ratio policy gradient optimization, i.e., the policy gradient theorem/GPOMDP, optimal baselines and the compatible function approximation.

3.1. Policy gradient theorem and G(PO)MDP

The intuitive observation that future actions do not depend on past rewards (unless policy changes take place continuously during the trajectory) can result in a significant reduction of the variance of the policy gradient estimate. This insight can be formalized as

$$E_{x_{k:l},u_{k:l}}\left\{\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_l|\boldsymbol{x}_l) r_k\right\} = E_{x_{k:l},u_{k:(l-1)}}\left\{\int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_l|\boldsymbol{x}_l) \mathrm{d}\boldsymbol{u}_l r_k\right\} = 0 \tag{23}$$

as $\int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_l|\boldsymbol{x}_l) \mathrm{d}\boldsymbol{u}_l = 0$ for $k < l$. This allows two variations of the previous algorithm which are known as the policy gradient theorem (Sutton et al., 2000)

$$\mathbf{g}_{\mathrm{PGT}} = \left\langle \sum_{k=0}^{H} a_k \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k) \left(\sum_{l=k}^{H} a_{l-k} r_l - b_k\right) \right\rangle,$$

or G(PO)MD (Baxter & Bartlett, 2001)

$$\mathbf{g}_{\mathrm{GMDP}} = \left\langle \sum_{l=0}^{H} \left(\sum_{k=0}^{l} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k)\right) (a_l r_l - b_l) \right\rangle.$$

[4] This result makes an important difference: in stochastic system optimization, finite difference estimators are often preferred as the derivative through system is required but not known. In policy search, we always know the derivative of the policy with respect to its parameters and therefore we can make use of the theoretical advantages of likelihood ratio gradient estimators.

[5] In the general case, the gradient length scales linearly in the reward magnitude and, thus, learning rates which include the inverse of the expected return can be very efficient (Peters, 2007).

While these algorithms *look* different, they are *exactly equivalent* in their gradient estimate,[6] i.e.,

$$\mathbf{g}_{\mathrm{PGT}} = \mathbf{g}_{\mathrm{GMPD}}, \tag{24}$$

which is a direct result from the summation theorem (Vachenauer, Rade, & Westergren, 2000) and from the fact that they both can be derived from REINFORCE. The G(PO)MDP formulation has previously been derived in the simulation optimization community (Glynn, 1990). An implementation of this algorithm is shown together with the optimal baseline in Table 4.

These two forms originally puzzled the community as they were derived from two separate points of view (Baxter & Bartlett, 2001; Sutton et al., 2000) and seemed to be different on first inspection. While their equality is natural when taking the path-based perspective, we will obtain the forms proposed in the original sources in a few steps. First, let us note that in $\mathbf{g}_{\mathrm{PGT}}$ the term $\sum_{l=k}^{\infty} a_{l-k} r_l$ in the policy gradient theorem is equivalent to a Monte Carlo estimate of the value function $Q^{\pi}(\boldsymbol{x}, \boldsymbol{u})$. Thus, we obtain

$$\mathbf{g}_{\mathrm{PGT}} = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\boldsymbol{u}|\boldsymbol{x}) \left(Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) - b(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{u} \mathrm{d}\boldsymbol{x},$$

for normalized weightings with infinite horizons (e.g., using the discounted or the average reward case). This form has a significant advantage over REINFORCE-like expressions, i.e., the variance does not grow with the planning horizon if a good estimate of $Q^{\pi}(\boldsymbol{x}, \boldsymbol{u})$ is given, e.g., by using traditional value function methods. Thus, the counterexample from Example 1 does no longer apply. Similarly, the term $\sum_{k=0}^{l} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k)$ becomes the log-derivative of the distribution of states $\mu_k^{\pi}(\boldsymbol{x}) = p(\boldsymbol{x} = x_k)$ at step k in expectation, i.e.,

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log d^{\pi}(\boldsymbol{x}) &= \sum_{l=0}^{H} a_l \nabla_{\boldsymbol{\theta}} \log \mu_k^{\pi}(\boldsymbol{x}) \\ &= \sum_{l=0}^{H} a_l \sum_{k=0}^{l} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k), \end{aligned} \tag{25}$$

which then can be used to rewrite the G(PO)MDP estimator into state-space form, i.e.,

$$\begin{aligned}\mathbf{g}_{\mathrm{GMDP}} = \int_{\mathbb{X}} \int_{\mathbb{U}} & \left(\pi_{\boldsymbol{\theta}}(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} d^{\pi}(\boldsymbol{x}) \right. \\ & \left. + d^{\pi}(\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\boldsymbol{u}|\boldsymbol{x})\right) \left(r(\boldsymbol{x}, \boldsymbol{u}) - b\right) \mathrm{d}\boldsymbol{u}\mathrm{d}\boldsymbol{x}. \end{aligned} \tag{26}$$

Note that this form only allows a baseline which is independent of the state unlike the policy gradient theorem. When either of the state-action value function or the state distribution derivative can be easily obtained by derivation or estimation, the variance of the gradient can be reduced significantly. Without a formal derivation of it, the policy gradient theorem has been applied in Gullapalli (1990, 1992) and Kimura and Kobayashi (1997) using estimated value functions $Q^{\pi}(\boldsymbol{x}, \boldsymbol{u})$ instead of the term $\sum_{l=k}^{H} a_l r_l$ and a baseline $b_k = V^{\pi}(\boldsymbol{x}_k, k)$. Note that the version introduced in Kimura and Kobayashi (1997) is biased[7] and does not correspond to the correct gradient unlike the one in Gullapalli (1990, 1992).

Note that the formulation over paths can be used in a more general fashion than the state-action form, e.g., it allows derivations for non-stationary policies, rewards and systems, than the state-action formulation in the paragraph above. However, for some results, it is more convenient to use the state-action based formulation and there we have made use of it.

Table 4
Specialized likelihood ratio policy gradient estimator "G(PO)MDP"/Policy Gradient with an optimal baseline

```
input: policy parameterization $\boldsymbol{\theta}$.
1 repeat
2     perform trials and obtain $\boldsymbol{x}_{0:H}, \boldsymbol{u}_{0:H}, r_{0:H}$
3     for each gradient element $g_h$
4         for each time step $k$
              estimate baseline for time step $k$ by
              $b_k^h = \frac{\left\langle \left(\sum_{\kappa=0}^{k} \nabla_{\theta_h} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_\kappa|\boldsymbol{x}_\kappa)\right)^2 a_k r_k \right\rangle}{\left\langle \left(\sum_{\kappa=0}^{k} \nabla_{\theta_h} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_\kappa|\boldsymbol{x}_\kappa)\right)^2 \right\rangle}$
5         end for.
6         estimate the gradient element
          $g_h = \left\langle \sum_{l=0}^{H} \left(\sum_{k=0}^{l} \nabla_{\theta_h} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k)\right) \left(a_l r_l - b_l^h\right) \right\rangle$.
7     end for.
8 until gradient estimate $\mathbf{g}_{\mathrm{GMDP}}$ converged.
return: gradient estimate $\mathbf{g}_{\mathrm{GMDP}}$.
```

3.2. *Optimal baselines*

Above, we have already introduced the concept of a baseline which can decrease the variance of a policy gradient estimate by orders of magnitude. Thus, an optimal selection of such a baseline is essential. An optimal baseline minimizes the variance $\sigma_h^2 = \mathrm{Var}\{g_h\}$ of each element g_h of the gradient $\mathbf{g}$ *without* biasing the gradient estimate, i.e., violating $E\{\mathbf{g}\} = \nabla_{\boldsymbol{\theta}} J$. This can be phrased as having a separate baseline b^h for every coefficient of the gradient,[8] i.e., we have

$$\min_{b_h} \sigma_h^2 = \mathrm{Var}\{g_h\}, \tag{27}$$

$$\text{s.t. } E\{g_h\} = \nabla_{\theta_h} J. \tag{28}$$

As the variance can generally be expressed as $\sigma_h^2 = E\{g_h^2\} - (\nabla_{\theta_h} J)^2$, and due to the application of Jensen's inequality

$$\min_{b_h} \sigma_h^2 \geq E\left\{\min_{b_h} g_h^2\right\} - \left(\nabla_{\theta_h} J\right)^2, \tag{29}$$

we know that we only need to determine $\min_{b_h} g_h^2$ on our samples. When differentiating

$$g_h^2 = \left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k) \left(\sum_{l=0}^{H} a_l r_l - b \right) \right)^2 \right\rangle$$

and solving for the minimum, we obtain the optimal baseline for each gradient element g_h can always be given by

$$b^h = \frac{\left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k) \right)^2 \sum_{l=0}^{H} a_l r_l \right\rangle}{\left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k|\boldsymbol{x}_k) \right)^2 \right\rangle}$$

for the general likelihood ratio gradient estimator, i.e., Episodic REINFORCE. The algorithmic form of the optimal baseline is shown in Table 3 in line 4. If the sums in the baselines are modified appropriately, we can obtain the optimal baseline for the policy gradient theorem or G(PO)MPD. We only show G(PO)MDP in this paper in Table 4 as the policy gradient theorem is numerically equivalent.

The optimal baseline which does not bias the gradient in Episodic REINFORCE can only be a single number for all trajectories

[6] Note that Baxter and Bartlett (2001) additionally add an eligibility trick for reweighting trajectory pieces. This trick can be highly dangerous in robotics; it can be demonstrated that even in linear-quadratic regulation, this trick can result in convergence to the worst possible policies for small planning horizons (i.e., small eligibility rates).

[7] See Peters (2007) for more information.

[8] A single baseline for all parameters can also be obtained and is more common in the reinforcement learning literature (Greensmith, Bartlett, & Baxter, 2001, 2004; Lawrence, Cowan, & Russell, 2003; Weaver & Tao, 2001a, 2001b; Williams, 1992). However, such a baseline is of course suboptimal.

and in G(PO)MPD it can also depend on the time step (Peters, 2007). However, in the policy gradient theorem it can depend on the current state and, therefore, if a good parameterization for the baseline is known, e.g., in a generalized linear form $b(\boldsymbol{x}_k) = \boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathrm{T}}\omega$, this can significantly improve the gradient estimation process. However, the selection of the basis functions $\boldsymbol{\phi}(\boldsymbol{x}_k)$ can be difficult and often impractical in practice. See Greensmith et al. (2001, 2004), Lawrence et al. (2003), Weaver and Tao (2001a, 2001b) and Williams (1992), for more information on this topic.

3.3. Compatible function approximation

As we previously discussed, the largest source of variance in the formulation of the policy gradient theorem is the state-action value function $Q^\pi(\boldsymbol{x},\boldsymbol{u})$, especially if the function $Q^\pi(\boldsymbol{x},\boldsymbol{u})$ is approximated by roll-outs as in this context. The natural alternative of using approximate value functions is problematic as these introduce bias in the presence of imperfect basis function in the function approximator. However, as demonstrated in Sutton et al. (2000) and Konda and Tsitsiklis (2000) the term $Q^\pi(\boldsymbol{x},\boldsymbol{u}) - b^\pi(\boldsymbol{x})$ can be replaced by a compatible function approximation

$$f_w^\pi(\boldsymbol{x},\boldsymbol{u}) = (\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}))^{\mathrm{T}}\boldsymbol{w} \equiv Q^\pi(\boldsymbol{x},\boldsymbol{u}) - b^\pi(\boldsymbol{x}), \quad (30)$$

parameterized by the vector $\boldsymbol{w}$, *without* affecting the unbiasedness of the gradient estimate and irrespective of the choice of the baseline $b^\pi(\boldsymbol{x})$. However, as mentioned in Sutton et al. (2000), the baseline may still be useful in order to reduce the variance of the gradient estimate when $\mathbf{g}_{\mathrm{PGT}}$ is approximated from samples. Thus, we derive an estimate of the policy gradient as

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \int_{\mathbb{X}} d^\pi(\boldsymbol{x}) \int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{u}|\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^{\mathrm{T}} \mathrm{d}\boldsymbol{u}\mathrm{d}\boldsymbol{x}\boldsymbol{w} \\ &= \int_{\mathbb{X}} d^\pi(\boldsymbol{x})\hat{\boldsymbol{G}}_{\boldsymbol{\theta}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\boldsymbol{w} = \boldsymbol{G}_{\boldsymbol{\theta}}\boldsymbol{w} \quad (31)\end{aligned}$$

where $\nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{u}|\boldsymbol{x}) = \pi(\boldsymbol{u}|\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})$. Since $\pi(\boldsymbol{u}|\boldsymbol{x})$ is chosen by the user, even in sampled data, the integral

$$\hat{\boldsymbol{G}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^{\mathrm{T}}\mathrm{d}\boldsymbol{u}$$

can be evaluated analytically or empirically without actually executing all actions. It is also noteworthy that the baseline does not appear in Eq. (31) as it integrates out, thus eliminating the need to find an optimal selection of this open parameter. Nevertheless, the estimation of $\boldsymbol{G}_{\boldsymbol{\theta}} = \int_{\mathbb{X}} d^\pi(\boldsymbol{x})\hat{\boldsymbol{G}}_{\boldsymbol{\theta}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ is still expensive since $d^\pi(\boldsymbol{x})$ is not known.

An important observation is that the compatible function approximation $f_{\boldsymbol{w}}^\pi(\boldsymbol{x},\boldsymbol{u})$ is mean-zero w.r.t. the action distribution, i.e.,

$$\int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x})f_{\boldsymbol{w}}^\pi(\boldsymbol{x},\boldsymbol{u})\mathrm{d}\boldsymbol{u} = \boldsymbol{w}^{\mathrm{T}}\int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{u}|\boldsymbol{x})\mathrm{d}\boldsymbol{u} = 0,$$

since from $\int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x})\mathrm{d}\boldsymbol{u} = 1$, differentiation w.r.t. to θ results in $\int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{u}|\boldsymbol{x})\mathrm{d}\boldsymbol{u} = \mathbf{0}$. Thus, $f_w^\pi(\boldsymbol{x},\boldsymbol{u})$ represents an *advantage function* $A^\pi(\boldsymbol{x},\boldsymbol{u}) = Q^\pi(\boldsymbol{x},\boldsymbol{u}) - V^\pi(\boldsymbol{x})$ in general. The advantage function *cannot* be learned with TD-like bootstrapping without knowledge of the value function as the essence of TD is to compare the value $V^\pi(\boldsymbol{x})$ of the two adjacent states — but this value has been subtracted out in $A^\pi(\boldsymbol{x},\boldsymbol{u})$. Hence, a TD-like bootstrapping using exclusively the compatible function approximator is impossible. As an alternative, Konda and Tsitsiklis (2000) and Sutton et al. (2000) suggested approximating $f_w^\pi(\boldsymbol{x},\boldsymbol{u})$ from unbiased estimates $\hat{Q}^\pi(\boldsymbol{x},\boldsymbol{u})$ of the action value function, e.g., obtained from roll-outs and using least-squares minimization between f_w and $\hat{Q}^\pi$. While possible in theory, one needs to realize that this approach implies a function approximation problem where the parameterization of the function approximator only spans a much smaller subspace than the training data — e.g., imagine approximating a quadratic function with a line. In practice, the results of such an approximation depends crucially on the training data distribution and has thus unacceptably high variance — e.g., fitting a line to only data from the right branch of a parabola, the left branch, or data from both branches. In the next section, we will see that there are more suitable ways to estimate the compatible function approximation (Section 4.1) and that this compatible function approximation has a special meaning (Section 4.2).

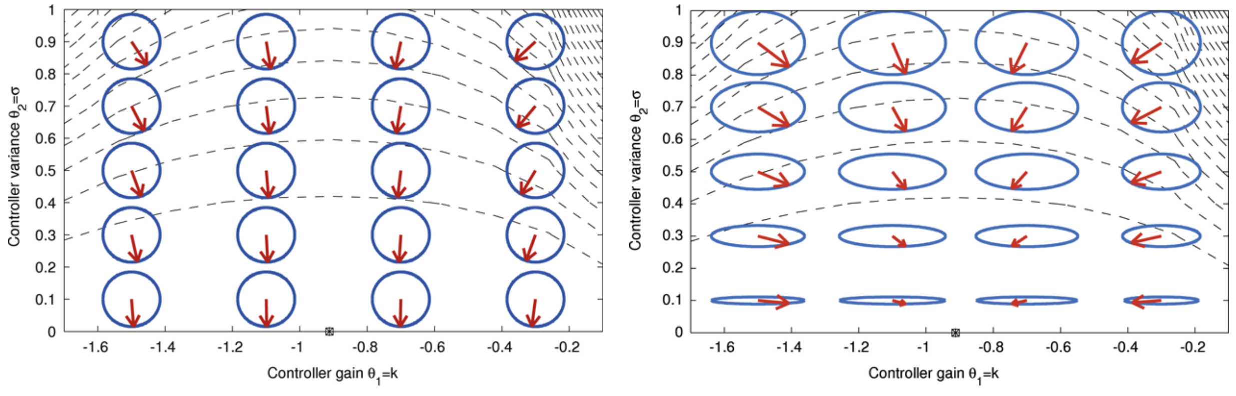
4. Natural Actor-Critic

Despite all the advances in the variance reduction of policy gradient methods, partially summarized above, these methods still tend to perform surprisingly poorly. Even when applied to simple examples with rather few states, where the gradient can be determined very accurately, they turn out to be quite inefficient — thus, the underlying reason cannot solely be the variance in the gradient estimate but rather must be caused by the large plateaus in the expected return landscape where the gradients are small and often do not point directly towards the optimal solution as demonstrated in Example 2. Thus, we are now turning to a second class of likelihood ratio gradient methods next to 'vanilla' policy gradient methods. Similarly as in supervised learning, the steepest ascent with respect to the Fisher information metric (Amari, 1998), called the 'natural' policy gradient, turns out to be significantly more efficient than normal gradients for such plateaus, as illustrated in Example 2.

Example 2. The classical example of linear-quadratic regulation is surprisingly hard with 'vanilla' policy gradient approaches. In this problem, we have a linear system with Gaussian noise $x_{t+1} \sim \mathcal{N}(Ax_t + Bu_t, \sigma_x^2)$, a linear policy with Gaussian exploration $u_t \sim \mathcal{N}(\theta_1 x_t, \theta_2)$ and a quadratic reward $r_t = -Qx_t^2 - Ru_t^2$. Traditionally 'vanilla' policy gradient approaches decrease the exploration rate θ_2 very fast and, thus, converge slowly towards the optimal solution as illustrated in Fig. 1(a). The reason is that a decrease of exploration has a stronger immediate effect on the expected return. Thus, if both parameters are seen as independent (as indicated by the circles), the plateau at zero exploration will result in the very slow convergence. If we can reweight the parameters, we could go along some directions faster than along others as indicated by the ellipses in Fig. 1(b). This reweighting can balance exploration and exploitation, resulting in faster convergence to the optimum. It is accomplished through the natural policy gradient.

This natural policy gradient approach was first suggested for reinforcement learning as the 'average natural policy gradient' in Kakade (2002), and subsequently shown to be the true natural policy gradient (Bagnell & Schneider, 2003; Peters, Vijayakumar, & Schaal, 2003). In this paper, we take this line of reasoning one step further by introducing the Natural Actor-Critic which inherits the convergence guarantees from gradient methods. Several properties of the natural policy gradient are worth highlighting before investigating some of the relevant derivations:

- Convergence to a local minimum is guaranteed, see Amari (1998).
- By choosing a more direct path to the optimal solution in parameter space, the natural gradient has, from empirical observations, faster convergence and avoids premature convergence of 'vanilla gradients' (see Fig. 1).
- The natural policy gradient can be shown to be **covariant**, i.e., independent of the coordinate frame chosen for expressing the policy parameters, see Peters, Vijayakumar, and Schaal (2005b).



(a) Vanilla policy gradient. (b) Natural policy gradient.

Fig. 1. The classical example of LQR can be used to illustrate why 'vanilla' policy gradients reduce the exploration to zero while natural policy gradients go for the optimal solution. The main difference is how the two approaches punish the change in parameters, i.e., the distance between current and next policy parameters. This distance is indicated by the blue ellipses in the contour plot while the dashed lines show the expected return. Obtaining a gradient then corresponds to finding a vector pointing from the center of the ellipses to the location with maximum expected return on the ellipse. A vanilla policy gradient (a) considers a change in all parameters as equally distant, thus, it is a search for a maximum on a circle while the natural gradient (b) uses scales determined by the Fisher information which results in a reduction in exploration. The slower reduction in exploration results into a faster convergence to the optimal policy. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

– As the natural gradient analytically averages out the influence of the stochastic policy (including the baseline of the function approximator), it requires fewer data points for a good gradient estimate than 'vanilla gradients'.

4.1. Motivation

One of the main reasons for using policy gradient methods is that we intend to make just a small change $\Delta\boldsymbol{\theta}$ to the policy $\pi_{\boldsymbol{\theta}}$ while improving the policy. However, the meaning of small is ambiguous. When using the Euclidian metric of $\sqrt{\Delta\boldsymbol{\theta}^{\mathrm{T}}\Delta\boldsymbol{\theta}}$, then the gradient is different for every parameterization $\boldsymbol{\theta}$ of the policy $\pi_{\boldsymbol{\theta}}$ even if these parameterizations are related to each other by a linear transformation (Kakade, 2002), often resulting in unnaturally slow learning even when higher-order gradient methods were employed (Baxter, Bartlett, & Weaver, 2001; Berny, 2000, chp 14; Kakade, 2001). This problem poses the question whether we can achieve a covariant gradient descent, i.e., gradient descent with respect to an invariant measure of the closeness between the current policy and the updated policy based upon the distribution of the paths generated by each of these. In statistics, a variety of distance measures for the closeness of two distributions (e.g., $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$ and $p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}(\boldsymbol{\tau})$) have been suggested, e.g., the Kullback–Leibler divergence[9] $d_{\mathrm{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \parallel p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}(\boldsymbol{\tau}))$, the Hellinger distance d_{HD} and others (Su & Gibbs, 2002). Many of these distances (e.g., the previously mentioned ones) can be approximated by the same second-order Taylor expansion, i.e., by

$$d_{\mathrm{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \parallel p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}(\boldsymbol{\tau})) \approx \frac{1}{2}\Delta\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{F}_{\boldsymbol{\theta}}\Delta\boldsymbol{\theta}, \tag{32}$$

where

$$\begin{aligned}\boldsymbol{F}_{\boldsymbol{\theta}} &= \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\nabla\log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\nabla\log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})^{\mathrm{T}}\mathrm{d}\boldsymbol{\tau}\\ &= \left\langle\nabla\log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\nabla\log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})^{\mathrm{T}}\right\rangle\end{aligned} \tag{33}$$

is known as the Fisher information matrix. Let us assume that we fix the amount of change in our policy using the step-size ε. We then have a restricted step-size gradient descent problem (Fletcher & Fletcher, 2000). Thus, we have an optimization problem

$$\max_{\Delta\theta} J(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) \approx J(\boldsymbol{\theta}) + \Delta\boldsymbol{\theta}^{\mathrm{T}}\nabla_{\boldsymbol{\theta}}J, \tag{34}$$

$$\text{s.t. } \varepsilon = d_{\mathrm{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \parallel p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}(\boldsymbol{\tau})) \approx \frac{1}{2}\Delta\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{F}_{\boldsymbol{\theta}}\Delta\boldsymbol{\theta},$$

(where $\varepsilon = \text{const}$ is constant) which is illustrated in Fig. 1 and has the solution

$$\Delta\boldsymbol{\theta} = \alpha_n\boldsymbol{F}_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}J \tag{35}$$

with $\alpha_n = [\varepsilon(\nabla J(\boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{F}_{\boldsymbol{\theta}}^{-1}\nabla J(\boldsymbol{\theta}))^{-1}]^{1/2}$, see Peters (2007) for derivations.[10] The direction $\Delta\boldsymbol{\theta}$ is called the natural gradient $\widetilde{\nabla}_{\boldsymbol{\theta}}J(\boldsymbol{\theta}) = \Delta\boldsymbol{\theta}/\alpha_n$ as introduced in Amari (1998). It is not necessary to use the learning rate α_n and it can be replaced by a constant learning rate as it does not affect the gradient direction.

This type of a gradient is known as Natural Policy Gradients and has its separate origin in supervised learning (Amari, 1998). It was first suggested in the context of reinforcement learning in Kakade (2002) and has been explored in greater depth in Bagnell and Schneider (2003), Peters et al. (2003) and Peters et al. (2005a). The strongest theoretical advantage of this approach is that its performance no longer depends on the parameterization of the policy and it is therefore safe to use for arbitrary policies.[11] In practice, the learning process converges significantly faster in most practical cases and requires less manual parameter tuning of the learning algorithm.

4.2. Connection to the compatible function approximation

Up to this point, we have left open the deciding question how to determine the Fisher information matrix. In the first work on natural policy gradients (Kakade, 2002), it appeared that this

[9] While being 'the natural way to think about closeness in probability distributions' (Balasubramanian, 1997), this measure is technically not a metric as it is not commutative.

[10] The value $d_{\mathrm{KL}}(p_{\boldsymbol{\theta}}, p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}})$ can also be seen as the *loss of information* resulting from a policy change $\Delta\boldsymbol{\theta}$. Thus, we could alternatively formulate the problem as the maximization of

$$J(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) - \alpha d_{\mathrm{KL}}(p_{\boldsymbol{\theta}}, p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}) \approx J(\boldsymbol{\theta}) + \Delta\boldsymbol{\theta}^{\mathrm{T}}\nabla_{\boldsymbol{\theta}}J - \frac{\alpha}{2}\Delta\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{F}_{\boldsymbol{\theta}}\Delta\boldsymbol{\theta},$$

with respect to $\Delta\theta$ which obviously has the same solution except to the freely selectable trade-off factor or forgetting rate α.

[11] There are a variety of interesting properties attributed to the natural policy gradient methods which are explored in Peters et al. (2005a).

question could not be answered straightforwardly; however this question was largely answered in subsequent work simultaneously by Bagnell and Schneider (2003) and Peters et al. (2003, 2005a). We summarize our results from Peters et al. (2003) and outline the derivation of Fisher information of paths here. In Moon and Stirling (2000), we can find the well-known lemma that by differentiating $\int_{\mathbb{T}} p(\boldsymbol{\tau})d\boldsymbol{\tau} = 1$ twice with respect to the parameters $\boldsymbol{\theta}$, we can obtain

$$\int_{\mathbb{T}} p(\boldsymbol{\tau})\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau})d\boldsymbol{\tau} = -\int_{\mathbb{T}} \nabla_{\boldsymbol{\theta}} p(\boldsymbol{\tau})\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau})^{\mathrm{T}}d\boldsymbol{\tau}$$

for any probability density function $p(\boldsymbol{\tau})$. Using Eqs. (17) and (18), we can obtain by differentiation

$$\nabla^2_{\boldsymbol{\theta}} \log p\left(\boldsymbol{\tau}_{0:H}\right) = \sum_{k=1}^{H} \nabla^2_{\boldsymbol{\theta}} \log \pi\left(\boldsymbol{u}_k \left|\boldsymbol{x}_k\right.\right). \quad (36)$$

Using the relationship above and Eq. (36), and the definition of the Fisher information matrix (Amari, 1998), we can determine the Fisher information matrix of paths for the average reward case in sample notation, i.e,

$$\begin{aligned}\boldsymbol{F}_{\boldsymbol{\theta}} &= -\left\langle \nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}_{0:H})\right\rangle, \\ &= -\left\langle \sum_{k=0}^{H} \nabla^2_{\boldsymbol{\theta}} \log \pi\left(\boldsymbol{u}_H \left|\boldsymbol{x}_H\right.\right)\right\rangle, \\ &= -\int_{\mathbb{X}} d^{\pi}_{H}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x})\nabla^2_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})d\boldsymbol{u}d\boldsymbol{x}, \\ &= \boldsymbol{G}_{\boldsymbol{\theta}}, \end{aligned} \quad (37)$$

where $d^{\pi}_{H}(\boldsymbol{x}) = \sum_{k=0}^{H} p\left(\boldsymbol{x}_k = \boldsymbol{x}\right)$ denotes the distribution of states along the trajectory. Similarly, if we replace $p(\boldsymbol{\tau}_{0:H})$ by a weighted path distribution $p_{\gamma}(\boldsymbol{\tau}_{0:n}) = p(\boldsymbol{\tau}_{0:n})\sum_{l=0}^{H} a_l \mathbb{I}_{x_i,u_i}$, we see that $\nabla^2_{\boldsymbol{\theta}} \log p\left(\boldsymbol{\tau}_{0:n}\right) = \nabla^2_{\boldsymbol{\theta}} \log p_{\gamma}(\boldsymbol{\tau}_{0:n})$. Thus, the proof above generalizes to reweighted path distributions, i.e., we have $d^{\pi}_{H}(\boldsymbol{x}) = \sum_{k=0}^{H} a_k p\left(\boldsymbol{x}_k = \boldsymbol{x}\right)$. Thus, we can estimate the Fisher information matrix with

$$\boldsymbol{F}_{\boldsymbol{\theta}} = \left\langle \sum_{l=0}^{H} a_l \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_l|\boldsymbol{x}_l)\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_l|\boldsymbol{x}_l)^{\mathrm{T}}\right\rangle \quad (38)$$

as we have shown in Peters et al. (2003). These results imply the equality of the matrix $\boldsymbol{G}_{\boldsymbol{\theta}}$ and the Fisher information $\boldsymbol{F}_{\boldsymbol{\theta}}$ of paths, i.e., we have

$$\boldsymbol{F}_{\boldsymbol{\theta}} = \boldsymbol{G}_{\boldsymbol{\theta}}. \quad (39)$$

Therefore, we have demonstrated that $\boldsymbol{F}_{\boldsymbol{\theta}}$ is indeed a true Fisher information matrix and does not have to be interpreted as the 'average' of the point Fisher information matrices (Kakade, 2002). Eqs. (37) and (35) combined imply that the natural gradient can be computed as

$$\widetilde{\nabla}_{\boldsymbol{\theta}}J(\boldsymbol{\theta}) = \boldsymbol{G}^{-1}_{\boldsymbol{\theta}}\boldsymbol{F}_{\boldsymbol{\theta}}\boldsymbol{w} = \boldsymbol{w}, \quad (40)$$

since $\boldsymbol{F}_{\boldsymbol{\theta}} = \boldsymbol{G}_{\boldsymbol{\theta}}$. Therefore we only need estimate $\boldsymbol{w}$ and *not* $\boldsymbol{G}_{\boldsymbol{\theta}}$. The resulting policy improvement step is thus significantly simplified to become $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha\boldsymbol{w}$ where α denotes a learning rate.

4.3. Natural actor-critic algorithms

We are now in the position to transform the insights from the previous sections into working algorithms, based on the idea of actor-critic methods (Barto, Sutton, & Anderson, 1983). The critic evaluates the current policy π in order to provide the basis for an actor improvement, i.e., the change $\Delta\boldsymbol{\theta}$ of the policy parameters. As we are interested in natural policy gradient updates $\Delta\boldsymbol{\theta} = \alpha\boldsymbol{w}$, we wish to employ the compatible function approximation $f^{\pi}_{w}(\boldsymbol{x}, \boldsymbol{u})$ from Section 3.3 for formulating the critic. In Section 3.3, we had realized that this function was hard to learn as it could only represent an impoverished version of the state-action value function. In order to remedy this situation we will derive more useful estimators from two different points of view, i.e., the state-action based point of view and the episodic roll-out based point of view. Both rely on the assumption of knowing an appropriate basis function representation of the critic's value function, although, as explained below, this assumption is easily fulfilled for the episodic case.

We observe that we can write the Bellman equations in terms of the advantage function and the state-value function (e.g., see Baird (1993))

$$\begin{aligned}Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) &= A^{\pi}(\boldsymbol{x}, \boldsymbol{u}) + V^{\pi}(\boldsymbol{x}) \\ &= r\left(\boldsymbol{x}, \boldsymbol{u}\right) + \gamma \int_{\mathbb{X}} p(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{u})V^{\pi}(\boldsymbol{x}')d\boldsymbol{x}'. \end{aligned} \quad (41)$$

Inserting $A^{\pi}(\boldsymbol{x}, \boldsymbol{u}) = f^{\pi}_{w}(\boldsymbol{x}, \boldsymbol{u})$ and an appropriate basis function representation of the value function as $V^{\pi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{v}$, we can rewrite the Bellman equation (41), as a set of linear equations

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{\phi}(\boldsymbol{x}_t)^{\mathrm{T}}\boldsymbol{v} &= r(\boldsymbol{x}_t, \boldsymbol{u}_t) \\ &\quad + \gamma\boldsymbol{\phi}(\boldsymbol{x}_{t+1})^{\mathrm{T}}\boldsymbol{v} + \epsilon(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1})\end{aligned} \quad (42)$$

where $\epsilon(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1})$ denotes a mean-zero error term, due to the stochastic policy in (41). We call algorithms that make use of Eq. (42) to obtain the natural gradient **Natural Actor-Critic Algorithms**.

The state-action based point of view of actor-critic algorithms and its difficulties were discussed in Peters et al. (2005a). Here, we will focus on the much simpler episodic case, which is also the most suitable one for our interest in employing motor primitives for control. Thus, in the next section, we introduce the family of Episodic Natural Actor-Critic algorithms.

4.3.1. Episodic natural actor-critic

As mentioned in the previous section, the critic in actor-critic algorithms needs to approximate the value function, which requires a suitable parameterization, usually in the form of a linear combination of basis functions. If chosen inappropriately, the estimate of the natural gradient becomes biased (Peters et al., 2005a). For the episodic case, however, this problem does not exist. We can derive the episodic natural actor-critic by summing up Eq. (42) along a sample path to obtain

$$\sum_{t=0}^{H} a_t A^{\pi}(\boldsymbol{x}_t, \boldsymbol{u}_t) = a_{H+1}V^{\pi}(\boldsymbol{x}_{H+1}) + \sum_{t=0}^{H} a_t r(\boldsymbol{x}_t, \boldsymbol{u}_t) - V^{\pi}(\boldsymbol{x}_0). \quad (43)$$

The term multiplied by a_{H+1} on the right side disappears for the discounted learning problem as $H \to \infty$ (or H just becomes large enough) or for episodic tasks (where $r(\boldsymbol{x}_H, \boldsymbol{u}_H)$ is the final reward). Therefore each roll-out yields one equation which is linear in the parameters $\boldsymbol{w}$. If we furthermore assume a single start-state (or a mean-zero start-state distribution), only one additional value is required to estimate $J_0 = V^{\pi}(\boldsymbol{x}_0)$, which corresponds to estimating the critic's value at the start-state $\boldsymbol{x}_0$. Therefore, we get a straightforward regression problem:

$$\sum_{t=0}^{H} a_t \nabla \log \pi(\boldsymbol{u}_t, \boldsymbol{x}_t)^{\mathrm{T}}\boldsymbol{w} + J_0 = \sum_{t=0}^{H} a_t r(\boldsymbol{x}_t, \boldsymbol{u}_t) \quad (44)$$

with exactly $\dim\theta + 1$ unknowns. This means that for non-stochastic tasks we can obtain a natural gradient after $\dim\theta + 1$ roll-outs using least-squares regression

$$\begin{bmatrix}\boldsymbol{w} \\ J_0\end{bmatrix} = \left(\boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{R}, \quad (45)$$

Table 5
Episodic natural actor-critic with a constant baseline

```
input: policy parameterization $\boldsymbol{\theta}$.
1  repeat
2      perform $M$ trials and obtain $\boldsymbol{x}_{0:H}, \boldsymbol{u}_{0:H}, r_{0:H}$ for each trial.
       Obtain the sufficient statistics
3      Policy derivatives $\boldsymbol{\psi}_k = \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k | \boldsymbol{x}_k)$.
4      Fisher matrix $\boldsymbol{F}_{\boldsymbol{\theta}} = \left\langle \left(\sum_{k=0}^{H} \boldsymbol{\psi}_k\right)\left(\sum_{l=0}^{H} \boldsymbol{\psi}_l\right)^{\mathrm{T}} \right\rangle$.
       Vanilla gradient $\boldsymbol{g} = \left\langle \left(\sum_{k=0}^{H} \boldsymbol{\psi}_k\right)\left(\sum_{l=0}^{H} a_l r_l\right) \right\rangle$.
5      Eligibility $\boldsymbol{\phi} = \left\langle \left(\sum_{k=0}^{H} \boldsymbol{\psi}_k\right) \right\rangle$.
6      Average reward $\bar{r} = \left\langle \sum_{l=0}^{H} a_l r_l \right\rangle$.
       Obtain natural gradient by computing
7      Baseline $b = \boldsymbol{Q}\left(\bar{r} - \boldsymbol{\phi}^{\mathrm{T}} \boldsymbol{F}_{\boldsymbol{\theta}}^{-1} \boldsymbol{g}\right)$
       with $\boldsymbol{Q} = M^{-1}\left(1 + \boldsymbol{\phi}^{\mathrm{T}}\left(M\boldsymbol{F}_{\boldsymbol{\theta}} - \boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{T}}\right)^{-1} \boldsymbol{\phi}\right)$
8      Natural gradient $\boldsymbol{g}_{\mathrm{eNAC1}} = \boldsymbol{F}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{g} - \boldsymbol{\phi} b)$.
9  until gradient estimate $\boldsymbol{g}_{\mathrm{eNAC1}}$ converged.
return: gradient estimate $\boldsymbol{g}_{\mathrm{eNAC1}}$.
```

Table 6
Episodic natural actor-critic with a time-variant baseline

```
input: policy parameterization $\boldsymbol{\theta}$.
1  repeat
2      perform $M$ trials and obtain $\boldsymbol{x}_{0:H}, \boldsymbol{u}_{0:H}, r_{0:H}$ for each trial.
       Obtain the sufficient statistics
3      Policy derivatives $\boldsymbol{\psi}_k = \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k | \boldsymbol{x}_k)$.
4      Fisher matrix $\boldsymbol{F}_{\boldsymbol{\theta}} = \left\langle \sum_{k=0}^{H} \left(\sum_{l=0}^{k} \boldsymbol{\psi}_l\right) \boldsymbol{\psi}_k^{\mathrm{T}} \right\rangle$.
       Vanilla gradient $\boldsymbol{g} = \left\langle \sum_{k=0}^{H} \left(\sum_{l=0}^{k} \boldsymbol{\psi}_l\right) a_k r_k \right\rangle$,
5      Eligibility matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_K]$
       with $\boldsymbol{\phi}_h = \left\langle \left(\sum_{k=0}^{h} \boldsymbol{\psi}_k\right) \right\rangle$.
6      Average reward vector $\bar{\boldsymbol{r}} = [\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_K]$
       with $\bar{r}_h = \langle a_h r_h \rangle$.
       Obtain natural gradient by computing
7      Baseline $\boldsymbol{b} = \boldsymbol{Q}\left(\bar{\mathbf{r}} - \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{F}_{\boldsymbol{\theta}}^{-1} \boldsymbol{g}\right)$
       with $\boldsymbol{Q} = M^{-1}\left(\boldsymbol{I}_K + \boldsymbol{\Phi}^{\mathrm{T}}\left(M\boldsymbol{F}_{\boldsymbol{\theta}} - \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\right)^{-1} \boldsymbol{\Phi}\right)$.
8      Natural gradient $\boldsymbol{g}_{\mathrm{NG}} = \boldsymbol{F}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{g} - \boldsymbol{\Phi}\boldsymbol{b})$.
9  until gradient estimate $\boldsymbol{g}_{\mathrm{eNACn}}$ converged.
return: gradient estimate $\boldsymbol{g}_{\mathrm{eNACn}}$.
```

with

$$\boldsymbol{\Psi}_i = \left[\sum_{t=0}^{H} a_t \nabla \log \pi(\boldsymbol{u}_t, \boldsymbol{x}_t)^{\mathrm{T}}, 1\right], \tag{46}$$

$$\boldsymbol{R}_i = \sum_{t=0}^{H} a_t r(\boldsymbol{x}_t, \boldsymbol{u}_t). \tag{47}$$

This regression problem, can be transformed into the form shown in Table 5 using the matrix inversion lemma, see Peters (2007) for the derivation.

4.3.2. *Episodic natural actor-critic with a time-variant baseline*

The episodic natural actor-critic described in the previous section suffers from one drawback: it does not make use of intermediate reward data along the roll-out, just like REINFORCE. An improvement was suggested by G(PO)MDP, which left out terms which would average out in expectation. One can argue that this omission of terms is equivalent to using a time-dependent baseline. We can make use of the same argument and reformulate the Episodic Natural Actor-Critic which results in the algorithm shown in Table 6. The advantage of this type of algorithms is two-fold: the variance of the gradient estimate is often lower and it can take time-variant rewards significantly better into account.

The time-variant baseline can also be seen in a slightly different light as it is not only a baseline but at the same time an additional basis function for the critic. As we argued in the previous section, a single constant offset suffices as additional basis function as it only needs to represent the value of the first state (or the average of first states as the start-state distribution does not depend on the policy). Obviously, using more basis functions than one constant offset representing the value of the first step can reduce the variance of the gradient estimate even further without introducing bias. We know from the non-episodic Natural Actor-Critic, that if we had large experience with the system and knew state-dependent basis functions, we can obtain the much better gradient estimate. However, we also know that we rarely have these good additional basis functions. Nevertheless, in some problems, we stay close to certain kinds of trajectories. In this important case, the state-dependent basis functions can be replaced by time-dependent or time-variant basis functions with an equivalent result.

Nevertheless, if we have a time-variant baseline in an episodic setup, this will increase the complexity of the current setup and the regression problem has a solution

$$\begin{bmatrix} \boldsymbol{g}_{\mathrm{eNACn}} \\ \bar{\boldsymbol{r}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_2 & \bar{\boldsymbol{\Phi}} \\ \bar{\boldsymbol{\Phi}}^{\mathrm{T}} & m\boldsymbol{I}_H \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{g} \\ \bar{\boldsymbol{r}} \end{bmatrix} \tag{48}$$

where we have a Fisher information matrix estimate $\boldsymbol{F}_{\boldsymbol{\theta}} = \left\langle \sum_{k=0}^{H} \left(\sum_{l=0}^{k} \boldsymbol{\psi}_l\right) \boldsymbol{\psi}_k^{\mathrm{T}} \right\rangle$ with log-policy derivatives $\boldsymbol{\psi}_k = \nabla_\theta \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_k | \boldsymbol{x}_k)$, the vanilla gradient estimate $\boldsymbol{g} = \left\langle \sum_{k=0}^{H} \left(\sum_{l=0}^{k} \boldsymbol{\psi}_l\right) a_k r_k \right\rangle$, an eligibility matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_K]$ with $\boldsymbol{\phi}_h = \left\langle \left(\sum_{k=0}^{h} \boldsymbol{\psi}_k\right) \right\rangle$ and $\bar{\boldsymbol{r}} = [\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_K]$ with $\bar{r}_h = \langle a_h r_h \rangle$. Computing $\boldsymbol{g}_{\mathrm{eNACn}}$ thus involves the inversion of a matrix which has the size $H + n$, i.e., the sum of the episode length H and parameters n. Obviously, such a matrix is prone to be ill-conditioned if inverted with a brute-force approach and computationally expensive. Nevertheless, as this matrix contains the identity matrix $\boldsymbol{I}_H$ in the largest block of the matrix, we can reduce the whole problem to an inversion of a matrix which has the number of policy parameters n as size. When making use of the matrix inversion theorem (using Harville (2000), pages 98–101, or Moon and Stirling (2000), pages 258–259; for details see also Peters (2005)), we can simplify this approach to

$$\boldsymbol{w} = \boldsymbol{\beta}_1 = \boldsymbol{F}_2^{-1}\left(\boldsymbol{g} - \bar{\boldsymbol{\Phi}} b\right) = \boldsymbol{F}_2^{-1} \boldsymbol{g}_2, \tag{49}$$

$$\boldsymbol{b} = \beta_2 = \boldsymbol{Q}^{-1}\left(\bar{r} - \bar{\boldsymbol{\Phi}}^{\mathrm{T}} \boldsymbol{F}_2^{-1} \boldsymbol{g}\right), \tag{50}$$

with $\boldsymbol{Q}^{-1} = m^{-1}(\boldsymbol{I}_n + \bar{\boldsymbol{\Phi}}^{\mathrm{T}}(m\boldsymbol{F}_2 - \bar{\boldsymbol{\Phi}}\bar{\boldsymbol{\Phi}}^{\mathrm{T}})^{-1}\bar{\boldsymbol{\Phi}})$. This algorithm is depicted in Table 6, and detailed derivations can be found in Peters (2005).

5. Empirical evaluations

In the previous section, we outlined five model-free policy gradient algorithms. From our assessment, these are among the most relevant for learning motor primitives in robotics as they can be applied without requiring additional function approximation methods and as they are suitable for episodic settings. These algorithms were (i) finite-difference gradient estimators, the vanilla policy gradient estimators (ii) with a constant baseline and (iii) a time-variant baseline, as well as the episodic Natural Actor-Critic with both (iv) an offset as additional basis functions and (v) a time-variant offset as additional basis functions.

In this section, we will demonstrate how the different algorithms compare in practice. For this purpose, we will show experiments on both a simulated plant as well as on a real robot and we will compare the algorithms for the optimization of control laws and for learning of motor skills.

5.1. Comparing policy gradient methods on motor primitives

Our goal in this section is to evaluate policy gradient methods for the applicability to learning motor primitives. As outlined in Section 1.2, we are interested in two different kinds of representations, i.e., a spline-based and dynamical systems.

In a spline-based approach (e.g., Miyamoto et al. (1996) and Sciavicco and Siciliano (2007)), we have a series of concatenated splines, which together form a desired trajectory — third-order splines or fifth-order splines are among the most popular choices. Boundary conditions between the individual splines ensure smooth trajectories with continuous derivatives up to a certain order, which are determined by the order of the spline polynomial. Depending on the design principle, the splines can have equal or different durations. The pros of spline-based trajectory planning are the ease of use of splines and the compact representation that consists only of the spline boundary conditions and timing. Among the main short-comings (Miyamoto et al., 1996; Sciavicco & Siciliano, 2007) are that between spline nodes, splines can sometimes create rather complex and undesirable behavior, and that the explicit time indexing of splines makes it hard to have on-line modifications of the spline plan, e.g., in terms of potential fields due to obstacle avoidance. We included spline-based trajectory planning as a base-line comparison for our learning algorithms, as the spline parameters are perfectly suited as a representation for parameterized movement primitives. In our implementation, we used third-order splines of equal time duration to span the desired movement duration. Third-order splines have four boundary conditions, i.e., position and velocity at each node, and between splines, the boundary conditions of the ending spline are the same as those of the starting spline. We chose six splines for the entire trajectory, resulting in a total of 10 free parameters (the start and end conditions of the desired trajectory are given).

As an alternative to splines, a dynamical system approach was suggested in Ijspeert et al. (2002, 2003). The basic principle here is that a by-design globally stable attractor system automatically creates a desired trajectory through its unfolding in time. Given that the dynamical system is autonomous, potential fields can easily be incorporated to allow for on-line modification of the desired trajectory, e.g., due to the need of obstacle avoidance. Additionally, scaling in speed and amplitude of movement is easily accomplished, derived from a framework of structural equivalence of dynamical systems. One dynamical system motor primitive represents a large family of movements, as it is valid for any initial condition. More details on this approach can be found in Appendix or in Ijspeert et al. (2002, 2003). The open parameters in the dynamical system motor primitive are the coefficients of a function approximator with linear parameters (cf. Appendix). As we used ten basis functions in the function approximator, we obtained 10 free parameters in the motor primitive, i.e., the same number of free parameters as in the spline approach.

We compared the algorithms (i)–(v) in four different scenarios. Each of these scenarios consists of the usage of one of the kinematic plan representations (i.e., splines or motor primitives) and two different tasks. Each task represents a motor plan in joint space represented by joint angle q and velocity $\dot{q}$. These tasks were chosen such that we can evaluate the different policy gradient methods for learning standard motor primitives as they appear frequently in the motor control literature (Ben-Itzhak & Karniel, in press; Flash & Hochner, 2005) as well as for applicability in the T-ball task presented in Section 5.2. The optimal policies for these tasks can be found in the literature, e.g., in Ben-Itzhak and Karniel (in press) and Flash and Hochner (2005). In all the experiments in this section, we used a single Degree-of-Freedom (DoF) scenario for the ease of illustration. To optimize the performance, we improved every gradient estimator as much as possible. Therefore each algorithm has learning rates that were manually chosen to maximize the performance while not destabilizing the gradient descent. In order to curb the variance of the gradient estimates, we make use of the PEGASUS trick (Ng & Jordan, 2000) for the generation of exploration (i.e., the perturbation of our motor commands is achieved using the same history of random numbers). All other settings were optimized in order to make each of these algorithms as good as possible, e.g., the random perturbation frequency of the different algorithms is tuned for maximum performance, as perturbations at too high frequency result in very slow learning, and perturbations at too low frequency can destabilize motor control.

The first task is to achieve a goal with a minimum-squared movement acceleration and a given movement duration, i.e., a reward of

$$r(\boldsymbol{x}_{0:H}, \boldsymbol{u}_{0:H}) = -\sum_{i=0}^{H/2} c_1 \ddot{q}_i^2 - \sum_{i=\frac{H}{2}+1}^{H} c_2[(q_i - g)^2 + \dot{q}_i^2] \tag{51}$$

is being optimized, where $c_1 = 1/100$ is the weight of the transient rewards for the movement duration $H/2$, while $c_2 = 1000$ is the importance of the final reward, extended over the time interval $[H/2 + 1, H]$, which insures that the goal state $g = 1.0$ is reached and maintained properly. The start-state of the motor primitive is always zero in this toy evaluation. The optimal movement with respect to this reward is known to be a third-order polynomial in time, with boundary conditions set according to the movement start and goal (Ben-Itzhak & Karniel, in press). We use this analytically optimal solution as a baseline, which is unachievable for the motor primitives due to the pre-requisite of smooth acceleration profiles — the third-order spline optimal solution has discontinuous accelerations at the start and end of the movement.

In Fig. 2, the task and the different solutions are illustrated. For the spline representation, all algorithms achieve the same good result. For the motor primitives, the best solution is a very close approximation of the analytically optimal solutions. The finite-difference methods do not reach this optimal solution but get stuck in a local minimum.

The second task involves passing through an intermediate point during the trajectory, while minimizing the squared accelerations, i.e., we have a similar reward with an additional punishment term for missing the intermediate point p_F at time F given by

$$\begin{aligned} r(\boldsymbol{x}_{0:H}, \boldsymbol{u}_{0:H}) &= -\sum_{i=0}^{H/2} \tilde{c}_1 \ddot{q}_i^2 - \sum_{i=H/2+1}^{H} \tilde{c}_2[(q_i - g)^2 + \dot{q}_i^2] \\ &\quad - \tilde{c}_3 (q_F - p_F)^2, \end{aligned} \tag{52}$$

where $\tilde{c}_1 = 1$, $\tilde{c}_2 = 20$, $\tilde{c}_3 = 20\,000$. The goal has a value of $g = 1$, the intermediate point a value of $p_F = 0.5$ at $F = 7H/20$ and the start-state was zero. This reward yields a smooth movement which first passes through the intermediate point before reaching the goal. This toy experiment is in preparation for hitting a baseball in the robot experiment below.

In Fig. 3, the second task and the different solutions are illustrated. Again, we compare against the analytically optimal solution described by two third-order splines, whose velocity boundary conditions at the intermediate point are optimized according to the minimum-squared acceleration cost. As in the first task, the motor primitives can only approximate this optimal solution due to smoothness constraints of the acceleration profiles. The results are very similar as those for the first task: for the motor primitives, a good approximation of the analytically optimal solution is achieved by all but the finite-difference method, which seems to get stuck in local minima. The spline representation shows no significant differences among the five different learning

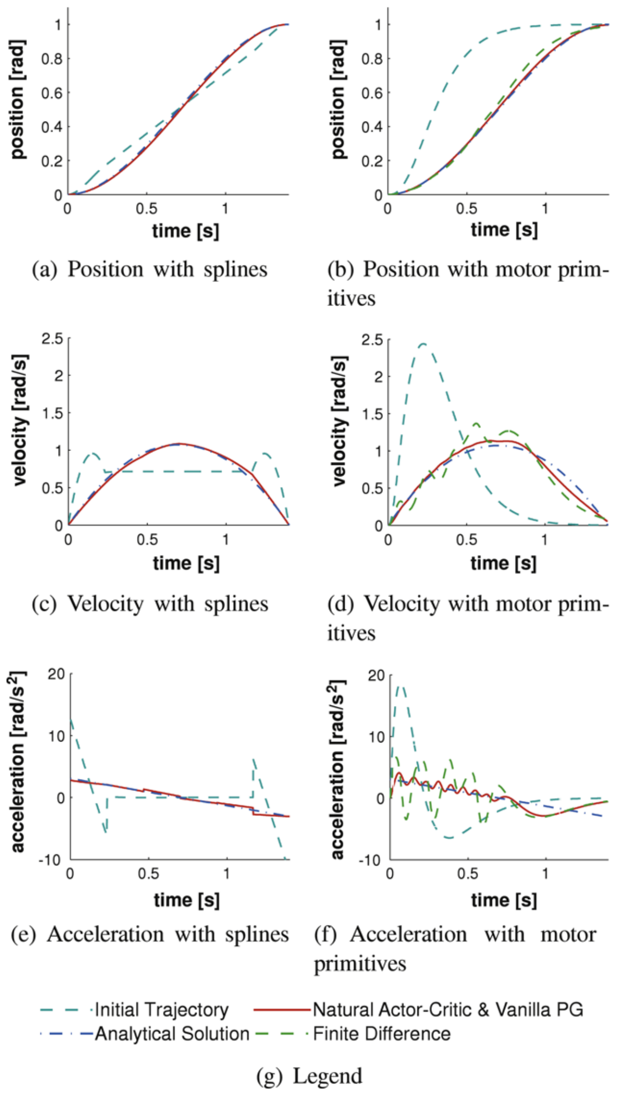


Fig. 2. This figure illustrates the task accomplished in the minimum motor command learning example. In (a, b), the positions of splines and motor primitives are shown, in (c, d) the velocities and in (e, f) the accelerations. The cyan dashed line shows the initial configurations, which is accomplished by straight-line movement initializations for the splines, and zero parameters for the motor primitives. The dash-dotted dark blue line shows the analytically optimal solution, which is unachievable for the motor primitives, but nicely approximated by their best solution, presented by the red solid line. This best solution is reached by all learning methods except for the finite-difference method, which gets stuck in a local minimum — shown as the green dashed line. The wiggly curve in the middle of the motor primitive acceleration results from the basis function approximation of this curve with Gaussian basis functions — the number of wiggles corresponds to the number of basis functions. The spline representation achieves very close to optimal results for all algorithms. The legend is given in (g). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

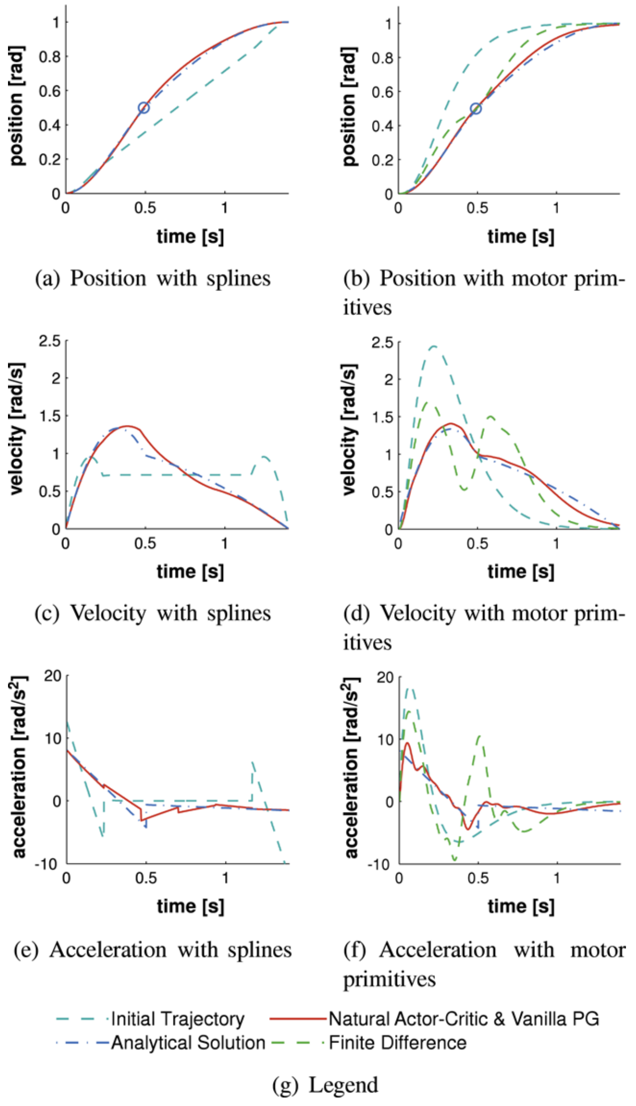
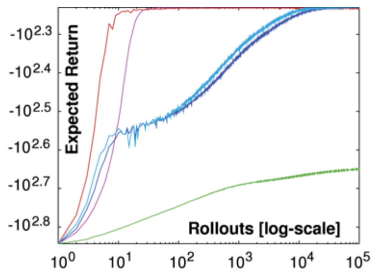


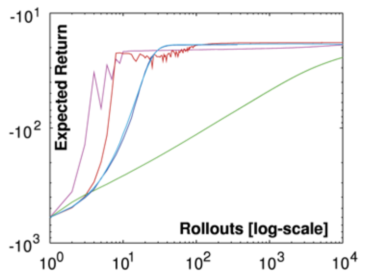
Fig. 3. This figure illustrates the results when learning to pass through an intermediate point while having smooth motion. In (a, b), the positions of splines and motor primitives are shown, in (c, d) the velocities and in (e, f) the accelerations. The dashed cyan line shows the initial configurations, which is accomplished by straight-line movement initializations for the splines, and zero parameters for the motor primitives. The dash-dotted dark blue line shows the analytically optimal solution, which is unachievable for the motor primitives, but nicely approximated by their best solution, presented by the red solid line. This best solution is reached by all learning methods except for the finite-difference method, which gets stuck in a local minimum — shown as the green dashed line. The best solution achieved by learning is presented by the red solid line, which was very close to the analytically optimal one. The legend is given in (g). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

methods, but the learned trajectory does not quite coincide with the optimal solution, as none of the spline nodes corresponded exactly to the intermediate point p_F.

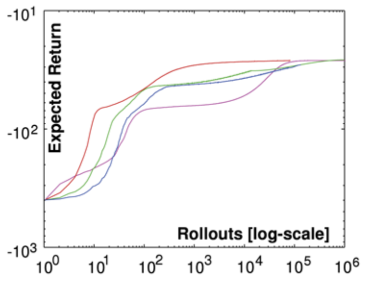
While the trajectory plots do not really distinguish the different learning algorithms, except for the finite-difference method, the learning curves demonstrate significant and interesting differences. In Fig. 4(a) and (b), we show a comparison for the first task (goal achievement with a minimum-squared motor command) for both splines and dynamic motor primitives, respectively. In Fig. 4(c) and (d) we show similar plots for the task with the intermediate target. These comparisons clearly establish that both versions of the episodic natural actor-critic methods outperform both the vanilla policy gradient methods as well as the finite-difference methods. The difference between the episodic natural actor-critic methods with single-offset and the time-variant additional basis functions is relatively small; however, for practical tasks this can still make a significant difference. The time-invariant baseline barely improves over the constant baseline for the vanilla policy gradients. For finite-difference gradient methods, it is best not to apply exploration during the trajectory but only perturb the parameters at the start of the episode. Otherwise, the gradient estimate is too noisy for any practical purposes. While this lack of exploration results into an increased accuracy of the gradient, the lack of stochasticity in the system results in an increase of plateaus and local minima (Ng & Jordan, 2000; Spall, 2003). This problem could be overcome by collecting sufficiently many samples with additional stochastic, exploratory



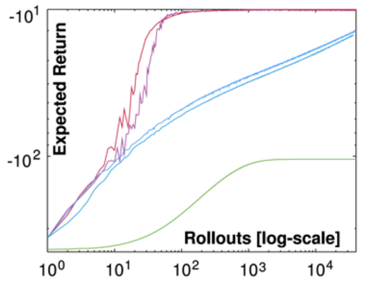
(a) Minimum motor command with splines



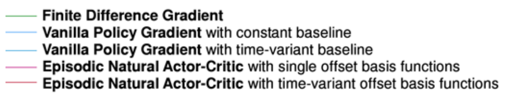
(b) Minimum motor command with motor primitives



(c) Passing through a point with splines



(d) Passing through a point with motor primitives


Finite Difference Gradient
Vanilla Policy Gradient with constant baseline
Vanilla Policy Gradient with time-variant baseline
Episodic Natural Actor-Critic with single offset basis functions
Episodic Natural Actor-Critic with time-variant offset basis functions

(e) Legend

Fig. 4. This figure shows different experiments with learning motor tasks. In (a, b), we see how the learning system creates plans for the minimum motor command cost using both (a) splines and (b) motor primitives. For this problem, the natural actor-critic methods learns the optimal solution faster by several orders of magnitude than any of the other methods. In (c, d), the plan has to achieve an intermediary goal. While the natural actor-critic methods still outperform previous methods, the gap is lower as the learning problem is easier. Note that these are double logarithmic plots. Both 'vanilla' policy gradient methods and natural ones achieve the same final solution — just at a different pace. Finite-difference methods on the other hand often do not attain a similarly good solution as in (a, d).

actions, but at the cost of many more roll-outs. Both vanilla policy gradient methods and the natural actor-critic methods avoid these local minima due to the usage of exploratory actions instead of parameter perturbations.

From the results presented in Fig. 4(a–d), we conclude that natural actor-critic methods seem to have a significant advantage for our learning goals, i.e., optimizing the parameters of control policies.

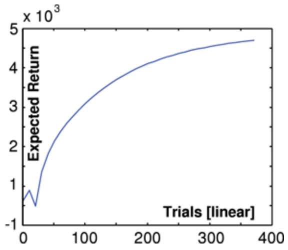
5.2. *Robot application: Motor primitive learning for baseball*

In this section, we apply the Episodic Natural Actor-Critic in a multi-DOF robot motor learning task, using a Sarcos Master Arm. The task of the robot is to hit a (soft) baseball placed on a T-stick such that it flies as far as possible — this game is also known as T-Ball and is used in the US to teach children how to hit a baseball. The state of the robot is given by its joint angles and velocities while the actions are the joint accelerations. A given inverse dynamics controller is used to obtain motor torques from these kinematic variables. The ball position and velocity are observed through a color vision system (Newton Labs, MA) at 60 Hz video frequency.

There are several complex components in this setup. The ball has to be hit at a proper location and proper angle with maximal velocity while avoiding to saturate the robot torques — this is modeled by minimizing the squared motor commands, i.e., acceleration of the motor primitives. Additionally, the task has to be distributed appropriately among the seven degrees of freedom of the robot, which are redundant during the hitting movement. As each degree of freedom has a single motor primitive representing its behavior over time, the redundancies require two implementational simplifications. The first simplification is the usage of the same random number generator for exploration across all seven motor primitives. This simplification is well known to introduce similar exploration over all degrees of freedom and has been shown to reduce the variance in the gradient estimate (Fu, 2002). It is necessary, as otherwise the exploration noise added in one DOF will often "fight" the exploration noise of other DOFs, resulting in very slow learning. Second, we need to cope with a 70-dimensional Fisher information matrix, which is hard to stabilize numerically. However, it is a reasonable assumption that this Fisher information matrix is dominated by its block diagonal as the parameters of the same motor primitive are more tightly coupled than between different motor primitives. This results into treating the motor primitive of each DOF as if they could be learned independently, i.e., we do not need to treat all the parameters of all motor primitives as one big optimization problem. Note that this approximation of the Fisher information matrix is positive definite and, thus, inherits all convergence guarantees from the 'vanilla' policy gradients while still yielding speeds comparable to an implementation of the Episodic Natural Actor-Critic without this simplification. The joint reward signal for each motor primitive for degree of freedom k was

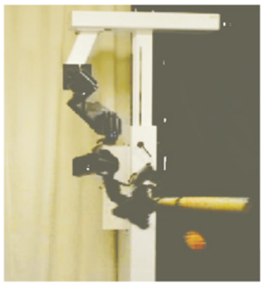
$$r_k(\boldsymbol{x}_{0:H}, \boldsymbol{u}_{0:H}) = +\hat{c}_1 \boldsymbol{p}_x - \sum_{i=0}^{H} \hat{c}_2 \ddot{\boldsymbol{q}}_{k;i}^2 \tag{53}$$

where $\boldsymbol{p}_x$ denotes the distance of the ball travelled (as estimated from the equations of a ballistic flight using initial trajectory captured by the vision system — the scope of the vision system was zoomed in the area around the ball to obtain high resolution information and could not see where the ball landed) along the x-axis while $\ddot{\boldsymbol{q}}_{k;i}^2$ punishes high joint accelerations of joint k at time i, where the weight of each cost component is given by $\hat{c}_1 = 1000$, $\hat{c}_2 = 1$. Note, that for a single DoF this Eq. (53) would be a more abstract version of Eq. (52). As policy gradient methods are local learning methods, we cannot expect to learn this complex, high-dimensional behavior in a short time unless we initialize the motor primitives with some prior knowledge. For motor control,



(a) Performance.

(b) Imitation learning.


(c) Initial reproduction.

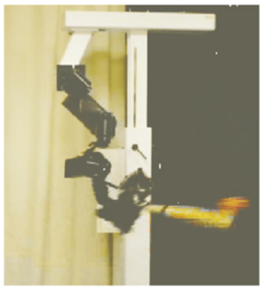

(d) After reinforcement learning.

Fig. 5. This figure shows (a) the average of the reward, Eq. (53), over all DoFs of the baseball swing task when using the motor primitives for learning. In (b), the learning system is initialized by imitation learning, in (c) it is initially failing at reproducing the motor behavior, and (d) after several hundred episodes exhibiting a nicely learned batting, which achieves about 5 m distance of the ball flight.

a suitable approach is to first imitate a teacher and, subsequently, improve by trial and error. This setup mimics how children learn how to a hit a baseball where first the parent presents and, subsequently, the child improves on its own. Similarly, we use the one-shot supervised learning setup[12] presented in Ijspeert et al. (2002, 2003) to teach the robot a rudimentary T-ball stroke as can be seen in Fig. 5(b); however, it fails to reproduce the behavior as shown in (c) due to control inaccuracies of the robot; subsequently, we improve the accuracy of the hitting angle using the episodic Natural Actor-Critic which yields the performance shown in (a) and the behavior in (d). After approximately 200–300 trials, the ball can be hit properly by the robot.

6. Conclusion & discussion

We have presented an extensive survey of policy gradient methods. While some developments needed to be omitted as they are only applicable for very low-dimensional state-spaces, this paper largely summarized the state of the art in policy gradient methods as applicable in robotics with high degree-of-freedom movement systems. All the three major ways of estimating first-order gradients, i.e., finite-difference gradients, vanilla policy gradients and natural policy gradients are discussed in this paper and practical algorithms are given.

One of the presented classes of algorithms, the Natural Actor-Critic algorithms was developed for this paper. This class of algorithms has been widely accepted by now and has been applied in a variety of settings (Guenter, Hersch, Calinon, & Billard, in press; Mori et al., 2004; Mori, Nakamura, & Ishii, 2005; Nakamura et al., 2004; Park, Kim, & Kang, 2005; Richter, Aberdeen, & Yu, 2007; Sato et al., 2002; Ueno et al., 2006). The Natural Actor-Critic is considered the "Current method of choice" (Aberdeen, 2006) among the policy gradient methods in the reinforcement learning community. It also allows the derivation of several previously presented algorithms in the literature and has very useful theoretical properties (Peters, 2007).

The experiments presented here show that the time-variant episodic natural actor-critic is the preferred method among the presented methods when applicable; however, if a policy cannot be differentiated with respect to its parameters, the finite-difference methods may be the only method applicable. The example of motor primitive learning for baseball underlines the efficiency of natural gradient methods for complex movement systems.

[12] This setup first extracts the duration of the desired movement and adjusts all time constants of the motor primitives. Based on the resulting dynamical systems, it computes the targets for the a locally weighted regression performed based on LWPR. See Ijspeert et al. (2002, 2003) for all details on this approach.

Appendix. Motor primitive equations

The motor primitives from Ijspeert et al. (2002, 2003) in their most recent reformulation are given by a canonical system

$$\tau^{-1}\dot{v} = \alpha_v\left(\beta_v\left(g - x\right) - v\right), \tag{A.1}$$

$$\tau^{-1}\dot{x} = v, \tag{A.2}$$

which represents the phase of the motor process. It has a goal g, a time constant τ and some parameters α_v,β_v which are chosen so that the system is stable. Additionally, we have a transformed system

$$\tau^{-1}\dot{z} = \alpha_z\left(\beta_z\left(s - x\right) - v\right) + f\left(x, v, g\right), \tag{A.3}$$

$$\tau^{-1}\dot{y} = z, \tag{A.4}$$

$$\tau^{-1}\dot{s} = \alpha_s\left(g - s\right), \tag{A.5}$$

which has the same time constant τ as the canonical system, appropriately set parameters α_z,β_z,α_s, and a transformation function $f(x, v, g)$. The transformation function transforms the output of the canonical system so that the transformed system can represent complex nonlinear patterns and is given by

$$f\left(x, v, g\right) = \frac{\sum_{i=1}^{N}\psi_i\left(x\right)\theta_i v}{\sum_{i=1}^{N}\psi_i\left(x\right)}, \tag{A.6}$$

where θ_i are adjustable parameters and it has localization weights defined by

$$\psi_i\left(x\right) = \exp\left(-h_i\left(\frac{x - x_0}{g - x_0} - c_i\right)^2\right) \tag{A.7}$$

with offset x_0, centers c_i and width h_i.

References

Aberdeen, D. (2006). POMDPs and policy gradients, presentation at the Machine Learning Summer School (MLSS).

Aleksandrov, V., Sysoyev, V., & Shemeneva, V. (1968). Stochastic optimization. *Engineering Cybernetics*, *5*, 11–16.

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*, 251.

Atkeson, C. G. (1994). Using local trajectory optimizers to speed up global optimization in dynamic programming. In J. E. Hanson, S. J. Moody, & R. P. Lippmann (Eds.), *Advances in neural information processing systems 6* (pp. 503–521). Morgan Kaufmann.

Bagnell, J., & Schneider, J. (2003). Covariant policy search. In *Proceedings of the international joint conference on artificial intelligence* (pp. 1019–1024).

Baird, L. (1993). Advantage updating. *Technical Report WL-TR-93-1146. Wright laboratory, Wright–Patterson air force base*. OH.

Balasubramanian, V. (1997). Statistical inference, occam's razor, and statistical mechanics on the space of probability distributions. *Neural Computation*, *9*(2), 349–368.

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

Baxter, J., & Bartlett, P. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, *15*, 319–350.

Baxter, J., Bartlett, P., & Weaver, L. (2001). Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research*, *15*, 351–381.

Ben-Itzhak, S., & Karniel, A. (2008). Minimum acceleration criterion with constraints implies bang–bang control as an underlying principle for optimal trajectories of arm reaching movements. *Neural Computation*, *20*(3), 779–812.

Benbrahim, H., Doleac, J., Franklin, J., & Selfridge, O. (1992). Real-time learning: A ball on a beam. In *Proceedings of the international joint conference on neural networks* (pp. 92–103).

Benbrahim, H., & Franklin, J. (1997). Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems*, *22*, 283–302.

Berny, A. (2000). Statistical machine learning and combinatorial optimization. In *Lecture notes in natural computing*: *Vol. 33* (pp. 287–306). Heidelberg, Germany: Springer-Verlag.

Dyer, P., & McReynolds, S. R. (1970). *The computation and theory of optimal control*. New York, NY: Academic Press.

Endo, G., Morimoto, J., Matsubara, T., Nakanishi, J., & Cheng, G. (2005). Learning cpg sensory feedback with policy gradient for biped locomotion for a full-body humanoid. In *Proceedings of the national conference on artificial intelligence* (pp. 1267–1273).

Flash, T., & Hochner, B. (2005). Motor primitives in vertebrates and invertebrates. *Current Opinions in Neurobiology*, *15*, 660–666.

Fletcher, R., & Fletcher, R. (2000). *Practical methods of optimization*. New York, NY: John Wiley & Sons.

Fu, M. C. (2002). Feature article: Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, *14*(3), 192–215.

Glynn, P. (1987). Likelihood ratio gradient estimation: An overview. In *Proceedings of the winter simulation conference* (pp. 366–375).

Glynn, P. (1990). Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, *33*(10), 75–84.

Greensmith, E., Bartlett, P., & Baxter, J. (2001). Variance reduction techniques for gradient estimates in reinforcement learning. *Advances in Neural Information Processing Systems*, *14*(34).

Greensmith, E., Bartlett, P. L., & Baxter, J. (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, *5*, 1471–1530.

Guenter, F., Hersch, M., Calinon, S., & Billard, A. (2007). Reinforcement learning for imitating constrained reaching movements. *In Imitative Robots [Special issue]. RSJ Advanced Robotics, 21*, 1521–1544.

Gullapalli, V. (1990). A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, *3*(6), 671–692.

Gullapalli, V. (1992). Learning control under extreme uncertainty. In *Advances in neural information processing systems* (pp. 327–334).

Gullapalli, V., Franklin, J., & Benbrahim, H. (1994). Aquiring robot skills via reinforcement learning. *IEEE Control Systems Journal, Special Issue on Robotics: Capturing Natural Motion*, *4*(1), 13–24.

Harville, D. A. (2000). *Matrix algebra from a statistician's perspective*. Heidelberg, Germany: Springer Verlag.

Hasdorff, L. (1976). *Gradient optimization and nonlinear control*. New York, NY: John Wiley & Sons.

Ijspeert, J.A., Nakanishi, J., & Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of IEEE international conference on robotics and automation* (pp. 1398–1403).

Ijspeert, A., Nakanishi, J., & Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in neural information processing systems*: *Vol. 15* (pp. 1547–1554). Cambridge, MA: MIT Press.

Jacobson, D. H., & Mayne, D. Q. (1970). *Differential dynamic programming*. New York, NY: American Elsevier Publishing Company, Inc.

Kakade, S. (2001). Optimizing average reward using discounted rewards. In *Proceedings of the conference on computational learning theory (pp. 605–615)*.

Kakade, S. A. (2002). Natural policy gradient. In *Advances in neural information processing systems*: *Vol. 14* (pp. 1531–1538). CA: Vancouver.

Kakade, S.M. (2003). On the sample complexity of reinforcement learning. *Ph.D. thesis*, Gatsby computational Neuroscience Unit. University College London, London, UK.

Kimura, H., & Kobayashi, S. (1997). Reinforcement learning for locomotion of a two-linked robot arm. In *Proceedings of the Europian workshop on learning robots* (pp. 144–153).

Kimura, H., & Kobayashi, S. (1998). Reinforcement learning for continuous action using stochastic gradient ascent. In *Proceedings of the international conference on intelligent autonomous systems (IAS)*: *Vol. 5* (pp. 288–295).

Kleinman, N., Spall, J., & Naiman, D. (1999). Simulation-based optimization with stochastic approximation using common random numbers. *Management Science*, *45*, 1570–1578.

Kohl, N., & Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 2619–2624).

Konda, V., & Tsitsiklis, J. (2000). Actor-critic algorithms. *Advances in Neural Information Processing Systems*, 12.

Lawrence, G., Cowan, N., & Russell, S. (2003). Efficient gradient estimation for motor control learning. In *Proceedings of the international conference on uncertainty in artificial intelligence* (pp. 354–361).

Mitsunaga, N., Smith, C., Kanda, T., Ishiguro, H., & Hagita, N. (2005). Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 1594–1601).

Miyamoto, H., Gandolfo, F., Gomi, H., Schaal, S., Koike, Y., & Osu, R. et al. (1995). A kendama learning robot based on a dynamic optimization theory. In *Proceedings of the IEEE international workshop on robot and human communication* (pp. 327–332).

Miyamoto, H., Gandolfo, F., Gomi, H., Schaal, S., Koike, Y., & Rieka, O. et al. (1996). A kendama learning robot based on a dynamic optimization principle. In *Proceedings of the international conference on neural information processing* (pp. 938–942).

Moon, T., & Stirling, W. (2000). *Mathematical methods and algorithms for signal processing*. Upper Saddle River, NJ: Prentice Hall.

Mori, T., Nakamura, Y., aki Sato, M., & Ishii, S. (2004). Reinforcement learning for cpg-driven biped robot. In *Proceedings of the national conference on artificial intelligence* (pp. 623–630).

Mori, T., Nakamura, Y., & Ishii, S. (2005). Efficient sample reuse by off-policy natural actor-critic learning. In *Advances in neural information processing systems (NIPS '05 workshop presentation)*.

Morimoto, J., & Atkeson, C. A. (2003). Minimax differential dynamic programming: an application to robust biped walking. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in neural information processing systems 15* (pp. 1539–1546). Cambridge, MA: MIT Press.

Nakamura, Y., Mori, T., & Ishii, S. (2004). Natural policy gradient reinforcement learning for a CPG control of a biped robot. In *Proceedings of the international conference on parallel problem solving from nature* (pp. 972–981).

Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., & Kawato, M. (2004). Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, *47*(2–3), 79–91.

Ng, A.Y., & Jordan, M. (2000). PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the international conference on uncertainty in artificial intelligence* (pp. 406–415).

Park, J., Kim, J., & Kang, D. (2005). An RLS-based natural actor-critic algorithm for locomotion of a two-linked robot arm. In Y. Hao, J. Liu, Y. Wang, Y. ming Cheung, H. Yin, L. Jiao, J. Ma, & Y.-C. Jiao (Eds.), *Lecture notes in computer science*: *Vol. 3801*. *Proceedings of the international conference on computational intelligence and security (CIS)* (pp. 65–72). Xi'an, China: Springer.

Peters, J. (2005). Machine learning of motor skills for robotics. *Technical Report CS-05-867*. University of Southern California, Los Angeles, CA.

Peters, J. (2007). Machine learning of motor skills for robotics. *Ph.D. thesis* University of Southern California, Los Angeles, CA, 90089, USA.

Peters, J., & Schaal, S. (2006). Policy gradient methods for robotics. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 2219–2225).

Peters, J., Vijayakumar, S., & Schaal, S. (2003). Reinforcement learning for humanoid robotics. In *Proceedings of the IEEE-RAS international conference on humanoid robots (HUMANOIDS)* (pp. 103–123).

Peters, J., Vijayakumar, S., & Schaal, S. (2005a). Natural actor-critic. In *Proceedings of the European machine learning conference* (pp. 280–291).

Peters, J., Vijayakumar, S., & Schaal, S. (2005b). Natural actor-critic. In *Proceedings of the European conference on machine learning* (pp. 280–291). Springer.

Pongas, D., Billard, A., & Schaal, S. (2005). Rapbid synchronization and accurate phase-locking of rhythmic motor primitives. In *Proceedings of the IEEE international conference on intelligent robots and systems (IROS 2005)*: *Vol. 2005* (pp. 2911–2916).

Richter, S., Aberdeen, D., & Yu, J. (2007). Natural actor-critic for road traffic optimisation. In B. Schoelkopf, J. Platt, & T. Hofmann (Eds.), *Advances in neural information processing systems*: *Vol. 19*. Cambridge, MA: MIT Press, p. Online Preproceedings.

Sadegh, P., & Spall, J. (1997). Optimal random perturbations for stochastic approximation using a simultaneous perturbation gradient approximation. In *Proceedings of the american control conference* (pp. 3582–3586).

Sato, M., Nakamura, Y., & Ishii, S. (2002). Reinforcement learning for biped locomotion. In *Lecture notes in computer science*, *Proceedings of the international conference on artificial neural networks (ICANN)* (pp. 777–782). Springer-Verlag.

Schaal, S. (1997). Learning from demonstration. In M. Mozer, M. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems (NIPS)*: *Vol. 9* (pp. 1040–1046). Cambridge, MA: MIT Press.

Schaal, S., Peters, J., Nakanishi, J., & Ijspeert, A. (2004). Learning movement primitives. In *Springer tracts in advanced robotics*, *International symposium on robotics research (ISRR2003)* (pp. 561–572). Ciena, Italy: Springer.

Sciavicco, L., & Siciliano, B. (2007). *Modeling and control of robot manipulators*. Heidelberg, Germany: MacGraw-Hill.

Spall, J. C. (2003). *Introduction to stochastic search and optimization: Estimation, simulation, and control*. Hoboken, NJ: Wiley.

Su, F., & Gibbs, A. (2002). On choosing and bounding probability metrics. *International Statistical Review*, *70*(3), 419–435.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, & K.-R. Mueller (Eds.), *Advances in neural information processing systems (NIPS)* (pp. 1057–1063). Denver, CO: MIT Press.

Tedrake, R., Zhang, T.W., & Seung, H.S. (2005). Learning to walk in 20 min. In *Proceedings of the Yale workshop on adaptive and learning systems* (pp. 10–22). Yale University, New Haven, New Haven, CT.

Ueno, T., Nakamura, Y., Takuma, T., Shibata, T., Hosoda, K., & Ishii, S. (2006). Fast and stable learning of quasi-passive dynamic walking by an unstable biped robot based on off-policy natural actor-critic. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 5226–5231).

Vachenauer, P., Rade, L., & Westergren, B. (2000). *Springers Mathematische Formeln: Taschenbuch für Ingenieure, Naturwissenschaftler, Informatiker, Wirtschaftswissenschaftler*. Heidelberg, Germany: Springer-Verlag.

Wada, Y., & Kawato, M. (1994). Trajectory formation of arm movement by a neural network with forward and inverse dynamics models. *Systems and Computers in Japan*, *24*, 37–50.

Weaver, L., & Tao, N. (2001a). The optimal reward baseline for gradient-based reinforcement learning. In: *Proceedings of the international conference on uncertainty in artificial intelligence* (pp. 538–545). *Vol. 17*. Seattle, Washington.

Weaver, L., & Tao, N. (2001b). *The variance minimizing constant reward baseline for gradient-based reinforcement learning. Technical Report 30*. Australian National University (ANU).

Werbos, P. (1979). Changes in global policy analysis procedures suggested by new methods of optimization. *Policy Analysis and Information Systems*, *3*(1).

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(23).