Machine Learning (2022) 111:4081–4137
https://doi.org/10.1007/s10994-022-06232-6

Smoothing policies and safe policy gradients

Matteo Papini[1] · **Matteo Pirotta**[2] · **Marcello Restelli**[3]

Received: 18 November 2021 / Revised: 13 May 2022 / Accepted: 9 August 2022 /
Published online: 20 October 2022

Abstract
Policy gradient (PG) algorithms are among the best candidates for the much-anticipated applications of reinforcement learning to real-world control tasks, such as robotics. However, the trial-and-error nature of these methods poses safety issues whenever the learning process itself must be performed on a physical system or involves any form of human-computer interaction. In this paper, we address a specific safety formulation, where both goals and dangers are encoded in a scalar reward signal and the learning agent is constrained to never worsen its performance, measured as the expected sum of rewards. By studying actor-only PG from a stochastic optimization perspective, we establish improvement guarantees for a wide class of parametric policies, generalizing existing results on Gaussian policies. This, together with novel upper bounds on the variance of PG estimators, allows us to identify meta-parameter schedules that guarantee monotonic improvement with high probability. The two key meta-parameters are the step size of the parameter updates and the batch size of the gradient estimates. Through a joint, adaptive selection of these meta-parameters, we obtain a PG algorithm with monotonic improvement guarantees.

Keywords Reinforcement learning · Safe learning · Policy gradient · Monotonic improvement

Editors: Dana Drachsler Cohen, Javier Garcia, Mohammad Ghavamzadeh, Marek Petrik, Philip S. Thomas.

✉ Matteo Papini
matteo.papini@upf.edu

Matteo Pirotta
pirotta@fb.com

Marcello Restelli
marcello.restelli@polimi.it

[1] Universitat Pompeu Fabra, Barcelona, Spain

[2] Meta, Paris, France

[3] Politecnico di Milano, Milan, Italy

1 Introduction

Reinforcement Learning (RL) (Sutton & Barto, 2018) has achieved astounding successes in games (Mnih et al., 2015; Silver et al., 2018; OpenAI, 2018; Vinyals et al., 2019), matching or surpassing human performance in several occasions. However, the much-anticipated applications of RL to real-world tasks such as robotics (Kober et al., 2013), autonomous driving (Okuda et al., 2014) and finance (Li & Hoi, 2014) seem still far. This technological delay may be due to the very nature of RL, which relies on the repeated interaction of the learning machine with the surrounding environment, e.g., a manufacturing plant, a trafficked road, a stock market. The trial-and-error process resulting from this interaction is what makes RL so powerful and general. However, it also poses significant challenges in terms of sample efficiency (Recht, 2019) and safety (Amodei et al., 2016).

In reinforcement learning, the term *safety* can actually refer to a variety of problems (Garca & Fernandez, 2015). The general concern is always the same: avoiding or limiting *damage*. In financial applications, it is typically a loss of money. In robotics and autonomous driving, one should also consider direct damage to people and property. In this work, we do not make assumptions about the nature of the damage, but we assume it is entirely encoded in the scalar reward signal that is presented to the agent in order to evaluate its actions. Other works (e.g., Turchetta et al., 2016) employ a distinct safety signal, separate from rewards.

A further distinction is necessary on the scope of safety constraints with respect to the agent's life. One may simply require the final behavior, the one that is deployed at the end of the learning process, to be safe. This is typically the case when learning is performed in simulation, but the final controller has to be deployed in the real world. The main challenges there are in transferring safety properties from simulation to reality (e.g., Tan et al., 2018). In other cases, learning must be performed, or at least finalized, on the actual system, because no reliable simulator is available (e.g., Peters & Schaal 2008). In such a scenario, safety must be enforced for the whole duration of the learning process. This poses a further challenge, as the agent must necessarily go through a sequence of sub-optimal behaviors before learning its final policy. The problem of learning *while* containing the damage is also known as *safe exploration* (Amodei et al., 2016) and will be the focus of this work.[1]

Garca and Fernandez (2015) provide a comprehensive survey on safe RL, where the existing approaches are organized into two main families: methods that modify the exploration process directly in order to explicitly avoid dangerous actions (e.g., Gehring & Precup 2013), and methods that constrain exploration in a more indirect way by modifying the reward optimization process. The former typically require some sort of external knowledge, such as human demonstrations or advice (e.g., (Abbeel et al., 2010; Clouse & Utgo, 1992)). In this work, we only assume online access to a sufficiently informative reward

[1] We only use "safe exploration" in the general sense of (Amodei et al., 2016). Indeed, in this work, we are not concerned with how exploration should be performed to maximize efficiency, but only with ensuring safety in a context where some form of exploration is necessary. Exploration in RL is a very profound problem with a vast research tradition (Fruit et al., 2019). The problem of efficient exploration under performance-improvement constraints has been studied in the multi-armed-bandit literature under the name of *conservative bandits* (Wu et al., 2016; Kazerouni et al., 2017; Garcelon et al., 2020b), with recent extensions to finite MDPs (Garcelon et al., 2020). In RL, safe exploration is mainly concerned with avoiding unsafe states during the learning process (Hans et al., 2008; Pecka & Svoboda, 2014; Dalal et al., 2018; Turchetta et al., 2016; Berkenkamp, 2019).

signal and prior knowledge of some worst-case constants that are easy to obtain. Optimization-based methods (those belonging to the second class) are more suited for this scenario. A particular kind, identified by García and Fernández as *constrained criteria* (Moldovan & Abbeel, 2012; Castro et al., 2012; Kadota et al., 2006), enforces safety by introducing constraints in the optimization problem, i.e., reward maximization.[2]

A typical constraint is that the agent's performance, i.e., the sum of rewards, must never be less than a user-specified threshold (Geibel & Wysotzki, 2005; Thomas et al., 2015), which may be the average performance of a trusted *baseline policy*. Under the assumption that the reward signal also encodes danger, low performances can be matched with dangerous behaviors, so that the performance threshold works as a safety threshold. This falls into the general framework of *Seldonian machine learning* introduced by Thomas et al. (2019).

If we only cared about the safety of the final controller, the traditional RL objective—maximizing cumulated reward—would be enough. However, most RL algorithms are known to yield oscillating performances *during* the learning phase. Regardless of the final solution, the intermediate ones may violate the threshold, hence yield unsafe behavior. This problem is known as *policy oscillation* (Bertsekas, 2011; Wagner, 2011).

A similar constraint, which confronts the policy oscillation problem even more directly, is *Monotonic Improvment* (MI, S. Kakade & Langford, 2002; Pirotta et al., 2013), and is the one adopted in this work. The requirement is that each new policy implemented by the agent during the learning process does not perform worse than the previous one. In this way, if the initial policy is safe, so will be all the subsequent ones.

The way safety constraints such as MI can be imposed on the optimization process depends, of course, on what kind of policies are considered as candidates and on how the optimization itself is performed. These two aspects are often tied and will depend on the specific kind of RL algorithm that is employed. *Policy Search* or *Optimization* (PO, Deisenroth et al., 2013) is a family of RL algorithms where the class of candidate policies is fixed in advance and a direct search for the best one within the class is performed. This makes PO algorithms radically different from value-based algorithms such as Deep Q-Networks (Mnih et al., 2015), where the optimal policy is a byproduct of a learned value function. Although value-based methods gained great popularity from their successes in games, PO algorithms are better suited for real-world tasks, especially the ones involving cyber-physical systems. The main reasons are the ability of PO methods to deal with high-dimensional continuous state and action spaces, convergence guarantees (Sutton et al., 2000), robustness to sensor noise, and the superior control on the set of feasible policies. The latter allows introducing domain knowledge into the optimization process, possibly including some safety constraints.

In this work, we focus on Policy Gradient methods (PG, Peters & Schaal, 2008; Sutton et al., 2000), where the set of candidate policies is a class of parametric distributions and the optimization is performed via stochastic gradient ascent on the performance objective as a function of the policy parameters. In particular, we analyze the prototypical PG algorithm, REINFORCE (Williams, 1992) and see how the MI constraints can be imposed by adaptively selecting its meta-parameters during the learning process. To achieve this, we study in more depth the stochastic gradient-based optimization process that is at the core

[2] Notably, the approach proposed by Chow et al. (2018) lays between the two classes. It relies on the framework of constrained MDPs to guarantee the safety of a behavior policy during training via a set of local, linear constraints defined using an external cost signal. Similar techniques have been used by Berkenkamp et al. (2017) to guarantee the ability to re-enter a "safe region" during exploration.

of all PG methods (Robbins & Monro, 1951). In particular, we identify a general family of parametric policies that makes the optimization objective Lipschitz-smooth (Nesterov, 2013) and allows easy upper-bounding of the related smoothness constant. This family, referred to as *smoothing policies*, includes commonly used policy classes from the PG literature, namely Gaussian and Softmax policies. Using known properties of Lipschitz-smooth functions, we then provide lower bounds on the performance improvement produced by gradient-based updates, as a function of tunable meta-parameters. This, in turn, allows identifying those meta-parameter schedules that guarantee MI with high probability. In previous work, a similar result was achieved only for Gaussian policies (Pirotta et al., 2013; Papini et al., 2017).[3]

The meta-parameters studied here are the *step size* of the policy updates, or learning rate, and the *batch size* of gradient estimates, i.e., the number of trials that are performed within a single policy update. These meta-parameters, already present in the original REINFORCE algorithm, are typically selected by hand and fixed for the whole learning process (Duan et al., 2016). Besides guaranteeing monotonic improvement, our proposed method removes the burden of selecting these meta-parameters. This safe, automatic selection within the REINFORCE algorithmic framework yields SPG, our Safe Policy Gradient algorithm.

The paper is organized as follows: in Sect. 2 we introduce the necessary background on Markov decision processes, policy optimization, and smooth functions. In Sect. 3, we introduce smoothing policies and show the useful properties they induce on the policy optimization problem, most importantly a lower bound on the performance improvement yielded by an arbitrary policy parameter update (Theorem 7). In Sect. 4, we exploit these properties to select the step size of REINFORCE in a way that guarantees MI with high probability when the batch size is fixed, then we achieve similar results with an adaptive batch size. In Sect. 5, we design a monotonically improving policy gradient algorithm with adaptive batch size, called safe policy gradient (SPG), and show how the latter can also be adapted to weaker improvement constraints. In Sect. 6, we offer a detailed comparison of our contributions with the most closely related literature. In Sect. 7 we empirically evaluate SPG on simulated control tasks. Finally, we discuss the limitations of our approach and propose directions for future work in Sect. 8.

2 Preliminaries

In this section, we revise continuous Markov decision processes (MDPs, Puterman, 2014), actor-only Policy Gradient algorithms (PG, Deisenroth et al., 2013), and some general properties of smooth functions.

2.1 Markov decision processes

A Markov decision process (MDP, Puterman, 2014) is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma, \mu \rangle$, comprised of a measurable state space $\mathcal{S}$, a measurable action space $\mathcal{A}$, a Markovian transition kernel $p : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$, where $\Delta_{\mathcal{S}}$ denotes the set of probability distributions over $\mathcal{S}$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a discount factor $\gamma \in (0, 1)$ and an initial-state

[3] See Sect. 6 for a discussion of related approaches, like the popular TRPO (Schulman et al., 2015).

distribution $\mu \in \Delta_{\mathcal{S}}$. We only consider bounded-reward MDPs, and denote with $R \geq \sup_{s\in\mathcal{S},a\in\mathcal{A}} |r(s,a)|$ (a known upper bound on) the maximum absolute reward. This is the only prior knowledge we have on the task. The MDP is used to model the interaction of a rational agent with the environment. We model the agent's behavior with a policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$, a stochastic mapping from states to actions. The initial state is drawn as $s_0 \sim \mu$. For each time step $t = 0, 1, \ldots$, the agent draws an action $a_t \sim \pi(\cdot|s_t)$, conditional on the current state s_t. Then, the agent obtains a reward $r_{t+1} = r(s_t, a_t)$ and the state of the environment transitions to $s_{t+1} \sim p(\cdot|s_t, a_t)$. The goal of the agent is to maximize the expected sum of discounted rewards, or *performance measure*:

$$J(\pi) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 \sim \mu, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)\right]. \tag{1}$$

We focus on continuous MDPs, where states and actions are real vectors: $\mathcal{S} \subseteq \mathbb{R}^{d_{\mathcal{S}}}$ and $\mathcal{A} \subseteq \mathbb{R}^{d_{\mathcal{A}}}$. However, all the results naturally extend to the discrete case by replacing integrals with summations. See (Puterman, 2014; Bertsekas & Shreve, 2004) on matters of measurability and integrability, which just require common technical assumptions. We slightly abuse notation by denoting probability measures (assumed to be absolutely continuous) and density functions with the same symbol.

Given an MDP, the purpose of RL is to find an optimal policy $\pi^* \in \arg\max_\pi J(\pi)$ without knowing the transition kernel p and the reward function r in advance, but only through interaction with the environment. To better characterize this optimization objective, it is convenient to introduce further quantities. We denote with p_π the transition kernel of the Markov Process induced by policy π, i.e., $p_\pi(\cdot|s) := \int_{\mathcal{A}} \pi(a|s)p(\cdot|s,a)\,\mathrm{d}a$. The t-step transition kernel under policy π is defined inductively as follows:

$$\begin{aligned} p_\pi^0(\cdot|s) &= \mathbb{1}\{s = s'\}, \\ p_\pi^1(\cdot|s) &:= p_\pi(\cdot|s), \\ p_\pi^{t+1}(\cdot|s) &:= \int_{\mathcal{S}} p_\pi^t(s'|s) p_\pi(\cdot|s')\,\mathrm{d}s', \end{aligned} \tag{2}$$

for all $s \in \mathcal{S}$ and $t \geq 1$. The t-step transition kernel allows to define the following *conditional state-occupancy* measure:

$$\rho_s^\pi(\cdot) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t p_\pi^t(\cdot|s), \tag{3}$$

measuring the (discounted) probability of visiting a state starting from s and following policy π. The following property of ρ_s^π —a variant of the *generalized eigenfunction property* by Ciosek and Whiteson (2020, Lemma20)—will be useful (proof in "Appendix A.1"):

Proposition 1 *Let π be any policy and f be any integrable function on $\mathcal{S}$ satisfying the following recursive equation*:

$$f(s) = g(s) + \gamma \int_{\mathcal{S}} p_\pi(s'|s) f(s')\,\mathrm{d}s',$$

for all $s \in \mathcal{S}$ and some integrable function g on $\mathcal{S}$. Then:

$$f(s) = \frac{1}{1-\gamma} \int_{\mathcal{S}} \rho_s^{\pi}(s') g(s') \, \mathrm{d}s',$$

for all $s \in \mathcal{S}$.

The state-value function $V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} r(S_t, A_t) | S_0 = s\right]$ is the discounted sum of rewards obtained, in expectation, by following policy π from state s, and satisfies Bellman's equation (Puterman, 2014):

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{a \sim \pi(\cdot|s)} \left[r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim p(\cdot|s,a)} \left[V^{\pi}(s') \right] \right], \tag{4}$$

Similarly, the action-value function:

$$Q^{\pi}(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim p(\cdot|s,a)} \left[V^{\pi}(s') \right], \tag{5}$$

is the discounted sum of rewards obtained, in expectation, by taking action a in state s and following π afterwards.

The two value functions are closely related:

$$V^{\pi}(s) = \int_{\mathcal{A}} \pi(a|s) Q^{\pi}(s,a) \, \mathrm{d}a, \tag{6}$$

$$Q^{\pi}(s,a) = r(s,a) + \gamma \int_{\mathcal{S}} p(s'|s,a) V_{\pi}(s') \, \mathrm{d}s'. \tag{7}$$

For bounded-reward MDPs, the value functions are bounded for every policy π:

$$\|V^{\pi}\|_{\infty} \leq \|Q^{\pi}\|_{\infty} \leq \frac{R}{1-\gamma}, \tag{8}$$

where $\|V^{\pi}\|_{\infty} = \sup_{s \in \mathcal{S}} |V^{\pi}(s)|$ and $\|Q^{\pi}\|_{\infty} = \sup_{s \in \mathcal{S}, a \in \mathcal{A}} |Q^{\pi}(s,a)|$. Using the definition of state-value function we can rewrite the performance measure as follows:

$$J(\pi) = \int_{\mathcal{S}} \mu(s) V^{\pi}(s) \, \mathrm{d}s = \frac{1}{1-\gamma} \int_{\mathcal{S}} \rho^{\pi}(s) \int_{\mathcal{A}} \pi(a|s) r(s,a) \, \mathrm{d}a \, \mathrm{d}s, \tag{9}$$

where:

$$\rho^{\pi}(\cdot) = \int_{\mathcal{S}} \mu(s) \rho_s^{\pi}(\cdot) \, \mathrm{d}s, \tag{10}$$

is the state-occupancy probability under the starting-state distribution μ.

2.2 Parametric policies

In this work, we only consider parametric policies. Given a d-dimensional parameter vector $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$, a parametric policy is a stochastic mapping from states to actions parametrized by $\boldsymbol{\theta}$, denoted with $\pi_{\boldsymbol{\theta}}$. The search for the optimal policy is thus limited to the policy class $\Pi_{\Theta} = \{\pi_{\boldsymbol{\theta}} \mid \boldsymbol{\theta} \in \Theta\}$. This corresponds to finding an optimal parameter, i.e., $\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta} \in \Theta} J(\pi_{\boldsymbol{\theta}})$. For ease of notation, we often write $\boldsymbol{\theta}$ in place of $\pi_{\boldsymbol{\theta}}$ in function

arguments and superscripts, e.g., $J(\boldsymbol{\theta})$, $\rho^{\boldsymbol{\theta}}(s)$ and $V^{\boldsymbol{\theta}}(s)$ in place of $J(\pi_{\boldsymbol{\theta}})$, $\rho^{\pi_{\boldsymbol{\theta}}}$ and $V^{\pi_{\boldsymbol{\theta}}}(s)$, respectively.[4] We restrict our attention to policies that are twice differentiable w.r.t. $\boldsymbol{\theta}$, for which the gradient $\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(a|s)$ and the Hessian $\nabla^2_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(a|s)$ are defined everywhere and finite. For ease of notation, we omit the $\boldsymbol{\theta}$ subscript in $\nabla_{\boldsymbol{\theta}}$ when clear from the context. Given any twice-differentiable scalar function $f : \Theta \to \mathbb{R}$, we denote with $D_i f$ the i-th gradient component, i.e., $\frac{\partial f}{\partial \theta_i}$, and with $D_{ij} f$ the Hessian element of coordinates (i, j), i.e., $\frac{\partial^2 f}{\partial \theta_i \partial \theta_j}$. We also write $\nabla f(\boldsymbol{\theta})$ to denote $\nabla_{\widetilde{\boldsymbol{\theta}}} f(\widetilde{\boldsymbol{\theta}})\Big|_{\widetilde{\boldsymbol{\theta}}=\boldsymbol{\theta}}$ when this does not introduce any ambiguity.

The Policy Gradient Theorem (Sutton et al., 2000; Konda & Tsitsiklis, 1999) allows us to characterize the gradient of the performance measure $J(\boldsymbol{\theta})$ as an expectation over states and actions visited under $\pi_{\boldsymbol{\theta}}$:[5]

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{1-\gamma} \int_{\mathcal{S}} \rho^{\boldsymbol{\theta}}(s) \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) \nabla \log \pi_{\boldsymbol{\theta}}(a|s) Q^{\boldsymbol{\theta}}(s,a) \, \mathrm{d}a \, \mathrm{d}s. \tag{11}$$

The gradient of the log-likelihood $\nabla \log \pi_{\boldsymbol{\theta}}(\cdot|s)$ is called *score function*, while the Hessian of the log-likelihood $\nabla^2 \log \pi_{\boldsymbol{\theta}}(\cdot|s)$ is sometimes called *observed information*.

2.3 Actor-only policy gradient

In practice, we always consider finite episodes of length T. We call this the *effective horizon* of the MDP, chosen to be sufficiently large so that the problem does not lose generality.[6] We denote with $\tau := (s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1})$ a *trajectory*, i.e., a sequence of states and actions of length T such that $s_0 \sim \mu$, $a_t \sim \pi(\cdot|s_t)$, $s_t \sim p(\cdot|s_{t-1}, a_{t-1})$ for $t = 0, \ldots, T-1$ and some policy π. In this context, the performance measure of a parametric policy $\pi_{\boldsymbol{\theta}}$ can be defined as:

$$J(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau \sim p_{\boldsymbol{\theta}}} \left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) \right], \tag{12}$$

where $p_{\boldsymbol{\theta}}(\tau)$ is the probability density of the trajectory τ that can be generated by following policy $\pi_{\boldsymbol{\theta}}$, i.e., $p_{\boldsymbol{\theta}}(\tau) = \mu(s_0)\pi_{\boldsymbol{\theta}}(a_0|s_0)p(s_1|s_0, a_0) \ldots \pi_{\boldsymbol{\theta}}(a_{T-1}|s_{T-1})$. Let $\mathcal{D} \sim p_{\boldsymbol{\theta}}$ be a batch $\{\tau_1, \tau_2, \ldots, \tau_N\}$ of N trajectories generated with $\pi_{\boldsymbol{\theta}}$, i.e., $\tau_i \sim p_{\boldsymbol{\theta}}$ i.i.d. for $i = 1, \ldots, N$. Let $\widehat{\nabla} J(\boldsymbol{\theta}; \mathcal{D})$ be an estimate of the policy gradient $\nabla J(\boldsymbol{\theta})$ based on $\mathcal{D}$. Such an estimate can be used to perform stochastic gradient ascent on the performance objective $J(\boldsymbol{\theta})$:

[4] Note that $J : \Theta \to \mathbb{R}$, as a function of policy parameters, may have a different geometry than $J : \Pi_\Theta \to \mathbb{R}$, as a function of the policy. In particular, policy parametrization can be an additional source of non-convexity.

[5] As observed by Nota and Thomas (2020), it is important that the state-occupancy measure is discounted as in (3) for the Policy Gradient Theorem to hold. An intuitive way to see the discounted occupancy $\rho^{\pi}(s)$ is as the probability of visiting state s in an indefinite-horizon undiscounted MDP that is reset to the initial state distribution with probability $1 - \gamma$ at each step.

[6] We consider infinite-horizon discounted MDPs in our theoretical analysis, but consider a finite horizon when introducing specific policy gradient estimators. This mismatch is justified by the following result: when the reward is uniformly bounded by R, by setting $T = O(\log(R/\epsilon)/(1-\gamma))$, the discounted truncated sum of rewards is ϵ-close to the infinite sum (see, e.g., (Kakade, 2003), Sec. 2.3.3). See "Appendix C.2" for a way to remove this bias by randomizing the horizon.

$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} + \alpha \widehat{\nabla} J(\boldsymbol{\theta}; \mathcal{D}), \tag{13}$$

where $\alpha \geq 0$ is a *step size* and $N = |\mathcal{D}|$ is called *batch size*. This yields an *Actor-only Policy Gradient* method, summarized in Algorithm 1.

Algorithm 1 Actor-only policy gradient

1: **Input:** initial policy parameters $\boldsymbol{\theta}_0$, step size α, batch size N, number of iterations K
2: **for** $k = 0, \ldots, K-1$ **do**
3: Collect N trajectories with $\boldsymbol{\theta}_k$ to obtain dataset $\mathcal{D}_k$
4: Compute policy gradient estimate $\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)$
5: Update policy parameters as $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha \widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)$
6: **end for**

Under mild conditions, this algorithm is guaranteed to converge to a local optimum (Sutton et al., 2000). This is reasonable since the objective $J(\boldsymbol{\theta})$ is non-convex in general.[7] As for the gradient estimator, we can use REINFORCE (Williams, 1992; Glynn, 1986):[8]

$$\widehat{\nabla} J(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T-1} \gamma^t r(a_t^i, s_t^i) - b \right) \left(\sum_{t=0}^{T-1} \nabla \log \pi_{\boldsymbol{\theta}}(a_t^i | s_t^i) \right), \tag{14}$$

or its refinement, G(PO)MDP (Baxter & Bartlett, 2001), which typically suffers from less variance (Peters & Schaal, 2008):

$$\widehat{\nabla} J(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left[\left(\gamma^t r(a_t^i, s_t^i) - b_t \right) \sum_{h=0}^{t} \nabla \log \pi_{\boldsymbol{\theta}}(a_h^i | s_h^i) \right], \tag{15}$$

where the superscript on states and actions denotes the i-th trajectory of the dataset and b is a (possibly time-dependent and vector-valued) control variate, or *baseline*. Both estimators are unbiased for any action-independent baseline.[9] Peters and Schaal (2008) prove that Algorithm 1 with the G(PO)MDP estimator is equivalent to Monte-Carlo PGT (Policy Gradient Theorem, (Sutton et al., 2000)), and provide variance-minimizing baselines for both REINFORCE and G(PO)MDP.

Algorithm 1 is called *actor-only* to discriminate it from *actor-critic* policy gradient algorithms (Konda & Tsitsiklis, 1999), where an approximate value function, or *critic*, is employed in the gradient computation. In this work, we will focus on actor-only algorithms, for which safety guarantees are more easily proven.[10] Generalizations of Algorithm 1 include reducing the variance of gradient estimates through baselines and other stochastic-optimization techniques (e.g., (Papini et al., 2018; Shen et al., 2019; Xu et al.,

[7] Recent works show that policy gradient algorithms can converge to *globally* optimal policies in some interesting special cases (Bhandari & Russo, 2019; Zhang et al., 2020; Agarwal et al., 2020).

[8] In the literature, the term REINFORCE is often used to denote actor-only policy gradient methods in general. In this paper, REINFORCE refers to the algorithm by Williams (1992), which also applies to more general stochastic optimization problems.

[9] Also valid action-dependent baselines have been proposed. See (Tucker et al., 2018) for a discussion.

[10] The distinction is not so sharp, as a critic can be seen as a baseline and vice-versa. We call *critic* an *explicit* value function estimate used in policy gradient estimation.

2020)) using a vector step size (Yu et al., 2006; Papini et al., 2017); making the step size *adaptive*, i.e., iteration and/or data-dependent (Pirotta et al., 2013); making the batch size N also adaptive (Papini et al., 2017); applying a preconditioning matrix to the gradient, as in Natural Policy Gradient (Kakade, 2002) and second-order methods (Furmston & Barber, 2012).

2.4 Smooth functions

In the following we denote with $\|\boldsymbol{x}\|_p$ the ℓ_p-norm of vector $\boldsymbol{x}$, which is the Euclidean norm for $p = 2$. . For a matrix A, $\|A\|_p = \sup\{\|Ax\|_p : \|x\|_p = 1\}$ denotes the induced norm, which is the spectral norm for $p = 2$. When the p subscript is omitted, we always mean $p = 2$.

Let $g : \mathcal{X} \subseteq \mathbb{R}^d \to \mathbb{R}^n$ be a (non-convex) vector-valued function. We call g *Lipschitz continuous* if there exists $L > 0$ such that, for every $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$:

$$\left\|g(\boldsymbol{x}') - g(\boldsymbol{x})\right\| \leq L\left\|\boldsymbol{x}' - \boldsymbol{x}\right\|. \tag{16}$$

Let $f : \mathcal{X} \subseteq \mathbb{R}^d \to \mathbb{R}$ be a real-valued differentiable function. We call f *Lipschitz smooth* if its gradient is Lipschitz continuous, i.e., there exists $L > 0$ such that, for every $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$:

$$\left\|\nabla f(\boldsymbol{x}') - \nabla f(\boldsymbol{x})\right\| \leq L\left\|\boldsymbol{x}' - \boldsymbol{x}\right\|. \tag{17}$$

Whenever we want to specify the Lipschitz constant L of the gradient, we call f L-smooth.[11] We also call L the *smoothness constant* of f. For a twice-differentiable function, the following holds:[12]

Proposition 2 *Let $\mathcal{X}$ be a convex subset of $\mathbb{R}^d$ and $f : \mathcal{X} \to \mathbb{R}$ be a twice-differentiable function. If the Hessian is uniformly bounded in spectral norm by $L > 0$, i.e., $\sup_{\boldsymbol{x} \in \mathcal{X}} \left\|\nabla^2 f(\boldsymbol{x})\right\|_2 \leq L$, then f is L-smooth.*

Lipschitz smooth functions admit a quadratic bound on the deviation from linear behavior:

Proposition 3 (Quadratic Bound) *Let $\mathcal{X}$ be a convex subset of $\mathbb{R}^d$ and $f : \mathcal{X} \to \mathbb{R}$ be an L-smooth function. Then, for every $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$:*

$$\left|f(\boldsymbol{x}') - f(\boldsymbol{x}) - \left\langle \boldsymbol{x}' - \boldsymbol{x}, \nabla f(\boldsymbol{x}) \right\rangle\right| \leq \frac{L}{2}\left\|\boldsymbol{x}' - \boldsymbol{x}\right\|^2, \tag{18}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product.

This bound is often useful for optimization purposes (Nesterov, 2013).

[11] The Lipschitz constant is usually defined as the *smallest* constant satisfying the Lipschitz condition. In this paper, we accept *any* constant for which the Lipschitz condition holds.

[12] The results from this section are well known in the optimization literature (Nesterov, 2013). However, proofs of Lemma 2 and 3 are reported in "Appendix A.2" for the sake of completeness.

Table 1 Smoothing constants ξ_1, ξ_2, ξ_3 and smoothness constant L for Gaussian and Softmax policies, where M is an upper bound on the Euclidean norm of the feature function, R is the maximum absolute-value reward, γ is the discount factor, σ is the standard deviation of the Gaussian policy and τ is the temperature of the Softmax policy. We also report the improved smoothness constant by Yuan et al. (2021) as $L^\star$

	Gaussian	Softmax
ξ_1	$\frac{2M}{\sqrt{2\pi}\sigma}$	$\frac{2M}{\tau}$
ξ_2	$\frac{M^2}{\sigma^2}$	$\frac{4M^2}{\tau^2}$
ξ_3	$\frac{M^2}{\sigma^2}$	$\frac{2M^2}{\tau^2}$
L	$\frac{2M^2R}{\sigma^2(1-\gamma)^2}\left(1+\frac{2\gamma}{\pi(1-\gamma)}\right)$	$\frac{2M^2R}{\tau^2(1-\gamma)^2}\left(3+\frac{4\gamma}{1-\gamma}\right)$
$L^\star$	$\frac{2M^2R}{\sigma^2(1-\gamma)^2}$	$\frac{6M^2R}{\tau^2(1-\gamma)^2}$

3 Smooth policy gradient

In this section, we provide lower bounds on performance improvement based on general assumptions on the policy class.

3.1 Smoothing policies

We introduce a family of parametric stochastic policies having properties that we deem desirable for policy-gradient learning. We call them *smoothing*, as they are characterized by the smoothness of the performance measure:

Definition 1 Let $\Pi_\Theta = \{\pi_\theta \mid \theta \in \Theta\}$ be a class of twice-differentiable parametric stochastic policies, where $\Theta \subset \mathbb{R}^d$ is convex. We call it *smoothing* if there exist non-negative constants ξ_1, ξ_2, ξ_3 such that, for every state and in expectation over actions, the Euclidean norm of the score function:

$$\sup_{s\in\mathcal{S}} \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\left\|\nabla \log \pi_\theta(a|s)\right\|\right] \leq \xi_1, \tag{19}$$

the squared Euclidean norm of the score function:

$$\sup_{s\in\mathcal{S}} \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\left\|\nabla \log \pi_\theta(a|s)\right\|^2\right] \leq \xi_2, \tag{20}$$

and the spectral norm of the observed information:

$$\sup_{s\in\mathcal{S}} \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\left\|\nabla^2 \log \pi_\theta(a|s)\right\|\right] \leq \xi_3, \tag{21}$$

are upper-bounded.

Note that the definition requires that the bounding constants ξ_1, ξ_2, ξ_3 be independent of the policy parameters and the state. For this reason, the existence of such constants

depends on the policy parameterization.[13] We call a policy class (ξ_1, ξ_2, ξ_3)-smoothing when we want to specify the bounding constants. In "Appendix B", we show that some of the most commonly used policies, such as the Gaussian policy for continuous actions and the Softmax policy for discrete actions, are smoothing. The smoothing constants for these classes are reported in Table 1. In the following sections, we will exploit the smoothness of the performance measure induced by smoothing policies to develop a monotonically improving policy gradient algorithm. However, smoothing policies have other interesting properties. For instance, variance upper bounds for REINFORCE/G(PO)MDP with Gaussian policies (Zhao et al., 2011; Pirotta et al., 2013) can be generalized to smoothing policies (see "Appendix D" for details). Other nice properties of smoothing policies, such as Lipschitzness of the performance measure, are discussed in Yuan et al. (2021, Lemma D.1)

3.2 Policy Hessian

We now show that the Hessian of the performance measure $\nabla^2 J(\boldsymbol{\theta})$ for a smoothing policy has bounded spectral norm. We start by writing the policy Hessian for a general parametric policy as follows. The result is well known (Kakade, 2001), but we report a proof in "Appendix A.4" for completeness. Also, note that our smoothing-policy assumption is weaker than the typical one (uniformly bounded policy derivatives). See "Appendix A.3" for details.

Proposition 4 *Let $\pi_{\boldsymbol{\theta}}$ be a smoothing policy. The Hessian of the performance measure is*:

$$\nabla^2 J(\boldsymbol{\theta}) = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s \sim \rho^{\boldsymbol{\theta}} \\ a \sim \pi_{\boldsymbol{\theta}}(\cdot|s)}} \Big[\nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top Q^{\boldsymbol{\theta}}(s,a) + \nabla Q^{\boldsymbol{\theta}}(s,a) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s) + \big(\nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s) + \nabla^2 \log \pi_{\boldsymbol{\theta}}(a|s) \big) Q^{\boldsymbol{\theta}}(s,a) \Big].$$

For smoothing policies, we can bound the policy Hessian in terms of the constants from Definition 1:

Lemma 5 *Given a (ξ_1, ξ_2, ξ_3)-smoothing policy $\pi_{\boldsymbol{\theta}}$, the spectral norm of the policy Hessian can be upper-bounded as follows*:

$$\left\| \nabla^2 J(\boldsymbol{\theta}) \right\| \leq \frac{R}{(1-\gamma)^2} \left(\frac{2\gamma \xi_1^2}{1-\gamma} + \xi_2 + \xi_3 \right).$$

Proof By the Policy Gradient Theorem (see the proof of Theorem 1 by, (Sutton et al., 2000)):

$$\nabla V^{\boldsymbol{\theta}}(s) = \frac{1}{1-\gamma} \int_{\mathcal{S}} \rho_s^{\boldsymbol{\theta}}(s') \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s') \nabla \log \pi_{\boldsymbol{\theta}}(a|s') Q^{\boldsymbol{\theta}}(s,a) \, \mathrm{d}a \, \mathrm{d}s'. \tag{22}$$

[13] Notice that, by Jensen's inequality, one can always remove the first requirement (19) by letting $\xi_1 = \sqrt{\xi_2}$, as observed by Yuan et al. (2021). However, a smaller value of ξ_1 can sometimes be obtained. See Lemma 23 for an example.

Using (22), we bound the gradient of the value function in Euclidean norm:

$$\begin{aligned}\left\|\nabla V^{\theta}(s)\right\| &\leq \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s' \sim \rho_s^{\theta} \\ a \sim \pi_{\theta}(\cdot|s')}} \left[\left\|\nabla \log \pi_{\theta}(a|s') Q^{\theta}(s', a)\right\|\right] \\ &\leq \frac{R}{(1-\gamma)^2} \mathop{\mathbb{E}}_{\substack{s' \sim \rho_s^{\theta} \\ a \sim \pi_{\theta}(\cdot|s')}} \left[\left\|\nabla \log \pi_{\theta}(a|s')\right\|\right] \end{aligned} \tag{23}$$

$$\begin{aligned} &\leq \frac{R}{(1-\gamma)^2} \sup_{s' \in \mathcal{S}} \mathop{\mathbb{E}}_{a \sim \pi_{\theta}(\cdot|s')} \left[\left\|\nabla \log \pi_{\theta}(a|s')\right\|\right] \\ &\leq \frac{\xi_1 R}{(1-\gamma)^2}, \end{aligned} \tag{24}$$

where (23) is from the Cauchy-Schwarz inequality and (8), and (24) is from the smoothing-policy assumption. Next, we bound the gradient of the action-value function. From (7):

$$\left\|\nabla Q^{\theta}(s,a)\right\| = \left\|\nabla\left(r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim p(\cdot|s,a)}\left[V^{\theta}(s')\right]\right)\right\| \tag{25}$$

$$= \gamma \left\| \mathop{\mathbb{E}}_{s' \sim p(\cdot|s,a)}\left[\nabla V^{\theta}(s')\right]\right\| \tag{26}$$

$$\leq \gamma \mathop{\mathbb{E}}_{s' \sim p(\cdot|s,a)}\left[\left\|\nabla V^{\theta}(s)\right\|\right] \leq \frac{\gamma \xi_1 R}{(1-\gamma)^2}, \tag{27}$$

where the interchange of gradient and expectation in (26) is justified by the smoothing-policy assumption (see "Appendix A.3" for details) and (27) is from (24). Finally, from Proposition 4:

$$\begin{aligned}(1-\gamma)\left\|\nabla^2 J(\boldsymbol{\theta})\right\| \leq & \mathop{\mathbb{E}}_{\substack{s \sim \rho^{\theta} \\ a \sim \pi_{\theta}(\cdot|s)}} \left[\left\|\nabla \log \pi_{\theta}(a|s) \nabla^{\top} Q^{\theta}(s,a)\right\|\right] \\ &+ \mathop{\mathbb{E}}_{\substack{s \sim \rho^{\theta} \\ a \sim \pi_{\theta}(\cdot|s)}} \left[\left\|\nabla Q^{\theta}(s,a) \nabla^{\top} \log \pi_{\theta}(a|s)\right\|\right] \\ &+ \mathop{\mathbb{E}}_{\substack{s \sim \rho^{\theta} \\ a \sim \pi_{\theta}(\cdot|s)}} \left[\left\|\nabla \log \pi_{\theta}(a|s) \nabla^{\top} \log \pi_{\theta}(a|s) Q^{\theta}(s,a)\right\|\right] \\ &+ \mathop{\mathbb{E}}_{\substack{s \sim \rho^{\theta} \\ a \sim \pi_{\theta}(\cdot|s)}} \left[\left\|\nabla^2 \log \pi_{\theta}(a|s) Q^{\theta}(s,a)\right\|\right] \end{aligned} \tag{28}$$

$$
\begin{aligned}
&\leq 2 \mathop{\mathbb{E}}_{\substack{s \sim \rho^\theta \\ a \sim \pi_\theta(\cdot|s)}} \left[\|\nabla \log \pi_\theta(a|s)\| \left\|\nabla Q^\theta(s,a)\right\| \right] \\
&\quad + \mathop{\mathbb{E}}_{\substack{s \sim \rho^\theta \\ a \sim \pi_\theta(\cdot|s)}} \left[\|\nabla \log \pi_\theta(a|s)\|^2 \left| Q^\theta(s,a) \right| \right] \\
&\quad + \mathop{\mathbb{E}}_{\substack{s \sim \rho^\theta \\ a \sim \pi_\theta(\cdot|s)}} \left[\left\|\nabla^2 \log \pi_\theta(a|s)\right\| \left| Q^\theta(s,a) \right| \right]
\end{aligned} \tag{29}
$$

$$
\begin{aligned}
&\leq \frac{2\gamma\xi_1 R}{(1-\gamma)^2} \mathop{\mathbb{E}}_{\substack{s \sim \rho^\theta \\ a \sim \pi_\theta(\cdot|s)}} \left[\|\nabla \log \pi_\theta(a|s)\| \right] \\
&\quad + \frac{R}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s \sim \rho^\theta \\ a \sim \pi_\theta(\cdot|s)}} \left[\|\nabla \log \pi_\theta(a|s)\|^2 \right] \\
&\quad + \frac{R}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s \sim \rho^\theta \\ a \sim \pi_\theta(\cdot|s)}} \left[\left\|\nabla^2 \log \pi_\theta(a|s)\right\| \right]
\end{aligned} \tag{30}
$$

$$
\leq \frac{R}{(1-\gamma)} \left(\frac{2\gamma\xi_1^2}{1-\gamma} + \xi_2 + \xi_3 \right), \tag{31}
$$

where (28) is from Jensen inequality (all norms are convex) and the triangle inequality, (29) is from $\left\|\boldsymbol{x}\boldsymbol{y}^\top\right\| = \|\boldsymbol{x}\| \|\boldsymbol{y}\|$ for any two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, (30) is from (8) and (27), and the last inequality is from the smoothing-policy assumption. □

3.3 Smooth performance

For a smoothing policy, the performance measure $J(\boldsymbol{\theta})$ is Lipschitz smooth with a smoothness constant that only depends on the smoothing constants, the reward magnitude, and the discount factor. This result is of independent interest as it can be used to establish convergence rates for policy gradient algorithms (Yuan et al., 2021).

Lemma 6 *Given a* (ξ_1, ξ_2, ξ_3)*-smoothing policy class* Π_Θ*, the performance measure* $J(\boldsymbol{\theta})$ *is L-smooth with the following smoothness constant*:

$$
L = \frac{R}{(1-\gamma)^2} \left(\frac{2\gamma\xi_1^2}{1-\gamma} + \xi_2 + \xi_3 \right). \tag{32}
$$

Proof From Lemma 5, L is a bound on the spectral norm of the policy Hessian. From Lemma 2, this is a valid Lipschitz constant for the policy gradient, hence the performance measure is L-smooth. □

The smoothness of the performance measure, in turn, yields the following property on the guaranteed performance improvement:

Theorem 7 *Let Π_Θ be a (ξ_1, ξ_2, ξ_3)-smoothing policy class. For every $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \Theta$:*

$$J(\boldsymbol{\theta}') - J(\boldsymbol{\theta}) \geq \langle \Delta\boldsymbol{\theta}, \nabla J(\boldsymbol{\theta}) \rangle - \frac{L}{2}\|\Delta\boldsymbol{\theta}\|^2,$$

where $\Delta\boldsymbol{\theta} = \boldsymbol{\theta}' - \boldsymbol{\theta}$ and $L = \frac{R}{(1-\gamma)^2}\left(\frac{2\gamma\xi_1^2}{1-\gamma} + \xi_2 + \xi_3\right)$.

Proof It suffices to apply Lemma 3 with the Lipschitz constant from Lemma 6. □

The smoothness constant L for Gaussian and Softmax policies is reported in Table 1.

In the following, we will exploit this property of smoothing policies to enforce safety guarantees on the policy updates performed by Algorithm 1, i.e., stochastic gradient ascent updates. However, Theorem 7 applies to any policy update $\Delta\boldsymbol{\theta} \in \mathbb{R}^d$ as long as $\boldsymbol{\theta} + \Delta\boldsymbol{\theta} \in \Theta$.

Very recently, R. Yuan et al. (2021, Lemma4.4) provided an improved smoothness constant for smoothing policies:

$$L^\star = \frac{R(\xi_2 + \xi_3)}{(1-\gamma)^2}. \tag{33}$$

This is a significant step forward since it improves the dependence on the effective horizon by a $(1-\gamma)^{-1}$ factor. In Table 1 we report explicit expressions for $L^\star$ in the case of linear Gaussian and Softmax policies. We will use these superior smoothness constant in the numerical simulations of Sect. 7.

4 Optimal safe meta-parameters

In this section, we provide a step size for Algorithm 1 that maximizes a lower bound on the performance improvement for smoothing policies. This yields safety in the sense of Monotonic Improvement (MI), i.e., non-negative performance improvements at each policy update:

$$J(\boldsymbol{\theta}_k) - J(\boldsymbol{\theta}_{k+1}) \geq 0, \tag{34}$$

at least with high probability.

In policy optimization, at each learning iteration k, we ideally want to find the policy update $\Delta\boldsymbol{\theta}$ that maximizes the new performance $J(\boldsymbol{\theta}_k + \Delta\boldsymbol{\theta})$, or equivalently:

$$\max_{\Delta\boldsymbol{\theta}} J(\boldsymbol{\theta}_k + \Delta\boldsymbol{\theta}) - J(\boldsymbol{\theta}_k), \tag{35}$$

since $J(\boldsymbol{\theta}_k)$ is fixed. Unfortunately, the performance of the updated policy cannot be known in advance.[14] For this reason, we replace the optimization objective in (35) with a lower

[14] The performance of the updated policy could be estimated with off-policy evaluation techniques, but this would introduce an additional, non-negligible source of variance. The idea of using off-policy evaluation to select meta-parameters was explored by Paul et al. (2019).

bound, i.e., a *guaranteed improvement*. In particular, taking Algorithm 1 as our starting point, we maximize the guaranteed improvement of a policy gradient update (line 5) by selecting optimal meta-parameters. The solution of this meta-optimization problem provides a lower bound on the actual performance improvement. As long as this is always non-negative, MI is guaranteed.

4.1 Adaptive step size: exact framework

To decouple the pure optimization aspects of this problem from gradient estimation issues, we first consider an *exact* policy gradient update, i.e., $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha \nabla J(\boldsymbol{\theta}_k)$, where we assume to have a first-order oracle, i.e., to be able to compute the exact policy gradient $\nabla J(\boldsymbol{\theta}_k)$. This assumption is clearly not realistic, and will be removed in Sect. 4.2. In this simplified framework, performance improvement can be guaranteed *deterministically*. Furthermore, the only relevant meta-parameter is the step size α of the update. We first need a lower bound on the performance improvement $J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k)$. For a smoothing policy, we can use the following:

Theorem 8 *Let Π_Θ be a (ξ_1, ξ_2, ξ_3)-smoothing policy class. Let $\boldsymbol{\theta}_k \in \Theta$ and $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla J(\boldsymbol{\theta}_k)$, where $\alpha > 0$. Provided $\boldsymbol{\theta}_{k+1} \in \Theta$, the performance improvement of $\boldsymbol{\theta}_{k+1}$ w.r.t. $\boldsymbol{\theta}_k$ can be lower bounded as follows*:

$$J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \alpha \left\| \nabla J(\boldsymbol{\theta}_k) \right\|^2 - \alpha^2 \frac{L}{2} \left\| \nabla J(\boldsymbol{\theta}_k) \right\|^2 := B(\alpha; \boldsymbol{\theta}_k),$$

where $L = \frac{R}{(1-\gamma)^2} \left(\frac{2\gamma \xi_1^2}{1-\gamma} + \xi_2 + \xi_3 \right)$.

Proof This is a direct consequence of Theorem 7 with $\Delta\boldsymbol{\theta} = \alpha \nabla J(\boldsymbol{\theta}_k)$. □

This bound is in the typical form of performance improvement bounds (e.g., (Kakade & Langford, 2002; Pirotta et al., 2013; Schulman et al., 2015; Cohen et al., 2018)): a positive term accounting for the anticipated advantage of $\boldsymbol{\theta}_{k+1}$ over $\boldsymbol{\theta}_k$, and a penalty term accounting for the mismatch between the two policies, which makes the anticipated advantage less reliable. In our case, the mismatch is measured by the curvature of the performance measure w.r.t. the policy parameters, via the smoothness constant L. This lower bound is quadratic in α, hence we can easily find the optimal step size α^*.

Corollary 9 *Let $B(\alpha; \boldsymbol{\theta}_k)$ be the guaranteed performance improvement of an exact policy gradient update, as defined in Theorem 8. Under the same assumptions, $B(\alpha; \boldsymbol{\theta}_k)$ is maximized by the constant step size $\alpha^* = \frac{1}{L}$, which guarantees the following non-negative performance improvement*:

$$J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \frac{\left\| \nabla J(\boldsymbol{\theta}_k) \right\|^2}{2L}.$$

Proof We just maximize $B(\alpha; \boldsymbol{\theta}_k)$ as a (quadratic) function of α. The global optimum $B(\alpha^*; \boldsymbol{\theta}_k) = \frac{\left\| \nabla J(\boldsymbol{\theta}_k) \right\|^2}{2L}$ is attained by $\alpha^* = \frac{1}{L}$. The improvement guarantee follows from Theorem 8. □

4.2 Adaptive step size: approximate framework

In practice, we cannot compute the exact gradient $\nabla J(\boldsymbol{\theta}_k)$, but only an estimate $\widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D})$ obtained from a finite dataset $\mathcal{D}$ of trajectories. In this section, N denotes the *fixed* size of $\mathcal{D}$. To find the optimal step size, we just need to adapt the performance-improvement lower bound of Theorem 8 to stochastic-gradient updates. Since sample trajectories are involved, this new lower bound will only hold with high probability. To establish statistical guarantees, we make the following assumption on how the (unbiased) gradient estimate concentrates around its expected value:

Assumption 1 Fixed a parameter $\boldsymbol{\theta} \in \Theta$, a batch size $N \in \mathbb{N}$ and a failure probability $\delta \in (0,1)$, with probability at least $1-\delta$:

$$\left\| \widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D}) - \nabla J(\boldsymbol{\theta}) \right\| \leq \frac{\epsilon(\delta)}{\sqrt{N}},$$

where $|\mathcal{D}|$ is a dataset of N i.i.d. trajectories collected with $\pi_{\boldsymbol{\theta}}$ and $\epsilon : (0,1) \to \mathbb{R}$ is a known function.

We will discuss how this assumption is satisfied in cases of interest in Sect. 5 and "Appendix C". Under the above assumption, we can adapt Theorem 8 to the stochastic-gradient case as follows:

Theorem 10 *Let Π_Θ be a (ξ_1,ξ_2,ξ_3)-smoothing policy class. Let $\boldsymbol{\theta}_k \in \Theta \subseteq \mathbb{R}^d$ and $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)$, where $\alpha \geq 0$, $N = |\mathcal{D}_k| \geq 1$. Under Assumption 1, provided $\boldsymbol{\theta}_{k+1} \in \Theta$, the performance improvement of $\boldsymbol{\theta}_{k+1}$ w.r.t. $\boldsymbol{\theta}_k$ can be lower bounded, with probability at least $1-\delta_k$, as follows*:

$$\begin{aligned} J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \alpha & \left(\left\| \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k) \right\| - \frac{\epsilon(\delta_k)}{\sqrt{N}} \right) \\ & \times \max \left\{ \left\| \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k) \right\|, \frac{\left\| \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k) \right\| + \frac{\epsilon(\delta_k)}{\sqrt{N}}}{2} \right\} \\ & - \frac{\alpha^2 L}{2} \left\| \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k) \right\|^2 := \widetilde{B}_k(\alpha;N), \end{aligned}$$

where $L = \frac{R}{(1-\gamma)^2}\left(\frac{2\gamma\xi_1^2}{1-\gamma} + \xi_2 + \xi_3 \right)$.

Proof Consider the good event $E_k = \left\{ \left\| \widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D}) - \nabla J(\boldsymbol{\theta}) \right\| \leq \epsilon(\delta_k)/\sqrt{N} \right\}$. By Assumption 1, E_k holds with probability at least $1-\delta_k$. For the rest of the proof, we will assume E_k holds.

Let $\epsilon_k := \epsilon(\delta_k)/\sqrt{N}$ for short. Under E_k, by the triangular inequality:

$$\begin{aligned} \left\| \nabla J(\boldsymbol{\theta}_k) \right\| &\geq \left\| \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k) \right\| - \left\| \nabla J(\boldsymbol{\theta}_k) - \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k) \right\| \\ &\geq \left\| \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k) \right\| - \epsilon_k, \end{aligned} \tag{36}$$

thus:

$$\|\nabla J(\boldsymbol{\theta}_k)\|^2 \geq \max\left\{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\| - \epsilon_k, 0\right\}^2. \tag{37}$$

Then, by the polarization identity:

$$\begin{aligned}\left\langle \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k), \nabla J(\boldsymbol{\theta}_k)\right\rangle &= \frac{1}{2}\left(\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2 + \|\nabla J(\boldsymbol{\theta}_k)\|^2 \right.\\ &\qquad \left. -\left\|\nabla J(\boldsymbol{\theta}_k) - \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2\right)\\ &\geq \frac{1}{2}\left(\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2 + \max\left\{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\| - \epsilon_k, 0\right\}^2 - \epsilon_k^2\right),\end{aligned}$$

where the latter inequality is from (37). We first consider the case in which $\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\| > \epsilon_k$:

$$\begin{aligned}\left\langle \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k), \nabla J(\boldsymbol{\theta}_k)\right\rangle &\geq \frac{1}{2}\left(\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2 + \left(\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\| - \epsilon_k\right)^2 - \epsilon_k^2\right)\\ &= \left(\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\| - \epsilon_k\right)\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|.\end{aligned} \tag{38}$$

Then, we consider the case in which $\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\| \leq \epsilon_k$:

$$\left\langle \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k), \nabla J(\boldsymbol{\theta}_k)\right\rangle \geq \frac{1}{2}\left(\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2 - \epsilon_k^2\right) \tag{39}$$

$$= \left(\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\| - \epsilon_k\right)\frac{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\| + \epsilon_k}{2}. \tag{40}$$

The two cases can be unified as follows:

$$\begin{aligned}\left\langle \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k), \nabla J(\boldsymbol{\theta}_k)\right\rangle &\geq \left(\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\| - \epsilon_k\right)\\ &\qquad \times \max\left\{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|, \frac{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\| + \epsilon_k}{2}\right\}.\end{aligned} \tag{41}$$

From Theorem 7 with $\Delta\boldsymbol{\theta} = \alpha\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)$ we obtain:

$$
\begin{aligned}
J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) &\geq \left\langle \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k, \nabla J(\boldsymbol{\theta}_k) \right\rangle - \frac{L}{2}\left\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\right\|^2 \\
&= \alpha \left\langle \widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k), \nabla J(\boldsymbol{\theta}_k) \right\rangle - \frac{\alpha^2 L}{2}\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|^2 \\
&\geq \alpha \left(\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\| - \epsilon_k \right) \\
&\quad \times \max \left\{ \left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|, \frac{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\| + \epsilon_k}{2} \right\} \\
&\quad - \frac{\alpha^2 L}{2}\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|^2,
\end{aligned}
\tag{42}
$$

where the last inequality is from (41). □

From Theorem 10 we can easily obtain an optimal step size, as done in the exact setting, provided the batch size is sufficiently large:

Corollary 11 *Let $\widetilde{B}(\alpha, N; \boldsymbol{\theta}_k)$ be the guaranteed performance improvement of a stochastic policy gradient update, as defined in Theorem 10. Under the same assumptions, provided the batch size satisfies*:

$$
N \geq \frac{\epsilon^2(\delta_k)}{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|^2},
\tag{43}
$$

$\widetilde{B}(\alpha, N; \boldsymbol{\theta}_k)$ is maximized by the following adaptive step size:

$$
\alpha_k^* = \frac{1}{L}\left(1 - \frac{\epsilon(\delta_k)}{\sqrt{N}\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|}\right),
\tag{44}
$$

which guarantees, with probability at least $1 - \delta_k$, the following non-negative performance improvement:

$$
J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \frac{\left(\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\| - \frac{\epsilon(\delta_k)}{\sqrt{N}}\right)^2}{2L}.
\tag{45}
$$

Proof Let $N_0 = \epsilon^2(\delta_k)\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|^{-2}$. When $N \leq N_0$, the second argument of the max operator in (41) is selected. In this case, no positive improvement can be guaranteed and the optimal non-negative step size is $\alpha = 0$. Thus, we focus on the case $N > N_0$. In this case, the first argument of the max operator is selected. Optimizing $\widetilde{B}(\alpha, N)$ as a function of α alone, which is again quadratic, yields (44) as the optimal step size and (45) as the maximum guaranteed improvement. □

In this case, the optimal step size is adaptive, i.e., time-varying and data-dependent. The constant, optimal step size for the exact case (Corollary 9) is recovered in the limit of infinite data, i.e., $N \to \infty$. In the following we discuss why this adaptive step size should not be used in practice, and propose an alternative solution.

4.3 Adaptive Batch Size

The safe step size from Corollary 11 requires the batch size to be large enough. As soon as the condition (43) fails to hold, the user is left with the decision whether to interrupt the learning process or collect more data — an undesirable property for a fully autonomous system. To avoid this, a large batch size must be selected *from the start*, which results in a pointless waste of data in the early learning iterations. Even so, Eq. (43), used as a stopping condition, would be susceptible to random oscillations of the stochastic gradient magnitude, interrupting the learning process prematurely.

As observed in (Papini et al., 2017), controlling also the batch size N of the gradient estimation can be advantageous. Intuitively, a larger batch size yields a more reliable estimate, which in turn allows a safer policy gradient update. The larger the batch size, the higher the guaranteed improvement, which would lead to selecting the highest possible value of N. However, we must take into account the cost of collecting the trajectories, which is non-negligible in real-world problems (e.g., robotics). For this reason, we would like the meta-parameters to maximize the *per-trajectory* performance improvement:

$$\alpha_k, N_k = \arg\max_{\alpha,N} \frac{J(\boldsymbol{\theta}_k + \alpha \widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D})) - J(\boldsymbol{\theta}_k)}{N}, \tag{46}$$

where $\mathcal{D}$ is a dataset of N i.i.d. trajectories sampled with $\pi_{\boldsymbol{\theta}_k}$. We can then use the lower bound from Theorem 10 to find the jointly optimal safe step size and batch size, similarly to what was done in (Papini et al., 2017) for the special case of Gaussian policies:

Corollary 12 *Let $\widetilde{B}_k(\alpha;N)$ be the lower bound on the performance improvement of a stochastic policy gradient update, as defined in Theorem 10. Under the same assumptions, the continuous relaxation of $\widetilde{B}_k(\alpha;N)/N$ is maximized by the following step size α^* and batch size N_k^*:*

$$\begin{cases} \alpha^* = \frac{1}{2L} \\ N_k^* = \frac{4\epsilon^2(\delta_k)}{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2}. \end{cases} \tag{47}$$

Using α^* and $\lceil N_k^* \rceil$ in the stochastic gradient ascent update guarantees, with probability at least $1-\delta_k$, the following non-negative performance improvement:

$$J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \frac{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2}{8L}. \tag{48}$$

Proof Fix k and let $\Upsilon(\alpha, N) = \widetilde{B}_k(\alpha;N)/N$ and $N_0 = \epsilon^2(\delta_k)/\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2$. We consider the continuous relaxation of $\Upsilon(\alpha, N)$, where N can be any positive real number. For $N \geq N_0$, the first argument of the max operator in (36) can be selected. Note that the second argument is always a valid choice, since it is a lower bound on the first one for every $N \geq 1$. Thus, we separately solve the following constrained optimization problems:

Table 2 Gradient estimation error bound $\epsilon(\delta)$ for Gaussian and Softmax policies using REINFORCE (RE.), GPOMDP (GP.), or the random-horizon estimator discussed in "Appendix C.2" (RH.) as gradient estimator, where d is the dimension of the policy parameter, M is an upper bound on the max norm of the feature function, R is the maximum absolute-valued reward, γ is the discount factor, T is the task horizon, σ is the standard deviation of the Gaussian policy and τ is the temperature of the Softmax policy

	Gaussian	Softmax
RE.	$\frac{4MRT(1-\gamma^T)}{\sigma(1-\gamma)}\sqrt{14d\log(6/\delta)}$	$\frac{4MRT(1-\gamma^T)}{\tau(1-\gamma)}\sqrt{2d\log(6/\delta)}$
GP.	$\frac{4MR[1-\gamma^T-T(\gamma^T-\gamma^{T+1})]}{\sigma(1-\gamma)^2}\sqrt{14d\log(6/\delta)}$	$\frac{4MR[1-\gamma^T-T(\gamma^T-\gamma^{T+1})]}{\tau(1-\gamma)^2}\sqrt{2d\log(6/\delta)}$
RH.	$\frac{4MR}{\sigma(1-\gamma^{1/2})^2}\sqrt{14d\log(6/\delta)}$	$\frac{4MR}{\tau(1-\gamma^{1/2})^2}\sqrt{2d\log(6/\delta)}$

$$\begin{cases} \max_{\alpha,N} \frac{1}{N}\left(\alpha\left\|\widehat{\nabla}J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|\left(\left\|\widehat{\nabla}J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\| - \frac{\epsilon(\delta_k)}{\sqrt{N}}\right)\right. \\ \qquad\qquad \left. -\alpha^2\frac{L}{2}\left\|\widehat{\nabla}J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2\right) \\ \text{s.t.} \quad \alpha \geq 0, \\ \text{s.t.} \quad N > \frac{\epsilon^2(\delta_k)}{\left\|\widehat{\nabla}J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2}, \end{cases} \tag{49}$$

and:

$$\begin{cases} \max_{\alpha,N} \frac{1}{N}\left(\frac{\alpha}{2}\left(\left\|\widehat{\nabla}J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2 - \frac{\epsilon^2(\delta_k)}{N}\right) - \alpha^2\frac{L}{2}\left\|\widehat{\nabla}J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2\right) \\ \text{s.t.} \quad \alpha \geq 0, \\ \text{s.t.} \quad N > 0. \end{cases} \tag{50}$$

Both problems can be solved in closed form using KKT conditions. The first one (49) yields $\Upsilon^* = \left\|\widehat{\nabla}J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^4 / \left(32L\epsilon^2(\delta_k)\right)$ with the values of α^* and N_k^* given in (47). The second one (50) yields a worse optimum $\Upsilon^* = \left\|\widehat{\nabla}J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^4 / \left(54L\epsilon^2(\delta_k)\right)$ with $\alpha = \frac{1}{3L}$ and $N = 3\epsilon^2(\delta) / \left\|\widehat{\nabla}J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2$. Hence, we keep the first solution. From Theorem 10, using α^* and N_k^* would guarantee $J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \left\|\widehat{\nabla}J(\boldsymbol{\theta}_k;\mathcal{D}_k)\right\|^2 / (8L)$. Of course, only integer batch sizes can be used. However, for $N \geq N_0$, the right-hand side of (36) is monotonically increasing in N. Since $N_k^* \geq N_0$ and $\lceil N_k^* \rceil \geq N_k^*$, the guarantee (48) is still valid when α^* and $\lceil N_k^* \rceil$ are employed in the stochastic gradient ascent update. □

In this case, the optimal step size is constant, and is exactly half the one for the exact case (Corollary 9). In turn, the batch size is adaptive: when the norm of the (estimated) gradient is small, a large batch size is selected. Intuitively, this allows to counteract the variance of the estimator, which is large relatively to the gradient magnitude. One may worry about the recursive dependence of N_k^* on itself through $\mathcal{D}_k$. We will overcome this issue in the next section.

5 Algorithm

In this section, we leverage the theoretical results of the previous sections to design a reinforcement learning algorithm with monotonic improvement guarantees. For the reasons discussed above, we adopt the adaptive-batch-size approach from Sect. 4.3.

Corollary 12 provides a constant step size α^* and a schedule for the batch size $(\lceil N_k^* \rceil)_{k\geq 1}$ that jointly maximize per-trajectory performance improvement under a monotonic-improvement constraint. Plugging these meta-parameters into Algorithm 1, we could obtain a safe policy gradient algorithm. Unfortunately, the closed-form expression for N_k^* provided in (47) cannot be used directly. We must compute the batch size *before* collecting the batch of trajectories $\mathcal{D}_k$, but N_k^* depends on $\mathcal{D}_k$ itself. To overcome this issue, we collect trajectories in an incremental fashion until the optimal batch size is achieved. We call this algorithm Safe Policy Gradient (SPG), outlined in Algorithm 2. The user specifies the failure probability δ_k for each iteration k, while the smoothness constant L and the concentration bound $\epsilon : (0,1) \to \mathbb{R}$ can be computed depending on the policy class and the gradient estimator (see Tables 1 and 2).

Algorithm 2 Safe Policy Gradient (SPG)

Input: initial policy parameter $\boldsymbol{\theta}_0$, smoothness constant L, concentration bound ϵ, failure probabilities $(\delta_k)_{k\geq 1}$, mini-batch size n
$\alpha = \frac{1}{2L}$ ▷ fixed step size
for $k = 1, 2, \ldots$ **do**
  $i = 0$, $\mathcal{D}_{k,0} = \emptyset$
  do
    $i = i + 1$
    Collect trajectory $\tau_{k,i} \sim p_{\boldsymbol{\theta}_k}$
    $\mathcal{D}_{k,i} = \mathcal{D}_{k,i-1} \cup \{\tau_{k,i}\}$
    Compute policy gradient estimate $g_{k,i} = \widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_{k,i})$
    $\delta_{k,i} = \frac{\delta_k}{i(i+1)}$
  while $i < \frac{4\epsilon^2(\delta_{k,i})}{\|g_{k,i}\|^2}$
  $N_k = i$, $\mathcal{D}_k = \mathcal{D}_{k,i}$ ▷ adaptive batch size
  Update policy parameters as $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha \widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)$
end for

We can study the data-collecting process of SPG as a stopping problem. Fixed an iteration k, let $\mathcal{F}_{k,i} = \sigma(\{\tau_{k,1}, \ldots, \tau_{k,i-1}\})$ be the sigma-algebra generated by the first i trajectories collected at that iteration. Let $\mathbb{E}_i[X]$ be short for $\mathbb{E}[X|\mathcal{F}_{i-1}]$.[15] In Sects. 4 and 4.3 we assumed the Euclidean norm of the gradient estimation error to be bounded by $\epsilon(\delta)/\sqrt{N}$ with probability $1-\delta$ for some function $\epsilon : (0,1) \to \mathbb{R}_+$. For Algorithm 2 to be well-behaved, we need gradient estimates to concentrate *exponentially*, which translates into the following, stronger assumption:

Assumption 2 Fixed a parameter $\boldsymbol{\theta} \in \Theta$, a batch size $N \in \mathbb{N}$ and a failure probability $\delta \in (0,1)$, with probability at least $1-\delta$:

$$\left\| \widehat{\nabla} J(\boldsymbol{\theta}; \mathcal{D}) - \nabla J(\boldsymbol{\theta}) \right\| \leq \frac{\epsilon(\delta)}{\sqrt{N}},$$

[15] In the analysis that follows, expectation without a subscript actually denotes $\mathbb{E}[\cdot|\boldsymbol{\theta}_k]$ for a fixed (outer) iteration k of Algorithm 2. However, we do not need this level of detail since data are discarded at the end of each iteration, dependence between different iterations is only through $\boldsymbol{\theta}_k$, and we can analyze each iteration in isolation until the very end of the section.

where $|\mathcal{D}|$ is a dataset of N i.i.d. trajectories collected with π_θ and $\epsilon(\delta) = C\sqrt{d\log(6/\delta)}$ for some problem-dependent constant C that is independent of δ, d and N.

This is satisfied by REINFORCE/G(PO)MDP with Softmax and Gaussian policies, as shown in "Appendix C". In Table 2 we summarize the value of the error bound $\epsilon(\delta)$ to be used in the different scenarios. Equipped with this exponential tail bound we can prove that, at any given (outer) iteration of SPG, the data-collecting process (inner loop) terminates:

Lemma 13 *Fix an iteration k of Algorithm 2 and let N_k the number of trajectories that are collected at that iteration. Under Assumption 2, provided $\|\nabla J(\boldsymbol{\theta}_k)\| > 0$, $\mathbb{E}[N_k] < \infty$.*

Proof First, note that N_k is a stopping time w.r.t. the filtration $(\mathcal{F}_{k,i})_{i\geq 1}$. Consider the event $E_{k,i} = \left\{\|g_{k,i} - \nabla J(\boldsymbol{\theta}_k)\| \leq \epsilon(\delta_{k,i})/\sqrt{i}\right\}$. By Assumption 2, $\mathbb{P}(\neg E_{k,i}) \leq \delta_{k,i}$. This allows to upper bound the expectation of N_k as follows:

$$\mathbb{E}[N_k] \leq \mathbb{E}\left[\sum_{i=1}^{\infty} \mathbb{I}\left(\sqrt{i} < \frac{2\epsilon(\delta_{k,i})}{\|g_{k,i}\|}\right)\right] \tag{51}$$

$$= \mathbb{E}\left[\sum_{i=1}^{\infty} \mathbb{I}\left(\sqrt{i} < \frac{2\epsilon(\delta_{k,i})}{\|g_{k,i}\|}, E_{k,i}\right)\right] + \mathbb{E}\left[\sum_{i=1}^{\infty} \mathbb{I}\left(\sqrt{i} < \frac{2\epsilon(\delta_{k,i})}{\|g_{k,i}\|}, \neg E_{k,i}\right)\right] \tag{52}$$

$$\leq \sum_{i=1}^{\infty} \mathbb{I}\left(\sqrt{i} < \frac{2\epsilon(\delta_{k,i})}{\|\nabla J(\boldsymbol{\theta}_k)\| - \epsilon(\delta_{k,i})/\sqrt{i}}\right) + \sum_{i=1}^{\infty} \mathbb{P}(\neg E_{k,i}) \tag{53}$$

$$\leq \min_{i\geq 1}\left\{\sqrt{i} \geq \frac{2\epsilon(\delta_{k,i})}{\|\nabla J(\boldsymbol{\theta}_k)\| - \epsilon(\delta_{k,i})/\sqrt{i}}\right\} + \sum_{i=1}^{\infty} \delta_{k,i} \tag{54}$$

$$\leq \min_{i\geq 1}\left\{\|\nabla J(\boldsymbol{\theta}_k)\|\sqrt{i} \geq 3\epsilon(\delta_{k,i})\right\}) + \delta_k \sum_{i=1}^{\infty} \frac{1}{i(i+1)} \tag{55}$$

$$\leq \min_{i\geq 1}\left\{\|\nabla J(\boldsymbol{\theta}_k)\|\sqrt{i} \geq 3C\sqrt{d\log(6i(i+1)/\delta_k)}\right\} + 1 \tag{56}$$

$$\leq \min_{i\geq 1}\left\{\|\nabla J(\boldsymbol{\theta}_k)\|^2 i \geq 18C^2 d\log(6i/\delta_k)\right\} + 1 \tag{57}$$

$$\leq \left\lceil \frac{36C^2 d}{\|\nabla J(\boldsymbol{\theta}_k)\|^2} \log \frac{108C^2 d}{\|\nabla J(\boldsymbol{\theta}_k)\|^2 \delta_k} \right\rceil + 1, \tag{58}$$

where (56) is by Assumption 2 and the last inequality is by Lemma 21 assuming $\|\nabla J(\boldsymbol{\theta}_k)\| \leq C$. If the latter is not true, we still get:

$$\mathbb{E}[N_k] \leq \min_{i \geq 1} \left\{ \left\|\nabla J(\boldsymbol{\theta}_k)\right\|^2 i \geq 18C^2 d \log(6i/\delta_k) \right\} + 1 \tag{59}$$

$$\leq \min_{i \geq 1} \left\{ i \geq 18d \log(6i/\delta_k) \right\} + 1 \tag{60}$$

$$\leq \lceil 36d \log(108d/\delta_k) \rceil + 1. \tag{61}$$

□

We can now prove that the policy updates of SPG are safe.

Theorem 14 *Consider Algorithm* 2 *applied to a smoothing policy, where* $\widehat{\nabla} J$ *is an unbiased policy gradient estimator. Under Assumption* 2*, for any iteration* $k \geq 1$*, provided* $\nabla J(\boldsymbol{\theta}_k) \neq 0$*, with probability at least* $1 - \delta_k$*:*

$$J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \frac{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|^2}{8L} \geq 0.$$

Proof Fix an (outer) iteration k of Algorithm 2 and let $g_{k,i} = \widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_{k,i})$ for short. Using an unbiased policy gradient estimator we ensure $\mathbb{E}_i[g_{k,i} - \nabla J(\boldsymbol{\theta}_k)] = 0$, so $X_i = g_{k,i} - \nabla J(\boldsymbol{\theta}_k)$ is a martingale difference sequence adapted to $(\mathcal{F}_{k,i})_{i \geq 1}$. We use an optional stopping argument to show that g_{k,N_k} is an unbiased policy gradient estimate. Lemma 13 shows that N_k is a stopping time w.r.t. the filtration $(\mathcal{F}_{k,i})_{i \geq 1}$ that is finite in expectation. Furthermore, by Assumption 2, integrating the tail:

$$\mathbb{E}_i[\|X_i\|] = \int_0^\infty \mathbb{P}\big(\|X\| > x | \mathcal{F}_{k,i}\big)\, \mathrm{d}x \tag{62}$$

$$\leq 6 \int_0^\infty \exp(-x^2 i/(C^2 d))\, \mathrm{d}x \tag{63}$$

$$\leq 6C\sqrt{\frac{\pi d}{4i}} \leq 6C\sqrt{\frac{\pi d}{4}} \quad \text{for all } i \geq 1. \tag{64}$$

Hence, by optional stopping (Lemma 22), $\mathbb{E}[X_{N_k}] = 0$. Since $X_{N_k} = \widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k) - \nabla J(\boldsymbol{\theta}_k)$, we have $\mathbb{E}[\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)] = \nabla J(\boldsymbol{\theta}_k)$. This shows that the policy update of Algorithm 2 is an unbiased policy-gradient update. By the stopping condition:

$$N_k \geq \frac{4\epsilon^2(\delta_{k,N_k})}{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|^2}. \tag{65}$$

Now consider the following good event:

$$E_k = \left\{ \forall i \geq 1 : \left\|g_{k,i} - \nabla J(\boldsymbol{\theta}_k)\right\| \leq \epsilon(\delta_{k,i})/i \right\}. \tag{66}$$

Under Assumption 2, by union bound:

$$\mathbb{P}\big(\neg E_k\big) \leq \sum_{i=1}^{\infty} \delta_{k,i} = \sum_{i=1}^{\infty} \frac{\delta_k}{i(i+1)} = \delta_k. \tag{67}$$

So E_k holds with probability at least $1 - \delta_k$. Under E_k, the performance improvement guarantee is by Corollary 12, Eq. (65), and the choice of the step size α. □

We have shown that the policy updates of SPG are safe with probability $1 - \delta_k$, where the failure probability δ_k can be specified by the user for each iteration k. Typically, one would like to ensure monotonic improvement for the whole duration of the learning process. This can be achieved by appropriate confidence schedules. If the number of updates K is fixed a priori, $\delta_k = \delta/K$ guarantees monotonic improvement with probability $1 - \delta$. The same can be obtained by using an adaptive confidence schedule $\delta_k = \frac{\delta}{k(k+1)}$, even when the number of updates is not known in advance. Both results are easily shown by taking a union bound over $k \geq 1$. Notice how having an exponential tail bound like the one from Assumption 2 is fundamental for the batch size to have a logarithmic dependence on the number of policy updates.

5.1 Towards a practical algorithm

The version of SPG we have just analyzed is very conservative. The price for guaranteeing monotonic improvement is slow convergence, even in small problems (see Sect. 7.1 for an example). In this section, we discuss possible variants and generalizations of Algorithm 2 aimed at the development of a more practical method. In doing so, we still stay faithful to the principle of satisfying the safety requirement specified by the user with no compromises. We just list the changes here. See "Appendix E" for a more rigorous discussion.

Improved smoothness constant

As mentioned in Sect. 3.3, we can use the improved smoothness constant by Yuan et al. (2021), denoted $L^\star$ in the following, which has a better dependence on the effective horizon. This yields a larger step size with the same theoretical guarantees, and allows to tackle problems with longer horizons in practice.

Mini-batches

In the inner loop of Algorithm 2, instead of just one trajectory at a time, we can collect mini-batches of n independent trajectories. For instance, $n \geq 2$ is required to employ the variance-reducing baselines discussed in Sect. 2. Moreover, a carefully picked mini-batch size n can make the early gradient estimates more stable, leading to an earlier stopping of the inner loop and a smaller batch size N_k. We leave the investigation of the optimal value of n to future work.

Largest safe step size

The meta parameters of Algorithm 2 were selected to maximize a lower bound on the per-trajectory performance improvement. Although we believe this is the most theoretically justified choice, we could gain some convergence speed by using a larger step size. From Theorem 10, it is easy to check that $\alpha = 1/L$ is the largest constant step size we can use with our choice of adaptive batch size from Algorithm 2. We leave the investigation of alternative safe combinations of batch size and (possibly adaptive) step size to future work.

Empirical Bernstein bound

The stopping condition of Algorithm 2 (line 11) is based on a Hoeffding-style bound on the gradient estimation error. In the case of policies with bounded score function, such

as Softmax policies (see "Appendix B.2"), we can use an empirical Bernstein bound instead (Maurer & Pontil, 2009). This requires some modifications to the algorithm, but yields a smaller adaptive batch size with the same safety guarantees. See "Appendix E" for details. Unfortunately, we cannot use the empirical Bernstein bound with the Gaussian policy because of its unbounded score function (see "Appendix B.1").

Weaker safety requirements

Monotonic improvement is a very strong requirement, so we do expect an algorithm with strict monotonic improvement guarantees like SPG to be very data-hungry and slow to converge. However, with little effort, Algorithm 2 can be modified to handle weaker safety requirements. A common one is the *baseline constraint* (Garcelon et al., 2020; Laroche et al., 2019), e.g.,), where the performance of the policy is required to never be (significantly) lower than the performance of a baseline policy π_b. In a real safety-critical application, the reward could be designed so that policies with performance greater than $J(\pi_b)$ are always safe. In other applications, π_b can be an existing, reliable controller that the user wants to replace with an adaptive RL agent. In this case, assuming $\pi_{\boldsymbol{\theta}_0} = \pi_b$, the baseline constraint guarantees that the learning agent never performs worse than the original controller. In our numerical simulations of Sect. 7, we will consider a stronger version of the baseline constraint that we call *milestone constraint*. In this case, the agent's policy must never perform (significantly) worse than the best performance observed so far. Formally, for all $k \geq 1$:

$$J(\boldsymbol{\theta}_{k+1}) \geq \lambda \max_{j=1,2,\dots,k} \{J(\boldsymbol{\theta}_j)\}, \tag{68}$$

where $\lambda \in [0, 1]$ is a user-defined significance parameter. The idea is as follows: every time the agent reaches a new level of performance (a *milestone*), it should never do significantly worse than that. When $\lambda = 1$, this reduces to monotonic improvement. When $\lambda < 1$, some amount of performance oscillation is allowed, but this relaxation can significantly improve the learning speed. Of course, the user has full control on this trade-off through the meta-parameter λ. In "Appendix E" we show that variants of Algorithm 2 satisfy the milestone constraint (and other requirements, such as the baseline constraint) with probability $1 - \delta$ for given significance λ and failure probability δ. We experiment with the milestone constraint in Sect. 7.2.

6 Related works

In this section, we discuss previous results on MI guarantees for policy gradient algorithms.

The seminal work on monotonic performance improvement is by Kakade and Langford (2002). In this work, policy gradient approaches are soon dismissed because of their lack of exploration, although they guarantee MI in the limit of an infinitesimally small step size. The authors hence focus on value-based RL, proposing the Conservative Policy Iteration (CPI) algorithm, where the new policy is a mixture of the old policy and a greedy one. The guaranteed improvement of this new policy (S. Kakade & Langford, 2002, Theorem 4.1) depends on the coefficient of this convex combination, which plays a similar role as the learning rate in our Theorem 8:

$$J(\pi_{k+1}) - J(\pi_k) \geq \frac{\alpha}{(1-\gamma)} \mathop{\mathbb{E}}_{\substack{s \sim \rho^{\pi_k} \\ a \sim \pi_k^+}} \left[A^{\pi_k}(s,a)\right] - \frac{2\alpha^2\gamma\epsilon}{(1-\gamma)^2(1-\alpha)}, \tag{69}$$

where $\epsilon = \max_{s \in \mathcal{S}} |\mathbb{E}_{a \sim \pi_k^+(s,a)} [A^{\pi_k}(s,a)]|$ and $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$ denotes the advantage function of policy π. In fact, both lower bounds have a positive term that accounts for the expected improvement of the new policy w.r.t. the old one, and a penalization term due to the mismatch between the two. The CPI approach is refined by Pirotta et al. (2013), who propose the Safe Policy Iteration (SPI) algorithm (see also, (Metelli et al., 2021)).

Specific performance improvement bounds for policy gradient algorithms were first provided by Pirotta et al. (2013) by adapting previous results on policy iteration (Pirotta et al., 2013) to continuous MDPs. However, the penalty term can only be computed for shallow Gaussian policies ("Appendix B.1") in practice. The bound for the exact framework is:

$$J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \alpha_k \left\|\nabla J(\boldsymbol{\theta}_k)\right\|^2 - \alpha_k^2 \frac{M^2 R}{\sigma^2(1-\gamma)^2} \left(\frac{|\mathcal{A}|}{\sqrt{2\pi}\sigma} + \frac{\gamma}{2(1-\gamma)} \right) \times \left\|\nabla J(\boldsymbol{\theta}_k)\right\|_1^2, \tag{70}$$

where $|\mathcal{A}|$ denotes the volume of the action space. From Table 1, our bound for the same setting is (Corollary 9):

$$J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \alpha_k \left\|\nabla J(\boldsymbol{\theta}_k)\right\|^2 - \alpha_k^2 \frac{M^2 R}{\sigma^2(1-\gamma)^2} \left(1 + \frac{2\gamma}{\pi(1-\gamma)} \right) \left\|\nabla J(\boldsymbol{\theta}_k)\right\|^2,$$

which has the same dependence on the step size, the policy standard deviation σ, the effective horizon $(1-\gamma)^{-1}$, the maximum reward R and the maximum feature norm M. Besides being more general, our penalty term does not depend on the problematic $|\mathcal{A}|$ term (the action space is theoretically unbounded for Gaussian policies) and replaces the l_1 norm of (70) with the smaller l_2 norm. Due to the different constants, we cannot say our penalty is always smaller, but the change of norm could make a big difference in practice, especially for large parameter dimension d. Pirotta et al. (2013) also study the approximate framework. However, albeit formulated in terms of the estimated gradient, their lower bound (Theorem 5.2) *still pertains exact policy gradient updates*, since $\boldsymbol{\theta}_{k+1}$ is defined as $\boldsymbol{\theta}_k + \alpha_k \nabla J(\boldsymbol{\theta}_k)$. This easy-to-overlook observation makes our Theorem 10 the first rigorous monotonic improvement guarantee for stochastic policy gradient updates of the form $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_k \widehat{\nabla} J(\boldsymbol{\theta}_k)$. Pirotta et al. (2013) use their results to design an adaptive step-size schedule for REINFORCE and G(PO)MDP, similarly to what we propose in this paper, but limited to Gaussian policies. Papini et al. (2017) rely on the same improvement lower bound (70) to design an adaptive-batch size algorithm, the most similar to our SPG. Again, their monotonic improvement guarantees are limited to shallow Gaussian policies.

Another related family of performance improvement lower bounds, inspired once again by Kakade and Langford (2002), is that of TRPO. These are very general results that apply to arbitrary pairs of stochastic policies, although they are mostly used to construct policy gradient algorithms in practice. Specializing Theorem 1 by Schulman et al. (2015) to our setting and applying the KL lower bound suggested by the authors we can get the following:

$$J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s \sim \rho^{\boldsymbol{\theta}_k} \\ a \sim \pi_{\boldsymbol{\theta}_{k+1}}}} \left[A^{\boldsymbol{\theta}_k}(s,a)\right] - \frac{2\gamma R}{(1-\gamma)^3} \max_{s \in \mathcal{S}} \left\{ \mathcal{KL}(\pi_{\boldsymbol{\theta}_k}(\cdot|s) \| \pi_{\boldsymbol{\theta}_{k+1}})(\cdot|s) \right\}, \tag{71}$$

where $\pi_{\boldsymbol{\theta}}$ is a stochastic policy. Unfortunately, the lower bound for a policy gradient update (exact or stochastic) cannot be computed exactly. Approximations can lead to very good practical algorithms such as TRPO, but not to actually implementable algorithms with rigorous monotonic improvement guarantees like our SPG. Achiam et al. (2017) and Pajarinen et al. (2019) are able to remove some approximations, but not all.[16] If we were to derive a computable worst-case lower bound starting from (71), we would get a result similar to (70). In fact, Pirotta et al. (2013) explicitly upper-bound the KL divergence in their derivations, which is why the final result is limited to Gaussian policies. We overcome this difficulty by directly upper-bounding the curvature of the objective function (Lemma 5). Furthermore, Theorem 7 suggests that our theory is not limited to policy gradient updates. Arbitrary update directions are considered in (Papini et al., 2020).

Pirotta et al. (2015) provide performance improvement lower bounds (Lemma 8) and adaptive-step algorithms for policy gradients under Lipschitz continuity assumptions on the MDP and the policy. Our assumptions on the environment are much weaker since we only require boundedness of the reward. Intuitively, stochastic policies *smooth out* the irregularities of the environment in computing expected return objectives. In turn, the results of Pirotta et al. (2015) also apply to deterministic policies.

Cohen et al. (2018) provide a general safe policy improvement strategy that can be applied also to policy gradient updates. However, it requires to maintain and evaluate a *set* of policies per iteration instead of a single one.

As mentioned, R. Yuan et al. (2021, Lemma4.4) also study policy gradient with smoothing policies, providing an improved smoothness constant and proving Lipschitz continuity of the objective function. However, their main focus is sample complexity of vanilla policy gradient.

7 Experiments

In this section, we test our SPG algorithm on simulated control tasks. We first test Algorithm 2 with monotonic improvement guarantees on a small continuous-control problem. We then experiment with the milestone-constraint relaxation proposed in Sect. 5.1 on a classic RL benchmark—cart-pole balancing.

7.1 Linear-quadratic regulator with Gaussian policy

The first task is a 1-dimensional Linear-Quadratic Regulator (LQR, Dorato et al. (1994)), a typical continuous-control benchmark. See "Appendix F.1" for a detailed

[16] This is not a critique of the TRPO algorithm per se. Besides the celebrated empirical results, TRPO is also theoretically justified (Neu et al., 2017), only not best as a monotonically improving gradient-descent algorithm (see also, (Shani et al., 2020)).

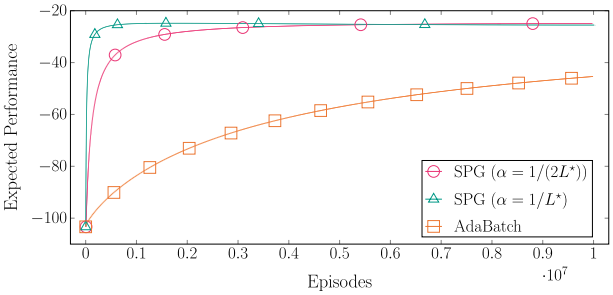

Fig. 1 Performance of SPG and AdaBatch (Papini et al., 2017) on the LQR task with Gaussian policy. Results are averaged over 5 independent runs. The shaded areas correspond to 10 standard deviations. A marker corresponds to 100 policy updates

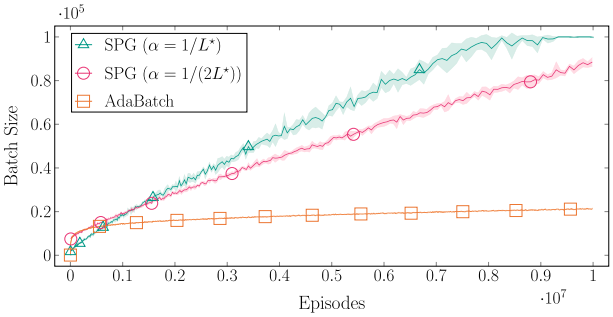

Fig. 2 Batch size of SPG and AdaBatch on the LQR task. Results are averaged over 5 independent runs. The shaded areas correspond to one standard deviation. A marker corresponds to 100 policy updates

task specification. We use a Gaussian policy ("Appendix B.1") that is linear in the state, $\pi_\theta(a|s) = \mathcal{N}(a;\theta s, \sigma^2)$. The task horizon is $T = 10$ and we use $\gamma = 0.9$ as a discount factor. The policy mean parameter is initialized to $\theta_0 = 0$ and the variance is fixed as $\sigma = 1$. For this task, the maximum reward (in absolute value) is $R = 1$ and the only feature is the state itself, giving $M = 1$. Hence, the smoothness constant $L^\star \simeq 200$ is easily computed (see Table 1). Similarly, the error bound can be retrieved from Table 2. We compare the SPG (Algorithm 2) with an existing adaptive-batch-size policy gradient algorithm for Gaussian policies (Papini et al., 2017), discussed in the previous section and labeled *AdaBatch* in the plots. SPG is run with a mini-batch size of $n = 100$ (see Sect. 5.1), and AdaBatch (in the version with Bernstein's inequality as recommended in the original paper) with an initial batch size of $N_0 = 100$. Both use the adaptive confidence schedule $\delta_k = \delta/(k * (k + 1))$ discussed in Sect. 5, with an overall failure probability of $\delta = 0.05$. We also consider SPG with a twice-as-large step size $\alpha = 1/L^\star$, as discussed in Sect. 5.1.

Figure 1 shows the expected performance of the algorithms on the LQR task. For this task, we are able to compute the expected performance in closed form given the policy parameters (Peters, 2002). This allows to filter out the oscillations due to the stochasticity of policy and environment, focusing on actual (expected) performance oscillations. It is also why the variability among different seeds is so small (note that, for this figure, shaded areas correspond to 10 standard deviations. They correspond to a single standard deviation in the other figures). Performance is plotted against the total number of collected trajectories for fair comparison. The distribution of policy updates can be deduced from the markers. We can see that indeed all the safe PG algorithms exhibit monotonic improvement. SPG converges faster than AdaBatch. This is mostly due to the larger step size of SPG (we observed that the step size of SPG was more than 100 times larger than the one of AdaBatch in most of the updates). This allows SPG to converge faster even with fewer policy updates. The variant of SPG with a larger step size ($\alpha = 1/L^\star$) converges faster to a

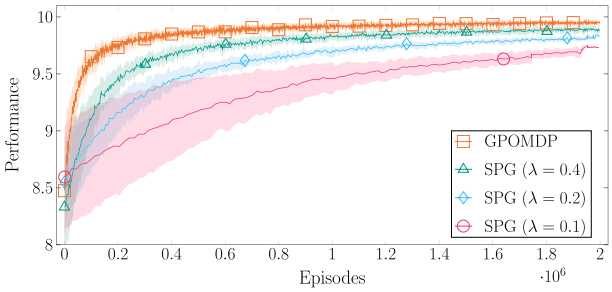

Fig. 3 Performance of GPOMDP and SPG (for different values of the significance parameter λ) on the cart-pole task with Softmax policy. Results are averaged over 5 independent runs. The shaded areas correspond to one standard deviation. A marker corresponds to 1000 policy updates

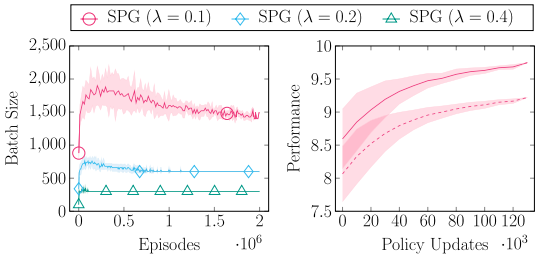

Fig. 4 Further results for SPG on the cart-pole task. On the left, the batch size is plotted against the total number of trajectories. A marker corresponds to 1000 policy updates. On the right, the performance at each policy update (solid line) is compared with the performance threshold (dashed line) when $\lambda = 0.1$. In both plots, shaded areas correspond to one standard deviation

good policy, but the original version from Algorithm 2 achieves higher performance on the long run. This indicates that maximizing the lower bound on per-trajectory performance improvement from Theorem 10 is indeed meaningful.

Figure 2 shows the batch size of the different algorithms. The batch size of SPG is mostly larger than that of AdaBatch. From Sect. 6 we know that the monotonic improvement guarantee of SPG is more rigorous, so a larger batch size is justified. Notice also that the batch size of SPG is smaller than that of AdaBatch in the early iterations, suggesting that the former is more adaptive.

7.2 Cart-pole with softmax policy

The second task is cart-pole (Barto et al., 1983). We use the implementation from `openai/gym`, which has 4-dimensional continuous states and finite actions, $a \in \{1, 2\}$. See "Appendix F.2" for further details. The policy is Softmax ("Appendix B.2"), linear in the state: $\pi_{\boldsymbol{\theta}}(a|s) \propto \exp(\boldsymbol{\theta}_a^\top s)$, with a separate parameter for each action ($\boldsymbol{\theta} = [\boldsymbol{\theta}_1; \boldsymbol{\theta}_2]$). We use a fixed temperature $\tau = 1$, initial policy parameters set to zero (this corresponds to a uniform policy) and $\gamma = 0.9$ as a discount factor. For SPG, we employ all the practical variants proposed in Sect. 5.1. In particular, since the Softmax policy has a bounded score function, we can use the empirical Bernstein bound. Note that we could not have done the same for the LQG task since the score function of the Gaussian policy is unbounded (see "Appendix C"). Moreover, we consider the relaxed milestone constraint for different values of the significance parameter, $\lambda \in \{0.1, 0.2, 0.4\}$. The overall failure probability is always

$\delta = 0.2$, the mini-batch size is $n = 100$, and the step size is $\alpha = 1/L^\star$.[17] We compare with GPOMDP with the same step size but a fixed batch size of $N = 100$, which comes with no safety guarantees, and corresponds to $\lambda = 0$. In Fig. 3 we plot the performance against the total number of collected trajectories. As expected, a more relaxed constraint yields faster convergence. However, no significant performance oscillations are observed, not even in the case of GPOMDP, suggesting that the choice of meta-parameters is still over-conservative. In Fig. 4 (left) we report the evolution of the batch size of SPG during the learning process. Note how, in this case, the batch size seems to converge to a constant value. In Fig. 4 (right) we illustrate the milestone constraint. The solid line is the performance of SPG with $\lambda = 0.1$, while the dotted line is the performance lower-threshold enforced by the milestone constraint, representing 90% of the highest performance achieved so far. As desired, the actual performance never falls under the threshold.

8 Conclusion

We have identified a general class of policies, called smoothing policies, for which the performance measure (expected total reward) is a smooth function of policy parameters. We have exploited this property to select meta-parameters for actor-only policy gradient that guarantee monotonic performance improvement. We have shown that an adaptive batch size can be used in combination with a constant step size for improved efficiency, especially in the early stages of learning. We have designed a monotonically improving policy gradient algorithm, called Safe Policy Gradient (SPG), with adaptive batch size. We have shown how SPG can also be applied to weaker performance-improvement constraints. Finally, we have tested SPG on simulated control tasks.

Albeit the safety motivations are clearly of practical interest, our contribution is mostly theoretical. The meta-parameters proposed in Sect. 4 and used in SPG are based on worst-case problem-dependent constants that are known and easy to compute, but can be very large. This would lead to over-conservative behavior in most problems of interest. However, we believe that this work provides a solid starting point to develop safe and efficient policy gradient algorithms that are rooted in theory.

To conclude, we propose some possible ideas for future work that are aimed to close this gap between theory and practice. While we used the empirical Bernstein bound to characterize the gradient estimation error for Softmax policies, the same cannot be done for Gaussian policies due to their unbounded score function. Tighter concentration inequalities should be studied for this case. The convergence rate of SPG should also be studied. The main challenge here is the growing batch size. The numerical simulations of Sect. 7.1 suggest the the growth is sublinear. Moreover, we have observed convergence to a fixed batch size under the weaker milestone constraint in Sect. 7.2. It is also worth to investigate whether SPG can be combined with stochastic variance-reduction techniques (e.g., (Papini et al., 2018; Yuan et al., 2020)). Convergence to global optima should also be investigated, as is now common in the policy optimization literature (Bhandari & Russo, 2019; Zhang et al., 2020; Agarwal et al., 2020). Actor-critic algorithms (Konda & Tsitsiklis, 1999) are more used than actor-only algorithms in practice (e.g., (Haarnoja et al., 2018)) due to their reduced variance. Thus, extending our improvement guarantees to this class of algorithms is also important. The main challenge lies in handling the bias due to the critic. A promising first step is

[17] Although both theory and our LQR experiments indicate that $\alpha = 1/(2L^\star)$ is ultimately the best choice, we prioritize convergence speed over long-term performance on this larger task.

to consider *compatible* critics that yield unbiased gradient estimates (Sutton et al., 2000; Konda & Tsitsiklis, 1999). Although the class of smoothing policies is very broad, we have restricted our attention to Gaussian and Softmax policies with given features. Other policy classes, such as beta policies (Chou et al., 2017) should be considered. Most importantly, *deep* policies should be considered that also learn the features from data, especially given their success in practice (Duan et al., 2016). See "Appendix B.3" for a brief discussion. Other possible extensions include generalizing the monotonic improvement guarantees to other concepts of safety, such as learning under constraints, or risk-averse RL (Bisi et al., 2020). Finally, the conservative approach adopted in this work could prevent *exploration*, making some tasks very hard to learn. We studied the case of Gaussian policies with adaptive standard deviation in (Papini et al., 2020). Future work should consider the trade-off between safety, efficiency and exploration in greater generality.

Appendix A: Omitted proofs

A.1: Markov decision processes

Lemma 15 *For all* $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ *and* $s_0 \in \mathcal{S}$:

$$\rho^{\pi}_{s_0}(\cdot) = (1-\gamma)\mathbb{1}\{s = s_0\} + \gamma \int_{\mathcal{S}} \rho^{\pi}_{s_0}(s) p(\cdot|s)\, \mathrm{d}s.$$

Proof

$$\begin{aligned}
\gamma \int_{\mathcal{S}} \rho^{\pi}_{s_0}(s) p(\cdot|s)\, \mathrm{d}s &= \gamma \int_{\mathcal{S}} (1-\gamma) \sum_{t=0}^{\infty} \gamma^t p^t_{\pi}(s|s_0) p_{\pi}(\cdot|s)\, \mathrm{d}s \\
&= \gamma(1-\gamma) \sum_{t=0}^{\infty} \gamma^t \int_{\mathcal{S}} p^t_{\pi}(s|s_0) p_{\pi}(\cdot|s)\, \mathrm{d}s \\
&= (1-\gamma) \sum_{t=0}^{\infty} \gamma^{t+1} \int_{\mathcal{S}} p^t_{\pi}(s|s_0) p_{\pi}(\cdot|s)\, \mathrm{d}s \\
&= (1-\gamma) \sum_{t=0}^{\infty} \gamma^{t+1} p^{t+1}_{\pi}(\cdot|s_0) \\
&= (1-\gamma) \sum_{t=1}^{\infty} \gamma^t p^t_{\pi}(\cdot|s_0) \\
&= (1-\gamma) \sum_{t=0}^{\infty} \gamma^t p^t_{\pi}(\cdot|s_0) - (1-\gamma) \\
&= \rho^{\pi}_{s_0}(\cdot) - (1-\gamma)\mathbb{1}\{s = s_0\}.
\end{aligned}$$

□

Proposition 16 *Let* π *be any policy and* f *be any integrable function on* $\mathcal{S}$ *satisfying the following recursive equation*:

$$f(s) = g(s) + \gamma \int_{\mathcal{S}} p_\pi(s'|s) f(s')\, \mathrm{d}s',$$

for all $s \in \mathcal{S}$ and some integrable function g on $\mathcal{S}$. Then:

$$f(s) = \frac{1}{1-\gamma} \int_{\mathcal{S}} \rho_s^\pi(s') g(s')\, \mathrm{d}s',$$

for all $s \in \mathcal{S}$.

Proof

$$\begin{aligned}
\int_{\mathcal{S}} \rho_s^\pi(s') g(s')\, \mathrm{d}s' &= \int_{\mathcal{S}} \rho_s^\pi(s') f(s')\, \mathrm{d}s' - \int_{\mathcal{S}} \rho_s^\pi(s') \gamma \int_{\mathcal{S}} p_\pi(s''|s') f(s'')\, \mathrm{d}s''\, \mathrm{d}s' \\
&= \int_{\mathcal{S}} \rho_s^\pi(s') f(s')\, \mathrm{d}s' - \int_{\mathcal{S}} \gamma \int_{\mathcal{S}} \rho_s^\pi(s') p_\pi(s''|s')\, \mathrm{d}s' f(s'')\, \mathrm{d}s'' \\
&= \int_{\mathcal{S}} \rho_s^\pi(s') f(s')\, \mathrm{d}s' - \int_{\mathcal{S}} \left(\rho_s^\pi(s'') - (1-\gamma)\mathbb{1}\left\{s'' = s\right\}\right) f(s'')\, \mathrm{d}s'' \\
&= (1-\gamma) f(s),
\end{aligned} \tag{A1}$$

where (A1) is from Lemma 15. □

A.2: Lipschitz-smooth functions

The following results, reported in Sect. 2, are well known in the literature (Nesterov, 1998), but we also report proofs for the sake of completeness:

Proposition 17 *Let $\mathcal{X}$ be a convex subset of $\mathbb{R}^d$ and $f : \mathcal{X} \to \mathbb{R}$ be a twice-differentiable function. If the Hessian is uniformly bounded in spectral norm by $L > 0$, i.e., $\sup_{\boldsymbol{x} \in \mathcal{X}} \left\|\nabla^2 f(\boldsymbol{x})\right\|_2 \leq L$, then f is L-smooth.*

Proof Let $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, $\boldsymbol{h} := \boldsymbol{x}' - \boldsymbol{x}$ and $g : [0,1] \to \mathbb{R}$, $g(\lambda) \equiv \nabla_{\boldsymbol{x}} f(\boldsymbol{x} + \lambda \boldsymbol{h})$. Convexity of $\mathcal{X}$ guarantees $\boldsymbol{x} + \lambda \boldsymbol{h} \in \mathcal{X}$ for $\lambda \in [0,1]$. Twice-differentiability of f implies $\nabla_{\boldsymbol{x}} f$ is continuous, which in turn implies g is continuous. From the Fundamental Theorem of Calculus:

$$\begin{aligned}
\nabla_{\boldsymbol{x}} f(\boldsymbol{x}') - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x} + \boldsymbol{h}) - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = g(1) - g(0) = \int_0^1 g'(\lambda)\, \mathrm{d}\lambda \\
= \int_0^1 \boldsymbol{h}^\top \nabla_{\boldsymbol{x}}^2 f(\boldsymbol{x} + \lambda \boldsymbol{h})\, \mathrm{d}\lambda.
\end{aligned} \tag{A2}$$

Hence:

$$\begin{aligned}\left\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}') - \nabla_{\boldsymbol{x}} f(\boldsymbol{x})\right\| &= \left\| \int_0^1 \boldsymbol{h}^\top \nabla_{\boldsymbol{x}}^2 f(\boldsymbol{x} + \lambda \boldsymbol{h}) \, \mathrm{d}\lambda \right\|_2 \\ &\leq \int_0^1 \left\| \nabla_{\boldsymbol{x}}^2 f(\boldsymbol{x} + \lambda \boldsymbol{h}) \boldsymbol{h} \right\|_2 \mathrm{d}\lambda \qquad \text{(A3)} \\ &\leq \int_0^1 \left\| \nabla_{\boldsymbol{x}}^2 f(\boldsymbol{x} + \lambda \boldsymbol{h}) \right\|_2 \|\boldsymbol{h}\|_2 \, \mathrm{d}\lambda \end{aligned}$$

$$\leq L\|\boldsymbol{h}\|_2 = L\left\|\boldsymbol{x}' - \boldsymbol{x}\right\|_2, \qquad \text{(A4)}$$

where (A3) is from the consistency of induced norms, i.e., $\|A\boldsymbol{x}\|_p \leq \|A\|_p \|\boldsymbol{x}\|_p$. □

Proposition 18 (Quadratic Bound) *Let $\mathcal{X}$ be a convex subset of $\mathbb{R}^d$ and $f : \mathcal{X} \to \mathbb{R}$ be an L-smooth function. Then, for every $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$:*

$$\left| f(\boldsymbol{x}') - f(\boldsymbol{x}) - \left\langle \boldsymbol{x}' - \boldsymbol{x}, \nabla f(\boldsymbol{x}) \right\rangle \right| \leq \frac{L}{2} \left\| \boldsymbol{x}' - \boldsymbol{x} \right\|^2, \qquad (18)$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product.

Proof Let $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, $\boldsymbol{h} := \boldsymbol{x}' - \boldsymbol{x}$ and $g : [0,1] \to \mathbb{R}$, $g(\lambda) \equiv f(\boldsymbol{x} + \lambda \boldsymbol{h})$. Convexity of $\mathcal{X}$ guarantees $\boldsymbol{x} + \lambda \boldsymbol{h} \in \mathcal{X}$ for $\lambda \in [0,1]$. Lipschitz smoothness implies continuity of f, which in turn implies g is continuous. From the Fundamental Theorem of Calculus:

$$f(\boldsymbol{x}') - f(\boldsymbol{x}) = g(1) - g(0) = \int_0^1 g'(\lambda) \, \mathrm{d}\lambda. \qquad \text{(A5)}$$

Hence:

$$\begin{aligned}\left| f(\boldsymbol{x}') - f(\boldsymbol{x}) - \left\langle \boldsymbol{x}' - \boldsymbol{x}, \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \right\rangle \right| &= \left| \int_0^1 g'(\lambda) \, \mathrm{d}\lambda - \langle \boldsymbol{h}, \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \rangle \right| \\ &= \left| \int_0^1 \langle \boldsymbol{h}, \nabla_{\boldsymbol{x}} f(\boldsymbol{x} + \lambda \boldsymbol{h}) \rangle \, \mathrm{d}\lambda - \langle \boldsymbol{h}, \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \rangle \right| \\ &= \left| \int_0^1 \langle \boldsymbol{h}, \nabla_{\boldsymbol{x}} f(\boldsymbol{x} + \lambda \boldsymbol{h}) - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \rangle \, \mathrm{d}\lambda \right| \qquad \text{(A6)} \\ &\leq \int_0^1 \left| \langle \boldsymbol{h}, \nabla_{\boldsymbol{x}} f(\boldsymbol{x} + \lambda \boldsymbol{h}) - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \rangle \right| \mathrm{d}\lambda \\ &\leq \int_0^1 \left\| \nabla_{\boldsymbol{x}} f(\boldsymbol{x} + \lambda \boldsymbol{h}) - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \right\|_2 \|\boldsymbol{h}\|_2 \, \mathrm{d}\lambda \end{aligned}$$

$$\begin{aligned} &\leq L\|\boldsymbol{h}\|_2^2 \int_0^1 \lambda \, \mathrm{d}\lambda \\ &= \frac{L}{2} \left\| \boldsymbol{x}' - \boldsymbol{x} \right\|_2^2, \end{aligned} \qquad \text{(A7)}$$

where (A6) is from the Cauchy-Schwartz inequality and (A7) is from the Lipschitz smoothness of f. □

A.3: Smoothing policies and differentiability

Our proofs of the results of Sect. 3.2 rely on the interchange of integrals (w.r.t. states and actions) and derivatives (w.r.t. policy parameters). In the policy gradient literature (cf.. (Sutton et al., 2000; Konda & Tsitsiklis, 1999; Kakade, 2001)), these are typically justified by assuming the derivatives of the policy are bounded uniformly over states and actions, that is:

$$\left|\frac{\partial}{\partial\theta_i}\pi_{\boldsymbol{\theta}}(a|s)\right| \leq C_1, \left|\frac{\partial^2}{\partial\theta_i\partial\theta_j}\pi_{\boldsymbol{\theta}}(a|s)\right| \leq C_2, \tag{A8}$$

for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$, and $i, j = 1, 2, \ldots, d$. The policy gradient itself originally relies on this assumption (Konda & Tsitsiklis, 1999), although weaker requirements are possible (see (Bhandari & Russo, 2019), Sect. 5.1, for a recent discussion). The main problem with (A8) is that the uniform bounds may depend on huge quantities such as the diameter of the parameter space. Even worse, for (linear) Gaussian policies, the first derivative is unbounded:

$$\nabla\pi_{\boldsymbol{\theta}}(a|s) = \pi_{\boldsymbol{\theta}}(a|s)\frac{a - \boldsymbol{\theta}^\top\boldsymbol{\phi}(s)}{\sigma^2}\boldsymbol{\phi}(s), \tag{A9}$$

even when $\boldsymbol{\phi}(s)$ is bounded, since $a \in \mathcal{A} = \mathbb{R}$. However, these policies are smoothing (see "Appendix B.1").

The following application of the Leibniz Integral Rule (cf. (Klenke, 2013), Theorem6.28) shows that our smoothing-policy assumption (Definition 1), can replace the stronger (A8) in differentiating expectations:

Lemma 19 *Let* $\{\pi_{\boldsymbol{\theta}}|\boldsymbol{\theta} \in \Theta\}$, *be a class of smoothing policies and* $f : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ *be any function such that* $\sup_{a\in\mathcal{A}} |f(s,a)|$ *is integrable on* $\mathcal{S}$. *Then* $\int_{\mathcal{S}}\int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s)f(s,a)\,\mathrm{d}a\,\mathrm{d}s$ *is twice differentiable and*:

$$\frac{\partial}{\partial\theta_i}\int_{\mathcal{S}}\int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s)f(s,a)\,\mathrm{d}a\,\mathrm{d}s = \int_{\mathcal{S}}\int_{\mathcal{A}} \frac{\partial}{\partial\theta_i}\pi_{\boldsymbol{\theta}}(a|s)f(s,a)\,\mathrm{d}a\,\mathrm{d}s, \tag{A10}$$

$$\frac{\partial^2}{\partial\theta_i\partial\theta_j}\int_{\mathcal{S}}\int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s)f(s,a)\,\mathrm{d}a\,\mathrm{d}s = \int_{\mathcal{S}}\int_{\mathcal{A}} \frac{\partial^2}{\partial\theta_i\partial\theta_j}\pi_{\boldsymbol{\theta}}(a|s)f(s,a)\,\mathrm{d}a\,\mathrm{d}s, \tag{A11}$$

for all $i, j = 1, 2, \ldots, d$.

Proof Let $B_s = \sup_{a\in\mathcal{A}} |f(s,a)|$ and fix an index $i \leq d$. Let:

$$u_s(\boldsymbol{\theta}) = \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s)f(s,a)\,\mathrm{d}a. \tag{A12}$$

By definition:

$$\frac{\partial}{\partial\theta_i}u_s(\boldsymbol{\theta}) = \lim_{h\to 0}\frac{u(\boldsymbol{\theta} + h\boldsymbol{e}_i) - u(\boldsymbol{\theta})}{h}, \tag{A13}$$

where $\boldsymbol{e}_i$ is the element of the canonical basis of $\mathbb{R}^d$ corresponding to the i-th coordinate. By linearity of integration:

$$\frac{\partial}{\partial \theta_i} u_s(\boldsymbol{\theta}) = \lim_{h \to 0} \int_{\mathcal{A}} \underbrace{\frac{\pi_{\boldsymbol{\theta}+h\boldsymbol{e}_i}(a|s) - \pi_{\boldsymbol{\theta}}(a|s)}{h} f(s,a)}_{g_{\boldsymbol{\theta}}(s,a)} \, \mathrm{d}a. \tag{A14}$$

By assumption, $\pi_{\boldsymbol{\theta}}(a|s)$ is differentiable, so it is continuous. Fix an $h \in \mathbb{R}$. By the mean value theorem, there exist a $\overline{\boldsymbol{\theta}}$ on the segment connecting $\boldsymbol{\theta}$ and $\boldsymbol{\theta} + h\boldsymbol{e}_i$ such that:

$$\frac{\pi_{\boldsymbol{\theta}+h\boldsymbol{e}_i}(a|s) - \pi_{\boldsymbol{\theta}}(a|s)}{h} = \frac{\partial}{\partial \theta_i} \pi_{\boldsymbol{\theta}}(a|s) \bigg|_{\boldsymbol{\theta}=\overline{\boldsymbol{\theta}}}. \tag{A15}$$

Hence, by upper bounding the l_∞ norm with the l_2 norm:

$$\left| g_{\boldsymbol{\theta}}(s,a) \right| \leq B_s \left\| \nabla_{\overline{\boldsymbol{\theta}}} \pi_{\overline{\boldsymbol{\theta}}}(a|s) \right\|. \tag{A16}$$

By the smoothing-policy assumption, Θ is convex, so $\overline{\boldsymbol{\theta}} \in \Theta$, and again by the smoothing-policy assumption:

$$\int_{\mathcal{A}} \left\| \nabla_{\overline{\boldsymbol{\theta}}} \pi_{\overline{\boldsymbol{\theta}}}(a|s) \right\| \mathrm{d}a \leq \int_{\mathcal{A}} \pi_{\overline{\boldsymbol{\theta}}}(a|s) \left\| \nabla_{\overline{\boldsymbol{\theta}}} \log \pi_{\overline{\boldsymbol{\theta}}}(a|s) \right\| \mathrm{d}a \leq \xi_1, \tag{A17}$$

showing that $g_{\boldsymbol{\theta}}(s,a)$ is bounded by a function that is integrable w.r.t. a. By the dominated convergence theorem, we can interchange the limit and the integral in (A14) to obtain:

$$\frac{\partial}{\partial \theta_i} u_s(\boldsymbol{\theta}) = \int_{\mathcal{A}} \lim_{h \to 0} \frac{\pi_{\boldsymbol{\theta}+h\boldsymbol{e}_i}(a|s) - \pi_{\boldsymbol{\theta}}(a|s)}{h} f(s,a) \, \mathrm{d}a = \int_{\mathcal{A}} \frac{\partial}{\partial \theta_i} \pi_{\boldsymbol{\theta}}(a|s) f(s,a) \, \mathrm{d}a.$$

By (A17) and Holder's inequality, $|\partial / \partial \theta_i u_s(\boldsymbol{\theta})| \leq B_s \xi_1$, which is integrable on $\mathcal{S}$. We can then use the same interchange argument to show that:

$$\frac{\partial}{\partial \theta_i} \int_{\mathcal{S}} u_s(\boldsymbol{\theta}) \, \mathrm{d}s = \int_{\mathcal{S}} \frac{\partial}{\partial \theta_i} u_s(\boldsymbol{\theta}) \, \mathrm{d}s = \int_{\mathcal{S}} \int_{\mathcal{A}} \frac{\partial}{\partial \theta_i} \pi_{\boldsymbol{\theta}}(a|s) f(s,a) \, \mathrm{d}a. \tag{A18}$$

For the second derivative, we can just repeat the whole argument from the previous paragraph on $\frac{\partial}{\partial \theta_i} u_s(\boldsymbol{\theta})$. Continuity of the integrand, which is necessary to apply the mean value theorem, follows from twice differentiability of the policy. To apply the dominated convergence theorem, we use the following:

$$\begin{aligned}
\int_{\mathcal{A}} \left\| \nabla^2 \pi_{\boldsymbol{\theta}}(a|s) \right\| \mathrm{d}a &= \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) \left\| \nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s) + \nabla^2 \log \pi_{\boldsymbol{\theta}}(a|s) \right\| \mathrm{d}a \\
&\leq \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) \left\| \nabla \log \pi_{\boldsymbol{\theta}}(a|s) \right\|^2 \mathrm{d}a + \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) \left\| \nabla^2 \log \pi_{\boldsymbol{\theta}}(a|s) \right\| \mathrm{d}a \\
&\leq \xi_2 + \xi_3,
\end{aligned}$$

by the triangular inequality and the smoothing-policy assumption. □

With some work, one can use Lemma 19 to justify all the interchanges of differentiation and integrals from Sect. 3.2 and "Appendix A.4", as the original derivations (Sutton et al., 2000; Kakade, 2001) were justified by (A8).

A.4 Policy Hessian

In the following, the interchange of differentiation and integrals is justified by our smoothing-policy assumption. See "Appendix A.3" for details.

Proposition 20 *Let $\pi_{\boldsymbol{\theta}}$ be a smoothing policy. The Hessian of the performance measure is*:

$$\nabla^2 J(\boldsymbol{\theta}) = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s \sim \rho^{\boldsymbol{\theta}} \\ a \sim \pi_{\boldsymbol{\theta}}(\cdot|s)}} \Big[\nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top Q^{\boldsymbol{\theta}}(s,a) + \nabla Q^{\boldsymbol{\theta}}(s,a) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s) + \big(\nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s) + \nabla^2 \log \pi_{\boldsymbol{\theta}}(a|s) \big) Q^{\boldsymbol{\theta}}(s,a) \Big].$$

Proof We first compute the Hessian of the state-value function:

$$\begin{aligned} \nabla^2 V^{\boldsymbol{\theta}}(s) &= \nabla^2 \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) Q^{\boldsymbol{\theta}}(s,a) \, \mathrm{d}a \\ &= \int_{\mathcal{A}} \nabla \big[\pi_{\boldsymbol{\theta}}(a|s) \big(\nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s) Q^{\boldsymbol{\theta}}(s,a) + \nabla^\top Q^{\boldsymbol{\theta}}(s,a) \big) \big] \, \mathrm{d}a \end{aligned} \tag{A19}$$

$$\begin{aligned} = \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) \big[&(\nabla^2 \log \pi_{\boldsymbol{\theta}}(a|s) + \nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s)) Q^{\boldsymbol{\theta}}(s,a) \\ &+ \nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top Q^{\boldsymbol{\theta}}(s,a) + \nabla Q^{\boldsymbol{\theta}}(s,a) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s) + \nabla^2 Q^{\boldsymbol{\theta}}(s,a) \big] \, \mathrm{d}a \end{aligned} \tag{A20}$$

$$\begin{aligned} = \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) \Bigg[\int &(\nabla^2 \log \pi_{\boldsymbol{\theta}}(a|s) + \nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s)) Q^{\boldsymbol{\theta}}(s,a) \\ &+ \nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top Q^{\boldsymbol{\theta}}(s,a) + \nabla Q^{\boldsymbol{\theta}}(s,a) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s) \\ &+ \nabla^2 \Bigg(r(s,a) + \gamma \int_{\mathcal{S}} p(s'|s,a) V^{\boldsymbol{\theta}}(s') \, \mathrm{d}s' \Bigg) \Bigg] \, \mathrm{d}a \end{aligned} \tag{A21}$$

$$\begin{aligned} = &\int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) \big[(\nabla^2 \log \pi_{\boldsymbol{\theta}}(a|s) + \nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s)) Q^{\boldsymbol{\theta}}(s,a) \\ &+ \nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top Q^{\boldsymbol{\theta}}(s,a) + \nabla Q^{\boldsymbol{\theta}}(s,a) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s) \big] \, \mathrm{d}a \\ &+ \gamma \int_{\mathcal{S}} p_{\boldsymbol{\theta}}(s'|s) \nabla^2 V^{\boldsymbol{\theta}}(s') \, \mathrm{d}s' \\ = &\, g(s) + \frac{\gamma}{1-\gamma} \int_{\mathcal{S}} \rho_s^{\boldsymbol{\theta}}(s') g(s') \, \mathrm{d}s', \end{aligned} \tag{A22}$$

where

$$\begin{aligned} g(s) = \int_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) \big[&\big(\nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s) + \nabla^2 \log \pi_{\boldsymbol{\theta}}(a|s) \big) Q^{\boldsymbol{\theta}}(s,a) \\ &+ \nabla \log \pi_{\boldsymbol{\theta}}(a|s) \nabla^\top Q^{\boldsymbol{\theta}}(s,a) + \nabla Q^{\boldsymbol{\theta}}(s,a) \nabla^\top \log \pi_{\boldsymbol{\theta}}(a|s) \big] \, \mathrm{d}a, \end{aligned}$$

(A19) is from the log-derivative trick, (A20) is from another application of the log-derivative trick, (A21) is from (5), and (A22) is from Lemma 1 with $\nabla^2 V^{\boldsymbol{\theta}}(s')$ as the recursive term. Computing the Hessian of the performance measure is then trivial:

$$\nabla^2 J(\boldsymbol{\theta}) = \nabla^2 \int_{\mathcal{S}} \mu(s) V^{\boldsymbol{\theta}}(s)\, \mathrm{d}s = \int_{\mathcal{S}} \mu(s) \nabla^2 V^{\boldsymbol{\theta}}(s)\, \mathrm{d}s, \tag{A23}$$

where the first equality is from (9). Combining (A22), (A23) and (10) we obtain the statement of the lemma. □

A.5: Auxiliary lemmas

Lemma 21 *For any $a, b > 0$ such that $ab > 1$, a sufficient condition for $x \geq a \log(bx)$ is $x \geq 2a \log(ab)$.*

Proof This can be deduced from the properties of the Lambert function. However, it is easier to verify it directly. Letting $x = 2a \log(ab)$, the first inequality becomes:

$$2a \log(ab) \geq a \log(2ab \log(ab)) = a \log(ab) + a \log(2 \log(ab)), \tag{A24}$$

and $\log(2 \log(y)) \leq \log y$ for any $y > 1$. Finally, notice that $x - a \log(bx)$ is increasing for $x > a$, and $2a \log(ab) > a$ for $ab > 1$. □

Lemma 22 (Optional Stopping) *Let $(X_t)_{t \geq 1}$ be a d-dimensional vector-valued martingale difference sequence and τ be a stopping time, both with respect to a filtration $(\mathcal{F}_t)_{t \geq 0}$. If $\mathbb{E}[\tau] < \infty$ and there exists $c \geq 0$ such that $\mathbb{E}[\|X_t\| \,|\, \mathcal{F}_{t-1}] \leq c$ for every $t \geq 1$, then $\mathbb{E}[X_\tau] = 0$.*

Proof Consider any martingale Y_t such that $X_t = Y_t - Y_{t-1}$. We are going to apply Doob's optional stopping theorem (See Thm 12.5.9 from Grimmett & Stirzaker, 2020)[18] to each element $Y_t^{(i)}$ of Y_t, where $i = 1, \ldots, d$. Sufficient conditions for $\mathbb{E}[Y_\tau^{(i)}] = \mathbb{E}[Y_0^{(i)}]$ are:

1. $\mathbb{E}[\tau] < \infty$,
2. $\mathbb{E}[|Y_{t+1}^{(i)} - Y_t^{(i)}| \,|\, \mathcal{F}_t] \leq c$ for all $t \geq 0$.

The first one is by hypothesis. For the second one:

$$\begin{aligned} \max_{i \in [d]} \mathbb{E}[|Y_{t+1}^{(i)} - Y_t^{(i)}| \,|\, \mathcal{F}_t] &= \max_{i \in [d]} \mathbb{E}[|X_{t+1}^{(i)}| \,|\, \mathcal{F}_t] \\ &\leq \mathbb{E}[\|X_{t+1}\|_\infty | \,|\, \mathcal{F}_t] \\ &\leq \mathbb{E}[\|X_t\|_2 | \,|\, \mathcal{F}_t] \leq c, \end{aligned} \tag{A25}$$

where the last inequality is by hypothesis. So, by optional stopping, $\mathbb{E}[Y_\tau^{(i)}] = \mathbb{E}[Y_0^{(i)}]$ for all $i \in [d]$. We can repeat the same argument for $\tau - 1$. Hence $\mathbb{E}[X_\tau] = \mathbb{E}[Y_\tau] - \mathbb{E}[Y_{\tau-1}] = \mathbb{E}[Y_0] - \mathbb{E}[Y_0] = 0$. □

[18] In the theorem, it is also required that $\mathbb{P}(\tau < \infty) = 1$, but this is implied by $\mathbb{E}[\tau] < \infty$ since τ is nonnegative.

Appendix B: Common smoothing policies

In this section, we show that some of the most commonly used parametric policies are smoothing and provide the corresponding Lipschitz constants for the policy gradient.

B.1: Gaussian policy

Consider a scalar-action, fixed-variance, shallow Gaussian policy:[19]

$$\pi_\theta(a|s) = \mathcal{N}\big(a|\boldsymbol{\theta}^\top \boldsymbol{\phi}(s), \sigma^2\big) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2}\left(\frac{a - \boldsymbol{\theta}^\top \boldsymbol{\phi}(s)}{\sigma}\right)^2\right\}, \tag{B26}$$

where $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$, $\sigma > 0$ is the standard deviation, and $\boldsymbol{\phi} : \mathcal{S} \to \mathbb{R}^d$ is a vector-valued feature function that is bounded in *Euclidean norm*, i.e., $\sup_{s\in\mathcal{S}} \|\boldsymbol{\phi}(s)\| < \infty$. This common policy turns out to be smoothing.

Lemma 23 *Let Π_Θ be the set of Gaussian policies defined in (B26), with parameter set Θ, standard deviation σ and feature function $\boldsymbol{\phi}$. Let M be a non-negative constant such that $\sup_{s\in\mathcal{S}} \|\boldsymbol{\phi}(s)\| \le M$. Then Π_Θ is (ξ_1, ξ_2, ξ_3)-smoothing with the following constants*:

$$\xi_1 = \frac{2M}{\sqrt{2\pi}\sigma}, \quad \xi_2 = \xi_3 = \frac{M^2}{\sigma^2}.$$

The corresponding Lipschitz constant of the policy gradient is:

$$L = \frac{2M^2 R}{\sigma^2(1-\gamma)^2}\left(1 + \frac{2\gamma}{\pi(1-\gamma)}\right). \tag{B27}$$

Proof Fix a $\boldsymbol{\theta} \in \Theta$. Let $x \equiv \frac{a - \boldsymbol{\theta}^\top \boldsymbol{\phi}(s)}{\sigma}$. Note that $\mathcal{A} = \mathbb{R}$ and $\mathrm{d}a = \sigma\, \mathrm{d}x$. We need the following derivatives:

$$\nabla \log \pi_\theta(a|s) = \frac{\boldsymbol{\phi}(s)}{\sigma} x, \tag{B28}$$

$$\nabla^2 \log \pi_\theta(a|s) = -\frac{\boldsymbol{\phi}(s)\boldsymbol{\phi}(s)^\top}{\sigma^2}. \tag{B29}$$

First, we compute ξ_1:

$$\begin{aligned} \mathop{\mathbb{E}}_{a\sim\pi_\theta(\cdot|s)}\big[\big\|\nabla \log \pi_\theta(a|s)\big\|\big] &= \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2} \left\|\frac{\boldsymbol{\phi}(s)}{\sigma} x\right\| \sigma\, \mathrm{d}x \\ &\le \frac{M}{\sqrt{2\pi}\sigma} \int_{\mathbb{R}} e^{-x^2/2} |x|\, \mathrm{d}x \\ &= \frac{2M}{\sqrt{2\pi}\sigma} := \xi_1. \end{aligned} \tag{B30}$$

[19] In this section, π with no subscript always denotes the mathematical constant.

Then, we compute ξ_2:

$$\begin{aligned}\mathop{\mathbb{E}}_{a\sim\pi_\theta(\cdot|s)}\left[\left\|\nabla\log\pi_\theta(a|s)\right\|^2\right] &= \int_{\mathbb{R}}\frac{1}{\sqrt{2\pi}\sigma}e^{-x^2/2}\left\|\frac{\boldsymbol{\phi}(s)}{\sigma}x\right\|^2\sigma\,\mathrm{d}x\\ &\leq \frac{M^2}{\sqrt{2\pi}\sigma^2}\int_{\mathbb{R}}e^{-x^2/2}x^2\,\mathrm{d}x\\ &= \frac{M^2}{\sigma^2}:=\xi_2.\end{aligned}\tag{B31}$$

Finally, we compute ξ_3:

$$\begin{aligned}\mathop{\mathbb{E}}_{a\sim\pi_\theta(\cdot|s)}\left[\left\|\nabla^2\log\pi_\theta(a|s)\right\|\right] &= \int_{\mathbb{R}}\frac{1}{\sqrt{2\pi}\sigma}e^{-x^2/2}\left\|\frac{\boldsymbol{\phi}(s)}{\sigma}x\right\|^2\sigma\,\mathrm{d}x\\ &\leq \frac{M^2}{\sqrt{2\pi}\sigma^2}\int_{\mathbb{R}}e^{-x^2/2}x^2\,\mathrm{d}x\\ &= \frac{M^2}{\sigma^2}:=\xi_3.\end{aligned}\tag{B32}$$

From these constants, the Lipschitz constant of the policy gradient is easily computed (Lemma 6). □

B.2 Softmax policy

Consider a fixed-temperature, shallow Softmax policy for a discrete action space:

$$\pi_\theta(a|s)=\frac{\exp\left\{\frac{\theta^\top\boldsymbol{\phi}(s,a)}{\tau}\right\}}{\sum_{a'\in\mathcal{A}}\exp\left\{\frac{\theta^\top\boldsymbol{\phi}(s,a')}{\tau}\right\}},\tag{B33}$$

where $\boldsymbol{\theta}\in\Theta\subseteq\mathbb{R}^d$, $\tau>0$ is the temperature, and $\boldsymbol{\phi}:\mathcal{S}\times\mathcal{A}\to\mathbb{R}^d$ is a vector-valued feature function that is bounded in *Euclidean norm*, i.e., $\sup_{s\in\mathcal{S},a\in\mathcal{A}}\|\boldsymbol{\phi}(s,a)\|<\infty$. This policy is smoothing.

Lemma 24 *Let Π_Θ be the set of Softmax policies defined in (B33), with parameter set Θ, temperature τ and feature function $\boldsymbol{\phi}$. Let M be a non-negative constant such that $\sup_{s\in\mathcal{S},a\in\mathcal{A}}\|\boldsymbol{\phi}(s,a)\|\leq M$. Then, Π_Θ is (ξ_1,ξ_2,ξ_3)-smoothing with the following constants*:

$$\xi_1=\frac{2M}{\tau},\quad \xi_2=\frac{4M^2}{\tau^2},\xi_3=\frac{2M^2}{\tau^2}.$$

The corresponding Lipschitz constant of the policy gradient is:

$$L=\frac{2M^2R}{\tau^2(1-\gamma)^2}\left(3+\frac{4\gamma}{1-\gamma}\right).\tag{B34}$$

Proof In this case, we can simply bound $\left\|\nabla\log\pi_\theta(a|s)\right\|$ and $\left\|\nabla^2\log\pi_\theta(a|s)\right\|$ uniformly over states and actions. The smoothing conditions follow trivially. We need the following derivatives:

$$\nabla \log \pi_\theta(a|s) = \frac{1}{\tau}\left(\boldsymbol{\phi}(s,a) - \mathop{\mathbb{E}}_{a'\sim\pi_\theta(\cdot|s)}\left[\boldsymbol{\phi}(s,a')\right]\right), \tag{B35}$$

$$\nabla^2 \log \pi_\theta(a|s) = \frac{1}{\tau^2}\mathop{\mathbb{E}}_{a'\sim\pi_\theta(\cdot|s)}\left[\boldsymbol{\phi}(s,a')\left(\mathop{\mathbb{E}}_{a''\sim\pi_\theta(\cdot|s)}\left[\boldsymbol{\phi}(s,a'')\right] - \boldsymbol{\phi}(s,a')\right)^\top\right]. \tag{B36}$$

First, we compute ξ_1 and ξ_2:

$$\begin{aligned}\left\|\nabla \log \pi_\theta(a|s)\right\| &\le \frac{1}{\tau}\left(\|\boldsymbol{\phi}(s,a)\| + \left\|\mathop{\mathbb{E}}_{a'\sim\pi_\theta(\cdot|s)}\left[\boldsymbol{\phi}(s,a')\right]\right\|\right)\\ &\le \frac{2M}{\tau},\end{aligned} \tag{B37}$$

hence $\sup_{s\in\mathcal{S}} \mathbb{E}_{a\sim\pi_\theta}\left[\left\|\nabla \log \pi_\theta(a|s)\right\|\right] \le \frac{2M}{\tau} := \xi_1$ and $\sup_{s\in\mathcal{S}} \mathbb{E}_{a\sim\pi_\theta}\left[\left\|\nabla \log \pi_\theta(a|s)\right\|^2\right] \le \frac{4M^2}{\tau^2} := \xi_2$.

Finally, we compute ξ_3:

$$\begin{aligned}\left\|\nabla^2 \log \pi_\theta(a|s)\right\| &\le \frac{1}{\tau^2}\mathop{\mathbb{E}}_{a'\sim\pi_\theta(\cdot|s)}\left[\left\|\boldsymbol{\phi}(s,a')\left(\mathop{\mathbb{E}}_{a''\sim\pi_\theta(\cdot|s)}\left[\boldsymbol{\phi}(s,a'')\right] - \boldsymbol{\phi}(s,a')\right)^\top\right\|\right]\\ &\le \frac{1}{\tau^2}\mathop{\mathbb{E}}_{a'\sim\pi_\theta(\cdot|s)}\left[\|\boldsymbol{\phi}(s,a')\|\left\|\mathop{\mathbb{E}}_{a''\sim\pi_\theta(\cdot|s)}\left[\boldsymbol{\phi}(s,a'') - \boldsymbol{\phi}(s,a')\right]\right\|\right]\\ &\le \frac{1}{\tau^2}\mathop{\mathbb{E}}_{a'\sim\pi_\theta(\cdot|s)}\left[\|\boldsymbol{\phi}(s,a')\|\mathop{\mathbb{E}}_{a''\sim\pi_\theta(\cdot|s)}\left[\|\boldsymbol{\phi}(s,a'')\| + \|\boldsymbol{\phi}(s,a')\|\right]\right]\\ &\le \frac{2M^2}{\tau^2},\end{aligned} \tag{B38}$$

hence $\sup_{s\in\mathcal{S}} \mathbb{E}_{a\sim\pi_\theta}\left[\left\|\nabla^2 \log \pi_\theta(a|s)\right\|\right] \le \frac{2M^2}{\tau^2} := \xi_3$. From these constants, the Lipschitz constant of the policy gradient is easily computed (Lemma 6). □

Note the similarity with the Gaussian constants from Lemma 23. The temperature parameter τ plays a similar role to the standard deviation σ.

The smoothness constants for Gaussian and Softmax policies are summarized in Table 1.

B.3 Preliminary results on deep policies

The policies we have considered so far rely on given feature maps from state (and action) space to low-dimensional linear space. For many applications, a linear map is not expressive enough to represent good policies. Deep policies (Duan et al., 2016) use Neural Networks (NN) to extract more powerful representations from data. Here we provide a first analysis on how the properties of the NN affect the smoothing properties of the policy.

As an example, consider a Gaussian policy with mean parametrized by a NN, that is:

$$\pi_\theta(a|s) \sim \mathcal{N}(a|\mu_\theta(s), \sigma^2), \tag{B39}$$

where $\mu_\theta : \mathcal{S} \to \mathcal{A}$ is a NN with weights θ. The score function is then:

$$\nabla_\theta \log \pi_\theta(a|s) = \frac{a - \mu_\theta(s)}{\sigma^2} \nabla_\theta \mu_\theta(s), \tag{B40}$$

and its second-order counterpart:

$$\nabla_\theta \log \pi_\theta(a|s) = \frac{a - \mu_\theta(s)}{\sigma^2} \nabla^2_\theta \mu_\theta(s) - \frac{\nabla_\theta \mu_\theta(s) \nabla^\top_\theta \mu_\theta(s)}{\sigma^2}. \tag{B41}$$

For the policy to be smoothing, we need bounds on the gradient and Hessian of the NN w.r.t. its weights (in Euclidean and spectral norm, respectively), both uniformly over the state space. This may suggest the use of activation functions that are smooth and have bounded derivatives for any input, such as tanh or sigmoid activations. We shall study the impact of the network architecture on the smoothing constants in future work.

Appendix C: Exponential concentration of policy gradient estimators

In this section, we provide exponential tail inequalities for REINFORCE and G(PO)MDP (see Sect. 2) policy gradient estimators with policy classes of interest. For the G(PO)MDP estimator, it is useful to notice that it can be equivalently written as (Sutton et al., 2000; Peters & Schaal, 2008):

$$\widehat{\nabla} J(\theta; \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left[\nabla \log \pi_\theta(a_t^i | s_t^i) \sum_{h=t}^{T-1} \gamma^h R(a_h^i, s_h^i) \right], \tag{C42}$$

just by reordering. For simplicity, we will consider estimators without variance-reducing baselines.

First, let us consider the case of a bounded score function:

Lemma 25 *Let* $\|\nabla \log \pi_\theta(a|s)\| \le W$ *for all* $\theta \in \mathbb{R}^d$, $s \in \mathcal{S}$ *and* $a \in \mathcal{A}$. *Then, for any* $\theta \in \mathbb{R}^d$, *with probability* $1 - \delta$:

$$\left\| \widehat{\nabla} J(\theta) - \nabla J(\theta) \right\| \le 2WR_T \sqrt{\frac{2d \log(6/\delta)}{N}}, \tag{C43}$$

where $R_T = \frac{RT(1-\gamma^\top)}{1-\gamma}$ *for REINFORCE and* $R_T = R\frac{1-\gamma^\top - T(\gamma^T - \gamma^{T+1})}{(1-\gamma)^2}$ *for G(PO)MDP.*

Proof For REINFORCE let:

$$\mathcal{R}_t(\tau) := \sum_{h=0}^{T-1} \gamma^h r_h \le \frac{R(1-\gamma^\top)}{1-\gamma} := \overline{\mathcal{R}}_t \qquad \text{for all } t \ge 0, \tag{C44}$$

$$R_T := \sum_{t=0}^{T-1} \overline{\mathcal{R}}_t = \frac{RT(1-\gamma^\top)}{1-\gamma}. \tag{C45}$$

For G(PO)MDP, let:

$$\mathcal{R}_t(\tau) := \sum_{h=t}^{T-1} \gamma^h r_h \leq \frac{R(\gamma^t - \gamma^\top)}{1-\gamma} := \overline{\mathcal{R}}_t, \tag{C46}$$

$$R_T := \sum_{t=0}^{T-1} \overline{\mathcal{R}}_t = R\frac{1 - \gamma^\top - T(\gamma^T - \gamma^{T+1})}{(1-\gamma)^2}. \tag{C47}$$

Let $S^{d-1} = \{v \in \mathbb{R}^d : \|v\| = 1\}$ be the unit sphere in $\mathbb{R}^d$. Fix a vector $v \in S^{d-1}$ and let $\widehat{\nabla} J(\boldsymbol{\theta})$ denote the policy gradient estimate obtained from a single trajectory τ sampled from $p_{\boldsymbol{\theta}}$. For both gradient estimators:

$$\langle v, \widehat{\nabla} J(\boldsymbol{\theta})\rangle = \sum_{t=0}^{T-1} \langle v, \nabla \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\rangle \mathcal{R}_t(\tau) \tag{C48}$$

$$\leq \sum_{t=0}^{T-1} \left\|\nabla \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right\| \mathcal{R}_t(\tau) \tag{C49}$$

$$\leq W R_T, \tag{C50}$$

where the first inequality uses the fact that, for any $x \in \mathbb{R}^d$, $\|x\| = \max_{v \in S^{d-1}} \langle v, x\rangle$. By linearity of expectation and unbiasedness of the gradient estimator, $\mathbb{E}[\langle v, \widehat{\nabla} J(\boldsymbol{\theta})\rangle] = \langle v, \nabla J(\boldsymbol{\theta})\rangle$. Hence, by (C50) and Hoeffding's inequality, with probability $1 - \delta_v$:

$$\langle v, \widehat{\nabla} J(\boldsymbol{\theta}; \mathcal{D}) - \nabla J(\boldsymbol{\theta})\rangle \leq W R_T \sqrt{\frac{2\log(1/\delta_v)}{N}}, \tag{C51}$$

where $N = |\mathcal{D}|$. To turn this into a bound on the Euclidean norm, we need a covering argument. For arbitrary $\eta > 0$, consider an η-cover C_η of S^{d-1}, that is:

$$\max_{v \in S^{d-1}, w \in C_\eta} \|v - w\| \leq \eta. \tag{C52}$$

It is a well known result that a finite cover C_η exists such that $|C_\eta| \leq (3/\eta)^d$. Then, with probability $1 - \delta$:

$$\left\|\widehat{\nabla} J(\boldsymbol{\theta}) - \nabla J(\boldsymbol{\theta})\right\| = \max_{v \in S^{d-1}} \langle v, \widehat{\nabla} J(\boldsymbol{\theta}; \mathcal{D}) - \nabla J(\boldsymbol{\theta})\rangle \tag{C53}$$

$$\leq \max_{v \in C_\eta} \langle v, \widehat{\nabla} J(\boldsymbol{\theta}; \mathcal{D}) - \nabla J(\boldsymbol{\theta})\rangle + \eta \left\|\widehat{\nabla} J(\boldsymbol{\theta}) - \nabla J(\boldsymbol{\theta})\right\| \tag{C54}$$

$$\leq W R_T \sqrt{\frac{2\log(|C_\eta|/\delta)}{N}} + \eta \left\|\widehat{\nabla} J(\boldsymbol{\theta}) - \nabla J(\boldsymbol{\theta})\right\| \tag{C55}$$

$$\leq W R_T \sqrt{\frac{2d\log\left(\frac{3}{\eta\delta}\right)}{N}} + \eta \left\|\widehat{\nabla} J(\boldsymbol{\theta}) - \nabla J(\boldsymbol{\theta})\right\|, \tag{C56}$$

where the first inequality is by Cauchy-Schwarz inequality and definition of C_η, the second one is by union bound over the finite elements of C_η, and the last inequality uses the covering number of the sphere in $\mathbb{R}^d$. Finally, by letting $\eta = 1/2$:

$$\left\|\widehat{\nabla} J(\boldsymbol{\theta}) - \nabla J(\boldsymbol{\theta})\right\| \leq \frac{WR_T}{1-\eta}\sqrt{\frac{2d\log\left(\frac{3}{\eta\delta}\right)}{N}} = 2WR_T\sqrt{\frac{2d\log(6/\delta)}{N}}. \tag{C57}$$

□

The Softmax policy described in "Appendix B.2" satisfies the assumption of Lemma 25 with $W = \frac{2M}{\tau}$ where M is an upper bound on $\|\boldsymbol{\phi}(s)\|$, as shown in the proof of Lemma 24.

Unfortunately, the Gaussian policy class from "Appendix B.1" is not covered by Lemma 25, since its score function is unbounded. Motivated by the broad use of Gaussian policies in applications, we provide an ad-hoc bound for this class:

Lemma 26 *Let Π_Θ be the class of shallow Gaussian policies from Lemma 23. Then, for any $\boldsymbol{\theta} \in \Theta$, with probability $1-\delta$:*

$$\left\|\widehat{\nabla} J(\boldsymbol{\theta}) - \nabla J(\boldsymbol{\theta})\right\| \leq \frac{4MR_T}{\sigma}\sqrt{\frac{14d\log(6/\delta)}{N}}, \tag{C58}$$

where $R_T = \frac{RT(1-\gamma^\top)}{1-\gamma}$ for REINFORCE and $R_T = R\frac{1-\gamma^\top - T(\gamma^T-\gamma^{T+1})}{(1-\gamma)^2}$ for G(PO)MDP.

Proof Let $\mathcal{R}_t(\tau)$, $\overline{\mathcal{R}}_t$, and R_T be defined (differently for the two gradients estimators) as in the proof of Lemma 25. Again, let S^{d-1} be the unit sphere in $\mathbb{R}^d$, fix a vector $v \in S^{d-1}$ and let $\widehat{\nabla} J(\boldsymbol{\theta})$ denote the policy gradient estimate obtained from a single trajectory τ sampled from $p_{\boldsymbol{\theta}}$. Consider the filtration $(\mathcal{F}_t)_{t=0}^{T-1}$ where $\mathcal{F}_t = \sigma(s_0, a_0, \ldots, s_t)$ is the sigma-algebra representing all the knowledge up to the t-th state included. Conditional on s_t, $a_t \sim \mathcal{N}(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s), \sigma^2)$. Hence, conditionally on $\mathcal{F}_t$:

$$\langle v, \nabla \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\rangle = \frac{a_t - \boldsymbol{\theta}^\top \boldsymbol{\phi}(s_t)}{\sigma^2}\langle v, \boldsymbol{\phi}(s_t)\rangle \sim \mathcal{N}\left(0, \frac{\langle v, \boldsymbol{\phi}(s_t)\rangle^2}{\sigma^2}\right). \tag{C59}$$

Let $X_t = \langle v, \nabla \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\rangle$ for brevity. Since X_t is $\mathcal{F}_t$-measurable and $\mathbb{E}[X_t|\mathcal{F}_{t-1}] = 0$, $(X_t)_t$ is a martingale difference sequence adapted to $(\mathcal{F}_t)_t$. Furthermore, (C59) shows that, for any $\lambda > 0$:

$$\mathbb{E}[\exp(\lambda X_t)|\mathcal{F}_t] = \exp\left(\frac{\lambda\langle v, \boldsymbol{\phi}(s_t)\rangle^2}{2\sigma^2}\right) \leq \exp\left(\frac{\lambda\left\|\boldsymbol{\phi}(s_t)\right\|^2}{2\sigma^2}\right) \leq \exp\left(\frac{\lambda M^2}{2\sigma^2}\right), \tag{C60}$$

where the first inequality is by $\|x\| = \max_{v\in S^{d-1}}\langle v, x\rangle$ for any $x \in \mathbb{R}^d$. Hence, X_t is conditionally (M/σ)-subgaussian and $\overline{\mathcal{R}}_t X_t$ is conditionally $(\overline{\mathcal{R}}_t M/\sigma)$-subgaussian. Using Azuma's inequality, for any $b > 0$:[20]

[20] We use the version by Shamir (2011).

$$\mathbb{P}\left(\langle v, \widehat{\nabla} J(\boldsymbol{\theta})\rangle > b\right) = \mathbb{P}\left(\sum_{t=0}^{T-1} X_t \mathcal{R}_t(\tau) > b\right) \tag{C61}$$

$$\leq \mathbb{P}\left(\sum_{t=0}^{T-1} X_t \overline{\mathcal{R}}_t > b\right) \tag{C62}$$

$$\leq \exp\left(-\frac{\sigma^2 b^2}{56M^2 \sum_{t=0}^{T-1} \overline{\mathcal{R}}_t^2}\right) \tag{C63}$$

$$\leq \exp\left(-\frac{\sigma^2 b^2}{56M^2 R_T^2}\right), \tag{C64}$$

showing that $\langle v, \widehat{\nabla} J(\boldsymbol{\theta})\rangle$ is $\sqrt{28} M R_T/\sigma$-subgaussian. From this and $\mathbb{E}[\langle v, \widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D})\rangle] = \langle v, \nabla J(\boldsymbol{\theta})\rangle$, using Hoeffding's inequality for averages of i.i.d. subgaussian random variables:

$$\langle v, \widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D}) - \nabla J(\boldsymbol{\theta})\rangle \leq \frac{M R_T}{\sigma}\sqrt{\frac{56\log(1/\delta_v)}{N}}, \tag{C65}$$

with probability $1-\delta_v$. Finally, using the same covering argument as in the proof of Lemma 25, with probability $1-\delta$:

$$\left\|\widehat{\nabla} J(\boldsymbol{\theta}) - \nabla J(\boldsymbol{\theta})\right\| \leq \frac{2M R_T}{\sigma}\sqrt{\frac{56d\log(6/\delta)}{N}}. \tag{C66}$$

□

The values of $\epsilon(\delta)$ for Gaussian and Softmax policies are summarized in Table 2.

C.1: Empirical Bernstein bound

For bounded-score policies (such as the Softmax), we can improve Lemma 25 using an empirical Bernstein inequality (Maurer & Pontil, 2009):

Lemma 27 *Let* $\left\|\nabla \log \pi_{\boldsymbol{\theta}}(a|s)\right\| \leq W$ *for all* $\boldsymbol{\theta} \in \mathbb{R}^d$, $s \in \mathcal{S}$ *and* $a \in \mathcal{A}$. *Then, for any* $\boldsymbol{\theta} \in \mathbb{R}^d$, *with probability* $1-\delta$:

$$\left\|\widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D}) - \nabla J(\boldsymbol{\theta})\right\| \leq \sqrt{\frac{8d\widehat{V}\log(12/\delta)}{N}} + \frac{14dWR_T\log(6/\delta)}{3(N-1)}. \tag{C67}$$

where $\hat{V} = \frac{1}{N-1}\sum_{i=1}^{N}\left\|\widehat{\nabla} J(\boldsymbol{\theta};\tau_i) - \widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D})\right\|^2$, *and* R_T *is defined as in Lemma 25.*

Proof Recall that $\mathcal{D} = \{\tau_1, \ldots, \tau_N\}$ is a set of trajectories sampled independently from $p_{\boldsymbol{\theta}}$. Let $\widehat{\nabla} J(\boldsymbol{\theta};\tau_i)$ denote the policy gradient estimate obtained from trajectory τ_i, and recall

$J(\boldsymbol{\theta};\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\widehat{\nabla} J(\boldsymbol{\theta};\tau_i)$ denotes the sample mean. Fix a vector $v \in S^{d-1}$, the unit sphere in $\mathbb{R}^d$, and let $X_i = \langle v, \widehat{\nabla} J(\boldsymbol{\theta};\tau_i)\rangle$ for short. Then, as shown in (C50):

$$|X_i| \leq WR_T, \tag{C68}$$

and $\mathbb{E}[X_i] = \langle v, \nabla J(\boldsymbol{\theta})\rangle$. Moreover, $(X_i)_{i=1}^N$ are i.i.d. (conditionally on $\boldsymbol{\theta}$, which is fixed in this case). By Theorem 4 from (Maurer & Pontil, 2009), with probability $1-\delta_v$:

$$\langle v, \widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D}) - \nabla J(\boldsymbol{\theta})\rangle \leq \sqrt{\frac{2\widehat{V}_v \log(2/\delta_v)}{N}} + \frac{7WR_T \log(2/\delta_v)}{3(N-1)}, \tag{C69}$$

where and $\widehat{V}_v$ is the (unbiased) sample variance of the $(X_i)_{i=1}^N$:

$$\widehat{V}_v = \frac{1}{N-1}\sum_{i=1}^{N}\left\langle v, \widehat{\nabla} J(\boldsymbol{\theta};\tau_i) - \widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D})\right\rangle^2 \tag{C70}$$

$$\leq \frac{1}{N-1}\sum_{i=1}^{N}\left\|\widehat{\nabla} J(\boldsymbol{\theta};\tau_i) - \widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D})\right\|^2 := \hat{V}, \tag{C71}$$

where the inequality is by Cauchy-Schwarz and $\|v\| = 1$. Since $\hat{V}$ does not depend on v, we can use the same covering argument as in the proof of Lemma 25 to obtain the desired result. □

To use this concentration inequality in SPG, Algorithm 2 must be modified, as discussed in "Appendix E".

C.2: Infinite-Horizon estimators

To obtain an unbiased estimate of the gradient for the original infinite-horizon performance measure considered in the paper, we can modify our simulation protocol as suggested in (Bedi et al., 2021). Consider a random-horizon G(PO)MDP estimator that, for each episode:

1. Samples a random horizon $T \sim \text{Geom}(1-\gamma^{t/2})$ from a geometric distribution;
2. Generates a trajectory τ of length T with the current policy $\pi_{\boldsymbol{\theta}}$;
3. Outputs $\widehat{\nabla} J(\boldsymbol{\theta};\tau,T) = \sum_{t=0}^{T-1}\left(\gamma^{t/2} r(a_t^i, s_t^i)\sum_{h=0}^{t}\nabla \log \pi_{\boldsymbol{\theta}}(a_h^i|s_h^i)\right)$.

The result can be averaged over a batch of independent trajectories, each with its own independently sampled random length. This policy gradient estimator is unbiased (Bedi et al., 2021, Lemma1). The random horizon should be accounted for in the concentration bounds of Lemma 25, 26, and 27. However, note that the term R_T, for the G(PO)MDP estimator, is bounded as follows:

$$R_T = R\frac{1-\gamma^{\top} - T(\gamma^T - \gamma^{T+1})}{(1-\gamma)^2} \leq \frac{R}{(1-\gamma)^2}, \tag{C72}$$

for any $T \geq 0$. Hence, Lemma 25, 26, and 27 all hold for the random-horizon estimator with $R_T = R/(1-\gamma^{1/2})^2$. The corresponding error bounds are reported in Table 2. We leave a more refined analysis of the variance and tail behavior of this random-horizon estimator to future work.

Appendix D: Variance of policy gradient estimators

In this section, we provide upper bounds on the variance of the (finite-horizon) REINFORCE and G(PO)MDP estimators, generalizing existing results for Gaussian policies (Zhao et al., 2011; Pirotta et al., 2013) to smoothing policies. We begin by bounding the variance of the REINFORCE estimator:

Lemma 28 *Given a* (ξ_1, ξ_2, ξ_3)*-smoothing policy class* Π_Θ *and an effective task horizon* T*, for every* $\boldsymbol{\theta} \in \Theta$*, the variance of the REINFORCE estimator* (*with zero baseline*) *is upper-bounded as follows*:

$$\mathbb{V}\mathrm{ar}\left[\widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D})\right] \leq \frac{T\xi_2 R^2 (1-\gamma^\top)^2}{N(1-\gamma)^2}. \tag{D73}$$

Proof Let $g_{\boldsymbol{\theta}}(\tau) := \left(\sum_{t=0}^{T-1} \gamma^t r(a_t, s_t)\right)\left(\sum_{t=0}^{T-1} \nabla \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right)$ with $s_t, a_t \in \tau$ for $t = 0, \dots, T-1$. Using the definition of REINFORCE (14) with $b = 0$:

$$\begin{aligned}
\underset{\mathcal{D}\sim p_{\boldsymbol{\theta}}}{\mathbb{V}\mathrm{ar}}\left[\widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D})\right] &= \frac{1}{N} \underset{\tau\sim p_{\boldsymbol{\theta}}}{\mathbb{V}\mathrm{ar}}\left[g_{\boldsymbol{\theta}}(\tau)\right] \\
&\leq \frac{1}{N} \underset{\tau\sim p_{\boldsymbol{\theta}}}{\mathbb{E}}\left[\left\|g_{\boldsymbol{\theta}}(\tau)\right\|^2\right] \\
&\leq \frac{R^2(1-\gamma^\top)^2}{N(1-\gamma)^2} \underset{\tau\sim p_{\boldsymbol{\theta}}}{\mathbb{E}}\left[\left\|\sum_{t=0}^{T-1} \nabla \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right\|^2\right] \\
&\leq \frac{R^2(1-\gamma^\top)^2}{N(1-\gamma)^2} \sum_{i=1}^{m} \underset{\tau\sim p_{\boldsymbol{\theta}}}{\mathbb{E}}\left[\sum_{t=0}^{T-1} \left(D_i \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right)^2 \right. \\
&\qquad \left. + 2\sum_{t=0}^{T-2}\sum_{h=t+1}^{T-1} D_i \log \pi_{\boldsymbol{\theta}}(a_t|s_t) D_i \log \pi_{\boldsymbol{\theta}}(a_h|s_h)\right] \\
&= \frac{R^2(1-\gamma^\top)^2}{N(1-\gamma)^2} \underset{\tau\sim p_{\boldsymbol{\theta}}}{\mathbb{E}}\left[\sum_{t=0}^{T-1} \left\|\nabla \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right\|^2\right]
\end{aligned} \tag{D74}$$

$$\begin{aligned}
&= \frac{R^2(1-\gamma^\top)^2}{N(1-\gamma)^2} \sum_{t=0}^{T-1} \underset{s_0\sim\mu}{\mathbb{E}}\left[\dots \underset{a_t\sim\pi_{\boldsymbol{\theta}}(\cdot|s_t)}{\mathbb{E}}\left[\left\|\nabla \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right\|^2 \,|\, s_t\right] \dots\right] \\
&\leq \frac{T\xi_2 R^2(1-\gamma^\top)^2}{N(1-\gamma)^2},
\end{aligned} \tag{D75}$$

where (D74) is from the following:

$$\begin{aligned}&\mathop{\mathbb{E}}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-2}\sum_{h=t+1}^{T-1} D_i \log \pi_{\boldsymbol{\theta}}(a_t|s_t) D_i \log \pi_{\boldsymbol{\theta}}(a_h|s_h)\right]\\ &\quad=\sum_{t=0}^{T-2}\mathop{\mathbb{E}}_{s_0\sim\mu}\left[\dots \mathop{\mathbb{E}}_{a_t\sim\pi_{\boldsymbol{\theta}}(\cdot|s_t)}\left[D_i \log \pi_{\boldsymbol{\theta}}(a_t|s_t)\right.\right.\end{aligned} \tag{D76}$$

$$\begin{aligned}&\sum_{h=t+1}^{T-1}\mathop{\mathbb{E}}_{s_{t+1}\sim p(\cdot|s_t,a_t)}\left[\dots \mathop{\mathbb{E}}_{a_h\sim\pi_{\boldsymbol{\theta}}(\cdot|s_h)}\left[D_i \log \pi_{\boldsymbol{\theta}}(a_h|s_h)\,\middle|\middle|\, s_h\right]\dots\,\middle|\middle|\, a_t\right]\,\middle|\middle|\, s_t\right]\dots\Bigg]\\ &=0,\end{aligned} \tag{D77}$$

where the last equality is from $\mathbb{E}_{a_h\sim\pi_{\boldsymbol{\theta}}(\cdot|s_h)}\left[D_i \log \pi_{\boldsymbol{\theta}}(a_h|s_h)\right]=0$. □

This is a generalization of Lemma 5.3 from Pirotta et al. (2013), which in turn is an adaptation of Theorem 2 from Zhao et al. (2011). In the Gaussian case, the original lemma is recovered by plugging the smoothing constant $\xi_2=\frac{M^2}{\sigma^2}$ from Lemma 23. Note also that, from the definition of smoothing policy, only the second condition (20) is actually necessary for Lemma 28 to hold.

For the G(PO)MDP estimator, we obtain an upper bound that does not grow linearly with the horizon T:

Lemma 29 *Given a (ξ_1,ξ_2,ξ_3)-smoothing policy class Π_Θ and an effective task horizon T, for every $\boldsymbol{\theta}\in\Theta$, the variance of the G(PO)MDP estimator (with zero baseline) is upper-bounded as follows*:

$$\mathbb{V}\mathrm{ar}\left[\widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D})\right]\leq\frac{\xi_2 R^2\left(1-\gamma^{\top}\right)}{N(1-\gamma)^3}. \tag{D78}$$

Proof Let $g_{\boldsymbol{\theta}}(\tau):=\sum_{t=0}^{T-1}\gamma^t r(a_t,s_t)\left(\sum_{h=0}^{t}\nabla\log\pi_{\boldsymbol{\theta}}(a_h|s_h)\right)$ with $s_t,a_t\in\tau$ for $t=0,\dots,T-1$. Using the definition of G(PO)MDP (15) with $b=0$:

$$\mathop{\mathbb{V}\mathrm{ar}}_{\mathcal{D}\sim p_\theta}\left[\widehat{\nabla} J(\boldsymbol{\theta};\mathcal{D})\right]=\frac{1}{N}\mathop{\mathbb{V}\mathrm{ar}}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1}\gamma^t r(a_t,s_t)\left(\sum_{h=0}^{t}\nabla\log\pi_{\boldsymbol{\theta}}(a_h|s_h)\right)\right] \tag{D79}$$

$$\begin{aligned}&\leq\frac{1}{N}\mathop{\mathbb{E}}_{\tau\sim p_\theta}\left[\left(\sum_{t=0}^{T-1}\gamma^{t/2} r(a_t,s_t)\gamma^{t/2}\left(\sum_{h=0}^{t}\nabla\log\pi_{\boldsymbol{\theta}}(a_h|s_h)\right)\right)^2\right]\\ &\leq\frac{1}{N}\mathop{\mathbb{E}}_{\tau\sim p_\theta}\left[\left(\sum_{t=0}^{T-1}\gamma^t r(a_t,s_t)^2\right)\left(\sum_{t=0}^{T-1}\gamma^t\left(\sum_{h=0}^{t}\nabla\log\pi_{\boldsymbol{\theta}}(a_h|s_h)\right)^2\right)\right]\end{aligned} \tag{D80}$$

Table 3 Upper bounds on the variance $\mathbb{V}(\boldsymbol{\theta})$ for common policy gradient estimators (single trajectory, no baseline), assuming the policy is smoothing (Definition 1). Here R is the maximum absolute-valued reward, γ is the discount factor, T is the task horizon, and the smoothing constant ξ_2 can be retrieved from Table 1 depending on the policy class

REINFORCE	**G(PO)MDP**
$\frac{T\xi_2 R^2(1-\gamma^\top)^2}{(1-\gamma)^2}$	$\frac{\xi_2 R^2(1-\gamma^\top)}{(1-\gamma)^3}$

$$\leq \frac{R^2(1-\gamma^\top)}{N(1-\gamma)} \mathop{\mathbb{E}}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1}\gamma^t\left(\sum_{h=0}^{t}\nabla\log\pi_{\boldsymbol{\theta}}(a_h|s_h)\right)^2\right]$$
$$\leq \frac{\xi_2 R^2(1-\gamma^\top)}{N(1-\gamma)}\sum_{t=0}^{T-1}\gamma^t(t+1) \tag{D81}$$

$$= \frac{\xi_2 R^2(1-\gamma^\top)}{N(1-\gamma)^3}\left[1-T\Big(\underbrace{\gamma^\top-\gamma^{T+1}}_{\geq 0}\Big)-\gamma^\top\right]$$
$$\leq \frac{\xi_2 R^2\Big(1-\gamma^\top\Big)}{N(1-\gamma)^3}, \tag{D82}$$

where (D79) is from the fact that the trajectories are i.i.d., (D80) is from the Cauchy-Schwarz inequality, (D81) is from the same argument used for (D74) in the proof of Lemma 28, and (D82) is from the sum of the arithmetico-geometric sequence. □

This is a generalization of Lemma 5.5 from Pirotta et al. (2013). Again, in the Gaussian case, the original lemma is recovered by plugging the smoothing constant $\xi_2 = \frac{M}{\sigma^2}$ from Lemma 23 (Table 1). Note that this variance upper bound stays finite in the limit $T \to \infty$, which is not the case for REINFORCE. The variance upper bounds are summarized in Table 3.

Appendix E: Analysis of relaxed algorithm

In this section, we will analyze in more detailed the variants of SPG introduced in Sect. 5.1. In particular, we will consider a very general *relaxed* improvement guarantee, then we will specialize it to the baseline and milestone constraints discussed in the main paper.

The pseudocode for the relaxed version of SPG is provided in Algorithm 3.

Algorithm 3 Relaxed SPG

Input: initial policy parameter $\boldsymbol{\theta}_0$, smoothness constant L, concentration bound ϵ, failure probabilities $(\delta_k)_{k\geq 1}$, degradation thresholds $(\Delta_k)_{k\geq 1}$, mini-batch size n
$\alpha = \frac{1}{L}$ ▷ fixed step size
for $k = 1, 2, \ldots$ **do**
  $i = 0$, $\mathcal{D}_{k,0} = \emptyset$
  do
    $i = i + 1$
    Collect trajectories $\tau_{k,i,1} \ldots \tau_{k,i,n} \overset{\text{iid}}{\sim} p_{\boldsymbol{\theta}_k}$
    $\mathcal{D}_{k,i} = \mathcal{D}_{k,i-1} \cup \{\tau_{k,i,1} \ldots \tau_{k,i,n}\}$
    Compute policy gradient estimate $g_{k,i} = \widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_{k,i})$
    $\delta_{k,i} = \frac{\delta_k}{i(i+1)}$
  while $\epsilon(ni, \delta_{k,i}) > \frac{\|g_{k,i}\|}{2} + \frac{L\Delta_k}{\|g_{k,i}\|}$
  $N_k = ni$, $\mathcal{D}_k = \mathcal{D}_{k,i}$ ▷ adaptive batch size
  Update policy parameters as $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha \widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)$
end for

The algorithm takes as additional inputs the mini-batch size n and a sequence of degradation thresholds $\Delta_k \geq 0$. Moreover, it assumes access to a generic gradient estimation error function ϵ with the following property:

Assumption 3 Fixed a parameter $\boldsymbol{\theta} \in \Theta$, a batch size $N \in \mathbb{N}$ and a failure probability $\delta \in (0, 1)$, with probability at least $1 - \delta$:

$$\left\| \widehat{\nabla} J(\boldsymbol{\theta}; \mathcal{D}) - \nabla J(\boldsymbol{\theta}) \right\| \leq \epsilon(N, \delta),$$

where $|\mathcal{D}|$ is a dataset of N i.i.d. trajectories collected with $\pi_{\boldsymbol{\theta}}$ and:

$$\epsilon(N, \delta) = \mathcal{O}\left(\frac{\log(1/\delta)}{\sqrt{N}} \right). \tag{E83}$$

The assumption, as the analysis that will follow, is less precise than Assumption 2, but more general. Indeed, it allows to use the empirical Bernstein bound from Lemma 27 for Softmax and other bounded-score policies. We can prove that the per-iteration performance degradation of Algorithm 3 is bounded by the user-defined threshold Δ_k with high probability. Of course, when $\Delta_k = 0$, this is still a monotonic improvement guarantee.

Theorem 30 *Consider Algorithm* 3 *applied to a smoothing policy, where* $\widehat{\nabla} J$ *is an unbiased policy gradient estimator. Under Assumption* 3*, for any iteration* $k \geq 1$*, provided* $\nabla J(\boldsymbol{\theta}_k) \neq 0$*, with probability at least* $1 - \delta_k$:

$$J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq -\Delta_k.$$

Proof Let the filtration $(\mathcal{F}_{k,i})_{i\geq 1}$ be defined as in Sect. 5 and note that N_k is a stopping time w.r.t. this filtration. Consider the event $E_{k,i} = \left\{ \left\| g_{k,i} - \nabla J(\boldsymbol{\theta}_k) \right\| \leq \epsilon(i, \delta_{k,i}) \right\}$. By Assumption 2, $\mathbb{P}(\neg E_{k,i}) \leq \delta_{k,i}$. Hence, by the same arguments used in the proof of Lemma 13:

$$\mathbb{E}[N_k] \leq \mathbb{E}\left[\sum_{i=1}^{\infty} \mathbb{I}\left(\epsilon(ni, \delta_{k,i}) > \frac{\|g_{k,i}\|}{2} + \frac{L\Delta_k}{\|g_{k,i}\|} \right) \right] \tag{E84}$$

$$\leq \mathbb{E}\left[\sum_{i=1}^{\infty} \mathbb{I}\left(\epsilon(ni, \delta_{k,i}) > \frac{\|g_{k,i}\|}{2}\right)\right] \tag{E85}$$

$$\begin{aligned} &= \mathbb{E}\left[\sum_{i=1}^{\infty} \mathbb{I}\left(\epsilon(ni, \delta_{k,i}) > \frac{\|g_{k,i}\|}{2}, E_{k,i}\right)\right] \\ &\quad + \mathbb{E}\left[\sum_{i=1}^{\infty} \mathbb{I}\left(\epsilon(i, \delta_{k,i}) > \frac{\|g_{k,i}\|}{2}, \neg E_{k,i}\right)\right] \end{aligned} \tag{E86}$$

$$\leq \sum_{i=1}^{\infty} \mathbb{I}\left(\epsilon(ni, \delta_{k,i}) > \frac{\left(\|\nabla J(\boldsymbol{\theta}_k)\| - \epsilon(ni, \delta_{k,i})\right)}{2}\right) + \sum_{i=1}^{\infty} \mathbb{P}(\neg E_{k,i}) \tag{E87}$$

$$\leq \min_{i \geq 1}\left\{\epsilon(ni, \delta_{k,i}) \leq \frac{\left(\|\nabla J(\boldsymbol{\theta}_k)\| - \epsilon(ni, \delta_{k,i})\right)}{2}\right\} + \sum_{i=1}^{\infty} \delta_{k,i} \tag{E88}$$

$$\leq \min_{i \geq 1}\left\{\epsilon(ni, \delta_{k,i}) \leq \frac{\|\nabla J(\boldsymbol{\theta}_k)\|}{3}\right\}) + \delta_k \sum_{i=1}^{\infty} \frac{1}{i(i+1)} \tag{E89}$$

$$= \min_{i \geq 1}\left\{\epsilon(ni, \delta_{k,i}) \leq \frac{\|\nabla J(\boldsymbol{\theta}_k)\|}{3}\right\}) + \delta_k, \tag{E90}$$

which is finite since, by Assumption 3:

$$\epsilon(ni, \delta_{k,i}) = O\left(\frac{\log(1/\delta_{k,i})}{\sqrt{ni}}\right) = O\left(\frac{\log i}{\sqrt{i}}\right). \tag{E91}$$

This shows that the inner loop of Algorithm 3 always terminates with a finite batch size. By the same optional-stopping argument as in the proof of Theorem 14, $\mathbb{E}[\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)] = \nabla J(\boldsymbol{\theta}_k)$, which means the gradient estimate is unbiased. By the stopping condition, for all k:

$$\epsilon(N_k, \delta_{k,N_k}) \leq \frac{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|}{2} + \frac{L\Delta_k}{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|}, \tag{E92}$$

with probability at least $1 - \sum_{i=1}^{\infty} \delta_{k,i} = 1 - \sum_{i=1}^{\infty} \delta_k/(i(i+1)) = 1 - \delta_k$. By Theorem 10 and the choice of step size $\alpha = 1/L$, with the same probability:

$$J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \frac{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|}{L}\left(\frac{\left\|\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)\right\|}{2} - \epsilon(N_k, \delta_{k,N_k})\right) \tag{E93}$$

$$\geq -\Delta_k, \tag{E94}$$

where the last inequality is from (E92). □

In the following we discuss some applications of Algorithm 3 to specific safety requirements.

Baseline Constraint.

A common requirement is for the updated policy not to perform (significantly) worse than a known baseline policy (e.g., Garcelon et al., 2020; Laroche et al. 2019). The safety constraint is thus:

$$J(\boldsymbol{\theta}_{k+1}) \geq \lambda J_b, \tag{E95}$$

where J_b is the (discounted) performance of the baseline policy and $\lambda \in [0,1]$ is a user-defined significance parameter. Equivalently, $J(\boldsymbol{\theta}_{k+1}) - J(\boldsymbol{\theta}_k) \geq \lambda J_b - J(\boldsymbol{\theta}_k)$, and Algorithm 3 satisfies this safety requirement if we set the degradation threshold as follows:

$$\Delta_k = \max\{J(\boldsymbol{\theta}_k) - \lambda J_b, 0\}. \tag{E96}$$

However, the performance of the current policy must also be estimated from data, and accidentally over-estimating it may result in excessive performance degradation. Hence, we replace it with a lower confidence bound based on the empirical Bernstein inequality (Maurer & Pontil, 2009). See Algorithm 4 for details. Note how the failure probability in line 10 is adjusted w.r.t. Algorithm 3 to account for this additional estimation step. With this small caveat, the analysis of Algorithm 4 can be carried out analogously to the one of Algorithm 3.

Algorithm 4 SPG with baseline constraint

1: **Input:** initial policy parameter $\boldsymbol{\theta}_0$, smoothness constant L, concentration bound ϵ, failure probabilities $(\delta_k)_{k\geq 1}$, baseline performance J_b, significance parameter λ, mini-batch size n
2: $\alpha = \frac{1}{L}$ ▷ fixed step size
3: **for** $k = 1, 2, \ldots$ **do**
4: $\quad i = 0$, $\mathcal{D}_{k,0} = \emptyset$
5: $\quad$ **do**
6: $\quad\quad i = i + 1$
7: $\quad\quad$ Collect trajectories $\tau_{k,i,1} \ldots \tau_{k,i,n} \overset{\text{iid}}{\sim} p_{\boldsymbol{\theta}_k}$
8: $\quad\quad \mathcal{D}_{k,i} = \mathcal{D}_{k,i-1} \cup \{\tau_{k,i,1} \ldots \tau_{k,i,n}\}$
9: $\quad\quad$ Compute policy gradient estimate $g_{k,i} = \widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_{k,i})$
10: $\quad\quad \delta_{k,i} = \frac{\delta_k}{2i(i+1)}$
11: $\quad\quad$ Estimate performance mean $\hat{J}$ and variance $\hat{V}$ from $\mathcal{D}_{k,i}$
12: $\quad\quad \underline{J} = \hat{J} - \sqrt{\frac{2\hat{V}\log(2/\delta_{k,i})}{ni}} - \frac{7R\log(2/\delta_{k,i})}{3(1-\gamma)(ni-1)}$
13: $\quad\quad \Delta_{k,i} = \max\{\underline{J} - \lambda J_b, 0\}$ ▷ baseline constraint
14: $\quad$ **while** $\epsilon(ni, \delta_{k,i}) > \frac{\|g_{k,i}\|}{2} + \frac{L\Delta_{k,i}}{\|g_{k,i}\|}$
15: $\quad N_k = ni$, $\mathcal{D}_k = \mathcal{D}_{k,i}$ ▷ adaptive batch size
16: $\quad$ Update policy parameters as $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)$
17: **end for**

Milestone Constraint.

In our numerical simulations we consider the following safety constraint:

$$J(\boldsymbol{\theta}_{k+1}) \geq \lambda \max_{h=1,\ldots,k} J(\boldsymbol{\theta}_h), \tag{E97}$$

which can be enforced by setting the degradation threshold in Algorithm 3 as:

$$\Delta_k = \max\left\{ J(\boldsymbol{\theta}_k) - \lambda \max_{h=1,\ldots,k} J(\boldsymbol{\theta}_h), 0 \right\}. \tag{E98}$$

Again, we must replace the unknown performance $J(\boldsymbol{\theta}_k)$ with a lower confidence bound. In this case, we also need to overestimate the best historical performance. See Algorithm 5 for details. Note that this safety constraint reduces to monotonic improvement if the significance parameter is set to $\lambda = 1$, since $\max_{h=1,\ldots,k} J(\boldsymbol{\theta}_h) \geq J(\boldsymbol{\theta}_k)$.

Algorithm 5 SPG with milestone constraint

Input: initial policy parameter $\boldsymbol{\theta}_0$, smoothness constant L, concentration bound ϵ, failure probabilities $(\delta_k)_{k\geq 1}$, baseline performance J_b, significance parameter λ, mini-batch size n
$\alpha = \frac{1}{L}$ ▷ fixed step size
$J^\star = -\infty$
for $k = 1, 2, \ldots$ **do**
  $i = 0$, $\mathcal{D}_{k,0} = \emptyset$
  do
    $i = i + 1$
    Collect trajectories $\tau_{k,i,1} \ldots \tau_{k,i,n} \overset{\text{iid}}{\sim} p_{\boldsymbol{\theta}_k}$
    $\mathcal{D}_{k,i} = \mathcal{D}_{k,i-1} \cup \{\tau_{k,i,1} \ldots \tau_{k,i,n}\}$
    Compute policy gradient estimate $g_{k,i} = \widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_{k,i})$
    $\delta_{k,i} = \frac{\delta_k}{2i(i+1)}$
    Estimate performance mean $\hat{J}$ and variance $\hat{V}$ from $\mathcal{D}_{k,i}$
    $\underline{J} = \hat{J} - \sqrt{\frac{2\hat{V}\log(2/\delta_{k,i})}{ni}} - \frac{7R\log(2/\delta_{k,i})}{3(1-\gamma)(ni-1)}$
    $\overline{J} = \max\left\{ \hat{J} + \sqrt{\frac{2\hat{V}\log(2/\delta_{k,i})}{ni}} + \frac{7R\log(2/\delta_{k,i})}{3(1-\gamma)(ni-1)}, J^\star \right\}$
    $\Delta_{k,i} = \max\{\underline{J} - \lambda\overline{J}, 0\}$ ▷ milestone constraint
  while $\epsilon(ni, \delta_{k,i}) > \frac{\|g_{k,i}\|}{2} + \frac{L\Delta_{k,i}}{\|g_{k,i}\|}$
  $N_k = ni$, $\mathcal{D}_k = \mathcal{D}_{k,i}$ ▷ adaptive batch size
  $J^\star = \max\{\overline{J}, J^\star\}$
  Update policy parameters as $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha\widehat{\nabla} J(\boldsymbol{\theta}_k; \mathcal{D}_k)$
end for

Appendix F: Task specifications

In this "Appendix", we provide detailed descriptions of the control tasks used in the numerical simulations.

F.1 LQR

The LQR is a classical optimal control problem (Dorato et al., 1994). It models the very general task of controlling a set of variables to zero with the minimum effort. Given a state

space $\mathcal{S} \subseteq \mathbb{R}^n$ and an action space $\mathcal{A} \subseteq \mathbb{R}^m$, the next state is a linear function of current state and action:[21]

$$s_{t+1} = As_t + Ba_t, \tag{F99}$$

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$. The reward is quadratic in both state and action:

$$r_{t+1} = s_t^\top C s_t + a_t^\top D a_t, \tag{F100}$$

where $C \in \mathbb{R}^{n \times n}$ and $D \in \mathbb{R}^{m \times m}$ are positive definite matrices. A linear controller is optimal for this task (Dorato et al., 1994) and can be computed in closed form with dynamic-programming techniques. In our experiments, we always consider shallow Gaussian policies of the form:

$$\pi(\cdot|s_t) = \mathcal{N}(\boldsymbol{\theta}^\top s_t, \sigma^2 \mathbb{I}), \tag{F101}$$

where $\boldsymbol{\theta} \in \mathbb{R}^n$ and $\sigma > 0$ can be fixed or learned as an additional policy parameter. This version of LQR with Gaussian policies is also called LQG (Linear-Quadratic Gaussian Regulator, Peters & Schaal 2008). States and actions are clipped in practice when they happen to fall outside $\mathcal{S}$ and $\mathcal{A}$, respectively. We have ignored nonlinearities stemming from this fact.

The LQR problem used in Sect. 7 is 1-dimensional with $\mathcal{S} = \mathcal{A} = [-1, 1]$, $A = B = C = D = 1$.

F.2 Cart-pole

This is the `CartPole-v1` environment from `openai/gym` (Brockman et al., 2016). It has 4-dimensional continuous states and finite (two) actions. The goal is to keep a pole balanced by controlling a cart to which the pole is attached. Reward is +1 for every time-step until the pole falls. We set a maximum episode length of 100. See the official documentation for more details (https://gym.openai.com/envs/CartPole-v1/).

Author contributions All authors—Matteo Papini, Matteo Pirotta, and Marcello Restelli—have given substantial contributions to the conception and design of this work.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. There has been no significant financial support for this work that could have influenced its outcome. M. Papini was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No.~950180).

[21] A zero-mean Gaussian noise is typically added to the next state to model disturbances. However, since we always consider Gaussian policies with fixed standard deviation, we can ignore the system noise without loss of generality. Indeed, from linearity of the next state, said $\overline{a}_t$ the expected action under (F101), $s_{t+1} = As_t + B\overline{a}_t + B\epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I})$. From the property of Gaussians, we can write $\epsilon = \epsilon_a + \epsilon_b$ where $\epsilon_a \sim \mathcal{N}(0, \sigma_a^2 \mathbb{I})$ is from the actual stochasticity of the agent and $\epsilon_b \sim \mathcal{N}(0, \sigma_b^2 B^\dagger)$ is the system noise, which can be subsumed by the policy noise in numerical simulations for simplicity.

Availability of data and material Not applicable.

Declarations

Conflict of interest There are no known conflicts of interest associated with this publication.

Ethics approval: Not applicable.

Consent to participate: Not applicable.

Consent for publication: Not applicable.

Code availability custom code was used for the numerical evaluations and is available at https://github.com/T3p/potion.

References

Abbeel, P., Coates, A., & Ng, A. Y. (2010). Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research, 29*(13), 1608–1639.

Achiam, J., Held, D., Tamar, A., & Abbeel, P. (2017). Constrained policy optimization. *ICML, 70*, 22–31. PMLR.

Agarwal, A., Kakade, S. M., Lee, J. D., & Mahajan, G. (2020). Optimality and approximation with policy gradient methods in Markov decision processes. *COLT, 125*, 64–66. PMLR.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., Mane, D. (2016). Concrete problems in ai safety. arXiv preprint arXiv:1606.06565 .

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve dicult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics, 13*(5), 834–846.

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

Bedi, A.S., Parayil, A., Zhang, J.,Wang, M., & Koppel, A. (2021). On the sample complexity and metastability of heavy-tailed policy search in continuous control. *CoRR*. https://arxiv.org/abs/2106.08414 .

Berkenkamp, F. (2019). Safe exploration in reinforcement learning: Theory and applications in robotics (Unpublished doctoral dissertation). ETH Zurich.

Berkenkamp, F., Turchetta, M., Schoellig, A.P., & Krause, A. (2017). Safe modelbased reinforcement learning with stability guarantees. *NIPS* (pp. 908– 919).

Bertsekas, D. P. (2011). Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications, 9*(3), 310–335.

Bertsekas, D.P., & Shreve, S. (2004). Stochastic optimal control: the discrete- time case.

Bhandari, J., & Russo, D. (2019). Global optimality guarantees for policy gradient methods. *CoRR*. https://arxiv.org/abs/1906.01786.

Bisi, L., Sabbioni, L., Vittori, E., Papini, M., & Restelli, M. (2020). Risk-averse trust region optimization for reward-volatility reduction. *IJCAI* (pp. 4583–4589). ijcai.org.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym.

Castro, D.D., Tamar, A., & Mannor, S. (2012). Policy gradients with variance related risk criteria. ICML. icml.cc / Omnipress.

Chou, P., Maturana, D., & Scherer, S. A. (2017). Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. *ICML, 70*, 834–843. PMLR.

Chow, Y., Nachum, O., Due nez-Guzman, E.A., & Ghavamzadeh, M. (2018). A lyapunov-based approach to safe reinforcement learning. *Neurips* (pp. 8103–8112).
Ciosek, K., & Whiteson, S. (2020). Expected policy gradients for reinforcement learning. *Journal of Machine Learning Research 21*, 52:1-52:51.
Clouse, J. A., & Utgo, P. E. (1992). A teaching method for reinforcement learning. *Machine Learning Proceedings, 1992*, 92–101. Elsevier.
Cohen, A., Yu, L., & Wright, R. (2018). Diverse exploration for fast and safe policy improvement. arXiv preprint arXiv:1802.08331 .
Dalal, G., Dvijotham, K., Vecerk, M., Hester, T., Paduraru, C., & Tassa, Y. (2018). Safe exploration in continuous action spaces. *CoRR*. https://arxiv.org/abs/1801.08757.
Deisenroth, M.P., Neumann, G., & Peters, J., et al. (2013). A survey on policy search for robotics. *Foundations and Trends®in Robotics*, 2 (1-2), 1-142.
Dorato, P., Cerone, V., & Abdallah, C. (1994). Linear-quadratic control: an introduction. Simon & Schuster, Inc.
Duan, Y., Chen, X., Houthooft, R., Schulman, J., & Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. *ICML, 48*, 1329–1338. JMLR.org.
Fruit, R., Lazaric, A., & Pirotta, M. (2019). Regret minimization in infinite-horizon finite markov decision processes. Tutorial at ALT'19. Retrieved from https://rlgammazero.github.io/
Furmston, T., & Barber, D. (2012). A unifying perspective of parametric policy search methods for markov decision processes. Advances in neural information processing systems (pp. 2717-2725).
Garcelon, E., Ghavamzadeh, M., Lazaric, A., & Pirotta, M. (2020). Conservative exploration in reinforcement learning. *AISTATS, 108*, 1431–1441. PMLR.
Garcelon, E., Ghavamzadeh, M., Lazaric, A., & Pirotta, M. (2020b). Improved algorithms for conservative exploration in bandits. AAAI (pp. 3962– 3969). AAAI Press.
Garca, J., & Fernandez, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research 16*, 1437–1480.
Gehring, C., & Precup, D. (2013). Smart exploration in reinforcement learning using absolute temporal difference errors. In *Proceedings of the 2013 international conference on autonomous agents and multi-agent systems* (pp. 1037–1044).
Geibel, P., & Wysotzki, F. (2005). Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research, 24*, 81–108.
Glynn, P.W. (1986). Stochastic approximation for monte carlo optimization. *WSC* (pp. 356–365).
Grimmett, G., & Stirzaker, D. (2020). *Probability and random processes*. Oxford University Press.
Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ICML, 80*, 1856–1865. JMLR.org.
Hans, A., Schneega, D., Schäfer, A.M., & Udluft, S. (2008). Safe exploration for reinforcement learning. *Esann* (pp. 143–148).
Kadota, Y., Kurano, M., & Yasuda, M. (2006). Discounted markov decision processes with utility constraints. *Computers & Mathematics with Applications, 51*(2), 279–284.
Kakade, S. (2001). Optimizing average reward using discounted rewards. Inter- national conference on computational learning theory (pp. 605–615).
Kakade, S. (2002). A natural policy gradient. Advances in neural information processing systems (pp. 1531–1538).
Kakade, S., & Langford, J. (2002). Approximately optimal approximate reinforcement learning..
Kakade, S. M., et al. (2003). *On the sample complexity of reinforcement learning (Unpublished doctoral dissertation)*. England: University of London London.
Kazerouni, A., Ghavamzadeh, M., Abbasi, Y., & Roy, B.V. (2017). Conservative contextual linear bandits. *NIPS* (pp. 3913–3922).
Klenke, A. (2013). *Probability theory: A comprehensive course*. Springer Science & Business Media.
Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research, 32*(11), 1238–1274.
Konda, V.R., & Tsitsiklis, J.N. (1999). Actor-critic algorithms. *NeurIPS* (pp. 1008–1014).
Laroche, R., Trichelair, P., & Des Combes, R.T. (2019). Safe policy improvement with baseline bootstrapping. In *International conference on machine learning* (pp. 3652–3661).
Li, B., & Hoi, S. C. (2014). Online portfolio selection: A survey. *ACM Computing Surveys (CSUR), 46*(3), 35.
Maurer, A., & Pontil, M. (2009). Empirical Bernstein bounds and samplevariance penalization. COLT.
Metelli, A. M., Pirotta, M., Calandriello, D., & Restelli, M. (2021). Safe policy iteration: A monotonically improving approximate policy iteration approach. *Journal of Machine Learning Research, 22*(97), 1–83.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529.
Moldovan, T.M., & Abbeel, P. (2012). Safe exploration in markov decision processes. In *Proceedings of the 29th international conference on international conference on machine learning* (pp. 1451-+1458).
Nesterov, Y. (1998). Introductory lectures on convex programming volume i: Basic course. Lecture notes.
Nesterov, Y. (2013). *Introductory lectures on convex optimization: A basic course (Vol. 87)*. Springer Science & Business Media.
Neu, G., Jonsson, A., & Gomez, V. (2017). A unified view of entropy-regularized markov decision processes. *CoRR*. https://arxiv.org/abs/1705.07798.
Nota, C., & Thomas, P.S. (2020). Is the policy gradient a gradient? AAMAS (pp. 939–947). International Foundation for Autonomous Agents and Multiagent Systems.
Okuda, R., Kajiwara, Y., & Terashima, K. (2014). A survey of technical trend of adas and autonomous driving. Technical papers of 2014 international symposium on VLSI design, automation and test (pp. 1–4).
OpenAI (2018). Openai five. https://blog.openai.com/openai-ve/.
Pajarinen, J., Thai, H.L., Akrour, R., Peters, J., & Neumann, G. (2019). Compatible natural gradient policy search. arXiv preprint arXiv:1902.02823 .
Papini, M., Battistello, A., & Restelli, M. (2020). Balancing learning speed and stability in policy gradient via adaptive exploration. *AISTATS, 108*, 1188–1199. PMLR.
Papini, M., Binaghi, D., Canonaco, G., Pirotta, M., & Restelli, M. (2018). Stochastic variance-reduced policy gradient. *ICML, 80*, 4023–4032. JMLR.org.
Papini, M., Pirotta, M., & Restelli, M. (2017). Adaptive batch size for safe policy gradients. In *Advances in neural information processing systems* (pp. 3591–3600).
Paul, S., Kurin, V., & Whiteson, S. (2019). Fast efficient hyperparameter tuning for policy gradients. CoRR. https://arxiv.org/abs/1902.06583.
Pecka, M., & Svoboda, T. (2014). Safe exploration techniques for reinforcement learning-an overview. In: *International workshop on modelling and simulation for autonomous systems* (pp. 357–375).
Peters, J. (2002). Policy gradient methods for control applications (Tech. Rep.). Technical Report TR-CLMC-2007-1,. University of Southern California.
Peters, J., & Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Networks, 21*(4), 682–697.
Pirotta, M., Restelli, M., & Bascetta, L. (2013). Adaptive step-size for policy gradient methods. *Advances in Neural Information Processing Systems, 26*, 1394–1402.
Pirotta, M., Restelli, M., & Bascetta, L. (2015). Policy gradient in Lipschitz Markov decision processes. *Machine Learning, 100*(2–3), 255–283.
Pirotta, M., Restelli, M., Pecorino, A., & Calandriello, D. (2013). Safe policy iteration. In: *International conference on machine learning* (pp. 307-315).
Puterman, M. L. (2014). *Markov decision processes: Discrete stochastic dynamic programming*. Wiley.
Recht, B. (2019). A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*.
Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics, 22*, 400–407.
Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., & Moritz, P. (2015). Trust region policy optimization. *ICML, 37*, 1889–1897. JMLR.org.
Shamir, O. (2011). A variant of Azuma's inequality for martingales with subGaussian tails. CoRR. https://arxiv.org/abs/1110.2392.
Shani, L., Efroni, Y., & Mannor, S. (2020). Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. AAAI (pp. 5668–5675). AAAI Press.
Shen, Z., Ribeiro, A., Hassani, H., Qian, H., & Mi, C. (2019). Hessian aided policy gradient. *ICML, 97*, 5729–5738. PMLR.
Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science, 362*(6419), 1140–1144.
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.
Sutton, R.S., McAllester, D.A., Singh, S.P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. Advances in neural information processing systems (pp. 1057–1063).

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., . . . & Vanhoucke, V. (2018). Sim-to-real: Learning agile locomotion for quadruped robots. arXiv preprint arXiv:1804.10332 .

Thomas, P. S., da Silva, B. C., Barto, A. G., Giguere, S., Brun, Y., & Brunskill, E. (2019). Preventing undesirable behavior of intelligent machines. *Science, 366*(6468), 999–1004.

Thomas, P. S., Theocharous, G., & Ghavamzadeh, M. (2015). High confidence policy improvement. *ICML, 37*, 2380–2388. JMLR.org.

Tucker, G., Bhupatiraju, S., Gu, S., Turner, R., Ghahramani, Z., & Levine, S. (2018). The mirage of action-dependent baselines in reinforcement learning. In *International conference on machine learning* (pp. 5015–5024).

Turchetta, M., Berkenkamp, F., & Krause, A. (2016). Safe exploration in finite markov decision processes with Gaussian processes. Advances in neural information processing systems (pp. 4312–4320).

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W.M., . . . & Silver, D. (2019). AlphaStar: Mastering the real-time strategy game starCraft II. https://deepmind.com/blog/alphastar -mastering-real-time-strategy-game-starcraft-ii/.

Wagner, P. (2011). A reinterpretation of the policy oscillation phenomenon in approximate policy iteration. Advances in neural information processing systems (pp. 2573–2581).

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8*(3–4), 229–256.

Wu, Y., Shari, R., Lattimore, T., & Szepesvari, C. (2016). Conservative bandits. *ICML, 48*, 1254–1262. JMLR.org.

Xu, P., Gao, F., & Gu, Q. (2020). *Sample efficient policy gradient methods with recursive variance reduction*. ICLR: OpenReview.net.

Yu, J., Aberdeen, D., & Schraudolph, N.N. (2006). Fast online policy gradient learning with SMD gain vector adaptation. *Advances in neural information processing systems* (pp. 1185–1192).

Yuan, H., Lian, X., Liu, J., & Zhou, Y. (2020). Stochastic recursive momentum for policy gradient methods. *CoRR*. https://arxiv.org/abs/2003.04302.

Yuan, R., Gower, R.M., & Lazaric, A. (2021). A general sample complexity analysis of vanilla policy gradient. *CoRR*. https://arxiv.org/abs/2107.11433.

Zhang, J., Kim, J., O'Donoghue, B., & Boyd, S.P. (2020). Sample efficient reinforcement learning with REINFORCE. *CoRR*. https://arxiv.org/abs/2010.11364.

Zhao, T., Hachiya, H., Niu, G., & Sugiyama, M. (2011). Analysis and improvement of policy gradient estimation. *NIPS* (pp. 262–270).