Simultaneous Perturbation Algorithms for Batch Off-Policy Search

Raphael Fonteneau† and Prashanth L.A.⋆

***Abstract*— We propose novel policy search algorithms in the context of off-policy, batch mode reinforcement learning (RL) with continuous state and action spaces. Given a batch collection of trajectories, we perform off-line policy evaluation using an algorithm similar to that in [1]. Using this Monte-Carlo like policy evaluator, we perform policy search in a class of parameterized policies. We propose both first order policy gradient and second order policy Newton algorithms. All our algorithms incorporate simultaneous perturbation estimates for the gradient as well as the Hessian of the cost-to-go vector, since the latter is unknown and only biased estimates are available. We demonstrate their practicality on a simple 1-dimensional continuous state space problem.**

I. Introduction

This paper stands within the field of optimal control in the context of infinite horizon discounted cost Markov decision processes (MDPs) [2]. In particular, we consider the batch mode setting [3], where we are given a set of noisy trajectories of a system without access to any model or simulator of that system. Formally, we are given a set of n samples (also called transitions) $\{(x^l, u^l, c^l, y^l)\}_{l=1}^n$, where, for every $l \in \{1, \ldots, n\}$, the 4-tuple (x^l, u^l, c^l, y^l) denotes the state x^l, the action u^l, a (noisy) cost received in (x^l, u^l) and a (noisy) successor state reached when taking action u^l in state x^l. The samples are generated according to some unknown policy and the objective is to develop a (off-policy) control scheme that attempts to find a near-optimal policy using this batch of samples.

For this purpose, we first parameterize the policy and hence the cost-to-go, denoted by $J^\theta(x_0)$. Here θ is the policy parameter, x_0 is a given initial state and $J^\theta(x_0)$ is the expected cumulative discounted sum of costs under a policy governed by θ (see (1)). Note that the policy parameterization is not constrained to be linear. We develop algorithms that perform descent using estimates of the cost-to-go $J^\theta(x_0)$. For obtaining these estimates from the batch data, we extend a recent algorithm proposed for finite horizon MDPs [1], to the infinite horizon, discounted setting. The advantage of this estimator, henceforth referred to as MFMC, is that it is off-policy in nature, computationally tractable and consistent under Lipschitz assumption on the transition dynamics, cost function and policy. Moreover, it does not require the use of function approximators, but only needs a metric on the state and action spaces.

Being equipped with the MFMC policy evaluator that outputs an estimate of the cost-to-go $J^\theta(x_0)$ for any policy parameter θ, the requirement is for a control scheme that uses these estimated values to update the parameter θ in the negative descent direction. However, closed form expressions of the gradient/Hessian of the cost-to-go are not available and MFMC estimates possess a non-zero bias. To alleviate this, we employ the well-known simultaneous perturbation principle (cf. [4]) to estimate the gradient and Hessian, respectively, of $J^\theta(x_0)$ using estimates from MFMC and propose two first order and two second order algorithms. Our algorithms are based on two popular simultaneous perturbation methods - Simultaneous Perturbation Stochastic Approximation (SPSA) [5] and Smoothed Functional [6].

The first-order algorithms perform gradient descent using either SPSA or SF estimates to update the policy parameter. On the other hand, the second order algorithms incorporate a Newton step by estimating the gradient as well as the Hessian of the cost-to-go $J^\theta(x_0)$ using SPSA or SF. We demonstrate the empirical usefulness of our algorithms on a simple 1-dimensional continuous state space problem.

To the best of our knowledge, the algorithms presented in this paper are the first to solve batch, off-policy stochastic control in continuous state and action spaces without using function approximators for evaluating policies. Our approach only requires (i) a (random) set of trajectories, (ii) metrics on the state and action spaces, and (iii) a set of parameterized policies.

II. Related work

The work presented in this paper mainly relates to two fields of research: batch mode reinforcement learning and policy gradient methods.

Genesis of batch mode RL may be found in [7], where the authors use least-squares techniques in the context of temporal difference (TD) learning methods for estimating the return of control policies. This approach has been extended to the problem of optimal control in [8]. Algorithms similar to value iteration have also been proposed in the batch mode RL setting and the reader is referred to [9] (using kernel approximators), [3] (using ensembles of regression trees) and [10] (using neural networks). More recently, new batch mode RL techniques have been proposed in [11] and this does not require the use of function approximators for policy evaluation. Our policy evaluator is based on the Monte Carlo-like technique proposed in [11].

Policy gradient methods [12] can be seen as a subclass of direct policy search techniques [13] that aim at finding a near-optimal policy within a set of parameterized policies. Actor-critic algorithms are relevant in this context and the reader is referred to [14] and the references therein. The

†Department of Electrical Engineering and Computer Science, University of Liège, 4000 Liège, Belgium raphael.fonteneau@ulg.ac.be

⋆SEQUEL project, INRIA Lille - Nord Europe, 59650 Villeneuve d'Ascq, FRANCE prashanth.la@inria.fr

actor-critic algorithms mentioned above work in an approximate dynamic programming setting. In other words, owing to the high-dimensional state spaces encountered often in practice, the algorithms approximate the value function with a (usually linear) function approximation architecture. Thus, the quality of the policy obtained by the algorithms are contingent upon the quality of the approximation architecture and selection of approximation architecture is in itself a hot topic of research in RL. In contrast, we do not employ function approximators and instead, use a Monte-Carlo like policy evaluation scheme to develop policy gradient/Newton algorithms.

III. The Setting

We consider a stochastic discrete-time system with state space $\mathcal{X} \subset \mathbb{R}^{d_\mathcal{X}}$, $d_\mathcal{X} \in \mathbb{N}$ and action space $\mathcal{U} \subset \mathbb{R}^{d_\mathcal{U}}$, $d_\mathcal{U} \in \mathbb{N}$. The dynamics of this system is governed by:

$$x_{t+1} = f\left(x_t, u_t, w_t\right), \qquad \forall t \in \mathbb{N}$$

where x_t and u_t denote the state and action at time $t \in \mathbb{N}$, while $w_t \in \mathcal{W}$ denotes a random disturbance drawn according to a probability distribution $p_\mathcal{W}(\cdot)$. Each system transition from time t to $t+1$ incurs an instantaneous cost $c\left(x_t, u_t, w_t\right)$. We assume that the cost function is bounded and translated into the interval $[0, 1]$.

Let $\mu : \mathcal{X} \to \mathcal{U}$ be a control policy that maps states to actions. In this paper, we consider a class of policies parameterized by $\theta \in \Theta$, i.e., $\mu^\theta : \mathcal{X} \to \mathcal{U}$. We assume that Θ is a compact and convex subset of $\mathbb{R}^N, N \in \mathbb{N}$. Since a policy μ is identifiable with its parameter θ, we shall use them interchangeably in the paper.

The classical performance criterion for evaluating a policy μ is its (expected) cost-to-go, which is the discounted sum of costs that an agent receives, while starting from a given initial state x_0 and then following a policy μ, i.e.,

$$J^\mu(x_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c(x_t, \mu(x_t), w_t) \mid x_0, \mu\right], \qquad (1)$$

where $x_{t+1} = f(x_t, \mu(x_t), w_t)$ and $w_t \sim p_\mathcal{W}(\cdot), \forall t \in \mathbb{N}$.

In the above, $\gamma \in (0, 1)$ denotes the discount factor.

In a batch mode RL setting, the objective is to find a policy that minimizes the cost-to-go $J^\mu(x_0)$. However, the problem is challenging since the functions f, c and $p_\mathcal{W}(\cdot)$ are unknown (not even accessible to simulation). Instead, we are provided with a batch collection of $n \in \mathbb{N} \setminus \{0\}$ one-step system transitions $\mathcal{F}_n$, defined as

$$\mathcal{F}_n = \left\{\left(x^l, u^l, c^l, y^l\right)\right\}_{l=1}^n,$$

where $c^l := c\left(x^l, u^l, w^l\right)$ is the instantaneous cost and $y^l := f\left(x^l, u^l, w^l\right)$ is the next state. Here, both c^l and y^l are governed by the disturbance sequence $w^l \sim p_\mathcal{W}(\cdot)$, for all $l \in \{1, \ldots, n\}$.

The algorithms that we present next incrementally update the policy parameter θ in the negative descent direction using either the gradient or Hessian of $J^\theta(x_0)$. The underlying policy evaluator that provides the cost-to-go inputs for any θ is based on MFMC, while the gradient/Hessian estimates are based on the principle of simultaneous perturbation [4].

IV. Algorithm Structure

In a deterministic optimization setting, an algorithm attempting to find the minima of the cost-to-go $J^\theta(x_0)$ would update the policy parameter in the descent direction as follows:

$$\theta_i(t+1) = \Gamma_i(\theta(t) - a(t) A_t^{-1} \nabla_\theta J^\theta(x_0)), \qquad (2)$$

where A_t is a positive definite matrix and $a(t)$ is a step-size that satisfies standard stochastic approximation conditions: $\sum_t a(t) = \infty$ and $\sum_t a(t)^2 < \infty$. Further, $\Gamma(\theta) = (\Gamma_1(\theta_1), \ldots, \Gamma_N(\theta_N))$ is a projection operator that projects the iterate θ to the nearest point in the set $\Theta \in \mathbb{R}^N$. The projection is necessary to ensure stability of the iterate θ and hence the overall convergence of the scheme (2).

For the purpose of obtaining the estimate of the cost-to-go vector $J^\theta(x_0)$ for any θ, we adapt the MFMC (for Model-Free Monte Carlo) estimator proposed in [1]) to our (infinite-horizon discounted) setting[1]. The MFMC estimator works by rebuilding (from one-step transitions taken in $\mathcal{F}_n$) artificial trajectories that emulate the trajectories that could be obtained if one could do Monte Carlo simulations. An estimate $\hat{J}^\theta$ of the cost-to-go J^θ is obtained by averaging the cumulative discounted cost of the rebuilt artificial trajectories.

Using the estimates of MFMC, it is necessary to build a higher-level control loop to update the parameter θ in the descent direction as given by (2). However, closed form expressions of the gradient and the Hessian of $J^\theta(x_0)$ are not available and instead, we only have (biased) estimates of $J^\theta(x_0)$ from MFMC. Thus, the requirement is for a simulation-optimization scheme that approximates the gradient/Hessian of $J^\theta(x_0)$ using estimates from MFMC.

Simultaneous perturbation methods [4] are well-known simulation optimization schemes that perturb the parameter uniformly in each direction in order to find the minima of a function observable only via simulation. These methods are attractive since they require only two simulations irrespective of the parameter dimension. Our algorithms are based on two popular simultaneous perturbation methods - Simultaneous Perturbation Stochastic Approximation (SPSA) [5] and Smoothed Functional [6]. The algorithms that we propose mainly differ in the choice of A_t in (2) and the specific simultaneous perturbation method used:

MCPG-SPSA. Here $A_t = I$ (identity matrix). Thus, MCPG-SPSA is a first order scheme that updates the policy parameter in the descent direction. Further, the gradient $\nabla_\theta J^\theta(x_0)$ is estimated using SPSA.

MCPG-SF. This is the Smoothed functional (SF) variant of MCPG-SPSA.

[1]Besides being adapted to the batch mode setting, the MFMC estimator also has the advantage of having a linear computational complexity and consistency properties (see Section V).

Algorithm 1 Structure of our algorithms.

Input: θ_0, initial parameter vector; $\delta > 0$; Δ;
MFMC(θ), the model free Monte Carlo like policy evaluator
for $t = 0, 1, 2, \ldots$ **do**
  Call MFMC($\theta(t) + p_1(t)$)
  Call MFMC($\theta(t) + p_2(t)$)
  Compute $\theta(t+1)$ (Algorithm-specific)
end for
Return $\theta(t)$

MCPN-SPSA. Here $A_t = \nabla^2 J^\theta(x_0)$, i.e., the Hessian of the cost-to-go. Thus, MCPN is a second order scheme that update the policy parameter using a Newton step. Further, the gradient/Hessian are estimated using SPSA.

MCPN-SF. This is the SF variant of MCPN-SPSA.

As illustrated in Fig. 1, our algorithms operate on the principle of simultaneous perturbation and involve the following steps:

(i) estimate, using MFMC, the cost-to-go for two perturbation sequences $\theta(t) + p_1(t)$ and $\theta(t) + p_2(t)$;

(ii) obtain the gradient/Hessian estimates (see (4)–(8)) from the cost-to-go values $J^{\theta(t)+p_1(t)}(x_0)$ and $J^{\theta(t)+p_2(t)}(x_0)$;

(iii) update the parameter θ in the descent direction using the gradient/Hessian estimates obtained above.

The choice of perturbation sequences $p_1(t)$ and $p_2(t)$ is specific to the algorithm (see Sections VI and VII).

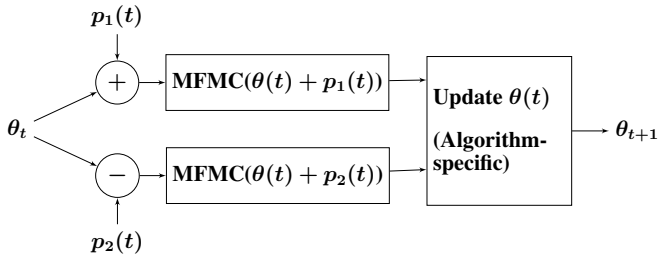


Fig. 1. Overall flow of simultaneous perturbation algorithms.

Remark 1: From a theoretical standpoint, the setting considered here is of deterministic optimization and the estimates from MFMC have non-zero, albeit bounded, non-stochastic bias for a given sample of transitions. This is unlike earlier work on SPSA, which mostly feature a stochastic noise component that is zero-mean. While we establish bounds on the bias of MFMC (see Lemmas 1 and 2 in [15]), it is a challenge to establish asymptotic convergence and in this regard, we note the difficulties involved in Section VIII.

V. MFMC Estimation of a Policy

For the purpose of policy evaluation given a batch of samples, we adapt the Model-free Monte Carlo estimator (MFMC) algorithm, proposed in [1], to an infinite horizon discounted setting.

From a sample of transitions $\mathcal{F}_n$, the MFMC estimator rebuilds $p \in \mathbb{N} \setminus \{0\}$ (truncated) artificial trajectories. These artificial trajectories are used as approximations of p trajectories that could be generated by simulating the policy μ^θ we want to evaluate. The final MFMC estimate $\hat{J}^\theta(x_0)$ is obtained by averaging the cumulative discounted costs over these truncated artificial trajectories.

The trajectories here are rebuilt in a manner similar to the procedure outlined in [1]. However, in our (infinite horizon) setting, the horizon needs to be truncated for rebuilding the trajectories. To this end, we introduce a truncation parameter T that defines the length of the rebuilt trajectories. To limit the looseness induced by such a truncation, the value of the parameter T should be chosen as a function of the discount factor γ, for instance, $T = \Omega\left(\frac{1}{1-\gamma}\right)$. The MFMC estimation

Algorithm 2 MFMC algorithm.

Input: $\mathcal{F}_n, \mu^\theta(.,.), x_0, d(.,.), T, p$
$\mathcal{G}$: current set of not yet used one-step transitions in $\mathcal{F}_n$;
Initially, $\mathcal{G} \leftarrow \mathcal{F}_n$;
for $i = 1$ to p **do**
  $t \leftarrow 0$; $x_t^i \leftarrow x_0$;
  while $t < T$ **do**
    $u_t^i \leftarrow \mu^\theta\left(x_t^i\right)$;
    $\mathcal{H} \leftarrow \underset{(x,u,c,y)\in\mathcal{G}}{\arg\min} \; d\left((x,u),\left(x_t^i,u_t^i\right)\right)$;
    $l_t^i \leftarrow$ lowest index in $\mathcal{F}_n$ of the transitions that belong to $\mathcal{H}$;
    $t \leftarrow t+1$; $x_t^i \leftarrow y^{l_t^i}$;
    $\mathcal{G} \leftarrow \mathcal{G} \setminus \left\{\left(x^{l_t^i}, u^{l_t^i}, c^{l_t^i}, y^{l_t^i}\right)\right\}$;
  end while
end for
Return $\hat{J}^\theta(x_0) = \frac{1}{p}\sum_{i=1}^{p}\sum_{t=0}^{T-1}\gamma^t c^{l_t^i}$.

can be computed using the algorithm provided in Algorithm 2.

Definition 1 (Model-free Monte Carlo Estimator):

$$\hat{J}^\theta(x_0) = \frac{1}{p}\sum_{i=1}^{p}\sum_{t=0}^{T-1}\gamma^t c^{l_t^i}.$$

where $\left\{l_t^i\right\}_{i=1,t=0}^{i=p,t=T-1}$ denotes the set of indices of the transitions selected by the MFMC algorithm (see Algorithm 2).

Note that the computation of the MFMC estimator $\hat{J}^\theta(x_0)$ has a linear complexity with respect to the cardinality n of $\mathcal{F}_n$, the number of artificial trajectories p and the optimization horizon T.

Remark 2: Through Lemmas 1 and 2 in [15], we bound the distance between the MFMC estimate $\hat{J}^\theta(x_0)$ and the true cost-to-go $J^\theta(x_0)$ in expectation and high probability, respectively.

VI. First order algorithms

A. Gradient estimates

SPSA based estimation of the gradient of the cost-to-go, originally proposed in [5], is illustrated as follows: For the

simple case of a scalar parameter θ,

$$\frac{dJ^\theta}{d\theta} \approx \left(\frac{J^{\theta+\delta} - J^\theta}{\delta} \right). \tag{3}$$

The correctness of the above estimate, in the limit as $\delta \to 0$, can be seen by first order Taylor series expansions of $J^{\theta+\delta}$ and $J^{\theta-\delta}$ around θ using a Taylor expansion. The above idea of simultaneous perturbation can be extended to a vector-valued parameter θ by perturbing each co-ordinate of θ uniformly using Rademacher random variables. The resulting SPSA based estimate of the gradient $\nabla_\theta J^\theta(x_0)$ is as follows:

$$\nabla_{\theta_i} J^\theta(x_0) \approx \frac{J^{\theta+\delta\Delta}(x_0) - J^{\theta-\delta\Delta}(x_0)}{2\delta\Delta_i}, \tag{4}$$

where $\Delta = (\Delta_1, \ldots, \Delta_N)^T$ with each Δ_i being Rademacher random variables.

SF is another popular simultaneous perturbation scheme, proposed first in [6]. The gradient of the cost-to-go is estimated via the SF approach as follows:

$$\nabla_{\theta_i} J^\theta(x_0) \approx \frac{\Delta_i}{\delta} \left(J^{\theta+\delta\Delta}(x_0) - J^{\theta-\delta\Delta}(x_0) \right), \tag{5}$$

where Δ is a $(|N|)$-vector of independent $\mathcal{N}(0,1)$ random variables. The main idea in proving the correctness of (5) in the limit as $\delta \to 0$, is to convolve the gradient of the cost-to-go $J^\theta(x_0)$ with a Gaussian density function, apply an integration-by-parts argument and then observe that it is equivalent to convolving the gradient of Gaussian density function with the cost-to-go (see Chapter 6 of [4] for a detailed description).

B. MCPG-SPSA and MCPG-SF algorithms

On the basis of the gradient estimates in (4)–(5), the SPSA and SF variants update the policy parameter θ as follows: For all $t \geq 1$, update

$$\textbf{SPSA: } \theta_i(t+1) = \Gamma_i \Bigg(\theta_i(t) - a(t) \frac{\hat{J}^{\theta(t)+\delta\Delta(t)}(x_0) - \hat{J}^{\theta(t)-\delta\Delta(t)}(x_0)}{2\delta\Delta_i(t)} \Bigg), \tag{6}$$

$$\textbf{SF: } \theta_i(t+1) = \Gamma_i \Bigg(\theta_i(t) - a(t) \frac{\Delta_i(t)}{2\delta} (\hat{J}^{\theta(t)+\delta\Delta(t)}(x_0) - \hat{J}^{\theta(t)-\delta\Delta(t)}(x_0)) \Bigg), \tag{7}$$

for all $i = 1, 2, \ldots, N$. In the above,
(i) $\delta > 0$ is a small fixed constant and $\Delta(t)$ is a N-vector of independent Rademacher random variables for SPSA and standard Gaussian random variables for SF;
(ii) $\hat{J}^{\theta(t)+\delta\Delta(t)}(x_0)$ and $\hat{J}^{\theta(t)-\delta\Delta(t)}(x_0)$ are the MFMC policy evaluator's estimates of the cost-to-go corresponding to the parameters $\theta + \delta\Delta$ and $\theta - \delta\Delta$, respectively.
(iii) $\Gamma(\theta) = (\Gamma_1(\theta_1), \ldots, \Gamma_N(\theta_N))^T$ is an operator that projects the iterate θ to the closest point in a compact and convex set $\Theta \in \mathbb{R}^N$;
(iv) $\{a(t), t \geq 1\}$ is a step-size sequence that satisfies the standard stochastic approximation conditions.

VII. Second order algorithms

For the second order methods, we also need an estimate of the Hessian $\nabla^2_\theta J^\theta(x_0)$, in addition to the gradient.

A. Hessian estimates

SPSA based estimate of the Hessian $\nabla^2_\theta J^\theta(x_0)$ is as follows:

$$\nabla^2_{\theta_i} J^\theta(x_0) \approx \frac{J^{\theta+\delta\Delta+\delta\widehat{\Delta}}(x_0) - J^{\theta+\delta\Delta}(x_0)}{\delta^2 \Delta_i \widehat{\Delta}_i}, \tag{8}$$

where Δ and $\widehat{\Delta}$ represent N-vectors of Rademacher random variables.

SF based estimate of the Hessian $\nabla^2_\theta J^\theta(x_0)$ is as follows:

$$\nabla^2_{\theta_i} J^\theta(x_0) \approx \frac{1}{\delta^2} \bar{H}(\Delta) \big(J^{\theta+\delta\Delta}(x_0) + J^{\theta-\delta\Delta}(x_0) \big), \tag{9}$$

where Δ is a N vector of independent Gaussian $\mathcal{N}(0,1)$ random variables and $\bar{H}(\Delta)$ is a $N \times N$ matrix defined as

$$\bar{H}(\Delta) \stackrel{\triangle}{=} \begin{bmatrix} (\Delta_1^2 - 1) & \Delta_1\Delta_2 & \cdots & \Delta_1\Delta_N \\ \Delta_2\Delta_1 & (\Delta_2^2 - 1) & \cdots & \Delta_2\Delta_N \\ \cdots & \cdots & \cdots & \cdots \\ \Delta_N\Delta_1 & \Delta_N\Delta_2 & \cdots & (\Delta_N^2 - 1) \end{bmatrix}. \tag{10}$$

The reader is referred to Chapters 7 and 8 in [4] for a proof of correctness of the above Hessian estimates.

B. MCPN-SPSA and MCPN-SF algorithms

Let $H(t) = [H_{i,j}(t)]_{i=1,j=1}^{|N|,|N|}$ denote the estimate of the Hessian w.r.t. θ of the cost-to-go $J^\theta(x_0)$ at instant t, with $H(0) = \omega I$ for some $\omega > 0$. On the basis of (8), MCPN-SPSA would estimate the individual components $H_{i,j}(t)$ as follows: For all $t \geq 1$, $i, j \in \{1, \ldots, N\}$, $i \leq j$, update

$$H_{i,j}(t+1) = H_{i,j}(t) + a(t) \Bigg(\frac{\hat{J}^{\theta(t)+\delta\Delta(t)+\delta\widehat{\Delta}(t)}(x_0) - \hat{J}^{\theta(t)+\delta\Delta(t)}(x_0)}{\delta^2 \Delta_j(t) \widehat{\Delta}_i(t)} - H_{i,j}(t) \Bigg),$$

and for $i > j$, set $H_{i,j}(t+1) = H_{j,i}(t+1)$. In the above, $\delta > 0$ is a small fixed constant and $\Delta(t)$ and $\hat{\Delta}(t)$ are N vectors of Rademacher random variables. Now form the Hessian inverse matrix $M(t) = \Upsilon(H(t))^{-1}$. The operator $\Upsilon(\cdot)$ ensures that the Hessian estimates stay within the set of positive definite and symmetric matrices. This is a standard requirement in second-order methods (See [16] for one possible definition of $\Upsilon(\cdot)$). Using these quantities, MCPN-SPSA updates the parameter θ along a descent direction as follows: $\forall t \geq 1$,

$$\theta_i(t+1) = \Gamma_i \Bigg(\theta_i(t) - a(t) \sum_{j=1}^{N} M_{i,j}(t) \frac{\hat{J}^{\theta(t)+\delta\Delta(t)}(x_0) - \hat{J}^{\theta(t)-\delta\Delta(t)}(x_0)}{2\delta\Delta_j(t)} \Bigg). \tag{11}$$

Along similar lines, using (9), the SF variant of the above algorithm would update the Hessian estimate as follows: For

all $t \geq 1$, $i, j, k \in \{1, \ldots, N\}$, $j \leq k$, update

$$H_{i,i}(t+1) = H_{i,i}(t) + a(t)\Bigg(\frac{(\Delta_i^2(t)-1)}{\delta^2}(\hat{J}^{\theta(t)+\delta\Delta(t)}(x_0) + \hat{J}^{\theta(t)-\delta\Delta(t)}(x_0)) - H_{i,j}(t)\Bigg), \quad (12)$$

$$H_{j,k}(t+1) = H_{j,k}(t) + a(t)\Bigg(\frac{\Delta_i(t)\Delta_j(t)}{\delta^2}(\hat{J}^{\theta(t)+\delta\Delta(t)}(x_0) + \hat{J}^{\theta(t)-\delta\Delta(t)}(x_0)) - H_{j,k}(t)\Bigg), \quad (13)$$

and for $j > k$, we set $H_{j,k}(t+1) = H_{k,j}(t+1)$. In the above, $\Delta(t)$ is a N vector of independent Gaussian $\mathcal{N}(0,1)$ random variables. As before, form the Hessian estimate matrix $H(t)$ and its inverse $M(t) = \Upsilon(H(t))^{-1}$. Then, the policy parameter θ is then updated as follows: $\forall t \geq 1$,

$$\theta_i(t+1) = \Gamma_i\Bigg(\theta_i(t) - a(t)\sum_{j=1}^{N} M_{i,j}(t)\Delta_j(t)\frac{(\hat{J}^{\theta(t)+\delta\Delta(t)}(x_0) - \hat{J}^{\theta(t)-\delta\Delta(t)}(x_0))}{2\delta}\Bigg). \quad (14)$$

Remark 3: A computationally efficient alternative to inverting the Hessian H is to use the Woodbury's identity. See Section 6.2 of [15] for a detailed description.

VIII. Notes on Convergence

In this section, we describe the difficulty in establishing the asymptotic convergence for the MCPG-SPSA algorithm - a difficulty common to all our algorithms. An important step in the analysis is to prove that the bias in the MFMC estimator contributes a asymptotically negligible term to the θ-recursion (6). In other words, it is required to show that (6) is asymptotically equivalent to the following in the sense that the difference between the two updates is $o(1)$:

$$\theta_i(t+1) = \Gamma_i\Bigg(\theta_i(t) - a(t)\frac{J^{\theta(t)+\delta\Delta(t)}(x_0) - J^{\theta(t)-\delta\Delta(t)}(x_0)}{2\delta\Delta_i(t)}\Bigg). \quad (15)$$

As a first step towards establishing this equivalence, we first re-write the θ-update in (6) as follows:

$$\theta_i(t+1) = \Gamma_i\Bigg(\theta_i(t) - a(t)\frac{J^{\theta(t)+\delta\Delta(t)}(x_0) - J^{\theta(t)-\delta\Delta(t)}(x_0)}{2\delta\Delta_i(t)} + a(t)\xi(t)\Bigg),$$

where $\xi(t) = \dfrac{\epsilon^{\theta(t)+\delta\Delta(t)} - \epsilon^{\theta(t)-\delta\Delta(t)}}{2\delta\Delta_i(t)}$. In the above, we have used the fact MFMC returns an estimate $\hat{J}^\theta(x_0) = J^\theta(x_0) + \epsilon^\theta$, with ϵ^θ denoting the bias.

Let $\zeta(t) = \sum_{s=0}^{t} a(s)\xi_{s+1}$. Then, a critical requirement for establishing the equivalence of (6) with (15) is the following condition:

$$\sup_{s \geq 0}(\zeta(t+s) - \zeta(t)) \to 0 \text{ as } t \to \infty.$$

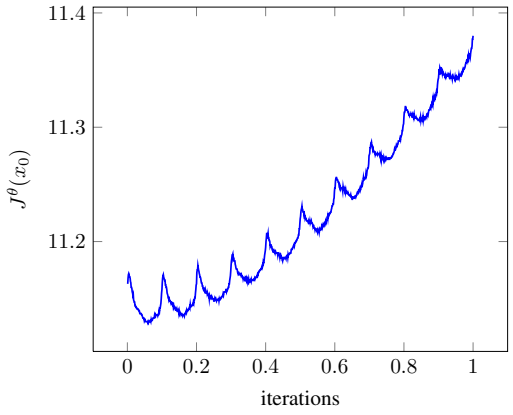


Fig. 2. $J^\theta(x_0)$ vs. θ. Note that the global minimum is $\theta_{min} = 0.06$.

While the bias ϵ^θ of MFMC can be bounded (see Lemma 2 in [15]), it is difficult to ensure that the above condition holds.

Assuming that the bias is indeed asymptotically negligible, the asymptotic convergence of MCPG can be established in a straightforward manner. In particular, using the ordinary differential equation (ODE) approach [17], it can be shown that (15) is a discretization (and hence converges to the equilibria) of the following ODE:

$$\dot{\theta} = \bar{\Gamma}\left(\nabla_\theta J^\theta(x_0)\right), \quad (16)$$

where $\bar{\Gamma}$ is a projection operator that ensures θ evolving according to (16) remains bounded. The reader is referred to Section 7 in [15] for the precise convergence claims and the associated proofs.

IX. Numerical Illustration

We consider the 1-dimensional system ruled by the following dynamics:

$$f(x,u,w) = \text{sinc}(10 * (x+u+w)), \text{ where}$$
$$\text{sinc}(x) = \sin(\pi x)/(\pi x).$$

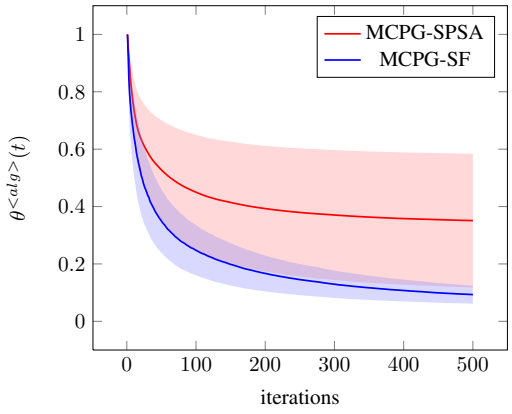
The cost function is defined as follows:

$$c(x,u,w) = -\frac{1}{2\pi}\exp\left(-\frac{x^2+u^2}{2} + w\right).$$

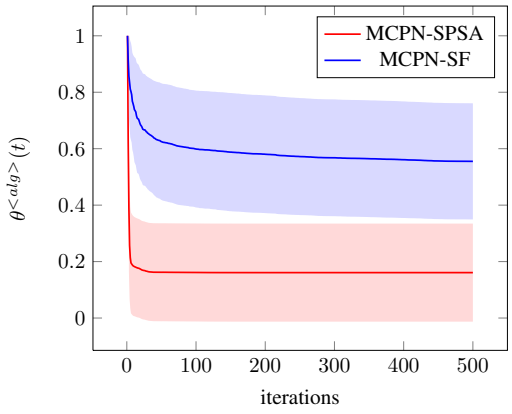
We consider a class of linearly parameterized policies:

$$\mu^\theta(x) = \theta x, \quad \forall \theta \in [0,1].$$

The disturbances are drawn according to a uniform distribution between $[-\frac{\epsilon}{2}, \frac{\epsilon}{2}]$ with $\epsilon = 0.01$. The initial state of the system is fixed to $x_0 = -1$ and the discount factor is set to $\gamma = 0.95$. The truncation of artificial trajectories is set to $T = \frac{1}{1-\gamma} = 20$, and the number of artificial trajectories rebuilt by the MFMC estimator is set to $p = \lceil \ln(n/T) \rceil$. We give in Figure 2 a plot of the evolution of the expected return $J^\theta(x_0)$ as a function of θ (obtained through extensive Monte Carlo simulations). We observe that the expected cost-to-go $J^\theta(x_0)$ is minimized for values of θ around 0.06.



(a) MCPG-SPSA vs. MCPG-SF



(b) MCPN-SPSA vs. MCPN-SF

Fig. 3. Empirical illustration of the MCPG and MCPN algorithms on an academic benchmark.

In order to observe the impact of the randomness of the set of transitions (induced by the disturbances) on the algorithms, we generate 50 samples of transitions $\mathcal{F}_n^1, \ldots, \mathcal{F}_n^{50}$, each sample containing $n = 200$ transitions. For each set $\mathcal{F}_n^i, i = 1 \ldots 50$, the set of state-action pairs $\mathcal{P}_n = \{(x^l, u^l)\}_{l=1}^n$ is the same and generated deterministically from a grid, i.e. $\mathcal{P}_n = \{(-1 + 2 * i/\sigma, -1 + 2 * j/\sigma)\}_{i,j=0}^{\sigma-1}$ with $\sigma = \lfloor \sqrt{n} \rfloor$. The randomness of each set $\mathcal{F}_n^i$ comes from the disturbances $w^l \quad l = 1 \ldots n$ along which transitions are generated.

Then, for each sample $\mathcal{F}_n^i$, we run all the four algorithms - MCPG-SPSA, MCPG-SF, MCPN-SPSA and MCPN-SF - for 500 iterations. This generates the sequences $\left(\theta^{i,<alg>}(t)\right)_t$, where $< alg >$ denotes the algorithm. For each algorithm run, we set $\delta = 0.1$ and the step-size $a(t) = \frac{1}{t}$, for all t. Further, the operator Γ projects $\theta(t)$ into the interval $[0, 1]$, while the Hessian operator Υ projects into $[0.1, \infty)$.

Figure 3 presents the average evolution of the parameter sequence in each of the 50 runs for all the algorithms (bands around the average curves represent 95% confidence intervals). From these plots, we observe that the MCPG-SF approach outperforms the other algorithms on this academic benchmark, with a much lower variance and higher precision.

X. Conclusions

We proposed novel policy search algorithms in a batch, off-policy setting. All these algorithms incorporate simultaneous perturbation estimates for the gradient as well as the Hessian of the cost-to-go vector, since the latter is unknown and only biased estimates are available. We proposed both first order policy gradient as well as second order policy Newton algorithms, using both SPSA as well as SF simultaneous perturbation schemes. We noted certain difficulties in establishing asymptotic convergence of the proposed algorithms, owing to the non-stochastic (and non-zero) bias of the MFMC policy evaluation scheme. As a future direction, we plan to investigate conditions under which the bias of MFMC is asymptotically negligible.

Acknowledgements

The first author was supported by a post-doctoral fellowship of the F.R.S. - FNRS (Belgian Funds for Scientific Research). The second author would like to thank the European Community's Seventh Framework Programme (FP7/2007 – 2013) under grant agreement n^o 270327 for funding the research leading to these results.

References

[1] R. Fonteneau, S. Murphy, L. Wehenkel, and D. Ernst, "Model-free Monte Carlo–like policy evaluation," in *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2010, pp. 217–224.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, May 1996.

[3] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, pp. 503–556, 2005.

[4] S. Bhatnagar, H. L. Prasad, and L. A. Prashanth, *Stochastic Recursive Algorithms for Optimization*. Springer, 2013, vol. 434.

[5] J. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.

[6] V. Katkovnik and Y. Kulchitsky, "Convergence of a class of random search algorithms," *Automatic Remote Control*, vol. 8, pp. 81–87, 1972.

[7] S. Bradtke and A. Barto, "Linear least-squares algorithms for temporal difference learning," *Machine Learning*, vol. 22, pp. 33–57, 1996.

[8] M. Lagoudakis and R. Parr, "Least-squares policy iteration," *Jounal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[9] D. Ormoneit and S. Sen, "Kernel-based reinforcement learning," *Machine Learning*, vol. 49, no. 2-3, pp. 161–178, 2002.

[10] M. Riedmiller, "Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method," in *European Conference on Machine Learning*, 2005, pp. 317–328.

[11] R. Fonteneau, S. Murphy, L. Wehenkel, and D. Ernst, "Batch mode reinforcement learning based on the synthesis of artificial trajectories," *Annals of Operations Research*, vol. 208, pp. 383–416, 2013.

[12] P. L. Bartlett and J. Baxter, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, vol. 15, pp. 319–350, 2001.

[13] J. Schmidhuber and J. Zhao, "Direct policy search and uncertain policy evaluation," In AAAI Spring Symposium on Search under Uncertain and Incomplete Information, Stanford Univ, Tech. Rep., 1998.

[14] S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor-critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.

[15] R. Fonteneau and L. A. Prashanth, "Simultaneous Perturbation Algorithms for Batch Off-Policy Search," *arXiv preprint arXiv:1403.4514*, 2014.

[16] P. Gill, W. Murray, and M. Wright, *Practical Optimization*. Academic press, 1981.

[17] V. Borkar, *Stochastic Approximation: a Dynamical Systems Viewpoint*. Cambridge University Press, 2008.