Delft University of Technology

Delft Center for Systems and Control

Technical report 10-009

Online least-squares policy iteration for reinforcement learning control*

L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška

If you want to cite this report, please use the following reference instead:

L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, "Online least-squares policy iteration for reinforcement learning control," *Proceedings of the 2010 American Control Conference*, Baltimore, Maryland, pp. 486–491, June–July 2010.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://pub.bartdeschutter.org/abs/10_009.html

Online least-squares policy iteration for reinforcement learning control

Lucian Buşoniu, Damien Ernst, Bart De Schutter, and Robert Babuška

***Abstract*— Reinforcement learning is a promising paradigm for learning optimal control. We consider policy iteration (PI) algorithms for reinforcement learning, which iteratively *evaluate* and *improve* control policies. State-of-the-art, least-squares techniques for policy evaluation are sample-efficient and have relaxed convergence requirements. However, they are typically used in offline PI, whereas a central goal of reinforcement learning is to develop *online* algorithms. Therefore, we propose an online PI algorithm that evaluates policies with the so-called least-squares temporal difference for Q-functions (LSTD-Q). The crucial difference between this *online least-squares policy iteration* (LSPI) algorithm and its offline counterpart is that, in the online case, policy improvements must be performed once every few state transitions, using only an incomplete evaluation of the current policy. In an extensive experimental evaluation, online LSPI is found to work well for a wide range of its parameters, and to learn successfully in a real-time example. Online LSPI also compares favorably with offline LSPI and with a different flavor of online PI, which instead of LSTD-Q employs another least-squares method for policy evaluation.**

I. Introduction

Reinforcement learning (RL) algorithms [1], [2] can in principle solve nonlinear, stochastic optimal control problems without using a model. A RL controller learns how to control the process by interacting with it. The immediate performance is measured by a scalar reward, and the goal is to find an optimal control policy that maximizes the value function, i.e., the cumulative long-term reward as a function of the process state and possibly of the control action. A value function that depends on the state and action is called a Q-function. RL solutions cannot always be represented exactly, and approximation must be used in general. State-of-the-art RL algorithms use weighted summations of basis functions to approximate the value function, and least-squares techniques to find the weights [3]–[6].

This paper concerns approximate policy iteration (PI), which in every iteration *evaluates* the current policy, by computing its approximate value function, and then finds a new, *improved* policy using this value function. Least-squares techniques for policy evaluation have relaxed convergence requirements and approach their solution quickly as the number of samples increases [7], [8]. They have mainly been employed in offline PI, which improves the policy only after an accurate value function has been found using many samples. This approach is feasible in the offline case, because only the performance of the final policy is important. However, a central goal of RL is to develop algorithms that learn online, in which case the performance should improve once every few transition samples.

Therefore, in this paper, we propose and empirically evaluate an *online* PI algorithm that evaluates policies with the least-squares temporal difference for Q-functions (LSTD-Q). We call this algorithm *online LSPI*, after its offline counterpart, called LSPI [5]. The crucial difference from the offline case is that policy improvements must be performed once every few samples, before an accurate evaluation of the current policy can be completed. Such policy improvements are called 'optimistic' [2]. Moreover, online LSPI has to collect its own samples, which makes exploration necessary.

Many existing online PI algorithms rely on gradient-based policy evaluation [1], [9], which is less efficient than least-squares policy evaluation. While using least-squares methods online has been proposed [2], [5], little is known about how they behave in practice. In particular, to the best of our knowledge, the combination of LSTD-Q with optimistic policy updates has not been studied yet. A competing algorithm to LSTD-Q, called least-squares policy evaluation for Q-functions (LSPE-Q) [4], has previously been used in optimistic PI [10]. We focus on LSTD-Q, while also providing a comparison with LSPE-Q. The authors of [11] evaluate LSPI with online sample collection, focusing on the issue of exploration. However, unlike our online LSPI variant, their method does not perform optimistic policy updates, but fully executes LSPI between consecutive sample-collection episodes; this incurs large computational costs.

After describing the necessary theoretical background in Section II, we introduce online LSPI in Section III. Section IV provides an extensive experimental evaluation of online LSPI, including real-time control results, for the problem of swinging up an underactuated inverted pendulum. Section V concludes the paper.

II. The RL problem. Approximate PI with least-squares policy evaluation

Consider a Markov decision process with state space X and action space U. Assume for now that X and U are countable. The probability that the next state x_{k+1} is reached after action u_k is taken in state x_k is $f(x_k, u_k, x_{k+1})$, where $f : X \times U \times X \to [0, 1]$ is the transition probability function. After the transition to x_{k+1}, a reward $r_{k+1} = \rho(x_k, u_k, x_{k+1})$ is received, where $\rho : X \times U \times X \to \mathbb{R}$ is the reward function. The expected infinite-horizon discounted return of initial state x_0 under a policy $h : X \to U$ is:

$$R^h(x_0) = \lim_{K \to \infty} \mathrm{E}_{x_{k+1} \sim f(x_k, h(x_k), \cdot)} \left\{ \sum_{k=0}^{K} \gamma^k r_{k+1} \right\} \quad (1)$$

This research was financially supported by the BSIK-ICIS project (grant no. BSIK03024) and by the STW-VIDI project DWV.6188.

L. Buşoniu, B. De Schutter, and R. Babuška are with the Delft Center for Systems and Control of Delft University of Technology, Netherlands (email: i.l.busoniu@tudelft.nl, b@deschutter.info, r.babuska@tudelft.nl). D. Ernst is a Research Associate of the Belgian FNRS; he is with the Systems and Modeling Unit of University of Liège, Belgium (email: dernst@ulg.ac.be).

where $\gamma \in [0,1)$ is the discount factor, and the notation $x_{k+1} \sim f(x_k, h(x_k), \cdot)$ means that x_{k+1} is drawn from the distribution $f(x_k, h(x_k), \cdot)$. The goal is to find an optimal policy h^*, i.e., a policy that maximizes the return (1) from every $x_0 \in X$.

The Q-function $Q^h : X \times U \to \mathbb{R}$ of the policy h gives, for every pair (x,u), the expected return when starting in x, applying u, and following h thereafter. For any policy, Q^h is unique and can be found by solving the Bellman equation $Q^h = T^h(Q^h)$, where the policy evaluation mapping T^h is:

$$[T^h(Q)](x,u) = \mathrm{E}_{x' \sim f(x,u,\cdot)} \{\rho(x,u,x') + \gamma Q(x', h(x'))\}$$

The PI algorithm starts with an arbitrary initial policy h_0. At every iteration $\ell \geq 0$, the algorithm *evaluates* the current policy, i.e., computes its Q-function Q^{h_ℓ}, and then finds an *improved* policy using:

$$h_{\ell+1}(x) = \arg\max_u Q^{h_\ell}(x,u) \tag{2}$$

The PI algorithm converges to an optimal policy h^*.

In general, the Q-function must be approximated (e.g., when X or U contain an infinite number of elements). In this paper, we consider linearly parameterized Q-function approximators, which use a vector of n basis functions (BFs) $\phi(x,u) = [\phi_1(x,u), \ldots, \phi_n(x,u)]^{\mathrm{T}}$, and a parameter vector $\theta \in \mathbb{R}^n$. Approximate Q-values are computed with:

$$\widehat{Q}(x,u) = \phi^{\mathrm{T}}(x,u)\theta \tag{3}$$

To find an approximate Q-function for a policy h, a *projected* form of the Bellman equation can be solved:

$$\widehat{Q}^h = P^w(T^h(\widehat{Q}^h)) \tag{4}$$

where P^w performs a weighted least-squares projection on the space of representable Q-functions, i.e., the space $\left\{\phi^{\mathrm{T}}(x,u)\theta \mid \theta \in \mathbb{R}^n\right\}$. The weight function $w : X \times U \to [0,1]$ has to satisfy $\sum_{x,u} w(x,u) = 1$, because it is also interpreted as a probability distribution. Equation (4) can be written as a linear equation in the parameter vector:

$$\Gamma\theta^h = \gamma\Lambda\theta^h + z \tag{5}$$

where $\Gamma, \Lambda \in \mathbb{R}^{n \times n}$ and $z \in \mathbb{R}^n$. Using a solution of this equation in (3) gives $\widehat{Q}^h$. For more details about projection-based policy evaluation, see, e.g., Ch. 6 of [2].

The matrices Γ, Λ and the vector z can be estimated from transition samples. Consider a set of samples $\{(x_{l_\mathrm{s}}, u_{l_\mathrm{s}}, x'_{l_\mathrm{s}} \sim f(x_{l_\mathrm{s}}, u_{l_\mathrm{s}}, \cdot), r_{l_\mathrm{s}} = \rho(x_{l_\mathrm{s}}, u_{l_\mathrm{s}}, x'_{l_\mathrm{s}})) \mid l_\mathrm{s} = 1, \ldots, n_\mathrm{s}\}$, constructed by drawing state-action samples (x,u) and then computing corresponding next states and rewards. The probability of each (x,u) must be $w(x,u)$. The estimates of Γ, Λ, and z are initialized to zeros and updated with:

$$\begin{aligned}
\Gamma_{l_\mathrm{s}} &= \Gamma_{l_\mathrm{s}-1} + \phi(x_{l_\mathrm{s}}, u_{l_\mathrm{s}})\phi^{\mathrm{T}}(x_{l_\mathrm{s}}, u_{l_\mathrm{s}}) \\
\Lambda_{l_\mathrm{s}} &= \Lambda_{l_\mathrm{s}-1} + \phi(x_{l_\mathrm{s}}, u_{l_\mathrm{s}})\phi^{\mathrm{T}}(x'_{l_\mathrm{s}}, h(x'_{l_\mathrm{s}})) \\
z_{l_\mathrm{s}} &= z_{l_\mathrm{s}-1} + \phi(x_{l_\mathrm{s}}, u_{l_\mathrm{s}})r_{l_\mathrm{s}}
\end{aligned} \tag{6}$$

LSTD-Q is a policy evaluation algorithm that processes the batch of samples using (6) and then solves the equation:

$$\frac{1}{n_\mathrm{s}}\Gamma_{n_\mathrm{s}}\widehat{\theta}^h = \gamma\frac{1}{n_\mathrm{s}}\Lambda_{n_\mathrm{s}}\widehat{\theta}^h + \frac{1}{n_\mathrm{s}}z_{n_\mathrm{s}} \tag{7}$$

to find an approximate parameter vector $\widehat{\theta}^h$. When $n_\mathrm{s} \to \infty$, we have that $\frac{1}{n_\mathrm{s}}\Gamma_{n_\mathrm{s}} \to \Gamma$, $\frac{1}{n_\mathrm{s}}\Lambda_{n_\mathrm{s}} \to \Lambda$, and $\frac{1}{n_\mathrm{s}}z_{n_\mathrm{s}} \to z$, and therefore that $\widehat{\theta}^h \to \theta^h$. To obtain an algorithm for approximate PI, the solution $\widehat{\theta}^h$ found by LSTD-Q is substituted in (3) to obtain an approximate Q-function, which is used to perform a policy improvement with (2). Then, the procedure repeats at the next iteration. The resulting algorithm is called LSPI [5].

An alternative to LSTD-Q is LSPE-Q [10], [12], which starts with an arbitrary initial parameter vector θ_0 and updates it using:

$$\begin{aligned}
\theta_{l_\mathrm{s}} &= \theta_{l_\mathrm{s}-1} + \beta(\theta^\dagger_{l_\mathrm{s}} - \theta_{l_\mathrm{s}-1}), \text{ where:} \\
\frac{1}{l_\mathrm{s}}\Gamma_{l_\mathrm{s}}\theta^\dagger_{l_\mathrm{s}} &= \gamma\frac{1}{l_\mathrm{s}}\Lambda_{l_\mathrm{s}}\theta_{l_\mathrm{s}-1} + \frac{1}{l_\mathrm{s}}z_{l_\mathrm{s}}
\end{aligned} \tag{8}$$

with β a step size parameter. The matrix Γ should be initialized to a small multiple of the identity matrix. The Q-function given by $\widehat{\theta}^h = \theta_{n_\mathrm{s}}$ can be used for policy improvement in approximate PI. Note that, to guarantee the asymptotical convergence of LSPE-Q to θ^h, the weight of each state-action pair $w(x,u)$ must be identical to the steady-state probability of this pair along an infinitely-long trajectory generated with the policy h [2]. In contrast, LSTD-Q (7) may have meaningful solutions for many weight functions w.

As long as the policy evaluation error is bounded, approximate PI algorithms eventually produce policies with a bounded suboptimality. Although for the derivation above it was assumed that X and U are countable, LSTD-Q and LSPE-Q can also be applied in uncountable (e.g., continuous) state-action spaces. The remainder of this paper will focus on LSTD-Q and LSPI, although LSPE-Q will also be revisited.

III. Online LSPI

LSPI is a state-of-the-art algorithm for approximate PI. However, it only works offline: it improves the policy only after an accurate Q-function has been obtained by running LSTD-Q on a large batch of samples. In contrast, one of the main goals of RL is to develop algorithms that learn online, in which case the policy should improve once every few samples. Therefore, in this paper, we introduce an online variant of LSPI. The crucial difference from the offline case is that policy improvements must be performed once every few transitions, before an accurate evaluation of the current policy can be completed. In the extreme case, the policy is improved after every transition, and then applied to obtain a new transition sample. Then, another policy improvement takes place, and the cycle repeats. Such a variant of PI is called fully optimistic [2], [13]. In general, online LSPI improves the policy once every several (but not too many) transitions; this variant is partially optimistic.

A second major difference between offline and online LSPI is that, while offline LSPI is supplied with a set of transition samples by the experimenter, online LSPI is responsible for collecting its own samples, by interacting with the controlled process. This immediately implies that online LSPI has to *explore*, i.e., try other actions than those given by the current policy. Without exploration, only the

actions dictated by the current policy would be performed in every state, and samples of the other actions in that state would not be available. This would lead to a poor estimation of the Q-values of these other actions, and the resulting Q-function would not be reliable for policy improvement [2]. Furthermore, exploration helps to obtain data from regions of the state space that would not be reached using only the greedy policy. In this paper, ε-greedy exploration is used: at every step k, a uniform random exploratory action is applied with probability $\varepsilon_k \in [0,1]$, and the greedy (maximizing) action with probability $1-\varepsilon_k$, see, e.g., [1]. Typically, ε_k decreases over time, as k increases, so that the algorithm increasingly *exploits* the current policy, as this policy (expectedly) approaches the optimal one.

Algorithm 1 Online LSPI with ε-greedy exploration

Input: BFs ϕ_l, $l=1,\ldots,n$; γ; K_θ; $\{\varepsilon_k\}_{k\geq 0}$; δ

1: $\ell \leftarrow 0$; initialize policy h_0
2: $\Gamma_0 \leftarrow \delta I_{n\times n}$; $\Lambda_0 \leftarrow 0_{n\times n}$; $z_0 \leftarrow 0_n$
3: measure initial state x_0
4: **for** each time step $k \geq 0$ **do**
5: $\quad u_k \leftarrow \begin{cases} h_\ell(x_k) & \text{w.p. } 1-\varepsilon_k \\ \text{a uniform random action} & \text{w.p. } \varepsilon_k \end{cases}$
6: $\quad$ apply u_k, measure next state x_{k+1} and reward r_{k+1}
7: $\quad \Gamma_{k+1} \leftarrow \Gamma_k + \phi(x_k,u_k)\phi^{\mathrm{T}}(x_k,u_k)$
8: $\quad \Lambda_{k+1} \leftarrow \Lambda_k + \phi(x_k,u_k)\phi^{\mathrm{T}}(x_{k+1},h_\ell(x_{k+1}))$
9: $\quad z_{k+1} \leftarrow z_k + \phi(x_k,u_k)\, r_{k+1}$
10: $\quad$ **if** $k = (\ell+1)K_\theta$ **then**
11: $\quad\quad$ solve $\frac{1}{k+1}\Gamma_{k+1}\theta_\ell = \frac{1}{k+1}\Lambda_{k+1}\theta_\ell + \frac{1}{k+1}z_{k+1}$
12: $\quad\quad h_{\ell+1}(x) \leftarrow \arg\max_u \phi^{\mathrm{T}}(x,u)\theta_\ell \quad \forall x$
13: $\quad\quad \ell \leftarrow \ell+1$
14: $\quad$ **end if**
15: **end for**

Algorithm 1 presents online LSPI with ε-greedy exploration. Online LSPI uses two new, essential parameters that are not present in offline LSPI: the number $K_\theta \in \mathbb{N}$, $K_\theta > 0$ of transitions between consecutive policy improvements, and the exploration schedule $\{\varepsilon_k\}_{k\geq 0}$. When $K_\theta = 1$, the policy is updated after every sample and online LSPI is fully optimistic. When $K_\theta > 1$, the algorithm is partially optimistic. The number K_θ should not be chosen too large, and a significant amount of exploration is recommended, i.e., ε_k should not approach 0 too fast. In this paper, the exploration probability is initially set to a value ε_0, and decays exponentially once every second with a decay rate of $\varepsilon_{\mathrm{d}} \in (0,1)$:

$$\varepsilon_k = \varepsilon_0\, \varepsilon_{\mathrm{d}}^{\lfloor kT_{\mathrm{s}} \rfloor} \tag{9}$$

where T_{s} is the sampling time of the process, and $\lfloor\cdot\rfloor$ denotes the floor operator.[1] Note that, in practice, improved policies do not have to be explicitly computed in online LSPI (line 12), but can be computed on demand using (2). To ensure its invertibility, Γ is initialized to a small multiple $\delta I_{n\times n}$ of the identity matrix, where $\delta > 0$.

Offline LSPI rebuilds Γ, Λ, and z from scratch before every policy improvement. Online LSPI cannot do this, because the few samples that arrive before the next policy improvement are not sufficient to construct informative new estimates of Γ, Λ and z. Instead, these estimates are continuously updated. The underlying assumption is that the Q-functions of subsequent policies are similar, which means that the previous values of Γ, Λ, and z are also representative for the improved policy. Note that the computational and memory demands of online LSPI are independent of the number of samples observed.

IV. Experimental study

This section provides an extensive experimental evaluation of online LSPI, for the problem of swinging up an underactuated inverted pendulum. This problem is challenging and highly nonlinear, but low-dimensional (the pendulum has two state variables and one action variable), which means extensive simulations can be performed with reasonable computational costs. We study the effects of the policy improvement interval and of the exploration decay rate on the performance of online LSPI. Then, we compare online LSPI with its offline counterpart, and with an online PI algorithm with LSPE-Q. Finally, we provide real-time learning results.

The inverted pendulum (Figure 1) consists of a weight of mass m attached to a disk, which is actuated by a DC motor and rotates in a vertical plane. The motor power is insufficient to push the pendulum up in a single rotation from every initial state. Instead, from certain states (e.g., pointing down), the pendulum needs to be swung back and forth to gather energy, prior to being pushed up and stabilized.

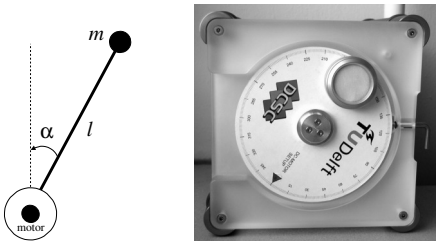

Fig. 1. Inverted pendulum schematic (left) and the real system (right).

A continuous-time model of the pendulum dynamics is:

$$\ddot{\alpha} = 1/J \cdot [mgl\sin(\alpha) - b\dot{\alpha} - K^2\dot{\alpha}/R + Ku/R]$$

where $J = 1.91\cdot 10^{-4}\,\mathrm{kgm}^2$, $m = 0.055\,\mathrm{kg}$, $g = 9.81\,\mathrm{m/s}^2$, $l = 0.042\,\mathrm{m}$, $b = 3\cdot 10^{-6}\,\mathrm{Nms/rad}$, $K = 0.0536\,\mathrm{Nm/A}$, $R = 9.5\,\Omega$. The angle α varies in the interval $[-\pi,\pi)\,\mathrm{rad}$, with $\alpha = 0$ pointing up, and 'wraps around' so that, e.g., a rotation of $3\pi/2$ corresponds to $\alpha = -\pi/2$. The state is $x = [\alpha,\dot{\alpha}]^{\mathrm{T}}$. The control action u is constrained to $[-3,3]\,\mathrm{V}$, and the velocity $\dot{\alpha}$ is restricted to $[-15\pi,15\pi]\,\mathrm{rad/s}$, using saturation. The sampling time is $T_{\mathrm{s}} = 0.005\,\mathrm{s}$, and the discrete-time transitions are obtained by numerically integrating the continuous-time dynamics between consecutive time steps. The goal is to stabilize the pendulum in the

[1] Exponential decay does not asymptotically lead to infinite exploration, which is required by some online RL algorithms [14]. Nevertheless, for an experiment having a finite duration, ε_{d} can be chosen large enough to provide any desired amount of exploration.

unstable equilibrium $x = 0$ (pointing up), and is expressed by the reward function:

$$r_{k+1} = \rho(x_k, u_k) = -x_k^{\mathrm{T}} Q_{\mathrm{rew}} x_k - R_{\mathrm{rew}} u_k^2$$
$$\text{where: } Q_{\mathrm{rew}} = \mathrm{diag}[5, 0.1],\ R_{\mathrm{rew}} = 1$$

Here, Q_{rew} is chosen to penalize nonzero values of the two state variables to a similar extent, given their relative magnitudes; and R_{rew} penalizes energy consumption, to a smaller extent than the state deviations. The discount factor is $\gamma = 0.98$, sufficiently large to lead to a good control policy.

1) Approximator and performance criterion: To approximate the Q-function, an equidistant 11×11 grid of Gaussian radial BFs (RBFs) is defined over the state space, and the action space is discretized into 3 discrete values: $U_{\mathrm{d}} = \{-3, 0, 3\}$. The RBFs are normalized, axis-parallel, and have identical radii. The RBF radius along each dimension is identical to the distance between two adjacent RBFs along that dimension (the grid step). To obtain the $n = 3 \cdot 11^2 = 363$ state-action BFs, the RBFs are replicated for every discrete action, and all the BFs that do not correspond to the current discrete action are taken equal to 0. So, if the vector of RBFs is $\bar{\phi}(x) = [\bar{\phi}_1(x), \ldots, \bar{\phi}_{121}(x)]^{\mathrm{T}}$, then the vector of state-action BFs is $\phi(x, u) = [\mathcal{I}(u = -3) \cdot \bar{\phi}^{\mathrm{T}}(x), \mathcal{I}(u = 0) \cdot \bar{\phi}^{\mathrm{T}}(x), \mathcal{I}(u = 3) \cdot \bar{\phi}^{\mathrm{T}}(x)]^{\mathrm{T}}$, where the indicator function $\mathcal{I}$ is 1 when its argument is true, and 0 otherwise.

After each simulated experiment with online LSPI is completed, snapshots of the current policy at increasing moments of time are evaluated. This produces a curve recording the control performance of the policy over time. During performance evaluation, learning and exploration are turned off. Policies are evaluated using simulation, by estimating their average return over the grid of initial states $X_0 = \{-\pi, -\pi/2, 0, \pi/2\} \times \{-10\pi, -3\pi, -\pi, 0, \pi, 3\pi, 10\pi\}$. The return from each state is estimated with a precision $\varepsilon_R = 0.1$.

2) Effects of the tuning parameters: In this section, we study the effects of varying the tuning parameters of online LSPI, in particular the number of transitions between consecutive policy improvements, K_θ, and the exploration decay rate, ε_{d}. Each experiment is run for 600 s, and is split into trials having a length of 1.5 s, which is sufficient for a good policy to swing up and stabilize the inverted pendulum. The initial state of each trial is drawn from a uniform random distribution over X. The decaying exploration schedule (9) is used, with $\varepsilon_0 = 1$, which means that a fully random policy is initially used. Any small positive value is appropriate for δ; we set this parameter to 0.001.

To study the influence of K_θ, the following values are used: $K_\theta = 1, 10, 100, 1000$, and 5000. The first experiment ($K_\theta = 1$) is fully optimistic: the policy is improved after every sample. The exploration decay rate is $\varepsilon_{\mathrm{d}} = 0.9962$. Figure 2 shows how the performance of the policies learned by online LSPI evolves. The mean performance across 20 independent runs of each experiment is reported. To avoid cluttering, confidence intervals are shown only for the extreme values of K_θ, in a separate graph. The performance converges quickly, in roughly 120 s, i.e., 80 trials. The algorithm is robust to changes in the K_θ parameter, with

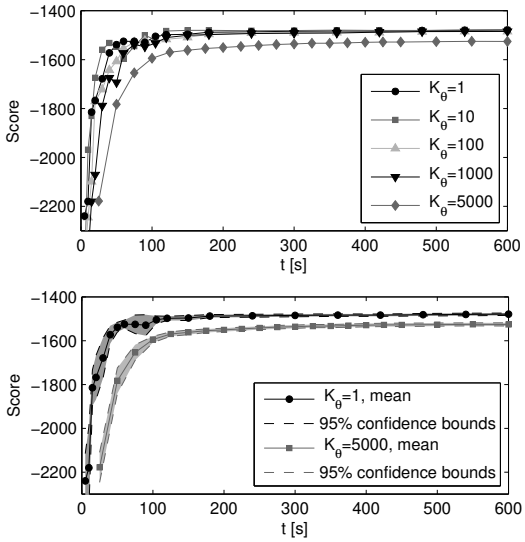

Fig. 2. Performance of online LSPI for varying K_θ. Top: mean performance for all the experiments; bottom: mean performance with 95% confidence intervals, for the extreme values of K_θ. The marker locations indicate the moments in time when the policies were evaluated.

all the values leading to a similar performance except $K_\theta = 5000$. For this large value, the performance is worse, and the difference from smaller K_θ is statistically significant, as illustrated in the bottom graph. Thus, policy improvements in online LSPI should not be performed too rarely.

To study the influence of ε_{d}, the following values are used: $\varepsilon_{\mathrm{d}} = 0.8913, 0.9550, 0.9772,\ 0.9924, 0.9962$, and 0.9996. Larger values of ε_{d} correspond to more exploration; in particular, for $\varepsilon_{\mathrm{d}} = 0.9996$, most of the actions taken during learning are exploratory. The policy is improved once every $K_\theta = 10$ transitions. Figure 3 presents the performance of online LSPI across 20 independent runs. There is no discernible effect of ε_{d} on the learning rate, but the final performance improves with more exploration. The difference between the performance of large and small

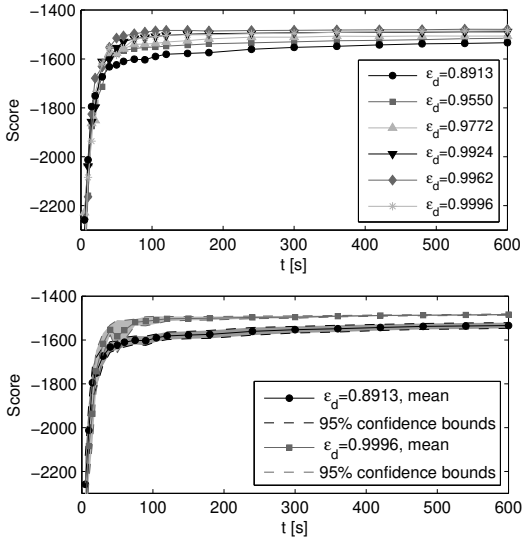

Fig. 3. Performance of online LSPI for varying ε_{d}. Top: mean performance for all the experiments; bottom: mean performance with 95% confidence intervals, for the extreme values of ε_{d}.

exploration schedules is statistically significant, as illustrated in the bottom part of the figure. These results are not surprising, since the considerations in Section III already indicated that online LSPI requires significant exploration. (Note however that too much exploration will decrease the control performance obtained *during* learning. This effect is not visible in Figure 3, because exploration is turned off when evaluating policies.)

Figure 4 shows the mean execution time for varying K_θ. The 95% confidence intervals are left out, since they are too small to be visible at the scale of the figure. The execution time is larger for smaller K_θ, because the most computationally expensive operation is solving the linear system in (7), which must be done once every K_θ steps. The execution time for all values of ε_d is around 330 s (it does not change much with the exploration schedule, since choosing random actions is computationally cheap).

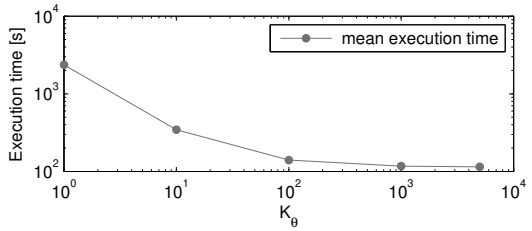

Fig. 4. Execution time of online LSPI for varying K_θ.

3) Comparison of online LSPI and offline LSPI: In this section, offline LSPI is used to find policies for the swingup problem. These policies are compared with the *final* policies found by online LSPI, at the end of the learning process. Offline LSPI employs the same approximator as online LSPI. Whereas online LSPI generates its own samples during learning, a number of $n_\mathrm{s} = 20000$ pre-generated random samples are used for offline LSPI, uniformly distributed throughout the state-discrete action space $X \times U_\mathrm{d}$. The offline experiment is run 20 times with independent sets of samples. Table I compares the performance and the execution time of offline and online LSPI. Two representative online experiments from the study of K_θ are selected for comparison: the experiment with the best mean performance, and the experiment with the worst mean performance. Two experiments from the study of ε_d are similarly selected. The performance and execution times are rounded to integer precision.

TABLE I

OFFLINE VERSUS ONLINE LSPI (MEAN; 95% CONFIDENCE INTERVAL).

Experiment	Performance	Execution time [s]
Offline	-1497; $[-1504, -1490]$	83; $[80, 86]$
$K_\theta = 10$ (best)	-1478; $[-1483, -1473]$	345; $[344, 345]$
$K_\theta = 5000$ (worst)	-1526; $[-1533, -1519]$	115; $[115, 115]$
$\varepsilon_\mathrm{d} = 0.9962$ (best)	-1479; $[-1482, -1476]$	336; $[333, 339]$
$\varepsilon_\mathrm{d} = 0.8913$ (worst)	-1534; $[-1547, -1521]$	334; $[332, 336]$

The final performance of online LSPI is comparable with the performance of offline LSPI, and is better for good selections of the parameters. On the other hand, online LSPI is more computationally expensive than offline LSPI, because it performs more policy improvements. Note that offline LSPI employs 20000 samples, whereas the online algorithm processes the same number of samples in 100 s, and 120000 samples during the entire learning process. Nevertheless, Figures 2 and 3 showed that the online performance is already good after 120 s, i.e., 24000 samples. Also, offline LSPI loops through the samples once at every iteration, whereas online LSPI processes samples only once. Therefore, online LSPI compares favorably with offline LSPI in the number of samples required to reach a good performance.

4) Comparison of online LSPI and online PI with LSPE-Q: In this section, we consider an online PI algorithm that evaluates policies with LSPE-Q (8), rather than with LSTD-Q as online LSPI does. Unlike online LSPI, online PI with LSPE-Q updates the parameter vector after every transition:

$$\theta_{k+1} = \theta_k + \beta(\theta^\dagger_{k+1} - \theta_k), \text{ where:}$$
$$\frac{1}{k+1}\Gamma_{k+1}\theta^\dagger_{k+1} = \gamma\frac{1}{k+1}\Lambda_{k+1}\theta_k + \frac{1}{k+1}z_{k+1} \tag{10}$$

where Γ, Λ, and z are computed as in online LSPI (Algorithm 1). The policy is improved once every K_θ transitions.

We apply online PI with LSPE-Q to the swingup problem, using the same values of K_θ as for online LSPI. The approximator, exploration schedule, and trial length are also the same; Γ is initialized to $0.001 \cdot I_{n\times n}$. Online PI with LSPE-Q has an additional step size parameter, β, which was not present in online LSPI. In order to choose β, preliminary experiments were performed for each value of K_θ, using several values of β: $0.001, 0.005, 0.01, 0.05, 0.1, 0.5$, and 1. In these experiments, the following values of β performed reasonably: $0.005, 0.01, 0.01, 0.1$, and 0.1, for, respectively, $K_\theta = 1, 10, 100, 1000$, and 5000. With these values of β, 20 independent runs are performed for every K_θ.

Figure 5 presents the performance of online PI with LSPE-Q across these 20 runs; compare with Figure 2. Online PI with LSPE-Q is less reliable than online LSPI: there is a larger variation in performance across the 20 runs,

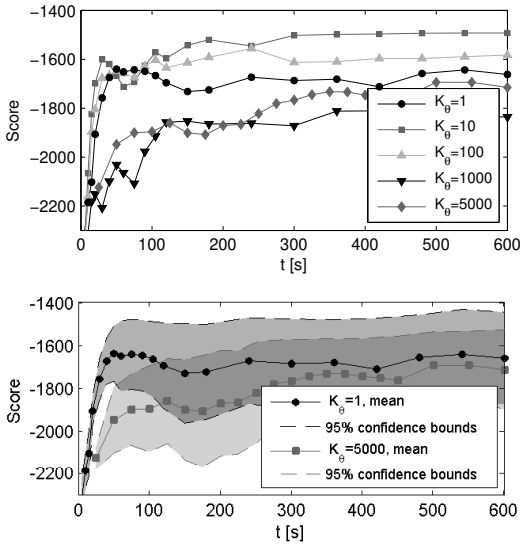

Fig. 5. Performance of online PI with LSPE-Q for varying K_θ. Top: mean performance for all the experiments; bottom: mean performance with 95% confidence intervals, for the extreme values of K_θ.

which can be seen, e.g., in the much larger 95% confidence intervals. To explain this, recall that in order to guarantee the convergence of LSPE-Q, state-action samples must be generated according to their steady-state probabilities along an infinitely-long trajectory generated with the current policy (Section II). In online PI, the policy is changed often and many exploratory actions are taken, which severely violates this requirement, destabilizing the update (10). While online LSPI is also affected by the imprecision in the values of Γ, Λ, and z, it may be more stable because it only uses them to compute 'one-shot' solutions, rather than updating the parameter vector recursively like online PI with LSPE-Q.

The mean execution time of online PI with LSPE-Q is around $1200\,\mathrm{s}$ for all values of K_θ. Compared to online LSPI (Figure 4), online PI with LSPE-Q is more computationally expensive when $K_\theta > 1$, since it must solve a linear system at every step; in contrast, online LSPI only solves a linear system before policy improvements.

5) Online LSPI for the real pendulum: Next, online LSPI is used to control the inverted pendulum system in real time, rather than in simulation as in the earlier sections. To make the problem slightly easier for the learning controller, the sampling time is increased to $T_\mathrm{s} = 0.02\,\mathrm{s}$ (from $0.005\,\mathrm{s}$), and the maximum available control is increased to $3.2\,\mathrm{V}$ (from $3\,\mathrm{V}$); even so, a swingup is still required to turn the pendulum upright. The same approximator is used as in the simulation experiments, and online LSPI is run for $300\,\mathrm{s}$, split in trials of $2\,\mathrm{s}$ each. Half of the trials start in the stable equilibrium (pointing down), and half in a random initial state obtained by applying a sequence of random actions. The initial exploration probability is $\varepsilon_0 = 1$ and decays with $\varepsilon_\mathrm{d} = 0.9848$, which leads to a final value of $\varepsilon = 0.01$. Policy improvements are performed only after each trial, because solving the linear system at line 11 of Algorithm 1 may take longer than the sampling time.

Figure 6 presents a subsequence of learning trials, containing 1 out of each 10 trials. These trajectories *include* the effects of exploration. The controller successfully learns how to swing up and stabilize the pendulum, giving a good performance roughly $120\,\mathrm{s}$ into learning. This is similar to the learning rate observed in the simulation experiments.

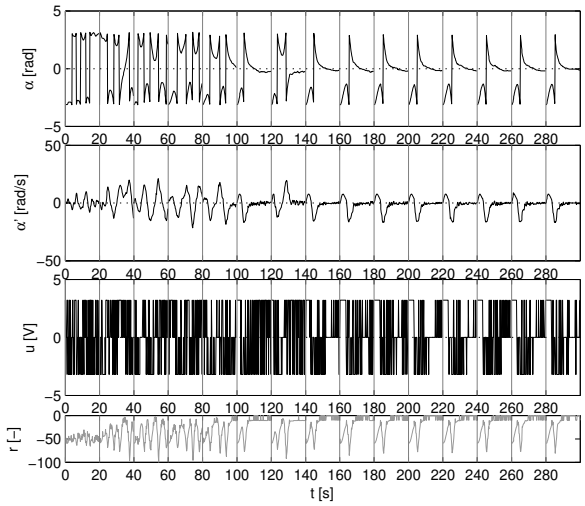

Fig. 6. A representative subsequence of learning trials for the real inverted pendulum. Each trial is $2\,\mathrm{s}$ long, and only 1 out of every 10 trials is shown. The starting time of each trial is given on the horizontal axis, and trials are separated by vertical lines. Thus, each line corresponds to a 'gap' of $18\,\mathrm{s}$ in real time.

V. Conclusions

In this paper, we have introduced online least-squares policy iteration: an online PI algorithm with LSTD-Q policy evaluation. We have provided an extensive experimental study of online LSPI for the problem of swinging up an inverted pendulum. In this study, the algorithm learned fast and reliably, without much sensitivity to its tuning parameters. Online LSPI also performed well in comparison to its offline counterpart, worked in real-time control, and was more stable than online PI with LSPE-Q, even though LSPE-Q has previously been deemed more appropriate for online learning than LSTD-Q [2], [10]. These results indicate that online LSPI is a promising algorithm for learning control.

Analyzing whether the asymptotical performance of online LSPI can be guaranteed is an important research topic. The performance guarantees of offline PI rely on bounded policy evaluation errors. Because online LSPI improves the policy before an accurate value function is available, the policy evaluation error can be very large, and the guarantees for offline PI cannot be directly applied to the online case.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific, 2007, vol. 2.

[3] J. Boyan, "Technical update: Least-squares temporal difference learning," *Machine Learning*, vol. 49, pp. 233–246, 2002.

[4] A. Nedić and D. P. Bertsekas, "Least-squares policy evaluation algorithms with linear function approximation," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 13, no. 1–2, pp. 79–110, 2003.

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

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

[7] V. Konda, "Actor-critic algorithms," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, US, 2002.

[8] H. Yu and D. P. Bertsekas, "Convergence results for some temporal difference methods based on least squares," *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1515–1531, 2009.

[9] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM Journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[10] T. Jung and D. Polani, "Kernelizing LSPE(λ)," in *Proceedings 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL-07)*, Honolulu, US, 1–5 April 2007, pp. 338–345.

[11] L. Li, M. L. Littman, and C. R. Mansley, "Online exploration in least-squares policy iteration," in *Proceedings 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-09)*, vol. 2, Budapest, Hungary, 10–15 May 2009, pp. 733–739.

[12] D. P. Bertsekas and S. Ioffe, "Temporal differences-based policy iteration and applications in neuro-dynamic programming," Massachusetts Institute of Technology, Cambridge, US, Tech. Rep. LIDS-P-2349, 1996, available at http://web.mit.edu/dimitrib/www/Tempdif.pdf.

[13] R. S. Sutton, "Learning to predict by the method of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.

[14] S. Singh, T. Jaakkola, M. L. Littman, and Cs. Szepesvári, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 287–308, 2000.