Reinforcement Learning - Homework 3

Mehdi Boubnan

December 16, 2018

1 On-Policy Reinforcement Learning with Parametric Policy

1.1 REINFORCE with Gaussian policy model

In this section we'll test the REINFORCE algorithm on the linear-quadratic Gaussian regulation (LQG) problem.

We'll consider these parameters for the problem:

- Gaussian policy $a_t/s_t \sim \mathcal{N}(\theta.s_t, \sigma^2)$ with $\sigma = 0.4$
- Transition model $s_{t+1} \sim \mathcal{N}(s_t + a_t, 0)$
- Reward $r_t = -0.5(s_t^2 + a_t^2)$
- Discount factor : $\gamma = 0.9$

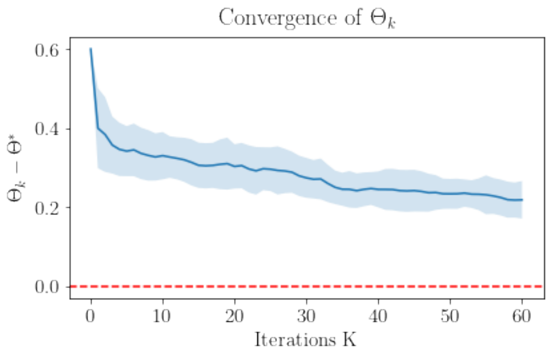
The parameters for the REINFORCE algorithm are as follows : - We will collect **N=50** or **N=100** trajectories per iteration. - Each trajectory will have at most **T=100** time steps. - We'll update the policy parameter **n_itr = 60** times for the constant stepper and **n_itr = 100** times for the Adam optimizer. - We'll average results of **nb_simu=5** experiments.

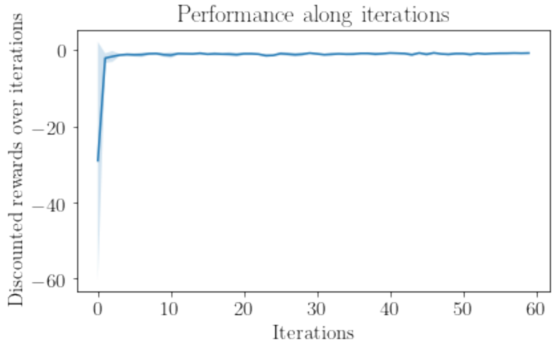
1.1.1 LQG Problem without exploration bonus

We initialize the environment and run the reinforce with two steppers.

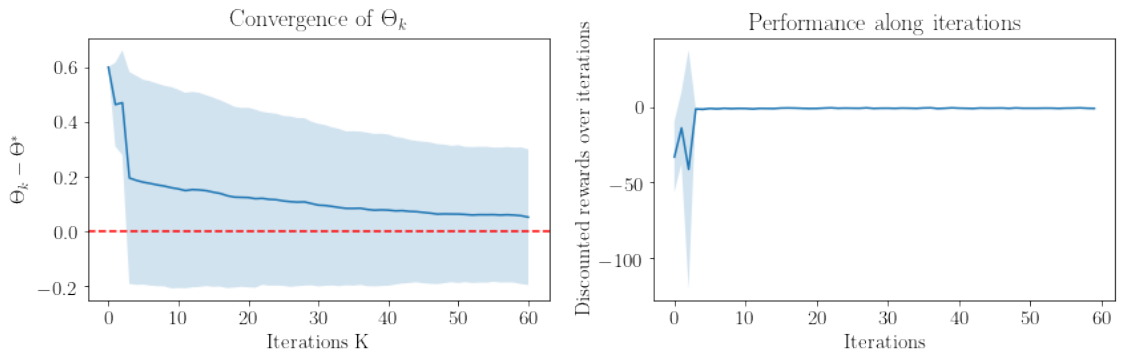
Using Constant stepper For the constant stepper, we'll use a learning rate equal to 2.10^{-5}.

N=50



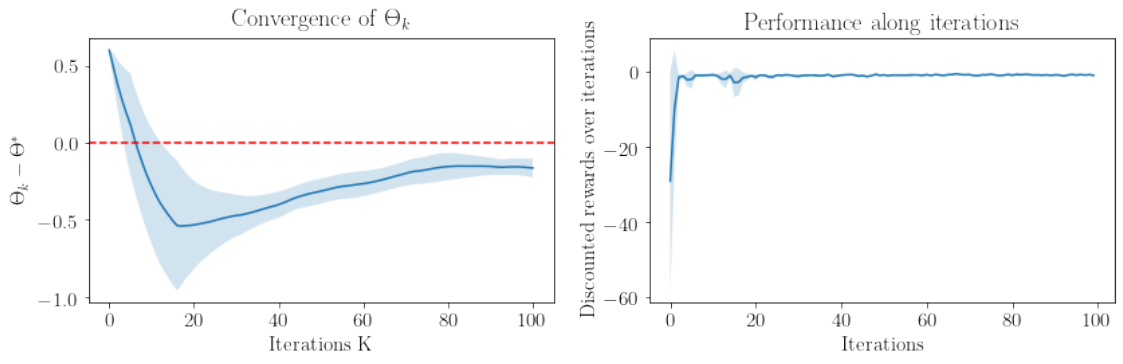


N=100

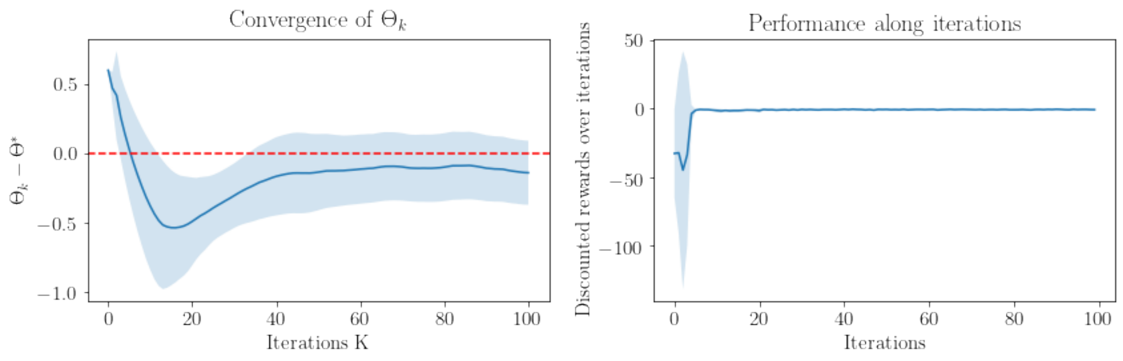


Using Adam stepper We'll now use the Adam stepper, with $\alpha = 0.2$, and the other parameters as default.

N=50



N=100



Effect of N We notice that the variance of the REINFORCE algorithm is very high, as we end with differents results even if we use the same parameters. However, this variance is countered if we increase the number of episodes N. Indeed the variance increase but the convergence is more accurate. Indeed, since we cannot compute the gradient over all the possible trajectories, we estimate the gradient using the Monte-Carlo algorithm over N episode; therefore, if we increase N, the Monte-Carlo estimation is more accurate, and so is the gradient which makes the convergence easier.

Effect of α_t For the constant stepper, as the gradients are big (order of 10^5), we need to take a value for the learning-rate in the same order. Moreover, the more we decrease the learning rate, the slower the algorithm converges. On the other hand, if we increase the learning-rate, we may overshoot the local minima and therefore the algorithm diverges.

1.1.2 LQG Problem with exploration bonus

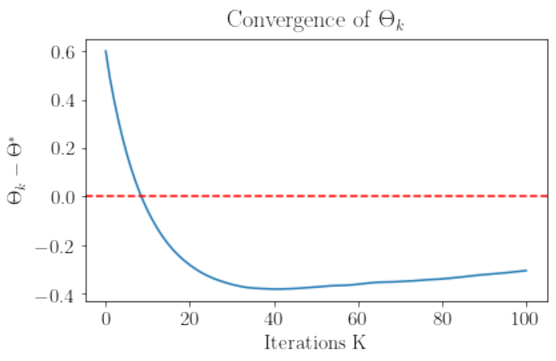
We'll now adopt an optimistic approach called MBIE-BE based on exploration bonus to enforce exploration toward unseen state-action areas.
For that, we'll adopt a new reward : $\tilde{r}(s,a) = r(s,a) + b(s,a)$ with :

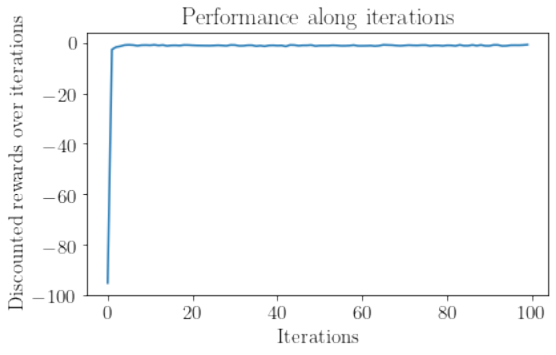
$$b(s,a) = \beta\sqrt{\frac{1}{N_t(\phi(s,a))}}$$

with $N_t(\phi(s,a))$ number of visits to the bin containing (s,a), and $\phi(s,a)$ a hashing function.
We'll take $\beta = \frac{reward_{max}}{1-\gamma}$ for each trajectory.

We run the algorithm with Adam optimizer with the same parameters as before :

```
Running 5 simulations: 100%
```





2 Off-Policy Reinforcement Learning with Value Function Approximation

For this section, we'll implement the Linear Fitted Q-iteration. We will approximate the action-value functions by :

$$\mathcal{F} = \left\{ f_\theta(s,a) = \theta^T \phi(s,a), \theta \in \mathbb{R}^3 \right\}$$

with

$$\phi(s,a) = (a, sa, s^2 + a^2)^T$$

We take :

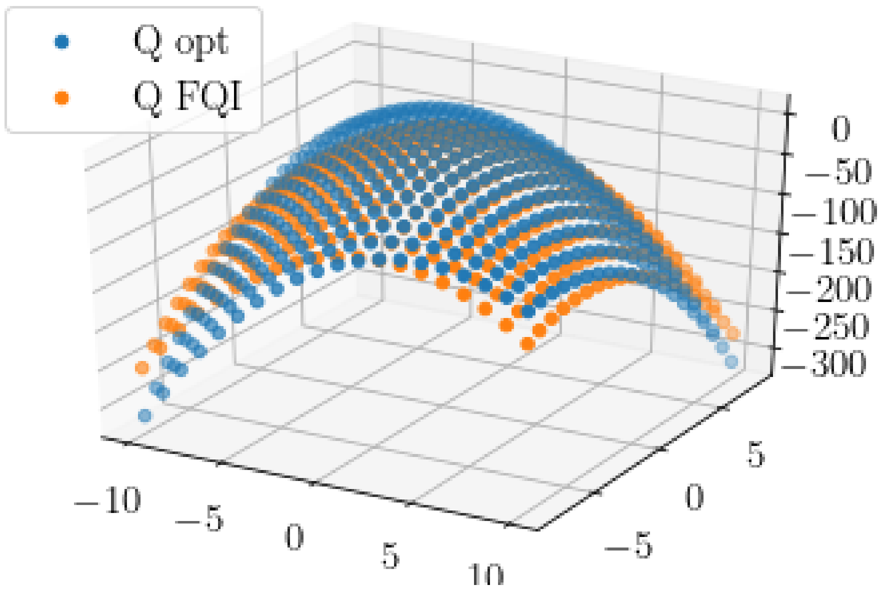
- A discritized state space constitued by 20 points in the interval [-10,10]
- A discritized action space constitued by 20 points in the interval [-8,8]

We initialize the environment, create a dataset of 200 episodes with a horizon equal to 100, and run the Linear Fitted Q-iteration with the following parameters :

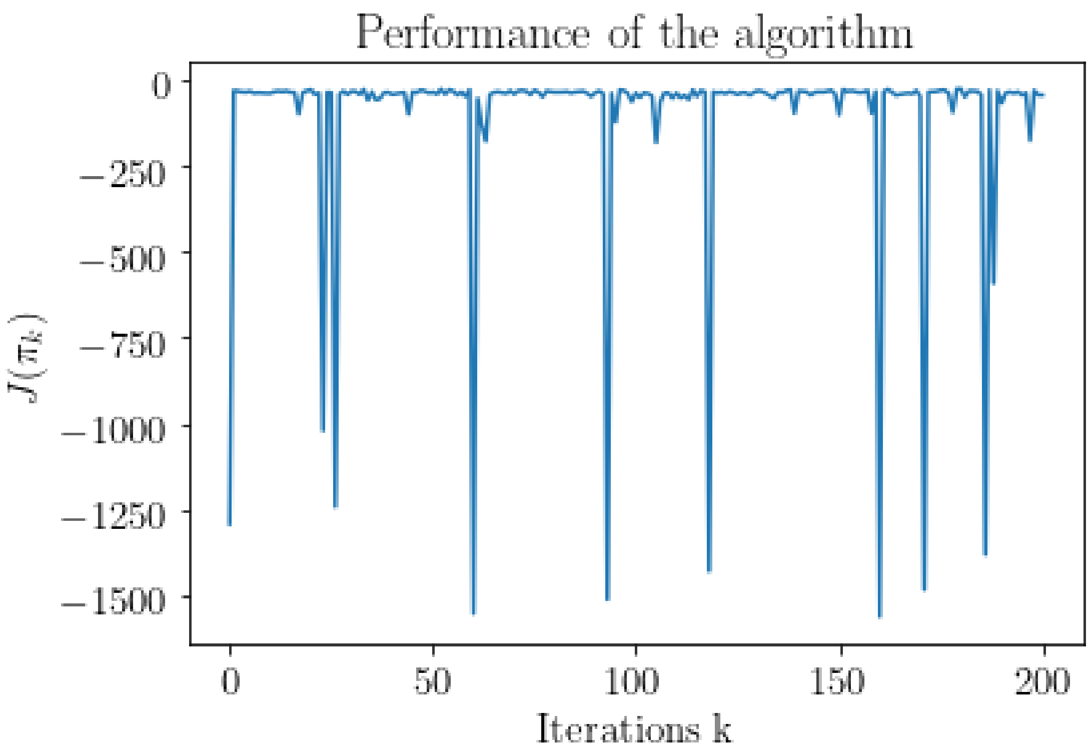
- Discount $\gamma = 0.9$
- Ridge regularization coefficient $\lambda = 10^{-3}$

We plot the Q function estimated by the FQI algorithm and the optimal Q function. The algorithm yields effectively to good results :

```
Optimal K: [[-0.58840335]] Covariance S: 0.001
```



We finally plot the performance of the algorithm along iterations :


Performance of the algorithm
$J(\pi_k)$
Iterations k