Faculty of Applied Sciences
University of Liege
February 27, 2019

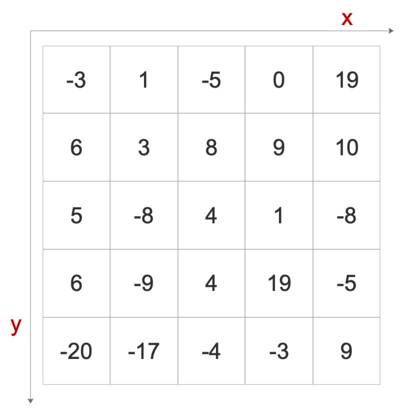
-

Optimal decision making for complex problems:

Assignment 1 - Section 6 to 7

s141770
Derroitte Natan

Continuing this project, the axis system remains a key choice that will influence each of the algorithms implemented. Therefore, it is recalled at the beginning of this report on the second part of the project. 1.



-3	1	-5	0	19
6	3	8	9	10
5	-8	4	1	-8
6	-9	4	19	-5
-20	-17	-4	-3	9

Figure 1: Representation of the axis system

6 Q-learning in batch setting

The purpose of this last part was to implement the Q-learning algorithm. The latter is based on a simple idea but contained many parameters. Regarding the number of episodes, the number of transitions and the learning ratio, the values recommended in the statement will be used (Number of episodes = 100, $t = 1000$, $\alpha = 0.05$) if not state otherwise. Concerning the size of the batch as well as the value of ϵ, their study will be reserved for the section 6.2.

Concerning the first part of this question, the value of $\hat{Q}(x, u)$ is updated using pre-calculated trajectories. The results are presented in figure 2.

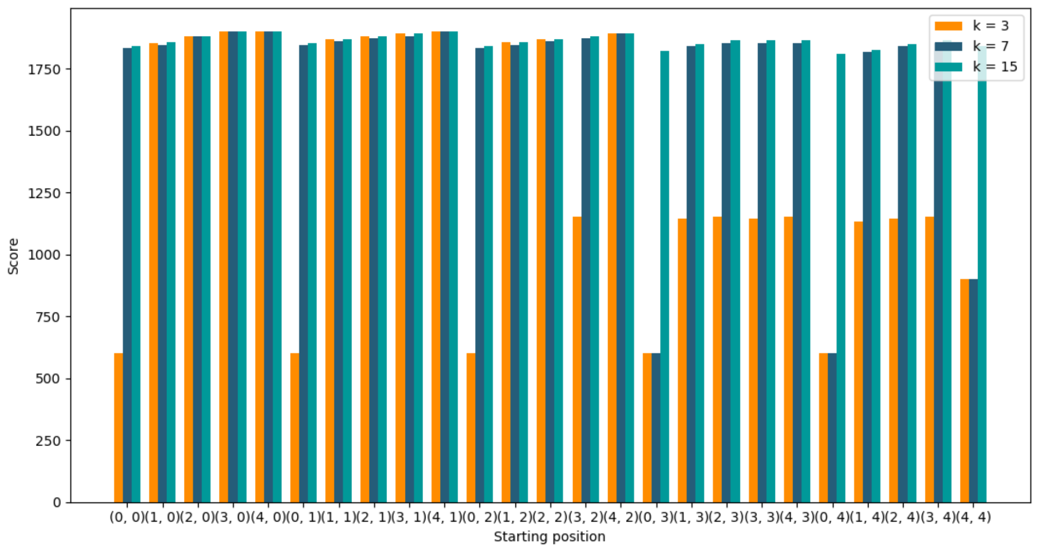


Figure 2: Expected cumulative reward computed using the Q-learning algorithm for different number of trajectories (1000 transitions).

It can be seen that, as expected, the higher the number of trajectories, the better the results. The reason why the higher value of k is only 15 is because after this value, all results are the same : the optimal J is reached.
Identically, by increasing the size of the trajectories, the results improve as shown in figure 3.

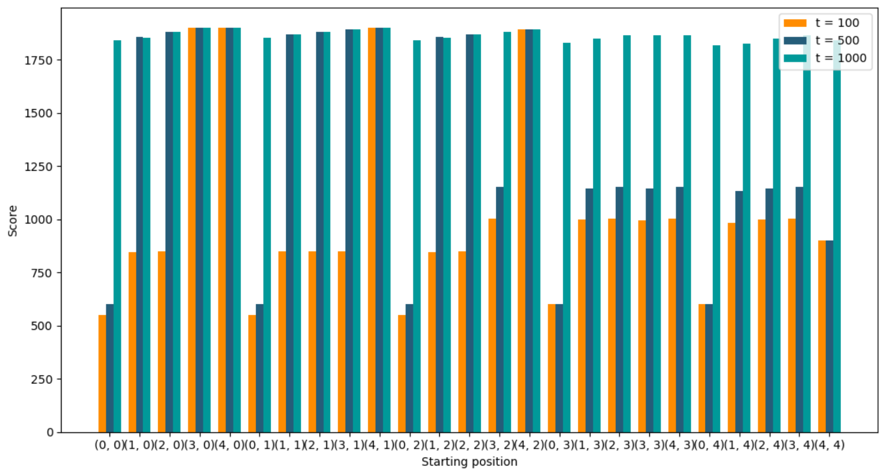


Figure 3: Expected cumulative reward computed using the Q-learning algorithm for different number of transitions in the trajectories used (10 trajectories).

6.1 Convergence of $\hat{Q}$

It is possible to analyse the convergence from $\hat{Q}$ to Q in the figure 4.

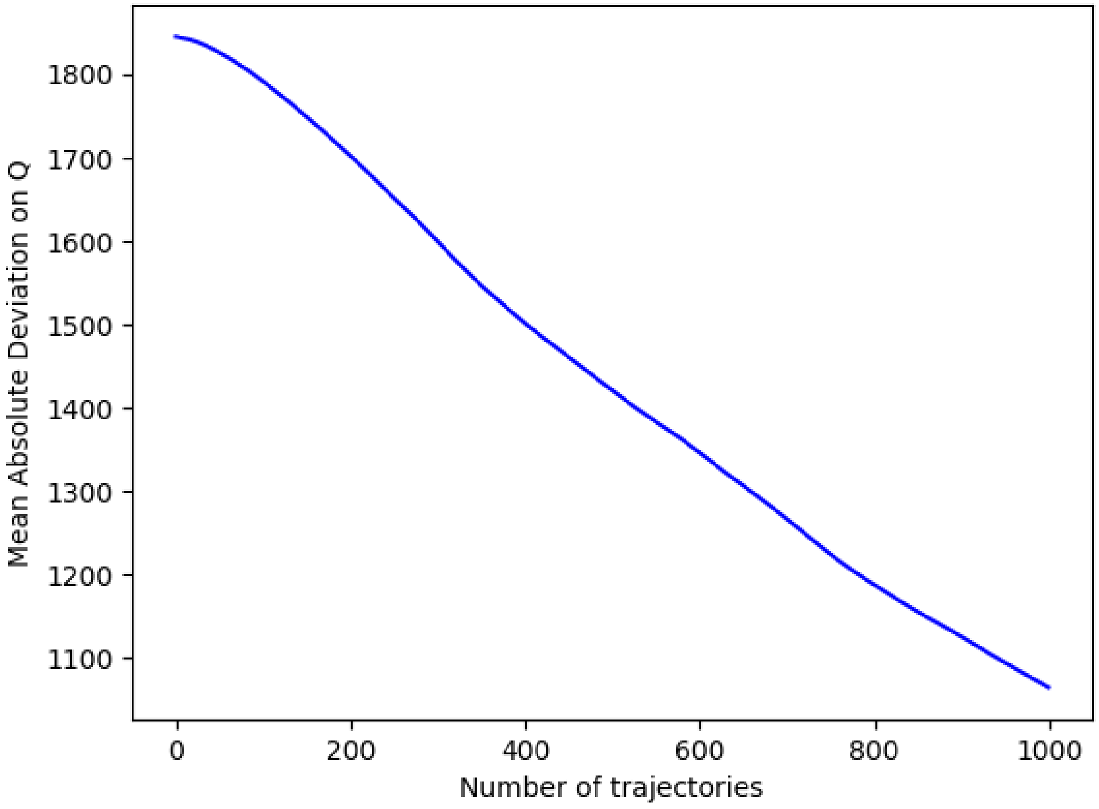


Figure 4: Convergence of $\hat{Q}$ towards Q with respect to the number of trajectories (100 transitions per trajectories).

Even if the error decreases with the number of trajectories, it appears that the error remains important. What explains why the estimated cumulative rewards are not so far from the real J_{μ^*}[1]is that the calculation of the optimal policy is based on the ratio, the proportion, between the values of $Q(x, u)$ and not on the exact values inside the matrix.
Moreover, this lack of convergence can easily be explained knowing that one of the convergence conditions of the Q-learning algorithm is not satisfied. Indeed, the equation

$$\lim_{t->\infty} \sum_{k=0}^{t-1} \alpha_k^2 < \infty$$

is not respected for the constant value of α in this case.

6.2 ϵ-greedy policy

In this part, the algorithm no longer relies on pre-recorded trajectories to approximate the values of $Q(x, u)$. At each iteration of each episode, an experiment is generated from the previous state. This experience is

[1]See figure 2.

then added to the batch. When this one is full, the algorithm updates the $\hat{Q}$ value using the Q-learning formula and generates a new policy from this new $\hat{Q}$. When a new episode begins, the current state is reset to the starting position.
Each experiment has a $1-\epsilon$ chance of being generated directly using the most recent policy computed, *ie* to take the optimal action. To encourage exploration, the action has an ϵ chance of being randomly selected. In the initial case, the policy is always to go upwards since $\hat{Q}_0(x,u) = 0 \ \forall x, u$.

The batch size is an important parameter. If it is 1, $\hat{Q}$ and the corresponding policy will be calculated for each iteration of each episode. If it is 1000 on the contrary, $\hat{Q}$ and the corresponding policy will be calculated once per episode. This parameter will be discussed in detail in the next section. For the moment, we consider only one update per episode, so taking the batch size equals to 1000.

The -greedy approach is mainly based on the ϵ parameter. This represents how much we want to encourage exploration ($\epsilon = 1$) or the exploitation of our knowledge ($\epsilon = 0$). It is therefore important to take a value of ϵ that allows these two aspects to be taken into account.

First of all, different value of ϵ are considered. The impact of the batch size will later be studied for the best value and no experience replay are done for these first results presented in the figure 5.

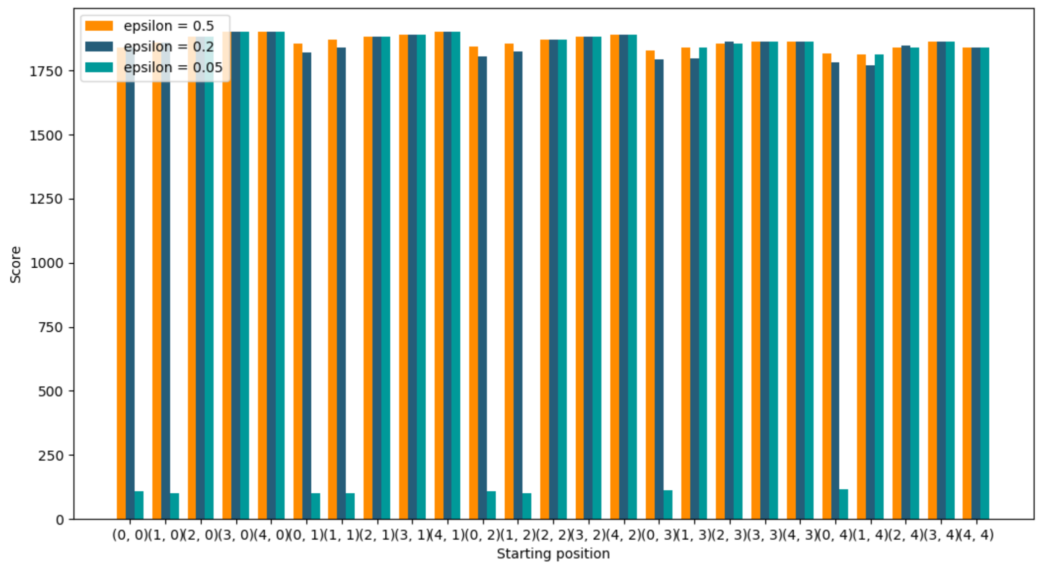


Figure 5: Comparison between the expected cumulative reward computed using the Q-learning algorithm of three epsilon value : 0.5, 0.2 and 0.05. 100 episodes of 1000 transitions

It is clear that when ϵ is too low, exploration is not sufficiently encouraged. So the results are bad. To have a better representation of the influence of ϵ in the other two cases, the table 1 is introduced.

ϵ	0.5	0.2	0.05
Error on $J(x)$	7.65	7.89	23.88

Table 1: Absolute mean deviation on J(x) for different value of epsilon. 100 episodes of 1000 transitions

From this table, it seems that the results are quit similar for the two values $\epsilon = 0.2$ and $\epsilon = 0.5$.
In order to obtain better results, a other method is considered. It seems consistent that exploration should

be encouraged in the first episodes of the algorithm and less in the last ones. It is with this in mind that the following protocol is introduced: start with a $\epsilon = 0.5$ and reduce it by steps of 0.0045 to reach $\epsilon = 0.05$ during the hundredth episode. The results are shown in figure 6.

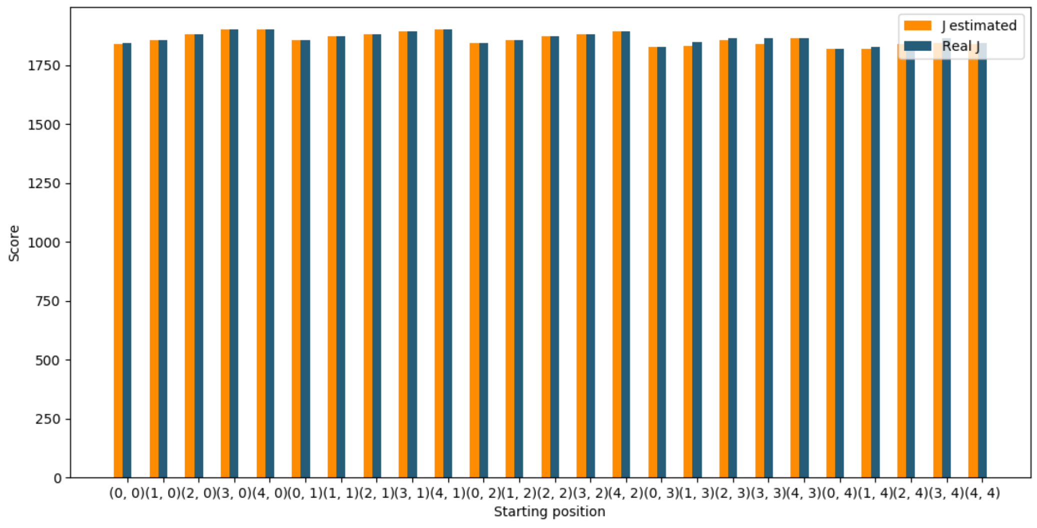


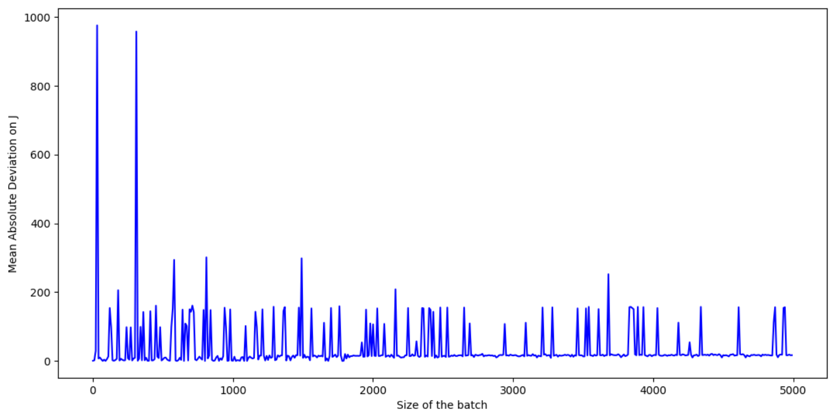
Figure 6: Expected cumulative reward computed using the Q-learning algorithm with a decreasing value of ϵ : from 0.5, to 0.05. 100 episodes of 1000 transitions

The error made on $J(x)$ is then 2.84. Without being totally convincing, it may be noted that this method can also provide very good results.

However, all these values are highly variable from one iteration to another. Generally speaking, it should be remembered that ϵ should not be valued too high, favouring exploration too much, or too small, favouring exploitation too much.

6.2.1 Influence of the batch size

As explained beforehand, the batch size is a parameter that influences the update frequency of $\hat{Q}(x, u)$ and therefore the corresponding policy according to which the actions are drawn (with a chance $1 - \epsilon$).
This section presents the results obtained for different batch sizes using a decreasing ϵ, from 0.5 to 0.05.



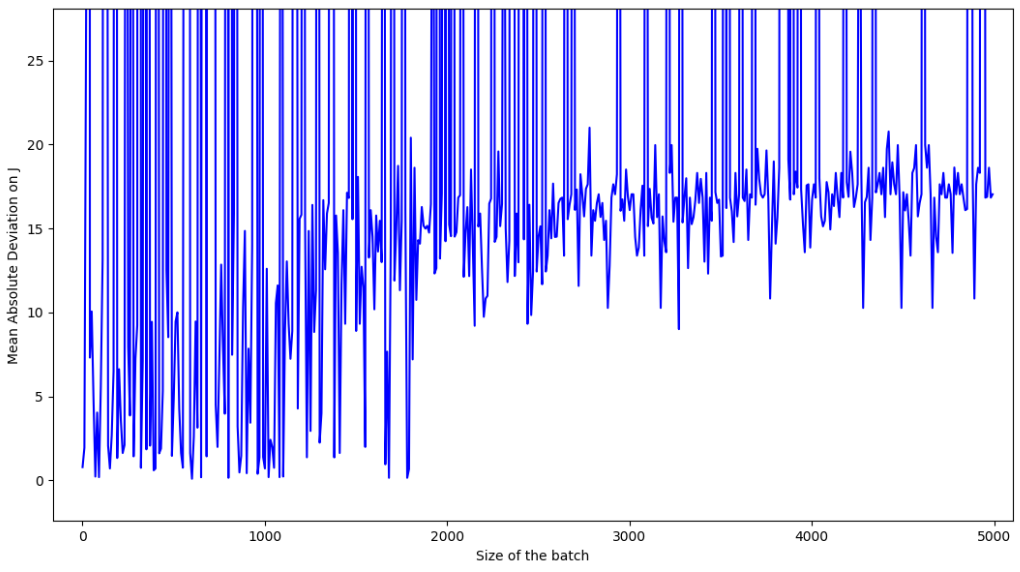


Figure 7: Expected cumulative reward computed using the Q-learning algorithm with different size of batch. 100 episodes of 1000 transitions. The second image is the same as the first one but zoomed.

Two important elements stand out from this figure. First, when the batch size is smaller, the average error on $J(x)$ is lower. However, for a smaller size, some iterations may not converge and therefore lead to very large errors.
Choosing the optimal value for this parameter is therefore tricky. A trade-off must be made between accuracy on the one hand and lower variance on the other.

Another observation that is not reflected in these graphs is that the smaller the batch size, the higher the computation time. This seems natural: even if the total number of updates remains the same, the number of times the optimal policy from Q is calculated decreases. However, the execution time remains low in most cases and should not be considered a major disadvantage.

The replay experience also has advantages that do not concern our implementation: to remove the dependency between experiments recorded when using Deep Q-Network. In addition, it allows the reuse of past experiences when it is expensive to acquire new ones.

6.3 Other policy

In order to influence the ratio between exploration and exploitation, several policies can be considered: *greedy approach*, *Boltzmann approach*,...
One of the simplest one remains however to chose the next action at random. Faced with the work already carried out within the framework of this project, this simpler approach was favoured.

From an implementation point of view, we just need to use the method developed in subsection 6.2 and set the parameter ϵ to 1. In doing so, we move back to the initial case where the algorithm was studied for a random sequence of trajectories 100 fixed in advance. The results are shown in the figure 8.

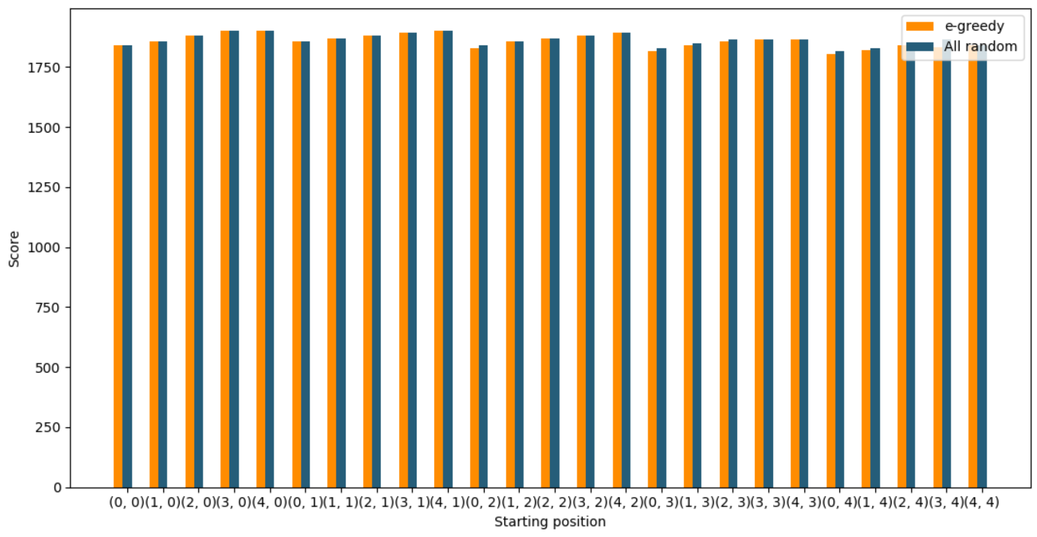


Figure 8: Comparison between J_{μ^*} and the expected cumulative reward for two different policy : all random or ϵ-greedy. 100 episodes of 1000 transitions. Batch size = 1000 (one update per episode).

It can be observed that the results do not differ. This is due to the number of transitions and the number of episodes. By reducing this number to 100 transitions with only 10 episode, the differences are much greater. This is reflected in the figure 9.

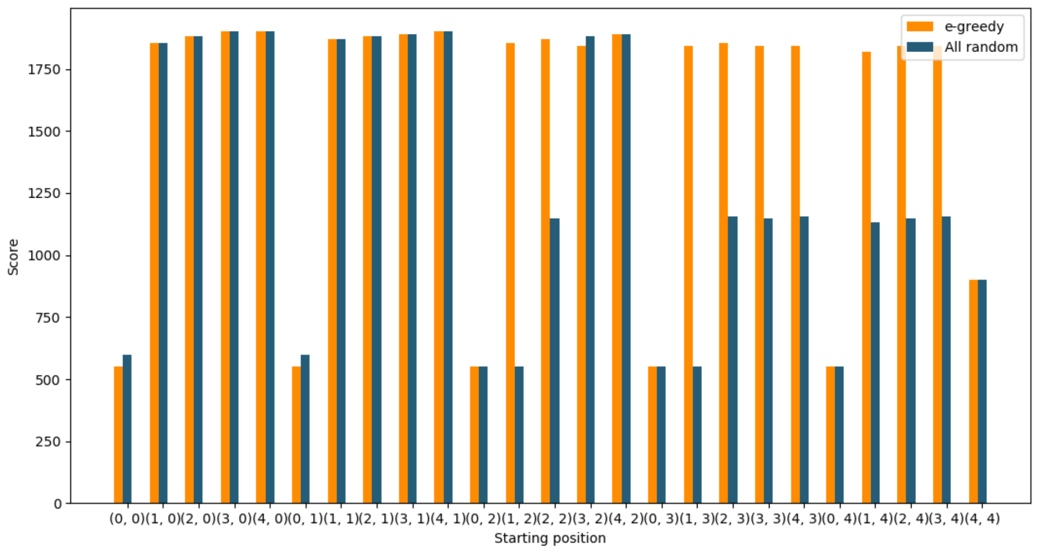


Figure 9: Comparison between J_{μ^*} and the expected cumulative reward for two different policy : all random or ϵ-greedy. 100 episodes of 100 transitions. Batch size = 100 (one update per episode).

When the number of iterations is reduced, it is normal that a policy, even simple as -greedy will work way better then just selecting every thing randomly

7 Discount factor

The discount factor represents the importance we give to the previous steps of our calculations. Therefore, in the deterministic case, it immediately appears that a smaller number of iterations will be necessary to achieve convergence towards the optimal solution.
In trying this, it can be observed that the convergence of cumulative rewards for questions 3 and 4 seems to be achieved for only 10 iterations. This can be explained very simply by using the following formula:

$$||J_\mu^N - J_\mu||_\infty \leq \frac{\gamma^N}{1-\gamma} Br$$

For $N = 10$, the error bound is below 0.00332. It takes 1320 iterations for $\gamma = 0.99$ to obtain the same error bound.
In question 5, the number of iterations can also be reduced. However, the size of the trajectory cannot be reduced, as these two parameters are indeed independent.
In the stochastic case, having a lower value of γ also allows you to have a better result in a smaller number of iterations.
It should also be noted that overall, the results obtained are generally much lower. This is simply explained by looking at the J_μ formula, which does not offer any compensation when γ decreases. From then on, J_μ decreases with γ.