Task 1

Implement the Q-learning algorithm, using the reward function as given in task1.mat and with the ε-greedy exploration algorithm by setting $\varepsilon_k, \partial_k$ and γ.

It is required to run 10 times and record the running time. The maximum number of trails in each run is set to 3000. The final output is an optimal policy and the associated reward.

In order to avoid that the learning rate is too small, I set a threshold for learning rate to early stop the program when less than 0.005. The difference of Q-values can help us find it is converged or not. Here, I set 0.005 as the threshold of two maximum Q-value difference. If it is less than the threshold, we can regard the this run has converged and then terminate the trails to process the next run.

Obtained Results:

epsilon alpha	No. of goal-reached runs		Execution time (sec.)	
	gamma = 0.5	gamma = 0.9	gamma = 0.5	gamma = 0.9
1/k	0	0	N/A	N/A
100/(100 + k)	0	7	N/A	38.8805
(1 + log(k))/k	0	0	N/A	N/A
(1 + 5log(k))/k	0	5	N/A	33.5326

Optimal Trajectory Output:

The final reward is 1861.9157

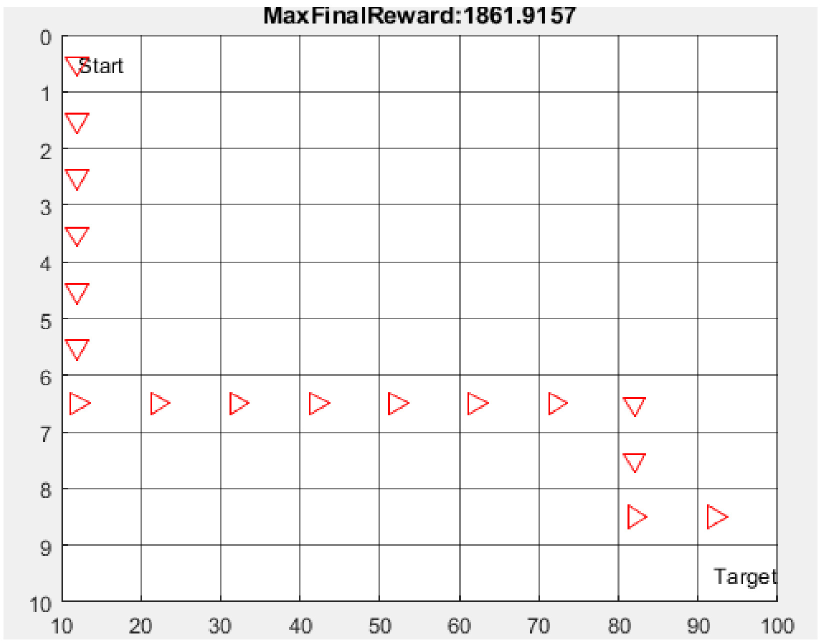

Figure 1. Robot Trajectory

Comments:

a. From the table, only two conditions can reach the goal within 10 runs. The parameter of discount rate is 0.9 in these two satisfied situations. Therefore, it means, given this reward matrix, we have to set up a larger discount rate which takes into account future rewards more strongly.

b. Compared the learning rate/exploration rate curve from two satisfied situations with the other two, the larger rate can help reach the optimal goal. As you can see from the following curve figures (Figure 2), the rate calculated from 100/ (100 + k) and (1 + 5log(k))/k are larger than the rest. Even for far future steps (Figure 3), these two can still generate larger learning rate than the others and update the Q-value.

c. Here I implemented the ε-greedy exploration. For those feasible actions with maximal Q-values in a given state, I assigned the probability of (1 – exportation rate) to them. For the rest, I assigned the probability of (exploration rate/ count_number). However, when there are more than two maximal Q-values, we have to randomly select one rather than selecting the first appeared one.

In conclusion, for the task 1 with this given reward matrix, we should select the rate function that can generate larger and more stable values. The same as the discount factor since we care more about the future steps in this situation.

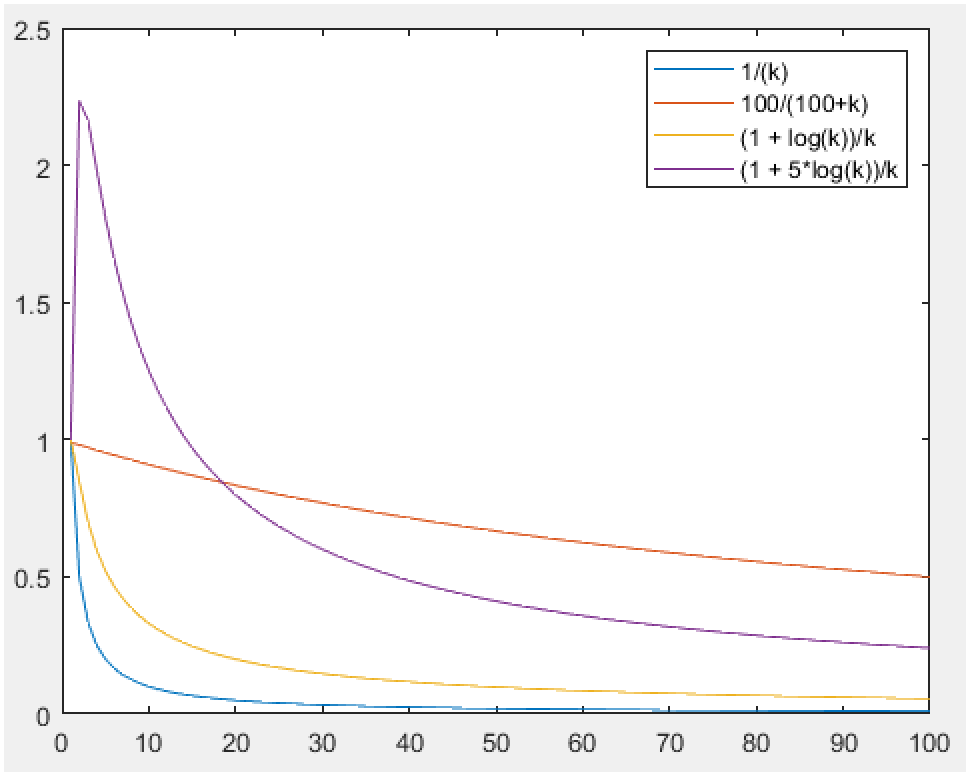

Figure 2. Curve comparison within 100 iterations

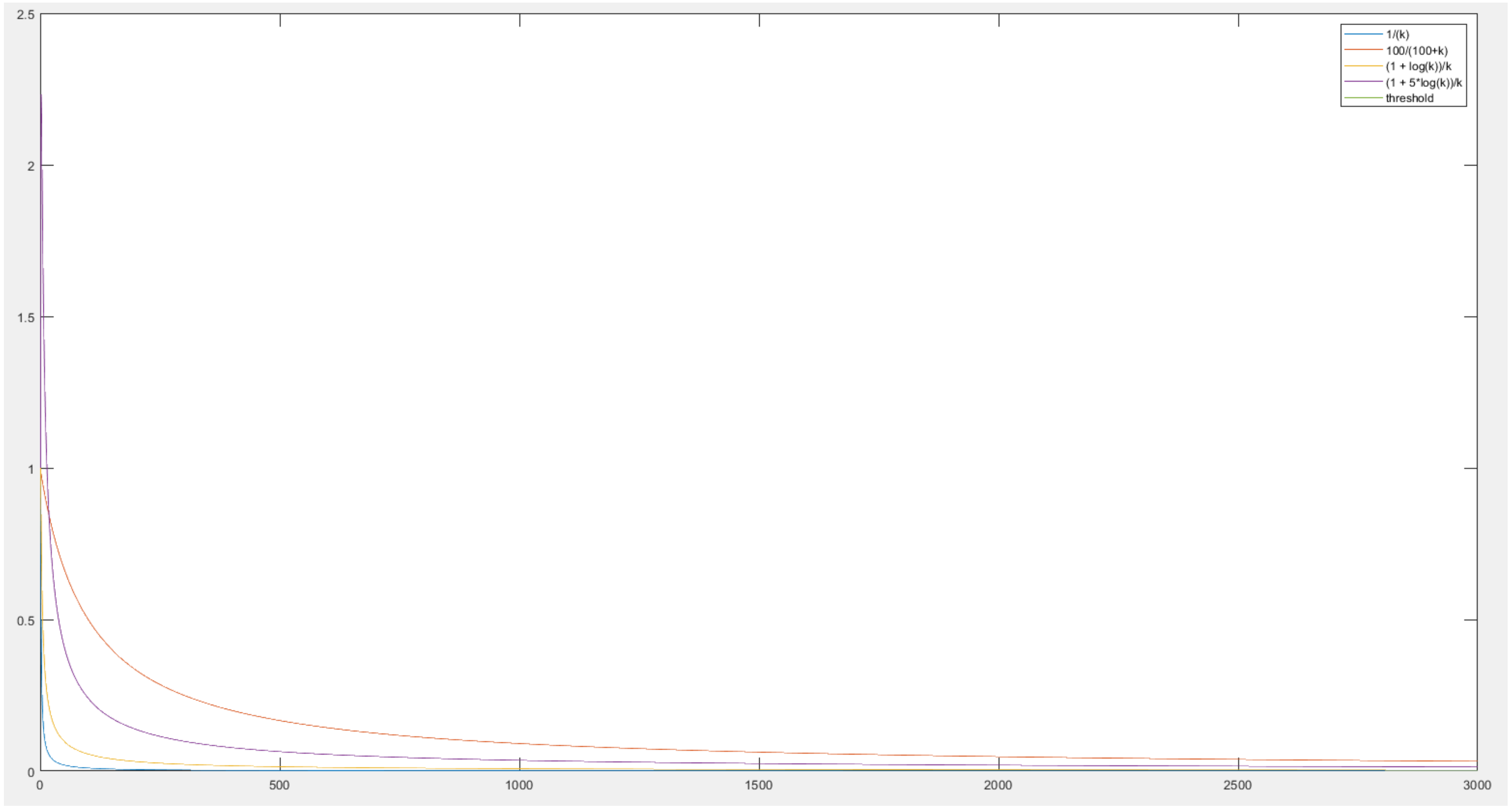

Figure 3. Curve comparison within 3000 iterations

Task 2

Own parameter setting for unknown evaluation test.

Parameters tuning:

Here I tuned the parameter of discount rate, learning rate functions, and exploration rate to find the optimal policy with least running time.

a. Learning rate functions.

	Random created MAT		Task 1 MAT	
epsilon alpha	No. of goal-reached runs	Execution time (sec.)	No. of goal-reached runs	Execution time (sec.)
	gamma = 0.9	gamma = 0.9	gamma = 0.9	gamma = 0.9
exp(-0.001k)	**10**	**0.74756**	**10**	**0.79637**
exp(-0.01k)	6	0.95567	7	1.1989
exp(-0.0001k)	10	2.5136	10	3.115
exp(-0.005k)	10	0.90757	10	0.8492
100/(100 + k)	9	31.6223	9	41.2752
1000/(1000 + k)	10	46.198	10	38.6753
0.5	10	0.88352	10	0.9494
0.8	10	4.3945	10	4.5086

Based on this results, I select exp(-0.001k) as the learning rate function.

b. Discount rate.

	Random created MAT		Task 1 MAT	
gamma	No. of goal-reached runs	Execution time (sec.)	No. of goal-reached runs	Execution time (sec.)
	exp(-0.001k)	exp(-0.001k)	exp(-0.001k)	exp(-0.001k)
0.9	10	0.74756	10	0.79637
0.8	10	0.70341	10	0.81875
0.75	**10**	**0.605**	**10**	**0.79878**
0.7	0	/	10	0.80433

Based on this discount rate table, I found 0.75 can have better performance.

c. Various fixed exploration rate with exp(-0.001k) as learning rate function.

	Random created MAT		Task 1 MAT	
Exploration rate	No. of goal-reached runs	Execution time (sec.)	No. of goal-reached runs	Execution time (sec.)
	exp(-0.001k), gamma=0.75	exp(-0.001k), gamma=0.75	exp(-0.001k), gamma=0.75	exp(-0.001k), gamma=0.75
0.3	9	1.0895	5	3.4908
0.5	10	0.27857	10	0.91737
0.7	**10**	**0.40077**	**10**	**0.42599**

Here I found when exploration rate is fixed as 0.5, it has least execution time in random created reward matrix. But it doesn't for Task 1. However, when exploration rate is set as 0.7, task 1 can have least running time and time of random created mat is not bad. They all run less time than (a) where the parameter is changing for each iteration (around 0.7). Thus, I select 0.7 as the fixed exploration probability.

In conclusion, my parameter setting is as below.

Learning rate function: exp(-0.001k), where k is iteration when attempting to reach end state.

Discount rate: 0.75

Fixed exploration probability: 0.7