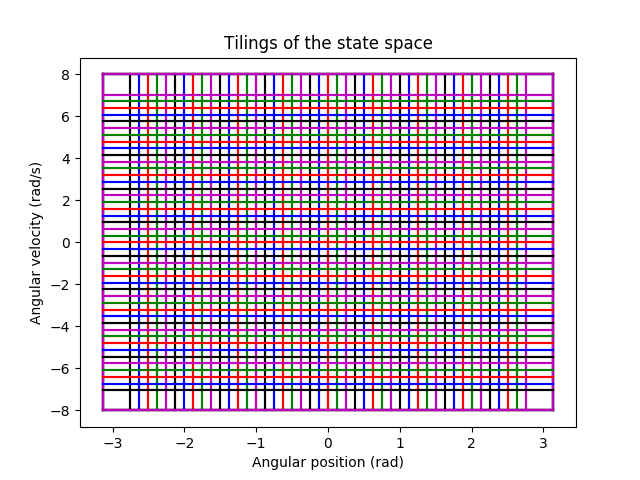
**Reinforcement Learning Assignment 2**

**Function Approximation 2.a**

For this experiment, we set the discount rate, γ = 0.9. We also set the amplitude of the torque being applied to the absolute value of 0.7. The policy followed was the one that with 0.9 probability, torque will be applied in the same direction as the angular velocity and with 0.1 probability, torque will be applied in the opposite direction. When angular velocity is equal to 0, torque was applied randomly in any direction.

The states are composed of angular position (theta) ([-pi,pi]) and angular velocity ([-8,8]). The state space was discretized in 10 bins for angular position and 10 bins for angular velocity. There were 5 overlapping tilings of the state space, as one can observe in Figure 1. Each set of colors represents one tiling (black, red, blue, green, and magenta). The weights of each tile were randomly initialized between [-0.001 , 0.001].



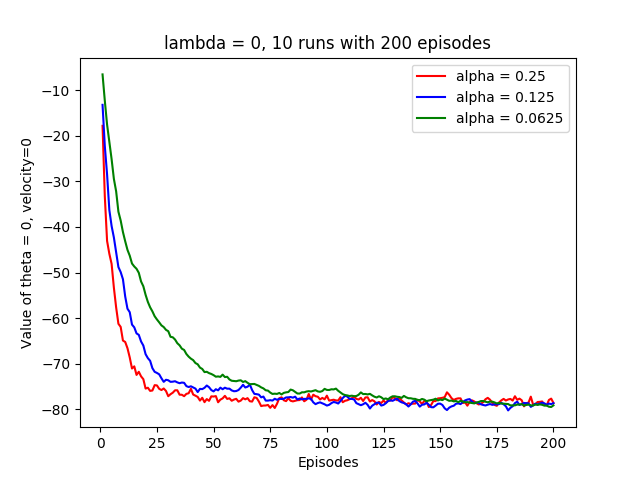
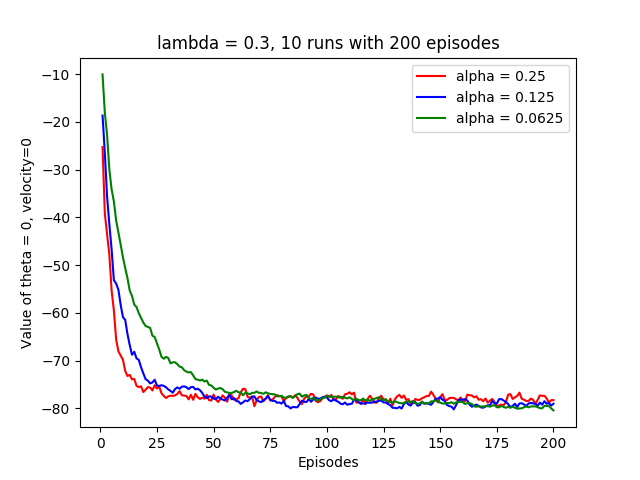
***Figure 1*: Tilings of the state space**

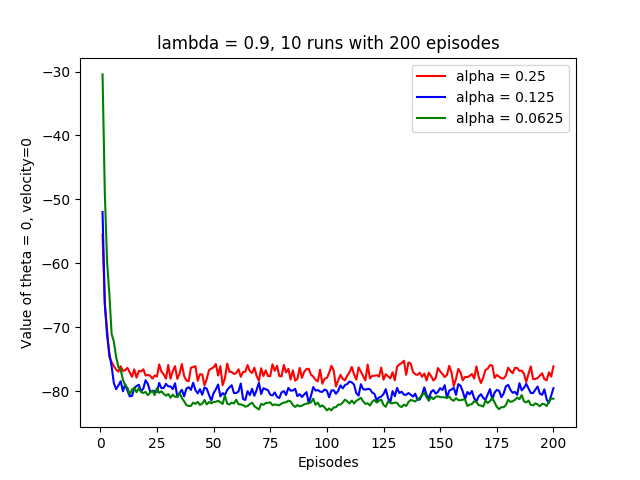
For the experiment five λ values were evaluated (0, 0.3, 0.7, 0.9 and 1). For each of these λ values, 3 learning rates α were evaluated (1/4, 1/8, and 1/16). For each combination of λ and α, 10 independent runs (random seeds memorized and reused for each combination of parameters) consisting of 200 episodes were done. The state values were reset for each run and the eligibility traces reset for each episode. Each episode was restarted at the same state. The TD(λ) algorithm implemented is the one from page 293 of the Sutton book. Since it is unclear what 0 for angular position means we ran 2 experiments.

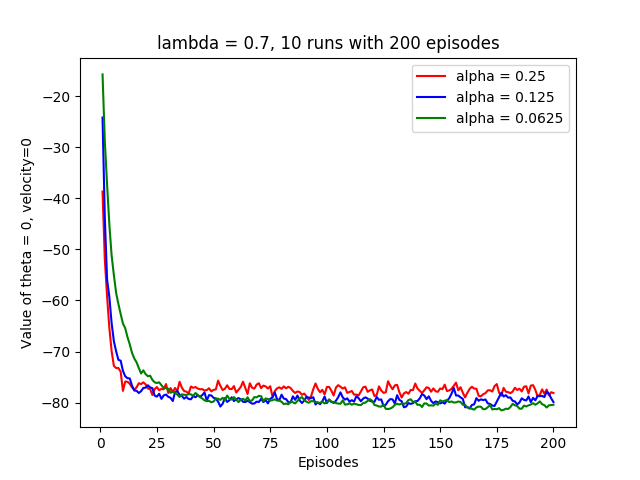
1. State0 = angular position is pi (pendulum at bottom) and velocity = 0 rad/s
2. State­0 = angular position is 0 (pendulum at top) and velocity = 0 rad/s

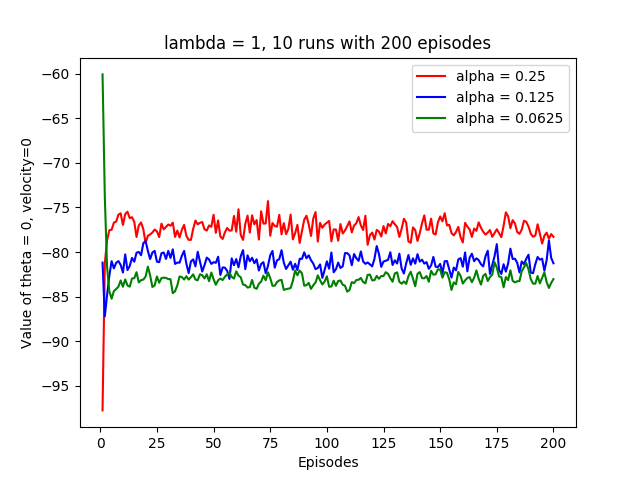
**Expirement 1:**

The following 5 graphs show the results for each λ for experiment 1.









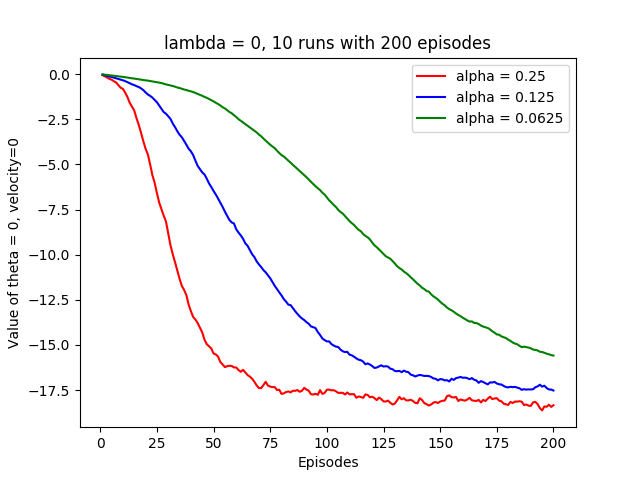
One can observe from the graphs that all combinations of parameters seem to be converging towards a negative value of about 80 (target) for this policy when starting at the bottom. The further the pendulum is from being balanced at the top, the, less the reward will be. We can clearly see that the policy of applying torque in the same direction of velocity will never make the pendulum balance but will make it spin in one direction.

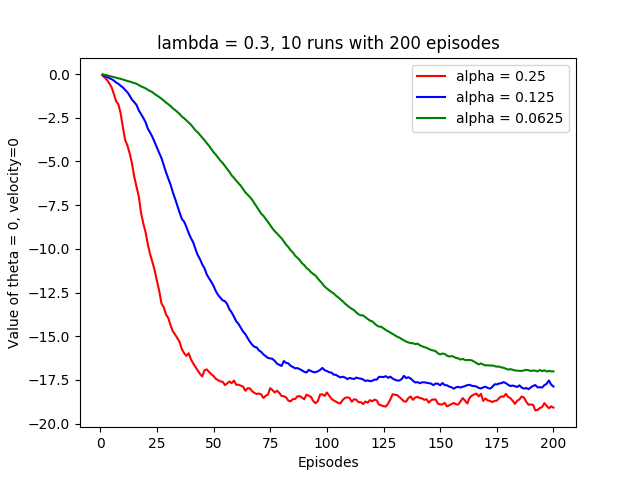
We can see that a bigger learning rate helps the agent converge to a value for the state faster but with bigger variance (noise). We can observe this since the smallest learning rate has the smoothest and most stable curve, but reaches the convergence target after more episodes, whereas the largest learning reaches it in fewer episodes but is noisier even after convergence.

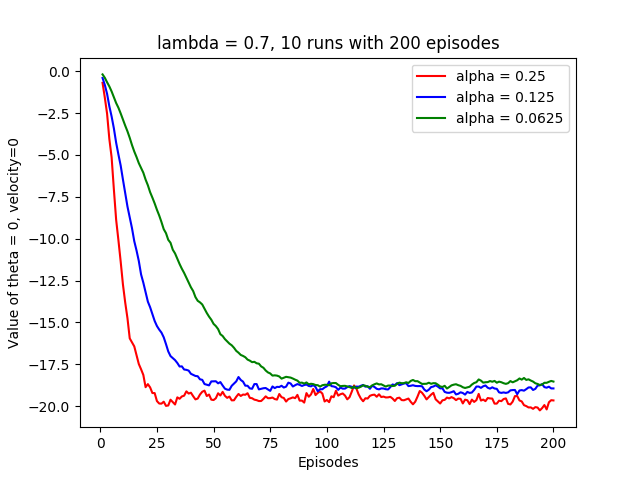
One can also see that the value of λ has a big impact on the convergence rate of the algorithm. When λ=0, we find TD(0) and when λ=1, we obtain MC. However for this method of MC, the values are updated at each step of the episode and not at the end of the episode. It can be observed that an intermediate value of λ (0.7) has better performances. It is also observed that as λ increases so does the rate of convergence. This makes sense since in this particular environment, the start state’s total reward depends on all the next steps and may be visited multiple times in one episode. Each step actually generates a reward so by taking into account more steps allows for the algorithm to converge faster. When λ is small, less successive steps and their rewards are considered in the update since the eligibility traces decay with each step if not reset. Therefore more visits to the start state are required to converge to the value since the eligibility traces are reset, while when λ is greater less visits are required since the eligibility traces decay less and therefore more steps and their rewards are considered in the updates. This is why a greater λ converges faster for this environment. However, after a certain value of λ the performance drops which is in line with Sutton’s empirical analysis of the TD(λ) algorithm.

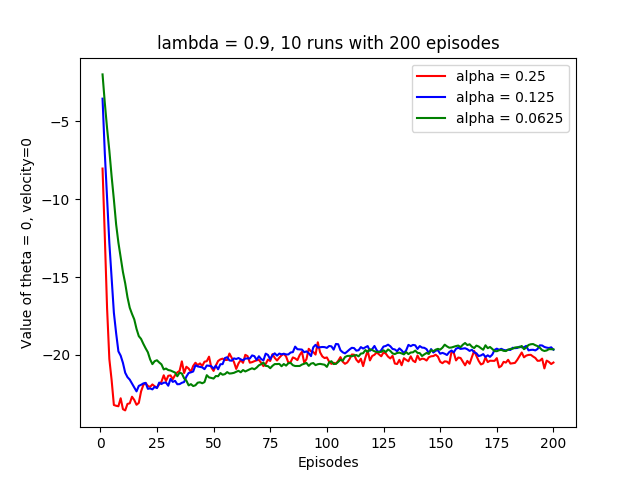
We also notice that TD(1) which is also MC converges to a higher target for certain learning rates (0.25). This is closer to the true value since MC converges closer to the true value of the training examples than TD(0), but does not generalise as well as TD(0).

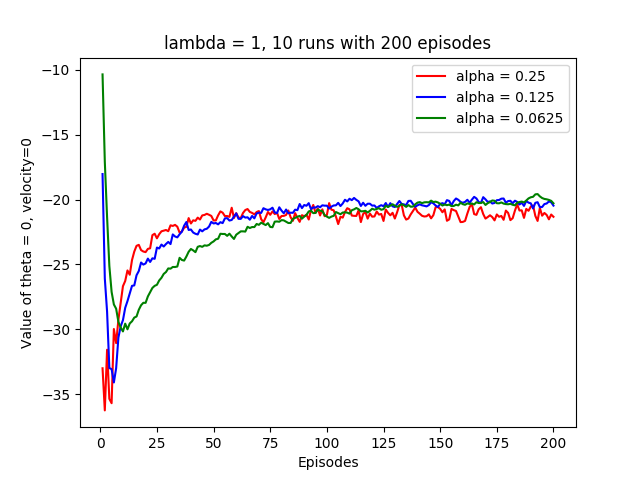
**Experiment 2:**

The following 5 graphs show the results for each λ for experiment 2.









One can see from the five graphs that the same behaviour as in experiment one takes place but this time the value at which the algorithm converges is around -18. We also see that the convergence is much slower. Again this can be attributed to the fact that a state needs to be visited more often to converge faster. In experiment 1, we started at the bottom of the pendulum and stationary. The pendulum then goes back and forth picking up speed with each swing because torque is applied to it in the same direction. At the beginning, the state corresponding to an angular position of almost pi and 0 velocity is visited a lot thus resetting the eligibility traces for the tiles corresponding to the start state more often and thus having updates with greater impacts. In experiment 2 we start at the top and then go into one direction and spin in that direction. Since torque is applied continuously to the pendulum and toque is force, it causes the pendulum to accelerate in one angular direction. Revisiting the start states, or states that are similar to it happens rarely since angular speed increases and then maxes out at 8 rad/s and the start state is 0 rad/s. The eligibility traces corresponding to the tiles of start state are rarely reset thus making less meaningful updates and therefore converging slower. When λ=1, the tiles corresponding to the start state are updated at each step. However, since the state is rarely visited the eligibility trace for that state decays and rarely resets making the updates les impactful with each step and thus explains why it converges slower than in experiment 1.

The reason why it converges at a higher value that experiment 1 is because it obtains greater rewards at each step since the rewards are a function of being close to the top and experiment 1 spends a lot more of its time at the bottom (negative rewards).

**Conclusion**

One can clearly see that the learning rate affects the speed of convergence but also the precision of that convergence. A higher learning rate convergences faster but creates more variance on the convergence value (noisy).

One can also observe that λ has in impact on the convergence. Intermediate values of λ provided better performances.

Finally from comparing the convergence rates of the 2 experiments one can observe that more visits to the state helps convergence since eligibility traces for that states are reset more often thus rendering the successive updates more impactful.