CS 5180: Exercise 5
Harin Kumar Nallaguntla

1.a. The mentioned scenario book proves to be highly effective for Temporal Difference (TD) learning due to the stability of states in the highway section, allowing for a consistent estimation of state values. TD facilitates swift adjustments to new states, with each updated state expediting the refinement of its predecessor. In contrast, Monte Carlo necessitates traversing to the end and averaging improvements across all states, including the highway sections that require minimal adjustments but are still affected by Monte Carlo's fluctuations. TD also excels in scenarios where reaching the terminal state is challenging. Even under the horizon method, which omits states with negligible value decay, traditional Monte Carlo remains inefficient, if not impractical.

b. Consider the game of chess, where the episodic structure of the gameplay favors the application of Monte Carlo (MC) methods. Chess games unfold as finite sequences of moves, leading to a conclusive win, loss, or draw. The episodic nature of chess aligns seamlessly with MC methods, allowing the agent to simulate entire games and update state value estimates based on the actual outcomes. This distinctive feature empowers MC to effectively leverage the final result as a crucial signal for learning, eliminating the need for intermediate value estimates during the ongoing game.

Moreover, the practicality of MC methods in chess is underscored by their alignment with the game's rules, where value estimates are naturally derived from playing complete games. In contrast, Temporal Difference (TD) learning may be less suited for chess due to its reliance on estimating values through the expected sum of rewards over time, posing challenges in providing accurate value estimates in a dynamic game with long-term consequences for each move. The episodic structure of chess, coupled with the direct availability of game outcomes, positions MC as a more fitting choice for effective learning and improvement in the game.

2.a. Q-learning is classified as an off-policy algorithm due to its methodology of updating Q-values by considering the Q-value of the next state (s) and the action taken greedily (a). In essence, Q-learning estimates the return, encompassing total discounted future rewards, for state-action pairs under the assumption that a greedy policy is pursued, despite the algorithm not strictly adhering to a greedy policy itself. Conversely, SARSA is characterized as an on-policy algorithm because it updates its Q-values using the Q-value of the next state (s) and the action dictated by the current policy (a). SARSA estimates the return for state-action pairs under the assumption that the ongoing policy is consistently followed.

b. When actions are determined by a greedy strategy, the algorithms appear identical at a superficial level, but variations in action choices and weight updates can emerge due to the arbitrary initialization of Q and S. For example, if random values within the range (0, 1] are assigned to each state-action pair as Q(S, A), the selection of greedy actions will differ in each instance. The divergence in action selection leads to disparate updates, and given that neither algorithm explores, there is no guarantee of convergence to the same solution.

3.a. The result of the first episode suggests the episode terminated at the left block for no reward. The value of state A was updated as follows:

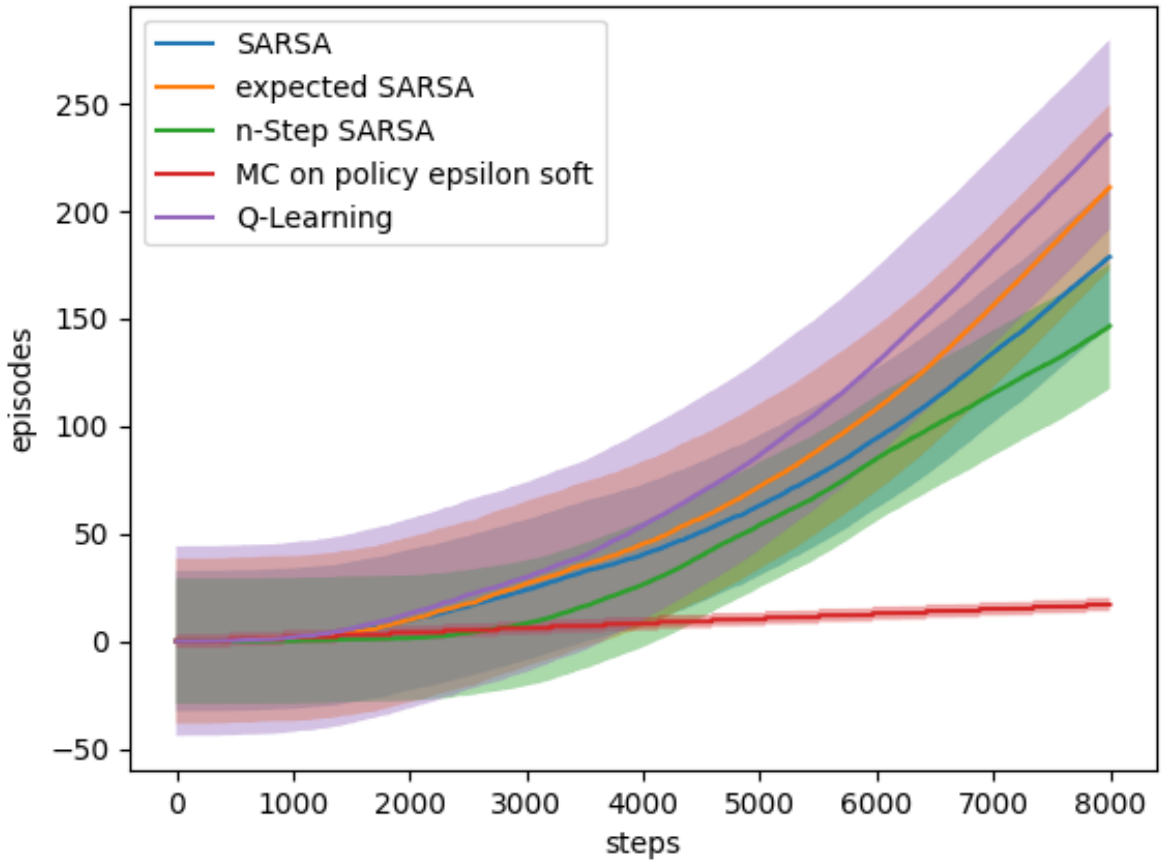
$$\begin{aligned} V(A) &= V(A) + \alpha \left[R_{t+1} + V(S_T) - V(A)\right] \\ &= 0.5 + 0.1 \left[0 + 0 - 0.5\right] \\ &= 0.45 \end{aligned}$$

b. No, the convergence criterion for both TD and Monte Carlo is a suitably small alpha. Therefore, it can be asserted that a small alpha in both TD and Monte Carlo consistently outperforms larger values in the long run, approaching the minimum achievable error limit. The presented alpha has already reached the performance limit for each method, indicating that there is no universally optimal fixed alpha for substantial improvements. Conversely, one might anticipate that a dynamic alpha or varying weights for different states could be advantageous.

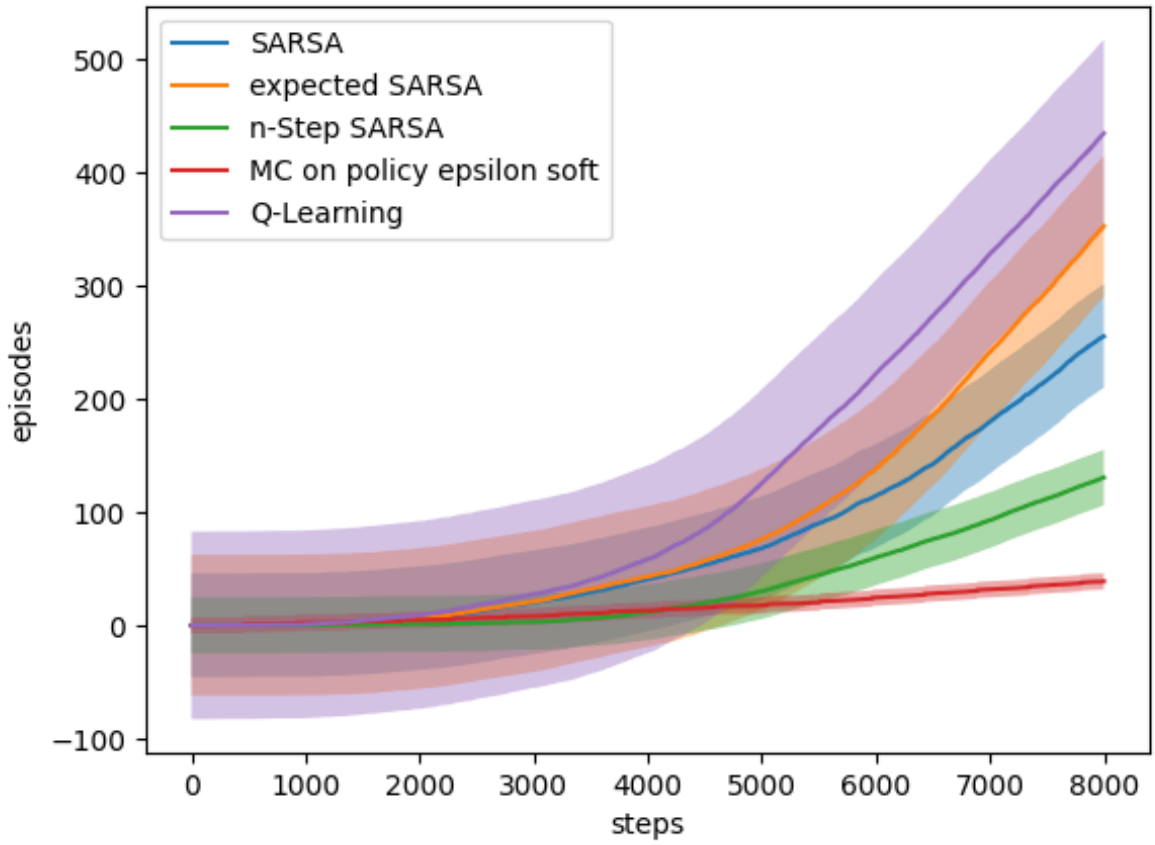
c. As previously mentioned, I believe that this phenomenon is inevitable with high alpha values, as the emphasis placed on the TD error tends to magnify minor errors. This amplification effect leads to the estimated value function oscillating between values without converging. Consequently, the estimate for V(c), initially set at its true value in this illustration, gradually deviates from the accurate estimate, thereby augmenting the RMS error.

d. Had the smaller 5-state random walk been employed, the optimal value for "n" would likely have been smaller. In the given scenario, selecting "n" as 3 and observing an episode starting at C, transitioning through C → D → E → Terminal would result in updating C toward the reward of 1, leading to an inaccurate estimation of the true value of C. It appears that a probable optimal choice in this context would be "n = 2," suggesting a general assumption that, for smaller state-spaces, smaller values of "n" are more suitable. If "n ≥ 2," then for longer episodes, updates would no longer bootstrap on other state values but instead rely on their own values. Additionally, the alteration of the left value to -1 from 0 in this example likely diminished the optimal value of "n." By setting the left value to -1, states on the left side of the walk are effectively closer to a reward, reducing the need for updates to back-propagate extensively to enhance the value of these states.

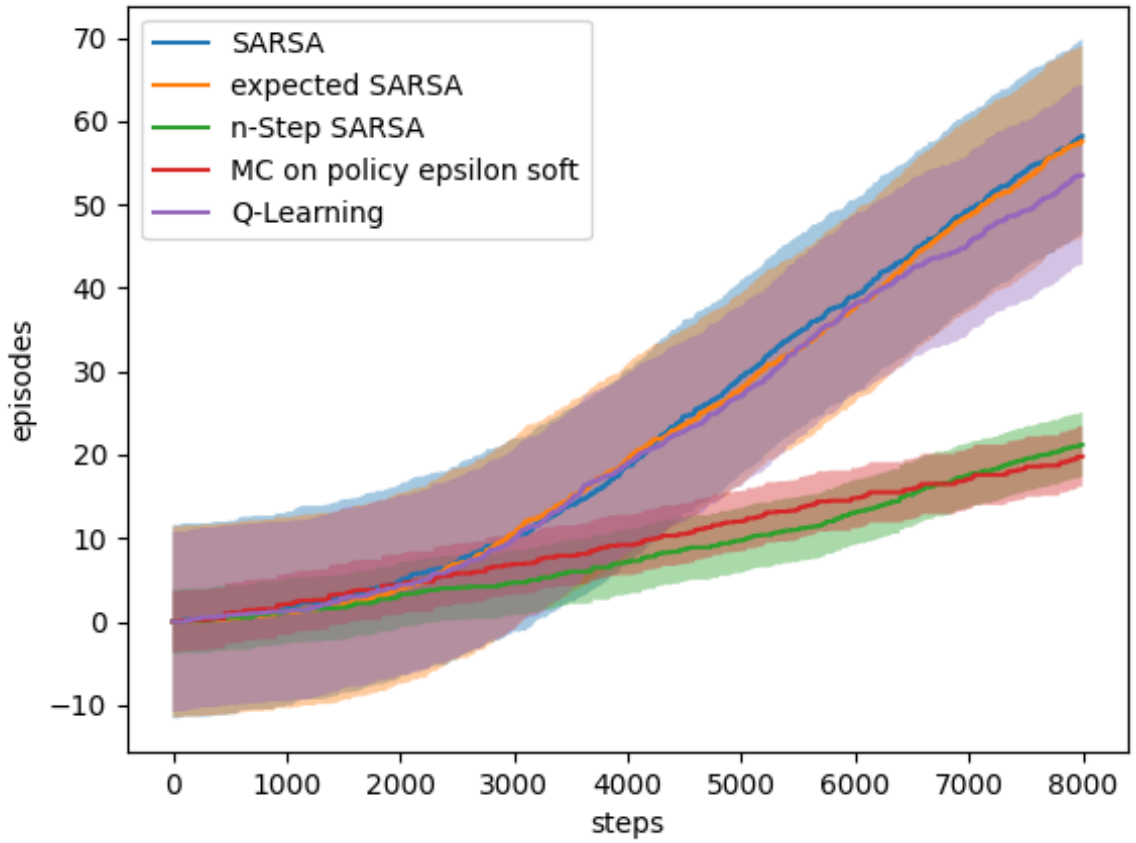
4.b



c.

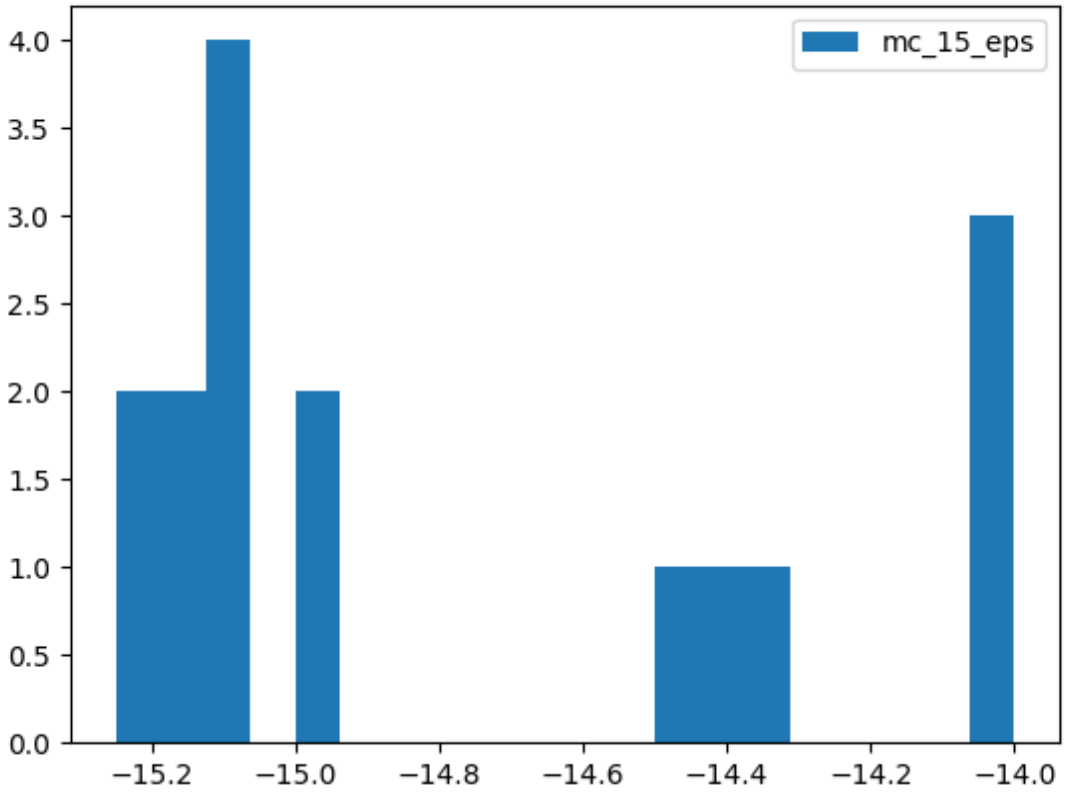

SARSA
expected SARSA
n-Step SARSA
MC on policy epsilon soft
Q-Learning
episodes
steps
500
400
300
200
100
0
−100
0
1000
2000
3000
4000
5000
6000
7000
8000

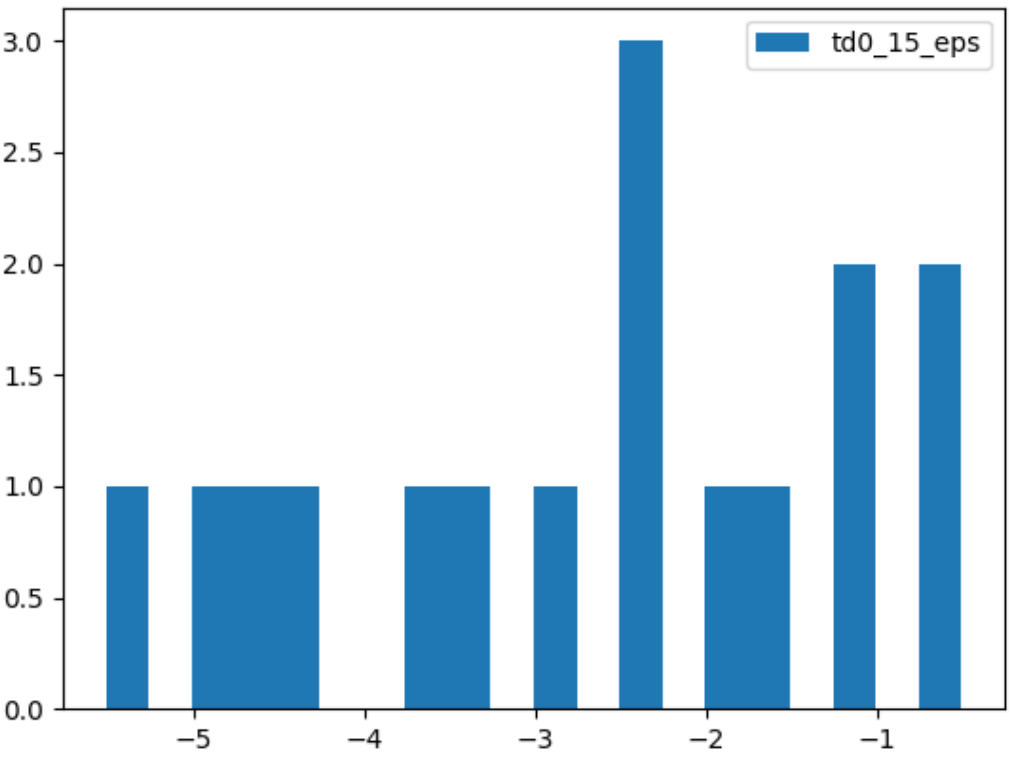
d.

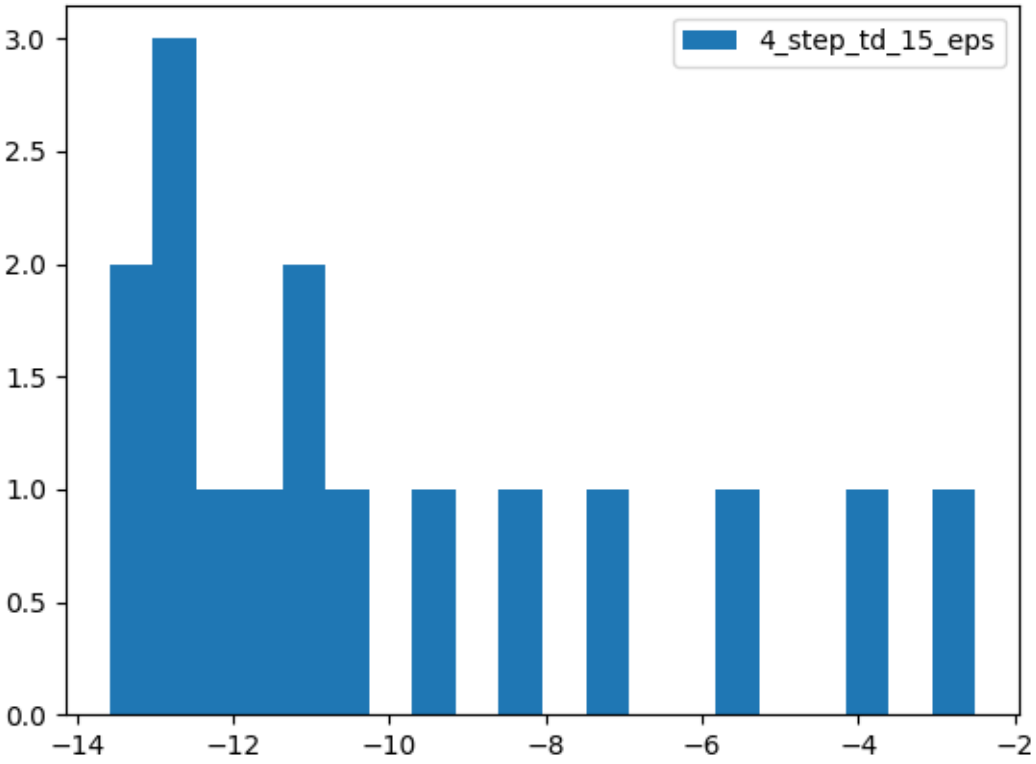

SARSA
expected SARSA
n-Step SARSA
MC on policy epsilon soft
Q-Learning
episodes
steps
70
60
50
40
30
20
10
0
−10
0
1000
2000
3000
4000
5000
6000
7000
8000

5.a. I used Q-learning to get optimal stochastic policy

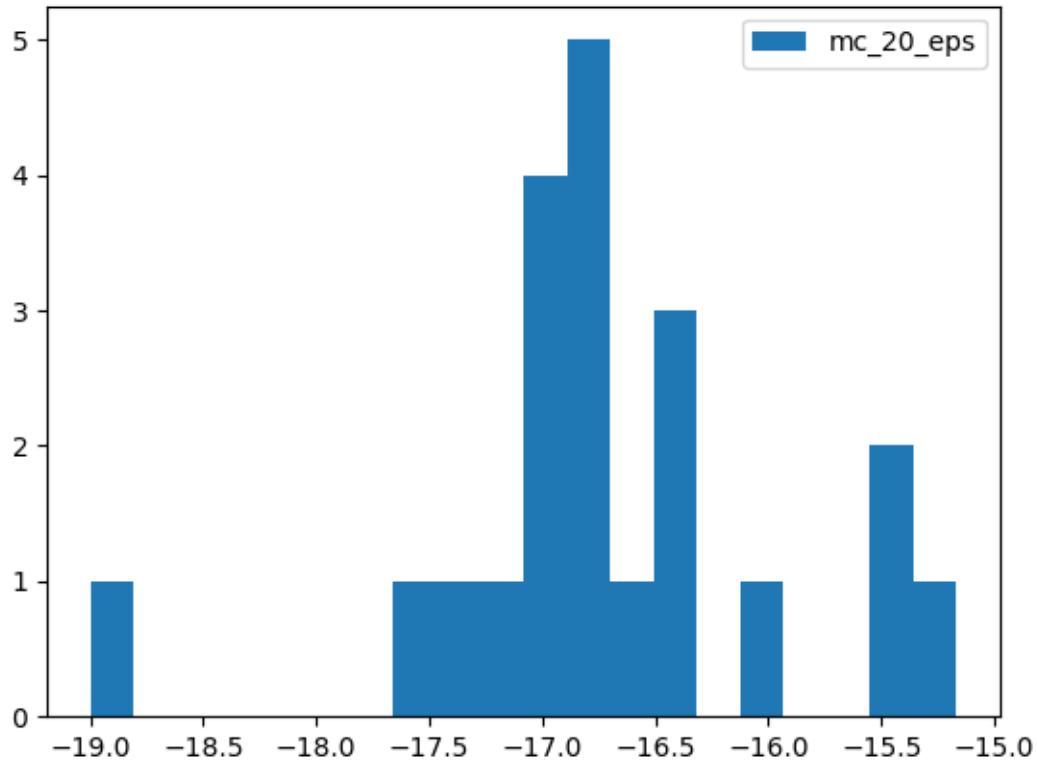
for 15 epsisodes:

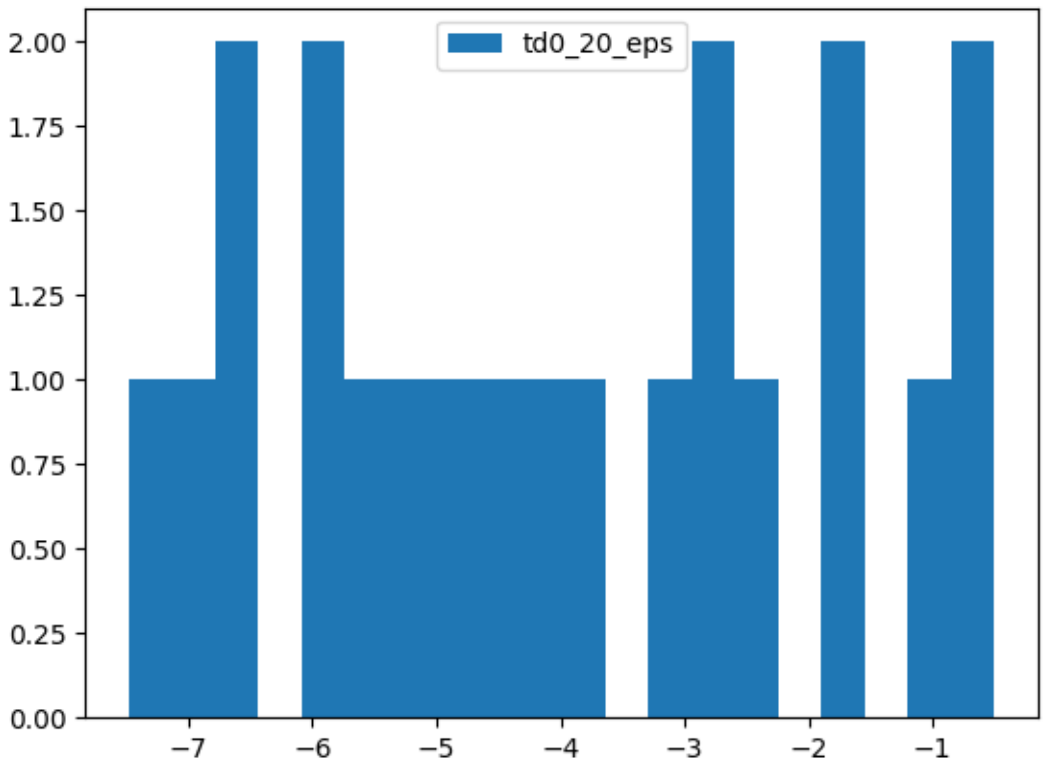


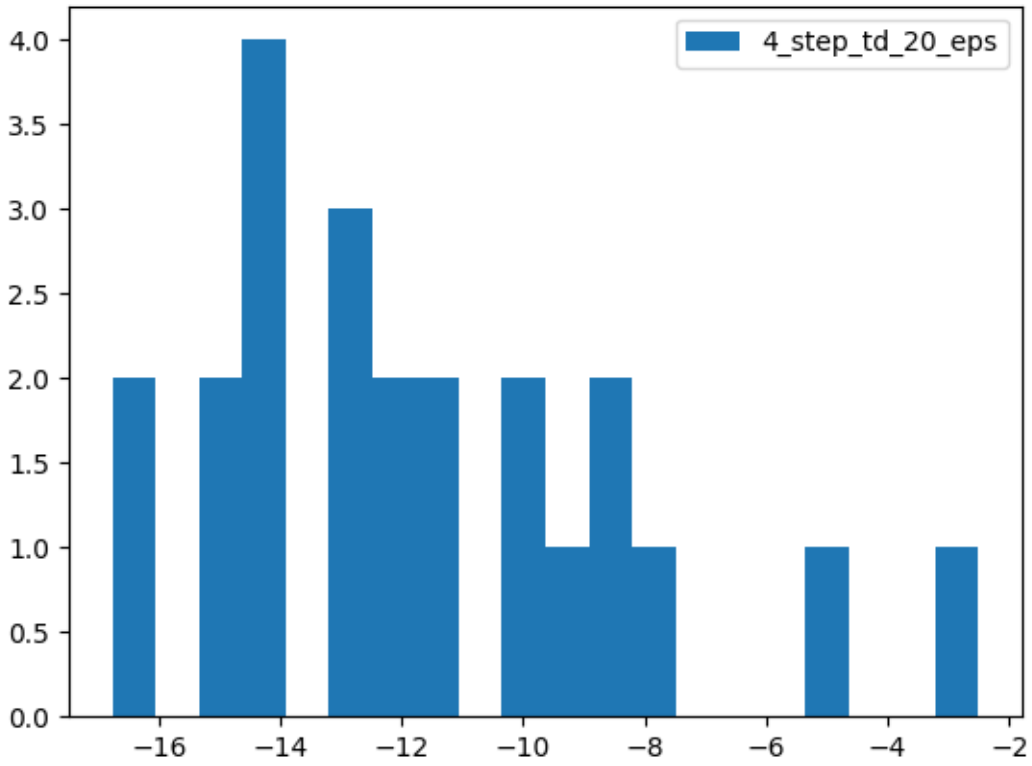




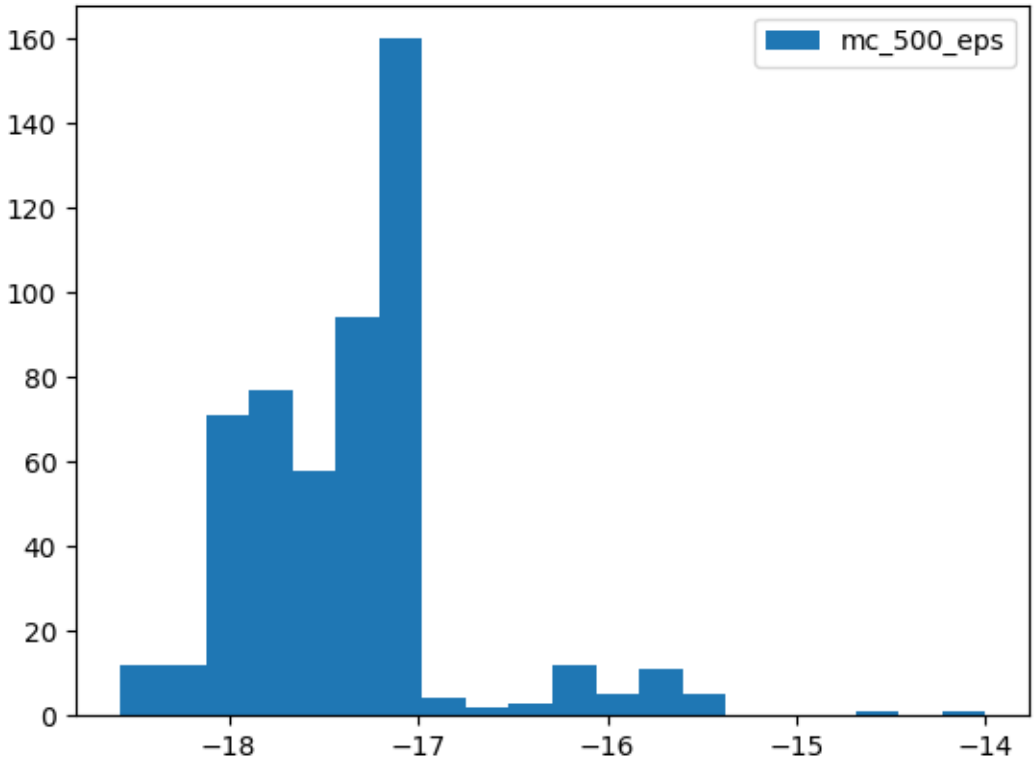
for 20 episodes:

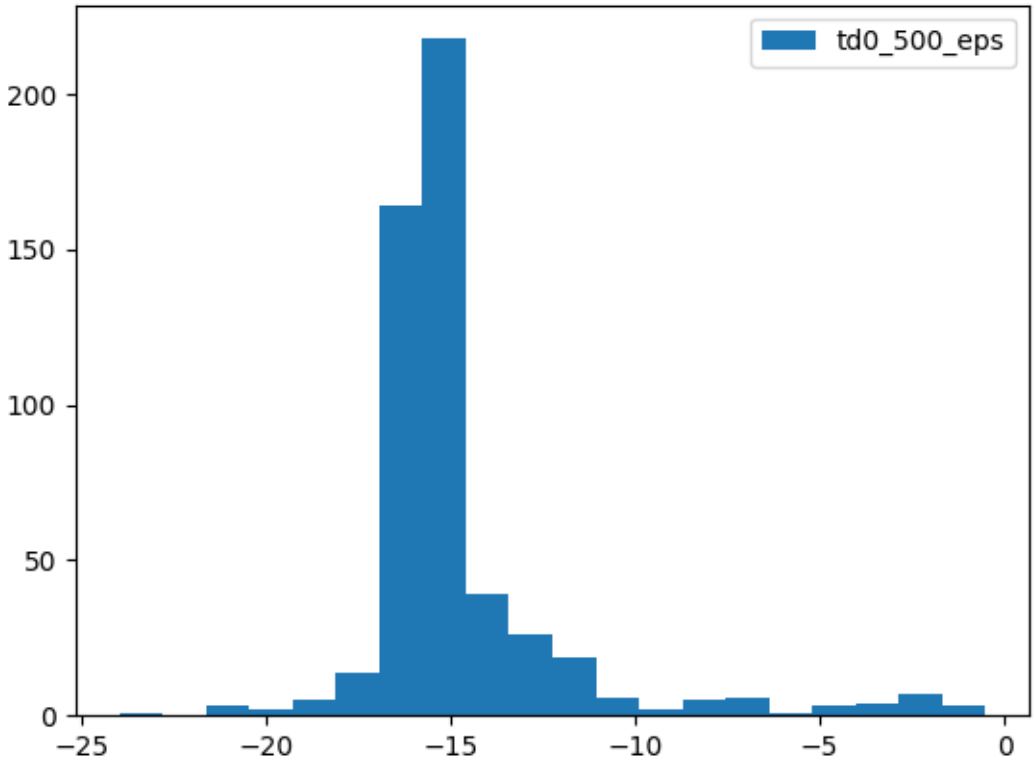


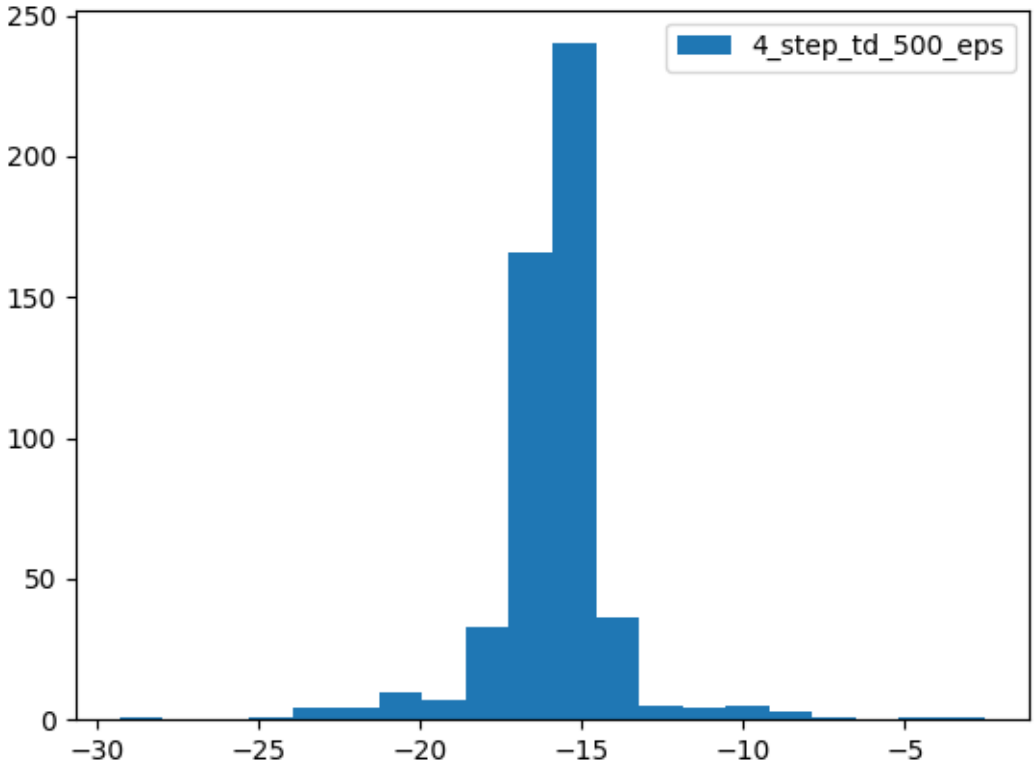

td0_20_eps
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−7
−6
−5
−4
−3
−2
−1


4_step_td_20_eps
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−16
−14
−12
−10
−8
−6
−4
−2

for 500 episodes:







b. Monte-Carlo Method:

- High variance: Since Monte-Carlo methods are typically high-variance, we observe a wide range of learning targets in the histograms. This is because Monte-Carlo uses the full return G as the target, leading to more variability in the estimates.
- Unbiased: Despite the high variance, the estimates are unbiased over a large number of episodes, as they are based on the true return.

TD(0) Method:

- Lower variance: TD(0) has lower variance compared to Monte-Carlo. We observe a narrower distribution of learning targets.
- Some bias: TD(0) introduces bias by using the immediate reward and the estimated value of the next state as the target. This bias results in a slight deviation from the true values.

4-step TD Method:

- Trade-off: n-step TD strikes a balance between bias and variance. With n=4, we see less variance compared to Monte-Carlo but possibly more than TD(0).
- The impact of n: Trying different values of n can show how the trade-off changes with the length of the trajectory considered.

Effect of Training Episodes (N):

- With a small number of training episodes (e.g., N=1), we observe larger variances in all methods, especially in Monte-Carlo.
- The bias introduced by TD methods becomes more apparent with fewer training episodes.