CS698R: Deep Reinforcement Learning

Mid-Semester Exam

Name: Romit Mohanty
Roll NO.: 190720

Solution to Problem 1: Random-Maze Environment Implementation

1. Actions:
   0:left
   1:up
   2:right
   3:down

   Environment is correct as can be seen from the implementation.
   timeStamp:0 CurrState:8 Action:0 NextState:8 Reward:-0.04 Score:-0.04
   timeStamp:1 CurrState:8 Action:3 NextState:9 Reward:-0.04 Score:-0.08
   timeStamp:2 CurrState:9 Action:3 NextState:9 Reward:-0.04 Score:-0.12
   timeStamp:3 CurrState:9 Action:0 NextState:8 Reward:-0.04 Score:-0.16
   timeStamp:4 CurrState:8 Action:1 NextState:4 Reward:-0.04 Score:-0.2
   timeStamp:5 CurrState:4 Action:3 NextState:8 Reward:-0.04 Score:-0.24000000000000002
   timeStamp:6 CurrState:8 Action:3 NextState:8 Reward:-0.04 Score:-0.28
   timeStamp:7 CurrState:8 Action:2 NextState:9 Reward:-0.04 Score:-0.32
   timeStamp:8 CurrState:9 Action:1 NextState:10 Reward:-0.04 Score:-0.36
   timeStamp:9 CurrState:10 Action:0 NextState:10 Reward:-0.04 Score:-0.39999999999999997
   timeStamp:10 CurrState:10 Action:3 NextState:10 Reward:-0.04 Score:-0.43999999999999995
   timeStamp:11 CurrState:10 Action:2 NextState:11 Reward:-0.04 Score:-0.4799999999999999
   timeStamp:12 CurrState:11 Action:0 NextState:11 Reward:-0.04 Score:-0.5199999999999999
   timeStamp:13 CurrState:11 Action:1 NextState:7 Reward:-1 Score:-1.52

   Most of the time agent goes in the desired direction except some cases like timestamp 11 where in spite of taking a left action it comes to 11. Also we can see rebounding from the boundary and the wall at 5 in time step 8 On the other hand if I set the goInDirection probability to 1 we can see no stochasticity as expected.

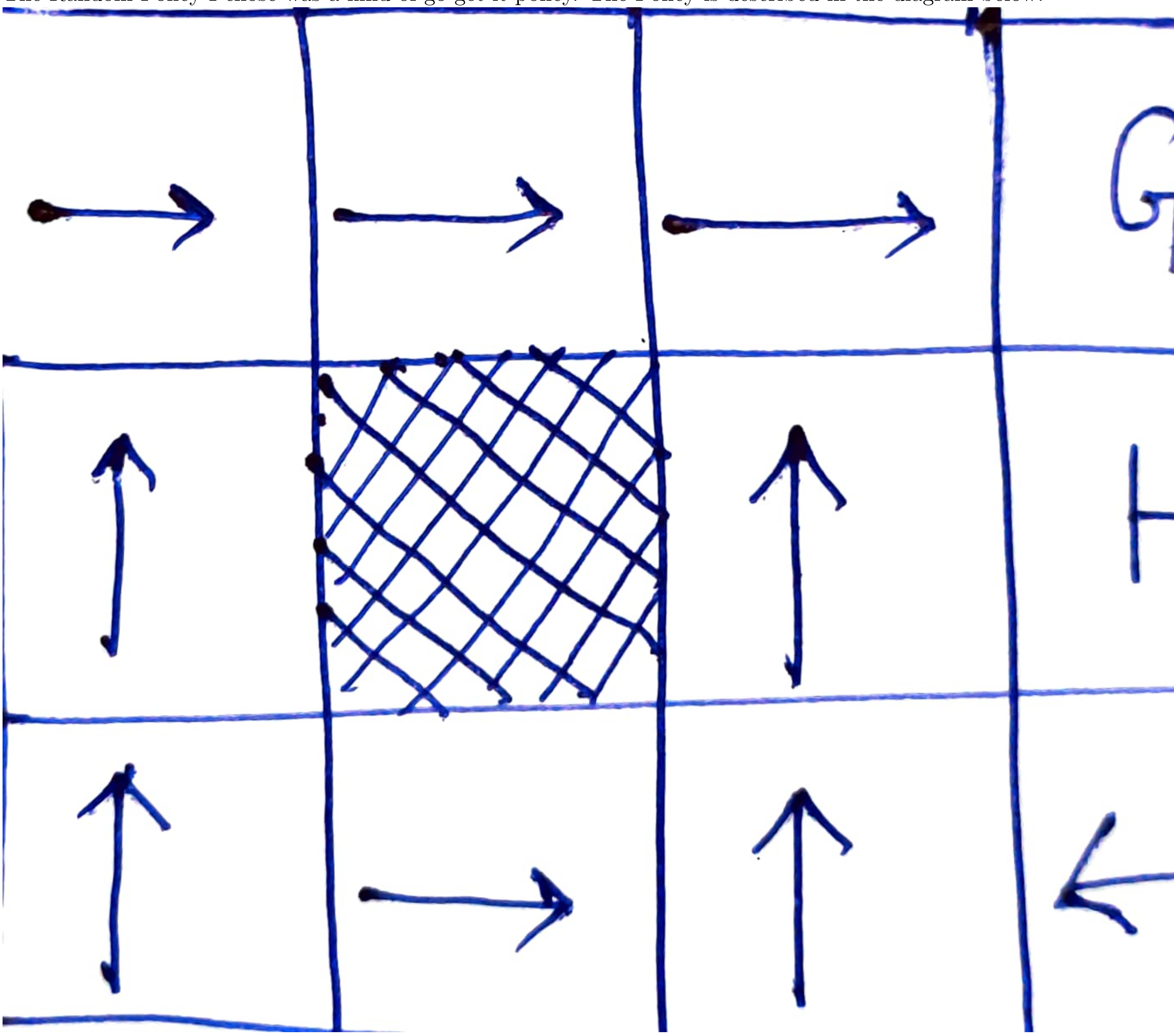
   timeStamp:0 CurrState:8 Action:0 NextState:8 Reward:-0.04 Score:-0.04
   timeStamp:1 CurrState:8 Action:3 NextState:8 Reward:-0.04 Score:-0.08
   timeStamp:2 CurrState:8 Action:3 NextState:8 Reward:-0.04 Score:-0.12
   timeStamp:3 CurrState:8 Action:0 NextState:8 Reward:-0.04 Score:-0.16
   timeStamp:4 CurrState:8 Action:1 NextState:4 Reward:-0.04 Score:-0.2
   timeStamp:5 CurrState:4 Action:3 NextState:8 Reward:-0.04 Score:-0.24000000000000002
   timeStamp:6 CurrState:8 Action:3 NextState:8 Reward:-0.04 Score:-0.28
   timeStamp:7 CurrState:8 Action:2 NextState:9 Reward:-0.04 Score:-0.32
   timeStamp:8 CurrState:9 Action:1 NextState:9 Reward:-0.04 Score:-0.36
   timeStamp:9 CurrState:9 Action:0 NextState:8 Reward:-0.04 Score:-0.39999999999999997
   timeStamp:10 CurrState:8 Action:3 NextState:8 Reward:-0.04 Score:-0.43999999999999995
   timeStamp:11 CurrState:8 Action:2 NextState:9 Reward:-0.04 Score:-0.4799999999999999
   timeStamp:12 CurrState:9 Action:0 NextState:8 Reward:-0.04 Score:-0.5199999999999999
   timeStamp:13 CurrState:8 Action:1 NextState:4 Reward:-0.04 Score:-0.5599999999999999
   timeStamp:14 CurrState:4 Action:2 NextState:4 Reward:-0.04 Score:-0.6
   timeStamp:15 CurrState:4 Action:0 NextState:4 Reward:-0.04 Score:-0.64
   timeStamp:16 CurrState:4 Action:3 NextState:8 Reward:-0.04 Score:-0.68
   timeStamp:17 CurrState:8 Action:1 NextState:4 Reward:-0.04 Score:-0.7200000000000001
   timeStamp:18 CurrState:4 Action:2 NextState:4 Reward:-0.04 Score:-0.7600000000000001

timeStamp:19 CurrState:4 Action:3 NextState:8 Reward:-0.04 Score:-0.8000000000000002
timeStamp:20 CurrState:8 Action:3 NextState:8 Reward:-0.04 Score:-0.8400000000000002
timeStamp:21 CurrState:8 Action:2 NextState:9 Reward:-0.04 Score:-0.8800000000000002
timeStamp:22 CurrState:9 Action:2 NextState:10 Reward:-0.04 Score:-0.9200000000000003
timeStamp:23 CurrState:10 Action:3 NextState:10 Reward:-0.04 Score:-0.9600000000000003
timeStamp:24 CurrState:10 Action:1 NextState:6 Reward:-0.04 Score:-1.0000000000000002
timeStamp:25 CurrState:6 Action:2 NextState:7 Reward:-1 Score:-2.0

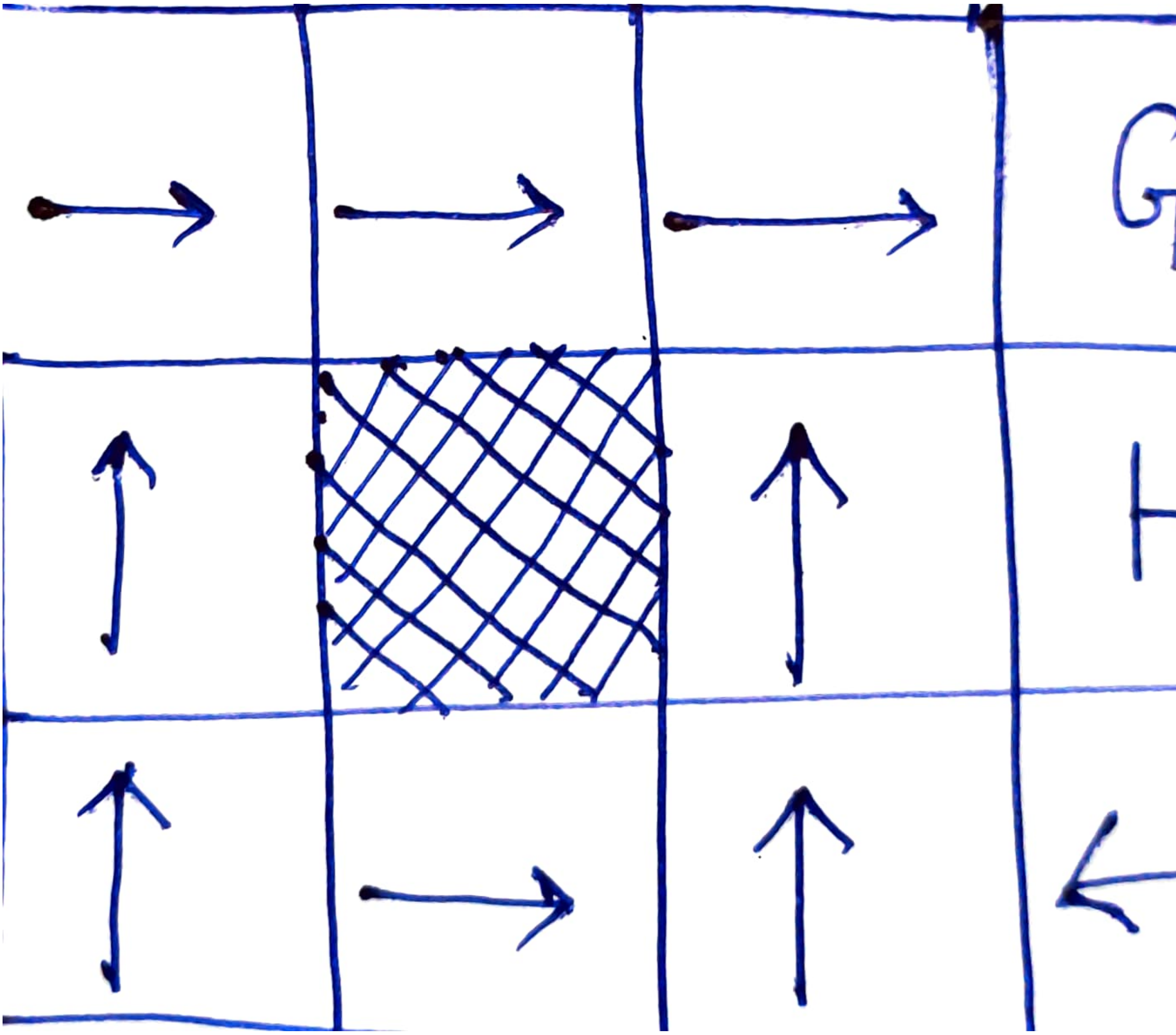
So, my environment implementation is correct.

Solution to Problem 2: RME Optimal Policy via Dynamic Programming

1. The Random Policy I chose was a kind of go get it policy. The Policy is described in the diagram below.

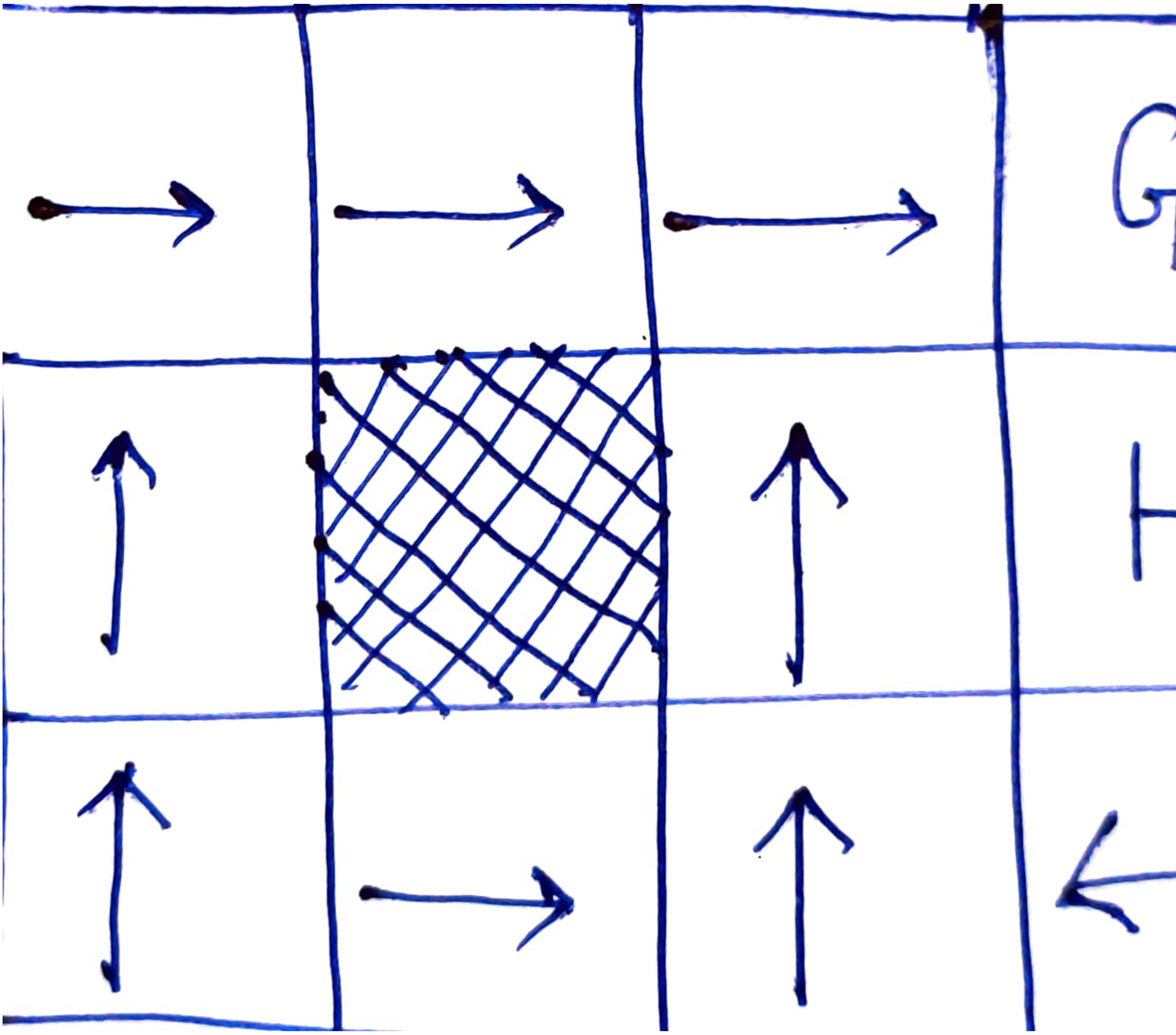


The optimal policy which I got from Policy Iteration was.
2,2,2,0,1,0,1,0,1,2,1,0 which translates to the given policy

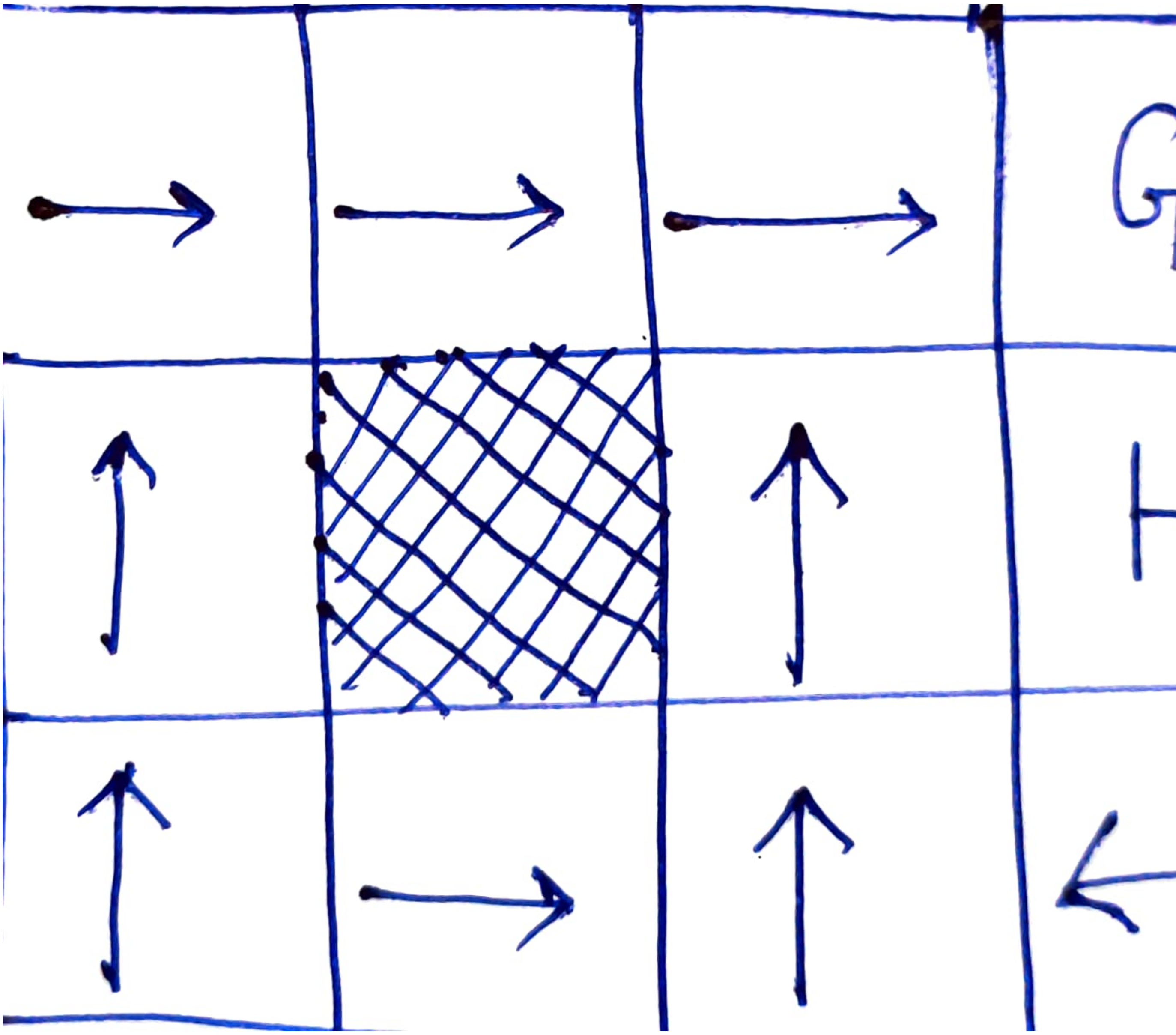


My policy Iteration converged in only 1 iteration

2. The Random Policy I chose was a kind of go get it policy. The Policy is described in the diagram below.



The optimal policy which I got from Value Iteration was.
2,2,2,0,1,0,1,0,1,2,1,0 which translates to the given policy



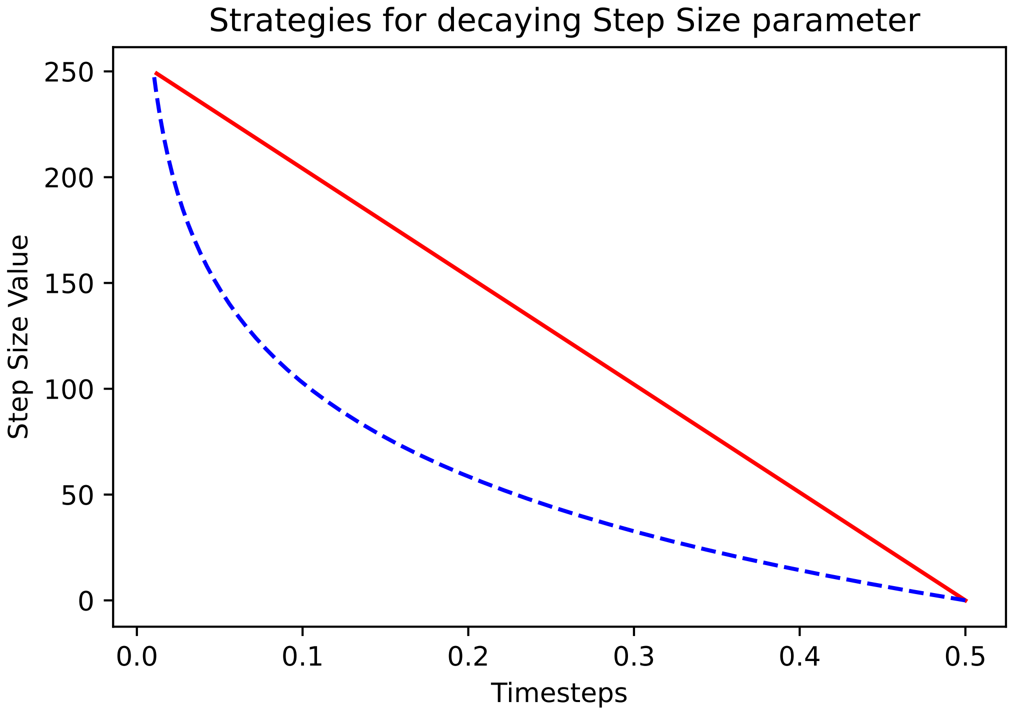
My value iteration took 741 iterations to converge

3. Policy Iteration converged faster than value iteration

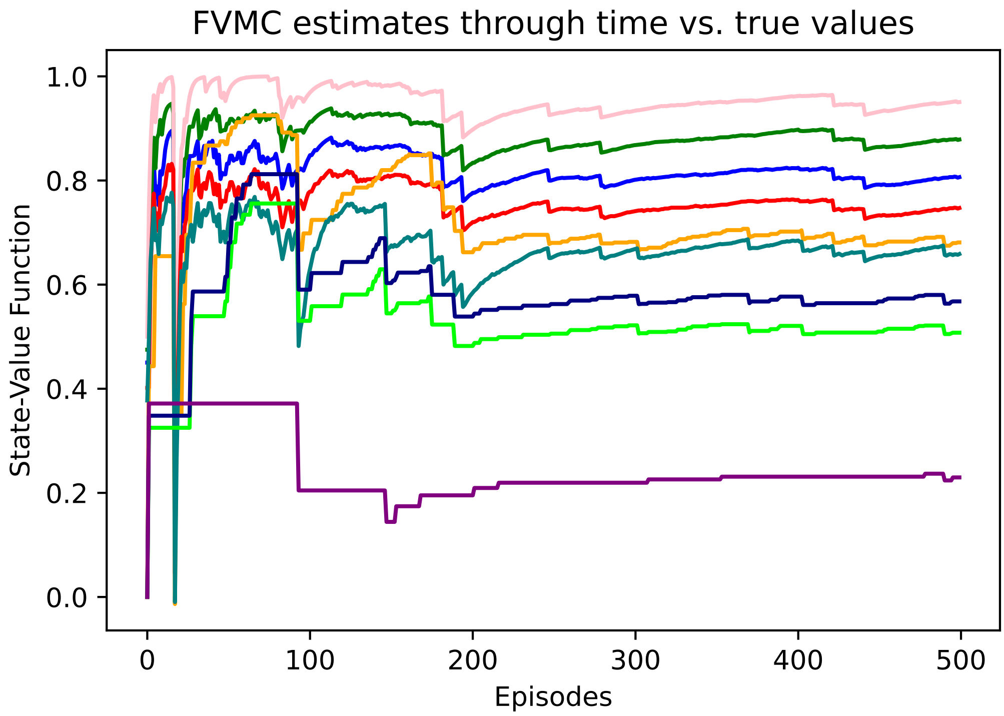
Solution to Problem 3: RME Prediction with MDP Unknown

1. My trajectory for the above chosen go get it policy is as follows
   CurrState:8 Action:1 NextState:4 Reward:-0.04
   CurrState:4 Action:1 NextState:4 Reward:-0.04
   CurrState:4 Action:1 NextState:0 Reward:-0.04
   CurrState:0 Action:2 NextState:1 Reward:-0.04
   CurrState:1 Action:2 NextState:2 Reward:-0.04
   CurrState:2 Action:2 NextState:3 Reward:+1.00

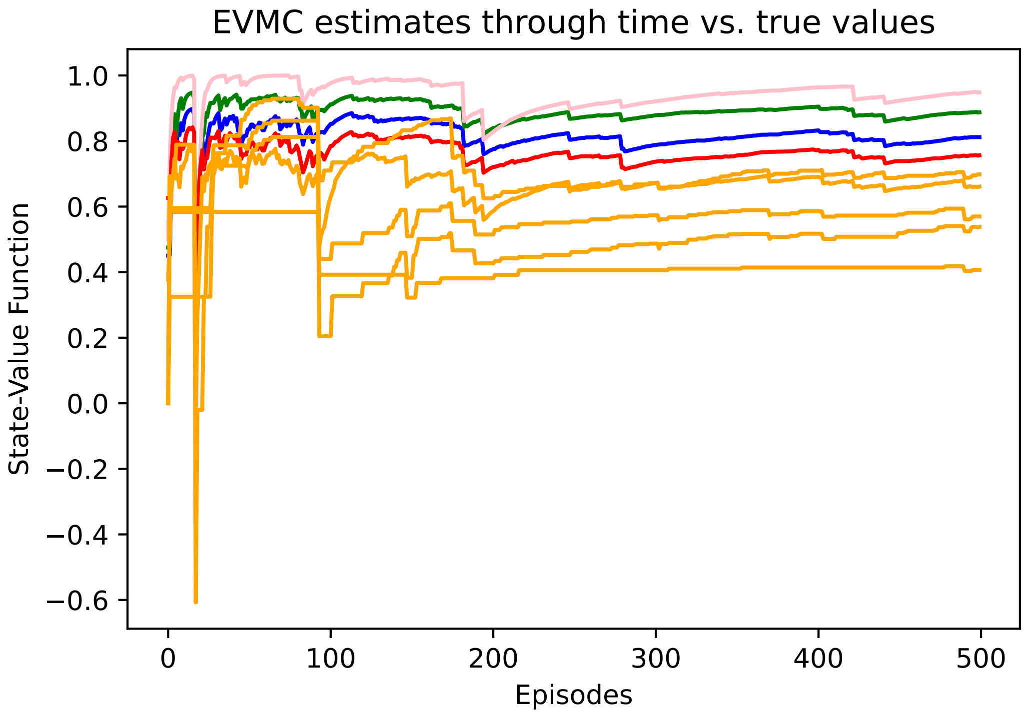
2. Step size parameter decay



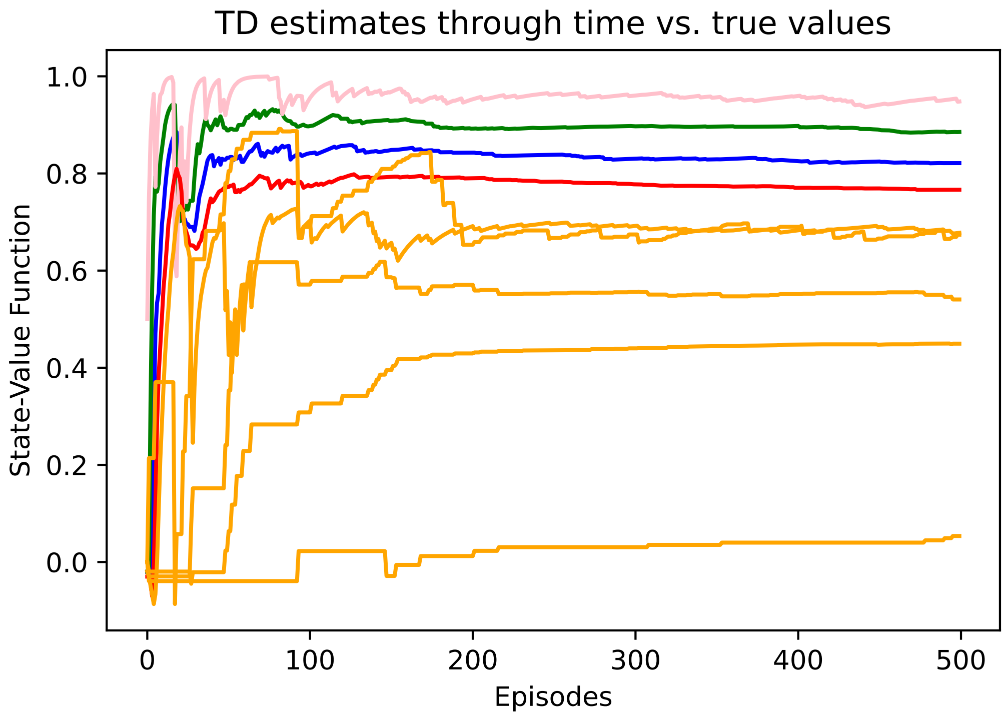
3. MC-FVMC estimate



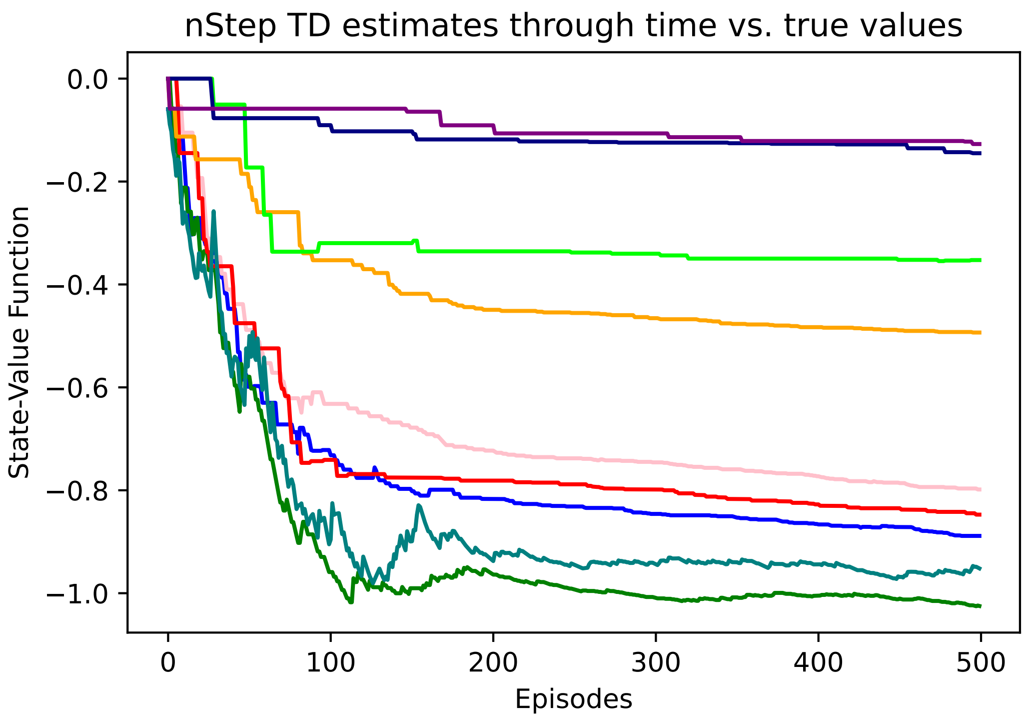
4. MC-EVMC estimate



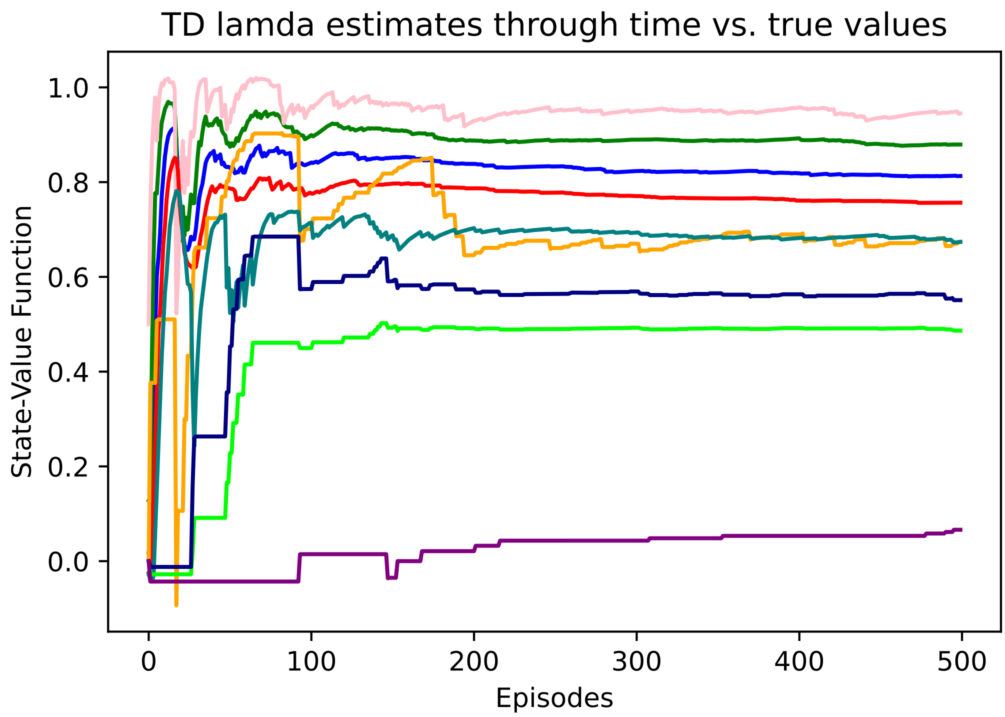
5. TD estimate



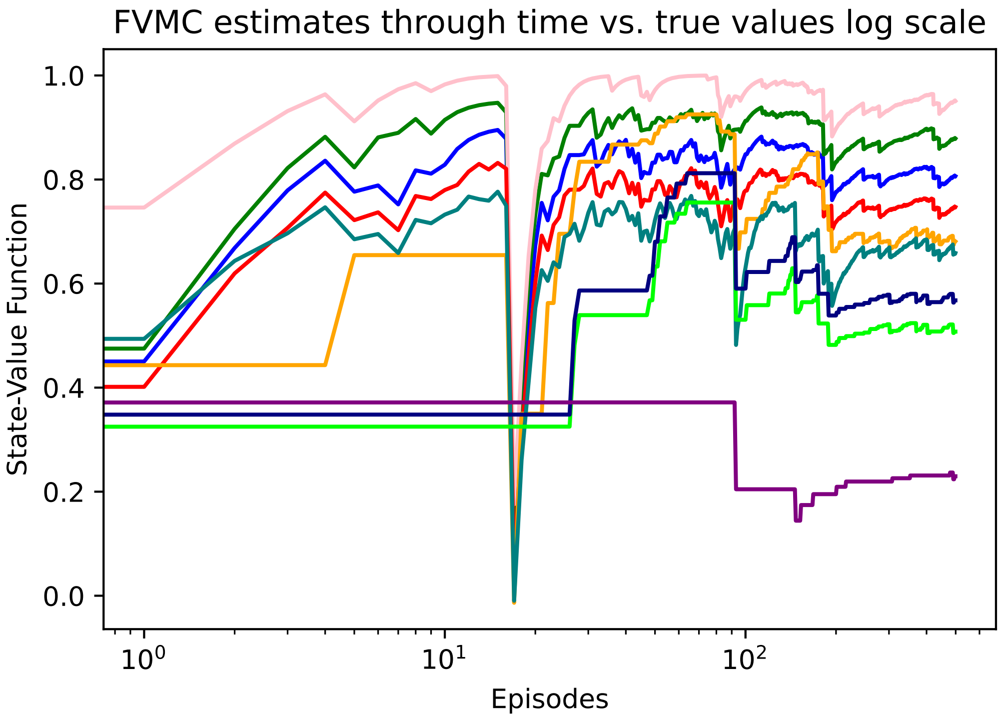
6. nStep TD estimate



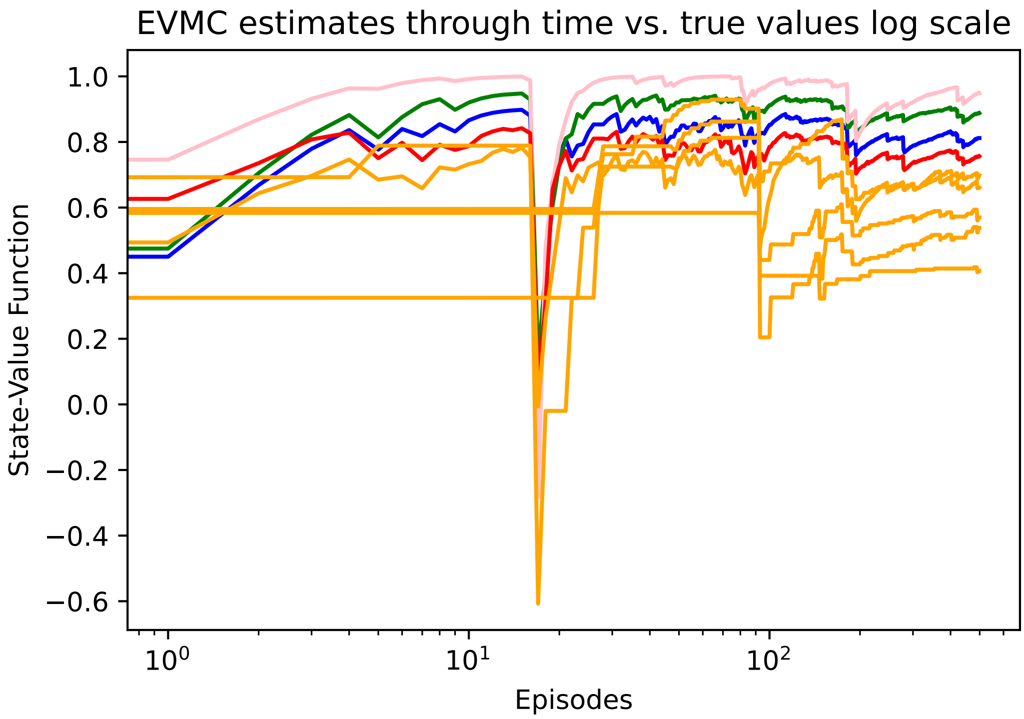
7. TD(λ) estimate



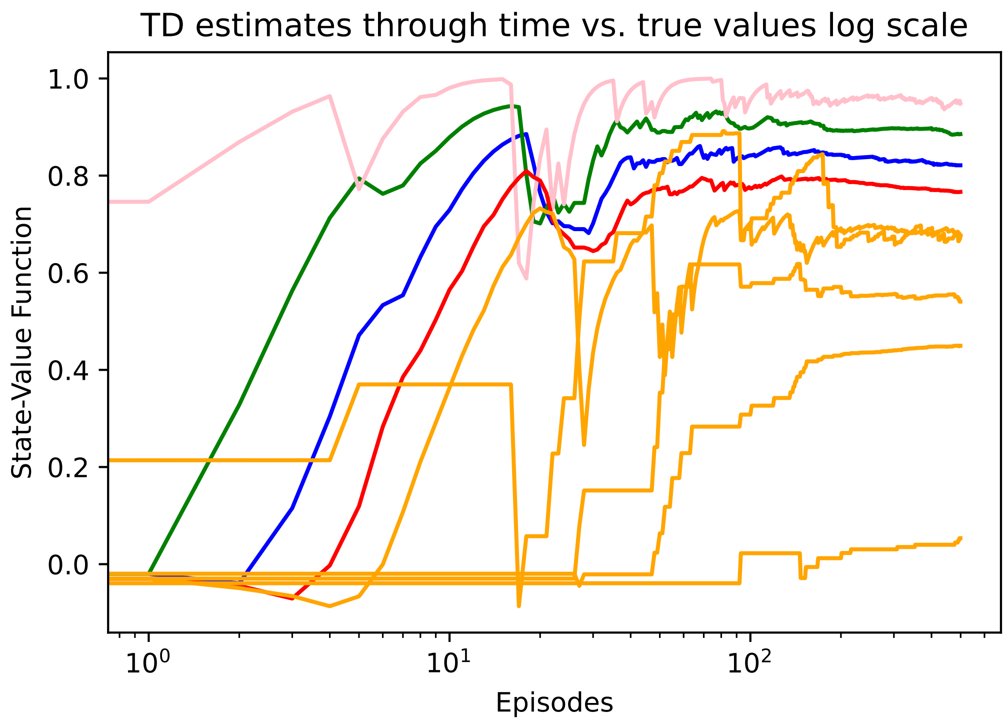
8. MC-FVMC estimate



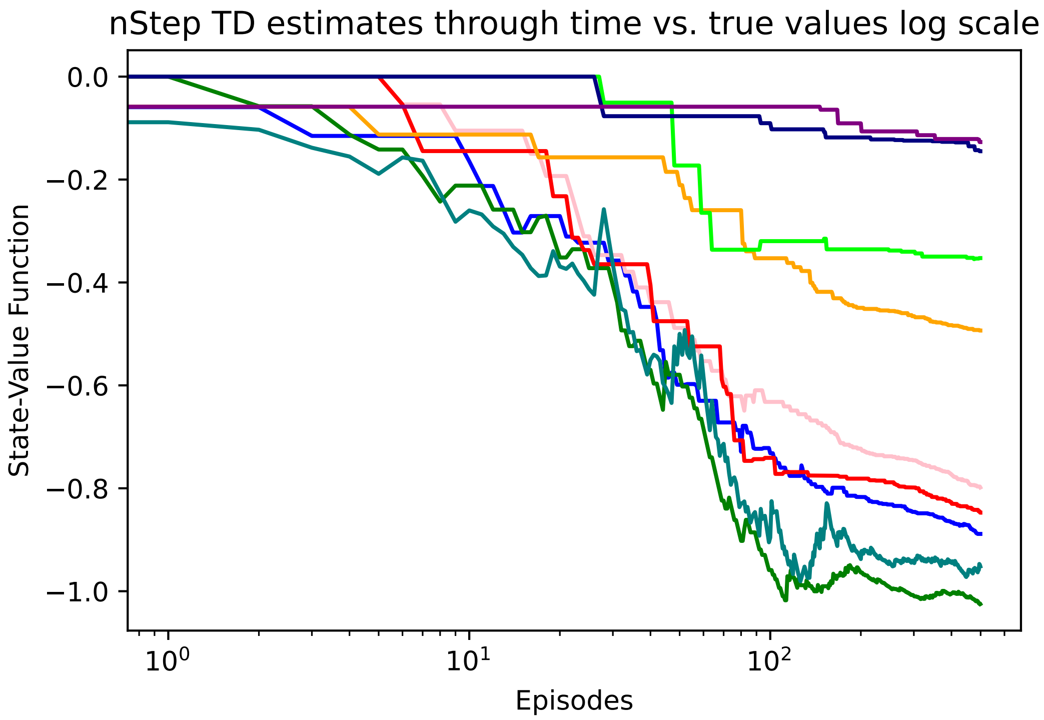
9. MC-EVMC estimate



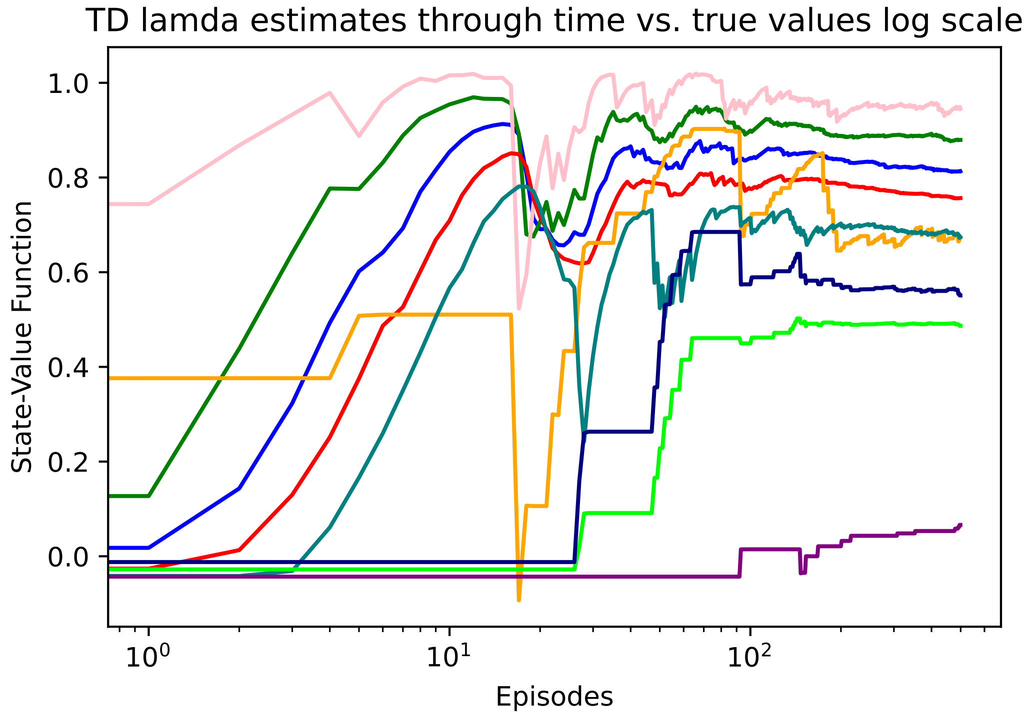
10. TD estimate



11. n Step TD estimate

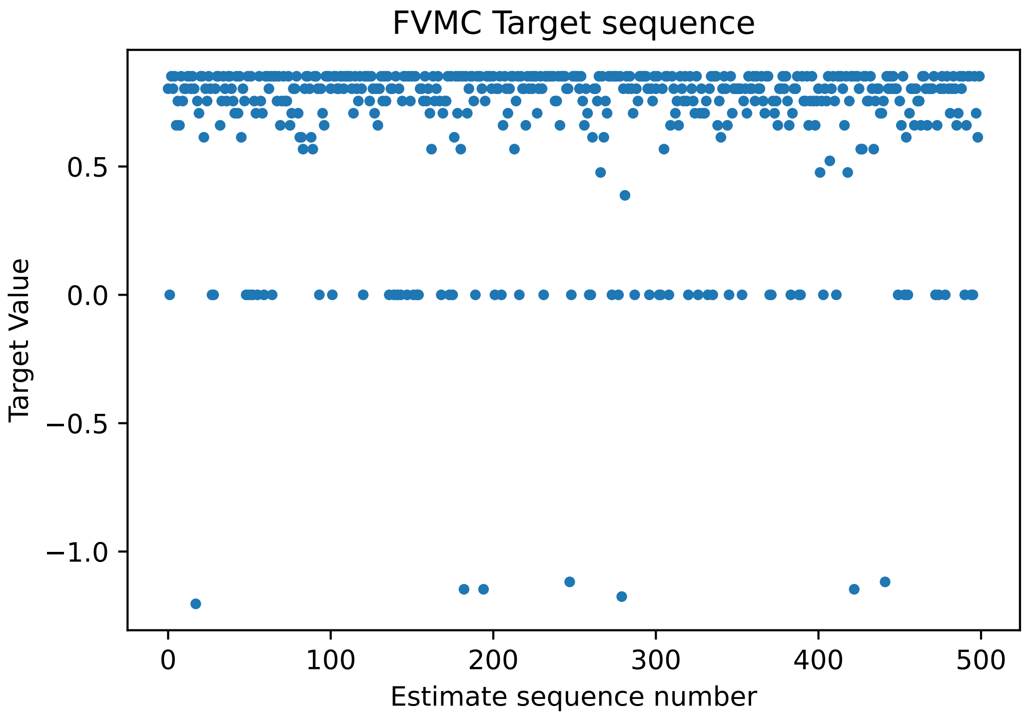


12. TD(λ) estimate

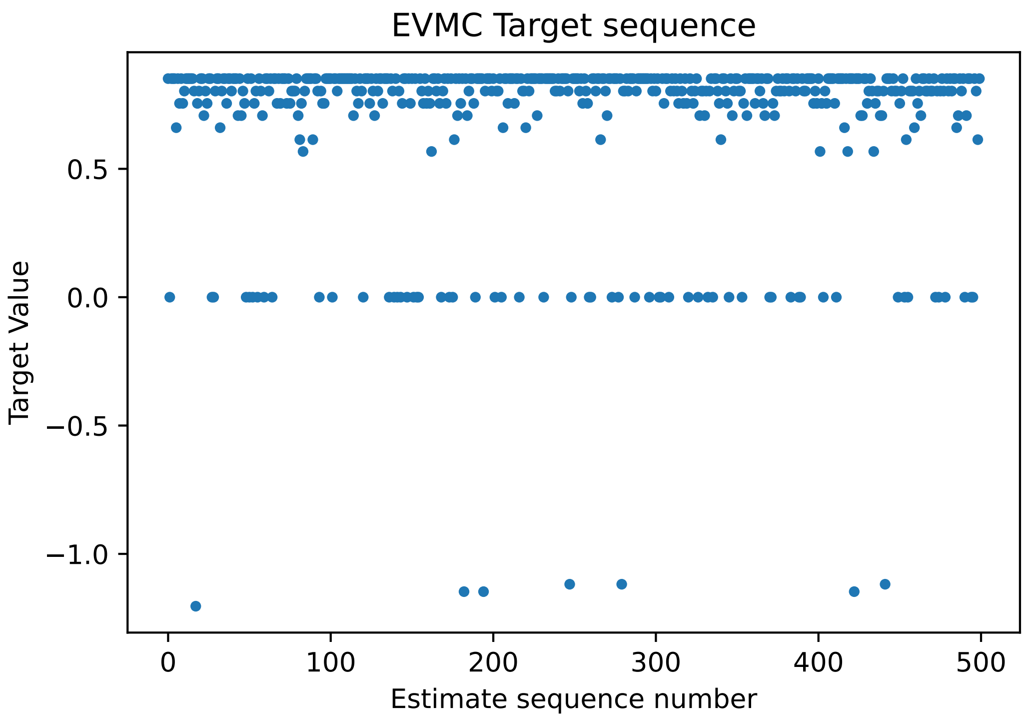


13. We can see that TD fares better than EVMC and FVMC. Also convergence in TD(λ) is much more faster than the rest because it takes the best from both worlds. In the plot of target estimate we observe quite differences from the previous plots in assignment as we have a living reward for each transition of -0.04 so the reward at each step is not binary (0 or 1). So this deviates from the original values due to this.

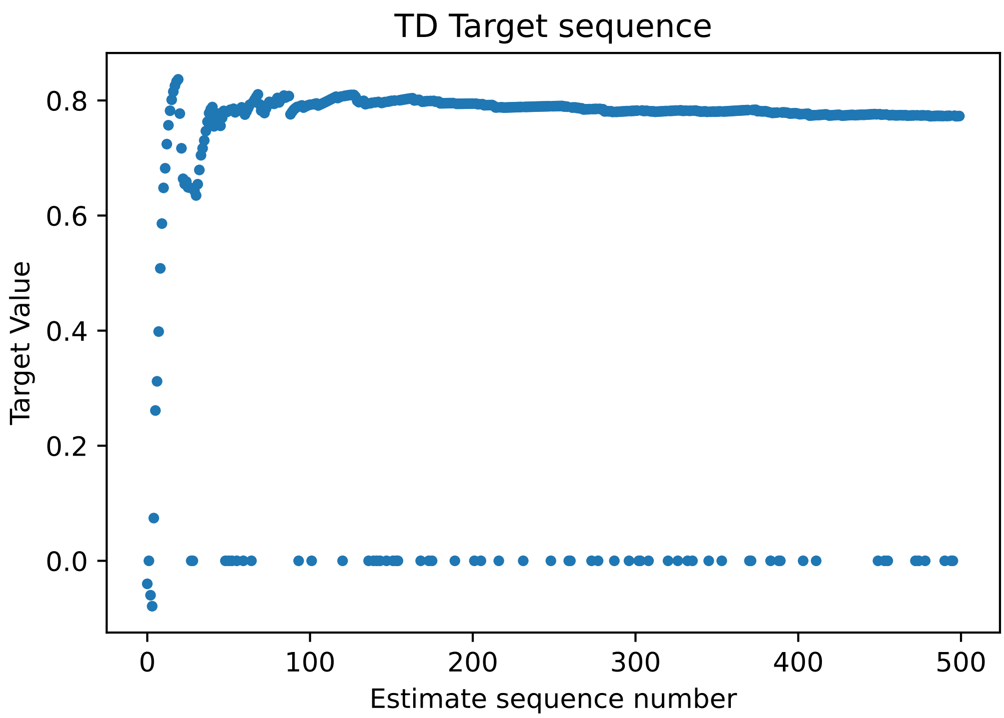
14. Gt FVMC



15. Gt EVMC



16. Gt TD



Random Seed used everywhere is 10