Assignment 1

3) The oscillations in early state are caused because the algorithm thinks all the bad options are also good, and try them out several times before realizing that they are actually bad. At this point the algorithm would be trying to reduce the q estimate values for bad options, at the same time trying to slowly increase q estimate values for good options. Suppose consider the true arm values of 10 arms are -1, 1, 2, 0, 0.5, 0.3, 0, 0.7, 0.8, 0.1, -0.5, we take optimistic estimate of +5 and choose greedy method. At first all the arms become more likely to be selected because the first estimate of even the best arm is 5+2/2 = 3.5. But once the 10 arms are picked and averaging of estimates is completed, the average of best arm becomes more than the other arms. So the best estimate is likely to be picked up on 11^{th} step. Then again the estimate of best arm reduces no, 2+2+5/3 = 3. So the other arms are more likely to be picked. So at step 11^{th} or somewhere close to you there is a spike and it is followed by some oscillations for a while before the estimates get completely averaged out.

In the initial stages the performance is poor; because the bad options had to be selected out many times before they get averaged out. How poor it is, depends on how fast the averaging out happens.

4) Self driving car: The states could be the direction in which it is heading, velocity, acceleration, fuel, an image captured with camera on top of car (this could give information about traffic signal inputs, whether there are other vehicles in front, distance between left and right side of car, weather there is a turn etc). The actions are to turn the driving wheel to turn left or right, change velocity or acceleration through gears, stop the car in side walk. The rewards could be zero if car is going on smoothly, a -0.5 if it's giving jerks, -1 if it collides, -0.25 if fuel is about to run out.

Chess AI: The state is position of pieces on board. The actions are all the moves available for each piece on board. The reward of +1 for win and -1 for lose.

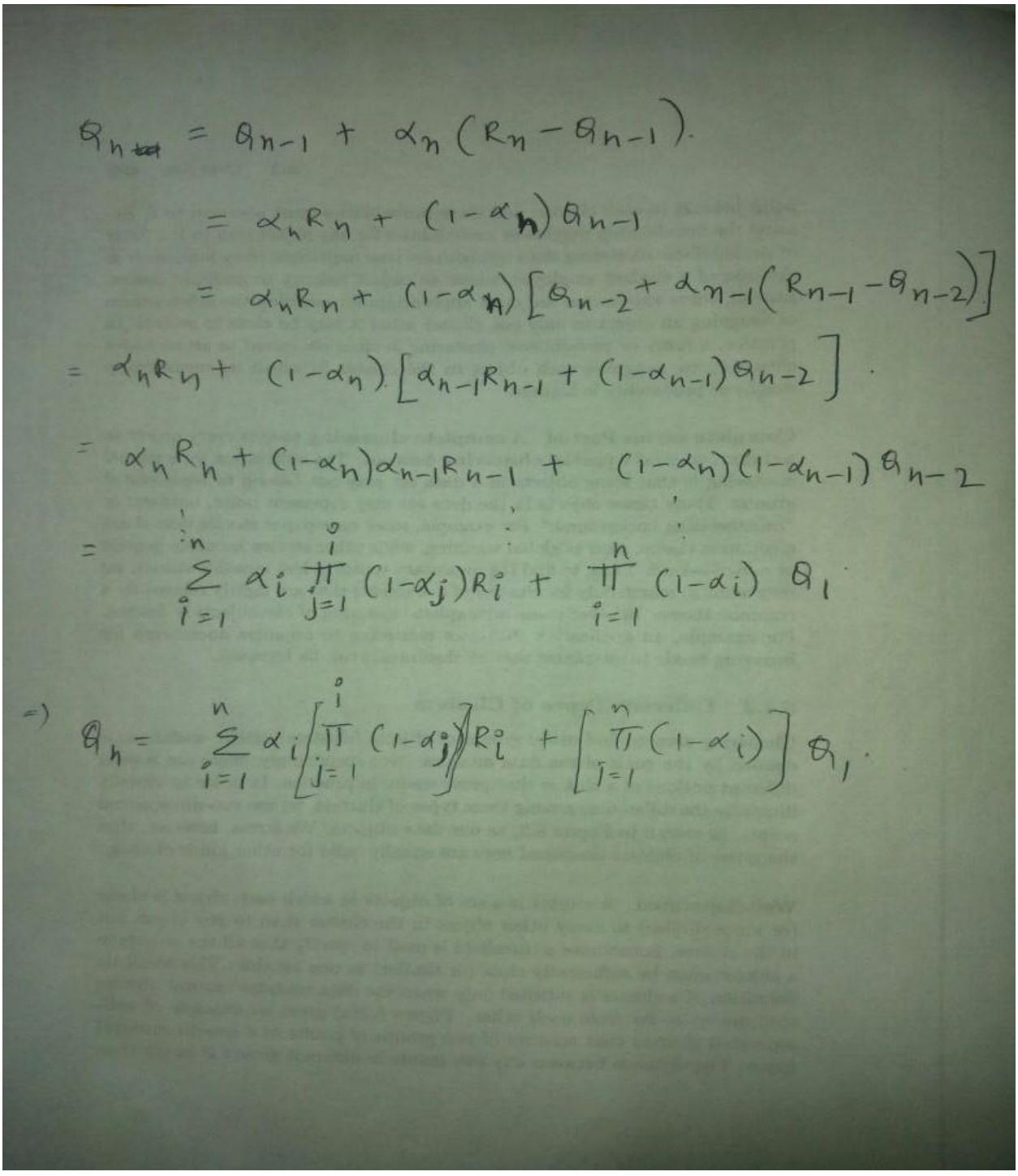
Atari game playing agent: The state is pixel input from screen. The actions are joystick keys. The reward is tied to increasing game score.

5) We have given reward +1 for escaping the maze and zero at all other times. The agent can explore the maze until the episode time ends and accumulate zero reward since there is no loss for just randomly exploring maze. Essentially we have not set any time limits for the robot to exit the mage. This is why the robot was not able to learn over time. To make the robot learn to solve the maze we need to introduce a -1 penalty for staying in the maze for each time step. This would allow the robot to seek out exit faster.

6) At present the photograph captured by the camera won't contain entire information needed to predict next image, the camera could capture. The scene before the camera is dynamic and anything could happen by the time the camera could capture the next picture. Furthermore the image will have no information about moving things, such as their velocity, acceleration etc. So we could say we don't have access to Markov state of image unless until everything before camera is perfectly stationary. If the

camera is broken for a day, we won't be able to predict what the next image is as the last known image is not Markov state. We won't have access to Markov state now either.

1)



7)

$$Q_\pi(s,a) = E_\pi\{R_t \mid s_t = s, a_t = a\}$$

$$= E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}$$

$$= E_\pi\left\{\sum_{s'}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a, s_{t+1} = s'\right\} \times P(s_{t+1} = s' \mid s_t = s, a_t = a)\right\}$$

$$= E_\pi\left\{\sum_{s'}\left\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\right\} P(s_{t+1} = s' \mid s_t = s, a_t = a)\right\}$$

$$+ E_\pi\left\{\gamma \sum_{s'}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s, a_t = a, s_{t+1} = s'\right\} P(s_{t+1} = s' \mid s_t = s, a_t = a)\right\}$$

$$= \sum_{s'} R(s_{t+1} = s' \mid s_t = s, a_t = a)\, P(s_{t+1} = s' \mid s_t = s, a_t = a) +$$

$$\gamma \sum_{s'} E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_{t+1} = s'\right\} P(s_{t+1} = s' \mid s_t = s, a_t = a)$$

$$E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_{t+1} = s'\right\} =$$

$$= E_\pi\{R_{t+1} \mid s_{t+1} = s'\}$$

$$= \sum_{a'} E_\pi\{R_{t+1} \mid s_{t+1} = s', a_{t+1} = a'\} P(a_{t+1} = a' \mid s_{t+1} = s')$$

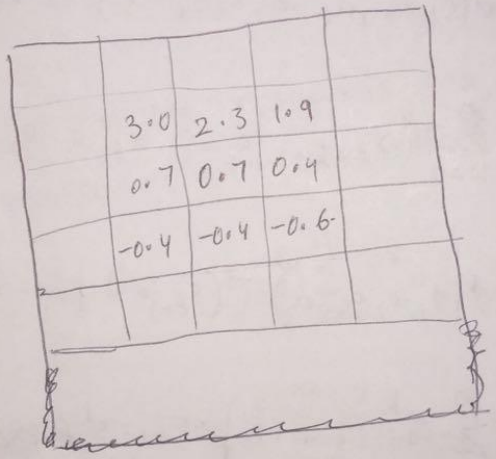
$$= \sum_{a'} E_\pi\{R_{t+1} \mid s_{t+1} = s', a_{t+1} = a'\}\, \pi(s', a')$$

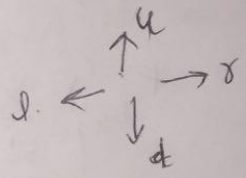
$$= \sum_{a'} q_\pi(s', a')\,\pi(s', a')$$

$$q^\pi(s,a) = \sum_{s'} R(s_{t+1}=s' \mid s_t=s, a_t=a)\; P(s_{t+1}=s' \mid s_t=s, a_t=a)$$

$$+ \gamma \sum_{s'} \sum_{a'} q_\pi(s', a')\,\pi(s', a')\, P(s_{t+1}=s' \mid s_t=s, a_t=a)$$

8)





$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s', r} P(s', r \mid s, a)\left[r + \gamma V_\pi(s')\right].$$

$$= \pi(a=u \mid s) \sum_{s'} P(s', r \mid s, a=u)\left[0 + \gamma V_\pi(s')\right] +$$

$$\pi(a=r \mid s) \sum_{s'} P(s', r \mid s, a=r)\left[0 + \gamma V_\pi(s')\right] +$$

$$\pi(a=d \mid s) \sum_{s'} P(s', r \mid s, a=r)\left[0 + \gamma V_\pi(s')\right] +$$

$$\pi(a=l \mid s) \sum_{s'} P(s', r \mid s, a=d)\left[0 + \gamma V_\pi(s')\right].$$

$$= \frac{1}{4}\gamma V^{\pi}(s'=u) + \frac{1}{4}\gamma V^{\pi}(s'=r) + \frac{1}{4}\gamma V^{\pi}(s'=d) + \frac{1}{4}\gamma V_{\pi}(s'=l).$$

@ $\gamma = 0.9$.

$$\Rightarrow \frac{0.9}{4}(2.3 + 0.4 - 0.4 + 0.7) \approx 0.7$$

9) The signs of the rewards are important. while the intervels are not important.

$$V_{\pi}(s) = E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} + C \mid s_t = s\right].$$

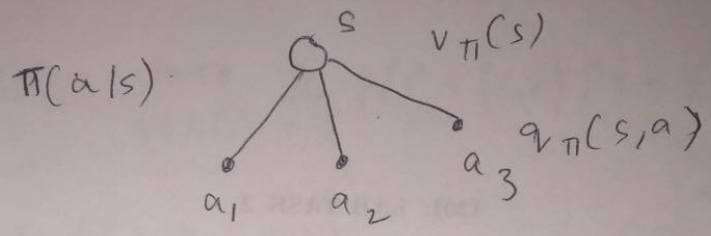
$$= E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} \mid s_t = s\right] + E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} C \mid s_t = s\right]$$

$$= V^{\pi}(s) + C \times \sum_{k=0}^{\infty}\gamma^{k}$$

$$= V_{\pi}(s) + \frac{C}{1-\gamma}.$$

Thus As, we can see adding constant to all values of states doesn't affect relative values of any states under any policies.

10)



$$V_\pi(s) = E_\pi[R_t | S_t = s]$$

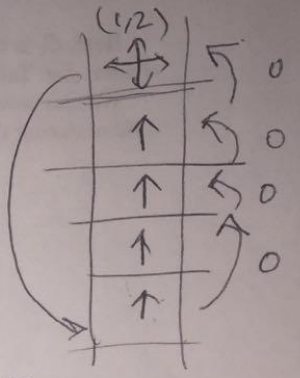
$$= \sum_{a=\{a_1,a_2,a_3\}} E_\pi[R_t | S_t = s, a_t = a] \pi(a_t = a | s_t = s)$$

$$= \sum_a E_\pi[R_t | S_t = s, a_t = a] \pi(a|s)$$

$$V_\pi(s) = \sum_a q_\pi(s,a) \pi(a|s)$$

11)

$$V_*(s) = \max_{a \in A(s)} \sum_{s',r} p(s',r|s,a)\left[r + \gamma V_*(s')\right]$$



at A(1,2) +10.

B(1,4) +5.

otherwise 0.

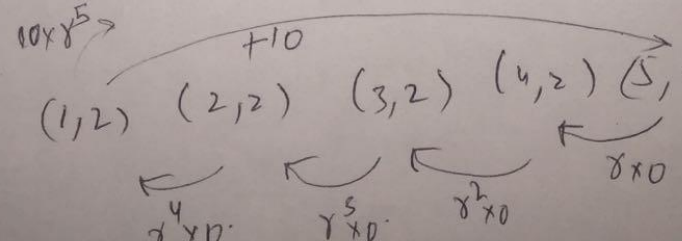
$$V_*(s) = \max_{a \in A(s)} E_{\pi^*}\left[R_t | s_t = s, a_t = a\right].$$

$$= \max_{a \in A(s)} E_{\pi^*}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a\right]$$

$$= 10 + \gamma \times 0 + \gamma^2 \times 0 + \gamma^3 \times 0 + \gamma^4 \times 0 + \gamma^5 \times 10 \ldots$$

$$= 10\left(1 + \gamma^5 + \gamma^{10} \ldots\right)$$

$$= \frac{10}{1-\gamma^5} = \frac{10}{1-(0.9)^5} = 24.4$$



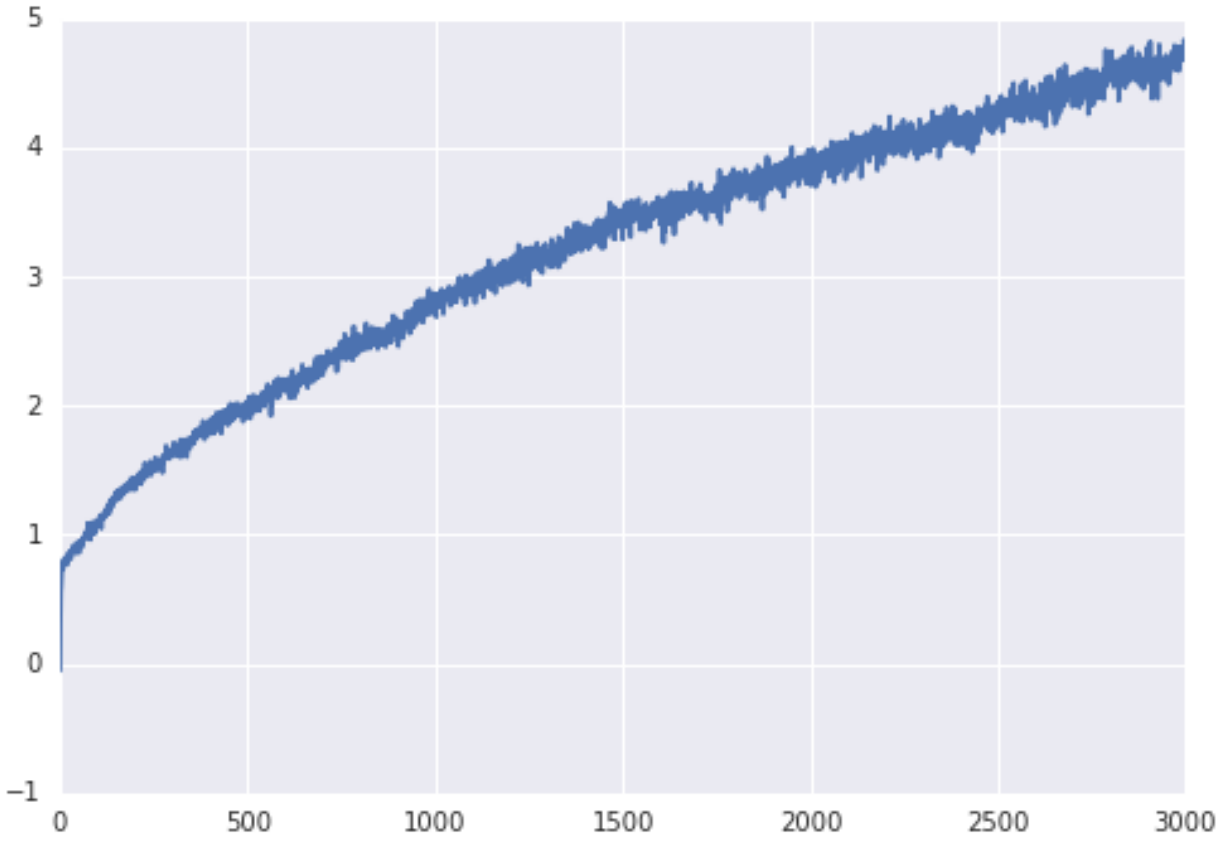
The patten as shown in above figure follows.

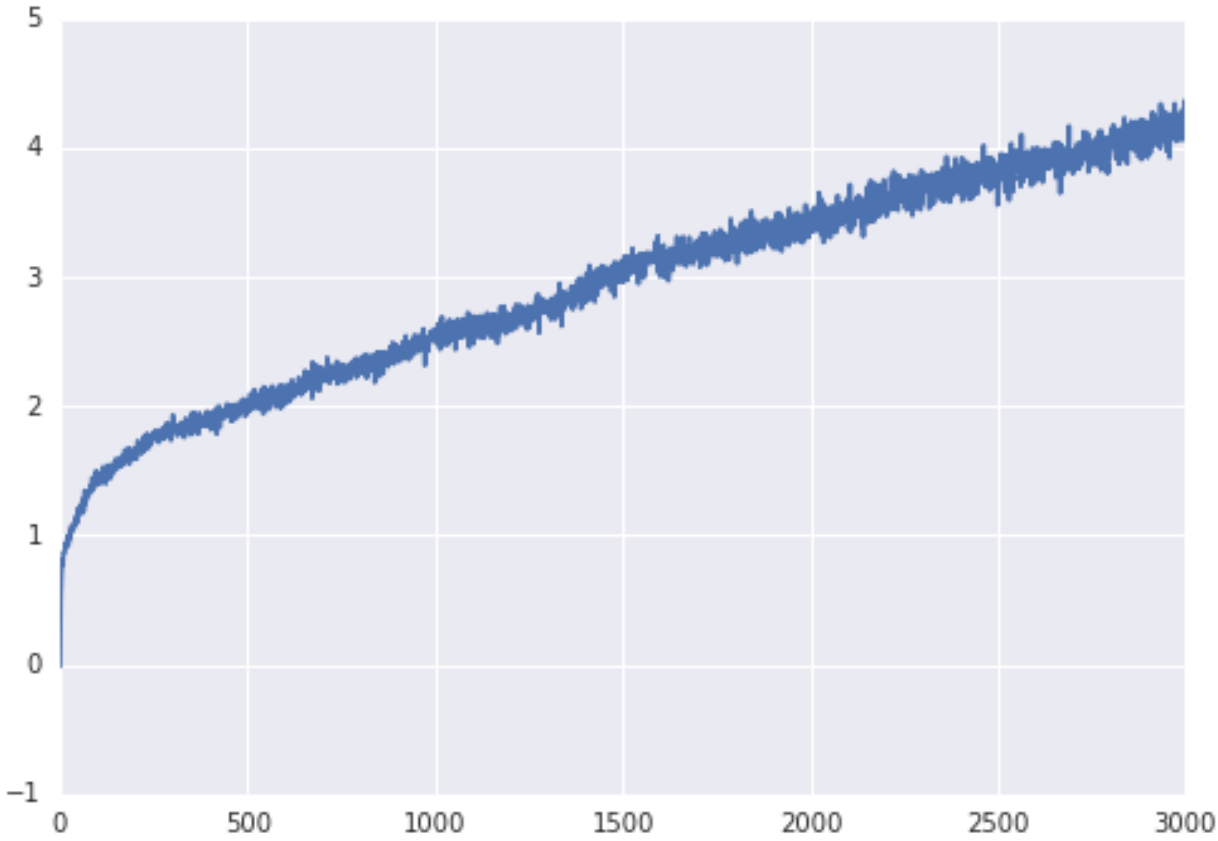
2) Notes about code.

It works using make and then make run on Silo. Initially the code gives the output for constant alpha incremental and by commenting the code in agent_step function in SimpleAgent.cxx sample average method can be seen.

For sample average methods the reward will not perform well as we increase the shift of random walk. It gives less reward compared to the constant alpha method.

For 3000 steps and 500 runs with a shift of 0.5 this plots show the difference clearly. Random walk happens after every 10 steps. First image is constant alpha and second one is sample average.




5
4
3
2
1
0
−1
0
500
1000
1500
2000
2500
3000