1 Requirements

2 Exercises

2.1 Rewrite the psuedo code of Monte Carlo ES as mentioned in Exercise 5.4. You must explain your code and why it is equivalent to the code provided in the book.

2.1.1 Explanation

$G_{s,a,t}$ is the return (sum reward uptil episode termination) recieved after encountering state s and taking action a on it the t^{th} time (we only consider the first encounter in an epsiode, later ecnounters in an episode are ignored) over the course of all the episodes.

$Q(s,a) = \frac{1}{n}.\sum_{k=1}^{n} G_{s,a,k}$
where n is the last time action a is taken on state s in the episodes generaed by Monte Carlo.

$Q(s,a)_t = \frac{1}{t}.\sum_{k=1}^{t} G_{s,a,k}$

$$
\begin{aligned}
Q(s,a)_{t+1} &= \frac{1}{t+1}.\sum_{k=1}^{t+1} G_{s,a,k} \\
&= \frac{1}{t+1}.[G_{s,a,k+1} + \sum_{k=1}^{t} G_{s,a,k}] \\
&= \frac{1}{t+1}.[G_{s,a,k+1} + \frac{1}{t+1}\sum_{k=1}^{t} G_{s,a,k}] \\
&= \frac{1}{t+1}.[G_{s,a,k+1} + t.\frac{1}{t}.\sum_{k=1}^{t} G_{s,a,k}] \\
&= \frac{1}{t+1}.[G_{s,a,k+1} + t.Q(s,a)_t] \\
&= Q(s,a)_t + \frac{1}{t+1}.[G_{s,a,k+1} - Q(s,a)_t]
\end{aligned}
$$

2.1.2 Pseudocode

Initialize:
$\pi(s) \in A(s)$ (arbitrarily), for all $s \in S$
$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in S, a \in A(s)$
$Count(s,a) \leftarrow 2D\ list\ of\ zeros$, for all $s \in S, a \in A(s)$
Loop forever (for each episode):
Choose $S_0 \in S, A_0 \in A(S_0)$ randomly such that all pairs have probability > 0
Generate an episode from S_0, A_0, following $\pi : S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1, \ldots, S_{t-1}, A_{t-1}$:

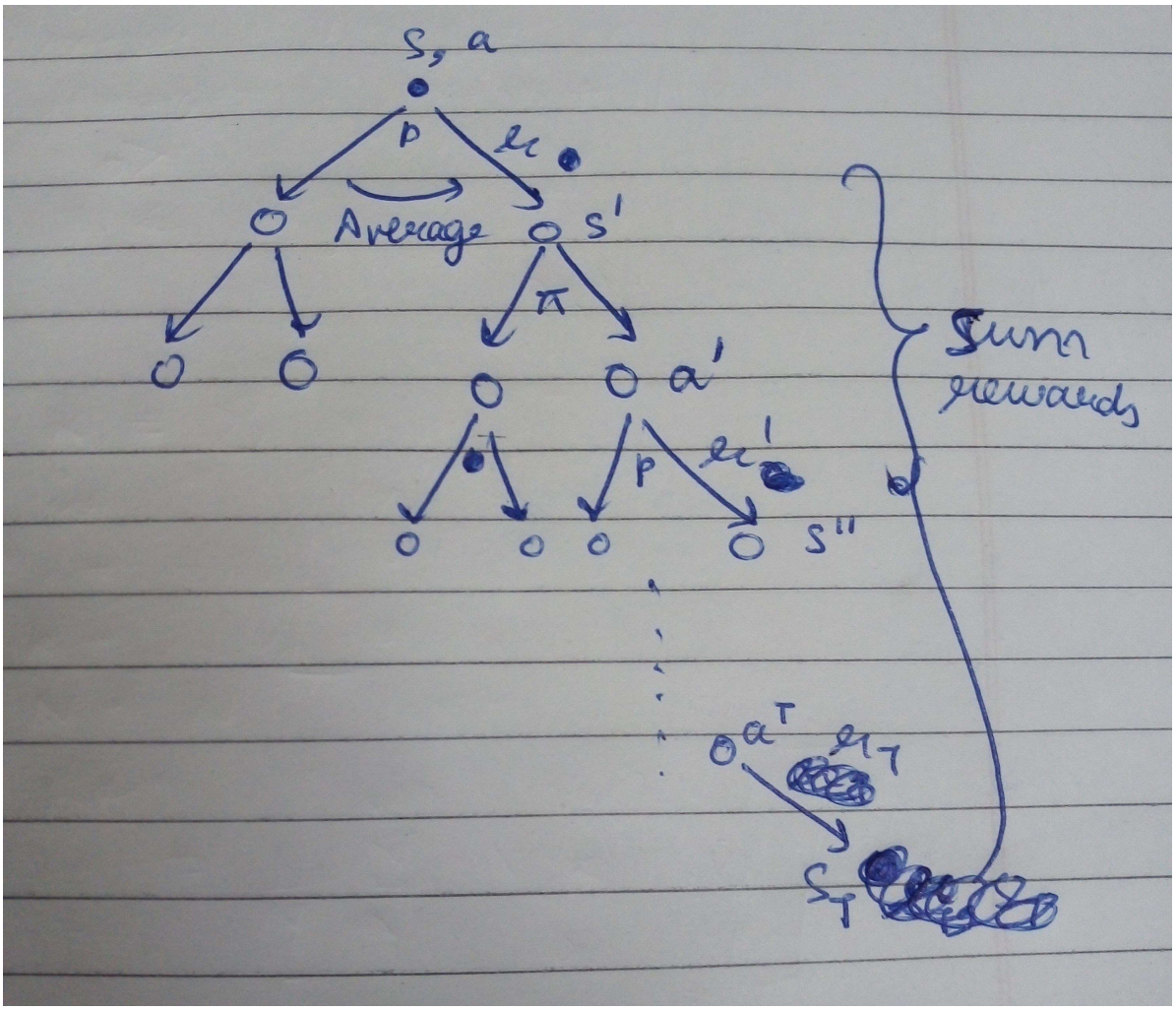
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + [G - Q(S_t, A_t)] \,/\, [Count(S_t, A_t) + 1]$$
$$Count(S_t, A_t) \leftarrow Count(S_t, A_t) + 1$$
$$\pi(S_t) \leftarrow argmax_a Q(S_t, a)$$

2.2 Draw the backup diagram asked for in exercise 5.3.

In [121]:

```
1 from IPython.display import Image
2 Image(filename='backup_diag.jpg')
```

Out[121]:



2.3 Solve exercise 5.6.

$$\rho_{t:T-1} \doteq \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

$$q_b(s,a) = \mathbb{E}_b[G_t|S_t = s, A_t = a]$$

$$q_\pi(s,a) = \mathbb{E}_\pi[\rho_{t:T-1} G_t|S_t = s, A_t = a]$$

$\mathfrak{T}(s,a)$ is the set of all time steps when state s is visited and action a is taken on it for an every-visit method.
$\mathfrak{T}(s,a)$ is the set of all time steps that were first visits to s when action a is taken on it within their episodes, in

case of first-visit Monte Carlo.
$T(t)$ is the time of termination of the episode in which time t occurs.
G_t is the return after t uptil $T(t)$.

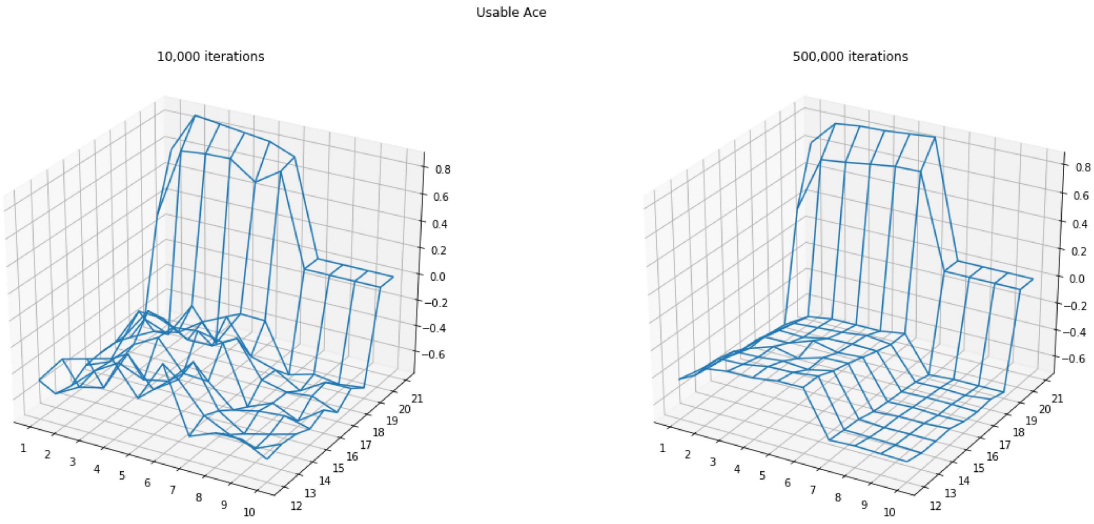
$$Q(s,a) \doteq \frac{\sum_{t \in \mathfrak{T}(s,a)} \rho_{t:T-1} G_t}{|\mathfrak{T}(s,a)|}$$

$$Q(s,a) \doteq \frac{\sum_{t \in \mathfrak{T}(s,a)} \rho_{t:T-1} G_t}{\sum_{t \in \mathfrak{T}(s,a)} \rho_{t:T-1}}$$

2.4 Solve the blackjack game and generate figures 5.1, 5.2, and 5.3. Submit your figures and code. You must explain your code.

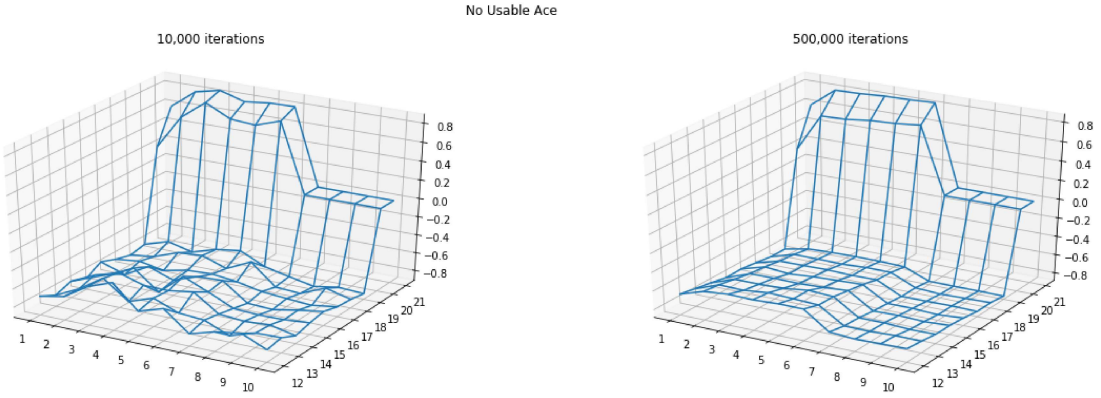
In [7]:

```
xv, yv = np.meshgrid(np.arange(10)+12, np.arange(10)+1, indexing='ij')

fig = plt.figure(figsize=(20, 8))
fig.suptitle('Usable Ace')
ax = fig.add_subplot(121, projection='3d')
ax.plot_wireframe(yv, xv, V_mid[:-1, :-1, 0])
plt.xticks(np.arange(10)+d_map)
plt.yticks(np.arange(10)+ps_map)
# ax.set_zticks([np.amin(V_mid[:, :, 0]), 0, np.amax(V_mid[:, :, 0])])
ax.title.set_text('10,000 iterations')

ax = fig.add_subplot(122, projection='3d')
ax.plot_wireframe(yv, xv, V[:-1, :-1, 0])
plt.xticks(np.arange(10)+d_map)
plt.yticks(np.arange(10)+ps_map)
# ax.set_zticks([np.amin(V[:, :, 0]), 0, np.amax(V[:, :, 0])])
ax.title.set_text('500,000 iterations')
```



In [10]:

```
fig = plt.figure(figsize=(20, 6))
fig.suptitle('No Usable Ace')
ax = fig.add_subplot(121, projection='3d')
ax.plot_wireframe(yv, xv, V_mid[:-1, :-1, 1])
plt.xticks(np.arange(10)+d_map)
plt.yticks(np.arange(10)+ps_map)
# ax.set_zticks([np.amin(V_mid[:, :, 1]), 0, np.amax(V_mid[:, :, 1])])
ax.title.set_text('10,000 iterations')

ax = fig.add_subplot(122, projection='3d')
ax.plot_wireframe(yv, xv, V[:-1, :-1, 1])
plt.xticks(np.arange(10)+d_map)
plt.yticks(np.arange(10)+ps_map)
# ax.set_zticks([np.amin(V[:, :, 1]), 0, np.amax(V[:, :, 1])])
ax.title.set_text('500,000 iterations')
```

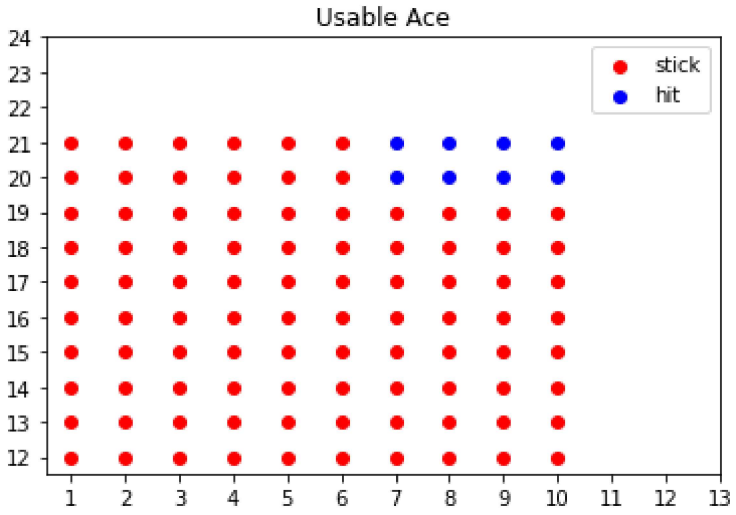


2.4.2 Policy Control

In [15]:

```
from IPython.display import Image
Image(filename='usable_ace_pi.png')
```

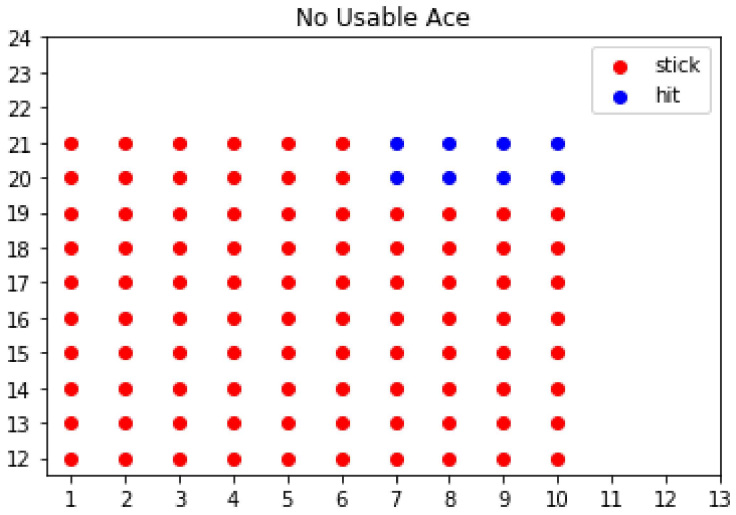
Out[15]:



In [17]:

```
from IPython.display import Image
Image(filename='unusable_ace_pi.png')
```

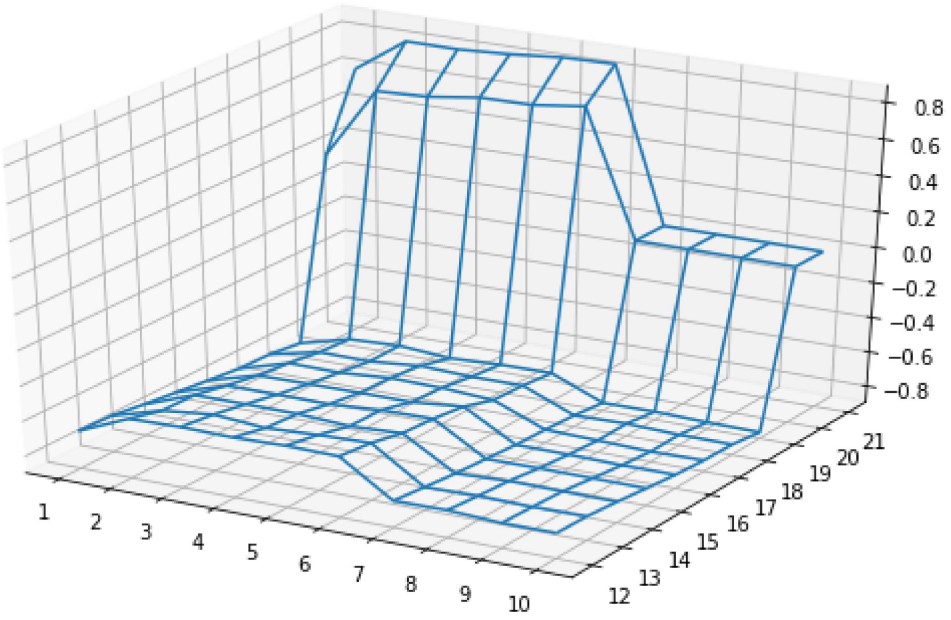
Out[17]:



In [18]:

```
from IPython.display import Image
Image(filename='unusable_ace_q.png')
```

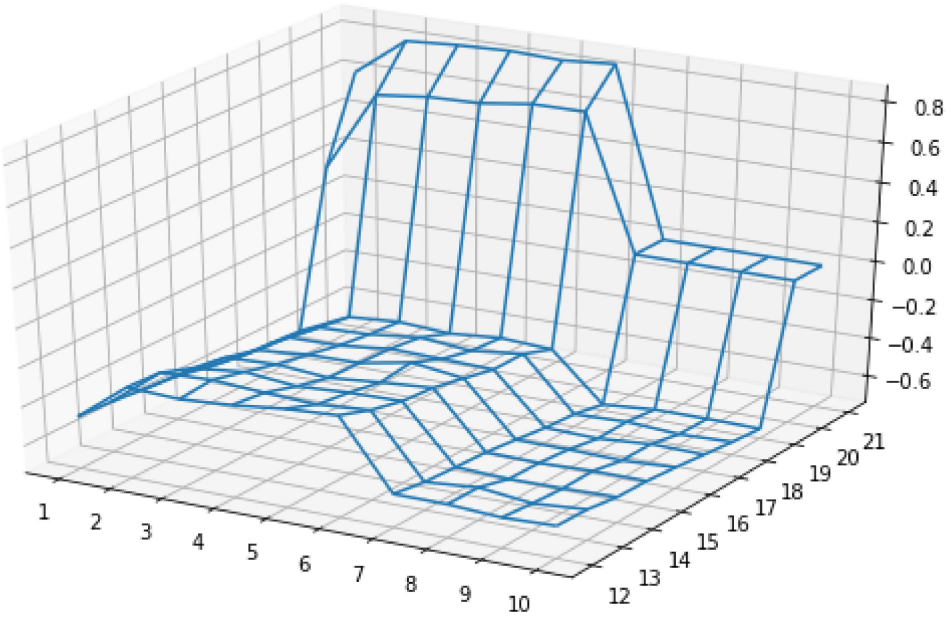
Out[18]:



In [16]:

```
from IPython.display import Image
Image(filename='usable_ace_q.png')
```

Out[16]:



2.5 Solve exercise 6.2.

In the example explained the intial updates of TD would be better because we already have a godd estimate from the old building we just need to improve upon our inter-building estimate. While in Monte Carlo we'd have to start all over again and generate full episodes and then update.

2.6 Write the necessary code and generate the figures in Example 6.2. Answer the related questions asked in exercises 6.3, 6.4, and 6.5.

In [2]:

```
1
```

2.6.1 Exercise 6.3

From the results shown in the left graph of the random walk example it appears that the first episode results in a change in only V (A). What does this tell you about what happened on the first episode? Why was only the estimate for this one state changed? By exactly how much was it changed?

Type *Markdown* and LaTeX: α^2

2.6.2 Exercise 6.4

The specific results shown in the right graph of the random walk example are dependent on the value of the step-size parameter, α. Do you think the conclusions about which algorithm is better would be affected if a wider range of α values were used? Is there a different, fixed value of α at which either algorithm would have performed significantly better than shown? Why or why not?

Type *Markdown* and LaTeX: α^2

2.6.3 Exercise 6.5

In the right graph of the random walk example, the RMS error of the TD method seems to go down and then up again, particularly at high α's. What could have caused this? Do you think this always occurs, or might it be a function of how the approximate value function was initialized?

Type *Markdown* and LaTeX: α^2

2.7 Write the code and generate the figure that compares the sum of rewards during episodes when using Q-learning and SARSA.

2.7.1 Q-learning

In [ ]:

```
1
```

2.7.2 SARSA

In [ ]:

```
1
```

2.7.3 Comparison

In [ ]:

```
1
```

2.8 Solve Exercise 6.12.

Suppose action selection is greedy. Is Q-learning then exactly the same algorithm as Sarsa? Will they make exactly the same action selections and weight updates?

No. Even if action selection is greedy, SARSA has stochastic action selection with highest probability of the greedy action, so, though with low probability, it still may choose the non-greedy aciton, while Q-learning with greedy action selection will deterministically always choose the greedy action. Since they choose different actions they'll have different weight updates.