INDENG 290 HW2 Solutions

We are selecting the solutions from Question from Stanislas Bucaille, Question 2 from April (Yutong) Yang, Question 3 from Kevin Shi, Mia Albery, Yihua Huang, and Question 4 from Yihua Huang and Mia Albery.

Problem 3 and Problem 4 are model-based problems and customized state space and reward function in the model, so the answers differ among students. The solutions I select for these two problems are students' examples of unique thoughts and designs, which students could inspire by these ideas. The grading criteria is not based on these examples.

Problem 1

Student: Stanislas Bucaille

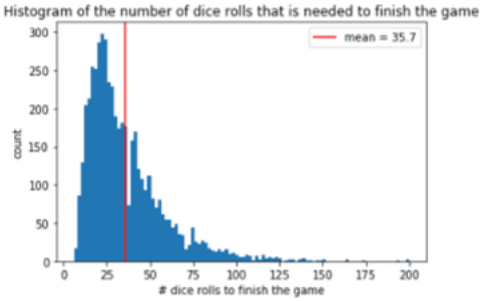
Question 1

In [3]:
```
# Bridges
bridges = {1:18, 4:14, 9:31, 21:42, 28:84, 36:44, 51:67, 71:91, 80:100,          # scales
           98:78, 95:75, 93:73, 87:24, 64:60, 62:19, 56:53, 49:11, 47:26, 16:6}   # snakes

# Simulation
def run_simulation():
    X = 0
    state = 0
    stop = False
    while stop == False:
        action = rd.randint(1,6)
        temp_state = state + action
        if temp_state in bridges.keys():
            state = bridges[temp_state]
        else:
            state = temp_state
        X += 1
        if state >= 100:
            stop = True
    return X

# Run
runs = 5000
simulations = []
for i in range(runs):
    X = run_simulation()
    simulations.append(X)
```

In [4]:
```
mean_sim = np.mean(simulations)
print('Expected number of dice rolls needed to finish the game: ', mean_sim)
```

```
Expected number of dice rolls needed to finish the game:  35.666
```

In [5]:
```
plt.hist(simulations, bins=100)
plt.axvline(x = mean_sim, color = 'r', label = f'mean = {round(mean_sim,1)}')
plt.title('Histogram of the number of dice rolls that is needed to finish the game')
plt.xlabel('# dice rolls to finish the game')
plt.ylabel('count')
plt.legend()
plt.show()
```



Problem 2:

Student: April (Yutong) Yang

For the two MDPs, we want to find the policy π^* that maximize the total cumulative rewards. Accordingly we have

- the optimal state-value function: $V^*(s) = \underset{\pi}{max} V^\pi(s)$
- the optimal action-value function: $Q^*(s,a) = \underset{\pi}{max} Q^\pi(s,a)$

From the Bellman equation, we have

$$\begin{aligned} V^*(s) &= \underset{a}{max} Q^\pi(s,a) \\ &= \underset{a}{max} E_{\pi^*}[R(s)|s_t = s, a_t = a] \end{aligned}$$

We plug in M_1 and M_2,

$$\begin{aligned} V_1^*(s) &= \underset{a}{max} Q_1^\pi(s,a) \\ &= \underset{a}{max} E_{\pi^*}[R_1(s)|s_t = s, a_t = a] \\ V_2^*(s) &= \underset{a}{max} Q_2^\pi(s,a) \\ &= \underset{a}{max} E_{\pi^*}[R_2(s)|s_t = s, a_t = a] \\ &= \underset{a}{max} E_{\pi^*}[R_1(s) + c|s_t = s, a_t = a] \\ &= c + \underset{a}{max} E_{\pi^*}[R_1(s)|s_t = s, a_t = a] \end{aligned}$$

If we seek the π^* that maximizes $V_2^*(s)$, it is clear that π^* is not dependent on c. Since the two MDPs share the same properties other than R, the optimal policy would be the same for M_1 and M_2.

Problem 3

(a) Express with clear mathematical notation the state space, action space, transition probabilities and rewards of an MDP so that the above frog-escape problem can be solved by arriving at the Optimal Action Value Function Q^* of this MDP

State Space: $\mathcal{S} = \{0, 1, \cdots, n\}$

Terminal States: $\mathcal{T} = \{0, n\}$

Non-Terminal States: $\mathcal{N} = \{1, 2, \cdots, n-1\}$

Action space: $\mathcal{A} = \{A, B\}$

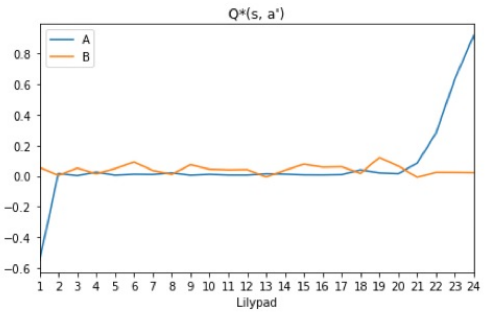
Transition probabilities:

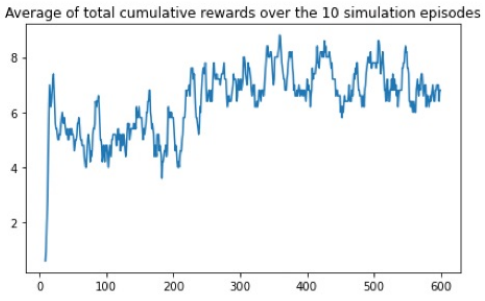
$$p(s \,|\, i, A) = \begin{cases} 0 & \text{if } i = 0 \\ 0 & \text{if } i = n \\ 0 & \text{if } s \neq i-1, i+1 \\ \frac{i}{n} & \text{if } i \neq 0, n, \ s = i-1 \\ \frac{n-i}{n} & \text{if } i \neq 0, n, \ s = i+1 \end{cases} \qquad p(s \,|\, i, B) = \begin{cases} 0 & \text{if } i = 0 \\ 0 & \text{if } i = n \\ 0 & \text{if } s = i \\ \frac{1}{n} & \text{if } i \neq 0, n, \ s \neq i \end{cases}$$

Rewards:

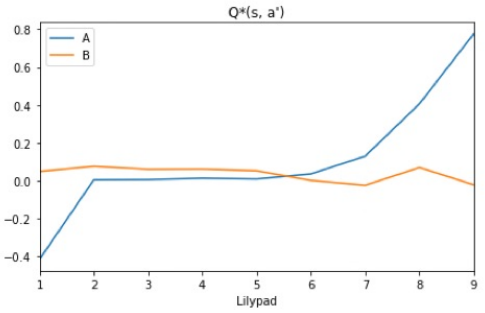
$$R(s) = \begin{cases} 1 & \text{if } s = n \\ -1 & \text{if } s = 0 \\ 0 & \text{otherwise} \end{cases}$$

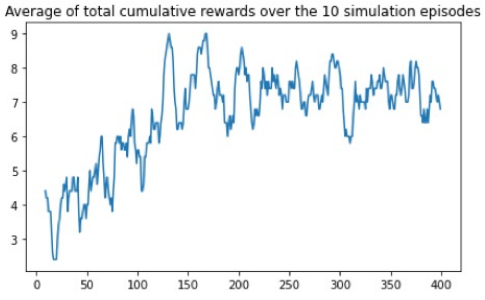
[6]:
```
F = frog(25, 0.95, 0.01, 0.9, 0.001)
F.q_learning_n(600)
F.all_plot()
```



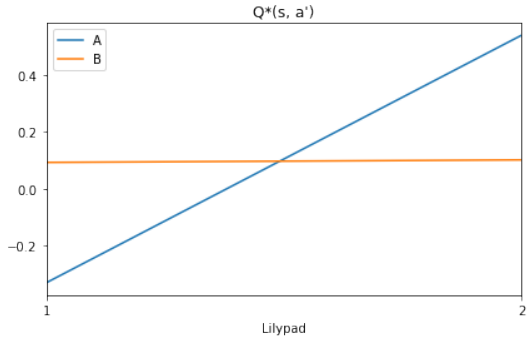


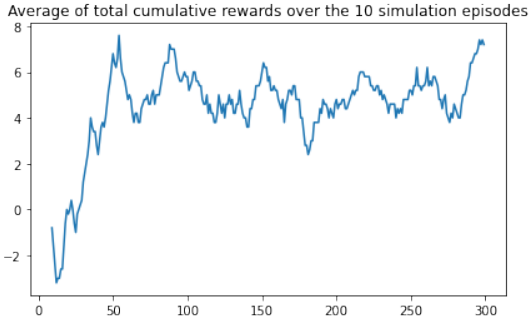
[7]:
```
F = frog(10, 0.95, 0.01, 0.9, 0.001)
F.q_learning_n(400)
F.all_plot()
```





[8]:
```
F = frog(3, 0.95, 0.01, 0.9, 0.001)
F.q_learning_n(300)
F.all_plot()
```





Student : Kevin Shi

Problem 3

(a)

Let n denoted the total number of lilypads.

State Place:

The lilypad where the frog is located in time t, denoted by s_t,$0 \leq s_t \leq n,\ and\ s_t \in N$.

Action Place:

Let a_t denote the action which the frog choose at time t. Then:

$$a_t = \begin{cases} 0 & when\ the\ frog\ croaks\ sound\ A \\ 1 & when\ the\ frog\ croaks\ sound\ B \end{cases}$$

Transition Probability:

Let $P_{s_t,s_{t+1}}(a_t)$ denote the probability to transit from state s_t to s_{t+1} using action a_t at time t. Then:

$$P_{s_t,s_{t+1}}(a_t) = \begin{cases} \frac{s_t}{n} & if\ s_{t+1} = s_t - 1 \\ \frac{n - s_t}{n} & if\ s_{t+1} = s_t + 1 \\ 0 & if\ else \end{cases}$$

when $a_t = 0$.

$$P_{s_t,s_{t+1}}(a_t) = \frac{1}{n} \quad for\ any\ s_{t+1}$$

when $a_t = 1$.

Reward Function:

Let $R(s_t)$ denote the reward when the MDP goes in s_t at time t. Then:

$$R(s_t) = \begin{cases} 100 & when\ s_t = n \\ -100 & when\ s_t = 0 \end{cases}$$

(b)

Solving the problem using a ϵ-greedy algorithm to solve the problem.

I set the learning rate to follow the following equation:

$$\alpha_{s,a}(n_s) = \frac{0.001}{{n_s}^{0.6}}$$

where n_s denote the number of visit of state s.

Then easy to know that the following condition of convergence is satisfied:

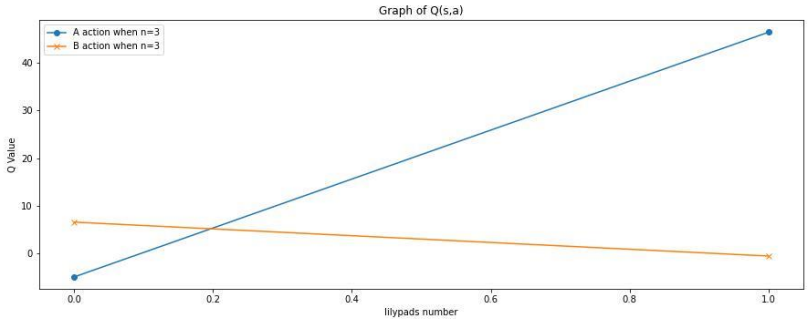
$$\sum_{i=0}^{\infty} \alpha_{i(s,a)} = \infty, \quad \sum_{i=0}^{\infty} \alpha_{i(s,a)}^2 < \infty, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}.$$

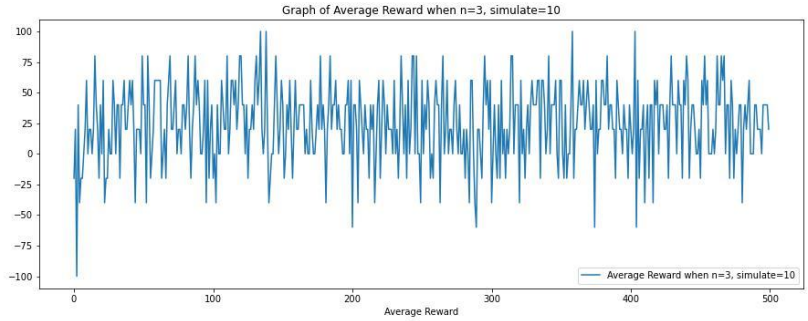
I also set the ϵ to follow the following equation:

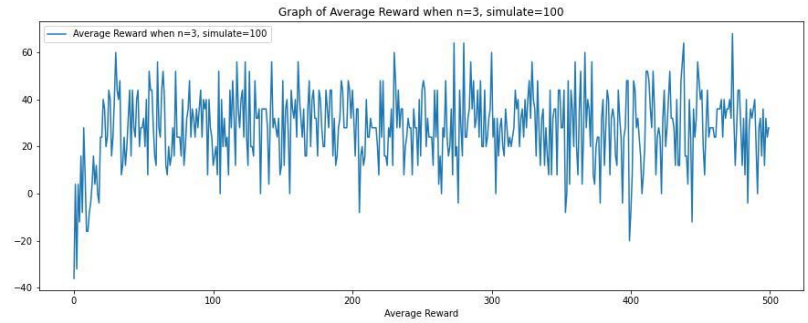
$$\epsilon = \frac{0.7}{{n_s}^{0.6}}$$

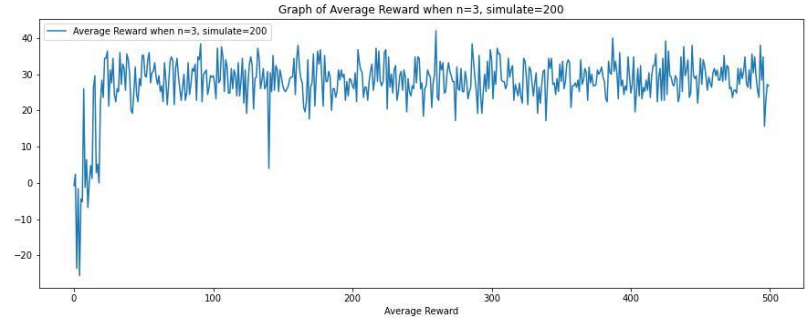
So when the training begins, the algorithm will search more randomly. And with training going on, the algorithm will focus on the best policy which the algorithm found.

Result when n=3(Episode=500, Simulate = 10,100,200):

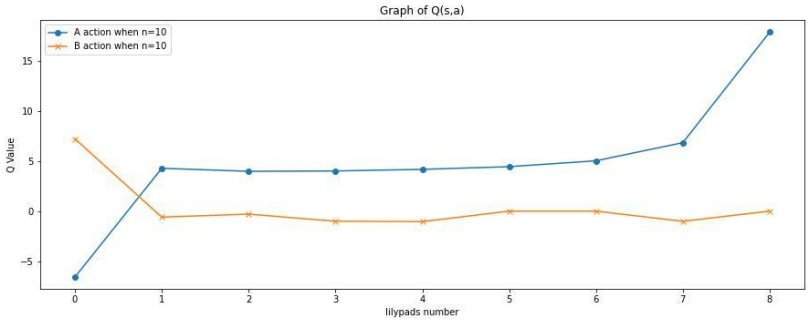


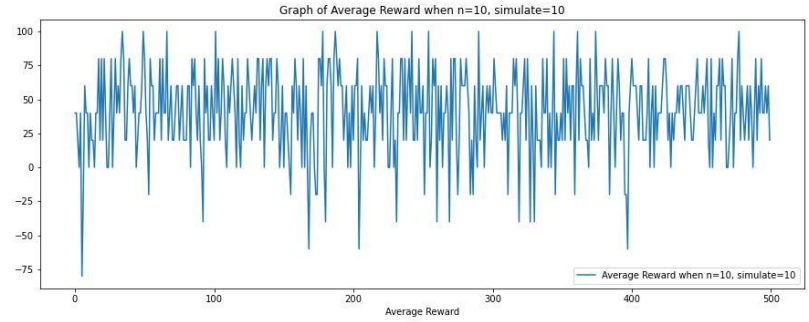


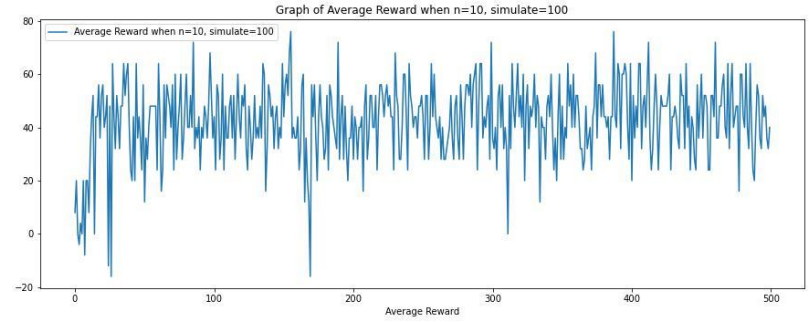


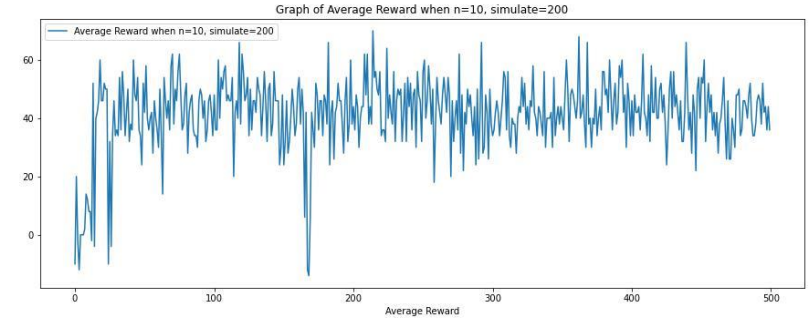


Result when n=10(Episode = 500, simulate =10,100,200):

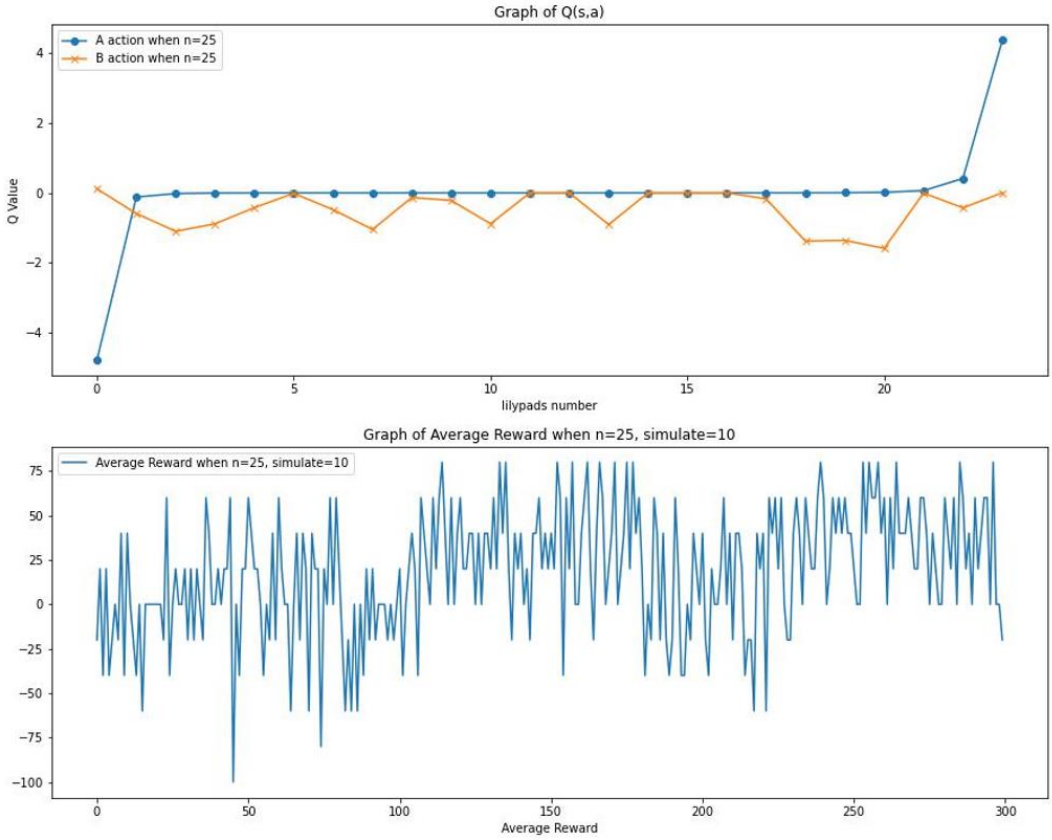








Result when n=25(Episode = 300, simulate =10) (Running too many times will let my computer shut down):



These are the results I got from running my computer more than 20 hours when n=25. I tried to leave it running for two days, but I have an old computer model, so it crashed straight away. Nevertheless I can see that the reinforcement learning algorithm is moving it towards the optimum.

I'm pretty sure that if I have a good computer and let it run for long enough, it will reach the optimal solution, and the optimal policy will be except the first lilypad, in all the other place we should choose sound A to maximize the survive rate.

Mia Albery
October 12, 2022
INDENG290: Homework 2

3. **Problem 3 (40 pts).** Consider an array of $n+1$ lilypads on a pond, numbered 0 to n. A frog sits on a lilypad other than the lilypad numbered 0 or n. When on lilypads i $(1 \leq i \leq n-1)$, the frog can croak one of two sounds A or B. If it croaks A when on lilypad i $(1 \leq i \leq n-1)$, it is thrown to lilypad $i-1$ with probability $\frac{i}{n}$ and is thrown to lilypad $i+1$ with probability $\frac{n-i}{n}$. If it croaks B when on lilypad i $(1 \leq i \leq n-1)$, it is thrown to one of the lilypads $0, \ldots, i-1, i+1, \ldots n$ with uniform probability $\frac{1}{n}$. A snake, located on lilypad 0, will eat the frog if the frog lands on lilypad 0. The frog can escape the pond (and hence, escape the snake!) if it lands on lilypad n. What should the frog croak when on each of the lilypads $1, 2, \ldots, n-1$ in order to maximize the probability of escaping the pond (i.e., reaching lilypad n before reaching lilypad 0)? Although there are more than one ways of solving this problem, we'd like to solve it by modeling it as an MDP and identify its Optimal Action Value function Q^*.

   (a) Express with clear mathematical notation the state space, action space, transition probabilities and rewards of an MDP so that the above frog-escape problem can be solved by arriving at the Optimal Action Value Function Q^* of this MDP (20 pts).

   (b) Write working Python code that models this MDP and solves the for the Optimal Action Value Function Q^* of this MDP. For $n = 3$, $n = 10$ and $n = 25$, plot a graph of $Q^*(s, a')$ as a function of the states of this MDP for each action a'. For every training episode, fix a state value function and use it to simulate total cumulative rewards of the frog-escape MDP with 10 random choices of the starting lilypad i. Then for every training episode, plot the average of total cumulative rewards over the 10 simulation episodes (20 pts).

(a)

- States: {lilypad i} where i = 0 to n;
  - Terminal States: {lilypad n, lilypad 0}
  - Non-terminal states: {lilypad i} where i = 1…n-1;
- Actions:
  - Croak A = {A}
  - Croak B = {B}
- Transition Probabilities:
  - $P(i-1|\{A\}) = \frac{i}{n}$
    - Given Action A while on Lilypad i, the probability of moving to lilypad i-1 is $\frac{i}{n}$
  - $P(i+1|\{A\}) = \frac{n-i}{n}$
    - Given Action A while on Lilypad i, the probability of moving to lilypad i+1 is $\frac{n-i}{n}$
  - $P(z|\{B\}) = \frac{1}{n}$
    - Given Action B while on Lilypad i, the probability of moving to lilypad z, where z s one of the lilypads 0,…,i—1,i + 1, n is uniform with probability $\frac{1}{n}$
- Rewards:
  - At Lilypad n:
    - Reward = 1
  - At Lilypad 0,…n-1:
    - Reward = 0

Mia Albery
October 12, 2022
INDENG290: Homework 2

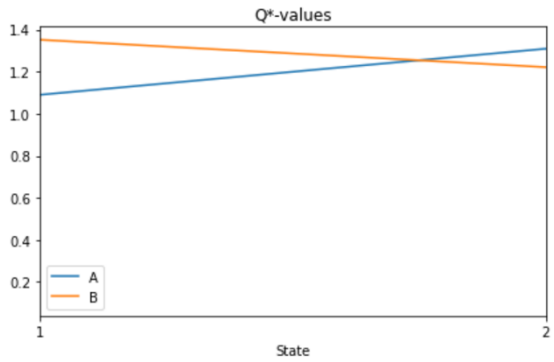
(b) Note: the reward function was originally defined as above. However, upon experimentation, it was found to be more effective in certain instances to use a different reward function. Each set of graphs shows which reward function was used. Additionally, the following hyperparameter scheme was found to converge best:

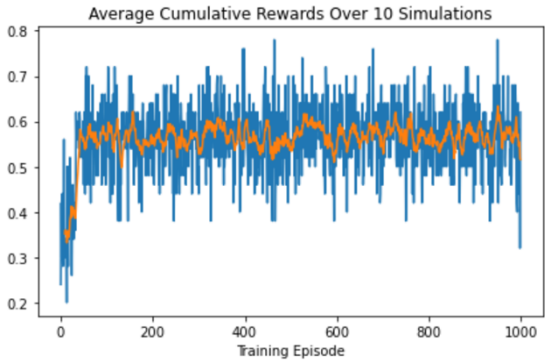
Initial alpha, $\alpha_0 = 0.01$;
Epsilon $\varepsilon_0 = 0.999$
Gamma $\gamma = 0.9$
To discount α:

$$\alpha_i = \left(\frac{\alpha_0 * (total\ training\ episodes - i)}{total\ training\ episodes}\right)$$

where i indicates the current training episode number. This will allow α to converge as the number of training episodes increases.

n = 3:

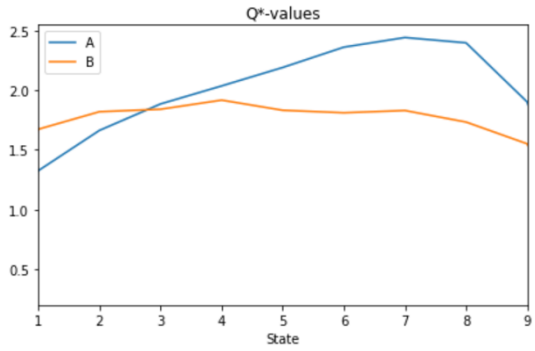


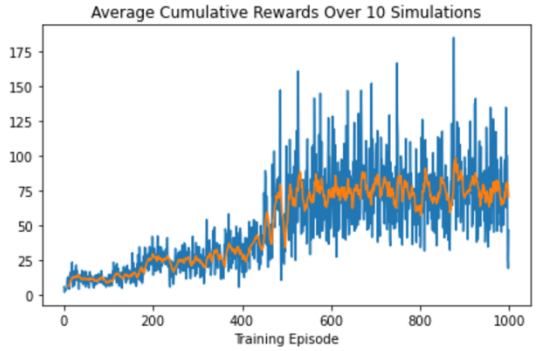


Reward scheme:
State n → reward = 1
State 0,…n-1 → reward = 0

Mia Albery
October 12, 2022
INDENG290: Homework 2

n = 10:



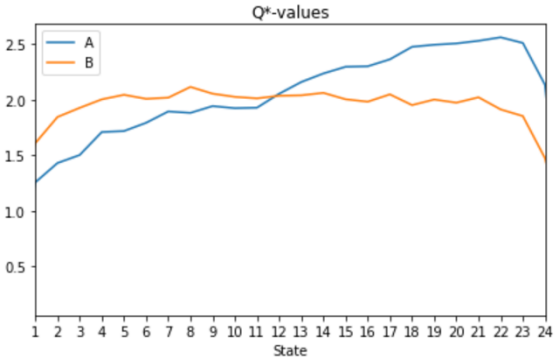


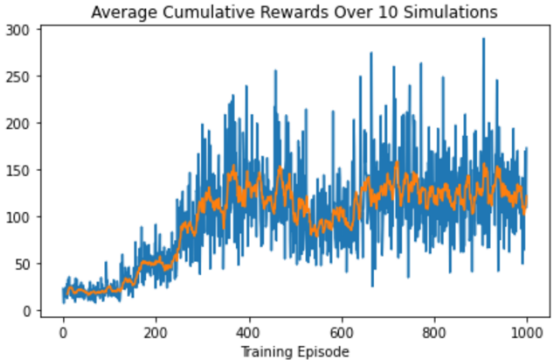
Reward scheme:

Reward = $\frac{i}{n}$

where i indicates the current position on lilypad i, and n indicates the total number of lilypads.

n =25:





Reward scheme:

Reward = $\frac{i}{n}$

where i indicates the current position on lilypad i, and n indicates the total number of lilypads.

Problem 4

Action Space

- A pair of limit orders closer to the top of the orderbook
- A pair of limit orders deeper in the orderbook
- Buy limit order closer to the top of the orderbook
- Buy limit order deeper in the orderbook
- Sell limit order closer to the top of the orderbook
- Sell limit order deeper in the orderbook
- Buy limit order with larger volume and sell limit order with smaller volume
- Sell limit order with larger volume and buy limit order with smaller volume
- Market order to clear its inventory

Reward

Let t_i be the partition of the time, $V_a(t_i), V_b(t_i)$ respectively be the executed volume in the ask or bid side of the LOB since the last time t_i.

Let $m(t_i)$ be the mid price at time t_i. Let $I(t_i)$ be the inventory at time t_i.

Let $p_a(t_i), p_b(t_i)$ be respectively the quoted price of the sell or buy limit order.

Then we define the reward function as follows

$$\text{Reward}(t_i) = V_a(t_i)[p_a(t_i) - m(t_i)] + V_b(t_i)[m(t_i) - p_b(t_i)] + I(t_i)[m(t_i) - m(t_{i-1})]$$

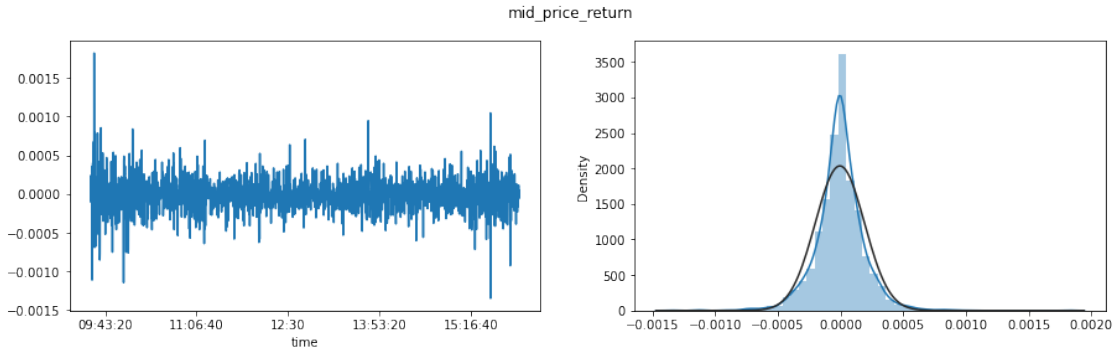
State Space

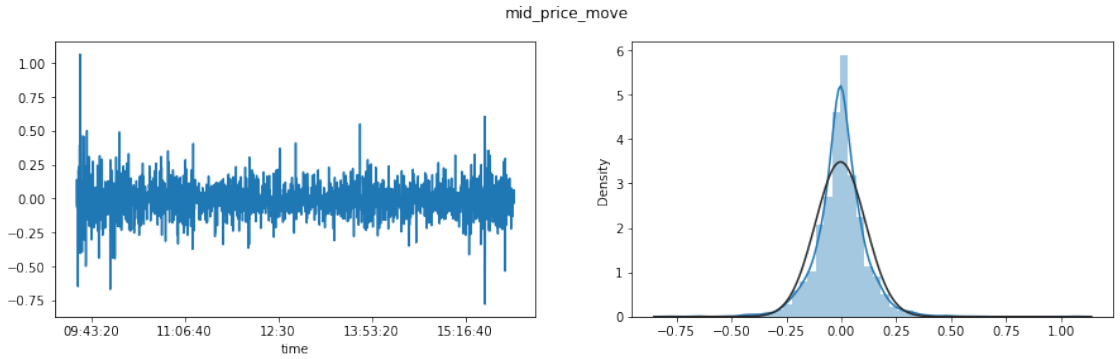
Let $P_a(t_i), P_b(t_i)$ be respectively the best ask price with volume $\nu_a(t_i)$ and the best bid price with volume $\nu_b(t_i)$ at time t_i. Then the micro price is defined as $micro(t_i) = \frac{P_a(t_i)\nu_b(t_i)+P_b(t_i)\nu_a(t_i)}{\nu_a(t_i)+\nu_b(t_i)}$.

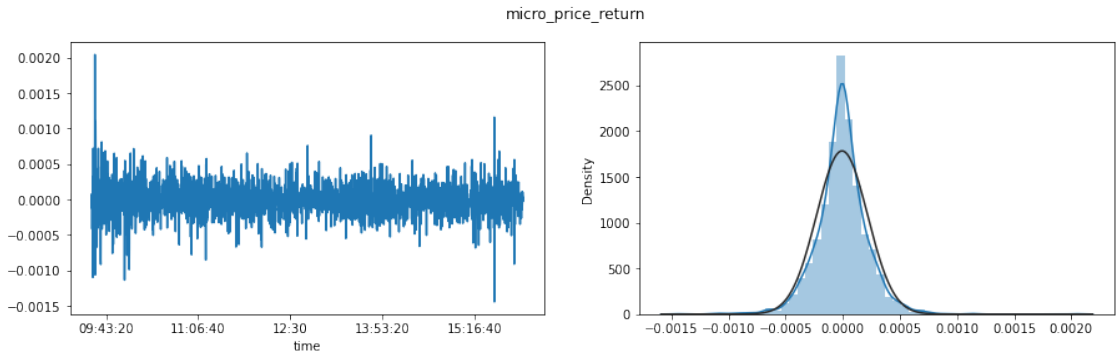
- Mid price move: $m(t_i) - m(t_{i+1})$
- Mid price return: $\log m(t_i) - \log m(t_{i-1})$
- Micro price move: $micro(t_i) - micro(t_{i+1})$
- Micro price return: $\log micro(t_i) - \log micro(t_{i-1})$
- Spread: $P_a(t_i) - P_b(t_i)$
- New spread: $\frac{P_a(t_i)}{P_b(t_i)} - 1$
- Ask spread: $P_a(t_i) - m(t_i)$
- Big spread: $m(t_i) - P_b(t_i)$
- Volume imbalance: sum of the volume from ask side − sum of the volume from bid side
- New volume imbalance: $\frac{\text{sum of the volume from ask side}}{\text{sum of the volume from bid side}} - 1$

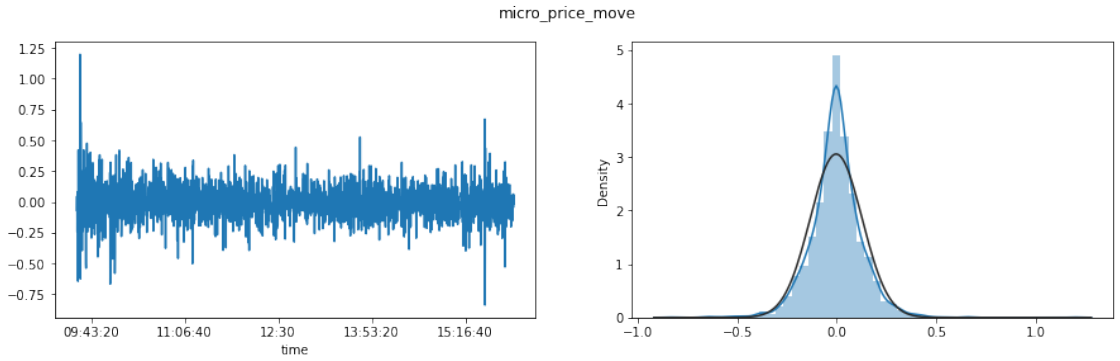
- Volume imbalance ratio: $\frac{\text{sum of the volume from ask side}-\text{sum of the volume from bid side}}{\text{sum of the volume from ask side}+\text{sum of the volume from bid side}}$
- Volatility of mid price: the standard deviation of the mid price within every 10 seconds
- Volatility of mid price: the standard deviation of the micro price within every 10 seconds
- Buy volume: sum of the volume of all executed buy orders within every 10 seconds
- Sell volume: sum of the volume of all executed sell orders within every 10 seconds
- Buy volume ratio: $\frac{\text{Buy volume}}{\text{Buy volume}+\text{Sell volume}}$
- Sell volume ratio: $\frac{\text{Sell volume}}{\text{Buy volume}+\text{Sell volume}}$
- RSI of mid price: RSI of mid price within every 10 seconds
- RSI of micro price: RSI of micro price within every 10 seconds

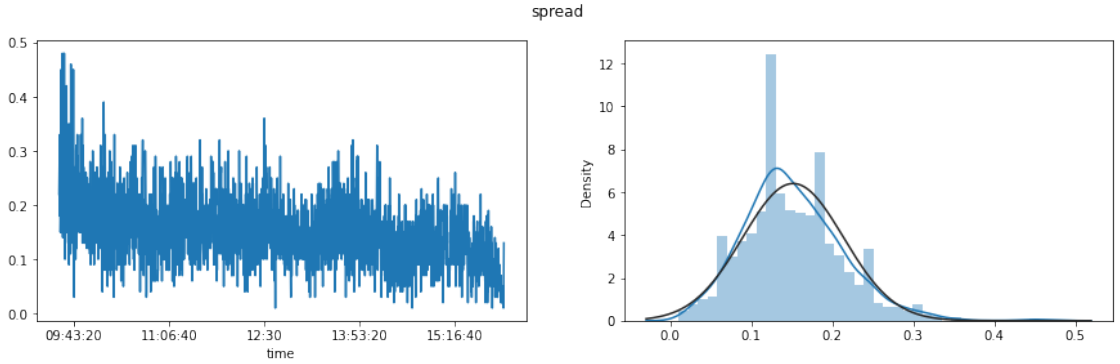
[10]:
```
t = test('AAPL', 10)
t.make_variables()
t.plot_hist()
```

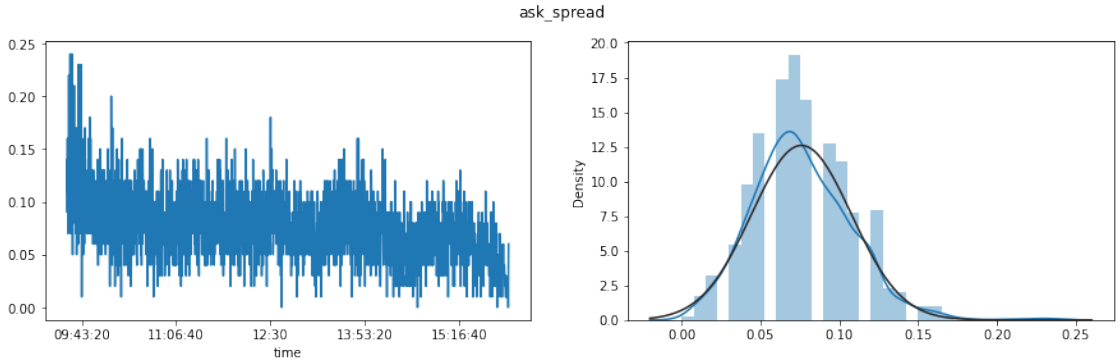


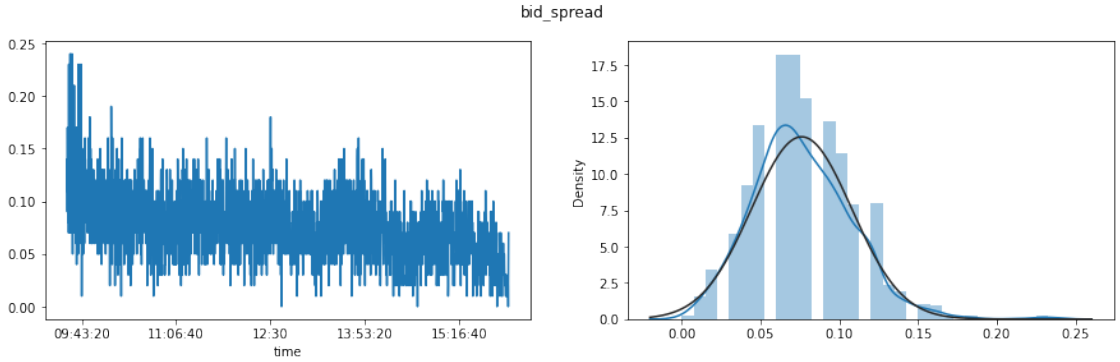


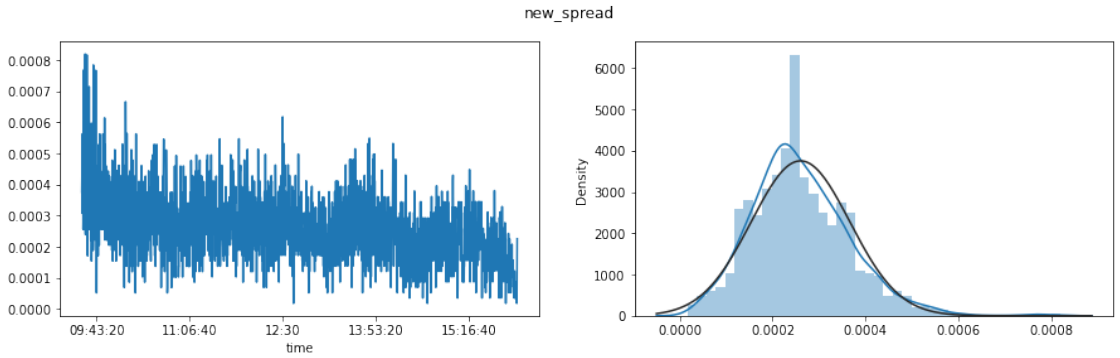


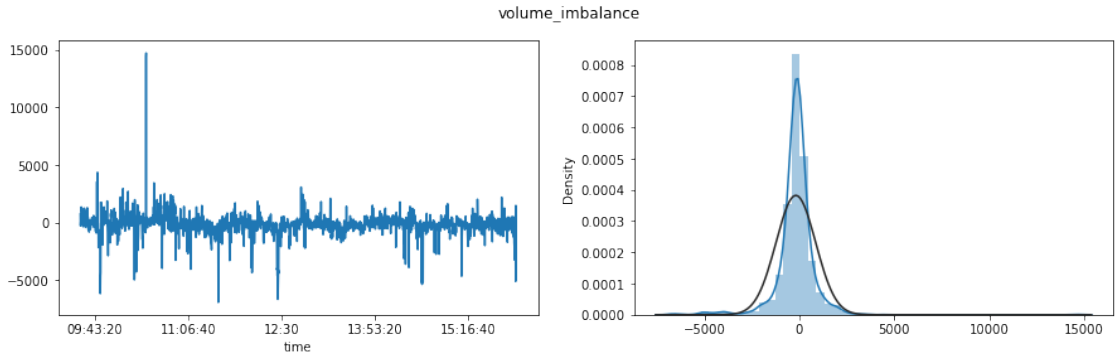

micro_price_move
time
Density

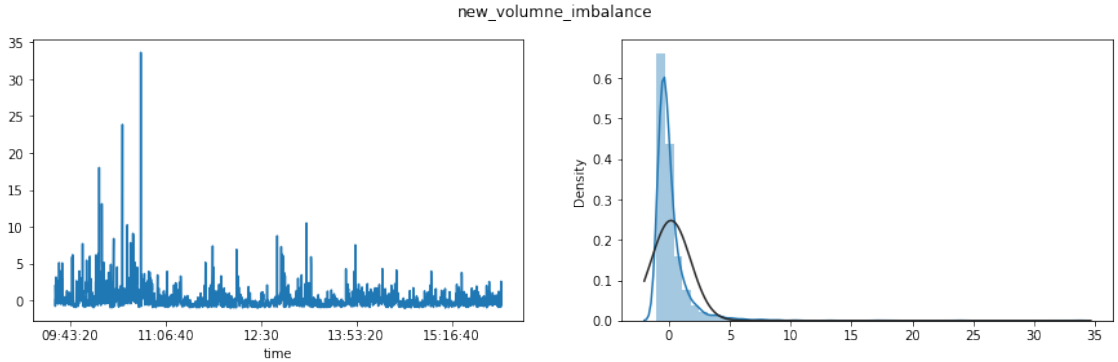

spread
time
Density

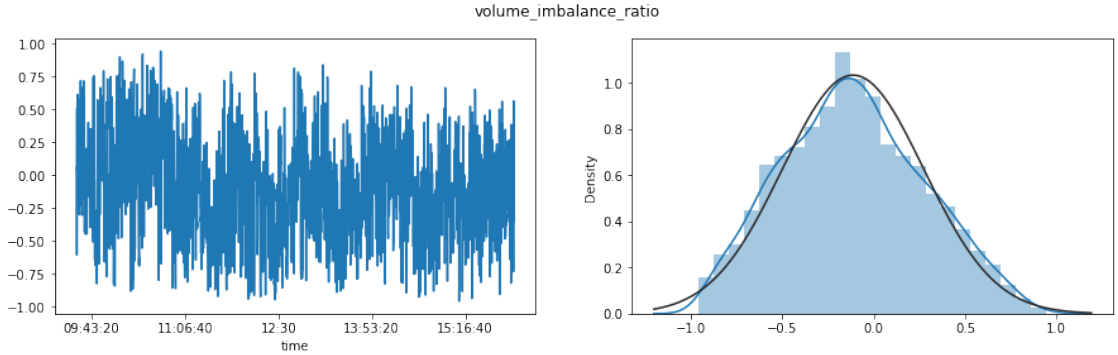

ask_spread
time
Density

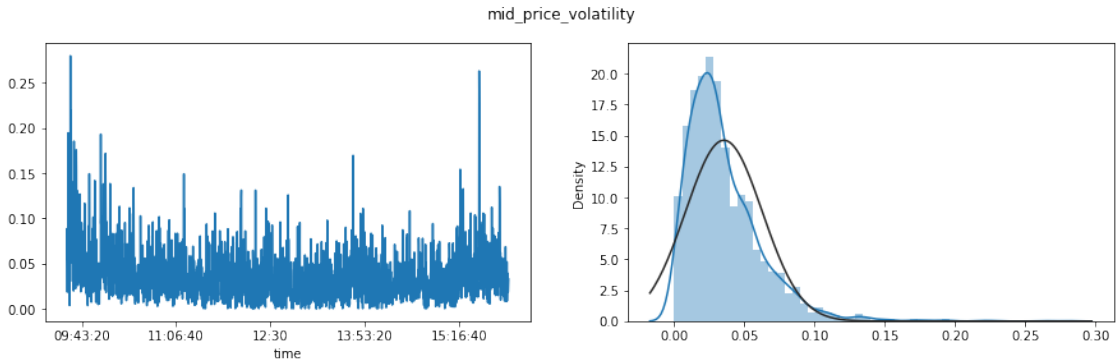

bid_spread
0.25
0.20
0.15
0.10
0.05
0.00
09:43:20
11:06:40
12:30
13:53:20
15:16:40
time
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Density
0.00
0.05
0.10
0.15
0.20
0.25

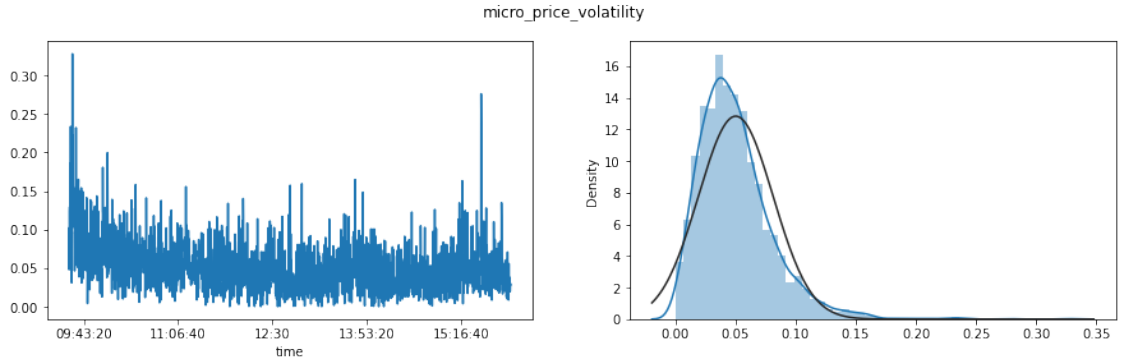

new_spread
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
09:43:20
11:06:40
12:30
13:53:20
15:16:40
time
6000
5000
4000
3000
2000
1000
0
Density
0.0000
0.0002
0.0004
0.0006
0.0008

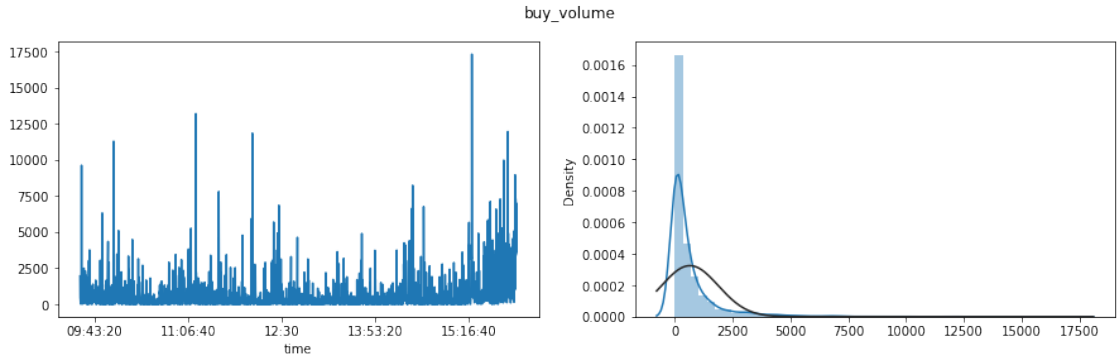

volume_imbalance
15000
10000
5000
0
−5000
09:43:20
11:06:40
12:30
13:53:20
15:16:40
time
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
Density
−5000
0
5000
10000
15000

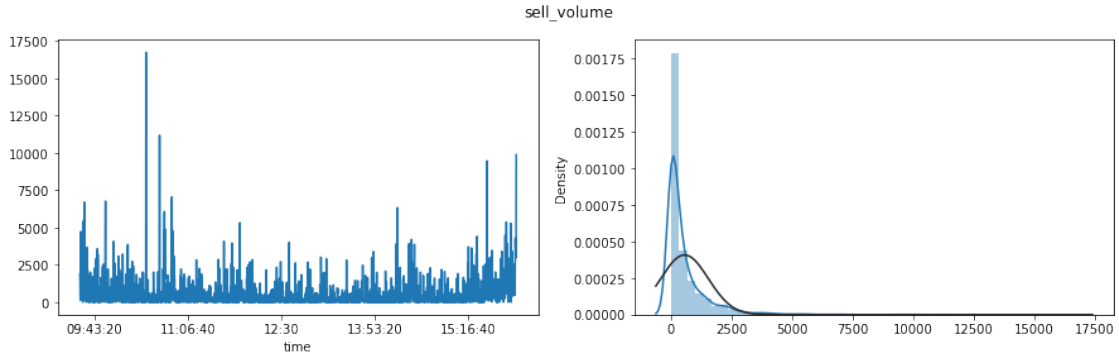

new_volumne_imbalance
time
Density

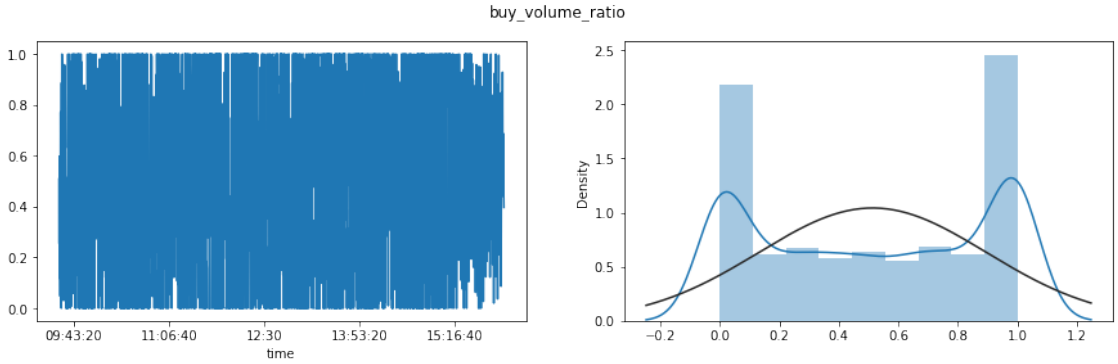

volume_imbalance_ratio
time
Density

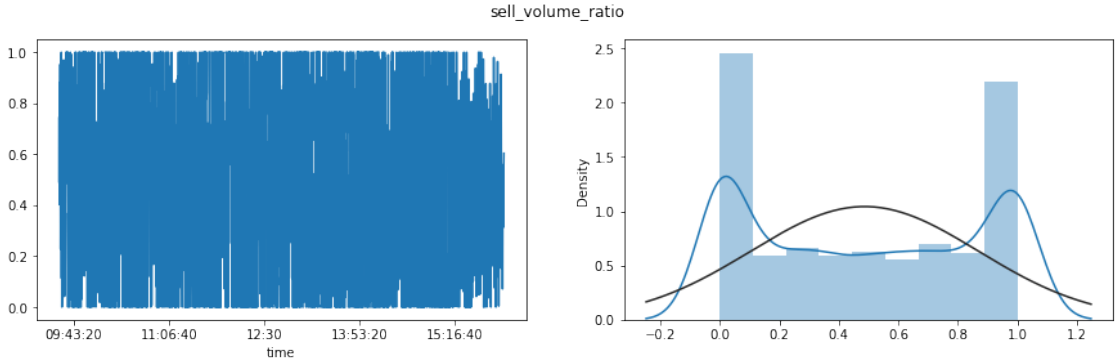

mid_price_volatility
time
Density

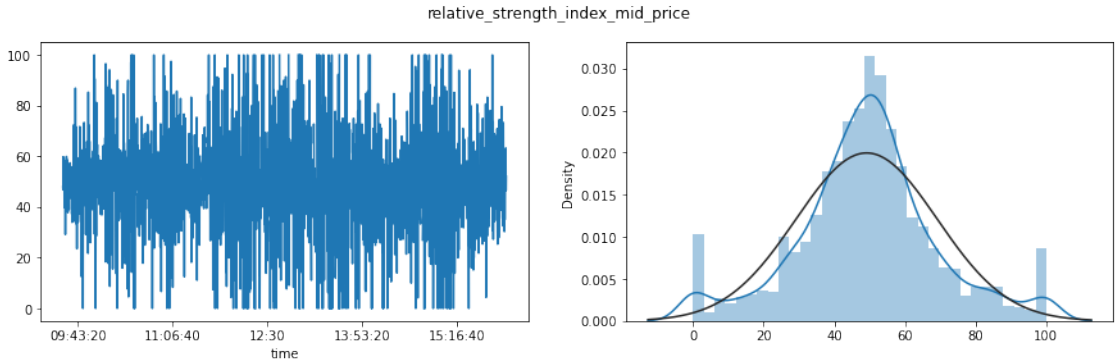

micro_price_volatility
0.30
0.25
0.20
0.15
0.10
0.05
0.00
09:43:20
11:06:40
12:30
13:53:20
15:16:40
time
Density
16
14
12
10
8
6
4
2
0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35

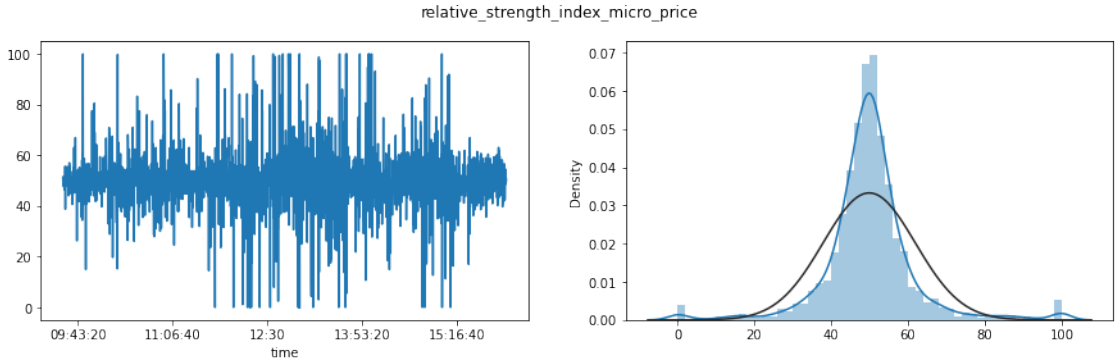

buy_volume
17500
15000
12500
10000
7500
5000
2500
0
09:43:20
11:06:40
12:30
13:53:20
15:16:40
time
Density
0.0016
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
2500
5000
7500
10000
12500
15000
17500


sell_volume
17500
15000
12500
10000
7500
5000
2500
0
09:43:20
11:06:40
12:30
13:53:20
15:16:40
time
Density
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
0
2500
5000
7500
10000
12500
15000
17500


buy_volume_ratio
time
Density


sell_volume_ratio
time
Density


relative_strength_index_mid_price
time
Density



Analysis of State Space

- First of all, I think useful market state variables should be ones that are not normally distributed. Therefore, state variables like "spread", "ask spread", "bid spread", "new spread" and "volume imbalance ratio" are not expected to be useful.
- Both mid price and micro price are considered as an estimate of the fair price. They are not very different when calculaing the move or the return of the price. We can see they all have heavy tails. However, when calculating the volatility, the volatility of mid price are less normally distributed, which means it's better to use "mid price volatility". Nevertheless, when calculating the RSI, we can see the performance of the micro price is much better than the mid price. Therefore, in this case "RSI micro price" should be more useful.
- In terms of the volume imbalance, we can see both "volume imbalance" and "new volume imbalance" skew from the normal distribution, which means they are expected to be useful.
- "Buy volume" and "sell volume" both skew heavily from normal distribution, which shows potential usefulness.
- "Buy volume ratio" and "sell volume ratio" seem to be two very interesting features since its distribution plots look very strange. They seem to show predominant power from buy or sell side.
- At last, although we say the "RSI mid price" is more similar to the normal distribution than that of micro price, we can see "interesting features" from the plot. The density becomes abnormally high when it closes to 0 or 100, which actually shows predominant power from buy or sell side.

Mia Albery
October 12, 2022
INDENG290: Homework 2

Problem 4:

- State Space: {Total Imbalance, Level 1 Imbalance, Spread, Cumulative Executed Volumes, Real-Time Executed Volume, Inventory, 10-second Midpice Move, 60-second Midprice Move, Volatility, Microprice}
- Action Space: {Place Limit Order Size X, Place Market Order Size X, or Hold (do nothing)}
- Rewards: {Goal: Minimize Trading Cost → Receive reward for executing orders at minimal cost}
    - More formally:

$$R_t = \begin{cases} (p_t - p_\sigma) \times f_t & sell\ order \\ (p_\sigma - p_t) \times f_t & buy\ order \end{cases}$$

where at time t, the reward, R_t is calculated using p_t (the price at time t), p_σ (the mid price), and f_t is the size. This calculation occurs when the order the agent places is matched with the opposite order in the market, resulting in an immediate reward [1].

In implementing 10 state variables, it was very helpful to plot their distributions to identify which might be useful in learning an optimal execution agent (all plots of variables versus time, and their distributions, can be seen below). First, I expect the Level 1 imbalance, as well as total Level 1—Level 5 imbalance to be useful in learning an optimal execution agent. There were certain periods throughout the trading day where the imbalance spiked very large amounts, and so having the imbalance as a state is important for the agent to know what to do if a spike is expected or if a spike just happened. Furthermore, the distributions of both of these variables are visibly non-normal, and they are heavily right-skewed. These interesting features mean that they will be important in training the agent. Next, I think the executed volumes and inventory will be useful for the agent in training because the plots of these variables versus time indicate that there are certain times throughout the day when higher trading happens, such as right before the day's close. This is important for the agent to know because it might mean that they want to execute more when other trading volume is low, or also trade when other trading volume is high, based on whatever has a better value. It should be noted that the distributions of these variables are very odd looking, but that is because the cumulative aspect of the variables indicate that a distribution is not as helpful to look at. Lastly, the micro-price distribution has very interesting features, which indicate that it might be very useful for the agent in training. Because micro-price is a price-weighted imbalance, it can give the agent even more specific information about how the price of a stock, and how the available volumes, are changing. The micro-price seems to decline as time progresses, and this might be useful for the agent to optimize its strategy.

On the other hand of this, certain variables might not be as helpful. The 10-second and 60-second midprice moves had relatively normal distributions, and so they might lack distinguishing features for the agent to utilize in its decision-making process. It is worth noting that the midprice moves do have spikes at the beginning of the day, and this might be useful for the agent, but continuously analyzing this over the entire trading day might not be valuable. The spread also has a relatively normal distribution. This indicates that there are not strong features that would help the agent make a decision based on what the spread is, and so solely using

spread as a market state variable will not provide the agent with great training. Lastly, the volatility is pretty consistent throughout the day, except for an initial spike. Thus, similar to the spread, it might not be useful as a training mechanism on its own, but knowing the volatility spikes early in the day might help the agent make some initial decisions.

Overall, analyzing the distributions and plots in time of these variables is very helpful in identifying which might be useful to the agent. Furthermore, as discussed in "*Reinforcement Learning for Optimized Trade Execution*" by Yuriy Nevmyvaka, they might be even more valuable if they are combined [2]. Thus, looking to combine perhaps the spread and the volatility might indicate very valuable features that were not seen in the variables when they were analyzed individually.

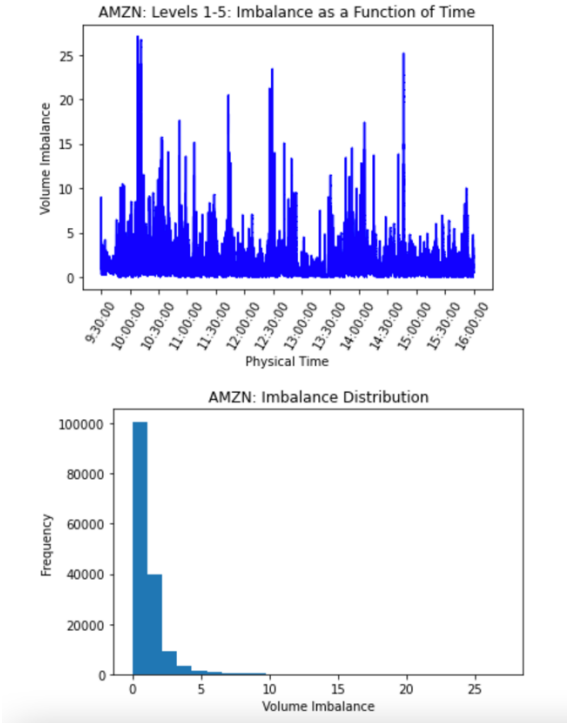
Resources:

[1] S. Vyetrenko and S. Xu, "Risk-Sensitive Compact Decision Trees for Autonomous Execution in Presence of Simulated Market Response." arXiv, Jan. 06, 2021. Accessed: Oct. 12, 2022. [Online]. Available: http://arxiv.org/abs/1906.02312

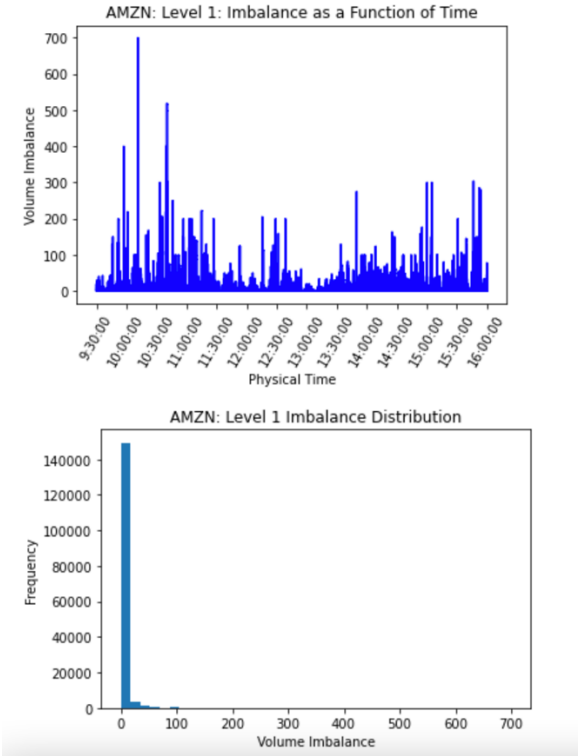
[2] Y. Nevmyvaka, Y. Feng, and M. Kearns, "Reinforcement learning for optimized trade execution," in *Proceedings of the 23rd international conference on Machine learning - ICML '06*, Pittsburgh, Pennsylvania, 2006, pp. 673–680. doi: 10.1145/1143844.1143929.

Mia Albery
October 12, 2022
INDENG290: Homework 2

State Variable #1:

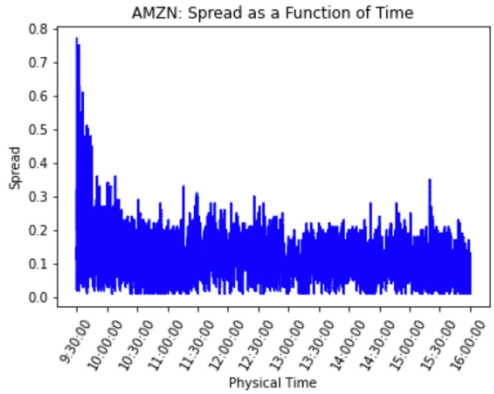


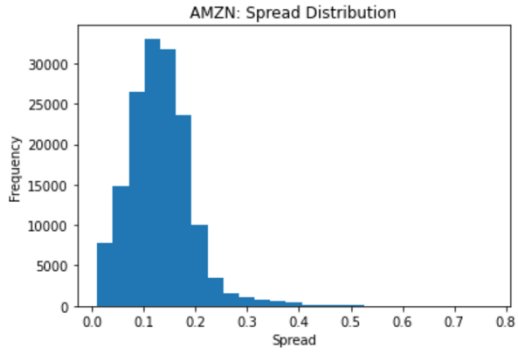
State Variable #2:



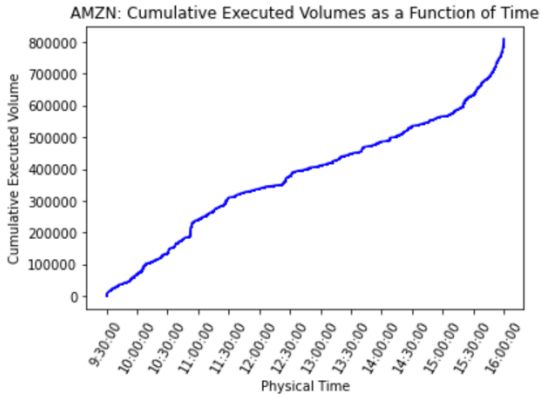
Mia Albery
October 12, 2022
INDENG290: Homework 2

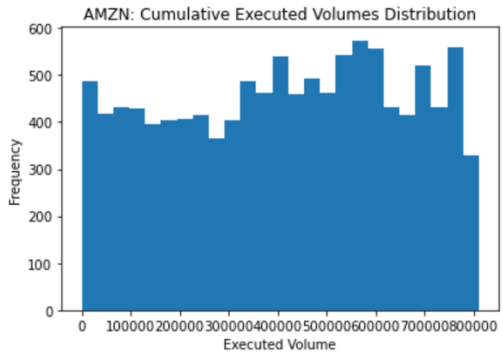
State Variable #3:





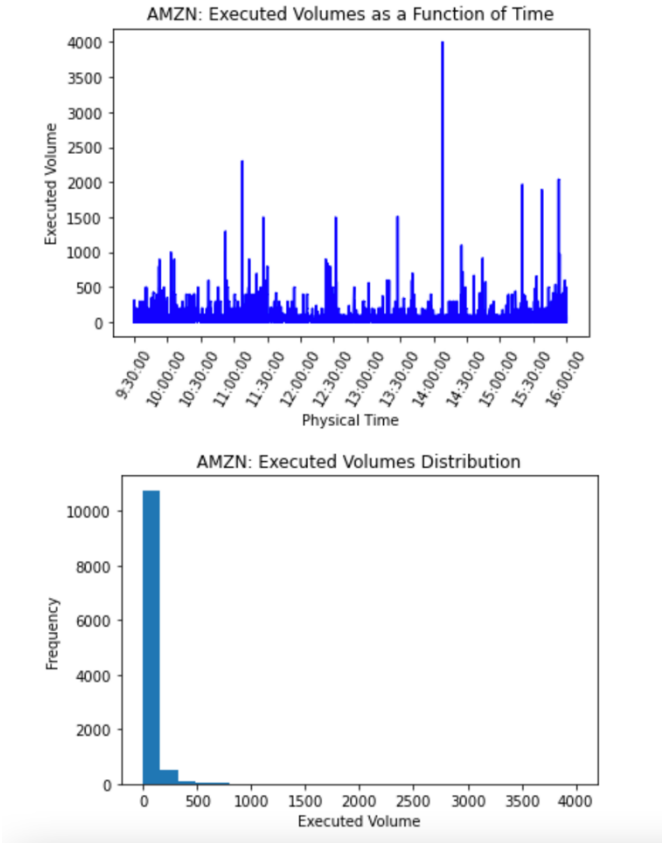
State Variable #4:



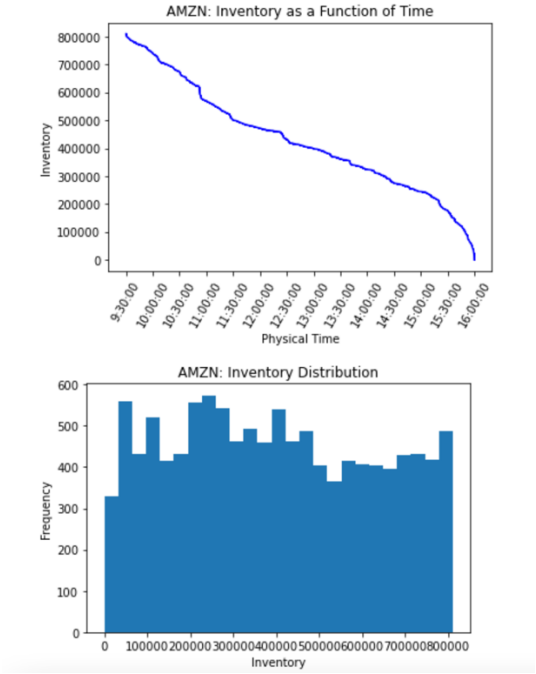


Mia Albery
October 12, 2022
INDENG290: Homework 2

State Variable #5:

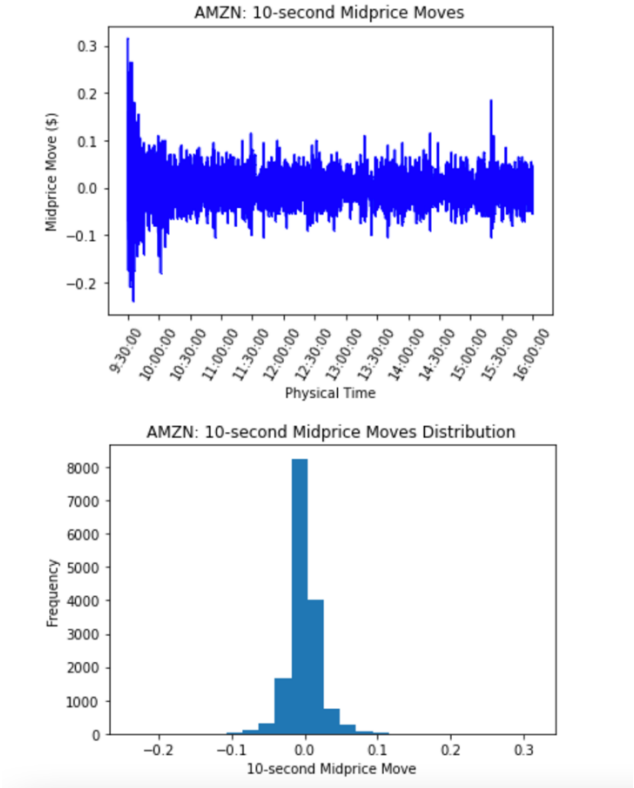


State Variable #6:

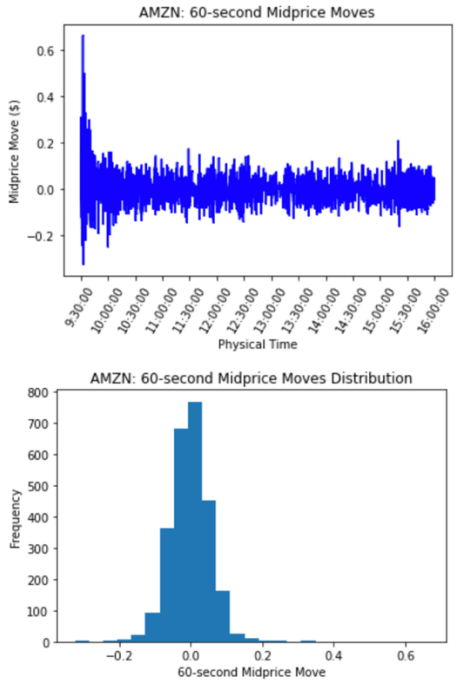


Mia Albery
October 12, 2022
INDENG290: Homework 2

State Variable #7:

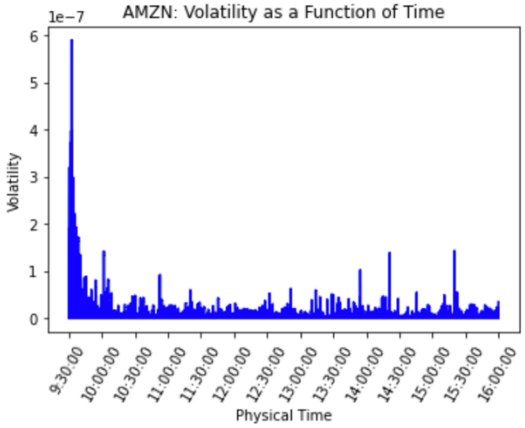


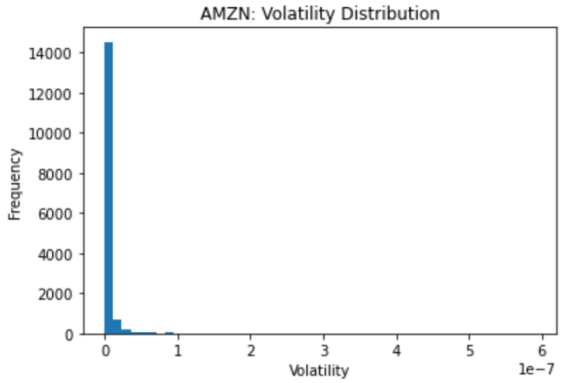
State Variable #8:



Mia Albery
October 12, 2022
INDENG290: Homework 2

State Variable #9:





State Variable #10: