RL PROJECT

CALL OPTIONS PRICING

USING REINFORCEMENT LEARNING ALGORITHMS FOR CALL OPTION PRICING

INTRODUCTION

- *A call option is a financial contract that gives its holder the right, but not the obligation, to buy specified securities at a predetermined price known as the strike price until a specific fixed date known as the expiry date.*

- In our project, an investor has a call option to purchase one share of a stock for a fixed price p and has T days to exercise it. For simplicity, we assume that the investor takes a decision at the start of each day.

- The investor may decide not to exercise the option but if he does exercise the option when the stock price is s, he effectively gets (s - p).

- The price of the stock is assumed to be varied with independent increments, *i.e.*, the price on day t + 1 is $S_{t+1} = S_t + W_t$

INTRODUCTION

- We assume $p \in N$ and W_t to be (discrete) uniformly distributed with endpoints $-\varepsilon$ and $+\varepsilon$ for some $\varepsilon \in N$. For example, for $\varepsilon=5$, W_t $U\{-5,5\}$.

- We developed an agent that determines whether to exercise (or not exercise) the call option over a certain time period of T days.

- Specifically, our agent provides an optimal price for each of the T days, and if the stock price on that day exceeds the optimal price, the investor should sell his call option on that day.

ENVIRONMENT

- Action Space is defined as { 0,1 }.
    - a_t = 0 indicates that we hold
    - a_t = 1 indicates that we sell (i.e. exercise the call option)

- A state is defined as s_t = { X_t , t } where X_t is the current price and t is the days spent.

ENVIRONMENT

- The Reward Function is defined as :
  - $R_t(s_t, a_t) = 0$ if $a_t = 0$
  - $R_t(s_t, a_t) = max(0, X_t - P_s)$ if $a_t = 1$ where P_s is the strike price.
  - The reward function is basically defined as the profit we gain by selling our option.

- The transition from one state to another is defined as $S_{t+1} = S_t + W_t$ where ,W_t Uniform(-val, val) {val is the max deviation possible}.

RL ALGORITHMS USED

We implemented a finite horizon version of the following algorithms:

1. Value Iteration
2. Policy Iteration
3. Monte Carlo control
4. Q-Learning
5. TD control using SARSA(λ)

VALUE ITERATION

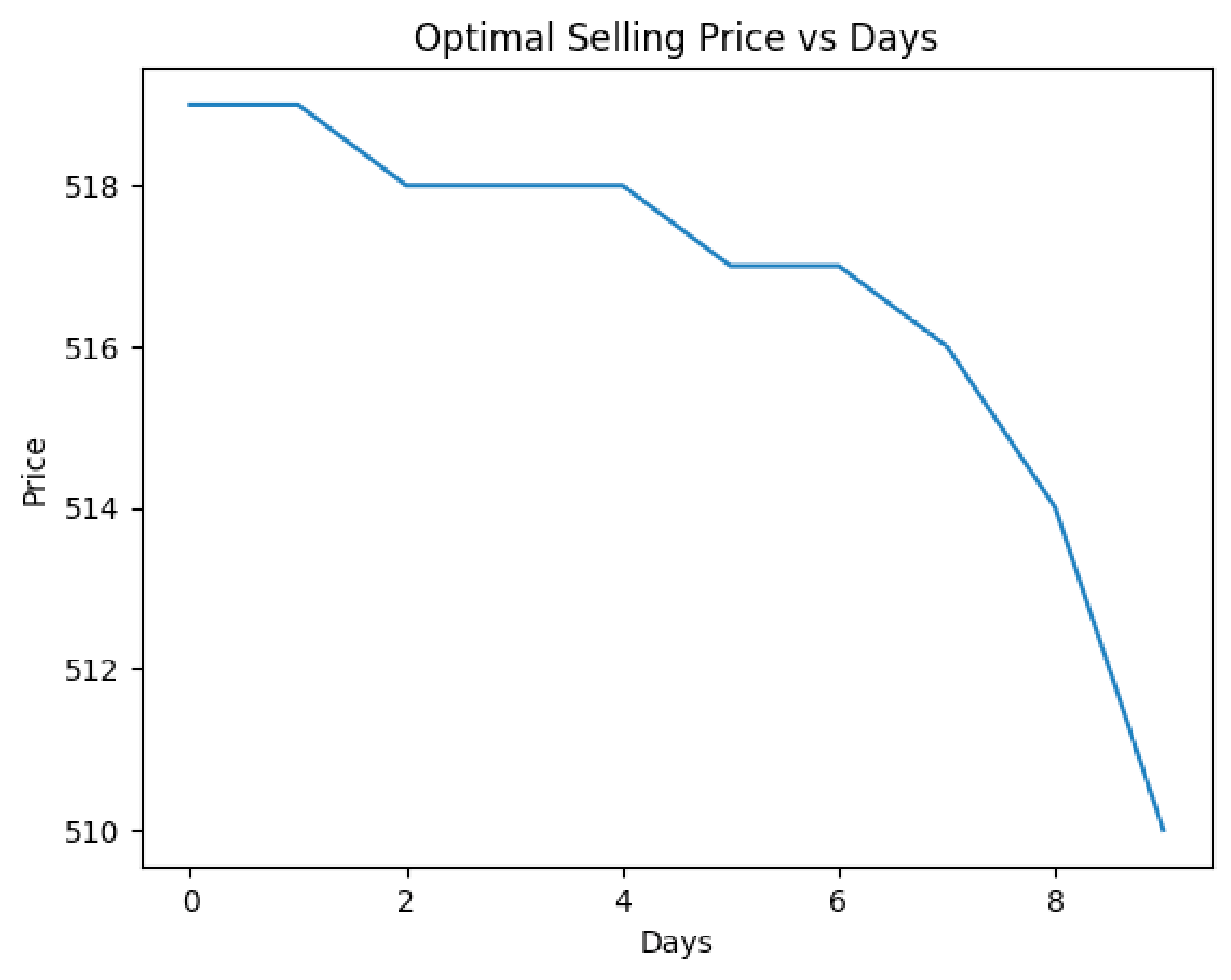
Pseudo Code:

Algorithm 1 Finite Horizon Value Iteration

for $t = T-1, T-2, \ldots, 0$ **do**
  for $s \in \mathcal{S}$ **do**
    $\pi_t(s), V_t(s) = \text{maximize}_a \mathbb{E}\left[r_t + V_{t+1}(s_{t+1})\right]$
  end for
end for

Result Obtained:

(strike price = 510, start price = 500, drift = 5 and T = 10)



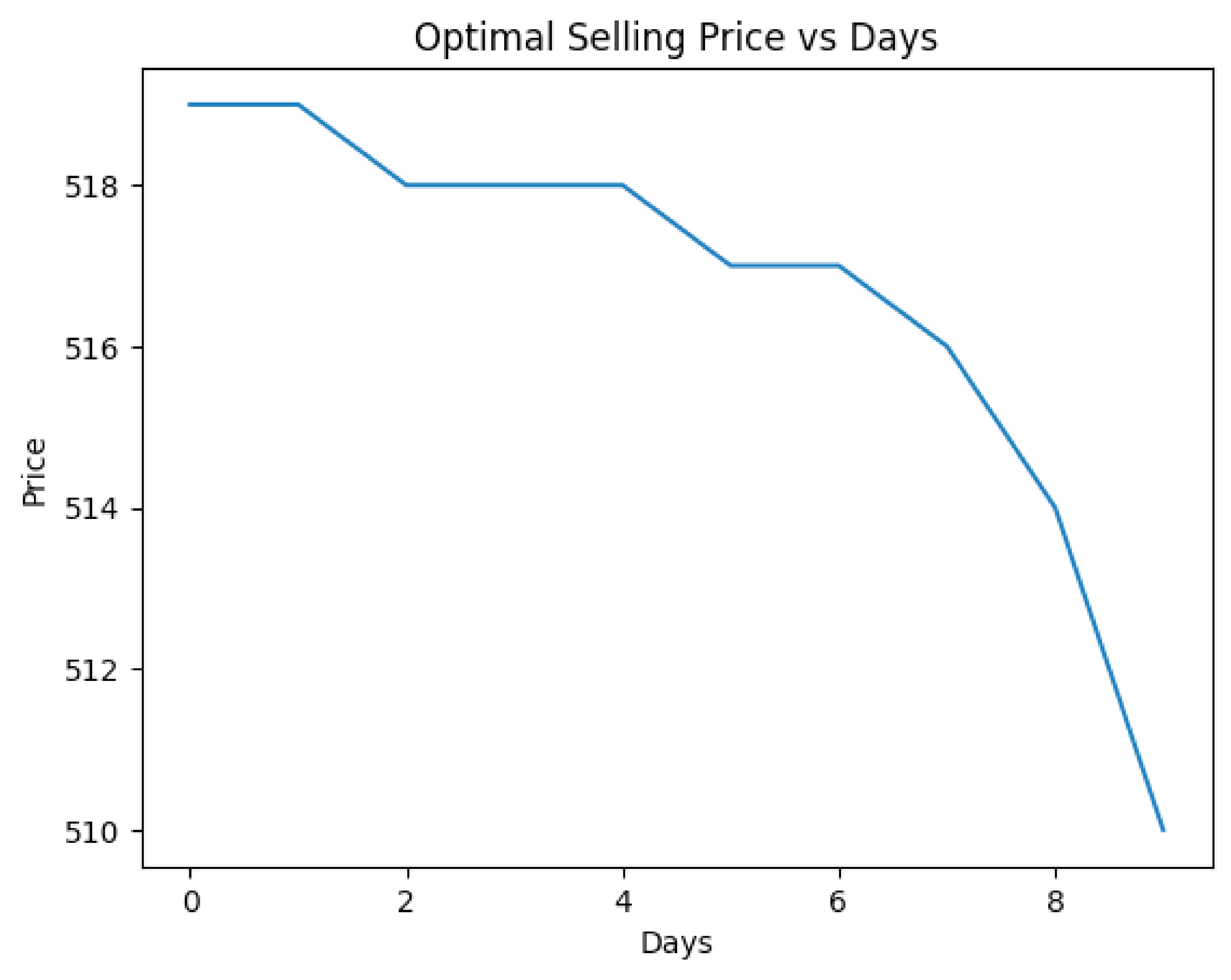
POLICY ITERATION

Pseudo Code

Algorithm 4 Policy Iteration

Initialize $\pi^{(0)}$.
for $n = 1, 2, \ldots$ **do**
  $V^{(n-1)} = \mathrm{Solve}[V = \mathcal{T}^{\pi^{(n-1)}} V]$
  $\pi^{(n)} = \mathcal{G} V^{\pi^{(n-1)}}$
end for

Result Obtained:
(strike price = 510, start price = 500, drift = 5 and T = 10)



MONTE CARLO

We have implemented On-policy every-visit Monte Carlo.

Pseudo Code Followed

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$
Initialize:
  $\pi \leftarrow$ an arbitrary ε-soft policy
  $Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
  $Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

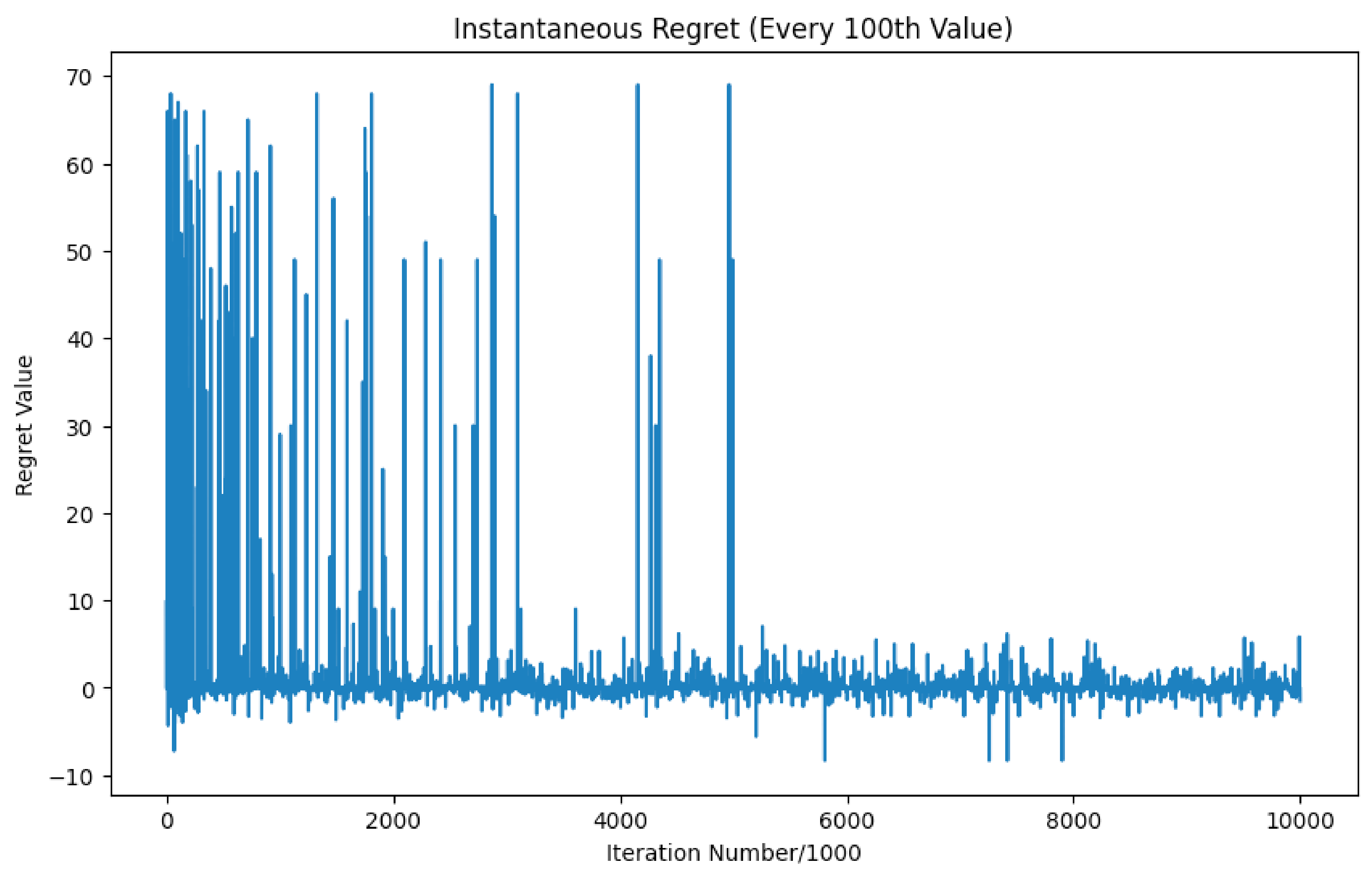
Repeat forever (for each episode):
  Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
      $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
      For all $a \in \mathcal{A}(S_t)$:

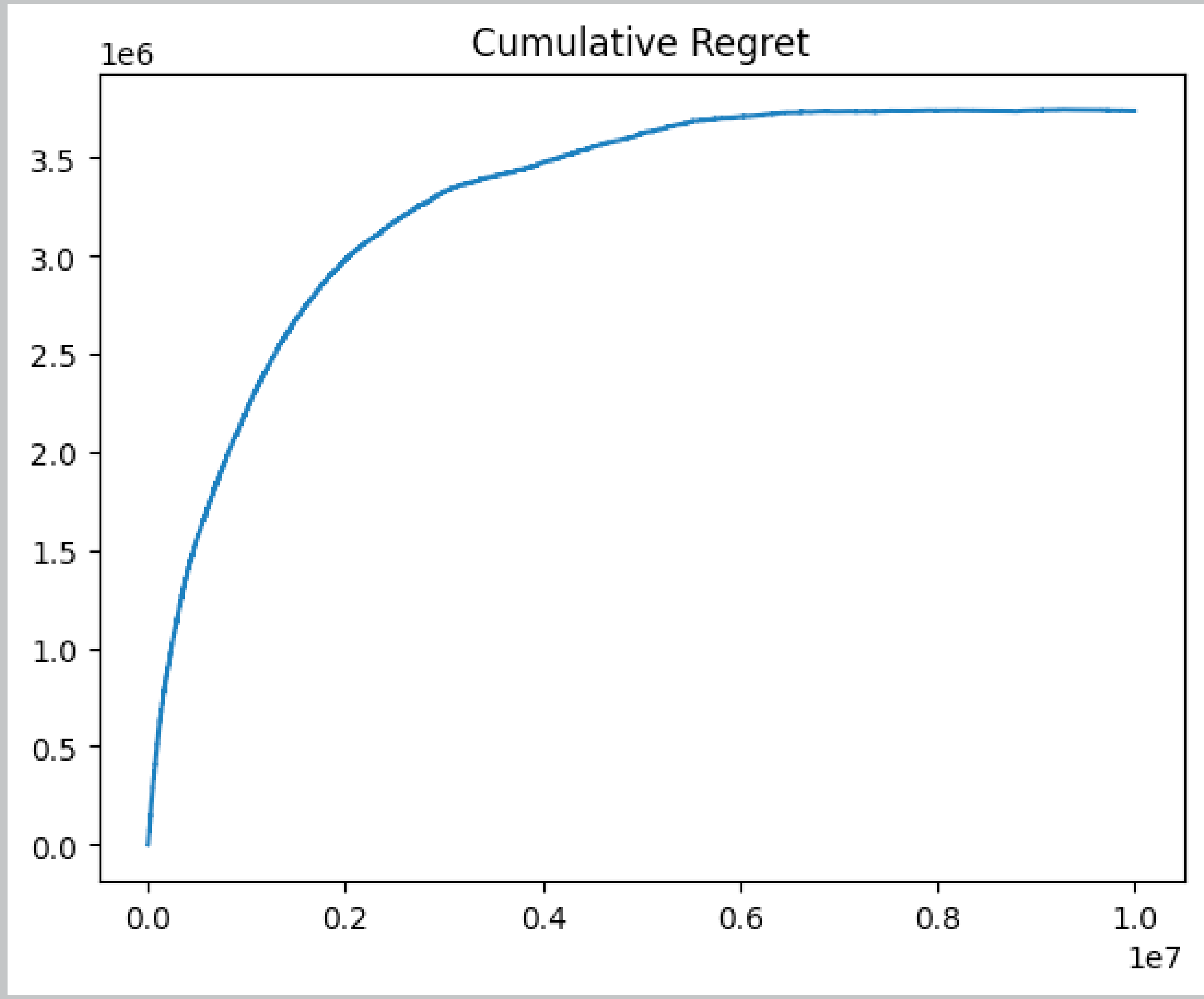
$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

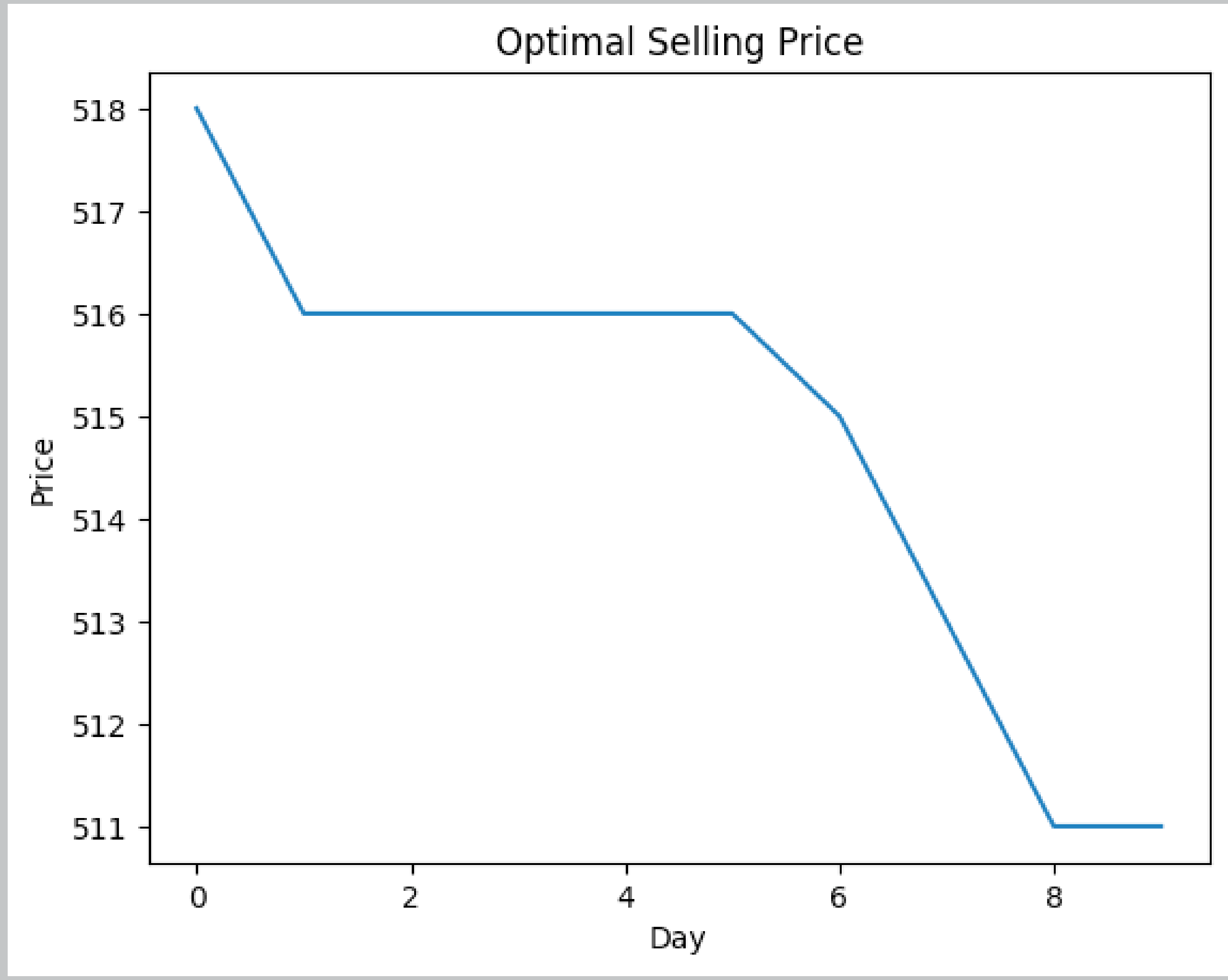
MONTE CARLO

Results Obtained:

(strike price = 510, start price = 500, drift = 8 and T = 10)





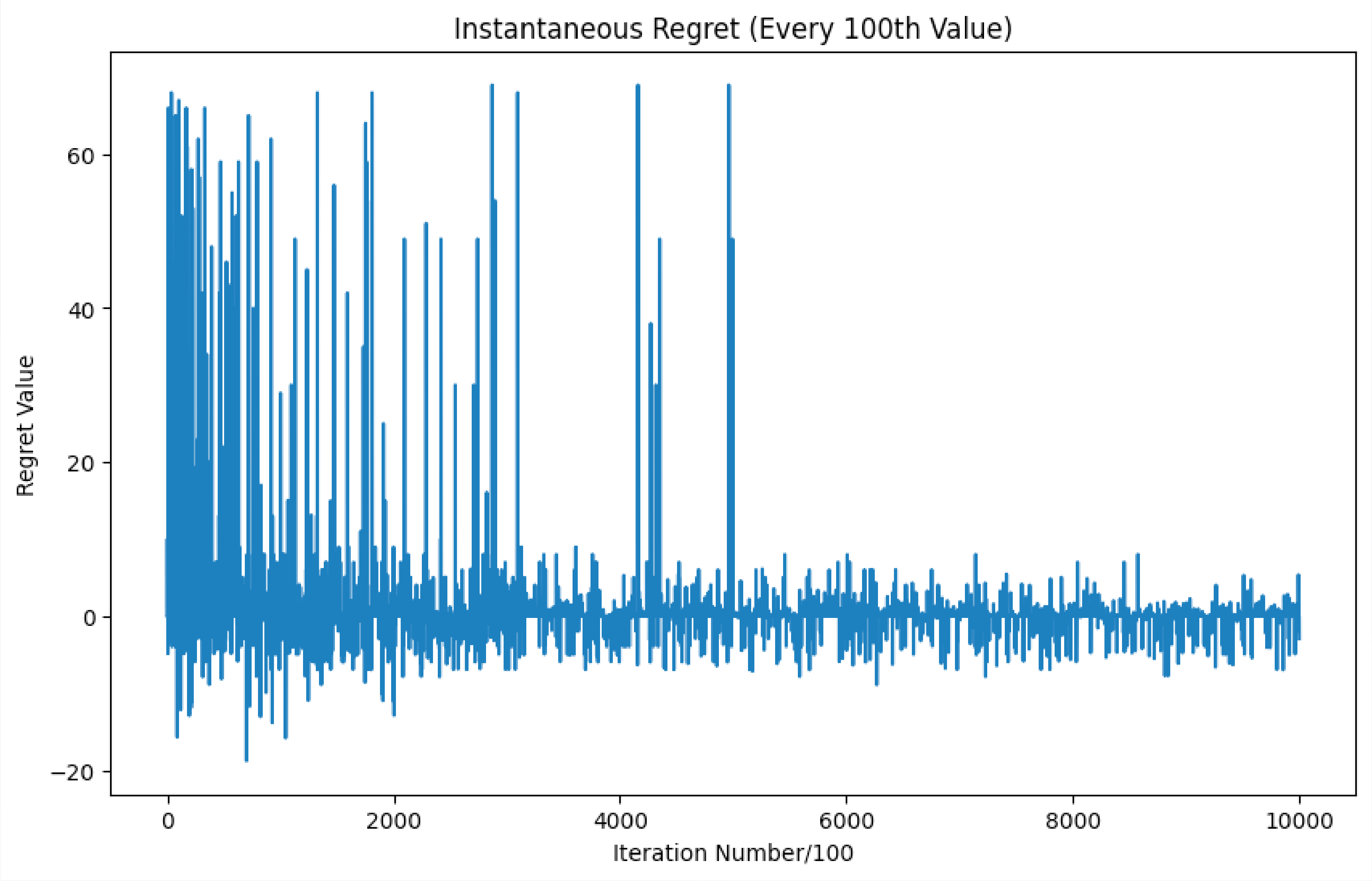


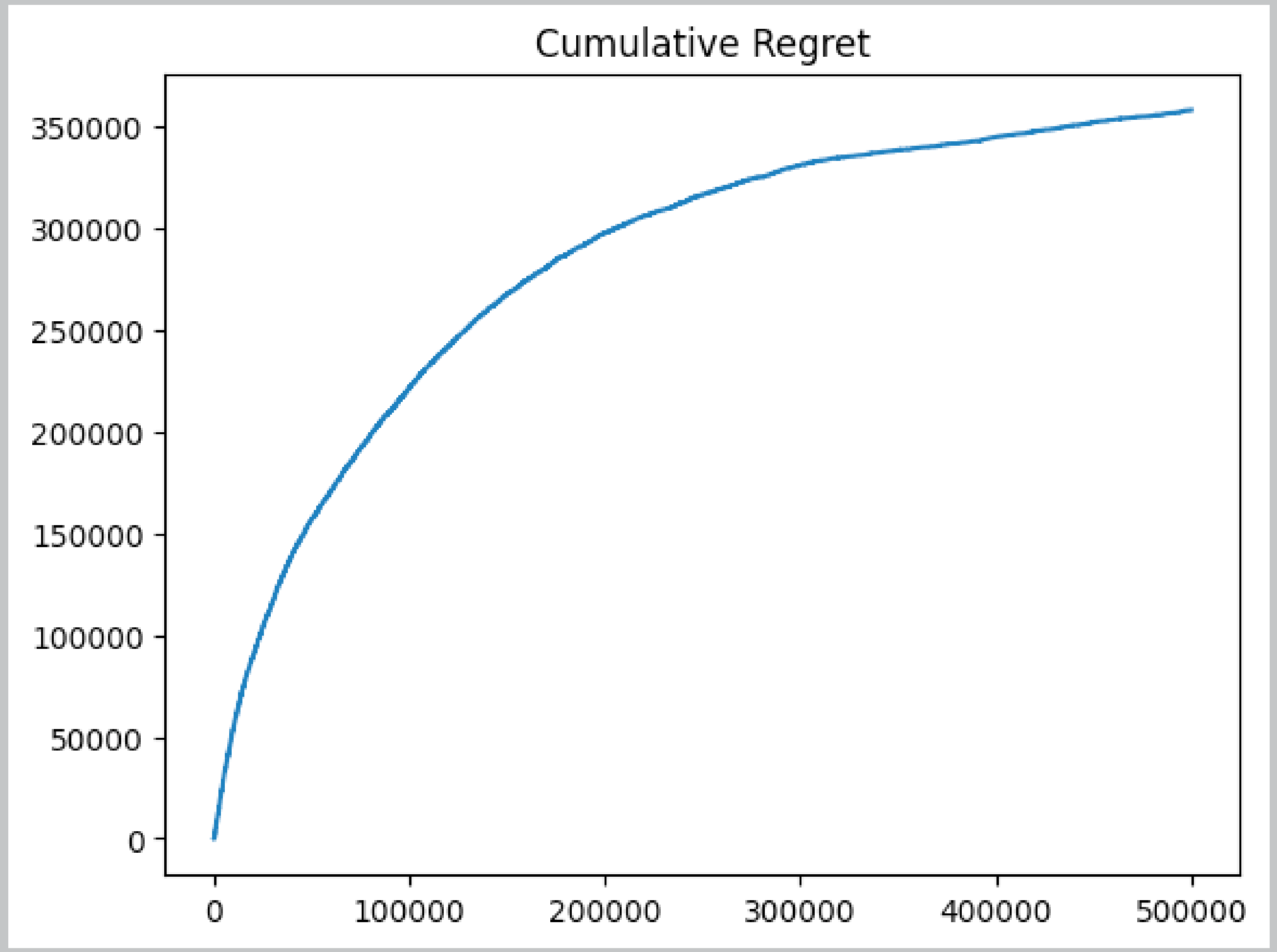
MONTE CARLO

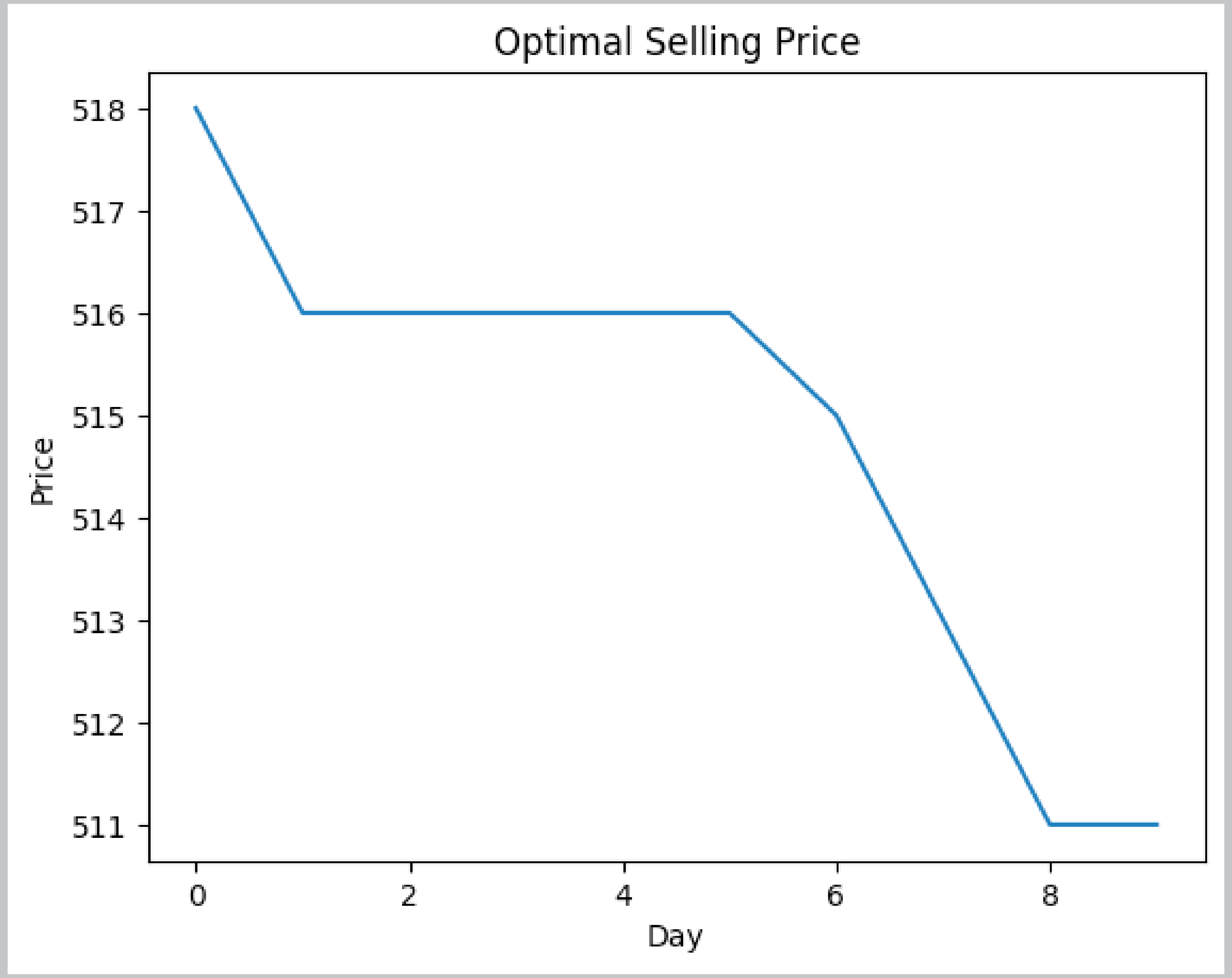
Updated Values of Regret:

Results Obtained:

(strike price = 510, start price = 500, drift = 8 and T = 10)







Q-LEARNING

Pseudo Code followed:

Algorithm 1 Finite Horizon Q-Learning

Notation:

$Q_n^m(i,a)$: Q-value for state i, action a, stage n, recursion m.

$a(m)$: step-size at recursion index m

$Q_N(i,a)$: Q-value for state i and action a at terminal stage (N).

$g_n(i,a,j)$: Single stage cost for stage n where current state is i, action is a and next state is j.

$g_N(i)$: Terminal reward at the N^{th} stage when terminal state is i.

$A(j)$: Set of feasible actions in state j.

$\eta(i,a)$: Sampling function taking input (i,a) as state-action pair that returns the next state.

Input: Samples of the form $\big(i$ (current state), a (action), g (cost), j (next state)$\big)$.

Output: Updated Q-value $Q_n^{m+1}(i,a)$ estimated after m iterations of the algorithm.

Initialization: $Q_n^0(i,a) = 0,\ \forall(i,a), n = 0,\dots,N-1,$ and $Q_N^0(i,a) = g_N(i), \forall(i,a)$

procedure Finite Horizon Q-Learning:

$a(m) = \left\lceil \frac{1}{(m+1)/10} \right\rceil$

$j = \eta(i,a)$ (from samples)

$Q_n^{m+1}(i,a) = \big(1 - a(m)\big)\Big(Q_n^m(i,a)\Big) + a(m)$

$\times\Big(g_n(i,a) + \min_{b \in A(j)} Q_{n+1}^m(j,b)\Big), n = 0, 1, \dots, N-1,$

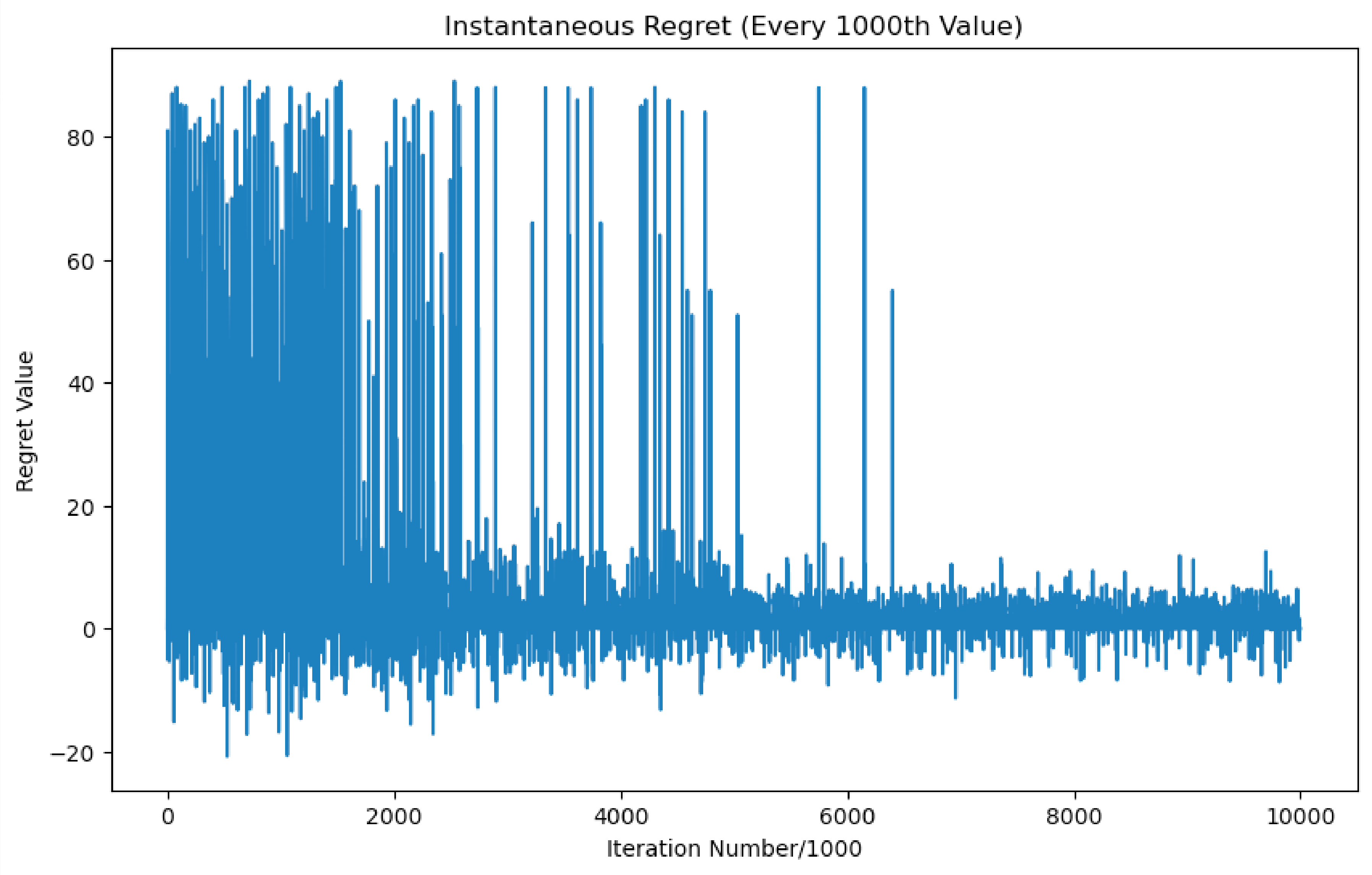
$Q_N^{m+1}(i,a) = g_N(i),\ \forall(i,a)$ tuples.

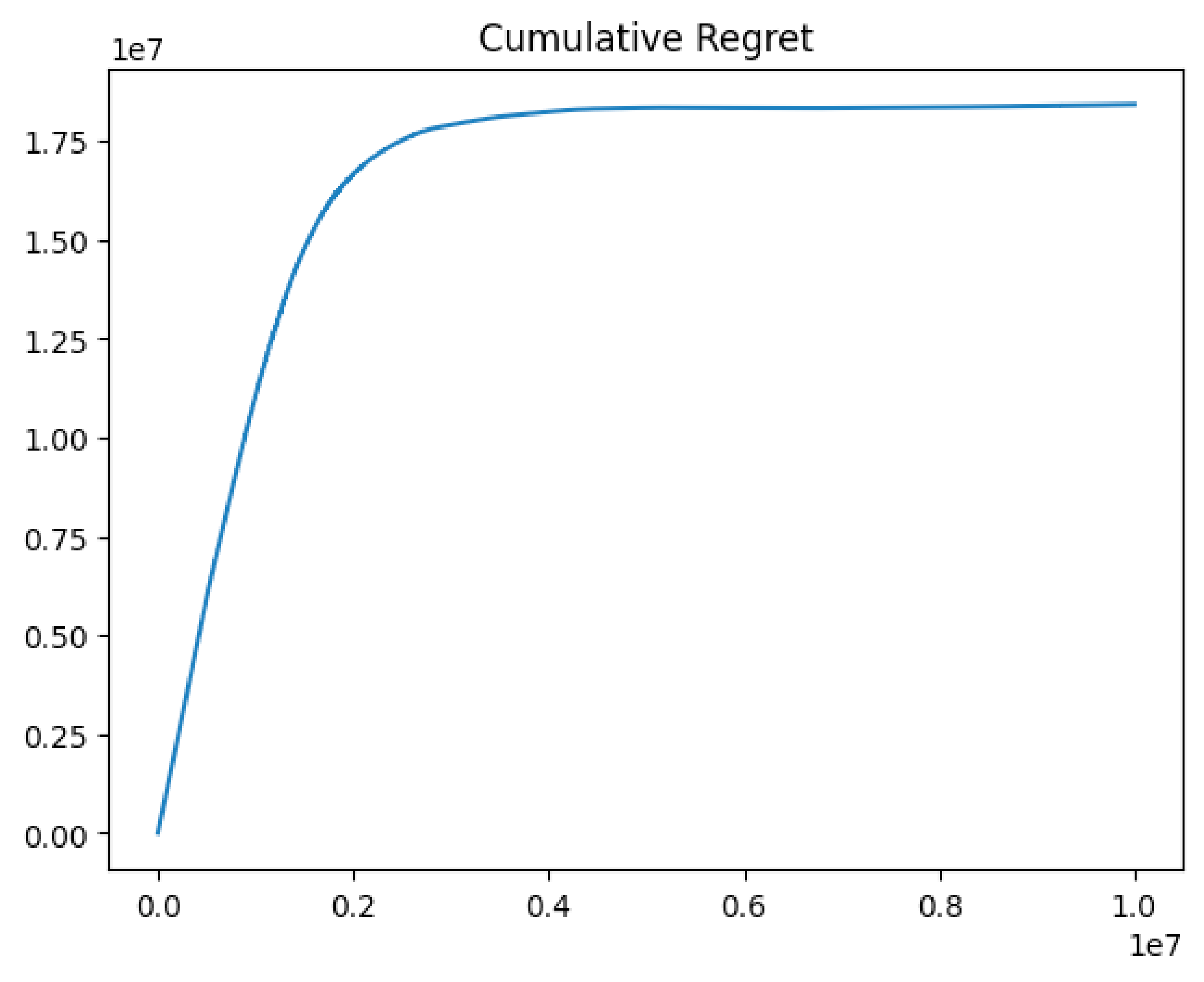
return $Q_n^{m+1}(i,a)$

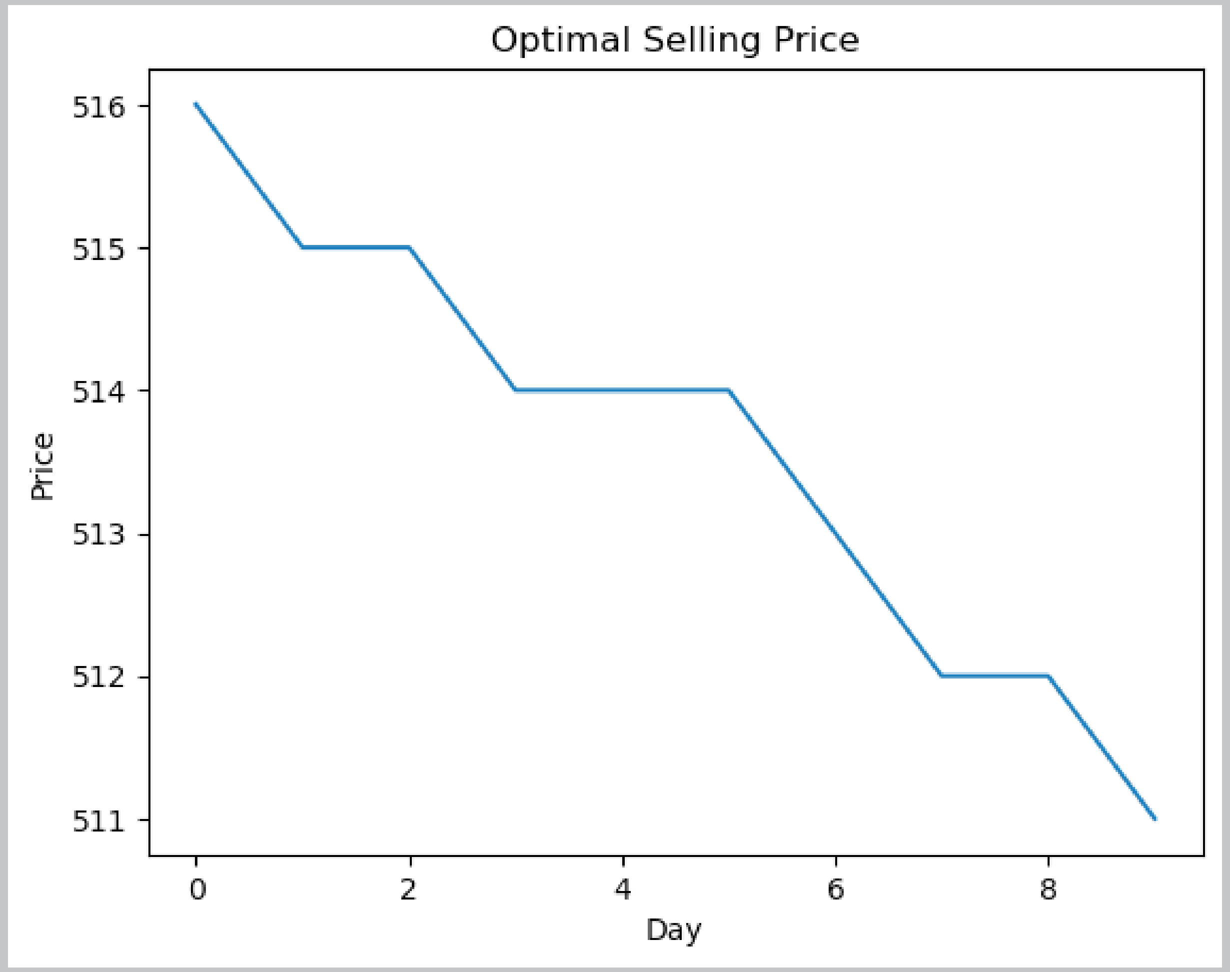
Q-LEARNING

Results Obtained:

(strike price = 510, start price = 500, drift = 5 and T = 10)





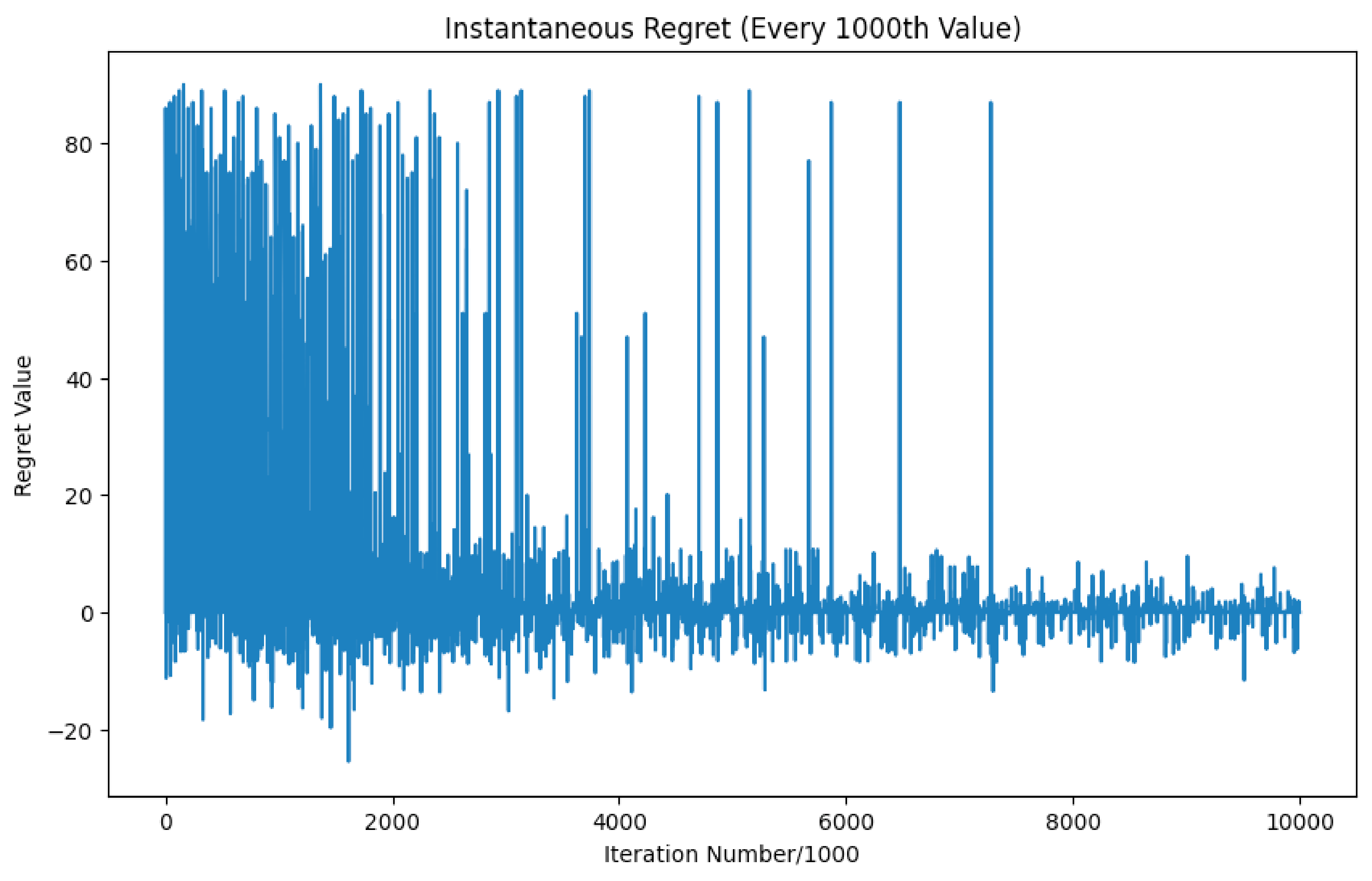


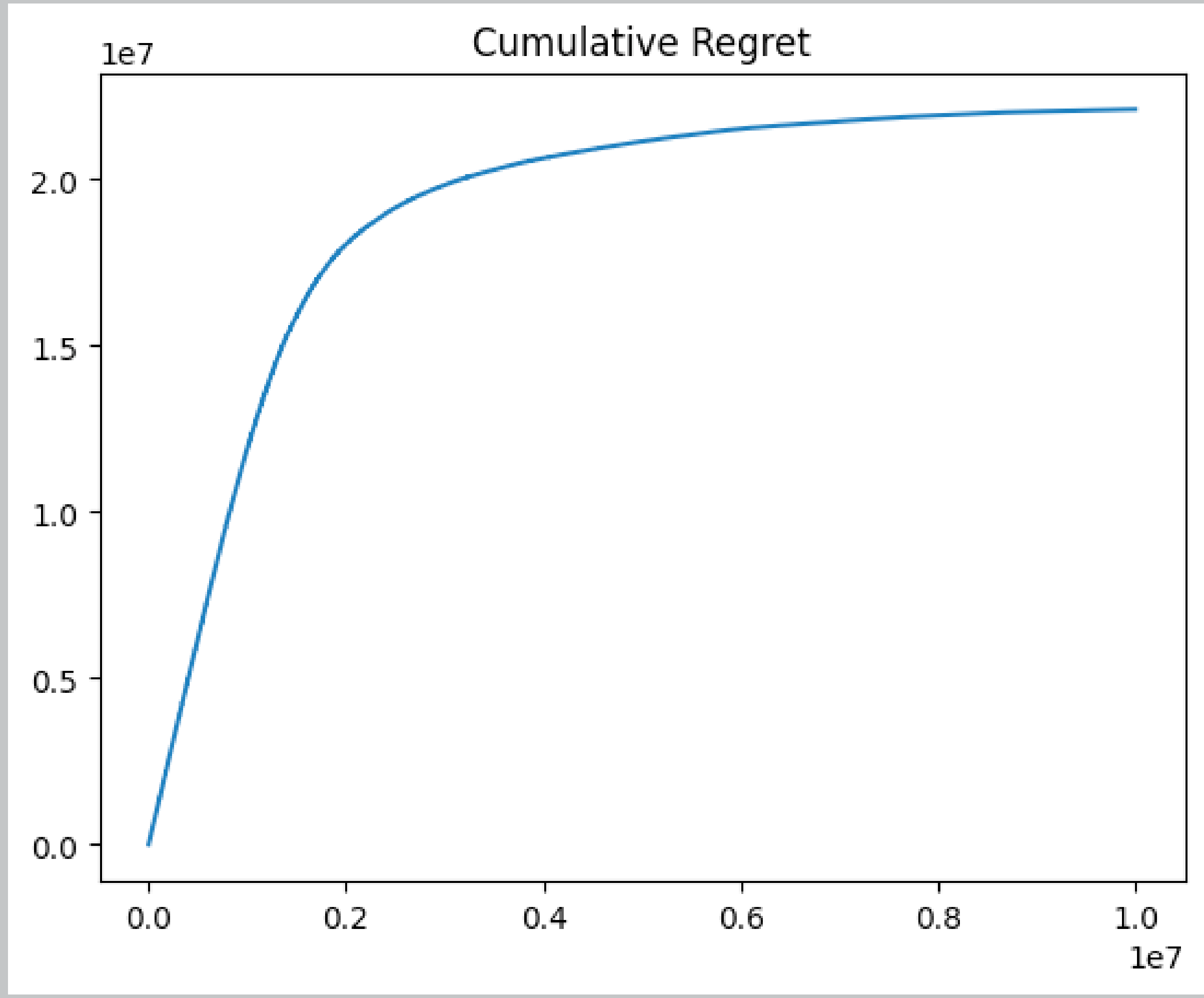
Q-LEARNING

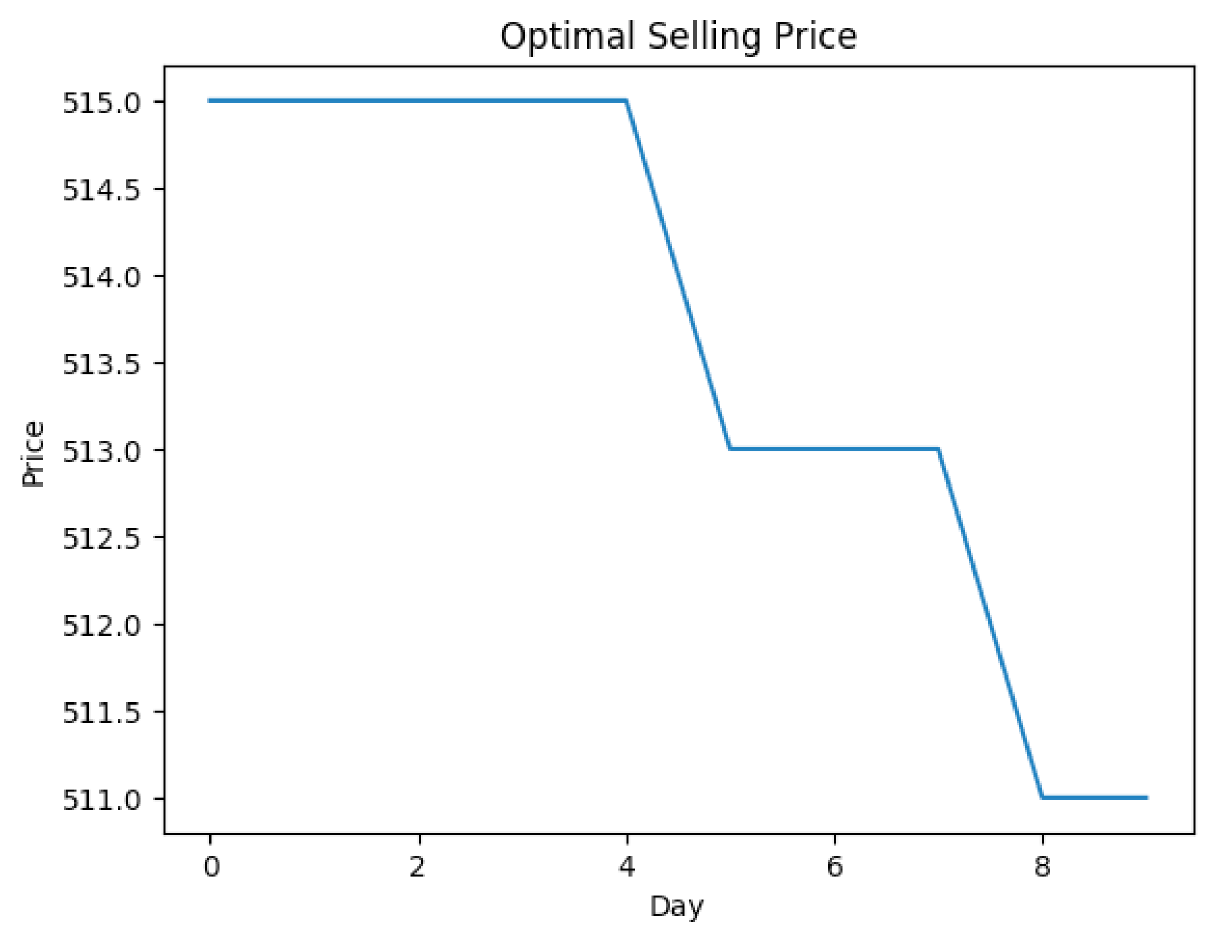
Updated Values of Regret:

Results Obtained:

(strike price = 510, start price = 500, drift = 10 and T = 10)







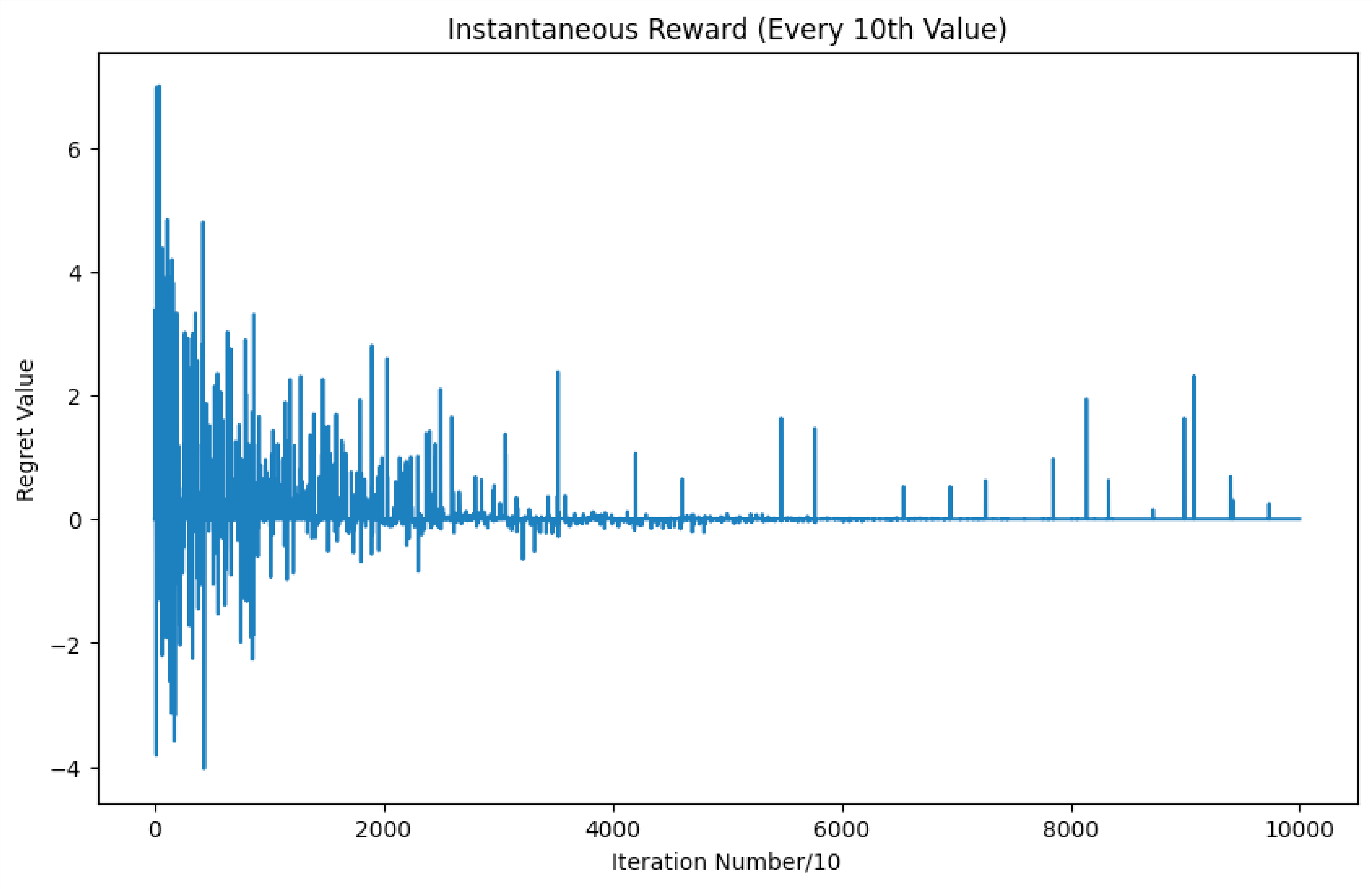
TD CONTROL
USING SARSA(λ)

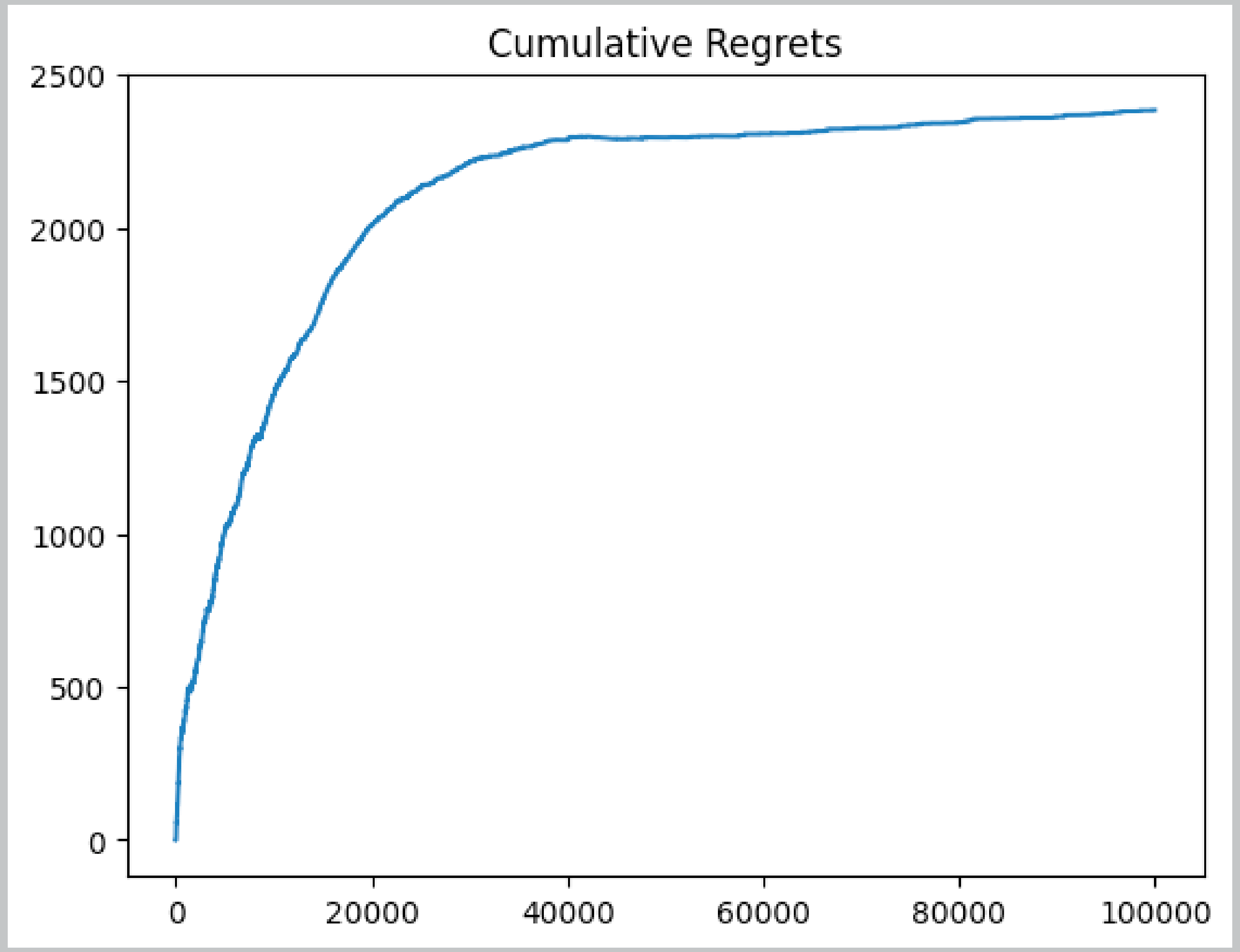
Pseudo Code followed:

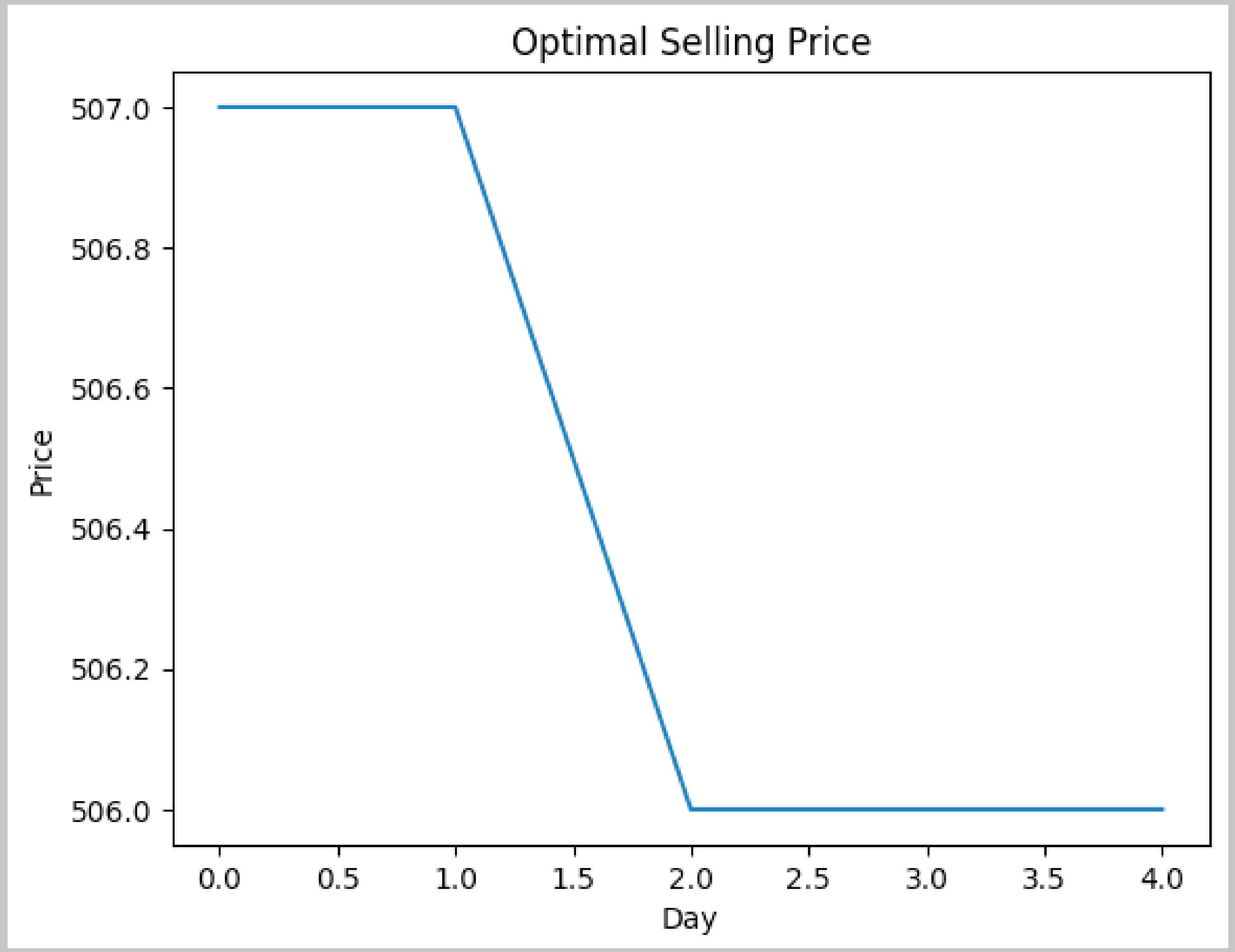
Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Repeat (for each episode):
    $E(s,a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
    Initialize S, A
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$
        $E(S,A) \leftarrow E(S,A) + 1$
        For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
            $Q(s,a) \leftarrow Q(s,a) + \alpha \delta E(s,a)$
            $E(s,a) \leftarrow \gamma \lambda E(s,a)$
        $S \leftarrow S'$; $A \leftarrow A'$
    until S is terminal

TD CONTROL
USING SARSA(λ)

Results Obtained:
(strike price = 505, start price = 500, drift = 5 and T = 5)





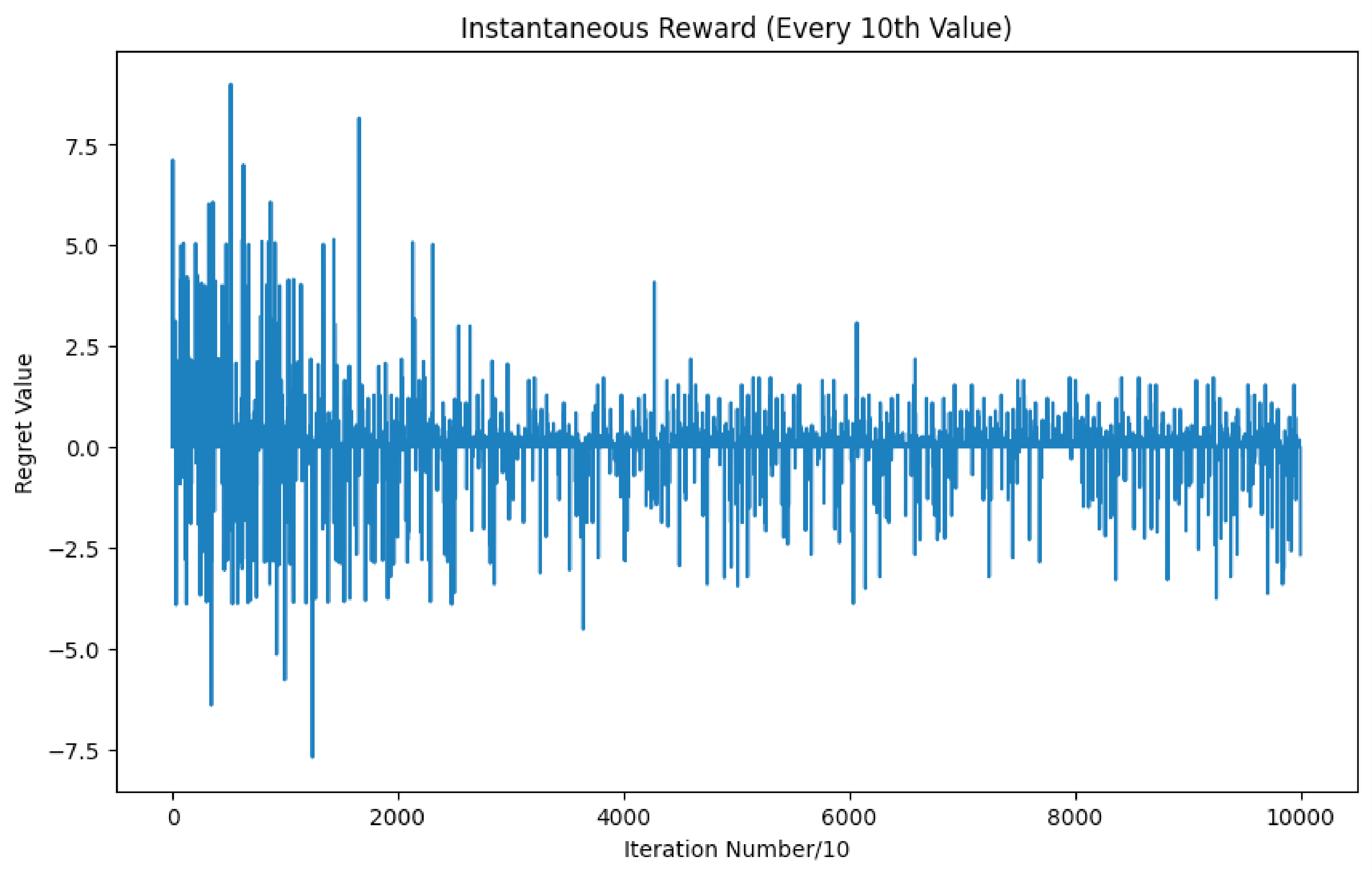


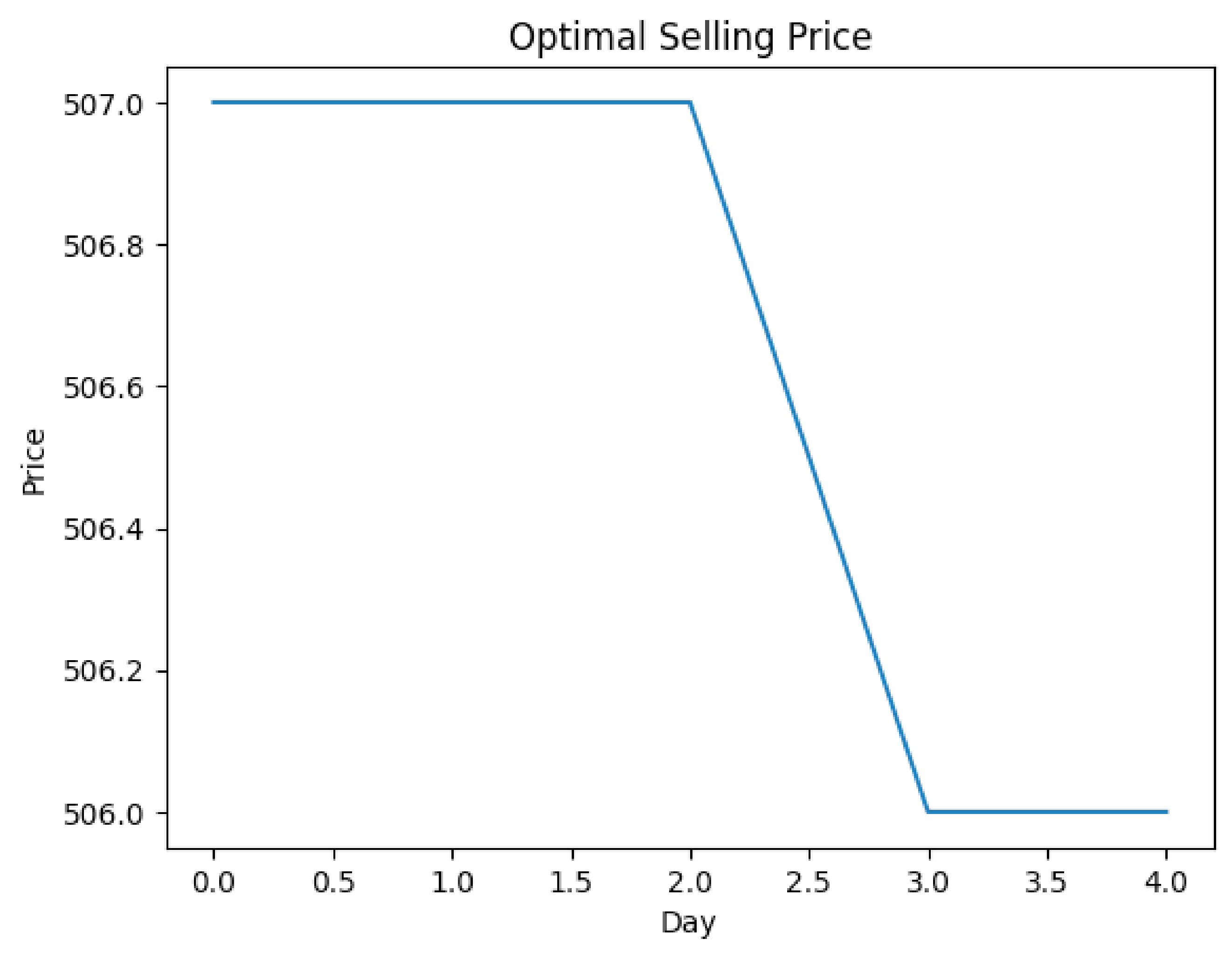
TD CONTROL
USING SARSA(λ)

Updated Values of Regret:

Results Obtained:

(strike price = 505, start price = 500, drift = 5 and T = 5)





Conclusion

An optimal policy will follow the below given theorem :

There exist numbers $s_1 \geqslant s_2 \geqslant \cdots \geqslant s_T$ such that it is optimal to exercise an option at time t iff $x_t \geqslant s_t$. Hence, the optimal strategy is of threshold type.

We see an identical trend with the RL algorithms we implemented, confirming the results' correctness.

TEAM MEMBERS

PRATHAM PRIYANK THAKKAR (2021101077)
NIPUN TULSIAN (2021101055)
VYOM GOYAL (2021101099)
PRAYUSH RATHORE (2020114009)
YASH AGARWAL (2020114005)

Thank You!