

# APPLIED PROBABILITY FOR MATHEMATICAL FINANCE

## PROJECT 1 REPORT

A. XINYI SHEN AND B. SHAOFENG KANG

**ABSTRACT.** This report focuses on pricing American put option through a variation of the Cox, Ross, Rubenstein (CRR) model, by deriving branching probabilities under risk-neutral measure, finding exercise boundary, simulating stock paths for not exercised option and early exercised option, computing hedging strategy, looking into P&L and stopping time distributions under different value of  $\sigma$  and  $r$ .

### 1. INTRODUCTION

Suppose that an asset price process  $S = (S_{t_k})_{k \in \{0,1,\dots,N\}}$ , where  $t_k = k\Delta t$  and  $\Delta t = \frac{T}{N}$ , for a fixed  $N$ , are given by the stochastic dynamics

$$S_{t_k} = S_{t_{k-1}} e^{r\Delta t + \sigma\sqrt{\Delta t}\epsilon_k},$$

where  $\epsilon_k$  are i.i.d random variables with  $\epsilon_k \in \{+1, -1\}$  and

$$\mathbb{P}(\epsilon_k = \pm 1) = \frac{1}{2} \left( 1 \pm \frac{(\mu - r) - \frac{1}{2}\sigma^2}{\sigma} \sqrt{\Delta t} \right).$$

Here,  $r \geq 0$  and  $\sigma > 0$  are constants. This is a variation of the Cox, Ross, Rubenstein (CRR) model.

Moreover, let  $B = (B_{t_k})_{k \in \{0,1,\dots,N\}}$  denote the bank account with  $B_t = e^{rt}$ .

### 2. METHODOLOGY

**Derive Branching Probabilities under Risk-Neutral Measure:** We are deriving the branching probabilities  $\mathbb{Q}(\epsilon_k = \pm 1)$  and  $\mathbb{Q}$ , where  $\mathbb{Q}$  refers to the martingale measure induced by using the bank account  $B$  as the numéraire.

Under the risk-neutral measure  $\mathbb{Q}$ , the stochastic dynamics of the asset price under  $\mathbb{Q}$  should be

$$S_{t_k} = S_{t_{k-1}} e^{r\Delta t + \sigma\sqrt{\Delta t}\epsilon_k},$$

where  $\epsilon_k$  are i.i.d random variables with  $\epsilon_k \in \{+1, -1\}$ .

To maintain the martingale property under  $\mathbb{Q}$ , the expected growth rate of the asset must match the risk-free rate  $r$ , that is, the discounted asset price  $S_{t_k} e^{-rt_k}$  should be a martingale under  $\mathbb{Q}$ .

By definition of martingale, for all  $t_k, t_{k-1}, k \in \{0,1,\dots,N\}$ , we have

$$\mathbb{E} [S_{t_k} | S_{t_{k-1}}] = S_{t_{k-1}} e^{r\Delta t}$$

Expanding the conditional expectation, we get

$$\mathbb{Q}(\epsilon_k = +1) S_{t_{k-1}} e^{r\Delta t + \sigma\sqrt{\Delta t}} + \mathbb{Q}(\epsilon_k = -1) S_{t_{k-1}} e^{r\Delta t - \sigma\sqrt{\Delta t}} = S_{t_{k-1}} e^{r\Delta t}$$

Let the risk-neutral probabilities be  $\mathbb{Q}(\epsilon_k = +1) = \tilde{p}$  and  $\mathbb{Q}(\epsilon_k = -1) = \tilde{q} = 1 - \tilde{p}$ . Substituting  $\tilde{p}$  and  $\tilde{q} = 1 - \tilde{p}$  into the above equation and dividing  $S_{t_{k-1}}$  on both sides, we have

$$\begin{aligned}
\tilde{p}e^{r\Delta t + \sigma\sqrt{\Delta t}} + (1 - \tilde{p})e^{r\Delta t - \sigma\sqrt{\Delta t}} &= e^{r\Delta t} \\
\tilde{p}e^{\sigma\sqrt{\Delta t}} + (1 - \tilde{p})e^{-\sigma\sqrt{\Delta t}} &= 1 \\
\tilde{p}e^{\sigma\sqrt{\Delta t}} + e^{-\sigma\sqrt{\Delta t}} - \tilde{p}e^{-\sigma\sqrt{\Delta t}} &= 1 \\
\tilde{p}(e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}}) &= 1 - e^{-\sigma\sqrt{\Delta t}} \\
\implies \tilde{p} &= \frac{1 - e^{-\sigma\sqrt{\Delta t}}}{e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}}}
\end{aligned}$$

Then we solve for  $\tilde{q} = 1 - \tilde{p}$ :

$$\begin{aligned}
\tilde{q} = 1 - \tilde{p} &= 1 - \frac{1 - e^{-\sigma\sqrt{\Delta t}}}{e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}}} \\
&= \frac{e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}} - (1 - e^{-\sigma\sqrt{\Delta t}})}{e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}}} \\
&= \frac{e^{\sigma\sqrt{\Delta t}} - 1}{e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}}}
\end{aligned}$$

Therefore, the branching probabilities under the risk-neutral measure  $\mathbb{Q}$  are

$$\mathbb{Q}(\epsilon_k = +1) = \tilde{p} = \frac{1 - e^{-\sigma\sqrt{\Delta t}}}{e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}}} \quad \text{and} \quad \mathbb{Q}(\epsilon_k = -1) = \tilde{q} = \frac{e^{\sigma\sqrt{\Delta t}} - 1}{e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}}}.$$

### 3. RESULTS

**Evaluate American Put Option:** In this part, we evaluate an American put option. Assume that  $T = 1$ ,  $S_0 = 10$ ,  $\mu = 5\%$ ,  $\sigma = 20\%$ , and the risk-free rate  $r = 2\%$ . Use  $N = 5000$  and a strike  $K = 10$ . We set `np.random.seed(0)` to ensure the reproducibility of randomness in this whole section.

**3.1. Part (a).** Implement the valuation and exercise boundary of the American put option with  $B$  as the numéraire.

**3.1.1. i.** Plot the exercise boundary as a function of  $t$ .

To plot the exercise boundary in Python, we first set the parameters. The total time  $T = 1$  is divided into  $N = 5000$  intervals. The time step size is calculated as  $dt = \frac{T}{N}$ . A time grid `time.steps` is generated to represent the time evolution between  $t = 0$  and  $t = T$ . Since the stochastic dynamic for the asset price process is  $S_{t_k} = S_{t_{k-1}}e^{r\Delta t + \sigma\sqrt{\Delta t}\epsilon_k}$  where  $\epsilon_k \in \{+1, -1\}$ , then the up factor is  $u = \frac{S_{t_k}}{S_{t_{k-1}}} = e^{r\Delta t + \sigma\sqrt{\Delta t}}$ , and the down factor is  $d = \frac{S_{t_k}}{S_{t_{k-1}}} = e^{r\Delta t - \sigma\sqrt{\Delta t}}$ . We will use the risk-neutral branching probabilities we derived earlier to discount option value, which is  $\mathbb{Q}(\epsilon_k = +1) = \frac{e^{r\Delta t} - d}{u - d} = \frac{1 - e^{-\sigma\sqrt{\Delta t}}}{e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}}}$ .

```
def plot_exercise_boundary(sigma, r, ax, T=1, S0=10, K=10, mu=0.05, N=5000):

    # Time step
    dt = T / N
    time_steps = np.linspace(0, 1, N + 1)

    # Up and down factors (u and d)
    u = np.exp(r * dt + sigma * np.sqrt(dt))
    d = np.exp(r * dt - sigma * np.sqrt(dt))

    # Risk-neutral up-branch probability
    p_risk_neutral = (1 - np.exp(-sigma * np.sqrt(dt))) / (
        np.exp(sigma * np.sqrt(dt)) - np.exp(-sigma * np.sqrt(dt)))
```

Then we generate a binomial stock price tree. At each node, the stock moves up or down by multiplying up or down factors. We also generate option values by using backward induction on the discounted expected payoff of the American put option. The discount factor,  $e^{-rt}$ , accounts for the time value of money. We use the risk-neutral probability for the expectation when discounting back the put option. The intrinsic value of option, which is  $\max(K - S_t, 0)$  is compared with the continuation value (expected payoff under the risk-neutral measure) to ensure the optimal exercise of the American put option. We also calculate the option premium, i.e., initial cost, for later use.

```
# Binomial tree for stock prices
stock_tree[0, 0] = S0
for i in range(1, N+1):
    stock_tree[i, 0] = stock_tree[i-1, 0] * d
    for j in range(1, i+1):
        stock_tree[i, j] = stock_tree[i-1, j-1] * u

# Backward induction using risk-neutral measure to calculate option values
discount_factor = np.exp(-r * dt)
option_value[N, :] = np.maximum(K - stock_tree[N, :], 0)
for i in range(N - 1, -1, -1):
    option_value[i, :i + 1] = discount_factor * (
        p_risk_neutral * option_value[i + 1, 1:i + 2]
        + (1 - p_risk_neutral) * option_value[i + 1, :i + 1])
    # Intrinsic value for early exercise
    intrinsic_value = K - stock_tree[i, :i + 1]
    # Take max between intrinsic value and continuation value
    option_value[i, :i + 1] = np.maximum(option_value[i, :i + 1], intrinsic_value)
premium = option_value[0, 0]
```

The exercise boundary is determined by finding the stock price at which the intrinsic value equals the option value and is stored as an array of time and stock prices.

```
# Exercise boundary calculation
exercise_boundary = []
```

```

for i in range(len(stock_tree)):
    intrinsic_value = K - stock_tree[i, :i+1]
    optimal_indices = np.where(option_value[i, :i+1] == intrinsic_value)[0]
    if len(optimal_indices) > 0:
        boundary_price = stock_tree[i, optimal_indices[-1]]
    else:
        boundary_price = np.nan
    exercise_boundary.append([time_steps[i], boundary_price])
exercise_boundary = np.array(exercise_boundary)

```

We return the exercise boundary, boundary time, boundary prices, and premium for later use and plot the exercise boundary as a function of  $t$  in Figure 1.

```

# Plot the exercise boundary
ax.plot(exercise_boundary[:, 0], exercise_boundary[:, 1],
        label="Exercise Boundary", color="steelblue")
ax.set_title(f"American Put Option Exercise Boundary ( $\sigma=\{\sigma\}$ ,  $r=\{r\}$ )")
ax.set_xlabel("Time")
ax.set_ylabel("Stock Price")
ax.legend()

valid_indices = ~np.isnan(exercise_boundary[:, 1])
boundary_time = (np.arange(N + 1) * dt)[valid_indices]
boundary_prices = exercise_boundary[:, 1][valid_indices]

return exercise_boundary, boundary_time, boundary_prices, premium

```

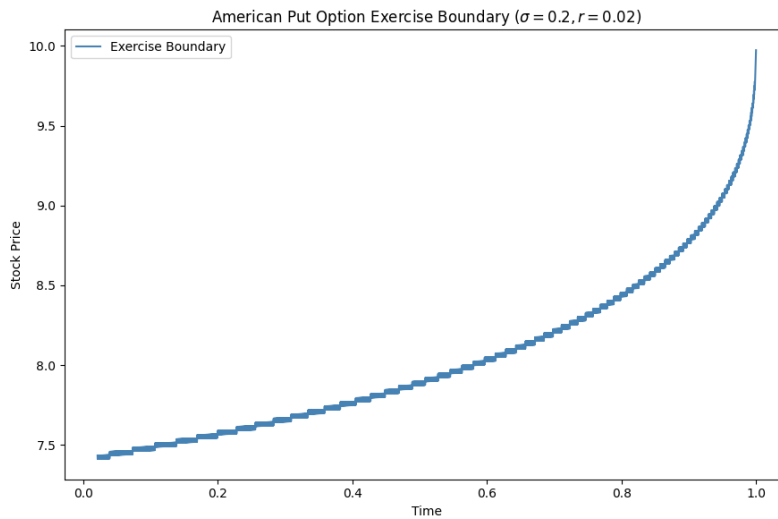


FIGURE 1. Plot the exercise boundary of American put option with  $B = e^{rt}$  as the numéraire as a function of  $t$  when  $\sigma = 20\%$  and  $r = 2\%$ .

3.1.2. *ii.* Generate two sample paths where in sample path 1) the option is exercised early (say around  $t = \frac{1}{2}$ ), 2) the option is not exercised.

To simulate stock paths, we incorporate randomness in each time step, where the stock price either goes up or down based on the physical-measure probability,  $\mathbb{P}(\epsilon_k = +1) = \frac{1}{2} \left( 1 + \frac{(\mu-r)-\frac{1}{2}\sigma^2}{\sigma} \sqrt{\Delta t} \right)$ . The `generate_path` function creates a single stock price path.

```
def generate_path(S0, N, u, d, p):
    path = np.zeros(N + 1)
    path[0] = S0 # Initial stock price
    for j in range(1, N + 1):
        if np.random.rand() < p:
            path[j] = path[j - 1] * u # Up move
        else:
            path[j] = path[j - 1] * d # Down move
    return path
```

The function `generate_paths_with_exercise_boundary` generates two paths:

- Path 1: A path that remains above the exercise boundary at all times and thus is not exercised until maturity.
- Path 2: A path that touches the exercise boundary at around  $t = \frac{1}{2}$  (we allow a 0.01 error) and exercises the option at that time point.

```
def generate_paths_with_exercise_boundary(sigma, r, ax, T=1, S0=10, K=10, mu=0.05, N=5000):
    ...
    # Generate Path 1: No early exercise (stays above the boundary)
    path_no_exercise = generate_path(S0, N, u, d, p_physical)
    for i in range(1, N + 1):
        if path_no_exercise[i] < boundary[i, 1]: # Ensure path stays above the boundary
            path_no_exercise[i] = path_no_exercise[i - 1]
    # Generate Path 2: Exercise early at t = 0.5
    path_early_exercise = None
    while path_early_exercise is None:
        candidate_path = generate_path(S0, N_hit, u, d, p_physical)
        if math.isclose(candidate_path[N_hit], boundary[N_hit, 1], rel_tol=0.01):
            path_early_exercise = candidate_path
```

We return the not exercised path and the early exercised path for later use and plot the simulated two stock paths in Figure 2.

```
ax.plot(time_steps, path_no_exercise, label="No Early Exercise", color="seagreen")
ax.plot(np.linspace(0, T_hit, len(path_early_exercise)), path_early_exercise,
        label="Early Exercise at t = 0.5", color="firebrick")
ax.set_title(f"Simulated Stock Paths with Exercise Boundary ( $\sigma$ = $\{sigma\}$ ,  $r$ = $\{r\}$ )")
ax.set_xlabel("Time")
ax.set_ylabel("Stock Price")
ax.legend()
```

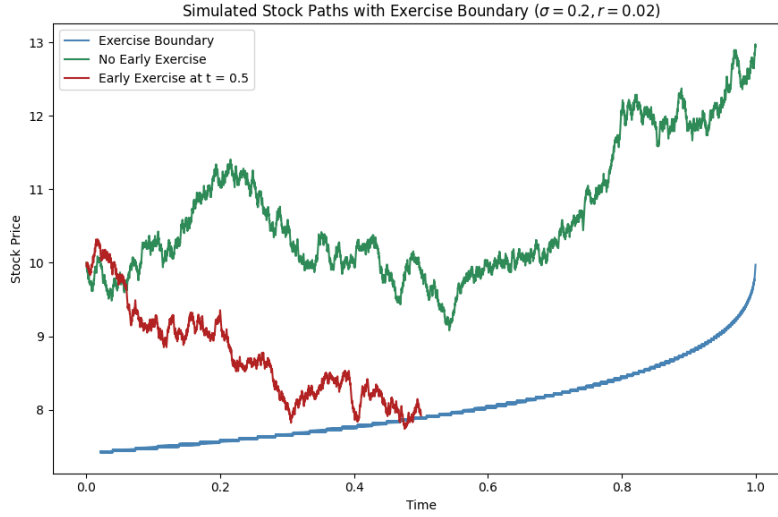


FIGURE 2. Simulated stock paths, one is never exercised until maturity, another is early exercised around  $t = \frac{1}{2}$  when  $\sigma = 20\%$  and  $r = 2\%$ .

3.1.3. *iii.* Along the two sample paths above, plot the hedging strategy that a trader would use to hedge the option as a function of time.

Delta hedging is a dynamic strategy to maintain a portfolio's sensitivity to small changes in the underlying asset's price. To compute the delta for the two simulated stock paths, we first create an array of time steps corresponding to each point in paths and locate the closest stock tree values away from the simulated stock paths at each path step. Use the found time steps and indices to populate stock prices for up and down moves as well as option values for up and down moves. Now, we can calculate the delta values,  $\Delta = \frac{V_{up} - V_{down}}{S_{up} - S_{down}}$ , by computing the changes in option values divided by changes in stock prices between up and down moves. Considering edge case, we ensure that delta is set to 0 when up moving stock prices and down moving stock prices are the same.

```
def delta_hedging(path, V=option_value, S=stock_tree):
    time_steps = np.arange(len(path))
    stock_indices = np.argmin(np.abs(S[time_steps, :len(path)] - path[:, None]), axis=1)
    S_up = S[time_steps[1:], stock_indices[:len(path) - 1]]
    S_down = S[time_steps[1:], stock_indices[:len(path) - 1] + 1]
    V_up = V[time_steps[1:], stock_indices[:len(path) - 1]]
    V_down = V[time_steps[1:], stock_indices[:len(path) - 1] + 1]
    delta = np.where(S_up != S_down, (V_up - V_down) / (S_up - S_down), 0)
    return delta
```

Applying the `delta_hedging` function to the two stock paths we simulated before, we plot the hedging strategy for both the no early exercise stock path and the early exercise at  $t = \frac{1}{2}$  stock path in Figure 3.

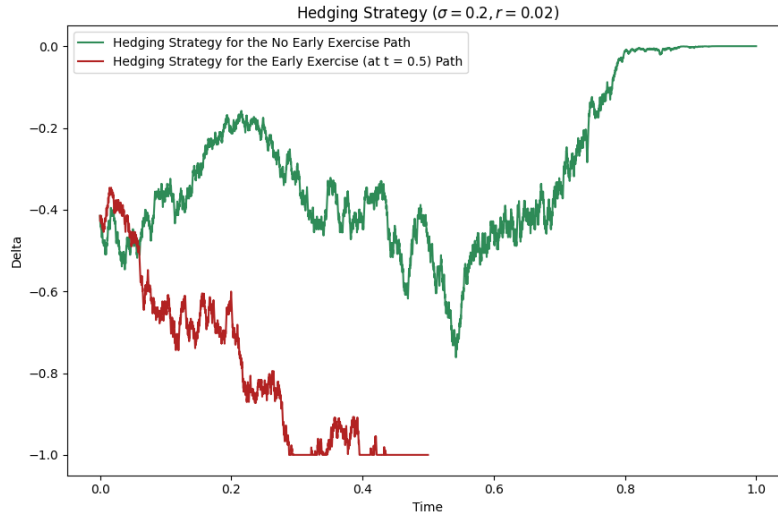


FIGURE 3. Delta-hedging strategy along the two simulated stock paths as a function of  $t$  when  $\sigma = 20\%$  and  $r = 2\%$ .

3.1.4. *iv.* Illustrate how the results in i, ii, and iii vary as volatility and risk-free rate change (pair-wise). [For example,  $\sigma = 10\%, 20\%, 30\%$  and  $r = 0\%, 2\%, 4\%$ ]

Exercise Boundary for Different Combinations of  $\sigma$  and  $r$

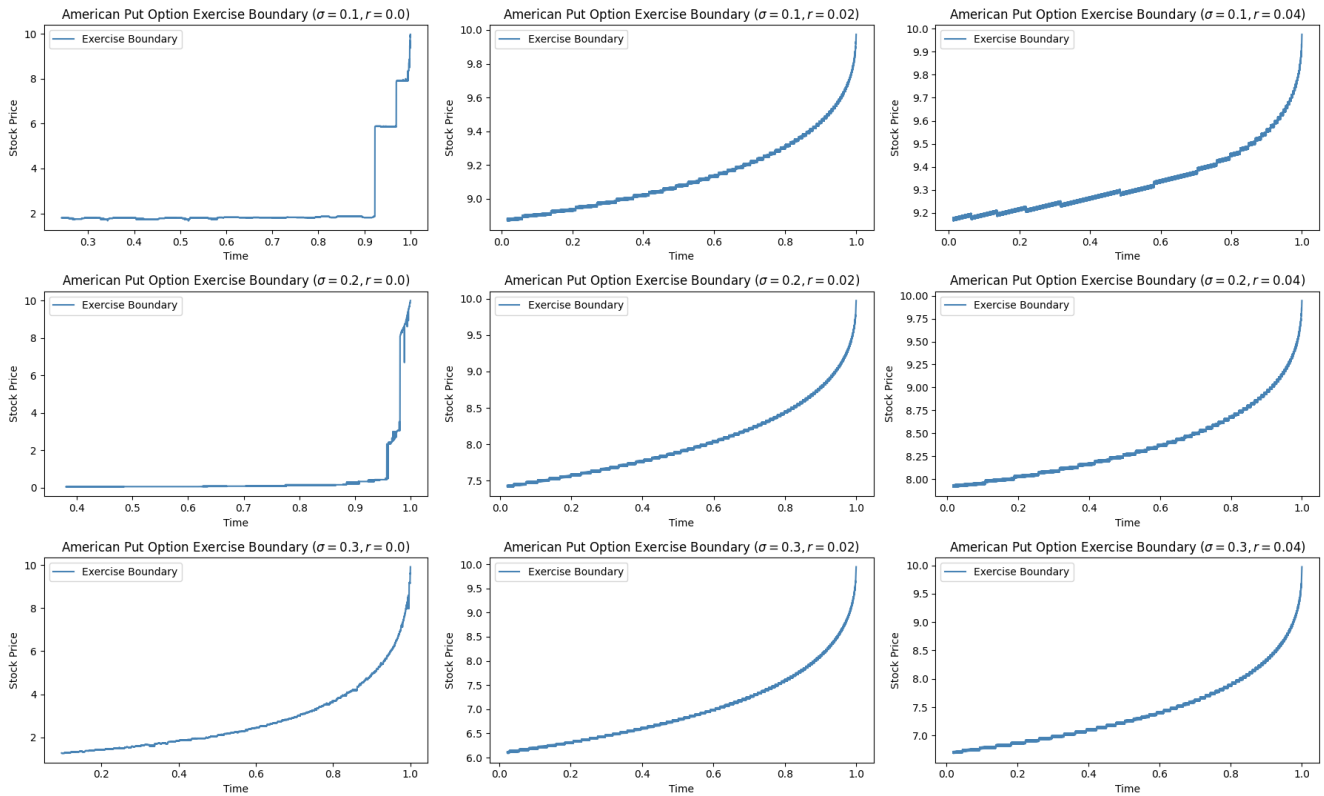


FIGURE 4. Exercise boundaries for  $\sigma = 10\%, 20\%, 30\%$  and  $r = 0\%, 2\%, 4\%$ .

Illustrate how the result in i vary as volatility and risk-free rate change pair-wise:

As shown in Figure 4, higher volatility makes the exercise boundary smoother while lower volatility produces more sudden jumps. Also, higher volatility makes the boundary **lower** at earlier time points. However, all the boundaries converge near the strike price as time approaches maturity. This reflects the typical behaviour of an American put option - where high volatility encourages waiting longer before exercising, but the decision at maturity becomes binary (exercise if  $S_T < K$ , otherwise let it expire).

In contrast, a higher risk-free interest rate makes the boundary **higher** at earlier time points. Holding the put option incurs an opportunity cost because the investor could exercise it now, receive cash, and earn interest on that cash. With higher interest rates, the opportunity cost increases, making it more attractive to exercise earlier.

Simulated Stock Paths for Different Combinations of  $\sigma$  and  $r$

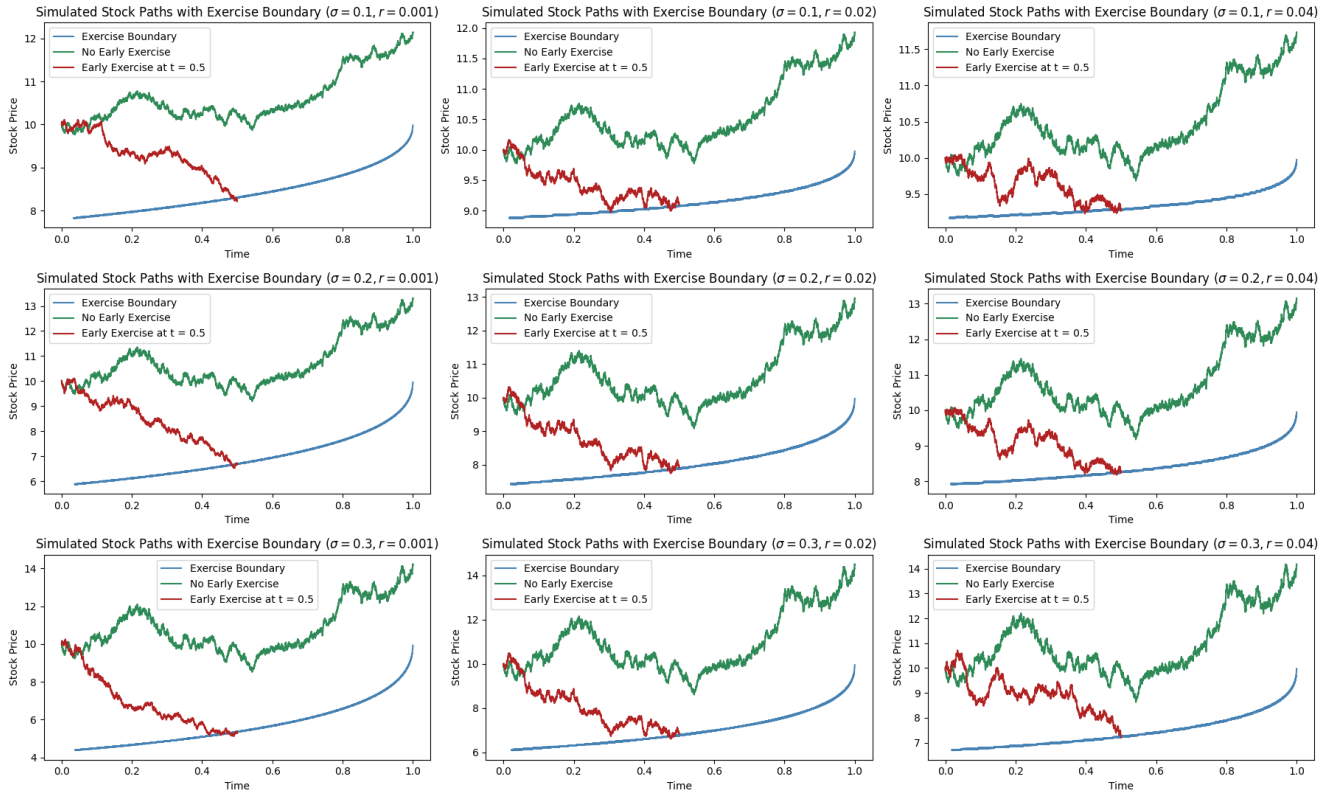


FIGURE 5. Simulated no exercise and early exercise around  $t = \frac{1}{2}$  stock paths for  $\sigma = 10\%, 20\%, 30\%$  and  $r = 0\%, 2\%, 4\%$ .

Illustrate how the result in ii vary as volatility and risk-free rate change pair-wise: Note that when  $r = 0\%$ , the stock paths cannot be simulated, so we choose a extremely small  $r = 0.1\%$  to replace for  $r = 0\%$ .

In Figure 5, higher volatility makes the simulated stock paths fluctuate more widely as you can see that the scale range increases as volatility increases. if volatility increases, the stock price



is more likely to bounce above the boundary in addition to moving below it. This means the likelihood of the stock price crossing and staying below the exercise boundary can decrease with higher volatility. Even if the stock price drops temporarily, it may quickly recover, reducing the chance of early exercise being optimal.

The early exercise stock paths tend to touch the rising exercise boundary earlier as the risk-free interest rate increases. This reflects the fact that the threshold for delaying exercise becomes harder to meet under high interest rates since when the interest rates are higher, the value of receiving the strike price  $K$  sooner increases because cash can be reinvested at a higher rate.

Hedging Strategy for Different Combinations of  $\sigma$  and  $r$

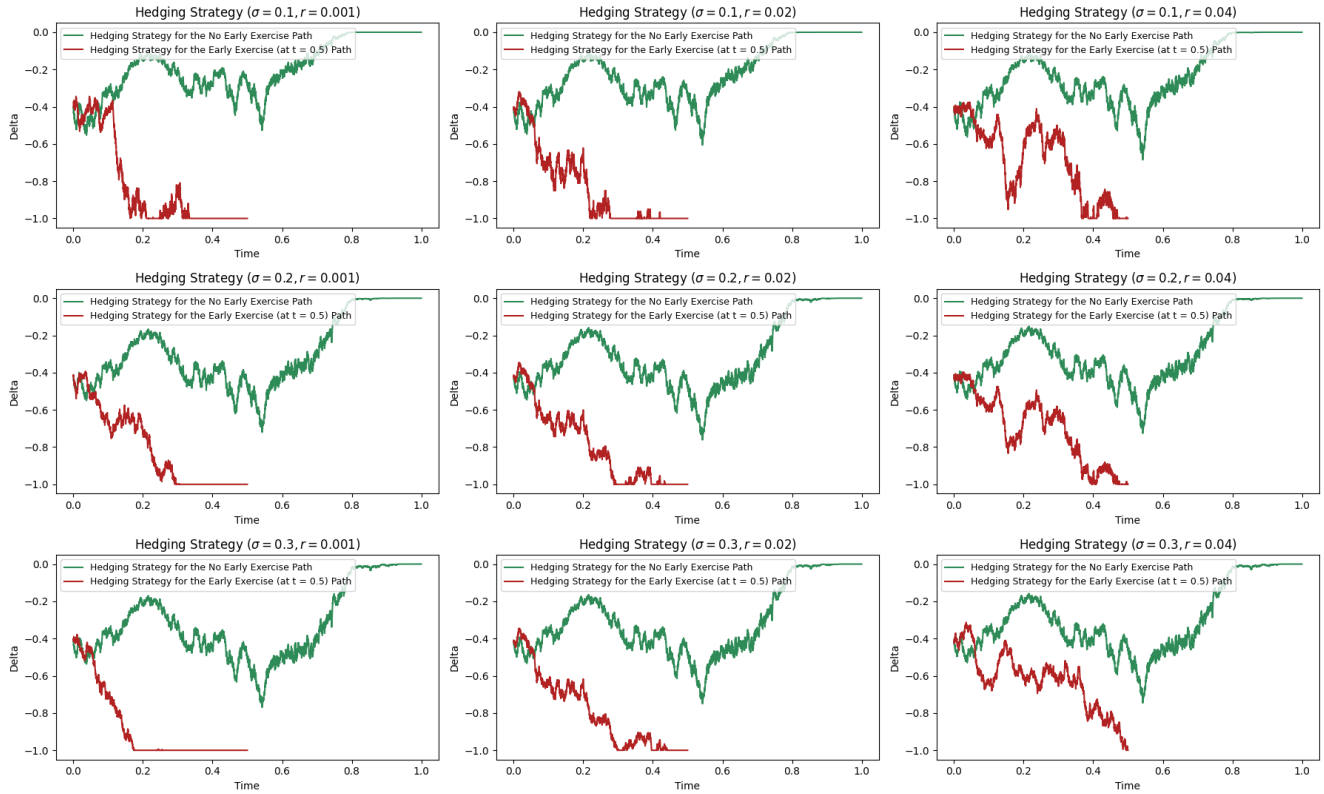


FIGURE 6. Simulated no exercise and early exercise around  $t = \frac{1}{2}$  stock paths for  $\sigma = 10\%, 20\%, 30\%$  and  $r = 0\%, 2\%, 4\%$ .

Illustrate how the result in iii vary as volatility and risk-free rate change pair-wise: Note that when  $r = 0\%$ , the stock paths cannot be simulated, so we choose a extremely small  $r = 0.1\%$  to replace for  $r = 0\%$ . Looking at the second and third columns in Figure 6, the increased volatility makes the early exercised stock paths slower to converge to  $-1$ . This reflects the option holder's hesitation to exercise early, given the higher chance of a favourable movement in the underlying asset; while, the speeds of converging to  $0$  for not exercised stock paths are approximately the same because the impact of volatility is less pronounced when the option becomes deep out-of-the-money. However, looking at the bottom left subplot, the interaction of interest rates and volatility shows that, as rates increase, even volatile paths are more likely to be exercised early.

**3.2. Part (b).** Assume you have purchased the above American put option using the parameters  $T = 1$ ,  $S_0 = 10$ ,  $\mu = 5\%$ ,  $\sigma = 20\%$ , and the risk-free rate  $r = 2\%$ . Use  $N = 5000$  and a strike  $K = 10$ .

3.2.1. *i.* Simulate 10,000 sample paths of the asset and generate distributions for the P&L and distribution and the stopping time for a trader who purchased the option. Conditioned on those paths that are exercised (otherwise the distribution will have point masses), but record the probability of exercise to account for the point mass.

The function `calculate_pnl_stopping_time` evaluates the profit and loss of each path and determines the stopping time when the option is exercised early if applicable. For the early exercise stock paths, the function checks if the stock price crosses the exercise boundary at anytime. If so, it discounts the payoff  $K - S_t$  to the exercise time  $t$  and subtracts the option premium to arrive at the net profit and loss. For the no exercise stock paths, this function computes the payoff at maturity  $\max(K - S_T, 0)$  and discounts the payoff to the present, subtracting premium.

```
def calculate_pnl_stopping_time(paths, boundary_prices, times, K, premium, r):
    ...
    for i in range(num_paths):
        exercised = False
        for t_idx, time in enumerate(times):
            if paths[i, t_idx] < boundary_prices[t_idx]:
                pnl[i] = (K - paths[i, t_idx]) * np.exp(-r * time) - premium
                stopping_time[i] = time
                exercised = True
                break
        if not exercised:
            final_payoff = max(K - paths[i, -1], 0) * np.exp(-r * times[-1])
            pnl[i] = final_payoff - premium
```

The function `plot_pnl_stopping_time` plots the profit and loss distribution and the stopping time distribution only for paths where the option is exercised early by using `~np.isnan(stopping_time)`.

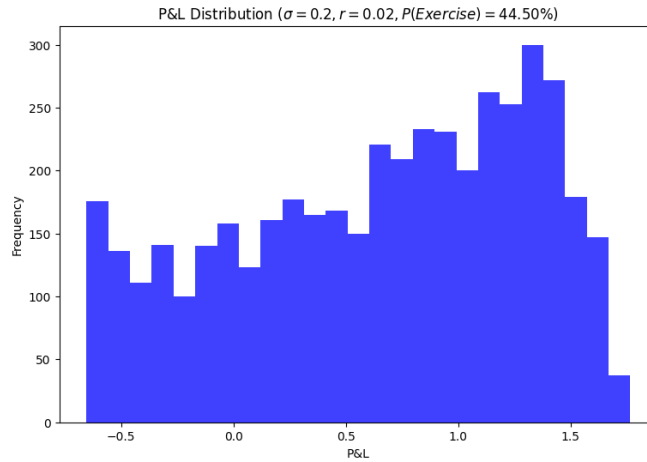


FIGURE 7. Profit and loss distribution when  $\sigma = 20\%$  and  $r = 2\%$ .

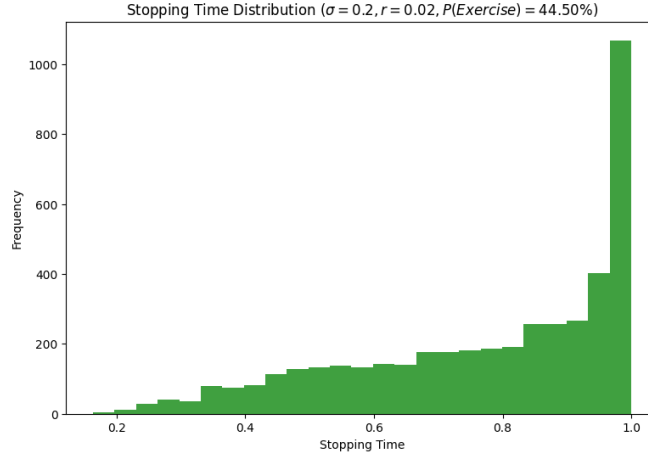


FIGURE 8. Stopping time distribution when  $\sigma = 20\%$  and  $r = 2\%$ .

3.2.2. *ii.* Repeat the above for various values of  $r$  and  $\sigma$ .

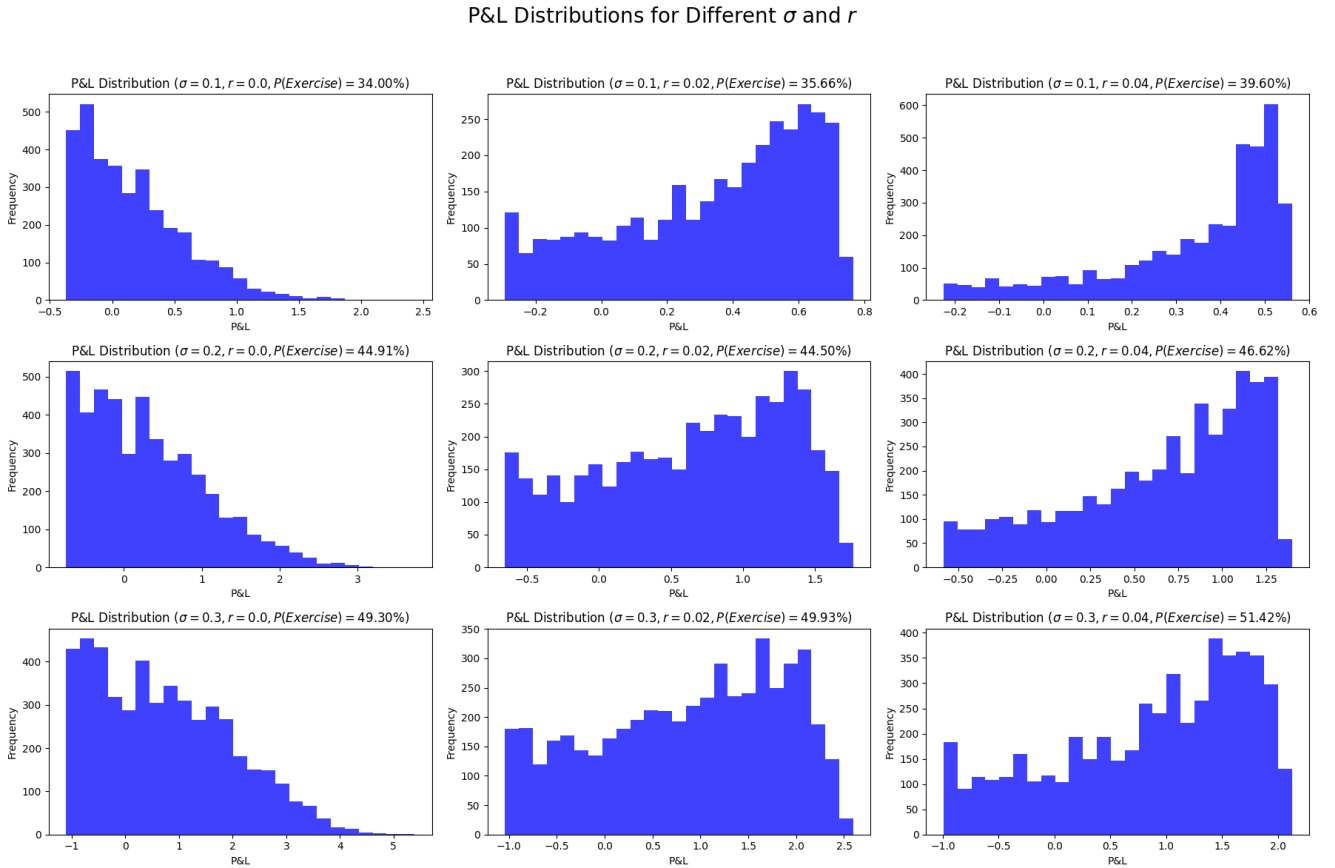
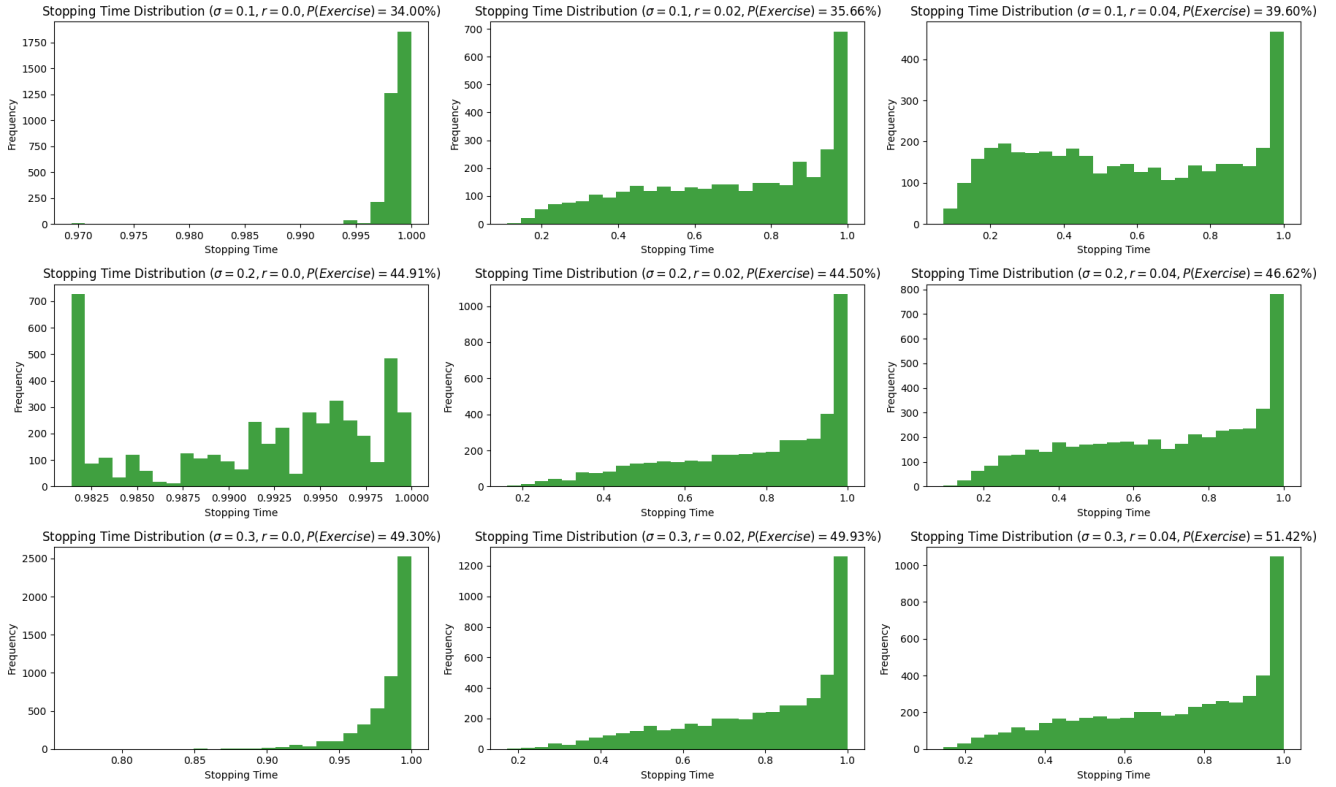


FIGURE 9. Profit and loss distributions for  $\sigma = 10\%, 20\%, 30\%$  and  $r = 0\%, 2\%, 4\%$ .

Stopping Time Distributions for Different  $\sigma$  and  $r$ FIGURE 10. Stopping time distributions for  $\sigma = 10\%, 20\%, 30\%$  and  $r = 0\%, 2\%, 4\%$ .

By leaving  $\sigma$  and  $r$  as variables in the `plot_exercise_boundary` function inside the `plot_pnl_stopping_time` function, we are able to generate profit and loss distributions and stopping time distributions for different  $\sigma$  and  $r$  as above presented.

```
def plot_pnl_stopping_time(sigma, r, ax_pnl, ax_stop, T=1, S0=10, K=10, mu=0.05,
    N=5000, num_paths=10000):

    boundary, times, boundary_prices, premium = plot_exercise_boundary(sigma, r, ax=ax)

    paths = simulate_sample_paths(sigma, r, T, S0, K, mu, N, num_paths)

    pnl, stopping_time = calculate_pnl_stopping_time(paths, boundary_prices, times, K,
        premium, r)

    probab_exercise = np.sum(~np.isnan(stopping_time)) / num_paths

    # Plot pnl and stopping time distributions
    ...
```

The analysis reveals that as volatility increases, the profit and loss (P&L) distributions widen, with higher P&L outcomes becoming more frequent. This indicates that greater volatility leads to a higher probability of profitable scenarios for the option holder. In other words, increased volatility amplifies the range of possible outcomes, which benefits the holder by increasing the likelihood of favorable payoffs. Furthermore, the shape of the P&L distribution is influenced by the risk-free interest rate. When the interest rate is low, the distribution tends to be right-skewed, indicating more positive outcomes. Conversely, when the interest rate is high, the distribution becomes left-skewed, suggesting a higher frequency of negative outcomes. This relationship reflects the impact of discounting: higher interest rates reduce the present value of future cash flows, which diminishes the profitability of the option.

The stopping time distributions provide insights into the exercise behavior under different market conditions. As volatility increases, the stopping times tend to cluster closer to maturity. This suggests that in highly volatile markets, traders are more likely to wait until the end of the option's life before exercising, as greater fluctuations increase the potential for more favorable outcomes near expiration. On the other hand, the risk-free rate significantly affects the timing of exercise. With higher interest rates, the stopping time shifts earlier, indicating that traders prefer to exercise the option sooner to capture the intrinsic value and avoid the adverse impact of discounting over time. In contrast, when the interest rate is low, the stopping times tend to gather near maturity, as the absence of significant discounting allows the trader to delay exercise without substantial loss in value.

These observations align with the results from part (a).

3.2.3. *iii.* Suppose that the realized volatility is  $\sigma = 10\%, 15\%, 20\%, 25\%, 30\%$ , but you were able to purchase the option with a volatility of  $\sigma = 20\%$  and you use the  $\sigma = 20\%$  exercise boundary in your trading strategy. Explore how the distributions of profit and loss and exercise time vary in this case.

By setting `sigma` and `r` as fixed parameters in the `plot_exercise_boundary` function inside the `plot_pnl_stopping_time` function, we are able to generate profit and loss distributions and stopping time distributions for different realized volatilities  $\sigma_r = 10\%, 15\%, 20\%, 25\%, 30\%$  under exercise boundary volatility  $\sigma = 20\%$  and a fixed risk-free rate  $r = 2\%$ .

```
def plot_pnl_stopping_time(sigma, r, ax_pnl, ax_stop, T=1, S0=10, K=10, mu=0.05,
    N=5000, num_paths=10000):
    boundary, times, boundary_prices, premium = plot_exercise_boundary(sigma=0.2, r=0.02, ax=ax)
    paths = simulate_sample_paths(sigma, r, T, S0, K, mu, N, num_paths)
    pnl, stopping_time = calculate_pnl_stopping_time(paths, boundary_prices, times, K,
        premium, r)
    prob_exercise = np.sum(~np.isnan(stopping_time)) / num_paths
    # Plot pnl and stopping time distributions
    ...
    # Loop over each realized sigma value to plot P&L and Stopping Time
    for i, sigma_realized in enumerate(sigma_realized_values):
        np.random.seed(0) # Ensure reproducibility
        plot_pnl_stopping_time(sigma_realized, r=0.02,
            ax_pnl=axes[i, 0], ax_stop=axes[i, 1])
```

## Distributions for Different Realized Volatilities

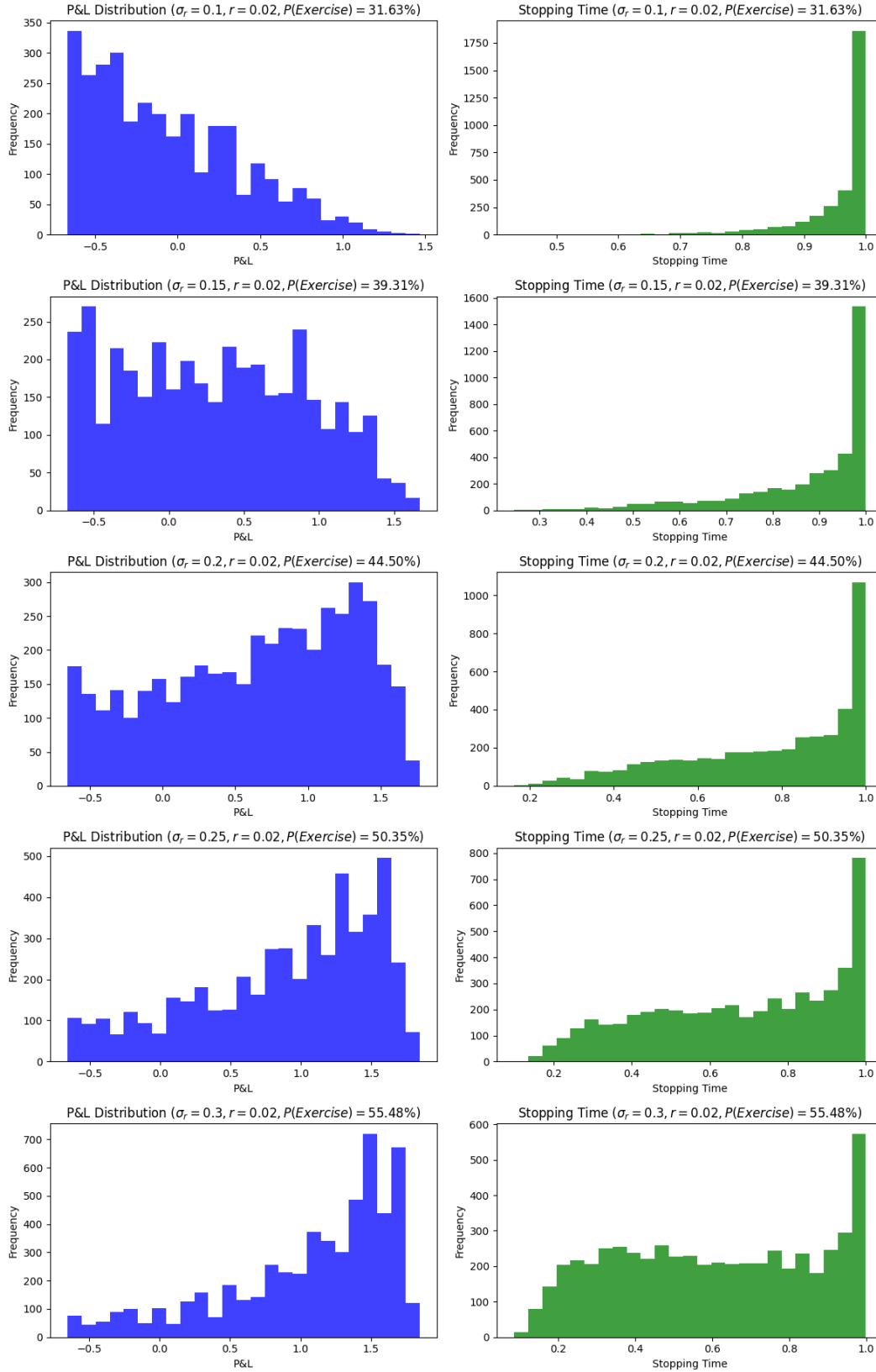


FIGURE 11. P&L and stopping time distributions for different realized volatility  $\sigma_r = 10\%, 15\%, 20\%, 25\%, 30\%$ .

As the realized volatility ( $\sigma_r$ ) increases, the P&L distribution widens, leading to a higher frequency of positive outcomes and indicating an increased probability of profitable scenarios for the option holder. At lower volatilities (e.g., 10% and 15%), the distributions are skewed towards smaller positive or negative P&L values, suggesting fewer profitable outcomes. In contrast, at higher volatilities (e.g., 25% and 30%), the distributions become more spread out, with a greater occurrence of higher P&L outcomes. Additionally, the probability of exercise  $P(\text{Exercise})$  increases with higher realized volatility, from 31.63% at  $\sigma_r = 10\%$  to 55.48% at  $\sigma_r = 30\%$ , indicating that higher realized volatility increases the chances of the option finishing in the money.

The stopping time distributions reveal that as realized volatility increases, the stopping times spread out, with a greater tendency towards earlier exercise. At lower volatilities (e.g., 10% and 15%), most exercises occur near maturity (i.e., close to 1), suggesting that traders prefer to wait until the end of the option's life. However, with higher volatilities (e.g., 25% and 30%), the stopping time shifts earlier, indicating that traders are more inclined to exercise the option before maturity. This behavior reflects the increased opportunities for favorable outcomes under higher volatility, reducing the incentive to wait until expiration.

These observations highlight the important role that realized volatility plays in both the P&L outcomes and the timing of exercise decisions.

#### 4. CONCLUSIONS

In this report, we explored the behaviour of American put options using a variation of the Cox-Ross-Rubinstein (CRR) model by deriving the branching probabilities under the risk-neutral measure and using the branching probabilities provided under the real-world measure. We analyzed how volatility and the risk-free interest rate influence the exercise boundary, simulated stock paths, delta hedging strategy, P&L outcomes, and stopping times.

The analysis in part (a)(iv) provided comprehensive insights into how these dynamics manifest across different scenarios. We found that higher volatility not only leads to later exercise but also widens the exercise boundary, offering more opportunities for the option holder. On the other hand, as interest rates rise, early exercise becomes optimal, reducing the option's time value. The interplay between volatility, interest rates, and exercise timing shows that traders tend to delay exercise in volatile markets to capture more upside potential, while in high-interest environments, the incentive shifts towards locking in intrinsic value earlier.

These findings align with part (b), where higher volatility results in both later exercise and greater probability of profitability, while higher interest rates drive earlier exercise and more negative P&L outcomes. This part also investigated P&L and stopping time distributions under different realized volatilities but fixed volatility and risk-free rate in the exercise boundary. The findings are that higher realized volatility leads to more profitable outcomes and later exercise as the option holder waits to capture favourable price movements.

In conclusion, this report highlights the complex interplay between volatility, interest rates, and option exercise decisions, emphasizing the importance of these factors for both pricing and optimal exercise strategies.

## CONTRIBUTION

Xinyi Shen and Shaofeng Kang drafted, discussed, and completed this project report together.

Xinyi Shen mainly contributes to transforming the logic into Python code and report writing.

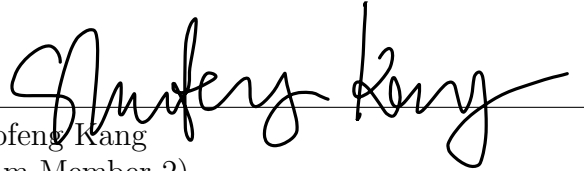
Shaofeng Kang mainly contributes to vectorizing the Python code to improve efficiency and report proofreading.

## Signatures



---

Xinyi Shen  
(Team Member 1)



---

Shaofeng Kang  
(Team Member 2)