Numerical Methods in Finance
Group Project

Zhaozhi Liu(zhaozhi.liu22@imperial.ac.uk, 01977539)
Zhengnan Dong(zhengnan.dong22@imperial.ac.uk, 02298872)
Weiyi Han (weiyi.han22@imperial.ac.uk, 02254742)

Department of Mathematics
Imperial College London

March 12, 2023

1 Part I

1.1 Model description

In this section, we utilize the trinomial tree model to price European options, American options, and look-back options. Before proceeding with our analysis, we provide a detailed introduction to the trinomial tree model.

The trinomial tree model assumes that the stock price can move in three directions with different probabilities from time t to time $t+1$. We define the following model parameters: σ as the volatility of the stock price, r as the riskless interest rate, λ as the parameter that controls the probability and fluctuation of the price movement, T as the maturity, K as the strike price, S_0 as the initial stock price, N as the number of time periods in the trinomial tree, and $\Delta t := \frac{T}{N}$ as the time step size. It is important to note that for all the questions presented in this section, we fix $\sigma = 0.2$, $r = 0.01$, and $T = 1$.

The dynamics of stock price can be formulated as following,

$$S_n = S_0 \prod_{i=1}^{n} \xi_i \tag{1}$$

with random variable ξ_i being iid, and suppose

$$\xi_i = \begin{cases} u, & \text{with probability } q_u; \\ m, & \text{with probability } q_m; \\ d, & \text{with probability } q_d, \end{cases} \tag{2}$$

where

$$u = e^{\lambda\sigma\sqrt{\Delta t}}, \quad m = 1, \quad d = e^{-\lambda\sigma\sqrt{\Delta t}} \tag{3}$$

$$q_u = \frac{1}{2\lambda^2} + \frac{\left(r - \frac{\sigma^2}{2}\right)\sqrt{\Delta t}}{2\lambda\sigma}, \quad q_m = 1 - \frac{1}{\lambda^2}, \quad q_d = \frac{1}{2\lambda^2} - \frac{\left(r - \frac{\sigma^2}{2}\right)\sqrt{\Delta t}}{2\lambda\sigma} \tag{4}$$

For the convenience of computations, we use the matrix M to store all the price happens in each time point.

$$M = \begin{bmatrix} S_0 & S_0u & S_0u^2 & S_0u^3 & \dots & S_0u^N \\ 0 & S_0 & S_0u & S_0u^2 & \dots & S_0u^{N-1} \\ 0 & S_0d & S_0 & S_0u^1 & \dots & \vdots \\ 0 & 0 & S_0d & S_0 & \dots & S_0u \\ 0 & 0 & S_0d^2 & S_0d & \dots & S_0 \\ 0 & 0 & 0 & S_0d^2 & \dots & \vdots \\ \vdots & \vdots & \vdots & S_0d^3 & \ddots & S_0d^{N-1} \\ 0 & 0 & 0 & \dots & \dots & S_0d^N \end{bmatrix} \tag{5}$$

We have that (k,n)-th entry of M is (indexing starting from 0)

$$s_k^n = S_0u^nd^k, n = 0, 1, ..., N; k = 0, 1, ..., 2n, \tag{6}$$

To generate the matrix M (5), we need a way to index the stock price described by our model. We consider a unit in the trinomial tree(i.e. a one-period trinomial tree) suppose at time $t = n$, the stock price is s_k^n, let it be the root node, we can then describe this sub-trinomial tree as following

$$\begin{cases} s_k^{n+1} = S_0 u^{n+1} d^k = s_k^n u \quad (s_n \text{ go up}) \\ s_{k+1}^{n+1} = S_0 u^{n+1} d^{k+1} = s_k^n ud = s_k^n \quad (s_n \text{ stay unchanged}) \\ s_{k+2}^{n+1} = S_0 u^{n+1} d^{k+2} = s_k^n ud^2 = s_k^n d \quad (s_n \text{ go down}) \end{cases}$$

In order to put these dynamics in a matrix, we consider u as a rightward shift operator, which shifts each element one index to the right, and d as a downward shift operator, which shifts each element one index down. When we multiply these operators with matrix elements, the direction of the shift corresponds to the direction of the operator. So we generate this matrix M from left to right, from top to bottom. Thus, we have an indexing system that is consistent with our model. To be more precise, the entry in the k-th row and n-th column of M can be expressed as $S_0 u^n d^k$, where S_0 is the entry in the top-left corner of the matrix. Intuitively, this expression can be interpreted as starting from S_0 and then shifting right n times and down k times to reach the desired position in the matrix. By utilizing this indexing system, we can effectively represent the dynamics of the stock price in a concise and efficient manner.

M is in $\mathbb{R}^{(2N+1)\times(N+1)}$, the column index represents the time point, and the row index represents each possible price at that column, in descending order. At each column j, the first $2j + 1$ elements are non-zero. We then give the pseudo code to generate matrix M.

Algorithm 1 Generate stock matrix M

```
Generate a zero matrix $M$ with $2N + 1$ rows and $N + 1$ columns.
Set the initial stock price be $s$.
for $n = 0, 1, 2, ..., N$ do
  for $k = 0, 1, 2, ..., 2n$ do
    M[k,n]=$su^n d^k$
  end for
end for
```

1.2 European option

Question 1.1.a

We construct a matrix V to demonstrate the option price flow, V has the same shape as M. We denote V_k^n to be the option price at time $t = n$ and at which the stock price is s_k^n. To generate V, we need backward induction starting from the terminal time $t = N$. Setting the function g to be the payoff of the interested option, in this question, $g(x) = \max(x - K, 0)$. Then we have

$$V_k^N = g(s_k^N); k = 0, 1, 2, ..., 2N \tag{7}$$

Then by looping back in time for $n = N - 1, N - 2, ..., 0$, we get

$$V_k^n = e^{-r\Delta t} \left[q_u V_k^{n+1} + q_m V_{k+1}^{n+1} + q_d V_{k+2}^{n+1} \right] \tag{8}$$

the price of the European option is the (0,0)-th entry of matrix V. The algorithm is shown in (Algorithm 2).

Algorithm 2 Generate option price matrix V

```
Generate a zero matrix $V$ with $2N+1$ rows and $N+1$ columns.
for $k = 0, 1, 2, ..., 2N$ do
  V[k,N]= $g$(M[k,N])
end for
for $n = N-1, N-2, ..., 0$ do
  for $k = 0, 1, 2, ..., 2n$ do
    Calculate (8)
  end for
end for
return V[0,0]
```

Question 1.1.b

The price of a European option will be convergent to the theoretical price calculated by Black-Scholes formula as the number of time period N increasing. Figure 1 demonstrates this convergence of a European call option with initial stock price $S_0 = 100$, strike price $K = 100$.

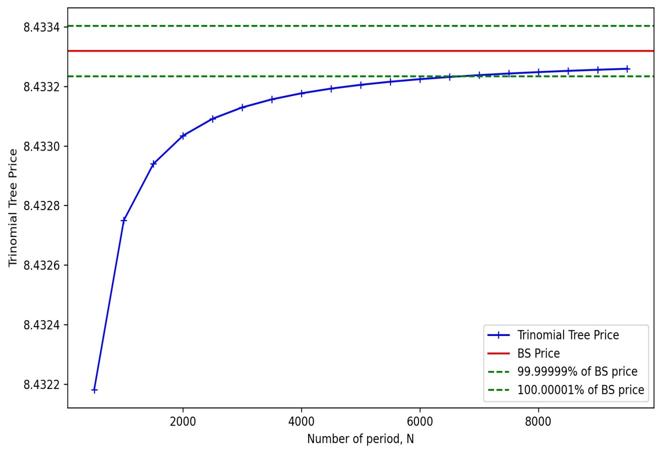

Figure 1: Trinomial Tree Prices of number of period $N \in (500, 10000)$, length of period is 501

The red line in Figure 1 indicates the theoretical European call option price computed using Black-Scholes formula, the dashed green lines are the upper and lower boundary of the interval between 99.99999% and 100.00001% of the value of the theoretical price. The blue line is the price value computed using the trinomial tree method.

The free parameter λ could impact the speed of convergence as shown in Figure 2 below.

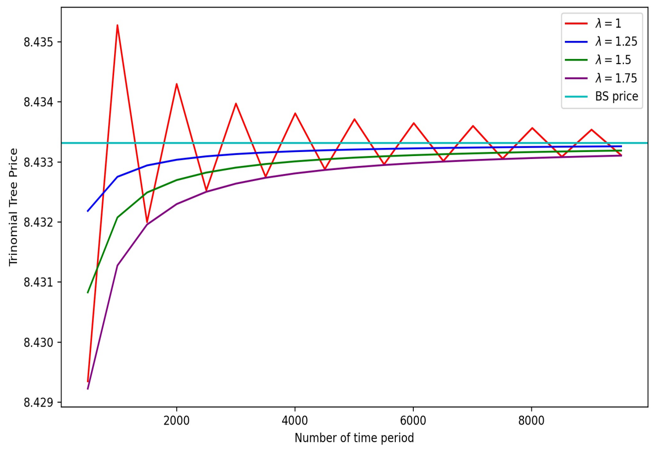

Figure 2: Converging Trinomial Tree Prices with free parameter $\lambda \in \{1, 1.25, 1.5, 1.75\}$

For $\lambda = 1$, $q_m = 0$, the trinomial model degenerates into a binary tree. The price oscillates over time, and the amplitude of oscillation decreases as the number of periods increases and slowly converges to the true value. For $\lambda > 1$, the price increases monotonically. As we can see from the graph, the speed of convergence of the trinomial tree method decreases when λ increases. We may infer from the graph that for a European at-the-money call option, choosing a λ slightly larger than 1 can provide a good convergence result. But this result may change if we use different S_0.

Question 1.1.c

To check the impact of the initial stock price on the European Option price, Figure 3 demonstrates the behavior of the European Option price with varying initial stock price. It can be seen that the prices obtained using the trinomial tree method perfectly fit the price calculated from the closed-form formula, regardless of the value of λ.

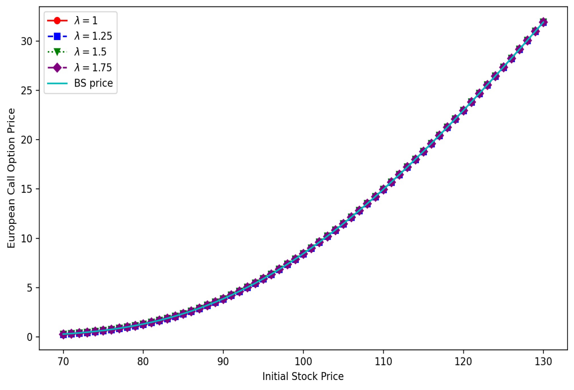

Figure 3: The European Option prices for initial stock price $S_0 \in [70, 130]$, with free parameter $\lambda \in \{1, 1.25, 1.5, 1.75\}$, $N = 500$

To tell the difference between different choices of λ, Figure 4 below visualizes the absolute differences(in log scale) between the trinomial tree prices and the exact Black-Scholes prices for a European call option under different initial stock prices for each value of $\lambda \in \{1, 1.25, 1.5, 1.75\}$.

From the error results obtained with different initial values(4), we may infer that the trinomial tree model provides a relatively good result when λ is set to 1.25. For a deep in-the-money, deep out-of-the-money, and at-the-money option, among the four choices of λ, $\lambda = 1.25$ gives the relatively lowest error result. Therefore, we set λ to 1.25 for the following analysis.

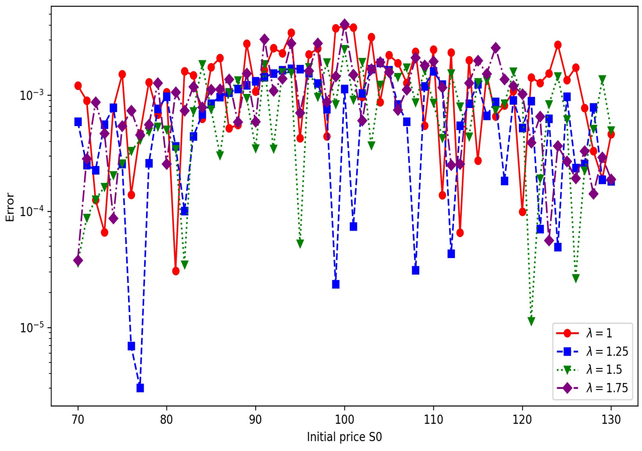

Figure 4: Absolute differences between the trinomial tree prices and the exact Black-Scholes prices for initial stock price $S_0 \in [70, 130]$, with free parameter $\lambda \in \{1, 1.25, 1.5, 1.75\}$, $N = 500$

1.3 American option

Question 1.3.a

For the American option, we need to modify the algorithm (2). Due to its early-exercise property, we first need to calculate the pay-off at all time points(in the case of the European option, we only calculated the pay-off at time $t = N$) and compare it with the so-called continuation value. So equation (8) now becomes,

$$V^n_{\text{American}} = \begin{cases} g\left(S_N\right) & \text{for } n = N \\ \max\left\{g\left(S_n\right), e^{-r\Delta t}\left[q_u V^{n+1}_k + q_m V^{n+1}_{k+1} + q_d V^{n+1}_{k+2}\right]\right\} & \text{for } n = 0, 1, \ldots, N-1 \end{cases} \tag{9}$$

Question 1.3.b

Time-zero value of an American call option and the intrinsic value of the option against the initial stock price is shown in Figure 5. One can prove[1] that for a non-dividend paying stock used as the underlying asset for an American call option, the price of the option is equivalent to that of a European call option.

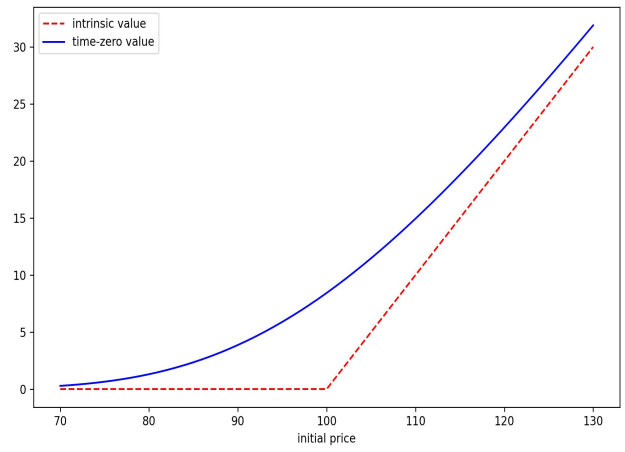

Figure 5: Time-zero value of an American call option and the intrinsic value of the option against the initial stock price S_0.

This conclusion indicates that the early exercise feature of American call options does not contribute to its value. This is because the discounted intrinsic value process $\{e^{-rt}(S_t - K)^+\}_{t\geq 0}$ of a call option is a submartingale under risk-neutral probability. Therefore, the optimal exercise strategy for an American call option at time $t = 0$ is to wait for the natural maturity of the option.

Question 1.3.c

There exists a significant difference in the pricing of American and European put options, whereby American put options exhibit higher prices than their European counterparts. Notably, deep in-the-money European put options are priced lower than the intrinsic value of American put options. This can be seen in Figure 6 that when the initial price is at an extremely low level, the intrinsic value of the American put option is larger than the value of the European put option. Consequently, the optimal exercise strategy for such options will undergo significant changes compared to call options. One can show that the discounted intrinsic value process $\{e^{-rt}(K - S_t)^+\}_{t\geq 0}$ of an American put option is not a submartingale anymore. In this case, the possibility of receiving the strike price K upon exercise of a put option may motivate the option holder to exercise the option prematurely to avoid the erosion of the payment's value through discounting. When the stock price is low, this effect becomes more significant and the option holder wants to exercise the option

[1]the proof can be found in Shreve's book Stochastic calculus for finance I/II

as soon as possible to lock the payoff K and prevent it from being discounted. Therefore, the optimal exercise strategy for an American put option at time $t = 0$ is that for the initial price such that $S_0 < K$.

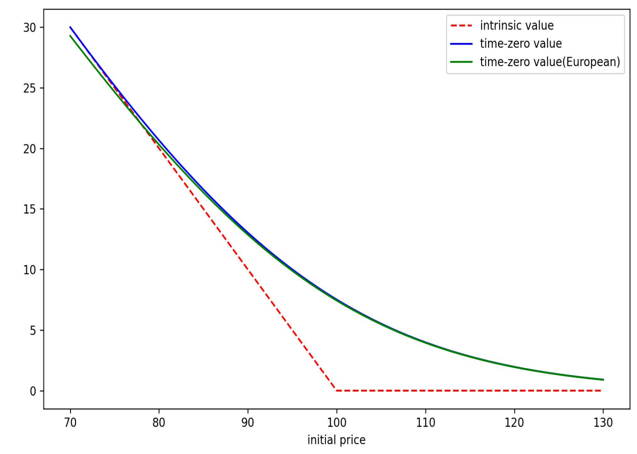

Figure 6: Time-zero value of an American put option and the intrinsic value of the option against initial stock price S_0.

We then analyze how changes in σ and r could affect the optimal strategy. Let V_n denote the time n value of the American put option given by (9) and g_n be the intrinsic value at time n. We denote the stopping time $\tau := \min\{n; V_n = g_n\}$. Based on the above analysis, we can give the formula for the price of American put at time 0.

$$V = \mathbb{E}^{\mathbb{Q}}\left[\mathbb{I}_{\tau \le N} e^{-r\tau} g_\tau\right] \tag{10}$$

The above formula can be simply generated at time $t > 0$. Therefore, we can plot the exercise boundary at each time point as in Figure 8. In this graph, we will not exercise the option if the stock price $S_n(\sigma, r)$ is above the line. The dotted line indicates the strike price.

As the value of σ increases, the exercise boundary decreases, implying that more time is required to wait for the stock price to fall within this boundary compared to a larger σ. This is because the option price increases with an increase in σ as shown in Figure 7, as the number of days to expiration decreases, the time value of an option decreases while the intrinsic value relatively increases. This makes it more likely for us to exercise the option.

As for the risk-free rate r, according to Figure 7 and Figure 8, as the risk-free interest rate increases, the exercise boundary of the American put options also expands, leading to the possibility of exercising options earlier compared to scenarios with lower risk-free interest rates, in order to secure the payoff of the option without excessive discounting.

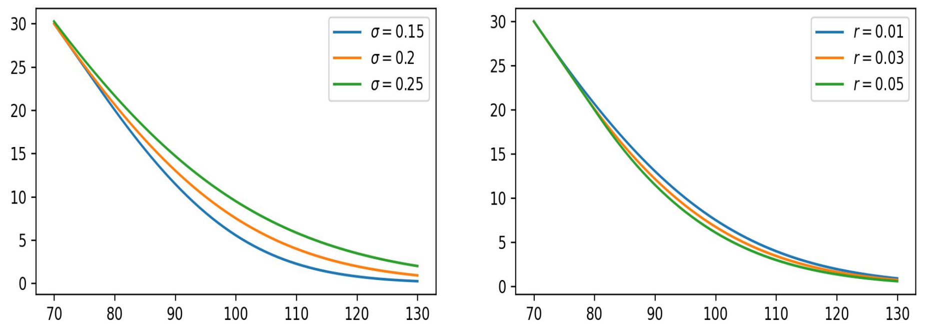

Figure 7: Price of American put with different σ and r, x-axis represents the initial price of the stock

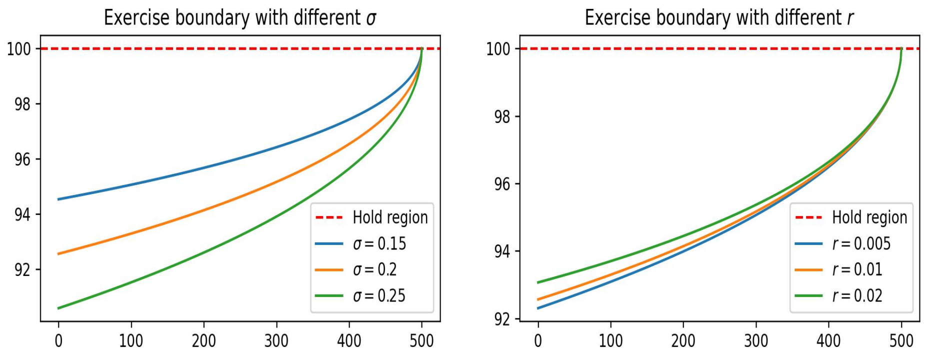

Figure 8: Exercise boundary with different σ and r, x-axis represents the i-th time points, $i = 0, 1, ..., N$, $N = 500$

1.4 Look-back option

Question 1.4.a

In this problem, we define the payoff function for a European floating look-back put option as $g(S_T, \omega) = \max_{0 \le u \le T} S_u - S_T$. This function depends not only on the final value S_T of the underlying asset but also on the entire path $\{S_t(\omega)\}_{t \le T}$ for each path ω in the underlying probability space. To remove this dependence, we should choose an auxiliary function that correctly indexes the information of $\max_{0 \le u \le T} S_u$. We introduce a variable $M_n = \max_{i=0,1,\dots,n} S_i$ and define the payoff function as $g(s, m) = m - s$. To solve this problem, we will use the shooting method.

Firstly, we need to find a suitable method to correctly index $M_n = \max_{i=0,1,\dots,n} S_i$ as what we did on the stock price (6). Since the initial stock price is S_0 and the largest possible price at time n is $S_0 u^n$, the maximum price recorded between time zero and time n must take value in the set

$$\{S_0, S_0 u, S_0 u^2, \dots, S_0 u^n\}$$

Hence we construct the grid for M_n as

$$m_j^n = S_0 u^{n-j} \quad \text{for } n = 0, 1, \dots, N \text{ and } \quad j = 0, 1, \dots, n$$

which represents the j-th possible value of M_n. Note that we have $m_j^n > s_k^n$, and based on $ud = 1$, we have

$$s_k^n = S_0 u^n d^k = S_0 u^{n-k}, n = 0, 1, \dots, N; k = 0, 1, \dots, 2n,$$

hence, we have the relation $j < k$.

Secondly, we generate the payoff at time $t = N$, but in this case, the option price matrix defined in question 1.1.a cannot be used in this question due to the path-dependence property of the look-back option. Before we introduce how to generate the option price matrix V for the look-back option, we sketch some properties of the matrix V.

- $V \in \mathbb{R}^{(N+1)\times(2N+1)\times(N+1)}$
- $V^n_{(k,j)}$ represents the price of the option at time $t = n$, given that the stock price passes through the node $M_{k,n}$ (defined in (5)), and that the j-th possible value of M_n (the running maximum up to time n) is used as the running maximum.

We can easily generate the final payoff, which we will describe in the algorithm later. However, since we use backward induction to calculate the option price, we need to determine how the dynamics of $M_{n+1} = \max(M_n, S_{n+1})$ can be represented in a matrix indexing system. For each k_{new}, we have the following

$$\begin{aligned} m^{n+1}_{j_{new}} &= \max(m^n_j, s^{n+1}_{k_{new}}) \\ n + 1 - j_{new} &= \max(n - j, n + 1 - k_{new}) \\ j_{new} &= n + 1 + \min(j - n, k_{new} - (n + 1)) \\ &= \min(j + 1, k_{new}) =: \phi(j, k_{new}) \end{aligned}$$

Finally, we can compute the price of the option follows the equation:

$$V^n_{k,j} = e^{-r\Delta t}\left[q_u V^{n+1}_{k,\phi(k,j)} + q_m V^{n+1}_{k+1,\phi(k+1,j)} + q_d V^{n+1}_{k+2,\phi(k+2,j)}\right], \tag{11}$$

where $\phi(k, j) = \min(k, j + 1)$. The algorithm is given below.(3) The algorithm (3) is for the European floating look-back option. For the American floating look-back option, similar modifications can be made by referring to question 1.2.a.

Algorithm 3 Compute the price of a floating strike look-back put option

```
Generate a zero matrix $V$ with shape $(N+1,2N+1,N+1)$.
Set the initial stock price be $s$.
for $k = 0,1,2,...,2N$ do
    for $j = 0,1,2,...,\min(k,N)$ do
        V[N,k,j]=$s(u^{N-j} - u^{N-k})$ (The final payoff)
    end for
end for
Define $\phi(k,j) = \min(k,j+1)$
for $n = N-1, N-2, ..., 0$ do
    for $k = 0,1,2,...,2n$ do
        for $j = 0,1,2,...,\min(k,n)$ do
            Calculate (11)
        end for
    end for
end for
return V[0,0,0]
```

Question 1.4.b

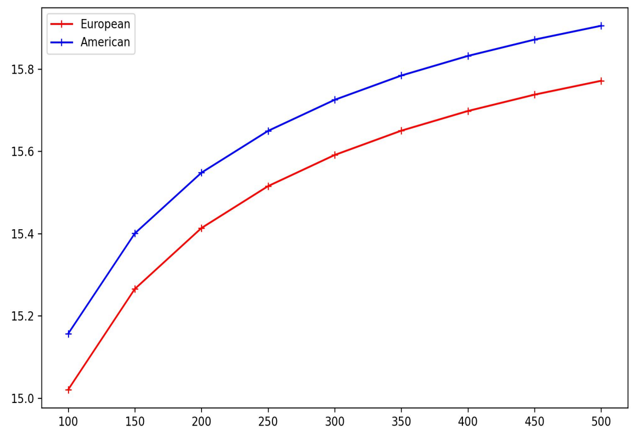

Figure 9: The price of floating strike look-back put option calculated using trinomial tree method, the x-axis represent the number of time period N and the y-axis represent the price

Figure 9 shows the price of the two floating strike look-back put options, we can see that the two calculated prices are still converging to their theoretical price(the theoretical formula is given in the jupyter notebook). Due to the feature of the floating look-back option, using the trinomial tree method to calculate its price is a computationally expensive task both in terms of running time and memory usage. Therefore, we only provide the convergence process before $N = 500$. When $N = 1300$, the calculated price for the European style look-back option is 16.01, which is still converging to the theoretical result (which is 16.41), but at a very slow pace, thus, when N needs to be a relatively large number (e.g., 5000), the trinomial tree algorithm may provide a relatively good result.

For the time complexity, due to the presence of triple nested for loops in the algorithm(3), the time and space complexity of the code are both $\mathcal{O}(N^3)$. Therefore, we used the vectorization technique of NumPy to optimize some operations in terms of running time, we can see from the following figures that the computation speed has been increased by more than ten times with the use of vectorization. The specific results are shown in the following three figures. In the first two figures, we also performed polynomial fitting to demonstrate that the execution time follows a polynomial growth of order 3. In fact, decorating non-vectorized functions with Numba can achieve faster speeds than vectorized functions. However, in this specific context, we aim to demonstrate that algorithm 3 grows at a cubic polynomial rate. Therefore, we have refrained from using Numba to accelerate the code. By maintaining the original implementation, we can accurately depict the algorithm's performance and remain consistent with previous research.

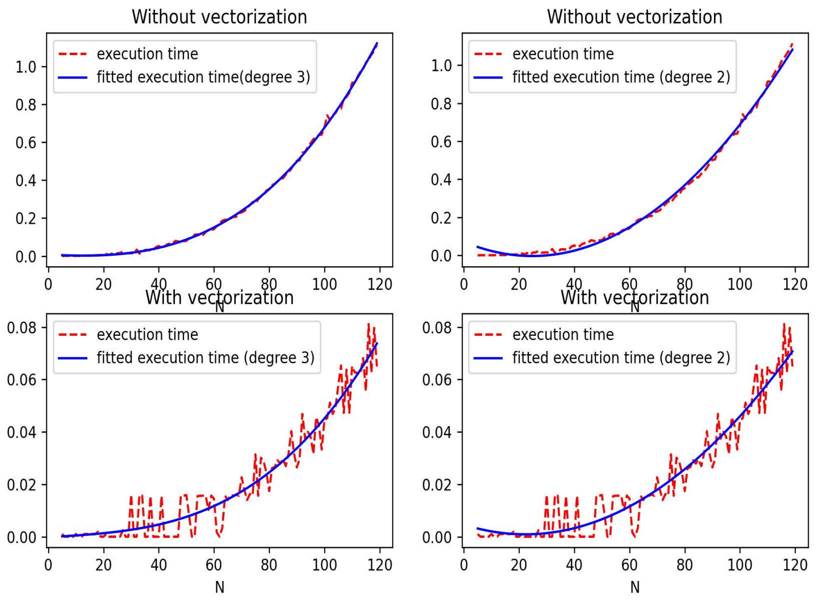

Figure 10: Execution time for calculate the price of European style look-back option for N in $(1, 120)$, with polynomial fitted result.

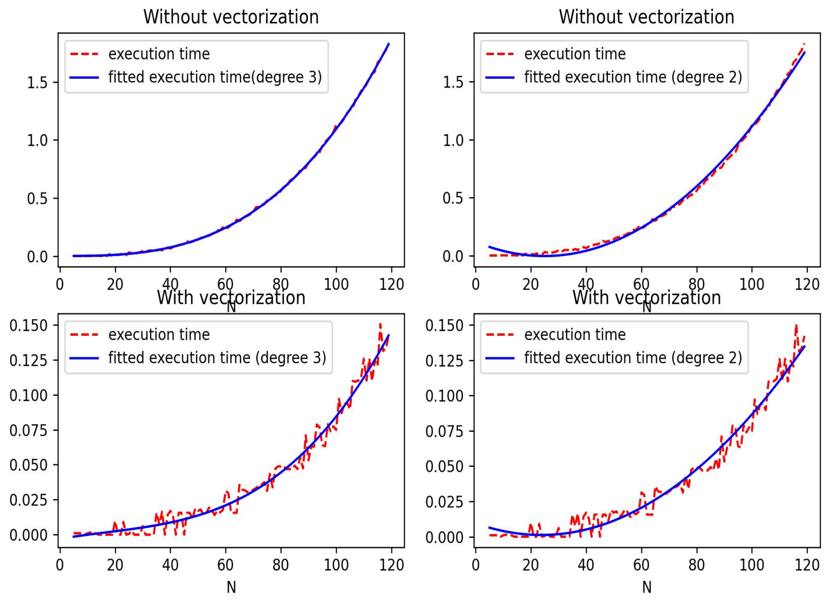

Figure 11: Execution time for calculate the price of American style look-back option for N in $(1, 120)$, with polynomial fitted result.

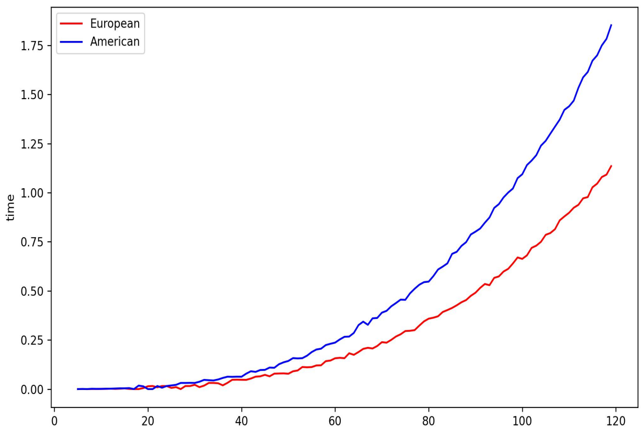

Figure 12: Comparison of the execution time between American/European look-back option as N increasing.

2 Part II

In this question, we mainly focus on one generalization to the Black-Scholes model, where the underlying stock price is assumed to follow a geometric brownian motion. Since the volatility of financial assets is not constant, a local volatility model is analyzed here as

$$\frac{dS_t}{S_t} = rdt + \sigma\left(t, S_t\right) dB_t$$

And the local volatility function is simply assumed to be

$$\sigma(t,s) = \left(1 + \frac{t}{30}\right)\left[0.1 + 0.4\exp\left(-\frac{s}{50}\right)\right]$$

2.1 Solution of the PDE

In this section, several finite difference methods are implemented to solve the following PDE, the derivation follows the idea in the lecture notes.

Question 2.1.a

To reformulate the PDE in Feynman-Kac formula into an initial condition problem, we start with the Feynman-Kac formula:

$$\begin{cases} \frac{\partial V}{\partial t} + \frac{\sigma^2(t,s)s^2}{2}\frac{\partial^2 V}{\partial s^2} + rs\frac{\partial V}{\partial s} - rV = 0, & t < T \\ V(T,s) = g(s), & t = T \end{cases}$$

where $V(t,s)$ is the solution to the PDE, and $V(T,s)$ is the terminal condition. Let $\tau = T - t$, and then by calculation, we have

$$\begin{cases} \frac{\partial V}{\partial \tau} = \frac{\psi^2(\tau,s)s^2}{2}\frac{\partial^2 V}{\partial s^2} + rs\frac{\partial V}{\partial s} - rV, & \tau < T \\ V(0,s) = g(s), & \tau = 0 \end{cases}$$

where $\psi(\tau,s) = \left(1 + \frac{T-\tau}{30}\right)\left[0.1 + 0.4\exp\left(-\frac{s}{50}\right)\right]$
.

The variation of ψ with $(T-\tau) \in [0,1]$ and $S \in [10,300]$ can be observed by Figure 13.

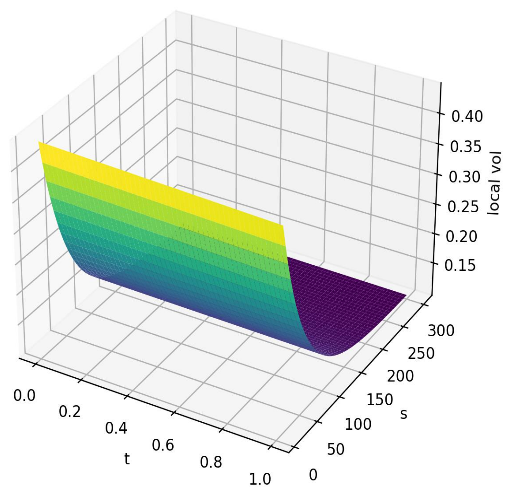

Figure 13: 3D surface: local volatility for $s \in [0, 300]$ and $t \in [0, 1.0]$

Question 2.1.b

Explicit, implicit, and Crank-Nicolson schemes are then derived to solve the PDE. Firstly, we approximate the open domain of S by a truncated interval $[10, 300]$. And according to the analysis in Lecture notes in week 3, for call option with initial condition $V(0, s) = (s - K)^+$:

- Call option becomes worthless at zero stock price,so $V(\tau, s) = 0$
- For the very large initial stock price, $V(\tau, s_{max}) \approx s_{max} - Ke^{-t\tau}$

Moreover, we are supposed to construct a uniform grid over D with $N+1$ points along the time dimension and $M+1$ points along the space dimension. Here we set $N = 2000, M = 200$. let

$$\Delta\tau := \frac{T}{N}, \quad \Delta x := \frac{x_{\max} - x_{\min}}{M}.$$

Then the values of the grid points are given by

$$\begin{aligned} \tau_n &:= n\triangle\tau, & n = 0, 1, \ldots, N \\ x_k &:= x_{\min} + k\triangle x, & k = 0, 1, \ldots, M \end{aligned}$$

Then we start to discretize the PDE by finite difference. Let V_k^n be our numerical approximation of $V(\tau_n, x_k)$, and $a_k^n := a(\tau_n, x_k), b_k^n := b(\tau_n, x_k)$ and $c_k^n := c(\tau_n, x_k)$.

Explicit Scheme

Supposing we use the forward difference for the time derivative, the discretized system by induction is

$$V_k^{n+1} = A_k^n V_{k-1}^n + (1 + B_k^n) V_k^n + C_k^n V_{k+1}^n$$

$$A_k^n := \frac{\Delta\tau}{\Delta x^2}a_k^n - \frac{\Delta\tau}{2\triangle x}b_k^n, \quad B_k^n := -\frac{2\triangle\tau}{\triangle x^2}a_k^n - \triangle\tau c_k^n, \quad C_k^n := \frac{\Delta\tau}{\Delta x^2}a_k^n + \frac{\Delta\tau}{2\triangle x}b_k^n$$

Implicit Scheme

Supposing we use the backward difference for the time derivative, the discretized system by induction is

$$V_k^{n-1} = -A_k^n V_{k-1}^n + (1 - B_k^n)\, V_k^n - C_k^n V_{k+1}^n$$

$$A_k^n := \frac{\Delta\tau}{\Delta x^2}a_k^n - \frac{\Delta\tau}{2\triangle x}b_k^n, \quad B_k^n := -\frac{2\triangle\tau}{\triangle x^2}a_k^n - \triangle\tau c_k^n, \quad C_k^n := \frac{\Delta\tau}{\Delta x^2}a_k^n + \frac{\Delta\tau}{2\triangle x}b_k^n$$

Crank-Nicolson Scheme

By choosing $\theta = 1/2$ in the weighted average of the explicit and implicit schemes, the Crank-Nicolson scheme is also applied here.

Prices of European option under different schemes for $s \in [10, 300]$ are plotted in Figure 14, which indicates the consistency of the results under the given setting.

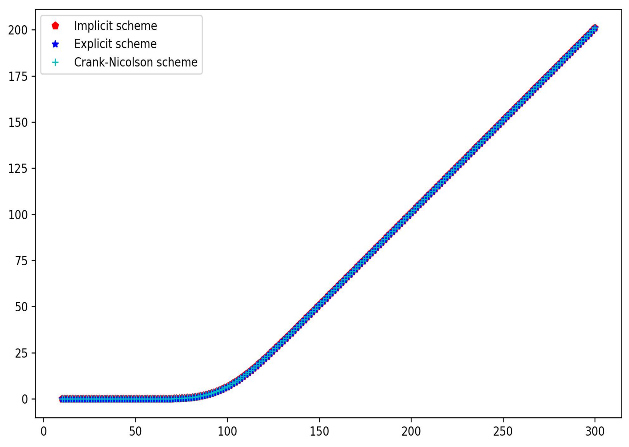

Figure 14: Prices of European option under different schemes for $s \in [10, 300]$

Question 2.1.c

Setting $K = 100$ and $T = 1$, the prices of the European call option under the local volatility model are reported in Table 1, which further verified the consistency of results.

2.2 Implied Volatility

Given the price of a call option, implied volatility σ_{imp} can be calculated by

$$BS_{\text{call}}\ (S_0, K, \sigma_{imp}, r, T) = C$$

Initial price	Explicit scheme	Implicit scheme	Crank-Nicolson scheme
80	0.750779	0.751032	0.750906
85	1.474107	1.474072	1.474090
90	2.643789	2.643370	2.643580
95	4.371698	4.370945	4.371321
100	6.725446	6.724551	6.724998
105	9.710965	9.710168	9.710566
110	13.273144	13.272624	13.272884
115	17.313230	17.313038	17.313134
120	21.714387	21.714460	21.714423

Table 1: Prices of European Call Option under Different Schemes

Question 2.2.a

A function in the form of $f(x) = 0$ is formulated to compute the implied volatility σ_{imp} given (S_0, K, r, T, C) as input values.

$$f(s, K, r, T) = BS_{\text{call}}\ (S_0, K, \sigma_{imp}, r, T) - C,$$

and the root can be calculated using `scipy.optimize.brentq`.

Question 2.2.b

Crank-Nicolson scheme is chosen here to solve the PDE. The price of a call option is separately obtained by using the local volatility model and constant $\sigma(t; s) = 15.7\%$, and the results are plotted in Figure 15.

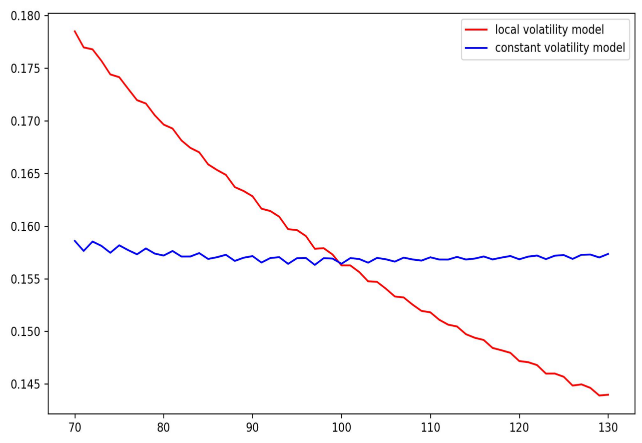

Figure 15: Implied volatility calculated by constant/local volatility model.

It can be seen in Figure 15 that the implied volatility is a decreasing function of the strike price, and the implied volatility corresponding to deep in-the-money call options is much higher than that of deep out-of-the-money call options. As stock prices and volatility usually exhibit a negative correlation, when stock prices rise, volatility tends to decrease, which can be regarded as a signal for making trading strategies.

As for the constant volatility, we get a roughly straight line as expected.

2.3 Down-and-in Barrier Put Option

The payoff of a down-and-in barrier put option is

$$(K - S_T)^+ \mathbb{I}_{(L_T \leq B_{in})}$$

where $L_t := \inf_{0 \leq u \leq t} S_u$ is the running minimum of the stock price up to time t, and B_{in} is the knock-in level of the option.

Question 2.3.a

We derived two methods to solve this question, one is directly starting from Feynman-Kac Theorem, and the second one utilizes the in-out parity to derive the price of the down-and-in barrier put option.

Method 1

By Feynman-Kac Theorem, and the property of the payoff, the price V of a down-and-in barrier put option must satisfy the following PDE with boundary conditions as follows:

$$\begin{cases} \frac{\partial V}{\partial \tau} = \frac{\psi^2(\tau,s)s^2}{2}\frac{\partial^2 V}{\partial s^2} + rs\frac{\partial V}{\partial s} - rV, & \tau < T \\ V(0,s) = (K-s)^+ \mathbb{I}_{(s \leq B_{in})}, & \tau = 0 \\ V(\tau,s) = Put(s,K,r,\tau,\psi(\tau,s)), & \tau < T; s \leq B_{in} \\ V(\tau, s_{min}) = Ke^{-r\tau} - s_{min} & \tau < T \\ V(\tau, s_{max}) = 0 & \tau < T \end{cases}$$

Where $Put(s,K,r,\tau,\psi(\tau,s))$ represent the price of the European put option with parameters $(s,K,r,\tau,\psi(\tau,s))$. This method requires us to numerically solve two PDEs over the same interval $[s_{min}, s_{max}]$: one for the down-and-in barrier put, as mentioned above, and another for the European put option, which is nested within the PDE of the down-and-in barrier put.

Method 2

We can also use the in-out parity to derive the price of the down-and-in barrier put option.

$$\text{European Put} = \text{down-and-in put} + \text{down-and-out put}$$

The payoff function for down-and-out put is

$$(K - S_T)^+ \mathbb{I}_{(L_T > B_{in})}$$

The boundary condition of the PDE of the down-and-out put option is simpler than that of down-and-in put, which is

$$\begin{cases} \frac{\partial V}{\partial \tau} = \frac{\psi^2(\tau,s)s^2}{2}\frac{\partial^2 V}{\partial s^2} + rs\frac{\partial V}{\partial s} - rV, & \tau < T \\ V(0,s) = (K-s)^+ \mathbb{I}_{(s>B_{in})}, & \tau = 0 \\ V(\tau, B_{in}) = 0 & \tau < T \\ V(\tau, s_{max}) = 0 & \tau < T \end{cases}$$

For European option, we are interested in the interval $[s_{min}, s_{max}]$, for down-and-out, we are interested in the interval $[B_{in}, s_{max}]$. Therefore, compared to Method 1, since $[B_{in}, s_{max}]$ is contained in $[s_{min}, s_{max}]$, we will have a smaller error(we presented the difference of the two methods in the Jupyter Notebook).

Question 2.3.b

In this part, the running results using the local volatility model and a constant volatility model with $\sigma(t,s) = 15.7\%$ are calculated and compared. Firstly, the Crank-Nicolson scheme is chosen here to solve the PDE based on the local volatility model. Moreover, in order to compute the price of the down-and-in barrier put option, the theoretical solution under the Black-Sholes model is inducted here:
Assuming $m_t^T = \min_{t \le u \le T} S_u$, Bl_{put} is put option price under BS model,

$$\delta_{\pm}^{\tau}(s) = \frac{1}{\sigma\sqrt{\tau}}\left(\log s + \left(r \pm \frac{\sigma^2}{2}\right)\tau\right)$$

$$\begin{aligned}
&\mathrm{e}^{-(T-t)r}\mathbb{E}^*\left[(K-S_T)^+ \mathbb{I}_{\{m_t^T<B\}} \mid \mathcal{F}_t\right] \\
&= \mathbb{I}_{\{m_0^t \leqslant B\}}\mathrm{Bl}_{\text{put}}\ (S_t, K, r, T-t, \sigma) - S_t\mathbb{I}_{\{m_0^t>B\}}\Phi\left(-\delta_+^{T-t}\left(\frac{S_t}{B}\right)\right) \\
&+ B\mathbb{I}_{\{m_0^t>B\}}\left(\frac{B}{S_t}\right)^{2r/\sigma^2}\left(\Phi\left(\delta_+^{T-t}\left(\frac{B^2}{KS_t}\right)\right) - \Phi\left(\delta_+^{T-t}\left(\frac{B}{S_t}\right)\right)\right) \\
&+ \mathrm{e}^{-(T-t)r}K\mathbb{I}_{\{m_0^t>B\}}\Phi\left(-\delta_-^{T-t}\left(\frac{S_t}{B}\right)\right) \\
&- \mathrm{e}^{-(T-t)r}K\mathbb{I}_{\{m_0^t>B\}}\left(\frac{S_t}{B}\right)^{1-2r/\sigma^2}\left(\Phi\left(\delta_-^{T-t}\left(\frac{B^2}{KS_t}\right)\right) - \Phi\left(\delta_-^{T-t}\left(\frac{B}{S_t}\right)\right)\right), \\
&0 \leqslant t \leqslant T
\end{aligned}$$

And here $t = 0$. After setting $\sigma = 15.7\%$, we finally can compare the different results by plotting them in Figure 16.

	Local volatility	Constant volatility	Constant volatility(Theoretical price)
Barrier			
60	0.227615	0.039294	0.047244
70	1.229601	0.704472	0.705837
80	3.507021	3.097747	3.104249
90	5.457144	5.420689	5.431394

Table 2: Prices of down-and-in barrier put option under Different models

	Implied volatility
Barrier	
60	0.182818
70	0.173016
80	0.164107
90	0.157582

Table 3: Implied volatility of down-and-in barrier put option under local volatility model.

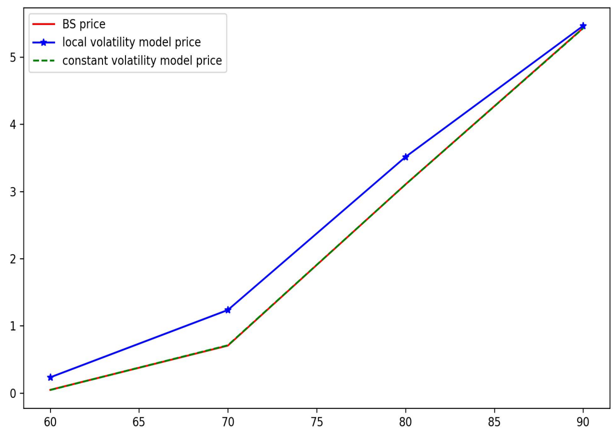

Figure 16: Comparison of the down-and-in barrier put option prices over different knock-in levels under both the local volatility model and a constant volatility model

Based on the calculated implied volatilities(see table 3) on this down-and-in put option, it can be observed that the first three terms is greater than 15.7%, whereas the result of the last term is approximately equal to 15.7%. Therefore, assuming a volatility of 15.7%, the prices predicted by the Black-Scholes (BS) model for barrier levels of 60, 70, and 80 are all lower than those generated by our local volatility model. This is because higher volatility increases the future fluctuation of the underlying asset, thereby increasing the potential for profit for the option holder, and hence the price of the option also increases.