PRICING OPTIONS USING MONTE CARLO SIMULATION AND LATTICE

XUANYI DUAN
LINGQI KONG
FAN ZHU

CONTENTS

1. INTRODUCTION

The objective of this report is to evaluate the prices of different options. An options contract is a type of derivatives that applies to an underlying asset, like stocks, enables the contract holder to have the right but not the obligation to buy or sell an asset under specified terms. Therefore, options are asymmetric because they benefit one party over the other. An option has 3 configurations: the underlying asset, strike price, and maturity. The strike price is the price at which the asset can be purchased when exercising the option. Maturity specifies the expiration date of the option. Furthermore, the two basic types of options are call option and put option. A call option gives the right to purchase something, whereas a put option gives the right to sell something. This report mainly emphasizes the pricing of Asian call and put options, lookback call and put options, floating lookback call and put options, and American put options. Two different methods are tried for the valuation of these options: the Monte Carlo Simulation approach and the Lattice approach.

The implements of these two approaches are through applying software Python programming. Rationales and details of each approach are demonstrated as below.

2. SIMULATING A STOCK PRICE PATH

Brownian motion is the random motion of microscopic particles suspended in liquids or gases caused by the impact of molecules of the surrounding medium, the particle moves randomly in d-dimensional space with small displacement at any time step. This motion is named after the Scottish botanist Robert Brown, who first discovered the phenomena while looking at the pollen immersed in water under the microscope. Later in 1918, the rigorous mathematical formulation of the theory of Brownian motion was developed by Norbert Wiener, so it is often also called the Wiener process.

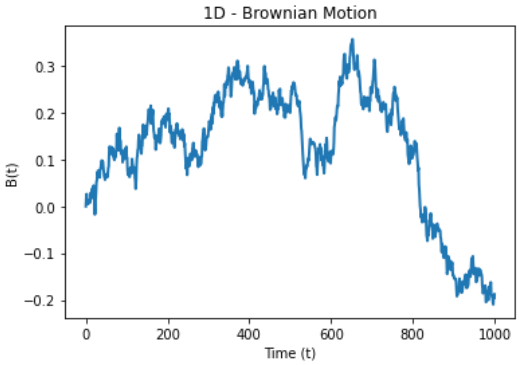


Figure 1

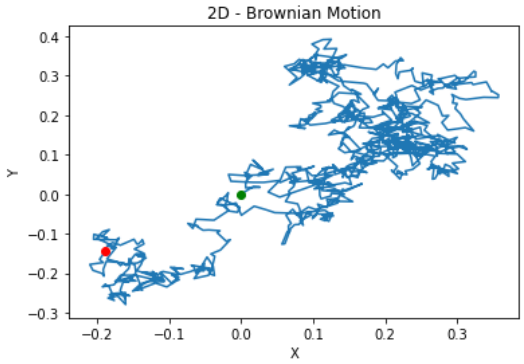


Figure 2

Figure 1 & 2 shows the linear and 2-dimensional Brownian motion. The pattern of linear Brownian motion looks very similar to the trend of the stock price, which arouses people's interest in using it to explain the movement of stock price.

Definition A real-valued stochastic process $\{B(t): t \geq 0\}$ is called a (linear) **Brownian motion** with start in $x \in R$ if the following holds:

- $B(0) = x$
- The process has independent increments, i.e. for all times $0 \leq t_1 \leq t_2 \leq \cdots \leq t_n$ the increments $B(t_n) - B(t_{n-1})$, $B(t_{n-1}) - B(t_{n-2})$, … , $B(t_2) - B(t_1)$ are independent random variables
- For all $t \geq 0$ and $h > 0$, the increments $B(t+h) - B(t)$ are normally distributed with expectation zero and variance h
- Almost surely, the function $B(t)$ is continuous

$\{B(t): t \geq 0\}$ is a **standard Brownian motion** if $x = 0$.
(Mörters & Peres, 2010)

The increment $B(t+\Delta t)-B(t)$ during a small amount of time Δt follows a normal distribution $N(0,\Delta t)$, the variance increases as the time interval increases. The increments that occur in disjoint time intervals are independent, which illustrates the Brownian motion is a Markov process. The Markov process is a stochastic process where the future is irrelevant to the past, the current market price contains all the information needed to predict its future price, which is consistent with the Weak Form Efficiency Market Hypothesis.

Consider Brownian motion with a drift term r and scale term σ, where r is the risk-free rate and σ is volatility.

$$dS_t = rd_t + \sigma dB(t)$$

The trajectory fluctuates around zero in Figure 1. The Brownian motion may not be a good choice to describe the movement of stock price, since the price of the stock cannot be negative. The price can be converted into the return, and S_t follows a **geometric Brownian motion**

$$\frac{dS_t}{S_t} = rd_t + \sigma dB(t)$$
$$dS_t = rS_t d_t + \sigma S_t dB(t)$$

Theorem Suppose that the sequence of partitions

$$0 = t_0^{(n)} \le t_1^{(n)} \le \cdots \le t_{k(n)-1}^{(n)} \le t_{k(n)}^{(n)} = t$$

is nested, i.e. at each step one or more partition points are added, and the mesh

$$\Delta(n) := \sup_{1\le j\le k(n)} \{t_j^{(n)} - t_{j-1}^{(n)}\}$$

converges to zero. Then almost surely,

$$\lim_{n\to\infty} \sum_{j=1}^{k(n)} \left(B\left(t_j^{(n)}\right) - B\left(t_{j-1}^{(n)}\right)\right)^2 = t$$

And therefore Brownian motion is of unbounded variation.
(Mörters & Peres, 2010)

Brownian motion is nowhere differentiable, which was shown by Paley, Wiener, and Zygmund in 1933 (Paley, Wiener, & Zygmund, 1933). Figure 1 also made this point, the function is continuous but not differentiable. Non-differentiable makes it difficult to solve for the price by applying the calculus. But thanks to the **Ito's Lemma** and Taylor series expansions, which provide a possible solution.

For given diffusion $X(t,w)$: $dX(t,w) = r(t,w)d_t + \sigma(t,w)dB_t$

Ito's lemma can be derived by forming Taylor series expansion. The theorem states the quadratic variation of Brownian motion is not zero, only keep the first-order term and second-order term $dB_t \times dB_t$. Other terms will be ignored.

$$df\big(t, X(t,w)\big) = \frac{\partial f}{\partial t}dt + \frac{\partial f}{\partial X}dX(t,w) + \frac{1}{2}\frac{\partial^2 f}{\partial X^2}\big(dX(t,w)\big)^2$$

$$= \left[\frac{\partial f}{\partial t} + \frac{\partial f}{\partial X}r(t,w) + \frac{1}{2}\frac{\partial^2 f}{\partial X^2}\sigma^2(t,w)\right]d_t + \frac{\partial f}{\partial x}\sigma(t,w)dB_t$$

Consider $f(t, S_t) = In(S_t)$

$$d\ln(\mathrm{S_t}) = \left(r - \frac{\sigma^2}{2}\right)dt + \sigma\sqrt{t}Z$$

$$S_t = S_0 e^{\left(r-\frac{\sigma^2}{2}\right)t+\sigma\sqrt{t}Z}$$

(Sadr, 2009)

The stock price can be described by the formula $S_t = S_0 e^{\left(r-\frac{\sigma^2}{2}\right)t+\sigma\sqrt{t}Z}$

Figure 3 shows a simulated path for the stock price. Simulation with initial stock price $s_0 = 100$, risk-free rate $r = 2\%$, volatility $\sigma = 25\%$, maturity is $T = 1/6$ year from now, unit time $\frac{1}{6000}$ year, number of steps 1000.

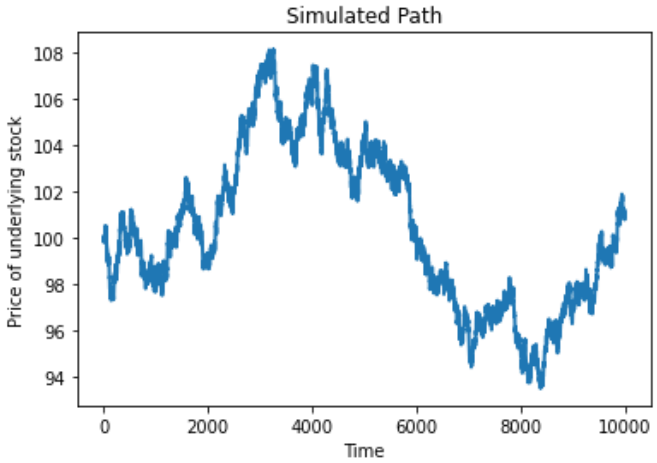


Figure 3

3. MONTE CARLO SIMULATION

Monte Carlo simulation can be applied to calculate the price of financial derivatives including options. The first step is always to generate stock price paths. Running enough simulations is sufficient, more paths will contain more possible scenarios and will lead to reliable results. Figure 4 shows 95% Confidence Intervals of the estimated option price, the range of confidence interval become smaller as the number of simulated paths increased. The confidence interval can be computed

$$\bar{h} \pm 1.96 \frac{s}{\sqrt{n}}$$

where n is the number of stock price paths, $\bar{h}$ is the estimate of the option price, s is the sample variance.

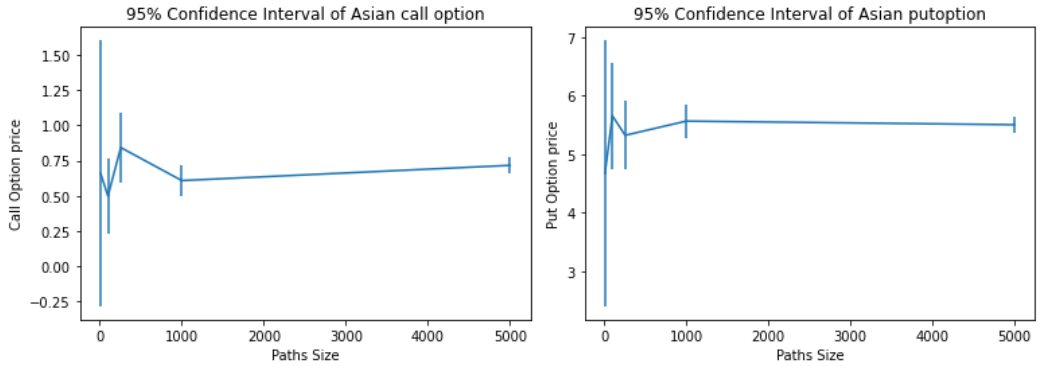


Figure 4

The project states maturity is 2 months, and the **lattice** has a unit time of 1 week. The stock price will either decrease or increase at any node, so there are $2^8 = 256$ possible paths. Set the number of simulation paths to 256 to make it consistent.

Setting all required parameters for part 1 of this project.

```
r = 0.02          # risk-free rate
S0 = 100          # current price of the underlying stock
sigma = 0.25      # volatility
K = 105           # strike price at expiry
T = 1/6           # years to expiry (2 months)
numSteps = 8      # number of steps
numPaths = 256    # number of sample paths
```

Simulate 256 stock price paths $S_t(w_1)$ ... $S_t(w_{256})$. Use the same set of stock price paths to compute the price for Asian Option, Lookback Option, and Floating lookback Option. This step allows us to compare prices of different options under the same circumstance, usually, it is not necessary.

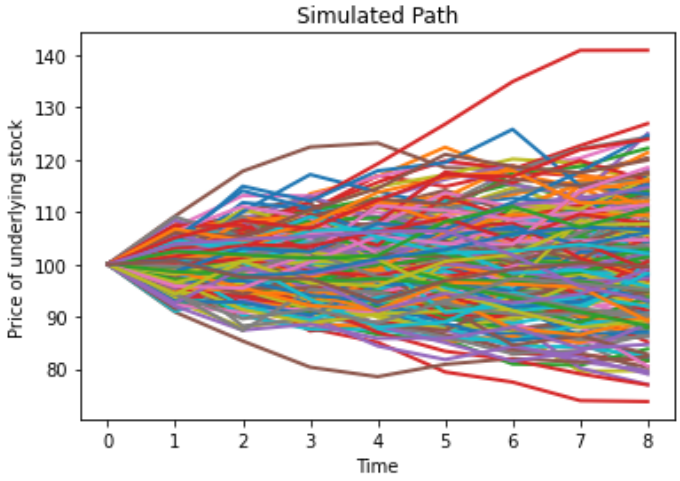


Figure 5

Pseudo-code (generate paths)

dT = T / numSteps # dT is the time increment (in years)
For $i = 1, \dots,$ numPaths: # Generate paths
$S_{t_0}(w_i) = S_0$ # For path $S_{t_0}(w_i)$, initial stock price is S_0
For $j = 1, \dots,$ numSteps:

$$S_{t_{j+1}}(w_i) = S_{t_j}(w_i) e^{\left(r - \frac{\sigma^2}{2}\right) dT + \sigma\sqrt{dT}Z}$$

ASIAN OPTION, FIXED & FLOATING LOOKBACK OPTION

Options can be divided into vanilla (European and American) options and exotic options. The latter one is more complex than the traditional one. Exotic options including the Asian option, lookback option, and floating lookback option, are all path-dependent, which means the payoff depends on the historical stock price. The payoff formulas are defined in the following table. Assuming no early exercise and no dividend-paying, the next step is to compute the option price.

Option Type	Payoff
European call	$\max(S_T - K, 0)$
European put	$\max(K - S_T, 0)$
Asian call	$\max(\bar{S} - K, 0)$
Asian put	$\max(K - \bar{S}, 0)$
Fixed Lookback call	$\max(S_{max} - K, 0)$
Fixed Lookback put	$\max(K - S_{min}, 0)$
Floating Lookback call	$\max(S_T - S_{min}, 0)$
Floating Lookback put	$\max(S_{max} - S_T, 0)$

Pseudo-code (compute option price)

For $i = 1, \ldots,$ numPaths:
Apply formulas to compute payoff for Call & Put according to the option type
$h_i = \cdots$
$\widetilde{h_i} = h_i \cdot e^{-rT}$ # Discount back

Estimate option price
$\bar{h} = \frac{1}{\text{numPaths}} \sum_{i=1}^{\text{numPaths}} \widetilde{h_i}$

Compute the 95% CI (explained in the previous part)

The output of Python Implementation

```
MC price of an Asian call option =  0.7251
MC price of an Asian put option =  5.4702

MC 95% CI of an Asian call option =  [0.4695 0.9806]
MC 95% CI of an Asian put option =  [4.8871 6.0532]

MC price of a Fixed Lookback call option =  3.6995
MC price of a Fixed Lookback put option =  11.0112

MC 95% CI of a Fixed Lookback call option =  [3.0203 4.3787]
MC 95% CI of a Fixed Lookback put option =  [10.3084 11.714 ]

MC price of a Floating Lookback call option =  6.9279
MC price of a Floating Lookback put option =  6.0127

MC 95% CI of a Floating Lookback call option =  [6.0633 7.7925]
MC 95% CI of a Floating Lookback put option =  [5.2856 6.7397]
```

For the Asian option, the payoff is the difference between the average historical stock price and the strike price, which is less volatile than European and American options. Therefore, the Asian option is less expensive than the vanilla option. In addition, the averaging feature help to prevent possible market manipulation, since it is hard to control the price all the time.

For the Fixed Lookback option, the payoff is the difference between the best beneficial historical stock price and the predetermined strike price K. On the other hand, the Floating Lookback option using floating strike price rather than fixed-price K. Floating Lookback call option sets the strike at the minimum value of the price path, put option set the strike at the maximum value of the price path. The lookback option allows the holder to look back and choose the optimal payoff over the lifetime. The price of the Lookback option is usually larger than the Asian option.

AMERICAN PUT OPTION

American put differ from European option in a way that it could be executed early. The buyer's right to the American option is relatively large. The seller experience higher risk. Therefore, under the same conditions as the European option, the price of American options is relatively high.

The main problem of the American option is to find the appropriate stop point between [t, T] and thus the value of the American option at Time t could be defined as:

$$Zt = (K - St) +$$

We will use a simple log-normal random walk defined as

$$dS(t) = \mu S(t)d_t + \sigma S(t)\, dz$$

Simulate N numbers of length M+1 random stock price. Enter the stock prices on all paths at a certain time, the corresponding option price P, strike price K

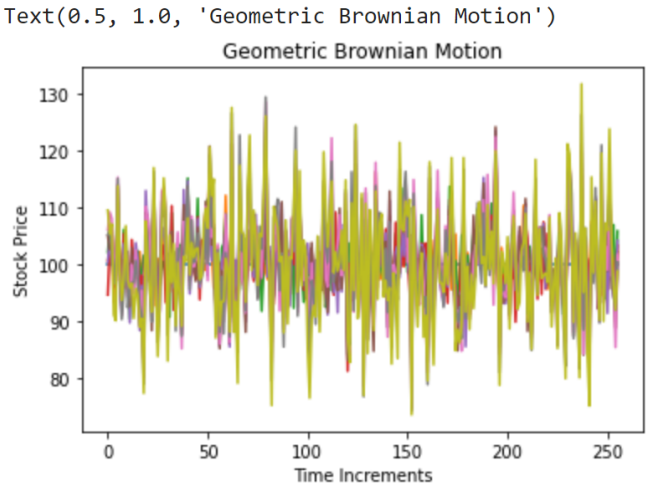


Figure 6

According to the discretization process of stock price, N stock price change paths are sampled and saved. Determine the option price at the end of each path.

The option price on each path is discounted to the previous time node so that each stock price at this time node corresponds to a discounted reference option price.

The stock price and reference option price on each path at that time are composed of (stock price, reference option price) pairs, and then all pairs are sorted from small to large according to stock prices.

Define a function that evaluates optimal boundary with the best execution price of the American put is named cut-off. This function returns the option price (corresponding to the order of input stock price) after applying the cut-off, also returns the cut-off value.

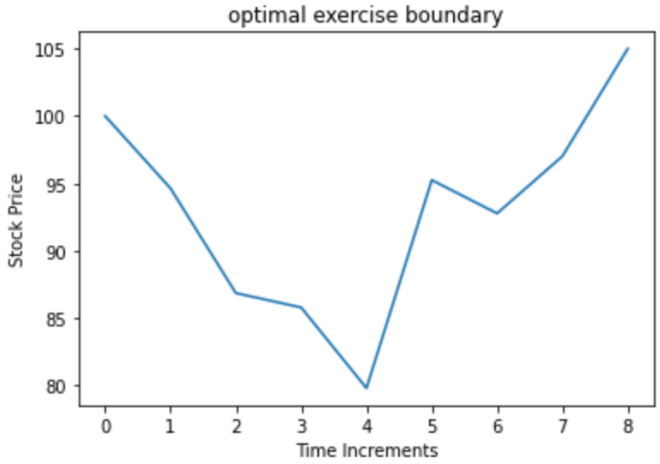


Figure 7

The exercise boundary means that there is an exercise price at any time. When the price is lower than the price, we exercise the option.

Let the variable cumulative price total = 0 and the maximum cumulative price total_ max = 0, cut off = 0. Use the backward method to calculate the end of the sequence first then calculate one before step by step, add the reference option price to the total each time, subtract max (k-stock price, 0), and then judge whether the total is greater than the total_ max, if greater than, then total_ max is updated to total, and the value cut-off is updated to the share price of the pair (stock price, reference option price).

After traversing the price pair, the cut-off is the optimal execution boundary price. We update the reference option price corresponding to the stock price on each path at that time. If the stock price is greater than or equal to the cut-off, the option price remains the reference option price, otherwise, the option price is updated to max (k-stock price, 0).

Repeat till the initial time. The average option price is the American put option price calculated by Monte Carlo simulation.

We assume the risk-free rate is 2%, the current price of the underlying stock is $100, and volatility 25%, with no dividend payments. The strike price is $105 and the maturity 2 months. The simulation has the unit time = 1 week, thus with 60days (2 months*30 days) / 7 days approximately equal to 8. So set the number of steps to be 8. And with the number of simulations be 256.
The American put option price is 7.0912.

4. LATTICE

ASIAN OPTION

The payoff of an Asian option is determined by the average values of a particular stock price path over the lifetime of the stock. The payoff of an Asian call option is max($\bar{S}$-K,0), where $\bar{S}$ denotes the average values of a particular stock price path over the lifetime of the stock and K is the strike price. Furthermore, the payoff of an Asian put option is max(K-$\bar{S}$,0). We will apply the lattice approach to evaluate the Asian option price and implement it via python.

The parameters applied for the lattice approach are:
$u = e^{\sigma\sqrt{\Delta t}}$, u is the amount the current stock price can go up.
$d = e^{-\sigma\sqrt{\Delta t}}$, d is the amount the current stock price can go down, and $d = \frac{1}{u}$.
$p = \frac{e^{r\Delta t}-d}{u-d}$, p is the probability the stock price is going up. Additionally, 1-p is the probability of a downward movement.
σ represents the volatility.
$\Delta\, t = \frac{T}{m}$ is the time between observations, m denotes the number of steps.
r denotes the risk-free interest rate compounded continuously.
At each time step t_i, the stock price value will be $S_{t_i} = Su^j d^{i-j}$,
j=0,…,i,and j indicates the number of upward movement.

In our problem, we have r=2%, S = \$100, σ = 25%, K = %105, T = $\frac{2}{12}$, m = 8.

The first stage is to build a lattice. For our problem, there are 8 steps, so we have 9 layers (including S=100, at t=0) and 2^8 possible paths. Then, we build a 9 by 256 matrix, and each row contains the possible stock prices (S_{t_i}). We can also create another 9 by 256 matrices with each row contains the possible probabilities of payoffs. The two matrices are as shown below. After that, each column of the matrix represents a particular stock price path. This enables us to calculate the option price at maturity by taking the average of each column, computing payoff. For a call option, its payoff is max(paths.mean(axis=0)-K, 0) and that of a put option is max(K-paths.mean(axis=0), 0). The last stage is to discount the payoff at maturity to the current price. Eventually, the current option price is determined. Back to our problem, the current price of the Asian call option is 0.6925 and that of the Asian put option is 5.509

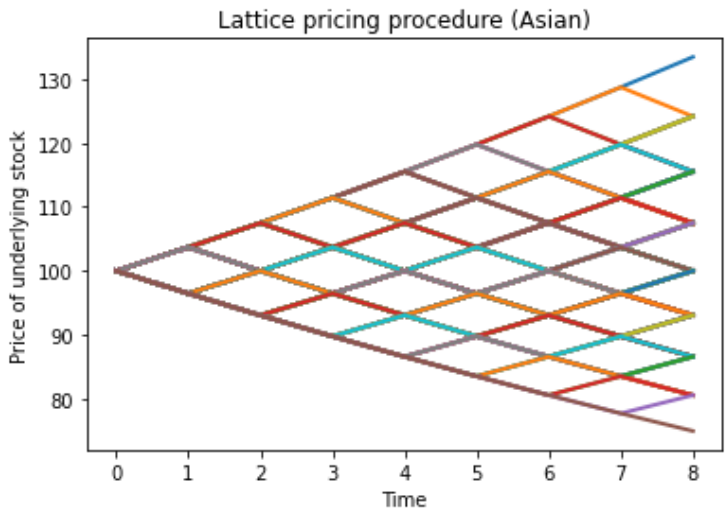


Figure 8

```
Stock price paths matrix
[[100.    100.    100.    ... 100.    100.    100.   ]
 [103.674 103.674 103.674 ...  96.456  96.456  96.456]
 [107.483 107.483 107.483 ...  93.037  93.037  93.037]
 ...
 [124.172 124.172 124.172 ...  80.533  80.533  80.533]
 [128.734 128.734 119.771 ...  83.492  77.679  77.679]
 [133.464 124.172 124.172 ...  80.533  80.533  74.926]]
Probability of payoff matrix
[[1.    1.    1.    ... 1.    1.    1.   ]
 [0.497 0.497 0.497 ... 0.503 0.503 0.503]
 [0.247 0.247 0.247 ... 0.253 0.253 0.253]
 ...
 [0.015 0.015 0.015 ... 0.016 0.016 0.016]
 [0.007 0.007 0.008 ... 0.008 0.008 0.008]
 [0.003 0.004 0.004 ... 0.004 0.004 0.004]]
```

FIXED LOOKBACK OPTION:

The lookback option is a type of path-dependent option whose price depends on the historical maximum or minimum stock price. The payoff of a lookback call option is max(0, S_{max} - K). S_{max} is the maximum stock price within the maturity time. The payoff of a lookback put option is max(0, K-S_{min}). Here we will introduce how to calculation lookback option price through the lattice approach.

For the European option, we know that there exists an option price that corresponds to the relevant strike price on each node of the lattice. However, unlike the European option, the lookback option does not obtain an option price on each node that corresponds to each stock price. For the lookback option, only the node that has the probability to exist maximum or minimum stock price has a corresponding option value. Therefore, at each node, we first compute the possible maximum or minimum stock price, then stock it at that node. After that, at the terminal of the lattice, there will exist maximum or minimum stock price on the nodes, then we can calculate the corresponding option price. Eventually, we get the option price at t=0 via discounting the option price at t=T to t=0.

We apply Python to implement the lattice approach as well. Same as the Asian option, our first step is to build a lattice that describes the change of stock price. We define 'tree' as a list, and denote 'row' as another list. The number of layers i is from 0 to N. The number of nodes at each layer j is from 0 to i. Then we build a dictionary called node to store the stock price, and describe the information of maximum and minimum stock price on each node. Defining 'S' as a key of the dictionary 'node' and storing the stock price in 'S' ($S_{t_i} = Su^j d^{i-j}$). 'Smax_val' and 'Smin_val' are the other two keys of the dictionary 'node', and we assign two dictionaries to them respectively. Their keys are the number of times that S times d and values are the corresponding option values. If S times d once, the key of 'Smin_val' is 1, if S times u once, the key of 'Smax_val' is -1, because of the relation $d = \frac{1}{u}$.

Furthermore, all possible minimum historical stock price at the i^{th} layer and the j^{th} node is between $S*d^{i-j}$ and $\min(S, S*d^{i-2j})$. Since at i^{th} layer and j^{th} node, there are j upward movements and i-j downward movements. Then, the lowest historical price of the path that reaches this node should have a minimum value of $S*d^{i-j}$. The largest value of the minimum value should be $\min(S, S*u^j*d^{i-j})$, which is equivalent to $\min(S, S*d^{i-2j})$, because $d = \frac{1}{u}$. Likewise, all possible minimum historical stock prices at i^{th} layer and j^{th} node should be from $\max(S, S*d^{i-2j})$ to $S*d^{-j}$. Since the highest historical price of the path that reaches this node should have a maximum value of $S*u^j = S*d^{-j}$. The minimum value of the historically highest price should be $\max(S, S*d^{i-2j})$. Also, for any nodes on the tree, the maximum or minimum value of the stock price of the path to reach this node can be described as $S*d^k$, where k is an integer ranging from 0 to i-j-max(0,i-2*j) for minimum value and from 0 to j-max(0,2*j - 1) for maximum value. Part of the lattice tree we build is presented below. Tree[0] corresponds to the initial layer and tree[N] reflects the terminal layer. Tree[N][j] indicates the j^{th} node at the N^{th} layer.

```
The initial layer of the tree, tree[0]
[{'S': 100.0, 'Smin_val': {0: None}, 'Smax_val': {0: None}}]
The terminal layer of the tree, tree[8]
[{'S': 74.926, 'Smax_val': {0: None}, 'Smin_val': {8: None}},
 {'S': 80.533,
  'Smax_val': {-1: None, 0: None},
  'Smin_val': {6: None, 7: None}},
 {'S': 86.56,
  'Smax_val': {-2: None, -1: None, 0: None},
  'Smin_val': {4: None, 5: None, 6: None}},
 {'S': 93.037,
  'Smax_val': {-3: None, -2: None, -1: None, 0: None},
  'Smin_val': {2: None, 3: None, 4: None, 5: None}},
 {'S': 100.0,
  'Smax_val': {-4: None, -3: None, -2: None, -1: None, 0: None},
  'Smin_val': {0: None, 1: None, 2: None, 3: None, 4: None}},
 {'S': 107.484,
  'Smax_val': {-5: None, -4: None, -3: None, -2: None},
  'Smin_val': {0: None, 1: None, 2: None, 3: None}},
 {'S': 115.527,
  'Smax_val': {-6: None, -5: None, -4: None},
  'Smin_val': {0: None, 1: None, 2: None}},
 {'S': 124.173,
  'Smax_val': {-7: None, -6: None},
  'Smin_val': {0: None, 1: None}},
 {'S': 133.466, 'Smax_val': {-8: None}, 'Smin_val': {0: None}}]
```

```
The 8th layer and 6th node of the tree, tree[8][6]
{'S': 115.527, 'Smin_val': {2: None, 1: None, 0: None}, 'Smax_val': {-6: None, -5: None, -4: None}}
```

After building the lattice and obtaining the historically maximum or minimum stock price at the corresponding node, we can compute the option price at maturity, which is the option price on the terminal layer. The payoff for the lookback call option at maturity is max(0, S*(d**number of d)-k), and number of d is the keys of 'Smax_val' at the terminal layer. The payoff for the lookback put option at maturity is max(0, k- S*(d**number of d)), and number of d is the keys of 'Smin_val' at the terminal layer. After obtaining the option price at maturity, we discount them to the current price. Assuming we are at layer i and node j, and to calculate call price.
Then, there is a probability of p that the stock price increases and call price at layer i+1 and node j+1 is tree[i+1][j+1]['Smax_val'][min(num_d, i-2*j-1)]. Also, there is a (1 - p) % probability the stock price becomes layer i+1 and node j with a call price of tree[i+1][j]['Smax_val'][num_d]. After that, we discount the expected call price by $e^{-r\Delta t}$ and get the call price at layer i and node j. We repeat this process until reach the initial node and get the call price at t=0. In our problem, we get a call price of $3.575. For a put option, the probability of an increase in the stock price is also p and the put price at layer i+1 and node j+1 is tree[i+1][j+1]['Smin_val'][num_d]. The probability the stock price is at layer i+1 and node j is (1-p) with a put price of tree[i+1][j]['Smin_val'][max(num_d, i-2*j+1)]. Same as the call option, we then discount the expected put price by $e^{-r\Delta t}$ and obtain the put price at layer i and node j. Repeating this process, and eventually get put price at t=0. For our problem, the put price at t=0 is $11.2485.

```
Lookback call price
3.574
Lookback Call option price at maturity
[{'S': 74.926, 'Smax_val': {0: 0}, 'Smin_val': {8: None}},
 {'S': 80.533, 'Smax_val': {-1: 0, 0: 0}, 'Smin_val': {6: None, 7: None}},
 {'S': 86.56,
  'Smax_val': {-2: 2.484, -1: 0, 0: 0},
  'Smin_val': {4: None, 5: None, 6: None}},
 {'S': 93.037,
  'Smax_val': {-3: 6.433, -2: 2.484, -1: 0, 0: 0},
  'Smin_val': {2: None, 3: None, 4: None, 5: None}},
 {'S': 100.0,
  'Smax_val': {-4: 10.527, -3: 6.433, -2: 2.484, -1: 0, 0: 0},
  'Smin_val': {0: None, 1: None, 2: None, 3: None, 4: None}},
 {'S': 107.484,
  'Smax_val': {-5: 14.772, -4: 10.527, -3: 6.433, -2: 2.484},
  'Smin_val': {0: None, 1: None, 2: None, 3: None}},
 {'S': 115.527,
  'Smax_val': {-6: 19.173, -5: 14.772, -4: 10.527},
  'Smin_val': {0: None, 1: None, 2: None}},
 {'S': 124.173,
  'Smax_val': {-7: 23.736, -6: 19.173},
  'Smin_val': {0: None, 1: None}},
 {'S': 133.466, 'Smax_val': {-8: 28.466}, 'Smin_val': {0: None}}]
```

```
Lookback put price
11.248
Lookback Put option price at maturity
[{'S': 74.926, 'Smax_val': {0: None}, 'Smin_val': {8: 30.074}},
 {'S': 80.533,
  'Smax_val': {-1: None, 0: None},
  'Smin_val': {6: 24.467, 7: 27.321}},
 {'S': 86.56,
  'Smax_val': {-2: None, -1: None, 0: None},
  'Smin_val': {4: 18.44, 5: 21.508, 6: 24.467}},
 {'S': 93.037,
  'Smax_val': {-3: None, -2: None, -1: None, 0: None},
  'Smin_val': {2: 11.963, 3: 15.26, 4: 18.44, 5: 21.508}},
 {'S': 100.0,
  'Smax_val': {-4: None, -3: None, -2: None, -1: None, 0: None},
  'Smin_val': {0: 5.0, 1: 8.544, 2: 11.963, 3: 15.26, 4: 18.44}}
 {'S': 107.484,
  'Smax_val': {-5: None, -4: None, -3: None, -2: None},
  'Smin_val': {0: 5.0, 1: 8.544, 2: 11.963, 3: 15.26}},
 {'S': 115.527,
  'Smax_val': {-6: None, -5: None, -4: None},
  'Smin_val': {0: 5.0, 1: 8.544, 2: 11.963}},
 {'S': 124.173,
  'Smax_val': {-7: None, -6: None},
  'Smin_val': {0: 5.0, 1: 8.544}},
 {'S': 133.466, 'Smax_val': {-8: None}, 'Smin_val': {0: 5.0}}]
```

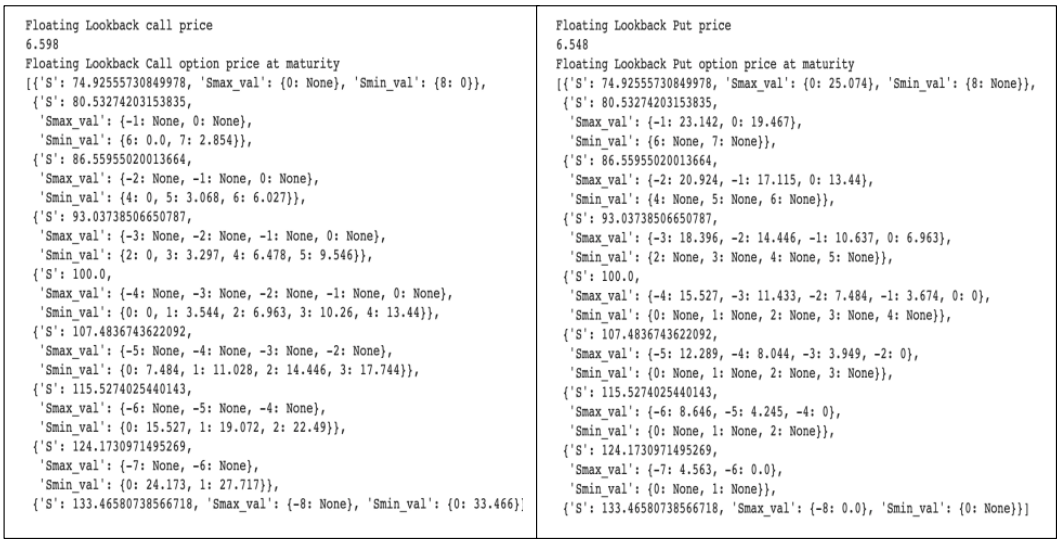
Floating Lookback Option:

Unlike the lookback option, the payoff of the floating lookback call option is max(0, S_T - S_{min}). The payoff of the lookback put option is max(0, S_{max} -S_T). S_{min} and S_{max} represent the minimum and maximum values of a particular stock price path during the life of the stock. S_T is the stock price at t = T. We can still use the lattice built for the lookback option. However, the way to calculate option price at the terminal layer and to discount option price is different.

The payoff for the floating lookback call option at maturity is max(0, node['S']-S*(d**number of d)). The number of d is the keys of 'Smin_val' at the terminal layer and node['S'] denotes S_T. The payoff for the lookback put option at maturity is max(0, S*(d**number of d)-node['S']), and number of d is the keys of 'Smax_val' at the terminal layer. The next stage is discounting. First, assuming we are at layer i and node j, and to calculate floating lookback call price. Then, there is a probability of p that the stock price increases and that call price at layer i+1 and node j+1 is tree[i+1][j+1]['Smin_val'][num_d]. Also, there is a (1 - p) % probability the stock price becomes layer i+1 and node j with a call price of tree[i+1][j]['Smin_val'][max(num_d, i-2*j+1)]. After that, we discount the expected floating lookback call price by $e^{-r\Delta t}$ and get the call price at layer i and node j. Repeating this process, until reaching the initial node and get the call price at t=0. In our problem, we get a floating lookback call price of $6.598.

For a put option, the probability of an increase in the stock price is also p and the floating lookback put price at layer i+1 and node j+1 is
tree[i+1][j+1]['Smax_val'][min(num_d, i-2*j-1)].

The probability the stock price is at layer i+1 and node j is (1-p) with a floating lookback put price of tree[i+1][j]['Smax_val'][num_d]. Then, discounting the expected put price by $e^{-r\Delta t}$ and obtain the put price at layer i and node j. Repeating this process, and eventually get put price at t=0. For our problem, the floating lookback put price at t=0 is $6.5478.



```
Floating Lookback call price
6.598
Floating Lookback Call option price at maturity
[{'S': 74.92555730849978, 'Smax_val': {0: None}, 'Smin_val': {8: 0}},
 {'S': 80.53274203153835,
  'Smax_val': {-1: None, 0: None},
  'Smin_val': {6: 0.0, 7: 2.854}},
 {'S': 86.55955020013664,
  'Smax_val': {-2: None, -1: None, 0: None},
  'Smin_val': {4: 0, 5: 3.068, 6: 6.027}},
 {'S': 93.03738506650787,
  'Smax_val': {-3: None, -2: None, -1: None, 0: None},
  'Smin_val': {2: 0, 3: 3.297, 4: 6.478, 5: 9.546}},
 {'S': 100.0,
  'Smax_val': {-4: None, -3: None, -2: None, -1: None, 0: None},
  'Smin_val': {0: 0, 1: 3.544, 2: 6.963, 3: 10.26, 4: 13.44}},
 {'S': 107.4836743622092,
  'Smax_val': {-5: None, -4: None, -3: None, -2: None},
  'Smin_val': {0: 7.484, 1: 11.028, 2: 14.446, 3: 17.744}},
 {'S': 115.5274025440143,
  'Smax_val': {-6: None, -5: None, -4: None},
  'Smin_val': {0: 15.527, 1: 19.072, 2: 22.49}},
 {'S': 124.1730971495269,
  'Smax_val': {-7: None, -6: None},
  'Smin_val': {0: 24.173, 1: 27.717}},
 {'S': 133.46580738566718, 'Smax_val': {-8: None}, 'Smin_val': {0: 33.466}]
```

```
Floating Lookback Put price
6.548
Floating Lookback Put option price at maturity
[{'S': 74.92555730849978, 'Smax_val': {0: 25.074}, 'Smin_val': {8: None}},
 {'S': 80.53274203153835,
  'Smax_val': {-1: 23.142, 0: 19.467},
  'Smin_val': {6: None, 7: None}},
 {'S': 86.55955020013664,
  'Smax_val': {-2: 20.924, -1: 17.115, 0: 13.44},
  'Smin_val': {4: None, 5: None, 6: None}},
 {'S': 93.03738506650787,
  'Smax_val': {-3: 18.396, -2: 14.446, -1: 10.637, 0: 6.963},
  'Smin_val': {2: None, 3: None, 4: None, 5: None}},
 {'S': 100.0,
  'Smax_val': {-4: 15.527, -3: 11.433, -2: 7.484, -1: 3.674, 0: 0},
  'Smin_val': {0: None, 1: None, 2: None, 3: None, 4: None}},
 {'S': 107.4836743622092,
  'Smax_val': {-5: 12.289, -4: 8.044, -3: 3.949, -2: 0},
  'Smin_val': {0: None, 1: None, 2: None, 3: None}},
 {'S': 115.5274025440143,
  'Smax_val': {-6: 8.646, -5: 4.245, -4: 0},
  'Smin_val': {0: None, 1: None, 2: None}},
 {'S': 124.1730971495269,
  'Smax_val': {-7: 4.563, -6: 0.0},
  'Smin_val': {0: None, 1: None}},
 {'S': 133.46580738566718, 'Smax_val': {-8: 0.0}, 'Smin_val': {0: None}}]
```

Lattice result summary:

```
Lattice price of an Asian call option =  0.6925
Lattice price of an Asian put option =  5.5095

Lattice price of a Fixed Lookback call option =  3.575
Lattice price of a Fixed Lookback put option =  11.2485

Lattice price of a Floating Lookback call option =  6.5979
Lattice price of a Floating Lookback put option =  6.5478
```

AMERICAN PUT OPTION:

The lattice method assumes that in a risk-neutral world, the present value of derivatives = the discount value of future expected return, and the discount rate is the risk-free rate r. The probability of the stock price rising is defined as p, and the probability of the stock price falling is 1-p. q is the risk-neural probability. The increasing range is u, and the decreasing range is d; The current time is t = 0, the current underlying asset price is S0, and the current derivative price is C; After a unit of time, the stock price will either rise to S0*u, and the corresponding derivative value will be Cu; Or down to S0*d, the corresponding derivative value is Cd.

Option value is

$$C = e^{-rT}[qC_u + (1-q)Cd]$$

Where $q=\frac{e^{rT}-d}{u-d}$

The steps are: The time t is divided into N parts, each part is dt, calculate u,d,p,q by their formula. First, find the value of the put option at maturity, then use backward induction to find the option price.

Check whether exercise early or not by Comparing the Option value of early exercise $S0d^8$ with the discount value max{K-$S0d^8$, 0}, choose the larger one.

The underlying tree of stock prices:

```
Lattice price of an American put option =  7.0322

array([[100. , 103.7, 107.5, 111.4, 115.5, 119.8, 124.2, 128.7, 133.5],
       [  0. ,  96.5, 100. , 103.7, 107.5, 111.4, 115.5, 119.8, 124.2],
       [  0. ,   0. ,  93. ,  96.5, 100. , 103.7, 107.5, 111.4, 115.5],
       [  0. ,   0. ,   0. ,  89.7,  93. ,  96.5, 100. , 103.7, 107.5],
       [  0. ,   0. ,   0. ,   0. ,  86.6,  89.7,  93. ,  96.5, 100. ],
       [  0. ,   0. ,   0. ,   0. ,   0. ,  83.5,  86.6,  89.7,  93. ],
       [  0. ,   0. ,   0. ,   0. ,   0. ,   0. ,  80.5,  83.5,  86.6],
       [  0. ,   0. ,   0. ,   0. ,   0. ,   0. ,   0. ,  77.7,  80.5],
       [  0. ,   0. ,   0. ,   0. ,   0. ,   0. ,   0. ,   0. ,  74.9]])
```

The underlying tree of put option payoffs:

```
array([[ 7. ,  4.6,  2.6,  1.2,  0.3,  0. ,  0. ,  0. ,  0. ],
       [ 0. ,  9.4,  6.6,  4.1,  2. ,  0.6,  0. ,  0. ,  0. ],
       [ 0. ,  0. , 12.2,  9.1,  6.1,  3.4,  1.3,  0. ,  0. ],
       [ 0. ,  0. ,  0. , 15.3, 12. ,  8.8,  5.5,  2.5,  0. ],
       [ 0. ,  0. ,  0. ,  0. , 18.4, 15.3, 12. ,  8.5,  5. ],
       [ 0. ,  0. ,  0. ,  0. ,  0. , 21.5, 18.4, 15.3, 12. ],
       [ 0. ,  0. ,  0. ,  0. ,  0. ,  0. , 24.5, 21.5, 18.4],
       [ 0. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. , 27.3, 24.5],
       [ 0. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. ,  0. , 30.1]])
```

Comparison of the two approaches for an American put option:

The option prices derived by using the Monte Carlo approach and Lattice approach are very close. The American put option price using the Monte Carlo approach is 7.0912.
The lattice price is 7.0322, within the 95% Confidence interval of the first method.

5. CONCLUSION

The results for the four options are consistent for both approaches. We do not know the actual outcomes, but by knowing the possible outcomes, we can find the expected present value of all future cash flows.

The calculation method of the lattice method could be summarized as the following steps: 1. Establish a binary tree model of stock price. At each node of the tree, the price could rise or fall; 2. Calculate the price of the option on the maturity date. 3. Any price of the option before the maturity date can be calculated by the backtracking method.

The advantage of the lattice approach is that it has a faster running speed than the Monte Carlo method. However, the lattice approach assumes a risk neural world, while the Monte Carlo simulation is more flexible on the risk assumptions under all parameters and thus models a range of possible outcomes.

Overall, both methods yield similar results for option pricing.

Bibliography

Mörters , P., & Peres, Y. (2010). *Brownian Motion*. New York: Cambridge University Press .

Paley, R. E., Wiener, N., & Zygmund, A. (1933). Notes on random functions. *Mathematische Zeitschrift 37*, 647-688.

Sadr, A. (2009). *Interest Rate Swaps and Their Derivatives: A Practitioner's Guide*. Wiley.