FE621 - Homework #4

Author: Sid Bhatia

Date: May 2nd, 2024

Pledge: I pledge my honor that I have abided by the Stevens Honor System.

Professor: Sveinn Olafsson

TA: Dong Woo Kim

Problem #1 (Barrier Options)

Price Formulation

The price of an **up-and-out put option/knock-out** (**UOP**) with strike price K and barrier H is given by:

$$P = e^{-rT}\mathbb{E}^Q[(K - S_T)_+ \mathbf{1}_{\{\tau > T\}}] \tag{1}$$

where τ is the *stopping time* of the asset price process $\{S_t\}_{t \geq 0}$ to the barrier H:

$$\tau = \inf\{t > 0 : S_t \geq H\} \tag{2}$$

Indicator Formulation

The **indicator function** $\mathbf{1}_{\{\tau > T\}}$ in the formula for the up-and-out put option is defined as follows:

$$\mathbf{1}_{\{\tau > T\}} = \begin{cases} 1 & \text{if } \tau > T \\ 0 & \text{otherwise} \end{cases}$$

where τ is the **stopping time** defined as the first instance when the stock price S_t reaches or exceeds the barrier level H.

Indicator Definition

The **infimum function** used to define the stopping time τ for the up-and-out put option is as follows:

$$\tau = \inf\{t > 0 : S_t \geq H\}$$

In this expression:

- τ represents the **stopping time**, the earliest time at which the stock price S_t reaches or exceeds a predetermined barrier level H.
- The set $\{t > 0 : S_t \geq H\}$ includes all times t where the stock price is greater than or equal to the barrier H.

- The function $\inf\{\cdot\}$ denotes the **infimum** of a set, which is the greatest lower bound of that set. In this case, it identifies the *smallest time value* from the set of all times where S_t is at least H.

If the set $\{t > 0 : S_t \geq H\}$ is empty (i.e., the stock price never reaches or exceeds H during the option's life), τ is considered infinite, and the indicator function $\mathbf{1}_{\{\tau > T\}}$ equals 1, implying that the option behaves like a standard put option throughout its lifetime.

Payoff

The payoff is the **same** as that of a *vanilla put option*, unless the stock price goes above H during the life of the option, in which case the payoff is **zero**. Assume the process $\{S_t\}_{t\geq 0}$ to follow a GBM.

a. Is an UOP option cheaper or more expensive than a vanilla put option? Explain.

An **up-and-out put option** (UOP) is generally cheaper than a vanilla put option. This difference in pricing comes from the **additional condition** involved in the UOP, where the option becomes worthless if the stock price exceeds the barrier H before expiration. In a vanilla put option, the holder has the right to sell the stock at the strike price K **regardless of how high the stock price has climbed during the option's life**.

This restriction in the UOP **reduces the probability** of a payout compared to a vanilla put option, where there is no upper limit on the stock price affecting the payoff.

Therefore, the UOP has a **lower premium** due to its *reduced likelihood of exercising profitably*. Essentially, the risk of the option knocking out (i.e., becoming worthless if the stock price exceeds the barrier H reduces its cost.

*b. The standard MC estimator for the price of an **UOP (put) option** is given by:*

$$\hat{P}_{n,m} = e^{-rT}\frac{1}{N}\cdot\sum_{k=1}^{N}(K - \hat{S}_m(k))^+\mathbf{1}_{\{\hat{\tau}(k)>T\}} \tag{3}$$

where $\{\hat{S}_i(k)\}_{i\geq 0}$ is the k-th simulated path of GBM at times $\{t_i\}_{i\geq 0}$ where $t_i = i\cdot\frac{T}{m}$ and

$$\hat{\tau}(k) = \inf\{i \geq 0 : \hat{S}_i(k) > H\} \tag{4}$$

is the **stopping time** of the simulated path to the barrier H.

(i) What is the definition of $\hat{P}_{n,m}$ being an unbiased/biased high/biased low estimator for P?

The definition of $\hat{P}_{n,m}$ being an **unbiased**, **biased high**, or **biased low** estimator for P relates to its **expected value** compared to the true value P:

- **Unbiased Estimator**: $\mathbb{E}[\hat{P}_{n,m}] = P$, or the estimator *equals* to the true price.
- **Biased High Estimator**: $\mathbb{E}[\hat{P}_{n,m}] > P$, or the estimator systematically *overestimates* the true price.

- **Biased Low Estimator**: $\mathbb{E}[\hat{P}_{n,m}] < P$, or the estimator systematically *underestimates* the true price.

(ii) Do you expect $\hat{P}_{n,m}$ to be biased (high/low)? Explain.

Bias Analysis of the Monte Carlo Estimator $\hat{P}_{n,m}$

Given the Monte Carlo estimator for the up-and-out put option:

$$\hat{P}_{n,m} = e^{-rT} \frac{1}{N} \cdot \sum_{k=1}^{N} (K - \hat{S}_m(k))^+ \mathbf{1}_{\{\hat{\tau}(k) > T\}} \tag{3}$$

This estimator calculates the option price by simulating stock price paths and applying the indicator function to determine if these paths exceed the barrier H before time T. The estimator's potential bias depends significantly on how well the simulation captures the dynamics of the stock price and its interaction with the barrier.

Factors Influencing Bias:

1. **Barrier Breach Underestimation**:
   - If the Monte Carlo simulation underestimates the frequency of the barrier breach (due to coarse time grid or insufficient number of paths), the result is an overestimated option price, leading to a **biased high** estimator.
2. **Barrier Breach Overestimation**:
   - Conversely, an over-sensitive simulation (possibly due to a very fine time grid) might overestimate barrier breaches, leading to a conservative estimation where the payoff is zeroed out more often than it should be, making the estimator **biased low**.
3. **Volatility and Path Dependency**:
   - The inherent variability in the GBM model and path dependency can affect the frequency of barrier breaches. Differences in how volatility is modeled and the stochastic nature of the paths can influence whether the simulated paths breach the barrier, thus impacting the bias of the estimator.

Bias Synthesis:

The bias of $\hat{P}_{n,m}$ largely depends on the fidelity of the simulation in representing the true stock price dynamics and its interaction with the barrier H. Without precise tuning of simulation parameters and thorough model validation against real-world data, there is a risk that $\hat{P}_{n,m}$ might be either biased high or biased low. Therefore, careful calibration and extensive validation of the Monte Carlo model are crucial for ensuring the accuracy of the estimator.

c. Use the following parameter table to compute the estimator $\hat{P}_{n,m}$ along with a 95% confidence interval. Use $m = 63$ and $n = 100,000$.[1]

Parameter	Symbol	Value
Initial price	S_0	50
Volatility	σ	30%
Interest rate	r	5%
Strike	K	60
Expiration	T	0.25 years
Barrier	H	55

Given the exact option price, are your simulation results consistent with your guess about the bias in part (b)-(ii)?

1. In the Black-Scholes-Merton model, there exist explicit formulas for a variety of barrier option (Bjork). The exact price of the UOP option in this problem is $6.869. ↩

Monte Carlo Simulation of Up-and-Out (KO) Put Option

The following Python code simulates the price of an up-and-out put option using the given parameters and a Monte Carlo approach. We compute the estimated option price along with a 95% confidence interval to compare against the exact price of $\mathbf{\$6.869}$. This comparison will help assess the bias discussed in part (b)-(ii):

```
import numpy as np

# Parameters
S0 = 50        # Initial stock price
K = 60         # Strike price
T = 0.25       # Time to expiration in years
r = 0.05       # Risk-free rate
sigma = 0.30   # Volatility
H = 55         # Barrier
m = 63         # Number of time steps
n = 100000     # Number of simulations

dt = T / m  # Time step size
discount_factor = np.exp(-r * T)  # Discount factor

# Simulating n paths
np.random.seed(0)  # For reproducibility
paths = S0 * np.exp(np.cumsum((r - 0.5 * sigma**2) * dt + sigma *
np.sqrt(dt) * np.random.randn(n, m), axis=1))
paths = np.hstack([np.full((n, 1), S0), paths])  # Adding the initial
price
```

```
# Calculating the payoffs
payoffs = np.maximum(K - paths[:, -1], 0)  # Payoffs at maturity
knockout = (paths > H).any(axis=1)  # Indicator for knocking out
payoffs[knockout] = 0  # Zeroing payoffs for knocked out paths

# Estimating the option price
estimated_price = discount_factor * np.mean(payoffs)
std_error = discount_factor * np.std(payoffs) / np.sqrt(n)
confidence_interval = (estimated_price - 1.96 * std_error, estimated_price
+ 1.96 * std_error)

# Display results
print(f"Estimated Price: {estimated_price:.4f}")
print(f"95% Confidence Interval: {confidence_interval}")
print(f"Exact Price: 6.869")
Estimated Price: 7.3396
95% Confidence Interval: (7.29108737732994, 7.388124622238721)
Exact Price: 6.869
```

Monte Carlo Analysis

The estimated price from the Monte Carlo simulation is $\mathbf{\$7.3396}$, which is higher than the exact price of $\$6.869$. This result falls outside the 95% confidence interval of $(7.2911, 7.3881)$, indicating that the simulation might be **biased high**. This is consistent with the earlier discussion about potential biases in the Monte Carlo estimator for the up-and-out put option.

This comparison helps us understand the reliability of the simulation and can guide adjustments in the model or parameters to reduce bias and improve accuracy.