Partial Differential Equations for Option Pricing

Sina Baghal

June 9, 2024

Exotic options, a category of derivative securities, feature complex structures and unique payoff mechanisms that differ from standard options like European and American options. These financial instruments include a wide variety of types, such as barrier options, Asian options, and lookback options, each with distinctive characteristics tailored to specific investor needs and market conditions. Pricing exotic options involves advanced mathematical models and computational techniques, often utilizing stochastic processes and partial differential equations to capture the intricate behaviors of their underlying assets. This note provides some details on the pricing of the following three types of exotic options based on geometric Brownian motion assets.

- Barrier options (e.g., up & out)
- Lookback options
- Asian options

Contents

0 Definition **1**

1 Reflection Principle **3**

2 Partial Differential Equations **4**
2.1 Barrier 4
2.1.1 Delta-hedging 5
2.2 Lookback 5
2.3 Asian 5

3 Python Codes **6**
3.1 Reflected Brownian mption 6
3.2 Barrier 8
3.3 Lookback 11
3.4 Asian 13

0 Definition

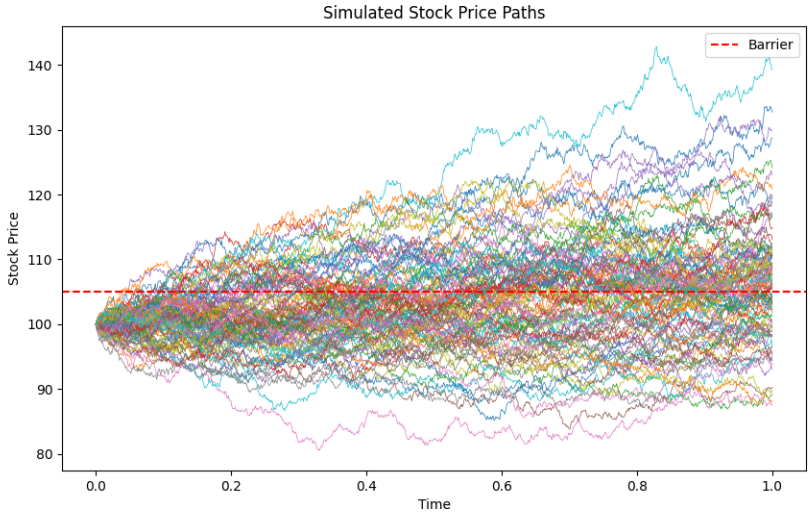
Throughout, denote the underlying asset dynamic by $S(t)$ and let

$$Y(T) := \max_{t \leq T} S(t)$$

A *Barrier option* are a type of financial derivative whose payoff depends on whether the underlying asset's price reaches a specific predetermined level, called the barrier, within a certain timeframe. If the asset's price hits this barrier, the option is either activated (knock-in) or deactivated (knock-out). These options can be either calls or puts and usually have lower premiums than standard options due to their conditional nature. For instance, an up & out barrier call option has the following payoff

$$V(T) = (S(T) - K)^+ \mathbf{1}_{Y(T) \leq b} \qquad \text{(Barrier Payoff)}$$

Plotting simulated paths generates Figure 0.



A *Lookback option* are a type of exotic financial derivative that allows the holder to "look back" over the life of the option and choose the optimal underlying asset's price to determine the payoff. There are two main types of lookback options:

- Fixed Lookback Options: The strike price is determined at the inception of the option, but the payoff is based on the maximum or minimum underlying asset price during the option's life.
- Floating Lookback Options: The strike price is determined based on the maximum or minimum underlying asset price during the option's life, providing the holder with the most favorable price movement.

The payoff formula for a lookback call option expiring at time T is for instance:

- Fixed-Strike: $\max(Y(T) - K, 0)$
- Floating-Strike: $S(T) - \min_{t \leq T} S(t)$ or $Y(T) - S(T)$

These options are more expensive than standard options because they offer a significant advantage in eliminating the uncertainty of timing market highs and lows. In this note, we only consider the following floating-strike lookback options with the following payoff:

$$V(T) := Y(T) - S(T) \qquad \text{(Lookback Payoff)}$$

An *Asian option* is a type of financial derivative where the payoff is determined by the average price of the underlying asset over a specific period, rather than its price at a single point in time. This averaging feature can reduce the option's volatility and, typically, its premium compared to standard options. We consider fixed-strike Asian call options whose payoff at time T is:

$$V(T) = \left(\frac{1}{T} \int_0^T S(t)\mathrm{d}t - K \right)^+ \qquad \text{(Asian Payoff)}$$

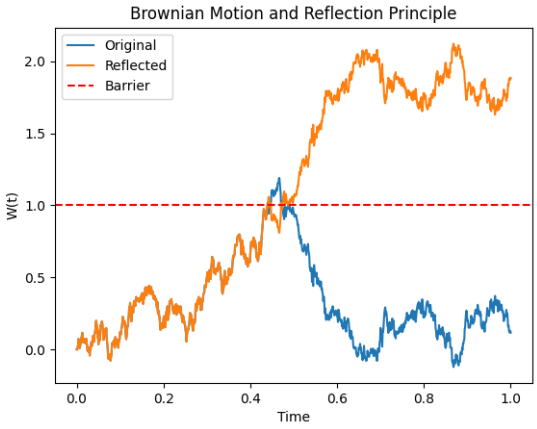
In this note, we provide insights into the pricing of these exotic options. Pricing barrier and lookback options involves understanding the joint density of the maximum of a Brownian motion $W(t)$ over the interval $[0, T]$ and $W(T)$. This distribution is derived using the *Reflection Principle*, which we define in the next section.

1 Reflection Principle

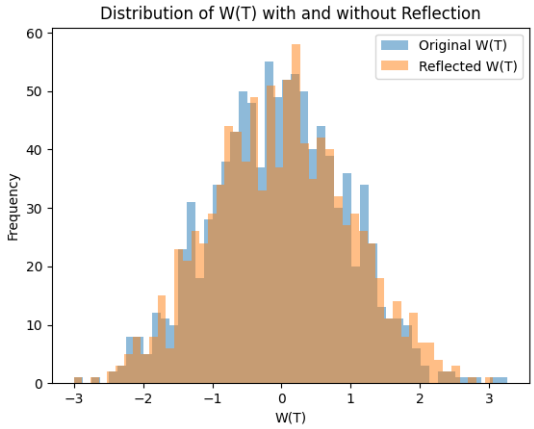
Consider a Brownian motion $W(t)$ and define the hitting time τ_m as below:

$$\tau_m = \min\{t : W(t) = m\}$$

The reflected path is defined inside Python code inside Section 3.1. Figure 1 also illustrates how a Brownian motion is reflected at the barrier level.



Reflection principle asserts that the reflected path is also a Brownian motion. For instance, Figure 1 illustrates that distribution of $W(T)$ and reflected path at T are both normal Gaussian.



Some simple manipulation computes the joint density of $M(t) := \max_{s \le t} W(s)$ and $W(t)$

$$\mathbb{P}\left(M(t) \ge m, W(t) \le \omega\right) = \mathbb{P}\left(W(t) \ge 2m - \omega\right), \quad w \le m, m \ge 0.$$

Using this, we can compute analytic formula for the present value of (Barrier Payoff) and (Lookback Payoff) via risk-neutral pricing formula

$$V(t) = \tilde{\mathbb{E}}\left[e^{-r(T-t)} V(T) | \mathcal{F}(t)\right] \qquad \text{(Risk-Neutral Pricing)}$$

It is emphasized that no analytic formula is known for Asian options. For these options, we will provide numerically friendly PDEs later below.

2 Partial Differential Equations

The primary purpose of this section is to provide numerically friendly partial differential equations for Asian options. However, before that we will also provide PDEs for lookback and barrier options.

2.1 Barrier

Denote

$$\mathcal{R}_{\text{Barrier}} = \{(t, x) : 0 \le t < T, 0 \le x \le B\}$$

If a barrier call has not been knocked out by t and $S(t) = x$, then the pricing formula $v^B(t, x)$ satisfies

$$v_t^B(t, x) + rx v_x^B(t, x) + \frac{1}{2}\sigma^2 x^2 v_{xx}^B(t, x) = r v^B(t, x) \qquad \forall (t, x) \in \mathcal{R}_{\text{Barrier}}.$$

More importantly, the following boundary conditions must hold

$$\begin{aligned} v^B(t, 0) &= 0, 0 \le t \le T \\ v^B(t, B) &= 0, 0 \le t < T \\ v^B(T, x) &= (x - K)^+, 0 \le x \le B \end{aligned}$$

Regarding the second boundary condition, note that $S(t)$ oscillates infinitely often immediately right after t and, therefore, will almost surely hit the barrier before T.

2.1.1 Delta-hedging

One important note regarding the delta-hedging practice for Barrier options is that $v^B(t,x)$ is discontinuous at the corner of $\mathcal{R}_{\text{Barrier}}$ at which delta (*i.e.*, $v^B_x(t,S(t))$) and gamma (*i.e.*, $v^B_{xx}(t,S(t))$) are large negative values. Normal delta-hedging becomes impractical as the large volume of trades renders significant the presumably negligible bid-ask spread. The common industry practice is to price and hedge the up-and-out call as if the barrier were at a level slightly higher than B.

2.2 Lookback

Denote

$$\mathcal{R}_{\text{Lookback}} = \{(t,x,y) : 0 \le t < T, 0 \le x \le y\}$$

Suppose $S(t) = x$ and $Y(t) = y$. Then the pricing formula $v^L(t,x,y)$ satisfies the following PDE

$$v^L_t(t,x,y) + rxv^L_x(t,x,y) + \frac{1}{2}\sigma^2x^2v^L_{xx}(t,x,y) = rv^L(t,x,y) \qquad \forall (t,x,y) \in \mathcal{R}_{\text{Lookback}}.$$

More importantly, the following boundary conditions must hold

$$\begin{aligned} v^L(t,0,y) &= e^{-r(T-t)}y, 0 \le t \le T, y \ge 0 \\ v^L_y(t,y,y) &= 0, 0 \le t < T, y > 0 \\ v^L(T,x,y) &= y - x, 0 \le x \le y \end{aligned}$$

The second boundary condition requires some explanation. It is important to note that $\mathrm{d}Y(t)$ is different from $\mathrm{d}S(t)$ and $\mathrm{d}t$. This follows from the fact that $Y(t)$ is monotonic and thus has zero quadratic variation. Moreover, $Y(t)$'s flat regions has Lebesgue measure 1 and hence $\mathrm{d}Y(t) \neq \Theta(t)\mathrm{d}t$ for any process $\Theta(t)$. It is straightforward to show that

$$\mathrm{d}Y(t)\mathrm{d}Y(t) = 0, \mathrm{d}Y(t)\mathrm{d}S(t) = 0.$$

Itô calculus gives

$$\mathrm{d}e^{-rt}v^L(t,S(t),Y(t)) = e^{-rt}[\cdots]\mathrm{d}t + e^{-rt}\sigma S(t)v^L_x(t,S(t),Y(t))\mathrm{d}\widetilde{W}(t) + e^{-rt}v^L_y(t,S(t),Y(t))\mathrm{d}Y(t)$$

Since $\mathrm{d}Y(t) \neq 0$ if and only if $S(t) = Y(t)$, the second boundary condition follows.

2.3 Asian

In this section, we consider fixed-strike Asian call with payoff (Asian Payoff). Denote

$$A(t) = \int_0^t S(u)\mathrm{d}u$$

The pair $(S(t), A(t))$ forms a Markov process, and Equation (Risk-Neutral Pricing) is rewritten as follows.

$$V(t) = v(t, S(t), A(t))$$

Taking into account the fact that $\mathrm{d}A(t) = S(t)\mathrm{d}t$, Itô calculus gives

$$v_t(t,x,a) + rxv_x(t,x,a) + xv_a(t,x,a) + \tfrac{1}{2}\sigma^2x^2v_{xx}(t,x,a) = rv(t,x,a), \quad x \ge 0.$$

Boundary conditions are also as below.

$$\begin{aligned} v(t,0,a) &= e^{-r(T-t)}\left(\frac{a}{T}-K\right)^{+} && ,0\leq t<T, a\in\mathbb{R} \\ \lim_{a\downarrow-\infty} v(t,x,a) &= 0 && ,0\leq t<T, x\geq 0 \\ v(T,x,a) &= \left(\frac{a}{T}-K\right)^{+} && ,x\geq 0, a\in\mathbb{R} \end{aligned}$$

These boundary conditions however suffer from some few drawbacks:

1. There is a degeneracy created by absence of $v_{aa}(t,x,a)$
2. It is unclear how $v(t,x,a)$ behaves as $x\uparrow+\infty$ and $a\downarrow-\infty$

A change of numeraire and a dimensionality reduction argument provides a different PDE for pricing Asian options. This idea is due to Vecer and is presented below in steps:

1. Construct a porfolio process $X(t)$ such that $V(T)=X^{+}(T)$ and for deterministic $\gamma(t)$

$$\mathrm{d}X(t)=\gamma(t)\mathrm{d}S(t)+r\left(X(t)-\gamma(t)S(t)\right)\mathrm{d}t$$

2. Consider the following PDE

$$g_t(t,a)+\tfrac{1}{2}\sigma^2\left(\gamma(t)-a\right)^2 g_{aa}(t,a)=0, 0\leq t<T, a\in\mathbb{R}$$

With boundary conditions as below:

$$g(T,a)=a^{+},\ \lim_{a\downarrow-\infty} g(t,a)=0,\ \lim_{a\uparrow+\infty} g(t,a)-a=0, a\in\mathbb{R}, 0\leq t<T.$$

The following is then true:

$$V(t)=S(t)g\left(t,\frac{X(t)}{S(t)}\right).$$

3 Python Codes

3.1 Reflected Brownian mption

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
T = 1.0  # Time interval
N = 1000  # Number of steps
dt = T / N  # Time step
a = 1.0  # Barrier level
n_simulations = 1000  # Number of simulations

# Function to simulate Brownian motion
def simulate_brownian_motion(T, N, dt):
    W = np.zeros(N+1)
    for i in range(1, N+1):
        W[i] = W[i-1] + np.sqrt(dt) * np.random.normal()
    return W

# Function to apply reflection principle
def reflect_brownian_motion(W, a):
    tau = np.argmax(W > a)  # First hitting time of level 'a'

    if tau > 0:
        W_reflected = W.copy()
        W_reflected[tau:] = 2 * a - W[tau:]
        import pdb;pdb.set_trace()
        return W_reflected
    else:
        return W
```

3.2 Barrier

```
import numpy as np
import matplotlib.pyplot as plt

def monte_carlo_barrier_option(S0, K, T, r, sigma, B, option_type='call',
                                        barrier_type='up-and-out', N_sim=10000,
                                        N_steps=1000):
    dt = T / N_steps
    disc_factor = np.exp(-r * T)

    # Generate paths
    S = np.zeros((N_sim, N_steps + 1))
    S[:, 0] = S0

    for t in range(1, N_steps + 1):
        Z = np.random.standard_normal(N_sim)
        S[:, t] = S[:, t - 1] * np.exp((r - 0.5 * sigma ** 2) * dt + sigma * np.
                                        sqrt(dt) * Z)

    # Apply barrier condition
    if barrier_type == 'down-and-out':
        # Set payoff to 0 if path hits the barrier
        payoff = np.maximum(S[:, -1] - K, 0) if option_type == 'call' else np.
                                        maximum(K - S[:, -1], 0)
        payoff[np.min(S, axis=1) <= B] = 0

    elif barrier_type == 'up-and-out':
        # Set payoff to 0 if path hits the barrier
        payoff = np.maximum(S[:, -1] - K, 0) if option_type == 'call' else np.
                                        maximum(K - S[:, -1], 0)
        payoff[np.max(S, axis=1) >= B] = 0

    elif barrier_type == 'down-and-in':
        # Payoff is 0 unless path hits the barrier
        hit_barrier = np.min(S, axis=1) <= B
        payoff = np.where(hit_barrier, np.maximum(S[:, -1] - K, 0) if option_type
                                        == 'call' else np.maximum(K - S[:
                                        , -1], 0), 0)

    elif barrier_type == 'up-and-in':
        # Payoff is 0 unless path hits the barrier
        hit_barrier = np.max(S, axis=1) >= B
        payoff = np.where(hit_barrier, np.maximum(S[:, -1] - K, 0) if option_type
                                        == 'call' else np.maximum(K - S[:
                                        , -1], 0), 0)

    # Discount the expected payoff
    price = disc_factor * np.mean(payoff)

    return price

# Parameters
S0 = 100        # Initial stock price
K = 100         # Strike price
T = 1.0         # Time to maturity
r = 0.07        # Risk-free rate
```

```
sigma = 0.1     # Volatility
B = 105          # Barrier level
N_sim = 10000   # Number of simulations

# Price the up-and-out call option
price = monte_carlo_barrier_option(S0, K, T, r, sigma, B, option_type='call',
                                   barrier_type='up-and-out', N_sim=N_sim)
print(f"The price of the up-and-out call option is: {price:.2f}")
# Prints  0.07
```

3.3 Lookback

```python
import numpy as np
def monte_carlo_lookback_option(S0, T, r, sigma, option_type='call', N_sim=10000,
                                N_steps=1000):
    dt = T / N_steps
    disc_factor = np.exp(-r * T)

    # Generate paths
    S = np.zeros((N_sim, N_steps + 1))
    S[:, 0] = S0

    for t in range(1, N_steps + 1):
        Z = np.random.standard_normal(N_sim)
        S[:, t] = S[:, t - 1] * np.exp((r - 0.5 * sigma ** 2) * dt + sigma * np.
                                       sqrt(dt) * Z)

    # Calculate payoff
    if option_type == 'call':
        max_S = np.max(S, axis=1)
        payoff = max_S - S[:, -1]
    elif option_type == 'put':
        min_S = np.min(S, axis=1)
        payoff = S[:, -1] - min_S


    # Discount the expected payoff
    price = disc_factor * np.mean(payoff)
    return price

# Parameters
S0 = 100        # Initial stock price
T = 1.0         # Time to maturity
r = 0.05        # Risk-free rate
sigma = 0.2     # Volatility
N_sim = 10000   # Number of simulations

# Price the lookback call option
price_call = monte_carlo_lookback_option(S0, T, r, sigma, option_type='call',
                                         N_sim=N_sim)
print(f"The price of the lookback call option is: {price_call:.2f}")

# Price the lookback put option
price_put = monte_carlo_lookback_option(S0, T, r, sigma, option_type='put', N_sim=
                                        N_sim)
print(f"The price of the lookback put option is: {price_put:.2f}")
# The price of the lookback call option is: 13.88
# The price of the lookback put option is: 16.96
```

3.4 Asian

```python
import numpy as np
def monte_carlo_asian_option(S0, K, T, r, sigma, option_type='call', N_sim=10000,
                                        N_steps=1000):
    dt = T / N_steps
    disc_factor = np.exp(-r * T)

    # Generate paths
    S = np.zeros((N_sim, N_steps + 1))
    S[:, 0] = S0

    for t in range(1, N_steps + 1):
        Z = np.random.standard_normal(N_sim)
        S[:, t] = S[:, t - 1] * np.exp((r - 0.5 * sigma ** 2) * dt + sigma * np.
                                            sqrt(dt) * Z)

    # Calculate arithmetic average price
    average_price = np.mean(S[:, 1:], axis=1)  # Exclude the initial price

    # Calculate payoff
    if option_type == 'call':
        payoff = np.maximum(average_price - K, 0)
    elif option_type == 'put':
        payoff = np.maximum(K - average_price, 0)

    # Discount the expected payoff
    price = disc_factor * np.mean(payoff)

    return price

# Parameters
S0 = 100        # Initial stock price
K = 100         # Strike price
T = 1.0         # Time to maturity
r = 0.05        # Risk-free rate
sigma = 0.2     # Volatility
N_sim = 10000   # Number of simulations
N_steps = 1000 # Number of time steps

# Price the Asian call option with arithmetic average
price_call = monte_carlo_asian_option(S0, K, T, r, sigma, option_type='call',
                                        N_sim=N_sim, N_steps=N_steps)
print(f"The price of the Asian call option with arithmetic average is: {price_call
                                        :.2f}")

# Price the Asian put option with arithmetic average
price_put = monte_carlo_asian_option(S0, K, T, r, sigma, option_type='put', N_sim=
                                        N_sim, N_steps=N_steps)
print(f"The price of the Asian put option with arithmetic average is: {price_put:.
                                        2f}")

#The price of the Asian call option with arithmetic average is: 5.84
#The price of the Asian put option with arithmetic average is: 3.38
```