Pricing Derivatives Using Black-Scholes-Merton Model

Rafiq Islam

2023-11-12

Table of contents

## Introduction

In this blog, we will explore how to price simple equity derivatives using the Black-Scholes-Merton (BSM) model. We will derive the mathematical formula and then provide Python code to implement it.

### Background and Preliminaries

Before proceeding to the deep of the discussion, we need to know some definition and terminology

**Brownian Motion:** Brownian motion is a concept with definitions and applications across various disciplines, named after the botanist Robert Brown, is the random, erratic movement of particles suspended in a fluid (liquid or gas) due to their collisions with the fast-moving molecules of the fluid.

*Brownian motion is a stochastic process* $\left(B\_{t}\right)\_{t\geq 0}$ *defined as a continuous-time process with the following properties:*

* $B\_{0}=0$ almost surely.
* $B\_{t}$ has independent increments.
* For $t>s$, $B\_{t}−B\_{s}∼N\left(0,t−s\right)$ (normally distributed with mean 0 and variance $t−s$).
* $B\_{t}$ has continuous paths almost surely.

from mywebstyle import plot\_style
plot\_style('#f4f4f4')
import numpy as np
import matplotlib.pyplot as plt

# Parameters
n\_steps = 100 # Number of steps
n\_paths = 20 # Number of paths
time\_horizon = 1 # Total time
dt = time\_horizon / n\_steps # Time step
t = np.linspace(0, time\_horizon, n\_steps) # Time array

# Generate Brownian motion
def generate\_brownian\_paths(n\_paths, n\_steps, dt):
 # Standard normal increments scaled by sqrt(dt)
 increments = np.random.normal(0, np.sqrt(dt), (n\_paths, n\_steps))
 # Cumulative sum to generate paths
 return np.cumsum(increments, axis=1)

# Generate one path and multiple paths
single\_path = generate\_brownian\_paths(1, n\_steps, dt)[0]
multiple\_paths = generate\_brownian\_paths(n\_paths, n\_steps, dt)

# Plotting
fig, axes = plt.subplots(1, 2, figsize=(7.9, 3.9))

# Single path
axes[0].plot(t, single\_path, label="Single Path")
axes[0].set\_title("Brownian Motion: Single Path")
axes[0].set\_xlabel("Time")
axes[0].set\_ylabel("Position")
axes[0].legend()

# Multiple paths
for path in multiple\_paths:
 axes[1].plot(t, path, alpha=0.5, linewidth=0.8)
axes[1].set\_title(f"Brownian Motion: {n\_paths} Paths")
axes[1].set\_xlabel("Time")
axes[1].set\_ylabel("Position")

plt.tight\_layout()
plt.show()



**Geometric Brownian Motion (GBM)**
A stochastic process $S\_{t}$ is said to follow a geometric Brownian motion if it satisfies the following equation:

$$dS\_{t}=μS\_{t}dt+σS\_{t}dB\_{t}$$

Which can be written as

$$S\_{t}−S\_{0}=\int\_{0}^{t}μS\_{u}du+\int\_{0}^{t}σS\_{u}dB\_{u}$$

To solve the GBM, we apply Ito’s formula to the function $Z\_{t}=f\left(t,S\_{t}\right)=ln\left(S\_{t}\right)$ and then by Taylor’s expansion, we have

By definition we have

The term $\left(dt\right)^{2}$ is negligible compared to the term $dt$ and it is also assume that the product $dtdB\_{t}$ is negligible. Furthermore, the quadratic variation of $B\_{t}$ i.e., $\left(dB\_{t}\right)^{2}=dt$. With these values, we obtain

with $Z\_{0}=lnS\_{0}$. Now we have the following

## Black-Scholes-Merton Formula

Now we are ready to derive the BSM PDE. The payoff of an *option* $V\left(S,T\right)$ at maturity is is known. To find the value at an earlier stage, we need to know how V behaves as a function of $S$ and $t$. By Ito’s lemma we have

Now let’s consider a portfolio consisting of a short one option and long $\frac{∂V}{∂S}$ shares at time $t$. The value of this portfolio is

$$Π=−V+\frac{∂V}{∂S}S$$

over the time $\left[t,t+Δt\right]$, the total profit or loss from the changes in the values of the portfolio is

$$ΔΠ=−ΔV+\frac{∂V}{∂S}ΔS$$

Now by the discretization we have,

At this point, if $r$ is the risk-free interest rate then we will have following relationship

$$rΠΔt=ΔΠ$$

The rationale of this relation is that no-aribtrage assumption. Thus, we have

This is the famous Black-Scholes-Merton PDF, formally written with the boundary conditions as follows

This Black-Scholes-Merton PDE can be reduced to the heat equation using the substitutions $S=Ke^{x}$, $t=T−\frac{τ}{\frac{1}{2}σ^{2}}$, and $c\left(S,t\right)=Kv\left(x,τ\right)$. Let’s derive the solution step by step in full mathematical detail and show how this leads to the normal CDF.

#### Step 1: Substitutions

We aim to reduce the BSM PDE:

$$\frac{∂c}{∂t}+\frac{1}{2}σ^{2}S^{2}\frac{∂^{2}c}{∂S^{2}}+rS\frac{∂c}{∂S}−rc=0$$

to the heat equation. Using the substitutions:

* $S=Ke^{x}$, where $x=ln\left(S/K\right)$, and $S\in \left(0,\infty \right)$ maps $x\in \left(−\infty ,\infty \right)$,
* $t=T−\frac{τ}{\frac{1}{2}σ^{2}}$, so $τ=\frac{1}{2}σ^{2}\left(T−t\right)$,
* $c\left(S,t\right)=Kv\left(x,τ\right)$, where $v\left(x,τ\right)$ is the transformed function.

#### Step 2: Derivative Transformations

For $c\left(S,t\right)=Kv\left(x,τ\right)$, we compute derivatives.

1. The first derivative of $c$ with respect to $S$:

$$\frac{∂c}{∂S}=\frac{∂}{∂S}\left(Kv\left(x,τ\right)\right)=K\frac{∂v}{∂x}\frac{∂x}{∂S},$$

* where $x=ln\left(S/K\right)$ implies $\frac{∂x}{∂S}=\frac{1}{S}$. Thus:

$$\frac{∂c}{∂S}=K\frac{∂v}{∂x}\frac{1}{S}.$$

1. The second derivative of $c$ with respect to $S$:

$$\frac{∂^{2}c}{∂S^{2}}=\frac{∂}{∂S}\left(K\frac{∂v}{∂x}\frac{1}{S}\right).$$

* Using the product rule:

$$\frac{∂^{2}c}{∂S^{2}}=K\frac{∂^{2}v}{∂x^{2}}\frac{1}{S^{2}}−K\frac{∂v}{∂x}\frac{1}{S^{2}}.$$

1. The time derivative:

$$\frac{∂c}{∂t}=K\frac{∂v}{∂τ}\frac{∂τ}{∂t}, and \frac{∂τ}{∂t}=−\frac{1}{\frac{1}{2}σ^{2}}.$$

#### Step 3: Transforming the PDE

Substituting the above derivatives into the BSM PDE, we rewrite each term.

1. For $\frac{∂c}{∂t}$:

$$\frac{∂c}{∂t}=−\frac{1}{\frac{1}{2}σ^{2}}K\frac{∂v}{∂τ}.$$

1. For $\frac{∂c}{∂S}$:

$$S\frac{∂c}{∂S}=S⋅\left(K\frac{∂v}{∂x}\frac{1}{S}\right)=K\frac{∂v}{∂x}.$$

1. For $\frac{∂^{2}c}{∂S^{2}}$:

$$\frac{1}{2}σ^{2}S^{2}\frac{∂^{2}c}{∂S^{2}}=\frac{1}{2}σ^{2}S^{2}\left(K\frac{∂^{2}v}{∂x^{2}}\frac{1}{S^{2}}−K\frac{∂v}{∂x}\frac{1}{S^{2}}\right)=\frac{1}{2}σ^{2}K\frac{∂^{2}v}{∂x^{2}}.$$

Substituting all these into the BSM PDE:

$$−\frac{1}{\frac{1}{2}σ^{2}}K\frac{∂v}{∂τ}+\frac{1}{2}σ^{2}K\frac{∂^{2}v}{∂x^{2}}+rK\frac{∂v}{∂x}−rKv=0.$$

Divide through by $K$:

$$−\frac{∂v}{∂τ}+\frac{∂^{2}v}{∂x^{2}}+\frac{2r}{σ^{2}}\frac{∂v}{∂x}−\frac{2r}{σ^{2}}v=0.$$

To simplify, let $v\left(x,τ\right)=e^{αx+βτ}u\left(x,τ\right)$, where $α$ and $β$ are constants. Substituting and choosing $α=−\frac{r}{σ^{2}}$ and $β=−\frac{r^{2}}{2σ^{2}}$, the equation reduces to:

$$\frac{∂u}{∂τ}=\frac{∂^{2}u}{∂x^{2}}.$$

#### Step 4: Solving the Heat Equation

The heat equation $\frac{∂u}{∂τ}=\frac{∂^{2}u}{∂x^{2}}$ has a well-known solution using Fourier methods:

$$u\left(x,τ\right)=\frac{1}{\sqrt{2πτ}}\int\_{−\infty }^{\infty }e^{−\frac{\left(x−y\right)^{2}}{2τ}}f\left(y\right) dy,$$

where $f\left(y\right)$ is the initial condition.

For the BSM problem, the initial condition is the payoff:

$$f\left(y\right)=max\left(e^{y}−1,0\right).$$

Performing the integration leads to the final solution involving the cumulative normal distribution function:

$$v\left(x,τ\right)=N\left(d\_{1}\right)−e^{−x}N\left(d\_{2}\right),$$

where:

$$d\_{1}=\frac{x+\frac{1}{2}τ}{\sqrt{τ}}, d\_{2}=\frac{x−\frac{1}{2}τ}{\sqrt{τ}}.$$

Transforming back to the original variables gives the Black-Scholes formula:

$$C\left(S,t\right)=Se^{−q\left(T−t\right)}N\left(d\_{1}\right)−Ke^{−r\left(T−t\right)}N\left(d\_{2}\right),$$

where:

$$d\_{1}=\frac{ln\left(S/K\right)+\left(r−q+\frac{σ^{2}}{2}\right)\left(T−t\right)}{σ\sqrt{T−t}}, d\_{2}=d\_{1}−σ\sqrt{T−t}.$$

Similarly, we can derive the price of a European put option:

$$P=Ke^{−rT}N\left(−d\_{2}\right)−Se^{−qT}N\left(−d\_{1}\right)$$

Where:

$$d\_{1}=\frac{ln\left(\frac{S}{K}\right)+\left(r−q+\frac{σ^{2}}{2}\right)T}{σ\sqrt{T}}, d\_{2}=d\_{1}−σ\sqrt{T}$$

### Asymptotic Behavior of the BSM formula for call and put options

What if $K\rightarrow 0$? In that case,

1. $ln\left(S\_{0}/K\right)\rightarrow \infty $, causing $d\_{1}\rightarrow \infty $ and $d\_{2}\rightarrow \infty $
2. The cdf $N\left(d\_{1}\right)\rightarrow 1$ and $N\left(d\_{2}\right)\rightarrow 1$
3. The second term $Ke^{−rT}N\left(d\_{2}\right)\rightarrow 0$ as $K\rightarrow 0$

In this case, the price of a call option $C\rightarrow S\_{0}$ and the price of a put option $P\rightarrow 0$

## Greeks: Delta and Gamma

**Delta** ($Δ$) is the sensitivity of the option price to changes in the underlying asset price:

$$Δ=\frac{∂C}{∂S}≈\frac{C\left(S\_{0}+h\right)−C\left(S\_{0}−h\right)}{2h}$$

This is the **central difference approximation**, which provides a more accurate estimate of delta compared to the forward or backward difference methods.

* $C\left(S\_{0}+h\right)$: Calculate the option price with the spot price increased by $h$.
* $C\left(S\_{0}−h\right)$: Calculate the option price with the spot price decreased by $h$.

**Gamma** ($Γ$) measures the rate of change of delta with respect to the underlying asset price:

$$Γ=\frac{∂^{2}C}{∂S^{2}}≈\frac{Δ\left(S\_{0}+h\right)−Δ\left(S\_{0}−h\right)}{2h}≈\frac{C\left(S\_{0}+h\right)−2C\left(S\_{0}\right)+C\left(S\_{0}−h\right)}{h^{2}}$$

Gamma ($Γ$) measures the rate of change of delta ($Δ$) with respect to the underlying spot price ($S\_{0}$).

* $C\left(S\_{0}+h\right)$: Option price with the spot price increased by $h$.
* $C\left(S\_{0}\right)$: Option price at the current spot price.
* $C\left(S\_{0}−h\right)$: Option price with the spot price decreased by $h$.

**Relationship Between Delta and Gamma:**

* Gamma represents how much delta changes for a small change in $S\_{0}$.
* If gamma is high, delta is more sensitive to changes in $S\_{0}$, which is important for hedging strategies.

## Implementation

### Notation

* $S$: Spot price of the stock.
* $K$: Strike price of the option.
* $T$: Time to maturity (in years).
* $r$: Risk-free rate (continuously compounded).
* $q$: Dividend yield (continuously compounded).
* $σ$: Volatility of the stock.
* $N\left(⋅\right)$: Cumulative distribution function of the standard normal distribution.

from dataclasses import dataclass
import numpy as np
from scipy.stats import norm

@dataclass
class Equity:
 spot: float
 dividend\_yield: float
 volatility: float

@dataclass
class EquityOption:
 strike: float
 time\_to\_maturity: float
 put\_call: str

@dataclass
class EquityForward:
 strike: float
 time\_to\_maturity: float

def bsm(underlying: Equity, option: EquityOption, rate: float) -> float:
 S = underlying.spot
 K = option.strike
 T = option.time\_to\_maturity
 r = rate
 q = underlying.dividend\_yield
 sigma = underlying.volatility

 # Handle edge case where strike is effectively zero
 if K < 1e-8:
 if option.put\_call.lower() == "call":
 return S
 else:
 return 0.0

 d1 = (np.log(S / K) + (r - q + 0.5 \* sigma\*\*2) \* T) / (sigma \* np.sqrt(T))
 d2 = d1 - sigma \* np.sqrt(T)

 if option.put\_call.lower() == "call":
 price = S \* np.exp(-q \* T) \* norm.cdf(d1) \
 - K \* np.exp(-r \* T) \* norm.cdf(d2)
 elif option.put\_call.lower() == "put":
 price = K \* np.exp(-r \* T) \* norm.cdf(-d2) \
 - S \* np.exp(-q \* T) \* norm.cdf(-d1)
 else:
 raise ValueError("Invalid option type. Must be 'call' or 'put'.")

 return price

def delta(underlying: Equity, option: EquityOption, rate: float) -> float:
 bump = 0.01 \* underlying.spot
 bumped\_up = Equity(spot=underlying.spot + bump,
 dividend\_yield=underlying.dividend\_yield,
 volatility=underlying.volatility)
 bumped\_down = Equity(spot=underlying.spot - bump,
 dividend\_yield=underlying.dividend\_yield,
 volatility=underlying.volatility)
 price\_up = bsm(bumped\_up, option, rate)
 price\_down = bsm(bumped\_down, option, rate)
 return (price\_up - price\_down) / (2 \* bump)

def gamma(underlying: Equity, option: EquityOption, rate: float) -> float:
 bump = 0.01 \* underlying.spot
 bumped\_up = Equity(spot=underlying.spot + bump,
 dividend\_yield=underlying.dividend\_yield,
 volatility=underlying.volatility)
 bumped\_down = Equity(spot=underlying.spot - bump,
 dividend\_yield=underlying.dividend\_yield,
 volatility=underlying.volatility)
 original\_price = bsm(underlying, option, rate)
 price\_up = bsm(bumped\_up, option, rate)
 price\_down = bsm(bumped\_down, option, rate)
 return (price\_up - 2 \* original\_price + price\_down) / (bump\*\*2)

def fwd(underlying: Equity, forward: EquityForward, rate: float) -> float:
 S = underlying.spot
 K = forward.strike
 T = forward.time\_to\_maturity
 r = rate
 q = underlying.dividend\_yield
 forward\_price = S \* np.exp((r - q) \* T) - K

 return forward\_price

def check\_put\_call\_parity(
 underlying: Equity,
 call\_option: EquityOption,
 put\_option: EquityOption,
 rate: float
 ) -> bool:

 call\_price = bsm(underlying, call\_option, rate)
 put\_price = bsm(underlying, put\_option, rate)
 S = underlying.spot
 K = call\_option.strike
 T = call\_option.time\_to\_maturity
 r = rate
 q = underlying.dividend\_yield

 parity\_lhs = call\_price - put\_price
 parity\_rhs = S \* np.exp(-q \* T) - K \* np.exp(-r \* T)

 return np.isclose(parity\_lhs, parity\_rhs, atol=1e-4)

### Example Usage

Say, we want to price a call option on an equity with spot price $S\_{0}=450$ with dividend yield $q=1.4\%$, and volatility $14\%$. The strike price of the call is $K=470$, with time to maturity in years $T=0.23$ and the risk free rate $r=0.05$. Next, we want to see the asymptotic behavior of the call option if the strike price $K\rightarrow 0$ with interest rate 0. Next, we want to price a put option on the same equity but strike price $K=500$, time to maturity in years $T=0.26$ and interest rate is 0. Finally, we want to check if the put-call parity relationship is hold. In each case, we consider $h=0.01$ a bump or small change in the stock price.

if \_\_name\_\_ == "\_\_main\_\_":
 eq = Equity(450, 0.014, 0.14)
 option\_call = EquityOption(470, 0.23, "call")
 option\_put = EquityOption(500, 0.26, "put")

 print(bsm(eq, option\_call, 0.05))
 print(bsm(eq, EquityOption(1e-15, 0.26, "call"), 0.0))
 print(bsm(Equity(450, 0.0, 1e-9), option\_put, 0.0))

 # Check put-call parity
 eq = Equity(450, 0.015, 0.15)
 option\_call = EquityOption(470, 0.26, "call")
 option\_put = EquityOption(470, 0.26, "put")
 print(check\_put\_call\_parity(eq, option\_call, option\_put, 0.05))

5.834035584709966
450
50.0
True

## References

* Karatzas, I., & Shreve, S. E. (1991). *Brownian Motion and Stochastic Calculus*.
* Options, Futures, and Other Derivatives by John C. Hull
* Arbitrage Theory in Continuous Time Book by Tomas Björk

**Share on**

**You may also like**