1 Motivations

I want to document here my project of C++ pricing library. C++ is a compiled programming language that has many advantages over Python: it is faster, manages memory efficiently with no possible leakage and is much more rigorous regarding variables declaration. For all these reasons I wanted to improve my skills in C++. To start my C++ journey I decided to implement my self made pricing library. I have always been interested in mathematical finance and programming that's why this project is perfect: I can work on both.
In the following I will explain my code, all the steps and the mathematics behind.

2 Option pricing

2.1 Black&Scholes model

To begin with, what else than Black&Scholes model for option pricing ? In the following we consider European options (exercised at maturity only).
The following hypothesis are made :

- The underlying asset follows geometric Brownian motion.

$$dS_t = rS_t dt + \sigma S_t dW_t$$

where:

S_t is the underlying price
r is the risk free rate, supposed constant
σ is the volatility of the underlying asset, supposed constant
dW_t is a Wiener process

- There are no dividends paid during the option's life.
- There is no risk-free arbitrage opportunity.

Applying Ito's lemma on the price of a call leads to Black Scholes equation :

$$\frac{\partial C}{\partial t} + rS\frac{\partial C}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} - rC = 0$$

this is equation can be rearranged into a heat equation Let $u(S,t) = e^{-rt}C(S,t)$. The transformed Black-Scholes equation becomes:

$$\frac{\partial u}{\partial \tau} + \left(\frac{1}{2}\sigma^2 - r\right)\frac{\partial^2 u}{\partial x^2} = 0$$

where:

$$\tau = T - t \qquad\qquad x = \ln S + (r + \frac{\sigma^2}{2})\tau$$

using limit conditions, this leads to well known formula:

$$\boxed{C = S_0 N(d_1) - Ke^{-rT}N(d_2)}$$

where:

S_0 is the current price of the underlying asset,
K is the strike price of the option,
T is the time to expiration in years,
r is the risk-free interest rate,
N is the cumulative distribution function of the standard normal distribution,
$d_1 = \dfrac{\ln\left(\frac{S_0}{X}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$,
$d_2 = d_1 - \sigma\sqrt{T}$,
σ is the volatility of the underlying asset's returns.

The same way we can compute the put price:

$$P = Ke^{-rT}N(-d_2) - S_0N(-d_1)$$

which shows Put-Call parity :

$$C - P = S_0 - Ke^{-rT}$$

From those formulas, we can easily derive all greeks:

1. Delta (Δ) - Sensitivity to the underlying asset's price:

$$\Delta_{\text{call}} = \frac{\partial C}{\partial S} = N(d_1)$$

$$\Delta_{\text{put}} = \frac{\partial P}{\partial S} = N(d_1) - 1$$

2. Gamma (Γ) - Rate of change of Delta with respect to the underlying asset's price:

$$\Gamma_{\text{call}} = \Gamma_{\text{put}} = \frac{\partial^2 C}{\partial S^2} = \frac{N'(d_1)}{S\sigma\sqrt{T}}$$

3. Theta (Θ) - Sensitivity to time decay:

$$\Theta_{\text{call}} = -\frac{\partial C}{\partial T} = -\frac{S_0\sigma N'(d_1)}{2\sqrt{T}} - rKe^{-rT}N(d_2)$$

$$\Theta_{\text{put}} = -\frac{\partial P}{\partial T} = -\frac{S_0\sigma N'(d_1)}{2\sqrt{T}} + rKe^{-rT}N(-d_2)$$

4. Vega (ν) - Sensitivity to changes in implied volatility:

$$\nu_{\text{call}} = \nu_{\text{put}} = \frac{\partial C}{\partial \sigma} = S_0\sqrt{T}N'(d_1)$$

5. Rho (ρ) - Sensitivity to changes in the risk-free interest rate:

$$\rho = \frac{\partial C}{\partial r} = KTe^{-rT}N(d_2)$$

$$\rho_{\text{put}} = \frac{\partial P}{\partial r} = -KTe^{-rT}N(-d_2)$$

Now we have all the formulas required to implement my first Black&Scholes model. I defined a class "Option" that takes as input the spot price S_0, the strike price K, the volatility σ, the risk free rate r and the time to maturity T. This Option object can be fed to a "BlackScholesModel" class that compute the option price and all it's greeks according to the Black&Scholes model.

2.2 Monte Carlo pricing

In this section we will price the same options but using an alternative method which consist in simulating a large number of path for the underlying price S_t and and average the results.

To begin, let's place ourselves in the same Black&Scholes conditions and suppose S_t follows the following geometric Brownian motion:

$$dS_t = rS_t dt + \sigma S_t dW_t$$

then applying Ito's lemma to $\ln(S_t)$ leads to,

$$d\ln(S_t) = (\mu - \frac{\sigma^2}{2})dt + \sigma dW$$

$$\ln(S_T) - \ln(S_0) \sim N((\mu - \frac{\sigma^2}{2})T, \sigma^2 T)$$

$$\boxed{S_T = S_0 e^{(r - \frac{\sigma^2}{2})T + \sigma\sqrt{T}Z}}$$

where:

$$Z \sim N(0, 1)$$

In the end, I computed a large number of normally distributed random variables, computed the spot price at maturity and fianlly computing the average payoff of my option. This gives me the expected payoff of the chosen option, this is its undiscounted price. Then,

$$C = e^{-rT}\mathbb{E}[max(0, S_T - K)]$$

$$\boxed{C = \frac{e^{-rT}}{N}\sum_{i=1}^{N} max(0, S_T^i - K)}$$

To compute all the greeks I use finite difference approximation,

$$\Delta \approx \frac{V(S + dS) - V(S)}{dS}$$

Now I can compare the results obtained between BlackScholes closed forms solutions and Monte-Carlo simulations. I obtain really close results.

3 Variance swaps