C++ in Quantitative Finance part #1

Individual home project

Submitted by**: Nihad Alili (466258)**

Submitted to **: Dr. Pawel Sakowski**

Degree**: Quantitative Finance, Master's degree II year**

Assumptions

- price of the underyling at the moment of option pricing: S0 = 100,
- strike price K = 110,
- annualized volatility rate σ = 25%,
- annualized risk-free rate r = 5%,
- time to maturity t = 0.75,
- barrier level H = 125
- number of steps N = 1000,
- number of path M = 1000000.

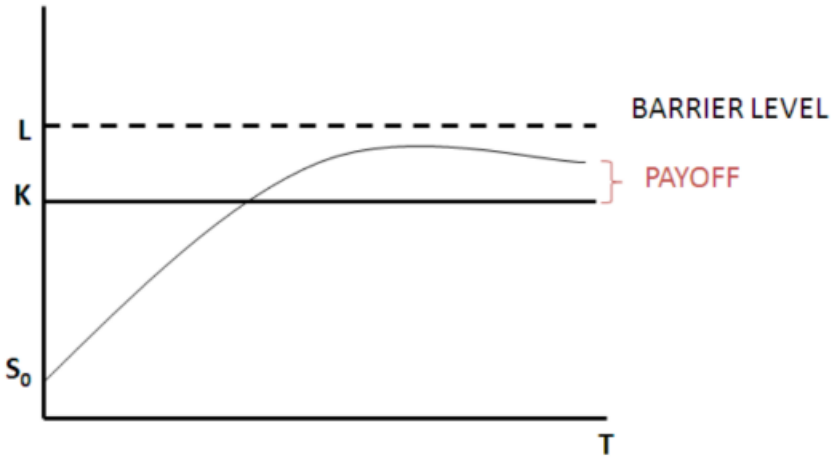
1. Methodology

This project aims to price barrier options using Monte Carlo Simulation on C++. First of all, details of barrier options and mathematical background of Monte-Carlo Simulation for this process and secondly, the codes are explained.

$$S(k\delta t) = S(0)\exp\left(\sum_{i=1}^{k}\left[\left(\mu - \frac{\sigma^2}{2}\right)\delta t + \sigma\varepsilon_i\sqrt{\delta t}\right]\right)$$

Formula 1. Geometric Brownian Motion Formula

$$\Phi_t = \max(S_T - X, 0) \quad \text{if} \quad \max_{0\leqslant t\leqslant T}(S_t) \leqslant L$$



1.1 Barrier option and Monte Carlo Simulation

Barrier options are a type of exotic options whose payoff do not only depends on strike price and forward price but also barrier level over time of stocks. It includes 8 option types including knock up(down) and in, knock up(down) and out, call(put) options. Knock up and out option is given in the project and is priced. Knock-up-and-out call option has the same logic with usual European options but additionally, barrier condition is set. This requires tracking the entire path of the asset price for each Monte Carlo simulation to determine if the barrier condition is triggered. In case of that barrier condition is triggered in whole life of the stock, option becomes worthless.

In Geometric Brownian Motion, deterministic part of the option is only calculated by parameters of risk-free rate, volatility, time to maturity/steps. Stochastic part, on the other hand, is calculated by parameters of volatility, time to maturity/steps, and random numbers generated by software. After simulating all possible values of stocks, payoffs are averaged arithmetically and continuously discounted. Only stock prices crossed strike price in the end and never crossed barrier level during life of the option are considered as positive payoff, so others are equalized to 0.

1.2 Code Implementation on C++

The zip file contains 3 files as source file, main file and header file for implementation of Monte Carlo Simulation on C++.

- **Header file**: Libraries **<vector>** and **<cmath>** are included to support numerical computations and container management. **#ifndef BARRIER_OPTION_H** and **#define BARRIER_OPTION_H** prevent multiple inclusions of the same header file, which can lead to redefinition errors during compilation. The **BarrierOption class** is declared, encapsulating all attributes and methods relevant to the barrier option. **simulatePrice()** simulates a single stock price path and calculate the corresponding payoff, considering the barrier condition. **PriceOption()** calculates the price of the barrier option by averaging the payoffs over multiple simulations and discounting to present value. These codes are used in source code.

- **Source File :** It gives the main 3 steps:

1. **BarrierOption::BarrierOption()** initizialise the BarrierOption object with user defined parameters.
2. **double BarrierOption::simulatePrice()** 1) simulate stock price over N path using the Geometric Brownian Motion (formula; 2) checks at each step whetether if the stock price exceeds the barrier, if so, it immediately returns a payoff of 0 (knock-out condition); 3) If the barrier is not breached, it calculates the payoff.
3. **double BarrierOption::priceOption()** 1) runs Monte Carlo simulations by repeatedly calling simulatePrice(); 2) Computes the average payoff across all simulations; 3) Discounts the average payoff to the present value

- **Main.cpp**: Overall, this file gathers input, creates an instance of the BarrierOption class, and calculates the option price using Monte Carlo simulation using **<iostream>.**

2. Result:

For time steps and number of path, numbers as a thousand and a million were added to the model taking into account of robustness of C++. As a barrier level, price level 125 is selected, so It is believed that It can have impact on the price of the option. Result is **0.43538** for the price of up-and-out barrier using call option with parameters S0=100, K=100, sigma=25%, r=5%, T=0.75 year after one million simulation.

In the beginning, H = 120 was selected as a barrier level and the result was 0,13 which is much lower than expected before experiment (Actually, It is not problem but I want to find higher value, that is why did not try lower barrier levels). After that, 130 was selected as a barrier level and the price of the option was 0,63 but It is not believable that price would increase to 130 so often. As a average, 125 was chosen as barrier level.

In order to be sure about the correctness of the model, barrier level was increased to 10000 which makes price even impossible to be breached and the result was compared to price of usual European call option which is calculated in Black-Scholes model with the same parameters. The results were so close.

In accordance with the Honor Code, I certify that my answers here are my own work, and I did not make my solutions available to anyone else.