Down-and-In Barrier Call Option Pricing

C++ Monte Carlo Simulation Project

Anindita Basu (466254)

January 25, 2025

1. Introduction and Objective

The goal of this project is to approximate the theoretical price of a *down-and-in barrier call option* via Monte Carlo simulations implemented in C++.

2. Assumptions

- **Underlying Asset Dynamics:** We assume the underlying asset follows a stochastic process with a constant risk-free rate r and a constant volatility σ. For simplicity, an Euler discretization of the geometric Brownian motion (GBM) is approximated by

$$S_{t+\Delta t} = S_t + S_t \times (r\,\Delta t + \sigma\sqrt{\Delta t}\,Z),$$

  where Z is a standard normal random variable generated via the Box-Muller method.

- **Monte Carlo Iterations:** We run a large number of simulated paths (num_exp = 100000) to achieve a stable approximation, each path consisting of n_steps(= 10000) time increments.

- **Constant Model Parameters:**
  - Initial stock price: $S_0 = 100$
  - Strike price: $K = 110$
  - Annualized volatility: $\sigma = 25\%$
  - Annualized risk-free rate: $r = 5\%$
  - Time to maturity: $t = 0.75$ (years)

- **Barrier Level** b: We experiment with reasonable values of b to *slightly* influence the price relative to a non-barrier option. In practice, we avoid setting b so low that it nearly never knocks in, or so high that it is immediately activated.

3. Down-and-In Barrier Call Option

A down-and-in barrier option is a type of exotic option that:

- **Activates (knocks in)** only if the underlying asset price S_t goes *below or equal* to a predefined barrier b at any point during the option's life.
- **Payoff (if activated)** for a call is $\max(S_T - K, 0)$ where S_T is the asset price at maturity.

Since the barrier must be breached for the option to become active, the barrier level b significantly influences the option value. If b is set too low, the activation is unlikely, leading to a low or zero option price. Conversely, if b is close to or higher than S_0, activation becomes more probable, and the option price approaches that of a plain vanilla call.

4. Description of the Code

4.1. Class `barrieroption`

We define a C++ class `barrieroption` with the following members:

- `double S0, K, sigma, r, t, b;`
- `int n_steps, num_exp;`
- `std::vector<double> payoffs;`

The constructor initializes these data members. Notable methods include:

- `get_one_gaussian_by_box_muller()`: Generates a single standard normal random variable using Box-Muller.
- `simulate()`:
  1. Initializes the payoffs vector.
  2. For each experiment, simulates a path for the underlying asset by discretizing time into n_steps intervals.

3. Checks if the barrier b is breached; if so, records the payoff $\max(S - K, 0)$ at maturity.
4. Computes the mean payoff and discounts it at the risk-free rate r.

- `calculate_stddev()`: Computes the standard deviation of the stored payoffs.

4.2. Main Program

In the main function, we create multiple `barrieroption` objects, each with a different barrier value b. By calling `simulate()`, we print the resulting discounted payoff, representing the approximate theoretical price of each barrier option.

5. Results

Using the following common parameters:

$$S_0 = 100, \quad K = 110, \quad \sigma = 0.25, \quad r = 0.05, \quad t = 0.75,$$

and varying the barrier b, we tested several scenarios:

Barrier	**Avg. Payoff**	**Discounted Price**	**Std. of Price**
95	2.80977	2.70635	7.95543
97	3.93535	3.79051	9.55912
99	5.50280	5.30027	11.4710

We observe that as the barrier b rises from 95 to near or above the initial stock price S_0, the option price approaches that of a vanilla call (since the option is more easily activated) and the payoff also increases.

6. Conclusions

- We successfully implemented a Monte Carlo simulation in C++ to price a down-and-in barrier call option.
- The option becomes more expensive as the barrier is raised closer to S_0, indicating a higher probability of activation.

- The code structure follows object-oriented principles: we encapsulate option parameters in a single class and perform simulations in `simulate()`.
- The results for barrier levels $\{95, 97, 99\}$ show a clear trend of increasing option price as activation becomes more likely.
- All computations rely on standard assumptions of geometric Brownian motion and constant parameters.

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.