C++ PROJECT REPORT

OPTION PRICING

FINANCIAL ENGINEERING

2022-2023

Authors:	*Id :*
THOMAS Nicolas	720948
STRIEBIG Maximilien	720808
ROUDAUT Antoine	711964
THOMASSIN Pablo	719295

Course Coordinator:

Juliette ACKET

26th December 2023

Contents

1 Abstract

This project presents a comprehensive analysis of option pricing methodologies employing the Cox-Ross-Rubinstein (CRR) model, the Black-Scholes model, and Monte Carlo simulation with generated data. The primary objective is to compare and contrast these widely used methods for pricing financial options and assess their accuracy and efficiency in different market scenarios.

2 Introduction

In the dynamic landscape of financial markets, the accurate pricing of options plays a pivotal role in risk management and investment decision-making. This project delves into the realms of option pricing, utilizing the Cox-Ross-Rubinstein (CRR) model, the Black-Scholes model, and Monte Carlo simulation with generated data. These models represent diverse approaches, each offering unique insights into the complex world of financial derivatives.

The Cox-Ross-Rubinstein model, rooted in discrete-time binomial pricing, provides a valuable framework for comprehending option pricing dynamics. By implementing this model in C++, our project seeks to unravel its intricacies and assess its efficacy in capturing option prices across various market scenarios. The discreet nature of the CRR model allows for a nuanced examination of market movements in a step-by-step manner.

Parallelly, the Black-Scholes model, an enduring cornerstone in option pricing theory, is also implemented in C++. Known for its closed-form solutions in continuous-time settings, the Black-Scholes model offers a contrasting perspective. Through this implementation, we aim to contrast the strengths and weaknesses of both models and gain a deeper understanding of their respective domains of applicability.

3 Deep view of the architecture of the code

3.1 PART I : CLASS DEFINITION AND STRUCTURES

3.1.1 OPTION CLASS AND HEREDITY

In this part of the project we are developing the founding architecture for the whole project, as such we will start by explaining the option part. The whole project evolves around pricing option, and for that we don't want in the future to re-do everything from scratch as us we determined what option shares as common trait to define it in the virtual class option.

As such we decided to create a daughter class, the VanillaOption class, which is a type of option (We are talking here about European Option) where the client has the right to exercise the option at maturity. The VanillaOption class has two daughter class herself : PUT and CALL option which are here to describe the structure of European put and call.

3.1.2 BLACK-SCHOLES PRICER

The other structures that have to be define is the Black-Scholes model for pricing option, starting to develop first it's parameter then a pricing methods for all kind of option. The model is using maths formula based on the solution of the partial equation (which in fact is a diffusion equation).
For our project we decided to price VanillaOption as such :

For a Call option:

$$C(S, K, T, r, \sigma) = S \cdot N(d_1) - K \cdot e^{-rT} \cdot N(d_2)$$

For a Put option:

$$P(S, K, T, r, \sigma) = K \cdot e^{-rT} \cdot N(-d_2) - S \cdot N(-d_1)$$

where:

$$d_1 = \frac{\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right) T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

3.1.3 OPTION CLASSES

We tested our pricer on different type of option as such we implemented Put and Call for :
- Vanilla option
- American option
- Asian option
- Digital option

In order to make some coherence in our code, each of the specific option are daughter class from mother class ; Option.
As such we tried to make some coherence in the pricer that we will introduce later. For the morphism we decided to make a selector based on a enum class. As first we tried to do in with pointer but it was hard and not so reliable because some pointer would be null and could cause memory leak errors :

Listing 1: enum class OptionNature

```
enum class OptionNature {
    American,
    Asian,
    Digital,
    Vanilla

};
```

3.2 PART II : DIVING IN CRR PRICING AND BINARY TREE

3.2.1 BINARY TREE CLASS

The BinaryTree class is designed to contain a 2D vector of any type, thanks to the use of templates. The class has two private members: depth, representing the height of the tree, and a vector of vectors named tree to store the node values. The default destructor is deemed sufficient for the class.
Key Methods:

- setDepth: This method resizes the tree to a depth of N+1 (where the first node doesn't count), and for each level, it resizes the vector to the depth plus one.
- setNode: Checks the validity of the selected indices and then sets the value of a specific node in the tree.
- getNode: Similar to setNode, this method checks if the selected indices are valid and then retrieves the value of a specific node.
- display: Displays the binary tree in the console. It utilizes an external function, size float, to determine the number of digits in each node. The method then uses cases to manage spaces between nodes, ensuring a well-formatted display.

3.2.2 CRR PRICER CLASS

The purpose of the CRRPricer class is to implement the Cox-Ross-Rubinstein (CRR) procedure for pricing financial options. The class has a private argument option which points to any option type, providing flexibility. The depth N is used to create a binary tree to store the asset price at any time.
Key Methods:
- Constructor: Checks for invalid parameter combinations to prevent arbitrage opportunities. compute:

- Computes the asset price at each time step using the formula $S(n,i) = S_0(1+U)^i(1+D)^{(n-i)}$.
- Applies the Cox-Ross-Rubinstein (CRR) procedure through backward induction, seeking previously computed prices in the tree.
- Adds the computed asset price to the tree.
- Get Method: Invokes the Compute method to populate the tree and then retrieves the option value at a specific node in the binomial tree.

- Operator(): Computes the option final price using either the closed-form solution (to calculate factorial we used the gamma function) alternatively, it uses the Compute method to populate the binomial tree and retrieves the computed value for the root node.

3.3 PART III : MONTE CARLO SIMULATION FOR BLACK-SCHOLES PRICER

3.3.1 DIGITAL OPTION

The class DigitalOption is made on the same basis as VanillaOption. It only differs in its nature which is Digital and its payoff method which only 1 if the option is executed.
The pricer of Black Scholes has been upgraded consequently with a way to price and compute the delta according to the nature of the option :
Price of a Digital Call :

$$C_{\text{digital}} = e^{-rT} \cdot N(d_2)$$

Where :

$$d_2 = \frac{\ln\left(\frac{S}{K}\right) + \left(r - \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$$

Delta of a Digital Call :

$$\Delta_{\text{Call digital}} = e^{-rT} \cdot N(d_2)$$

Price of a Digital Put:

$$P_{\text{digital}} = e^{-rT} \cdot N(-d_2)$$

Delta of a Digital Put :

$$\Delta_{\text{Put digital}} = -e^{-rT} \cdot N(-d_2)$$

In order to handle the type of option priced, the Black Scholes pricer check the nature of the option and then enters in the right pricing method.

3.3.2 ASIAN OPTION

The AsianOption class is defined on the same basis as the other classes but has some differences. The method getExpiry() of a Vanilla option is replaced by getTimeSteps() which returns the vector of dates named time in the constructor.

The payoff method has also been reviewed and is named payoffPath. It takes as an argument a vector of prices at different dates and then, if the average of the prices are in a condition of profit (depending of the type of the option), it will execute the right of the option. CALL AND PUT AsianCallOption and AsianPutOption are sister classes and derived from their mother classe AsianOption. hey inherite of the nature of the option and have a type : Call or Put.
The method payoffPath is set to be in accordance with the type of the option but the principle of average of prices stills the same.
Here the return for a call with zt the price of the asset at time t:

$$h(z_1, \ldots, z_t) = \left(\left(\frac{1}{t} \sum_{k=1}^{t} z_k \right) - K \right)^+$$

3.3.3 SINGLETON CLASS MT

The singleton class MT is designed to generate a unique random seed which will generate a random value in an N(0,1) or in an U(0, 1) through two methods : rand norm() and rand unif(). These random figures will be used in the following class BlackScholesMCpricer.

3.3.4 MONTE-CARLO BLACK-SCHOLES

The aim of the class BlackScholesMCpricer is to compute the price of an option thanks to randomly generated futur prices of the asset.
The user enters as an input the number of simulated prices, called paths. This means that for a Vanilla option, a number a paths prices will be generated. These simulated prices are stored in a vector called sim prices that are themself stored in another vector for a matter of generalized process which allows the program to compute the price of options disregarding their nature (Vanilla, Digital, Asian).
If for a Vanilla or Digital option the vector will be composed of vector of size 1, for an Asian option, the vector will be composed of vectors with same size as the vector of time steps.
These prices are simulated according to this formule :

$$S_{t_k} = S_{t_{k-1}} \cdot e^{\left(r - \frac{\sigma^2}{2}\right)(t_k - t_{k-1}) + \sigma\sqrt{t_k - t_{k-1}} \cdot Z_k}$$

Where Z1,..., Zm are i.i.d. random variables with distribution N (0, 1).

Once these prices simulated, we called the respecting payoff method for each price or series of prices.// Finally, the price of the option is computed using THIS EXACT FORMULA

$$H_0 \approx e^{-rT} \frac{1}{N} \sum_{i=0}^{N-1} h\left(S^i_{t_1}, \ldots, S^i_{t_m}\right)$$

Where S_t are vectors containing vectors of simulated prices.
N represents the total number of paths generated since the beginning.

3.4 PART IV : AMERICAN OPTION AND CRR PRICING METHOD

The final part of this project involves pricing an American option using the CRR model. American options differ from European options in that they give the holder the right to exercise the option at any time before maturity.

First, we created an American Option class, a child class of Option, and overloaded it with several IsAmericanOption(), AmericanCallOption() and AmericanPutOption() methods. All these methods return variables that will be used in the CRR model or Booleans to check certain conditions before setting the option price.

The AmericanCallOption() and AmericanPutOption() methods have been implemented with a constructor and overloaded payoff() methods specific to their option types in order to give a clear view of their functionality and improve the visibility and reusability of the code. visibility and reusability of the code. In the CRRPricer class we have introduced a BinaryTree for boolean variable to store the exercise conditions,accessible via the getExercise method, which takes two parameters.

This makes it possible to model the dynamic evolution of the payoff of an American option through different time steps along all the paths.

In addition, the CRR model can be used to approximate the Black-Scholes model if N is sufficiently large. This is why we have overloaded the constructor by adding the parameters U, D and N. These parameters are related to the Black-Scholes model using the same input data

such as volatility and interest rate.

$$U = e^{\left(r+\frac{\sigma^2}{2}\right)h+\sigma\sqrt{h}} - 1$$

$$D = e^{\left(r+\frac{\sigma^2}{2}\right)h-\sigma\sqrt{h}} - 1$$

$$R = e^{rh} - 1$$

In conclusion, the use of polymorphism, the hierarchical structure for clarity and maintainability and the implementation of error management provide a flexible tool for pricing multiple assets under a variety of market conditions.

4 Conclusion

To conclude the main objective of this project is attained by realizing a pricer for differents options type and for different methods, at first the normal Black and Scholes way, then using a discrete Cox Ross Rubinstein way with descretization of data involved. Finally using a Black Scholes model with data evolved from Monte Carlo algorithms simulation.
This Project made us reflect on how banks can price their option for different asset, which is in a certain way is an insurance for the customer.
During this project every member worked on average in the same load of work. Thus reflecting a group work in opposite to independent work.

Appendix on some notation and usage :
- Sometimes there is the usage of size t type instead of regular types existing this is due to the fact that we sometimes compare discrete index on long integer vector indexeds.