Eva CHENG – Alexis CABRAL IF1

Project in C++ for Finance

Report

I. Introduction

This project has been done within our course of C++ in the 4th year at ESILV in Financial Engineering. We, students of 4th-year has been working on this project for 2 months. First of all, the purpose of this project is to let us improve our knowledge in C++ by working on a project related to Finance. Secondly, its goal is allow us to understand financial concepts and the concept of pricing by algorithm. In this report, we will introduce the resources and software we have used, the procedure we have used to resolve each part of the project and the different functionalities and classes we have implemented.

II. Resources and Software

a. Resources

Since one of the students who have been working on this project was a former student at EFREI Paris and has had a course of C++ and has done several projects in C++ / Java, we were to a certain extent quite at ease with the syntax and the concepts used in C++ (inheritance, polymorphism, templates, enumeration, overloading, pointers). For the finance part and the algorithm part, we have used several resources such as John Hull's book, Wikipedia, Investopedia but also cplucplus.com and stack overflow.

b. Software

To code in C++ we have used Visual Studio 2019 and to work together, we have been using Git / GitHub to allow us to work on different part of the project together remotely. To work together, we have been using Discord, Zoom and Messenger for communication.

III. Procedure used to solve each TD

a. TD4

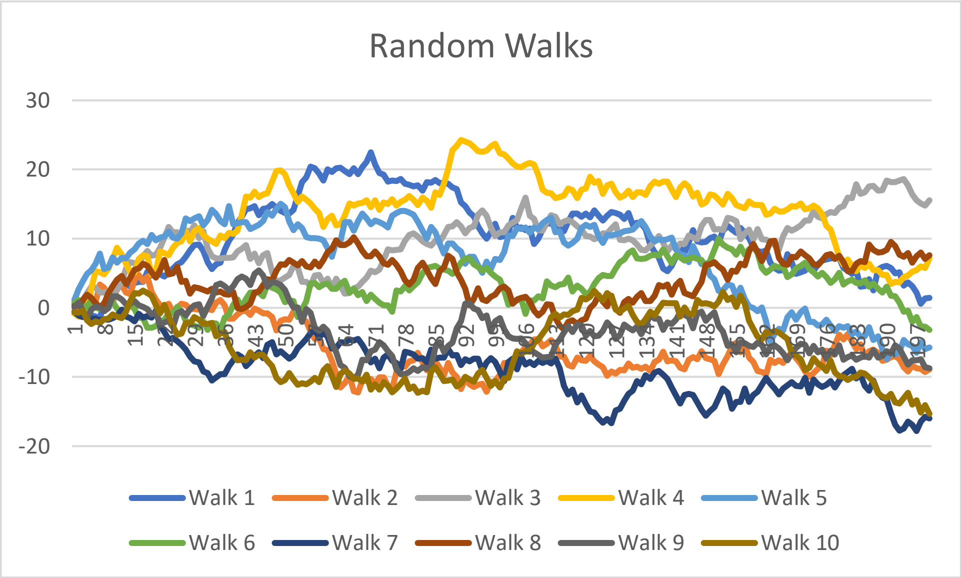
1) Random Walks and Random Walk Generator

The purpose of this TD was to simulate random walks by

The principal class we will use here is the Random Walk Generator which is basically a class that contains a vector of Random walks, its size and all the methods we will use to generate the random walks.

The Random walk class contains the values of a single random walk and has every methods that are associated to the random walks. It also allows the usage of different distribution by taking in parameter a function in the constructor.

The different distributions will be implemented in the main. The classes will be detailed in the part IV of the report and each value is stored in a csv file so we can create graphics such as :



b. TD5

1) Black and Scholes

The purpose of this TD was to understand the model of Black and Scholes used to price Call and Put. To that end, we have searched for the formulas of Black and Scholes on the internet and made up 3 classes. The main class is the Black Scholes Model class and then we have implemented 2 other classes which purpose is to calculate the integral of a Normal Law. Indeed, in the model of Black and Scholes, we have to calculate the cumulative function of d1 and d2 that follows a normal law.

To calculate the normal law and the cumulative distribution function of the normal law, we have implemented two different methods : the first one with the error function of the normal law and the second thanks to a Monte Carlo approach.

We will detail this approach later on this report.

The third class we have implemented is a generator which is a singleton in the program, it allows us to generate randomly the points for the Monte Carlo approach thanks to a singleton.

2) Cox Ross Rubinstein for European Option

The purpose of the exercise was to understand the method of CRR and to resolve this exercise, we have implemented three classes : Node, Binomial Tree and the main class that is Cox Ross Rubinstein European Option.

Since we have a binomial tree, we have chosen to create the node class which is each element of the tree, the tree class connects every nodes together and the CRR class is the main class.

Thanks to this, we can access easily to each node and go through the tree easily since every node is connected in the same way as the binomial tree is.

c. TD 6

1) Cox Ross Rubinstein Closed Method

This method is pretty close to the previous one but instead of using a tree, we use the fact that the moves follow a binomial law of parameter (n, p). We compute all the u, d, p and q following the same method as the previous one.

And we obtain a result that is close to the previous one.

2) Option

The main purpose of this exercise was to understand the concept of inheritance and polymorphism. To solve this TD, we have created a class option and 4 classes that derive from the option class. All the parameters and method that are usually private are put into protected to give the right to the derived classes to access to the different attributes and methods. For the method that are different depending on the derived class, we use a virtual method (meaning that the Option class is considered as abstract) in the principal class and implemented them in the sub classes.

3) Bin Lattice

The purpose of this exercise was to understand the usage of templates, this is why we have created a class that has an attribute T which vary depending on the type of the class. It also provides a display made up by an algorithm we imagined by thinking about a design firstly on a paper sheet and then via the program.

d. TD 7

1) Monte Carlo

We have created a class Monte Carlo Pricing Method that contains the Asian Options and the European Option and all the methods that goes with them.

e. TD 8

1) American Option

To solve this exercise, we have create a single class that has methods that are similar to the CRR pricer closed method since some part are the same as European Options and others that are unique to American options.

2) Black and Scholes as the Limit of the binomial tree

We have implemented the Black Scholes Limit class and a class Comparer that contains an instance from Black Scholes Limit and another from Monte Carlo Pricer and we compare the two of them to tell the difference.

IV. Classes

a. Random walk (TD 4)

1. Constructor with parameters

It takes in parameter the size of the walk, the different values that are generated outside the class and the distribution which is the name of the distribution. The constructor for other classes will not be detailed anymore.

2. Compute mean

It generates the mean of the values of the walk.

3. Setters and Getters

The setters set a value taken in parameter to an attribute of the class and the getters returns the value of the attributes that are private. (This will be the same for the other classes, so we will not mention the setters and the setters in the report again because it is the same for any other class even if they are present in all classes).

4. Bracket overloading

We have overloaded the brackets to allow the user to access to an element of the vector of the random walk by using the brackets and the integer between them only.

b. Random Walk Generator

1. Get Instance

Since the random walk generator is a singleton, we each iteration we use the getInstance method that return the only instance of generator.

2. Generate function

It take in parameter the number of step of a walk, two parameters, a distribution function returning a double which take is argument two doubles and the name of a distribution. Then we generate a vector that stores for each element, the previous value plus a random number by using the distribution function. Then it creates a vector a random walk and stores the random walk into a vector of random walk.

3. Bracket overloading

Same as the previous class.

c. Black-Scholes Model

1. Computable

It handles the errors by verifying all the parameters before computing anything.

2. Computers

They compute the parameters used in Black and Scholes with the formulas provided in the TD, Investopedia, the book of John Hull and the internet. (The will no longer be mentioned in the report for all the classes because they are almost the same).

3. Calculate CDF

We here use a Monte Carlo approach we taking in parameter the upper limit then we generate a big number of points between 0 and the upper limit (uniform distribution) and then applying the Monte Carlo approach between 0 and x adding 0.5 (since the normal law is centered in 0 (infinity to 0 is 50% of the cdf).

4. CDF with the error function

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int e^{-u^2} du$$

$$F_{Normal}(x) = \frac{1}{\sqrt{2\pi}} \int e^{-\frac{u^2}{2}} du = \frac{1}{2} + \frac{1}{2} \times \mathrm{erf}\left(\frac{u}{\sqrt{2}}\right)$$

d. Monte Carlo Normal Law

1. Method function

It represents the density function of the centered reduced normal law.

2. Compute

It generates a lot of point using the density function and it uses monte carlo to calculate the integral.

3. Operator ()

It returns the estimate.

e. MT19937 Generator

It is a singleton that returns a single generator to avoid declaring it a lot of times.

f. Cox Ross Rubinstein European Option

1. Create tree

It builds up a tree following the parameters of the CRR pricer

2. Create terminal nodes values / pay off

It creates a vector that contains all the terminal nodes (nodes that do not have children or nodes following them).

3. Create Node Vector

Creating a vector that contain all the nodes sorted by their level to allow the user to access to the nodes more easily.

g. *Node*

The comments on the code are sufficient to understand the code. Each node is located in the tree following its parents and its children.

h. *Binomial Tree*

The comments on the code are sufficient to understand the code.

i. *CRR Pricer Closed Method*

The comments on the code are sufficient to understand the code

j. *Option*

The comments on the code are sufficient to understand the code

k. *BinLattice*

The comments on the code are sufficient to understand the code

l. *Monte Carlo Option Pricing*

The comments on the code are sufficient to understand the code

m. *American Options*

The comments on the code are sufficient to understand the code

n. *Black Scholes Limit Tree*

The comments on the code are sufficient to understand the code

o. *Method comparer Monte Carlo VS Binomial tree*

The comments on the code are sufficient to understand the code

p. *Main*

The main tests all the functionalities implemented and all the TD, it of course handles the error and exceptions.