Option's Pricer

ALQUIER Mattéo, DEBOERDERE Gabriel, PINCON Axel

30 Décembre 2023

Contents

1 Introduction

Dans cette partie, nous allons définir le but du projet ainsi que les différentes hypothèses et conventions utilisées afin de réussir ce projet.

1.1 Contexte et But du projet

Le pricing d'options est une composante cruciale de la finance de marché, jouant un rôle central dans la gestion des risques, la spéculation, et la couverture d'actifs financiers. Il permet aux traders et aux institutions financières de déterminer la juste valeur d'options, des instruments financiers dérivés dont le prix est dérivé de la valeur d'autres actifs, tels que des actions, des indices ou des devises. Une compréhension précise et fiable du pricing est indispensable pour prendre des décisions éclairées, optimiser les stratégies de trading, et minimiser les risques financiers.

Parmi les méthodologies développées pour le pricing d'options, le modèle de *Black-Scholes-Merton* se distingue comme une référence incontournable. Introduit dans les années 1970, il fournit une formule analytique pour le pricing d'options européennes sous certaines hypothèses standardisées, facilitant ainsi l'évaluation des options de manière théorique et pratique. Cependant, la complexité des produits financiers et la diversité des conditions de marché ont conduit à la recherche de méthodes plus flexibles et puissantes.

Dans ce contexte, la méthode de *Monte-Carlo* émerge comme une solution robuste pour estimer le prix des options, surtout lorsqu'il est difficile d'obtenir des solutions analytiques. Par son approche de simulation, elle permet de modéliser une large gamme de scénarios de marché et de comprendre la distribution des résultats possibles, ce qui est particulièrement utile dans l'évaluation des options exotiques ou dans des marchés incomplets.

En outre, l'approche de *Longstaff et Schwartz*, connue pour sa méthode de régression pour évaluer les options américaines, enrichit l'arsenal des techniques de pricing. Cette méthode se base sur la simulation de trajectoires d'actifs sous-jacents et sur des techniques de régression pour estimer la continuation et la valeur d'exercice anticipé, offrant ainsi une alternative pratique pour le pricing d'options américaines qui ne peuvent être facilement évaluées par des formules closes.

Nous avons donc décidé de pricer des options selon ces 3 méthodes :

- Utilisation de la méthode *Black-Scholes-Merton* pour les options européennes
- Utilisation de la méthode *Monte-Carlo* pour les options asiatiques
- Utilisation de la méthode *Longstaff et Schwartz* pour les options américaines

Afin de décrire le fonctionnement de notre programme, nous allons tout d'abord faire la description générale du programme en expliquant le fonctionnement utilisateur du programme et en donnant l'archictecture du programme. Ensuite nous ferons la description fonctionnelle du programme en décrivant

les différentes fonctions implémentées. Enfin, nous mettrons en exergue les problèmes rencontrés ainsi que les solutions apportées. Nous laissons, pour une meilleure compréhension des méthodes de pricing, les formules de pricing en annexe

1.2 Hypothèses et Cadre d'utilisation

Afin de simplifier le problème, nous avons supposé que le **taux sans risque est constant**.

De plus, le pricing mis en place permet de pricer trois types d'options: **américaines, européennes et asiatiques**. Ces mêmes options peuvent avoir trois types de dividendes possibles:**pas de dividende**, **dividendes discrets** et **dividendes continus**. Toutes ces caractéristiques de l'option seront à préciser dans l'interface utilisateur.

2 Description générale du programme

2.1 Guide d'utilisation

Ce programme a nécessité deux librairies: **cmath** et **Eigen**. Il est nécessaire de les installer afin de rendre possible l'éxécution de notre projet.

Le pricer fonctionne à partir d'une interface utilisateur. Une fois le projet exécuté, une interface devrait s'ouvrir vous demandant de compléter diverses caractéristiques de votre option et la position que vous compter avoir ("call" ou "put"). Une fois toutes les informations récoltées, le pricer s'éxécute et fourni le pricing de l'option ainsi que, si souhaité en ajoutant une date future, l'estimation de la valeur de l'actif.

2.2 Architecture du programme

2.2.1 Vue générale : diagramme de classes UML

Sur ce diagramme, nous avons omis volontairement les getters et setters ainsi que les constructeurs. Egalement des fonctions permettant l'actualisation et l'estimation de la valeur de l'actif ne sont pas précisées dans le diagramme. Enfin, le fichier *tools*, contenant des fonctions permettant le pricing d'options, n'est pas précisée puisqu'il intervient de façon marginale dans l'architecture.

Le losange vide (de Asset vers Option) correspondent a une relation d'agrégation tandis que le losange noir correspond a une relation de composition. Quant aux flèches de Option vers les types d'options, elles indiquent un héritage.

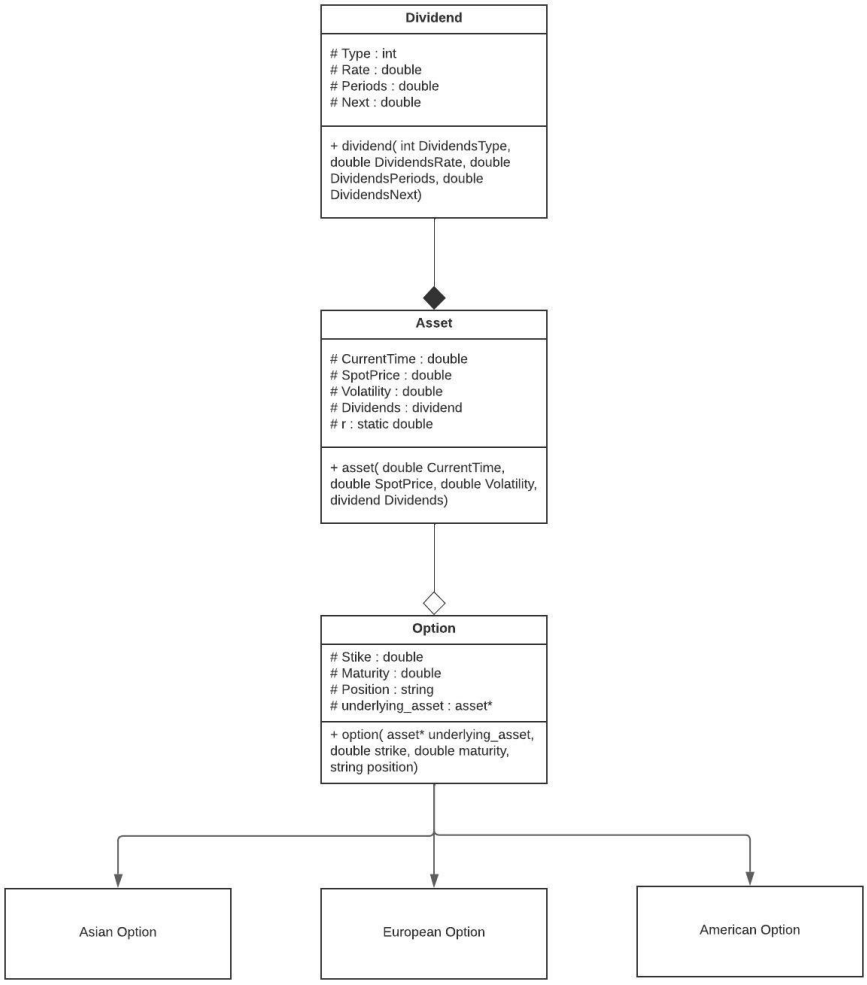

Figure 1: Enter Caption

2.2.2 Classe Asset et Dividend

La classe **Asset** permet de définir l'objet actif et les caractéristiques qui permettent de décrire un actif comme la date à laquelle est évalué l'actif, le prix spot de l'actif... Voici une description des variables membres de la classe Asset:

- CurrentTime : date à laquelle est évalué l'actif;
- SpotPrice : prix spot de l'actif;
- Volatility : volatilité de l'action;
- r : taux sans risque;
- Dividends : dividendes de l'actif (objet de la classe Dividend);

Nous avons créer une classe **Dividend** pour représenter l'objet qui correspond aux dividendes de l'action. Cette classe est agrégée à la classe Asset. Ses variables memebres sont :

- Type : type de la dividende, entier compris entre 0 et 2:
  - 0 : sans dividendes;
  - 1 : dividendes versées de façon discrète;
  - 2 : dividendes versées de façon continue;
- Rate : taux des dividendes;
- Periods : période à laquelle les dividendes de type discrète sont reversées en années (si la période est semestriel, alors on a une période de 0.5);
- Next : durée jusqu'au prochain paiement des dividendes de type 1;

2.2.3 Classe Option

La classe **Option** est une classe abstraite permettant de créer l'objet option. Ses variables membres sont :

- Stike : prix d'exercice de l'option;
- Maturity : date à laquelle l'option péri;
- Position : position que l'utilisateur veut prendre:
  - "call" : l'utilisateur a une option "call";
  - "put" : l'utilisateur a une option "put";
- UnderlyingAsset : pointeur vers un objet de la classe Asset

Suite à la définition de la classe Option, nous avons définis trois classes qui héritent de cette classe:

- EuropeanOption : pour le pricing des options européennes;
- AmericanOption : pour le pricing des options américaines;
- AsianOption : pour le pricing des options asiatiques;

3 Description fonctionnelle du programme

3.1 Les fonctions générales de classe

Les constructeurs et destructeurs ont été définis par défaut sauf pour la classe Asset où le constructeur par défaut a été implémenté ainsi que le destructeur. Les fonctions getter et setter sont implémentés pour chaque classe.

L'opérateur = a été surchargé dans les classes Asset et Dividend afin de simplifier l'implémentation dans le *main*. Ainsi lorsque dans le *main*, nous écrivons Asset1 = Asset2, on sous-entend que les deux objets ont les mêmes caractéristiques propres à la classe Asset.

De plus, les opérateurs iostream << et >> ont également été surchargé pour les classes Asset et Option. Cela permet d'éviter de surcharger le *main*.

3.2 Fonctions spécifiques aux classes

3.3 Classe Asset

Deux fonctions ont été implémentés et s'appliquent uniquement pour des objets Asset

- AssetActualization : permet de mettre à jour l'objet Asset. Cette fonction prend en argument:
  - NewTime : date à laquelle on veut actualiser l'actif;
  - SpotPrice : le prix actuel de l'actif;

  Elle retourne alors un objet Asset avec ses caractéristiques actualisées, en particulier ses dividendes.
- AssetEstimation : permet d'estimer le prix d'un actif à une date future. Elle prend en argument :
  - Time : date à laquelle on veut estimer l'actif;
  - RiskFreeRate : taux sans risque ;

  Elle retourne ensuite un objet Asset estimé à partir de l'espérance des trajectoires possibles.

3.3.1 Les fonctions de pricing

Comme exposé précédemment dans l'introduction, l'évaluation des options s'effectue en se basant sur le modèle de Black-Scholes. Pour les options européennes, on utilise des formules analytiques issues de ce modèle. Nous utilisons également la "put-call parity" afin de, à partir d'un option call européennes, trouver la valeur de l'option put. Cependant, pour les options asiatiques et américaines, on recourt à des techniques de Monte Carlo. Spécifiquement, le pricing des options asiatiques a été abordé avec l'application de la **méthode de Monte-Carlo**, réalisant l'évaluation par la simulation de multiples trajectoires du prix de l'actif sous-jacent et déterminant ensuite le prix moyen des options asiatiques à l'échéance pour chaque trajectoire simulée. Selon la loi des grands nombres, en moyennant et en actualisant ces prix à la date présente, on obtient la valeur estimée de l'option asiatique.

Le calcul précis du prix des options asiatiques requiert la moyenne du prix de l'actif pendant la période de détention de l'option. Pour ce faire, le temps a été discrétisé pour permettre la simulation d'une distribution normale N(0, 1) et ainsi générer un processus de Wiener, menant à une trajectoire des prix simulée.

Pour le pricing des options américaines, on utilise la **méthode des moindres carrés de Monte Carlo** (LSM), une technique introduite par **Longstaff et Schwartz** en 2001. Cette approche implique l'utilisation de régression polynomiale et la simulation de mouvements browniens. La discrétisation du temps et la simulation de multiples trajectoires sont représentées sous forme de matrices pour optimiser la gestion de la mémoire et faciliter les régressions polynomiales. Pour cela, l'intégration de la bibliothèque Eigen a été nécessaire. Les fonctions liées à cette matrice et à ces vecteurs de grandes dimensions acceptent uniquement des pointeurs en paramètres pour minimiser l'empreinte mémoire lors des appels de fonction. La simulation suit un processus similaire à celui utilisé pour le pricing des options asiatiques. Compte tenu de la complexité de la méthode de pricing et des nombreuses fonctions associées, nous avons créé un en-tête, tools.h, et un fichier d'implémentation, tools.cpp, pour structurer et organiser le code.

4 Problèmes rencontrés et solutions apportées

4.1 Prise en compte des problèmes liés à l'utilisateur

Un des problèmes rencontrés durant ce projet est la prise en compte des erreurs de l'utilisateur lorsqu'il doit rentrer les caractéristiques de l'actifs sur l'interface. En effet, lorsqu'il doit rentrer le type de l'option (american, asian ou european), s'il rentre un mauvais caractère, le programme continue sans relever l'erreur et donc ne peut pas fournir un pricing cohérent.

Pour éviter cela, nous avons mis en place une boucle booléenne qui permet de redemander le type de l'option tant qu'elle n'est pas correcte. Cette boucle est donc placée juste avant la requête interface de la position et se termine

que lorsque la réponse de l'utilisateur est "american", "asian" ou "european". De même une boucle a été créé afin de proposer à l'utilisateur de choisir si il veut pricer l'option, l'estimer, créer une nouvelle option ou alors quitter le programme.

4.2 Organisation du projet

Lorsque nous avions réaliser le pricer pour la première fois, nous avions mis toutes les commandes interfaces dans le *main*. Cependant, cela rendait le main très compliqué à lire et comprendre. Pour allerger le *main*, nous avons donc fait appel aux principes de surcharges. En effet, en surchargant les commandes de iostream << et >> dans les classes Asset et Option, nous avons pu répartir les commandes interfaces en fonctions de l'objet avec lequel nous travaillions (options ou dividendes) et ainsi rendre plus lisible et compréhensible le *main*.

5 Conclusion

Ce premier projet en C++ a été consacré à la mise en œuvre d'un pricer d'options. Durant cette initiative, notre attention s'est portée principalement sur l'utilisation de la programmation orientée objet, une approche qui se marie naturellement avec le langage C++. Nous avons apprécié la pertinence de C++ pour ce projet, notamment grâce à sa performance en termes de rapidité d'exécution et sa capacité à gérer efficacement les objets, offrant ainsi robustesse et intégrité au code. Cette rapidité était particulièrement notable lors des simulations intensives de nombreuses trajectoires nécessaires à la méthode de Monte Carlo pour le pricing d'options asiatiques et américaines.

En conclusion, notre programme a été conçu pour évaluer de manière efficiente les prix d'options européennes, avec ou sans dividendes, d'options asiatiques sans dividendes et d'options américaines également sans dividendes. Ce projet a non seulement renforcé notre compréhension des concepts financiers sous-jacents mais a également approfondi notre maîtrise de la programmation C++, démontrant la puissance de ce langage dans le domaine de la finance quantitative.

6 Annexe : Méthode de Pricing des options

6.1 European options

6.1.1 European sans dividends

- call: formule de Black-Scholes: $c = SN(d_1) - e^{-rT}N(d_2)$; où $N(\cdot)$ est la fonction de répartition de $\mathcal{N}(0,1)$
- put: call-put parity: $c - p = S - e^{-rT}K$

6.1.2 European lump payment dividend

- call: $c_{\text{lump payments}} = c((1-\delta)^n S_0, \sigma, T, K)$
- put: $p_{\text{lump payments}} = p_{\text{lump payments}} = (1-\delta)^n S_0 - e^{-rT}K$
- Remarque: dans le code, j'ai nommé $S_{\text{hat}} = e^{-\delta T}S_0$

6.1.3 European continuous dividend

- call: $c_{\text{continuous}} = c(e^{-\delta T}S_0, \sigma, T, K)$
- put: $p_{\text{continuous}} = p_{\text{continuous}} = e^{-\delta T}S_0 - e^{-rT}K$
- Remarque: dans le code, j'ai nommé $S_{\text{hat}} = e^{-\delta T}S_0$

6.2 Asian Options

6.2.1 Asian Call

Le prix estimé d'une option call asiatique peut être calculé en utilisant la formule suivante :

$$C = e^{-rT}\frac{1}{N}\sum_{i=1}^{N}\max\left(\bar{S}_i - K, 0\right)$$

où :

- C est le prix de l'option call asiatique.
- r est le taux d'intérêt sans risque annuel.
- T est le temps jusqu'à l'expiration.
- N est le nombre total de simulations.
- $\bar{S}_i$ est la moyenne arithmétique des prix simulés pour la i-ème trajectoire.
- K est le prix d'exercice de l'option.

6.2.2 Asian Put

De manière similaire, le prix estimé d'une option put asiatique est donné par :

$$P = e^{-rT} \frac{1}{N} \sum_{i=1}^{N} \max\left(K - \bar{S}_i, 0\right)$$

où P est le prix de l'option put asiatique.

6.3 American Options

Le pricing d'une option américaine sans dividende par simulation de Monte Carlo et la méthode de régression est résumé par les formules suivantes :

- Simulez les trajectoires du prix de l'actif sous-jacent.
- À chaque instant t, déterminez le payoff d'exercice immédiat:

$$\begin{aligned} &\text{Pour un call: } \max(S_t - K, 0), \\ &\text{Pour un put: } \max(K - S_t, 0). \end{aligned}$$

- Estimez la valeur de continuation et choisissez l'option qui maximise la valeur de l'option.
- La valeur initiale de l'option, V_0, est obtenue par:

$$V_0 = \max\left(\text{Payoff initial}, \text{Valeur actualisée des payoffs futurs}\right).$$

La valeur actualisée des payoffs futurs est calculée en actualisant les payoffs de toutes les trajectoires simulées à l'instant initial.