Projet C++ :
Pricer d'option

Gilles, Barthélémy, Cyprien, Gaetan

2024

Contents

1 Introduction

1.1 Contexte

Les marchés financiers jouent un rôle crucial dans l'économie mondiale, offrant des mécanismes permettant aux entreprises de lever des fonds et aux investisseurs de gérer leurs risques. Dans ce contexte, les produits dérivés, tels que les options financières, occupent une place centrale. Ces instruments complexes permettent de couvrir les risques associés à la volatilité des actifs sous-jacents tout en offrant des opportunités de spéculation. La modélisation financière est ainsi devenue un domaine incontournable pour les analystes et les ingénieurs financiers.

L'évaluation des options repose sur des outils mathématiques avancés et des algorithmes performants. Ces derniers permettent de prédire les prix des options en fonction de divers paramètres, notamment le prix de l'actif sous-jacent, la volatilité, les taux d'intérêt, et la durée jusqu'à l'échéance. Cependant, la complexité croissante des produits dérivés exige des solutions informatiques robustes et modulaires capables de traiter différents types d'options et de répondre à des besoins spécifiques. La modélisation financière n'est pas seulement une nécessité technique, mais un levier stratégique pour les institutions financières.

1.2 Objectifs du projet

Ce projet a pour ambition de modéliser et d'évaluer différents types d'options financières à l'aide de trois approches principales :

- Le modèle analytique de Black-Scholes, connu pour sa simplicité et son efficacité pour les options européennes.
- La méthode de simulation Monte Carlo, adaptée aux options dépendantes du chemin, telles que les options asiatiques.
- Le modèle Cox-Ross-Rubinstein (CRR), basé sur un arbre binomial, idéal pour les options américaines où l'exercice anticipé est autorisé.

L'objectif principal est de concevoir un outil en C++ à la fois modulaire et performant, exploitant les principes de la programmation orientée objet pour gérer une grande variété de scénarios financiers. Ce projet mettra en œuvre des concepts avancés, tels que l'héritage, le polymorphisme et les structures de données personnalisées.

2 Théorie et Concepts Financiers

2.1 Options financières : Définitions et types

Les options financières sont des contrats dérivés qui donnent à leur détenteur le droit, mais non l'obligation, d'acheter ou de vendre un actif sous-jacent à un prix prédéfini, appelé prix d'exercice (*strike*), avant ou à une date spécifique (*expiry*). Ces instruments sont couramment utilisés pour couvrir des risques financiers ou spéculer sur les mouvements de prix des actifs sous-jacents.

On distingue plusieurs types d'options:

- **Options *call***: Donnent le droit d'acheter l'actif sous-jacent à un prix fixé.
- **Options *put***: Donnent le droit de vendre l'actif sous-jacent à un prix fixé.

Les options peuvent également être classées selon leurs conditions d'exercice:

- **Options européennes**: Ne peuvent être exercées qu'à la date d'échéance.
- **Options américaines**: Peuvent être exercées à tout moment avant ou à la date d'échéance.
- **Options asiatiques**: Dépendent de la moyenne des prix de l'actif sous-jacent sur une période donnée.

Ces distinctions influencent fortement les modèles utilisés pour évaluer les prix des options.

2.2 Modèles mathématiques utilisés

L'évaluation des options repose sur plusieurs modèles mathématiques qui diffèrent par leurs hypothèses et leurs applications.

2.2.1 Modèle Black-Scholes

Le modèle Black-Scholes, introduit en 1973, repose sur des hypothèses strictes: un marché sans friction (pas de coûts de transaction), une volatilité constante et une absence d'opportunités d'arbitrage. Il fournit une solution analytique pour le prix des options européennes. La formule de prix pour une option *call* est donnée par:

$$C = S_0 N(d_1) - K e^{-rT} N(d_2),$$

avec:

$$d_1 = \frac{\ln(S_0/K) + \left(r + \frac{\sigma^2}{2}\right) T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T},$$

où S_0 est le prix initial de l'actif, K le prix d'exercice, T le temps à l'échéance, r le taux d'intérêt sans risque, σ la volatilité, et $N(x)$ la fonction de répartition de la loi normale.

Le *delta*, une mesure de la sensibilité du prix de l'option à une variation du prix de l'actif sous-jacent, est donné par:

$$\Delta = N(d_1).$$

2.2.2 Modèle Cox-Ross-Rubinstein (CRR)

Le modèle CRR, développé en 1979, est une approche discrète basée sur un arbre binomial. À chaque étape, le prix de l'actif peut augmenter ou diminuer selon des facteurs u (hausse) et d (baisse), avec des probabilités neutres au risque q et $1 - q$, respectivement. Ces paramètres sont définis par:

$$u = e^{\sigma\sqrt{\Delta t}}, \quad d = e^{-\sigma\sqrt{\Delta t}}, \quad q = \frac{e^{r\Delta t} - d}{u - d},$$

où $\Delta t = T/N$ est la taille d'une étape temporelle.

Le prix de l'option est calculé en remontant l'arbre, étape par étape, jusqu'à la racine:

$$H(i,j) = e^{-r\Delta t} \left[qH(i+1, j+1) + (1-q)H(i+1, j) \right],$$

où $H(i,j)$ est la valeur de l'option au niveau i et au nœud j.

2.2.3 Simulation Monte Carlo

La méthode Monte Carlo est utilisée pour modéliser des options dépendantes du chemin, comme les options asiatiques. Elle consiste à générer un grand nombre de trajectoires aléatoires pour l'actif sous-jacent en utilisant un processus de Wiener discretisé:

$$S_t = S_0 e^{\left(r - \frac{\sigma^2}{2}\right)t + \sigma\sqrt{t}Z},$$

où Z est une variable aléatoire suivant une loi normale standard.

Le prix de l'option est calculé comme la moyenne des *payoffs* actualisés:

$$C = e^{-rT} \frac{1}{M} \sum_{i=1}^{M} \text{payoff}(S^{(i)}),$$

où M est le nombre de trajectoires simulées.

2.2.4 Méthodes de réduction de variance (variables antithétiques)

Pour améliorer la précision des méthodes de simulation tout en limitant le nombre de trajectoires, on peut utiliser des techniques de **réduction de variance**. Parmi les plus courantes figure la méthode des *variables antithétiques* :

- On génère en paires les variables aléatoires $(Z, -Z)$ au lieu d'un seul Z.
- On calcule les deux payoffs correspondants et on en prend la moyenne.

L'idée est de *corréler négativement* les tirages pour qu'ils se compensent et réduisent la variance de l'estimateur, sans biaiser la moyenne. **Remarque:** D'autres techniques (comme l'importance sampling) peuvent également être employées pour des gains de précision similaires.

2.2.5 Volatilité stochastique : Modèle de Heston

En pratique, la volatilité n'est pas toujours constante comme dans Black-Scholes. Le **modèle de Heston** propose de rendre la variance ν_t *stochastique*:

$$\begin{cases} dS_t = \mu\, S_t\, dt + \sqrt{\nu_t}\, S_t\, dW_t^{(1)}, \\ d\nu_t = \kappa(\theta - \nu_t)\, dt + \xi\sqrt{\nu_t}\, dW_t^{(2)}, \end{cases}$$

avec une corrélation ρ entre $W_t^{(1)}$ et $W_t^{(2)}$. Ce modèle décrit mieux la réalité des marchés (effet de levier, *clustering* de la volatilité, etc.).

3 Approche Technique et Implémentation

L'architecture du projet repose sur une conception orientée objet, avec une hiérarchie de classes permettant une modularité et une extensibilité optimales. Le diagramme UML ci-dessous illustre les relations entre les classes principales:

3.1 Classe abstraite : Option

Description : La classe `Option` est une classe abstraite qui représente les propriétés communes à toutes les options financières. Elle sert de base pour les classes dérivées.
Attributs :

- `double _expiry` : Date d'échéance de l'option.

Méthodes :

- `Option()` : Constructeur par défaut.
- `Option(double expiry)` : Constructeur avec initialisation de l'échéance.
- `double getExpiry()` : Retourne la date d'échéance.
- `virtual double payoff(double) = 0` : Méthode virtuelle pure pour calculer le payoff d'une option.
- `virtual double payoffPath(vector<double>& prices)` : Méthode virtuelle pour calculer le payoff sur un chemin.
- `virtual bool isAsianOption()` : Identifie si l'option est asiatique.
- `virtual bool isAmericanOption()` : Identifie si l'option est américaine.

—

3.2 Classe dérivée : EuropeanVanillaOption

Description : La classe `EuropeanVanillaOption` est une classe dérivée qui représente les options européennes classiques.
Attributs :

- `double _strike` : Prix d'exercice de l'option.

Méthodes :

- `EuropeanVanillaOption(double expiry, double strike)` : Constructeur avec initialisation de l'échéance et du prix d'exercice.
- `double getStrike()` : Retourne le prix d'exercice.
- `virtual optionType GetOptionType()` : Retourne le type d'option (`call` ou `put`).

—

3.3 Sous-classes : CallOption et PutOption

Description : Ces classes représentent les options européennes *call* et *put*.
Attributs (hérités) :

- Aucun attribut supplémentaire.

Méthodes spécifiques :

- `double payoff(double)` : Calcule le payoff spécifique à une option *call* ou *put*.
- `optionType GetOptionType()` : Retourne le type d'option (`call` ou `put`).

—

3.4 Classe dérivée : AsianOption

Description : La classe `AsianOption` représente les options asiatiques, dont le payoff dépend du chemin suivi par le prix du sous-jacent.
Attributs :

- `double _strike` : Prix d'exercice.
- `vector<double> timeSteps` : Points temporels pour le calcul des prix moyens.

Méthodes :

- `AsianOption(vector<double>&, double strike)` : Constructeur pour initialiser les points temporels et le prix d'exercice.
- `double payoff(double)` : Calcule le payoff en fonction de la moyenne des prix.
- `double payoffPath(vector<double>&)` : Calcule le payoff basé sur un chemin donné.

—

3.5 Classe dérivée : AmericanOption

Description : La classe `AmericanOption` représente les options américaines, qui permettent l'exercice anticipé.
Attributs :

- `double _strike` : Prix d'exercice.

Méthodes :

- `AmericanOption(double expiry, double strike)` : Constructeur avec initialisation de l'échéance et du prix d'exercice.
- `double payoff(double)` : Calcule le payoff.
- `bool isAmericanOption()` : Retourne `true`, car l'option est américaine.

—

3.6 Classe utilitaire : BinaryTree

Description : `BinaryTree` est une structure générique utilisée pour représenter les arbres binomiaux nécessaires dans le modèle CRR.
Attributs :

- `int _depth` : Profondeur de l'arbre.
- `vector<vector<T>> _tree` : Conteneur pour les nœuds de l'arbre.

Méthodes :

- `BinaryTree()` : Constructeur par défaut.
- `void setDepth(int)` : Définit la profondeur de l'arbre.
- `void setNode(int step, int node, T value)` : Définit la valeur d'un nœud.
- `T getNode(int step, int node)` : Récupère la valeur d'un nœud.

—

3.7 Pricers : BlackScholesPricer, BlackScholesMCPricer, CRRPricer

3.7.1 BlackScholesPricer

Description : Implémente les formules analytiques du modèle Black-Scholes.
Méthodes :

- `double operator()()` : Calcule le prix de l'option.
- `double delta()` : Calcule la sensibilité du prix de l'option au prix du sous-jacent.

3.7.2 BlackScholesMCPricer

Description : Implémente la simulation Monte Carlo pour les options dépendantes du chemin.
Méthodes :

- `void generate(int nb_paths)` : Génère des trajectoires aléatoires.
- `double operator()()` : Retourne le prix calculé.
- `vector<double> confidenceInterval()` : Retourne un intervalle de confiance à 95

3.7.3 CRRPricer

Description : Implémente le modèle Cox-Ross-Rubinstein (CRR) pour les options.
Attributs :

- `BinaryTree<double>` : Arbre pour les prix des options.
- `BinaryTree<bool>` : Arbre pour les décisions d'exercice.

Méthodes :

- `void compute()` : Construit l'arbre et calcule les prix.
- `double operator()(bool closed_form)` : Retourne le prix de l'option, avec ou sans formule fermée.
- `bool getExercise(int, int)` : Retourne si l'option doit être exercée à une étape donnée.

Les classes sont organisées de manière à refléter les différents types d'options et les pricers associés, tout en séparant clairement les responsabilités.

3.8 Choix techniques

Le choix du langage C++ pour ce projet repose sur plusieurs avantages:

- **Gestion de la mémoire**: L'utilisation explicite de la mémoire permet un contrôle précis des allocations dynamiques, ce qui est essentiel pour les simulations Monte Carlo et la gestion des arbres binomiaux.
- **Vitesse d'exécution**: Grâce à sa compilation en code natif, C++ offre une performance optimale pour les calculs intensifs.
- **Modularité et extensibilité**: La programmation orientée objet facilite l'ajout de nouveaux types d'options ou de pricers.

Des concepts avancés comme l'héritage et le polymorphisme ont été largement utilisés pour permettre une gestion polymorphique des options. De plus, les exceptions ont été employées pour gérer les erreurs, par exemple lors de la validation des paramètres (dates négatives, valeurs hors limites, etc.).

3.9 Explications des algorithmes

3.9.1 Algorithme Monte Carlo et variables antithétiques

Pour la **simulation Monte Carlo**, nous générons des trajectoires du sous-jacent selon le processus géométrique brownien:

$$S_{t+\Delta t} = S_t \exp\Big(\big(r - \tfrac{1}{2}\sigma^2\big)\Delta t + \sigma\sqrt{\Delta t}\, Z\Big),$$

où $Z \sim \mathcal{N}(0,1)$. Nous calculons ensuite les *payoffs* correspondants et faisons la moyenne actualisée.

Implémentation en C++. Dans le code, nous disposons d'une classe `BlackScholesMCPricer` qui :

1. Stocke les paramètres `(S0, r, sigma, nb_paths, etc.)`.
2. Possède une méthode `generate(nb_paths)` qui, pour chaque trajectoire, tire des variables normales $\{Z_i\}$, met à jour le prix et calcule le payoff.
3. Cumule la somme et la somme des carrés des payoffs pour estimer la variance.

Réduction de variance par variables antithétiques. Pour réduire la variance, nous avons ajouté un `bool antithetic` dans notre pricer. Si `antithetic = true`, chaque tirage Z est complété par $-Z$. Ainsi, au lieu de calculer un seul payoff, on en calcule deux ($payoff(Z)$ et $payoff(-Z)$) et on prend leur moyenne. En pratique, le code prévoit:

- Un test sur `antithetic` dans `generate()`.
- Si `true`, on génère des paires $(Z, -Z)$ et on cumule la *moyenne* de leurs payoffs.

Ceci améliore la précision tout en gardant la même espérance.

3.9.2 Algorithme CRR

Le modèle Cox-Ross-Rubinstein (CRR) utilise un **arbre binomial**, où chaque nœud se ramifie selon une hausse (u) ou une baisse (d). Après avoir construit l'arbre des prix, on remonte en évaluant la valeur de l'option à chaque étape.

Implémentation en C++. Nous avons créé une classe `CRRPricer` qui :

1. Construit un arbre binaire de profondeur N pour les prix sous-jacents.
2. Calcule les payoffs à la date finale.
3. Remonte l'arbre en utilisant la formule de valorisation neutre au risque.

En cas d'option américaine, on compare la valeur de continuation avec le payoff d'exercice anticipé à chaque nœud.

3.9.3 Volatilité stochastique : Modèle de Heston

Pour tenir compte d'une volatilité *non constante*, nous avons implémenté le **modèle de Heston**, où la variance ν_t suit une SDE couplée:

$$\begin{cases} dS_t = \mu\, S_t\, dt + \sqrt{\nu_t}\, S_t\, dW_t^{(1)}, \\ d\nu_t = \kappa(\theta - \nu_t)\, dt + \xi\sqrt{\nu_t}\, dW_t^{(2)}, \end{cases}$$

avec ρ comme corrélation entre $W_t^{(1)}$ et $W_t^{(2)}$.

Implémentation en C++. Nous avons une classe `HestonMCPricer` qui :

- Gère deux vecteurs de variables normales $\{Z_1, Z_2\}$ corrélés par ρ.
- Simule pas à pas, pour chacun des $nSteps$, la mise à jour de S et de ν.
- Stocke le payoff final (ou la moyenne si option asiatique) avant de l'actualiser.

Pour chaque trajectoire, on initialise ν_0 (variance initiale) et S_0. À chaque pas Δt, on met à jour successivement $\nu_{t+\Delta t}$ et $S_{t+\Delta t}$. Les payoffs sont ensuite moyennés pour obtenir le prix. Le calcul de la *confidence interval* et de la variance est similaire à la Monte Carlo standard.

4 Conclusion

4.1 Conclusion de l'analyse

Cette étude a permis de valider les performances des modèles implémentés et de souligner leurs forces et faiblesses. Les résultats mettent en lumière l'importance de comprendre les spécificités de chaque modèle pour répondre aux besoins des acteurs financiers. Les pistes d'amélioration envisagées incluent l'optimisation des algorithmes existants et l'intégration de nouvelles fonctionnalités pour élargir les cas d'utilisation.

Bien que le projet ait atteint ses objectifs principaux, plusieurs pistes d'amélioration pourraient être envisagées pour enrichir et optimiser l'outil:

4.1.1 Optimisation des algorithmes

L'efficacité des algorithmes pourrait être améliorée en explorant les techniques suivantes:

- **Parallélisation** : Utilisation de calculs parallèles pour accélérer la méthode Monte Carlo, en répartissant les trajectoires sur plusieurs cœurs de processeur.
- **Réduction de variance** : Implémentation de méthode comme les variables antithétiques pour améliorer la précision des simulations Monte Carlo avec moins de trajectoires.
- **Compression d'arbre** : Optimisation de la structure de l'arbre binomial pour réduire la mémoire et le temps nécessaires dans le modèle CRR.
- **Modèle stochastic** : Le modèle de Heston présente un intérêt majeur en ce qu'il capture la nature stochastique de la volatilité, reflétant plus fidèlement la dynamique réelle des marchés financiers (volatilité non constante, effet de levier, etc.). De cette manière, il permet de mieux modéliser certaines caractéristiques empiriques observées dans les données de marché, que le modèle de Black–Scholes (à volatilité constante) ne peut pas prendre en compte

4.1.2 Intégration dans un logiciel financier

Enfin, l'intégration de l'outil dans une application logicielle complète pourrait élargir son utilisation pratique:

- **Interface utilisateur** : Développement d'une interface graphique pour permettre aux utilisateurs d'entrer des paramètres, de choisir un modèle, et de visualiser les résultats sous forme de graphiques interactifs.
- **Connexion à des flux de données** : Intégration avec des API financières (par exemple Bloomberg ou Yahoo Finance) pour récupérer les données de marché en temps réel.
- **Extensions analytiques** : Ajout de fonctionnalités pour calculer les sensibilités (grecs financiers) ou pour analyser les portefeuilles contenant plusieurs options.

—

4.2 Conclusion générale

En conclusion, ce projet démontre la faisabilité de construire un outil modulaire et performant pour la tarification des options financières. Les modèles implémentés couvrent les besoins fondamentaux des options européennes, américaines, et asiatiques. Cependant, les perspectives identifiées montrent qu'il existe encore un fort potentiel d'amélioration, tant sur le plan des fonctionnalités que des performances.

Ces évolutions permettraient de transformer cet outil académique en une solution pratique et efficace pour les professionnels des marchés financiers, tout en répondant aux défis croissants liés à la complexité des produits dérivés modernes.