Pricer

Projet de C++

Créer un programme pour déterminer le prix d'une option financière, ainsi que la stratégie de réplication dans le modèle de Black-Scholes-Merton. Dans la plupart des cas où les formules explicites ne s'appliquent pas, on calculera ces prix par méthode de Monte-Carlo.

Pierre-Nicolas Létocart
Mihnea Popa
Fangyu Xue

Sommaire

Introduction

Ce compte-rendu a pour but de présenter l'architecture du code de notre projet ainsi que la stratégie adoptée. L'objectif est de créer une classe Class_Option qui regroupe les caractéristiques de notre option ainsi qu'une fonction qui détermine le prix de l'option mais qui n'est pas codée. Puis, par le biais de deux classes nommées Black_Scholes_Merton et Monte_Carlo, nous allons calculer le prix de l'option supposée européenne suivant qu'il s'agisse d'une option put ou call.

1 Classe Class_Option

1.1 Constructeurs et destructeur

Nous utilisons le destructeur par défaut, ainsi, il y a besoin de définir uniquement le constructeur avec paramètres. Le constructeur sans paramètres et le constructeur par copie ne seront pas utilisés dans notre programme.

1.2 Les variables

Nous implémentons toutes les caractéristiques d'une option dans des variables protected pour y avoir accès dans la définition des classes filles. Il s'agit du prix initial de l'actif, du strike price, de la maturité de l'option, la volatilité de l'actif ainsi que le taux d'intérêt. Enfin, pour calculer le prix de l'option correctement, nous utilisons un booléen type qui vaut true si l'option est call et false si l'option est put.

1.3 Les fonctions à implémenter

Dans la classe mère, nous introduisons la fonction nommée price, qui est une fonction polymorphe qui sera redéfinie dans les classes héritées.

2 Classe Black_Scholes_Merton

2.1 L'héritage de la classe Class_Option

Dans cette classe fille, on considère les mêmes variables, le même constructeur et aussi le même destructeur que dans la classe mère.

2.2 Les nouvelles fonctions pour séparer les cas put et call

Afin d'éviter d'avoir une fonction price trop longue, nous décidons de créer deux fonctions propres à la classe qui permettent de calculer le prix d'une option call et d'une option put dans le modèle BSM en se basant sur la parité put-call pour les options européennes. Nous utilisons les formules calculées dans le cours d'instruments financiers à l'ENSAE qui font intervenir la fonction de répartition de la loi normale centrée réduite notée CDF. Cependant, dans C++, cette fonction de répartition n'est pas directement

implémentée, on va la déduire de la fonction d'erreur complémentaire erfc. La fonction d'erreur complémentaire erfc est définie par:

$$\text{erfc}(z) = \frac{2}{\sqrt{\pi}} \int_z^{\infty} e^{-t^2}\, dt$$

En effectuant le changement de variable $u = -\frac{t}{\sqrt{2}}$ dans l'expression de la fonction de répartition de la loi normale centrée réduite, on abouti à

$$CDF(x) = \frac{1}{2}\text{erfc}\left(-\frac{x}{\sqrt{2}}\right)$$

2.3 Le polymorphisme de la fonction price

On peut à présent redéfinir la fonction polymorphe price en utilisant le type de notre option pour faire appel à la bonne fonction.

3 Classe Monte_Carlo

3.1 L'héritage de la classe Class_Option

Dans cette classe fille, on considère les variables, le constructeur et aussi le destructeur définits dans la classe mère. Cependant, on rajoute une variable privée, qui correspond au nombre de simulations pour le calcul par Monte-Carlo.

3.2 Le polymorphisme de la fonction price

Dans cette classe, on a décidé d'écrire une unique fonction qui traite le cas put et le cas call. Cette fonction découle de l'algorithme de Mersenne Twister utilisé pour générer des nombres pseudo aléatoires et nous permet d'approximer le prix de l'option.

4 Main

4.1 Définition d'une option test

Dans le main nous définissons les variables qui caractérisent une option dont on va calculer le prix ensuite.

4.2 Calcul du prix de l'option

On calcule le prix du call et du put selon le modele de Black-Scholes-Merton et suivant la simulation. Enfin, on vérifie la parité put-call via la différence des prix des deux options obtenues via simulation.