

Rapport Projet C++ : Modèle de Volatilité Stochastique, Schema de Broadie-Kaya et Pricing Monte-Carlo

Samir GUEBLAOUI

7 avril 2024

Table des matières

1	Introduction	2
2	Structure du Projet	2
3	Résultats	4

1 Introduction

Le modèle de Heston est reconnu pour sa capacité à capturer les fluctuations de la volatilité des actifs financiers, en offrant une approche plus réaliste pour la tarification des options par rapport au modèle de Black-Scholes. Cependant, la complexité des équations différentielles stochastiques associées au modèle de Heston peut rendre sa mise en œuvre numérique ardue. Pour résoudre ce défi, le schéma de Broadie-Kaya émerge comme une méthode numérique efficace pour calculer les prix des options selon le modèle de Heston.

L'objectif principal de ce projet est de développer une bibliothèque en C++ permettant d'évaluer les prix des options en utilisant le modèle de Heston avec le schéma de Broadie-Kaya. Cette bibliothèque fournira des outils pour la tarification d'une gamme d'options.

2 Structure du Projet

Le code du projet sera divisé en cinq parties distinctes, chacune jouant un rôle crucial dans l'accomplissement des objectifs du projet.

1. Définition et Implémentation du Modèle : (**ModelHeston.h** et **ModelHeston.cpp**)

La première partie du code consistera à définir et à implémenter le modèle de volatilité stochastique, avec une classe mère décrivant le modèle de volatilité stochastique de base. Ensuite, une classe fille spécifique au modèle de Heston sera développée, héritant des propriétés de la classe mère et décrivant les caractéristiques spécifiques du modèle de Heston.

2. Simulation des Trajectoires : (**PathSimulatorHeston.h** et **PathSimulatorHeston.cpp**)

La deuxième partie du code sera dédiée à la simulation des trajectoires des prix des actifs financiers, en utilisant le schéma de Broadie-Kaya. Ce schéma discret permettra de générer des trajectoires réalistes en discrétisant les équations du modèle de Heston, ce qui sera essentiel pour évaluer les prix des options de manière précise.

3. Définition et Implémentation d'une Classe Option : **(PayoffHeston.h et PayoffHeston.cpp)**

La troisième partie du code consistera à définir et à implémenter une classe Payoff. Cette classe permettra de représenter différents types d'options financières, telles que les options européennes, américaines ou exotiques. Elle inclura une méthodes pour calculer le prix de l'option.

4. Pricing Monte Carlo : **(MonteCarloHeston.h et MonteCarloHeston.cpp)**

La quatrième partie du code sera consacrée à la tarification des options en utilisant la méthode de Monte Carlo. Cette méthode permettra de calculer le prix de l'option en simulant un grand nombre de trajectoires des prix des actifs et en moyennant les payoffs obtenus à la date d'expiration. Cette approche offre une solution numérique robuste pour évaluer les prix des options dans le cadre du modèle de Heston.

5. Transfert des Résultats sur Python et Traçage des Trajectoires : **(main.cpp et GrapheHeston.py)**

Enfin, la cinquième partie du code consistera à transférer les résultats obtenus en C++ vers Python, afin de profiter des fonctionnalités de traçage et de visualisation disponibles dans ce langage. Les trajectoires simulées des prix des actifs seront tracées pour permettre une analyse approfondie du comportement des options dans différents scénarios.

3 Résultats

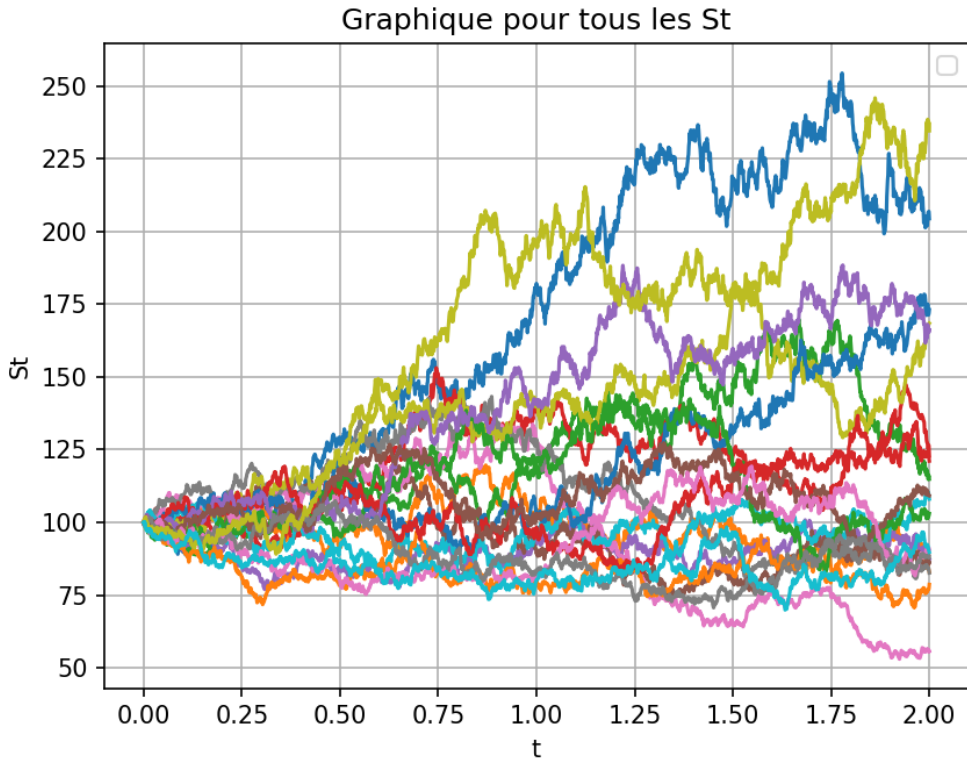
On a le modèle de Heston :

$$\begin{cases} \frac{dS_t}{S_t} = r(t)dt + \sqrt{V_t}dW_t^S \\ dV_t = \kappa(\theta - V_t)dt + \sigma\sqrt{V_t}dW_t^V \\ d\langle W_t^S, W_t^V \rangle = \rho dt \end{cases}$$

Appliquons notre code à une option européen de strike 105 de maturité 2 sur un modèle de Heston tel que :

$$S_0 = 100 \quad \kappa = 3 \quad \theta = 0.3 \quad \sigma = 0.2 \quad \rho = 0.5 \quad V_0 = 0.1 \quad r(t) = 0.1$$

En utilisant en python les resultats de 20 vecteurs path généré à l'aide d'objets de notre classe Modèle de Heston avec les paramètres ci-dessus on peut tracer le graphique suivant :



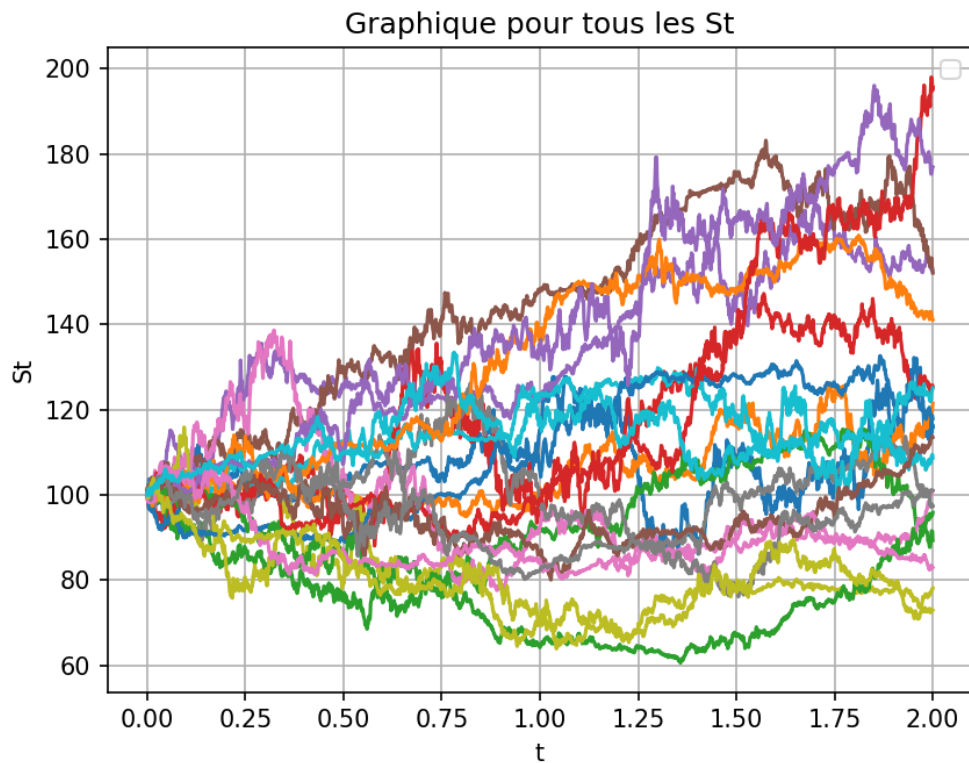
On obtient alors le prix de l'option avec la fonction `price()` de notre classe MonteCarlo :

```
Console de débogage Microsoft Visual Studio
Prix du CALL European: 12.8298
Prix du PUT European: 18.1362
```

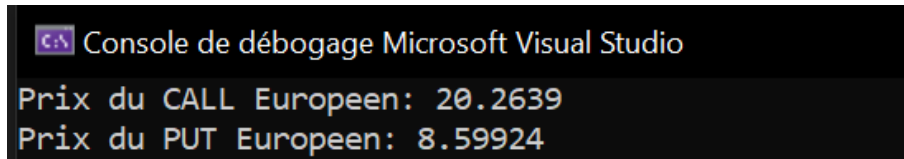
Appliquons cette fois notre code à une option européen de strike 105 de maturité 2 sur un modèle de Heston tel que :

$$S_0 = 100 \quad \kappa = 3 \quad \theta = 0.1 \quad \sigma = 0.7 \quad \rho = 0.7 \quad V_0 = 0.1 \quad r(t) = 0.1$$

Avec ces paramètres on obtient alors le graphe suivant :



Et on obtient pour l'option européenne le prix suivant :

A screenshot of the Microsoft Visual Studio debug console. The title bar at the top reads "Console de débogage Microsoft Visual Studio". Below the title bar, the text "Prix du CALL Europeen: 20.2639" is displayed on one line, and "Prix du PUT Europeen: 8.59924" is displayed on the line below it. The text is in a light-colored monospace font against a dark background.

```
Console de débogage Microsoft Visual Studio
Prix du CALL Europeen: 20.2639
Prix du PUT Europeen: 8.59924
```

Conclusion : En faisant varier uniquement θ , σ , ρ , le prix d'un call européen est moins élevé que celui du put dans le premier modèle de Heston, alors que pour le deuxième on obtient l'inverse.