Documentation Technique - Pricing de Stratégies Optionnelles et Produits Structurés

1. Introduction

Ce projet a pour objectif de développer une solution C# permettant de pricer des stratégies optionnelles et des produits structurés à l'aide de la simulation de Monte Carlo et de divers modèles de volatilité. Deux modèles sont principalement utilisés :

- Le modèle Stochastic Volatility Inspired (SVI), qui est utilisé pour calibrer une surface de volatilité en fonction des données du marché.
- Le modèle d'Heston, un modèle de volatilité stochastique utilisé dans la simulation de Monte Carlo pour générer des trajectoires de prix avec une incertitude sur la volatilité.

2. Modèle de Monte Carlo

Le modèle de Monte Carlo est une méthode de simulation numérique qui permet d'estimer le prix des produits dérivés en générant de nombreuses trajectoires de prix.

La méthode de Monte Carlo consiste à simuler de multiples scénarios aléatoires en suivant un processus stochastique donné, puis à calculer la moyenne des payoffs obtenus. Dans le contexte du pricing d'options, chaque trajectoire simule l'évolution du prix de l'actif sous-jacent sur la période de vie de l'option.

La simulation de Monte Carlo repose sur la discrétisation des processus stochastiques. Dans le cas des options, le prix de l'actif sous-jacent peut être modélisé, par exemple, par un processus de type Brownien Géométrique :

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

où S_t est le prix de l'actif à l'instant t, μ est le taux de drift, σ est la volatilité, et $W_t$est un processus de Wiener représentant le terme de bruit aléatoire.

La discrétisation de ce processus est réalisée en utilisant le schéma d'Euler-Maruyama, qui approxime les mouvements continus du prix sur de petits intervalles de temps Δt :

$$S_{\{t+\Delta t\}} = S_t + \mu S_t \Delta t + \sigma S_t \sqrt{\Delta t}\, Z$$

où Z est une variable aléatoire tirée d'une loi normale standard.

- **<u>Implémentation en C#</u>**

L'implémentation en C# repose sur la simulation de N trajectoires pour modéliser l'évolution du prix de l'actif. Chaque trajectoire représente une simulation indépendante du processus stochastique, et le prix final de l'option est estimé en prenant la moyenne des payoffs actualisés calculés sur l'ensemble des trajectoires simulées.

$$V^0 = e^{-rT} \frac{1}{N} \sum_{i=1}^{N} Payoff(S_t^i)$$

Pour les produits dont le payoff dépend pas seulement du prix final, comme les options à barrières, nous avons utilisé la méthode de discrétisation d'Euler pour simuler l'évolution des trajectoires à plusieurs points dans le temps. Cette méthode a également été appliquée lors du pricing des produits avec volatilité stochastique ainsi que pour les Autocalls, où un prix est simulé à chaque date d'observation.

Pour les autres produits, afin d'optimiser les temps d'exécution, nous ne simulons que le prix final pour chaque trajectoire.

<u>Tirages de variables normalement distribuées</u>

La méthode de Box-Muller est une technique utilisée pour générer des variables aléatoires suivant une loi normale à partir de deux variables uniformément distribuées.

La méthode fonctionne en prenant deux variables aléatoires indépendantes qui sont uniformément distribuées sur l'intervalle [0,1]. Ces deux variables sont ensuite transformées en deux variables normales indépendantes via la transformation suivante :

$$Z_1 = \sqrt{-2 \ln (U_1)} \times \cos(2\pi U_2)$$

$$Z_2 = \sqrt{-2 \ln (U_1)} \times \sin(2\pi U_2)$$

3. Produits

Notre objectif était de concevoir une solution permettant de pricer différents types de produits financiers, allant des options vanilles à des stratégies optionnelles complexes, ainsi que des produits structurés comme les autocalls.

Pour répondre à cette diversité de produits, nous avons développé plusieurs interfaces en C#, structurées de manière modulaire pour faciliter l'extension et la réutilisation du code. Deux interfaces principales ont été mises en place :

- Interface commune à tous les produits financiers : Celle-ci gère les fonctionnalités de base et les éléments partagés, tels que la fonction payoff.
- Interface spécifique aux produits optionnels : Dédiée aux produits dérivés complexes, cette interface gère les paramètres propres aux options et aux stratégies optionnelles, notamment l'intégration de formules fermées pour certaines options.

Ces interfaces ont été conçues pour offrir une expérience utilisateur fluide, tout en restant flexible afin de permettre l'ajout facile de nouveaux types de produits.

Nous avons ensuite créé plusieurs classes abstraites pour modéliser les différentes familles de produits financiers :

- **Classe abstraite pour les options** : Cette classe sert de base pour tous les types d'options simples, comme les options vanilles ou les options à barrières. Elle contient des méthodes et attributs communs (strike, maturité, etc.). Pour certains produits spécifiques, comme les options à barrières ou binaires, des classes abstraites supplémentaires ont été créées, intégrant des attributs spécifiques tels que la barrière ou le coupon.

- **Classe abstraite pour les stratégies optionnelles :** Elle regroupe les stratégies combinant plusieurs options, comme les spreads (call/put spread), les straddles ou encore d'autres combinaisons complexes. Chaque stratégie est décomposée en plusieurs options à l'aide d'un dictionnaire, où chaque option est définie et organisée avec ses quantités par produit.

Ces classes abstraites permettent une gestion efficace des produits en héritant des comportements communs, tout en permettant des spécialisations adaptées à chaque type de produit.

Gestion des produits structurés (Autocalls)

Pour les produits autocalls, une classe abstraite commune a été créée, avec des méthodes et attributs partagés. Les différences entre les produits Autocall résident principalement dans le payoff, qui varie d'un type d'autocall à l'autre.

- **Autocall Phoenix** : Ce produit structuré offre un coupon périodique tant que l'actif sous-jacent reste au-dessus d'une barrière à chaque date d'observation. Il inclut une option de remboursement anticipé si le sous-jacent dépasse un seuil prédéfini à une de ces dates, mettant ainsi fin au contrat avant la maturité. En cas de franchissement à la baisse d'une barrière, l'investisseur peut subir des pertes en capital à l'échéance.

- **Autocall Athena :** Contrairement au Phoenix, les coupons ne sont pas versés immédiatement à chaque date d'observation, mais sont gardés en mémoire. Ils ne sont payés qu'à l'échéance si l'actif sous-jacent se trouve au-dessus de la barrière ou si le produit est rappelé en anticiper. Le risque de perte en capital reste similaire, mais l'Athena est plus risqué que le Phoenix, offrant en contrepartie un coupon plus élevé pour des paramètres identiques.

L'utilisateur a la possibilité de définir les barrières et la fréquence des observations, offrant ainsi une flexibilité dans la personnalisation du produit.

4. Modèle de Volatilité SVI

Le modèle SVI (Stochastic Volatility Inspired) est utilisé pour construire une surface de volatilité en fonction des différentes expirations et des strike (structure par terme du smile de volatilité).

- **Formulation mathématique :**

Le modèle SVI repose sur la paramétrisation suivante de la variance implicite :

$$\sigma_{BS}(K)^2 = a + b \times (\rho(K - m) + \sqrt{\theta^2 + (K - m)^2})$$

où :

- $\sigma_{BS}(K)$ est la volatilité implicite en fonction du log strike K (ou de la log Moneyness K/S
- a représente le niveau global de la variance
- b est la pente de la courbe de variance
- ρ contrôle l'asymétrie du smile
- m est le décalage horizontal du smile
- σ est la volatilité au prix d'exercice.

- **Calibration :**

La calibration du modèle SVI consiste à ajuster ces paramètres pour qu'ils correspondent au mieux aux données de volatilité implicite des options cotées sur le marché. Cela se fait généralement par la minimisation d'une fonction objectif représentant l'erreur quadratique entre les valeurs observées et les valeurs fournies par le modèle.

On peut considérer de manière générale toutes les fonctions objectif de la forme suivante :

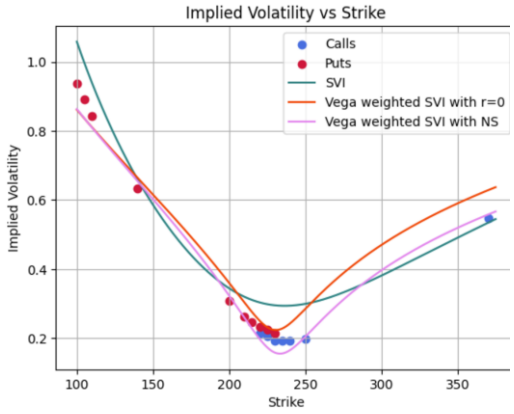
$$min_{a,b,\rho,m,\theta} \sum_{i=1}^{n} \lambda_i \left(\sigma_{SVI}(x; a, b, \rho, m, \theta)^2 - \sigma_i^2\right)^2$$

Avec :

- σ_i^2 : la variance implicite observée sur le marché
- $\sigma_{BS}(x; a, b, \rho, m, \theta)^2$: la variance SVI
- λ_i : les poids attribués à chaque observation

Dans notre implémentation du code en C#, nous avons choisi de rester sur un cas basique où $\lambda_i = 1$. Mais on peut noter que les poids peuvent souvent prendre la forme d'une fonction qui a des valeurs importantes près de la monnaie afin de mieux pondérer les options ATM (les plus nombreuses).
Pour cela, on utilise souvent le Vega de l'option observée.

Voici un exemple de résultat que nous avons obtenu en faisant l'implémentation en Python :



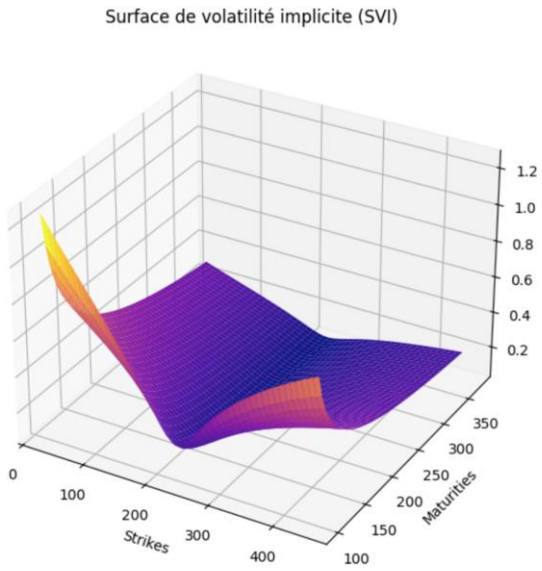
La courbe verte représente un SVI sans poids.

La courbe orange représente un SVI pondéré par les Vega des options en prenant un taux nul.

La courbe orange représente un SVI pondéré par les Vega des options en prenant un taux obtenu par interpolation de Nelson-Siegel

Une fois la calibration effectuée, cette surface de volatilité peut être utilisée pour pricer des produits dérivés plus complexes et alimenter des modèles de simulation comme celui de Monte Carlo.
Pour cela, on se place sur la surface de volatilité au couple (Strike, Maturité) correspondant.

On obtient une surface représentant au mieux la structure par terme des smiles de volatilité :



5. Modèle de Heston

Le modèle de Heston est un modèle de volatilité stochastique couramment utilisé dans les marchés financiers. Contrairement aux modèles à volatilité constante, il permet de modéliser la dynamique de la volatilité elle-même comme un processus stochastique. Le modèle de Heston repose sur le système d'équations différentielles stochastiques suivant :

$$dS_t = \mu S_t dt + \sqrt{V_t} S_t dW_t^S$$

$$dV_t = \kappa\,(\theta\ -\ V_t)\mathrm{dt} + \sigma\sqrt{V_t} dW_t^V$$

où :

- S_t est le prix de l'actif sous-jacent,
- $V_t$est la variance instantanée,
- W_t^S et W_t^V sont deux processus de Wiener corrélés avec une corrélation ρ
- κ représente la vitesse de réversion
- θ est la variance long terme
- σ est la volatilité de la volatilité

- **<u>Discrétisation pour la simulation de Monte Carlo</u>**

La discrétisation du modèle de Heston suit également le schéma d'Euler-Maruyama. Le principal défi réside dans la gestion de la variance instantanée, qui doit rester positive.

Le processus de Monte Carlo est similaire à celui décrit dans la section 2, avec l'ajout de la dynamique de la variance stochastique V_t pour chaque trajectoire simulée.

$$S_{\{t+\Delta t\}} = S_t + \mu S_t \Delta t + \sigma S_t \sqrt{\Delta t}\, Z$$

$$V_{\{t+\Delta t\}} = V_t + \kappa\,(\theta\ -\ V_t)\Delta t + \sigma\sqrt{V_t}\sqrt{\Delta t}\, Z$$

- **<u>Calibration :</u>**

La calibration du modèle d'Heston se fait de la même manière que le modèle SVI précédemment présenté, en utilisant une fonction objectif du type :

$$min_{a,b,\rho,m,\theta} \sum_{i=1}^{n} \lambda_i \left(\sigma_{Heston}(x; a, b, \rho, m, \theta)^2 - \sigma_i^2\right)^2$$

Cette calibration est trop complexe et sort du cadre du projet car on doit calculer numériquement la formule fermée d'Heston. Nous ne l'avons pas implémentée.