



MODÈLES STOCHASTIQUES POUR LA FINANCE

CENTRALESUPÉLEC

OPTION MATHÉMATIQUES APPLIQUÉES

Projet

Auteurs:

Laurent Sekonian
Antoine Gruson

Encadrants:

Ioane Muni-Toke

January 21, 2021

Contents

Introduction	3
1 Question I - Modèle de Black & Scholes et Monte-Carlo	4
1.1 Modèle de Black & Scholes	4
1.2 Greeks	8
1.3 Méthode de Monte Carlo	11
1.4 Panier d'actifs	14
2 Question II - Option asiatique et réduction de variance	16
2.1 Monte Carlo pour Option Asiatique	17
2.2 Réduction de Variance	20
2.2.1 Variables antithétiques	20
2.2.2 Variables de contrôle	21
3 Question III - Options à barrières	23
3.1 Monte Carlo pour Option à barrières	23
3.2 Surveillance continue de la barrière	26
3.3 Réduction de Variance par Monte Carlo Conditionnel	27
4 EDP pour le modèle de Black & Sholes	29
4.1 Résolution de l'EDP de Black Scholes	29
4.2 Prix de l'EDP en fonction de S_0	30
4.3 Tracé des l'erreur Moyenne (L_1, L_2, L_∞)	33
4.4 Convergence des schémas	35
4.5 Tracé de l'erreur ponctuelle en fonction de t et S	38
5 Discrétisation d'EDS et modèle de Heston	40
5.1 Discrétisation d'EDS	40
5.2 Modèle de Heston	43
5.3 Volatilité implicite et Smile/Smirt de volatilité	45
5.4 Smile en fonction des paramètres λ, η, ρ	46
5.5 Densité de transition du processus CIR	48

Introduction

Note

Ce rapport sera en lien avec le notebook Python  compagnon. La totalité des questions représentant le fil directeur du rapport seront rappelées et traitées, avec des analyses supplémentaires étudiées dans certains cas. La plupart des méthodes seront implémentées en tant que classes, afin que le lecteur puisse manier les objets à sa guise plus simplement.

Pour les simulations par MonteCarlo et le modèle de Heston, nous avons fait le choix d'effectuer l'estimation directement dans les fonctions `__init__` de chaque classe, afin d'avoir une estimation par instanciation.

Motivation

Le modèle de Black&Scholes met en lien les prix de produits dérivés avec les variations des sous-jacents, représenté mathématiquement comme des processus stochastiques continus.

En 1973, Fischer Black et Myron Scholes publient leur travaux, approfondis d'un point de vue mathématiques par Robert C. Merton.

Ce modèle, bien que limité par de nombreux aspects, a révolutionné le monde de la finance pour l'évaluation d'options.

Plan

Nous allons premièrement étudier l'implémentation du modèle de Black&Scholes et présenter les principales grecques associées. Puis, sur l'implémentation de méthodes de Monte-Carlo pour le call et put vanille, où nous illustrerons les différentes propriétés de la méthode.

Deuxièmement, les options asiatiques seront étudiées ainsi que les méthodes de réductions de variances par variable antithétiques et de contrôle pour l'estimateur de Monte-Carlo.

En troisième partie, nous nous pencherons sur les options Up&Out et Down&In, en analysant les bienfaits de la réduction de variance par Monte-Carlo conditionnel.

En quatrième partie, nous détaillerons l'implémentation de l'approche par EDP (Equations aux dérivées partielles) du modèle de Black&Scholes ainsi que les comparaisons associées.

Enfin, nous étudierons le modèle de Heston, prenant en compte la volatilité stochastique, et permettant notamment d'obtenir des smiles de volatilité.

Partie 1

Question I - Modèle de Black & Scholes et Monte-Carlo

1.1 Modèle de Black & Scholes

Rappels

On détaillera ici la construction et l'obtention des solutions du modèle de Black&Scholes.

Soit $t \in [0, T]$ avec T un horizon de temps fixé (maturité), $(\Omega, \mathcal{F}, \mathbb{P})$ un espace de probabilité muni de la filtration $(\mathcal{F}_t)_{t \in [0, T]}$ et B_t un (\mathcal{F}_t) -mouvement brownien standard unidimensionnel sous \mathbb{P} , de loi normale $\mathcal{N}(0, t)$.

On considère un **actif risqué** de prix S , stochastique, de dynamique :

$$dS_t = \mu S_t dt + \sigma S_t dB_t \quad (1.1)$$

Avec μ représentant le drift (tendance), et $\sigma > 0$ la volatilité.

En passant au log et en appliquant Itô sur $Y_t = \log(S_t)$, nous obtenons la solution :

$$S_t = S_0 e^{\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma B_t}$$

Nous admettons que $(S_t)_{t \in [0, T]}$ est un processus continu, dont les composantes suivent une loi log-normale. Les rendements sont donc, sous nos hypothèses, stationnaires et indépendants.

Également, nous considérons un **actif sans risque** de prix D déterministe, croissant au taux sans risque $r > 0$ tel que :

$$dD_t = r D_t$$

Avec $D_0 = 1$, nous avons donc en résolvant l'équation :

$$D_t = e^{rt}$$

Généralement, le taux r peut être vu comme le taux d'une obligation zéro coupon.

Ce taux sans risque permet d'introduire le principe d'**actualisation (discount)**, représentant la "valeur-temps" de l'argent, tel que :

$$\tilde{S}_t = e^{-rt} S_t$$

Nous avons ainsi :

$$d\tilde{S}_t = (\mu - r) \tilde{S}_t dt + \sigma \tilde{S}_t dB_t \quad (1.2)$$

On rappelle qu'un **produit dérivé** sur S est un produit financier qui, pour une maturité T , paye un certain **payoff** Z à maturité à celui ayant acheté le produit à $t = 0$.

Ce payoff Z est une variable aléatoire dépendant de la trajectoire $(S_t)_{t \in [0, T]}$.

Introduisons deux produits dérivés, options vanilles :

Call : Option délivrant le droit (non l'obligation) à son détenteur d'acheter le sous-jacent S à un prix déterminé à l'avance, le strike K . Payoff : $(S_T - K)_+$

Put : Option délivrant le droit (non l'obligation) à son détenteur de vendre le sous-jacent S à un prix déterminé à l'avance, le strike K . Payoff : $(K - S_T)_+$

On parle d'option européenne lorsque l'exercice ne se fait qu'à maturité. Et d'options américaines lorsque l'exercice peut se faire à tout moment de la durée de vie de l'option.

Formulation

Ce modèle s'établit dans le cadre de la probabilité risque-neutre \mathbb{Q} , c'est à dire qu'en appliquant le théorème de Girsanov (avec $\theta_t = \frac{\mu-r}{\sigma}$), pour les probabilités équivalentes \mathbb{Q} et \mathbb{P} , nous avons

$$B_t^Q = B_t + \frac{\mu - r}{\sigma} t$$

comme étant un (\mathcal{F}_t) -mouvement brownien sous \mathbb{Q} , mesure martingale équivalente à \mathbb{P} .

Ainsi, nous avons :

$$d\tilde{S}_t = \sigma \tilde{S}_t dB_t^Q$$

Et donc :

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

Nous n'avons donc pas de terme μ , terme représentant l'évolution de l'actif, sous la probabilité risque-neutre \mathbb{Q} .

Revenons au call, sa valeur à t représente l'espérance actualisée du payoff sous \mathbb{Q} (le portefeuille autofinancé actualisé associé est une martingale sous \mathbb{Q} et le call est un produit réplifiable car dans ce modèle le marché est complet). C'est à dire :

$$C_t = \mathbb{E}^{\mathbb{Q}}[e^{-r(T-t)}(S_T - K)_+ | \mathcal{F}_t]$$

Or

$$S_T = S_t e^{\left(r - \frac{\sigma^2}{2}\right)(T-t) + \sigma(B_T^Q - B_t^Q)}$$

Avec $(B_T^Q - B_t^Q)$ indépendant de \mathcal{F}_t . Nous avons donc :

$$\begin{aligned} C_t &= \mathbb{E}^{\mathbb{Q}}[e^{-r(T-t)}(S_T - K)_+] \\ &= e^{-r(T-t)} \mathbb{E}^{\mathbb{Q}}[S_T \mathbb{1}_{S_T > K}] - K e^{-r(T-t)} \mathbb{E}^{\mathbb{Q}}[\mathbb{1}_{S_T > K}] \end{aligned}$$

Avec $\mathbb{E}^{\mathbb{Q}}[\mathbb{1}_{S_T > K}] = \mathbb{P}(S_T > K)$.

Après calcul, pour le premier terme, on a :

$$e^{-r(T-t)} \mathbb{E}^{\mathbb{Q}}[S_T \mathbb{1}_{S_T > K}] = S_t \Phi(d_1)$$

Avec $\Phi(\cdot)$ la fonction de répartition de la loi normale, et :

$$d_1 = \frac{\ln(\frac{S_t}{K}) + (r + \frac{\sigma^2}{2})(T-t)}{\sigma \sqrt{T-t}}$$

Pour le deuxième terme, nous obtenons que $\mathbb{P}(S_T > K) = \Phi(d_2)$ (probabilité d'exercer le call), avec :

$$d_2 = d_1 - \sigma \sqrt{T-t}$$

Nous obtenons donc pour le prix du call :

$$C_t = S_t \Phi(d_1) - K e^{-r(T-t)} \Phi(d_2) \quad (1.3)$$

Découlant de l'Absence d'Opportunité d'Arbitrage (AOA), nous obtenons la **parité call/put** :

$$C_t - P_t = S_t - K e^{-r(T-t)}$$

Nous pouvons donc définir le prix du put :

$$P_t = -S_t \Phi(-d_1) + K e^{-r(T-t)} \Phi(-d_2) \quad (1.4)$$

Limites

Ce modèle est fort intéressant et a clairement révolutionné les mathématiques financières, en effet il permet de disposer d'une formule afin de connaître le prix des calls et puts à tout instant, permettant une stratégie de couverture et les activités de gestion de portefeuilles d'options. Cependant, la cadre de ce modèle comporte bien des limites :

- Volatilité σ et taux d'emprunt r supposé constant, nous verrons en partie 5 que des modèles peuvent prendre en compte la volatilité stochastique,
- On suppose que S_t a des rendements log-normaux, ce qui n'est pas toujours vrai dans la réalité,
- Pas de frais de transaction et pas de fiscalité,
- La liquidité et supposée parfaite / divisibilité des actifs,
- Le marché est censé fonctionner en continu,
- La vente à découvert est autorisée, ce qui n'est pas toujours le cas.

Paramètres par défaut utilisés pour les simulations

💡 : Les classes *Call* et *Put* héritant de la classe mère *BS* ont été implémentées afin de déterminer le prix des calls et puts standards en fonction des paramètres. Le taux de dividendes continu a a été ajouté. Il est important de noter que **par défaut, les paramètres sont :**

- $S_t = 100$,
- $t = 0$, donc $S_t = S_0$,
- $K = 100$,
- $T = 90d$,
- $\sigma = 0.25$,
- $r = 0.05$,
- $a = 0$ (modèle sans dividendes continues).

Evolution du prix de calls et puts standards en fonction de S_0

Paramètres : $S_0 \in [0, 200]$

Voici l'évolution d'un call et d'un put standards en fonction de S_0 pour un acheteur d'options.

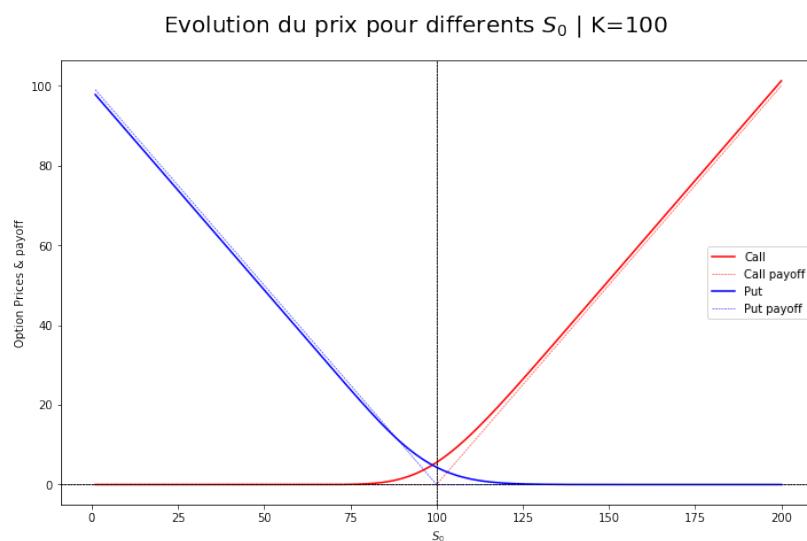


Figure 1.1: Evolution d'un call et d'un put standards en fonction de S_0

On peut remarquer que pour $S_0 = 100$ avec $K = 100$ (à la monnaie), le prix du put est inférieur au prix du call, et que lorsque le prix s'éloigne de S_0 , le prix du call est au dessus de son payoff, tandis que le prix du put est

en dessous de son payoff.

Cela peut se traduire par le fait que la tendance r est positive (l'actif à maturité a plus de chance d'être supérieur à S_0).

Nous pouvons également le voir de la manière suivante :

$$\mathbb{E}[S_t] = \mathbb{E}\left[S_0 e^{\left(r - \frac{\sigma^2}{2}\right)t + \sigma B_t}\right] = S_0 e^{\left(r - \frac{\sigma^2}{2}\right)t} \mathbb{E}[e^{\sigma B_t}] \stackrel{\text{Laplace}}{=} S_0 e^{\left(r - \frac{\sigma^2}{2}\right)t} e^{\left(\frac{\sigma^2}{2}\right)t} = S_0 e^{rt} > S_0$$

Inversement, si $r < 0$, alors le prix du put à la monnaie sera supérieur à celui du call car $\mathbb{E}[S_t] = S_0 e^{rt} < S_0$

Si nous annulons la tendance, $r = 0$, alors les prix seront identiques à la monnaie et les courbes totalement et symétriques superposées à leurs payoffs.

En effet :

$$\mathbb{E}[S_t] = S_0 e^{rt} \stackrel{r=0}{=} S_0$$

Evolution du prix de calls et puts standards pour différents T en fonction du strike K

Paramètres : $K \in [140, 240]$, $S_0 = 187$ et $T = \{30d, 95d, 340d\}$

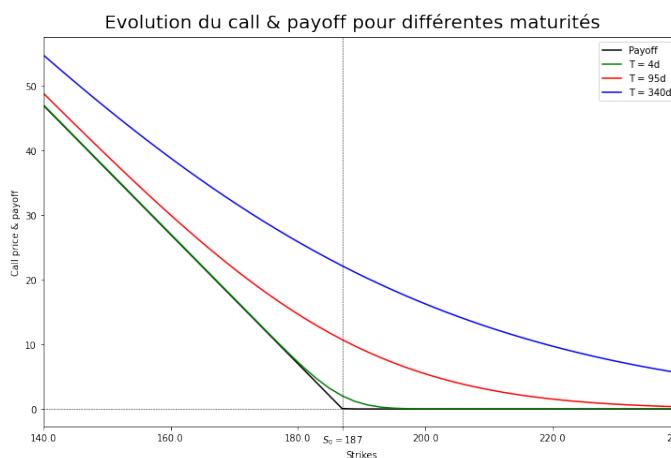


Figure 1.2: Evolution de prix de calls pour différentes maturités

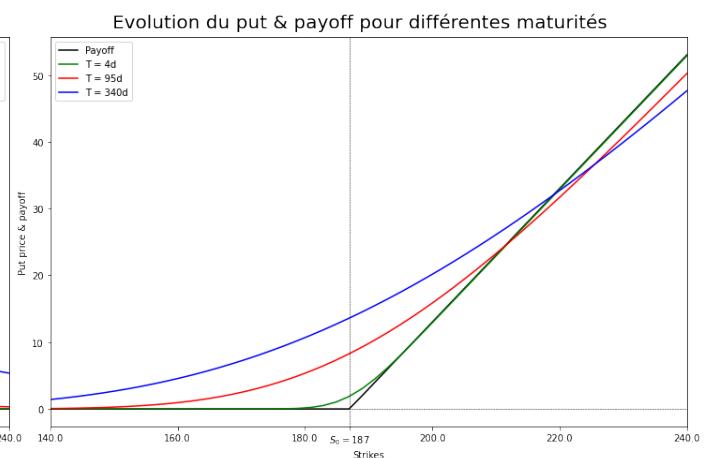


Figure 1.3: Evolution de prix de puts pour différentes maturités

Pour le call, on peut voir que plus la maturité est faible, plus le prix du call l'est aussi. En effet, une proche maturité est synonyme de bien moins d'incertitude sur le prix S_T , et donc le prix se rapproche naturellement du payoff.

Puis nous savons que la valeur de l'option décroît avec le temps ($\theta < 0$).

De plus, plus le strike K est proche de S en fin de vie de l'option, plus il y a incertitude quant à l'exercice, c'est pour cela que le prix de l'option se détache du payoff à la monnaie.

Pour le put, nous observons les mêmes conclusions sauf pour des strikes grands, on peut voir que les prix de put de maturités plus grandes passent en dessous du payoff, et on voit également une inversion d'ordre des prix du put pour des strikes plus grands.

En effet, comme dit précédemment, plus la maturité est grande et plus l'actif a de "chances de monter", et plus le sous-jacent est en hausse, moins le put est intéressant. Pour $K = 240$ par exemple, le sous-jacent a plus de chance d'effectuer une plus grande hausse en 340 jours qu'en 4 jours, le prix du put serait donc moins intéressant, d'où le "renversement".

Profit et Payoff acheteur/vendeur d'options

Paramètres : Par défaut.

Il peut être aussi intéressant de se pencher sur le payoff et le profit d'un acheteur (long) et vendeurs (short) d'options. Le profit est égal au payoff moins le prix de l'option payée en $t = 0$.

Dans le cas du call :

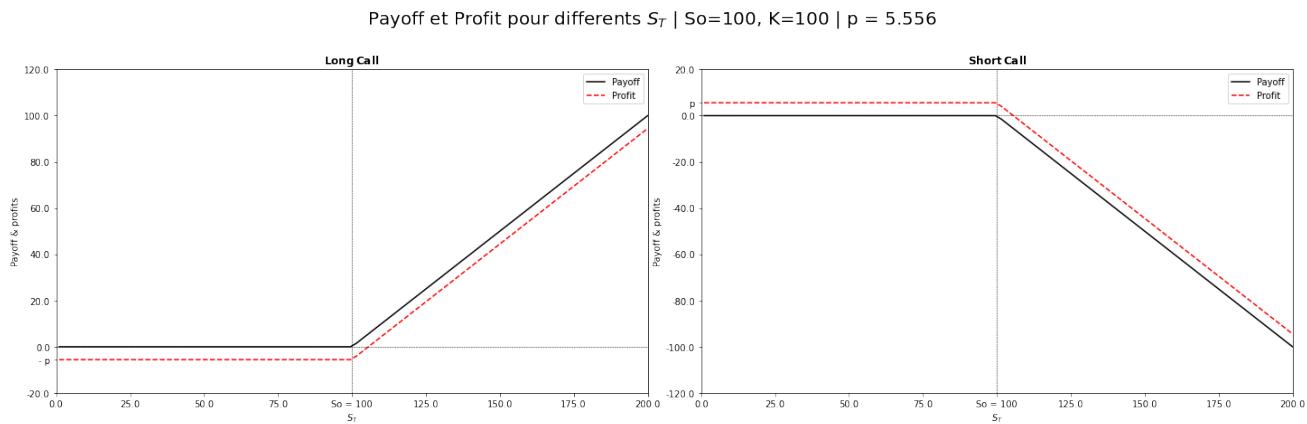


Figure 1.4: Evolution du profit et payoff pour achat (à gauche) et vente (à droite) d'un call et en fonction de S_T

On voit que dans le cas d'achat (vente) d'un call, le potentiel profit (perte) est illimité(e).

Dans le cas du put :

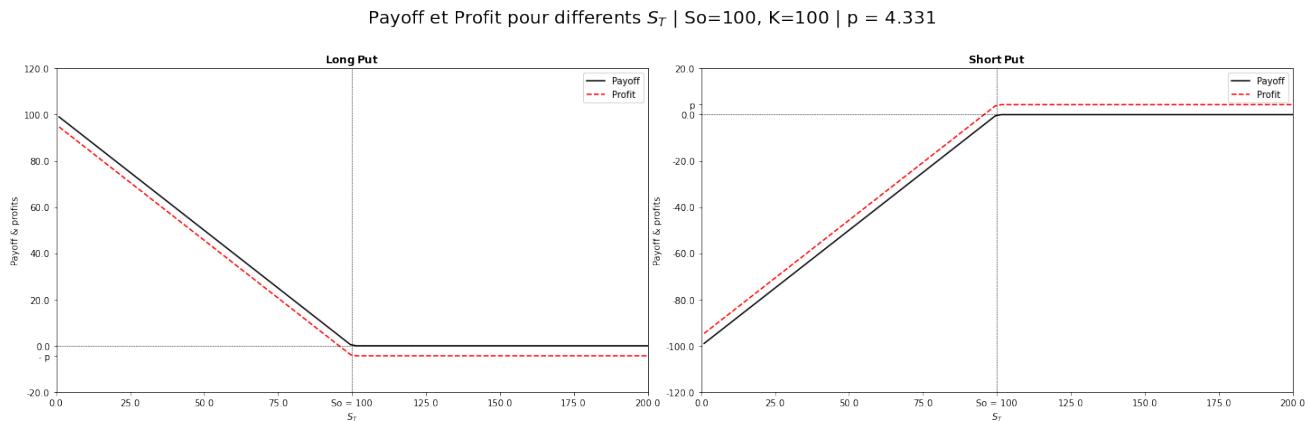


Figure 1.5: Evolution du profit et payoff pour achat (à gauche) et vente (à droite) d'un put et en fonction de S_T

Dans ce cas, le potentiel profit (perte) est limité(e) vu que le prix de l'actif ne peut pas être négatif.

1.2 Greeks

💡 : Le delta, gamma, theta, vega et rho sont implémentés dans les classes *Call* et *Put*.

Les *Greeks* représentent la sensibilité du prix de l'option par rapport aux paramètres. Elles sont notamment utiles pour la gestion de portefeuille d'options.

Nous pouvons introduire le "lemme de calcul des grecques" simplifiant l'obtention des formules :

$$S_t \Phi'(d_1) = K e^{-r(T-t)} \Phi'(d_2)$$

Delta Δ

Le delta mesure la sensibilité du prix de l'option par rapport à une variation du prix du sous-jacent, on peut le voir comme la vitesse d'augmentation du prix de l'option par rapport au sous-jacent. On a :

$$\Delta = \frac{\partial C}{\partial S}$$

Après calculs :

$$\Delta_{Call} = \Phi(d_1) > 0 \quad | \quad \Delta_{Put} = -\Phi(-d_1) < 0$$

En effet, le delta du call vaut d'autant plus que le prix du sous-jacent est élevé, il ne peut donc pas être négatif. De même, plus le cours du sous-jacent est bas et plus le prix du put est élevé, d'où son delta négatif. Nous avons donc toujours $\Delta_{Call} \in [0, 1]$, $\Delta_{Put} \in [-1, 0]$, et $\Delta_{Call} = 1 - \Delta_{Put}$.

A la monnaie ($S = K$), et en admettant que $r \ll \sigma$, nous avons :

$$\Delta_{atm} = \Phi\left(\frac{\sigma}{2}\sqrt{T-t}\right) \approx \Phi(0) = \frac{1}{2}$$

Donc, plus on s'approche de la maturité en étant à la monnaie, plus le delta est proche de 0.5.

Un portefeuille de calls achetés peut être répliqué en achetant Δ_{Call} actions. En effet, lorsqu'on achète un call, cela revient à "espérer" une hausse du sous-jacent.

Un portefeuille de puts achetés peut être répliqué en vendant Δ_{Put} actions. Par même raisonnement, acheter un put revient à "espérer" une baisse du sous-jacent.

Prenons l'exemple d'un market maker devant vendre un call à un client. Une chose très importante en market-making est qu'en réalisant cette transaction, le trader ne doit pas prendre de risques. Il se doit donc de couvrir sa position, se hedger.

En vendant le call, il s'expose à une possible montée de l'action (si $S_T > K$ à maturité alors il devra délivrer le payoff au client). Afin de se hedger, il va donc acheter Δ_{Call} actions du sous-jacent pour pouvoir lui livrer le payoff. Son portefeuille est donc delta-hedge.

Ci dessous nous traçons le delta pour un call en fonction de différentes maturités.

Paramètres : $S_0 \in [0, 200]$, $T = \{5d, 90d, 350d\}$

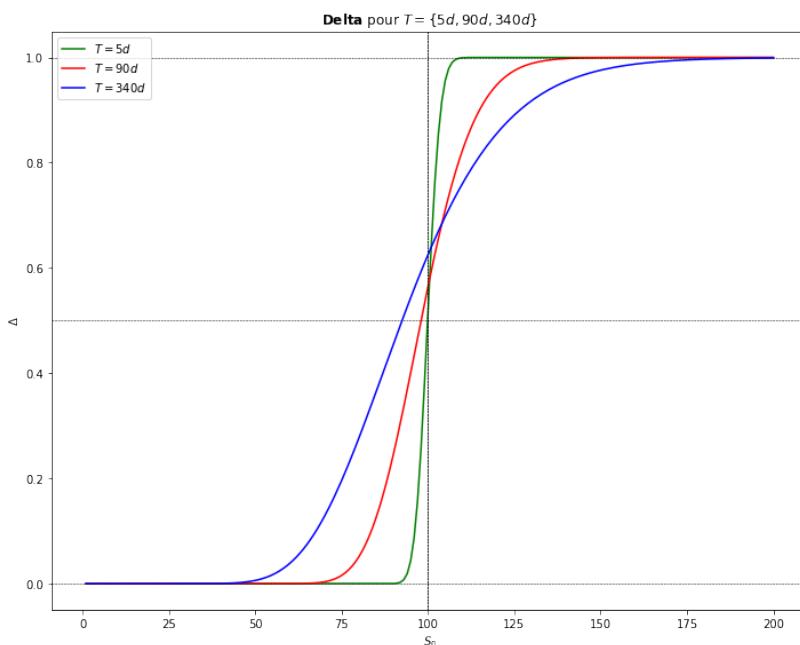


Figure 1.6: Delta pour différentes maturités en fonction de S_0

Nous voyons bien une allure de fonction de répartition. On peut également vérifier que plus la maturité est proche, plus le delta à la monnaie se rapproche de 0.5.

Gamma Γ

Le gamma mesure la convexité du prix de l'option par rapport au prix du sous-jacent, on peut le voir comme l'accélération du prix de l'option par rapport à S . C'est à dire si l'option évolue plus ou moins vite que le sous-jacent. On a :

$$\Gamma = \frac{\partial^2 C}{\partial S^2}$$

Après calculs :

$$\Gamma_{Call} = \Gamma_{Put} = \frac{\Phi'(d_1)}{S_t \sigma \sqrt{T-t}} > 0$$

Ci dessous nous traçons le gamma pour un call en fonction de différentes maturités.

Paramètres : $S_0 \in [0, 200]$, $T = \{5d, 90d, 340d\}$

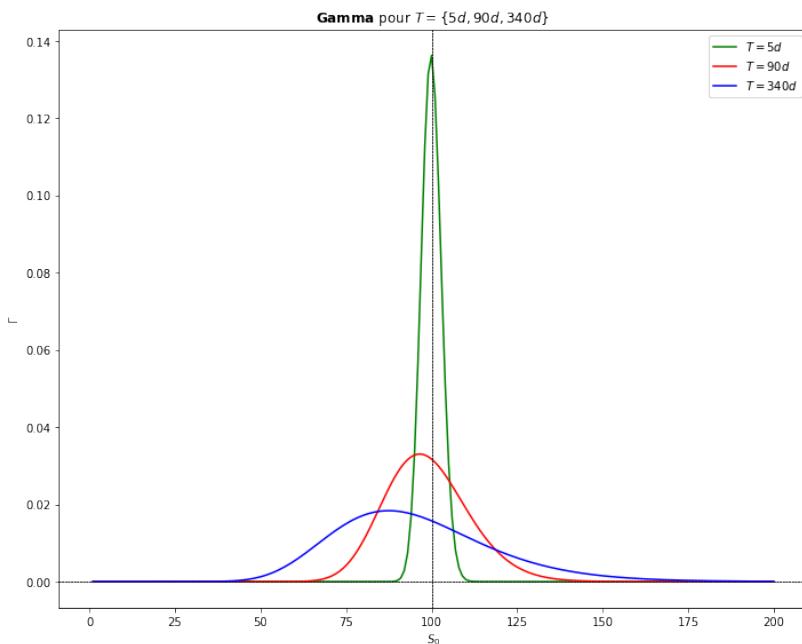


Figure 1.7: Gamma pour différentes maturités en fonction de S_0

On remarque donc que plus on se rapproche de la maturité, plus le gamma augmente. On peut voir analytiquement que gamma tend vers $+\infty$ quand T tend vers 0 (de même avec σ).

Vega ν , Theta θ et Rho ρ

Le vega ν représente la sensibilité de l'option par rapport à la volatilité (implicite). On a :

$$\nu_{call} = \nu_{put} = \frac{\partial C}{\partial \sigma}$$

Le vega est d'autant plus élevé que la maturité est grande. Dans le cas d'un acheteur d'options on a $\nu_{call} = \nu_{put} \geq 0$, l'inverse pour un vendeur.

Le theta θ représente le "coût" du temps du portefeuille. Il sera négatif pour un acheteur d'options (options longue) et négatif pour un vendeur. On a :

$$\theta = -\frac{\partial C}{\partial T}$$

Quant au rho ρ , il représente la sensibilité de l'option par rapport au taux sans risque. On a :

$$\rho = \frac{\partial C}{\partial r}$$

Voici le tracé des trois pour un call en fonction de $S_0 \in [0, 200]$:

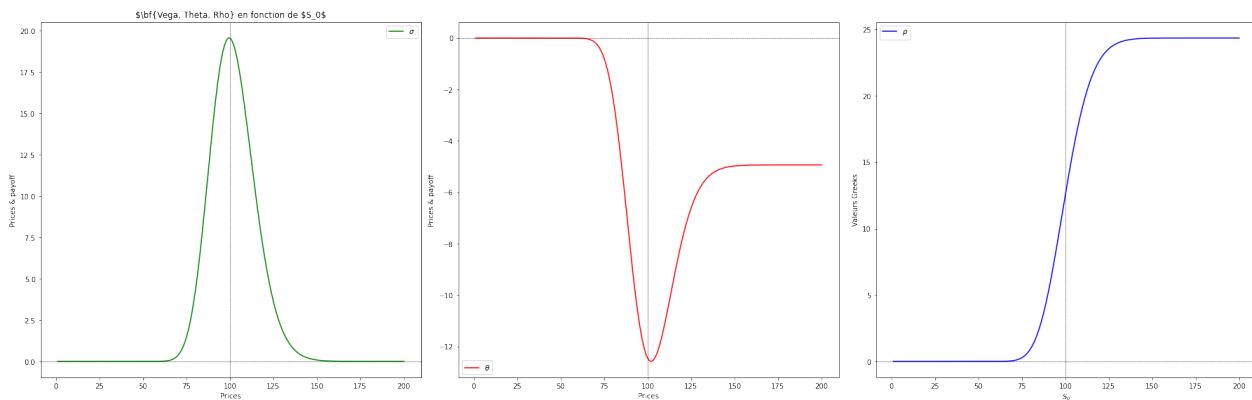


Figure 1.8: Gamma pour différentes maturités en fonction de S_0

1.3 Méthode de Monte Carlo

💡 : Nous avons également réalisé les estimateurs de MonteCarlo sous formes de classes afin de pouvoir aisément manipuler les objets par la suite.

Une classe mère *MonteCarloBS* est implémentée. Chaque classe décrite dans la suite héritera de celle-ci. Elle contient les méthodes principales et communes aux classes filles, telles que les fonctions *display_init* et *display_results* qui permettent d'afficher directement les outputs d'initialisation et de résultats (écart-type, estimation, intervalles de confiance) lors des instantiations. Un argument booléen *display_text* par défaut à False permet d'afficher ou non l'output. Un exemple d'output est affiché en début de partie 2.

Cette classe mère dispose également des fonctions de calcul de variance à la volée, via la fonction *var_est*, et du calcul des intervalles de confiance via la fonction *IC*.

Pour cette partie, une classe fille *MonteCarloBS_European* a été implémentée. Nous y traitons le cas du call et du put européens

Nous n'avons plus de paramètres t ici (S_t sera donc forcément S_0). Et nous rajoutons le paramètre N représentant le nombre d'estimations. Nous rappelons les paramètres ci-dessous :

- $N = 1000$,
- $S_0 = 100$,
- $K = 100$,
- $T = 90d$,
- $\sigma = 0.25$,
- $r = 0.05$,
- call = True et put = False.

Principe

La méthode de Monte Carlo est basée sur N simulations i.i.d d'une variable aléatoire d'une distribution que l'on connaît.

En admettant que nous souhaitons estimer l'espérance d'une fonction de cette variable aléatoire, on peut appliquer la loi forte des grands nombres, afin d'avoir :

$$C_N = \frac{1}{N} \sum_{i=1}^N h(X_i) \xrightarrow[N \rightarrow +\infty]{p.s.} \mathbb{E}[h(X)]$$

Ainsi, notre estimateur est convergent, sans biais, et asymptotiquement normal (théorème central limite).

Dans notre cas, nous souhaitons estimer l'espérance sous mesure risque neutre \mathbb{Q} du payoff actualisé. Nous avons donc $h(S_T) = (S_T - K)_+$ et :

$$C_0 = \mathbb{E}^{\mathbb{Q}}[e^{-rT}(S_T - K)_+]$$

Avec $S_T = S_0 e^{(r - \frac{\sigma^2}{2})T + \sigma B_T^Q}$. Et on sait que le mouvement brownien B_T^Q suit une loi normale $\mathcal{N}(0, T)$.

Nous avons donc :

$$h(X_i) = e^{-rT} \left(S_0 e^{\left(r - \frac{\sigma^2}{2}\right) T + \sigma X_i \sqrt{T}} - K \right)_+$$

Nous pouvons ainsi simuler N réalisation d'une loi normale centrée réduite pour les X_i , calculer nos fonctions $h(X_i)$, puis en faire la moyenne afin d'avoir une estimation sans biais de l'espérance.

Nous sommes donc en mesure d'estimer le prix d'un call standard avec la méthode de Monte-Carlo (pour le cas du put également, en notant le bon payoff en conséquence).

D'un point de vue général, nous sommes capables d'estimer tout produit dérivé européen dans le cadre de notre modèle de Black&Scholes via la méthode de Monte-Carlo.

Comparaison avec formule fermée

Pour une première comparaison, en évaluant le prix d'un call avec la formule de Black&Scholes, nous obtenons $C_0^{BS} = 5.556$. Avec Monte-Carlo, pour les mêmes paramètres et un nombre de simulations $N = 1\ 000\ 000$, nous obtenons $C_0^{MC} = 5.559$.

Convergence vers le prix exact

Afin de pouvoir calculer les intervalles de confiance de chaque estimateurs, nous avons défini une fonction permettant le **calcul à la volée de $\sigma_{h(X)}^2$** via l'algorithme stable (cohérence vérifiée avec la fonction `np.var`). Nous pouvons donc définir, pour chaque estimateur, un intervalle de confiance de niveau $1 - \alpha$:

$$\left[C_n - q_{1-\frac{\alpha}{2}} \sqrt{\frac{\sigma^2}{n}} ; C_n + q_{1-\frac{\alpha}{2}} \sqrt{\frac{\sigma^2}{n}} \right]$$

Avec $q_{1-\frac{\alpha}{2}}$ quantile de la loi normale centrée réduite.

En réalisant 100 simulations pour $N \in [100, 1\ 000\ 000]$, et en y calculant les intervalles de confiance pour chaque C_n , nous pouvons vérifier la convergence de notre estimateur vers le prix trouvé avec la formule de Black&Scholes ($C_0^{BS} = 5.556$):

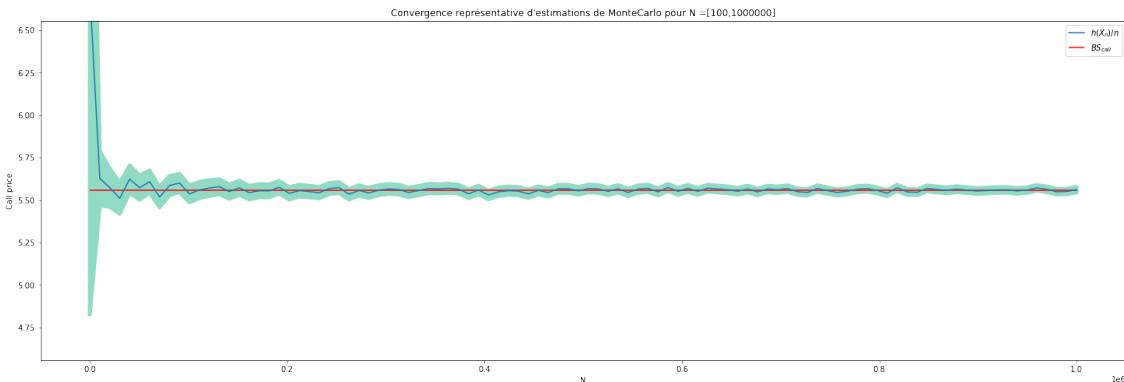


Figure 1.9: Convergence d'estimations de Monte-Carlo pour $N \in [100, 1000\ 000]$

Une fonction `plot_convergence` interne à la classe `MonteCarloBS_European` permet également d'avoir la convergence pour un estimateur en prenant la somme cumulée des $h(X_i)$. Voici un exemple avec $N = 1\ 000\ 000$. L'intervalle de confiance tracé représente l'intervalle de confiance général de l'estimateur.

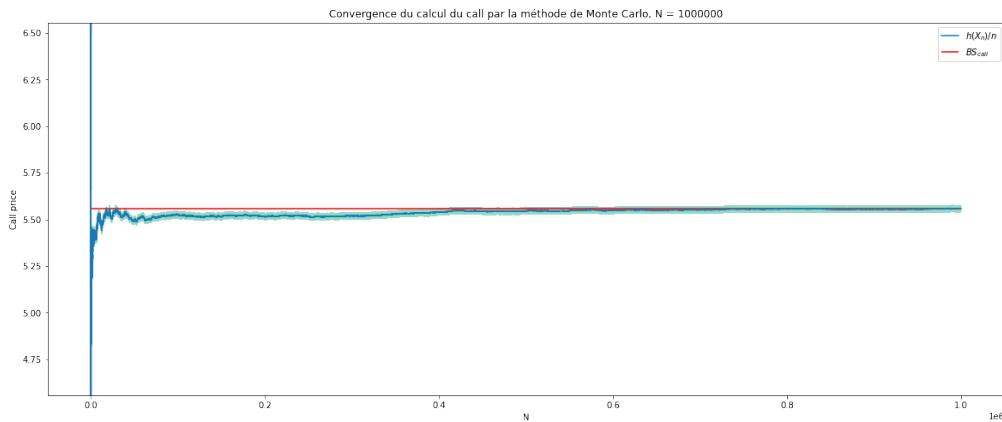


Figure 1.10: Convergence d'un estimateur de Monte-Carlo pour $N \in [100, 100\,000]$ avec somme cummulée

Normalité asymptotique

Notre estimateur est asymptotiquement normal, c'est à dire :

$$\sqrt{n} \frac{C_n - C}{\sigma_{h(X)}} \xrightarrow{n \rightarrow +\infty} \mathcal{N}(0, 1)$$

Afin de le démontrer, nous allons afficher la densité de N estimateurs pour N allant de 10 jusqu'à différents maximum : 100, 1000, 10 000. Nous affichons la moyenne μ et l'écart-type σ obtenu.

Voici ce que l'on obtient :

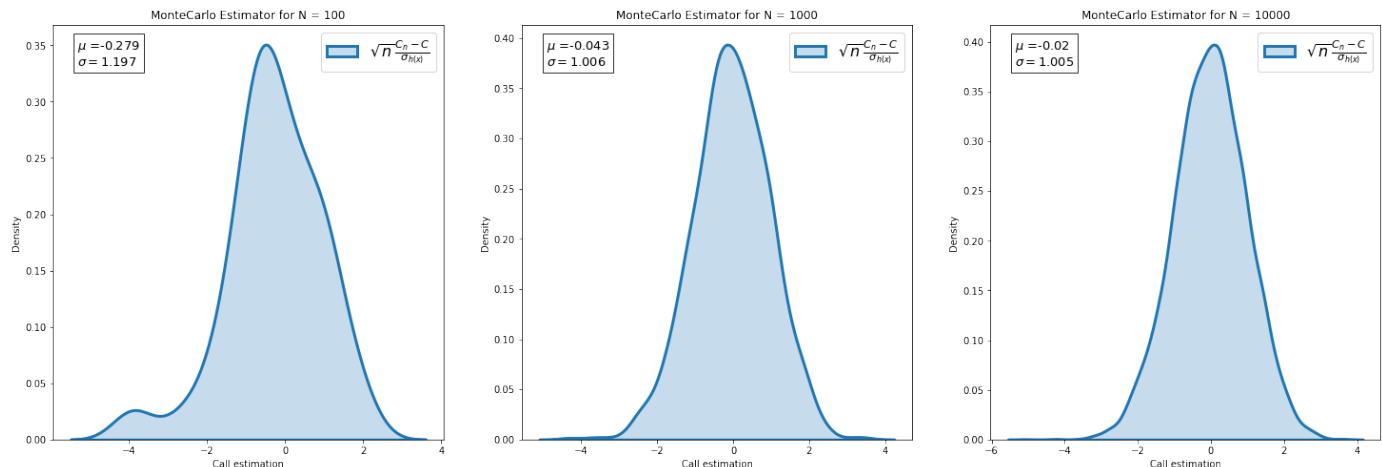


Figure 1.11: Densité de N estimateurs avec N allant de 10 jusqu'à différents maximum : 100, 1000, 10 000

Nous vérifions donc la convergence asymptotique de notre estimateur.

Vitesse de convergence

Théoriquement, la vitesse de convergence de notre estimateur est de \sqrt{n} . Afin de le vérifier, nous allons tracer un graphe log-log et vérifier que la pente est de $-\frac{1}{2}$, pour $N \in [100, 1000]_{step=1}$. Pour être plus précis, nous traçons en abscisse les N en échelle log et en ordonnée la valeur absolue de l'erreur $|C_n - C|$ en échelle log également.

💡 : nous calculons la pente avec la fonction `np.polyfit` pour nos deux log-sets de données.

Voici ce qu'on obtient :

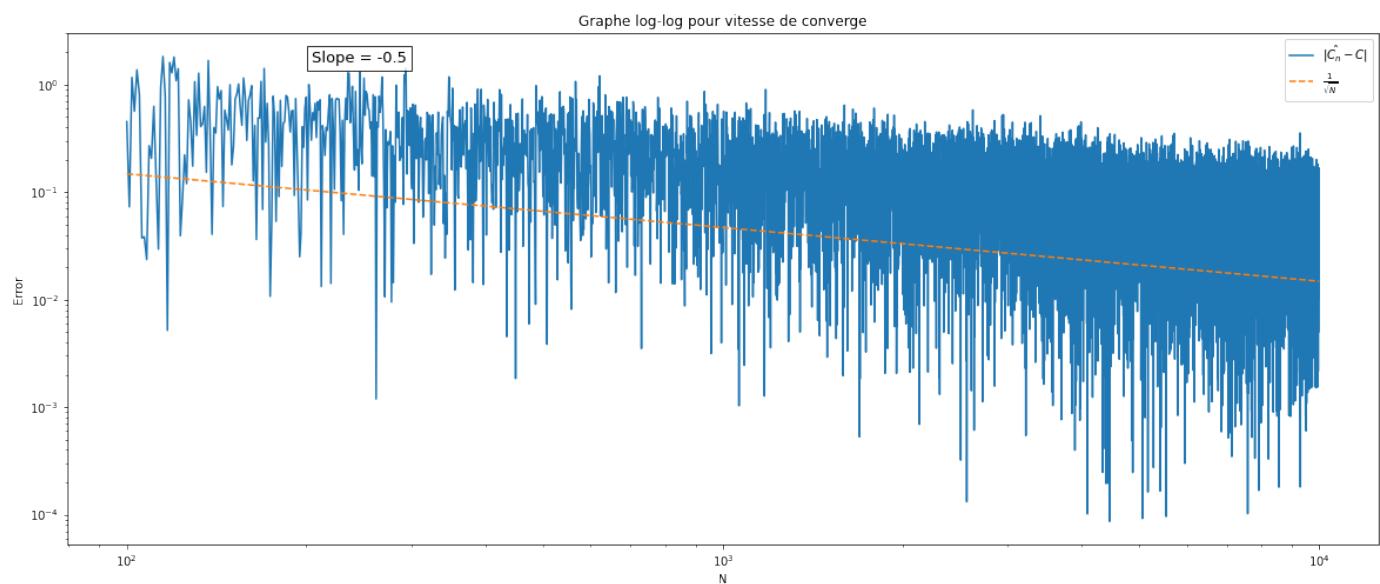


Figure 1.12: Graphique log-log de $|C_n - C|$ en fonction de $N \in [100, 1000]_{step=1}$

Nous vérifions donc la vitesse de convergence de l'estimateur.

1.4 Panier d'actifs

💡 : Nous avons rajouté une classe *MonteCarloBS_Basket* permettant de simuler un call ou put européen sur un panier de plusieurs actifs.

Voici les paramètres par défaut, pour deux actifs :

- $N = 1000$,
- $\text{eval} = \text{"max"}$ ("min", "max" ou "moyenne" disponibles),
- $S_0 = [100, 100]$, doit être obligatoirement en liste,
- $K = 100$,
- $\sigma = [0.25, 0.25]$, doit être obligatoirement en liste,
- $r = 0.05$,
- $T = 1$,
- $\rho = -0.5$.
- $\text{call} = \text{True}$ et $\text{put} = \text{False}$.

Afin de simuler les browniens corrélés, nous utilisons la méthode de décomposition de cholesky : pour un mouvement browien d-dimensionnel de composantes indépendantes et une matrice de corrélation ρ , nous pouvons générer W composé de mouvements browniens corrélés via

$$\rho = LL^T \longrightarrow W = LB$$

Si le paramètre ρ est donné en scalaire, alors le nombre d'asset doit être de deux, et la matrice de corrélation est initiée dans la classe. Si le nombre d'asset est supérieur à deux, l'argument ρ doit être directement la matrice de corrélation.

Exemple avec 4 actifs différents

Nous allons présenter un exemple avec 4 actifs différent :

Paramètres : Nous avons pour les 4 actifs $S_0 = 100$, $\sigma = 0.25$, et notre matrice de corrélation est :

$$\rho = \begin{pmatrix} 1 & 0.821 & -0.285 & -0.06 \\ 0.821 & 1 & -0.344 & -0.017 \\ -0.285 & -0.344 & 1 & 0.589 \\ -0.06 & -0.017 & 0.589 & 1 \end{pmatrix}$$

Nous obtenons les résultats suivants (on a $C_0^{BS} = 12.34$ et $P_0^{BS} = 7.46$ pour nos paramètres par défaut) :

	Call	Put
Max	29,02	0,64
Min	1,17	17,39
Moy	8,30	3,61

Table 1.1: Prix pour options call et put dans le cas d'un panier de 4 actifs

Pour le call, nous pouvons par exemple vérifier que le prix pour le mininimum des 4 actifs est bien plus bas que le prix pour le maximum, et l'inverse pour le put.

En remarque, nous pouvons dire qu'une bonne amélioration serait d'incorporer cette classe dans les classes parentes et de traiter directement le fait d'avoir un seul actif ou plusieurs. Ici nous avons montré séparément l'évaluation d'un panier et les autres méthodes pour un seul actif. De plus, l'évaluation de panier d'actifs n'est lié à aucune méthode de réduction de variance dans ce TP.

Partie 2

Question II - Option asiatique et réduction de variance

Note : Nous ne traitons que le cas du call pour l'option asiatique.

💡 Pour les Parties 2 et 3, une classe `MonteCarloPathDependant` dérivant de la parente `MonteCarloBS` est implémentée, afin d'y ajouter entre autre la simulation de trajectoires sous forme de dataframe `pandas`, pour donner une touche de "réalisme".

Un paramètre `fix_seed` booléen optionnel permet de simuler le même lot de trajectoires, utile pour la variation des paramètres. Il est mis à `False` par défaut.

Un paramètre `all_days` booléen optionnel permet de simuler tous les jours entre "aujourd'hui" et la maturité, indépendamment des dates d'observations (utile pour garder les mêmes trajectoires quand on estime sur plusieurs dates). Nous avons choisi de le mettre à `False` par défaut afin de gagner en temps de calcul : simuler seulement les trajectoires correspondant aux dates d'observations.

Les dates et la maturité T peuvent être fournies en fraction d'année numérique (ex : 0.5 pour 6 mois), ou également de type string (ex : "25/12/2021 12:45:33"), convertie en Timestamp pandas.

Les heures, minutes et secondes peuvent être ajoutées aux dates ou maturité, la différence entre deux dates discrètes en sera impactée (toujours calculée en tant que la différence de seconde entre les dates divisée par le nombre total de secondes d'une année).

Les dates doivent être entrées en liste ou `np.array`, ne peuvent ni être supérieures à la maturité, ni inférieures à la date d'aujourd'hui, seront triées si données dans le "désordre", et les doublons seront supprimés. Si une date est de type `string` elle doit être de format non-américain (jours en premier).

Cependant, le choix de `pandas` était à payer en temps de calcul. Nous avons donc choisi de réaliser tous les calculs en array `numpy` au lieu `pandas.Series`, en créant notre dataframe en fin d'initialisation, pour un gain en temps de calcul. De plus, un argument `kind_pandas` (mis à `True` par défaut) est disponible afin que la classe n'utilise que `numpy` et perde la caractérence de dates `pandas` (utile lorsque nous avons beaucoup de date d'observations comme dans le cas de la convergence vers le prix en barrière continue).

Si `kind_pandas` est mis à `False`, la classe vérifiera que les dates et la maturité soient données en fraction d'année.

Exemple : Voici un exemple d'instanciation pour prix de call asiatique et de l'output correspondant.

```
MonteCarloBS_Asian(dates=[1,0.2546, "05/06/2021 10:30:30",0.2], T = "18/01/2022", nb_sim_path=1000000, display_text=True);
----- MonteCarloBS_Asian(Call) -----
Nombre de simulations de trajectoires : 1000000. Aucune réduction de variance.
Pour les dates d'observations : ['01/04/2021', '20/04/2021', '05/06/2021', '18/01/2022']. Maturité: 18/01/2022
Ecart-type h(X): 9.306
Estimation : 6.464194821685318
Intervalle de confiance (95%) : [6.446,6.482]
```

Figure 2.1: Exemple d'instanciation avec la classe `MonteCarloBS_Asian` et output correspondant

En remarque, nous pouvons aussi dire que le plus judicieux aurait été de prendre 252 comme dénominateur pour la fraction d'année au lieu de 365, et de ne considérer que les dates de jours ouvrés. Puis, également de prendre en compte les horaires de marché, afin de mieux coller à la réalité. Cela peut être une bonne piste d'amélioration. Ici, nous nous intéressons surtout aux résultats numériques.

2.1 Monte Carlo pour Option Asiatique

💡 : Pour cette partie, une classe `MonteCarloBS_Asian` est implémentée. Le paramètre par défaut de maturité T change, nous le posons à $T = 1$ (un an). Les paramètres par défaut sont donc :

- dates (liste de dates discrètes à fournir obligatoire)
- `nb_sim_path` = 1000,
- $S_0 = 100$,
- $K = 100$,
- $T = 1$,
- $\sigma = 0.25$,
- $r = 0.05$.

Option Asiatique

Une option d'achat asiatique à moyenne arithmétique discrète est un produit dérivé payant à la maturité T le payoff :

$$\left(\frac{1}{N} \sum_{i=1}^N S_{t_i} - K \right)_+$$

Avec (t_1, \dots, t_N) représentant N dates déterministes dans l'intervalle $[0, T]$.

Nous pouvons ainsi en déduire l'évaluation par mesure martingale suivante :

$$C_0^A = \mathbb{E}^Q \left[e^{-rT} \left(\frac{1}{N} \sum_{i=1}^N S_{t_i} - K \right)_+ \right]$$

Nous pouvons ainsi obtenir le pricing d'une option asiatique par Monte Carlo.

Pour cela, nous allons donc simuler un certain nombre de trajectoire afin d'en déduire la moyenne et ainsi l'estimateur de Monte Carlo par la loi forte des grands nombres.

Dans le cadre du modèle de Black&Scholes, nous simulons les trajectoires de la manière suivante (mouvement brownien géométrique) :

$$S_{t_i} = S_{t_{i-1}} e^{\left(r - \frac{\sigma^2}{2}\right)(t_i - t_{i-1}) + \sigma(B_{t_i}^Q - B_{t_{i-1}}^Q)}$$

Avec $B_{t_i}^Q - B_{t_{i-1}}^Q \sim \mathcal{N}(0, t_i - t_{i-1}) \sim \sqrt{t_i - t_{i-1}} \mathcal{N}(0, 1)$.

Comparaison call asiatique (une date discrète à T) et européen pour $T = 90d$

Nous avons comparé un pricing de call asiatique pour une seule date discrète (même que T) pour T valant 90 jours afin d'avoir une correspondance avec le pricing du call européen.

Notre formule fermée de BS nous donne $C_0^{BS} = 5.5559$. Et nous obtenons $C_0^A = 5.5524$.

Simulation de 1000 trajectoires depuis $S_0 = 100$ jusqu'à T

Paramètres : Par défaut sauf $T = "19/04/2021"$.

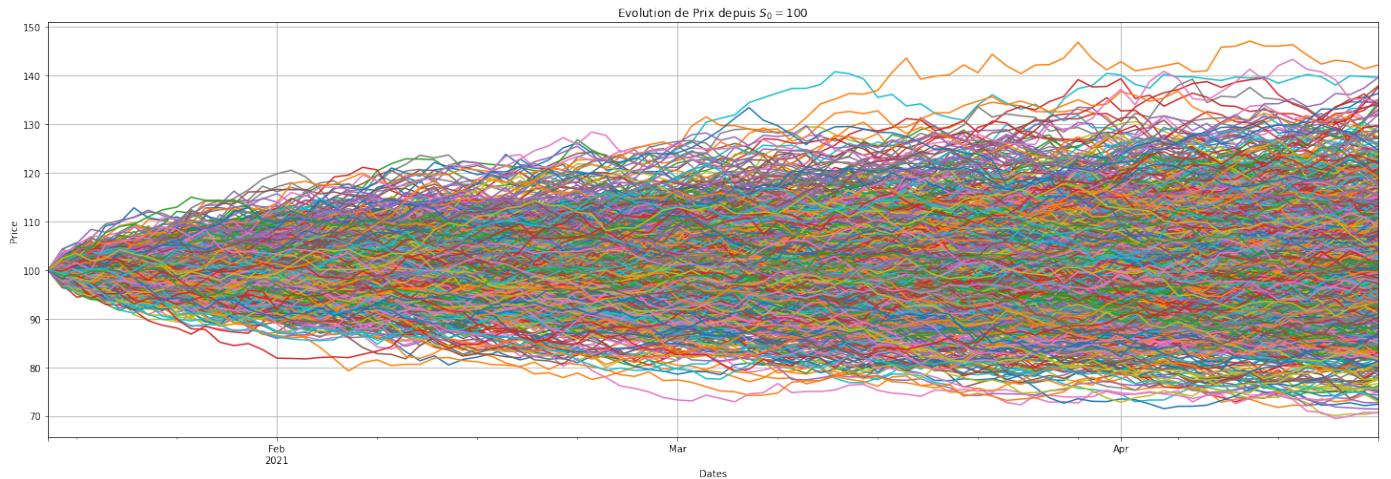


Figure 2.2: Simulation de 1000 trajectoires. $T = 19/04/2021$.

Evolution du prix pour $S_0 \in [0, 200]$ avec call européen en comparaison

Paramètres : Par défaut pour l'européen. Pour l'asiatique : $T = 90/365$, 10000 simulations de trajectoires avec `fix_seed=True`, et 5 dates d'observations partant "d'aujourd'hui" avec une fréquence de 10 business days (`freq="10B"`) en python).

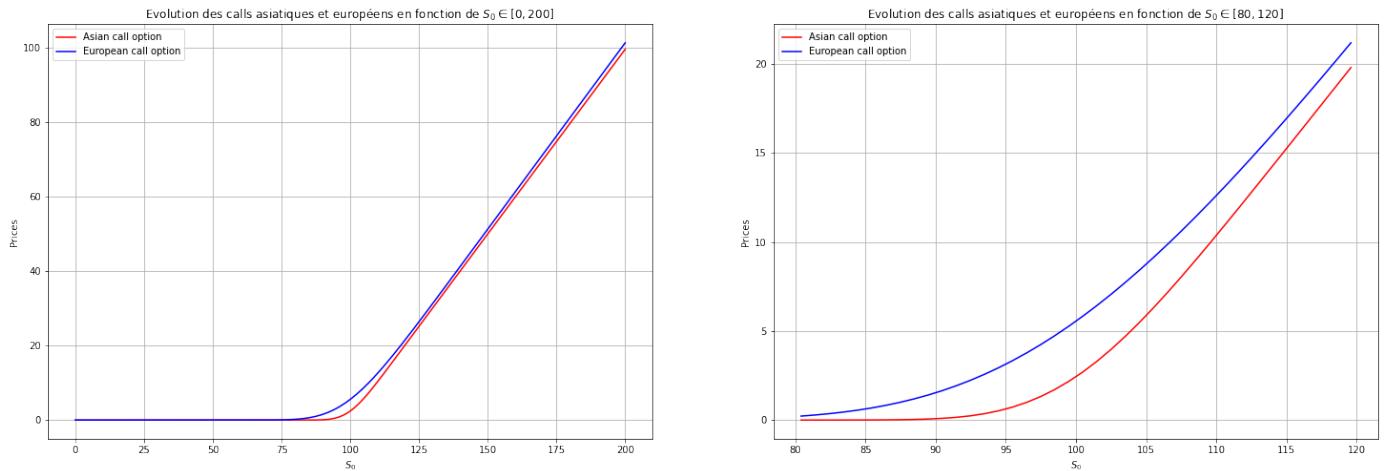


Figure 2.3: A gauche : Evolution des calls vanille et asiatique en fonction de $S_0 \in [0, 200]$.
A droite : En fonction de $S_0 \in [80, 120]$

Nous voyons que le prix du call asiatique est inférieur au prix d'un call européen, d'autant plus aux alentours du strike ($K = 100$ par défaut.). Cependant, nous remarquons également que plus S_0 augmente, plus le prix du call asiatique se rapproche du prix du call européen. Cela provient du fait que plus S_0 est grand comparé au strike, plus il y a de chances que le sous jacent S tisse une trajectoire supérieure au strike pour les dates discrètes données. Cependant, le prix est en théorie censé être toujours inférieur au prix du call, du fait de la moyenne des dates discrètes inférieures à T (même si on a une seule date discrète D inférieure à T , nous avons plus de chance d'avoir $S_D < S_T$ du au taux sans risque $r > 0$). Puis plus nous augmentons le nombre de dates d'observations, moins la moyenne sera "volatile", ce qui implique un prix inférieur à celui du call.

Donc, plus les dates discrètes D données sont inférieures à T , plus le prix de l'option asiatique baissera. Nous pouvons le vérifier ci-dessous.

Evolution du prix en fonction du nombres de dates d'observations : $Nb_{dates} \in [1, 100]$

En partant de T (dates d'observations décroissantes)

Paramètres : 10000 simulations de trajectoires avec `fix_seed=True` et `all_days=True` pour chaque simulation, afin d'avoir exactement chaque estimation basée sur les mêmes trajectoires.

La première date d'observation est la maturité ($T = 1$ par défaut). A chaque itération, nous rajoutons des dates antérieures de fréquence 3 jours, avec l'argument `freq` de la fonction `date_range` de `pandas`.

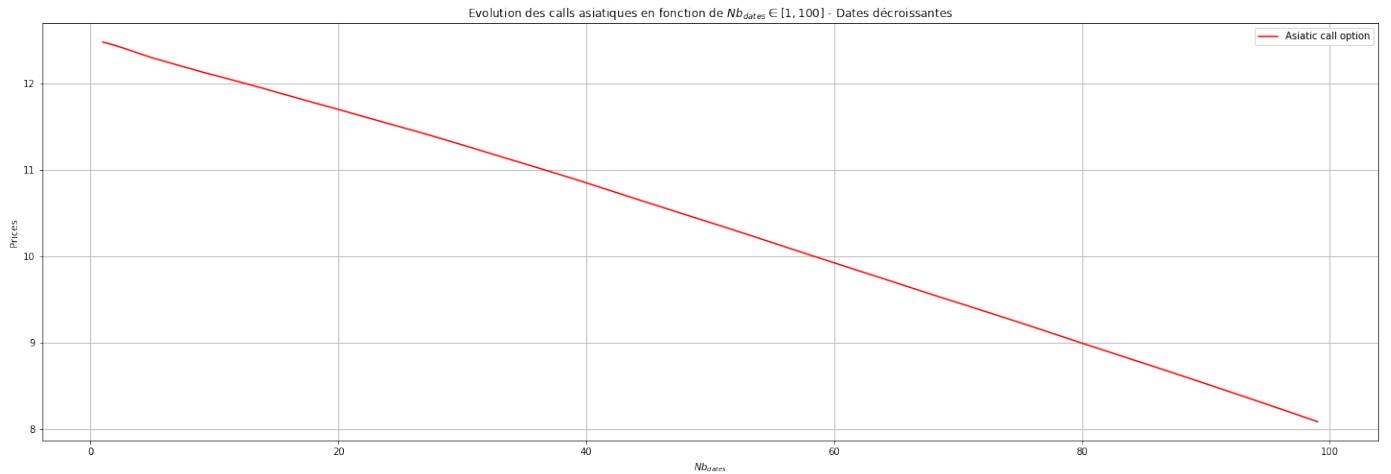


Figure 2.4: Evolution du prix de l'option asiatique pour $Nb_{dates} \in [1, 100]$ et 10000 simulations de trajectoires Dates décroissantes

On remarque que plus l'on soumet de dates d'observations, plus le prix de l'option en est diminué. Nous vérifions bien que plus on rajoute de dates éloignées de la maturité, plus le prix baisse.

On remarque également qu'avec seulement la date de maturité en date d'observation, nous retrouvons bien le prix du call vanille $C^{BS} = 12.34$.

En partant de $t_0 + 1$ (dates d'observations croissantes)

Paramètres : Identique sauf que la première date est "demain" (S_0 ne peut pas être pris en compte dans le payoff), et que nous ajoutons des dates ultérieures de même fréquence que précédemment.

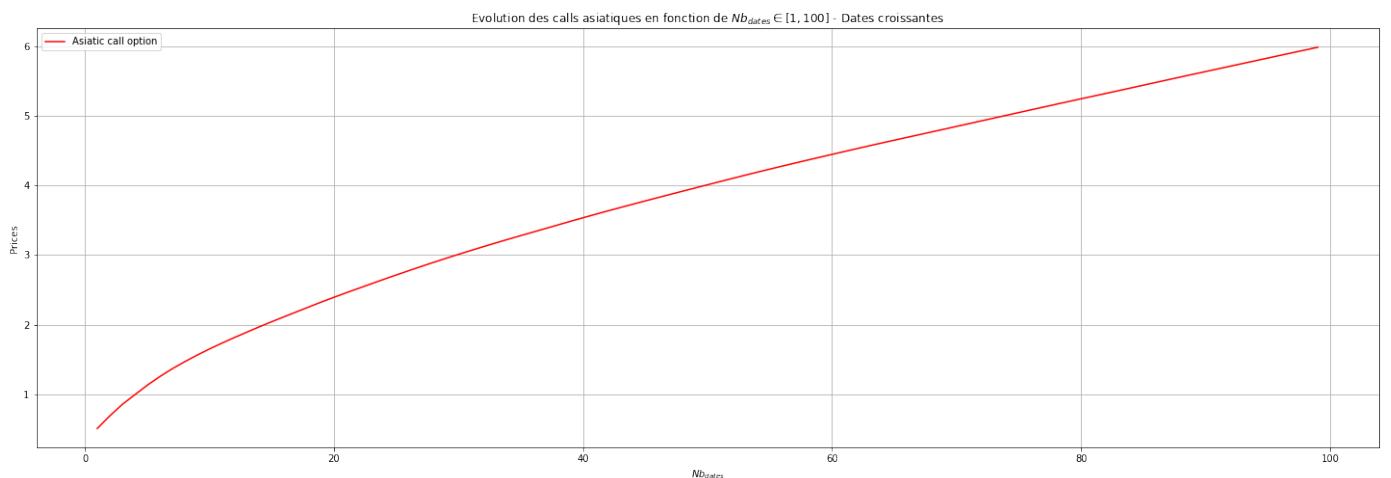


Figure 2.5: Evolution du prix de l'option asiatique pour $Nb_{dates} \in [1, 100]$ et 10000 simulations de trajectoires Dates croissantes

Inversement, nous voyons qu'en rajoutant des dates croissantes à chaque itération, le prix augmente. Cela est cohérent avec le fait que le prix est plus susceptible d'augmenter.

Convergence du prix en fonction du nombres de trajectoires simulées

Paramètres : Nous fournissons seulement la maturité ($T = 1$) en date discrète afin de comparer avec le call européen. Avec 500 nombre de trajectoires simulées allant de 2 à 10000, et intervalles de confiance.

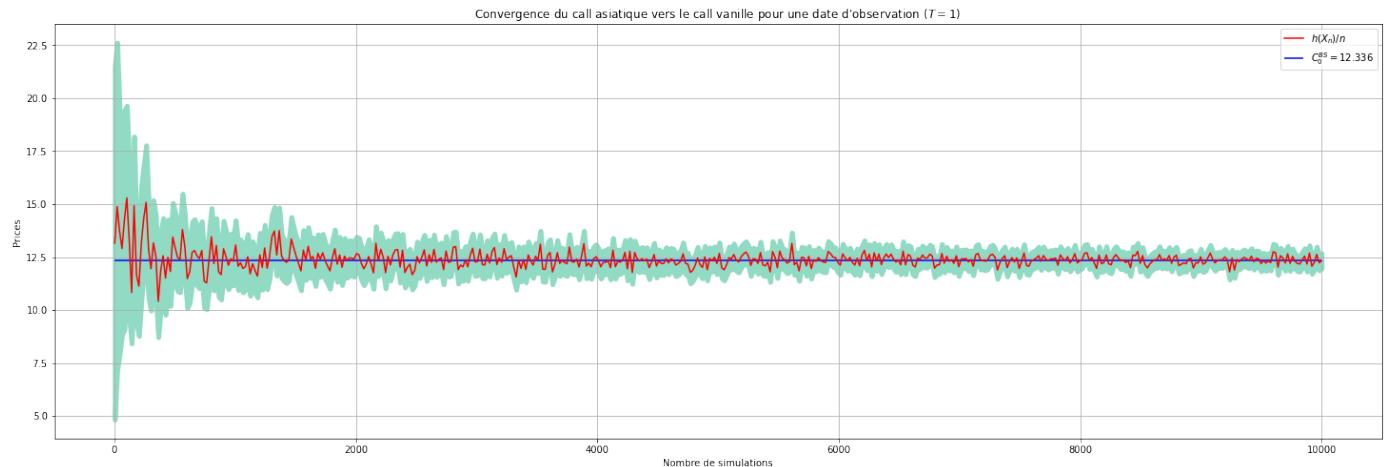


Figure 2.6: Convergence du prix pour 500 nombre de simulations dans l'intervalle [10,5000]

Nous pouvons donc voir la convergence vers le prix du call pour les mêmes paramètres et $T = 1$: $C_0^{BS} = 12.336$.

2.2 Réduction de Variance

Ces méthodes sont utiles pour réduire la variance de l'estimateur, donc réduire l'intervalle de confiance et améliorer la précision.

2.2.1 Variables antithétiques

Principe

Cette méthode profite du fait que la distribution de la loi normale centrée réduite est symétrique : X et $-X$ suivent la même loi $\mathcal{N}(0, 1)$.

Le principe est qu'en simulant un nombre $2N$ d'échantillons X aléatoires, nous ne pouvons prendre que les N premiers, créer deux estimateurs m_1 et m_2 avec respectivement les échantillons X et $-X$ associés, de taille N .

Nous avons donc deux estimateurs de Monte-Carlo, et désormais

$$C_a = \frac{m_1 + m_2}{2}$$

qui est un estimateur sans biais, asymptotiquement normal et convergent, de variance :

$$\mathbb{V}[C_a] = \frac{1}{4} (\mathbb{V}[m_1] + \mathbb{V}[m_2] + 2\text{Cov}(m_1, m_2))$$

Nous avons donc une réduction de variance par rapport au cas à $2N$ simulations si $\text{Cov}(m_1, m_2) < 0$.

Pour notre cas, cela permet de supprimer la variance due à la partie impaire du payoff. Si le payoff est symétrique nous n'avons pas de réduction de variance.

Résultats

💡 : Les méthodes de réduction de variance ont également été implémentées pour le pricing par MC d'options européennes dans la classe `MonteCarloBS_European`.

La fonction `display_results` de la classe mère `MonteCarloBS` a été modifiée afin de rajouter l'output du cas de

la réduction à celui existant, permettant de comparer directement (voir Notebook).

Nous affichons $\sigma_{m_1}, \sigma_{m_2}, \text{Cov}(m_1, m_2), \sigma_{h(X)}^{anth}$, l'estimation et l'intervalle de confiance, au dessus de l'output habituel. Nous décidons de mettre l'argument `fix_seed` à False.

En première comparaison, nous estimons le prix d'un call Asiatique par les deux méthodes pour dates = [0.2, 0.5, 1], maturité $T = 1$, 1000000 simulations, avec les écarts-types et intervalles de confiance correspondants :

	C_0	$\sigma_{h(X)}$	IC
MC	7,4996	10,894	[7,478; 7,521]
MC _{anth}	7,4978	5,595	[7,487; 7,509]

Table 2.1: Comparaisons des résultats pour Monte-Carlo classique et avec variables antithétiques

Afin d'avoir une meilleure vue sur la réduction de variance et de la convergence des prix lors de l'augmentation du nombre de simulations, un graphique est présenté pour 100 nombre de simulations $\in [100, 10000]$, avec les intervalles de confiance associées aux deux estimations :

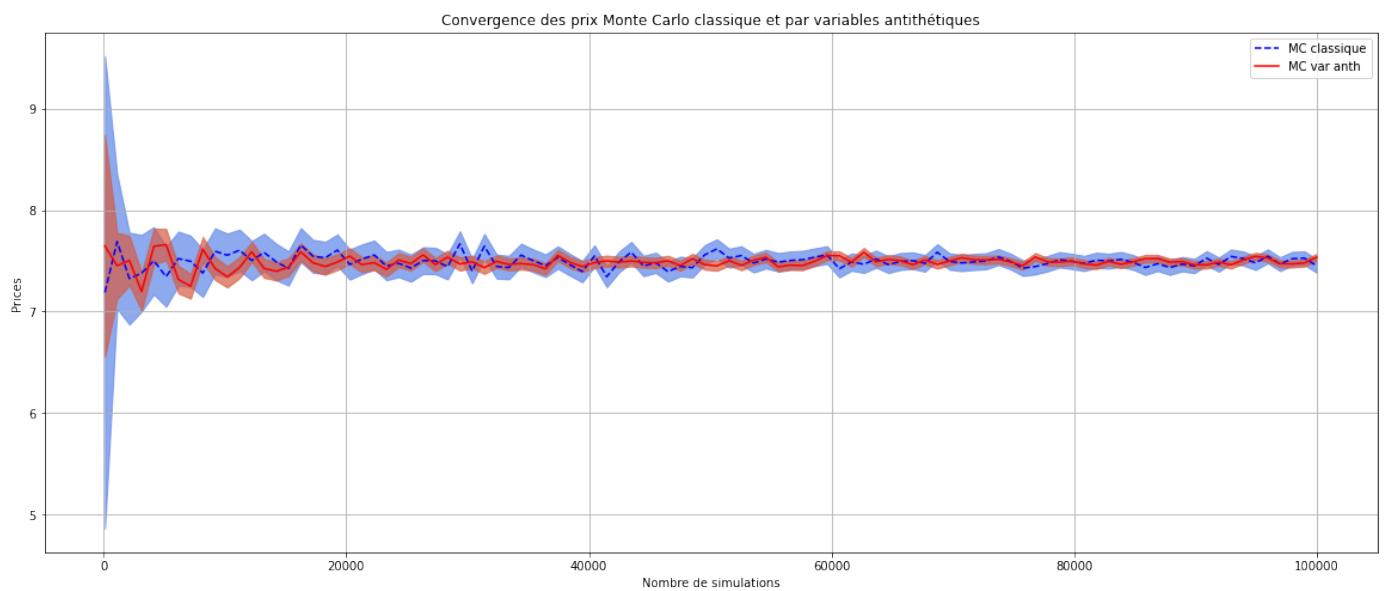


Figure 2.7: Convergence des prix par Monte-Carlo classique et variables antithétiques de l'option call Asiatique avec intervalles de confiance

Les intervalles de confiance dans le cas des variables antithétiques étant plus étroits, nous pouvons assurer qu'il y a réduction de variance.

2.2.2 Variables de contrôle

💡 : Une fonction `display_results_Z` a été ajoutée dans la classe `MonteCarloBS_Asian` afin de rajouter l'output du cas de la réduction à celui existant de la même manière que précédemment (voir Notebook).

Principe

On profite du fait que nous connaissons déjà analytiquement certaines formules.

On suppose que nous connaissons l'espérance d'une certaine fonction de nos N réalisations aléatoires : $\mathbb{E}[Z]$ avec $Z_i = f(X_i)$.

On a alors :

$$C_c = \frac{1}{N} \sum_{i=1}^N [h(X_i) + c(Z_i - \mathbb{E}[Z])]$$

qui est un estimateur sans biais, asymptotiquement normal et convergent.

Nous avons

$$\mathbb{V}[h(X_i) + c(Z_i - \mathbb{E}[Z])] = \mathbb{V}[h(X)] + c^2 \mathbb{V}[Z] + 2c \text{Cov}(h(X), Z)$$

On retrouve donc un polynôme d'ordre 2 en c , qui doit être tel que la variance soit minimale. Nous avons donc

$$c^* = -\frac{\text{Cov}(h(X), Z)}{\mathbb{V}[Z]}$$

Nous l'estimerons avec les variables simulées.

Il y a donc toujours réduction de variance, et plus les variables aléatoires $h(X)$ et Z sont corrélées, plus la réduction est forte.

Résultats

Nous allons utiliser trois variables de contrôles différentes :

$$Z_1 = S_T \quad | \quad Z_2 = \frac{1}{N} \sum_{i=1}^N S_{t_i} \quad | \quad Z_3 = e^{-rT} (S_T - K)_+$$

Dans notre cas, nous connaissons déjà les espérances des trois variables, nous avons :

$$\mathbb{E}[Z_1] = S_0 e^{rT} \quad | \quad \mathbb{E}[Z_2] = \frac{1}{N} \sum_{i=1}^N S_0 e^{r t_i} \quad | \quad \mathbb{E}[Z_3] = C_0^{BS}$$

Ainsi, nous sommes capable de réduire d'autant plus notre erreur d'estimation. En effet, calculer les espérances par une moyenne (MonteCarlo) fait perdre en précision du à l'approximation.

Nous fixons l'argument *fix_seed* à True ici afin de comparer les estimations entre espérances analytiquement connues ou non.

Pour les mêmes paramètres que précédemment et avec les espérances estimées, nous obtenons :

	C_0	$\sigma_{h(X)}$	IC
MC	7,5098	10,907	[7,488; 7,531]
MC_{Z_1}	7,5098	5,757	[7,499; 7,521]
MC_{Z_2}	7,5098	4,440	[7,501; 7,518]
MC_{Z_3}	7,5098	4,836	[7,500; 7,519]

Table 2.2: Comparaisons des résultats pour Monte-Carlo classique et avec variables de contrôle
(Espérances estimées)

Nous obtenons donc que la variable de contrôle Z_2 est celle réduisant le plus la variance dans ce cas de figure. On note également que le fait d'avoir mis la maturité comme date d'observation a d'autant plus réduit la variance avec la variable de contrôle Z_3 .

Avec les espérances "connues", nous obtenons :

	C_0	$\sigma_{h(X)}$	IC
MC	7,5098	10,907	[7,488; 7,531]
MC_{Z_1}	7,4920	5,757	[7,481; 7,503]
MC_{Z_2}	7,4972	4,440	[7,488; 7,506]
MC_{Z_3}	7,4907	4,836	[7,481; 7,500]

Table 2.3: Comparaisons des résultats pour Monte-Carlo classique et avec variables de contrôle
(Espérances connues)

On note que nous avons la même source aléatoire donc les mêmes variances que dans le cas précédent, mais le fait d'avoir utilisé des espérances analytiquement connues nous a permis d'avoir une meilleure estimation sur le prix.

Partie 3

Question III - Options à barrières

3.1 Monte Carlo pour Option à barrières

💡 : Pour cette partie, les classes *MonteCarloBS_UpAndOut* et *MonteCarloBS_DownAndIn* ont été implémentées. Les paramètres par défaut sont les mêmes que pour la classe de l'option asiatique avec en plus respectivement les barrières $M = 120$ et $B = 90$:

- dates (liste de dates discrètes à fournir obligatoire)
- nb_sim_path = 1000,
- $S_0 = 100$,
- $M = 120$ (Up&Out), $B = 90$ (Down&In),
- $K = 100$,
- $T = 1$,
- $\sigma = 0.25$,
- $r = 0.05$.

Options à barrière Up&Out et Down&In

Une option d'achat Up&Out discrète est un produit dérivé payant à la maturité T le payoff :

$$(S_T - K)_+ \mathbb{1}_{\max(S_{t_1}, \dots, S_{t_N}) < M}$$

Avec (t_1, \dots, t_N) représentant N dates déterministes dans l'intervalle $]0, T]$.

On définit de manière similaire le payoff d'une option Down&In discrète :

$$(S_T - K)_+ \mathbb{1}_{\min(S_{t_1}, \dots, S_{t_N}) \leq B}$$

Nous pouvons ainsi en déduire les évaluations par mesure martingale suivante :

$$C_0^{U\&O} = \mathbb{E}^Q \left[e^{-rT} (S_T - K)_+ \mathbb{1}_{\max(S_{t_1}, \dots, S_{t_N}) < M} \right] \quad | \quad C_0^{D\&I} = \mathbb{E}^Q \left[e^{-rT} (S_T - K)_+ \mathbb{1}_{\min(S_{t_1}, \dots, S_{t_N}) \leq B} \right]$$

Nous pouvons ainsi obtenir le pricing de ces deux options par Monte Carlo.

Pour cela, nous allons donc simuler un certain nombre de trajectoire et en estimer le prix de la même manière que pour l'option Asiatique.

Evolution du prix pour $S_0 \in [0, 200]$ avec call européen en comparaison

Paramètres : Par défaut pour l'européen.

Par défaut pour les Up&Out et Down&In, sauf $T = 90/365$, 10000 simulations de trajectoires avec *fix_seed=True*, avec respectivement 10 et 90 dates d'observations partant de "aujourd'hui" avec une fréquence journalière. Nous avons choisi 90 dates (toute la période) pour l'option Down&In afin de ne pas avoir un prix trop faible, par soucis de lisibilité. De même pour l'option Up&Out, où nous avons pris 10 dates et pas plus.

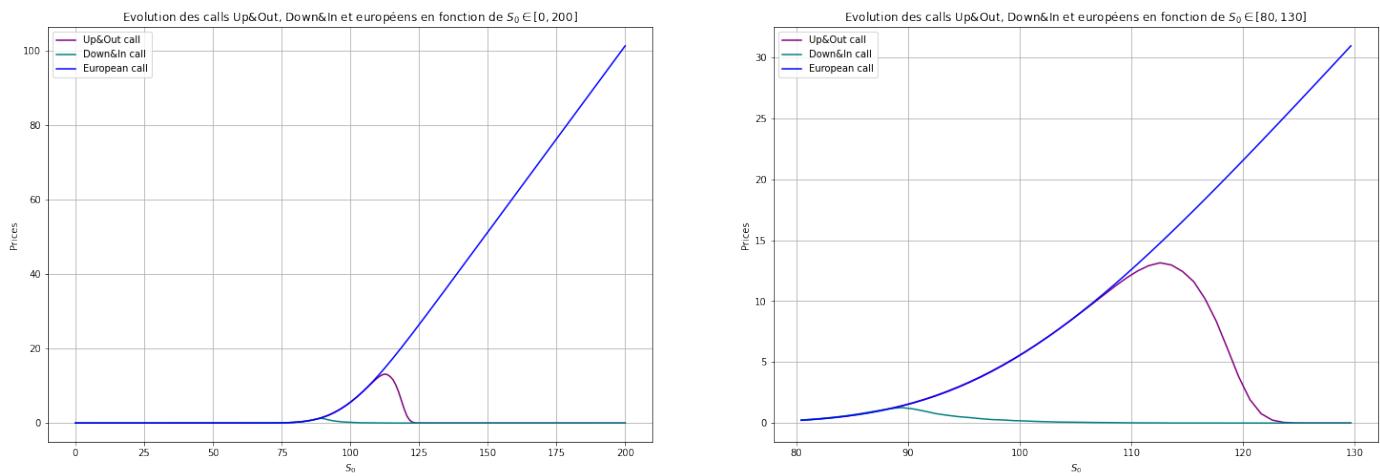


Figure 3.1: A gauche : Evolution des calls vanille, Up&Out et Down&In en fonction de $S_0 \in [0, 200]$.
A droite : En fonction de $S_0 \in [80, 130]$

Pour l'option Up&Out, on voit que le prix est le même que celui du call vanille jusqu'à environ $S_0 = 110$, où le prix commence à décroître, du fait de la plus grande probabilité de voir le sous-jacent franchir la barrière au cours des 10 prochains jours, jusqu'à s'annuler aux alentours de $S_0 = 123$, où le prix n'a presque plus de chance de ne pas dépasser M lorsque qu'il démarre quelque peu au dessus de celle-ci.

On note que si nous avions donnée les 10 derniers jours avant la maturité en date d'observations, le prix serait tombé à 0 bien plus au niveau de la barrière, car le sous-jacent aura moins de chances de ne pas l'avoir franchit pour des dates d'observations plus éloignées ($r > 0$).

Pour l'option Down&In, on voit que le prix est le même que celui du call vanille jusqu'à environ $S_0 = 90$, où le prix atteint son maximum puis décroît, jusqu'à s'annuler aux alentours de $S_0 = 100$. En effet plus S_0 est élevé moins il y a de chances que le sous-jacent atteigne la barrière activante.

Evolution du prix du call Up&Out en fonction du nombres de dates d'observations : $Nb_{dates} \in [1, 100]$

En partant de T (dates d'observations décroissantes)

Paramètres : 10000 simulations de trajectoires avec `fix_seed=True` et `all_days=True` pour chaque simulation, afin d'avoir exactement chaque estimation basée sur les mêmes trajectoires.

La première date d'observation est la maturité ($T = 1$ par défaut). A chaque itération, nous rajoutons des dates antérieures de fréquence 2 jours.

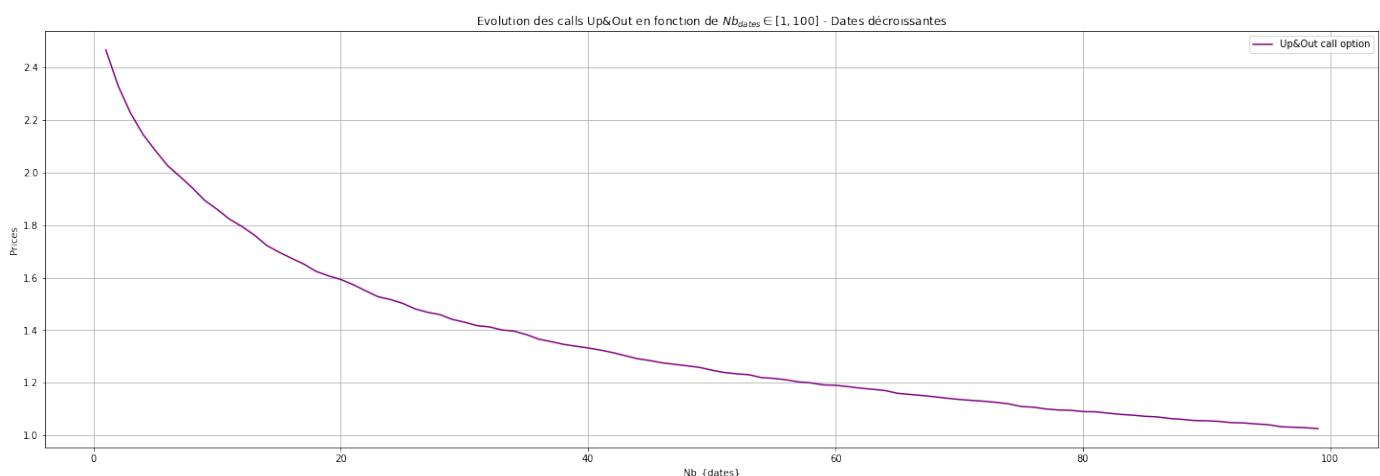


Figure 3.2: Evolution du prix de l'option Up&Out pour $Nb_{dates} \in [1, 100]$ et 10000 simulations de trajectoires
Dates décroissantes

On remarque que plus l'on soumet de dates d'observations, plus le prix de l'option en est diminué. Nous vérifions bien que l'ajout de dates fait baisser le prix de notre option, en effet, plus de dates d'observations signifie plus de chances de voir notre sous-jacent franchir la barrière.

En partant de $t_0 + 1$ (dates d'observations croissantes)

Paramètres : Identique sauf que la première date est "demain" (S_0 ne peut pas être pris en compte dans le payoff), et que nous ajoutons des dates ultérieures de même fréquence que précédemment.

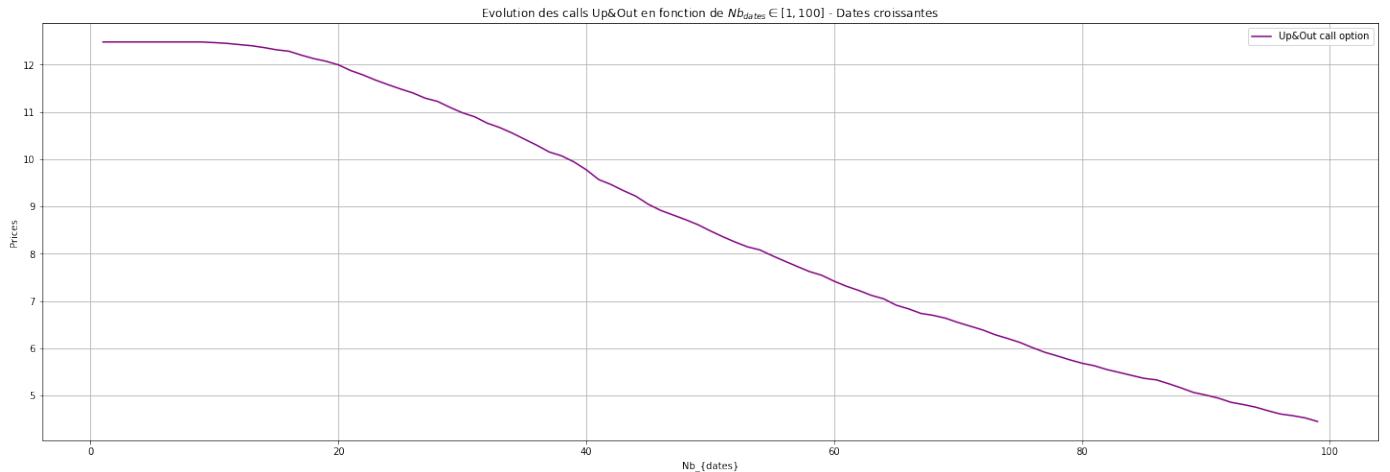


Figure 3.3: Evolution du prix de l'option Up&Out pour $Nb_{dates} \in [1, 100]$ et 10000 simulations de trajectoires Dates croissantes

De même, nous voyons que l'ajout de dates fait également baisser le prix de notre option, pour les mêmes raisons que précédemment.

Pour les premières dates, le prix est très proche de celui du call vanille pour les mêmes paramètres ($C^{BS} = 12.34$), du fait de la très faible probabilité que le sous-jacent ait franchi la barrière dans les premiers jours. On peut remarquer qu'au bout de 15 jours, le prix de l'option commence à décroître.

De plus, on remarque que le prix de l'option dans le cas d'ajout de date croissantes est bien supérieur au cas d'ajout de dates décroissantes partant de la maturité. Cela vient du fait que plus nous sommes éloignés de la date S_0 , plus l'actif aura de chance d'avoir dépassé la barrière ($r > 0$), le prix du call Up&Out baissera donc fortement, comparé aux dates proches de S_0 . Ce que nous vérifions ici.

Nous avons choisi de montrer les deux cas séparément afin de garder la même structure que pour le cas des options asiatiques.

On peut vérifier de la même manière que dans le cas de l'option Down&In, plus nous augmentons le nombre de dates d'observations et plus le prix augmentera, en effet, plus il y a de dates d'observations et plus on a de chance d'avoir un une date où le sous-jacent a franchi la barrière activante.

Evolution des prix en fonction de la volatilité

Paramètres : Par défaut sauf nb_sim_path=10000, $T = 90/365$, fix_seed=True, toute la période (les 90 jours) en date d'observations pour les deux options barrières, et 50 différents $\sigma \in [5\%, 100\%]$

Une chose intéressante dans le cas de ces deux options serait de regarder leur prix en fonction de l'évolution de la volatilité. Nous savons que dans le modèle de Black&Scholes le prix est une fonction croissante de la volatilité.

Intuitivement, nous imaginons que le prix de l'option Up&Out sera décroissant en fonction de la volatilité, et celui de l'option Down&In croissant. En effet, une augmentation de la volatilité augmentera les chances de voir le prix franchir la barrière, qu'elle soit activante ou désactivante.

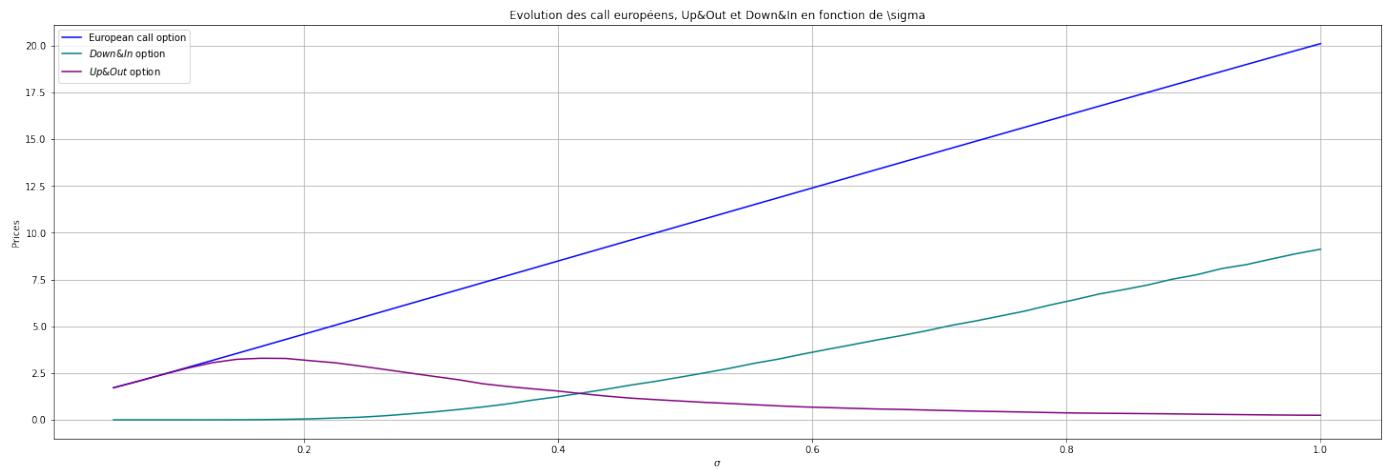


Figure 3.4: Evolution des calls vanille, Up&Out et Down&In en fonction de $\sigma \in [0.05, 100]$.

Nous vérifions donc bien les hypothèses. En effet le prix de l'option Up&Out décroît dès que $\sigma \approx 15\%$, et le prix de l'option Down&In augmente dès que $\sigma \approx 20\%$.

3.2 Surveillance continue de la barrière

💡 : Une fonction `upout_continuos_barrier` est créée pour estimer le prix de la barrière continue. Nous avons décidé de ne pas l'incorporer dans la classe afin de traiter cette partie à part.

Principe

Pour B_t un mouvement brownien, $\alpha \in \mathbb{R}$, et $X_t = \alpha t + B_t$, nous pouvons simuler des variables aléatoires suivant la même loi que la densité conditionnelle de $M_t^X = \sup_{0 \leq u \leq t} X_u$ sachant $X_t = x$. En effet, pour $u \sim \mathcal{U}(0, 1)$, $\frac{x + \sqrt{x^2 - 2T \log(u)}}{2}$ suit la même loi que M_T^X sachant $X_T = x$.

Dans notre cas, nous cherchons à simuler le maximum courant de chaque S_T afin d'avoir un prix de surveillance continue de la barrière.

Nous avons $S_T = S_0 e^{(r - \frac{\sigma^2}{2})T + \sigma B_T}$. Donc :

$$\frac{\log(\frac{S_T}{S_0})}{\sigma} = \left(\frac{r}{\sigma} - \frac{\sigma}{2} \right) T + B_T = \alpha T + B_T = X_T$$

Avec $\alpha = \left(\frac{r}{\sigma} - \frac{\sigma}{2} \right)$. Ainsi on a $S_T = S_0 e^{\sigma X_T}$.

Nous sommes donc en mesure de simuler le maximum courant M_T^X de X_T et d'en déduire celui de S_T : $M_T^S = S_0 e^{\sigma M_T^X}$.

Nous allons donc simuler un certain nombre de X_T , calculer les S_T correspondants, considérer seulement les $S_T > K$ et $S_T < M$ (sinon le payoff est nul), puis pour chacun, nous allons simuler le M_T^X afin d'en déduire le M_T^S . Nous calculons ensuite le payoff $\mathbb{E} \left[e^{-rT} (S_T - K) \mathbb{1}_{S_T < M_T^S} \right]$ (le $+$ est déjà compris dans le fait que nous ne considérons que les $S_T > K$).

Nous aurons donc une estimation par MonteCarlo du prix avec surveillance continue de la barrière. Notre but est de montrer que le prix avec dates d'observations discrètes tend vers ce prix quand on augmente le nombre d'observations, pour les mêmes paramètres.

Résultats

Paramètres : Nous estimons le prix de la barrière continue une seule fois avec les paramètres par défaut et $10^6 S_T$ différents afin d'avoir une estimation robuste pour pouvoir l'utiliser comme limite de convergence. Nous sélectionnons 100 fois un nombre de dates en échelle log, avec `np.logspace(1, 4, 100)`, allant de 10 à 10000

dates, en prenant pour chaque nombre de dates intermédiaire l'arrondi inférieur. Les dates sont fournies en fraction d'année dans cette partie, le paramètre `all_days` n'est pas utilisé car est limité aux **jours**, ce qui ne serait pas suffisant pour illustrer la convergence, et le paramètre `fix_seed` est laissé à `False`. Chaque estimation est faite avec `nb_sim_path= 10000`.

Nous obtenons $C_{cont}^{U\&O} = 0.69$ pour la surveillance continue de la barrière. Voici le tracé obtenu, en échelle log pour l'axe des x:

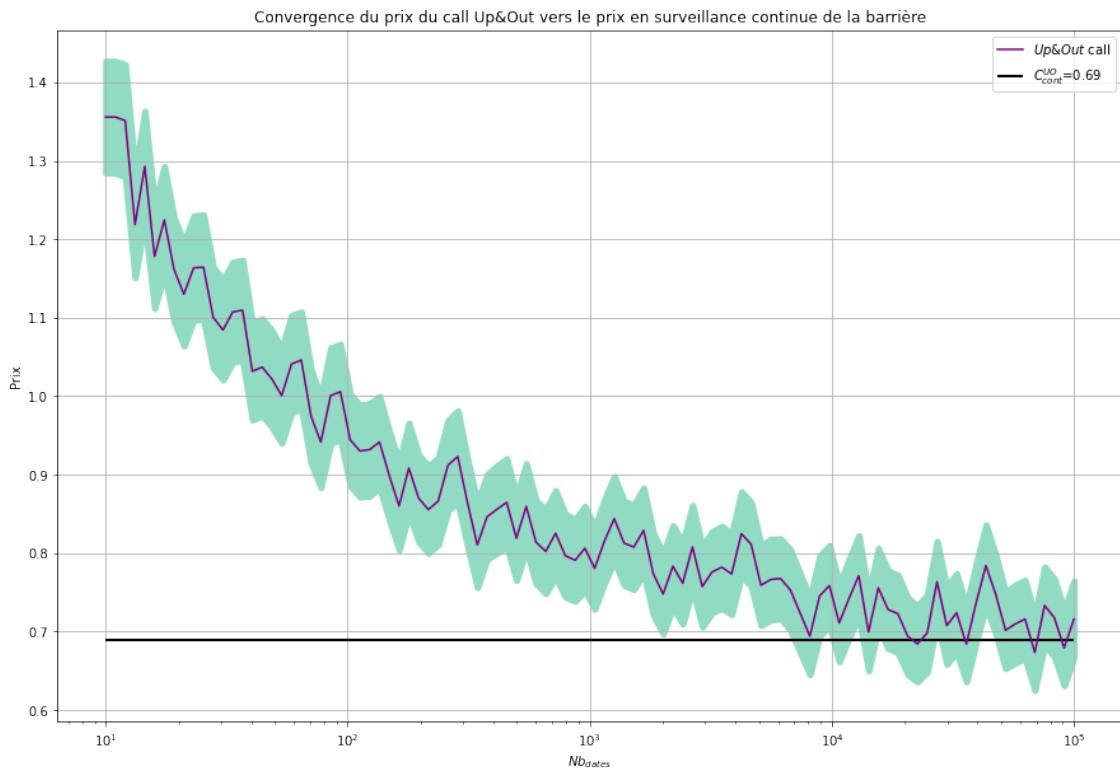


Figure 3.5: Convergence du prix de l'option call Up&Out vers le prix en surveillance continue avec intervalles de confiance

Nous voyons donc bien une convergence du prix lorsque nous augmentons le nombre de dates d'observations.

3.3 Réduction de Variance par Monte Carlo Conditionnel

💡 : Pour cette question, nous avons rajouté un argument `mc_cond` dans la classe `MonteCarloBS_Down&In` afin de redéfinir ("overidder") les méthodes :

- `simulate_path` pour intégrer directement le calcul du prix du call à notre vecteur de payoff `h_x`.
- `compute_h_x` qui retournera seulement le vecteur `h_x`, cela permet de ne pas modifier le code de la classe parente `MonteCarloBS_PathDependant`.

La date de maturité n'est donc jamais simulée ici (à part si donnée en date d'observation), et si la barrière est franchie à t^* , les simulations aux dates d'observations supérieures à t^* sont laissées à 0.

Nous avons décidé de laisser l'argument `fix_seed` à `False`.

Principe

On considère $\mathbb{E}[h(X)|\mathcal{G}]$ comme estimateur à la place de $h(X)$. Par le théorème de la variance totale, nous avons :

$$\mathbb{V}[h(X)] = \mathbb{V}[\mathbb{E}[h(X)|\mathcal{G}]] + \mathbb{E}[\mathbb{V}[h(X)|\mathcal{G}]]$$

Comme nous avons $\mathbb{V}[\mathbb{E}[h(X)|\mathcal{G}]] \leq \mathbb{V}[h(X)]$, nous obtenons une réduction de variance. En effet on utilise plus d'informations pour diminuer la variance de l'estimateur.

Dans notre cas, pour un call Down&In, nous allons considérer le fait que si pour un certain t^* le sous-jacent franchit la barrière, alors l'option est activée et nous pouvons directement calculer le prix du call $C^{BS}(T - t^*)$, qu'on actualise pour $t = 0$ (donc le discount sera de e^{-rt^*}).

Résultats

De la même manière que pour la partie 2, nous estimons le prix d'un call Down&In avec les deux méthodes pour 10 dates d'observations réparties entre aujourd'hui et la maturité $T = 1$, et 1000000 simulations, avec les écarts-types et intervalles de confiance correspondants :

	C_0	$\sigma_{h(X)}$	IC
MC	1,4963	5,940	[1,485; 1,508]
MC_{cond}	1,4961	1,905	[1,492; 1,500]

Table 3.1: Comparaisons des résultats pour Monte-Carlo classique et conditionnel

Comme en partie 2, un graphique est présenté pour 100 nombre de simulations $\in [100, 10000]$, avec les intervalles de confiance associées aux deux estimations.

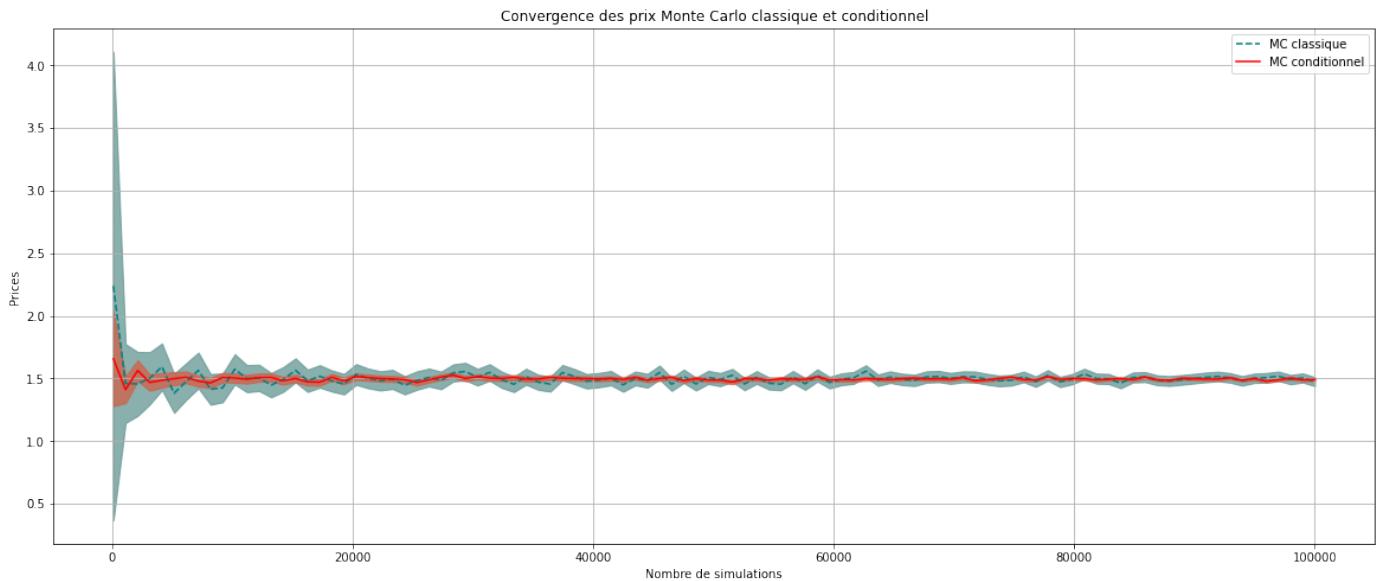


Figure 3.6: Convergence des prix par Monte-Carlo classique et conditionnel de l'option call Down&In avec intervalles de confiance

Nous voyons donc la réduction de variance pour le cas conditionnel, du aux intervalles de confiance plus étroits, ainsi que la convergence des deux prix.

Partie 4

EDP pour le modèle de Black & Sholes

 : Les classes `BS_PDE_FinitDiff` (héritant de la classe `BS`) et `BS_finitDiff_Schemes` sont implémentées.

4.1 Résolution de l'EDP de Black Scholes

Dans cette partie nous allons mettre en évidence l'ensemble des étapes de calculs réalisés pour arriver à la forme de l'EDP qui nous a servie pour l'utilisation des méthodes de différences finies.

On considère tout d'abord l'EDP de Black&Scholes sous sa forme initiale, où $v(t, x)$ le prix du produit dérivé à la date $t \in [0, T]$ dans l'état $S_t = x$ est solution de :

$$\frac{\partial v(t, x)}{\partial t} + r x \frac{\partial v(t, x)}{\partial x} + \frac{1}{2} \sigma^2 x^2 \frac{\partial^2 v(t, x)}{\partial x^2} = r v(t, x)$$

Munie de la condition terminale $v(T, x) = h(X)$, $x \in \mathbb{R}^+$. L'EDP ne dépend pas du produit dérivé mais seulement du payoff $h(x)$ pour la condition terminale.

On pose le changement de variable $y = \log(x)$ avec x le prix du sous jacent. On peut donc écrire $v(t, e^y) = v(t, x)$ et :

$$\frac{\partial v(t, e^y)}{\partial x} = \frac{\partial v(t, e^y)}{\partial y} * \frac{\partial y}{\partial x} = \frac{\partial v(t, e^y)}{\partial y} * \frac{1}{x}$$

On fait de même pour toutes les dérivées partielles.

On effectue une deuxième changement de variable (renversement du temps) en posant $\tau = T - t$:

$$\frac{\partial u(\tau, y)}{\partial \tau} = \frac{\partial u(t, y)}{\partial t} * \frac{\partial t}{\partial \tau} = \frac{\partial u(t, y)}{\partial t} * -1$$

On fait de même pour toutes les dérivées partielles.

On obtient alors l'EDP suivante :

$$\frac{\partial u(\tau, y)}{\partial \tau} - \frac{1}{2} \sigma^2 \frac{\partial^2 u(\tau, y)}{\partial y^2} - \left(r - \frac{\sigma^2}{2} \frac{\partial u(\tau, y)}{\partial y} + r u(\tau, y) \right) = 0$$

muni de la condition initiale : $u(0, y) = (e^y - K)_+$

On notera pour la suite les coefficients : $\alpha = \frac{\sigma^2}{2}$; $\beta = (r - \frac{\sigma^2}{2})$ et $\gamma = r$ tel que l'EDP s'écrit :

$$\frac{\partial u(\tau, y)}{\partial \tau} - \alpha \frac{\partial^2 u(\tau, y)}{\partial y^2} - \beta \frac{\partial u(\tau, y)}{\partial y} + \gamma u(\tau, y) = 0$$

On note que les paramètres α, β, γ de cette EDP sont bien des constantes.

4.2 Prix de l'EDP en fonction de S_0

Dans le cadre du pricing d'un Call ou d'un Put à l'aide de méthodes numériques de différences finies, il est nécessaire de discréteriser l'espace. Dans l'ensemble de tous les tracés, nous avons pris les mêmes hypothèses sur les bornes supérieures et inférieures en temps et en espace afin de garder une certaine cohérence.

Les hypothèses utilisées pour l'ensemble des tracés sont les suivantes :

- $S_0 = 100$,
- $K = 140$
- $T = 1$,
- $r = 0.02$
- $\alpha = 0$ (absence de versement de devividendes)
- $\sigma = 0.15$
- Les calculs sont tous effectués pour un N et un M de 400.

Schema Explicite

Nous avons discréterisé l'intervalle de 0 à T en $N+1$ intervalles uniformes. Chaque temps de discréterisation sera donc de la forme :

$$\forall n \in [0, N] \quad t^n = t^0 + n\delta t = t^0 + n\frac{T}{N}$$

De la même manière, nous avons discréterisé le cours du sous-jacent qui est compris entre x_{min} et x_{max} en $M+1$ pas d'espace.

$$\forall i \in [0, M] \quad x_i = x_0 + n\frac{x_M - x_0}{M}$$

avec $x_M = x_{max}$ et $x_0 = x_{min}$

Pour la résolution de l'EDP par une méthode explicite, nous avons utilisé les formules des différences finies suivantes :

$$\frac{\partial u(x_i, t^n)}{\partial x} = \frac{u_{i+1}^n - u_{i-1}^n}{2\delta x}$$

$$\frac{\partial u(x_i, t^n)}{\partial x^2} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\delta x^2}$$

$$\frac{\partial u(x_i, t^n)}{\partial t} = \frac{u_i^n - u_i^{(n-1)}}{\delta t}$$

En remplaçant les dérivées partielles dans l'EDP par les formules aux différences finies, nous obtenons :

$$u_i^{n-1} = au_{i+1}^n + bu_i^n + cu_{i-1}^n$$

avec $a = \delta t(\frac{\alpha}{\delta x^2} + \frac{\beta}{2\delta})$, $b = \delta t(\frac{1}{\delta t} - \frac{2\alpha}{\delta x^2} + \gamma)$ et $c = \delta t(\frac{\alpha}{\delta x^2} - \frac{\beta}{2\delta})$

A chaque itération n , On cherche à calculer le vecteur : $(u_1^{n-1}, u_2^{n-1}, \dots, u_{M-1}^{n-1})$

Avec comme conditions limites :

- $u(\tau, y) = 0$ quand y tend vers 0 pour le call
- $u(\tau, y) = e^y - Ke^{-r\tau}$ quand y tend vers l'infini pour le call

On notera que pour le put les conditions sont de 0 lorsque y tend vers l'infini et $Ke^{-r\tau} - e^y$ quand y tend vers 0.

Call

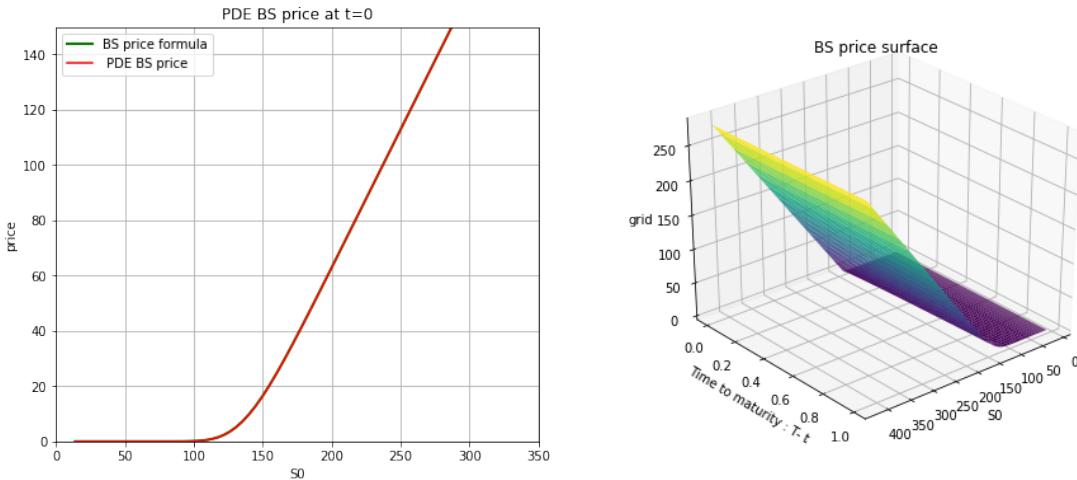


Figure 4.1: Evolution du prix du call par EDP en fonction de S_0
Schéma explicite

Put

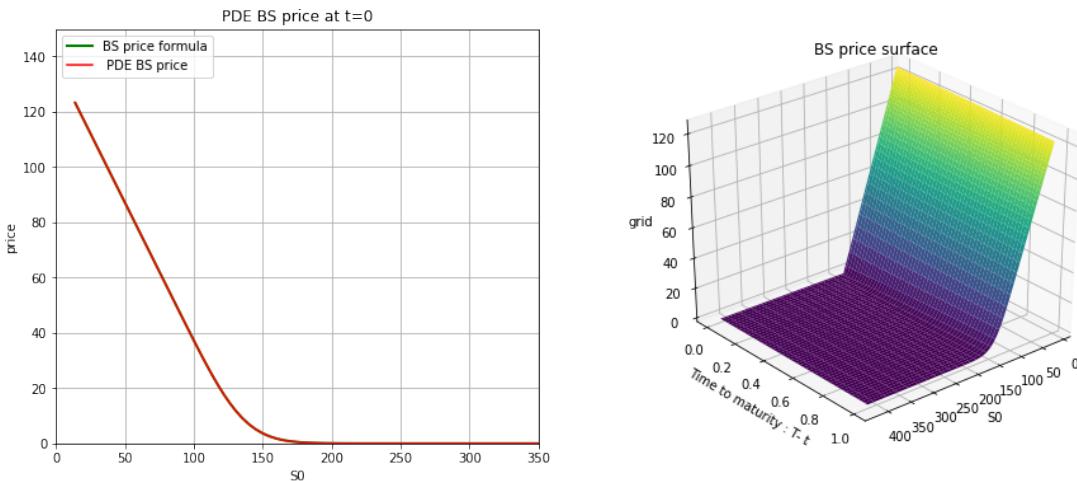


Figure 4.2: Evolution du prix du put par EDP en fonction de S_0
Schéma explicite

Schema Implicite

Pour la résolution de l'EDP par une méthode explicite, nous avons utilisé les formules des différences finies suivantes :

$$\frac{\partial u(x_i, t^{n-1})}{\partial x} = \frac{u_{i+1}^{n-1} - u_{i-1}^{n-1}}{2\delta x}$$

$$\frac{\partial u(x_i, t^{n-1})}{\partial x^2} = \frac{u_{i+1}^{n-1} - 2u_i^{n-1} + u_{i-1}^{n-1}}{\delta x^2}$$

$$\frac{\partial u(x_i, t^{n-1})}{\partial t} = \frac{u_i^n - u_i^{n-1}}{\delta t}$$

En remplaçant les dérivées partielles dans l'EDP par les formules aux différences finies, nous obtenons :

$$u_i^{n-1} = \zeta_1 u_{i+1}^{n-1} + \zeta_2 u_i^{n-1} + \zeta_3 u_{i-1}^{n-1}$$

avec $\zeta_1 = -\delta t(\frac{\alpha}{\delta x^2} + \frac{\beta}{2\delta})$, $\zeta_2 = \delta t(\frac{1}{\delta t} + \frac{2\alpha}{\delta x^2} - \gamma)$ et $\zeta_3 = \delta t(\frac{\beta}{2\delta} - \frac{\alpha}{\delta x^2})$

Les conditions limites sont les mêmes que pour le schéma explicite :

- $u(\tau, y) = 0$ quand y tend vers 0 pour le call
- $u(\tau, y) = e^y - Ke^{-r\tau}$ quand y tend vers l'infini pour le call

De même, pour le put les conditions sont de 0 lorsque y tend vers l'infini et $Ke^{-r\tau} - e^y$ quand y tend vers 0.

Call

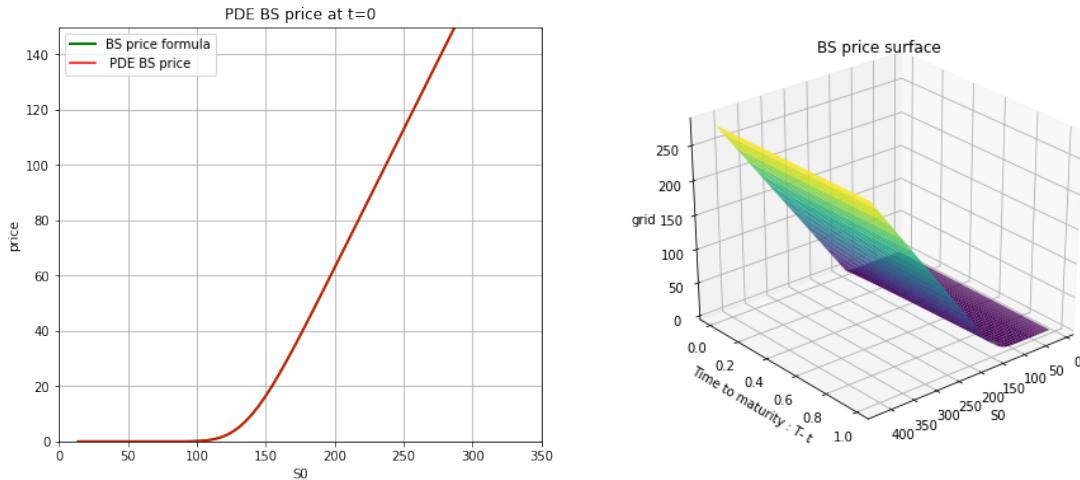


Figure 4.3: Evolution du prix du call par EDP en fonction de S_0
Schéma implicite

Put

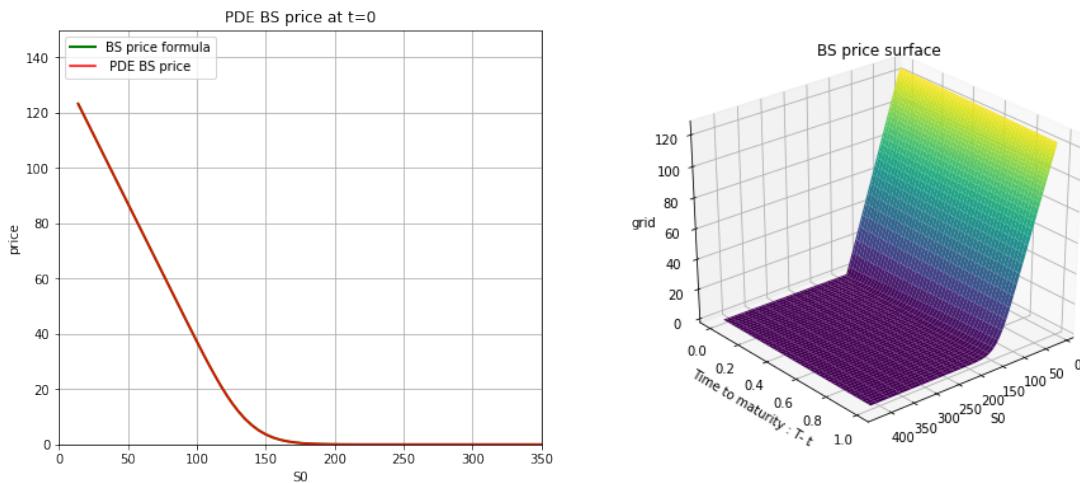


Figure 4.4: Evolution du prix du call par EDP en fonction de S_0
Schéma implicite

Crank-Nicolson

Le schéma de Crank–Nicolson est une méthode qui permet d'obtenir un résultat avec un ordre supérieur. Cette technique est en fait une sorte de combinaison linéaire entre la méthode implicite et la méthode explicite et plus précisément une moyenne entre les deux méthodes précédentes.

Call

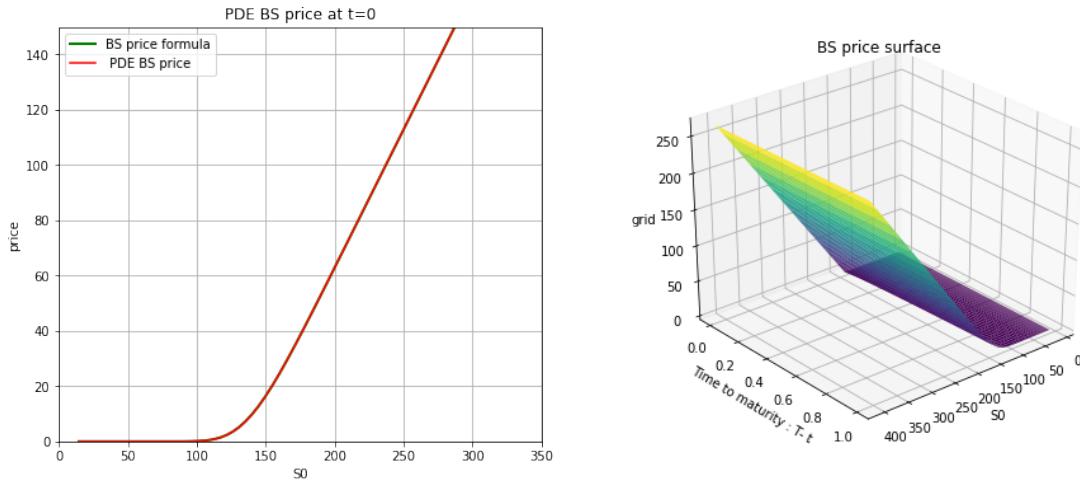


Figure 4.5: Evolution du prix du call par EDP en fonction de S_0
Schéma de Crank-Nickolson

Put

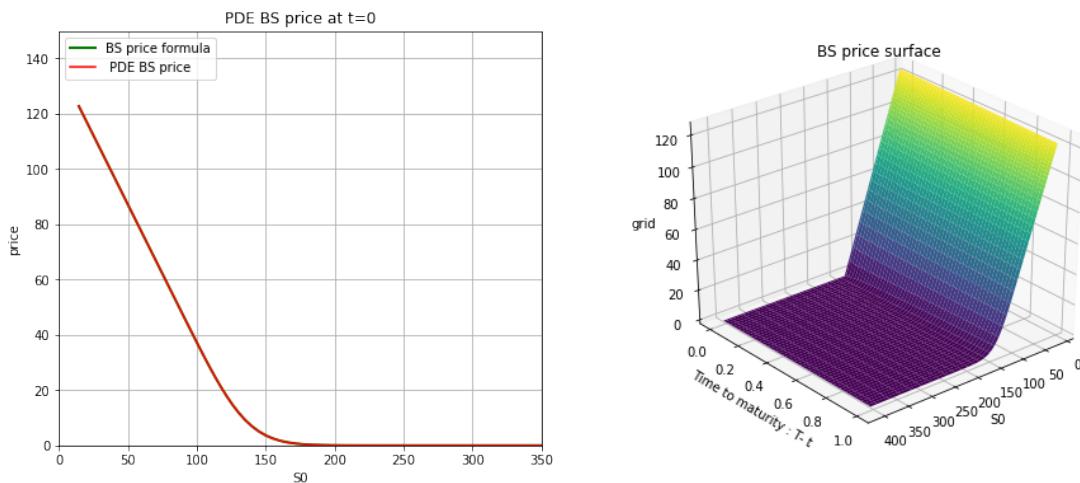


Figure 4.6: Evolution du prix du put par EDP en fonction de S_0
Schéma de Crank-Nickolson

4.3 Tracé des l'erreur Moyenne (L_1, L_2, L_∞)

En temps

Explicite

L'erreur en temps dans le cas du schéma explicite est peu réalisable ou alors dans des conditions qui ne permettent pas d'avoir des résultats fiables. En effet pour que le schéma explicite soit stable, il faut que la condition CFL soit inférieure à $\frac{1}{2}$. Or le rapport $\frac{\text{deltat}}{\delta x^2}$ rend impossible le choix d'un δ_x très petit devant δ_t . Il est impossible de tracer une erreur sans que nos résultats soit bruités par l'erreur au pas d'espace.

Implicite

Voici ce que l'on obtient dans le cas du schéma implicite :

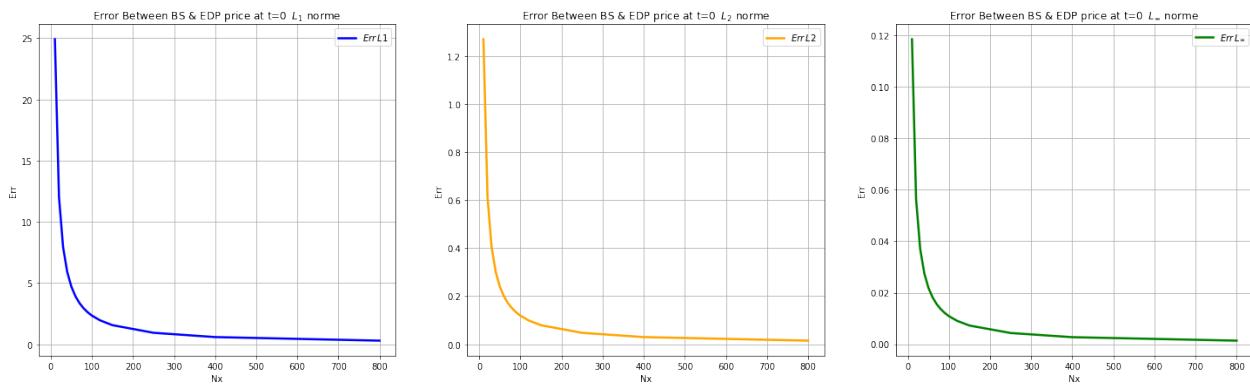


Figure 4.7: Tracé de l'erreur L_1 , L_2 et L_∞ en temps
Schéma implicite

Crank-Nickolson

De même, pour le schéma de Crank-Nickolson :

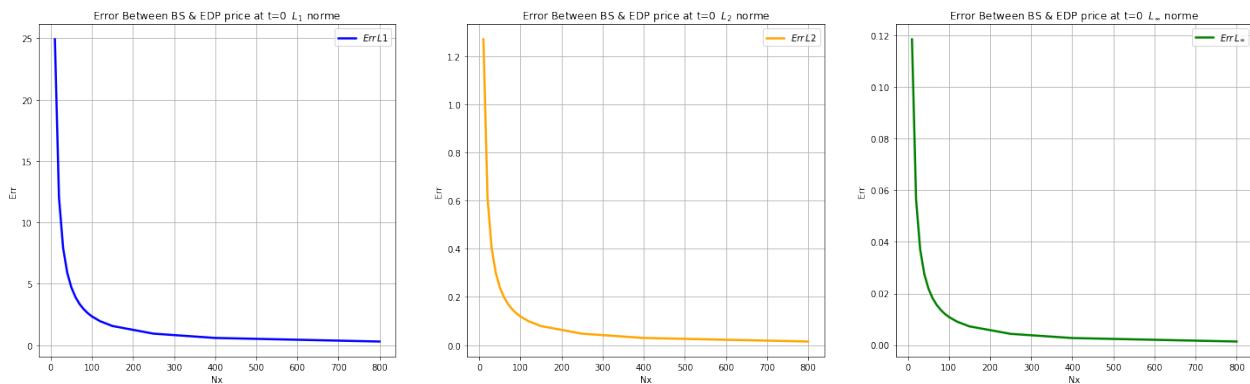


Figure 4.8: Tracé de l'erreur L_1 , L_2 et L_∞ en temps
Schéma de Crank-Nickolson

En espace

Explicite

Cette fois-ci, il est possible de tracer l'erreur car la condition CFL n'impose pas de restriction si on prend un δ_t infiniment petit devant δ_x

Voici nos erreurs en espace dans le cas du schéma explicite :

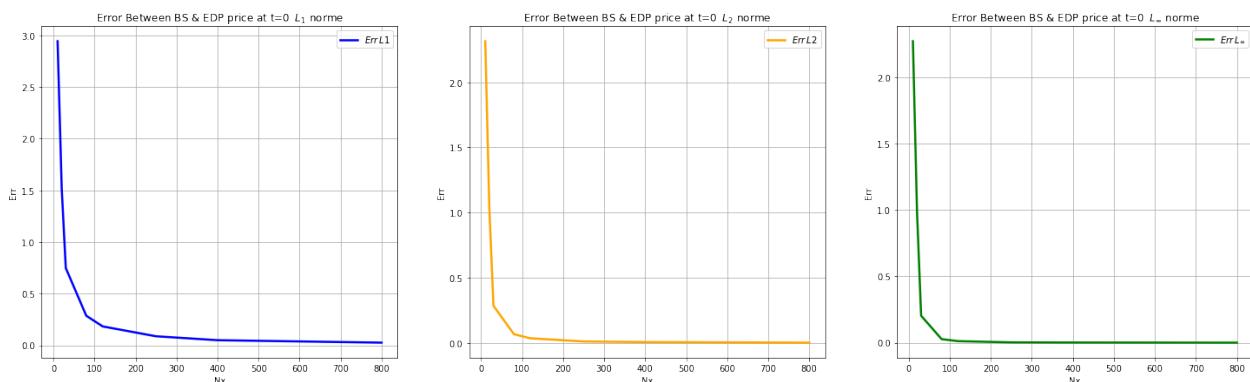


Figure 4.9: Tracé de l'erreur L_1 , L_2 et L_∞ en espace
Schéma explicite

Implicite

Ce que l'on obtient pour le cas du schéma implicite :

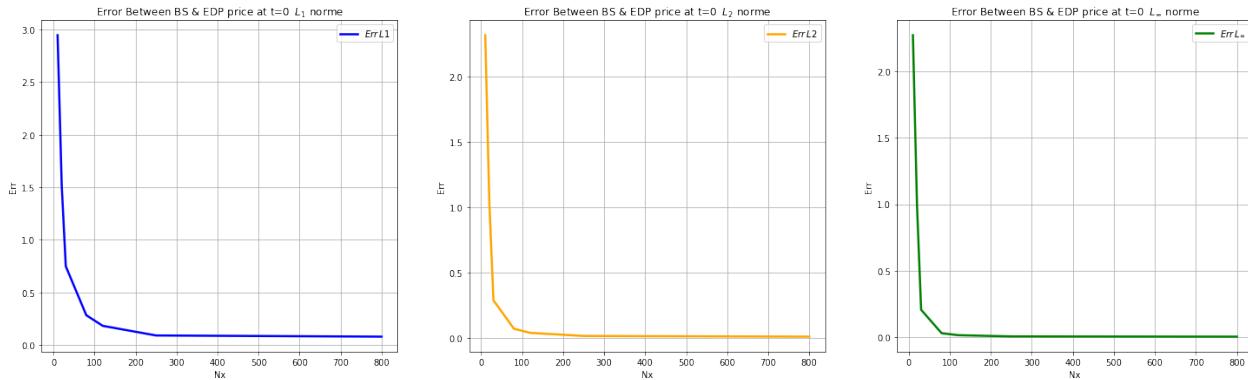


Figure 4.10: Tracé de l'erreur L_1 , L_2 et L_∞ en espace Schéma implicite

Crank-Nickolson

Et pour le schéma de Crank-Nickolson :

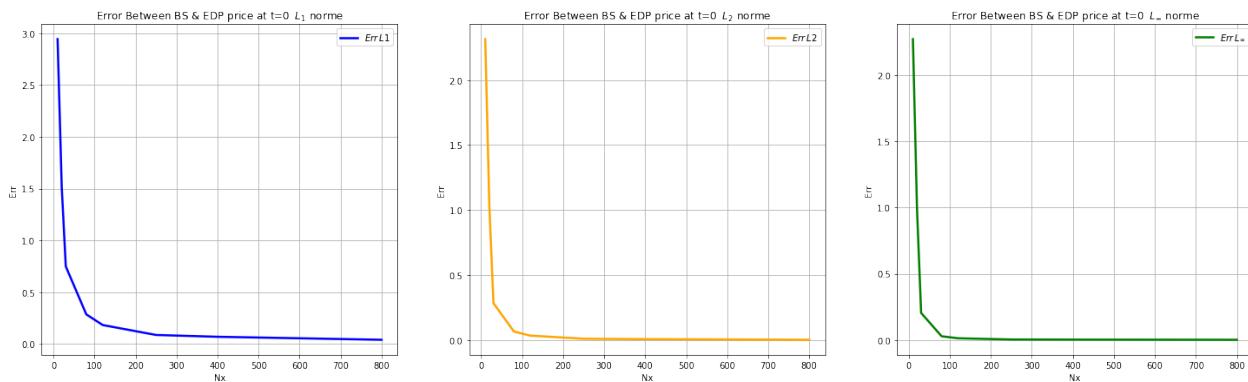


Figure 4.11: Tracé de l'erreur L_1 , L_2 et L_∞ en espace Schéma de Crank-Nickolson

Pour tous les tracés et tous les types d'erreurs analysés, on a bien une décroissance exponentielle vers 0 avec M et N .

4.4 Convergence des schémas

En temps

Pour illustrer la vitesse de convergence en Temps, on fait varier le pas de temps tout en fixant un pas d'espace très faible.

Explicite

La condition CFL dans ce cas pose problème pour tracer la convergence de notre schéma de manière robuste.

Implicite

Mais nous pouvons tracer celle-ci pour notre schéma implicite :

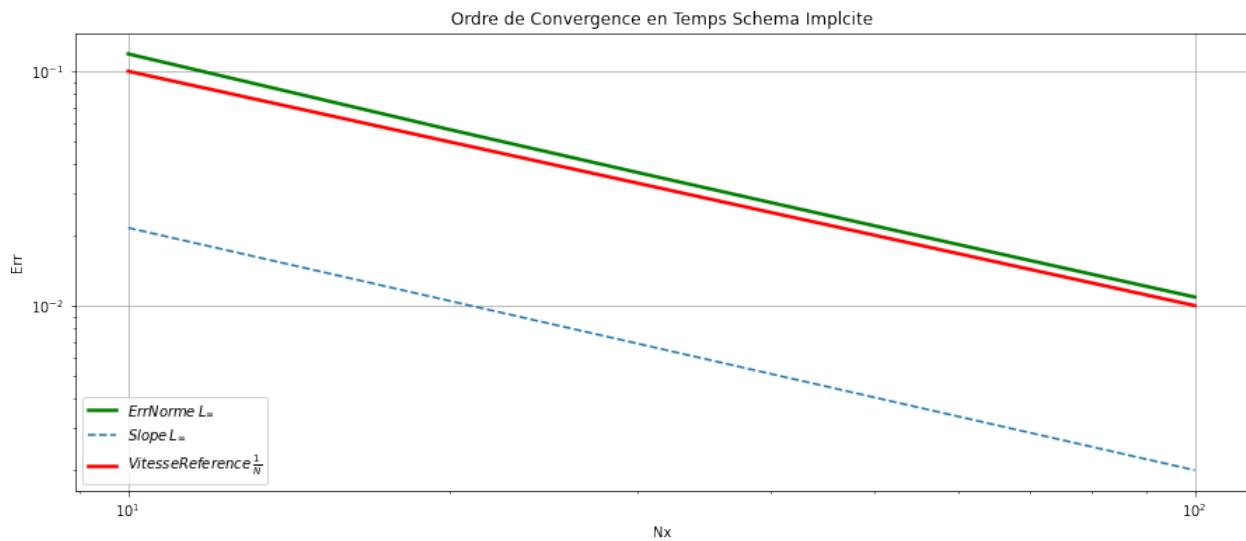


Figure 4.12: Convergence en temps pour le schéma implicite

Dans le cadre d'un schéma implicite, la convergence du schéma lorsque le pas d'espace tend vers 0 est à l'ordre 1. Lorsque que nous traçons l'erreur en norme infinie en loglog pour un pas d'espace très petit devant le pas de temps, alors nous pouvons valider empiriquement que notre schéma converge en la comparant à la vitesse de convergence de référence. Ici la courbe "slope" n'apporte aucune information supplémentaires car la courbe est déjà très régulière.

Crank-Nickolson

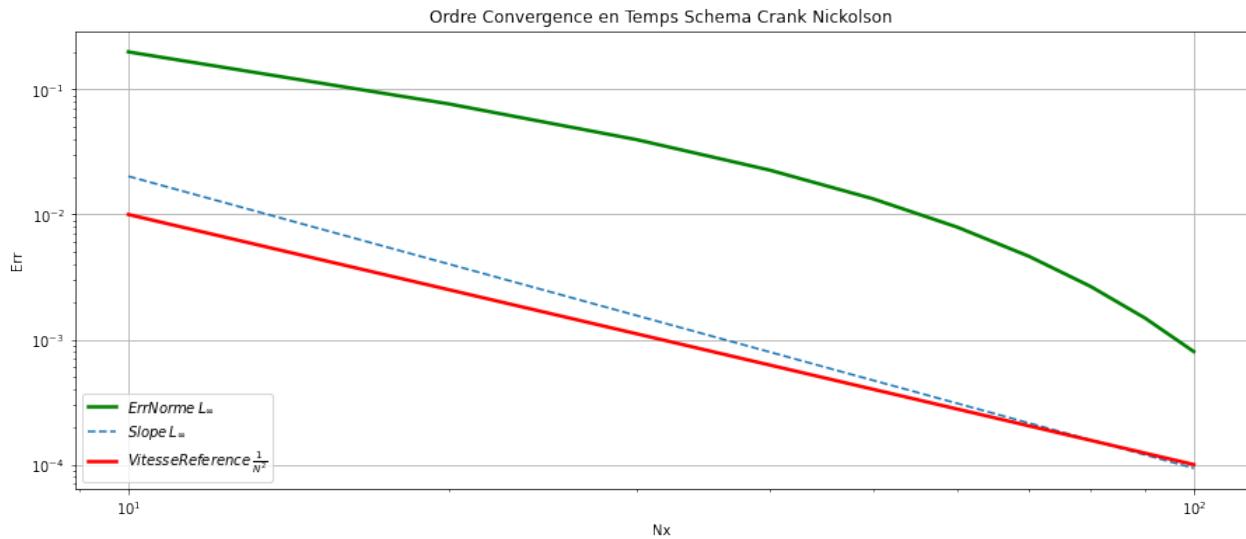


Figure 4.13: Convergence en temps pour le schéma de Crank-Nickolson

Dans le cadre d'un schéma de Crank-Nickolson, la convergence du schéma lorsque le pas d'espace tend vers 0 est à l'ordre 2. Lorsque que nous traçons l'erreur en norme infinie en loglog pour un pas d'espace très petit devant le pas de temps, alors nous pouvons valider empiriquement que notre schéma converge en la comparant à la vitesse de convergence de référence en $\frac{1}{N^2}$ en rouge. Ici la courbe "slope" nous permet de voir que la convergence n'est pas parfaite mais les tendances sont très proches.

En espace

Pour illustrer la vitesse de convergence en Espace, on fait varier le pas d'espace tout en fixant un pas de temps très faible.

Explicite

Nous pouvons ici tracer la convergence pour le schéma explicite :

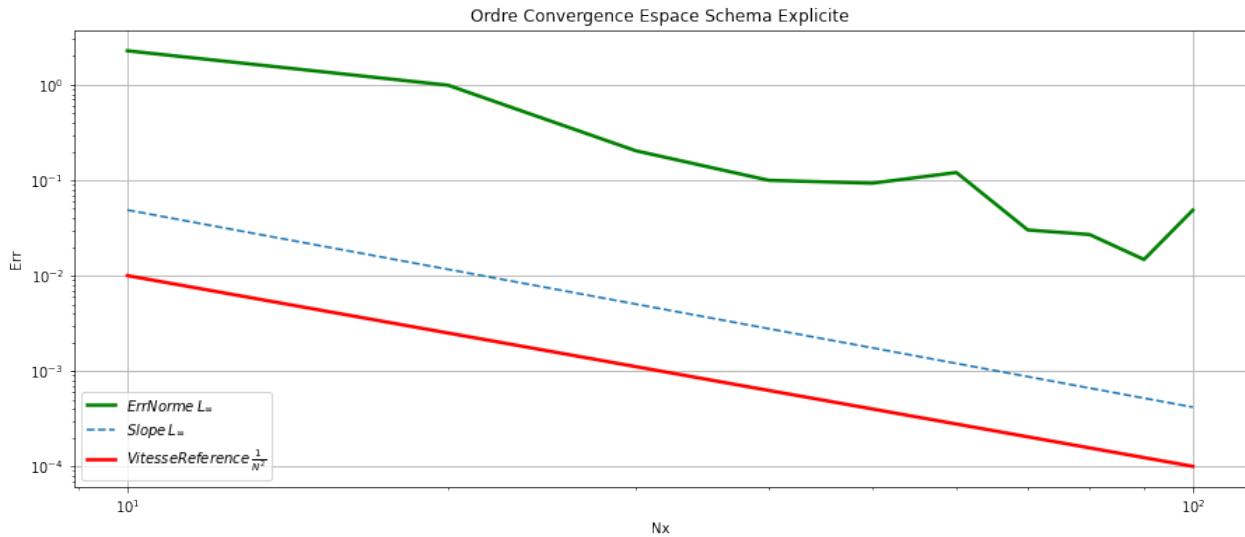


Figure 4.14: Convergence en espace pour le schéma explicite

Dans le cadre d'un schéma explicite, la convergence du schéma lorsque le pas de temps tend vers 0 est à l'ordre 2. Lorsque que nous traçons l'erreur en norme infinie en loglog pour un pas de temps très petit devant le pas d'espace, alors nous pouvons valider empiriquement que notre schéma converge en la comparant à la vitesse de convergence de référence. Ici la courbe "slope" nous permet de déterminer si les tendances entre la vitesse de référence et celle de notre schémas sont les mêmes. Il s'avère que oui.

Implicite

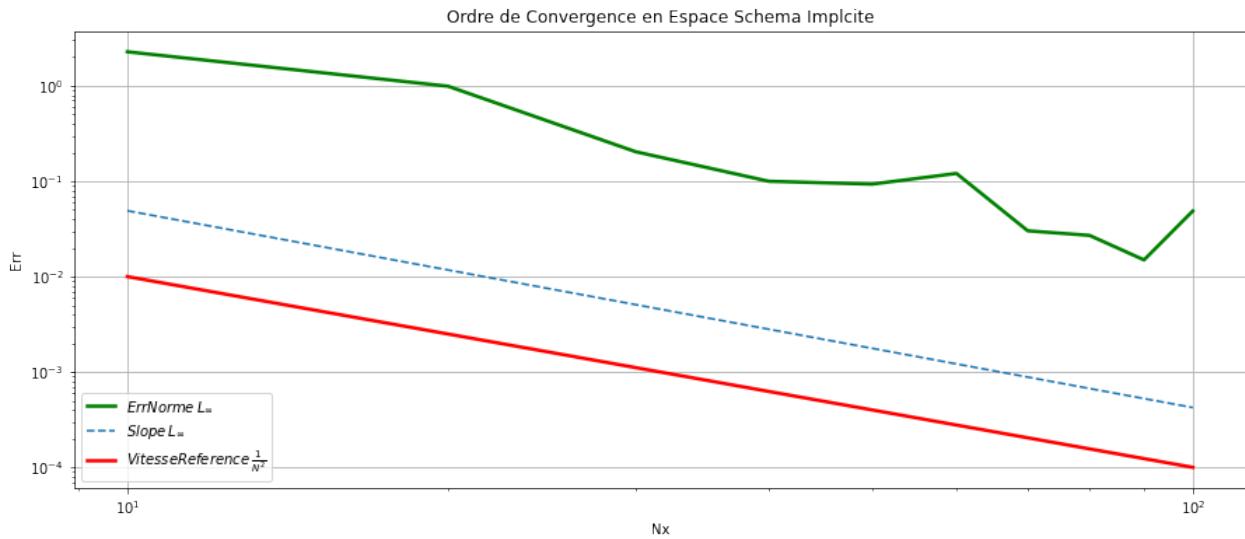


Figure 4.15: Convergence en espace pour le schéma implicite

Nous sommes exactement dans le mêmes cas de figure pour le schéma implicite.

Crank-Nickolson

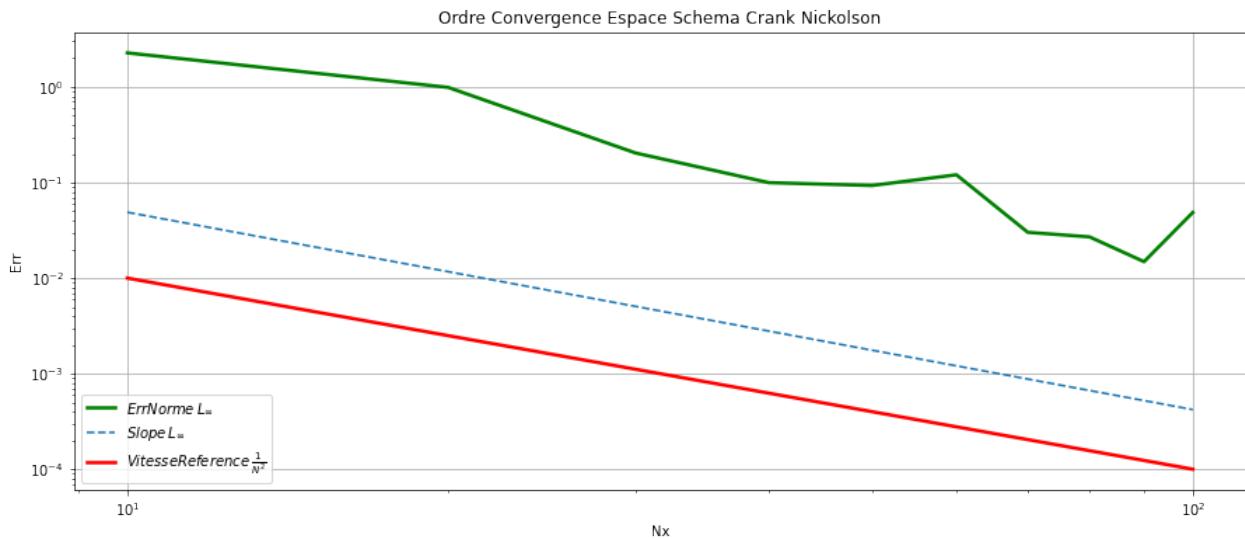


Figure 4.16: Convergence en espace pour le schéma de Crank-Nickolson

Nous sommes exactement dans le mêmes cas de figure pour le schéma de Crank-Nickolson.

4.5 Tracé de l'erreur ponctuelle en fonction de t et S

💡 : Nous réalisons nos graphiques pour $Ndt = 400$ et $Nx = 400$.

On notera que se soit pour le call ou le put les résultats obtenus sont les memes. Ils pourront être vérifiés dans le Notebook associé au rapport.

Call

On trace l'erreur ponctuelle ou l'erreur de troncature. Cette erreur correspond à la valeur absolue de l'écart entre la solution en un noeud approchée par la méthode de différence finie et la solution exacte en ce même point.

Explicite

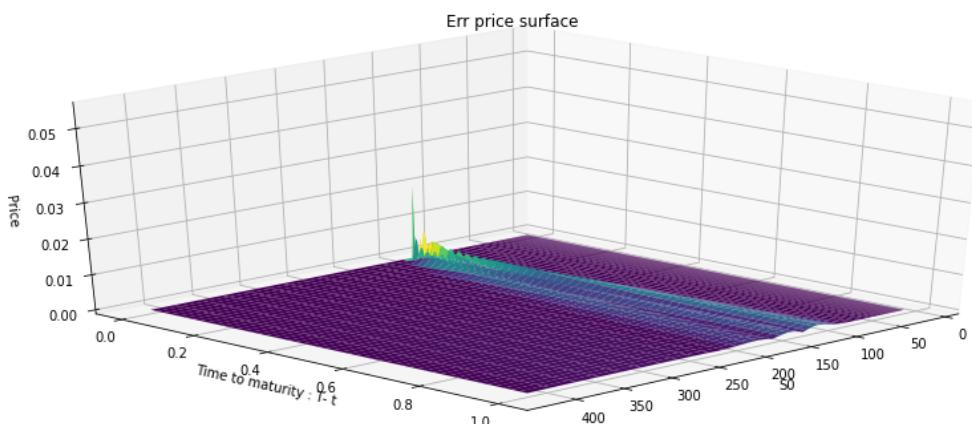


Figure 4.17: Erreur ponctuelle pour le schéma explicite dans le cas du call

Implicite

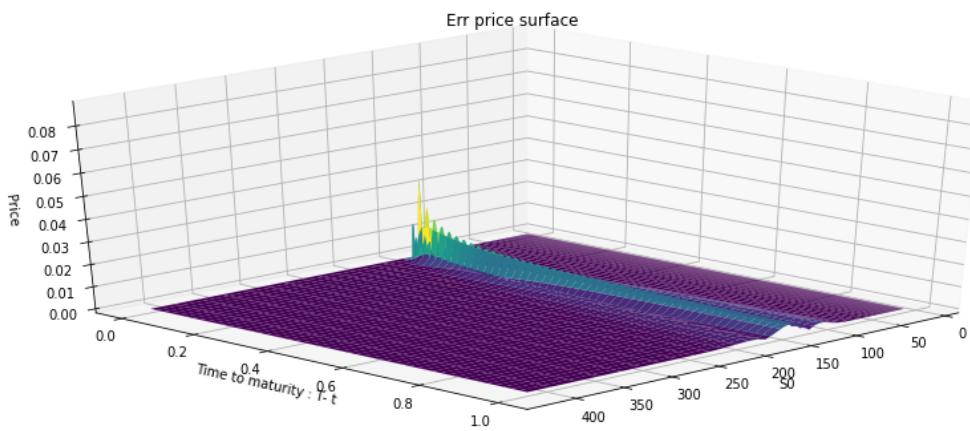


Figure 4.18: Erreur ponctuelle pour le schéma implicite dans le cas du call

Crank-Nickolson

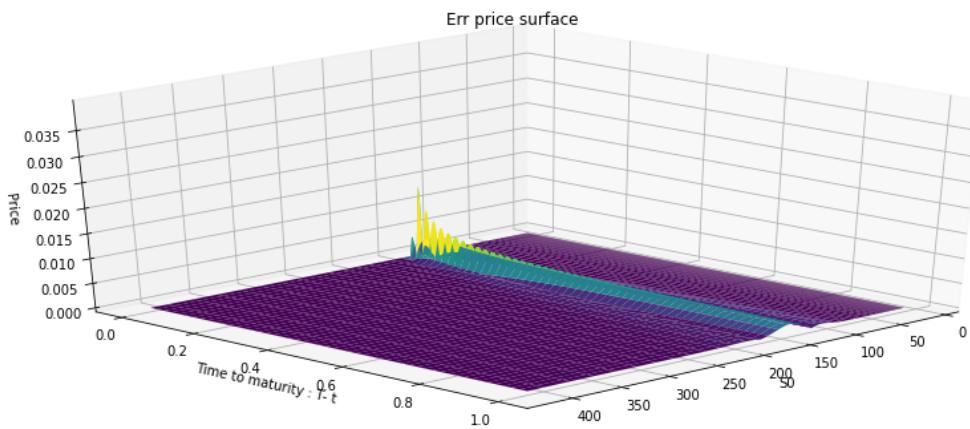


Figure 4.19: Erreur ponctuelle pour le de Crank-Nickolson dans le cas du call

Nous obtenons des résultats similaires pour les schémas explicites et implicites, ce qui est cohérent étant donné que les vitesses de convergence en temps et en espace sont les mêmes.

Pour le schéma de Crank Nickolson on note des erreurs de troncatures plus faible que pour les deux autres schémas car la convergence en temps est d'ordre 2 au lieu d'ordre 1.

Note : Par soucis de longueur du rapport, nous avons laissé le cas du put dans le Notebook, où nous obtenons exactement la même chose.

Partie 5

Discrétisation d'EDS et modèle de Heston

5.1 Discrétisation d'EDS

💡 : Une classe parente *EDS* est implémentée pour résoudre d'une manière générale une EDS de la forme :

$$dX_t = a(X_t, t)dt + b(X_t, t)dB_t$$

avec les schémas d'Euler et de Milstein, présentés ci-dessous.

Schémas

Euler-Maruyama

Selon ce schéma, l'EDS (que l'on prend homogène en temps ici) :

$$dX_t = a(X_t)dt + b(X_t)dX_t$$

peut être discrétisée de la manière suivante :

$$X_{i+1} = X_i + a(X_i)\Delta_t + b(X_i)\Delta B_i$$

avec $\Delta B_i = \sqrt{\Delta_t} Z_i$ et $Z_i \sim \mathcal{N}(0, 1)$ et Δ_t le pas de temps.

Milstein

Nous considérons également le schéma de Milstein :

$$X_{i+1} = X_i + a(X_i)\Delta_t + b(X_i)\Delta B_i + \frac{1}{2}b(X_i)b'(X_i)\Delta_t(Z_i^2 - 1)$$

Ordre d'erreurs

Il est intéressant de vérifier l'ordre de convergence des schémas de discrétisation.

Pour X_T la vrai valeur connue de la solution de l'EDS, on appelle erreur forte (ou trajectorielle) d'un schéma l'erreur suivante :

$$\mathbb{E}[|\hat{X}_T - X_T|]$$

Et erreur faible :

$$|\mathbb{E}[f(\hat{X}_T)] - \mathbb{E}[f(X_T)]|$$

On appelle ordre fort (faible) δ tel que l'erreur forte (faible) soit en $\mathcal{O}(\Delta_t^\delta)$.

Sous conditions de croissance et de régularité sur $a(X_t, t)$ et $b(X_t, t)$, le schéma d'Euler est d'ordre fort $\frac{1}{2}$ et d'ordre faible 1.

Sous les mêmes conditions, celui de Milstein est d'ordre fort 1 et d'ordre faible 1.

Application à l'EDS de Black Scholes

💡 : La classe `EDS_BS` héritant de la classe `EDS` est utilisée, cela permet donc d'étudier un cas de résolution d'EDS en définissant les fonction $a(X_t, t)$ et $b(X_t, t)$ et en utilisant les méthodes de la classe parente `EDS`. Un argument `calc_order` est optionnel afin de calculer la solution exacte de l'EDS ainsi que les erreurs, afin de déterminer les ordres.

Les paramètres sont comme ceux de l'estimateur de Monte-Carlo. Le nombre de pas est par défaut `Ndt= 1000` et le nombre de simulations `nb_sim_path= 1000`.

Nous rappelons l'EDS de Black-Scholes :

$$dS_t = rS_t dt + \sigma S_t dS_t$$

Nous avons donc $a(S_t, t) = rS_t$ et $b(S_t, t) = \sigma S_t$.

Dans notre cas, on rappelle que nous connaissons la solution de cette EDS. En effet, comme explicité en partie 1, en appliquant Itô à $Y_t = \log(S_t)$ nous avons que

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

Avec $Z \sim \mathcal{N}(0, 1)$.

Ordre fort et faible pour schéma d'Euler

De manière similaire à la vitesse de convergence de Monte-Carlo de la partie 1, nous allons tracer un graphique log-log afin de représenter la convergence du schéma d'Euler, dont la pente sera l'ordre correspondant.

Nous utilisons les mêmes sources aléatoires browniennes ainsi que les mêmes points pour le calcul de la solution exacte afin de pouvoir comparer les erreurs d'approximations des schémas. Nous fixons le nombre de simulations `nb_sim_path= 100`.

Voici ce que l'on obtient pour l'erreur forte, pour 100 valeurs de $N \in [100, 10000]$ $step=100$:

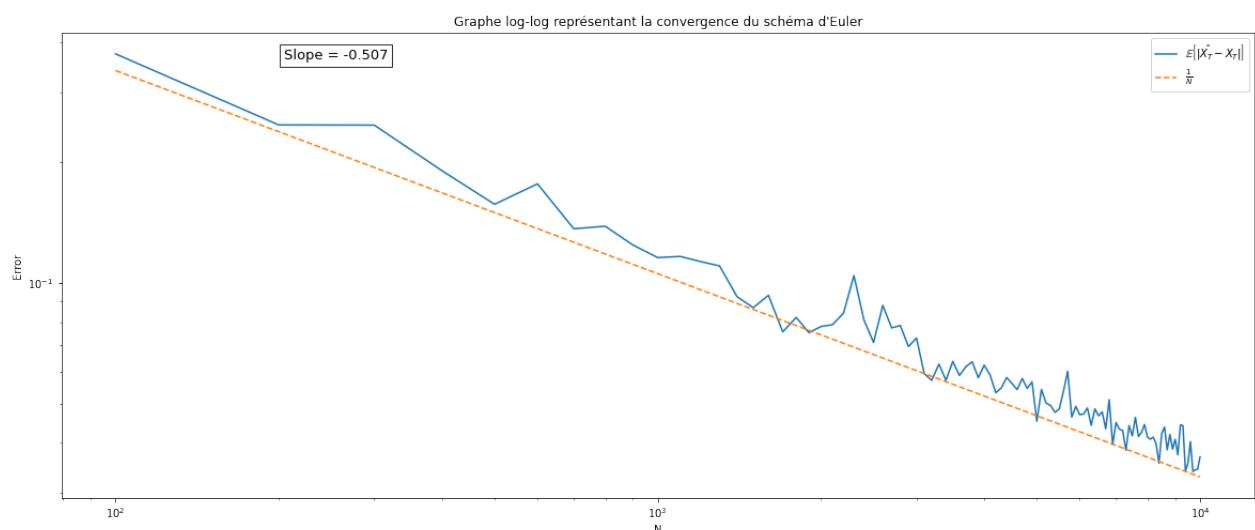


Figure 5.1: Convergence du schéma d'Euler pour l'ordre fort avec $N \in [100, 10000]$ $step=100$

Nous vérifions donc bien, avec approximations, que l'erreur forte est de $\frac{1}{2}$.

Voici le graphe pour l'erreur faible :

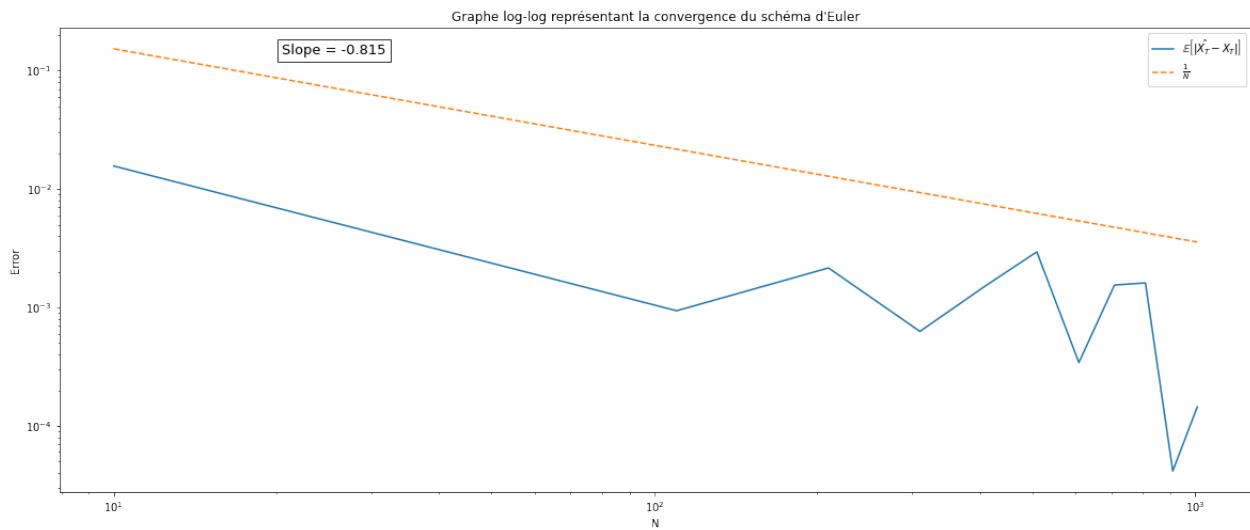


Figure 5.2: Convergence du schéma d'Euler pour l'ordre fort avec $N \in [10, 1000]$, $step=100$ pour 10000 simulations

Ordre fort et faible pour schéma de Milstein

Dans le cas du schéma de Milstein, Voici ce que l'on obtient pour l'erreur forte, toujours pour 100 valeurs de $N \in [100, 10000]$, $step=100$:

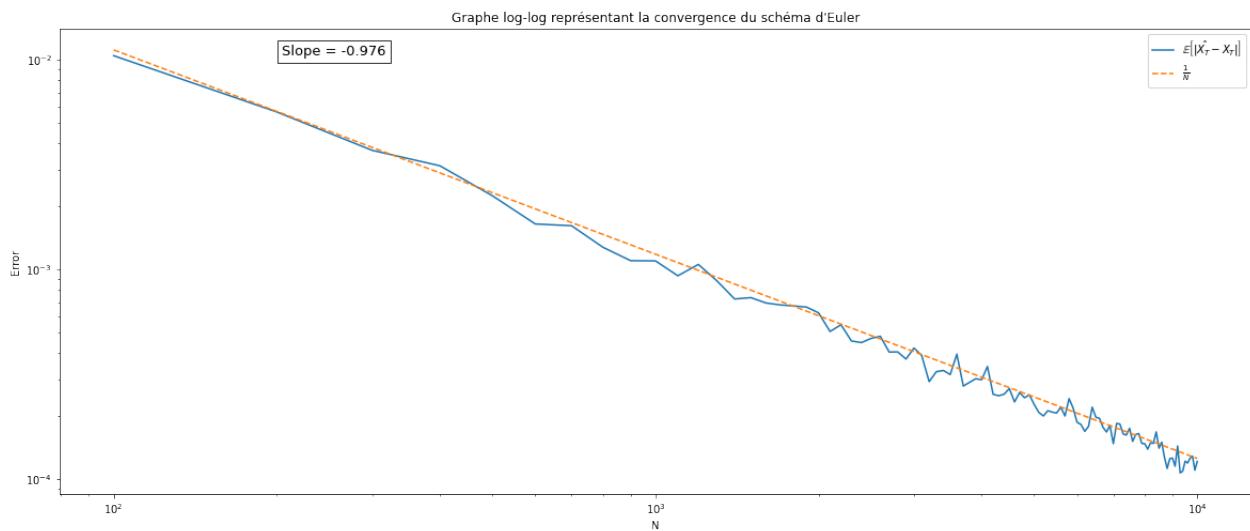


Figure 5.3: Convergence du schéma de Milstein pour l'ordre fort avec $N \in [100, 10000]$, $step=100$

Nou vérifions donc bien que l'erreur forte est de 1 à approximation près.

Et pour l'ordre faible :

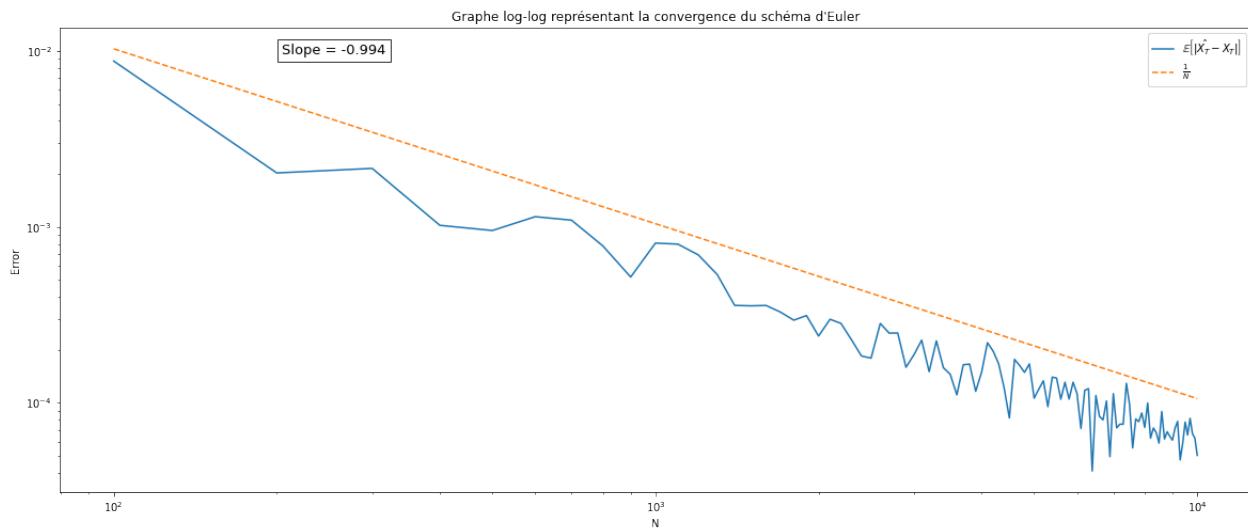


Figure 5.4: Convergence du schéma de Milstein pour l'ordre faible avec $N \in [100, 10000]_{step=100}$

Nous vérifions également que l'ordre faible est de 1.

5.2 Modèle de Heston

💡 : Une classe *Heston* héritant des classes *EDS* et *MonteCarloBS* a été créée pour cette partie, contenant une méthode *vol_implicit* pour le calcul des volatilités implicites par dichotomie (avec la fonction *bisect* de la librairie *scipy*).

Les calculs de S_t et de v_t via le schéma d'Euler se font directement à l'intérieur de la classe, il n'y a pas d'appels aux méthodes de la classe *EDS*, rendant la le code de la classe *Heston* plus lisible (et simple à implémenter).

De manière similaire à la partie 2, un argument *fix_seed* est optionnel afin de pouvoir réaliser des simulations sur les mêmes trajectoires lorsque nous changeons les paramètres.

Nous simulons les browniens corrélés avec la fonction *multivariate_normal* de *numpy* ici (nous aurions également pu les simuler par la méthode de la décomposition de cholesky présentée lors de l'évaluation de panier par Monte-Carlo en partie 1).

Les paramètres par défaut sont :

- $Ndt = 1000$
- $nb_sim_path = 1000$
- $S_0 = 100$,
- $K = 100$,
- $T = 1$,
- $r = 0.05$,
- $\eta = 0.5$,
- $\lambda = 5$,
- $\sqrt{\bar{v}} = \sqrt{v_0} = 0.2$,
- $\rho = -0.7$.

Nous prenons $\rho < 0$ car dans la réalité, la volatilité augmente lors de marchés baissiers (krachs par exemple), du à la peur des investisseurs. On note qu'ici $r > 0$ est pris constant mais peut également être pris comme étant stochastique.

Rappels

Le modèle de Heston est une spécificité des modèles à volatilité stochastique. Nous ne considérons désormais plus σ comme constant, mais prenons un processus stochastique adapté $(v_t)_{t \geq 0}$ (processus variance), B_t^1 et

B_t^2 deux (\mathcal{F}_t) -mouvements browniens univariés de corrélation $\rho \in]-1, 1[$, tels que :

$$\begin{aligned} dS_t &= \mu(t, S_t)dt + \sqrt{\nu_t} S_t dB_t^1 \\ dv_t &= \alpha(t, S_t, \nu_t)dt + \beta(t, S_t, \nu_t)dB_t^2 \end{aligned}$$

$\sqrt{\nu_t}$ est le processus de volatilité.

Nous pouvons également nous placer dans le cadre de la probabilité risque-neutre \mathbb{Q} , afin d'annuler le drift de la dynamique de S_t . A vrai dire il existe une infinité de mesures martingale équivalente, de manière générale le rôle de la calibration devient donc important lors des applications.

Le modèle de Heston est régi par les deux équations suivantes :

$$\begin{aligned} dS_t &= r S_t dt + \sqrt{\nu_t} S_t dB_t^1 \\ dv_t &= -\lambda(\nu_t - \bar{\nu})dt + \eta \sqrt{\nu_t} dB_t^2 \end{aligned}$$

Avec $\eta > 0$ la volatilité du processus variance, $\lambda > 0$ le coefficient de retour à la moyenne (force de rappel), et $\bar{\nu} > 0$ la variance moyenne long-terme.

A vrai dire, le processus $(\nu_t)_{t \geq 0}$ est le processus de diffusion racine-carré, connu sous le nom de processus CIR (proposé par Cox, Ingersoll et Ross afin de modéliser les taux d'intérêt ne pouvant être négatif, faisant suite au modèle de Vasicek).

Avec le changement de variable $X_t = \log(S_t)$, nous obtenons :

$$\begin{aligned} dS_t &= (r - \frac{\nu_t}{2})dt + \sqrt{\nu_t} dB_t^1 \\ dv_t &= -\lambda(\nu_t - \bar{\nu})dt + \eta \sqrt{\nu_t} dB_t^2 \end{aligned}$$

D'après la condition de Feller, ν_t admet des solutions positives \mathbb{Q} -p.s si $2\lambda\bar{\nu} > \eta^2$. En fait, cela n'est pas forcément vérifié d'un point de vue pratique. Nous avons donc imposé des conditions aux bord via l'argument optionnel `conds_bords` :

- "refl" pour réfléchissante (par défaut) : $\bar{\nu}_t = |\bar{\nu}_t|$
- "abs" pour absorbante : $\bar{\nu}_t = \bar{\nu}_t^+$.

Ce processus peut être discrétisé par le schéma d'Euler-Maruyama décrit précédemment. Afin de palier au fait que ν_t peut prendre des valeurs négatives, il est possible de faire un changement de variable $w_t = \log(\nu_t)$. Mais d'un point de vue numérique cela peut causer des erreurs d'approximations et produire des *Nan* quand Δ_t est trop proche de 0. Nous n'étudierons pas ce cas ici.

Le pricing d'une option via ce modèle peut être effectué par EDP (explicite et Fourier), ou par Monte-Carlo, que nous étudierons ici.

Enfin, on peut dire en limite que le modèle de Heston est efficace pour le moyen-long terme, mais peine plus pour le court terme (smile trop plat...), où ce sont donc plutôt les modèles à saut qui sont utilisés dans ce cas. De plus, d'autres modèles existent afin de modéliser les smiles de volatilité, tels que les modèles à volatilité locale par exemple.

Dynamique de l'actif S_t et de la volatilité ν_t

Note : Par défaut sauf `nb_sim_path=1`.

Voici donc un exemple des dynamiques du sous-jacent et de la volatilité stochastique associée :

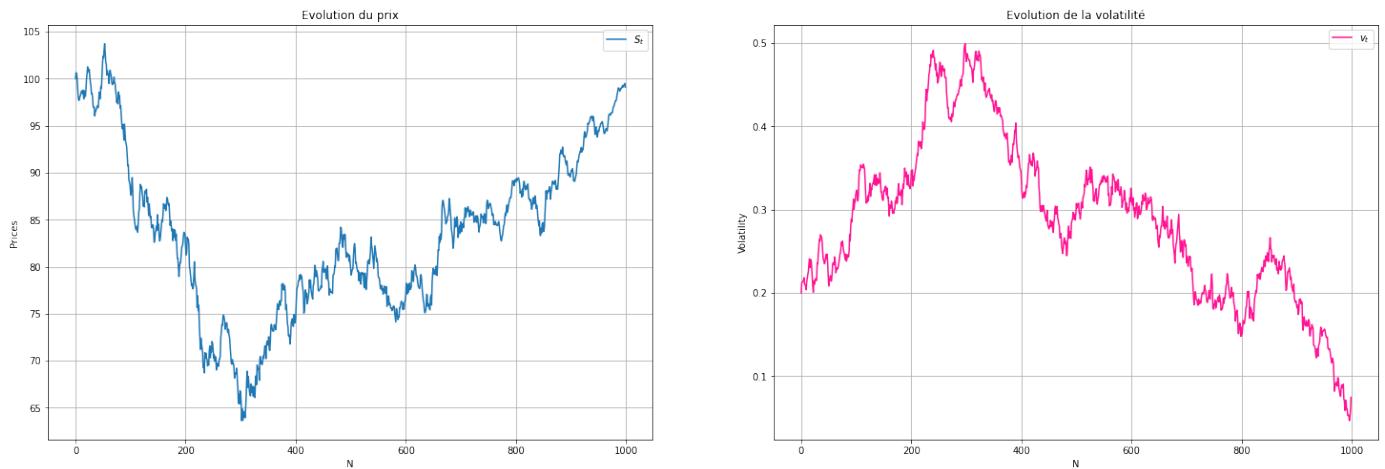


Figure 5.5: Evolution d'une paire de trajectoires S_t et v_t pour $Ndt = 1000$.

En remarque, nous voyons bien l'anti-corrélation entre la volatilité et l'actif, une montée de la volatilité est synonyme d'une baisse du sous-jacent, et vice versa.

5.3 Volatilité implicite et Smile/Smirt de volatilité

Volatilité implicite

La volatilité implicite est le paramètre σ qui découle du prix d'une option cotée sur le marché, que l'on devrait entrer dans la formule de Black&Scholes afin de retrouver le prix côté en question.

Le modèle de Black&Scholes stipule que pour une option sur un même sous-jacent, de même maturité, la volatilité implicite est constante quelque soit le strike. Dans la pratique, cela n'est pas vérifié. On parle alors de **skew de volatilité**.

Smile/Smirt de volatilité

Nous pouvons distinguer les smiles de volatilité des smirt de volatilité.

Pour un smile de volatilité, la volatilité prend une forme en U autour du strike, ce qui signifie que les options en dehors de la monnaie et dans la monnaie ont une bien plus grande volatilité implicite que celles à la monnaie. C'est généralement le cas pour les options sur actions et forex de courte maturité.

Dans le cas d'un smirt de volatilité, la volatilité implicite est plus grande pour les strikes inférieurs (pour les calls dans la monnaie et les puts en dehors de la monnaie). Cela est plus fréquent pour les options sur actions de plus grande maturité et les options d'index. Cela signifie donc que les calls dans la monnaie et les puts en dehors sont plus chers, cela peut par exemple s'expliquer par la peur des investisseurs vis à vis des krachs boursiers, ceux-ci achetant donc des puts pas forcément dans la monnaie en guise de protection.

Dans notre cas, avec le modèle de Heston, nous obtenons des **Smirt de volatilité**.

Voici un exemple de smirt de volatilité :

Paramètres : Par défaut sauf $\lambda = 1$, $nb_sim_path = 10000$ et $K \in [60, 170]$.

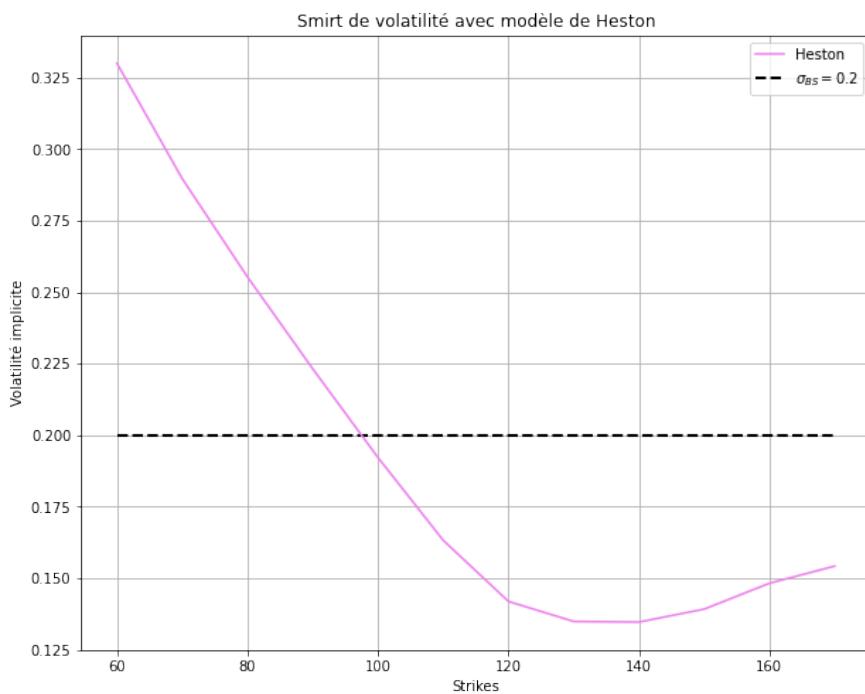


Figure 5.6: Smile (Smirt) de volatilité pour modèle de Heston pour $K \in [60, 170]$

Nous voyons donc bien le smile de volatilité obtenu avec le modèle de Heston, nous avons tracé jusqu'à $K = 170$ afin de voir la légère remontée de la volatilité implicite.

5.4 Smile en fonction des paramètres λ, η, ρ

Dans cette partie, nous allons faire varier les paramètres et en étudier l'impact sur le smile de volatilité, avec $K \in [60, 140]$ pour chaque cas.

Pour $\lambda \in \{1, 10, 100, 1000\}$

Paramètres : Les paramètres par défaut à part λ et K .

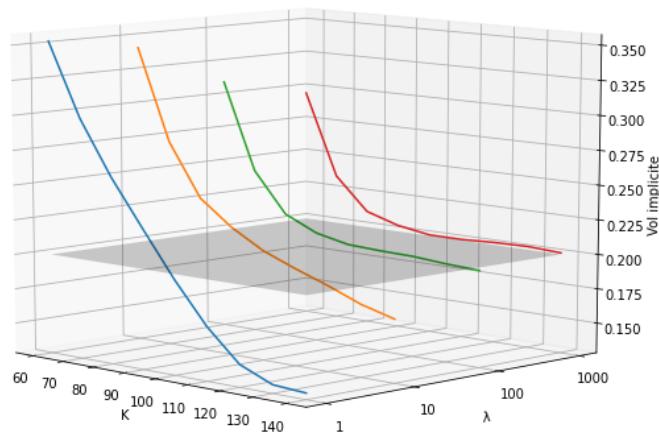


Figure 5.7: Evolution du smile en fonction de $\lambda \in \{1, 10, 100, 1000\}$

Nous voyons ici comme première courbe (bleue) le smile de volatilité tracé précédemment.

Nous voyons donc que plus λ augmente, plus le smile tend à s'aplatir et s'aligner sur le $\sigma_{BS} = 0.2$ pour les $K > 100$. Cela est du au fait que plus λ augmente, plus la force de rappel est importante, la volatilité sera donc "forcée" de revenir à sa moyenne long-terme $\bar{\nu} = 0.2^2 = 0.04$ (retour à la moyenne). Cela diminue donc la stochasticité de notre volatilité, ce qui cause l'aplatissement du smile.

Pour $\eta \in \{0, 0.3, 0.6, 0.9\}$

Paramètres : Les paramètres par défaut à part η et K .

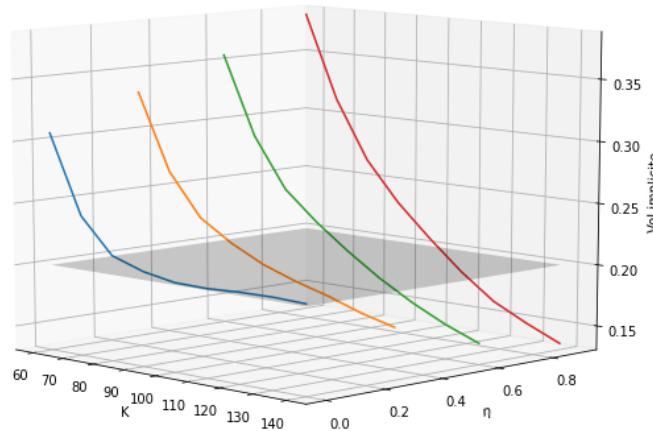


Figure 5.8: Evolution du smile en fonction de $\eta \in \{0, 0.3, 0.6, 0.9\}$

Nous voyons donc qu'à l'inverse, plus la volatilité de la volatilité η augmente, plus le smile s'intensifie. Nous pouvons dire que cela est du à l'augmentation de la stochasticité, en effet, le sous-jacent sera plus susceptible de subir de grande variation. Et nous savons que, pour une option d'achat, plus la volatilité augmente et plus le prix de l'option augmente. Nous avons donc une augmentation des prix de des options calls dans la monnaie lorsque la stochasticité augmente.

Cependant, ce graphique a été obtenu avec $\rho = -0.7$, donc avec S_t et v_t anti-corrélos. La forme du smile quand η augmente avec une corrélation positive n'aurait pas été la même, voyons cela en modifiant le paramètre ρ .

Pour $\rho \in \{-0.8, -0.2, 0.2, 0.8\}$

Paramètres : Les paramètres par défaut à part ρ et K .

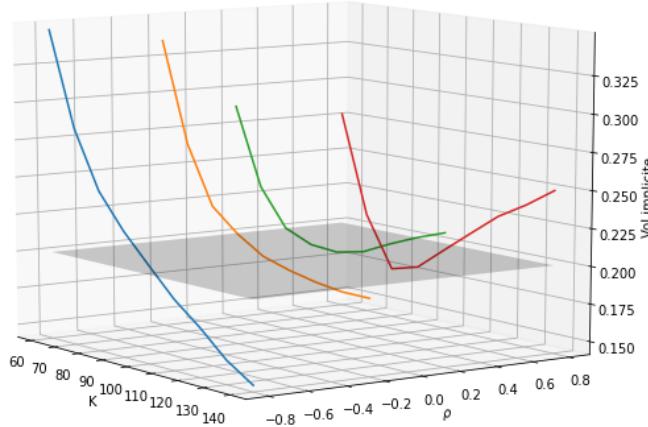


Figure 5.9: Evolution du smile en fonction de $\rho \in \{-0.8, -0.2, 0.2, 0.8\}$

Nous voyons ici que plus la volatilité est positivement corrélée au sous-jacent, plus la volatilité implicite des strikes supérieurs (en dehors de la monnaie pour les calls et dans la monnaie pour les puts) augmente, tandis que la volatilité implicite des strikes inférieurs diminue.

De plus, aux strikes aux alentours de la monnaie, la volatilité implacite est clairement en dessous du strike. On ne peut donc pas vraiment parler d'appatissement de smile au vu de cela.

5.5 Densité de transition du processus CIR

💡 : Une classe *CIR* est implémentée pour cette question, héritant de la classe *EDS*. Nous avons préféré ne pas la lier à la classe Heston afin de garder une instanciation par initialisation pour Heston, quitte à y redéfinir les attributs.

Les paramètres par défaut sont les mêmes que précédemment, mais sans K , S_0 , r et ρ , inutiles ici.

Principe

Dans le cadre du modèle de Heston, nous avons discrétisé le processus CIR. Mais nous savons que celui-ci a une densité de transition suivant une loi du χ^2 décentrée. Nous pouvons le simuler de manière exacte grâce à cette densité de transition, avec des lois du χ^2 , gaussiennes, et de Poisson.

Le principe de simulation est le suivant :

Pour $a \leftarrow \eta^2 \frac{(1-e^{-\lambda h})}{4h}$ et $b \leftarrow v_t \frac{e^{-\lambda h}}{a}$, on a :

- Si $d = \frac{4\lambda\bar{v}}{\eta^2} > 1$, alors $v_{t+h} = a \left(\chi^2_{d-1} + \left(\mathcal{N}(0, 1) + \sqrt{b} \right)^2 \right)$

- Sinon, $v_{t+h} = a \chi^2_{d+2\mathcal{P}\left(\frac{b}{2}\right)}$

Avec nos paramètres par défaut, on note que nous avons un d de

$$d = \frac{4\lambda\bar{v}}{\eta^2} = 3.2 > 1$$

Comparaison avec schéma d'Euler

Nous allons donc comparer avec le schéma d'Euler afin de se faire une idée de l'erreur de distribution. Notre but est de savoir quelles sont les valeurs de la distribution de la variance qui sont bien ou moins bien reproduites avec un schéma d'Euler pour le modèle de Heston.

Nous utiliserons l'argument `fix_seed=True` afin d'avoir les mêmes trajectoires lorsque nous faisons varier les paramètres.

Pour $N \in [10, 100, 1000, 10000]$

Paramètres : Par défaut sauf $N \in [10, 100, 1000, 10000]$ et `fix_seed=True`.

Nous simulons une trajectoire pour chaque nombre de pas de temps, voici les densités obtenues :

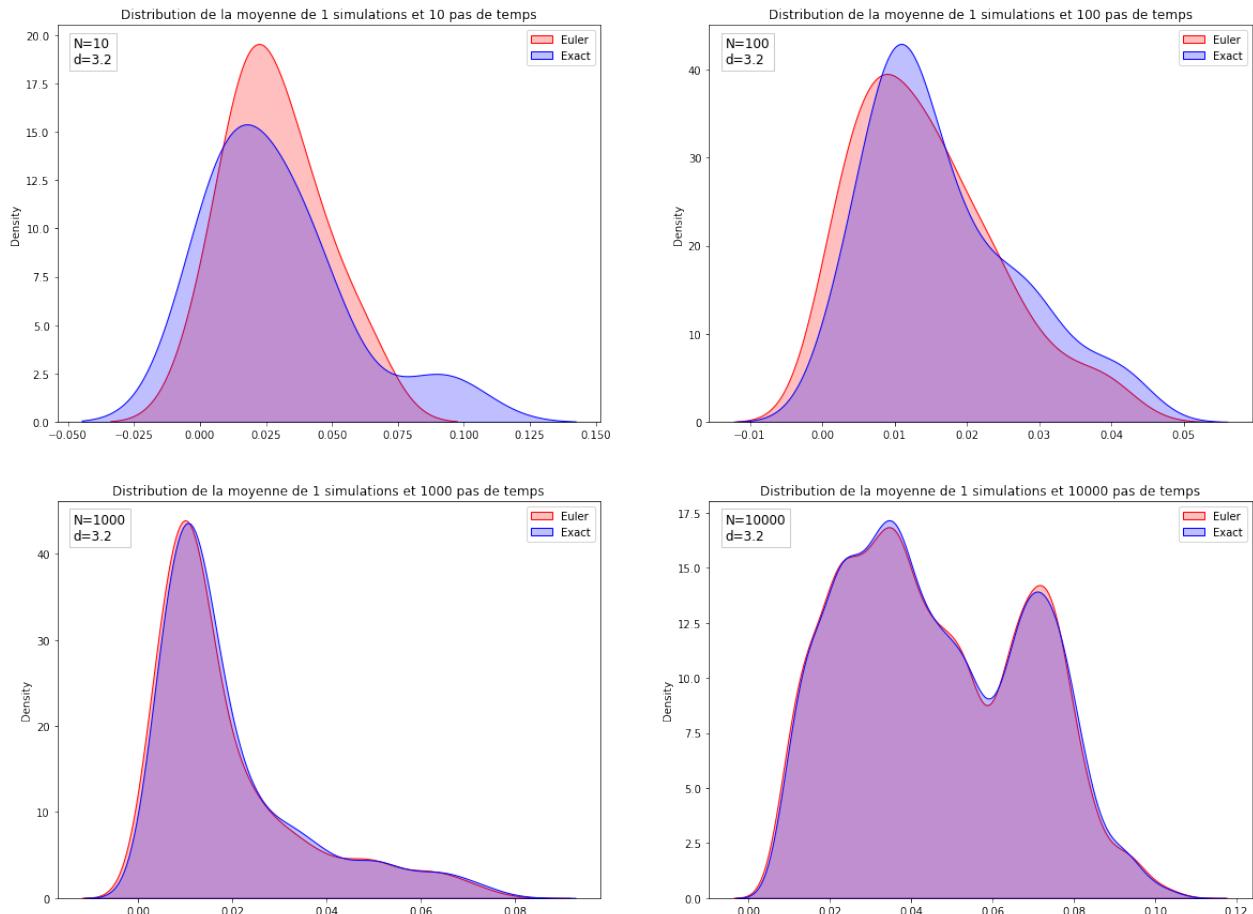


Figure 5.10: Densités de la trajectoire de v_t avec schéma d'Euler et méthode exacte pour $N \in [10, 100, 1000, 10000]$

Nous voyons que plus nous réduisons le pas de temps (plus N augmente), plus les deux densités deviennent indistinguables.

Pour $\lambda \in [1, 5, 100, 1000]$

Paramètres : Par défaut sauf $\lambda \in [1, 5, 100, 1000]$ et `fix_seed=True`.

Voici les densités obtenues en faisant varier le paramètre λ

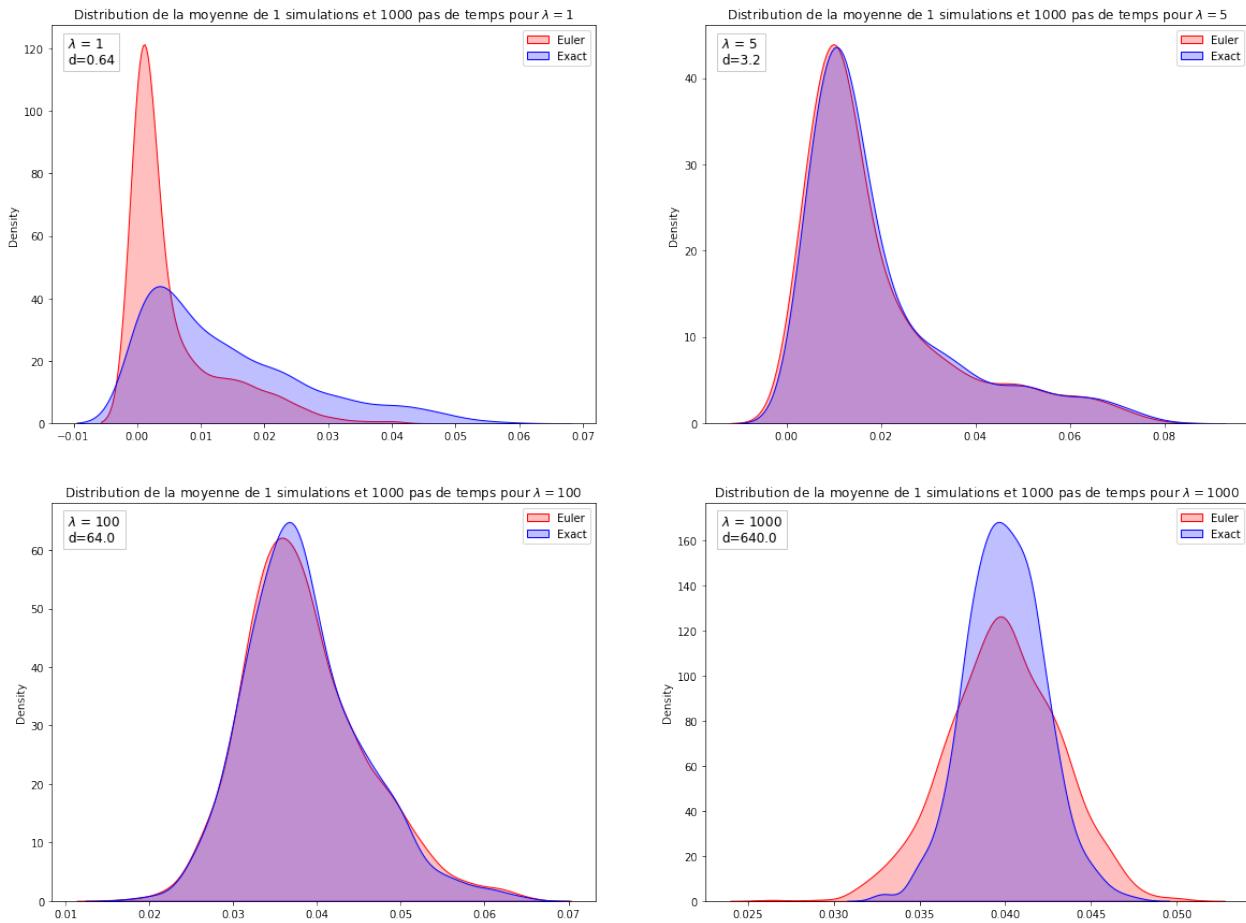


Figure 5.11

Nous voyons bien apparaître la loi du χ^2 décentrée pour la méthode exacte quand $\lambda = 1$ (et donc $d < 1$), il est important de noter que le caractère des courbes avec $\lambda = 1$ est extrêmement aléatoire, les courbes varient donc fortement et l'interprétation en est compliquée.

Nous voyons également que dès $\lambda = 5$, les deux densités se superposent, caractère qui tend à être perdu lorsque l'on augmente la force de rappel. Cependant, en l'augmentant, on distingue clairement que les deux densités tendent à se "normaliser".

Pour $\lambda = 1000$, la force de rappel est telle que la densité de la trajectoire par méthode exacte à l'allure d'une loi normale centrée en $v_0 = \bar{v} = 0.04$, et il est très intéressant de remarquer qu'avec la méthode exacte les valeurs de v_t sont bien moins dispersées que pour le schéma d'Euler. De plus, le cas de figure $\lambda = 1000$ est quasi-invariant, si nous enlevons l'argument `fix_seed=True` nous observons presque à chaque fois le même comportement.

Pour $\eta \in [0.2, 0.7, 1.2, 1.7]$

Paramètres : Par défaut sauf $\eta \in [0.2, 0.7, 1.2, 1.7]$ et `fix_seed=True`.

Nous simulons une trajectoire pour chaque η :

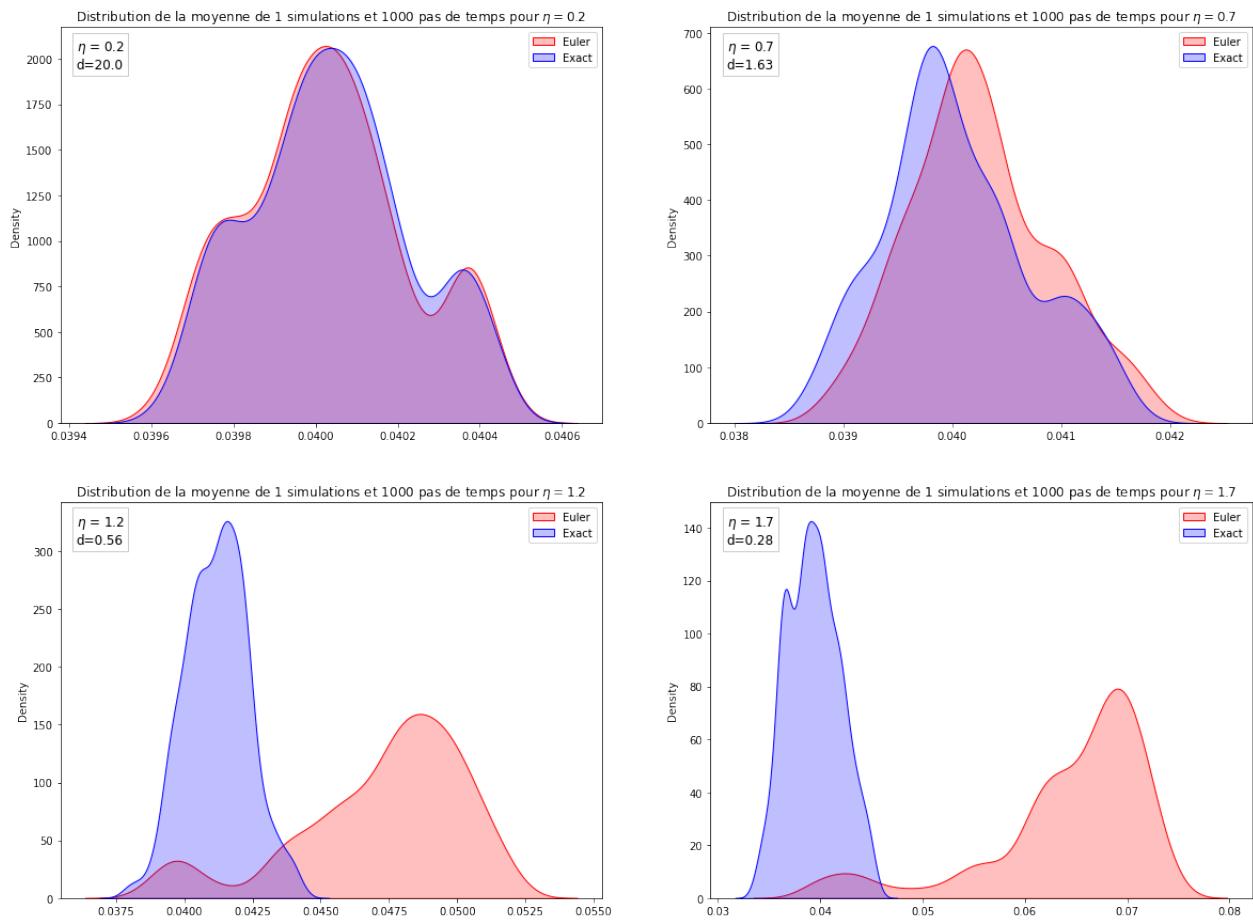


Figure 5.12: Densités de la trajectoire de v_t avec schéma d'Euler et méthode exacte

Nous voyons que plus η augmente et plus les deux densités se distinguent, car nous augmentons la stochasticité et donc les possibles variations. Cependant, nous voyons que cela reste beaucoup plus précis et localisé dans le cas de la méthode exacte.