Développement d'un Pricer d'Autocall Basé sur la Modélisation de Surfaces de Volatilité et l'Analyse de Données de Marché

Rapport de Stage
Année 2024

Kerlaouezo Erwan
Analyste Quantitatif - Parcours Recherche
A4 - IF

Qobalt Partners **ESILV**

Encadrant :
Luc Hazoume - Responsable de Stage

Lieu du Stage :
62 rue du Faubourg Saint-Antoine
75012, Paris

Contents

Résumé

Ce rapport présente le stage effectué chez Qobalt Partners, situé à Paris, durant la période du 2 avril 2024 au 9 août 2024.

L'objectif principal de ce stage était d'établir un pricer d'autocall compatible avec différents modèles. Durant cette période, j'ai acquis des compétences en programmation orientée objet, j'ai également développé mes connaissances en finance de marché et gestion des risques, et j'ai participé activement à un projet de site web où nous avons établi un outil en ligne de pricing de produits vanilles, exotiques et stratégies d'options.

1 Mission et Tâches Réalisées

L'objectif de ce stage était de développer un pricer d'autocall à partir de prix d'options vanilles récupérés sur Yahoo finance. Je vais expliquer en détail les étapes de A à Z que nous avons suivies pour établir ce pricer de qualité. Je vais également détailler chaque difficulté, chaque problème rencontré et expliquer la manière dont nous nous en sommes sortis.

Dans un premier temps, je vais résumer le processus de production que nous avons suivi pour atteindre notre objectif. Pour pricer un produit sur actions, il est indispensable de disposer de tous les paramètres du contrat, essentiels pour évaluer le produit, mais également de récupérer le maximum de données possible pour pouvoir valoriser ce produit de manière optimale. Les données essentielles, si nous avons un produit basé sur l'action Apple par exemple, sont les données de marché : du prix de l'action Apple et toutes les cotations d'options sur le marché(toutes les maturités, tous les prix bid et ask, tous les strikes et les volatilités implicites). Une fois ces données récupérées, il faut les trier et garder les plus pertinentes et les plus utiles pour notre objectif.

Une fois nos données récupérées et triées, un outil indispensable pour valoriser n'importe quelle option et n'importe quel produit est la surface de volatilité. Nous avons commencé par construire une surface de volatilité implicite en extrapolant les smiles de volatilité aux bornes puis en interpolant. Cette méthode peut s'avérer approximative, nous avons donc établi une deuxième méthode de construction de surface de volatilité implicite basée sur le modèle SVI (Stochastic Volatility Inspired), qui offre une surface plus lisse et précise.

L'objectif est qu'à partir de cette surface de volatilité implicite, nous pouvons avoir un pricer d'autocall:

- avec une volatilité constante
- à volatilité stochastique (Heston)
- à volatilité locale (Dupire)

Ce rapport montre que la construction de notre surface de volatilité implicite en première partie est indispensable pour utiliser nos différents modèles dans nos méthodes de pricing. Nous aurons ainsi un modèle de volatilité stochastique et un modèle de volatilité locale.

Une fois nos surfaces obtenues, nous nous intéresserons à la valorisation des produits. Nous utiliserons principalement la méthode Monte Carlo, dont nous détaillerons toutes les étapes suivies pour chaque modèle, avec des variantes pour améliorer son efficacité et sa rapidité. Nous aurons alors toute une pallette d'outils pour comparer, analyser et expliquer différents résultats.

En développant ce pricer, nous avons acquis une meilleure compréhension des mécanismes sous-jacents aux marchés financiers et des techniques avancées de modélisation financière. Cette expérience a renforcé nos compétences en programmation, en mathématiques financières et en analyse des données de marché, tout en nous offrant l'opportunité de travailler sur des problématiques concrètes et actuelles.

2 Récupération des données et tris

Nous récupérons des données de marchés pour les options Calls et Puts sur YahooFinance, cependant on remarque qu'il faut effectuer un tri car il est facile d'observer qu'il y a données manquantes et de mauvaises qualités. Il est également nécessaire de calculer les forwards, les taux ainsi que les dividendes pour chaque maturité, car ces données seront utilisées lors des calculs futurs (surface de volatilité, pricing d'options et des structurés).

Nous avons donc un ensemble de dimension n que l'on note $\mathcal{O} = \{O_i\}_{1 \leq i \leq n}$, où chaque option O_i est représentée tel que $\forall i \in [1, n]$, $O_i = (V_i, T_i, K_i, a_i, b_i, w_i)$, avec V_i le prix de l'option, T_i la maturité, K_i le strike, a_i le prix ask, b_i le prix bid et w_i le type de l'option.

2.1 Calculs des Forwards

Dans un premier temps nous allons calculer le forward pour chaque maturité. Pour cela, nous allons noter $\mathcal{O}' = \{(O_i, O_j)\}_{i \neq j}$ l'ensemble des couples d'options tel que pour chaque maturité, tous les calls et puts de même strike ayant un bid et un ask pour pouvoir calculer un forward à l'aide de la parité Call-Put soit gardée. Ainsi nous avons que:

$$\forall i \neq j,\ (O_i, O_j) \in \mathcal{O}' \Leftrightarrow T_i = T_j,\ K_i = K_j,\ w_i \neq w_j,\ (a_i, a_j, b_i, b_j) \in \mathbb{R}_*^4$$

Nous allons maintenant noter tout couple de $\mathcal{O}'$ comme étant $(C(t, S_t, T, K), P(t, S_t, T, K))$, où C et P représentent respectivement les options call et put. Avec ces objets on peut maintenant rapeller la parité Call-Put:

$$C(t, S_t, T, K) - P(t, S_t, T, K) = D(t, T)\,(F(t, T) - K) \tag{1}$$

Où $D(t, T) = e^{-r(T-t)}$ le facteur d'actualisation et $F(t, T) = S_t e^{(r-q)(T-t)}$ le forward.
Ce que l'on fait maintenant est que nous regroupons chaque couple de $\mathcal{O}'$ par maturité et pour chaque couple de même maturité on calcul un forward de la manière suivante à partir de (1):

$$F(t, T) = \frac{C(t, S_t, T, K) - P(t, S_t, T, K)}{D(t, T)} + K$$

Puis nous faisons une moyenne des forwards obtenus par maturité. Voici la courbe des forwards moyens calculés pour Apple obtenue avec cette méthode :

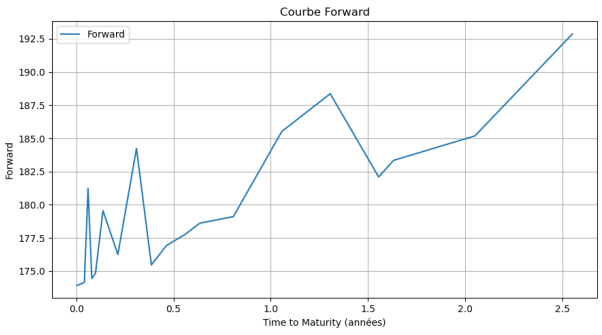

Figure 1: Courbe des Forwards moyen par maturité

Nous remarquons que cette courbe de forwards contient des valeurs qui ne sont pas convenables. Les deux principales raisons sont que dans cette méthode nous devons choisir arbitrairement un taux r, ce qui fausse légèrement les valeurs. Mais la raisons principales est que nous avons certains prix de marchés qui ne sont pas bons et qui entrainent un biais non négligeable dans notre moyenne de forwards.

Une façon d'observer ces mauvais prix est de tracer la parité Call-Put obtenue pour une maturité quelconque avec les données de marché et de la comparer à sa courbre théorique.

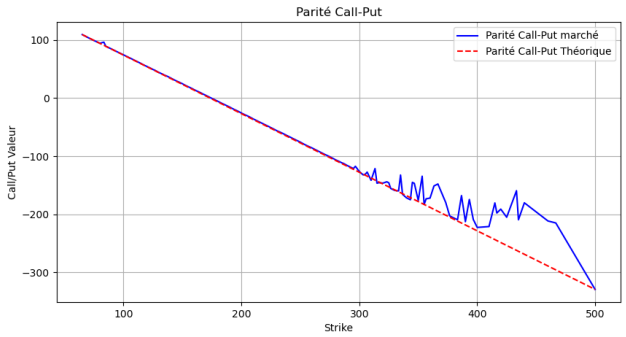

Figure 2: Parité Call-Put du marché comparée à la théorique

Avec cette courbe nous remarquons bien qu'il y a des prix qui biaisent notre moyenne de forwards, ce qui est normal car les options proches de la monnaie sont beaucoup plus liquides que les autres, donc nous avons des prix de meilleurs qualité proche de la monnaie. Il faut donc opter pour une autre méthode qui ne prend pas en compte ces mauvais prix. La méthode utilisée consiste à regarder la dernière valeur de parité positive et la première négative puis faire une interpolation linéaire pour trouver notre forward. Avec ces deux points, nous cherchons le K^* tel que $C(t, S_t, T, K^*) - P(t, S_t, T, K^*) = 0$ pour avoir $K^* = F(t, T)$.

2.1.1 Interpolation Linéaire

L'interpolation linéaire se fait comme suit : soit $K_1 < K_2$ les strikes correspondants respectivement à la dernière valeur de parité positive et à la première valeur de parité négative. On a alors les prix des options call et put associées, notés $y_1 = C_1 - P_1$ et $y_2 = C_2 - P_2$, avec $y_1 > 0 > y_2$. Nous avons alors la fonction d'interpolation suivante:

$$\forall x \in [K_1, +\infty[, \quad f(x) = \frac{y_2 - y_1}{K_2 - K_1}(x - y_1) + y_1$$

Nous pouvons maintenant calculer K^* par interpolation linéaire de la manière suivante :

$$f(K^*) = 0 \Leftrightarrow K^* = K_1 - \frac{y_1(K_2 - K_1)}{y_2 - y_1}$$

Cette méthode permet de mieux estimer le forward en éliminant les effets des prix des options moins liquides, et en se concentrant sur les prix des options les plus liquides, proches de la monnaie, ce qui devrait donner de meilleurs résultats.

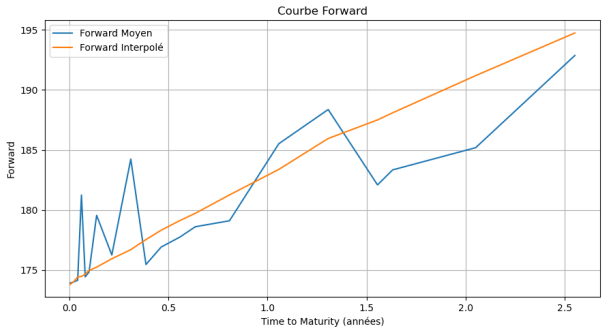

Figure 3: Courbe forwards interpolés comparé aux forwards moyens

On voit que cette courbe de forward est bien plus lisse que la première et semble indiquer que la méthode par interpolation linéaire semble être de bien meilleure qualité. En effet, l'interpolation linéaire se base uniquement sur deux options, mais ce sont deux options qui sont proches de la monnaie, sont donc très liquides et sont les meilleurs prix.

2.2 Calculs des Taux

Avec nos forwards nous pouvons maintenant calculer nos risk free rates pour chaque maturité. Une première approche consiste à utiliser la parité call-put pour tirer la relation suivante:

$$(1) \Leftrightarrow r = \frac{1}{T-t} \log\left(\frac{F(t,T)-K}{C(t,S_t,T,K)-P(t,S_t,T,K)}\right)$$

Ainsi pour chaque maturité nous calculons tous les rates associés à un couple Call-Put qui vérifient que $F(t,T)-K > C(t,S_t,T,K)-P(t,S_t,T,K)$, car sinon $r \leq 0$, puis nous faisons une moyenne de tous les taux gardés pour une maturité donnée. s Avec la figure 2, nous pouvons penser qu'une régression linéaire pour estimer r est une bonne approche.

2.2.1 Régression Linéaire

La régression linéaire est une méthode statistique utilisée pour modéliser la relation entre une variable dépendante y d'une ou plusieurs variables indépendantes X. Le modèle de régression linéaire simple peut être exprimé par l'équation suivante :

$$y = \beta_0 + \beta_1 x + \epsilon$$

où β_0 représente l'ordonnée à l'origine, β_1 est le coefficient de régression ou pente, et ϵ est l'erreur aléatoire.
Pour estimer les coefficients β_0 et β_1, on utilise généralement la méthode des moindres carrés. Cette méthode minimise la somme des carrés des résidus. Les solutions pour β_0 et β_1 sont données par les formules suivantes :

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

où $\bar{x}$ et $\bar{y}$ sont les moyennes des valeurs observées de x et y, respectivement.

Nous allons donc faire une régression linéaire sur les $C(t, S_t, T, K) - P(t, S_t, T, K)$ par rapport aux K pour avoir la relation suivante:

$$\beta_1 K + \beta_0 = e^{-r\tau} \left(F(t,T) - K\right)$$

Or pour $K = F(t,T)$ nous obtenons que $\beta_0 = -\beta_1 F(t,T)$ d'où l'on tire la relation:

$$r = -\frac{1}{\tau} \log(-\beta_1)$$

Ainsi à partir de cette relation, nous gardons uniquement les $\beta_1 > -1$ pour éviter des taux négatifs ou nuls. Nous décidons de faire cela pour chaque portion de données gardées (si nous avons 10 données nous calculons une slope pour les 10 puis nous calculons r, on fait pareil pour 9, pour 8, ect) puis nous faisons la moyenne de ces taux.

2.3 Calculs des Dividendes

Un autre paramètre très important à prendre en compte dans le pricing est les dividendes. Cependant, nous n'avons pas cette information directement dans nos données de marchés. Il faut donc trouver un moyen de pouvoir approximer ceux-ci pour chaque maturité et ainsi pouvoir gagner encore plus en précision dans nos calculs de prix. Par les deux sections d'avant, nous avons le forward ainsi que le taux pour chaque maturité. Donc à partir de la formule du forward $F_\tau = S_\tau e^{(r-q)\tau}$, on tire immédiatement que:

$$q = r - \frac{1}{\tau} \ln\left(\frac{F_\tau}{S_\tau}\right)$$

2.4 Convexité et Croissance/Décroissance

Nous décidons de garder seulement les calls et puts en dehors de la monnaie. Nous faisons également en sorte de garder la croissance/décroissance des prix pour les puts/calls par rapport au strike. Nous allons maintenant démontrer pourquoi il faut cette croissance/décroissance à l'aide des options digitales. On rapelle qu'une option digital de strike K et de maturité T a pour payoff:

$$P_{digitCall}(K,T) = 1 - P_{digitPut}(K,T) = 1_{\{S_T > K\}}$$

Cette option a comme valeur actualisée:

$$D_c(K,T) = \mathbb{E}(e^{-r(T-t)} 1_{\{S_T > K\}}) = e^{-r(T-t)} \mathbb{P}(S_T > K) \tag{2}$$

On a facilement que $D_c(K,T) = e^{-r(T-t)} - D_p(K,T)$ et que $D_c(K,T) \geq 0$. On sait également qu'une option digital peut être répliquée avec un call spread ou un put spread.

2.4.1 Réplication avec le Call/Put spread pour montrer la décroissance/croissance

On rapelle que le call spread se traduit par l'achat d'un call de strike K_1 et de la vente d'un strike K_2 avec $K_1 < K_2$.

$$\forall (K_1, K_2) \in \mathbb{R}_+^2, K_1 < K_2, \quad CallSpread(K_1, K_2, T) = C(K_1, T) - C(K_2, T)$$

On remarque qu'ici nous touchons à maturité l'écart entre les deux strikes et non 1 si $S_T \geq K_2$. En effet cela se montre facilement de la manière suivante:

$$\begin{aligned} S_T = K_2 &\Leftrightarrow CallSpread(K_1, K_2, T) = (K_2 - K_1)_+ - (K_2 - K_2)_+ \\ &\Leftrightarrow CallSpread(K_1, K_2, T) = K_2 - K_1 \end{aligned}$$

Il faut diviser notre $CallSpread$ par l'écart entre les deux strikes pour avoir 1 si $S_T > K_2$.

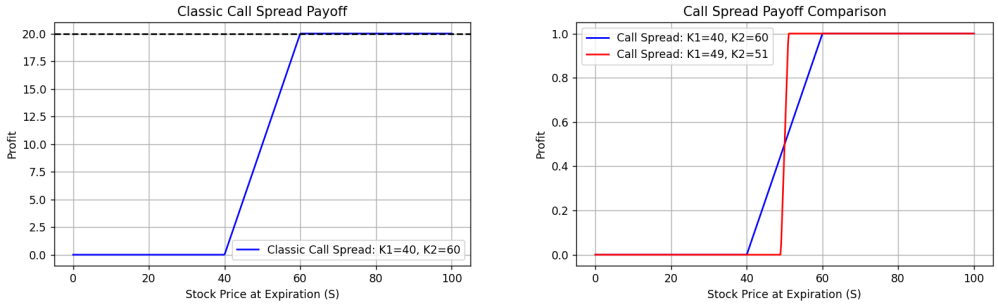

Figure 4: Valeur call spread vs comparaison call spread divisé par écart entre strikes

On voit bien qu'en rapprochant les strikes et en divisant le payoff par leur différence, nous tendons vers quelque chose qui tend vers une digitale. On peut alors introduire $\epsilon > 0$:

$$D_c(K,T) = \lim_{\epsilon \to 0} \left[\frac{1}{2\epsilon} \left(C(K-\epsilon, T) - C(K+\epsilon, T) \right) \right]$$

Or par développement de Taylor sur un schéma centré, on a:

$$\frac{\partial C}{\partial K} = \frac{C(K+\epsilon, T) - C(K-\epsilon, T)}{2\epsilon}$$

2.4.2 Résultats sur les conditions de croissance/décroissance

On a donc que pour les digitales call:

$$D_c(K,T) = \lim_{\epsilon \to 0} \left[\frac{1}{2\epsilon} \left(C(K-\epsilon, T) - C(K+\epsilon, T) \right) \right] = -\frac{\partial C}{\partial K} \tag{4}$$

Par un raisonnement similaire, en considérant une option digitale put notée D_p, dont le payoff est le complémentaire de celui de l'option digitale call, et en utilisant un put spread pour la répliquer, nous obtenons le résultat suivant :

$$D_p(K,T) = \lim_{\epsilon \to 0} \left[\frac{1}{2\epsilon} \left(P(K+\epsilon, T) - C(K-\epsilon, T) \right) \right] = \frac{\partial P}{\partial K} \tag{5}$$

Or par (2) on a que $D_c(K,T) \geq 0$ et de la même manière $D_p(K,T) \geq 0$. Nous arrivons donc enfin à nos conditions de coirssance/décroissance qui sont les suivantes:

- $\frac{\partial C}{\partial K} \leq 0$, donc prix du call décroissant par rapport au strike
- $\frac{\partial P}{\partial K} \geq 0$, donc prix du put croissant par rapport au strike

2.4.3 Convexité par rapport au strike

Finalement, après le tri des bons prix et les conditions de coirssance/décroissance établies, nous gardons uniquement les prix qui satisfaissent la convexité. Montrons que si nos prix d'options vanilles ne sont pas convexes, nous arrivons à une absurdité. Pour cela nous allons établir un raisonnement par l'absurde pour les calls. Tout d'abord rapellons la condition de convexité sur notre fonction pour une option vanille V:

$$\forall K_1 < K_2, \quad \frac{\partial V}{\partial K_1} \leq \frac{\partial V}{\partial K_2}$$

Raisonnement par l'absurde: Commencons notre raisonnement et assumons que l'inéquation du dessus est inversée. Nous démontrons cette inégalité pour les Calls mais le raisonnement est le même pour les Puts. Nous avons alors que $\forall K_1 < K_2$:

$$\begin{aligned} \frac{\partial C}{\partial K_1} \geq \frac{\partial C}{\partial K_2} &\Leftrightarrow \frac{\partial C}{\partial K_1} - \frac{\partial C}{\partial K_2} \geq 0 \\ &\Leftrightarrow -D_c(K_1, T) + D_c(K_2, T) \geq 0 \quad \text{par (4)} \end{aligned}$$

Or on a:

$$\begin{aligned} K_1 < K_2 &\Leftrightarrow \mathbb{P}(S_T > K_1) > \mathbb{P}(S_T > K_2) \\ &\Leftrightarrow D_c(K_1, T) > D_c(K_2, T) \quad \text{par (2)} \\ &\Leftrightarrow -D_c(K_1, T) + D_c(K_2, T) < 0 \end{aligned}$$

Ce qui contredit l'hypothèse. On arrive donc bien à une absurdité et nous avons donc bien que le prix des vanilles est convexe par rapport au strikes. ■

Finalement, la condition qu'iL faut respecter est que pour trois points $(K_1, p_1), (K, p), (K_2, p_2)$ avec $h_2 = K_2 - K$, $h_1 = K - k_1$ et $K_1 < K < K_2$ l'on vérifie l'inégalité suivante:

$$\frac{p_2 - p}{h_2} > \frac{p - p_1}{h_1}$$

2.5 Résultats Finaux

Ce ne sont que des approximations car il faut rapeller que nos forwards sont obtenus paar interpolation linéaire, nos taux par régression linéaire. On précise que le dernier spot enregistré était de 224.31.

Maturité	**Forwards**	**Taux**	**Dividendes**	**Spots**
0.110	224,8030	0,0546	0,0289	224,3099
0.131	225,1232	0,0482	0,0136	224,3099
0.157	225,7463	0,0552	0,0158	224,3099
0.229	226,6010	0,0688	0,0263	224,3099
0.318	227,4372	0,0456	0,0018	224,3099
0.403	228,2767	0,0640	0,0214	224,3099

Table 1: Valeurs de forwards, taux et dividendes en fonction de la maturité et du spot.

3 Construction de Surfaces de Volatilité Implicite

3.1 Extrapolation des Prix

Après notre tri des données de marché, nous remarquons que nous n'avons pas de cotations pour tous les strikes, il va donc nous manquer des informations pour pouvoir pricer notre autocall. Pour remédier à ce problème, il existe des méthodes d'extrapolation du smile de volatilité. La littérature propose deux fonctions qui sont présentées comme plus efficace que la méthode d'extrapolation basée sur un modèle SABR, [5].

Fonction pour les Calls
Le prix de notre option Call baisse si le son prix d'exercice augmente. En effet plus le prix d'exercice fixé est grand plus la probabilité de finir dans la monnaie est faible, donc le prix baisse. Pour pouvoir extrapoler correctement il nous faut une fonction f qui respecte les conditions suivantes :

- $f \in C^2, \frac{\partial f}{\partial K} \leq 0$, et $\frac{\partial^2 f}{\partial K^2} \geq 0$
- $f \xrightarrow[K \to +\infty]{} 0$

On prend donc la fonction de la forme suivante avec $\mu \in \mathbb{R}$ un paramètre que l'on choisit arbitrairement et $(a, b, c) \in \mathbb{R}^3$ des constantes, [1]:

$$\mathrm{C}(K) = K^{-\mu} \exp\left(a + \frac{b}{K} + \frac{c}{K^2}\right)$$

Fonction pour les Puts
Pour les Puts, si on augmente le strike alors la probabilité de finir dans la monnaie augmente donc le prix de notre option va lui aussi augmenter. Les conditions de notre fonction d'extrapolation f sont les suivantes :

- $f \in C^2, \frac{\partial f}{\partial K} \geq 0$, et $\frac{\partial^2 f}{\partial K^2} \geq 0$
- $f \xrightarrow[K \to 0]{} 0$ et $f(0) = 0$

La fonction que l'on va considérer est plus ou moins proche de celle pour les Calls :

$$\mathrm{P}(K) = K^{\mu} \exp\left(a + bK + cK^2\right)$$

Fonctions Générales
On peut généraliser les deux fonctions en fixant $\omega = 1$ pour les calls et $\omega = -1$ pour les puts de la manière suivante :

$$\mathrm{F}(K) = K^{-\omega\mu} \exp\left(a + bK^{-\omega} + cK^{-2\omega}\right)$$

Dans les deux cas nous avons quatres inconnues avec μ la paramètre d'aplatissement de la courbe choisi arbitrairement. Il suffit alors de calculer la dérivée première puis la dérivée seconde de chaque fonction pour avoir un système de trois équations à trois inconnues. Ces constantes sont calculées à partir des derniers prix et strikes d'options observés. Un

fait très important est que nous nous basons sur les trois derniers prix de marchés qui ont été gardés pour calculer nos constantes. Pour la résolution, nous nous basons sur les trois équations suivantes :

$$\mathrm{F}(K) = K^{-\omega\mu} \exp\left(a + bK^{-\omega} + cK^{-2\omega}\right) \tag{4.1}$$

$$\mathrm{F}'(K) = \mathrm{F}(K)\left(-\omega\mu K^{-1} - \omega K^{-\omega-1}b + 2\omega K^{-2\omega-1}c\right) \tag{4.2}$$

$$\begin{aligned}\mathrm{F}''(K) = \mathrm{F}(K)\big(&\mu(\mu+\omega)K^{-2} + (2\mu+1+\omega)K^{-\omega-2}b + (4\mu+4+2\omega)K^{-2\omega-2}c \\ &+K^{-2\omega-2}b^2 + 4K^{-3\omega-2}cb + 4K^{-4\omega-2}c^2\big)\end{aligned} \tag{4.3}$$

A partir de ces équations nous avons:

$$(4.2) \Leftrightarrow b(c) = -K^{\omega+1}\left(\omega\frac{F'(K)}{F(K)} + \mu K^{-1} - 2K^{-2\omega-1}c\right)$$

En prenant en compte que $\forall\omega \in \{-1, 1\}$, $\omega^2 = 1$ et $\frac{1}{\omega} = \omega$, on calcule nos paramètres en prenant en compte K_m, qui est le strike du dernier prix gardé, et en réinjectant $b(c)$ dans (4.3) on aboutit au résultat suivant:

$$c = K_m^{-2\omega}\left(\frac{\mu}{2} + \frac{\omega(\omega+1)}{2}\frac{F'(K_m)}{F(K_m)}K_m + \frac{\lambda}{2}K_m^2\right)$$

Puis par (4.2) et (4.3) on obtient les valeurs pour b et a en K_m:

$$(4.2) \Leftrightarrow b(c) = -\omega\frac{F'(K_m)}{F(K_m)}K_m^{1+\omega} - \mu K_m^{\omega} - 2K_m^{-\omega}c$$

$$(4.3) \Leftrightarrow a(b, c) = \ln(F(K_m)K^{\omega\mu}) - bK_m^{-\omega} - cK_m^{-2\omega}$$

Pour déterminer nos constantes, nous nous sommes basés sur les trois derniers prix de marché gardés notés p_i, p_{i-1} et p_{i-2}. En notant $p_i = F(K_m)$ et K_m le strike du dernier prix gardé on obtient les dérivées p_i' et p_i'' par développement de Taylor en notant $h_{i-1} = K_m - K_{i-1}$ et $h_{i-2} = K_m - K_{i-2}$:

$$p_i' = \frac{p_i - p_{i-1}}{h_{i-1}},$$

$$p_i'' = \frac{h_{i-1}p_{i-2} - h_{i-2}p_{i-1} - (h_{i-1} - h_{i-2})p_i}{\frac{1}{2}h_{i-1}h_{i-2}(h_{i-2} - h_{i-1})}$$

Un paramètre de notre résolution qui est capital et qui joue un rôle très important pour notre extrapolation est le λ qui est défini comme suit :

$$\lambda = \frac{1}{p_i}\left(p_i'' - \frac{p_i'^2}{p_i}\right) \tag{6}$$

Ce λ est très important car dans la pratique si il est négatif, nous arrivons a extrapoler en gardant les conditions de croissance/décroissance et de convexité pour n'importe quelle

valeur de μ, pour $\mu \in [1, \mu_{lim}]$ avec μ_{lim} un paramètre qui au dela de celui-ci nous donne des prix extrapolés qui ne marche pas.

En revanche si il est positif, les prix vont être extrapolés mais les conditions souhaitées ne vont plus être respectées. Cependant, dans la pratique il existe des cas où, lorsque $\lambda > 0$ et pour une valeur de μ bien précise, nous avons des prix extrapolés qui respectent les conditions souhaitées.

Dans cet exemple, les trois premiers points en partant de la gauche pour les Calls (à droite pour les Puts) correspondent aux derniers prix pour un strike que l'on ait et à partir de ces trois points nous traçons les points donnés par les fonctions d'extrapolations. On remarque que toutes les conditions souhaitées sont respectées.

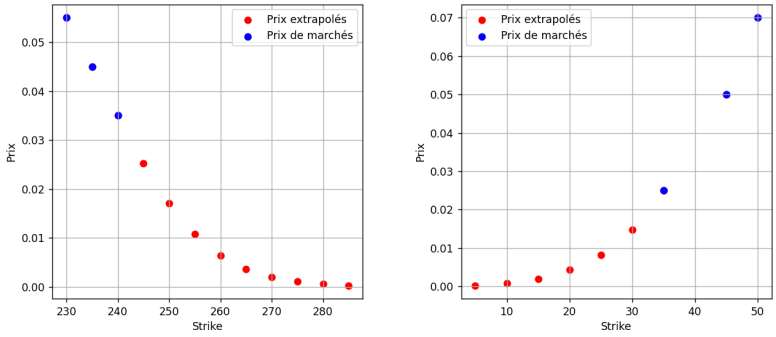

Figure 5: Extrapolation des prix de Calls et Puts par rapport au strike

Malgré la bonne forme de nos extrapolations de prix, voici le type de smile de volatilité que nous obtenons après passage des prix dans un calculateur de volatilité implicite utilisant la méthode de Brent :

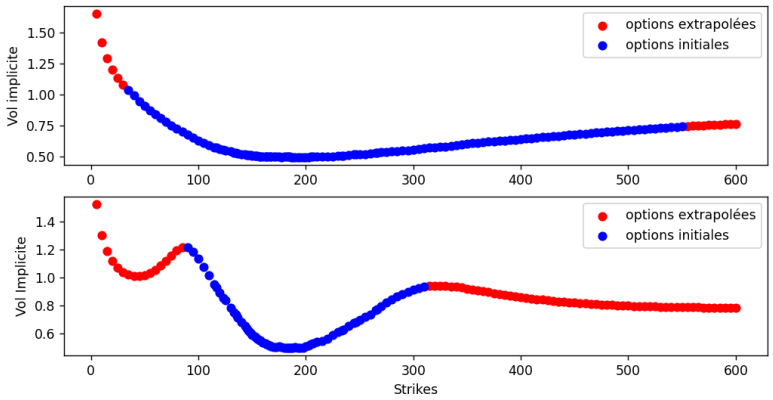

Figure 6: Smile extrapolé à partir des prix

Nous remarquons que pour le premier cas, où nous avons une maturité plus longue que la deuxième, le smile a une forme cohérente, en revanche pour le deuxième nous remarquons que pour les options extrapolées, la volatilité implicite est sous évaluée par rapport à ce qu'elle devrait être. Ceci indique donc que nos fonctions d'extrapolations sont, dans certains cas, de mauvaises qualités (sous pricing des options) bien que nos prix extrapolés semblent de bonnes qualité au vu de la figure 5.

3.2 Calculateur de volatilité implicite

3.2.1 Modèle de Black-Scholes

Maintenant que nous avons nos prix de marché intéressants et que nous les avons extrapolés, il nous faut un inverseur pour récupérer la volatilité implicite pour chaque options. Pour cela il faut introduire le modèle le plus connu, le modèle de Black-Scholes, qui permet de valoriser des options à l'aide de caractéristiques qui sont:

- T la maturité de l'option
- K le prix d'exercice de l'option
- S le sous-jacent
- σ la volatilité du sous-jacent
- r le taux sans risque
- q le taux de dividende instantané
- W_t un mouvement brownien

Une hypothèse fondamentale de ce modèle est que le sous-jacent est un processus stochastique tel que $\forall t \in [0, T]$:

$$dS_t = (r - q)S_t dt + \sigma S_t dW_t$$

Il est alors assez simple d'obtenir une formule fermée pour la valorisation d'une option vanille européenne en notant $F_t = S_t e^{(r-q)(T-t)}$ le forward de l'option:

$$V(t, S_t, \omega) = \omega e^{-r(T-t)} \left(F_t N(\omega d_+) - K N(\omega d_-)\right)$$

Où on a N la fonction de répartition de la loi normale centrée réduite et :

- Pour un Call Européen : $\omega = 1$
- Pour un Put Européen : $\omega = -1$

Avec :

$$d_\pm = \frac{\log(\frac{F_t}{K})}{\sigma\sqrt{T-t}} \pm \frac{1}{2}\sigma\sqrt{T-t}$$

Fonction utilisée:
Une fois ce modèle défini, nous allons l'utiliser pour pouvoir avoir nos volatilités implicites à partir des prix de marchés et des prix extrapolés. Pour ce faire, nous considérons la fonction suivante:

$$f(x; t, S_t, \omega) = \omega e^{-r(T-t)} \left(F_t N(\omega d_+(x)) - K N(\omega d_-(x))\right) - V_{market}$$

Avec :

$$d_\pm(x) = \frac{\log(\frac{F_t}{K})}{x\sqrt{T-t}} \pm \frac{1}{2}x\sqrt{T-t}, \quad \text{et} \quad V_{BS} \quad \text{le prix de marché.}$$

Une fois cette fonction créée, il faut choisir une méthode pour trouver le zéro de cette fonction et répéter l'opération sur touts les prix de marchés.

Il existe plusieurs méthodes comme la méthode de Newton-Raphson, cependant cette méthode dépend beaucoup de notre point initial, or pour des options avec maturité très courte et OTM, leur volatilité implicite a tendance à être plus élevée et le Newton-Raphson peut être amené à ne pas bien converger. Nous utiliserons la méthode de Brent, avec cette méthode nous pouvons fixer des bornes et ainsi éviter tous problèmes d'estimation de volatilité implicite. Nous savons pertinemment que ces méthodes d'inversion de prix ne sont pas les meilleures et ne sont sûrement pas les mieux adaptées mais elles offrent tout de même de très bonnes approximations.
La méthode de Brent est une méthode numérique utilisée pour trouver les racines d'une fonction. Elle combine les avantages de la méthode de la sécante, de la méthode de la bissection et de la méthode inverse par interpolation quadratique, afin de fournir une solution rapide et robuste. La méthode de Brent est particulièrement efficace pour les fonctions continues où une racine unique est cherchée dans un intervalle donné.[1]

Description de l'algorithme
Soit f une fonction continue définie sur un intervalle $[a, b]$, où $f(a)$ et $f(b)$ ont des signes opposés. La méthode de Brent procède comme suit :

Algorithm 1 Algorithme de Brent

Initialisation : Choisissez a et b tels que $f(a) \cdot f(b) < 0$. On note $f_a = f(a)$ et $f_b = f(b)$.
if $|f_a| < |f_b|$ **then**
  Échangez a et b.
end if
On choisit $c = a$, $f_c = f_a$.
repeat
  if $f_a \neq f_c$ et $f_b \neq f_c$ **then**
    Appliquez l'interpolation inverse quadratique pour calculer un nouveau point s.
  else
    Appliquez la méthode de la sécante pour calculer un nouveau point s.
  end if
  if le point s ne se trouve pas dans l'intervalle $\left(\frac{3a+b}{4}, b\right)$, ou si s n'améliore pas suffisamment la solution **then**
    Utilisez la méthode de la bissection pour calculer un nouveau point s.
  end if
  Évaluez $f(s)$. Si $f_s = 0$ ou si $|b - a| < \epsilon$, terminez.
  if $f_a \cdot f_s < 0$ **then**
    $b = s$.
  else
    $a = s$.
  end if
  Mettez à jour f_a et f_b.
  if $|f_a| < |f_b|$ **then**
    Échangez a et b.
  end if
until $|b - a| < \epsilon$

[1] Cette partie est reprise de Wikipédia.

Avantages de la méthode de Brent

- **Robustesse** : La méthode de Brent combine la robustesse de la méthode de la dichotomie avec la rapidité des méthodes de la sécante et d'interpolation inverse quadratique.
- **Convergence rapide** : La méthode de Brent assure une convergence superlinéaire, souvent plus rapide que les méthodes classiques comme la méthode de Newton.
- **Précision** : En utilisant plusieurs approches pour la recherche de racines, la méthode de Brent minimise les erreurs et améliore la précision de la solution.

3.3 Interpolation du Smile

Une fois nos prix inversés en volatilité implicite, il nous reste à interpoler nos volatilités implicites pour les options manquantes pour chaque maturité. La méthode que l'on a choisi est les splines cubiques. Une spline cubique $S_i(x)$ entre les points (x_i, y_i) et (x_{i+1}, y_{i+1}) est définie par un polynôme cubique :

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Principes de Continuité
Pour garantir la continuité des splines, nous imposons les conditions suivantes aux points internes x_i :
Continuité des valeurs :

$$S_i(x_i) = y_i \quad \text{et} \quad S_i(x_{i+1}) = y_{i+1}$$

Continuité des premières dérivées :

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$$

Continuité des deuxièmes dérivées :

$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$$

Formulation des Equations
Les dérivées première et seconde d'un polynôme cubique $S_i(x)$ sont :

$$S_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$$

$$S_i''(x) = 2c_i + 6d_i(x - x_i)$$

Conditions de continuités
Aux points x_{i+1}, nous avons en notant $h_i = x_{i+1} - x_i$. :

$$b_i + 2c_i h_i + 3d_i h_i^2 = b_{i+1}$$

$$2c_i + 6d_i h_i = 2c_{i+1}$$

Formulation du Système Tridiagonal
Pour prendre en considération les conditions sur les dérivées premières nous isolons d_i:

$$d_i = \frac{c_{i+1} - c_i}{3h_i}$$

Pour chaque intervalle $[x_i, x_{i+1}]$, nous avons les relations suivantes à partir des conditions des dérivées secondes:

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = \alpha_i$$

où :

$$\alpha_i = 3\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right)$$

Matrice Tridiagonale
La matrice tridiagonale peut être écrite comme suit :

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & 0 & \cdots & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 & \cdots & 0 & 0 \\ 0 & h_2 & 2(h_2+h_3) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & 0 & 0 & \cdots & h_{n-1} & 2(h_{n-1}+h_n) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_{n-1} \\ \alpha_n \end{bmatrix}$$

Résolution du Système Tridiagonal
Pour résoudre le système tridiagonal, nous utilisons les coefficients l, μ, et z définis comme suit :

1. Initialisation des conditions aux limites pour une spline naturelle:

$$l[0] = 1, \quad \mu[0] = 0, \quad z[0] = 0$$

2. Calcul des coefficients pour $i = 1$ à $n - 1$:

$$l[i] = 2(x_{i+1} - x_{i-1}) - h[i-1]\mu[i-1]$$

$$\mu[i] = \frac{h[i]}{l[i]}$$

$$z[i] = \frac{\alpha[i] - h[i-1]z[i-1]}{l[i]}$$

3. Finalisation des conditions aux limites pour n :

$$l[n] = 1, \quad z[n] = 0, \quad c[n] = 0$$

4. Substitution arrière pour obtenir les c_i :

$$c[j] = z[j] - \mu[j]c[j+1]$$

Calcul des coefficiants Restants
Une fois les c_i déterminés, nous calculons les coefficients a_i, b_i et d_i :

$$a[i] = y_i$$
$$b[i] = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}(2c_i + c_{i+1})$$
$$d[i] = \frac{c_{i+1} - c_i}{3h_i}$$

Interpolation
Pour interpoler une valeur x, nous utilisons le polynôme cubique correspondant :

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

L'interpolation par splines cubiques permet de garantir une continuité des dérivées premières et secondes, offrant ainsi une interpolation lisse et précise des volatilités implicites manquantes.

3.4 Résultat Final:

Finalement, après toutes ces étapes décrites et expliquées plus haut, nous pouvons obtenir une surface de volatilité pour n'importe quelle action dont nous avons des données d'options sur le marché. Cependant, de par le fait que nos extrapolations peuvent être sous-évaluées, nous pouvons retrouver ces défauts principalement pour des options à faible maturité et loin de la monnaie, plus particulièrement pour les puts. Voici la surface finale:

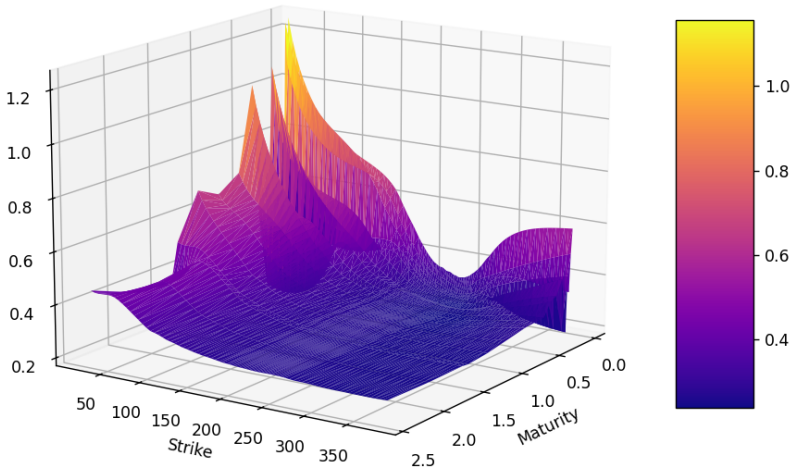

Figure 7: Surface de volatilité pour l'action Apple

3.5 Paramétrisation SVI

Les modèles SVI (Stochastic Volatility Inspired) sont utilisés pour représenter la surface de volatilité implicite. Ils offrent une paramétrisation à la fois parcimonieuse et flexible du smile de volatilité implicite. Deux paramétrisations courantes sont le SVI "Brut" (Raw SVI) et le SVI "Naturel" (Natural SVI).

3.5.1 Utilité des Modèles SVI

Les modèles SVI sont largement utilisés en finance quantitative pour ajuster les données de volatilité implicite des options et pour générer des surfaces de volatilité implicite plus lisses. Ces surfaces sont essentielles pour la valorisation et la gestion des risques des portefeuilles d'options. Les paramètres des modèles SVI permettent de capturer différentes caractéristiques du smile de volatilité, telles que le niveau, la pente et l'asymétrie.

On définit $\forall k \in \mathbb{R}$ et $t > 0$, $C_{BS}(k, \sigma_t^2)$ le prix de Black-Scholes d'une option d'achat européenne sur S avec un prix d'exercice $K = F_t e^k$, un time to maturity t et une volatilité $\sigma_t^2 > 0$. Nous désignerons la volatilité implicite de Black-Scholes par $\sigma_{BS}(k, t)$ et définirons la variance implicite totale par [9]:

$$w(k, t) = \sigma_{BS}^2(k, t)t$$

La variance implicite v sera également définie comme $v(k, t) = \sigma_{BS}^2(k, t) = \frac{w(k,t)}{t}$.

3.5.2 Paramétrisation Brute SVI

Pour un ensemble de paramètres donné $\chi_R = \{a, b, \rho, m, \sigma\}$, la paramétrisation SVI brute de la variance totale implicite se lit :

$$w(k; \chi_R) = a + b\left(\rho(k - m) + \sqrt{(k - m)^2 + \sigma^2}\right),$$

où $a \in \mathbb{R}$, $b \geq 0$, $|\rho| < 1$, $m \in \mathbb{R}$, $\sigma > 0$, et la condition que $a + b\sigma\sqrt{1 - \rho^2} \geq 0$, que l'on démontre plus bas et qui assure que $w(k; \chi_R) \geq 0$ pour tout $k \in \mathbb{R}$. Cette condition assure en effet que le minimum de la fonction $w(\cdot; \chi_R)$ est non-négatif. Notez également que la fonction $k \mapsto w(k; \chi_R)$ est (strictement) convexe sur toute la droite réelle.
où :

- $w(k, t)$ est la variance totale implicite au log-moneyness k et à la maturité t.
- a représente le niveau global de la variance.
- b est la pente de la courbe de variance.
- ρ contrôle l'asymétrie du smile.
- m est le décalage horizontal du smile.
- σ est la volatilité au prix d'exercice.

Il en découle immédiatement que les changements dans les paramètres ont les effets suivants :

- Augmenter a augmente le niveau général de la variance, une translation verticale du smile ;

- Augmenter b augmente les pentes des ailes des options put et call, resserrant le smile ;
- Augmenter ρ diminue (augmente) la pente de l'aile gauche (droite), une rotation antihoraire du smile ;
- Augmenter m translate le smile vers la droite ;
- Augmenter σ réduit la courbure du smile à la monnaie.

Preuve inégalité: Pour démontrer cette inégalité $a + b\sigma\sqrt{1-\rho^2} \geq 0$, il suffit de trouver le minimum de w car cette fonction est convexe. En effet:

$$\frac{dw}{dk}(k;\chi_R) = b\left(\rho + \frac{k-m}{\sqrt{(k-m)^2+\sigma^2}}\right)$$
$$\frac{d^2w}{dk^2}(k;\chi_R) = \frac{b\sigma^2}{((k-m)^2+\sigma^2)^{\frac{3}{2}}} \geq 0$$

Maintenant que nous avons que w est convexe, trouvons son minimum avec $b \neq 0$:

$$\begin{aligned}\frac{dw}{dk}(k;\chi_R) = 0 &\Leftrightarrow \frac{k-m}{(k-m)^2+\sigma^2} = \rho^2 \\ &\Leftrightarrow (k^2 - 2mk + m^2)(1-\rho^2) - \sigma^2\rho^2 = 0\end{aligned}$$

Ce qui nous donne en résolvant cette équation du second degré: $k_\pm = m \pm \frac{\sigma\rho}{\sqrt{1-\rho^2}}$
On réinjecte dans w:

$$\begin{aligned}w(k_\pm;\chi_R) &= a + b\left(\pm\frac{\sigma\rho^2}{\sqrt{1-\rho^2}} + \sqrt{\frac{\sigma^2\rho^2}{1-\rho^2}+\sigma^2}\right) \\ &= a + b\sigma\left(\pm\frac{\rho^2}{\sqrt{1-\rho^2}} + \sqrt{\frac{\rho^2}{1-\rho^2}+1}\right) \\ &= a + b\sigma\left(\pm\frac{\rho^2}{\sqrt{1-\rho^2}} + \frac{1}{\sqrt{1-\rho^2}}\right) \\ &= \begin{cases} a + b\sigma\sqrt{1-\rho^2}, \text{ si on a } k_- \\ a + b\sigma\left(\sqrt{1-\rho^2} + 2\frac{\rho^2}{\sqrt{1-\rho^2}}\right), \text{ si on a } k_+ \end{cases}\end{aligned}$$

Or il est évident que $\sqrt{1-\rho^2} \leq \sqrt{1-\rho^2} + 2\frac{\rho^2}{\sqrt{1-\rho^2}}$, donc comme il faut que $\forall k \in \mathbb{R}$, $w(k;\chi_R) \geq 0$, on a bien la condition souhaitée $a + b\sigma\sqrt{1-\rho^2} \geq 0$. ■

Nous excluons les cas triviaux $\rho = 1$ et $\rho = -1$, où le skew de volatilité est respectivement strictement positif et négatif. Nous excluons également le cas $\sigma = 0$ qui correspond à un smile linéaire (dérivée seconde de w nulle).

3.5.3 Paramétrisation Natural SVI

La paramétrisation SVI "Naturel" est une formulation alternative qui normalise les paramètres pour améliorer leur interprétation. Elle est définie comme suit :

Pour un ensemble de paramètres donné $\chi_N = \{\Delta, \mu, \rho, \omega, \zeta\}$, la paramétrisation SVI naturelle de la variance totale implicite se lit :

$$w(k; \chi_N) = \Delta + \frac{\omega^2}{2}\left(1 + \zeta\rho(k-\mu) + \sqrt{(\zeta(k-\mu)+\rho)^2 + (1-\rho^2)}\right),$$

où $\omega > 0$, $\Delta \in \mathbb{R}$, $\mu \in \mathbb{R}$, $|\rho| < 1$ et $\zeta > 0$.
Avec :

- $w(k; \chi_N)$ est la variance totale implicite au log-moneyness k.
- Δ représente le niveau global de la variance.
- ω est la pente de la courbe de variance.
- ρ contrôle l'asymétrie du smile.
- μ est le décalage horizontal du smile.
- ζ est la volatilité au prix d'exercice.

Nous avons la correspondance suivante entre les paramètres de la SVI brute et naturelle:

$$(a, b, \rho, m, \sigma) = \left(\Delta + \frac{\omega^2}{2}(1-\rho^2), \quad \frac{\omega\zeta}{2}, \quad \rho, \quad \mu - \frac{\rho}{\zeta}, \quad \frac{\sqrt{1-\rho^2}}{\zeta}\right),$$

et sa transformation inverse, entre la SVI naturelle et brute :

$$(\Delta, \mu, \rho, \omega, \zeta) = \left(a - \frac{\omega^2}{2}(1-\rho^2), \quad m + \frac{\rho\sigma}{\sqrt{1-\rho^2}}, \quad \rho, \quad \frac{2b\sigma}{\sqrt{1-\rho^2}}, \quad \frac{\sqrt{1-\rho^2}}{\sigma}\right).$$

3.5.4 Comparaison des Paramétrisations

Les deux paramétrisations visent à capturer les mêmes dynamiques sous-jacentes de la surface de volatilité implicite, mais elles le font avec des normalisations de paramètres différentes. Le choix entre SVI "Brut" et SVI "Naturel" dépend de l'application spécifique et de la préférence pour l'interprétabilité des paramètres.
La paramétrisation SVI "Brut" est souvent utilisée pour sa simplicité et sa capacité à ajuster rapidement les données du marché. La paramétrisation SVI "Naturel" est préférée lorsque l'on cherche à interpréter les paramètres de manière plus intuitive, notamment en ce qui concerne l'effet de la pente et de l'asymétrie sur le smile de volatilité.
Les modèles SVI, qu'ils soient "Bruts" ou "Naturels", fournissent des outils puissants pour modéliser la surface de volatilité implicite. Comprendre leurs formulations et leurs paramètres est crucial pour une application efficace en finance quantitative. Ces modèles permettent non seulement d'ajuster les données de marché de manière précise, mais aussi de générer des surfaces de volatilité qui facilitent la gestion des risques et la valorisation des portefeuilles d'options.

3.5.5 Calibration des SVI

Pour avoir une surface de volatiltié basée sur le modèle SVI, il faut calibrer les paramètres de notre fonction de variance implicite totale pour chaque maturité.

- **Bornes des paramètres**

Pour la calibration, il faut des bornes pour simplifier les calculs dans notre algorithme d'optimisation. Les bornes sont établies dans [11] et sont les suivantes:

$$\begin{cases} 0 \leq a \leq \max_i\{w_i\}, \\ 0 \leq b \leq \frac{4}{\tau(1+|\rho|)}, \\ -1 \leq \rho \leq 1, \\ \min_i\{x_i\} \leq m \leq \max_i\{x_i\}, \\ 0 < \sigma_{\min} \leq \sigma \leq 10. \end{cases}$$

Cette méthode est basée sur l'utilisation de la paramétrisation brute. Nous souhaitons trouver la meilleure adaptation de cette paramétrisation aux données de marché données et ainsi, en utilisant la méthode des moindres carrés, nous pouvons définir le problème d'optimisation comme suit :

$$\min_{a,b,\sigma,\rho,m} \sum_{i=1}^{n} \lambda_i \left(\omega_{\text{raw}}(x; a, b, \sigma, \rho, m) - \hat{\omega}_i\right)^2$$

où ω_{raw} est la formule de la paramétrisation brute en fonction du jeu de paramètres (a, b, σ, ρ, m), $\hat{\omega}_i$ représente les données de marché données définies en variance totale implicite selon la définition en début de chapitre et λ_i sont les poids pour définir la pertinence des différents points de données, ici nous choisissons le vega de l'option au carré pour calibrer au mieux les options proche de la monnaie. L'élévation au carré est uniquement pour des raisons d'homogénéité.

Ce que nous voulons réussir à faire, c'est bien calibrer les options les plus liquides, c'est à dire les options proches de la monnaie. Le vega est alors un bon choix car il est plus élevé pour les options dans la monnaie:

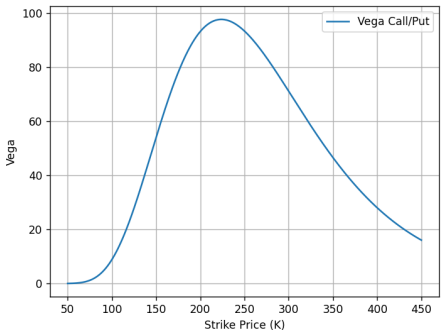

Figure 8: Evolution du vega pour les options vanilles avec forward à 220.

Ce problème d'optimisation non linéaire peut être assez lourd à résoudre directement en termes de temps de calcul. Pour simplifier le problème, nous appliquons la paramétrisation quasi-explicite proposée dans [10], [11]. Soit

$$y(x) = \frac{x - m}{\sigma},$$

sous ce changement de variable, la variance totale implicite dans la paramétrisation SVI brute se lit,

$$\omega_{\text{raw}}(x) = a + b\sigma\left(\rho y(x) + \sqrt{y(x)^2 + 1}\right) = \hat{a} + dy(x) + cz(x),$$

où

$$\hat{a} = a, \quad c = b\sigma, \quad d = \rho b\sigma, \quad z(x) = \sqrt{y(x)^2 + 1}.$$

Pour une paire fixée (σ, m), nous sommes confrontés à l'optimisation

$$\min_{(\hat{a},d,c)\in D} f_{x_i,\hat{\omega}_i}(\hat{a}, d, c),$$

où f_{x_i,w_i} est la fonction de coût quadratique,

$$f_{x_i,\hat{\omega}_i}(\hat{a}, d, c) = \sum_{i=1}^{n} \lambda_i(\hat{a} + dy(x_i) + cz(x_i) - \hat{\omega}_i)^2.$$

A noter que dans [11], ils oublient de considérer les poids λ_i dans la fonction f alors qu'ils apparaissent dans la formule pour β. Notez également que $f_{x_i,\hat{\omega}_i)}$ est une fonction lisse et convexe. Par conséquent, f a un minimum unique pour chaque paire fixée (σ, m) sur l'ensemble défini plus haut (les bornes des paramètres). De plus, le gradient de f est linéaire en $(\hat{a}, d, c)$, donc la solution peut être facilement calculée analytiquement en calculant l'inverse d'un système d'équations de 3×3. Cela permet de faire une optimisation extrêmement rapide des paramètres internes étant donnés les paramètres externes. Un algorithme d'optimisation intelligent devrait tirer parti de cela et impliquer une "optimisation externe" qui optimise (σ, m) et une "optimisation interne", qui à chaque étape de l'optimisation externe optimise $(\hat{a}, d, c)$.

Cela signifie qu'en choisissant une paire (σ, m), nous transformons notre problème non linéaire en un problème de régression linéaire multiple qui peut être résolu très rapidement par une seule opération matricielle.

Avec (σ, m) choisis, le problème d'optimisation est résolu par

$$\beta = (X^T \Lambda X)^{-1} X^T \Lambda Y,$$

où

$$\beta = \begin{pmatrix} \hat{a} \\ d \\ c \end{pmatrix}, \quad X = \begin{pmatrix} 1 & y(x_1) & z(x_1) \\ \vdots & \vdots & \vdots \\ 1 & y(x_n) & z(x_n) \end{pmatrix}, \quad Y = \begin{pmatrix} \hat{\omega}_1 \\ \vdots \\ \hat{\omega}_n \end{pmatrix}, \quad \Lambda = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

Preuve convexité, linéarité et égalité: Nous avons la fonction f qui s'écrit:

$$f_{x_i,\hat{\omega}_i}(\beta) = (X\beta - Y)^T \Lambda (X\beta - Y)$$

Et on a également pour le gradient et pour la Hessienne:

$$\begin{cases} \nabla f_{x_i,\hat{\omega}_i}(\beta) = 2X^T \Lambda (X\beta - Y) \\ \nabla^2 f_{x_i,\hat{\omega}_i}(\beta) = 2X^T \Lambda X \end{cases}$$

Pour démontrer la convexité de f, il faut que l'on démontre que $X^T \Lambda X$ est définie positive. Soit $X \in \mathbb{M}_{n,m}(\mathbb{R})$, considérons le vecteur $z \in \mathbb{R}^m$. Montrons que $z^T X^T \Lambda X z \geq 0$.

$$z^T X^T \Lambda X z = (Xz)^T \Lambda (Xz)$$

Alors en posant $y = Xz \in \mathbb{R}^n$ et en sachant que $\Lambda = Diag(\lambda_1, \cdots, \lambda_n) > 0$, on a:

$$z^T X^T \Lambda X z = y^T \Lambda y = \sum_{i=1}^{n} \lambda_i y_i^2 \geq 0$$

Ce qui montre que notre Hessienne est positive et nous avons donc bien la convexité de notre fonction.
De plus nous avons que notre gradient est linéaire car $\forall \beta$, il est de la forme $a\beta + b$ avec $a = 2X^T \Lambda X$ et $b = -2X^T \Lambda Y$. Il nous reste maintenant a trouver le minimum de f:

$$\begin{aligned} \nabla f_{x_i,\hat{\omega}_i}(\beta) = 0 &\Leftrightarrow 2X^T \Lambda (X\beta - Y) = 0 \\ &\Leftrightarrow \beta = (X^T \Lambda X)^{-1} X^T \Lambda Y \end{aligned}$$

Nous avons alors bien montré la convexité de f, la linéarité de son gradient ainsi que la solution de minimisation de f en trouvant β. ■

Λ est la matrice de poids définie comme une matrice diagonale avec chaque poids correspondant λ_i comme ses éléments diagonaux. Avec cette procédure, nous trouverons la meilleure adaptation possible pour la paramétrisation brute SVI.

Lors de la calibration de notre SVI, il est crucial de choisir un algorithme d'optimisation adapté pour obtenir des résultats précis et efficaces. Nous avons exploré deux méthodes principales : l'optimisation par PSO et la méthode de Nelder-Mead.

Optimisation par Essaims de particules (ou PSO pour Particle swarm optimization): L'algorithme de PSO est inspiré par le comportement collectif dans la nature des abeilles ou les bancs de poissons. Il s'agit d'une méthode d'optimisation heuristique qui ne nécessite pas de dérivées de la fonction à optimiser, ce qui est particulièrement utile pour les fonctions complexes ou non différentiables. Dans PSO, une population de particules explore l'espace de recherche. Chaque particule représente une solution candidate et se déplace dans l'espace en fonction de sa propre expérience et de celle des autres particules. Les mouvements des particules sont influencés par les meilleures positions trouvées par elles-mêmes et par l'ensemble de la population.

Cette optimisation est capable de trouver des solutions même dans des espaces complexes et ne nécessite pas de calcul de gradient. En revanche, cette méthode peut être amenée à être longue si on augmente la population initiale et le nombre de paramètres à calibrer.

Optimisation avec la méthode de Nelder-Mead La méthode de Nelder-Mead, également connue sous le nom de méthode du simplexe, est une technique d'optimisation basée sur des réflexions géométriques. Elle est particulièrement adaptée pour les problèmes d'optimisation sans contraintes et sans dérivées. L'algorithme utilise un simplexe, une figure géométrique de $n+1$ sommets en n dimensions, pour explorer l'espace de recherche. À chaque itération, le simplexe est modifié par des opérations de réflexion, d'expansion, de contraction et de réduction, en fonction des valeurs de la fonction à ses sommets.

Cette méthode converge très rapidement, 2h pour PSO contre 10 secondes pour Nelder-Mead. Elle est plus éfficace en temps et donne de bonnes solutions. En revanche, nous risquons, comme toute méthode de recherche de minimum local, de tomber sur un minimum qui n'est pas global.

Pour nos besoins de calibration de notre SVI, la méthode de Nelder-Mead s'est révélée être la plus efficace, permettant une calibration rapide et précise. Nous avons donc décidé de l'adopter pour nos processus de calibration, garantissant des résultats fiables tout en optimisant notre temps de calcul.

3.5.6 Surface Finale

Finalement, après notre choix de modèle, notre tri de données, et le choix de notre méthode de calibration, nous aboutissons à la surface de volatilité implicite suivante:

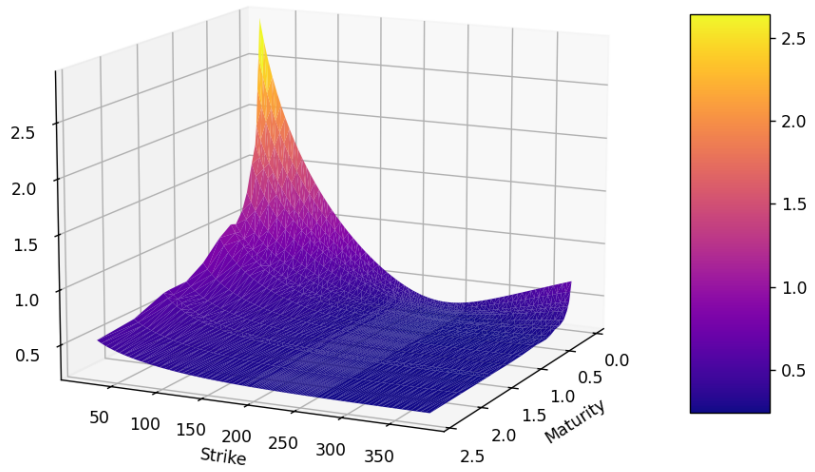

Figure 9: Surface de volatilité implicite pour Apple avec SVI.

3.6 Comparaison des Surfaces de Volatilité implicite

3.6.1 Comparaison des Smiles

Pour se donner un premier avis pour départager les deux surfaces, nous pouvons comparer différents smiles pour se faire une idée de comment se comporte nos smiles. On rappel que l'on calibre notre SVI uniquement sur les données de marché.

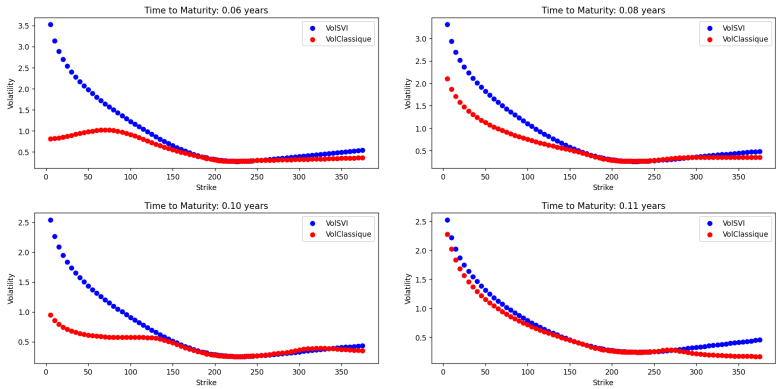

Figure 10: Comparaison des smiles SVI et classique sur différentes maturité.

On retrouve bien nos soucis d'extrapolation de smile (courbe rouge) et on a aussi que la calibration du modèle SVI est un succès car les options proches de la monnaie (forward environ à 220) sont très bien calibrées et on peut penser que le modèle SVI capte mieux les options loin de la monnaie.

3.6.2 Comparaison des surfaces

Nous pouvons comparer les deux surfaces de volatilité implicite obtenue en affichant une surface avec la différence entre les valeurs de la surface SVI et celles de la surface qui se basent sur nos fonctions d'extrapolations et d'interpolations.

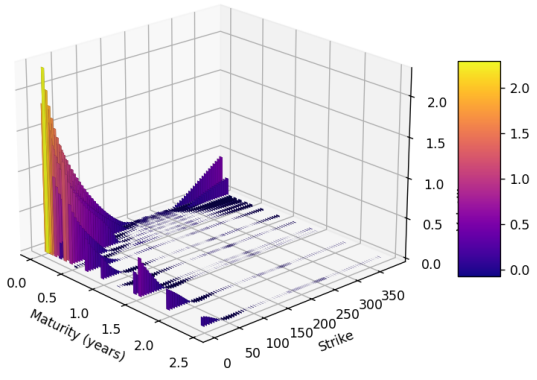

Figure 11: Surface de différence entre surface SVI et surface classique.

Nous pouvons remarquer que les options avec le plus d'incertitude concernant la valeur de leur volatilité implicite sont celles très OTM et à maturité faible. A ne pas oublier que notre surface de volatilité implicite classique extrapole très mal pour certaines maturités, ce qui se voit directement sur cette surface de différence.

4 Méthodes et modèles pour le pricing

4.1 Méthode Monte-Carlo

4.1.1 La Méthode

La méthode Monte-Carlo est une méthode couramment utilisée pour le pricing d'options dont il est difficile ou impossible d'obtenir une formule fermée de la valeur de ces mêmes options pour tout instant t. Cette méthode se base sur la loi forte des grands nombres qui nous dis que si on a un ensemble de variables $(X_i)_{1\leq i\leq n}$ indépendantes et identiquement distribuées alors:

$$\frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{p.s} \mathbb{E}(X_1)$$

Cette méthode est applicable car la valeur d'une option correspond simplement à l'espérance du payoff actualisé. Le principe de cette méthode consiste donc à simuler un certain nombre de trajectoires pour l'actif sous-jacent, à calculer le payoff pour chaque trajectoire, et, en faisant la moyenne de ces payoffs, on s'attend à ce que, selon la loi forte des grands nombres, cette moyenne converge vers l'espérance du payoff, et donc vers la valeur de l'option.

4.1.2 Précision de la méthode

Pour évaluer la précision de l'estimation de la méthode Monte Carlo, on utilise souvent un intervalle de confiance. Si $\hat{P}$ est l'estimation du prix de l'option basée sur n simulations, l'intervalle de confiance de niveau $1-\alpha$ pour le vrai prix P est donné par :

$$P \in \left[\hat{P} - \eta\frac{\sigma}{\sqrt{n}}, \hat{P} + \eta\frac{\sigma}{\sqrt{n}}\right]$$

où σ est l'écart-type des payoffs simulés, η est le quantile de la loi normale centrée réduite de niveau $1-\frac{\alpha}{2}$. Cet intervalle de confiance signifie que nous avons $1-\alpha$ % de chances que le vrai prix de l'option se situe dans cet intervalle. En augmentant le nombre de simulations n, on réduit la largeur de l'intervalle de confiance, améliorant ainsi la précision de l'estimation. Remarquons alors que si l'on veut diviser notre intervalle de confiance par deux il faut alors multiplier par quatre notre nombre de simulations.

4.1.3 Application au modèle de Black-Scholes

Pour utiliser la méthode monte carlo, il faut avant tout simuler des trajectoires de notre sous-jacent. Pour cela, nous allons utiliser dans la plupart des cas la méthode de discrétisation d'Euler qui consiste à approximer la solution de l'EDS du modèle en utilisant des intervalles de temps discrets. Supposons que nous voulons simuler le prix de l'actif sur un intervalle de temps $[0,T]$ en N étapes. L'intervalle de temps discret est alors $\triangle t = \frac{T}{N}$. L'idée est de remplacer les dérivées stochastiques par des différences finies. Ce qui donne la chose suivante:

$$\begin{aligned} S_{t+\triangle t} &= S_t + \triangle S_t \\ &= S_t\left(1 + (r-q)dt + \sigma W_{\triangle t}\right) \end{aligned}$$

A partir de cette équation il suffit de remarquer que $W_{\triangle t}$ étant un mouvement brownien, on peut l'écrire à l'aide d'une variable $Z \sim N(0,1)$ comme $W_{\triangle t} = \sqrt{\triangle t} Z$. Ainsi pour nos trajectoires il suffit d'avoir un S_0, de choisir un pas de temps et de simuler des variables gaussiennes centrées réduites.

Pour illustrer nos méthodes d'évaluation, prenons l'exemple des options vanilles. Le modèle de Black-Scholes nous fournit une formule fermée pour calculer leur prix de manière exacte. Parallèlement, nous pouvons également estimer ce prix en utilisant la méthode de Monte-Carlo. Cela nous permet d'observer comment le pricer basé sur Monte-Carlo converge vers le prix calculé par la formule fermée pour les options vanilles.

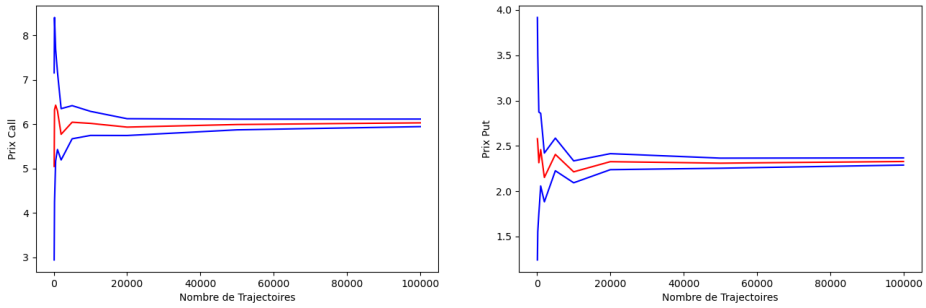

Figure 12: Prix du Call (à gauche) et Put (à droite) avec $S_0 = 100$, $K_c = 110$ et $K_p = 90$ par Monte-Carlo en fonction du nombre de trajectoires avec intervalle de confiance à 1%.

On peut également prendre comme deuxième exemple les options digitales Call et Put européenes qui donnent 1 si à maturité notre $S_T \geq K$ et 0 sinon. Il est assez simple d'obtenir une formule fermée de la valeur de ces options qui est la suivante:

$$D(S_t, t, \omega) = e^{-r(T-t)} N(\omega d_-), \text{ avec } \omega = 1 \text{ pour Call, } \omega = -1 \text{ pour Put}$$

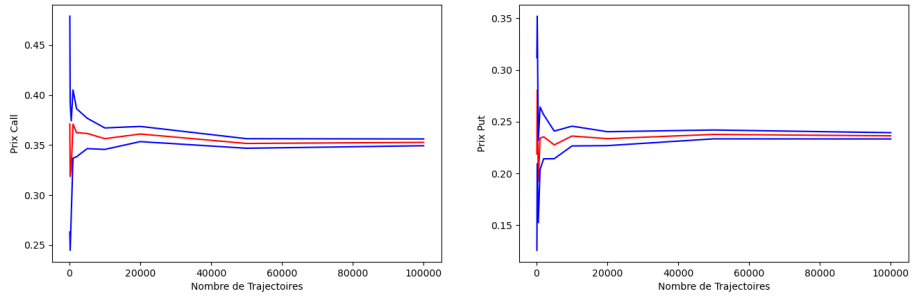

Figure 13: Prix de Digit Call (gauche) et Put (droite) avec $S_0 = 100$, $K_c = 110$ et $K_p = 90$ par Monte-Carlo en fonction du nombre de trajectoires avec intervalle de confiance à 1%.

Dans nos deux premiers exemples visant à évaluer l'efficacité de notre "pricer" Monte-Carlo, nous observons que les prix des options vanilles et digitales tendent à converger vers une valeur stable après un certain nombre de trajectoires. Il est important de souligner que dans ces cas, nous avons initié nos simulations avec un prix initial S_0 relativement proche du strike de chaque option, facilitant ainsi la convergence. En revanche, si le S_0 est significativement éloigné du strike, la convergence devient plus difficile à atteindre.

4.2 AutoCall Athéna

4.2.1 Termes dans le contrat du produit

Après s'être intéressé aux options vanilles et digitales, il serait maintenant plus ambicieux d'essayer de pricer des produits un peu plus complexe. L'autocall Athéna est un produit structuré qui a dans son contrat les termes suivants :

- B le niveau de rappel de l'autocall
- B_p le niveau de protection du capital
- K le niveau utilisé seulement à maturité
- S_0 le sous-jacent initial considéré dans le produit
- $t_1 < \ldots < t_n$ les temps d'observations du produit
- C le coupon
- N le nominal investi

Ainsi, avec ces notations on défini trois barrières qui sont la barrière de rappel de mon autocall BS_0, la barrière de protection de mon capital B_pS_0 et la barrière finale KS_0.

4.2.2 Payoff De l'AutoCall Athéna

Pour l'autocall, le payoff se distingue en deux parties : lors des dates d'observations t_j, pour $j \in [\![0, n-1]\!]$, en considérant que $\forall k < j, S_{t_k} < BS_0$, il y a deux situations possibles qui sont les suivantes :

- Si $S_{t_j} \geq BS_0$ alors l'acheteur de l'autocall perçoit le nominal N plus j coupons C. Ensuite le produit s'arête.
- Si $S_{t_j} < BS_0$ alors l'acheteur du produit ne reçoit rien et le produit continue jusqu'à la prochaine date d'observation.

Avant d'écrire le payoff pour tous les instants avant la maturité, comme le produit peut se faire rapeller, il faut instaurer un temps d'arrêt τ défini comme suit :

$$\tau = t_j \Leftrightarrow \forall k < j, S_{t_k} < BS_0 \text{ et } S_{t_j} \geq BS_0.$$

A partir des deux points précédants on peut donc maintenant écrire notre payoff V $\forall t_j \in \{t_1, \ldots, t_{n-1}\}$:

$$\begin{aligned} V(t_j) &= N(1+jC) \cdot \mathbf{1}_{\{\tau = t_j\}} \\ &= N(1+jC) \cdot \mathbf{1}_{\left\{\forall k<j, S_{t_k} < BS_0\right\}} \cdot \mathbf{1}_{\left\{S_{t_j} \geq BS_0\right\}} \\ &= N(1+jC) \cdot \prod_{k=1}^{j-1} \mathbf{1}_{\left\{S_{t_k} < BS_0\right\}} \cdot \mathbf{1}_{\left\{S_{t_j} \geq BS_0\right\}} \end{aligned}$$

Ce payoff traduit bien le fait que si à l'instant t_j le spot est supérieur à mon niveau de rappel alors nous percevons un montant puis nous sortons du produit.

Payoff à maturité
Il faut maintenant présenter le payoff final, censé être reçu à l'instant t_n qui se distingue en trois situations possibles :

- Si $S_{t_n} \geq KS_0$ alors l'acheteur de l'autocall perçoit le nominal N plus n coupons C (dans certains contrat on peut avoir $n+1$ ou même $n-1$ coupons).
- Si $B_pS_0 \leq S_{t_n} < KS_0$ alors l'acheteur du produit reçoit 100% du nominal N.
- Si $S_{t_n} < B_pS_0$ alors l'acheteur de l'autocall touchera $\frac{S_{t_n}}{S_0}N$. Il a donc une perte, qui est potentiellement totale.

A partir de ces trois points on peut donc construire le payoff final que peut toucher le détenteur du produit si et seulement si $\forall j \in [\![0, n-1]\!]$, $S_{t_j} < BS_0$:

$$V(t_n) = N(1+nC) \cdot \mathbf{1}_{\{S_{t_n} \geq KS_0\}} + N\left(\mathbf{1}_{\{B_pS_0 \leq S_{t_n} < KS_0\}} + \frac{S_{t_n}}{S_0}\mathbf{1}_{\{S_{t_n} < B_pS_0\}}\right) \tag{6.3.1}$$

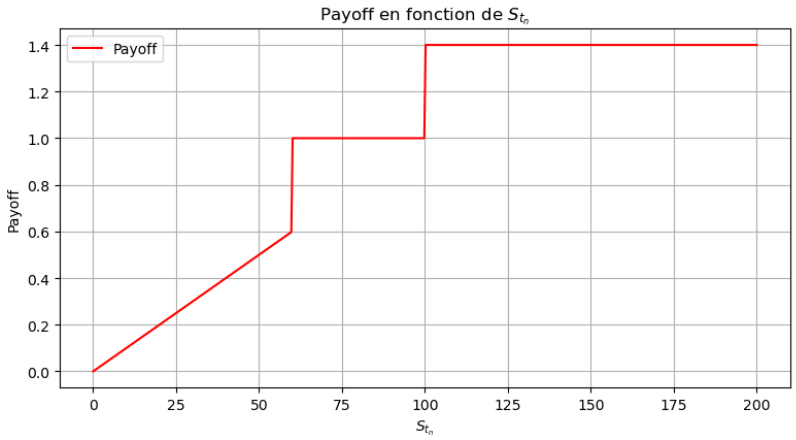

Figure 14: Payoff à maturité en fonction de S_T, $B_p = 0.6$, $K = 1$.

Nous remarquons que ce payoff n'est rien d'autre que la vente d'une option barrière européenne de type knock-in put down and in (on la note $EPDI$), de strike KS_0 et de barrière B_pS_0, et l'achat de digit. Nous verrons dans la section des variables de contrôles que ce payoff peut être réécris de manière à faire apparaître ce EPDI et ces digits. Maintenant que nous avons tous ces éléments, nous pouvons implémenter un payoff complet pour tout instant d'observation t_j et nous pouvons commencer à observer quelques résultats, notamment la convergence du prix de notre autocall calculé à partir du modèle de Black-Scholes. A noter que pricer un autocall avec le modèle de Blask-Scholes peut être une première approche qui donne une idée du prix mais dans notre autocall, nous avons plusieurs options digitales à pricer, or nous savons qu'une digitale se réplique par un call spread. Et dans notre pricer nous pricons nos digitales avec une volatilité constante or il faudrait en considérer au moins deux différentes du fait de la manière dont elles se répliquent. Nous verrons qu'un modèle à volatilité locale permet d'y remédier.

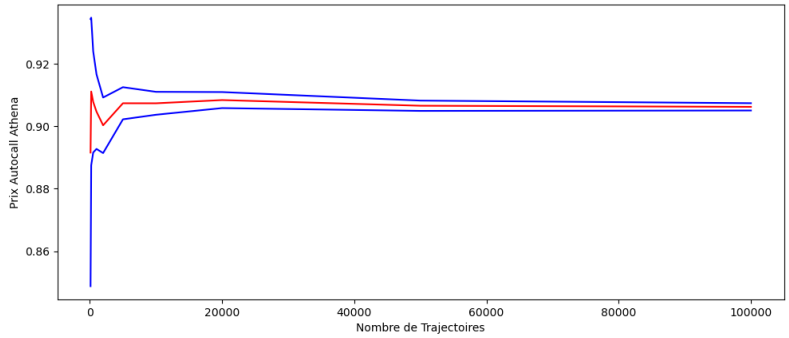

Figure 15: Prix de l'autocall Athéna avec $S_0 = 100$, $B = 100\%$, $K = 100\%$ et $B_p = 60\%$ par Monte-Carlo en fonction du nombre de trajectoires avec intervalle de confiance à 1%.

Cas de la volatilité :

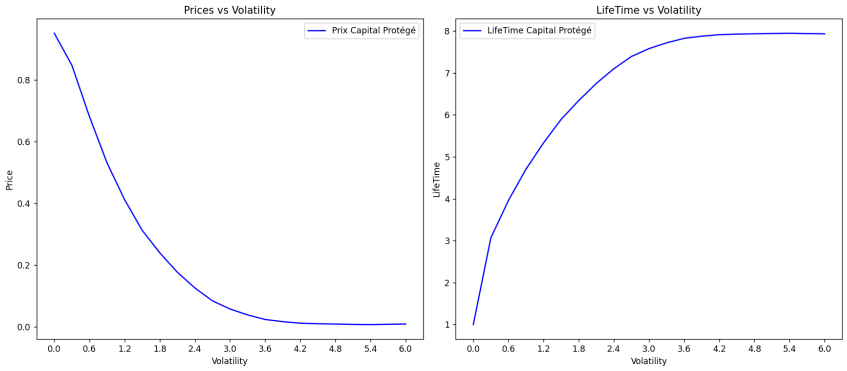

Figure 16: Evolution du prix de l'autocall ainsi que que de sa durée de vie avec $S_0 = 100$ et un spot de départ dans le pricer à 100.

Dans cet exemple, nous avons fixé le paramètre initial S_0 à 100 pour notre autocall, avec une maturité de 8 ans, et un prix de départ pour notre "pricer" à 100. Cette configuration place notre produit dans une position favorable, car il débute au seuil de rappel, fixé à 100% de notre S_0. Étant donné que le prix initial est égal à S_0, la probabilité que l'autocall soit exercé au cours des premières périodes est assez élevée. On remarque également que si notre volatilité augmente, la valeur du produit diminue. En effet, une hausse de la volatilité ne change pas la probabilité de gain ou de perte car avec une hausse de volatilité, la probabilité que le sous-jacent prenne ou perde de la valeur est la même qu'avant cette hausse. Ce qui fait que le prix diminue est que l'espérence de perte augmente considérablement comparé à l'espérence de gain qui elle ne varie que très peu avec une hausse de volatilité. Si une hausse de volatilité entraine une hausse du spot, dans presque tous les cas nous toucherons un coupon puis nous sortirons du produit, donc nous espérons presque à chaque fois le même gain. En revanche, si une augmentation de la volatilité entraîne une baisse du sous-jacent, alors plus cette augmentation est élevée, plus le risque de forte baisse du sous-jacent est important. Dans le cadre d'un autocall, plus le spot diminue, plus nous risquons de perdre, car nous n'avons qu'une protection en capital. Une facon de voir notre espérence de gain est de tracer le prix d'un autocall à capital garantie. Dans la suite nous allons illustrer ce propos en comparant le prix d'un autocall à capital garantie avec celui d'un capital protégé.

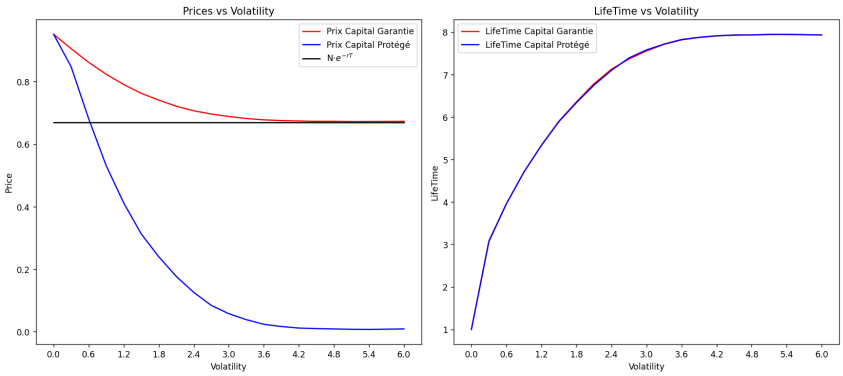

Figure 17: Evolution du prix de l'autocall à capital protégé VS capital garantie ainsi que leur durée de vie avec $S_0 = 100$ et un spot de départ dans le pricer à 100.

Observons tout d'abord que la courbe de durée de vie des produits est identique pour les deux types d'autocall. En effet, que l'autocall soit à capital garanti ou à capital protégé, cela n'affecte pas le moment où le produit sort lorsque le spot baisse considérablement. Dans les deux cas, si le niveau du spot chute significativement, nous atteignons le seuil de sortie au même moment, entraînant une durée de vie identique pour les deux types d'autocall.

La courbe rouge correspond au prix d'autocall à capital garantie, qui est l'espérence de gain car pour un tel produit a son espérence de perte en capital qui est nulle. Ce prix est théoriquement borné par Ne^{-rT} car c'est le gain minimal percu dans un produit à capital garantie. On voit que l'espérence de gain diminue très lentement avec la volatilité et on remarque alors que l'espérence de perte pour un autocall à capital protégé prend le dessus sur l'espérence de gain. A noter que l'espérence de perte est la différence entre la courbe bleue et la rouge. Ce graphe montre alors bien que plus la volatilité est élevée et plus l'espérence de perte prend le dessus sur celle des gains, ce qui entraine donc la baisse du pris de mon produit. Finalement, une hausse de volatilité fais baisser notre espérence de gain et augmente l'espérence de perte, il serait donc déconseillé d'investir dans un tel produit sur un sous-jacent volatile.

Intervalle de confiance:

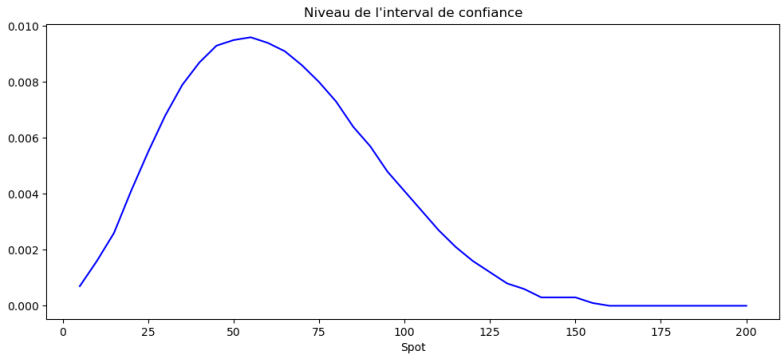

Figure 18: Evolution de l'intervalle de confiance en fonction du spot de départ ($S_0 = 100$).

On remarque qu'à certains endroits, nous avons une hausse de la variance dans l'estimation de nos prix. Notre objectif maintenant est de déterminer des méthodes qui permettent de

réduire cette variance pour avoir un prix encore plus précis de notre produit. Une technique couramment utilisée pour améliorer la précision des estimations est d'incorporer des informations supplémentaires pertinentes dans le processus de simulation.

4.2.3 Variables de Contrôles à une dimension

Cette approche vise à réduire la variance d'une variable aléatoire X en introduisant une nouvelle variable Z qui dépend de X et d'une variable Y de moyenne m, avec $\lambda \in \mathbb{R}^*$:

$$Z = X - \lambda(Y - m)$$

Il est alors immédiat que:

$$\begin{cases} \mathbb{E}(Z) = \mathbb{E}(X) \\ \mathbb{V}(Z) = \mathbb{V}(X) + \lambda^2\mathbb{V}(Y) + 2\lambda\mathbb{C}(X,Y) \end{cases}$$

L'objectif maintenant est de trouver le λ qui minimise $\mathbb{V}(Z)$. Pour cela on va donc dériver puis regarder quelle valeur de λ est optimale.

$$\begin{aligned} \frac{\partial V}{\partial \lambda}(Z) = 0 &\Leftrightarrow 2\mathbb{V}(Y)\lambda + 2\mathbb{C}(X,Y) = 0 \\ &\Leftrightarrow 2\mathbb{V}(Y)\lambda + 2\rho_{XY}\sqrt{\mathbb{V}(X)\mathbb{V}(Y)} = 0 \\ &\Leftrightarrow \lambda = -\rho_{XY}\sqrt{\frac{\mathbb{V}(X)}{\mathbb{V}(Y)}} \end{aligned}$$

On réinjecte ce λ optimal dans l'expression de $\mathbb{V}(Z)$, et on obtient:

$$\mathbb{V}(Z) = (1 - \rho_{XY}^2)\mathbb{V}(X)$$

Ainsi nous avons bien introduit une nouvelle variable dont la variance est inférieur à celle de X et d'espérance inchangée.

En observant la figure 18, on remarque que l'incertitude est la plus élevée lorsque le spot de départ est éloigné de la barrière de rappel de l'autocall. Si l'on commence loin de la monnaie, les chances que le produit se termine à maturité sont élevées. Il semble donc judicieux de prendre comme variable de contrôle le payoff terminal de l'autocall, car il peut être exprimé comme une stratégie d'options pour laquelle nous disposons d'une formule fermée par Black-Scholes.

Réécriture du payoff terminal

L'objectif suivant va être de démontrer que nous avons plusieurs écritures du payoff terminale possible et qui sont plus simple à pricer car, nous allons le démontrer, ce ne sont que des options dont nous avons une formule fermée et qui est donc facile à calculer. On note $EPDI(K,B,T)$ un $EPDI$ de strike K, de barrière B et de maturité T. Nous allons donc démontrer les équivalence suivante concernant le payoff terminal:

$$\begin{aligned} (6.3.1) \Leftrightarrow V(t_n) &= N\left(1 + nC \cdot \mathbf{1}_{\{S_{t_n} \geq KS_0\}} - \frac{1}{S_0}\left(B_pS_0 - S_{t_n}\right)_+ - (1 - B_p)\mathbf{1}_{\{S_{t_n} < B_pS_0\}}\right) \\ \Leftrightarrow V(t_n) &= N\left(1 + nC\mathbf{1}_{\{S_{t_n} \geq KS_0\}} - \frac{1}{S_0}EPDI(S_0, B_pS_0, t_n)\right) \end{aligned}$$

Preuve: Partons de (6.3.1), pour avoir:

$$\begin{aligned}
V(t_n) &= N(1+nC)\cdot \mathbf{1}_{\{S_{t_n}\geq KS_0\}} + N\left(\mathbf{1}_{\{B_pS_0\leq S_{t_n}<KS_0\}} + \frac{S_{t_n}}{S_0}\mathbf{1}_{\{S_{t_n}<B_pS_0\}}\right)\\
&= N\left(1-\mathbf{1}_{\{S_{t_n}<KS_0\}} + nC\mathbf{1}_{\{S_{t_n}\geq KS_0\}} + \mathbf{1}_{\{S_{t_n}\geq B_pS_0\}}\mathbf{1}_{\{S_{t_n}<KS_0\}} + \frac{S_{t_n}}{S_0}\mathbf{1}_{\{S_{t_n}<B_pS_0\}}\right)\\
&= N\left(1-\mathbf{1}_{\{S_{t_n}<KS_0\}} + nC\mathbf{1}_{\{S_{t_n}\geq KS_0\}} + \left(1-\mathbf{1}_{\{S_{t_n}<B_pS_0\}}\right)\mathbf{1}_{\{S_{t_n}<KS_0\}} + \frac{S_{t_n}}{S_0}\mathbf{1}_{\{S_{t_n}<B_pS_0\}}\right)\\
&= N\left(1+nC\mathbf{1}_{\{S_{t_n}\geq KS_0\}} - \mathbf{1}_{\{S_{t_n}<B_pS_0\}}\mathbf{1}_{\{S_{t_n}<KS_0\}} + \frac{S_{t_n}}{S_0}\mathbf{1}_{\{S_{t_n}<B_pS_0\}}\right)\\
&= N\left(1+nC\mathbf{1}_{\{S_{t_n}\geq KS_0\}} - \mathbf{1}_{\{S_{t_n}<B_pS_0\}} + \frac{S_{t_n}}{S_0}\mathbf{1}_{\{S_{t_n}<B_pS_0\}}\right) \text{ , car } S_{t_n}<B_pS_0 \Rightarrow S_{t_n}<KS_0\\
&= N\left(1+nC\mathbf{1}_{\{S_{t_n}\geq KS_0\}} - \mathbf{1}_{\{S_{t_n}<B_pS_0\}} + \left(\frac{S_{t_n}}{S_0} - B_p + B_p\right)\mathbf{1}_{\{S_{t_n}<B_pS_0\}}\right)\\
&= N\left(1+nC\mathbf{1}_{\{S_{t_n}\geq KS_0\}} + \frac{1}{S_0}\left(S_{t_n}-B_pS_0\right)\mathbf{1}_{\{S_{t_n}<B_pS_0\}} - (1-B_p)\mathbf{1}_{\{S_{t_n}<B_pS_0\}}\right)\\
&= N\left(1+nC\mathbf{1}_{\{S_{t_n}\geq KS_0\}} - \frac{1}{S_0}\left(B_pS_0-S_{t_n}\right)_+ - (1-B_p)\mathbf{1}_{\{S_{t_n}<B_pS_0\}}\right)
\end{aligned}$$

Et on retrouve bien notre $EPDI$ car nous savons qu'un put down and in se réplique avec l'achat d'un put et l'achat d'une digitale put. Soit un $EPDI$ de maturité T, de strike K et de barrière B, alors cette options se réplique de la manière suivante:

$$EPDI(K,B,T) = Put(B,T) + (K-B)Digit_{put}(B,T)$$

Et nous avons bien que:

$$\begin{aligned}
(B_pS_0-S_{t_n})_+ + (S_0-B_pS_0)\mathbf{1}_{\{S_{t_n}<B_pS_0\}} &= Put(B_pS_0,t_n) + (S_0-B_pS_0)Digit_{put}(B_pS_0,t_n)\\
&= EPDI(S_0,B_pS_0,t_n)
\end{aligned}$$

Et donc finalement nous avons que:

$$V(t_n) = N\left(1+nC\mathbf{1}_{\{S_{t_n}\geq KS_0\}} - \frac{1}{S_0}EPDI(S_0,B_pS_0,t_n)\right) \qquad \blacksquare$$

Ainsi notre payoff terminal s'écrit comme une somme de digitales et d'un Put Vanille, à noter que la constante 1 n'est rien d'autre qu'une digitale de strike 0 car nous avons que $\mathbb{P}(S_T>K) \xrightarrow[K\to 0]{} 1$. On a donc notre valeur m qui est le prix de ce payoff par formule fermée. Ainsi avec notre pricer Monte-Carlo nous sommes censés avoir $\mathbb{E}(Y-m)=0$ avec Y notre vecteur de simulation de prix de notre variable de contrôle pour chaque trajectoire.

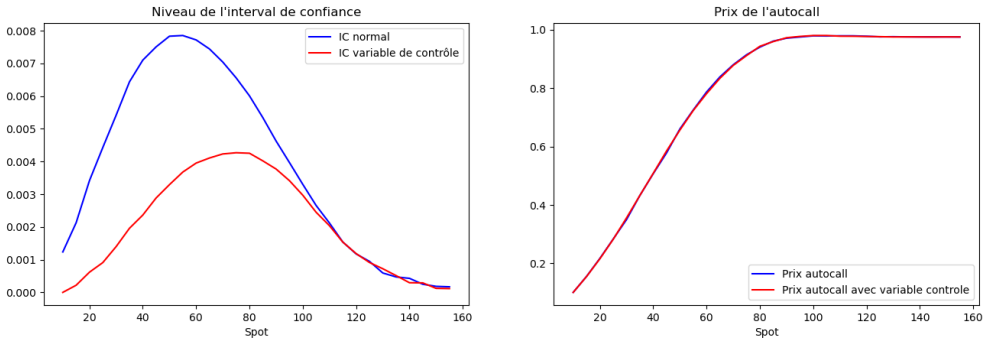

Figure 19: Evolution de l'intervalle de confiance en fonction du spot de départ du modèle (à gauche) et évolution du prix de mon produit en fonction du spot (à droite)

Ici on prend comme barrière de rappel de produit $BS_0 = 100$. On remarque bien que notre variable de contrôle permet de réduire notre variance sans changer l'espérance. Il reste une zone à améliorer qui est celle lorsque que notre spot de départ est au dessus de ma barrière de rappel, et dans cette configuration la probabilité de sortir du produit lors des premières dates d'observation est élevée donc on se dit qu'on pourrait rajouter quelques variables de contrôles en plus pour pouvoir encore plus réduire notre variance.

4.2.4 Variables de contrôles à d dimensions

Ici, nous traitons un cas général où nous choisissons d'introduire d nouvelles variables aléatoires telles que $Y = (Y_1, \dots, Y_d)$, avec un vecteur moyenne $M = (m_1, \dots, m_d)$, dans une variable Z, avec $\Lambda = (\lambda_1, \dots, \lambda_d)$ tel que:

$$Z = X + \sum_{i=1}^{d} \lambda_i (Y_i - m_i) = X + \Lambda(Y - M)$$

Il est immédiat que $\mathbb{E}(Z) = \mathbb{E}(X)$ et le calcul pour la variance est le suivant:

$$\begin{aligned}\mathbb{V}(Z) &= \mathbb{V}(X) + \sum_{i,j} \lambda_i \lambda_j \mathbb{C}(Y_i, Y_j) + 2 \sum_{i=1}^{d} \lambda_i \mathbb{C}(X, Y_i) \\ &= \mathbb{V}(X) + \Lambda^{\mathrm{T}} \Gamma \Lambda + 2 \Lambda^{\mathrm{T}} C_{XY}\end{aligned}$$

Ici l'objectif est le même que pour le cas à une dimension, on veut trouver un vecteur Λ qui minimise notre $\mathbb{V}(Z)$. On effectue une dérivation pour avoir notre solution optimale:

$$\begin{aligned}\frac{\partial V}{\partial \Lambda}(Z) = 0 &\Leftrightarrow 2\Gamma\Lambda + 2C_{XY} = 0 \\ &\Leftrightarrow \Lambda = -\Gamma^{-1} C_{XY}\end{aligned}$$

On peut donc maintenant réinjecter notre solution dans (4):

$$\begin{aligned}\mathbb{V}(Z) &= \mathbb{V}(X) + C_{XY}^{\mathrm{T}} (\Gamma^{-1})^{\mathrm{T}} \Gamma \Gamma^{-1} C_{XY} - 2 C_{XY}^{\mathrm{T}} (\Gamma^{-1})^{\mathrm{T}} C_{XY} \\ &= \mathbb{V}(X) - C_{XY}^{\mathrm{T}} \Gamma^{-1} C_{XY}\end{aligned}$$

Ainsi on a bien construit une variables aléatoire de variance inférieure à celle de X.
Le payoff d'un autocall à chaque date d'observation avant sa maturité se compose exclusivement de digitales conditionnelles, qui ne sont exercées que si les précédentes n'ont pas

été activées. De plus, le prix d'une option digitale déterminé par la méthode Monte-Carlo est simplement le quotient du nombre de fois où la barrière est franchie par le nombre total de trajectoires simulées. Un spot initial significativement éloigné du strike rend la convergence du prix de l'option digitale vers son équivalent en formule fermée difficile. Ceci peut entraîner une divergence entre l'espérance de prix de l'autocall utilisant ces digitales comme variables de contrôle et l'espérance du prix d'un autocall classique.

Par conséquent, notre stratégie consiste à intégrer les options digitales correspondant à chaque payoff pour chaque date d'observation. Nous définissons ensuite un seuil au-delà duquel une digitale est considérée comme une variable de contrôle pertinente. Si le prix en formule fermée d'une digitale est supérieur à 0.1, nous estimons que l'option ne rencontrera pas de difficulté à converger vers ce prix théorique, et par conséquent, elle n'impactera pas négativement l'espérance de notre nouvelle estimation.

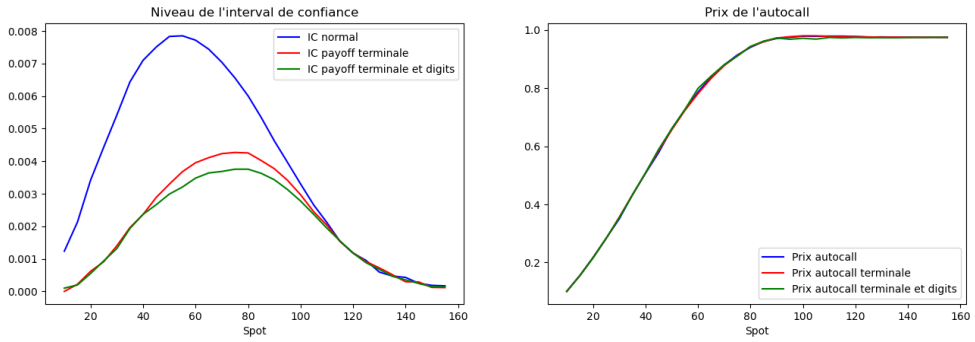

Figure 20: Evolution de l'interval de confiance en fonction du spot de départ du modèle (à gauche) et évolution du prix de mon produit en fonction du spot (à droite)

Dans le résultat du dessus, on remarque que nos nouvelles variables de contrôles sont efficaces car elles réduisent encore un peu plus notre variance du prix de mon autocall sans en changer son espérance. Nous avons donc maintenant un pricer d'autocall Athéna avec une incertitude la plus faible que l'on puisse avoir avec les outils utilisés pour le moment. Une intuition légitime est de se dire que plus notre spot initial est loin du strike plus les digits avec maturité longue ont de chances de finir dans la monnaie. Cependant il existe des cas particuliés où nous considérons une volatilité très élevée. En effet comme la payoff d'une digitale dépend du d_-, on remarque que pour un taux d'intéret r assez faible on a:

$$d_- \xrightarrow[T\to\infty]{} \left(\frac{r}{\sigma} - \frac{\sigma}{2}\right)\sqrt{T} \leq 0$$

Ainsi nous avons que d_- est décroissant en temps. Donc comme N est une fonction croissante on a que le prix de nos digitales va décroitre avec le temps pour des volatilités très élevées. Cela a pour impact que pour des volatilités élevées et un spot initial assez loin du strike de nos digitales, lors de notre sélection de variables de contrôles notre algorithme risque de garder les digits à maturité faible, or notre intuition nous dirait que notre algorithme devrait garder celles de plus grandes maturité.

4.3 Modèle de Heston

4.3.1 Le Modèle

Le modèle de Heston suppose que le prix de l'actif S_t et sa variance instantanée v_t sont déterminés par les équations différentielles stochastiques suivantes:

$$\begin{cases} dS_t = (r-q)S_t\,dt + \sqrt{v_t}S_t\,dW_t^s, \\ dv_t = \kappa(\theta - v_t)\,dt + \sigma\sqrt{v_t}\,dW_t^v, \end{cases}$$

où:

- r représente le taux de rendement sans risque de l'actif.
- $\sqrt{v_t}$ est le terme de volatilité stochastique.
- κ est le taux de retour vers la moyenne de la variance.
- θ est la variance long terme moyenne.
- σ est la volatilité de la variance.
- dW_t^s et dW_t^v sont des mouvements browniens sous le même espace de probabilité, avec une corrélation ρ, i.e., $d\langle W^s, W^v\rangle_t = \rho\,dt$.

4.3.2 La discrétisation pour Monte Carlo

Nous utilisons la méthode d'Euler-Maruyama pour discrétiser les équations stochastiques. La discrétisation des équations du modèle de Heston peut s'écrire comme suit :

$$\begin{aligned} S_{t+\Delta t} &= S_t + (r-q)S_t\Delta t + \sqrt{v_t}S_t W_{\Delta t}^s, \\ v_{t+\Delta t} &= v_t + \kappa(\theta - v_t)\Delta t + \sigma\sqrt{v_t}W_{\Delta t}^v, \end{aligned}$$

Quelque chose de compliqué est de pouvoir faire des simulations de brownien tout en sachant qu'ils sont corrélés. Pour remédier à cela nous pouvons introduire un troisième brownien W_t qui lui est indépendant des deux autres. On définit alors :

$$W_{\Delta t}^s = \lambda W_{\Delta t}^v + \mu W_{\Delta t}$$

Ainsi nous avons:

$$\mathbb{V}(W_{\Delta t}^s) = (\lambda^2 + \mu^2)\Delta t$$

Or comme $W_{\Delta t}^s$ est un brownien on a également $\mathbb{V}(W_{\Delta t}^s) = \Delta t$, on a donc que $\mu^2 = 1-\lambda^2$. De plus nous avons par définition que :

$$Cov(W_{\Delta t}^s, W_{\Delta t}^v) = \lambda t = \rho t$$

Ainsi nous avons que :

$$W_{\Delta t}^s = \rho W_{\Delta t}^v + \sqrt{1-\rho^2}W_{\Delta t}$$

Nous pouvons donc maintenant simuler nos deux browniens initiaux à condition de connaitre le paramètre ρ.

Et on peut donc finalement réecrire nos équations de discréditation:

$$S_{t+\Delta t} = S_t \left(1 + (r-q)\Delta t + \sqrt{v_t}\left(\rho W^v_{\Delta t} + \sqrt{1-\rho^2} W_{\Delta t}\right)\right),$$
$$v_{t+\Delta t} = v_t + \kappa(\theta - v_t)\Delta t + \sigma\sqrt{v_t} W^v_{\Delta t},$$

La discrétisation pour notre volatilité stochastique pose problème car dans ce cas de figure il peut y avoir des variances négatives, ce qui est inconvenant. On va donc utiliser une discrétisation étudiée de manière sérieuse [1] et on va donc choisir un schéma qui consiste en un drift-implicite de Milstein:

$$v_{t+\Delta t} = \frac{1}{1+\kappa\Delta t}\left(\left(\sqrt{v_t} + \frac{\sigma}{2} W^v_{\Delta t}\right)^2 + \left(\kappa\theta - \frac{\sigma^2}{4}\right)\Delta t\right)$$

Cette équation nous donne bien des valeurs de variances qui sont positives car pour que le modèle de Heston soit applicable il faut respecter la condition de Feller qui est que $2\kappa\theta \geq \sigma^2$.

4.3.3 Formules fermées pour les options vanilles

Nous nous intéressons maintenant à trouver une formule fermée pour, à partir des données de marché avoir des volatilités implicites pour pouvoir calibrer notre modèle. Il existe plusieurs articles qui présente ces formules mais nous prendrons les notations présentées par Heston [2] qui sont les suivantes pour le prix d'un Call Européen en notant $\tau = T - t$:

$$C(S_t, v, t) = S_t e^{-q\tau} P_1 - K e^{-r\tau} P_2$$

Il est alors trivial d'avoir le prix d'un Put Européen par la parité Call-Put:

$$P(S_t, v, t) = S_t e^{-q\tau}(P_1 - 1) - K e^{-r\tau}(P_2 - 1)$$

où nous avons pour i le nombre imaginaire tel que $i^2 = -1$ et $j \in \{1, 2\}$:

$$P_j = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty \Re\left(\frac{e^{-iu\ln(K)} f_j(S, v, \tau, u)}{iu}\right) du$$

avec $F(\tau) = S_t e^{(r-q)\tau}$:

$$f_j(S, v, \tau, u) = e^{C_j(\tau,u) + D_j(\tau,u)v + iu\ln(F(\tau))}$$

En introduisant $s_j = \frac{1}{2}1_{\{j=1\}} - \frac{1}{2}1_{\{j=2\}}$ et $b_j = \kappa(1+\sigma) - 1_{\{j=1\}}\rho\sigma$, on a:

$$\begin{cases} C_j(\tau, u) = \frac{\kappa\theta}{\sigma^2}\left(\left(b_j - \rho\sigma u i + d_j\right)\tau - 2\ln\left(\frac{1 - g_j e^{d_j\tau}}{1 - g_j}\right)\right) \\ D_j(\tau, u) = \frac{b_j - \rho\sigma u i + d_j}{\sigma^2}\left(\frac{1 - e^{d_j\tau}}{1 - g_j e^{d_j\tau}}\right) \end{cases}$$

Avec

$$\begin{cases} g_j = \frac{b_j - \rho\sigma u i + d_j}{b_j - \rho\sigma u i - d_j} \\ d_j = \sqrt{\left(\rho\sigma u i - b_j\right)^2 - \sigma^2\left(2 s_j u i - u^2\right)} \end{cases}$$

Comme le soulignent Albrecher et al. [3], un "piège" apparaît dans les équations mentionnées ci-dessus, qui peut poser des défis lors de l'implémentation. En effet, lorsque u tend vers l'infini, le module de d_j devient très grand, rendant ainsi le module du numérateur de g_j significativement plus grand que celui de son dénominateur. Cette disparité peut entraîner des difficultés dans les calculs numériques.

On prend donc ces nouvelles notations, qui sont équivalentes aux anciennes :

$$g_j = \frac{b_j - \rho\sigma ui - d_j}{b_j - \rho\sigma ui + d_j}$$

Ce qui entraine les modifications suivantes :

$$\begin{cases} C_j(\tau, u) = \frac{\kappa\theta}{\sigma^2}\left((b_j - \rho\sigma ui - d_j)\,\tau - 2\ln\left(\frac{1-g_j e^{-d_j\tau}}{1-g_j}\right)\right) \\ D_j(\tau, u) = \frac{b_j - \rho\sigma ui - d_j}{\sigma^2}\left(\frac{1-e^{-d_j\tau}}{1-g_j e^{-d_j\tau}}\right) \end{cases}$$

Implémentation et Complications:

Dans l'implémentation on utilise les notations suivantes:

$$g_j(u) = \Re\left(\frac{e^{-iu\ln(K)} f_j(S, v, \tau, u)}{iu}\right)$$

Puis on se ramène à une seule intégrale au lieu de deux pour gagner en temps de calcul et minimiser les approximations par :

$$C(S_t, v, t) = \frac{1}{2}\left(S_t e^{-q\tau} - Ke^{-r\tau}\right) + \frac{1}{\pi}\int_0^\infty \mathrm{H}(u)du$$

Où :

$$\mathrm{H}(u) = S_t e^{-q\tau} g_1(u) - Ke^{-r\tau} g_2(u)$$

Cependant, la formule fermée pour calculer le prix des options vanilles peut entrainer quelques complications. En effet, si nous voulons calculer le prix d'options qui sont OTM et avec une faible maturité, nous avons des erreurs de prix avec certains qui sont négatifs:

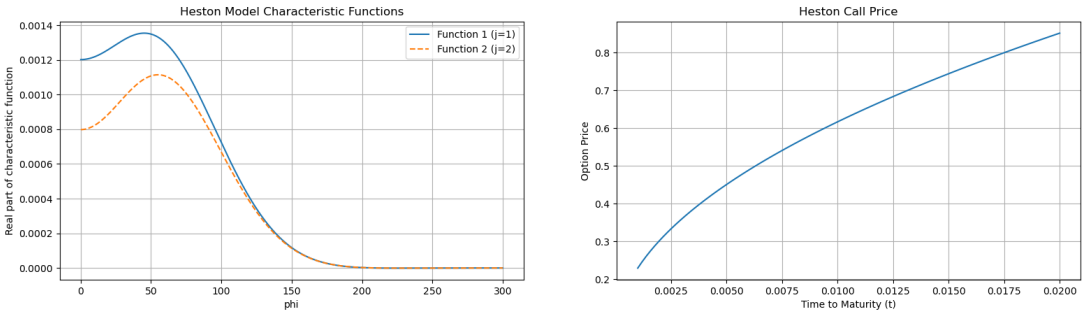

Figure 21: Fonction caractéristique pour pricer des Call at the money pour Heston, spot à 100 et strike à 100

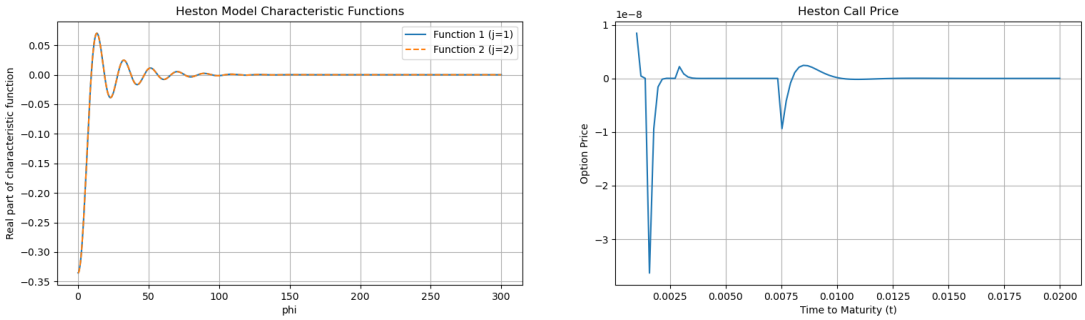

Figure 22: Fonction caractéristique pour les Call OTM pour Heston, $S_0 = 100$ et $K = 140$

Maturity	**Prix**
0.0198	8.41e-09
0.0196	4.45e-10
0.0194	9.89e-12
0.0192	-3.62e-08
0.0190	-9.42e-09
0.0188	-1.58e-09

Table 2: Evolution du prix en fonction de Tau qui représente le time to maturity.

4.3.4 Formules fermées pour les options digitales

Pour pricer une option digitale par formule fermée avec le modèle de Heston, il suffit de savoir que P_2 n'est rien d'autre que la probabilité qu'à maturité, le sous-jacent se trouve au dessus du strike K. Ainsi nous alors immédiatemant que le prix d'une option digitale call est :

$$D_C(S, v, \tau) = e^{-r\tau} P_2$$

Il est alors également évident de voir que pour la digitale put l'on ait :

$$D_P(S, v, \tau) = e^{-r\tau}(1 - P_2)$$

4.3.5 Calibration du modèle de Heston

Pour pouvoir pricer tout type de produit avec le modèle de Heston en utilisant la méthode de Monte-Carlo, il est essentiel de calibrer les paramètres du modèle aux conditions de marché. Cette étape de calibration est cruciale car elle permet d'ajuster le modèle de manière à ce qu'il reflète au mieux les réalités du marché et fournisse des estimations précises pour les prix des produits.

- **Calibration à partir des données de marché triées:** Cette méthode consiste à utiliser des données de marché pré-triées, telles que les prix des options calls et puts pour différentes maturités et différents strikes. En triant les données de marché, on s'assure de travailler avec des informations de haute qualité, en éliminant les anomalies et les erreurs de cotation. Cette approche est généralement plus rapide car elle repose sur un ensemble de données réduit et nettoyé.

En revanche, dans notre scénario de calibration il nous faut une fonction objectif. Pour ce faire, on va noter $\Theta = (\nu_0, \kappa, \theta, \rho, \sigma)$ notre vecteur de paramètres à calibrer. La première approche est de considérer la fonction objectif suivante :

$$G(\Theta) = \sum_{i=1}^{n} \left(P_{i,ma} - P_{mo}(\Theta)\right)^2$$

Avec n le nombre d'options de marchés considérées, $P_{i,ma}$ les prix de marché dépendant de la maturité et du strike, $P_{mo}(\Theta)$ le prix calculé par notre modèle de Heston avec le vecteur de paramètre Θ. Cependant, cette méthode n'est pas la meilleure pour calibrer notre modèle car il faudrait prendre en compte la liquidité de chaque option en considérant le bid et le ask car plus il y a de liquidité sur une option plus l'écart entre le bid et le ask sera faible et inversement, mais ce genre d'information n'est pas toujours donnée sur le marché ou peut être imprécise. Il devient alors naturel de considérer les volatilités implicites plutôt que les prix [6]. Notre fonction objectif devient alors:

$$G(\Theta) = \sum_{i=1}^{n} \left(\sigma_{i,ma} - \sigma_{mo}(\Theta)\right)^2$$

Où $\sigma_{i,ma}$ les volatilités implicites de Black-Scholes calculées par un inverseur dépendant de la maturité et du strike, $\sigma_{mo}(\Theta)$ les volatilités implicite calculé à partir des prix obtenu par notre modèle de Heston avec le vecteur de paramètre Θ.

Ce que nous voulons, c'est de vouloir valoriser de manière cohérente les options les plus liquides car ce sont avec ces options que les hedges se font. On veut donc réussir à mieux calibrer les options proches de la monnaie, c'est à dire celle avec la volatilité implicite les plus faibles pour une maturité donnée. Une première intuition est de prendre le vega comme pour les SVI. En effet le vega des options vanilles est plus élevé pour les options proche de la monnaie, on se dit donc que pour pouvoir mieux calibrer nos options proche de la monnaie il faut prendre en considération ce vega dans notre fonction objectif.

$$G(\Theta) = \sum_{i=1}^{n} \left(\frac{\partial P_{i,ma}}{\partial \sigma_{i,ma}}\right)^2 \left(\sigma_{i,ma} - \sigma_{mo}(\Theta)\right)^2$$

Où $\frac{\partial P_{i,ma}}{\partial \sigma_{i,ma}}$ est le νega de l'option, que l'on met au carré pour des soucis d'homogénéitée.

On récapitule dans le tableau suivant les résultats de nos calibrations avec OTM PSO qui signifie que l'on a calibré à partir des options OTM avec la méthode PSO, OTM NM avec la méthode de Nelder-Mead, et All signifie que l'on a calibrer par rapport à toutes les options et à toutes les maturités.

Θ	**OTM PSO**	**All PSO**	**OTM NM**	**All NM**
ν_0	0.14224	0.12373	0.36224	0.17916
κ	5.49127	3.78309	60.37127	9.75562
θ	0.15817	0.10156	0.12817	0.12400
ρ	-0.66529	-0.06842	-0.91529	0.09040
σ	1.29896	0.50416	0.97896	0.76890
Temps	210	225	100	115
$\mathbf{G(\Theta)}$	1.6865	63.8773	0.5748	69.8054

Table 3: Paramètres calibrés avec options OTM, toutes les options (All), avec méthode PSO et Nelder-Mead. Affichage du temps de calcul en minutes et moyenne des erreurs.

On montre maintenant les résultats en comparant nos smiles de Heston avec ceux obtenus à partir de nos prix de marchés ainsi que ceux obtenus par paramétrisation SVI pour pouvoir identifier laquelle des méthodes de calibration est la plus efficace pour calibrer les options dans la monnaie:

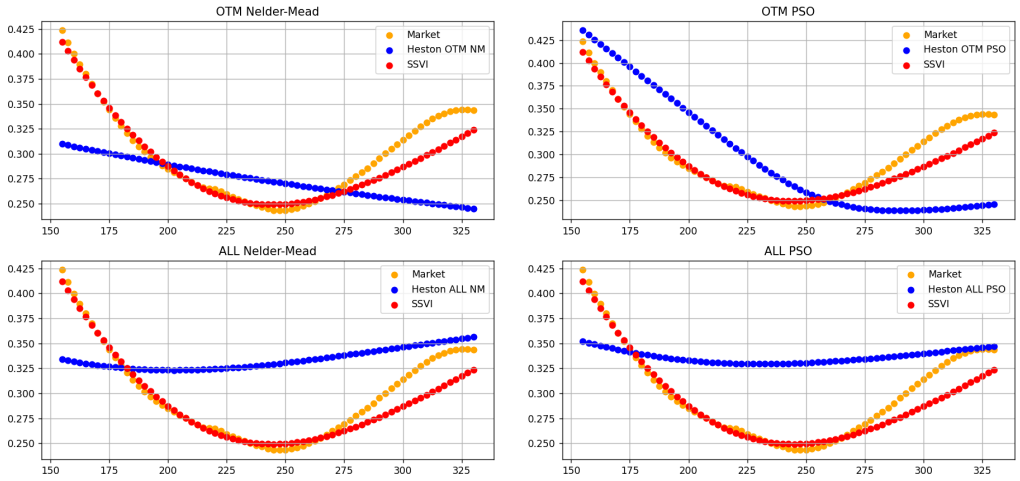

Figure 23: Calibration Heston comparée à vol implicite marché et SVI avec toutes les maturités.

On remarque qu'en considérant uniquement les options OTM, nous obtenons une calibration de meilleure qualité que si nous considérions toutes les options. On remarque tout de même qu'il y a un écart non négligeable entre les smiles SVI et Heston pour les options hors de la monnaie. La première intuition est de se dire que comme le modèle de Heston a du mal à pricer des options loin de la monnaie et en particulier pour des options à maturité assez faible, nous choisissons de garder toutes les options qui ont un time to maturity supérieur à 0.1 pour la calibration.

Voici maintenant les paramètres calibrés des quatres même manières que plus haut mais cette fois nous considérons toutes les options qui ont un time to maturity supérieur à 0,1.

Θ	**OTM PSO**	**All PSO**	**OTM NM**	**All NM**
ν_0	0.06803	0.11282	0.07078	0.12400
κ	1.33171	2.52134	2.6967	3.45729
θ	0.11161	0.13356	0.13251	0.14883
ρ	-0.31845	0.14999	-0.32892	0.09999
σ	0.49712	0.82087	0.84534	0.99795
Temps	115	95	7	43
$\mathbf{G(\Theta)}$	0.0757	63.7205	0.0601	63.8611

Table 4: Paramètres calibrés avec options OTM, toutes les options (All), avec méthode PSO et Nelder-Mead. Affichage du temps de calcul en minutes et moyenne des erreurs.

On remarque que la meilleure méthode est sans aucun doute celle de Nelder-Mead, que ce soit en temps et en précision. On montre alors aussi les résultats en comparant nos smiles de Heston avec ceux obtenus à partir de nos prix de marchés ainsi que ceux obtenus par paramétrisation SVI pour pouvoir essayer de confirmer quelle méthode de calibration est la plus efficace pour calibrer les options dans la monnaie:

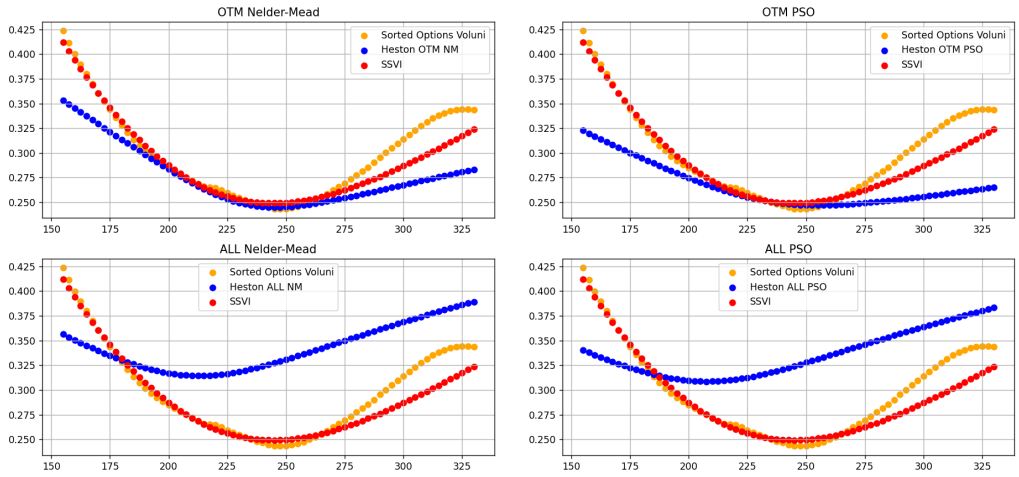

Figure 24: Calibration Heston comparée à vol implicite marché et SVI avec toutes maturité supérieure à 0,1.

On remarque que nous avons une calibration de bien meilleure qualité en supprimant quelques maturités de nos datas et pour les datas contenant uniquement les options OTM, nous avons une calibration assez précise pour les options proches de la monnaie. Pour les options OTM, la méthode PSO et Nelder-Mead nous donne une solution relativement proche. On peut donc finalement dire que calibrer le modèle de heston fonctionne mieux en considérant uniquement les options OTM qu'en prenant toutes les options. Et on peut également prétendre qu'en considérant les options OTM, nous aurons plus tendance à converger vers le minimum globale de notre fonction de calibration.

Expliquons maintenant cette différence de qualité de calibration entre différentes selection de maturité. Si la maturité est plus faible, la courbure du smile de volatilité associé à cette maturité sera plus prononcée que pour une maturité plus élevée. Ainsi les plus grands écarts possible entre nos volatilités de marché et nos volatilités calibrées sont pour les options à faible maturité, donc notre algorithme va chercher à minimiser ces grands écarts en premier et cela peut donc amener à une calibration de mauvaise qualité. De plus, comm il est dit plus haut, le modèle de Heston price assez mal les options à faible maturité et OTM, donc notre algorithme va calibrer là où les écarts sont les plus importants à partir de peu de données, donc cela risque augmente encore le risque d'avoir une mauvaise calibration. Finalement, une fois nos paramètres calibrés, nous pouvons calculer n'importe quel prix d'options que nous mettons dans notre calculateur de volatilité implicite pour avoir une surface de volatilité basée sur le modèle de Heston calibré par rapport aux données de marché. Et voici la représentation des différences entre nos volatilités implicites obtenues par paramétrisation SVI et par le modèle de Heston.

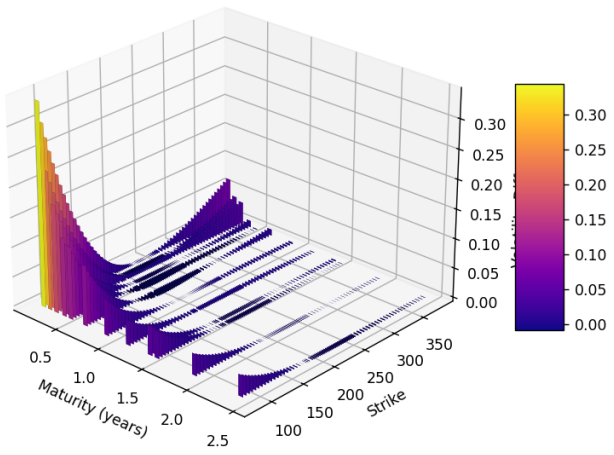

Figure 25: Comparaison surface de Heston et SVI.

4.4 Modèle à Volatilité Locale de Dupire

4.4.1 Le Modèle

Le modèle de volatilité locale de Dupire [7] permet de décrire la dynamique des prix des options en fonction de la volatilité locale, qui dépend du prix du sous-jacent et du temps. Ce document présente les principales formules et concepts associés à ce modèle.
Le modèle à volatilité locale de Dupire est une extension des modèles de Black-Scholes, permettant de mieux capturer les sourires et les skews observés dans la volatilité implicite des options. Contrairement aux modèles à volatilité constante, le modèle de Dupire suppose que la volatilité dépend à la fois du prix du sous-jacent S et du temps t.
La dynamique du prix du sous-jacent S_t sous la mesure risque-neutre $\mathbb{Q}$ est donnée par :

$$dS_t = (r - q)S_t\,dt + \sigma_{loc}(S_t, t)S_t\,dW_t,$$

- r est le taux d'intérêt sans risque,

- $\sigma_{loc}(S_t, t)$ est la volatilité locale,
- W_t est un mouvement brownien sous $\mathbb{Q}$.

Equation de volatilité locale: L'équation de Dupire pour la volatilité locale est obtenue à partir de la formule de Fokker-Planck et est donnée par [8]:

$$\sigma_{loc}^2(K,T) = \frac{\frac{\partial C}{\partial T} + (r-q)K\frac{\partial C}{\partial K} + qC}{\frac{1}{2}K^2\frac{\partial^2 C}{\partial K^2}},$$

où :

- $C(K,T)$ est le prix de l'option call pour un prix d'exercice K et une maturité T,
- $\frac{\partial C}{\partial T} = -\theta$ est la dérivée partielle de C par rapport à T,
- $\frac{\partial C}{\partial K}$ est la dérivée partielle de C par rapport à K,
- $\frac{\partial^2 C}{\partial K^2}$ est la dérivée seconde de C par rapport à K.

Les dérivées partielles des options dans le cadre de ce modèle peuvent être exprimées comme suit :

$$\theta = -\frac{\partial C}{\partial T} = -\frac{S \cdot n(d_1) \cdot \sigma \cdot e^{-q \cdot T}}{2 \cdot \sqrt{T}} - \omega \cdot r \cdot K \cdot e^{-r \cdot T} \cdot N(d_2) + \omega \cdot q \cdot S \cdot e^{-q \cdot T} \cdot N(d_1)$$

$$\Delta_{dual} = \frac{\partial C}{\partial K} = -\omega \cdot e^{-r \cdot T} \cdot N(d_2),$$

$$\Gamma_{dual} = \frac{\partial^2 C}{\partial K^2} = \frac{e^{-r \cdot T} \cdot n(d_2)}{K \cdot \sigma \cdot \sqrt{T}},$$

où :

- σ la volatilité implicite de l'optin de strike K et de maturité T
- $n(d_1)$ est la densité de la distribution normale standard évaluée en d_1,
- ω est 1 pour les calls et -1 pour les puts,
- $N(d_1)$ et $N(d_2)$ sont les fonctions de répartition de la distribution normale standard,

A noter que cette formule est équivalente pour les puts. Cela se démontre à l'aide de la parité Call-Put:

Preuve Put:

En effet, à l'aide de la parité Call-Put, nous avons que:

$$\begin{aligned} C &= P + e^{-r\tau}(F(\tau) - K), \\ \frac{\partial C}{\partial T} &= \frac{\partial P}{\partial T} - qS_t e^{-q\tau} + re^{-r\tau}, \\ \frac{\partial C}{\partial K} &= \frac{\partial P}{\partial K} - e^{-r\tau}, \\ \frac{\partial^2 C}{\partial K^2} &= \frac{\partial^2 P}{\partial K^2}. \end{aligned}$$

Et en remplacant ces expressions dans la formule pour la volatilité locale on arrive à:

$$\sigma_{loc}^2(K,T) = \frac{\frac{\partial P}{\partial T} + (r-q)K\frac{\partial P}{\partial K} + qP}{\frac{1}{2}K^2\frac{\partial^2 P}{\partial K^2}}$$

■

4.4.2 La discrétisation pour Monte Carlo

Pour le modèle à volatilité locale de Dupire, il n'y a pas de formule fermée pour le prix des options vanilles. Pour pricer une option avec une volatilité locale, il faut utiliser la méthode de Monte Carlo. Pour ce faire, il nous faut une nouvelle discrétisation. Nous choisissons encore une fois celle d'Euler.

$$S_{t+dt} = S_t\left(1 + (r-q)dt + \sigma_{loc}(t, S_t)W_{dt}\right)$$

L'enjeux principale de nos simulations de trajectoires du sous-jacent, est qu'il faut que l'on puisse avoir accès, pour n'importe quelle valeur de spot, à la volatilité locale associée. Pour le moment nous avons une surface de volatilité locale qui n'est pas "continue" donc pour une infinité de spot possible, il n'y a pas de volatilité locale associée. Donc pour y remédier nous proposerons plus bas deux méthodes pour interpoler notre surface et ainsi pouvoir avoir une approximation pour n'importe quelle volatilité locale souhaitée.

4.4.3 Surface de volatilité locale

Pour obtenir notre surface de volatilité locale, il faut que l'on mette en input dans notre modèle de Dupire une surface de volatilité implicite. La volatilité locale $\sigma_{loc}(K,T)$ est alors calculée à partir des volatilités implicites des options en utilisant l'équation de Dupire.
Le modèle à volatilité locale de Dupire offre une flexibilité accrue par rapport aux modèles à volatilité constante, permettant de mieux capturer les caractéristiques observées sur les marchés des options.
Dans la pratique, ce que nous avons fait est que nous avons mis en input la surface de volatilité implicite basée sur la paramétrisation SVI. Voici la surface de volatilité locale pour Apple :

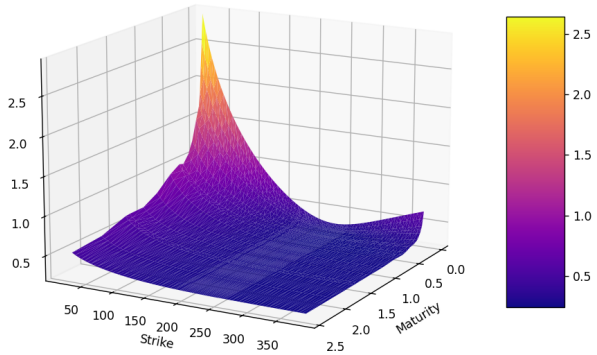

Figure 26: Surface de volatilité Locale

Et voici la surface de différences entre notre surface de volatilité locale et notre surface de volatilité implicite basée sur la paramétrisation SVI:

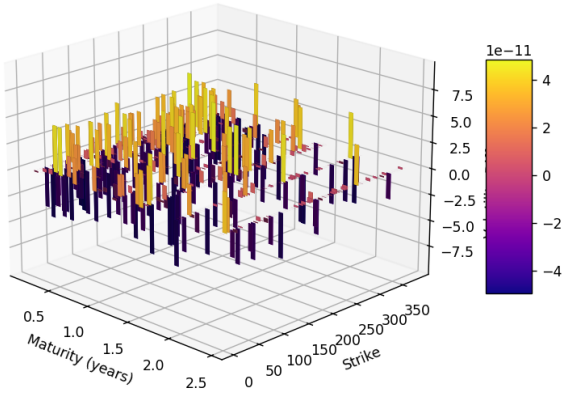

Figure 27: Surface de volatilité Locale comparée à surface SVI

4.4.4 Interpolation pour la discrétisation

Lors du pricing à volatilité locale, on veut que $\forall(t, K) \in \mathbb{R}_+^* \times \mathbb{R}_+$ associés à nos simulations, l'on ait la possibilité d'avoir accès à une volatilité locale. Les deux méthodes que l'on propose pour y arriver sont une double interpolation par spline cubique sur notre surface et une méthode d'interpolation bilinéaire. Ces deux méthodes sont expliquées ci-dessous:

Spline Cubique
Le nouveau problème est que si notre spot ou que notre dt n'est pas dans notre grille de valeur de surface, il faut trouver un moyen d'obtenir une valeur. Une première méthode est d'interpoler par spline cubique. Par exemple si S_t et dt ne sont pas dans la grille de valeur, alors il faut d'abord interpoler dt sur chaque valeur de K de la grille puis interpoler à S_t une fois cette ligne d'interpolation effectuée. Par exemple si nous avons une grille de points de marchés comme suit (vue du dessus):

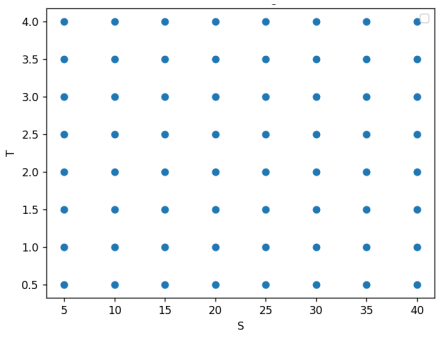

Figure 28: Surface de volatilité Locale vue du dessus

Et que à un moment dans nos simulations de prix nous avons $dt = 2.2$ et $S_t = 22.5$ alors nous allons interpoler par spline cubique pour chaque ligne verticale la valeur $dt = 2.2$ pour avoir cette grille:

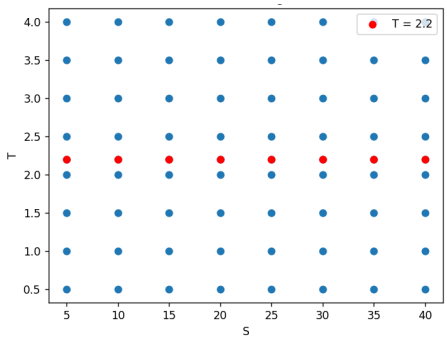

Figure 29: Surface de volatilité Locale vue du dessus avec interpolation sur tous les Strikes pour $dt = 2.2$

Puis ensuite si pour $dt = 2.2$ nous avons $S_t = 22.5$ alors il suffit de faire une dernière interpolation par spline cubique sur la ligne horizontale rouge en S_t:

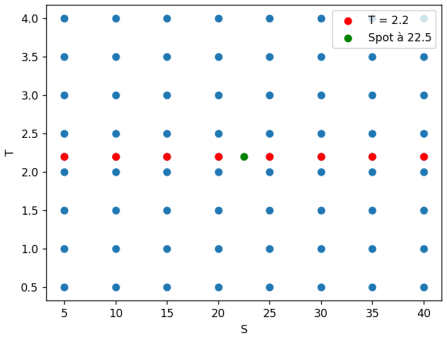

Figure 30: Surface de volatilité Locale vue du dessus avec interpolation sur tous les Strikes pour $dt = 2.2$ puis pour $S_t = 22.5$

Finalement, le point vert obtenu est notre volatilité locale au point $(2.2, 22.5)$ et nous reproduisons ce schéma pour chaque step dans notre simulation. Si notre spot ou notre dt est hors borne de la grille, on prend les valeurs aux bornes. Bien que cette méthode semble de bonne qualité, les temps de calculs sont trop importants car à chaque step de simulation nous faisons $n+1$ interpolations par spline cubique avec n le nombre de valeur de la grille en ordonnée. Il faut donc trouver une méthode plus rapide sans perdre en précision.

Interpolation Bilinéaire

Pour cela, nous choissisons de partir sur une interpolation bilinéaire plûtot qu'une interpolation par spline cubique.

Pour interpoler chaque point rouge à l'intérieur de ce carré formé par quatres points de notre grille, il faut considérer la fonction d'interpolation suivante:

$$f(x, y) = ax + by + cxy + d$$

Avec nos quatres points nous avons donc le système suivant:

$$\begin{aligned} f_{11} = f(x_1, y_1) &= ax_1 + by_1 + cx_1y_1 + d, \\ f_{12} = f(x_1, y_2) &= ax_1 + by_2 + cx_1y_2 + d, \\ f_{21} = f(x_2, y_1) &= ax_2 + by_1 + cx_2y_1 + d, \\ f_{22} = f(x_2, y_2) &= ax_2 + by_2 + cx_2y_2 + d. \end{aligned}$$

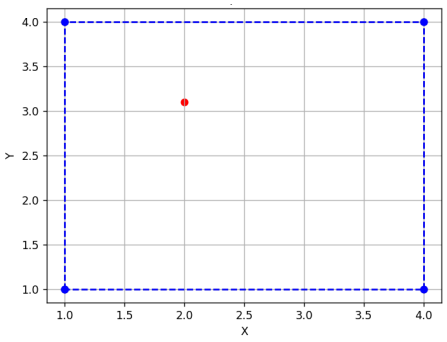

Figure 31: Zone à interpoler en fonction de quatres points.

La méthode pour simplifier ce système d'équation est de faire un changement de variable de sorte à ce que (x_1, y_1) soit le point d'origine $(0, 0)$. On prend donc comme nouvelle fonction d'interpolation. On prend donc les notations suivantes: $dx = x - x_1$ et $dy = y - y_1$.

$$f(dx, dy) = adx + bdy + cdxdy + d$$

Ainsi nous avons en notant $\triangle x = x_2 - x_1$ et $\triangle y = y_2 - y_1$. Ce qui nous ramène aux quatre points suivants:

$$\begin{aligned}(x_1, y_1) &= (0, 0),\\ (x_1, y_2) &= (0, \triangle y),\\ (x_2, y_1) &= (\triangle x, 0),\\ (x_2, y_2) &= (\triangle x, \triangle y).\end{aligned}$$

Et le système d'équations devient plus simple:

$$\begin{aligned}f_{11} &= f(x_1, y_1) = d,\\ f_{12} &= f(x_1, y_2) = b\triangle y + d,\\ f_{21} &= f(x_2, y_1) = a\triangle x + d,\\ f_{22} &= f(x_2, y_2) = a\triangle x + b\triangle y + c\triangle x\triangle y + d.\end{aligned}$$

Et ce système est équivalent à :

$$\begin{bmatrix} 0 & 0 & 0 & 1\\ 0 & \triangle y & 0 & 1\\ \triangle x & 0 & 0 & 1\\ \triangle x & \triangle y & \triangle x\triangle y & 1 \end{bmatrix} \begin{bmatrix} a\\ b\\ c\\ d \end{bmatrix} = \begin{bmatrix} f_{11}\\ f_{12}\\ f_{21}\\ f_{22} \end{bmatrix}$$

Et donc en introduisant les notations, $\triangle f_x = f_{21} - f_{11}$, $\triangle f_y = f_{12} - f_{11}$ et $\triangle f_{xy} = f_{11} + f_{22} - f_{21} - f_{12}$, on a la fonction d'interpolation pour chaque point dans la grille qui est de la forme suivante :

$$f(x, y) = \triangle f_x \frac{dx}{\triangle x} + \triangle f_y \frac{dy}{\triangle y} + \triangle f_{xy} \frac{dx}{\triangle x}\frac{dy}{\triangle y} + f_{11}$$

Avec cette technique, les temps de calculs sont considérablement plus faibles qu'avec une interpolation par spline cubique. Avec cette méthode, le code est plus lisible que pour les splines cubique et la méthode est plus compréhensible. Nous avons maintenant tous les éléments nécéssaire pour pouvoir pricer nos autocalls avec un modèle à volatilité locale.

4.4.5 Comparaison des deux méthodes d'interpolation de surface

On peut maintenant comparer nos méthodes d'interpolations pour montrer que celle-ci fonctionnent correctement. Rapellons que notre surface de volatilité a un range de strikes qui va de 5 à 380 et un range de time to maturity allant d'environ 0.001 à 2.40 années. Voici les résultats:

Time to maturity	**Strike**	**Vol_BiLinear**	**Vol_CubicSpline**	**Erreur (%)**
0,0009664	10	7,5258779	7,5258779	0,00
0,0201313	100	1,6394705	1,5488824	5,53
0,1460725	220	0,2577803	0,2572244	0,21
1,3945324	300	0,2473402	0,2473742	0,02
1,9749568	370	0,2474620	0,2479248	0,18
2,4787214	390	0,2487115	0,2487115	0,00

Table 5: Comparaison des volatilités obtenues par interpolation bilinéaire et spline cubique, et pourcentage d'erreur.

Les erreurs nulles correspondent aux cas où l'on sort de notre grille de valeurs possibles. Dans ces situations, nous choisissons de prendre la dernière valeur observée, bien qu'une régression linéaire aurait pu être utilisée pour chaque point hors de la grille. L'incertitude entre les deux méthodes d'interpolation est la plus élevée lorsque nous avons une option à maturité faible et loin de la monnaie, avec un strike proche de 0. En effet, comme le montre la figure 26, la zone la plus pentue se situe dans celle des options avec un strike proche de 0 et à faible maturité. En revanche, lorsque nous nous trouvons dans une zone plutôt plate, les erreurs sont faibles et non significatives. Il est important de noter que dans le cadre du calcul de prix d'un autocall, les simulations réalisées avec la méthode d'interpolation par spline cubique sont beaucoup plus longues qu'avec la méthode d'interpolation bilinéaire. Dans la dernière partie qui et celle du pricing d'autocall, lorsque nous devrons valoriser un autocall à l'aide de la volatilité locale, nous emploierons la méthode d'interpolation par bi-linéarité pour diminuer les temps de calculs.

5 Pricing de l'autocall Athéna

Nous avons à présent tous les objets nécessaires pour pouvoir pricer un autocall. Nous pouvons le pricer avec une volatilité constante, une volatilité stochastique et une volatilité locale. Dans ce qui suit, nous allons comparer les prix des différentes méthodes et modèles. Un premier résultat pour observer si notre pricer fonctionne correctement pour nos différents modèles et méthodes. Nous allons calculer, pour un niveau de marge donné, le coupon associé à cette marge, et ce pour différentes maturités.

5.1 Choix de la volatilité constante

Pour pouvoir pricer nos autocall via le modèle de Black-Scholes, il faut que l'on choisisse une volatilité constante. Comme notre surface de volatilité a un time to maturity maximale d'environ 2.5 ans et que nous pricons des produits de maturité plus longue, nous allons prendre comme volatilité constante la volatilité correspondante à l'option la plus proche de la monnaie que l'on a, pour la maturité la plus lointaine.

5.2 Choix du produit autocall

Le choix de nos paramètres et de payoff pour l'autocall sont les suivants: nous prenons $B = 1.0$, $K = B$, $B_p = 0.6$, sous jacent APPL, date d'observation chaque année à partir de l'émission du produit.

Payoff du produit
Le payoff se distingue en deux parties : Lors des dates d'observations t_j, pour $j \in [\![1, n-1]\!]$, en considérant que $\forall k < j, S_{t_k} < BS_0$, il y a deux situations possible qui sont les suivantes :

- Si $S_{t_j} \geq BS_0$ alors l'acheteur de l'autocall perçoit le nominal N plus j coupons C et on sort du produit.
- Si $S_{t_j} < BS_0$ alors l'acheteur du produit ne reçoit rien et le produit continu jusqu'à la prochaine date d'observation.

Payoff à maturité
Le payoff final, censé être reçu à l'instant t_n se distingue en trois situations possibles :

- Si $S_{t_n} \geq KS_0$ alors l'acheteur de l'autocall perçoit le nominal N plus n coupons C.
- Si $B_pS_0 \leq S_{t_n} < KS_0$ alors l'acheteur du produit reçoit 100% du nominal N.
- Si $S_{t_n} < B_pS_0$ alors l'acheteur de l'autocall touchera $\frac{S_{t_n}}{S_0}N$. Il a donc une perte, potentiellement totale.

5.3 Resultats du pricing

Pour ce premier graphe, nous prenons comme volatilité implicite constante celle obtenue par calibration des options de marché avec le modèle de Black-Scholes. Nous prenons les paramètres calibrés pour le Heston avec les options OTM et méthode PSO. Et enfin, nous construisons notre surface de volatilité locale avec la surface SVI en input. Nous obtenons alors ces résultats pour 150000 trajectoires:

Maturity	Coupon_BS	Coupon_Heston	Coupon_Dupire
5	3.58	7.35	2.97
8	4.35	7.15	3.56
10	4.43	7.01	3.67
12	4.46	6.70	3.63
15	4.38	6.12	3.48

Table 6: Comparaison des coupons calculés avec les modèles BS, Heston et Dupire pour un prix d'autocall à 88%.

Ce tableaux représente le cas de calcul de coupon à verser au détenteur du produit si il y a une marge de 12% à faire sur ce produit. On peut aussi regarder pour quelle valeur de coupon la marge sur ce produit serait de 0%:

Maturity	Coupon_BS	Coupon_Heston	Coupon_Dupire
5	14.38	18.97	13.31
8	13.23	16.45	12.01
10	12.57	15.72	11.53
12	11.99	14.87	11.04
15	11.41	13.90	10.42

Table 7: Comparaison des coupons calculés avec les modèles BS, Heston et Dupire pour un prix d'autocall à 100%.

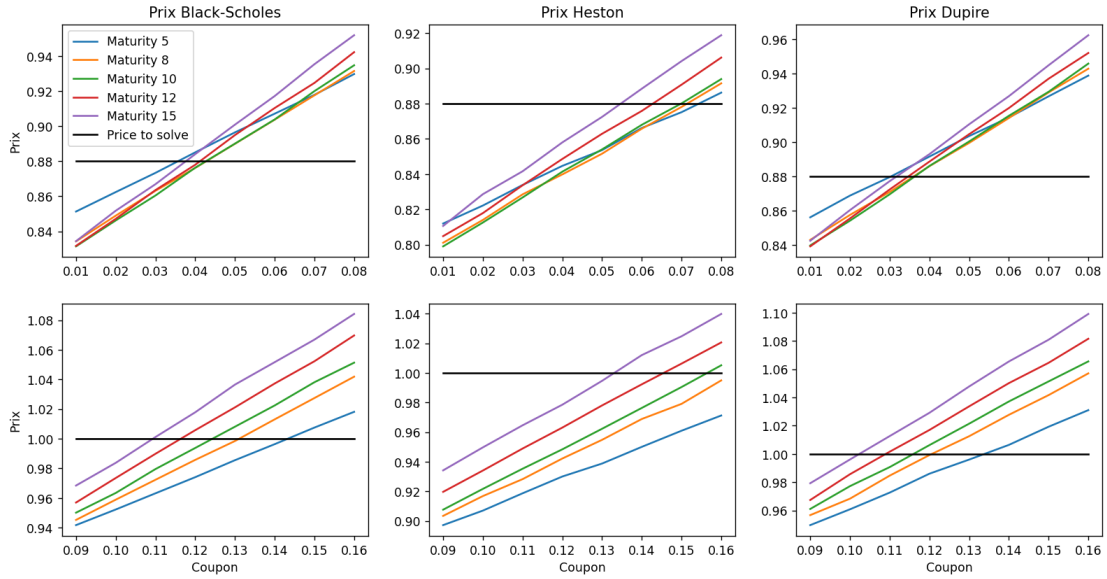

Figure 32: Courbe des prix en fonction du modèle et du coupon.

Avec la figure ci-dessus, l'hypothèse est vérifiée et alors cela explique les résutats. On remarque alors que pour des prix assez faibles, il n'y a pas de logique de croissance ou décroissance des coupons selon la maturité du produit. Cependant, cette méthode est très couteuse en calculs car à chaque itération de notre méthode de Brent, nous refaisons des

simulations pour notre sous-jacent, or on voit bien que pour un prix de 88%, il faut une standard deviation très réduite pour être certain d'atteindre le bon coupon sans entrer dans la "zone" d'un autre coupon pour une maturité différente, donc il faut un grand nombre de trajectoires car nous n'avons de méthodes de réduction de variances pour le modèle de Dupire car nous n'avons de formules fermées pour avoir le prix d'options vanilles ou digitales. Contrairement à un prix de 100% qui lui peut donner des résultats satisfaisants.

De plus, comme à chaque itération nous re-simulons des trajectoires, la méthode de Brent peut être amenée à ne pas converger car comme nous n'avons jamais les mêmes trajectoires, les chances d'avoir à un moment une erreur à 10^{-8} est presque impossible.

Ce que nous pouvons faire est de faire nos simulations de trajectoires une seule fois pour chaque maturité et chaque modèle puis nous les stockons dans une variable et nous les réutilisons dans l'optimisation pour avoir une estimation plus précise. Cette manière de procéder devrait marcher si l'on considère un nombre suffisant de simulations au départ pour avoir une estimation plus précise. Cette méthode nous permet également de considérablement gagner en temps de calcul, au lieu de refaire des simulations à chaque itération, nous les faisons une unique fois et ensuite c'est très rapide pour que notre solver de coupon ait une bonne précision avec la méthode de Brent. Voici les résultats pour 400000 trajectoires.

Maturity	**Coupon_BS**	**Coupon_Heston**	**Coupon_Dupire**
5	3.62	7.40	3.01
8	4.23	7.16	3.57
10	4.33	6.95	3.69
12	4.11	6.22	3.46
15	3.70	5.48	3.21

Table 8: Comparaison des coupons calculés avec les modèles BS, Heston et Dupire pour un prix d'autocall à 88%.

Maturity	**Coupon_BS**	**Coupon_Heston**	**Coupon_Dupire**
5	14.42	18.74	13.32
8	12.97	16.54	12.05
10	12.48	15.78	11.61
12	11.73	14.49	10.89
15	10.92	13.35	10.27

Table 9: Comparaison des coupons calculés avec les modèles BS, Heston et Dupire pour un prix d'autocall à 100%.

On peut aussi appliquer cela pour retracer les 6 graphes de prix en fonction du coupon pour chaque marturité où pour chaque maturité on fais des simulations et on les réutilisent pour tous les coupons. On s'attends à avoir des courbes de prix bien plus lissent qu'avant.

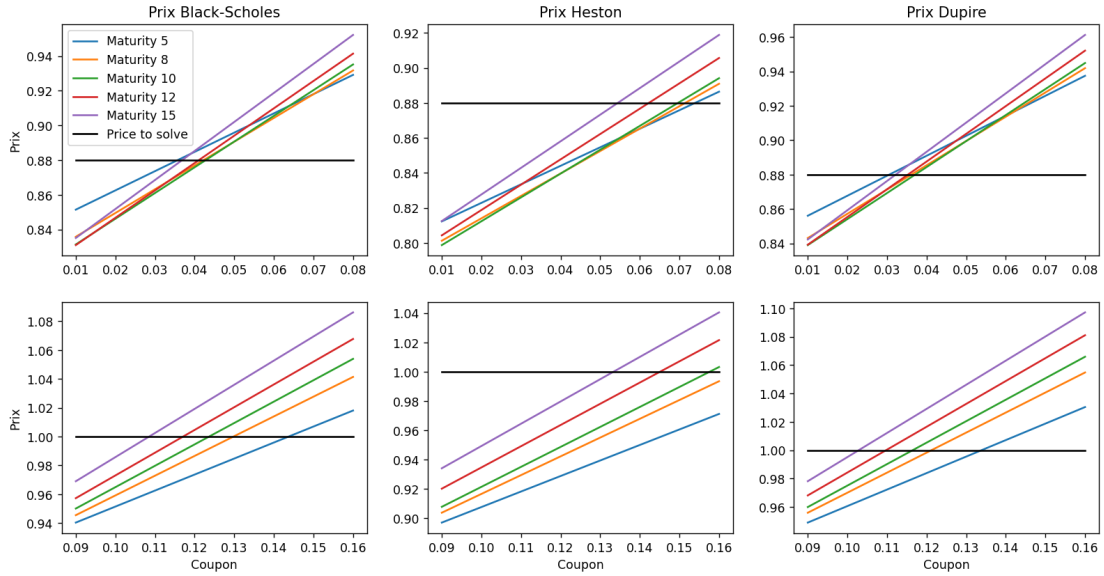

Figure 33: Courbe des prix en fonction du modèle et du coupon.

Ces résultats montrent bien que cette deuxième manière de faire est plus rapide sans perdre en précision. Cette méthode permet de passer de près de 10 heures de calculs à environ 0.5 heure. Pour savoir laquelle est la plus précise, il faudrait pouvoir comparer nos prix avec ceux émis sur le marché et ainsi avoir un avis plus tranché sur notre pricer.

Pour finir, nous pouvons conclure que le modèle le plus aproprié pour pricer ce genre de produit est le modèle à volatilité stochastique de Heston principalement car une fois calibré, nous pouvons pricer toutes options, et ce pour n'importe quelle maturité à condition qu'elle ne soit pas trop faible mais en générale les dates d'observations du produit sont à des maturités suffisement élevées pour pouvoir pricer correctement notre produit avec heston. Alors que pour le modèle à volatilité locale de Dupire, nous sommes restreints car nous avons une surface finie en termes de maturité, bien que des techniques d'extrapolations existes, cela n'aurait surement pas vraiment de sens d'extrapoler une surface 3 ans à une surface 10 ans par exemple. On peut donc conclure que si nous avons une méthode de calibration suffisement robuste pour notre Heston, alors nous pouvons tirer un bon prix de notre autocall.

6 Conclusion et Recommandations

7 Bibliographie

1. M. Altmayer & A. Neuenkirch, *Discretizing the Heston Model: An Analysis of the Weak Convergence Rate*, 2010.

2. Steven L. Heston, *A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options*, 1993.

3. H. Albrecher et al., *The Little Heston Trap*, 2006.

4. C. Kahl & P. Jackel, *Not-so-complex logarithms in the Heston model*, 2006.

5. S. Benaim et al., *An arbitrage-free method for smile extrapolation*, 2008.

6. R. Cont & P. Tankov, *Nonparametric calibration of jump-diffusion option pricing models.*, 2004.

7. B. Dupire, *Pricing and hedging with smiles*, 1993.

8. G. Deelstra & G. Rayée, *Local Volatility Pricing Models for Long-dated FX Derivatives*, 2012

9. J. Gatheral & A. Jacquier†, *Arbitrage-free SVI volatility surfaces*, 2013

10. A. Öhman, *The Calibrated SSVI Method - Implied Volatility Surface Construction*, 2019

11. A. Aurell, *The SVI implied volatility model and its calibration*, 2014

12. P. Tankov, *Surface de volatilité*, 2015

13. A. Ben Haj Yedder, *Modèle de Heston, Pricing d'options européennes et calibration*, 2007