Options barrières dans le LIBOR Market Model

Yannis Oulefki - Grégoire Szymanski

Résumé

Implémenter l'algorithme introduit dans [KT12] (similaire à la méthode du pont brownien pour le pricing des options barrières dans un modèle de diffusion). Faire le pricing de barrier cap et de barrier swaption par cette méthode et comparer à un Monte-Carlo classique.

1 Présentation de l'algorithme

1.1 Le modèle

Soit $(\Omega, \mathcal{F}, (\mathcal{F}_t)_t, \mathbb{P})$ un espace filtré complet sur lequel on définit un mouvement brownien de dimension r noté $(B_t)_t$. On fixe $T > t_0$. Dans ce qui suit, on prendra souvent $t_0 = 0$.

Introduisons le systeme d'equations differentielles stochastiques :

$$\begin{cases} dX_t &= b(t, X_t)\, dt + \sigma(t, X_t)\, dB_t \\ dY_t &= Y_t\, c(t, X_t)\, dt \\ dZ_t &= Y_t\, g(t, X_t)\, dt \end{cases} \tag{1}$$

où X, Y et Z sont trois diffusions à valeurs dans $\mathbb{R}^d$, $\mathbb{R}_+^*$ et $\mathbb{R}$, et où $b(t,x)$ est un vecteur dans $\mathbb{R}^d$, $\mathbb{R}$, $\sigma(t,x)$ est une matrice de taille $d \times r$ et $c(t,x)$ et $g(t,x)$ sont des scalaires. Nous supposerons toutes ces fonctions continues. Lorsque cela sera nécéssaire, nous noterons $X^{t,x}$, $Y^{t,x,y}$ et $Z^{t,x,y,z}$ les diffusions où nous fixons le point de départ. Soit G un domaine borné de $\mathbb{R}^d$. On note $Q = [t_0, T[\times G$ le cylindre correspondant. Enfin, on note τ le premier temps de sortie de X de G avant T, c'est-à-dire le temps de sortie de (t, X) de Q. Remarquons que si Q est régulier, τ est un temps d'arrêt. Nous nous placerons uniquement dans ce cadre. On définit enfin

$$u : \left| \begin{array}{ccc} Q & \longrightarrow & \mathbb{R} \\ (t,x) & \longmapsto & \mathbb{E}^{t,x}\left[\varphi(\tau, X_\tau)\, Y_\tau + Z_\tau\right] \end{array} \right. .$$

Remarque 1. *Pour le moment, les diffusions Y et Z peuvent être considérées comme superflues, puisque celles-ci s'expriment simplement en fonction de X. En fait, on peut facilement voir que :*

$$u(t,x) = \mathbb{E}^{t,x}\left[\varphi(\tau, X_\tau) \exp\left(-\int_t^\tau c(r, X_r) dr\right) + \int_t^\tau \exp\left(-\int_t^s c(r, X_r) dr\right) g(s, X_s) ds\right].$$

La forme plus concise permet cependant un raisonnement plus approfondi sur ces équations différentielles stochastiques aboutissant à des méthodes de réduction de variables.

Proposition 1. *Introduisons maintenant $\mu(t,x)$ et $F(t,x)$ deux vecteurs dans $\mathbb{R}^d$. On suppose que μ et F sont continus sur $\bar{Q}$. Alors la valeur de u n'est pas modifiée si on remplace les diffusions de X, Y et Z par :*

$$\begin{cases} dX_t &= (b(t, X_t) - \sigma(t, X_t)\mu(t, X_t))\, dt + \sigma(t, X_t)\, dB_t \\ dY_t &= Y_t\, c(t, X_t)\, dt + Y_t\, {}^t\mu(t, X_t)\, dB_t \\ dZ_t &= Y_t\, g(t, X_t)\, dt + Y_t\, {}^tF(t, X_t)\, dB_t \end{cases} \tag{2}$$

Preuve. Fixons (t,x) et calculons $u(t,x)$. Pour tout $s \geq t$, on pose :

$$L_s = \exp\left(-\int_t^s {}^t\mu(r, X_r) dB_r - \frac{1}{2}\int_t^s |\mu(r, X_r)|^2\, dr\right).$$

On sait que L_s est une martingale (car μ est bornée car continue sur un compact) donc on peut définir une probabilité $\mathbb{Q}$ sur $(\Omega, \mathcal{F}_T)$ équivalente à $\mathbb{P}$ de densité L_T. Par le théorème de Girsanov, on sait que $W_s = B_s + \int_t^s \mu(r, X_r)dr$ est un mouvement brownien standard sous $\mathbb{Q}$. On en déduit donc que :

$$\begin{aligned} u(t,x) &= \mathbb{E}_{\mathbb{Q}}^{t,x}\left[\frac{\varphi(\tau, X_\tau)\, Y_\tau + Z_\tau}{L_\tau}\right] = \mathbb{E}_{\mathbb{Q}}^{t,x}\left[\varphi(\tau, X_\tau)\,\frac{Y_\tau}{L_\tau} + \frac{Z_\tau}{L_\tau}\right] \\ &= \mathbb{E}_{\mathbb{Q}}^{t,x}\left[\varphi(\tau, X_\tau)\,\widetilde{Y}_\tau + Z_\tau + \int_t^\tau \widetilde{Y}_r\, {}^t(F-\mu)(r, X_r)dW_r\right] = \mathbb{E}_{\mathbb{Q}}^{t,x}\left[\varphi(\tau, X_\tau)\,\widetilde{Y}_\tau + \widetilde{Z}_\tau\right] \end{aligned}$$

où $\widetilde{Y}_s = \frac{Y_s}{L_s}$ et où $\widetilde{Z}_s = \frac{Z_s}{L_s} + \int_t^s \widetilde{Y}_r\, {}^t(F-\mu)(r, X_r)dW_r$. On peut alors vérifier que :

$$\begin{aligned} dX_t &= b(t, X_t)\ dt + \sigma(t, X_t)\ dB_t \\ &= b(t, X_t)\ dt + \sigma(t, X_t)\,(\,dW_t - \mu(t, X_t)\ dt) \\ &= (b(t, X_t) - \sigma(t, X_t)\mu(t, X_t))\ dt + \sigma(t, X_t)\ dW_t. \end{aligned}$$

Et en utilisant la formule d'Ito :

$$\begin{aligned} d\widetilde{Y}_t &= \widetilde{Y}_t\, c(t, X_t)\, dt + \widetilde{Y}_t\, {}^t\mu(t, X_t)\, dW_t, \\ d\widetilde{Z}_t &= \widetilde{Y}_t\, g(t, X_t)\, dt + \widetilde{Y}_t\, {}^tF(t, X_t)\, dW_t, \end{aligned}$$

ce qui permet de conclure.

□

Le but de ce rapport est de présenter une méthode alternative de calcul (approché) de u.

L'une des méthode usuelle consiste à expliciter une EDP satisfaite par u puis de la résoudre (voir Proposition 2). Cependant, cette approche est difficilement implémentable en grande dimension. Il faut donc se tourner vers d'autres méthodes telles que celle de Monte Carlo. Pour implémenter celle-ci, deux points sont à considérer.

Pour cela, il faut tout d'abord simuler la variable $\varphi(\tau, X_\tau)\, Y_\tau + Z_\tau$. Nous rappellerons brièvement l'algorithme basé sur une discrétisation de l'EDS par un schéma d'Euler dans la section 1.3. Puis nous allons montrer comment celui-ci peut etre modifié en se basant sur les marches aléatoires afin de limiter la complexité des simulations.

Cependant, avant de discuter de la simulation en soi, nous allons présenter des méthodes génériques de réduction de variances basée sur un choix judicieux d'EDS à discrétiser.

1.2 Réduction de variance

La simulation de $\varphi(\tau, X_\tau)\, Y_\tau + Z_\tau$ passe par l'approximation de la trajectoire $(s, X(s))$ tant que celle-ci reste dans le domaine Q.

Lemme 1. *Le procéssus $(u(t, X_t)Y_t + Z_t)_t$ arreté au temps τ est une martingale.*

Preuve. Soit $H \leq \tau$ un temps d'arret borné presque surement. On remarque alors que $(X_{H+s}, \frac{Y_{H+s}}{Y_H}, \frac{Z_{H+s} - Z_H}{Y_H})_s$ avec valeur initiale pour $X_t = x$ suit la même loi que (X_s, Y_s, Z_s) avec valeur initiale X_H à l'instant H. Cela provient de l'equation 2 et de l'ecriture explicite des processus Y et Z lorsque ceux-ci ont pour valeurs initiales 1 et 0. De plus, comme $H \leq \tau$, on sait que $\tau \circ X = \tau \circ (X_{H+t})_t$. Par conséquent :

$$\begin{aligned} \mathbb{E}^{t,x}\left[u(H, X_H)Y_H + Z_H\right] &= \mathbb{E}^{t,x}\left[\mathbb{E}^{H, X_H}\left[\varphi(\tau, X_\tau)\, Y_\tau + Z_\tau\right] Y_H + Z_H\right] \\ &= \mathbb{E}^{t,x}\left[\mathbb{E}\left[\varphi(\tau, X_\tau)\,\frac{Y_\tau}{Y_H} + \frac{Z_\tau - Z_H}{Y_H} | \mathcal{F}_H\right] Y_H + Z_H\right] \\ &= \mathbb{E}^{t,x}\left[u(\tau, X_\tau)Y_\tau + Z_\tau\right] \\ &= u(t, x). \end{aligned}$$

Cela permet bien de conclure que $(u(t, X_t)Y_t + Z_t)_t$ est une martingale.

□

Proposition 2. *u est solution de l'equation*

$$\partial_t u + \frac{1}{2}\operatorname{Tr}\left(\sigma {}^t\sigma \nabla^2 u\right) + {}^t b \nabla u + cu + g = 0$$

et de plus $d\left(u(t,X_t)Y_t + Z_t\right) = {}^t\left(F + \mu u + {}^t\sigma \nabla u\right) dB_t$

Preuve. Il suffit de verifier que

$$d\left(u(t,X_t)Y_t + Z_t\right) = {}^t\left(F + \mu u + {}^t\sigma \nabla u\right) dB_t + \left(\partial_t u + \frac{1}{2}\operatorname{Tr}\left(\sigma {}^t\sigma \nabla^2 u\right) + {}^t b \nabla u + cu + g\right) dt$$

puis d'utiliser le lemme précédent pour annuler le terme en dt. □

Cette proposition est tres intéressante d'un point de vue pratique. En effet, on a : $u(\tau, X_\tau)Y_\tau + Z_\tau = \int_0^\tau {}^t\left(F + \mu u + {}^t\sigma \nabla u\right) dB_t$ et donc

$$\operatorname{var}\left(u(\tau, X_\tau)Y_\tau + Z_\tau\right) = \mathbb{E}\left[\int_0^\tau \left\|\left(F + \mu u + {}^t\sigma \nabla u\right)\right\|^2 dt\right].$$

Par conséquent, si F et μ sont tels que $F + \mu u + {}^t\sigma \nabla u = 0$, la variance de la simulation Monte-Carlo est nulle ! Bien qu'impossible à utiliser dans la pratique (car cela nécessiterait de connaitre à priori u), on peut utiliser des approximations de u afin de tenter des choix de F et de μ.

Remarquons également que cette technique de réduction de variance est une généralisation de méthodes bien connues. En effet, si on fixe $\mu = 0$, l'utilisation du F revient à modifier la variable simulée par un facteur additif d'esperance nulle (l'integrale stochastique $\int_t^\tau Y_r {}^t F(r, X_r) dW_r$). Cela correspond donc à la technique dite des variables de controle. A l'inverse, en fixant $F = 0$, l'introduction du μ correspond à un changement de probabilités via le théoreme de Girsanov. Cela correspond donc à la technique d'importance sampling.

Finalement, d'autres techniques de réduction de variances peuvent être employées. Nous utiliserons par la suite la technique des variables antithétiques. Cela correspondra à automatiquement simuler les bruits Ξ par $-\Xi$ et les variables uniformes U sur $[0,1]$ nécessaires pour la simulation des lois de Bernouillis par $1-U$. La justification théorique de variables antithétiques n'est pas valable ici car les payoffs ne sont pas toujours des fonctions monotones des aléas. (Et quand ils le sont, il est souvent difficile de le vérifier compte tenu de la discrétisation des équations stochastiques). Cependant, nous avons tout de même essayé cette technique. En pratique, nous n'utilisons pas la technique antithétique pour les variables de Bernoulli (simulées à l'aide de variables uniformes) utilisées dans l'algorithme des marches aléatoires. En effet, cela nécessiterait de stocker les variables simulées et donc soit de rajouter plusieurs branchements conditionnels, soit de simuler cette variable même lorsqu'on n'en a pas besoin. Dans tous les cas, les gains sont négligeables comparés aux couts supplémentaires étant donné qu'il est très rare que les deux branches de la simulation (la branche classique et antithétique) aient besoin de cette variable lors de la même itération.

1.3 Le schéma d'Euler

Commençons par fixer (t,x) les conditions initiales de notre simulation ainsi qu'un pas de discrétisation $h = \frac{T-t}{n} > 0$ supposé petit . On note $t_k := t + kh$ les instants de discrétisation. Appliquons l'algorithme de discrétisation d'Euler au système 2 :

$$\begin{cases} \bar{X}^{t,x}(t_{k+1}) & = \bar{X}^{t,x}(t_k) + h(b(t,\bar{X}^{t,x}(t_k)) - \sigma(t,\bar{X}^{t,x}(t_k))\mu(t,\bar{X}^{t,x}(t_k))) + \sqrt{h}\sigma(t,\bar{X}^{t,x}(t_k))\xi_k \\ \bar{Y}^{t,x}(t_{k+1}) & = \bar{Y}^{t,x}(t_k) + hc(t,x)\bar{Y}^{t,x}(t_k) + \sqrt{h}{}^t\mu(t,\bar{X}^{t,x}(t_k))\bar{Y}^{t,x}(t_k)\xi_k \\ \bar{Z}^{t,x}(t_{k+1}) & = \bar{Z}^{t,x}(t_k) + hg(t,\bar{X}^{t,x}(t_k))\bar{Y}^{t,x}(t_k) + \sqrt{h}{}^t F(t,\bar{X}^{t,x}(t_k))\bar{Y}^{t,x}(t_k)\xi_k \end{cases} \tag{3}$$

avec bien sûr $\bar{X}^{t,x}(t) = x$, $\bar{Y}^{t,x}(t) = 1$ et $\bar{Z}^{t,x}(t) = 0$. Les variables ξ_k sont i.i.d. et sont toutes des vecteurs gaussiens standards dans $\mathbb{R}^d$.

Cette simulation se poursuit tant que $\bar{X}^{t,x}(t_k)$ est dans Q. Si c'est le cas jusqu'à $k = n$, alors on peut poser $\tau = T$. Sinon, il existe k tel que $\bar{X}^{t,x}(t_k) \in Q$ mais $\bar{X}^{t,x}(t_{k+1}) \notin Q$. On peut alors considérer λ le plus petit réel de $[0,1]$ tel que

$$\bar{X}^{t,x}(t_k) + \lambda(\bar{X}^{t,x}(t_{k+1}) - \bar{X}^{t,x}(t_k)) \notin Q.$$

Puis, $\bar{\tau}$, $\bar{X}_{\bar{\tau}}$, $\bar{Y}_{\bar{\tau}}$ et $\bar{Z}_{\bar{\tau}}$ sont définis comme les interpolations linéaires de points $1-\lambda$ et λ entre leurs valeurs en t_k et en t_{k+1}.

Cette méthode a de nombreux défauts. Par exemple, on voit que le temps de sortie $\bar{\tau}$ est toujours sous-estimé. Cela vient du fait que l'on peut éventuellement sortir entre t_k et t_{k+1} puis rentrer de nouveau dans Q. Une des solutions possible pour corriger ce biais est la méthode du pont brownien mais celle-ci ne fonctionne que dans des cas particuliers.

1.4 L'algorithme des marches aléatoires

Nous allons maintenant présenter les algorithmes introduits dans [KT12]. Reprenons les notations de la méthode précédente. L'algorithme des marches aléatoires se base sur une simulation différente des trajectoires de $(s, X(s))$ que celle d'Euler. Dans le cas général, c'est-à-dire tant que $\bar{X}$ reste "loin" du bord de G, l'expression analytique de $\bar{X}^{t,x}(t_{k+1})$, $\bar{Y}^{t,x}(t_{k+1})$ et $\bar{Z}^{t,x}(t_{k+1})$ reste celle donnée par 3, mais la loi des Ξ_k est changée. Dorénavant, $\xi_k = {}^t(\xi_k^1, ..., \xi_k^d)$ est un vecteur de loi uniforme dans $\{-1, +1\}^d$.

Le choix d'une telle loi pour le vecteur ξ_k peut être motivée par plusieurs observations. Tout d'abord, lorsque le pas de discrétisation est suffisament fin, n sera elevé et le théoreme central limite nous assure que la somme normalisée des ξ_k converge bien vers une Gaussienne. De plus, la loi uniforme dans $\{-1, 1\}$ est, tout comme une $\mathcal{N}(0,1)$, centrée et de variance 1. Cette loi est également plus facile à simuler.

Nous allons maintenant distinguer le comportement de cette simulation lorsque celle-ci s'approche du bord.

On introduit l'ensemble des points proches du bord $S_{t,h} \subset \overline{G}$ définit tel que $x \in S_{t,h}$ si au moins une des 2^r valeurs possibles du vecteur $\bar{X}^{t,x}(t+h)$ simulé avec 3 est en dehors de $\overline{G}$. La compacité de $\overline{Q}$ implique l'existence d'une constante $\lambda > 0$ telle que si la distance entre $x \in G$ et le bord de G est supérieure ou égale à $\lambda\sqrt{h}$, alors x est en dehors de $S_{t,h}$. Cela vient du fait que $\left\|\bar{X}^{t,x}(t+h) - x\right\| \leq \text{const}(h + \sqrt{h})$ dès que b, σ, μ sont continus sur le compact $\overline{Q}$. [1]

Considérons $x \in S_{t,h}$. Notons $x^\pi \in \partial G$ la projection de x sur le bord de G et $n(x^\pi)$ le vecteur unitaire orthogonale à ∂G en x^π [2]. On note $\mu^{x,h}$ la loi du vecteur $A^{t,h}$ prenant les valeurs x^π et $x + \sqrt{h}\lambda n(x^\pi)$ avec probabilité $p = p_{x,h}$ et $q = 1 - p_{x,h}$. Nous choisissons alors $p_{x,h}$ de sorte à pouvoir écrire l'approximation $v(x) = \mathbb{E}\left[v(A^{t,h})\right]$ pour v une fonction lisse. On a ensuite le développement de Taylor suivant (ou l'on utilise $x^\pi - x = O(\sqrt{h})$) :

$$\begin{aligned}
\mathbb{E}\left[v(A^{t,h})\right] &= p_{x,h} v\left(x^\pi\right) + (1 - p_{x,h}) v\left(x + \sqrt{h}\lambda n\left(x^\pi\right)\right) \\
&= p_{x,h}\left(v\left(x\right) + {}^t(x^\pi - x)\nabla v(x)\right) + (1 - p_{x,h})\left(v\left(x\right) + \sqrt{h}\lambda {}^t n\left(x^\pi\right)\nabla v(x)\right) + O(\sqrt{h}) \\
&= v(x) + p_{x,h}\left({}^t(x^\pi - x)\nabla v(x) - \sqrt{h}\lambda {}^t n\left(x^\pi\right)\nabla v(x)\right) + \sqrt{h}\lambda {}^t n\left(x^\pi\right)\nabla v(x) + O(\sqrt{h}).
\end{aligned}$$

Pour annuler le terme de reste, on doit donc prendre :

$$p_{x,h}\left({}^t(x^\pi - x)\nabla v(x) - \sqrt{h}\lambda {}^t n\left(x^\pi\right)\nabla v(x)\right) = -\sqrt{h}\lambda {}^t n\left(x^\pi\right)\nabla v(x).$$

Cela nous suggère l'expression simplifiée suivante :

$$p_{x,h}\left(x + \sqrt{h}\lambda n\left(x^\pi\right) - x^\pi\right) = \sqrt{h}\lambda {}^t n\left(x^\pi\right).$$

Cette dernière équation impliquerait que $n(x^\pi)$ soit colinéaire à $\sqrt{h}\lambda n(x^\pi) - x^\pi$ ce qui n'est pas forcément vérifié en réalité. Cependant, c'est "presque" le cas pour h suffisament petit et ∂G suffisamment régulière. Le choix suivant pour $p_{x,h}$ semble donc approprié :

$$p_{x,h} = \frac{\sqrt{h}\lambda}{\left|x + \sqrt{h}\lambda n(x^\pi) - x^\pi\right|}. \tag{4}$$

1. En fait, cela reste vrai dans un cadre plus général. Par exemple, cela reste vrai si $\overline{Q}$ n'est pas compact mais de la forme $[t_0, T] \times]-\infty; a]$ avec a un réel.

2. $x\pi$ et $n(x^\pi)$ ne sont pas toujours bien définis. Ils le sont en revanche dès que G est suffisament lisse et h suffisamment petit.

Finalement, lorsque $\bar{X}^{t,x}(t_k)$ est dans $S_{t_k,h}$, on prend $A^{t,x}(t_k)$ de loi $\mu_{t_k,\bar{X}^{t,x}(t_k)}$. Lorsque $A^{t,x}(t_k) = \left(\bar{X}^{t,x}(t_k)\right)^{\pi}$, on arrête l'algorithme, et sinon, on continue en définissant $\bar{X}^{t,x}(t_{k+1})$ comme précédemment mais en remplacant $\bar{X}^{t,x}(t_k)$ par $A^{t,x}(t_k)$.

Deux algorithmes sont alors possibles. Dans le premier, l'algorithme s'arrête lorsque X entre dans la zone "proche" du bord tandis que dans le second, on continue en tirant A comme présenté précédemment. Le pseudo code détaillant ces algorithmes peut se trouver dans [KT12]. On peut alors montrer que :

Théorème 1 (Algorithme 1 - ordre $\frac{1}{2}$)**.** *Soit $\bar{\tau}$ le premier instant $t_k < T$ tel que $\bar{X}^{t,x}(t_k)$ soit dans $S_{t,h}$ et T s'il ne l'atteint jamais. On pose $\hat{X} = \left(\bar{X}^{t,x}(t_k)\right)^{\pi}$ si $\tau = t_k < T$ et $\hat{X} = \bar{X}^{t,x}(T)$ si $\tau = T$. Alors on a :*

$$u(t,x) = \mathbb{E}^{t,x}\left[\varphi\left(\bar{\tau}, \hat{X}\right) Y_{\bar{\tau}} + Z_{\bar{\tau}}\right] + O(\sqrt{h}).$$

Théorème 2 (Algorithme 2 - ordre 1)**.** *Soit $\bar{\tau}$ le premier instant t_k tel que $\bar{X}^{t,x}(t_k)$ soit dans $S_{t_k,h}$ et tel que $A^{t,x}(t_k) = \left(\bar{X}^{t,x}(t_k)\right)^{\pi}$, et T s'il n'en sort jamais. Alors on a :*

$$u(t,x) = \mathbb{E}^{t,x}\left[\varphi\left(\bar{\tau}, \bar{X}_{\bar{\tau}}\right) Y_{\bar{\tau}} + Z_{\bar{\tau}}\right] + O(h).$$

2 Un exemple jouet : les options barrieres sur Black and Scholes

Avant de passer au sujet principal du mémoire, nous avons souhaité implémenter ces algorithmes sur le modèle de Black and Scholes afin d'illustrer leur fonctionnement dans un cadre simplifié. En fait, il n'y a aucune différence conceptuelle avec le cadre des barrier caplets étudiés dans la section 3.2 puisqu'il n'y a alors qu'un libor sous-jacent. Cependant, le modèle de Black and Scholes étant beaucoup plus étudié, il permet de se familiariser avec ces algorithmes sans rentrer dans un formalisme important.

On considère un actif S ayant comme dynamique sous la probabilité risque neutre $\frac{dS_t}{S_t} = rdt + \sigma dW_t$ avec r le taux sans risque positif, σ une constante strictement positive et W_t un $(\mathcal{F}_t)_t$ mouvement brownien.

Dans ce modèle, les prix des options barrières les plus simples admettent des formules fermées. Par exemple, considérons une option de type Down and In Call. Il s'agit d'un contrat identique à un Call classique mais qui est activé uniquement si le titre sous-jacent franchit une barrière H à la baisse (à noter que si $S_0 < H$ nous avons simplement un Call). Le payoff de cette option est donc $(S_T - K)_+ \mathbf{1}_{\tau_u < T}$ où $\tau_u = \inf\{t \geqslant 0, S_t \leqslant H\}$. Le prix d'une telle option est donné par la proposition suivante.

Proposition 3. *Considérons une option DIC regular ($H < K$) sur un sous-jacent ne versant aucun dividende. On suppose que le taux d'interêt r est nul. Son prix est donné par :*

- *Si $x \leqslant H$, alors* $\mathrm{DIC}(x,K,H) = \mathrm{Call}(x,K)$.
- *Si $x \geqslant H$, alors* $\mathrm{DIC}(x,K,H) = \frac{K}{H}\,\mathrm{Put}\left(x, \frac{H^2}{K}\right) = \mathrm{Call}\left(H, \frac{xK}{H}\right)$.

Preuve. Lorsque la valeur x du sous-jacent (à la date t) est inférieure à la barrière H, la contrainte est réalisée, l'option est donc une option vanille classique, d'où le résultat.

Supposons maintenant que la valeur du sous-jacent (à la date t) est supérieure à la barrière, nous pouvons par arbitrage choisir d'évaluer l'option à la date τ_u pour $\tau_u <$ T puis donner un prix en t pour ce flux aléatoire payé en τ_u . A la barrière, le niveau du sous-jacent est connu, seule la maturité restante $T - \tau_u$ est aléatoire et l'option DIC est équivalente à un $\mathrm{Call}(\tau_u, H, K, T)$. Comme le sous-jacent est martingale et la volatilité déterministe, la dynamique du sous-jacent initialisée au temps aléatoire τ_u et au point H, est, conditionnellement à l'observation du passé jusqu'en τ_u à distribution log-normale. Puis, La formule de parité Call/Put et l'homogénéité du prix du Put montrent que

$$\mathrm{Call}(\tau_u, H, K) = \mathrm{Put}(\tau_u, K, H) = \frac{K}{H}\,\mathrm{Put}\left(\tau_u, H, \frac{H^2}{K}\right).$$

On remarque que l'option qui à la barrière vaut $\mathrm{Put}\left(\tau_u, H, \frac{H^2}{K}\right)$ est un (Down and In Put) $\mathrm{DIP}\left(\frac{H^2}{K}, H\right)$ et donc l'option $\mathrm{DIC}(x,K,H)$ est équivalente à $\frac{K}{H}$ options $\mathrm{DIP}(\frac{H^2}{K}, H)$. A l'échéance, le Put a de la valeur seulement si le sous-jacent est inférieur à $\frac{H^2}{K}$ quantité inférieure à H puisque $H \leqslant K$. La barrière H a

donc été atteinte durant la vie de l'option avec une probabilité 1 ; la barrière n'a donc plus d'influence sur le prix. L'option barrière $\mathrm{DIP}(x, \frac{H^2}{K}, H)$ est donc égale à un $\mathrm{Put}(x, \frac{H^2}{K})$ pour $H \leqslant K$. On a donc

$$\mathrm{DIC}(x, K, H) = \frac{K}{H}\,\mathrm{DIP}\left(x, \frac{H^2}{K}, H\right) = \frac{K}{H}\,\mathrm{Put}\left(x, \frac{H^2}{K}\right) = \mathrm{Call}\left(H, \frac{xK}{H}\right).$$

□

Supposons par exemple que nous voulions estimer le prix d'un Down and In Call regular en $t = 0$ de parametres fixés, par exemple : $S_0 = 100$, $T = 1$, $K = 100$, $r = 0$, $\sigma = 0.2$ et $H = 95$. Le calcul du prix avec la formule explicite donnée par la proposition 3 donne 3.8667.

Remarquons tout d'abord que nous ne sommes pas dans le cadre de la partie 1 puisque le payoff depend de l'observation finale et ne s'arrête pas lorsque la barrière est franchie. Néanmoins, on peut les adapter et les implémenter. Finalement, nous avons implémenté ces différents algorithmes :

1. Utilisation de la formule explicite avec un pont brownien pour vérifier l'activation de la barrière.
2. Utilisation de la formule explicite $S_t = S_0 \exp\left(\sigma W_t + \left(r - \frac{\sigma^2}{2}\right) t\right)$ pour discrétiser le cours de l'actif et regarder si celui-ci active la barrière. (Cette simulation n'est pas exacte car on n'utilise pas le pont brownien ce qui induit un biais de simulation)
3. Algorithme d'Euler classique.
4. Algorithme d'Euler classique en remplaçant la loi Gaussienne par des lois uniformes sur $\{-1, +1\}$.

Remarquons que seul la première méthode est une simulation exacte du payoff et donc sans biais. Les autres méthodes peuvent ne pas détecter certaines activations de la barrière.

Table 1 – Simulation du prix d'un DIC avec $S_0 = 100$, $T = 1$, $K = 100$, $r = 0$, $\sigma = 0.2$ et $H = 95$

Algorithme	h	Resultat	Erreur	Variance	simulations	temps
Algo 1	$\varnothing$	3.8707	± 0.0049999	79.661	12241000	1.407 s
Algo 2	$h = 2 \cdot 10^{-2}$	2.9615	± 0.0049997	59.456	9137000	29.731 s
Algo 3	$h = 2 \cdot 10^{-2}$	2.9446	± 0.0049999	58.724	9024000	22.551 s
Algo 4	$h = 2 \cdot 10^{-2}$	3.3785	± 0.0049999	68.514	10528000	17.888 s
Algo 2	$h = 10^{-3}$	3.6459	± 0.0049999	74.481	11445000	782.4 s
Algo 3	$h = 10^{-3}$	3.6402	± 0.0049999	74.356	11426000	519.44 s
Algo 4	$h = 10^{-3}$	3.7117	± 0.0049998	75.994	11678000	350.07 s

Nous pourrions également appliquer ici des méthodes de réduction de variances. Par exemple, la méthode antithétique pourrait s'appliquer en changeant les lois normales par leurs opposés et les loi uniformes U sur $[0, 1]$ par $1 - U$. Nous détaillerons ces méthodes sur les exemples suivant concernant le LIBOR.

Nous allons conclure cette partie en remarquant que bien qu'inexacte, la simulation utilisant des lois uniformes sur $\{-1, +1\}$ permet d'une part de réduire les temps de calculs grâce à une exécution plus rapide [3] mais aussi en réduisant le biais !

3 Les options barrières dans le modèle LMM

3.1 Présentation du modèle

On se place à présent dans un marché sans opportunité d'arbitrage, sans friction et où les échanges sont effectués en continue. On se place dans un cadre à horizon fini $[t_0, T]$. Le LIBOR (London Interbank Offered Rate) est un taux d'intérêt qui est déterminé chaque jour pour plusieurs maturités en effectuant un benchmark quotidien auprès des banques londoniennes.

On définit le **taux LIBOR forward** taux LIBOR forward noté $L_t(T, T + \delta)$ comme le taux intérêt défini en $t < T$ pour la période $[T, T + \delta]$. Ainsi si on entre dans le contrat en t pour emprunter en T et rembourser en $T + \delta$, l'intérêt appliqué sera $\delta L_t(T, T + \delta)$.

3. Il est possible de réaliser environ 50% plus de simulation avec l'algorithme d qu'avec le c.

Le Libor Market Model, abrégé LMM est un modèle permettant de modéliser le taux LIBOR forward comme un modèle log-normal sous la probabilité forward-neutre.

Commençons par introduire certaines notations. On notera $P(t,T)$ le prix du zéro-coupon payé en t donnant le payoff 1 en T. On notera $P_t(T,T+\delta)$ le zéro-coupon forward de maturités T et $T+\delta$. Par absence d'opportunités d'arbitrage, on peut montrer que

$$P_t(T,T+\delta) = \frac{P(t,T+\delta)}{P(t,T)}.$$

On définit ensuite $L(t,T)$ le taux LIBOR en t de maturité T comme le taux linéaire associé au zéro-coupon et on définit de la même manière $L_t(T,T+\delta)$ à partir de $P_t(T,T+\delta)$ de sorte que : $L_t(T,T+\delta) = \frac{1}{\delta}\left(P_t(T,T+\delta)-1\right)$. On note également $\mathbb{Q}^T$ la probabilité T-forward, c'est-à-dire la probabilité associée au numéraire $P(t,T)$.

Nous pouvons maintenant enfin définir le modèle LMM. Considérons $T_0 < T_1 < \cdots < T_N = T^*$ une suite de maturités. On notera pour simplifier les écritures $L^i(t) = L_t(T_i, T_{i+1})$. Soit $0 \leq k, i < N$. Le modèle LMM se base sur l'hypothèse que la dynamique de $L^i(t)$ peut s'écrire pour $t \leq \inf(T_i, T_k)$

$$\frac{dL^i(t)}{L^i(t)} = \begin{cases} \sigma^i(t)dW_i^{T_{k+1}} + \sigma^i(t)\sum_{j=k+1}^{i} \frac{\delta L^j(t)}{1+\delta L^i(t)}\rho_{i,j}\sigma_j(t)dt & \text{si } i > k \\ \sigma^i(t)dW_i^{T_{k+1}} & \text{si } i = k \\ \sigma^i(t)dW_i^{T_{k+1}} - \sigma^i(t)\sum_{j=i+1}^{k} \frac{\delta L^j(t)}{1+\delta L^i(t)}\rho_{i,j}\sigma_j(t)dt & \text{si } i < k \end{cases} \tag{5}$$

avec $\left(W_i^{T_{k+1}}\right)_i$ un mouvement brownien de dimension N, centré et de matrice de corrélation $\rho = (\rho_{i,j})_{i,j}$ sous $\mathbb{Q}^{T_{k+1}}$.

Afin de pouvoir simuler ce processus, on décompose ρ sous la forme : $\rho = U\,{}^tU$ ou U est une matrice triangulaire supérieure. Soit $B^{T_{k+1}}(t)$ un mouvement brownien standard de dimension N tel que :$dW^{T_{k+1}} = UdB^{T_{k+1}}$.

De cette manière, on voit que l'équation 5 peut s'écrire sous la forme de la 1. On peut ensuite ajouter les paramètres supplémentaires F et μ comme dans l'équation 2 en prenant soin de les ajouter aux équations définies à partir de B. Mais $dB^{T_{k+1}} = U^{-1}dW^{T_{k+1}}$ donc quitte à remplacer F et μ (dans 2) par ${}^tUF$ et ${}^tU\mu$ (dans 5), on peut garder la même expression avec W.[4]. On peut également s'assurer par le calcul que l'équation que doivent vérifier F et μ pour avoir une variance nulle reste inchangée (tant que U est inversible).

Dans la suite, nous supposerons toujours que les différents incréments de maturités sont équidistants et on note $\delta = \frac{T^*-T_0}{N}$ cet incrément. Nous supposerons également pour simplifier que $\rho_{i,j} = \exp\left(-\beta|T_i - T_j|\right)$.

3.2 Les barrier cap

Définition 1. *Un caplet est un Call sur les prix LIBORs. Il est caractérisé par une maturité T, une date de paiement $T+\delta$ et un strike K. Le payoff, recu en $T+\delta$ est $\delta(L(T,T+\delta)-K)_+$.*

Un barrier caplet est un cap qui est désactivé si le taux Libor forward $L_t(T,T+\delta)$ atteint par le dessous un certain taux barrière H.

Enfin, un (barrier) cap est une succession de (barrier) caplets.

On peut définir de même des (barrier) floorlets et floor qui sont définis à partir de put plutôt que de call. Cependant, nous nous concentrerons sur les caps par la suite. Le pricing des (barrier) caps revient à faire la somme des pricing des caplets donc nous nous interesserons uniquement aux pricing des caplets.

Notons $V(t)$ la valeur d'un barrier caplet de maturité T, payé en $T+\delta$, de strike K et de barrière H. Notons τ le temps de sortie de $]0,H[$ arreté à T. On a alors :

$$\begin{aligned} V(t) &= \mathbb{E}\left[\exp\left(-\int_t^T r_s ds\right)\delta\left(L(T,T+\delta)-K\right)_+ \mathbb{1}_{\tau\geq T}\right] \\ &= \delta P(t,T+\delta)\mathbb{E}^{\mathbb{Q}^{T+\delta}}\left[\left(L(T,T+\delta)-K\right)_+ \mathbb{1}_{\tau\geq T}\right] \end{aligned}$$

4. On verifiera que dans l'expression de L, le terme $\sigma\mu$ est le terme correspondant car la fonction σ de 2 correspond à $U\sigma$ dans le modele LLM.

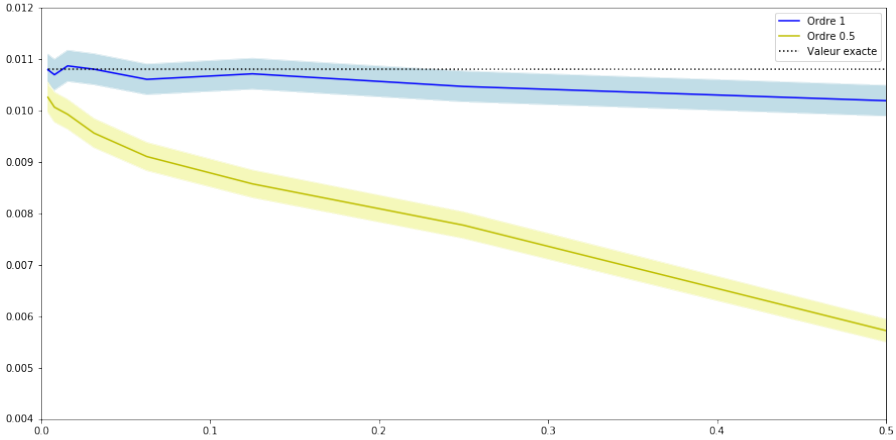

FIGURE 1 – Estimations Monte Carlo du prix d'un caplet

où r désigne le taux d'intérêt. Ce dernier étant inconnu, nous priviligerons la seconde égalité et nous calculerons plutôt le prix forward normalisé par δ, noté $\widetilde{V}(t) := \frac{V(t)}{\delta P(t,T+\delta)}$. On peut ensuite réécrire $\widetilde{V}$ de la manière suivante : $\widetilde{V}(t) = \mathbb{E}^{\mathbb{Q}^{T+\delta}}[\varphi(\tau, L_\tau(T, T+\delta))]$ avec $\varphi(t,x) = (x-K)_+ \mathbb{1}_{t=T}$ afin de rentrer dans le cadre de la partie 1.

Notons $L(t) = L_t(T, T+\delta)$. Afin de préserver la positivité du taux Libor dans le modèle LMM, nous allons simuler $\ln(L(t))$ plutôt que $L(t)$. Ce processus satisfait

$$d\ln(L(t)) = \sigma(t) dW^{T+\delta}(t) - \frac{\sigma^2(t)}{2} dt.$$

Nous appliquons maintenant l'agorithme des marches aléatoires etudié à la section 1.4.
On a alors :

$$\overline{\ln(L)}(t_{k+1}) = \overline{\ln(L)}(t_k) - \frac{\sigma^2(t_k)}{2} h + \sigma(t_k)\xi_k\sqrt{h}.$$

De plus, on voit que si $\overline{\ln(L)}(t_k) \leq \ln H - \sqrt{h}\lambda$, alors $\overline{\ln(L)}(t_{k+1}) \leq \ln H + (\|\sigma\| - \lambda)\sqrt{h} < lnH$ dès que $\lambda > \|\sigma\|$. En fait, si on fait dependre λ de l'iteration actuelle, on peut prendre :

$$\lambda_k = \frac{-\sigma^2(t_k)\sqrt{h}}{2} + \sigma(t_k).$$

Nous avons alors appliqué les deux algorithmes vus dans la section 1.4 avec comme parmètres $L(0) = 15\%$, $H = 20\%$, $K = 5\%$, σ constant égal à 25% et une maturité $T = 10$. Les résultats pour différents pas de discrétisation h pour 10^5 simulations sont données sur la figure 1.

On vérifie bien ici que l'algorithme 1 converge bien plus lentement que l'algorithme 2. Par conséquent, nous nous concentrerons par la suite sur le second. Il est également remarquable de voir les oscillations autour de la vraie valeur pour l'algorithme 2. Celles-ci sont typiques des algorithmes basé sur des marches aléatoires. Remarquons que dans cet exemple, la valeur exacte de $\widetilde{V}$ est connue et est égale à 0.01079. En effet, on dispose du résultat suivant :

Théorème 3. *Notons* $\delta_\pm(x) := \frac{1}{v}\left(\ln(x) \pm \frac{v^2}{2}\right)$ *ou* $v^2 = \int_t^T \sigma(s)^2 ds$. *Alors on a :*

$$\begin{aligned}\widetilde{V}(t) = {} & L(t)\mathcal{N}\left(\delta_+\left(\frac{L(t)}{K}\right)\right) - L(t)\mathcal{N}\left(\delta_+\left(\frac{L(t)}{H}\right)\right) - K\mathcal{N}\left(\delta_-\left(\frac{L(t)}{K}\right)\right) + K\mathcal{N}\left(\delta_-\left(\frac{L(t)}{H}\right)\right) \\ & + \frac{KL(t)}{H}\mathcal{N}\left(\delta_-\left(\frac{H^2}{KL(t)}\right)\right) - \frac{KL(t)}{H}\mathcal{N}\left(\delta_-\left(\frac{H}{L(t)}\right)\right) - H\mathcal{N}\left(\delta_+\left(\frac{H^2}{KL(t)}\right)\right) + H\mathcal{N}\left(\delta_+\left(\frac{H}{L(t)}\right)\right)\end{aligned}$$

Preuve. Il n'est pas possible d'utiliser ici un raisonnement similaire à celui de la proposition 3 puisque l'option est ici non regulière. On est dans le cadre une option d'achat mais la barrière vérifie $K < H$.

Il s'agit donc de pricer le payoff $(L(T)-K)_+\mathbb{1}_{\tau>T}$. Commençons par étudier $(L(T)-K)_+\mathbb{1}_{\tau\leq T}$. En utilisant le fait que $\tau\leq T$ dès que $L(T)\geq H$, on a :

$$\begin{aligned}(L(T)-K)_+\mathbb{1}_{\tau\leq T} &= (L(T)-K)_+\mathbb{1}_{S_T\geq H} + (L(T)-K)_+\mathbb{1}_{\tau\leq T, S_T<H}\\ &= (L(T)-H)_+ + (H-K)\mathbb{1}_{S_T\geq H} + \left((H-K)-(H-L(T))_+ + (K-L(T))_+\right)\mathbb{1}_{\tau\leq T, S_T<H}.\end{aligned}$$

On utilise ensuite le fait que $(H-L(T))_+$ et $(K-L(T))_+$ sont nuls si $S_T\geq H$ et de plus que $(L(T)-K)_+\mathbb{1}_{\tau<T} = (L(T)-K)_+ - (L(T)-K)_+\mathbb{1}_{\tau\leq T}$. On obtient alors en prenant l'espérance :

$$\begin{aligned}\widetilde{V}(t) = \mathrm{Call}(L(t),K) - \mathrm{Call}(L(t),H) &- (H-K)\mathbb{Q}^{T+\delta}(S_T\geq H)\\ &- (H-K)\mathbb{Q}^{T+\delta}(\tau\leq T, S_T<H)\\ &+ \mathrm{UIP}(L(t),H,H) - \mathrm{UIP}(L(t),K,H)\end{aligned}$$

où $\mathrm{UIP}(L(t),K,H)$ désigne une option barrière de type knock-in à la vente de strike K et de barrière H. Ces options sont régulières et on peut montrer que $\mathrm{UIP}(L(t),K,H) = \mathrm{Call}(\frac{KL(t)}{H},H)$ en suivant 3. Le calcul de $\mathbb{Q}^{T+\delta}(S_T\geq H)$ est standard et on peut calculer $\mathbb{Q}^{T+\delta}(\tau\leq T, S_T<H)$ en remarquant que :

$$\frac{d}{dK}\left(\mathrm{UIP}(L(t),K,H)\right) = \frac{d}{dK}\left(\mathbb{E}^{\mathbb{Q}^{T+\delta}}\left[(K-L(T))_+\mathbb{1}_{\tau\leq T}\right]\right) = \mathbb{E}^{\mathbb{Q}^{T+\delta}}\left[\mathbb{1}_{K\geq L(T)}\mathbb{1}_{\tau\leq T}\right]$$

puis en faisant tendre K vers H. Ainsi, on a :

$$\mathbb{Q}^{T+\delta}(\tau\leq T, S_T<K) = \frac{d}{dK}\left(\mathrm{Call}(\frac{KL(t)}{H},H)\right) = \frac{L(t)}{H}\Delta_{\mathrm{Call}(\frac{KL(t)}{H},H)}.$$

On conclut en regroupant tous les termes et en utilisant la formule de Black and Scholes. □

Nous allons maintenant implémenter les différentes méthodes de réduction de variables. Au vu de 2, si $F = -\sigma L\cdot\frac{\partial\widetilde{V}}{\partial L}$, la simulation devrait être exacte (sans variance). Compte-tenu de la discrétisation en temps réalisé, la simulation ne le sera pas, mais on peut donc s'attendre à une erreur très faible. On modifie ensuite la simulation afin d'y intégrer F. Concrêtement, on introduit Z de dynamique donnée par :

$$dZ_t = F(t,L(t))\,dW^{T+\delta}(t).$$

En dérivant par rapport à $L(t)$ l'expression obtenue par dans le théorème 3, on obtient :

$$\begin{aligned}\frac{\partial\widetilde{V}}{\partial L}(t) = \mathcal{N}\left(\delta_+\left(\frac{L(t)}{K}\right)\right) - \mathcal{N}\left(\delta_+\left(\frac{L(t)}{H}\right)\right) + \frac{K}{H}\mathcal{N}\left(\delta_-\left(\frac{H^2}{KL(t)}\right)\right) - \frac{K}{H}\mathcal{N}\left(\delta_-\left(\frac{H}{L(t)}\right)\right)\\ + \frac{2(K-H)}{vH}\phi\left(\delta_+\left(\frac{L(t)}{H}\right)\right)\end{aligned}$$

où ϕ désigne la densité d'une loi normale centrée réduite. On utilise notamment les identités suivantes pour simplifier le calcul : $x\phi\left(\delta_+(x)\right) = \phi\left(\delta_+(\frac{1}{x})\right)$, $\delta'_+(x) = \delta'_-(x) = \frac{1}{vx}$ et $\phi(\delta_-(x)) = x\phi(\delta_+(x))$. Cette expression permet de mettre en place la réduction de variance présentée précédemment. En utilisant les mêmes paramètres que précédemment, on obtient les résultats des tables 2 et 3.

Table 2 – Simulation du prix d'un caplet sans l'utilisation du F

h	Resultat	Erreur	Variance	simulations	temps
$h=0.1$	0.010556	$\pm 4.998\cdot 10^{-5}$	0.00057372	882000	1.935 s
$h=10^{-2}$	0.010784	$\pm 4.998\cdot 10^{-5}$	0.00058412	898000	17.153 s
$h=10^{-3}$	0.010771	$\pm 4.999\cdot 10^{-5}$	0.00058236	895000	168.98 s

On voit que la réduction de variance basée sur F fonctionne parfaitement. En effet, celle-ci permet de la diminuer d'un facteur 10 environ pour un pas de discrétisation élevé et d'un facteur 1000 pour les pas plus petits. Cela se traduit donc par des gains de temps de calculs importants, surtout pour des pas petits. Dans le cas des pas plus large, un gain de temps assez faible est obtenu car le temps de calcul

TABLE 3 – Simulation du prix d'un caplet en utilisant F

h	Resultat	Erreur	Variance	simulations	temps
$h = 0.1$	0.010586	$\pm 4.997 \cdot 10^{-5}$	$3.4462 \cdot 10^{-5}$	53000	1.438 s
$h = 10^{-2}$	0.010741	$\pm 4.990 \cdot 10^{-5}$	$8.1047 \cdot 10^{-6}$	12500	3.286 s
$h = 10^{-3}$	0.010782	$\pm 4.930 \cdot 10^{-5}$	$6.3292 \cdot 10^{-7}$	1000	2.544 s

de la fonction F est relativement long. Cela illustre bien la remarque précédente : lorsque nous utilisons la fonction F, la variance de la simulation provient uniquement de l'erreur due à la discrétisation de l'intervalle. Elle diminue donc fortement lorsque le taille de l'intervalle diminue.

Nous avons pouvons également appliquer la méthode antithétique que nous avions présentée à la section 1.2. Les résultats sont présentés sur la table 4.

TABLE 4 – Simulation du prix d'un caplet avec la méthode antithétique

h	Resultat	Erreur	Variance	simulations	temps
$h = 0.1$	0.010541	$\pm 4.997 \cdot 10^{-5}$	0.00026265	404000	1.354 s
$h = 10^{-2}$	0.010795	$\pm 4.995 \cdot 10^{-5}$	0.00026763	412000	13.418 s
$h = 10^{-3}$	0.010813	$\pm 4.995 \cdot 10^{-5}$	0.00026763	412000	134.23 s

Ainsi, on observe que même si la croissance du payoff n'est pas vérifiée, la méthode antithétique permet un gain de temps de calcul. Cependant, même si la variance asymptotique est divisée par deux, le temps de calcul n'est réduit que de 20% environ. La perte de temps observée est due à une augmentation du nombre de calculs réalisés pour calculer le payoff antithétique.

Finalement, il est possible de combiner la méthode antithétique et l'utilisation de F, permettant d'améliorer de nouveau les temps de calcul. (table 5)

TABLE 5 – Simulation du prix d'un caplet avec la méthode antithétique et en utilisant F

h	Resultat	Erreur	Variance	simulations	temps
$h = 0.1$	0.010555	$\pm 4.994 \cdot 10^{-5}$	$1.8672 \cdot 10^{-5}$	28750	1.533 s
$h = 10^{-2}$	0.010804	$\pm 4.983 \cdot 10^{-5}$	$4.7195 \cdot 10^{-6}$	7300	3.813 s
$h = 10^{-3}$	0.010767	$\pm 4.828 \cdot 10^{-5}$	$6.0698 \cdot 10^{-8}$	100	485 ms

On peut également comparer ces résultats à une simulation de Monte-Carlo classique, c'est à dire basée sur des aléas gaussiens. Avec ou sans ponts gaussiens [5]. Remarquons tout d'abord que sans pont gaussien, les algorithmes sont davantage biaisés que sans. En revanche, il reste un biais relativement important dans le cas du pont brownien. Celui-ci provient du fait que contrairement au cas de la section sur le modele de Black & Scholes, les caplets sont non-regulier, donc plus difficile à pricer. Il est également ici possible d'appliquer la méthode antithétique qui permet des gains de temps d'environ 20% et il est également possible d'utiliser la fonction F présentée précedemment. (tables 6 et 7). Enfin, on peut implémenter la méthode du pont Brownien. Cela diminue drastiquement le biais induit par la méthode classique de simulation (dans le cadre du bruit Gaussien). Les résultats ne sont pas présentés, mais sont reproductibles avec notre programme.

Finalement, on remarque que les simulations se basant sur une marche aléatoire sont non seulement moins biaisées, mais elles sont également plus rapides en pratique (en grande partie car les variables uniformes discrètes sont plus simples à simuler que les variables gaussiennes).

3.3 Les barrier swaption

On se place en $t_0 \leqslant T_0$ et on reprend les notations de la section 3.1. Un swap est un échange entre un taux variable et un taux fixe sur chaque intervalle de la forme $[T_i, T_{i+1}]$. En t_0, lorsque l'on signe le

5. Nous nous sommes contentés d'un pont brownien à chaque itération de temps. Bien sûr, dans le cadre du caplet, il est possible de le faire sans discrétiser l'intervalle comme dans la section 2. Cependant, notre méthode se généralisant à des payoffs plus compliqués, nous avons décidé de la garder.

TABLE 6 – Simulation Brownienne du prix d'un caplet avec la méthode antithétique sans utiliser F

h	Resultat	Erreur	Variance	simulations	temps
$h = 0.1$	0.013239	$\pm 4.996 \cdot 10^{-5}$	0.00035356	544000	2.618 s
$h = 10^{-2}$	0.011618	$\pm 4.996 \cdot 10^{-5}$	0.00029445	453000	20.823 s
$h = 10^{-3}$	0.01106	$\pm 4.997 \cdot 10^{-5}$	0.00027495	423000	191.62 s

TABLE 7 – Simulation Brownienne du prix d'un caplet avec la méthode antithétique et en utilisant F

h	Resultat	Erreur	Variance	simulations	temps
$h = 0.1$	0.013257	$\pm 4.998 \cdot 10^{-5}$	$4.7764 \cdot 10^{-5}$	73450	4.631 s
$h = 10^{-2}$	0.011578	$\pm 4.988 \cdot 10^{-5}$	$8.066 \cdot 10^{-6}$	12450	7.213 s
$h = 10^{-3}$	0.011015	$\pm 4.874 \cdot 10^{-5}$	$4.3293 \cdot 10^{-7}$	700	3.842 s

contrat, le taux fixe appelé **taux de swap** est choisi tel que le prix du contrat soit nul. On peut montrer que celui-ci verifie :

$$R_{swap}(t) = \frac{1 - \frac{1}{\prod_{j=0}^{N-1} 1+\delta L_t(T_j,T_{j+1})}}{\delta \sum_{i=0}^{N-1} \frac{1}{\prod_{j=0}^{i} 1+\delta L_t(T_j,T_{j+1})}}.$$

R_{swap} peut être vu comme une quantité de marché même si le contrat n'est pas échangeable sur le marché. Un swaption européen est une option qui donne à son détenteur le droit d'entrer dans un swap à la maturité et à un taux fixé K et qui paye $\delta(R_{swap}(T_0) - K)_+$ en chaque T_j , $j \geqslant 1$.Cela revient à considérer le payoff $\delta(R_{swap}(T_0) - K)_+ \sum_{j=0}^{N} P(T_0, T_j)$ en T_0. Un swaption barrière est un contrat identique à un swaption avec en plus une barrière (activante ou désactivante) portant sur le cours du R_{swap}. Nous allons nous intéresser aux swaptions barrières désactivante.

Un swaption *knockout* est un swaption standard à condition que le taux de swap sous-jacent reste au dessus d'une barrière R_{up} jusqu'à la maturité T_0. Autrement dit, le payoff d'un swaption peut s'écrire sous la forme $\delta(R_{swap}(T_0) - K)_+ \sum_{j=0}^{N} P(T_0, T_j)\mathbb{1}_{\theta>0}$ ou $\theta := \inf\{t \leqslant 0 : R_{swap}(t) \leqslant R_{up}\}$. Le prix d'un swaption *knockout* au temps $t = 0$ sous la probabilité forward neutre Q^{T_0} est donné par

$$V_{swaption}(0) = P(0,T_0)\mathbb{E}_{\mathbb{Q}^{T_0}}\left[\delta(R_{swap}(T_0) - K)_+ \sum_{j=1}^{N} P(T_0,T_j)\mathbb{1}_{\theta>T_0}\right] = \delta P(0,T_0)\widetilde{V}_{swaption}(0),$$

$$\widetilde{V}_{swaption}(0) = \mathbb{E}_{\mathbb{Q}^{T_0}}\left[(R_{swap}(T_0) - K)_+ \sum_{j=1}^{N} P(T_0,T_j)\mathbb{1}_{\theta>T_0}\right].$$

On rappelle que les dynamiques de $L^i(t)$ sous Q^{T_0} sont données par :

$$\frac{dL^i(t)}{L^i(t)} = \sigma^i(t)dW_i^{T_0}(t) + \sigma^i(t)\sum_{j=0}^{i} \frac{\delta L^j(t)}{1+\delta L^j(t)}\rho_{i,j}\sigma_j(t)dt,\ i = 0, \cdots, N-1.$$

De plus on peut montrer que

$$\begin{aligned}
\frac{\widetilde{V}_{swaption}(0)}{\sum_{j=1}^{N} P_0(T_0,T_j)} = {} & R_{swap}(0)\left[\mathcal{N}\left(\delta_+\left(\frac{R_{swap}(0)}{K}\right)\right) - \mathcal{N}\left(\delta_+\left(\frac{R_{swap}(0)}{R_{up}}\right)\right)\right] \\
& - K\left[\mathcal{N}\left(\delta_-\left(\frac{R_{swap}(0)}{K}\right)\right) - \mathcal{N}\left(\delta_-\left(\frac{R_{swap}(0)}{R_{up}}\right)\right)\right] \\
& - H\left[\mathcal{N}\left(\delta_+\left(\frac{R_{up}^2}{KR_{swap}(0)}\right)\right) - \mathcal{N}\left(\delta_+\left(\frac{R_{up}}{R_{swap}(0)}\right)\right)\right] \\
& + \frac{KR_{swap}(0)}{R_{up}}\left[\mathcal{N}\left(\delta_-\left(\frac{R_{up}^2}{KR_{swap}(0)}\right)\right) - \mathcal{N}\left(\delta_-\left(\frac{R_{up}}{R_{swap}(0)}\right)\right)\right]
\end{aligned}$$

où on note $\delta_\pm(x) = \frac{\ln(x)}{v_{R_{swap}}} \pm \frac{v_{R_{swap}}}{2}$avec $v^2_{R_{swap}} = \int_0^{T_0} (\sigma_{R_{swap}}(s))^2 ds$ où $\sigma_{R_{swap}}(s)$ est la volatilité instantané de la dynamique log-normale du taux de swap.

Bien que nous n'ayons pas de formule explicite pour $v^2_{R_{swap}}$, nous disposons d'une formule approchée, dite de Rebonato qui est :

$$v^{LMM}_{R_{swap}} = \sqrt{\sum_{i,j=0}^{N-1} \frac{\omega_i(0)\omega_j(0)L^i(0)L^j(0)\rho_{ij}}{R_{swap}(0)^2} \int_0^{T_0} \sigma_i(s)\sigma_j(s)ds} \text{ où } \omega_i(0) = \frac{1/\prod_{j=0}^{i}(1+\delta L^j(0))}{\delta \sum_{k=0}^{N-1} 1/\prod_{j=0}^{k}(1+\delta L^j(0))}.$$

Passons maintenant à l'algorithme. Comme précédemment, on simule $\ln(L_i(t))$. Par la formule d'Ito, on a $d\ln(L_i(t)) = \frac{dL_i(t)}{L_i(t)} - \frac{\sigma_i(t)^2 dt}{2}$, donc on utilisera les approximations :

$$\overline{\ln(L_i)}(t_{k+1}) = \overline{L_i}(t_k) + \sigma_i(t_k) \sum_{j=0}^{i} \frac{\delta \exp(\overline{\ln(L_i)}(t_k))}{1+\delta \exp(\overline{\ln(L_i)}(t_k))} \rho_{ij}\sigma_j(t_k)h - \frac{1}{2}\sigma_i(t_k)^2 h + \sigma_i(t_k)\sqrt{h}\xi_{k,i}$$

où $((\xi_{k,i})_i)_k$ est une suite iid de vecteurs aléatoires définies par $\xi_k = {}^tU \cdot \eta_k$ avec $\eta_k \sim \mathcal{U}(\{-1,1\})$. Posons $M_k := \max_i \overline{\ln(L_i)}(t_k)$ et $\widetilde{M_k} := M_k + \sigma^2_{max} hN + \sigma_{max}\sqrt{hN}$. On cherche a conditionner pour savoir quand il est possible de sortir de la zone admissible. La condition s'écrit

$$R_{swap}((\exp(\overline{\ln(L_i)}(t_k)) \cdot (1 + \sigma_i(t_k)\sigma_{max,k}h + \sigma_i(t_k)\sqrt{hN}))_{0 \leqslant i \leqslant N-1}) < R_{up}$$

avec $\sigma_{max,k} = \max_j \sigma_j(t_k)$. Cette condition est difficile à vérifier. Cependant, on remarque qu'elle est verifiée dès que $R_{swap}(\widetilde{M_k}, ..., \widetilde{M_k}) = \widetilde{M_k} < \ln(R_{up})$ l'est aussi. Quand l'une de ces conditions est bien vérifiée, on simule $\overline{\ln(L_i)}(t_{k+1})$ comme expliqué précédemment. Sinon, on suit la méthode des marches aléatoires introduite dans la section 1.4. Pour cela, il faut commencer par chercher la projection notée $\overline{\ln(L_i)}(t_k)^\pi$ du vecteur $\overline{\ln(L_i)}$ sur la surface définie par $\ln(R_{swap}(...)) = \ln(R_{up})$.

Autrement dit, on recherche le point $x = (x_0, \cdots, x_{N-1})$ tel que si y désigne le vecteur $\ln(\bar{L}_i)(t_k)$, la quantité $\|x-y\|$ soit minimale et le taux de swap forwards induit par les taux libors forwards $(\exp(x_0), \cdots, \exp(x_{N-1}))$ soit égal à R_{up}. On peut donc paramétriser de nouveau selon x_0 en écrivant :

$$x_0 = \psi(x) := \ln\left(\frac{R_{up}\delta \cdot \left(1 + \sum_{i=1}^{N-1} \prod_{j=i}^{N-1} (1+\delta e^{x_j})\right) + 1}{\delta \prod_{j=1}^{N-1}(1+\delta e^{x_j})} - \frac{1}{\delta} \right).$$

Après cette nouvelle paramétrisation, il faut maintenant minimiser une fonction du type $\phi(x) = \phi(x_1, \cdots, x_{N-1}) := (\psi(x) - y_0)^2 + \sum_{i=1}^{N-1}(x_i - y_i)^2$.

Pour effectuer cette minimisation, il faut calculer le gradient de ϕ, donc celui de ψ. En fait, il faut commencer par celui de e^ψ. Pour $1 \leq k \leq N-1$, on a :

$$\partial_k e^\psi(x) = -e^{x_k} \frac{R_{up}\delta \left(1 + \sum_{i=k+1}^{N-1} \prod_{j=i}^{N-1} (1+\delta e^{x_j})\right) + 1}{(1+\delta e^{x_k}) \prod_{j=1}^{N-1}(1+\delta e^{x_j})}$$

Puis :

$$\partial_k \phi(x) = 2\frac{\partial_k e^\psi(x)}{e^\psi(x)}(\psi(x) - y_0) + 2(x_k - y_k)$$

On applique ensuite l'algorithme de descente de gradient pour trouver le minimum. Cet algorithme n'est pas optimal, mais fonctionne bien tant que nous restons dans des conditions initiales classiques. Par exemple, cet algorithme dégénère si l'on prend des courbes des taux très pentue. (typiquement, $L^0 = 0.01$ et $L^i = 0.07$ pour $i \neq 0$, avec une barrière $H = 0.075$.)

Pour cela on y_0 en fonction de $y_1, ..., y_{N-1}$. On passe alors à un problème d'optimisation classique dans R^{N-1}. On pose ensuite $\lambda\sqrt{h} := \sqrt{N}\left(\sigma^2_{max} hN + \sigma_{max}\sqrt{hN}\right)$ ainsi que $p = \frac{\lambda\sqrt{h}}{\left\|\overline{\ln(L_i)}(t_k)^\pi - \overline{\ln(L_i)}(t_k)\right\| + \lambda\sqrt{h}}$. Le choix de p est en accord avec la formule 4 lorsque le vecteur normal $n(\overline{\ln(L_i)}(t_k)^\pi)$ est colinéaire à $\overline{\ln(L_i)}(t_k)^\pi - \overline{\ln(L_i)}(t_k)$. Ce n'est pas toujours le cas, mais ceux ci sont surement assez proches et nous les supposerons égaux dans notre implementation.

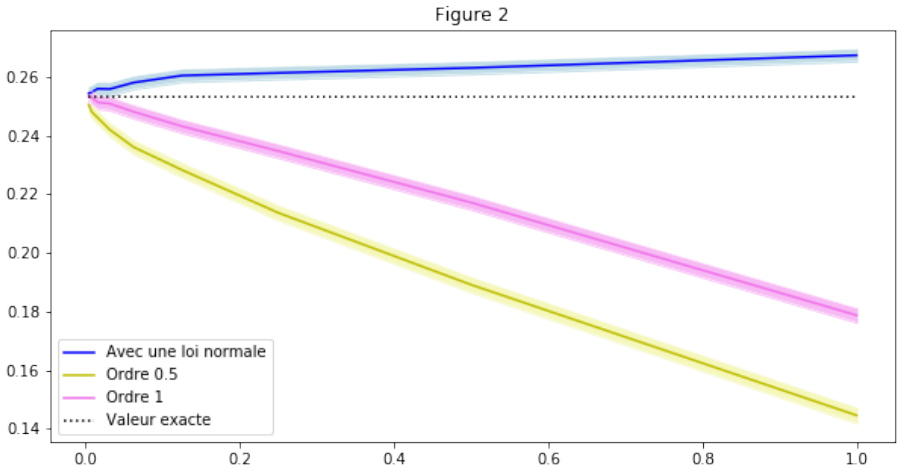

FIGURE 2 – Estimations Monte Carlo du prix d'un Swaption

On peut alors appliquer la méthode de Monte-Carlo associée aux simulations reposant sur un schema d'Euler de bruit gaussien et de bruit $\mathcal{U}(\{\pm 1\})$.

Nous les avons appliqués avec comme paramètres $\beta = 0.1$, $H = 7.5\%$, $K = 1\%$, σ constant égal à 10%, une courbe des taux initiale plate égale à 5% et des maturités $T_0 = 10$, $\delta = 1$, $T^* = 20$. Les résultats pour differents pas de discrétisation h pour 4000 simulations sont donnés sur la figure 2.

La valeur approchée est obtenue en utilisant la formule fermée du prix d'un swaption et l'approximation de Rebonato de la volatilité du swap. Celle-ci est environ égale à 0.2533.

On observe de nouveau de manière claire les différents ordres de convergence. Il semble que les simulations basées sur une loi normale soient plus précises pour un pas grand, mais cela tend à etre moins precise que l'algorithme d'ordre 1 basé sur les marches aléatoires alors même que notre algorithme de projection utilisé pour celui-ci pourrait etre amélioré.

Comme précédemment, on peut implémenter ces méthodes en version antithétique afin de les accélérer. Les résultats sont présentés dans les tableaux suivants. On observe alors que la méthode antithétique permet bien un gain de temps mais celui-ci est relativement faible. De plus, on vérifie que l'utilisation de l'algorithme d'ordre 1 basé sur des marches aléatoires est moins biaisé que celui basé sur des bruits gaussiens quand h devient suffisamment petit. Tous ces résultats se retrouvent sur les tables 8, 9, 10 et 11.

TABLE 8 – Simulation Brownienne du prix d'un Swaption sans la méthode antithétique

h	Resultat	Erreur	Variance	simulations	temps
$h = 1$	0.26732	± 0.00099912	0.011434	44000	964 ms
$h = 0.1$	0.25868	± 0.00098982	0.012497	49000	8.725 s
$h = 10^{-2}$	0.25539	± 0.00099756	0.012952	50000	86.189 s
$h = 10^{-3}$	0.2546	± 0.00099579	0.012906	50000	918.68 s

TABLE 9 – Simulation Brownienne du prix d'un Swaption avec la méthode antithétique

h	Resultat	Erreur	Variance	simulations	temps
$h = 1$	0.26734	± 0.0009973	0.0069907	27000	672 ms
$h = 0.1$	0.258	± 0.00099463	0.0079834	31000	7.705 s
$h = 10^{-2}$	0.25416	± 0.0009867	0.0083636	33000	78.245 s
$h = 10^{-3}$	0.25516	± 0.00099317	0.0082167	32000	770.43 s

TABLE 10 – Simulation avec l'algorithme des marches aléatoires d'ordre 1 du prix d'un Swaption sans la méthode antithétique

h	Resultat	Erreur	Variance	simulations	temps
$h = 1$	0.1792	± 0.00099358	0.015933	62000	5.967 s
$h = 0.1$	0.24569	± 0.00099283	0.0136	53000	8.835 s
$h = 10^{-2}$	0.25202	± 0.0009935	0.013104	51000	73.405 s
$h = 10^{-3}$	0.25398	± 0.00099997	0.013015	50000	665.37 s

TABLE 11 – Simulation avec l'algorithme des marches aléatoires d'ordre 1 du prix d'un Swaption avec la méthode antithétique

h	Resultat	Erreur	Variance	simulations	temps
$h = 1$	0.1789	± 0.00098832	0.006611	26000	4.812 s
$h = 0.1$	0.24574	± 0.0009986	0.0085664	33000	7.128 s
$h = 10^{-2}$	0.25238	± 0.00099105	0.0084374	33000	57.333 s
$h = 10^{-3}$	0.25295	± 0.00099039	0.0084263	33000	531.98 s

On peut également essayer de trouver un F qui réduise la variance comme expliqué précédemment. Pour cela, il faut prendre : $F = -{}^t\sigma \nabla \left(\tilde{V}_{swaption}\right)$. Le gradient est par rapport aux taux libors L_i. De plus, ${}^t\sigma$ doit correspondre à la forme demandée dans la section 1. Autrement dit, on a : $\sigma\left(t, (L_i)_i\right) = \text{diag}\left(L_i \sigma_i(t)\right)$. On peut alors calculer explicitement F. La formule est trop longue pour etre recopiée ici. Nous l'avons cependant implémentée. Nous avons également essayé deux options. La première consistait à dériver la formule exacte, puis à remplacer dans cette formule $v_{R_{swap}}$ par sa valeur approchée par la formule de Rebonato. La seconde consistait à remplacer $v_{R_{swap}}$ par sa valeur approchée puis à dériver (Le résultat est différent car la valeur approchée dépend des valeurs de L_i.) Nous avons tésté les deux et la première semble fonctionner beaucoup mieux donc nous ne présenterons que ses résultats. Nous retrouvons le même genre de résultats à la section précédente et les commentaires de la section caplets s'appliquent toujours ici. Nous présentons à titre d'illustration les résultats uniquement dans le cadre antithétique. Remarquons que la réduction de variance est un peu moins importante que pour les caplets. Cela vient notamment du fait que la formule est approchée. (tables 12 et 13)

TABLE 12 – Simulation Brownienne (antithétique) du prix d'un Swaption avec F optimal

h	Resultat	Erreur	Variance	simulations	temps
$h = 1$	0.26798	± 0.00099722	0.002977	11500	1.381 s
$h = 0.1$	0.25921	± 0.00098613	0.00065818	2600	3.107 s
$h = 10^{-2}$	0.25479	± 0.00091563	$4.3649 \cdot 10^{-5}$	200	2.369 s
$h = 10^{-3}$	0.25385	± 0.00072366	$1.3632 \cdot 10^{-5}$	100	11.896 s

4 Implementation en C++

Tout d'abord, nous avons découpé notre code C++ en plusieurs fichiers distincts. Un premier fichier contient le code de la fonction "main". Ensuite, nous avons construit un fichier appelé *Header.hpp* dans lequel nous avons écrit tout les instructions déstinées au préprocesseurs (tous les includes) ainsi que les utilisations des espaces noms. Nous avons également établi un fichier par simulation contenant une classe se chargant de cette simulation et un fichier pour le code (générique) permettant de faire des simulations Monte-Carlo. Enfin, nous avons également utilise deux couples de fichiers .cpp et .hpp déstinés respectivement aux tests unitaires et aux codes ayant servi aux simulations présentées dans notre rapport. Ces fichiers ne sont pas présents dans notre compte-rendu car ils sont remplacés par des fonctions dans le fichier main permettant à tout utilisateur d'entrer à l'exécution les paramètres qu'il souhaite sur chaque type de simulation.

TABLE 13 – Simulation avec l'algorithme des marches aléatoires (antithétique) du prix d'un Swaption avec F optimal

h	Resultat	Erreur	Variance	simulations	temps
$h = 1$	0.17942	± 0.00099714	0.0051248	19800	5.754 s
$h = 0.1$	0.2449	± 0.00099523	0.0018822	7300	8.698 s
$h = 10^{-2}$	0.25218	± 0.0009752	0.00042086	1700	18.778 s
$h = 10^{-3}$	0.25371	± 0.00075934	$3.002 \cdot 10^{-5}$	200	22.915 s

Venons en maintenant au point principal de notre implémentation : les classes servant aux simulations. Bien que le générateur de nombre aléatoire choisi dans notre projet soit un Mersene Twister 19937 en 64 bits, nous avons souhaité nous laisser la possibilité de changer en ne construisant que des classes templates et le paramètre template est le générateur de nombres aléatoires.

Ces classes disposent de structures internes permettant de gérer le format des paramètres ainsi que des résultats. On a également construit des méthodes payoff et différentes méthodes pour chaque type de simulation. Ces méthodes sont toutes dédoublées pour permettre d'indiquer si l'on souhaite une simulation antithétique ou non. Une méthode distincte est également présente pour chaque type de simulation. Par exemple, dans la classe pour simuler le prix d'un caplet, on dispose des méthodes suivantes :

— *rw_simulation* qui implémente l'algorithme 2.

— *normal_simulation* qui implémente un schema Euler classique

— *bridge_normal_simulation* qui implemente la simulation exacte sans discrétisation basée sur un pont brownien

Chacune de ses méthodes dispose également de son pendant antithétique.

Lorsque nous avons programmé ces methodes, une question cruciale s'est posée : ne devrions-nous pas plutôt chercher à synthétiser le code en le factorisant, quitte par exemple à adopter un mode de programmation générique. Différentes possibilités ont été étudiées pour refactoriser le code.

La première a été de réunir antithétique et non antithétique ensemble, voir de réunir les différents types de simulations ensemble et de mettre en paramètre le choix de celle-ci. Cependant, cela impliquait de rajouter des conditions qui d'une part allongeaient les méthodes ou alors rajoutaient des appels de fonction. Dans tous les cas, pour réduire la quantité globale de code, et donc pour que la refactorisation soit utile, il fallait mettre des conditionnements à l'intérieur des boucles de discrétisation. Cela impliquait donc de ralentir le code (parfois de près de 10% dans le cas du caplet). L'économie de code n'était pas suffisament conséquente pour justifier cette proposition.

La seconde possibilité était, notamment dans le cas des appels antithétiques, d'écrire une fonction annexe sans génération de nombre aléatoire qui effectuait le calcul déterministe. On pouvait alors appeler cette fonction avec l'aléatoire préalablement simulé et les paramètres actuels de la discrétisation. Cependant, cela impliquait de dédoubler certains calculs (ou alors d'augmenter le nombre d'arguments de la fonction) et surtout cela décuplait le nombre d'appels de fonction lors de l'exécution. Nous avions commencé par implémenter les simulations caplets de cette manière là et avions observé un gain de temps de 35% lorsque nous avions développé le code. [6] De plus, dans la classe dédiée aux swaptions, nous avons réuni les boucles dédiées au calcul des deux variables antithétiques.

Enfin, nous avons utilisé les bibliothèques Eigen (pour le calcul matriciel) et boost (pour calculer les quantiles de la loi normale ainsi que sa fonction de distribution).

Références

[KT12] Maria Krivko and Michael Tretyakov. Application of simplest random walk algorithms for pricing barrier options. page 22, November 2012.

6. Nous avons découvert récemment l'inlining de fonctions en C/C++ qui aurait pu être une solution à notre problème. Nous n'avons cependant pas souhaité modifier notre code pour le tester car celui-ci etait déja complet. Cela serait toutefois quelque chose à essayer et peut-être à changer dans notre code.