Pricing des FX Variances Swaps sous le modèle de Heston

John Sibony
Jeremy Chichportich
William Sauve
Khalil Belghouate

Objectif

Le projet a pour but de déterminer le prix d'un contrat forward sur swap de variance. On peut voir ces contrats comme le sentiment des marchés financiers vis à vis de la volatilité du prix d'un sous jacent. Le payoff du contrat sur position longue est la différence entre la variance réalisée et un strike de variance K â une date fixée auparavant T. La partie courte recoit l'opposé. En notant V_T le prix d'un tel contrat à maturité, RV la variance réalisée et L une constante de prix-normalisation on a :

$$V_T = (RV - K)L$$

Avant d'entré dans un contrat, le type de volatilité RV pris en compte doit être spécifié. Le plus souvent il s'agit d'une deux formules suivantes :

$$\frac{T}{N}\sum_{i=1}^{N} R^2 \qquad \frac{T}{N}\sum_{i=1}^{N}(R_i - \bar{R})^2$$

avec $R = \ln(S_{t_i}/S_{t_{i-1}})$, $\bar{R} = \frac{1}{N}\sum_{i=1}^{N} R_i$ et S le sous-jacent.
Il nous reste â déterminer la dynamique stochastique de S. On se propose de se mettre dans les conditions du modèle d'Heston où le processus des prix de l'asset S est représenté par l'EDS à volatilité stochastique suivante :

$$\begin{cases} dS_t = rS_t + \sqrt{v_t}S_t dB_t^1 \\ dv_t = \kappa(\theta - v_t)dt + \sigma_V\sqrt{v_t}dB_t^2 \end{cases}$$

avec B^1 et B^2 des mouvements browniens de corrélation ρ

But : Il n'y a pas de cout d'entré dans un contrat de variance swap. Ainsi, le prix d'un tel contrat à l'instant initial est nulle et l'on peut donc pricer le strike de variance K. Puisque le prix du contrat est l'espérance actualisée sous probabilité risque-neutre du payoff, on a trivialement $K = \mathbb{E}_{\mathbb{Q}}(V_0)$. Deux méthodes sont alors proposées : l'une analytique (formule fermée du prix), l'autre par simulation de Monte-Carlo selon différents schéma de discrétisation.

Structure du code

Le programme se découpe en 6 fichiers distincts (.cpp) avec leur fichier header où sont notés les prototypes des fonctions/classes. Chaque fichier a une propre utilité qui est expliqué dans son header avec des commentaires sur les fonctions utilisées. Le code brut se trouve dans les fichiers .cpp avec quelques commentaires lorsque nécessaire. Nous décrivons ci dessous chacun des 6 fichiers :

Nous avons implémentés le fichier utilsClass comprenant 2 classes. La première est une classe héritant de la classe complexe par défaut permettant d'élargir ses propriétés en acceptant les opérations classiques entre int/complexe et double/complexe. Cela nous a grandement facilité la tâche puisqu'il nous suffit plus qu'à créer un nombre complexe unité i puis sa partie réelle et imaginaire se complétera automatiquement lors de l'exécution des opérations. La seconde classe est une classe aléatoire permettant de simuler des lois normales corrélés par la méthode de Box Muller.

Le fichier analytiquePrice correspond au code du pricing par formule analytique. Dans cette méthode, deux fonctions principales C et D sont utilisées. Le fichier comprend 4 classes : une regroupant les fonctions utilisées par les fonctions C et D ; 2 autres classes correspondant aux fonctions C et D avec leurs dérivés première et seconde ; et une classe chargée de calculer le la variance des prix log-spot S (ln(S)) en prenant en attribut les objets C et D.

Le fichier montecarloPrice correspond au code du pricing par méthode de Monte Carlo. Ce fichier est le plus lourd plus qu'il contient 3 différents schéma de discrétisation pour la simulation (Euler, TG, QE). Il contient une class HestonModel chargée d'initialiser les paramètres de l'EDS Heston (dont le pas en temps delta) ainsi qu'un schéma de simulation du log-spot appelé dedie-Kaya utilisé par 2 sur 3 des méthodes (TG et QE). Les 3 classes suivantes permettent de simuler les prix log-spot selon leur propre schéma et en héritant de la classe HestonModel. Pour le schéma d'Euler, la méthode de simulation du log-spot est biensure réécrite afin de remplacer celle hériter du schéma de sa classe mère.

Le fichier varianceSwapPricer contient une unique classe calculant le prix du strike en prenant en attribut 4 objets correspondant aux prix log-spot analytique et aux 3 prix log-spot Monte Carlo.

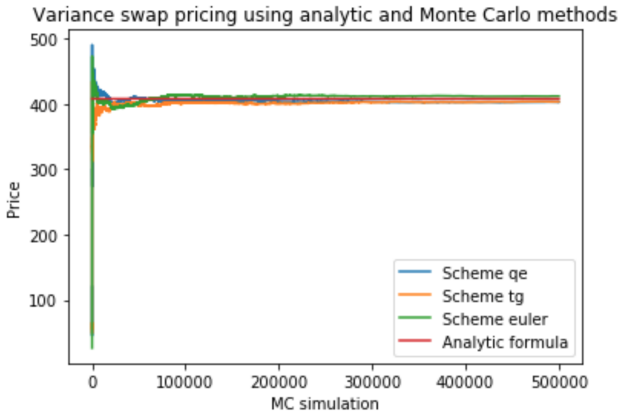
Le fichier europeanOptionPricer est écrit à la manière du fichier précédent varianceSwapPricer afin de calculer le prix initial d'une option européenne selon les 3 mêmes schéma Monte Carlo précédent et avec sa formule analytique fermée déterminée dans les années 1990 par SL Heston. Ce fichier était optionnel.

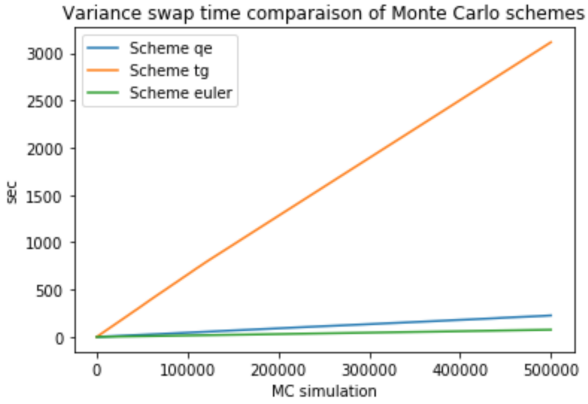
Le fichier main correspond à l'interface utilisateur : un objet des fichiers varianceSwapPricer et europeanOptionPricer est crée puis leur méthode de pricing est appelée selon l'attente de l'utilisateur.

Résultats

On trouve ci-dessus la comparaison graphique des prix Variance Swap (avec le temps de simulation) selon les différentes méthodes pour les paramètres fixés suivants :

$\kappa = 0.5,\ \rho = -0.9,\ \sigma = 1,\ \theta = 0.04,\ r = 0,\ T = 10,\ N = 500,\ s_0 = 100,\ v_0 = 0.04$



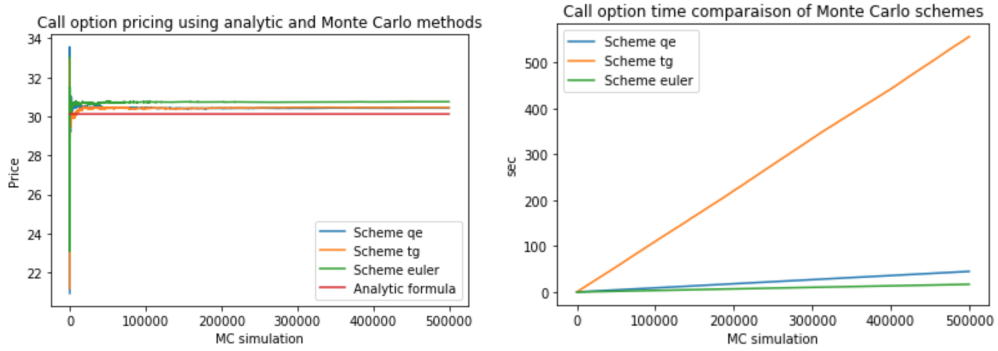


Après 500k simulations Monte Carlo on trouve les prix suivants :

Analytic = 407.713 Euler = 394.84 QE = 401.438 TG = 398.996

De même ci dessous la comparaison graphique des prix Call Européen (avec le temps de simulation) selon les différentes méthodes pour les paramètres fixés suivants :

$$\kappa = 1.5,\ \rho = 0.5,\ \sigma = 1,\ \theta = 0.02,\ r = 0,\ T = 1,\ N = 100,\ s_0 = 100,\ v_0 = 0.04, K = 70$$



Après 500k simulations Monte Carlo on trouve les prix suivants :

Analytic = 30.1193 Euler = 30.6707 QE = 30.4288 TG = 30.4179

On remarque que les schémas TG et QE donnent des performances similaires mais le schéma TG prend bien plus de temps que les autres dû au calcul du zéros d'une fonction par m'ethode de Newton à chaque simulation

Répartition des tâches

John et Jérémy se sont chargés d'implémenter la formule analytique, William et Khalil les 2 méthodes Monte Carlo.

Analytic pricing

L'objet du papier Zhu et Lian est d'obtenir une formule close du prix strike d'un contrat de variance swap. La résolution de l'EDS se passe en 2 temps afin de lever l'inconnue intégrale par rapport à la mesure de dirac. L'output de la première EDS sera l'input de la seconde par propriété de continuité. Le papier offre au lecteur la solution finale du problème.

Nous avons implémenté un code reproduisant les différentes fonctions analytiques. Pour se faire, nous avons créer une classe All_function ayant pour méthode les fonctions généraliste. Les classes C et D sont les plus importantes, celles qui définissent le prix analytique et prenant en héritage la classe All_function. Elles ont pour méthode leur dérivée première et seconde. Enfin, une dernière classe Analytic se charge d'agencer la formule analytique finale selon les objets C et D.

L'une des principales difficultés était de travailler avec des nombres complexes puisque le pricing demande la résolution de transformée de Fourier. Nous avons eu l'idée d'implémenter dans un autre fichier la classe My_complex qui permet l'opération entre un complexe et un double ou int. Cela nous a permis de réutiliser les équations du papier tel qu'elle sans à se soucier de trouver les bonnes partiels réelle et imaginaire de chaque complexe. Notre classe le fait automatiquement.

Nous avons également du faire attention à la fonction g dont le dénominateur peut être nulle. Dans ce cas, nous avons calculer analytiquement la limite en l'infini et avons pris en compte ce résultat dans notre algorithme.

Enfin, John s'est chargé de toute la structure des différents fichiers, de l'agencement des classes, méthodes et fonctions. Chaque personne lui a envoyé son code afin qu'il puisse l'intégrer dans le code finale. Il s'est également chargé d'implémenter le fichier utilsClass et s'est proposé d'implémenter le pricing de l'option européenne. Pour ce faire, il a dû utiliser la formule fermée d'un call européen par Heston lui même puisque la formule du papier d'Anderson était incorrecte. De plus, il a crée le schéma d'euler afin de pouvoir mieux comparer la performance des 2 autres schéma Monte Carlo. Il a implémenté le schéma Broade-Kaya pour la simulation des log-price utilisés dans les 3 schémas (euler, QE, TG). Pour le fichier main, il a implémenté l'interface avec l'utilisateur pour pouvoir choisir les contrats à pricer ainsi que leur schéma de discrétisation. Il s'est finalement chargé de runner/débugger les algorithmes afin que les prix converges. L'une des principales difficultés dont il a du faire face était sur le schéma TG où il a du réécrire le calcul de la fonction de répartition pour obtenir des résultats plus précis que la méthode de somme de Riemann. L'initialisation de la méthode de Newton a aussi était un de ses problèmes. Une grille de recherche a du être effectuée en amont afin de trouver la bonne initialisation. Pour finir, il s'est proposé d'écrire ce rapport latex (sauf la partie suivante) avec les graphiques.

Monte Carlo pricing

L'objet du papier d'Anderson est l'étude de m'ethodes de simulation permettant de diffuser le modèle de Heston et plus particulièrement de discrétiser la diffusion du processus de volatilité/variance sous certaines hypothèses (exemple : une corr'elation loin de 0). William s'est chargé d'implémenter la m'ethode de discrétisation Truncature Gaussian [TG] tandis que Khalil la méthode Quadratic Exponential [QE].

Le code est organisé selon une architecture orientée objet. Nous avons ainsi créé une classe TG qui va modéliser les trajectoires de Monte-Carlo. Cette classe hérite de la classe mère HestonModel et donc des paramètres du modèle d'Heston. De plus, on lui a rajouté un attribut de classe qui se nomme v0 et qui représente la variance initiale. La classe est dotée d'une méthode V qui permettra de simuler les différents paths de la variance. Cette méthode prend en paramètre deux valeurs : le nombre de pas temporel (delta) et le nombre de simulation (nb_steps) afin de laisser à l'utilisateur une liberté et un contrêle sur son pricer. En effet, plus ces deux paramètr ès sont grands, plus la précision augmente, cependant le temps de calcul augmente aussi. Il s'agira donc de trouver un équilibre entre ces deux paramètres.

Il faut bien noter que les méthodes de Monte-Carlo sont beaucoup plus lentes que les m'ethodes analytiques (quand elles existent), cependant certains types de produits dériv'es, comme les options américaines, qui sont path-dependant n'ecessitent d'être pricée par de telles méthodes.

L'autre schéma que nous avons implémenté pour pouvoir utiliser la méthode de Monte-Carlo est le schéma Quadratic Exponential, noté QE. De la même manière que pour la classe TG, nous avons créé une classe QE comme suit :

Nous avons hérité la classe HestonModel afin de ne pas redéfinir une nouvelle fois la fonction phi et afin d'initialiser les paramètres depuis la classe Heston. De plus, nous avons créé une méthode "V" qui permet de calculer le processus de variance à l'instant t+1 à partir du processus au temps t. Cette méthode fait appel à deux autres fonctions "Next_below" et "Next_above" selon que la valeur de phi, à pas de temps et à simulation fixés, soit respectivement inférieur ou supérieur à la valeur de phi_critic. Enfin, le choix de phi_critic (appelé threshold) peut être modifié par l'utilisateur en appelant le constructeur non par défaut de la classe VarianceSwapPricer.

Précisons que la martingale correction optionelle, n'a pas été implementée. La méthode QE s'est révélée être très efficace et relativement simple à impl'ementer. Plus intuitif, le schéma TG (transformation impliquant une gaussienne simple) donne par ailleurs de moins bons résultats que le schéma QE.