Comparaison des méthodes de pricing d'options vanilles européennes par Monte Carlo et Black-Scholes

Paul-Noël DIGARD, Nizar JELILA

1 Théorie du pricing d'options

1.1 Définitions

Commençons par définir dans un cas concret une option sur action :
Une option est un contrat qui donne le droit et non l'obligation à son détenteur de vendre ou d'acheter :

— un sous-jacent (ici une action, dont le prix à l'instant t sera noté S_t)
— à un prix donné (appelé strike et noté K)

Dans le cas des options européennes, cet achat peut s'effectuer uniquement à un instant T dans le futur. Il s'agit de la maturité de l'option.

À échéance, le détenteur d'un call vanille européen (seule option que l'on considérera dans ce projet ; il s'agit du droit d'acheter une action à un prix K à un instant futur T) reçoit donc (on appelle cette valeur le *pay-off* du call) :

$$(S_T - K)+ \tag{1}$$

En effet, si le cours de l'action S_T est plus faible que le strike K, alors le détenteur du call n'aura aucun intérêt à exercer son option et préférera acheter l'action au prix S_T.

1.2 Valorisation d'une option

Au vu de la définition d'un call et de son *pay-off*, on observe que l'acheteur du call s'expose à une perte limitée (il pert ce qu'il a payé pour acheter le call si $S_T - K < 0$ car dans ce cas $(S_T - K)+ = 0$), tandis que son gain potentiel est illimité (si S_T est "très grand" devant K). Le vendeur du call est lui dans la situation inverse. La détermination du prix de l'option (appelée la prime) est donc très importante.

Le but de ce projet étant d'élaborer un programme en C++ permettant de calculer le prix d'un Call à partir de certains paramètres (strike K, maturité T, taux d'intérêt r...), nous allons utiliser les résultats suivants sans les démontrer.

Tout d'abord, le résultat fondamental de la théorie de valorisation des options en absence d'opportunité d'arbitrage (ie pas de situation dans laquelle on peut gagner de l'argent sans prendre de risque) est le suivant (C_0 étant le prix (ie la prime) de l'option à l'instant t=0 ; r est le taux d'intérêt) :

$$C_0 = \mathbb{E}[e^{-rT}(S_T - K)+] \tag{2}$$

Ainsi, la prime (C_0) du dérivé est simplement égale à l'espérance des *pay-offs* actualisés du dérivé à maturité.

2 Formule de Black-Scholes

Ici nous allons voir que sous certaines hypothèses, l'équation (2) peut être réécrite sous forme d'une formule fermée.

2.1 Hypothèses sur le sous-jacent

Dans toute la suite, on supposera que la dynamique du sous-jacent (ie cours de l'action) est la suivante :

$$\frac{S_{t+1} - S_t}{S_t} = \mu dt + \sigma\sqrt{dt}Z \tag{3}$$

Où :
— μ représente la tendance
— σ représente la volatilité
— Z suit une loi normale centrée réduite $\mathcal{N}(0,\ 1)$

Cette équation peut se résoudre de la manière suivante (ici sans démonstration) :

$$S_t = S_0 * exp((\mu - \frac{\sigma^2}{2})t + \sigma\sqrt{dt}Z) \tag{4}$$

2.2 La formule de Black-Scholes

En injectant (4) dans (2), on obtient la fameuse formule de Black-Scholes :

$$C_0 = S_0 N(d_1) - Kexp(-rT)N(d_2) \tag{5}$$

Avec :
— N la fonction de répartition de la loi $\mathcal{N}(0,\ 1)$
— $d_1 = \frac{ln(S_0/K)+(r+\sigma^2/2)T}{\sigma\sqrt{T}}$
— $d_2 = d_1 - \sigma\sqrt{T}$

2.3 Implémentation de Back-Scholes en C++

Dans le fichier `BlackScholes.cpp` nous avons créé un programme qui calcule le prix d'une option en utilisant la formule de Black-Scholes (5).

Dans un premier temps nous avons codé la fonction de répartition de la loi $\mathcal{N}(0,\ 1)$. Il s'agit de la fonction nommée `fdr_norm` à la ligne 31 du programme.

Nous avons ensuite codé la fonction `call_price` ligne 60 qui prend en entrée le cours de l'action `S` à t=0, le strike `K`, la maturité `T`, le taux d'intérêt `r` et la volatilité `v` de l'action. La fonction renvoie alors le prix de l'option calculé par la formule de Black-Scholes et fait donc appel à la fonction `fdr_norm`.

Enfin, la fonction `main` du fichier `BlackScholes.cpp` permet de choisir les paramètres S, K, T, r et v et d'obtenir le prix de l'option associé.

Avec les paramètres S=100, K=100, r=0.05, v=0.2 et T=1 on obtient : $\boxed{C_0 = 10.4506}$.

3 Méthode de Monte Carlo

Dans certains, la formule de Black-Scholes ne s'applique pas et il faut donc revenir à la formule (2) pour obtenir le prix de l'option. Il peut alors être intéressant d'utiliser la méthode de Monte Carlo pour obtenir une approximation de cette espérance lorsqu'il n'est pas possible d'en obtenir une formule analytique.

Pour cela, on génère aléatoirement un grand nombre de valeur de S_T en utilisant (4) ainsi qu'une fonction qui génère aléatoirement des réels à partir d'une distribution $\mathcal{N}(0, 1)$.

Nous calculons alors la moyenne de tous les *pay-offs* ainsi générés. Cette moyenne, par la loi des grands nombres est une approximation de l'espérance des *pay-offs*.

Implémentation de la méthode de Monte Carlo en C++ Dans le fichier `MonteCarlo.cpp` nous avons créé un programme qui calcule le prix d'une option en approximant la formule (2) par la méthode de Monte Carlo.

Pour la génération de réels à partir d'une distribution de loi $\mathcal{N}(0, 1)$ nous avons utilisé la fonction `std::normal_distribution<>` de la bibliothèque `<random>`. La fonction `monte_carlo_call_pricer` (ligne 31) consiste alors à calculer la moyenne d'un nombre `num_sims` de *pay-offs* générés aléatoirement suivant la formule (4).

La fonction `main` du fichier `MonteCarlo.cpp` permet de choisir les paramètres S, K, T, r et v et d'obtenir le prix de l'option associé en faisant appel à la fonction `monte_carlo_call_pricer`.

Avec les paramètres S=100, K=100, r=0.05, v=0.2 et T=1 on obtient : $\boxed{C_0 = 10.4525}$. On remarque que cette valeur ne diffère que très peu de celle obtenue par la formule de Black-Scholes à la section précédente.