Méthode de Monte Carlo pour Black-Scholes

A rendre le 25/10/2010.

On s'efforcera à inclure le paradigme objets.

On se propose d'écrire un code C++ pour simuler le prix d'une option (Européenne). Les premiers exercices préparent les outils pour la simulation. Les équations sont décrites au dernier exercice.

Exercice-*1* : *Écrire une fonction de prototype*

`double uniforme(const double`$\&\ x_0$**`, const double`** $\&\ x_1$**`)`**

qui retourne un nombre aléatoire tiré uniformément dans l'intervalle $[x_0, x_1]$.

Pour cette question on rappelle que la fonction **`rand()`** *déclarée dans l'entête* **`<cstdlib>`** *retourne un nombre aléatoire tiré uniformément dans l'intervalle* $[0, \mathrm{RAND\_MAX}]$*, où* **`RAND_MAX`** *est le plus grand nombre réel représentable par votre machine et accessible via l'entête* **`<cstdlib>`**.

Exercice-*2* : *Écrire une fonction de prototype*

`double GetOneGaussian()`

qui simule une variable aléatoire normale centrée $\mathcal{N}(0, 1)$*. Pour cette question, on se renseignera sur les méthodes de simulation de variables aléatoires (Boxmuller, sommation, inversion,...).*

On fournit en exemple l'information suivante :

si U et V sont deux variables aléatoires indépendantes uniformément distribuées sur $[0, 1]$ (i.e. $\mathcal{U}[0, 1]$), alors

$$X = \sqrt{-2\log(U)}\cos(2\pi V) \qquad \textbf{et} \qquad Y = \sqrt{-2\log(U)}\sin(2\pi V)$$

sont indépendantes et de loi normale $\mathcal{N}(0, 1)$.

<u>Note</u> : les fonctions log, max, sqrt, cos, sin sont accessibles via le fichier entête **`<cmath>`**.

Exercice-*3* : *On se propose à présent d'évaluer la fonction suivante*

$$C(T) = e^{-rT}\mathbb{E}\left(f\left(S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}\mathcal{N}(0,1)}\right)\right).$$

où, T, K, r, S_0 *sont des réels donnés,* f *est la fonction définie par* $f(S) = \max(S - K, 0)$ *et* $\mathbb{E}(Z)$ *représente l'espérance de la variable aléatoire* Z.

Pour cela, on utilise la loi faible des grands nombres qui dit que :

Si $Z_i, i = 1, 2, \ldots$, est une suite de variables aléatoires indépendantes et identiquement distribuées, alors la variable aléatoire $\frac{1}{N}\sum_{i=1}^{N} Z_i$ converge en probabilité vers $\mathbb{E}(Z_1)$, lorsque N tend vers l'infini.

Écrire une fonction de prototype

`double simpleMonteCarlo(const double`$\&\ T$**`, const double`**$\&\ K$**`, const double`**$\&\ S_0$**`, const double`**$\&\ \sigma$**`, const double`**$\&\ r$**`, const unsigned long`**$\&\ N$**`)`**

qui retourne la valeur de $C(T)$ ci-dessus selon le principe décrit.

Exercice-*4* : *Pour déterminer le prix d'une option ("call" (C) ou "put" (P)) européenne de maturité T de taux d'intérêt r et de "pay-off" f, la théorie de Black-Scholes fournit*

$$C(T) = e^{-rT}\mathbb{E}(f(S_T)), \tag{1}$$

où S_T est la solution à l'instant T de l'équation différentielle stochastique

$$dS_t = rS_t dt + \sigma S_t dW_t \tag{2}$$

dans laquelle W_t est un mouvement Brownien.
Pour résoudre (2), on passe au log *et on utilise la formule d'Îto. On obtient alors*

$$d\log S_t = \left(r - \frac{1}{2}\sigma^2\right)dt + \sigma dW_t,$$

qui, du fait des coefficients constants conduit à

$$\log S_t = \log S_0 + \left(r - \frac{1}{2}\sigma^2\right)t + \sigma W_t.$$

Mais comme W_t est un mouvement Brownien, W_T suit une loi Gaussienne de moyenne 0 et de variance T. On écrit alors,

$$W_T = \sqrt{T}\mathcal{N}(0,1).$$

Par suite,

$$\log S_T = \log S_0 + \left(r - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}\mathcal{N}(0,1).$$

Ce qui de façon équivalente s'écrit

$$S_T = S_0 e^{(r-\frac{1}{2}\sigma^2)T + \sigma\sqrt{T}\mathcal{N}(0,1)}.$$

Par conséquent, (1) devient

$$C(T) = e^{-rT}\mathbb{E}\left(f\left(S_0 e^{(r-\frac{1}{2}\sigma^2)T + \sigma\sqrt{T}\mathcal{N}(0,1)}\right)\right). \tag{3}$$

Q-*4-1* : *Écrire un simulateur pour le "call" Européen.*

C'est-à-dire un programme interactif qui :
- *récupère en ligne de commande, les valeurs de la maturité (T), du strike (K), du taux d'intérêt (r), de la volatilité (σ), du nombre de trajectoires (N),*
- *fait varier S_0 dans un intervalle $[0, L]$, avec $L = 2 \times K$ par pas de $\Delta S = \frac{L}{M}$ (M étant libre de choix),*
- *et représente sur un même graphique les valeurs $f(S_0)$ et $C(T) \equiv C(S_0, T)$ en fonction de S_0.*

Interpréter les courbes obtenues.
(On pourra générer un fichier résultat et se servir de **`gnuplot`** *pour la visualisation).*

Q-*4-2* : *En utilisant la parité "put-call" suivante*

$$P(t) + S(t) - C(t) = Ke^{-r(T-t)}, \quad \forall 0 \le t \le T$$

écrire un simulateur pour le "put" Européen.

(On pourra remarquer qu'il suffit de remplacer f par $f(S) = \max(K - S, 0)$ dans (3), pour obtenir la formule pour le "put").

Exercice-*5* : *Encapsuler les fonctions écrites dans une classe de sorte que l'exécution suivante soit possible :*

fichier

```
int main(int argc, char** argv)
{
   //construction des paramètres
   BlackScholesParameters bs_params;
   bs_params.Init(...);//il faudra fournir les paramètres

   //constructions du Problem
   BlackScholesProblem    bs_problem;
   bs_problem.SetParam(bs_params);

   //constrcution du solveur
   BlackScholesRunner     bs_runner;
   bs_runner.SetTarget(bs_problem);

   //exécution
   bs_runner.execute(); // ceci fait tout
return 0;
}
```