ensiie
école nationale supérieure d'informatique
pour l'industrie et l'entreprise

Projet de PAP
2018-2019

MEHDI M'BITEL
CHEN ZEYU

09 Janvier 2019

Sommaire

Préambule

L'objectif de ce projet est de modéliser un marché financier et de déterminer le prix et la couverture d'options européennes.

Dans ce projet nous utilisions des méthodes numériques telle que la méthode de Crank-Nickelson ou encore celle des différence finies implicites afin d'estimer la solution de l'équation de Black-Scholes.

nous allons étudier différents modèle connus pour résoudre ce problème.

1 EDP complète

1.1 Discrétisation de l'espace-temps

Nous allons fixer le nombre N de points en temps à calculer. On pose $\Delta T = \frac{T}{N}$ puis $t_n = n\Delta T$.

Nous allons fixer le nombre M+1 de points en espace à calculer. On pose $\Delta s = \frac{L}{M+1}$ puis $s_i = i\Delta s$.

1.2 Discrétisation des dérivées partielles

On approche les dérivées partielles intervenant dans l'EDP :

$$\frac{\partial C_{n,i}}{\partial t} \approx \frac{1}{\triangle T}\left(C_{n+1,i} - C_{n,i}\right)$$

$$\frac{\partial C_{n,i}}{\partial S} \approx \frac{1}{2}\left(\left(\frac{1}{2\triangle s}C_{n+1,i+1} - C_{n+1,i-1}\right) + \frac{1}{2\triangle s}(C_{n,i+1} - C_{n,i-1})\right)$$

$$\frac{\partial^2 C_{n,i}}{\partial S^2} \approx \frac{1}{2\triangle s^2}(C_{n+1,i+1} - 2C_{n+1,i} + C_{n+1,i-1}) + \frac{1}{2\triangle s^2}(C_{n,i+1} - 2C_{n,i} + C_{n,i-1})$$

1.3 Résourde le problem sous la forme de la matrice

Tout d'abord, on a l'EDP :

$$\frac{\partial C}{\partial t} + rS\frac{\partial P}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 C}{\partial S^2} = rC$$

et on sait que $P(t_n, s_i)$ est remplacé par $C_{n,i}$.

donc, on a :

$$\frac{1}{\triangle T}(C_{n+1,i} - C_{n,i}) + \frac{1}{4}rS\frac{1}{\triangle s}((C_{n+1,i+1} - C_{n+1,i-1}) + (C_{n,i+1} - C_{n,i-1}))$$

$$+\frac{1}{4\triangle s^2}\sigma^2 S^2((C_{n+1,i+1} - 2C_{n+1,i} + C_{n+1,i-1}) + (C_{n,i+1} - 2C_{n,i} + C_{n,i-1})) = rC_{n,i}$$

après :

$$\left(-\frac{1}{\triangle T} - \frac{1}{2}\sigma^2 i^2 - r\right)C_{n,i} + \left(\frac{\sigma^2 i^2}{4} + \frac{1}{4}ri\right)C_{n,i+1} + \left(\frac{\sigma^2 i^2}{4} - \frac{1}{4}ri\right)C_{n,i-1} =$$

$$\left(-\frac{1}{\triangle T} + \frac{1}{2}\sigma^2 i^2\right)C_{n+1,i} + \left(-\frac{\sigma^2 i^2}{4} - \frac{1}{4}ri\right)C_{n+1,i+1} + \left(\frac{1}{4}ri - \frac{\sigma^2 i^2}{4}\right)C_{n+1,i-1}$$

Sur la matrice, on a :

$$a_i = \frac{\sigma^2 i^2}{4} - \frac{1}{4}ri$$

$$b_i = -\frac{1}{\triangle T} - \frac{1}{2}\sigma^2 i^2 - r$$

$$c_i = \frac{\sigma^2 i^2}{4} + \frac{1}{4}ri$$

$$d_i = \frac{1}{4}ri - \frac{\sigma^2 i^2}{4}$$

$$e_i = -\frac{1}{\triangle T} + \frac{1}{2}\sigma^2 i^2$$

$$f_i = -\frac{\sigma^2 i^2}{4} - \frac{1}{4}ri$$

donc, on a :

$$A = \begin{bmatrix} b_0 & c_0 & 0 & \cdots & \cdots & 0 \\ a_1 & b_1 & c_1 & \cdots & \cdots & 0 \\ 0 & a_2 & \ddots & \ddots & \cdots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & a_M & b_M & c_M \\ 0 & 0 & 0 & \cdots & a_{M+1} & b_{M+1} \end{bmatrix} \qquad B = \begin{bmatrix} e_0 & f_0 & 0 & \cdots & \cdots & 0 \\ d_1 & e_1 & f_1 & \cdots & \cdots & 0 \\ 0 & d_2 & \ddots & \ddots & \cdots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & d_M & e_M & f_M \\ 0 & 0 & 0 & \cdots & d_{M+1} & e_{M+1} \end{bmatrix}$$

A et B ont la taille de $M + 2 * M + 2$, parce que j'ai fait une discrétisation pour l'espace en fixant le nombre M+1 de points. Et puis, comme on change pas le Pc (y compris la premier et la derniere colonne et la derniere ligne de la matrice C), On peut déduire que $b_0 = e^{-r\Delta T}, c_0 = f_0 = d_{M+1} = 0, e_0 = e_{M+1} = 1$

Donc , on a :

$$A = \begin{bmatrix} e^{-r\Delta T} & 0 & 0 & \cdots & \cdots & 0 \\ a_1 & b_1 & c_1 & \cdots & \cdots & 0 \\ 0 & a_2 & \ddots & \ddots & \cdots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & a_M & b_M & c_M \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ d_1 & e_1 & f_1 & \cdots & \cdots & 0 \\ 0 & d_2 & \ddots & \ddots & \cdots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & d_M & e_M & f_M \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

Sachant que deux conditions :

$$P(t,0) = Ke^{r(t-T)} \quad \forall t \in [0,T]$$

$$P(t,L) = 0 \quad \forall t \in [0,T]$$

$$P(T,s) = max(K-s,0) \quad \forall s \in [0,L]$$

On sait donc C est :

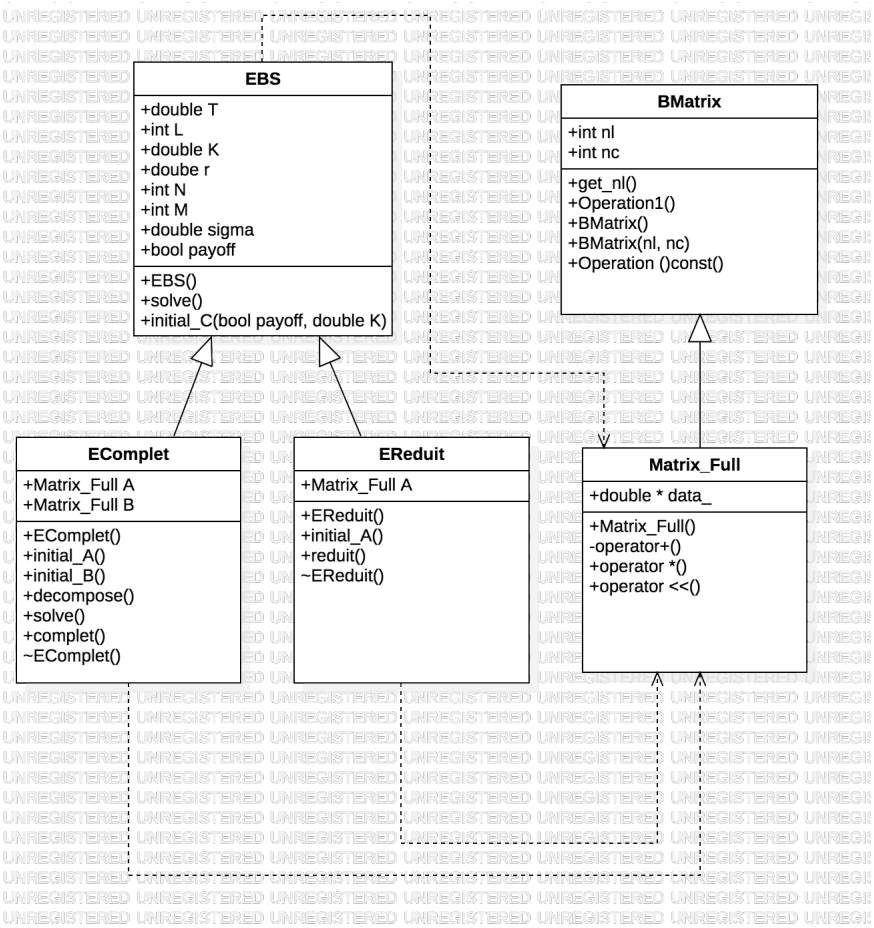
$$C = \begin{bmatrix} Ke^{-r(-T)} & 0 & 0 & \cdots & \cdots & 0 \\ Ke^{-r(\Delta T - T)} & 0 & 0 & \cdots & \cdots & 0 \\ Ke^{-r(2\Delta T - T)} & 0 & \ddots & \ddots & \cdots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & 0 & 0 & 0 \\ K & max(0, K - \Delta s) & max(0, K - 2\Delta s) & \cdots & 0 & 0 \end{bmatrix}$$

Finalement ,on a transformé notre problem sous la forme de la matrice :

$$\boxed{AC_n = BC_{n+1}}$$

1.4 L'implémentation de la méthode en C++

D'abord le diagramme UML :



Donc, à partir d'initialiser la matrice A et B et C, on va parcourir chaque ligne de la matrice C ,on commence par Cn et on derminera quand qu'on trouvra la C0. Pour chaque itération on cherche à resoudre le problem $AC_i = BC_{i+1}$,on passe la matrice A à la méthode decompose et on appelle la méthode solve pour avoir la solution en donnant sa valeur à C_i .

2 Différence finie implicite

2.1 Transformer l'équation de Black-Scholes en une equation de chaleur

NB : Pour simplifier les notations $\widetilde{S}$ sera vu comme x, $\widetilde{t}$ comme τ et $\widetilde{C}$ comme u. Nous commençons par l'équation de Black-Scholes fournie dans le sujet :

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} = rC, \qquad (1)$$

2.1.1 Première étape

L'équation peut être réecrite sous la forme équivalente :

$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 \left(S\frac{\partial}{\partial S}\right)^2 C + \left(r - \frac{1}{2}\sigma^2\right) S\frac{\partial C}{\partial S} - rC = 0.$

Nous faisons un changement de variables :

$S = e^y, \qquad t = T - \tau$

Ce qui donne :

$S\frac{\partial}{\partial S} \to \frac{\partial}{\partial y}, \qquad \frac{\partial}{\partial t} \to -\frac{\partial}{\partial \tau},$

Nous obtenons alors l'équation à coefficients constants suivante :

$\frac{\partial C}{\partial \tau} - \frac{1}{2}\sigma^2 \frac{\partial^2 C}{\partial y^2} - \left(r - \frac{1}{2}\sigma^2\right) \frac{\partial C}{\partial y} + rC = 0.$ (3)

2.1.2 Deuxième étape

Si l'on remplace $C(y, \tau)$ dans l'équation (3) par $u = e^{r\tau}C$, nous obtenons :
$\frac{\partial u}{\partial \tau} - \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial y^2} - \left(r - \frac{1}{2}\sigma^2\right) \frac{\partial u}{\partial y} = 0.$

2.1.3 Troisième étape

Finalement, le changement de variable $x = y + (r - \sigma^2/2)\tau$ nous permet d'éliminer le terme de premier ordre et de mettre l'équation précédente sous la forme suivante :
$\frac{\partial u}{\partial \tau} = \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial x^2}$

2.2 L'implémentation de la méthode des différences finies implicite

2.2.1 Documentation

Considérons une équation de chaleur à une dimension vérifiée par u
$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial^2 x}$
Pour numériser l'équation précédente, il faut approximer les dérivées partielles par des différences. Dans ce cas l'on considère une grille spacio-temporelle (x_i, t_i) homogène telle que la différence entre deux points de temps consécutifs soit Δt et deux points de l'espace consécutifs soit Δx.

Les points $u(x_j, t_n) = u_j^n$ représenterons une approximation numérique de $u(x_j, t_n)$
Ainsi :
$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial^2 x}$
Est approximée en :
$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \alpha \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}$
Ce qui donne :
$u_j^{n+1} = \alpha \Delta t \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} + u_j^n$
Nous posons $\beta = \frac{\alpha \Delta t}{\Delta x^2}$
L'équation précédente devient alors :
$u_j^{n+1} = \beta(u_{j+1}^n - 2u_j^n + u_{j-1}^n) + u_j^n$
Ce qui donne finalement :
$u_j^{n+1} = \beta u_{j+1}^n + (1 - 2\beta)u_j^n + \beta u_{j-1}^n$
Nous arrivons alors à une forme matricielle :

$$A = \begin{bmatrix} w & 0 & 0 & 0 & 0 & 0 & 0 & . & 0 \\ a & b & c & 0 & 0 & 0 & 0 & . & 0 \\ 0 & a & b & c & 0 & 0 & 0 & . & 0 \\ . & 0 & . & . & . & 0 & 0 & 0 & 0 \\ . & 0 & 0 & . & . & . & 0 & 0 & . \\ . & 0 & 0 & 0 & . & . & . & 0 & . \\ . & 0 & 0 & 0 & 0 & . & . & . & 0 \\ 0 & . & 0 & 0 & 0 & 0 & a & b & c \\ 0 & . & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

ie une matrice dont tridiagonale en (a,b,c) à laquelle l'on rajoute une dernière ligne en (1,0,...,0) et une première ligne en ($w = \frac{u_0^{n+1}}{u_0^n}$,0,..,0) à savoir $a = c = \beta$ et $b = 1 - 2\beta$

Dans ce cas nous obtenons :
$U^{n+1} = AU^n$ ie, $U^n = A^{-1}U^{n+1}$

où

$$U^{n+1} = \begin{bmatrix} u_0^{n+1} \\ . \\ . \\ . \\ u_{M+1}^{n+1} \end{bmatrix}$$

Et évidemment

$$U^{n} = \begin{bmatrix} u_0^{n} \\ . \\ . \\ . \\ u_{M+1}^{n} \end{bmatrix}$$

Commençant par la condition finale U^T, nous arrivons par boucle de T à 0 à U^0.

3 Affichage des courbes

Le plot pour un put de la courbe avec la méthode complète : Images/Complet_put.png

Le plot pour un call de la courbe avec la méthode complète : Images/Complet_call.png