Méthodes numériques pour le pricing d'options

Mohamed Ben Alaya

26 février 2013

Nous allons tester les différentes méthodes de différence finies vu dans le cours en l'appliquant au calcul du call ou le put européen. On considère, pour $t \in [0,T]$, le modèle de Black & Scholes solution de l'EDS

$$dS_t = rS_t dt + \sigma S_t dW_t, \quad S_0 = s_0 > 0,$$

où $(W_t)_{0\leq t\leq T}$ désigne un mouvement Brownien standard et $(\mathcal{F}_t)_{0\leq t\leq T}$ sa filtration canonique.

- On sait que l'EDS ci-dessus admet une solution unique donnée par

$$S_t = s_0 exp\left((r - \frac{\sigma^2}{2})t + \sigma W_t\right).$$

  On note par $(S_t^{0,s_0})_{0\leq t\leq T}$ la solution de l'EDS précédente issue de s_0 à l'instant 0. Le prix d'une option européenne, de payoff $g(S_T)$ où g est une fonction numérique, s'écrit comme

$$\mathbb{E}\left(e^{-rT} g(S_T^{0,s_0})\right) = \mathbb{E}\left(f(W_T)\right)$$

  où f est une fonction à déterminer à partir de g.

- On s'intéresse au calcul de la fonction

$$u(t,x) = \mathbb{E}\left(f(x + W_t)\right) \quad t \geq 0 \text{ et } x \in \mathbb{R}.$$

  Cette fonction est solution de l'équation de la chaleur

$$\begin{cases} \dfrac{\partial u}{\partial t} &= \dfrac{1}{2}\dfrac{\partial^2 u}{\partial x^2} \quad \forall (t,x) \in [0,T] \times \mathbb{R} \\ \\ u(0,x) &= f(x). \end{cases}$$

On va rappeler les différentes méthodes de différence finies et les résultats de convergence vu en cours. Ceci est dans le but de les programmer et d'analyser numériquement le choix des paramètres dans le cas du call et du put européen.

Rappel On rappelle que pour le call, le payoff $g(x) = (x-K)_+$ et la valeur de l'option, à l'instant $t \in [0,T]$, est égale à $V(t, S_t^{0,s_0})$ avec

$$V(t,x) = xN(d_1) - Ke^{-r(T-t)}N(d_2),$$

où

$$d_1 = \frac{\log(x/K) + (r + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}} \;\; et \;\; d_2 = d_1 - \sigma\sqrt{T-t}$$

Pour le put on a une formulation analogue avec $V(t,x) = Ke^{-r(T-t)}N(-d_2) - xN(-d_1)$.

1 Introduction aux méthodes d'analyse numérique

On considère une fonction numérique u sur $\mathbb{R}_+ \times \mathbb{R}$ solution de l'équation de la chaleur

$$\left\{ \begin{array}{lcl} \dfrac{\partial u}{\partial t} & = & \dfrac{1}{2}\dfrac{\partial^2 u}{\partial x^2} \quad \forall (t,x) \in \mathbb{R}_+ \times \mathbb{R} \\ u(0,x) & = & f(x). \end{array} \right.$$

Pour résoudre numériquement cette équation aux dérivées partielles (EDP), on discrétise l'espace et le temps en se donnant un pas de temps $k = \delta t$ et un pas d'espace $h = \delta x$, tous les deux strictement positifs. On obtient une grille dans $\mathbb{R}_+ \times \mathbb{R}$ par les points

$$(n\delta t, j\delta x) \text{ pour } n \in \mathbb{Z} \text{ et } j \in \mathbb{N}.$$

On cherche à approximer u sur la grille en trouvant une suite de nombres $U(n,j)$ telle que

$$U(n,j) \approx u(n\delta t, j\delta x) \text{ pour } n \in \mathbb{Z} \text{ et } j \in \mathbb{N}.$$

Pour cela on approxime les opérateurs $\frac{\partial}{\partial t}$ par ∂_t et $\frac{\partial}{\partial x}$, au choix, par ∂_x ou $\bar{\partial}_x$, où

$$\left\{ \begin{array}{lcl} \partial_t U(n,j) & = & \dfrac{U(n+1,j) - U(n,j)}{\delta t} \\ \partial_x U(n,j) & = & \dfrac{U(n,j+1) - U(n,j)}{\delta x} \\ \bar{\partial}_x U(n,j) & = & \dfrac{U(n,j) - U(n,j-1)}{\delta x} \end{array} \right.$$

Puis l'on approxime $\frac{\partial^2}{\partial x^2}$ par $\partial_x \bar{\partial}_x$. Soit :

$$\partial_x \bar{\partial}_x U(n,j) = \frac{U(n,j+1) - 2U(n,j) + U(n,j-1)}{(\delta x)^2}$$

1.1 Méthode explicite

En tenant compte de ces approximations, on obtient

$$\frac{U(n+1,j) - U(n,j)}{\delta t} = \frac{U(n,j+1) - 2U(n,j) + U(n,j-1)}{2(\delta x)^2}.$$

On appelle ce type de discrétisation un schéma explicite, car on calcule directement $U(n+1,.)$ en fonction de $U(n,.)$. Ou encore, si l'on pose $\lambda = \dfrac{k}{h^2} = \dfrac{\delta t}{(\delta x)^2}$:

$$U(n+1,j) = \frac{\lambda}{2} U(n,j+1) + (1-\lambda) U(n,j) + \frac{\lambda}{2} U(n,j-1)$$

Remarques :

1. Pour $0 < \lambda \leq 1$, ce schéma est stable dans L^∞, dans le sens que $||U(n+1,.)||_\infty \leq ||U(n,.)||_\infty$, pour tout $n \in \mathbb{N}$. Il converge aussi vers la vraie solution. Dans ce cas le schéma est dit conditionnellement convergent.
2. On peut également restreindre le problème sur un l'espace bornée $[0,T] \times [x_{min}, x_{max}]$. Les bornes des intervalles sont des paramètres à choisir soigneusement. Dans ce cas, il faut imposer des conditions au bord, on peut par exemple prendre des conditions du type $u(t, x_{min}) = g(t)$ et $u(t, x_{max}) = d(t)$.

Si l'on considère alors l'EDP suivante

$$\begin{cases} \dfrac{\partial u}{\partial t} & = & \dfrac{1}{2}\dfrac{\partial^2 u}{\partial x^2} & \forall (t,x) \in [0,T] \times [x_{min}, x_{max}] \\ u(t, x_{min}) = 0 & \text{et} & u(t, x_{max}) = 0 & \\ u(0,x) & = & f(x). & \end{cases}$$

Une discrétisation de ce problème, en posant $\delta t = T/N$ et un pas d'espace $\delta x = (x_{max} - x_{min})/(M+1)$, est de considérer les points

$$\begin{cases} x_0 & = & x_{min} \\ x_1 & = & x_{min} + \delta x \\ & \vdots & \\ x_j & = & x_{min} + j\delta x \\ & \vdots & \\ x_M & = & x_{min} + M\delta x \\ x_{M+1} & = & x_{max} \end{cases}$$

et d'approximer

$$u(n\delta t, x_j) \approx U(n,j), \quad n \in \{0, \cdots, N\} \text{ et } j \in \{0, \cdots, M+1\},$$

avec

$$U(n+1,j) = \frac{\lambda}{2}U(n,j+1) + (1-\lambda)U(n,j) + \frac{\lambda}{2}U(n,j-1), \ 1 \leq j \leq M,$$

et pour tenir compte des conditions aux limites en x_{min} et x_{max} compléter par $U(n+1,0) = 0$ et $U(n+1, M+1) = 0$. Maintenant, si l'on pose $U^n = \begin{pmatrix} U(n,1) \\ \vdots \\ U(n,M) \end{pmatrix}$ alors

$$U^{n+1} = AU^n \quad \text{avec} \quad A = \begin{pmatrix} a_2 & a_3 & 0 & 0 & 0 & 0 & 0 \\ a_1 & a_2 & a_3 & 0 & 0 & 0 & 0 \\ 0 & a_1 & a_2 & a_3 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & a_1 & a_2 & a_3 & 0 \\ 0 & 0 & 0 & 0 & a_1 & a_2 & a_3 \\ 0 & 0 & 0 & 0 & 0 & a_1 & a_2 \end{pmatrix}$$

$a_1 = a_3 = \frac{\lambda}{2}$ et $a_2 = 1 - \lambda$.

Exercice Ecrire un programme basé sur cette égalité pour calculer un call européen.

```
// TP 4
// Méthode des differences finies pour l'équation de la chaleur
// On introduit la fonction $f$
// La grille est uniforme en espace et en temps.

stacksize(1.e7);
clear
xbasc()

////////////////////////////
// Parametres financiers
////////////////////////////

// payoff : call

T=1.;
r=0;
S0=100;
K=100;
sigma=0.05;

////////////////////////////
// Parametres numeriques
////////////////////////////

// Schéma explicite
M=500; // nombres de pas en espace
N=100; // nombres de pas de temps
Smax=5*K; // valeur maximum du sous-jacent
xmax=(log(Smax/S0)-(r-sigma**2/2)*T)/sigma;
xmin=-xmax;
dx=(xmax-xmin)/(M+1);
dt=T/N;
lambda=dt/(dx*dx);


////////////////////////
// Matrices
////////////////////////

disp('Definition de la matrice A');

// Matrices diagonales, diagonale inférieure, diagonale centrée,
```

```
// diagonale supérieure.

for i=1:M-1
A(i,i)=1-lambda;
A(i,i+1)=lambda/2;
A(i+1,i)=lambda/2;
end
A(M,M)=1-lambda;

// Condition initiale
x=(xmin+dx:dx:xmax-dx)';
Unew=exp(-r*T)*max(0,S0*exp((r-sigma**2/2)*T+sigma*x)-K);


////////////////////////////////
// Boucle en temps
////////////////////////////////

disp('Boucle en temps...');

for i=1:M
Unew=A*Unew;
end


////////////////////////////////
// Affichage du prix
////////////////////////////////


i=floor(xmax/dx);
P=Unew(i);
// on peut interpoler pour avoir le prix en S0
disp(P,'prix calculé')

// calcul du vrai prix
d1=(1/(sigma*sqrt(T)))*(log(S0/K)+(r+sigma**2/2)*T);
d2=d1-sigma*sqrt(T);

// case 'call'
  prix=S0*cdfnor("PQ",d1,0,1)-K*exp(-r*T)*cdfnor("PQ",d2,0,1);
// case 'put'
// prix=-(S0-K*exp(-r*T))+(S0*cdfnor("PQ",d1,0,1)-K*exp(-r*T)*cdfnor("PQ",d2,0,1));
```

```
disp(prix,'prix exact');
```

1.2 Méthode implicite

On reprend la discrétisation précédente, que l'on peut écrire $\partial_t U_j^n = \frac{1}{2}\partial_x\bar{\partial}_x U_j^n$, mais, en calculant le second membre en $n+1$: $\partial_t U_j^n = \frac{1}{2}\partial_x\bar{\partial}_x U_j^{n+1}$. On obtient

$$U_j^n = -\frac{\lambda}{2}U_{j+1}^{n+1} + (1+\lambda)U_j^{n+1} - \frac{\lambda}{2}U_{j-1}^{n+1}, \;\; 1 \leq j \leq M, \quad \text{et } U_0^{n+1} = U_{M+1}^{n+1} = 0,$$

avec $\lambda = \dfrac{\delta t}{(\delta x)^2}$. Ceci est équivalent à écrire

$$BU^{n+1} = U^n \quad \text{avec} \quad B = \begin{pmatrix} b_2 & b_3 & 0 & 0 & 0 & 0 & 0 \\ b_1 & b_2 & b_3 & 0 & 0 & 0 & 0 \\ 0 & b_1 & b_2 & b_3 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & b_1 & b_2 & b_3 & 0 \\ 0 & 0 & 0 & 0 & b_1 & b_2 & b_3 \\ 0 & 0 & 0 & 0 & 0 & b_1 & b_2 \end{pmatrix}$$

$b_1 = b_3 = -\frac{\lambda}{2}$ et $b_2 = 1+\lambda$.

Remarques :

1. On montre que B est inversible, le schéma vérifie une propriété de stabilité inconditionnelle (stabilité pour tout $\lambda > 0$) et qu'il est convergent.
2. L'erreur est en $O(h^2 + k)$. Ceci suggère que pour calculer efficacement u, il faut prendre $k \approx h^2$.

Exercice Tester cette méthode dans le cas d'un call européen.

1.3 Méthode de Cranck Nicholson

C'est un mélange des deux précédentes, on considère

$$\partial_t U_j^n = \frac{1}{2}\left(\frac{1}{2}\partial_x\bar{\partial}_x U_j^n + \frac{1}{2}\partial_x\bar{\partial}_x U_j^{n+1}\right), \quad \text{pour } 1 \leq j \leq M, \quad \text{et } U_0^{n+1} = U_{M+1}^{n+1} = 0.$$

Ceci est équivalent, en posant $\lambda = \dfrac{\delta t}{(\delta x)^2}$, à écrire

$$-\frac{\lambda}{4}U_{j+1}^{n+1} + (1+\frac{\lambda}{2})U_j^{n+1} - \frac{\lambda}{4}U_{j-1}^{n+1} = \frac{\lambda}{4}U_{j+1}^n + (1-\frac{\lambda}{2})U_j^n + \frac{\lambda}{4}U_{j-1}^n, \;\; 1 \leq j \leq M,$$

et $U_0^{n+1} = U_{M+1}^{n+1} = 0$. En posant

$$B = \begin{pmatrix} b_2 & b_3 & 0 & 0 & 0 & 0 & 0 \\ b_1 & b_2 & b_3 & 0 & 0 & 0 & 0 \\ 0 & b_1 & b_2 & b_3 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & b_1 & b_2 & b_3 & 0 \\ 0 & 0 & 0 & 0 & b_1 & b_2 & b_3 \\ 0 & 0 & 0 & 0 & 0 & b_1 & b_2 \end{pmatrix} \text{ et } A = \begin{pmatrix} a_2 & a_3 & 0 & 0 & 0 & 0 & 0 \\ a_1 & a_2 & a_3 & 0 & 0 & 0 & 0 \\ 0 & a_1 & a_2 & a_3 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & a_1 & a_2 & a_3 & 0 \\ 0 & 0 & 0 & 0 & a_1 & a_2 & a_3 \\ 0 & 0 & 0 & 0 & 0 & a_1 & a_2 \end{pmatrix}$$

avec $b_1 = b_3 = -\frac{\lambda}{4}$, $b_2 = 1 + \frac{\lambda}{2}$, $a_1 = a_3 = \frac{\lambda}{4}$ et $a_2 = 1 - \frac{\lambda}{2}$. Ce problème se réécrit

$$BU^{n+1} = AU^n, \quad \text{ou encore } U^{n+1} = B^{-1}AU^n.$$

Remarques

1. On montre que B est inversible, le schéma vérifie une propriété de stabilité inconditionnelle (stabilité pour tout $\lambda > 0$) et qu'il est convergent.
2. L'erreur est en $O(h^2+k^2)$. Ceci suggère que pour calculer efficacement u, il faut prendre h et k du même ordre. Ceci représente une amélioration significative du schéma précédent. Ce schéma est généralement choisi pour ce genre de problèmes.