Méthodes Déterministes en Finance

TP du 12 Décembre 2011 [1]

Options américaines

On cherche une approximation numérique de la fonction put américain $p = p(t,s)$, $(t,s) \in \Omega := (0,T) \times (0, S_{max})$ solution du système d'inéquation aux dérivées partielles suivant:

$$\min(\partial_t p + \mathcal{A}p,\ p - \varphi) = 0, \quad \text{dans } \Omega, \tag{1a}$$
$$p(t, S_{max}) = 0, \quad t \in (0,T), \tag{1b}$$
$$p(0,s) = \varphi(s) \tag{1c}$$

avec $\varphi(s) = (K-s)_+$ et $\mathcal{A}v := -\frac{\sigma^2}{2}s^2\ \partial_{s,s}v - rs\ \partial_s v + rv$. où σ, r, K sont des constantes strictement positives. *Pour les applications numériques on prendra les paramètres financiers suivant:* $K = 100$, $S_{max} = 200$, $T = 1$, $\sigma = 0.2$ et $r = 0.1$.

1. Télécharger le fichier exécutable `amer_Q.sce`. Au début du fichier, le paramètre `SCHEMA='EE'` indique qu'on calcule le prix du put Européen par un un schéma d'Euler Explicite (différences finies décentrées à droite). On désire l'adapter à l'option américaine.

On considère un maillage uniforme en espace et en temps. $[0, S_{max}]$ est maillé avec I points intérieurs: $s_j = jh$, $h = S_{max}/(I+1)$, et $t_n = n\delta t$, $0 \le n \le N$. On cherche P_j^n une approximation de $P(t_n, s_j)$. On choisi de travailler avec comme inconnue le vecteur

$$P^n := \begin{pmatrix} P_0^n \\ \vdots \\ P_I^n \end{pmatrix}.$$

2. Schema d'Euler Explicite. On considère le schéma explicite décentré à droite: $P_j^0 := \varphi(s_j)$ puis pour $n = 0, \dots, N-1$, on calcule (P_j^{n+1}) par

$$\min\left(\frac{P_j^{n+1} - P_j^n}{\delta t} + \frac{\sigma^2}{2}s_j^2\frac{-P_{j-1}^n + 2P_j^n - P_{j+1}^n}{h^2} - rs_j\frac{P_{j+1}^n - P_j^n}{h} + rP_j^n,\ P_j^{n+1} - \varphi(s_j)\right) = 0. \qquad 0 \le j \le I,$$

avec $P_{I+1}^n = 0$ (et aussi par convention $P_{-1}^n = 0$). On note A la matrice de discrétisation de $\mathcal{A}$, de taille $I+1$, t.q.

$$(AP)_j := +\frac{\sigma^2}{2}s_j^2\frac{-P_{j-1} + 2P_j - P_{j+1}}{h^2} - rs_j\frac{P_{j+1} - P_j}{h} + rP_j, \quad 0 \le j \le I,$$

(avec la convention $P_{-1} = 0$).

[1] *http://people.math.jussieu.fr/∼boka/enseignement/2011/enpc/tp_amer/*

Notons aussi g le vecteur de composantes $g_j := \varphi(s_j)$. On obtient alors l'écriture équivalente du schema (EE), dans $\mathbb{R}^{I+1}$:

$$\min(\frac{P^{n+1}-P^n}{\delta t} + AP^n,\ P^{n+1}-g) = 0, \quad n = 0,\dots,N-1, \qquad (2)$$
$$P^0 = g.$$

On vérifie que l'itération principale s'écrit aussi

$$P^{n+1} = \max(P^n - \delta t A P^n, g).$$

• Programmer ce schema, noté `'AMER-EE'`.
On commencera par changer la variable `SCHEMA` à la valeur `'AMER-EE'`, puis adapter la ligne correspondant dans la boucle principale.
• Vérifier que le programme donne une solution stable avec les paramêtres `I+1=20` et `N=10`. Observer qu'il y a instabilité dans le cas de `I+1=50`.

3. Schéma d'Euler Implicite. On désire programmer le schéma d'Euler Implicite (EI) en temps qui s'écrit naturellement comme suit:

$$\min(\frac{P^{n+1}-P^n}{\delta t} + AP^{n+1},\ P^{n+1}-g) = 0, \quad n = 0,\dots,N-1, \qquad (3)$$
$$P^0 = g.$$

Posons $B = I + \delta t A$ et $b = P^n$: pour chaque n, on doit trouver $P^{n+1} = x$ solution du système non linéaire

$$\min(Bx - b, x - g) = 0. \qquad (4)$$

Plusieurs méthodes permettent de résoudre ce problème.

3.1. Algorithme PSOR (Projected Successive Over Relaxation).
Cette méthode itérative est basée sur une décomposition de $B = L + U$ où L est triangulaire inférieure et U triangulaire supérieure. [2]

• Pour tester la méthode, mettre `SCHEMA='AMER-EI-PSOR'`, et télécharger le fichier `PSOR.sci` qui contient la fonction `PSOR`. Pour que le programme reconnaisse la fonction, on a rajouté dans la boucle principale une instruction du type `getf PSOR.sci`.
• Observer un ralentissement de la méthode lorsque I est plus grand. (Tester par exemple avec $\sigma = 0.3$, $N = 10$, $I+1 = 100$; oberver un nombre important d'itérations de l'algorithme PSOR à chaque itération en temps).

3.2. Algorithme de Brennan et Schwartz - ou méthode PUL (ou méthode UL avec Projection)
Il existe une méthode "directe" pour résoudre $\min(Bx - b, x - g) = 0$, lorsque la

[2] On rappelle que cette méthode converge si par ex. B est à diagonale stritement dominante et vérifie $B_{i,i} > 0\ \forall i$.

solution cherchée à un "profil" particulier ($\exists i_0$, $x_i = g_i\ \forall i \leq i_0$ et $x_i > g_i\ \forall i > i_0$).[3] L'idée est d'écrire une décomposition de type $B = UL$ (L: matrice triangulaire inférieure et U: matrice triangulaire supérieure avec $U_{ii} = 1$). Puis sous l'hypothèse de profil (et en supposant aussi $U^{-1} \geq 0$) on peut montrer l'équivalence

$$\min(ULx - b, x - g) = 0 \Leftrightarrow \min(Lx - U^{-1}b, x - g) = 0.$$

Enfin, la deuxième écriture possède une résolution explicite simple, qui porte le nom d'algorithme de Brennan et Schwartz.

• Télécharger le fichier `PUL.sci`. Mettre `SCHEMA='AMER-EI-UL'`. Tester la méthode avec $N = 10$, $I + 1 = 50$.
• Reprendre le test avec le payoff particulier $P_0(s) = K\, 1_{50 \leq s \leq 100}$. Vérifier dans ce cas qu'on n'a pas $min(Bx - b, x - g) \neq 0$, dès la première itération $n = 0$.

3.3. Méthode de Newton [4] On veut résoudre $F(x) = 0$ par une méthode de type Newton, avec

$$F(x) := \min(Bx - b, x - g).$$

Plus précisement, on considère l'algorithme suivant: itérer sur $k \geq 0$ (x^0 point de départ à choisir)

$$x^{k+1} = x^k - F'(x^n)^{-1} F(x^k),$$

jusqu'à ce que $F(x^{k+1}) \simeq 0$. On prendra la définition suivante de $F'(x^k)$ (ligne par ligne):

$$F'(x^k)_{i,j} = \begin{cases} B_{i,j} & \text{si } (Bx^k - b)_i \leq (x^k - g)_i, \\ \delta_{i,j} & \text{sinon.} \end{cases}$$

• Programmer cet algorithme en complétant la fonction `Newton_Q.sci` (fichier à télécharger), apres avoir choisi `SCHEMA='AMER-EI-N'` dans le programme principal.
• Tester la méthode avec $N = 10$, $I + 1 = 50$ (Payoff classique).
• La méthode fonctionne-t-elle encore avec le payoff particulier $P_0(s) = K\, 1_{50 \leq s \leq 100}$?

4. Schéma de splitting. On propose une méthode simplifiée:

$$\text{Calculer } P^{n+1,(1)} \text{ t.q.} \quad \frac{P^{n+1,(1)} - P^n}{\delta t} + AP^{n+1,(1)} = 0, \tag{5}$$

$$\text{Calculer } P^{n+1} \text{ t.q.} \quad P^{n+1} = \max(P^{n+1,(1)}, g) \tag{6}$$

• Programmer cette méthode (`SCHEMA='EI-SPLIT'`). Quel avantage voyez vous par rapport à la méthode de Newton ?
• Proposer des schémas de type Crank-Nicolson (avec schéma centré), par une approche de Newton et une approche par splitting. Les tester avec les paramêtres $N = 100$, $I = 500$. (Solutions: $P_{CN} = 4.8156$, $P_{CN-Splitting} = 4.8056$; valeur de référence: $P = 4.8162$).

[3] Voir Jaillet, Lamberton, Lapeyre, 1990, dans le cas d'une approche par éléments finis.

[4] Cette méthode est équivalente à l'algorithme de Howard - ou méthode d'"Itération sur les Politiques", et est aussi connue sous le nom de méthode "Primale-Duale" pour des problèmes d'obstacle.