Rapport du projet de mathématique

Modélisation d'un marché financier et détermination du prix et de la couverture d'options européennes

Thomas Meyer*- El Mehdi Kossir†- Romain Beuzelin‡

Année 2019-2020
Semestre 2
UE PROJ

Mai 2020

Table des matières

*thomasmeyer@outlook.fr
†m.kossir@outlook.fr
‡romain.beuzelin@ensiie.fr

Le marché financier est composé de deux actifs :
— Un actif sans risque (S_t^0) que l'on suppose connu dès l'instant initial 0.
— Un actif risqué (S_t) , à l'instant t qui est une variable aléatoire.

1 Modèle de Cox-Ross-Rubinstens

Le but est de déterminer le prix de l'option a une date fixé T appelée date de maturité. On va utiliser le modèle de Cox-Ross-Rubinstens qui va nous permettre de modéliser le prix de manière discrète.

Nous supposons que les prix des deux actifs sur le marché évoluent de manière discrète et ne changent qu'aux dates $(t_i)0 \leq i \leq N$ avec $t_0 = 0$, $t_N = T$ et $t_{i+1} - t_i = \sigma$ pour $i \in (0, \ldots, N-1)$ où $\sigma = \frac{T}{N}$.

Le prix est calculé sous forme d'espérance.
— Le prix de l'actif risqué est : $S_{t_i}^0 = (1 + r_N)^i$ avec $0 \leq r_N$ une constante fixé à la date 0 qui représente le taux sans risque.
— Le prix de l'actif sans risque est : $S_{t_i^N} = T_i^N S_{t_{i-1}}^N$ avec $S_0^N = s > 0$.

Question 1

Déterminer q_N revient à déterminer la probabilité risque-neutre $\mathbb{Q}$ telle que $\mathbb{E}_Q\left[T_1^{(N)}\right] = 1 + r_N$, ce qui revient à dire :

$$(1 + h_N)q_N + (1 + b_N)(1 - q_N) = 1 + r_N$$

et par conséquent :

$$q_N = \frac{r_N - b_N}{h_N - b_N}$$

Question 2

Tout d'abord, remarquons que pour chaque $0 \leq k \leq N$, on a $S_{t_k}^{(N)} = s\prod_{i=1}^k T_i^{(N)}$ qui est une variable aléatoire de support :

$$\Delta_{S_{t_k}^{(N)}} = \{s(1 + h_N)^i(1 + b_N)^{k-i}; 0 \leq i \leq k\}$$

fini avec $k + 1$ valeurs distinctes possibles. Pour la suite, on va noter $\delta_{t_k}^i = s(1 + h_N)^i(1 + b_N)^{k-i}$ où i est le nombre d'évolutions favorables $(i + h_N)$ rencontrées. Pour chaque variable $S_{t_k}^{(N)}$, on a la probabilité P définie par :

$$P\left(S_{t_k}^{(N)} = \delta_{t_k}^i\right) = q_N^k(1 - q_N)^{k-i}\binom{k}{i}$$

qui s'obtient en sachant que l'on a i évolutions favorables $(1+h_N)$ avec la probabilité q_N, que l'on a $k-i$ évolutions $(1+b_N)$ avec la probabilité $(1-q_N)$ et que l'on a $\binom{k}{i}$ façons de choisir les variables $T_i^{(N)}$ pour obtenir ces évolutions.

Par conséquent, comme le théorème de transfert nous donne :

$$\begin{aligned}\mathbb{E}_{\mathbb{Q}}\left[f(S_{t_N}^{(N)})\right] &= \sum_{i=0}^{N} f(\delta_{t_N}^i) P\left(S_{t_k}^{(N)} = \delta_{t_k}^i\right) \\ &= \sum_{i=0}^{N} f\left(s(1+h_N)^i(1+b_N)^{N-i}\right) q_N^i (1-q_N)^{N-i} \binom{N}{i}\end{aligned}$$

on en déduit que le prix $p_{(N)}$ de l'option s'exprime de la manière suivante :

$$p_{(N)} = \frac{1}{(1+r_N)^N}\left[\sum_{i=0}^{N} f\left(s(1+h_N)^i(1+b_N)^{N-i}\right) q_N^i (1-q_N)^{N-i} \binom{N}{i}\right]$$

1.1 Premier pricer

Question 3

Pour obtenir numériquement le prix $p_{(N)}$, on implémente en Python l'algorithme 1 `price1` suivant qui consiste simplement à utiliser une boucle itérative pour sommer les N termes de l'expression de $p_{(N)}$ de la question 2. On mémorise R, H, B et q_n pour simplifier l'écriture du code et améliorer la vitesse d'exécution. On utilise l'expression de q_n déterminée à la question 1.

Algorithme 1 : price1

Données : $N \in \mathbb{N}^*, r_N \in \mathbb{R}, h_N \in \mathbb{R}, b_N \in \mathbb{R}, s \in \mathbb{R}, f : \mathbb{R} \mapsto \mathbb{R}$
$S \leftarrow 0$
$R \leftarrow 1 + r_N$
$H \leftarrow 1 + h_N$
$B \leftarrow 1 + b_N$
$q_N \leftarrow \frac{r_N - b_N}{h_N - b_N}$
pour *i allant de 0 à N* **faire**
  $S \leftarrow f(sH^iB^{N-i})q_N^i(1-q_N)^{N-i}\binom{N}{i}$
fin
Résultat : $p_{(N)} \leftarrow \frac{S}{R^N}$

Question 4

En utilisant les paramètres $s = 100$, $h_N = 0.05$, $b_N = -0.05$, $r_N = 0.01$, $N = 30$ et la fonction

$f(x) = \max(100 - x, 0)$, la fonction `price1` de la question 3 nous donne :

$$p_{(N)} \approx 1.614$$

1.2 Deuxième pricer

Question 5

Dans cette question, on implémente en Python l'algorithme 2 `price2` suivant pour obtenir numériquement le prix $p_{(N)}$.

On utilise le vecteur m pour mémoriser les valeurs des différents v_k. Au départ, m est de taille $N+1$ et contient les valeurs de v_N :

$$m = \{f(s(1+h_N)^i(1+b_N)^{N-i}); 0 \leq i \leq N\}$$

Puis, à chaque fin d'itération de la boucle d'indice j, m est de taille j et contient les valeurs de v_{j-1} calculées à partir des anciennes valeurs de m associées à v_j. On adopte la convention $m[i] = v_k(\delta^i_{t_k})$ pour un k donné. À la fin, m est de taille 1 et contient $p_{(N)}$.

Algorithme 2 : price2

Données : $N \in \mathbb{N}^*, r_N \in \mathbb{R}, h_N \in \mathbb{R}, b_N \in \mathbb{R}, s \in \mathbb{R}, f : \mathbb{R} \mapsto \mathbb{R}$
$R \leftarrow 1 + r_N$
$H \leftarrow 1 + h_N$
$B \leftarrow 1 + b_N$
$q_N \leftarrow \frac{r_N - b_N}{h_N - b_N}$
pour *i allant de 0 à N* **faire**
  $m[i] \leftarrow f\left(sH^iB^{N-i}\right)$
fin
pour *j allant de N à 1* **faire**
  pour *i allant de 0 à j* **faire**
    $m[i] \leftarrow \frac{m[i+1]q_N + m[i](1-q_N)}{R}$
  fin
fin
Résultat : $p_{(N)} \leftarrow m[0]$

Question 6

En utilisant les paramètres $s = 100$, $h_N = 0.05$, $b_N = -0.05$, $r_N = 0.01$, $N = 3$ et la fonction $f(x) = \max(100 - x, 0)$, la fonction `price2` nous donne :

$$p_{(N)} \approx 2.35$$

et si on utilise $N = 30$, on confirme le résultat obtenu à la question 4 : $p_{(N)} \approx 1.614$

Dans le graphique 1, on représente l'arbre des $v_k(x)$ obtenu via la fonction `price2`. Les flèches rouges et vertes correspondent respectivement à une évolution défavorable ou favorable de l'actif risqué.

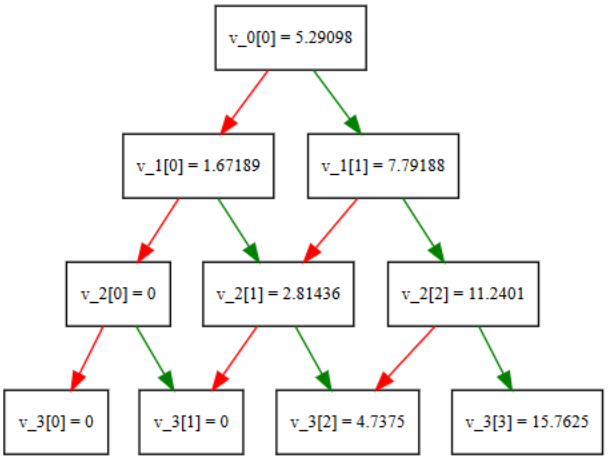

FIGURE 1 – Arbre d'évolution du prix de l'option dans le temps en fonction de l'évolution du prix de l'actif risqué. En rouge, des évolutions défavorables. En vert, des évolutions favorables.

1.3 Comparaison

Question 7

Pour cette question, on garde les paramètres s, h_N, b_N et r_N des questions 4 et 6. On va prendre la fonction $f(x) = \max(x - 100, 0)$ et $5 \leq N \leq 15$. On obtient les résultats suivant, en remarquant que les deux valeurs obtenues sont presque identiques :

`price1`	`price2`
$p_{(N)} = 2.343023$	$p_{(N)} = 2.340696$

1.4 La couverture

Dans cette partie on va déterminer la couverture qui est la stratégie d'investissement du vendeur de l'option afin de garantir les paiements.

Question 8

Pour la suite de cette question, notons $S = S_{t_{N-1}}$ pour simplifier les équations. À la date de maturité, on doit avoir le système d'équation (E) suivant :

$$\begin{aligned} \alpha_{N-1}(S)(1+h_N)S + \beta_{N-1}(S)(1+r_n)^N &= f\left((1+h_N)S\right) \\ \alpha_{N-1}(S)(1+b_N)S + \beta_{N-1}(S)(1+r_n)^N &= f\left((1+b_N)S\right) \end{aligned}$$

ce qui nous donne :

$$\begin{aligned} \alpha_{N-1}(S)(1+h_N)S - \alpha_{N-1}(S)(1+b_N)S &= f\left((1+h_N)S\right) - f\left((1+b_N)S\right) \\ \alpha_{N-1}(S) &= \frac{f\left((1+h_N)S\right) - f\left((1+b_N)S\right)}{S(h_N - b_N)} \end{aligned}$$

Donc, en introduisant l'expression de $\alpha_{N-1}(S)$ dans la première équation, on obtient :

$$\beta_{N-1}(S)(1+r_n)^N = f((1+h_N)S) - \alpha_{N-1}(S)(1+h_N)S$$

et par conséquent, on trouve finalement :

$$\begin{aligned} \alpha_{N-1}(S) &= \frac{f\left((1+h_N)S\right) - f\left((1+b_N)S\right)}{S(h_N - b_N)} \\ \beta_{N-1}(S) &= \frac{f((1+h_N)S)(-1-b_N) + (1+h_N)f((1+b_N)S)}{(h_N - b_N)(1+r_N)^N} \end{aligned}$$

Question 9

Pour les autres dates t_k, on doit avoir le système d'équation (E) suivant :

$$\begin{aligned} \alpha_{k-1}(S)(1+h_N)S + \beta_{k-1}(S)(1+r_n)^k &= v_k\left((1+h_N)S\right) \\ \alpha_{k-1}(S)(1+b_N)S + \beta_{k-1}(S)(1+r_n)^k &= v_k\left((1+b_N)S\right) \end{aligned}$$

ce qui nous donne :

$$\begin{aligned} \alpha_{k-1}(S)(1+h_N)S - \alpha_{k-1}(S)(1+b_N)S &= v_k\left((1+h_N)S\right) - v_k\left((1+b_N)S\right) \\ \alpha_{k-1}(S) &= \frac{v_k\left((1+h_N)S\right) - v_k\left((1+b_N)S\right)}{S(h_N - b_N)} \end{aligned}$$

et en introduisant l'expression de $\alpha_{k-1}(S)$ dans la première expression , on a

$$\beta_{k-1}(S) = \frac{v_k((1+h_N)S)(-1-b_N) + (1+h_N)v_k((1+b_N)S)}{(h_N - b_N)(1+r_N)^N}$$

Question 10

2 Modèle de Black-Scholes

2.1 Le modèle

Question 11

En appliquant la formule d'Ito à $ln(S_t)$, on obtient :

$$d\ln S_t = \frac{dS_t}{S_t} - \frac{|\sigma S_t|^2}{2S_t^2}dt$$

Puis en utilisant l'équation différentielle vérifiée par S_t :

$$\begin{aligned} d\ln S_t = \frac{dS_t}{S_t} - \frac{|\sigma S_t|^2}{2S_t^2}dt &\iff d\ln S_t = rdt + \sigma dB_t - \frac{|\sigma S_t|^2}{2S_t^2}dt \\ &\iff d\ln S_t = rdt + \sigma dB_t - \frac{|\sigma|^2}{2}dt \\ &\iff \frac{dln(S_t)}{dt} = r + \sigma\frac{dB_t}{dt} - \frac{|\sigma|^2}{2} \\ &\iff \ln S_t = rt + \sigma dB_t - \frac{|\sigma|^2}{2}t + cste \end{aligned}$$

Or, en $t = 0$, on a : $\ln(s) = cste$, donc :

$$\begin{aligned} d\ln S_t = \frac{dS_t}{S_t} - \frac{|\sigma S_t|^2}{2S_t^2}dt &\iff \ln S_t = t\left(r - \frac{|\sigma|^2}{2}\right) + \sigma B_t + ln(s) \\ &\iff S_t = s\exp\left(\left(r - \frac{|\sigma|^2}{2}\right)t + \sigma B_t\right) \end{aligned}$$

2.2 Le pricer par la méthode de Monte-Carlo

Question 12

Dans cette question, on implémente en Python l'algorithme 3 `price3` suivant pour obtenir numériquement l'estimateur $\hat{p}_{(n)}$ du prix de l'option p. Il s'agit simplement d'utiliser une boucle

itérative pour sommer les n termes de l'expression de $\hat{p}_{(n)}$.

Algorithme 3 : price3

Données : $n \in \mathbb{N}^*, s \in \mathbb{R}, r \in \mathbb{R}, \sigma \in \mathbb{R}, T \in \mathbb{R}, f : \mathbb{R} \mapsto \mathbb{R}$
$S \leftarrow 0$
$A \leftarrow \left(r - \frac{\sigma^2}{2}\right) T$
$B \leftarrow \sigma\sqrt{T}$
pour *i allant de 1 à n* **faire**
  $S \leftarrow S + f\left(s \exp\left(A + B\xi_i\right)\right)$
fin
Résultat : $\hat{p}_{(n)} \leftarrow \frac{e^{-rT}}{n} S$

On mémorise $A = \left(r - \frac{\sigma^2}{2}\right) T$ et $B = \sigma\sqrt{T}$ pour faciliter l'écriture de l'algorithme et pour accélérer sa vitesse d'exécution. Notons également qu'on a placé e^{-rT} en facteur de la somme. On utilise la fonction `random.gauss(mu, sigma)` du module `random` de la bibliothèque standard de Python.

Question 13

On donne dans la figure 2 le graphique pour le prix $\hat{p}_{(n)}$ obtenu via la fonction `price3` avec $r = 0.01, \sigma = 0.01, s = 100, T = 1, f(x) = \max(100 - x, 0)$ et $n = 10^5 k$ avec $1 \leq k \leq 10$.

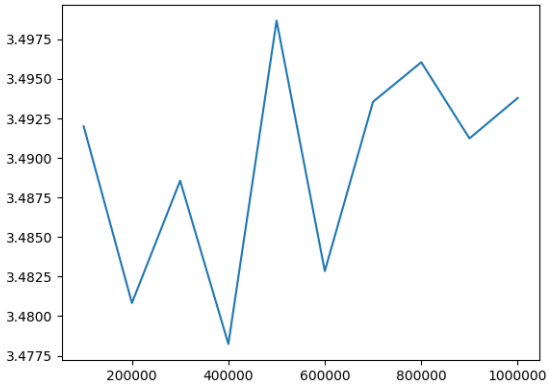

FIGURE 2 – Graphique d'évolution de $\hat{p}_{(n)}$ en fonction de $n = 10^5 k$ avec $1 \leq k \leq 10$ et les autres paramètres de la question 13.

Question 14

Comme $\hat{p}_{(n)}$ est la moyenne empirique de la variable aléatoire $\exp(-rT)f(S_t)$ où nous avons pris $B_t = \sqrt{T}\xi_i$. Donc, en appliquant le théorème central limite, on en déduit que $\hat{p}_{(n)}$ converge presque sûrement vers $\mathbb{E}[\exp(-rT)f(S_t)]$.

2.3 Le pricer par formule fermée

Question 15

Dans cette question, on implémente en Python l'algorithme 4 `put` suivant pour obtenir numériquement le prix p de l'option. Ici, il suffit d'appliquer la formule de p obtenue dans l'énoncé.

Algorithme 4 : put

Données : $s \in \mathbb{R}, r \in \mathbb{R}, \sigma \in \mathbb{R}, T \in \mathbb{R}, K \in \mathbb{R}$

$d \leftarrow \frac{1}{\sigma\sqrt{T}}\left[\ln\left(\frac{s}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T\right]$

Résultat : $p \leftarrow -sF(-d) + Ke^{-rT}F(-d + \sigma\sqrt{T})$

Pour calculer les valeurs de formule de répartition F de la loi normale $\mathcal{N}(0,1)$ on utilise la fonction `cdf(x, loc=0, scale=1)` de `scipy.stats.norm` dans la bibliothèque ScyPi de Python.

Question 16

En prenant les paramètres $r = 0.01$, $\sigma = 0.1$, $s = 100$, $T = 1$ et $K = 100$, la fonction `put` de la question 15 nous donne :

$$p = 1.0752585217052843 \times 10^{-60}$$

Question 17

Sur le graphique 3, On remarque que la courbe bleue de $\hat{p}_{(n)}$ évolue autour de la valeur constante p en orange et converge vers cette valeur conformément au résultat de la question 14.

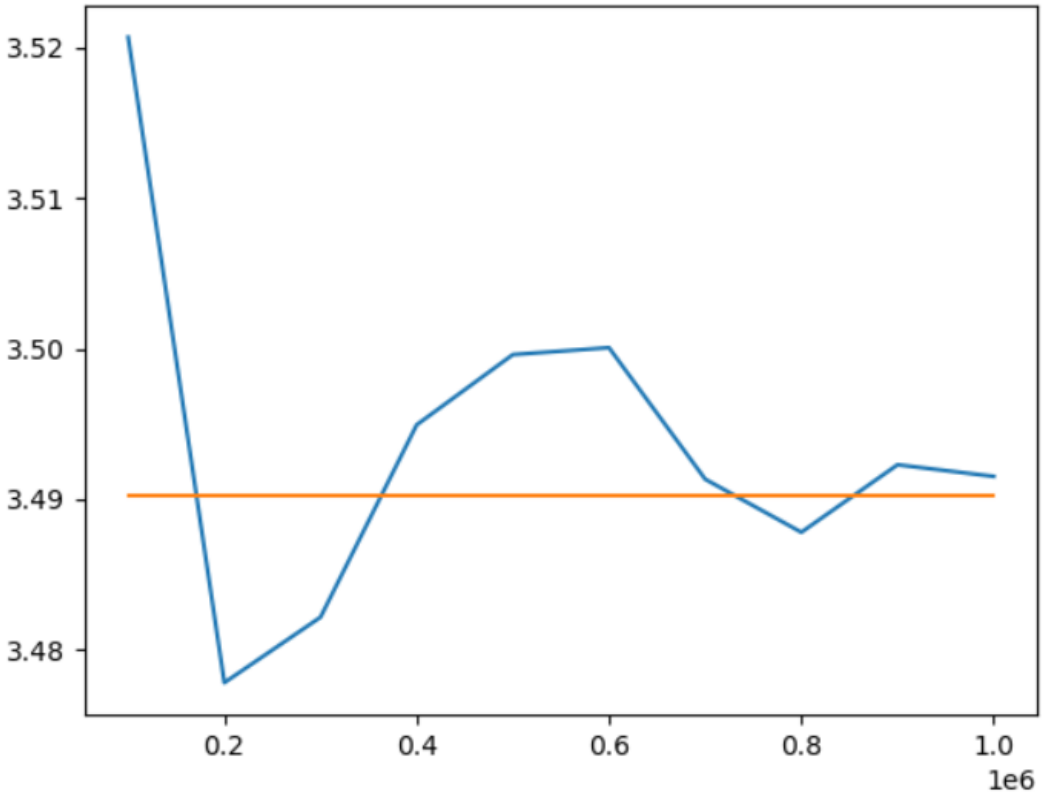

FIGURE 3 – Graphique représentant $\hat{p}_{(n)}$ et p en fonction de n : $\hat{p}_{(n)}$ est représenté en bleu et p en orange.

Question 18

On trace sur le graphique 4 l'évolution du prix de l'option payant $\max(K - S_T, 0)$ en T calculé par `put` en fonction du prix de l'actif risqué $s = 20k$ avec $1 \leq k \leq 10$ et en fonction da la maturité $T \in \{\frac{1}{12}, \frac{1}{6}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\}$.

On remarque sur le graphique 4 que plus le prix de l'actif risqué s est important, plus le prix de l'option est bas.

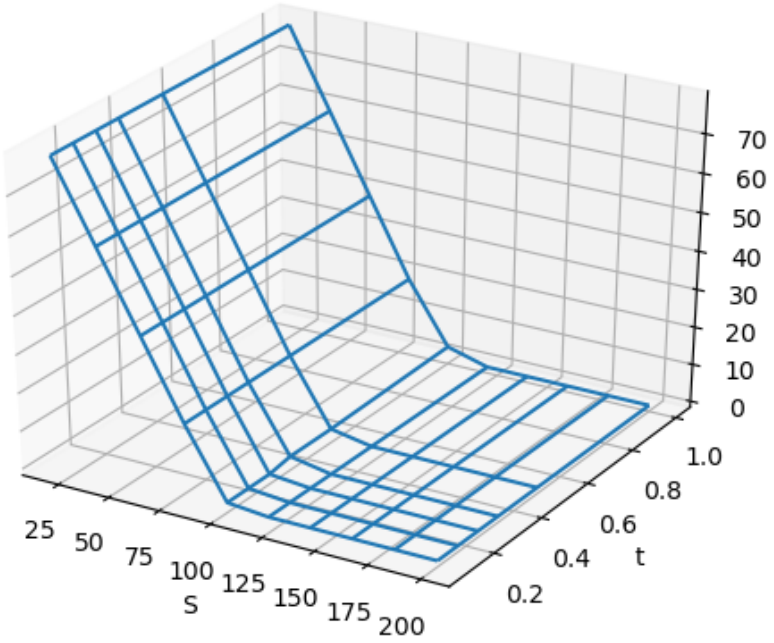

FIGURE 4 – Graphique représentant le prix de l'option en fonction de s et T.

3 Convergence des prix

Question 19

On remarque que le price2 semble converger vers une valeur finie. On a choisi $K = 182$ pour le calcul de p.

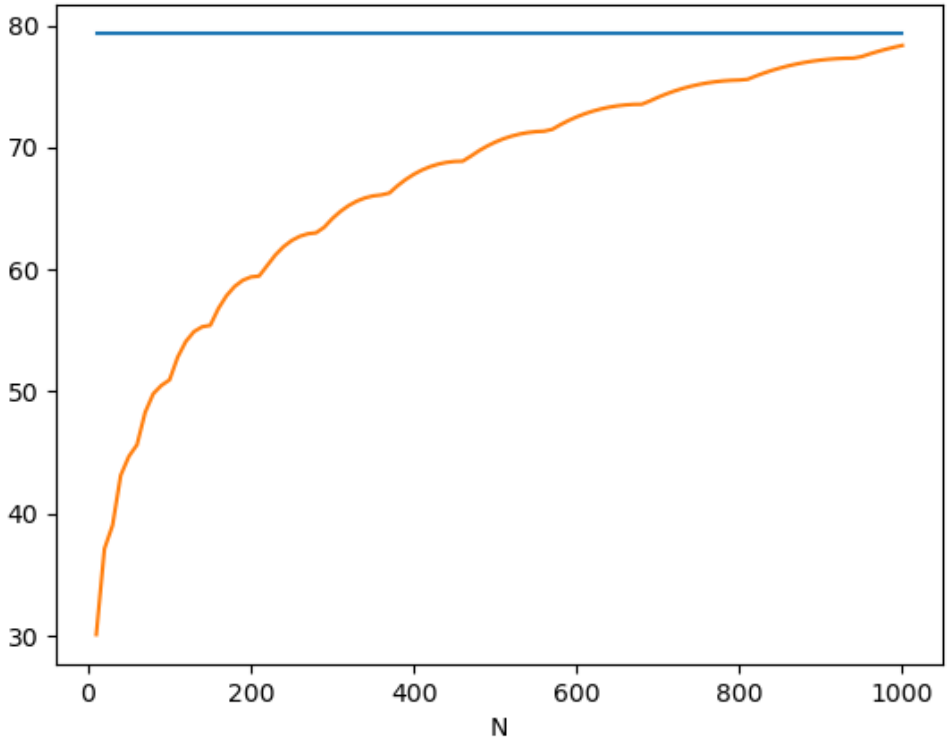

FIGURE 5 – Graphique représentant la valeur du price 3 en fonction de N

4 EDP de Black-Scholes

Question 20

On prend l'équation aux dérivées partielles vérifiée par P au point (t_n, s_i) :

$$\frac{\partial P}{\partial t}(t_n, s_i) + rS_i \frac{\partial P}{\partial t}(t_n, s_i) + \frac{1}{2}\sigma^2 s_i^2 \frac{\partial^2 P}{\partial^2 S}(t_n, s_i) = rP(t_n, s_i)$$

4.1 Méthode implicite

Puis, pour la méthode implicite, on remplace les termes avec les expressions de l'énoncé

$$\frac{\partial P}{\partial t}(t_n, s_i) + rS_i \frac{\partial P}{\partial t}(t_n, s_i) + \frac{1}{2}\sigma^2 s_i^2 \frac{\partial^2 P}{\partial^2 S}(t_n, s_i) = rP(t_n, s_i)$$

ce qui est équivalent à :

$$\frac{1}{\Delta T}(P_{n+1,i} - P_{n,i}) + \frac{rS}{\Delta S}(P_{n,i} - P_{n,i-1}) + \frac{\sigma^2 s_i^2}{2\Delta S^2}(P_{n,i+1} - 2P_{n,i} + P_{n,i+1}) = rP_{n,i}$$

Puis en isolant $P_{n+1,i}$ on obtient :

$$P_{n+1,i} = aP_{n,i-1} + bP_{n,i} + cP_{n,i+1}$$

avec :

$$\begin{aligned} a_i &= \frac{\Delta T}{\Delta s} rS_i - \frac{\Delta T}{\Delta s^2}\frac{\sigma^2 S_i^2}{2} \\ b_i &= 1 - \Delta T \frac{rS_i}{\Delta S} + \frac{\Delta T}{\Delta s^2}\sigma^2 S_i^2 + r\Delta T \\ c_i &= -\frac{\Delta T}{\Delta s^2}\frac{\sigma^2 S_i^2}{2} \end{aligned}$$

On obtient alors une relation matricielle entre P_{n+1} et P_n qui est $P_{n+1} = AP_n + B_n$ où A est une matrice tridiagonale :

$$A = \begin{pmatrix} b_i & c_i & 0 & ... & 0 \\ a_i & b_i & c_i & ... & 0 \\ 0 & ... & ... & ... & 0 \\ ... & 0 & a_i & b_i & c_i \\ 0 & ... & 0 & a_i & b_i \end{pmatrix}$$

et B_n est un vecteur colonne :

$$B_n = \begin{pmatrix} a_1 K \exp r(t_n - T) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

On en déduit que pour obtenir P_n à partir de P_{n+1}, nous allons résoudre le système linéaire : $P_{n+1} - B_n = AP_n$

Pour cela, on réalisera la décomposition LU de la matrice A, pour résoudre les deux systèmes linéaire avec L triangulaire inférieure et U triangulaire supérieure.

4.2 Méthode explicite

Pour la méthode explicite, on suit la même démarche en prenant :

$$\frac{\partial P}{\partial t} = \frac{P_{n,i} - P_{n-1,i}}{\Delta t}$$

Les calculs aboutissent à :

$$P_{n-1,i} = a_i P_{n,i-1} + b_i P_{n,i} + c_i P_{n,i+1}$$

avec :

$$\begin{aligned} a_i &= \frac{\Delta t \sigma^2 s_i^2}{2\Delta s_i^2} - \frac{\Delta t r s_i}{\Delta s} \\ b_i &= 1 - r\Delta t + \frac{\Delta T}{\Delta s} r s_i - \frac{\Delta t}{\Delta s^2}\sigma^2 s_i^2 \\ c_i &= \frac{\sigma^2 s_i^2}{2}\frac{\Delta t}{\Delta s} \end{aligned}$$

On a alors $P_{n-1} = AP_n + B_n$ avec A et B telles qu'elles sont définies plus haut avec les nouveaux coefficients. On notera que ce schéma est bien explicite car on n'effectue pas de résolution de systèmes d'équations pour trouver P_{n-1} à partir de P_n.

4.3 Méthode de Crank-Nicolson

Pour la méthode de Crank-Nicolson, on remplace les termes suivants :

$$\begin{aligned} \frac{\partial P}{\partial t} &= \frac{P_{n+1,i} - P_{n,i}}{\Delta T} \\ \frac{\partial P}{\partial S} &= \frac{1}{4\Delta s}(P_{n+1,i+1} - P_{n+1,i-1} - (P_{n,i+1} - P_{n,i-1})) \\ \frac{\partial^2 P}{\partial S^2} &= \frac{1}{2\Delta s^2}(P_{n+1,i+1} - 2P_{n+1,i} + P_{n+1,i-1} + P_{n,i+1} - 2P_{n,i} + P_{n,i-1}) \\ P &= (P_{n+,i} - P_{n,i}) \end{aligned}$$

en reportant ces termes dans l'équation :

$$a_{1,i}P_{n+1,i-1} + b_{1,i}P_{n+1,i} + c_{1,i}P_{n+1,i+1} = a_{0,i}P_{n,i-1} + b_{0,i}P_{n,i} + c_{0,i}P_{n,i+1}$$

avec :

$$\begin{aligned}
a_{1,i} &= \frac{\sigma^2 S^2}{4\Delta s^2} - \frac{r s_i}{4\Delta s} \\
b_{1,i} &= \frac{1}{\Delta t} - \frac{r}{2} - \frac{\sigma^2 S^2}{2\Delta s^2} \\
c_{1,i} &= \frac{\sigma^2 S^2}{4\Delta s^2} + \frac{r s_i}{4\Delta s} \\
a_{0,i} &= -\frac{\sigma^2 S^2}{4\Delta s^2} - \frac{r s_i}{4\Delta s} \\
b_{0,i} &= \frac{1}{\Delta t} - \frac{r}{2} + \frac{\sigma^2 S^2}{2\Delta s^2} \\
c_{0,i} &= \frac{r s_i}{4\Delta s} - \frac{\sigma^2 S^2}{4\Delta s^2}
\end{aligned}$$

On obtient une relation matricielle entre P_{n+1} et P_n qui est $AP_{n+1} = BP_n + C_n$ avec la matrice tridiagonale A :

$$A = \begin{pmatrix} b_{1,i} & c_{1,i} & 0 & ... & 0 \\ a_{1,i} & b_{1,i} & c_{1,i} & ... & 0 \\ 0 & ... & ... & ... & 0 \\ ... & 0 & a_{1,i} & b_{1,i} & c_{1,i} \\ 0 & ... & 0 & a_{1,i} & b_{1,i} \end{pmatrix}$$

la matrice tridiagonale B :

$$B = \begin{pmatrix} b_{0,i} & c_{0,i} & 0 & ... & 0 \\ a_{0,i} & b_{0,i} & c_{0,i} & ... & 0 \\ 0 & ... & ... & ... & 0 \\ ... & 0 & a_{0,i} & b_{0,i} & c_{0,i} \\ 0 & ... & 0 & a_{0,i} & b_{0,i} \end{pmatrix}$$

et le vecteur colonne C :

$$C_n = \begin{pmatrix} a_{0,1} K \exp\left(r(t_n - T)\right) - a_{1,1} K \exp\left(r(t_{n+1} - T)\right) \\ 0 \\ ... \\ 0 \end{pmatrix}$$

On peut alors obtenir P_n à partir de P_{n+1} :

$$AP_{n+1} - C_n = BP_n$$

À chaque itération, on calcule la quantité $Y = AP_{n+1} - C_n$, puis on résout avec une factorisation LU : $Y = BP_n$.

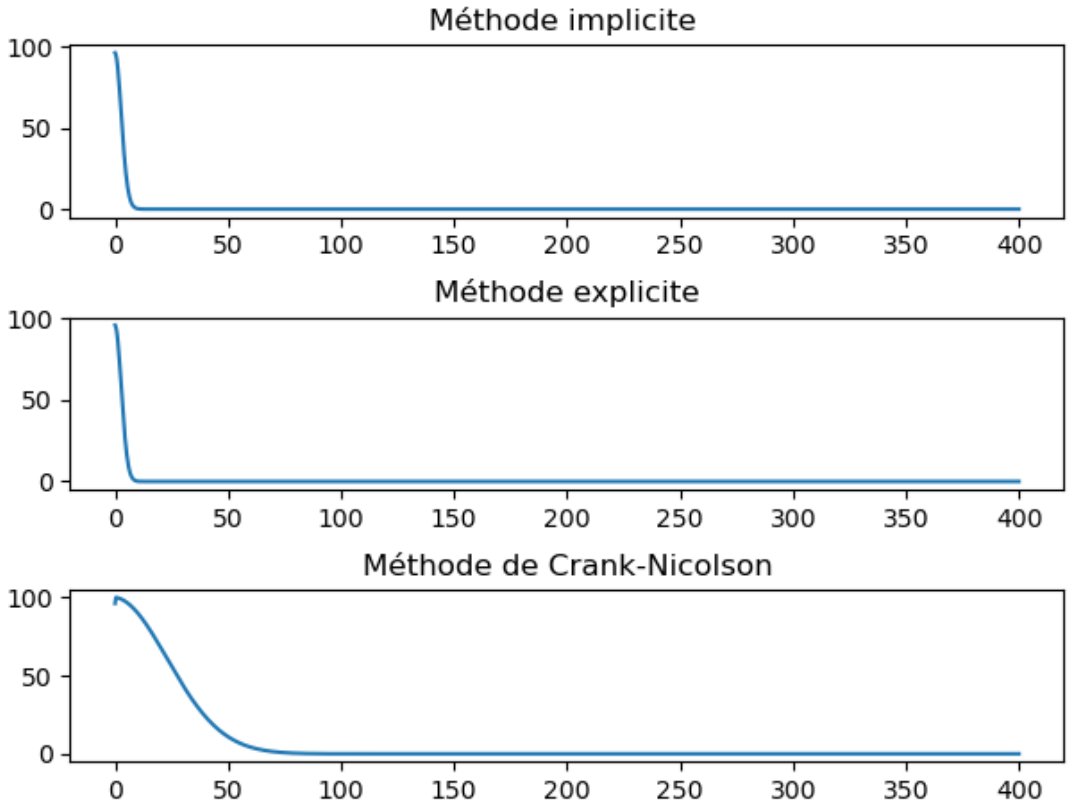

FIGURE 6 – Tracés des trois méthodes - implicite, explicite, de Crank-Nicolson - abordées dans la question 20.