Rapport final - Projet Mathématiques

Khalil BATTIKH, Lucas RODRIGUEZ, Avraham ROSENBERG

Mars/Juin 2021

Introduction

Ce rapport présente nos principaux résultats de recherche du projet mathématiques. Les résultats, démonstrations, pistes de recherches et remarques sont rédigés ci-dessous, tandis que l'implémentation des différentes méthodes numériques sont disponibles sur le Jupyter Notebook ci-joint.

Table des matières

1 Modèle de Cox-Ross-Rubinstein

1.1 Premier pricer

On note $\mathbb{Q}$ la probabilité risque-neutre tel que $\mathbb{E}_{\mathbb{Q}}[T_1^{(N)}] = 1 + r_N$.

Question 1 On cherche à déterminer l'expression de q_N.
Par définition, $\forall k \in \mathbb{N},\ T_k^{(N)} \in \{1+h_N, 1+b_N\}$ avec $q_N := \mathbb{Q}(T_1^{(N)} = 1+h_N)$ pour $N \in \mathbb{N}^*$

$$\begin{aligned}
\mathbb{E}_{\mathbb{Q}}[T_1^{(N)}] = 1 + r_N &\iff (1+h_N)q_N + (1+b_N)(1-q_N) = 1 + r_N \\
&\iff q_N + h_N q_N + 1 - q_N + b_N - b_N q_N = 1 + r_N \\
&\iff q_N(h_N - b_N) + 1 + b_N = 1 + r_N \\
&\iff \boxed{q_N = \frac{r_N - b_N}{h_N - b_N}}
\end{aligned}$$

Question 2 Cherchons le prix $\text{Prix}^{(N)}$ défini par : $\text{Prix}^{(N)} := \frac{1}{(1+r_n)^N}\,\mathbb{E}_{\mathbb{Q}}[f(S_{t_N}^{(N)})]$
On cherche à trouver une bonne expression de $S_{t_N}^{(N)}$. En retravaillant l'expression, on a :

$$\begin{aligned}
S_{t_N}^{(N)} &= T_N^{(N)} S_{t_{N-1}}^{(N)} && \text{par définition} \\
&= T_N^{(N)} T_{N-1}^{(N)} S_{t_{N-2}}^{(N)} \\
&= T_N^{(N)} T_{N-1}^{(N)} \cdots T_1^{(N)} S_{t_0}^{(N)} && \text{par récurrence immédiate}
\end{aligned}$$

On a donc :

$$S_{t_N}^{(N)} = S_{t_0}^{(N)} \prod_{i=1}^{N} T_i^{(N)} \tag{1}$$

or $\forall i \in [\![1, N]\!],\ T_i^{(N)} \in \{1+h_N, 1+b_N\}$ d'où :

$$\forall i \in [\![1, N]\!],\ T_i^{(N)} = (1+h_N)^{X_i}(1+b_N)^{1-X_i}$$

avec $X_i \rightsquigarrow \mathcal{B}(1, q_N)$ (loi de Bernoulli) [1] [2]
D'où en injectant ce résultat dans 1, on a :

$$S_{t_N}^{(N)} = S_{t_0}^{(N)}(1+h_N)^X(1+b_N)^{N-X}$$

où $X = \sum_{i=1}^{N} X_i$.
On remarque que, par construction, $X \rightsquigarrow \mathcal{B}(N, q_N)$ (loi binomiale)
De plus,

$$S_{t_N}^{(N)} = \psi(X)$$

avec :

$$\begin{array}{rccc} \psi & : & \mathbb{R} & \longrightarrow & \mathbb{R} \\ & & x & \longmapsto & S_{t_0}^{(N)}(1+h_N)^x(1+b_N)^{N-x} \end{array}$$

En appliquant le théorème de transfert appliqué à la fonction $f \circ \psi$ relativement à l'espérance $\mathbb{E}_{\mathbb{Q}}$, on trouve :

$$\mathbb{E}_{\mathbb{Q}}[f(S_{t_N}^{(N)})] = \sum_{i=0}^{N} \binom{N}{i} q_N^i (1-q_N)^{N-i} f(s(1+h_N)^i(1+b_N)^{N-i}))$$

Finalement, en multipliant avec la constante donnée dans l'énoncé, on tombe sur le résultat suivant :

$$\forall N \in \mathbb{N}^*,\ \boxed{\text{Prix}^{(N)} = \frac{1}{(1+r_N)^N} \sum_{i=0}^{N} \binom{N}{i} q_N^i (1-q_N)^{N-i} f(s(1+h_N)^i(1+b_N)^{N-i}))}$$

1. $X_i \in [0, 1]$
2. A i fixé, $X_i = 1$ implique que le prix monte et $X_i = 0$ implique que le prix descend.

Remarque 1 La somme démarre à 0 car la variable aléatoire X prend des valeurs dans $[\![0, N]\!]$. Numériquement, prendre $i = 0$ ou $i = 1$ revient au même résultat, du fait du choix de la fonction f.

Question 3 Nous implémentons le premier pricer.

Algorithme 1 : Premier Pricer

Input : N, r_N, h_N,b_N, s, f
$q_N \leftarrow \frac{r_N - b_N}{h_N - b_N}$;
$\text{Prix}^{(N)} = 0$;
for *indice* $i \in [\![0, N]\!]$ **do**
  $s_i \leftarrow (s(1 + h_N)^i (1 + b_N)^{N-i})$;
  $\text{Prix}^{(N)} \leftarrow \text{Prix}^{(N)} + \text{binomial(N,i)} \times f(s_i) \times q_N^i (1 - q_N)^{N-i}$;
end
Output : $\text{Prix}^{(N)} \times \frac{1}{(1+r_N)^N}$

La fonction `binomial(N, i)` définit la fonction de calcul du coefficient binomial de manière itérative

Question 4 Avec les paramètres donnés dans l'énoncé, nous trouvons :

$$\boxed{20.078658207286068}$$

1.2 Second pricer

Question 5 Cherchons une formule explicite, facile à implémenter numériquement de $v_k(S_{t_k}^{(N)})$, en retravaillant l'expression donnée en énoncé (en se débarassant du caractère conditionnel de l'espérance par exemple)

$$\mathbb{E}_{\mathbb{Q}}[v_{k+1}(S_{t_{k+1}}^{(N)}) \mid S_{t_k}^{(N)}] = \mathbb{E}_{\mathbb{Q}}[v_{k+1}(S_{t_k}^{(N)} T_{k+1}^{(N)}) \mid S_{t_k}^{(N)}]$$

Cette égalité est vraie pour toute les valeurs prises par $S_{t_k}^{(N)}$. En particulier pour $S_{t_k}^{(N)} = s_k$

$$\begin{aligned}
\mathbb{E}_{\mathbb{Q}}[v_{k+1}(S_{t_k}^{(N)} T_{k+1}^{(N)}) \mid S_{t_k}^{(N)}] &= v_{k+1}((1+h_N)s_k)\,\mathbb{Q}[S_{t_{k+1}}^{(N)} = (1+h_N)s_k \mid S_{t_k}^{(N)} = s_k] \\
&+ v_{k+1}((1+b_N)s_k)\,\mathbb{Q}[S_{t_{k+1}}^{(N)} = (1+b_N)s_k \mid S_{t_k}^{(N)} = s_k] \\
&= v_{k+1}((1+h_N)s_k)\,\mathbb{Q}[S_{t_k}^{(N)} T_{k+1}^{(N)} = (1+h_N)s_k \mid S_{t_k}^{(N)} = s_k] \\
&+ v_{k+1}((1+b_N)s_k)\,\mathbb{Q}[S_{t_k}^{(N)} T_{k+1}^{(N)} = (1+b_N)s_k \mid S_{t_k}^{(N)} = s_k] \\
&= v_{k+1}((1+h_N)s_k)\,\mathbb{Q}[T_{k+1}^{(N)} = (1+h_N) \mid S_{t_k}^{(N)} = s_k] \qquad \text{voir preuve } (\star) \\
&+ v_{k+1}((1+b_N)s_k)\,\mathbb{Q}[T_{k+1}^{(N)} = (1+b_N) \mid S_{t_k}^{(N)} = s_k] \\
&= v_{k+1}((1+h_N)s_k)\,\mathbb{Q}[T_{k+1}^{(N)} = (1+h_N)] \qquad \text{voir preuve } (\star\star) \\
&+ v_{k+1}((1+b_N)s_k)\,\mathbb{Q}[T_{k+1}^{(N)} = (1+b_N)] \\
&= v_{k+1}((1+h_N)s_k)q_N + v_{k+1}((1+b_N)s_k)(1-q_N)
\end{aligned}$$

D'où au final, en multipliant la réécriture déterminée ci-dessus par la constante donnée dans l'énoncé, on trouve :

$$\boxed{v_k(S_{t_k}^{(N)}) := \frac{1}{1+r_N}\left(v_{k+1}((1+h_N) \times S_{t_k}^{(N)})q_N + v_{k+1}((1+b_N) \times S_{t_k}^{(N)})(1-q_N)\right)} \quad (1)$$

Pour calculer $\text{Prix}^{(N)} = v_0(S_{t_0}^{(N)})$, il faut partir de $v_N(S_{t_N}^{(N)})$, puis remonter par récursivité vers l'instant t_0.

Remarque 2 Attention, pour chaque étape il faut gérer tous les cas représentés par les arêtes de l'arbre. Pour cela, on va "numériser" v_N avec un vecteur $\mathbf{v_N}(S_{t_N}^{(N)})$

$$\mathbf{v_N} = \begin{pmatrix} f(s(1+h_N)^N) \\ \vdots \\ f(s(1+h_N)^i(1+b_N)^{N-i} \\ \vdots \\ f(s(1+b_N)^N) \end{pmatrix} \in \mathbb{R}^N \text{ et } \forall k \in [\![0, N-1]\!],\ \mathbf{v_k} = \begin{pmatrix} v_k(s(1+h_N)^k) \\ \vdots \\ v_k(s(1+h_N)^i(1+b_N)^{k-i} \\ \vdots \\ v_k(s(1+b_N)^k) \end{pmatrix} \in \mathbb{R}^k$$

Pour le calcul de chaque vecteur, on peut utiliser une matrice nulle pour itérer sur les coefficients de la colonne précédente, pour le calcul d'un coefficient de la colonne suivante. Ainsi, par récurrence, il existe 2 types de configuration possible pour déterminer les composantes $\mathbf{v_k}^{(i)}$ de $\mathbf{v_k}$

$$\mathbf{v_k}^{(i)} = \frac{1}{1+r_N}(q_N \mathbf{v_{k+1}}^{(i+1)} + (1-q_N)\mathbf{v_{k+1}}^{(i)})$$

Remarque 3 Il existe une configuration équivalente :

$$\mathbf{v_k}^{(i)} = \frac{1}{1+r_N}(q_N \mathbf{v_{k+1}}^{(i)} + (1-q_N)\mathbf{v_{k+1}}^{(i+1)})$$

Par la suite, on utilise la première configuration :

Algorithme 2 : Deuxième Pricer

Input : N, r_N, h_N,b_N, s, f
$q_N \leftarrow \frac{r_N - b_N}{h_N - b_N}$;
$V_{size} \leftarrow N+1$;
for *element* $i \in [\![0, N]\!]$ **do**
  $V[i][0] \leftarrow f(s(1+h_N)^i(1+b_N)^{N-i})$;
end
for *element* $j \in [\![1, N]\!]$ **do**
  for *element* $i \in [\![j, N]\!]$ **do**
    $V[i][j] \leftarrow \frac{1}{1+r_N}(q_N \times V[i-1][j-1] + (1-q_N)V[i][j-1])$
  end
end
Output : $V[N][N]$

Remarque 4 (Inconvénient majeur de cette méthode) Dans les 2 cas, cette méthode présente un important inconvénient au niveau numérique. En effet, nous sommes obligé de stocker $\frac{N^2}{2}$ coefficients nuls. Une solution (non implémentée ici) serait d'implémenter l'algorithme sur des matrices creuses, via le module scipy.sparse

Une autre méthode pourrait être d'implémenter cette formule de manière récursive. Nous l'avons implémenté dans le Notebook en Python (cf. Annexes)

Démonstration de (⋆). (Transformation efficace de la forme de $\mathbb{Q}$)
Pour plus de simplicité, on va poser A et B 2 variables aléatoires réelles avec B non nul :

$$\mathbb{Q}[A \times B = a \times b | B = b] = \frac{\mathbb{Q}[A \times B = a \times b \cap B = b]}{\mathbb{Q}[B = b]}$$

or $(AB = ab) \cap (B = b) = (A = a) \cap (B = b) \cap (AB = ab) = (A = a) \cap (B = b)$
d'où :

$$\frac{\mathbb{Q}[A \times B = a \times b \cap B = b]}{\mathbb{Q}[B = b]} = \frac{\mathbb{Q}[A = a \cap B = b]}{\mathbb{Q}[B = b]} = \mathbb{Q}[A = a | B = b]$$

Le cas où $b = 0$ est impossible car dans notre situation, $B = S_{t_k}^{(N)} = s(1+h_N)^X(1+b_N)^{k-X}$ et $s > 0$ par définition du prix de l'actif risqué au temps initial. Les 2 autres quantités sont également strictement positives $\forall X \in [\![0, k]\!]$. D'où le résultat. ■

Démonstration de (⋆⋆). (Indépendance de $T_{k+1}^{(N)}$ et $S_{t_k}^{(N)}$)
Nous cherchons à montrer que $\forall k \in \mathbb{N}^*,\ T_{k+1}^{(N)} \perp S_{t_k}^{(N)}$.
En effet, on a par définition que :

$$\forall k \in [\![1, N]\!],\ S_{t_k}^{(N)} = S_{t_0}^{(N)} \prod_{i=1}^{k} T_i^{(N)}$$

Or, $\forall i \in \mathbb{N},\ T_{i+1}^{(N)} \perp T_i^{(N)}$ car $(T_i^{(N)})_{i \in \mathbb{N}}$ est une suite de variables aléatoires indépendantes et identiquements distribuées (i.i.d)
D'après le Lemme des Coalitions, on en déduit que $\forall k \in \mathbb{N}^*,\ T_{k+1}^{(N)} \perp S_{t_k}^{(N)}$. ■

Question 6 On implémente l'algorithme en Python

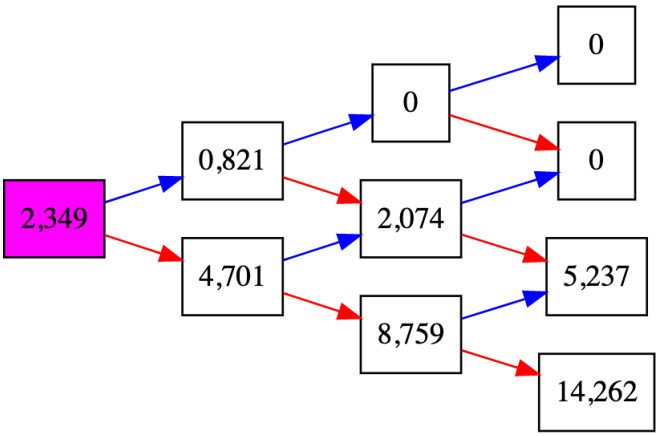

FIGURE 1 – Arbre des différents cas

On trouve un prix de $2,349$.

1.3 Etude comparative des deux pricers

Question 7 On compare les 2 pricers pour $N = 3$:

— Premier pricer : 5.2909780733979686

— Second pricer : 5.290978073397971

La distance entre les deux résultats de pricing est de $2.6645352591003757 \times 10^{-15}$. On remarque que cette distance est très très faible. Cela conclut donc à la bonne précision des techniques de pricing.

On peut également comparer les temps d'exécution des 2 pricers sur plusieurs valeurs de N :

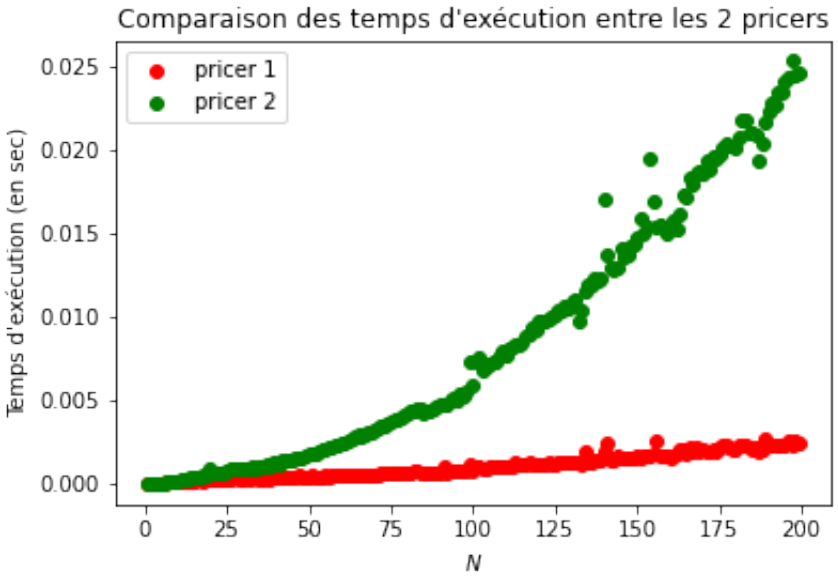

FIGURE 2 – Comparaison temporelle des 2 pricers

1.4 Etude de la couverture

Question 8 On note $(\mathcal{S})$ l'équation :

$$\alpha_{N-1}(S^{(N)}_{t_{N-1}})S^{(N)}_{t_N} + \beta_{N-1}(S^{(N)}_{t_{N-1}})S^0_{t_N} = f(S^{(N)}_{t_N})$$

Le problème s'apparente à la résolution d'un système de 2 équations à 2 inconnues. Nous allons procéder par substitution

$$
\begin{aligned}
(\mathcal{S}) &\iff \begin{cases} \alpha_{N-1}(S^{(N)}_{t_{N-1}})(1+h_N)S^{(N)}_{t_{N-1}} + \beta_{N-1}(S^{(N)}_{t_{N-1}})(1+r_N)^N = f((1+h_N)S^{(N)}_{t_{N-1}}) \\ \alpha_{N-1}(S^{(N)}_{t_{N-1}})(1+b_N)S^{(N)}_{t_{N-1}} + \beta_{N-1}(S^{(N)}_{t_{N-1}})(1+r_N)^N = f((1+b_N)S^{(N)}_{t_{N-1}}) \end{cases} \\
&\iff \begin{cases} \alpha_{N-1}(S^{(N)}_{t_{N-1}})(1+h_N)S^{(N)}_{t_{N-1}} + \beta_{N-1}(S^{(N)}_{t_{N-1}})(1+r_N)^N = f((1+h_N)S^{(N)}_{t_{N-1}}) \\ \alpha_{N-1}(S^{(N)}_{t_{N-1}})S^{(N)}_{t_{N-1}}[(1+b_N)-(1+h_N)] = f((1+b_N)S^{(N)}_{t_{N-1}}) - f((1+h_N)S^{(N)}_{t_{N-1}}) \end{cases} \\
&\iff \begin{cases} \alpha_{N-1}(S^{(N)}_{t_{N-1}})(1+h_N)S^{(N)}_{t_{N-1}} + \beta_{N-1}(S^{(N)}_{t_{N-1}})(1+r_N)^N = f((1+h_N)S^{(N)}_{t_{N-1}}) \\ \alpha_{N-1}(S^{(N)}_{t_{N-1}}) = \frac{1}{(b_N-h_N)S^{(N)}_{t_{N-1}}}\left[f((1+b_N)S^{(N)}_{t_{N-1}}) - f((1+h_N)S^{(N)}_{t_{N-1}})\right] \end{cases} \\
&\iff \begin{cases} \frac{1}{(b_N-h_N)S^{(N)}_{t_{N-1}}}\left[f((1+b_N)S^{(N)}_{t_{N-1}}) - f((1+h_N)S^{(N)}_{t_{N-1}})\right](1+h_N)S^{(N)}_{t_{N-1}} + \beta_{N-1}(S^{(N)}_{t_{N-1}})(1+r_N)^N \\ = f((1+h_N)S^{(N)}_{t_{N-1}}) \\ \alpha_{N-1}(S^{(N)}_{t_{N-1}}) = \frac{1}{(b_N-h_N)S^{(N)}_{t_{N-1}}}\left[f((1+b_N)S^{(N)}_{t_{N-1}}) - f((1+h_N)S^{(N)}_{t_{N-1}})\right] \end{cases} \\
&\iff \begin{cases} \beta_{N-1}(S^{(N)}_{t_{N-1}}) = \frac{1}{(1+r_N)^N}\left[f((1+h_N)S^{(N)}_{t_{N-1}}) - \frac{(1+h_N)}{(b_N-h_N)}\left(f((1+b_N)S^{(N)}_{t_{N-1}}) - f((1+h_N)S^{(N)}_{t_{N-1}})\right)\right] \\ \alpha_{N-1}(S^{(N)}_{t_{N-1}}) = \frac{1}{(b_N-h_N)S^{(N)}_{t_{N-1}}}\left[f((1+b_N)S^{(N)}_{t_{N-1}}) - f((1+h_N)S^{(N)}_{t_{N-1}})\right] \end{cases}
\end{aligned}
$$

A la fin, nous trouvons :

$$\boxed{\alpha_{N-1}(S_{t_{N-1}}^{(N)}) = \frac{1}{(b_N - h_N)S_{t_{N-1}}^{(N)}}\left[f((1+b_N)S_{t_{N-1}}^{(N)}) - f((1+h_N)S_{t_{N-1}}^{(N)})\right]}$$

$$\boxed{\beta_{N-1}(S_{t_{N-1}}^{(N)}) = \frac{1}{(b_n - h_N)(1+r_N)^N}\left[f((1+h_N)S_{t_{N-1}}^{(N)})(b_N+1) - f((1+b_N)S_{t_{N-1}}^{(N)})(1+h_N)\right]}$$

Question 9 En appliquant le même principe qu'à la question précédente avec :
- v_k à la place de f
- $S_{t_k}^0 = (1+r_N)^k$

On obtient exactement les formules ci-dessous :

$$\forall k \in [\![1, N-1]\!],\ \boxed{\alpha_{k-1}(S_{t_{k-1}}^{(N)}) = \frac{1}{(b_N - h_N)S_{t_{k-1}}^{(N)}}\left[v_k((1+b_N)S_{t_{k-1}}^{(N)}) - v_k((1+h_N)S_{t_{k-1}}^{(N)})\right]}$$

$$\forall k \in [\![1, N-1]\!],\ \boxed{\beta_{k-1}(S_{t_{k-1}}^{(N)}) = \frac{1}{(b_n - h_N)(1+r_N)^k}\left[v_k((1+h_N)S_{t_{k-1}}^{(N)})(b_N+1) - v_k((1+b_N)S_{t_{k-1}}^{(N)})(1+h_N)\right]}$$

Question 10 Les α_{k-1} et β_{k-1} dépendent de v_k ; or, on a généralisé le calcul de v_k via un protocole « vectoriel » :
- **Calcul de α_0 et β_0** : on applique la formule de la question 9 avec v_1 qui nécessite un calcul préliminaire pour les valeurs $s(1+h_N)$ et $s(1+b_N)$
- **Calcul de α_1 et β_1** : on applique la formule de la question 8 pour les 2 cas à traiter

• **Calcul de α_0 et β_0** On va utiliser les calculs démontrés dans la question 9 avec $k=1$.

$$\alpha_0(S_{t_0}^{(N)}) = \frac{1}{(b_N - h_N)S_{t_0}^{(N)}}\left[v_1((1+b_N)S_{t_0}^{(N)}) - v_1((1+h_N)S_{t_0}^{(N)})\right]$$

$$\beta_0(S_{t_0}^{(N)}) = \frac{1}{(b_N - h_N)(1+r_N)}\left[v_1((1+h_N)S_{t_0}^{(N)})(b_N+1) - v_1((1+b_N)S_{t_0}^{(N)})(1+h_N)\right]$$

Il faut donc calculer $v_1((1+b_N)s)$ et $v_1((1+h_N)s)$ avec $s = S_{t_0}^{(N)}$ afin de pouvoir déterminer numériquement α_0 et β_0.
Pour cela, on va utiliser la formule théorique déterminée lors de l'élaboration du second pricer. En effet :

$$v_1(s(1+h_N)) = \frac{1}{1+r_N}\Big(f(s(1+h_N)^2)q_N + f(s(1+h_N)(1+b_N))(1-q_N)\Big)$$

et

$$v_1(s(1+b_N)) = \frac{1}{1+r_N}\Big(f(s(1+b_N)(1+h_N))q_N + f(s(1+b_N)^2)(1-q_N)\Big)$$

D'où, en injectant $v_1(s(1+h_N))$ et $v_1(s(1+b_N))$ dans les formules de α_0 et β_0, et en effectuant le calcul de manière numérique, nous trouvons :

$$\boxed{\alpha_0(S_{t_0}^{(N)}) \simeq 0.796} \qquad \boxed{\beta_0(S_{t_0}^{(N)}) \simeq -73.43}$$

• **Calcul de α_1 et β_1** On cherche maintenant à déterminer α_1 et β_1. On utilise la formule déterminée dans la question 8 avec $N = 2$.

$$\alpha_1(S_{t_1}^{(N)}) = \frac{1}{(b_N - h_N)S_{t_1}^{(N)}}\left[f((1+b_N)S_{t_1}^{(N)}) - f((1+h_N)S_{t_1}^{(N)})\right]$$

$$\beta_1(S_{t_1}^{(N)}) = \frac{1}{(b_N - h_N)(1+r_N)^2}\left[f((1+h_N)S_{t_1}^{(N)})(b_N+1) - f((1+h_N)S_{t_1}^{(N)})(1+h_N)\right]$$

Ces calculs requièrent les valeurs numériques de $f((1+b_N)S_{t_1}^{(N)})$ et $f((1+h_N)S_{t_1}^{(N)})$ avec $S_{t_1}^{(N)} \in \{s(1+h_N); s(1+b_N)\}$ où $s = S_{t_0}^{(N)}$

— Cas où $S_{t_1}^{(N)} = s(1+h_N)$

$$\boxed{\alpha_1(S_{t_1}^{(N)}) \simeq 0.976} \qquad \boxed{\beta_1(S_{t_1}^{(N)}) \simeq -91.785}$$

— Cas où $S_{t_1}^{(N)} = s(1+b_N)$

$$\boxed{\alpha_1(S_{t_1}^{(N)}) = 0} \qquad \boxed{\beta_1(S_{t_1}^{(N)}) = 0}$$

2 Compléments de réponse

Nous proposons ici de détailler d'autres pistes de recherche pour les questions 2 et 5.

Complément pour la question 2 (Détails de la récurrence + Explication avec dénombrement)
Cherchons le prix $\text{Prix}^{(N)}$ défini par : $\text{Prix}^{(N)} := \frac{1}{(1+r_n)^N}\,\mathbb{E}_{\mathbb{Q}}[f(S_{t_N}^{(N)})]$
Soit $\mathcal{P}_k$, la propriété de récurrence suivante avec $k > 0$:

$$\mathcal{P}_k = \ll\ S_{t_k}^{(N)} = S_{t_0}^{(N)} \prod_{i=0}^{k-1} T_{k-i}^{(N)}\ \gg$$

Initialisation : Pour $k = 1$:
$S_{t_1}^{(N)} = T_1^{(N)} S_{t_0}^{(N)}$ (d'après l'énoncé)
$S_{t_0}^{(N)} \prod_{i=0}^{0} T_{1-i}^{(N)} = S_{t_0}^{(N)} T_1^{(N)} = S_{t_1}^{(N)}$

Hérédité : Supposons que le résultat est vrai au rang k, montrons le résultat au rang $k+1$

$$\begin{aligned} S_{t_k}^{(N)} &= T_{k+1}^{(N)} S_{t_k}^{(N)} \\ &= T_{k+1}^{(N)} S_{t_0}^{(N)} \prod_{i=0}^{k-1} T_{k-i}^{(N)} (HR) \\ &= S_{t_0}^{(N)} \prod_{i=0}^{k} T_{k+1-i}^{(N)} \end{aligned}$$

On obtient $\mathcal{P}_{k+1}$
En particulier, pour $k = N$:

$$S_{t_N}^{(N)} = S_{t_0}^{(N)} \prod_{i=0}^{N-1} T_{N-i}^{(N)} = S_{t_0}^{(N)} \prod_{i=1}^{N} T_i^{(N)}$$

Soit $i \in [\![1, N]\!]$

Détermination de $\mathbb{Q}(S_{t_N}^{(N)} = s(1+h_N)^i(1+b_N)^{N-i})$ par dénombrement Procédons au dénombrement :

Chaque liste de $S_{t_N}^{(N)} = s(1+h_N)^i(1+b_N)^{N-i}$ a une probabilité
- q_N^i pour obtenir $(1+h_N)^i$
- $(1-q_N)^{N-i}$ d'obtenir $(1+b_N)^{N-i}$

Or il y a $\binom{N}{i}$ combinaisons possibles d'obtenir cette liste.

On obtient ainsi :

$$\boxed{\mathbb{Q}(S_{t_N}^{(N)} = s(1+h_N)^i(1+b_N)^{N-i}) = \binom{N}{i} q_N^i (1-q_N)^{N-i}}$$

En appliquant le théorème de transfert à la fonction f, on trouve :

$$\mathbb{E}_{\mathbb{Q}}[f(S_{t_N}^{(N)})] = \sum_{i=1}^{N} \binom{N}{i} q_N^i (1-q_N)^{N-i} f(s(1+h_N)^i(1+b_N)^{N-i}))$$

Finalement, on trouve :

$$\forall N \in \mathbb{N}^*,\ \boxed{\text{Prix}^{(N)} = \frac{1}{(1+r_N)^N} \sum_{i=1}^{N} \binom{N}{i} q_N^i (1-q_N)^{N-i} f(s(1+h_N)^i(1+b_N)^{N-i}))}$$

Complément pour la question 5 (Implémentation récursive pour le second pricer)

```
Fonction Deuxieme Pricer(N,r,h,b,s,f)
    q ⟵ (r-b)/(h-b)
    p1 ⟵ q/(1+r)
    p2 ⟵ (1-q)/(1+r)

    Fonction Pricer_rec(n,N,p1,p2,h,b,s,f)
        Si (n==N) Alors
            y ⟵ f(s)
        Sinon
            y ⟵ p1*price_rec(n+1,N,p1,p2,h,b,s*(1+h),f)
            +p2*price_rec(n+1,N,p1,p2,h,b,s*(1+b),f)

        Retourner y
        Fin Si
    Fin Fonction

    Retourner price_rec(0,N,p1,p2,h,b,s,f)
Fin Fonction
```

3 Modèle de Black-Scholes

3.1 Présentation du modèle

Question 11 Nous allons utiliser la formule d'Itô

$$\mathrm{d}g(S_t) = g'(S_t)\mathrm{d}S_t + \frac{|\sigma S_t|^2}{2} g''(S_t)\mathrm{d}t \tag{ITO}$$

Avec : $g : \mathbb{R} \longrightarrow \mathbb{R}$ de régularité $\mathcal{C}^2$
On peut appliquer la formule d'Itô à $g = \ln$ car $x \longmapsto \ln x$ est $\mathcal{C}^2$ sur $\mathbb{R}_+^*$

Démonstration. En effet, soit

$$\begin{array}{rccc} g & : & \mathbb{R}_+^* & \longrightarrow & \mathbb{R} \\ & & x & \longmapsto & \ln x \end{array}$$

On a alors : $g'(x) = \frac{1}{x}$ qui est dérivable et continue. De plus, $g''(x) = -\frac{1}{x^2}$ qui est bien continue sur $\mathbb{R}_+^*$ ■

On a :

$$\mathrm{d}g(S_t) = g'(S_t)\mathrm{d}S_t + \frac{|\sigma S_t|^2}{2} g''(S_t)\mathrm{d}t$$

et

$$\mathrm{d}S_t = S_t \times (r\mathrm{d}t + \sigma \mathrm{d}B_t)$$

$$\begin{aligned} \mathrm{d}\ln S_t &= \frac{\mathrm{d}S_t}{S_t} + \frac{|\sigma|^2}{2}\Big(- \frac{|S_t|^2}{S_t^2}\Big)\mathrm{d}t \\ &= \frac{\mathrm{d}S_t}{S_t} - \frac{|\sigma|^2}{2}\mathrm{d}t \\ &= \frac{S_t * (r\mathrm{d}t + \sigma \mathrm{d}B_t)}{S_t} - \frac{|\sigma|^2}{2}\mathrm{d}t \\ &= (r - \frac{|\sigma|^2}{2})\mathrm{d}t + \sigma \mathrm{d}B_t \end{aligned}$$

On intègre la formule par rapport au temps sur l'intervalle $[0,t]$ avec $t \in \mathbb{R}_+^*$

$$\int_0^t \mathrm{d}\ln S_u = \int_0^t (r - \frac{|\sigma|^2}{2})\mathrm{d}u + \int_0^t \sigma \mathrm{d}B_u \iff \ln S_t - \ln S_0 = (r - \frac{|\sigma|^2}{2})t + \sigma(B_t - B_0) \quad \text{or par définition } B_0 = 0$$

$$\iff \ln \frac{S_t}{S_0} = (r - \frac{|\sigma|^2}{2})t + \sigma B_t$$

$$\iff \boxed{S_t = s \exp\left(rt - \frac{t|\sigma|^2}{2}\right) \exp \sigma B_t}$$

3.2 Le pricer par la méthode de Monte-Carlo

Question 12 Nous avons implémenté la fonction dans le Notebook ci-joint. Le pseudo-code du pricer est disponible ci-dessous :

```
Fonction Troisieme Pricer(n,s,r,σ,T,f)
    ksi ⟵ gener_echantillon_gaussien(0,1,n)

    prix_mc ⟵ 0
    Pour i allant de 0 A n−1
        prix_mc ⟵ prix_mc +exp(−r∗T)∗
        f(s∗exp(r − σ²/2 ∗T + σ∗√T ∗ksi[i]))
    Fin_Pour
    prix_mc ⟵ prix_mc ∗(1/n)
    Retourner prix_mc
Fin Fonction
```

Question 13 Nous traçons le graphique pour le prix avec $r = 0.01$, $\sigma = 0.1$, $s = 100$, $T = 1$ et $f(x) = \max(100 - x, 0)$ pour $n = 10^5 k$, $\forall k \in [\![1, 10]\!]$:

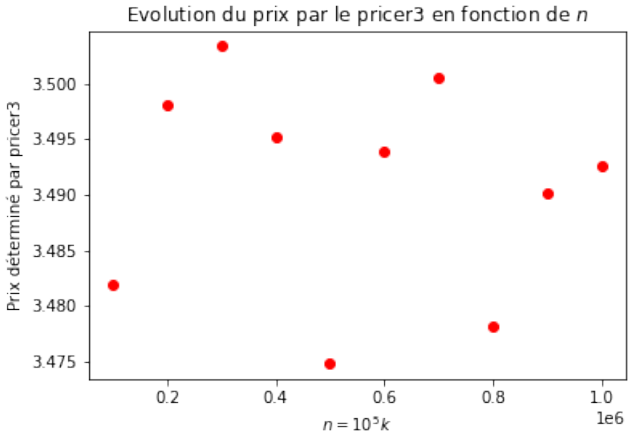

FIGURE 3 – Evolution du prix déterminé par le troisième pricer en fonction de n

Question 14 On note

$$\mathbb{P} := \mathbb{E}[e^{-rT} f(S_T)]$$

et

$$\widehat{\mathrm{PrixMC}}_{(n)} := \frac{1}{n} \sum_{i=1}^{n} e^{-rT} f\Big(s \exp((r - \frac{\sigma^2}{2})T + \sigma\sqrt{T}\xi_i)\Big)$$

On pose $\forall i \in [\![1,n]\!]$

$$X_i = e^{-rT} f\Big(s\exp((r-\frac{\sigma^2}{2})T + \sigma\sqrt{T}\xi_i)\Big)$$

On va montrer que les X_i sont identiquement et indépendamment distribués (iid).
On voit directement que $\forall i \in [\![1,n]\!],\ X_i = \phi(\xi_i)$ où

$$\begin{array}{rccc} \phi & : & \mathbb{R} & \longrightarrow & \mathbb{R} \\ & & x & \longmapsto & e^{-rT} f\Big(s\exp((r-\frac{\sigma^2}{2})T + \sigma\sqrt{T}x)\Big) \end{array}$$

De plus, la fonction ϕ est strictement monotone croissante.
Soient $(i,j) \in [\![1,n]\!]^2,\ i \neq j$
Déjà, on a $(\star)$:

$$\mathbb{Q}[\xi_i \leq x_i \cap \xi_j \leq x_j] = Q[\xi_i \leq x_i] \times \mathbb{Q}[\xi_j \leq x_j]$$

car $(\xi_i)_{i\in[\![1,n]\!]}$ iid par énoncé De plus,

$$\begin{aligned} \mathbb{Q}[\xi_i \leq x_i \cap \xi_j \leq x_j] &= \mathbb{Q}[X_i \leq \phi(x_i) \cap X_j \leq \phi(x_j)] \\ &= \mathbb{Q}[X_i \leq \phi(x_i)] \times \mathbb{Q}[X_j \leq \phi(x_j)] \text{ en composant } (\star) \text{ par } \phi \end{aligned}$$

Les X_i sont donc iid. On peut donc appliquer la Loi Forte des Grands Nombres :

$$\widehat{\text{PrixMC}}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{\text{ps}} \mathbb{E}[X_1]$$

où

$$\mathbb{E}[X_1] = \mathbb{E}[e^{-rT} f\Big(s\exp((r-\frac{\sigma^2}{2})T + \sigma\sqrt{T}\xi_1)\Big)]$$

On reconnaît intuitivement une partie de l'expression de $f(S_T)$ déterminée dans la question 11.

$$f(S_T) = f\Big(S_0 \exp\left(rT - \frac{T|\sigma|^2}{2}\right)\exp\sigma B_T\Big)$$

Par définition, on a $B_T \rightsquigarrow \mathcal{N}(0,T)$.
De plus, $\xi_1 \rightsquigarrow \mathcal{N}(0,1) \longrightarrow \xi_1\sqrt{T} \rightsquigarrow \mathcal{N}(0,T)$
On en conclut donc que les 2 expressions suivent la même loi.
Finalement,

$$\boxed{\left(\widehat{\text{PrixMC}}_{(n)}\right)_{n\in\mathbb{N}} \xrightarrow{\text{ps}} \mathbb{P} := \mathbb{E}[e^{-rT} f(S_T)]}$$

3.3 Le pricer par formule fermée

Question 15 Nous avons implémenté la fonction `put` en PYTHON. Voici le pseudo-code de la fonction

```
Fonction Put(r,s,σ,T,f)
    d ⟵ (1/(σ*√T)) * (log(s/K) + (r + (σ²/2) * T))
    Fonction F(x)
        retourner la probabilit  d'avoir x
    Fin Fonction
    prix_bs ⟵ -s*F(-d) + K*exp(-r*T)*F(-d + σ * √T)
    Retourner prix_bs

Fin Fonction
```

Question 16 Nous effectuons une application numérique pour le `put` avec les paramètres suivants : $r = 0.01$, $\sigma = 0.1$, $s = 100$, $T = 1$, $K = 90$:
Nous trouvons : $\boxed{0.5815000751362422}$

Question 17 Nous souhaitons afficher un graphique comparant les valeurs du `pricer3` et du `put` pour plusieurs valeurs de n

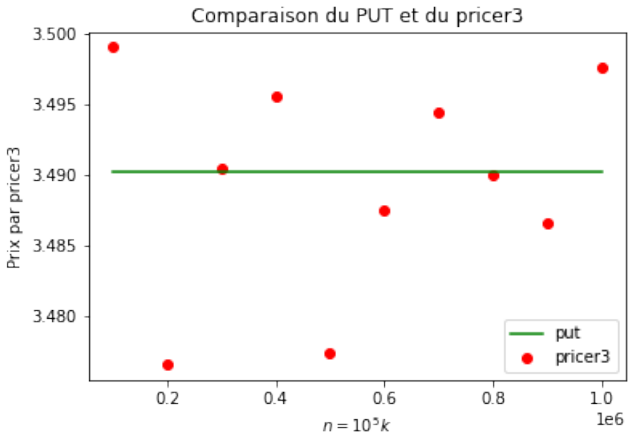

FIGURE 4 – Comparaison du put et du troisième pricer pour plusieurs valeurs de $n = 10^5 k,\ \forall k \in [\![1, 10]\!]$

(Complément) : Convergence asymptotique La simulation du pricer3 pour d'importantes valeurs de n permet de confirmer le caractère convergent du pricer de Monte-Carlo vers le put

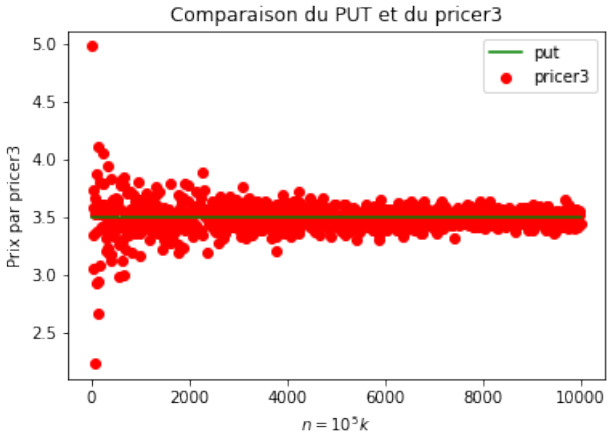

FIGURE 5 – Evolution asymptotique du pricer 3

On remarque alors une convergence très nette de la fonction du `pricer3` vers le put de l'option.

Question 18 On cherche à créer un graphique en 3 dimensions en affichant le prix de l'option grâce à la fonction `put` :

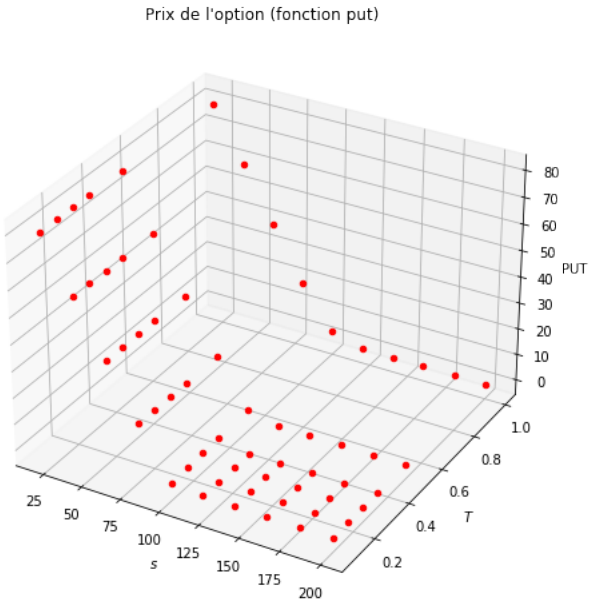

FIGURE 6 – Prix de l'option en fonction de s et de T

Nous remarquons que le put de l'option est très sensible aux évolutions des valeurs de s. En effet, plus s décroit, plus la valeur du put augmente fortement.
Il semble néamoins que les variations de T soient sans effet au pire, ou entraînent une très faible évolution du put.

En conclusion, le prix calculé par le pricer par formule fermée est très nettement influencé par la valeur prise par s.

4 Etude de la convergence des prix

Question 19 Nous souhaitons étudier la convergence du modèle discret (CRR) vers le modèle continu (BS). En employant les quantités décrites dans l'énoncé puis en faisant tendre $N \longrightarrow +\infty$, on obtient les résultats suivants :

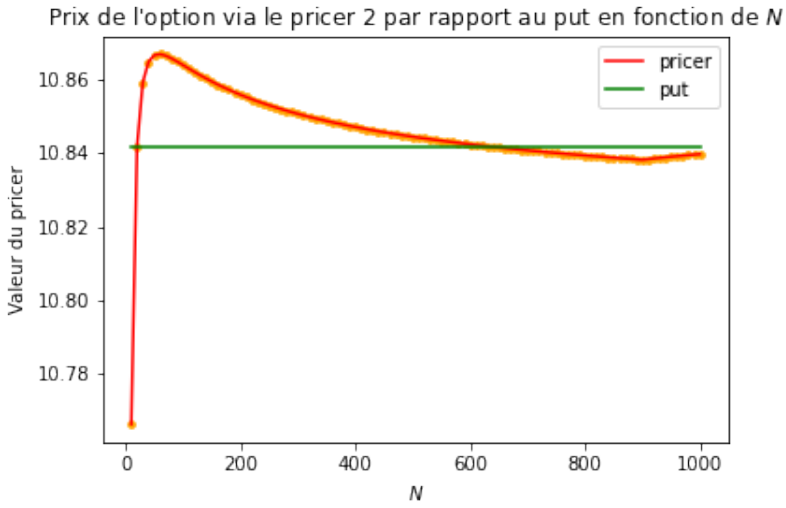

FIGURE 7 – Etude de la convergence de CRR vers BS

On observe que le pricer effectue un premier dépassement du put quand N est encore assez faible. Puis ce dernier tend à converger vers le put au fur et à mesure que N augmente.
On peut ainsi conclure que le modèle de Cox-Ross-Rubinstein converge vers le modèle de Black-Scholes lorsque le pas de discrétisation N tend vers d'importantes quantités.
Voici le pseudo-code de la question 19 :

```
Fonction F(x)
    Retourner max(100 - x, 0)
Fin Fonction

s ⟵ 100
σ ⟵ 0.3
r ⟵ 0.02
T ⟵ 1
Pour i allant de 1 A 100
    Nliste[k] ⟵ 10*k
Fin_Pour
K ⟵ 100
p2 ⟵ liste vide
Pour N dans Nliste
    r_N ⟵ r * T/N
    h_N ⟵ (1 + r_N)* exp(σ* sqrt(T/N)) - 1
    b_N ⟵ (1 + r_N)*exp(−σ*sqrt(T/N)) - 1
    p ⟵ price2(N, r_N, h_N, b_N, s, f)
    p2 ⟵ p2 ∪ { p }
Fin_Pour
```

5 EDP Black-Scholes

On considère l'Equation aux Dérivées Partielles de Black Scholes suivante :

On note $\Delta t = \frac{T}{M}$, $h = \frac{x_{\max} - x_{\min}}{N}$ et enfin $\forall m \in [\![0, M-1]\!]$ $\mathbf{P}_m = \begin{pmatrix} p_{m,1} \\ p_{m,2} \\ \vdots \\ p_{m,N-1} \end{pmatrix}$

$\forall (t,x) \in (0,T] \times (x_{\min}, x_{\max})$,

$$\frac{\partial p}{\partial t}(t,x) - \frac{1}{2}\sigma^2 \frac{\partial^2 p}{\partial x^2}(t,x) - \left(r - \frac{\sigma^2}{2}\right)\frac{\partial p}{\partial x}(t,x) + rp(t,x) = 0 \qquad \text{(BS-EDP)}$$

On réécrit cette formule afin de faire apparaitre des coefficients plus simples à manipuler :

$$\frac{\partial p}{\partial t}(t,x) + \alpha\sigma^2 \frac{\partial^2 p}{\partial x^2}(t,x) + \beta \frac{\partial p}{\partial x}(t,x) + \gamma p(t,x) = 0$$

avec :

$$\begin{cases} \alpha &= -\frac{\sigma^2}{2} \\ \beta &= \frac{\sigma^2}{2} - r \\ \gamma &= r \end{cases}$$

On note par la suite $\mathbf{p}_m^j$, l'approximation de $p(t_m, x_j$, $\forall (m,j) \in [\![0, M-1]\!] \times [\![1, N-1]\!]$

5.1 Différences finies par méthode explicite

Discrétisation des composantes différentielles de p Pour la résolution de l'EDP, il faut tout d'abord réaliser une discrétisation des composantes différentielles de p grâce aux formules des différences finies suivantes :

$$\begin{aligned} \frac{\partial p}{\partial t}(t_m, x_j) &\simeq \frac{1}{\Delta t}(p(t_{m+1}, x_j) - p(t_m, x_j)) \\ \frac{\partial p}{\partial x}(t_m, x_j) &\simeq \frac{1}{2h}(p(t_m, x_{j+1}) - p(t_m, x_{j-1})) \\ \frac{\partial^2 p}{\partial x^2}(t_m, x_j) &\simeq \frac{1}{h^2}(p(t_m, x_{j+1}) - 2p(t_m, x_j) + p(t_m, x_{j-1})) \end{aligned}$$

Détermination du système à résoudre On injecte ces termes dans (BS-EDP) :

$$\begin{aligned} \text{(BS-EDP)} &\iff \frac{1}{\Delta t}(\mathbf{p}_{m+1}^j - \mathbf{p}_m^j) + \frac{\alpha}{h^2}(\mathbf{p}_m^{j+1} - 2\mathbf{p}_m^j + \mathbf{p}_m^{j-1}) + \frac{\beta}{2h}(\mathbf{p}_m^{j+1} - \mathbf{p}_m^{j-1}) + \gamma \mathbf{p}_m^j = 0 \\ &\iff \frac{1}{\Delta t}\mathbf{p}_{m+1}^j = \mathbf{p}_m^j(\frac{1}{\Delta t} + \frac{2\alpha}{h^2} - \gamma) + \mathbf{p}_m^{j+1}(-\frac{\alpha}{h^2} - \frac{\beta}{2h}) + \mathbf{p}_m^{j-1}(-\frac{\alpha}{h^2} + \frac{\beta}{2h}) \\ &\iff \boxed{\mathbf{p}_{m+1}^j = a_1 \mathbf{p}_m^j + a_2 \mathbf{p}_m^{j+1} + a_3 \mathbf{p}_m^{j-1}} \end{aligned}$$

avec

$$\begin{cases} a_1 &= \Delta t(\frac{1}{\Delta t} + \frac{2\alpha}{h^2} - \gamma) \\ a_2 &= \Delta t(-\frac{\alpha}{h^2} - \frac{\beta}{2h}) \\ a_3 &= \Delta t(-\frac{\alpha}{h^2} + \frac{\beta}{2h}) \end{cases}$$

Numérisation du problème & Résolution Le schéma explicite de la méthode des différences finies est très simple[3] à implémenter numériquement. Il suffit de réaliser un double parcours imbriqué (spatial et temporel) sur une matrice P et remplir les coefficients avec la formule de récurrence trouvée ci-dessus. Néanmoins, comme nous pouvons le voir ci-dessous, elle présente un très grand désavantage : son instabilité.
Commençons par tester notre algorithme avec $N = M$:

3. En effet, la valeur de p à l'instant suivant est une combinaison linaire des valeurs de p du passé à des positions différentes.

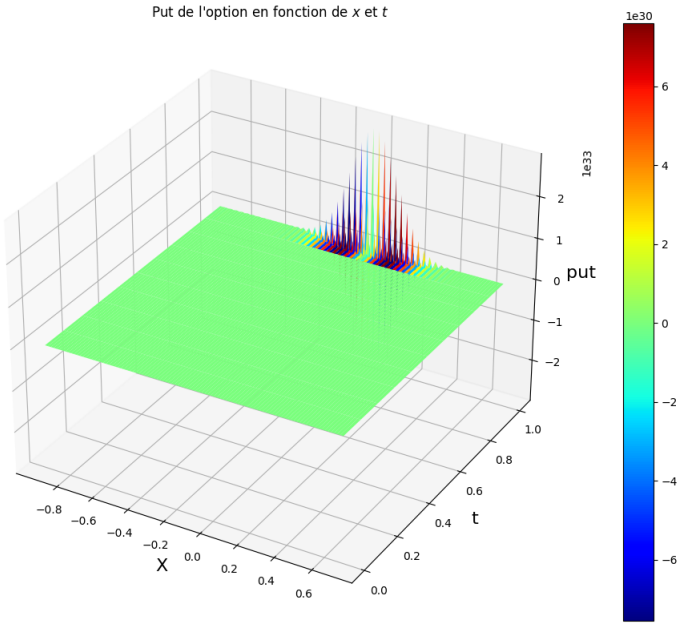

FIGURE 8 – Put de l'option en fonction des paramètres d'espace x et de temps t (à M fixé arbitrairement)

On dénote un comportement périodique à des valeurs élevées de t symétriquement par rapport à $x = 0$.
Nous pouvons mieux observer ce comportement en visualisant $x \longmapsto p(t, x)$ à t fixé :

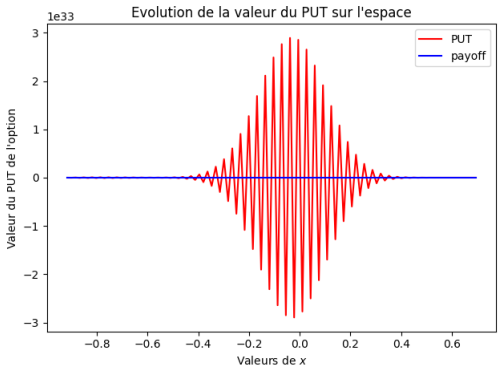

FIGURE 9 – Put de l'option en fonction du paramètre d'espace x (à M fixé arbitrairement) $(x \longmapsto p(t, x))$

On peut ici remarquer ce comportement dénotant une instabilité résidant dans notre schéma numérique.
Après plusieurs recherches bibliographiques, on s'aperçoit que la stabilité de la méthode des différences finies à schéma explicite est soumise à la contrainte de Courant-Friedrich-Lévy (CFL)

Proposition 1 (Contrainte de Courant-Friedrich-Lévy) La stabilité de la méthode des Différences Finies par implémentaiton d'un schéma explicite est entièrement garanti par la condition :

$$\boxed{C = \frac{\Delta t}{h^2} \leq C_{\max} = 1}$$

Remarque 5 A chaque fois que l'on doit effectuer une nouvelle approximation de (BS-EDP) via la méthode explicite des différences finies, il faudra **toujours** garantir que $\Delta t \leq h^2$.

Pour avoir cette garantie, nous allons fixer $\boxed{\Delta t = \dfrac{h^2}{2}}$

Remarque 6 (Paramètrage selon M) On peut également paramètre notre grille de discrétisation en fonction de M. Dans ce cas là,

$$\boxed{M = \left\lfloor \frac{2N^2T}{(x_{\max} - x_{\min})^2} \right\rfloor}$$

Démonstration. En effet,

$$\begin{aligned} \Delta t = \frac{h^2}{2} &\iff \frac{T}{M} = \frac{(x_{\max} - x_{\min})^2}{N} \\ &\iff M = \left\lfloor \frac{2N^2T}{(x_{\max} - x_{\min})^2} \right\rfloor \qquad \text{car } n \text{ doit être un entier} \end{aligned}$$

■

Au final, en respectant ce critère de stabilité, on obtient des résultats cohérents :

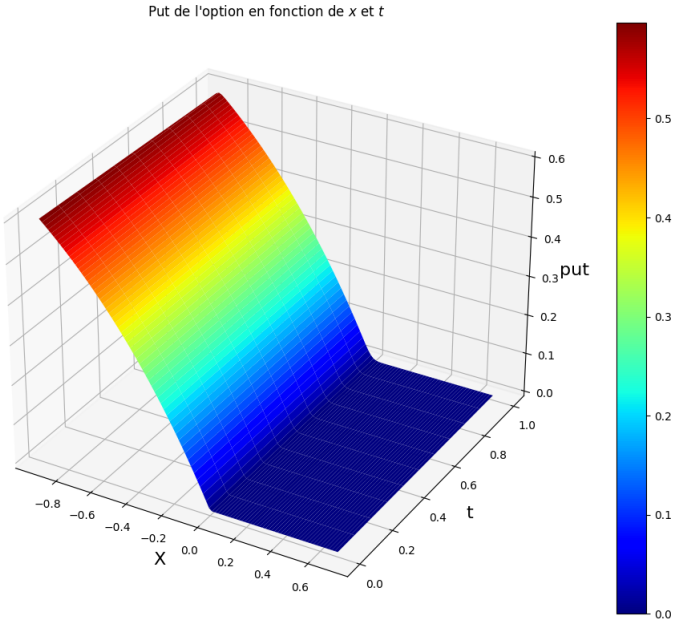

FIGURE 10 – Put de l'option en fonction des paramètres d'espace x et de temps t (sous contrainte de stabilité respectée)

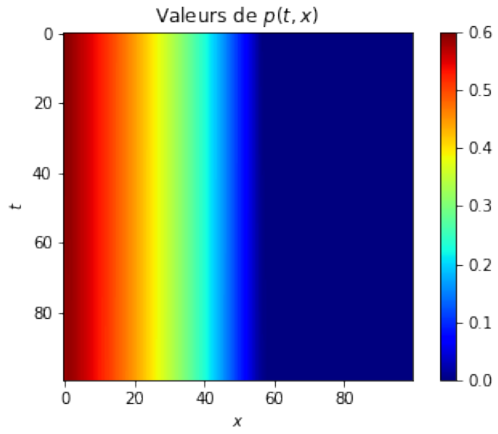

FIGURE 11 – Put de l'option en fonction des paramètres d'espace x et de temps t (sous contrainte de stabilité respectée) (vue en 2D)

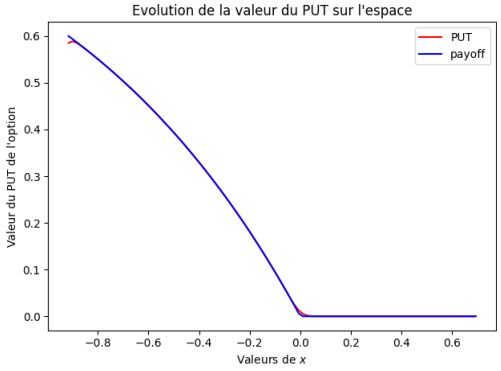

FIGURE 12 – Put de l'option en fonction du paramètre d'espace x (sous contrainte de stabilité respectée) ($x \longmapsto p(t, x)$)

On remarque que notre implémentation suit presque parfaitement la courbe du payoff. Nous analyserons en fin de question, l'évolution de l'erreur relative suivant nos implémentations des 3 types de schéma.
Voici le pseudo-code de la méthode explicite :

$h \longleftarrow \frac{x_{max} - x_{min}}{N}$
$dt \longleftarrow \frac{h^2}{2}$

```
P ⟵ la matrice nulle de taille m*n

# Discretisation de l'espace-temps X_liste et T_liste
Pour i allant de 1 A (N-1)
    P[0][i] ⟵ max(K - exp(X_liste[i]), 0)
Fin_Pour

Pour t allant de 0 A (M-1)
    P[t][0] ⟵ K*exp(-r*T_liste[t]) - exp(x_min)
    P[t][-1] ⟵ 0
Fin_Pour

P_m ⟵ vecteur nul de taille n
Pour t allant de 0 A (M-1)
    P_m[0] ⟵ K*exp(-r*T_liste[t]) - exp(x_min)
```

```
    P_m[-1] ⟵ 0

    a1 = dt*(1/dt + 2*α/h² − γ)
    a2 = dt*(−α/h² − β/(2*h))
    a3 = dt*(−α/h² + β/(2*h))
    Pour x allant de 1 A (M - 1)
        P_m[x] ⟵ a1*P[t][x] + a3*P[t][x - 1]+a2*P[t][x + 1]
    Fin_Pour
    P[t + 1] ⟵ copie de P_m
Fin_Pour
```

5.2 Différences finies par méthode implicite

Discrétisation des composantes différentielles de p

$$\begin{aligned}
\frac{\partial p}{\partial t}(t_m, x_j) &\simeq \frac{1}{\Delta t}(p(t_m, x_j) - p(t_{m-1}, x_j)) \\
\frac{\partial p}{\partial x}(t_m, x_j) &\simeq \frac{1}{2h}(p(t_m, x_{j+1}) - p(t_m, x_{j-1})) \\
\frac{\partial^2 p}{\partial x^2}(t_m, x_j) &\simeq \frac{1}{h^2}(p(t_m, x_{j+1}) - 2p(t_m, x_j) + p(t_m, x_{j-1}))
\end{aligned}$$

d'où en augmentant m de 1, on obtient les discrétisations suivantes :

$$\begin{aligned}
\frac{\partial p}{\partial t}(t_{m+1}, x_j) &\simeq \frac{1}{\Delta t}(p(t_{m+1}, x_j) - p(t_m, x_j)) \\
\frac{\partial p}{\partial x}(t_{m+1}, x_j) &\simeq \frac{1}{2h}(p(t_{m+1}, x_{j+1}) - p(t_{m+1}, x_{j-1})) \\
\frac{\partial^2 p}{\partial x^2}(t_{m+1}, x_j) &\simeq \frac{1}{h^2}(p(t_{m+1}, x_{j+1}) - 2p(t_{m+1}, x_j) + p(t_{m+1}, x_{j-1}))
\end{aligned}$$

Détermination du système à résoudre

$$\begin{aligned}
\text{(BS-EDP)} &\iff \frac{1}{\Delta t}(\mathbf{p}^j_{m+1} - \mathbf{p}^j_m) + \frac{\alpha}{h^2}(\mathbf{p}^{j+1}_{m+1} - 2\mathbf{p}^j_{m+1} + \mathbf{p}^{j-1}_{m+1}) + \frac{\beta}{2h}(\mathbf{p}^{j+1}_{m+1} - \mathbf{p}^{j-1}_{m+1}) + \gamma \mathbf{p}^j_{m+1} = 0 \\
&\iff \frac{1}{\Delta t}\mathbf{p}^j_m = \mathbf{p}^j_{m+1}(\frac{1}{\Delta t} - \frac{2\alpha}{h^2} + \gamma) + \mathbf{p}^{j+1}_{m+1}(\frac{\alpha}{h^2} + \frac{\beta}{2h}) + \mathbf{p}^{j-1}_{m+1}(\frac{\alpha}{h^2} - \frac{\beta}{2h}) \\
&\iff \boxed{\mathbf{p}^j_m = b_1 \mathbf{p}^j_{m+1} + b_2 \mathbf{p}^{j+1}_{m+1} + b_3 \mathbf{p}^{j-1}_{m+1}}
\end{aligned}$$

avec :

$$\begin{cases}
b_1 &= \Delta t(\frac{1}{\Delta t} - \frac{2\alpha}{h^2} + \gamma) \\
b_2 &= \Delta t(\frac{\alpha}{h^2} + \frac{\beta}{2h}) \\
b_3 &= \Delta t(\frac{\alpha}{h^2} - \frac{\beta}{2h})
\end{cases}$$

Numérisation du problème & Résolution Après plusieurs tentatives, nous n'avons pas réussi à implémenter la solution via une méthode itérative en remplissant successivement les différentes composantes de $\mathbf{P_m}$. Nous nous sommes alors tournés vers une solution matricielle pour déterminer la solution de systèmes.
Matriciellement, nous obtenons le système linéaire suivant :

$$\forall m \in [\![0, M-1]\!],\ \mathbf{P_m} = A\mathbf{P_{m+1}} \iff \boxed{\mathbf{P_{m+1}} = A^{-1}\mathbf{P_m}}$$

où la matrice A est une matrice tridiagonale de la forme

$$A = \begin{pmatrix} b_1 & b_2 & 0 & \cdots & \cdots & \cdots & 0 \\ b_3 & b_1 & b_2 & 0 & \cdots & \cdots & 0 \\ 0 & b_3 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \ddots & \ddots & b_2 \\ 0 & \cdots & \cdots & \cdots & 0 & b_3 & b_1 \end{pmatrix}$$

Ainsi, il suffit de calculer initialement les coefficients de la matrice tridiagonale A puis de calculer son inverse. Enfin, on effectue le produit matriciel entre A^{-1} et $\mathbf{P_m}$.

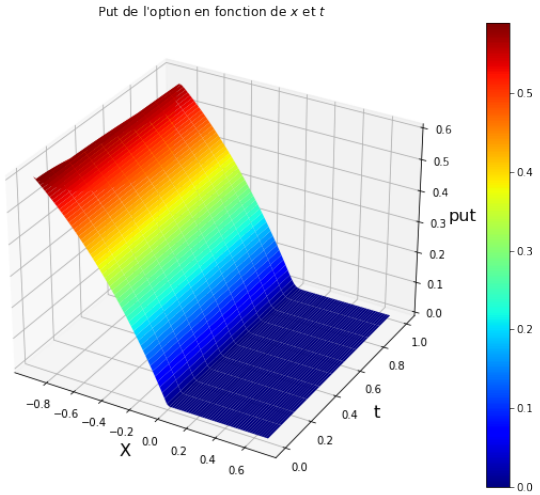

FIGURE 13 – Put de l'option en fonction des paramètres d'espace x et de temps t

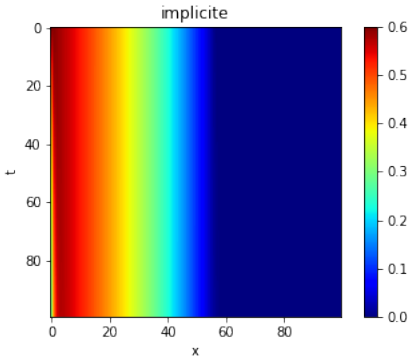

FIGURE 14 – Put de l'option en fonction du paramètre d'espace x (sous contrainte de stabilité respectée) $(x \longmapsto p(t,x))$

Optimisation du procédé de résolution Cela représente un certain coût temporel à la vue du nombre d'opérations à réaliser. Afin de résoudre ces systèmes, il faut adopter une bonne méthodologie.
On a d'abord voulu implémenter une méthode du pivot de Gauss afin de résoudre le système.

$$\begin{cases} \mathbf{P_0} \in \mathbb{R}^N \\ \forall m \in [\![0, M-1]\!],\ A\mathbf{P_{m+1}} = \mathbf{P_m} \end{cases}$$

Pour chaque étape, la méthode de Gauss coûte $\boxed{\frac{2}{3}N^3 + \text{toi}}$ [4] opérations.
Ce coût étant beaucoup trop important, nous pourrions alors réaliser au début du programme une décomposition LU de la matrice A :

$$A = LU$$

avec L matrice triangulaire inférieure à diagonale unitaire et U matrice triangulaire supérieure.
Pour chaque étape, la méthode LU coûte $\boxed{2N^2 + \text{toi}}$ opérations.
De plus, le protocole de résolution du système s'effectue en 2 étapes :

$$\forall m \in [\![0, M-1]\!],\ L\underbrace{U\mathbf{P_{m+1}}}_{y} = \mathbf{P_m} \iff Ly = \mathbf{P_m} \text{ puis } U\mathbf{P_{m+1}} = y$$

On peut encore rafiner le résultat. En effet, vu que A est tridiagonale. On peut donc appliquer le théorème suivant à A

Théorème 1 (Décomposition LU pour matrice tridiagonale) Soit $A \in \mathbb{GL}_n(\mathbb{R})$ telle que :

$$A = \begin{pmatrix} b_1 & c_1 & 0 & \cdots & \cdots & \cdots & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \ddots & \ddots & c_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & a_n & b_n \end{pmatrix}$$

On définit la suite suivante :

$$\begin{cases} \delta_0 &= 1 \\ \delta_1 &= b_1 \\ \delta_k &= b_k\delta_{k-1} - a_kc_{k-1}\delta_{k-2},\ \forall k \in [\![2, n]\!] \end{cases}$$

Si $\forall k \in [\![0, n]\!], \delta_k \neq 0$, alors, la matrice tridiagonale A admet **une unique décomposition LU** de la forme :

$$A = \underbrace{\begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ a_2\frac{\delta_0}{\delta_1} & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & a_n\frac{\delta_{n-2}}{\delta_{n-1}} & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} \frac{\delta_1}{\delta_0} & c_1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & \ddots & \ddots & c_{n-1} \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & \frac{\delta_n}{\delta_{n-1}} \end{pmatrix}}_{U}$$

Au niveau théorique, le coût des opérations $Lx = y$ et $Ux = y$ est en $\boxed{2N + \text{toi}}$ opérations.
Enfin, nous pouvons appliquer l'algorithme de Thomas [5], très utile pour la résolution de

4. **toi** pour termes d'ordre inférieur (ici à n^3)
5. Voir la ressource : https://en.wikipedia.org/wiki/Tridiagonal_matrix_algorithm

systèmes tridiagonaux.
Vous pourrez retrouver notre implémentation de l'algorithme de Thomas en Annexe du Notebook.

```
h ⟵ (x_max - x_min)/N
dt ⟵ (h**2)/2 # pour la stabilit

# Calcul des coefficients
b1 ⟵ (dt)*(1/dt - (2*alpha)/(h**2) + gamma)
b2 ⟵ (dt)*(alpha/(h**2) + beta/(2*h))
b3 ⟵ (dt)*(alpha/(h**2) - beta/(2*h))

# Cr ation de la matrice des p(t_m, x_j)
P ⟵ Matrice nulle de taille MxN
A ⟵ Matrice nulle de taille NxN


# Remplissage de la matrice A
Pour i allant de 0    N -1
    A[i][i]  ⟵ b1

Pour i allant de 0    N - 2
    A[i][i + 1]  ⟵ b2

Pour i allant de 1    N - 1
    A[i][i - 1]  ⟵ b3

# Inversion de la matrice
A_inv  ⟵ np.linalg.inv(A)

# Discr tisation de l'espace-temps
X_liste  ⟵ Subdivision    pas constant de [x_min, x_max] en N div
T_liste  ⟵ Subdivision    pas constant de [0, T] en M div


# Conditions initiales
Pour i allant de 1    N - 2
    P[0][i]  ⟵ max(K - np.exp(X_liste[i]), 0)

# Conditions aux limites
Pour t allant de 0    M - 1
    P[t][0]  ⟵ K*np.exp(-r*T_liste[t]) - np.exp(x_min)
    P[t][-1]  ⟵ 0


# Impl mentation du sch ma
Pour t allant de 0    M - 2
    P[t + 1, :]  ⟵ Produit de A_inv par P[t, :]
```

5.3 Différences finies par méthode de Crank-Nicholson

Cette méthode résulte d'une combinaison linéaire entre la méthode implicite et la méthode explicite. Plus précisément, il s'agit de la moyenne arithmétique entre les deux méthodes précédentes.

Discrétisation des composantes différentielles de p

$$\frac{\partial p}{\partial t}(t_{m+\frac{1}{2}}, x_j) \simeq \frac{1}{\Delta t}(p(t_{m+1}, x_j) - p(t_m, x_j))$$
$$\frac{\partial p}{\partial x}(t_{m+\frac{1}{2}}, x_j) \simeq \frac{1}{4h}(p(t_{m+1}, x_{j+1}) - p(t_{m+1}, x_{j-1}) + p(t_m, x_{j+1}) - p(t_m, x_{j-1}))$$
$$\frac{\partial^2 p}{\partial x^2}(t_{m+\frac{1}{2}}, x_j) \simeq \frac{1}{2h^2}(p(t_{m+1}, x_{j+1}) - 2p(t_{m+1}, x_j) + p(t_{m+1}, x_{j-1}) + p(t_m, x_{j+1}) - 2p(t_m, x_j) + p(t_m, x_{j-1}))$$

Détermination du système à résoudre Par injection dans l'EDP (BS-EDP) :

$$\begin{aligned}
\text{(BS-EDP)} &\iff \frac{1}{\Delta t}(\mathbf{p}^j_{m+1} - \mathbf{p}^j_m) + \frac{\alpha}{2h^2}(\mathbf{p}^{j+1}_{m+1} - 2\mathbf{p}^j_{m+1} + \mathbf{p}^{j-1}_{m+1} + \mathbf{p}^{j+1}_m - 2\mathbf{p}^j_m + \mathbf{p}^{j-1}_m) \\
&+ \frac{\beta}{4h}(\mathbf{p}^{j+1}_{m+1} - \mathbf{p}^j_{m+1} + \mathbf{p}^j_{m+1} - \mathbf{p}^j_{m+1}) + \frac{\gamma}{2}(\mathbf{p}^j_{m+1} + \mathbf{p}^j_{m+1}) = 0 \\
&\iff \mathbf{p}^j_{m+1}(\frac{1}{\Delta t} - \frac{\alpha}{h^2} + \frac{\gamma}{2}) + \mathbf{p}^{j+1}_{m+1}(\frac{\alpha}{2h^2} + \frac{\beta}{4h}) + \mathbf{p}^{j-1}_{m+1}(\frac{\alpha}{2h^2} - \frac{\beta}{4h}) \\
&+ \mathbf{p}^j_m(-\frac{1}{\Delta t} - \frac{\alpha}{h^2} + \frac{\gamma}{2}) + \mathbf{p}^{j+1}_m(\frac{\alpha}{2h^2} + \frac{\beta}{4h}) + \mathbf{p}^{j-1}_m(\frac{\alpha}{2h^2} - \frac{\beta}{4h}) = 0 \\
&\iff \mathbf{p}^j_{m+1}(\frac{1}{\Delta t} - \frac{\alpha}{h^2} + \frac{\gamma}{2}) + \mathbf{p}^{j+1}_{m+1}(\frac{\alpha}{2h^2} + \frac{\beta}{4h}) + \mathbf{p}^{j-1}_{m+1}(\frac{\alpha}{2h^2} - \frac{\beta}{4h}) \\
&= \mathbf{p}^j_m(\frac{1}{\Delta t} + \frac{\alpha}{h^2} - \frac{\gamma}{2}) + \mathbf{p}^{j+1}_m(-\frac{\alpha}{2h^2} - \frac{\beta}{4h}) + \mathbf{p}^{j-1}_m(-\frac{\alpha}{2h^2} + \frac{\beta}{4h}) \\
&\iff \boxed{c_1\mathbf{p}^j_{m+1} + c_2\mathbf{p}^{j+1}_{m+1} + c_3\mathbf{p}^{j-1}_{m+1} = c_4\mathbf{p}^j_m + c_5\mathbf{p}^{j+1}_m + c_6\mathbf{p}^{j-1}_m}
\end{aligned}$$

avec

$$\begin{cases}
c_1 &= \frac{1}{\Delta t} - \frac{\alpha}{h^2} + \frac{\gamma}{2} \\
c_2 &= \frac{\alpha}{h^2} + \frac{\beta}{2h} \\
c_3 &= \frac{\alpha}{h^2} - \frac{\beta}{2h} \\
c_4 &= \frac{1}{\Delta t} + \frac{\alpha}{h^2} - \frac{\gamma}{2} \\
c_5 &= -c_2 \\
c_6 &= -c_3
\end{cases}$$

Numérisation du problème & Résolution

$$\forall m \in [\![0, M-1]\!], A\mathbf{P_{m+1}} = B\mathbf{P_m} \iff \boxed{\mathbf{P_{m+1}} = A^{-1}B\mathbf{P_m}}$$

où A et B sont 2 matrices tridiagonales telles que :

$$A = \begin{pmatrix}
c_1 & c_2 & 0 & \cdots & \cdots & \cdots & 0 \\
b_3 & c_1 & c_2 & 0 & \cdots & \cdots & 0 \\
0 & b_3 & \ddots & \ddots & \ddots & & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\
\vdots & & & \ddots & \ddots & \ddots & c_2 \\
0 & \cdots & \cdots & \cdots & 0 & c_3 & c_1
\end{pmatrix}$$

et

$$B = \begin{pmatrix} c_4 & c_5 & 0 & \cdots & \cdots & \cdots & 0 \\ c_6 & c_4 & c_5 & 0 & \cdots & \cdots & 0 \\ 0 & c_6 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \ddots & \ddots & c_5 \\ 0 & \cdots & \cdots & \cdots & 0 & c_6 & c_4 \end{pmatrix}$$

Nous adoptons la même technique de résolution numérique que celles de la méthode implicite, car nous avons également choisi une méthode matricielle.
On obtient les courbes suivantes :

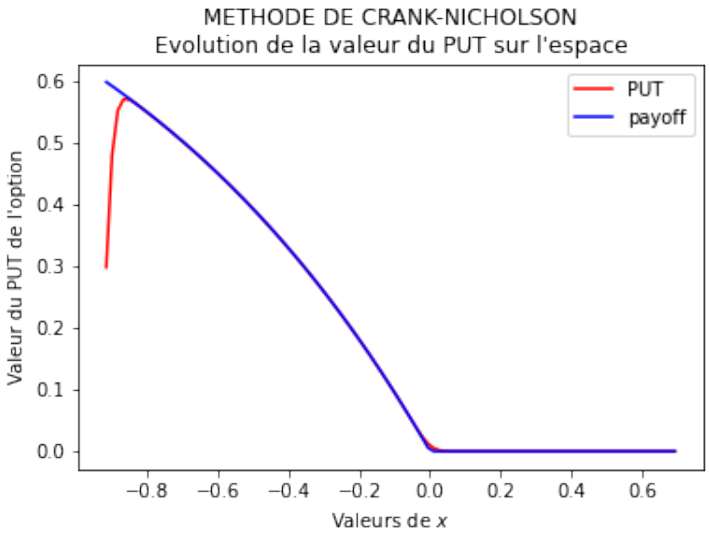

FIGURE 15 – Put de l'option en fonction des paramètres d'espace x et de temps t (sous contrainte de stabilité respectée)

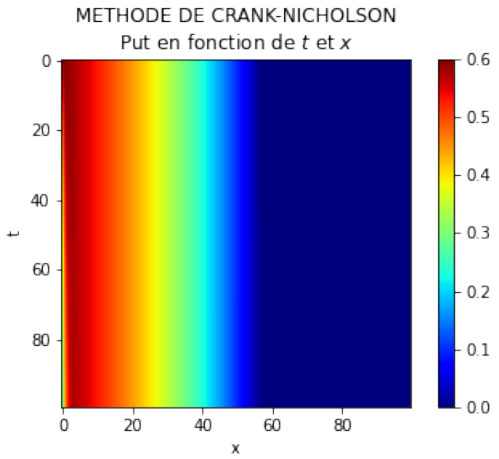

FIGURE 16 – Put de l'option en fonction des paramètres d'espace x et de temps t (sous contrainte de stabilité respectée) (vue en 2D)

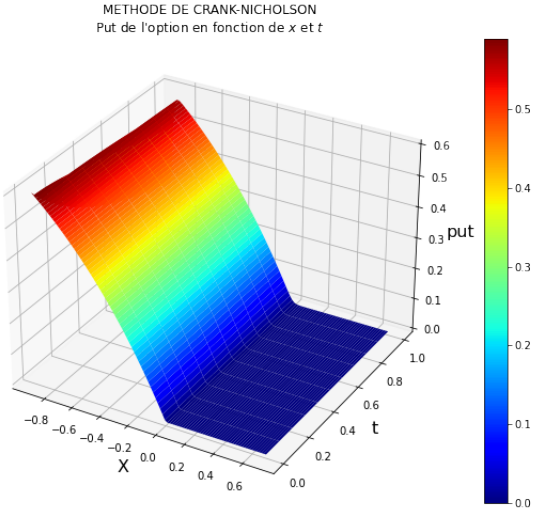

FIGURE 17 – Put de l'option en fonction du paramètre d'espace x (sous contrainte de stabilité respectée) $(x \longmapsto p(t,x))$

5.4 Evolution de l'erreur relative au sein des trois méthodes

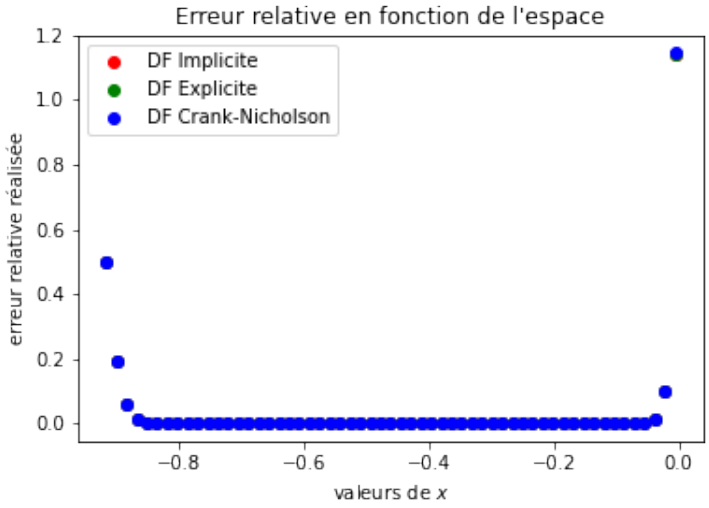

FIGURE 18 – Evolution de l'erreur relative selon les méthodes

5.5 Cours de $\mathbf{P}_M$

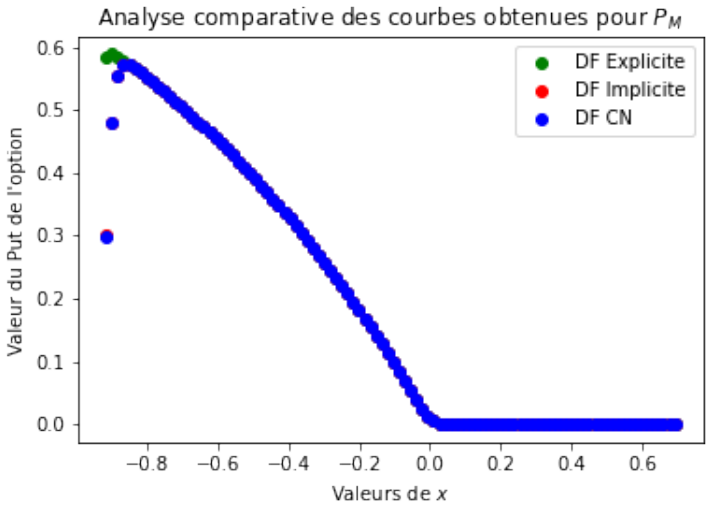

FIGURE 19 – Superposition des 3 courbes obtenues pour $\mathbf{P}_M$

Table des figures