Projet Maths

Gautier Poursin, Andrew Farndon, Théo Lazzaroni

22 mai 2020

1 Modèle de Cox-Ross-Rubinstein

1. Dans un premier temps, nous allons déterminer $q_N = Q(T_1^N = 1 + h_N)$. On sait que T_1^N ne prend seulement que 2 valeurs= $1 + h_N$ ou $1 + b_N$. Donc, par définition de l'espérance ,
$E_Q[T_1^N] = (1 + h_N)Q(T_1^N = 1 + h_N) + (1 + b_N)Q(T_1^N = 1 + b_N)$.

Or, $Q(T_1^N = 1 + b_N) = 1 - Q(T_1^N = 1 + h_N)$

Donc, $E_Q[T_1^N] = (1 + h_N)Q(T_1^N = 1 + h_N) + (1 + b_N)(1 - Q(T_1^N = 1 + h_N))$

Donc, $E_Q[T_1^N] = (1 + h_N)q_N + (1 + b_N)(1 - q_N) = 1 + r_N$ selon l'énoncé.
On peut alors en déduire une expression pour q_N.

On obtient: $\boxed{q_N = \frac{r_N - b_N}{h_N - b_N}}$

2.
Par définition, on a $p_{\prime(N):=\frac{1}{(1+r_N)^N}} E_Q[f(S_{t_N}^N)]$.
Pour N=2, on a la formule suivante :
$P_{(2)} = \frac{1}{(1+r_2)^2} f(s(1+h_N)^2)q_2^2 + \frac{2}{(1+r_2)^2} f(s(1+h_2)(1+b_2))q_2(1-q2) + \frac{1}{(1+r_2)^2} f(s(1+b_2)^2)(1-q_2)^2$.

Or, on sait qque $\binom{2}{0} = 1$, $\binom{2}{1} = 2$ et $\binom{2}{2} = 1$.
Donc, $P_{(2)} = \frac{\binom{2}{2}}{(1+r_2)^2} f(s(1+h_N)^2)q_2^2 + \frac{\binom{2}{1}}{(1+r_2)^2} f(s(1+h_2)(1+b_2))q_2(1-q2) + \frac{\binom{2}{0}}{(1+r_2)^2} f(s(1+b_2)^2)(1-q_2)^2$

On peut réitérer au rang k, cela forme une récurence. En effet, $S_{t_i}^{(N)} = T_i^{(N)} S_{t_{i-1}}^{(N)}$.
Or, les T_i sont 2 à 2 indépendants et prennent seulement 2 valeurs : $1 + h_N$ ou $1 + b_N$ donc, par récurence, $S_{t_i}^{(N)} = S_{t_0}(1 + h_N)^k(1 + b_N)^{N-k}$

Donc, $E_Q[f(S_{T_N})] = \sum_{k=0}^{N} \binom{N}{k} f(s(1 + h_N)^k(1 + b_N)^{N-k})(q_N^k)(1 - q_N)^{N-k}$ Finalement, $\boxed{P_{(N)} = \frac{1}{(1+r_N)^N} \sum_{k=0}^{N} \binom{N}{k} f(s(1 + h_N)^k(1 + b_N)^{N-k})(q_N^k)(1 - q_N)^{N-k}}$

Premier pricer

3. (voir fichier python)

4.

```
>>> pricer1(0.01,30,-0.05,0.05,100,f)
1.61398258566672
```

Figure 1: Résultat du premier pricer

Ce résultat semble cohérent avec la réalité puisque il est supérieur a 1. Il faudra comparer avec le pricer 2 puis le pricer 3 pour vérifier que nous obtenons des résultats identiques.

Deuxième pricer

5. (voir fichier python)
6.
Pour ce deuxième pricer, nous avons décidé d'afficher seulement l'élement qui se situe sur la première ligne, première colonne en position (1,1). Les résultats situés dans les autres cases de la matrice sont les valeurs de chaques noeuds l'arbre, lorsque j¿i avec j le nombre de colonnes.
Voici l'arbre de résultats obtenu:

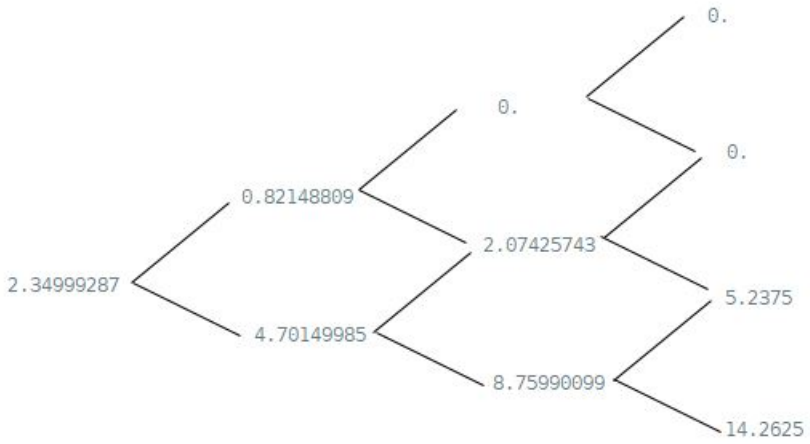

Figure 2: Arbre de valeurs pour le second pricer

Comparaison

7.
Dans cette question, nous souhaitons comparer les valeurs obtenues par ces 2 pricer. Logiquement, les 2 pricers doivent donner le même résultat.

```
def compare (f,rN,bN,hN,s):
    N=random.randint(5,15)
    print (N);
    return pricer2(f,N,rN,bN,hN,s)- pricer1(rN,N,bN,hN,s,f)
```

Figure 3: Fonction de comparaison

Nous ne donnons pas N en argument puisque il varie entre 5 et 15 donc lorsqu'on fait appel a la fonction, N variera automatiquement. Voici quelques résultats obtenu:

```
>>> compare(f,0.01,-0.05,0.05,100)
8
-4.4408920985006262e-16
```

Figure 4: Différence lorsque N=8

```
>>> compare(f,0.01,-0.05,0.05,100)
15
-4.4408920985006262e-16
```

Figure 5: Différence lorsque N=15

```
>>> compare(f,0.01,-0.05,0.05,100)
11
0.0
```

Figure 6: Différence lorsque N=11

On remarque que la différence entre les 2 pricers est presque nulle. Si elle n'est pas nulle, elle est de l'ordre de $e-16$ donc nos pricers sont cohérent. Ils permettent d'obtenir les mêmes résultats.

La couverture

8.
À la date terminale, le portefeuille doit vérifier l'égalité suivante :
$\alpha_{N-1}(S^{(N)}_{t_{N-1}})S^{(N)}_{t_N} + \beta_{N-1}(S^{(N)}_{t_{N-1}})S^0_{t_N} = f(S^{(N)}_{t_N})$
Or à un instant T, on a deux états de $S^{(N)}_{t_N}$:
- Soit $S^{(N)}_{t_N} = (1+h_N)S^{(N)}_{t_{N-1}}$
- Soit $S^{(N)}_{t_N} = (1+b_N)S^{(N)}_{t_{N-1}}$

On a donc le système à deux équations suivant :
$\alpha_{N-1}(S^{(N)}_{t_{N-1}})(1+h_N)S^{(N)}_{t_{N-1}} + \beta_{N-1}(S^{(N)}_{t_{N-1}})S^0_{t_N} = f((1+h_N)S^{(N)}_{t_{N-1}})$
$\alpha_{N-1}(S^{(N)}_{t_{N-1}})(1+b_N)S^{(N)}_{t_{N-1}} + \beta_{N-1}(S^{(N)}_{t_{N-1}})S^0_{t_N} = f((1+b_N)S^{(N)}_{t_{N-1}})$

En résolvant ce système, on obtient finalement :

$$\boxed{\alpha_{N-1}(S^{(N)}_{t_{N-1}}) = \frac{f((1+h_N)S^{(N)}_{t_{N-1}}) - f((1+b_n)S^{(N)}_{t_{N-1}})}{S^{(N)}_{t_{N-1}}(h_N - b_N)}}$$

$$\boxed{\beta_{N-1}(S^{(N)}_{t_{N-1}}) = \frac{f((1+b_n)S^{(N)}_{t_{N-1}})(1+h_N) - f((1+h_N)S^{(N)}_{t_{N-1}})(1+b_N)}{(1+r_N)^N(h_N - b_N)}}$$

9.
On réalise la même disjonction de cas qu'à la question précédente et on obtient:

$$\boxed{\alpha_{k-1}(S^{(N)}_{t_{k-1}}) = \frac{v_k((1+h_N)S^{(N)}_{t_{k-1}}) - v_k((1+b_n)S^{(N)}_{t_{k-1}})}{S^{(N)}_{t_{k-1}}(h_N - b_N)}}$$

$$\boxed{\beta_{k-1}(S_{t_{k-1}}^{(N)}) = \frac{v_k((1+b_n)S_{t_{k-1}}^{(N)})(1+h_N) - v_k((1+h_N)S_{t_{k-1}}^{(N)})(1+b_N)}{(1+r_N)^N(h_N-b_N)}}$$

10.
Nous allons tout d'abord calculer α_0 et α_1 avec les formules précédentes. On a :
$\alpha_0(S_{t_0}^{(N)}) = \frac{v_1((1+h_N)S_{t_0}) - v_1((1+b_N)S_{t_0})}{S_{t_0}(h_N-b_N)}$

Avec l'arbre de la question 6, on obtient $\boxed{\alpha_0 = \frac{4.70149985-0.82148809}{100*0.1} = 0.388}$
Après calcul grace à la question6, on obtient $\boxed{\beta_0 = -33.9}$

Pour l'instant 1, on a cette fois ci 2 cas possibles car $S_{t_N}^{(N)} = (1+h_N)S_{t_{N-1}}^{(N)}$ ou $S_{t_N}^{(N)} = (1+b_N)S_{t_{N-1}}^{(N)}$
On obtient donc 2 valeurs possibles pour α_1:
$\alpha_1(S_{t_1}^{(N)}) = \frac{f(s(1+h_N)^2) - f(s(1+h_N)(1+b_N))}{s(1+h_N)(h_N-b_N)}$ ou
$\alpha_1(S_{t_1}^{(N)}) = \frac{-f(s(1+b_N)^2) + f(s(1+h_N)(1+b_N))}{s(1+h_N)(h_N-b_N)}$.
Après un calcul sur python, on obtient : $\boxed{\alpha_1((1+h_N)S_{t_{N-1}}^{(N)}) = 0.976}$ ou bien $\boxed{\alpha_1((1+b_N)S_{t_{N-1}}^{(N)}) = 0.0}$
Concernant β_1, on a :
$\beta_1(S_{t_1}^{(N)}) = \frac{f((1+b_N)(1+h_N)s)(1+h_N) - f((1+h_N)^2 s)(1+b_N)}{(1+r_N)^N(h_N-b_N)}$ ou
$\beta_1(S_{t_1}^{(N)}) = \frac{f((1+b_N)^2 s)(1+h_N) - f((1+h_N)(1+b_N)s)(1+b_N)}{(1+r_N)^N(h_N-b_N)}$
Après un calcul sur python, on obtient: $\boxed{\beta_1((1+h_N)S_{t_{N-1}}^{(N)}) = -91.785}$ ou bien $\boxed{\beta_1((1+b_N)S_{t_{N-1}}^{(N)}) = 0.0}$.
De plus, on a $100\alpha_0 + \beta_0 = 4.9$ et lorsque l'on utilise **pricer2** avec la fonction call (max(x-100,0)) on obtient 5.2 ce qui nous conforte dans l'idée que le résultat est cohérent.

Modèle de Black-Scholes

11.
Nous allons dans cette partie appliquer la formule d'Ito pour déterminer une formule de S_t a la fonction $ln(S_t)$.
On a :
$dg(S_t) = g'(S_t)dS_t + \frac{\sigma^2 S_t^2}{2} g''(S_t)$
$dS_t = rS_t dt + \sigma S_t dB_t$

Donc, $dg(S_t) = (g'(S_t)rS_t + \frac{\sigma^2 S_t^2}{2} g''(S_t))dt + g'(S_t)S_t\sigma dB_t$

Or, $g(S_t) = ln(S_t)$ donc $g'(S_t) = \frac{1}{S_t}$ et $g''(S_t) = -\frac{1}{S_t^2}$

Donc, $\boxed{dln(S_t) = (r - \frac{\sigma^2}{2})dt + \sigma dB_t}$.
En divisant par dt, qui est non nul, on obtient $\frac{dln(S_t)}{dt} = (r - \frac{\sigma^2}{2}) + \sigma \frac{dB_t}{dt}$.
Ensuite, nous allons primitiver notre expression :
$ln(S_t) - ln(S_0) = (r - \frac{\sigma^2}{2})t + \sigma B_t + \sigma B_0$.

Or , $B_0 = 0$ donc $ln(S_t) = ln(S_0) + (r - \frac{\sigma^2}{2})t + \sigma B_t$.
Or, $\sigma B_t \sim N(0, \sigma^2 t)$. Donc $\boxed{ln(S_t) \sim N(ln(S_0) + (r - \frac{\sigma^2}{2})t, \sigma^2 t)}$ soit

$\boxed{S_t =^{(loi)} S_0 exp((r - \frac{\sigma^2}{2})t + \sigma\sqrt{t}\epsilon)}$ avec $\epsilon \sim N(0, 1)$

Le pricer par la méthode de Monte-Carlo

12. (voir fichier python)

13.
Voici le graphique obtenu par le pricer 3. L'inconvénient de cette fonction pourrait être la compléxité de l'algorithme. Il faut plusieurs minutes pour obtenir le graphe souhaité.

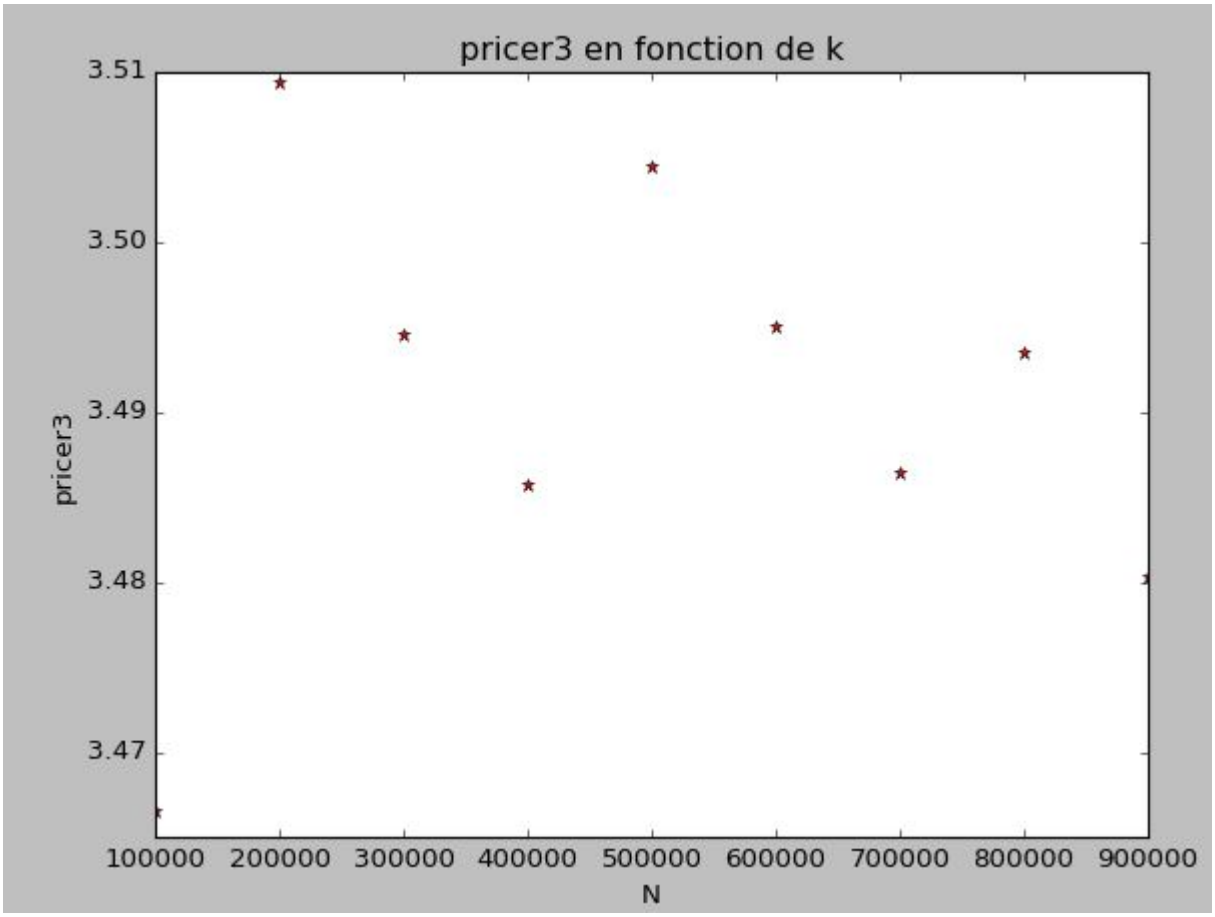

Figure 7: Graphe représentant Pricer 3

On remarque que le pricer varie entre 3.48 et 3.51. Il ne s'éloigne jamais de ces valeurs pour nos valeurs et intervalles choisis.

14. Posons $X_1, X_2, ...$ une suite de variables aléatoires suivant une loi $(e^{-rT} f(s* exp((r + \frac{\sigma^2}{2})T + \sigma\sqrt{T}\xi_i)))_{i\in 1,...,n}$. On sait que les $(\xi_i)_{i\in 1,...,n}$ sont i.i.d donc les $(X_i)_{i\in 1,...,n}$ sont i.i.d.

En utilisant la Loi Forte des Grands Nombres on obtient :
$P[lim\frac{1}{n}\sum_{i=0}^{n} X_i = E[X_1]] = 1$ i.e $\frac{1}{n}\sum_{i=0}^{n} X_i \longrightarrow E[X_1]$

Il nous suffit donc de montrer que $E[X_1] = p$ et on obtiendra le résultat désiré. On sait que B est un mouvement Brownien donc $B_t - B_s$ suit une loi N(0,t-s) donc en prenant s=0, puisque $B_0 = 0$, on divise par $\sqrt{T}$ et on obtient $\frac{B_T}{\sqrt{T}}$ suit

N(0,1). On définit donc $\xi_1 = \frac{B_T}{\sqrt{T}}$. Finalement :

$$E[X_1] = E[e^{-rT} f(s * exp((r - \frac{\sigma^2}{2}T + \sigma B_T)] = E[e^{-rT} f(S_T)] = p$$

Pour conclure :

$$\hat{p}_{(N)} \rightarrow_{p.s} p$$

Le pricer par formule fermée

15. (voir fichier python)

16.
Voici la formule obtenu avec la fonction put et les valeurs de l'énoncé:

```
>>> put(100,0.01,0.1,1,100)
3.4902197839388904
```

Figure 8: Valeur du put

Cette valeur est très proche des valeurs obtenues par le pricer3. On va donc tracer la droite correspondant à la valeur du put avec pricer3 et on va vérifier que les valeurs du pricer 3 se situent autour du put.

17.

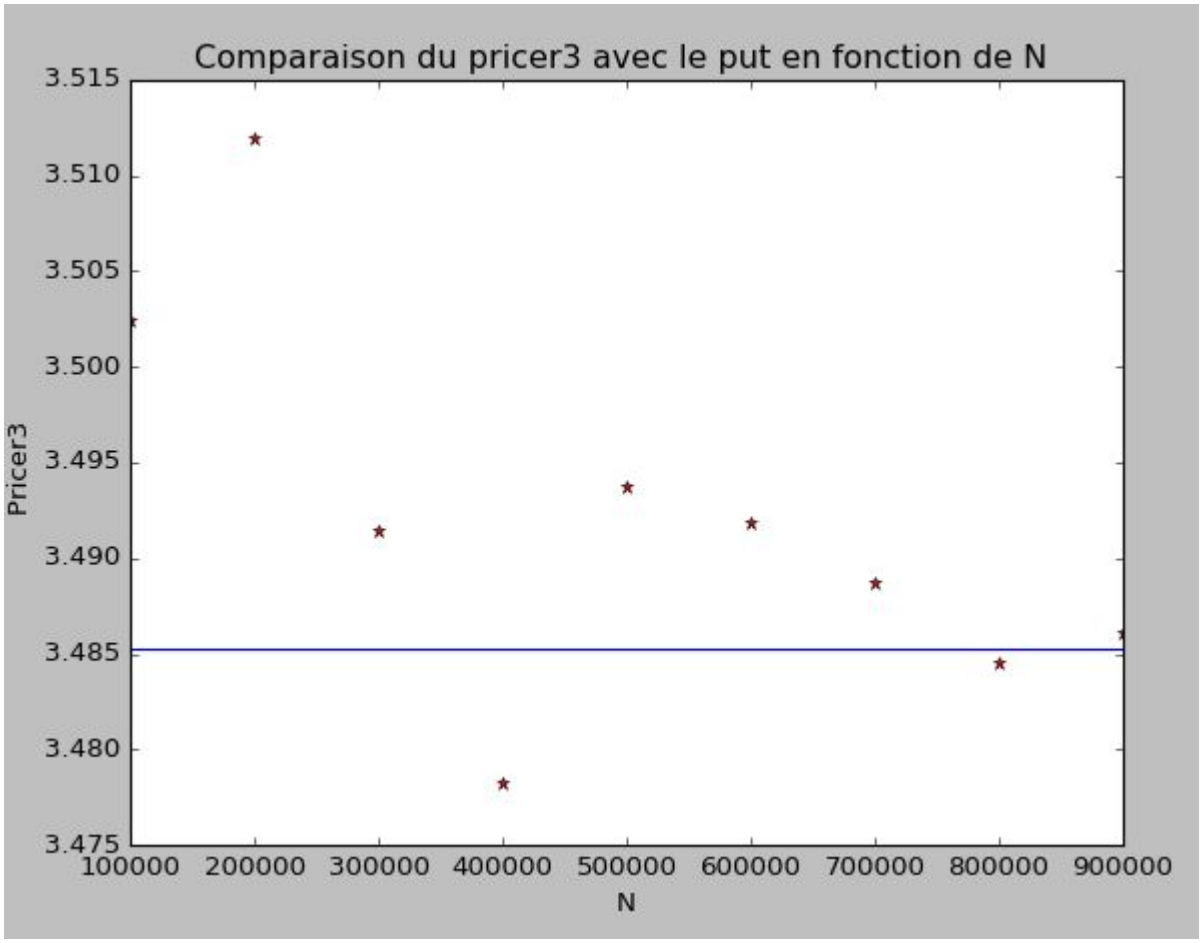

Figure 9: Pricer3 avec la droite de la fonction put

On remarque que les valeurs de la fonction pricer3 restent très proches de la valeur du **put**. Au plus, pricer3 s'éloigne de 0.014 de la valeur du put (lorsque N=500000). Ainsi, on peut considérer le pricer3 comme juste. Il est semblable au prix de l'option.

18.

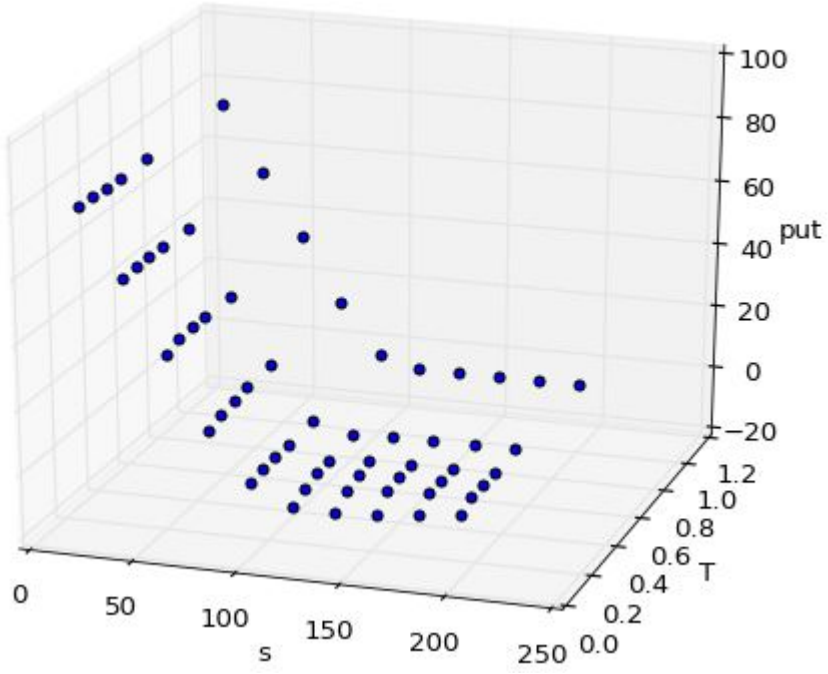

Figure 10: Allure du put en fonction de s et T

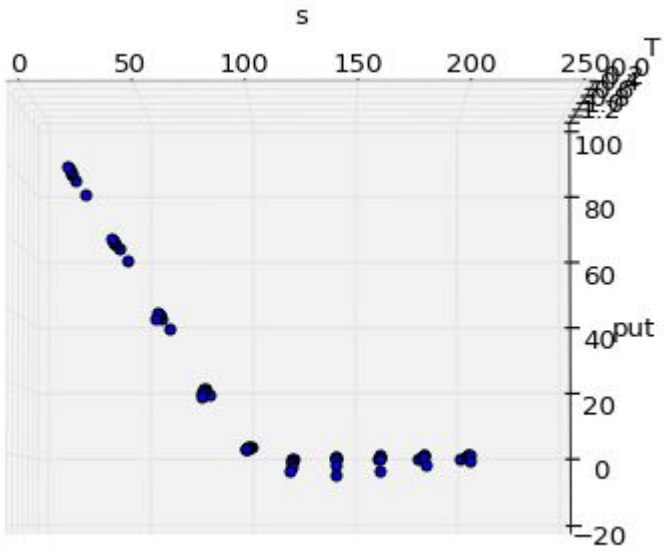

Figure 11: Allure du put en fonction de s et T

On remarque avec ces 2 graphiques que la valeur de T a très peu d'influence sur la valeur du put. La valeur du put est légèrement croissante selon T. L'influence de s est bien plus importante sur le put. Lorsque $s > 100$, alors la valeur du put est nulle. Cependant, lorsque $s \in [0, 100]$, alors la valeur du put décroit linéairement.

19.

Nous n'avons pas réussi à tracer cette fonction, nous avons mis en commentaire dans le fichier Python le code que nous avons établi, mais celui ci provoque une erreur de compilation.

20.

On pose ici $\frac{\partial P}{\partial t} + rS\frac{\partial P}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 P}{\partial^2 S} = rP$

Pour la résolution explicite, on pose $\frac{\partial P(t_n,s_i)}{\partial S} = \frac{1}{\Delta s}(P(t_n, s_i) - P(t_n, s_{i-1}))$, on obtient après remplacement dans l'équation:

$$\boxed{P(t_{n+1}, s_i) = \frac{1}{1-r\Delta T}((1 - \frac{rS\Delta T}{\Delta S} + \frac{\sigma^2 S^2 \Delta T}{\Delta S^2})P(t_n, s_i) + (\frac{rS\Delta T}{\Delta S} - \frac{\sigma^2 S^2 \Delta T}{2\Delta S^2})P(t_n, s_{i-1}) - \frac{\sigma^2 S^2 \Delta T}{2\Delta S^2}P(t_n, s_{i+1})))}$$

Pour la résolution implicite, on pose $\frac{\partial P(t_n,s_i)}{\partial S} = \frac{1}{\Delta s}(P(t_{n+1}, s_i) - P(t_{n+1}, s_{i-1}))$.
Cette fois ci, :

$$\boxed{P(t_n, s_i) = (1 - r\Delta T + \frac{rS\Delta T}{\Delta S} + \frac{\sigma^2 S^2 \Delta T}{\Delta S^2})P(t_{n+1}, s_i) + \frac{\sigma^2 S^2 \Delta T}{2\Delta S^2})P(t_{n+1}, s_{i+1}) + (\frac{\sigma^2 S^2 \Delta T}{2\Delta S^2} - \frac{rS\Delta T}{\Delta S})P(t_{n+1}, s_{i-1})}$$

La méthode de Crank-Nicholson consiste à faire la moyenne entre le méthode implicite et la méthode explicite ie calculer $\frac{P(t_n,s_i)+P(t_{n+1},s_i)}{2}$. On obtient l'équation suivante:

$(\frac{1}{\Delta T}+\frac{rS}{2\Delta S}+\frac{\sigma^2S^2}{4\Delta S^2})P(t_{n+1},s_i)-(\frac{rS}{2\Delta S}+\frac{\sigma^2S^2}{4\Delta S^2})P(t_{n+1},s_{i+1})+\frac{\sigma^2S^2}{4\Delta S^2}P(t_{n+1},s_{i-1}) =$
$(-r-\frac{rS}{2\Delta S}+\frac{\sigma^2S^2}{4\Delta S^2}+\frac{1}{\Delta T})P(t_n,s_i)+(\frac{rS}{2\Delta S}+\frac{\sigma^2S^2}{4\Delta S^2})(P(t_n,s_{i+1})-\frac{\sigma^2S^2}{4\Delta S^2}P(t_n,s_{i-1})$

On peut écrire cette équation de cette forme, qui sera plus lisible:
$a_{n+1,i}P(t_{n+1},s_i)+a_{n+1,i+1}P(t_{n+1},s_{i+1})+a_{n+1,i-1}P(t_{n+1},s_{i-1}) = b_{n,i}P(t_n,s_i)+$
$b_{n,i+1}P(t_n,s_{i+1}) + b_{n,i-1}P(t_n,s_{i-1})$

Cette équation est une équation vectorielle: $AP(t_{n+1},s_i) = BP(t_n,s_i)+C$ avec A,B,C des matrices carrées de taille $(M+2)*(M+2)$,, car on doit prendre en compte les conditions aux limites. Au vu de la forme de l'équation précédente, on déduit assez rapidement que A et B sont **des matrices tridiagonales** avec , $\forall i \in [1,M]$:
- $a_{i,i} = \frac{1}{\Delta T} + \frac{rS}{2\Delta S} + \frac{\sigma^2S^2}{4\Delta S^2}$
-$a_{i,i+1} = -(\frac{rS}{2\Delta S} + \frac{\sigma^2S^2}{4\Delta S^2})$
-$a_{i,i-1} = \frac{\sigma^2S^2}{4\Delta S^2}$
-$b_{i,i} = -r - \frac{rS}{2\Delta S} + \frac{\sigma^2S^2}{4\Delta S^2} + \frac{1}{\Delta T}$
-$b_{i,i+1} = (\frac{rS}{2\Delta S} + \frac{\sigma^2S^2}{4\Delta S^2})$
-$b_{i,i-1} = -\frac{\sigma^2S^2}{4\Delta S^2}$
Ces coefficients correspondent à la méthode de Crank-Nicholson.
De manière générale, voici la valeurs des coefficients:
-$a_{i,i} = \frac{1}{\Delta T} + (1-\theta)(\frac{rS}{\Delta S} + \frac{\sigma^2S^2}{2\Delta S^2})$
-$a_{i,i+1} = -(1-\theta)(\frac{rS}{\Delta S} + \frac{\sigma^2S^2}{2\Delta S^2})$
-$a_{i,i-1} = (1-\theta)(\frac{\sigma^2S^2}{2\Delta S^2})$
-$b_{i,i} = -r - \theta(\frac{rS}{\Delta S} - \frac{\sigma^2S^2}{2\Delta S^2}) + \frac{1}{\Delta T}$
-$b_{i,i+1} = \theta(\frac{rS}{\Delta S} + \frac{\sigma^2S^2}{2\Delta S^2})$
-$b_{i,i-1} = -\theta(\frac{\sigma^2S^2}{2\Delta S^2})$

Lorsque $\theta = 0$, on retombe sur la méthode implicite.
Lorsque $\theta = 1$, on retombe sur la méthode explicite.
Lorsque $\theta = \frac{1}{2}$, on retombe sur la méthode de Crank-Nicholson.

A l'aide des conditions aux limites, on peut déterminer certains coefficients des matrices A et B, on obtient:

$$\begin{cases} -a_{0,0} = 1 \\ -a_{1,0} = 0 \\ -a_{M+1,M+1} = 1 \\ -a_{M,M+1} = 0 \end{cases}$$

$$\begin{cases} -b_{0,0} = 0 \\ -b_{1,0} = 0 \\ -b_{M+1,M+1} = 0 \\ -b_{M,M+1} = 0 \end{cases}$$

L'objectif désormais est d'avoir tous les coefficients situés sur la première colonne de notre tableau. Pour cela, il va falloit calculé les valeurs au coeur de la matrice en partant de la dernière colonne. Il faut faire une récurence décroissante pour déterminer P_n grâce à P_{n+1} puisqu'on, via les conditions limites, on connait les valeurs de la dernière colonne. Pour optimiser nos calculs, nous allons faire appel aux cours d'Analyse Numérique. Nos matrices sont tridiagonales donc nous allons utiliser **la décomposition LU** des matrices tridiagonales. Pour résoudre une équation de la forme Ax=B avec A matrice tridiagonale, il y a:

$$\begin{cases} 3(n-1)\ additions \\ 3(n-1)\ soustractions \\ 2n\ divisions \end{cases}$$

avce n la taille de la matrice. En théroie, la décomposition LU pour une matrice carrée de taille n à pour complexité $O(n^2)$. Ici, la complexité serait de $O(M^2)$.

2 Conclusion

En conclusion de ce projet, nous avons vu différentes méthodes pour calculer l'option du prix à payer, pour obtenir une certaine somme à un instant T. Nous avons par exemple implémenter 2 fonctions pricer qui retournent le même résultat mais avec un raisonnement différent. La dernière partie, malheureusement non terminée puisque nous n'avons pas réussi à tracer les graphiques, est aussi une différente méthode permettant de mener au résultat. Toutes ces méthodes se rejoignent sur le résultat, mais certaines possèdent à n'en pas douter des avantages que les autres n'ont pas.