Projet Mathématiques

Godart Loan, Micciche Luca

1 Modèle de Cox-Ross-Rubinstein

Question 1)

On recherche Q la probabilité risque-neutre tel que $E_Q(T_1^{(N)}) = 1 + r_N$. Pour cela, on va développer l'espérance, sachant que les variables aléatoires $T_i^{(N)}$ prennent la valeure $1 + h_N$ avec la probabilité q_N et la valeure $1 + b_N$ avec la probabilité $1 - q_N$.
On a donc:

$$\begin{aligned} E_Q(T_1^{(N)}) &= (1 + h_N)q_N + (1 + b_N)(1 - q_N) \\ &= 1 + r_N \end{aligned}$$

On isole ensuite q_N et on trouve alors:

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

Question 2)

Par définition, le prix de l'option qui paye $f(S_{t_N}^N)$ est:

$$p_{(N)} = \frac{1}{(1 + r_N)^N} E_Q(f(S_{t_N}^N))$$

On sait que $\forall i \in [0; N], S_{t_i} = T_i^{(N)} S_{t_{i-1}}$. Ainsi, on a donc $S_{t_N}^{(N)} = T_{t_N}...Tt_1 s$. Les différentes valeurs prisent par $S_{t_i}^{(N)}$ sont donc $s(1 + h_N)^k(1 + b_N)^{N-k}$, où k prend ses valeurs dans [0:N].
On va alors poser $\forall i \in [1, N], X_i = 1_{T_i = 1 + h_N}$. Alors les X_i suivent toutes une loi bernouilly de paramètre q_N. Ainsi, $X = \sum_{i=1}^N X_i$ suit une binomial de paramètre (N, q_N).
Avec ce qu'on à écrit, et par la formule du transfert, on a donc:

$$E_Q(f(S_{t_N}^{(N)})) = \sum_{i=1}^N f(x_i) \binom{N}{i} q_N^i (1 - q_N)^{N-i} \text{ avec } x_i = (1 + h_N)^i (1 + b_N)^{N-i}$$

On peut donc conlcure :

$$\boxed{p_{(N)} = \frac{1}{(1 + r_N)^N} \sum_{i=1}^N f(x_i) \binom{N}{i} q_N^i (1 - q_N)^{N-i}} \text{ avec } x_i = (1 + h_N)^i (1 + b_N)^{N-i}$$

1.1 Premier pricer

Question 3)

Pour implémenter la fonction, j'ai défini une première fonction **coeff_binom** qui calcul le coefficient binomial en fonction de deux entier donner. Ensutie, j'ai ecrit une fonction **price1** qui renvoie la valeur du pricer 1.

Question 4)

On effectue alors le test avec $f(x) = max(100 - x, 0)$, $s = 100$, $h_N = 0.05$, $b_n = -0.05$, $r_n 0.01 et N = 30$.
On obtient alors un premier pricer qui vaut 1.61398258566672.

1.2 Deuxième pricer

Question 5)

Pour comprendre comment implémenter le pricer 2, il faut dans un premier temps expliciter la formule de récurence associé au prix de l'option à la date t_k, c'est à dire la formule suivante :

$$v_k(S_{t_k}^{(N)}) = \frac{1}{1 + r_N} E_Q(v_{k+1}(S_{t_{k+1}}^{(N)})|S_{t_k}^{(N)})$$

On sait que $S_{t_{k+1}}^{(N)} = T_{t_{k+1}} S_{t_k}^{(N)}$. On va donc pouvoir développer l'espérance, en posant (pour simplifier la lecture des calculs) $Y_k = S_{t_k}^{(N)}$. Ainsi on a donc $E_Q(v_{k+1}(S_{t_{k+1}}^{(N)})|S_{t_k}^{(N)}) = E_Q(Y_{k+1} = T_{t_{k+1}} Y_k | Y_k = y_k)$. Détaillons maintenant le calcul:

$$\begin{aligned}
E_Q(v_{k+1}(Y_{k+1} = T_{t_{k+1}} Y_k)|Y_k = y_k) &= v_{k+1}((1 + h_N)y_k)Q(Y_{k+1} = (1 + h_N)y_k | Y_k = y_k) \\
&+ v_{k+1}((1 + b_N)y_k)Q(Y_{k+1} = (1 + b_N)y_k | Y_k = y_k) \\
&= v_{k+1}((1 + h_N)y_k)Q(T_{k+1}^{(N)} = (1 + h_N)|Y_k = y_k) \\
&+ v_{k+1}((1 + b_N)y_k)Q(T_{k+1}^{(N)} = (1 + b_N)|Y_k = y_k) \\
&= v_{k+1}((1 + h_N)y_k)q_N + v_{k+1}((1 + b_N)y_k)(1 - q_N) \quad \text{par indépendance de } T_{k+1}^{(N)} \text{ et } Y_k
\end{aligned}$$

On va donc à chaque ittération avoir un vecteur V_k dont les éléments seront les $v_k((1 + h_N)^i (1 + b_N)^{k-i}$.

L'initialisation sera alors le vecteur V_N, puis on va décrémenter l'indice jusqu'à 0, pour obtenir le prix de $p_{(N)}$.

Voici les deux vecteurs que l'on aura: $V_N = \begin{bmatrix} f(s(1 + h_N)^N) \\ f(s(1 + b_N)(1 + h_N)^{N-1} \\ ... \\ f(s(1 + b_N)^N) \end{bmatrix}$, $V_k = \begin{bmatrix} v_k(s(1 + h_N)^k) \\ v_k(s(1 + b_N)(1 + h_N)^{k-1} \\ ... \\ v_k(s(1 + b_N)^k) \end{bmatrix}$

On a donc comme formule vectorielle:

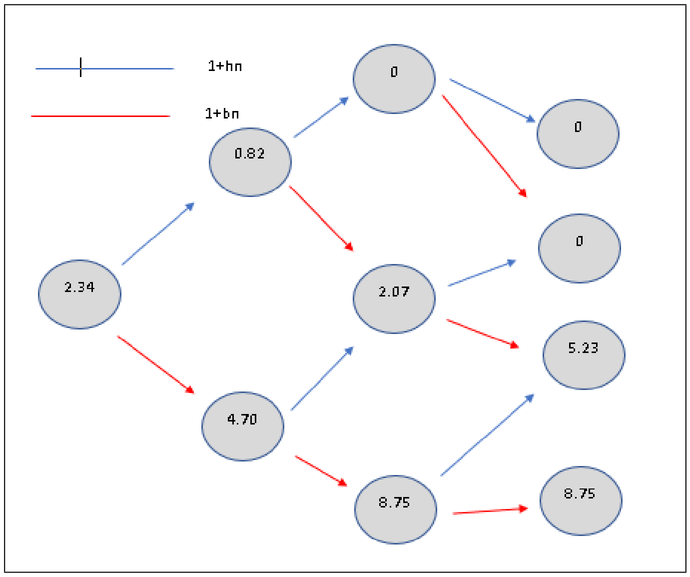
$$V_k[i] = \frac{1}{1 + r_N}(q_N V_{k+1}[i + 1] + (1 - q_N)V_{k+1}[i])$$

Question 5)

On effectue les tests avec les même paramètres, sauf N=3.

Question 6)

On obtient alors que le pricer 2 vaut: 2.349992866162413
Voici l'arbre des valeurs:



1.3 Comparaison

Question 7)

Pour N=13, on a:

- le pricer 1 vaut: 14.372799344018286
- le pricer2 vaut: 14.474435937892322

En valeur absolue, on a un écart de l'ordre de 0.1, soit très faible. Ainsi, ces deux méthodes semblent conduire au même résultat. On pourra faire d'autres test en utilisant le code, qui affiche directement la valeur absolue de la différence des deux valeurs.

1.4 Couverture

Question 8)

On note pour simplifier: $Y_N = S^{(N)}_{t_{N-1}}$. Le système d'équation à résoudre est celui-ci:

$$\left\{ \begin{array}{l} f((1+h_N)Y_N) = \alpha_{N-1}(Y_N)Y_N(1+h_N) + \beta_{N-1}(Y_N)(1+r_N)^N \\ f((1+b_N)Y_N) = \alpha_{N-1}(Y_N)Y_N(1+b_N) + \beta_{N-1}(Y_N)(1+r_N)^N \end{array} \right.$$

Par substitution:

$$\left\{ \begin{array}{l} \alpha_{N-1}(Y_N) = \frac{f((1+b_N)Y_N) - f((1+h_N)Y_N)}{Y_N(b_N-h_N)} \\ \beta_{N-1}(Y_N) = \frac{f((1+h_N)Y_N)(1+b_N) - f((1+b_N)Y_N)(1+h_N)}{(b_N-h_N)(1+r_N)^N} \end{array} \right.$$

Question 9)

On note cette fois-ci $Y_k = S^{(N)}_{t_{k-1}}$. Par le même principe on obtient:

$$\begin{cases} \alpha_{k-1}(Y_k) = \frac{v_k((1+b_N)Y_k)-v_k((1+h_N)Y_k)}{Y_k(b_N-h_N)} \\ \beta_{N-1}(Y_k) = \frac{v_k((1+h_N)Y_k)(1+b_N)-v_k((1+b_N)Y_k)(1+h_N)}{(b_N-h_N)(1+r_N)^N} \end{cases}$$

Question 10)

On considère ici N=2. On commence par calculer α_0 et β_0, qui dépendent de v_1, à l'aide de la formule suivante:

$$v_1(S^{(N)}_{t_1}) = \frac{1}{1+r_N} E_Q(v_{k+1}(S^{(N)}_{t_{k+1}})|S^{(N)}_{t_k})$$

On la développe suivant les deux valeurs prisent par $S^{(N)}_{t_0}$ et on obtient:

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

On remplace maintenant simplement v_1 dans ce que l'on a trouvé pour α_{k-1} et β_{k-1} pour k=1.
Les valeurs obtenues sont les suivantes:

- α_0 = 0.7961165048543688
- β_0 = -73.42822132151944

On veut maitenant calculer α_1 et β_1. Pour cela, on utilise les formules de la question 8. On distingue 2 cas, suivant la valeur de $S^{(N)}_{t_2}$.

Si $S^{(N)}_{t_2} = 1 + h_n$:

- α_1= 0.9761904761904762
- β_1= -91.78527665189932

Si $S^{(N)}_{t_2} = 1 + b_n$:

- α_1= 0
- β_1= 0

2 Modèle de Black-Scholes

2.1 Le modèle

Question 11)

On a $dS_t = S_t(rdt + \sigma dB_t)(1)$ et $dg(S_t) = g'(S_t)dS_t + \frac{|\sigma S_t|^2}{2}g''(S_t)dt(2)$.
On va appliquer (2) à $ln(S_t)$:

$$\begin{aligned} dg(ln(S_t)) &= \frac{dS_t}{S_t} + \frac{|\sigma S_t|^2}{2}(-\frac{1}{S_t^2})dt \\ &= \frac{dS_t}{S_t} - \frac{\sigma^2 dt}{2} && \text{on remplace avec (1)} \\ &= rdt + \sigma dB_t - \frac{\sigma^2 dt}{2} && \text{en divisant par dt} \\ \frac{dS_t}{dt} &= S_t(r - \frac{\sigma^2}{2} + \sigma\frac{dB_t}{ft}) \end{aligned}$$

On résout maintenant une équation différentielle du premier ordre sachant que $S_0(0) = s$.

La solution est donc $\boxed{S_t(t) = sexp((r - \frac{\sigma^2}{2})t + \sigma B_t)}$.

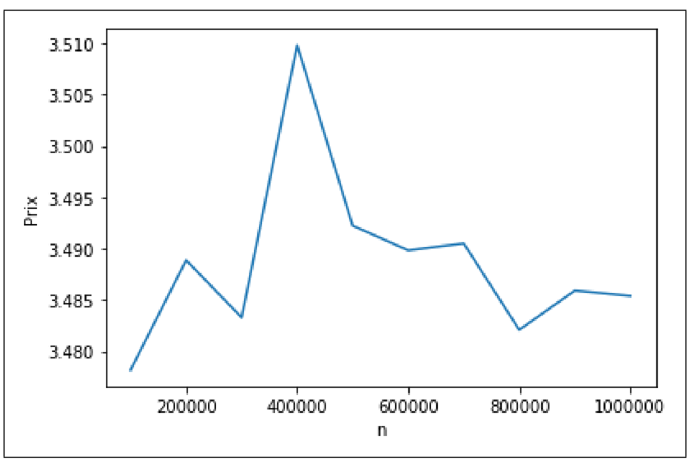
2.2 Le pricer par Monte-Carlo

Question 12)

Pour écrire le pricer 3 on génère les variables aléatoires de loi normale N(0,1) avec la bibliothèque scipy.math. Ensuite on a écrit price 3.

Question 13)

Voici le graphique obtenu avec les paramètres demandés:



Question 14)

Pour montrer la convergence presque sûr vers p on va utiliser la **loi forte des grand nombres**. Notons $Z_i = exp(-rT)f(s * exp(T(r - \frac{\sigma**2}{2}) + \sigma\sqrt{T}\xi_i))$. Montrons l'indépendance des Z_i. Pour cela on va noter $\psi(\xi_i) = Z_i$ où

ψ est une fonction croissante positive. Ainsi, on a:

$$\begin{aligned} P(\xi_i < x et \xi_j < y) &= P(\psi(\xi_i) < \psi(x) et \psi(\xi_j) < \psi(y)) \\ &= P(\xi_i < x)P(\xi_j < y) \qquad \text{par indépendance des } \xi \end{aligned}$$

On a donc en composant par ψ:

$$\begin{aligned} P(\psi(\xi_i) < \psi(x) et \psi(\xi_j) < \psi(y)) &= P(Z_i < \psi(x) et Z_j < \psi(y)) \\ &= P(\psi(\xi_i) < \psi(x))P(\psi(\xi_j) < \psi(y)) \\ &= Z_i < \psi(x)P(Z_j < \psi(y)) \end{aligned}$$

Ainsi, les X_i sont bien iid, et on peut appliquer LFGD, qui donne que:

$$\frac{1}{n}\sum_{i=1}^{n} X_i \longrightarrow E(Z_i)$$

Or on sait que B_t et $\sqrt{T}\xi_i$ suivent une loi normale (0,T), donc avec la question 11, on a bien $f(s * exp(T(r - \frac{\sigma**2}{2}) + \sigma\sqrt{T}\xi_i))$ qui possède la même loi que $f(S_T)$. Ainsi, on a prouvé le résultat souhaité, soit la convergence presque sûr de $p(n)$ vers p.

2.3 Le pricer par formule fermée

Question 15)

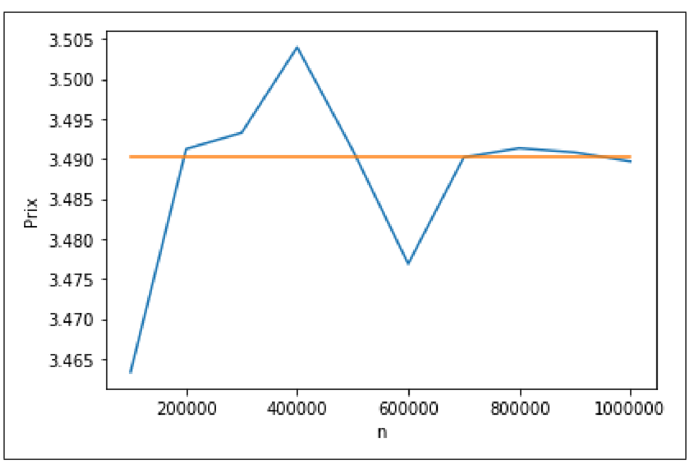
Voir le code python.

Question 16)

Par la formule de BS, le put vaut: 3.4902197839388904

Question 17)

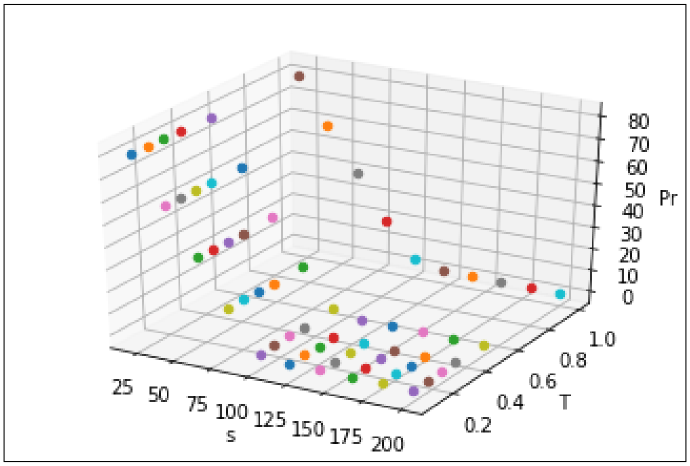
Voici le graphe obtenu:



On remarque alors une convergence de la fonction price3 vers le prix fourni par la méthode de Black-Scholes.

Question 18)

Voici le graphe en 3D que l'on obtient:

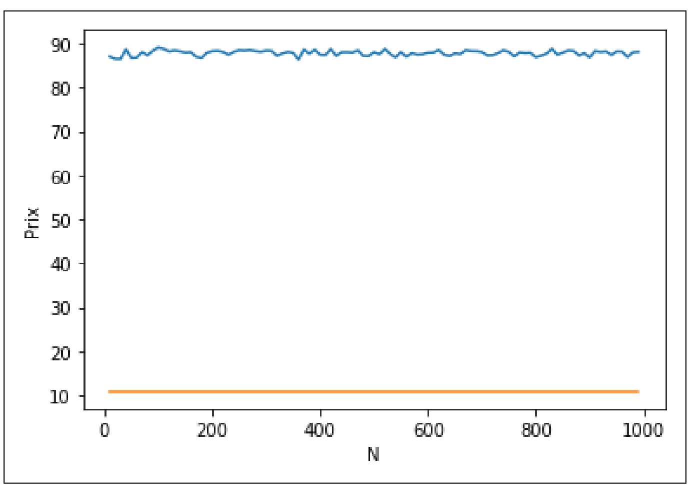


On remarque alors grâce à ce graphique que la fonction put est très dépendante de N. En effet, elle connait une forte décroissance vis-à-vis de celle-ci lorsque N augmente(surtout au début). Concernant T, on observe que ses variations impactent faiblement l'option. **En conclusion, le prix que permet de calculer le modèle de Black-Scholes est fortement dépendant du nombre d'itération.**

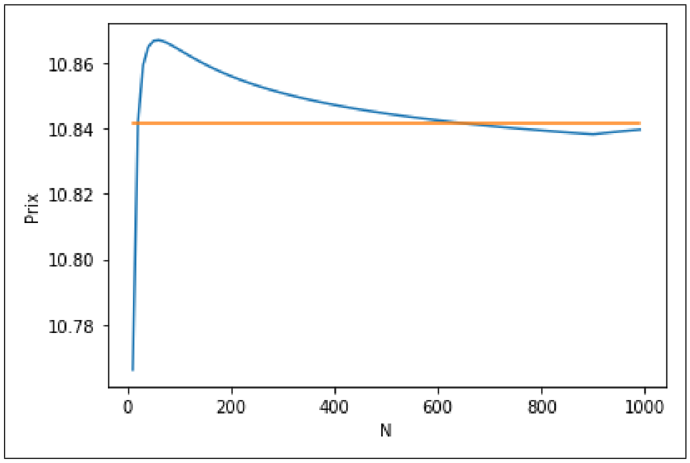
3 Convergence des prix

Question 19)

Pour cette question, l'objectif du graphique était de s'appercevoir que la fonction price2 allait tendre vers le put. Malheureusement, comme on peut le voir sur le graphe ci-dessous, je n'ai pas réussi à implémenter la fonction qui paye $max(100 - S_t, 0)$.



Pour voir si le contrat était tout de même rempli, j'ai testé avec price2 qui paye max(100-x,0). On voit bien qu'il y a convergence de la fonction price2 vers put.



Le problème vient donc de l'implémentation de S_T. Pour moi, on doit utiliser la fonction de la question 11, en génarant une loi normale (0,1) pour la valeur du mouvement brownien (T=1), et ensuite evaluer cette fonction en le point de la fonction price2 voulu. Le résultat n'est cependant pas celui escompté.

4 Black-Scholes

Question 20)

On veut donc ici approcher numériquement l'équation diférentielle de Black-Scholes par les méthodes d'Euler explicite, implicite et par la méthode de Crack-Nicholson.Voici l'EDP:

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

Pour cela, on se propose de discrétiser les dérivée partielle comme suit;

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

Détaillons maitenant ce que l'on obtient avec les différentes méthodes :

Pour Euler explicite, on prend l'EDP en (t_n, s_i), et on trouve:

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

Pour Euler implicite, on prend l'EDP en $(t_{n+1}, s_i$, et on trouve:

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

Pour Cranck-Nicholson, c'est la moyenne des deux méthodes, on obtient l'équation suivante:

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

On peut alors tirer le système suivant:

$$a_{i+1,i}P(t_n i+1, s_i) + a_{n+1,i-1}P(t_{n+1}, s_{i-1})a_{n+1,i+1}P(t_{n+1}, s_{i+1}) = b_{i,n}P(t_n, s_i) + b_{n,i-1}P(t_n, s_{i-1}) + b_{n,i+1}P(t_n, s_{i+1})$$

On a ainsi chaque coefficient $a_{i,i}$ et $b_{i,i}$ avec ceux de l'équation de Crack-Nicholson. On tombe donc sur un système matricielle de la forme $AP_{n+1} = BP_n + C_n$ où A,B sont deux matrices diagonales de tailles $2(M+2)$ et C_n la matrice contenant les conditions initiales.
Avec les conditions initiales, on peut déterminer certains coefficients de la matrice. Ensuite pour le reste, il faut utiliser les méthodes vu en MAN pour la résolution. On va procéder par récurence sur n.
Pour résoudre les systèmes, nous voulions procéder par décomposition LU (on a des matrices diagonales). D'après le cours, on serait en compléxité de 8(M+2) soit 8M + o(M).
Malheureusement, le temps nous a manqué pour parvenir jusqu'à l'implémentation sur python. Il s'agit ici que de la théorie !

5 Conclusion

Même si nous n'avons pas pu terminer la dernière question, ce projet nous a permis de découvrir les différentes méthode permettant d'approcher le prix d'une option à payer pour obtenir une certaine somme à une date T ultérieure. On peut observer que ces méthodes se rejoignent pour la plupart.