Devoir maison de calcul stochastique

Enzo Benbalit,
enzo.benbalit@edu.univ-paris13.fr

8 mai 2023

Table des matières

1 EXERCICE 1

Soit l'équation différentielle stochastique :

$$dS_t = rS_t dt + \sigma S_t dW_t \tag{1}$$

Avec $(W_t)_{t\geqslant 0}$ un mouvent-brownien issu de 0.

1.1 Question 1

Théorème 1.1 (Formule d'Itô)**.** *Soit $(X_t)_{t\geqslant 0}$ un processus d'Itô et f une fonction de classe $\mathscr{C}^2$ de $\mathbb{R}^+ \times \mathbb{R} \to \mathbb{R}$ alors :*

$$f(t, X_t) = f(0, X_0) + \int_0^t \frac{\partial f}{\partial s}(s, X_s)ds + \int_0^t \frac{\partial f}{\partial x}(s, X_s)dX_s + \frac{1}{2}\int_0^t \frac{\partial^2 f}{\partial x^2}(s, X_s)d < X, X >_s$$

$(W_t)_{t\geqslant 0}$ est un $(\mathscr{F}_t)_{t\geqslant 0}$-mouvement brownien, c'est donc un procesus d'Itô. D'après le théorème précedent appliqué à :

$$S_t = f(t, W_t) = S_0 e^{\sigma W_t + (r - \frac{\sigma^2}{2})t} \tag{2}$$

Nous avons :

$$\begin{aligned}
S_t &= S_0 e^{\sigma W_0 w} + \int_0^t S_0(r - \frac{\sigma^2}{2})e^{\sigma W_s + (r-\frac{\sigma^2}{2})s}ds + \int_0^t S_0 \sigma e^{\sigma W_s + (r-\frac{\sigma^2}{2})s}dW_s + \frac{1}{2}\int_0^t S_0\sigma^2 e^{\sigma W_s + (r-\frac{\sigma^2}{2})s}ds \\
&= S_0 e^{\sigma W_0} + \int_0^t rS_0 e^{\sigma W_s + (r-\frac{\sigma^2}{2})s}ds + \int_0^t \sigma S_0 e^{\sigma W_s + (r-\frac{\sigma^2}{2})s}dW_s \\
&= S_0 e^{\sigma W_0} + \int_0^t rS_s ds + \int_0^t \sigma S_s dW_s
\end{aligned}$$

On a donc $dS_t = rS_t dt + \sigma S_t dW_t$, $\quad S_t$ est donc bien solution de (1).

1.2 Question 2

Soit,

$$\pi_T = e^{-rT}\mathbb{E}[(S_T - K)_+ \mathbb{1}_{\{Max_{t\in[0,T]} S_T > B\}}]$$

le prix d'une option Barrière, de maturité T et de strike K. Avec S_t définie en (2).

1.2.1 simulerBrownien

Soit $(B_t)_{t\geqslant 0}$ un $(\mathscr{F})_{t\geqslant 0}$-mouvement brouwnien issu de 0. On a :

$$\begin{aligned}
&\forall s \leqslant t, B_t - B_s \perp\!\!\!\perp \mathscr{F}_s \\
&\forall s \leqslant t, B_t - B_s \sim N(0, t - s)
\end{aligned}$$

Soit n$\in \mathbb{N}^*$, $T > 0$, on note h$=\frac{T}{n}$ et $\forall i \in \{1, ..., N\}$,

$$\begin{aligned}
B_{ih} = B_{(i-1)h} + (B_{ih} - B_{(i-1)h}) \quad &avec \quad (B_{ih} - B_{(i-1)h}) \perp\!\!\!\perp \mathscr{F}_{(i-1)h} \\
&et \quad (B_{ih} - B_{(i-1)h}) \sim N(0, h)
\end{aligned}$$

Notons $N_h = B_{ih} - B_{(i-1)h} \sim N(0, h)$, on a $B_{ih} = B_{(i-1)h} + N_h, \forall i \in \{1, ..., n\}$

Remarque 1.2. *Une réalisation de notre mouvement brownien à l'instant i se calcul par l'addition de notre mouvement brownien à l'instant $i-1$ et de la réalisation d'une v.a. de loi $N(0,h)$*

Voici le code implémenté pour la simulation du mouvement browien issu de 0 :

```
def simulerBrownien(T,n):
    h=T/n
    W=np.zeros(n+1)
    for i in range(n):
        W[i+1]=npr.normal(loc=0,scale=np.sqrt(h))+W[i] #W[i+1]=N(0,h)+W[i]
    return W
```

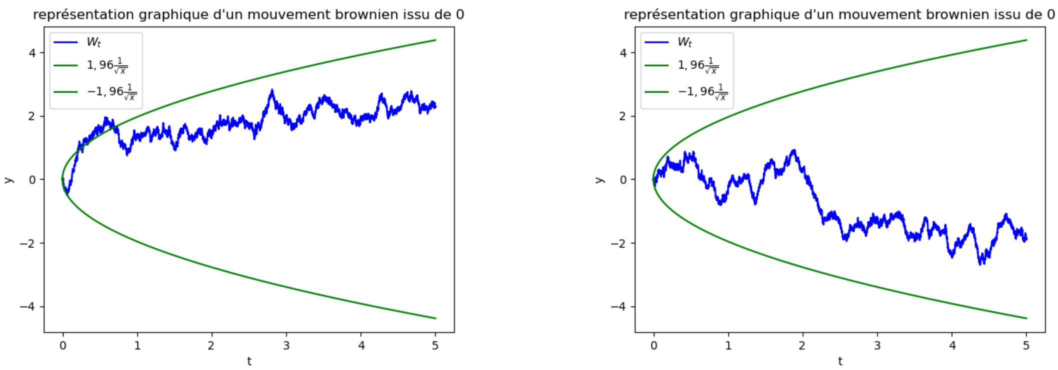


FIGURE 1 – Simulation d'un mouvement brownien issu de 0 dans l'interval $[0,T]$ avec $T=5$ (la simulation à été faite à partir de la fonction simulerBrownien et est représentée graphiquement à l'aide d'un script python).

Observation : *Les mouvements brownien simulés ont tendance à tendre vers l'infini avec une vitesse $\sqrt{n}$, cela s'observe théoriquement grâce à la propriété de scaling.*

Voici le code implémenté pour la simulation du mouvement brownien géométrique :

```
def simulerSt(r,s,T,n,S0):
    #on simule notre mouvement brownien W_t puis on calcul S_t
    #qui est une "fonction" de W_t
    W=q2a.simulerBrownien(T,n)
    h=T/n
    St=np.zeros(n+1)
    for i in range(n+1):
        St[i]=S0*np.exp(s*W[i]+(r-(s*s/2)*i*h))
    return St
```

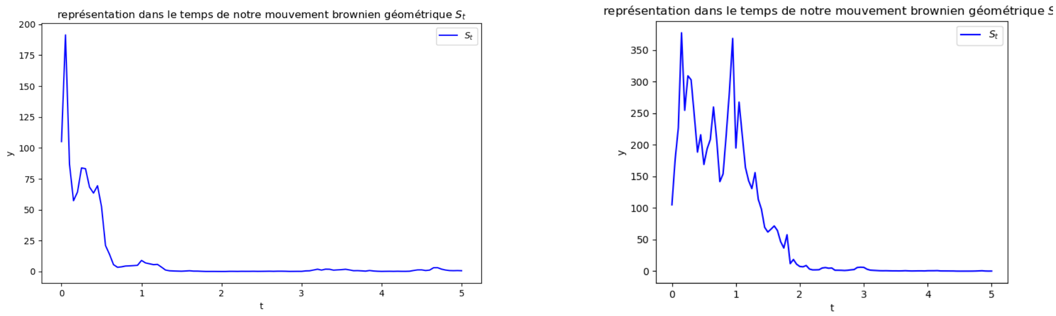


FIGURE 2 – Évolution du mouvement brownien géométrique pour les valeurs $T = 5$, $r = 0.05$, $\sigma = 2$, $S_0 = 100$, $K = 95$, $B = 105$, $n = 100$.

1.2.2 prixBarriere

i) Loi forte des grands nombres :

Théorème 1.3 (Loi Forte des Grands Nombres). *Soit $(X_n)_{n\geqslant 1}$ une suite de v.a intégrable et i.i.d, alors :*

$$S_n = \frac{X_1 + ... + X_n}{n} \xrightarrow[n\to\infty]{} \mathbb{E}[X_1] \quad p.s$$

$\forall T \geqslant 0$, $n \in \mathbb{N}^*$ on note $X_T^{(n)} = e^{-rT}(S_T^{(n)} - K)_+ \mathbb{1}_{\{Max_{t\in[0,T]} S_t^{(n)} > B\}}$, par construction, les $(X_T^{(i)})_{i\geqslant 1}$ sont i.i.d.

• Montrons que $\forall i \in \{1, ..., n\}$, $X_T^{(i)} \in L^2(\mathbb{R}_+, \mathscr{B}(\mathbb{R}_+), \mathbb{P})$:

$$\begin{aligned}\forall n \geqslant 1, \mathbb{E}[(X_T^{(n)})^2] = e^{-rT}\mathbb{E}\Big[((S_T^{(n)} - K)_+ \mathbb{1}_{\{Max_{t\in[0,T]} S_t^{(n)} > B\}})^2\Big] &= e^{-rT}\mathbb{E}\Big[((S_T - K)_+ \mathbb{1}_{\{Max_{t\in[0,T]} S_t > B\}})^2\Big] \quad \text{(par i.i.d)} \\ (e^{-rT} \leqslant 1 \text{ et } \mathbb{1}_{\{Max_{t\in[0,T]} S_t > B\}} \leqslant 1) &\leqslant \mathbb{E}\big[(S_T - K)_+^2\big] = \mathbb{E}\big[(S_T - K)^2 \mathbb{1}_{\{S_T > K\}}\big] \\ &\leqslant \mathbb{E}\big[(S_T - K)^2\big] \\ &= \mathbb{E}\big[S_T^2 - 2KS_T + K^2\big] \\ &= \mathbb{E}\big[S_T^2\big] - 2K\mathbb{E}\big[S_T\big] + K^2 \leqslant \infty\end{aligned}$$

En effet, Soit $X \sim N(m, \mu^2)$ alors $\forall \lambda \in \mathbb{C}$, $\mathbb{E}\big[e^{\lambda X}\big] = e^{\lambda m + \frac{\lambda^2}{2}\mu^2}$ (Transformée de Laplace).

Ainsi, $\forall m \geqslant 0$, $\mathbb{E}\big[S_T^m\big] = S_0\mathbb{E}\big[\big(e^{\sigma W_t + (r - \frac{\sigma^2}{2})t}\big)^m\big] = S_0\mathbb{E}\big[e^{\sigma m W_t + (r - \frac{\sigma^2}{2})tm}\big] = S_0 e^{(r - \frac{\sigma^2}{2})tm + \frac{(\sigma m)^2}{2}t} \leqslant \infty$

On a donc bien montrer que $\forall i \in \{1, ..., n\}$, $X_T^{(i)} \in L^2(\mathbb{R}_+, \mathscr{B}(\mathbb{R}_+), \mathbb{P})$, en faite, $X_T^{(i)} \in L^p$, $p \geqslant 1$, en effet, l'exponentielle de gaussienne admet des moments de tout ordre par la transformée de Laplace. Ainsi, par la loi forte des grands nombres,

$$\forall T \geqslant 0, \quad \bar{\pi_T} = \frac{X_T^{(1)} + ... + X_T^{(n)}}{n} \xrightarrow[n\to\infty]{} e^{-rT}\mathbb{E}\big[(S_T - K)_+ \mathbb{1}_{\{Max_{t\in[0,T]} S_t > B\}}\big] = \pi_T \quad p.s \qquad (3)$$

ii) construction de l'intervalle de confiance asymptotique à 95% :

Théorème 1.4 (Théorème Centrale Limite). *Soit $(X_i)_{i\geqslant 0}$ un suite de v.a i.i.d, de carré intégrable, d'espérance μ et de variance σ^2, alors :*

$$\frac{S_n - n\mu}{\sigma\sqrt{n}} \xrightarrow[n\to\infty]{Loi} N(0, 1)$$

On a montrer dans i) que les $(X_T^{(i)})_{i\geqslant 1} \in L^2(\mathbb{R}_+, \mathscr{B}(\mathbb{R}_+), \mathbb{P})$ ainsi par le théoreme centrale limite,

$$\frac{\sum_{i=1}^n X_T^{(i)} - n\mu}{\sigma\sqrt{n}} \xrightarrow[n\to\infty]{Loi} N(0,1)$$

Or dans notre cas σ^2 est inconnue, on l'approche donc en utilisant la loi des grands nombres par :

$$\bar{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(X_T^{(i)} - \bar{X_T^{(n)}})^2 \quad \text{avec } \bar{X_n} = \sum_{i=1}^{n} X_i$$

Ainsi, un interval de confiance pour la moyenne au seuil $\alpha = 0.05$ est :

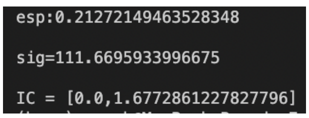
$$IC(95) = \left[\bar{X_T^{(n)}} - 1,96\sqrt{\frac{\bar{\sigma^2}}{n}}; \bar{X_T^{(n)}} - 1,96\sqrt{\frac{\bar{\sigma^2}}{n}}\right]$$

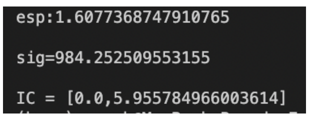
De plus, $\forall i \in \{1, ..., n\}$, $X_T^{(i)} \geqslant 0$ donc $\mathbb{E}[X_T] \geqslant 0$. On peut donc rogner la borne inférieur de l'interval de confiance sans perte de precision, on a donc l'interval de confiance pour la moyenne au seuil $\alpha = 0.05$:

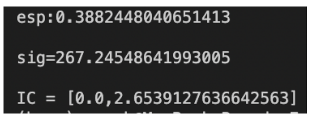
$$IC(95) = \left[\text{Max}\left(\bar{X_T^{(n)}} - 1,96\sqrt{\frac{\bar{\sigma^2}}{n}}, 0\right); \bar{X_T^{(n)}} - 1,96\sqrt{\frac{\bar{\sigma^2}}{n}}\right]$$

Voici le code implémenté de *prixBarriere* ainsi que quelques résultat pour $\bar{\pi_T}$, $\bar{\sigma^2}$ et $IC(95)$ pour $n = 100$:

```python
def simulerXTseed(S_t,r,s,S0,K,B,T):
# permets de simuler une realisation de X_T
    ST=0
    if max(S_t[-1]-K,0.)!=0 and max(S_t)>B:
        ST=np.exp(-r*T)*(S_t[-1]-K) #S_t[-1] donne le dernier elements
    return ST

def prixBarriere(r,s,S0,K,B,n,T):
    #on simule piT et sig2 par la methode de monte-carlo
    #on a piT l'esperance empirique et sig2 la variance empirique
    piT=0
    sig2=0
    ST=np.zeros(n)
    S_t=np.zeros(n)
    for i in range(n):
        S_t=simulerSt(r,s,T,n,S0)
        ST[i]=simulerXTseed(S_t,r,s,S0,K,B,T) #on garde en memoire chaque X_t
        piT=piT+ST[i]                         #afin de calculer sig2 avec le
    piT=piT/n                                 #meme n-echantillon de X_t

    for i in range(n):
        sig2=sig2+(ST[i]-piT)*(ST[i]-piT)
    sig2=sig2/n

    IC=[piT-1.96*np.sqrt(sig2/n),piT+1.96*np.sqrt(sig2/n)]
    IC[0]=max(0.,IC[0]) #etant donner que X_t>=0, on peut rogner l'IC
    return piT,sig2,IC  #sans perte de precision !
```







Observation : *pour la valeur $n = 100$ et $T = 5$, la condition pour ne pas être nulle dans l'éspérance à chaque étape est très contraignante. En effet, comme la figure (2) le montre, S_t converge rapidement vers 0. Cependant, lorsque la condition de positivité est validée, le caractère exponnentielle de S_t fait que la valeur simulée est plutôt grande. Pour ce qui est de $\bar{\sigma^2}$, la valeur est grande car les $ST[i]$ (dans l'algorithme) sont en grande partie nuls, les valeurs de $\bar{\pi_T}$ étant proche de 0, l'écart-type est grand. Ainsi leurs sommes au carré est d'autant plus grande.*

1.2.3 prixBarriereMaturite

Soit $N \in \mathbb{N}$, $T > 0$ et $i \in \{1, ..., N\}$. Soit t_i une discrétisation uniforme de $]0, T]$ t.q : $t_i = i\frac{T}{N}$, $i \in \{1, ..., N\}$. D'après (3),

$$\forall T \geqslant 0, \quad \bar{\pi_T} = e^{-rT}\frac{X_T^{(1)} + ... + X_T^{(n)}}{n} \xrightarrow[n \to \infty]{} e^{-rT}\mathbb{E}\big[(S_T - K)_+ \mathbb{1}_{\{Max_{t\in[0,T]}S_t > B\}}\big] = \pi_T \quad p.s$$

$$donc \quad \forall i \in \{1, ..., n\}, \quad \bar{\pi_{t_i}} = e^{-rt_i}\frac{X_{t_i}^{(1)} + ... + X_{t_i}^{(n)}}{n} \xrightarrow[n \to \infty]{} e^{-rt_i}\mathbb{E}\big[(S_{t_i} - K)_+ \mathbb{1}_{\{Max_{t\in[0,t_i]}S_t > B\}}\big] = \pi_{t_i} \quad p.s$$

Ainsi, Pour simuler la liste $\pi_{\frac{T}{N}}, \pi_{2\frac{T}{N}}, ..., \pi_T$ il suffit de simuler chaque π_{t_i} indépendamenent de la valeur de $\pi_{t_{i-1}}$.

Voici le code implémenté pour $prixBarriereMaturite$ ainsi que le graphe de $t \to \pi_t$:

```
def prixBarriereMaturite(r,s,S0,K,B,n,T):
    #on calcul a chaque etape le nouveau prix barriere pour le
    #temps (i+1)*h independament des precedents
    h=T/n
    pi=np.zeros(n)
    for i in range(n):
        A=prixBarriere(r,s,S0,K,B,n,(i+1)*h)
        pi[i]=A[0] #le premier element de A est le prix barriere
    return pi
```

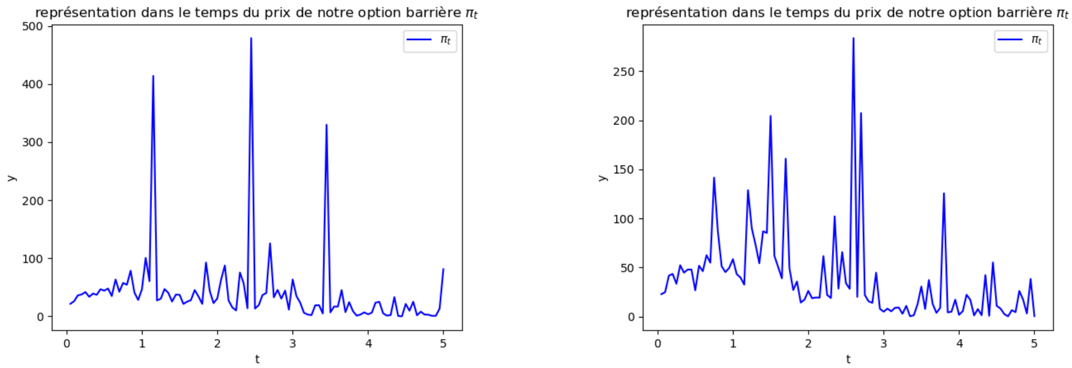


FIGURE 3 – Représentation dans le temps du prix $t \to \pi_t$ de notre option barrière pour $t \in]0, 5]$.

Observation : *Aucune tendance ne semble se dégager.*

1.2.4 Réduction de variance

Montrons que $(-W_t)_{t\geqslant 0}$ est bien un mouvement brownien lorsque $(W_t)_{t\geqslant 0}$ est un mouvement brownien :

i) Soit $t, s > 0$, $t > s$, on a : $-W_t + W_s = -(W_t - W_s) = -N_{t-s} = N_{t-s}$
Avec $N_{t-s} \sim N(0, t-s)$. En effet, comme W_t est un mouvement brownien, $W_t - W_s \sim N(0, t-s)$ et $\mathbb{V}ar(-(W_t - W_s)) = \mathbb{V}ar(W_t - W_s) = t - s$.

ii) Soit $t, s > r > 0$, on a :

$$\begin{aligned} Cov(-W_t + W_s, W_r) &= -Cov(W_t, W_r) + Cov(W_s, W_r) \\ &= s \wedge r - t \wedge r \\ &= r - r = 0. \end{aligned}$$

Ainsi, $(-W_t)t_{t\geqslant 0}$ est bien un mouvement brownien.

Notons $\varphi(x) = -x$, une fonction qui préserve la loi du mouvement brownien, alors par la méthode de réduction de variance par la variable antithétique appliqué à $h(x) = e^{-rT}(S_T(x) - K)_+ \mathbb{1}_{\{Max_{t \in [0,T]} S_t(x) > B\}}$ et tel que $S_t(x) = S_0 e^{\sigma x + (r - \frac{\sigma^2}{2})t}$, on a :

$$\begin{aligned}
\mathbb{E}[h(W_t)] &= \frac{\mathbb{E}[h(W_t)] + \mathbb{E}[h(W_t)]}{2} \\
&= \frac{\mathbb{E}[h(W_t)] + \mathbb{E}[h(\varphi^{-1}(W_t))]}{2} \\
&= \frac{\mathbb{E}[h(W_t)] + \mathbb{E}[h(-W_t)]}{2}
\end{aligned}$$

Cette transformation réduit bien la variance. En effet, on a :

$$\begin{aligned}
\mathbb{V}ar\left(\frac{h(W_t) + h(-W_t)}{2}\right) &= \frac{1}{4}\Big(\mathbb{V}ar(h(W_t)) + \mathbb{V}ar(h(-W_t)) + 2Cov(h(W_t), h(-W_t))\Big) \\
&= \frac{1}{2}\Big(2\mathbb{V}ar(h(W_t)) + 2Cov(h(W_t), h(-W_t))\Big) \text{ (car } h(W_t) \text{ et } h(-W_t) \text{ ont même loi)} \\
&\overset{(Holder)}{\leqslant} \frac{1}{4}\Big(2\mathbb{V}ar(h(W_t)) + 2\sqrt{\mathbb{V}ar(h(W_t))\mathbb{V}ar(h(-W_t))}\Big) \\
&\leqslant \frac{4\mathbb{V}ar(h(W_t))}{4} = \mathbb{V}ar(h(W_t))
\end{aligned}$$

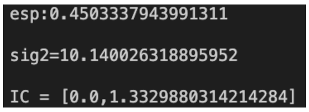
La variance par la méthode de la variable antithétique est théoriquement plus faible que la variance normale.

Objectif : *l'objectif de la méthode est de pouvoir utiliser dans les calculs deux fois plus de variable que celles simulées et de pouvoir affiner les calculs et l'intervalle de confiance pour le même nombre de simulation.*

Voici le code implémenté de *prixBarriereREDUCTVAR* ainsi que quelques résultat pour $\bar{\pi_T}$, $\bar{\sigma}^2$ et $IC(95)$ pour $n_1 = \frac{n}{2} = 50$.

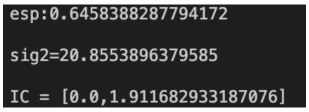
```python
def prixBarriereREDUCTVAR(r,s,S0,K,B,n,T):
    #on commence par simuler un n-echantillon de loi W et l'on calcul
    #S_t(W_t) et S_t(-W_t) afin de pouvoir faire la reduction de variance
    piT=0
    sig2=0
    ST1=np.zeros(n)
    S_T1=np.zeros(n)
    ST2=np.zeros(n)
    S_T2=np.zeros(n)

    for i in range(n):
        W=q2a.simulerBrownien(T,n)
        ST1=simulerStSeed(W,r,s,T,S0) #S_t(W_t)
        ST2=simulerStSeed(-W,r,s,T,S0)#S_t(-W_t)
        S_T1[i]=q2bc.simulerXTseed(ST1,r,s,S0,K,B,T) #calcul de X_t pour W_t
        S_T2[i]=q2bc.simulerXTseed(ST2,r,s,S0,K,B,T) #calcul de X_t pour -W_t
        piT=piT+(S_T1[i]+S_T2[i])/2 #formule de reduction de variance
    piT=piT/n

    for i in range(n):
        sig2=sig2+(((S_T1[i]+S_T2[i])/2)-piT)**2
    sig2=sig2/(n-1)

    IC=[piT-1.96*np.sqrt(sig2/n),piT+1.96*np.sqrt(sig2/n)]
    IC[0]=max(0.,IC[0]) #ici aussi l'IC peut etre rogner sans perte de precision
    return piT,sig2,IC
```

```python
def simulerStSeed(W,r,s,T,S0):
    #cette fonction donne la valeur de S_t pour un W_t
    #bien precis
    n=len(W)
    h=T/(n-1)
    St=np.zeros(n)
    for i in range(n):
        St[i]=S0*np.exp(s*W[i]+(r-(s*s/2)*i*h))
    return St
```


esp:0.4503337943991311

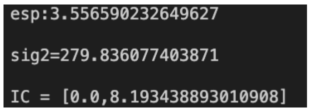
sig2=10.140026318895952

IC = [0.0,1.3329880314214284]


esp:0.6458388287794172

sig2=20.8553896379585

IC = [0.0,1.911682933187076]


esp:3.556590232649627

sig2=279.836077403871

IC = [0.0,8.193438893010908]

__Observation :__ *comme observé théoriquement, la variance calculé par la méthode de la variable antithétique est en moyenne moins élevé que la variance observé par la méthode classique.*

1.3 Option barrière asiatique

Soit,

$$\bar{\pi}_T = e^{-rT}\mathbb{E}\left[\left(\frac{1}{T}\int_0^T S_u du - K\right)_+ \mathbb{1}_{\{max_{t\in[0,T]}S_t > B\}}\right]$$

Le prix d'une option barrière asiatique.

Par l'approximation de l'intégralle par la méthode des trapèzes, on trouve, pour $(a_i)_{0\leqslant i\leqslant N_2}$ une discrétisation uniforme de $[0,T]$ t.q $\forall i \in \{0,...,N_2\}$, $a_i = ih_2 = i\frac{T}{N_2}$,

$$\begin{aligned}\int_0^T S_u du &= \sum_{i=0}^{N_2-1}\int_{a_i}^{a_{i+1}} S_u du \\ &\approx \sum_{i=0}^{N_2-1}(a_{i+1}-a_i)\frac{S_{a_i}+S_{a_{i+1}}}{2} = h_2\sum_{i=0}^{N_2-1}\frac{S_{a_i}+S_{a_{i+1}}}{2}\end{aligned}$$

Ainsi, $\forall T \geqslant 0$,

$$\tilde{\pi}_T^{(N_2)} = e^{-rT}\mathbb{E}\left[\left(\frac{h_2}{T}\sum_{k=0}^{N_2-1}\frac{S_{a_i}+S_{a_{i+1}}}{2} - K\right)_+ \mathbb{1}_{\{\underset{i\in\{0,...,N_2\}}{\text{Max}} S_{a_i} > B\}}\right] \xrightarrow[N_2\to\infty]{} e^{-rT}\mathbb{E}\left[\left(\frac{1}{T}\int_0^T S_u du - K\right)_+ \mathbb{1}_{\{\underset{t\in[0,T]}{\text{Max}} S_t > B\}}\right] = \bar{\pi}_T$$

__Mise en place de l'algorithme :__ *On simule N_1 N_2-échantillon de notre mouvement brownien géométrique afin d'approcher par la loi forte des grands nombres le prix de l'option barrière asiatique pour pour tout temps $T > 0$.*

Remarque 1.5. *Comme pour prixBarriereMaturite, il n'est pas nécessaire de calculer $\bar{\pi}_{b_i}$ avec $\bar{\pi}_{b_{i-1}}$, $b_i = i\frac{T}{N_3}$, $i \in \{1,...,N_3\}$.*

Voici le code implémenté pour l'approximation du prix de l'option barrière asiatique à un temps T fixé :

```
def int_trapeze(St,T):
    #calcul par la methode des trapezes l'approximation
    #de l'integrale de St dans [0,T]
    N2=np.size(St)
    h2=T/N2
    I=0
    for i in range(N2-1):
        I=I+(St[i]+St[i+1])
    I=I*(h2/2)
    return I

def XTseed_int(I,St,T,K,B,r):
    #calcul l'interieur de l'esperance
    #si la condition du if n'est pas respecter alors Xt=0
    Xt=0
    if max((1/T)*I-K,0)!=0 and max(St)>B:
        Xt=np.exp(-r*T)*((1/T)*I-K)
    return Xt
```

```
def prixBarriereAsiatique(r,s,S0,K,B,N1,N2,T):
    #calcul du prix barriere de l'option asiatique pour un temps fixe
    #par la methode de monte-carlo
    piT=0
    for i in range(N1):
        #a chaque etape on calcul un nouvel St et son integral approche par
        #la methode des trapezes, puis on calcule Xt
        St=q2bc.simulerSt(r,s,T,N2,S0)
        I=int_trapeze(St,T)
        Xt=XTseed_int(I,St,T,K,B,r)
        piT=piT+Xt
    piT=piT/N1
    return piT
```

Voici le code implémenté pour obtenir $\bar{\pi_{b_i}}$, $i \in \{1, ..., N_3\}$:

```
def prixBarriereAsiatiqueMaturite(r,s,S0,K,B,N1,N2,T,N3):
    #a chaque etape on calcul \bar{\pi_{b_i}} independament des valeurs precedentes
    h=T/N3
    PB=np.zeros(N3)
    for i in range(N3):
        PB[i]=prixBarriereAsiatique(r,s,S0,K,B,N1,N2,(i+1)*h)
    return PB
```

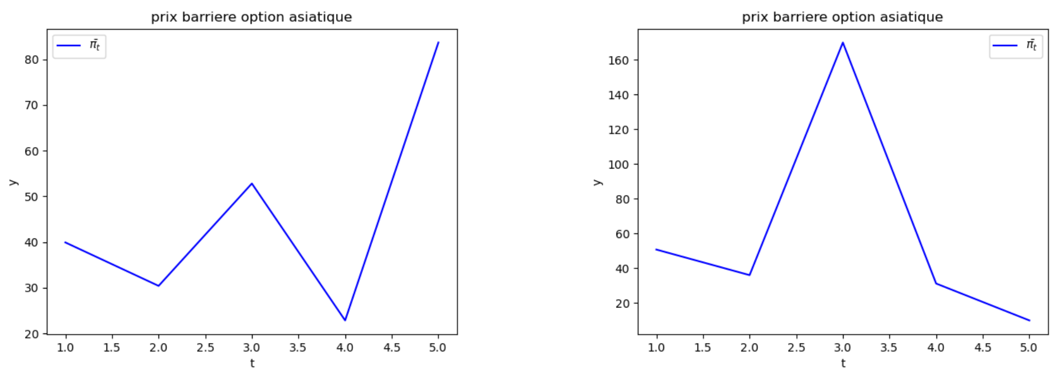


FIGURE 4 – Représentation dans le temps du prix de notre option barrière asiatique $\bar{\pi}_t$ avec $T = 5$ et $b_i = i$, $i \in \{1, ..., 5\}$.

2 EXERCICE 2 : Modèle de Cox, Ross & Rubinstein

2.1 introductions des différentes variables et outils utile

Dans cette section, nous nous efforcerons de définir l'ensemble des variables du modèle de Cox, Ross & Rubinstein :

- S_0 : valeur de l'actif risque à l'instant 0.
- r : taux d'intérêt sans risque.
- σ : volatilité de l'actif risqué.
- K : prix d'exercice de l'option européenne.
- N : le nombre de période.
- T : le temps qui sépare la date de signature du contrat et la date d'exercice du Call exprimé en année (ici $T = 1$).
- u : scalaire qui caractérise un mouvement vers le haut de l'actif risqué de volatilité σ,
$$u = e^{\sigma\sqrt{\Delta_t}} \ , \ \Delta_t = \frac{T}{N}$$
- d : scalaire qui caractérise un mouvement vers le bas de l'actif risqué de volatilité σ,
$$d = e^{-\sigma\sqrt{\Delta_t}}$$
- $(S_n^{(0)})_{n\in[\![0,N]\!]}$: l'actif sans risque de randement certain r sur une période,
$$S_n^{(0)} = (1+r)^n \ , \ \forall n \in [\![0,N]\!]$$
- S : l'actif risqué de prix S_n à l'intant $n \in [\![0,N]\!]$,
$$S_n = \begin{cases} uS_{n-1} \ , \text{ si l'actif est monté} \\ dS_{n-1} \ , \text{ si l'actif est descendu} \end{cases}$$
- $T_n = \frac{S_n}{S_{n-1}}$, $n \in [\![1,N]\!]$: la variable aléatoire i.i.d qui décris si l'actif risqué est monté ou descendu,
$$\mathbb{P}(T_n = u) = p = 1 - \mathbb{P}(T_n = d) \ , \ \forall n \in [\![0,N]\!]$$
- C_n (resp. P_n) : la valeur à l'instant $n \in [\![0,N]\!]$ d'un Call (resp. Put) européen sur unité d'actif risqué au prix d'exercice K et d'échéance N, la relation Call-Put nous donne :
$$C_n - P_n = S_n - K(1+r)^{-(N-n)} \ , \ \forall n \in [\![0,N]\!] \tag{4}$$
- $H_n^{(0)}$: la quantité d'actifs non risqués à détenir à l'instant n pour avoir la couverture appropriée pour le Call.
- H_n : la quantité d'actifs risqués à détenir à l'instant n pour avoir la couverture appropriée pour le Call.
- $J_n^{(0)}$: la quantité d'actifs non risqués à détenir à l'instant n pour avoir la couverture appropriée pour le Put.
- J_n : la quantité d'actifs risqués à détenir à l'instant n pour avoir la couverture appropriée pour le Put.

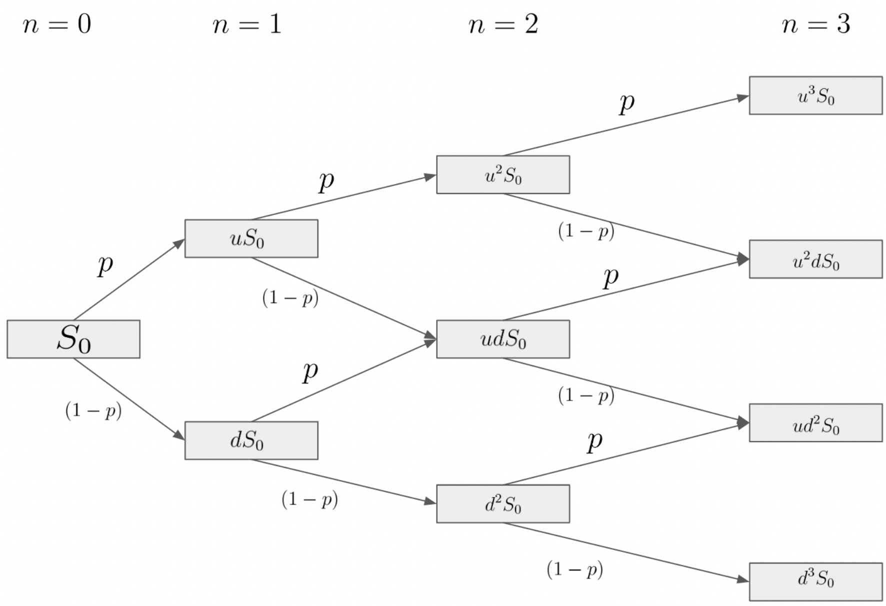


FIGURE 5 – Arbre représentant l'évolution de notre actif risqué pour les périodes 0, 1, 2 et 3 par la méthode de Cox, Ross & Rubinstein.

2.2 Le modèle

Le modèle de Cox, Ross & Rubinstein (CRR) est un modèle discret pour l'evaluation et la dynamique de sous-jacent. Ce modèle, aussi appelé "modèle binomial" peut se représenté à l'aide d'un arbre de probabilité (c.f (5)). Ce modèle est une version discrétisé du modèle de Black-Scholes qui suppose qu'à chaque instant, l'actif risqué S_n monte (resp. descent) avec une probabilité p (resp. $(1-p)$) par un facteur u (resp. d). À chaque instant, la probabilité de monté ou de descendre est supposé être la même et indépendante des précedentes variations du prix de l'actif. De plus, le modèle de CRR suppose que le produit d'un mouvement vers le haut par un mouvement vers le bas est 1. En effet, si le prix augmente puis baisse (ou baisse puis augmente), alors il retourne à son état initial ce qui nous donne $u \times d = 1$.

Dans la suite de cette partie nous allons déterminer la probabilité qu'a l'actif de monter ou descendre, la fonction qui caractérise la valeur d'un Call européen sur une unité d'actif risqué puis la quantité d'actifs risqués et non-risqués à détenir afin d'avoir la couverture appropriée.

2.2.1 probabilité qu'à l'actif de monté ou descendre et viabilité du marché.

i) Montrons que $\tilde{S_n}$ est une $(\mathbb{P}, \mathscr{F}_n)$-martingale ssi $\mathbb{E}[T_{n+1}|\mathscr{F}_n] = 1 + r$.

Soit, $\tilde{S_n} = \frac{S_n}{(1+r)^n}$, $\tilde{S_n}$ est une $(\mathbb{P}, \mathscr{F}_n)$-martingale ssi $\mathbb{E}\left[\tilde{S_{n+1}}|\mathscr{F}_n\right] = \tilde{S_n}$, ainsi on veut :

$$\begin{aligned} &\mathbb{E}\left[\frac{\tilde{S_{n+1}}}{\tilde{S_n}}|\mathscr{F}_n\right] = 1 \\ \Leftrightarrow &\mathbb{E}\left[\frac{S_{n+1}}{S_n}\frac{1}{1+r}|\mathscr{F}_n\right] = 1 \\ \Leftrightarrow &\mathbb{E}\left[\frac{T_{n+1}}{1+r}|\mathscr{F}_n\right] = 1 \\ \Leftrightarrow &\mathbb{E}\left[T_{n+1}|\mathscr{F}_n\right] = 1 + r \end{aligned}$$

ii) Montrons que le marché est viable si $d < 1 + r < u$

Le marché est viable ssi il existe une proba risque neutre $\mathbb{P}^*$ tq $(\tilde{S_n})_{n \in [\![0,N]\!]}$ est une $\mathbb{P}^*$-martingale. On a $\tilde{S_n}$ une martingale ssi $\mathbb{E}[T_{n+1}] = 1 + r$, $\forall n \in [\![0, N]\!]$.
En particulier, on a $\mathbb{E}^*[T_{n+1}] = 1 + r$ avec $\mathbb{E}^*[T_{n+1}] = u\mathbb{P}^*(T_{n+1} = u) + d\mathbb{P}^*(T_{n+1} = d$.
Donc $\tilde{S_n}$ est un martingale si :

$$\begin{aligned} 1 + r &< u[\mathbb{P}^*(T_{n+1} = u) + \mathbb{P}^*(T_{n+1} = d)] \\ 1 + r &> d[\mathbb{P}^*(T_{n+1} = u) + \mathbb{P}^*(T_{n+1} = d)] \end{aligned}$$

Le marché est donc viable pour $u > 1 + r > d$.

iii) Determinons p, la probabilité que l'actif risqué monte.

Afin de déterminer p, Nous allons montrer :

$$\tilde{S_n} \text{ est une } (\mathbb{P}, \mathscr{F}_n)\text{-martingale } \Leftrightarrow \text{ Les } (T_i)_{1 \leqslant i \leqslant N} \text{ sont i.i.d et } \mathbb{P}(T_i = u) = p = 1 - \mathbb{P}(T_i = d)$$

$\Rightarrow$: Soit $\tilde{S_n}$ une $(\mathbb{P}, \mathscr{F}_n)$-martingale, on sait que $\forall n \in [\![0, N-1]\!]$:

$$\begin{aligned} \mathbb{E}\Big[\tilde{S_{n+1}}|\mathscr{F}_n\Big] = \tilde{S_n} \Leftrightarrow \mathbb{E}\Big[T_{n+1}|\mathscr{F}_n\Big] = 1 + r \\ \text{Et,} \quad \mathbb{E}[\mathbb{1}_{\{T_{n+1}=u\}}|\mathscr{F}_n] + \mathbb{E}[\mathbb{1}_{\{T_{n+1}=d\}}|\mathscr{F}_n] = 1 \end{aligned}$$

On a donc :

$$\begin{aligned} \mathbb{E}[T_{n+1}|\mathscr{F}_n] = 1 + r &\Leftrightarrow u\mathbb{E}[\mathbb{1}_{\{T_{n+1}=u\}}|\mathscr{F}_n] + d\mathbb{E}[\mathbb{1}_{\{T_{n+1}=d\}}|\mathscr{F}_n] \\ &\Leftrightarrow u\alpha_n + d\beta_n = 1 + r \text{ avec } \alpha_n + \beta_n = 1 \end{aligned}$$

Ainsi on a le systeme suivant,

$$\begin{cases} u\alpha_n + d\beta_n = 1 + r \\ \alpha_n + \beta_n = 1 \end{cases} \Leftrightarrow \begin{cases} u\alpha_n + d(1 - \alpha_n) = 1 + r \\ \beta_n = 1 - \alpha_n \end{cases} \Leftrightarrow \begin{cases} \alpha_n = \dfrac{1 + r - d}{u - d} \\ \beta_n = 1 - \alpha_n \end{cases}$$

De plus, par proriété de l'espérance conditionnelle, on a $\mathbb{E}\Big[\mathbb{E}[X|\mathscr{G}]\Big] = \mathbb{E}[X]$. Donc, pour $X = \mathbb{1}_{\{T_{n+1}=u\}}$ et $\mathscr{G} = \mathscr{F}_n$, on trouve :

$$\begin{aligned} \mathbb{P}(T_{n+1} = u) = \mathbb{E}[\alpha_n] = p = \frac{1 + r - d}{u - d} \\ \text{Et,} \quad \mathbb{P}(T_{n+1} = d) = \mathbb{E}[\beta_n] = \mathbb{E}[1 - \alpha_n] = 1 - p = \frac{u - 1 - r}{u - d} \end{aligned}$$

Les T_i sont donc identiquement distribuée. L'indépendance viens du fait que $\mathbb{E}[T_{n+1}|\mathscr{F}_n] = \mathbb{E}[T_{n+1}]$, $\forall n \in [\![0, N-1]\!]$.

$\Leftarrow$: Supposons que les T_i soit i.i.d, on a $\forall n \in [\![0, N-1]\!]$:

$$\begin{aligned} \mathbb{E}[T_{n+1}|\mathscr{F}_n] &= \mathbb{E}[T_{n+1}] \quad \text{(Par i.i.d des } T_i\text{)} \\ &= u\mathbb{P}(T_{n+1} = u) + d\mathbb{P}(T_{n+1} = d) \\ &= up + d(1 - p) \\ &= 1 + r \end{aligned}$$

D'après i), ceci montre que $\tilde{S_n}$ est une $(\mathbb{P}, \mathscr{F}_n)$-martingale.

Proposition 2.1. *Le modèle de CRR est un modèle viable et complet*

Démonstration. Le marché est viable par ii). Il est complet si il existe une unique probabilité risque neutre, l'existance viens de la viabilité du marché et est satisfaite pour $d < 1 + r < u$, l'unicité viens du fait que les T_i soient i.i.d. □

iv) Determinons la fonction qui caractérise le prix d'un Call européen.

Soit $C_n = \mathbb{E}^*\left[\frac{(S_N-K)_+}{(1+r)^{N-n}}|\mathscr{F}_n\right]$, avec $\mathbb{P}^*$ (un probabilité equivalente à $\mathbb{P}$) la probabilité sous laquelle l'actifs risqué réactualisé $(\tilde{S_n})_{n\in[\![0,N]\!]}$ est une martingale. On a :

$$C_n = \mathbb{E}^*\left[\frac{(S_N-K)_+}{(1+r)^{N-n}}|\mathscr{F}_n\right]$$
$$(1+r)^{N-n}C_n = \mathbb{E}^*\left[(S_N-K)_+|\mathscr{F}_n\right]$$

Avec, $S_N = S_n \times \frac{S_{n+1}}{S_n} \times ... \times \frac{S_N}{S_{N-1}} = S_n \prod_{i=n+1}^N T_i$.

Notons $Z_{n+1} = \prod_{i=n+1}^N T_i$, on a, $Z_{n+1} \amalg \mathscr{F}_n$ et S_n est $\mathscr{F}_n$-mesurable. Ainsi,

$$(1+r)^{N-n}C_n = \mathbb{E}^*\left[\left(S_n\left(\prod_{i=n+1}^N T_i\right) - K\right)_+|\mathscr{F}_n\right] = \mathbb{E}^*\left[(S_nZ_{n+1}-K)_+\right]$$

Ainsi on trouve :

$$C_n = c(n,S_n) = \frac{1}{(1+r)^{N-n}}\mathbb{E}^*\left[(S_n\left(\prod_{i=n+1}^N T_i\right) - K)_+\right] \tag{5}$$

De plus, on a :

$$\begin{aligned}
(1+r)^{N-n}c(n,x) &= \mathbb{E}^*\left[(x\left(\prod_{i=n+1}^N T_i\right) - K)_+\right] \\
&= \sum_{(t_{n+1},...,t_N)\in\{u,d\}^{(N-n)}} (xt_{n+1}\times ... \times t_N - K)_+\mathbb{P}^*(T_{n+1}=t_{n+1},...,T_N=t_N) \\
&= \sum_{(t_{n+1},...,t_N)\in\{u,d\}^{(N-n)}} (xt_{n+1}\times ... \times t_N - K)_+\mathbb{P}^*(T_{n+1}=t_{n+1})\times ... \times \mathbb{P}^*(T_N=t_N) \\
&= \sum_{(t_0,...,t_{N-n})\in\{u,d\}^{(N-n)}} (xt_1\times ... \times t_{N-n} - K)_+\left(\prod_{i=1}^{N-n}\mathbb{P}^*(T_1=t_i)\right) \\
&= \sum_{k=0}^{N-n}\binom{N-n}{k}(xu^kd^{N-n-k}-K)_+\mathbb{P}^*(T_1=u)^k\mathbb{P}^*(T_1=d)^{N-n-k} \\
&= \sum_{k=0}^{N-n}\binom{N-n}{k}(xu^kd^{N-n-k}-K)_+p^k(1-p)^{N-n-k}
\end{aligned}$$

Ici $k \in [\![0, N-n]\!]$ car "il esiste un chemin qui ne fait que descendre", cela se caractérise par $k = 0$. Ainsi,

$$C_n = c(n,S_n) = \frac{1}{(1+r)^{N-n}}\sum_{k=0}^{N-n}\binom{N-n}{k}(S_nu^kd^{N-n-k}-K)_+p^k(1-p)^{N-n-k} \tag{6}$$

Or, pour n fixé, $(S_n u^k d^{N-n-k} - K)_+$ est une valeur qui ne dépend plus que de k, on cherche donc le plus petit k tq :

$$
\begin{aligned}
& S_n u^k d^{N-n-k} - K > 0 \\
\Leftrightarrow\ & S_n u^k d^{N-n-k} > K \\
\Leftrightarrow\ & ln(S_n u^k d^{N-n-k}) > ln(K) \\
\Leftrightarrow\ & ln(S_n) + kln(u) + (N-n-k)ln(d) > ln(K) \\
\Leftrightarrow\ & k(ln(u) - ln(d)) > ln(K) - ln(S_n) - (N-n)ln(d) \\
\Leftrightarrow\ & kln(\frac{u}{d}) > ln(\frac{K}{S_n d^{N-n}}) \\
\Leftrightarrow & k > \frac{ln(\frac{K}{S_n d^{N-n}})}{ln(\frac{u}{d})}
\end{aligned}
$$

On a donc trouver l'itéré initial $\tilde{k}$ tel que $\forall k \in [\![\tilde{k}, N-n]\!]$, $S_n u^k d^{N-n-k} - K > 0$. Ainsi on trouve une nouvelle formule pour le calcul du prix du Call :

$$C_n = c(n, S_n) = \frac{1}{(1+r)^{N-n}} \sum_{k=\tilde{k}}^{N-n} \binom{N-n}{k} (S_n u^k d^{N-n-k} - K) p^k (1-p)^{N-n-k}$$

avec $\tilde{k}$=inf$\left\{ k \in \mathbb{N},\ k > \frac{ln(\frac{K}{S_n d^{N-n}})}{ln(\frac{u}{d})} \right\}$

Donc,

$$C_n = S_n \sum_{k=\tilde{k}}^{N-n} \binom{N-n}{k} \left(\frac{pu}{1+r}\right)^k \left(\frac{(1-p)d}{1+r}\right)^{N-n-k} - \frac{K}{(1+r)^{N-n}} \sum_{k=\tilde{k}}^{N-n} \binom{N-n}{k} p^k (1-p)^{N-n-k} \quad (7)$$

Et par la relation Call-Put (4), on trouve une formule pour le prix du Put à l'intant n :

$$P_n = p(n, S_n) = c(n, S_n) - S_n + \frac{K}{(1+r)^{N-n}}\ ,\ \forall n \in [\![0, N]\!] \quad (8)$$

v) Déterminons la quantité d'actifs risqués H_n et non risqué $H_n^{(0)}$ à détenir à l'intant n afin d'avoir la couverture appropriée dans le cas du Put et du Call.

Le modèle de CRR étant un modèle complet et viable (par la proposition (2.1)) , il existe une stratégie d'auto-financement qui garantie une couverture appropriée. Cette possibilité d'arbitrage nous donne à chaque instant la relation :

$$H_n^{(0)} S_n^{(0)} + H_n S_n = c(n, S_n)\ \text{, avec } S_n^{(0)} = (1+r)^n \text{ et } n \in [\![1, N]\!] \quad (9)$$

C'est à dire qu'à chaque instant, le prix du Call peut être amorti par l'achat de H_n actif risqué et de $H_n^{(0)}$ actif non-risqué.

Ainsi pour $S_n = uS_{n-1}$ et $S_n = dS_{n-1}$, on a :

$$\begin{cases} H_n^{(0)}(1+r)^n + H_n d S_{n-1} = c(n, dS_{n-1}) \text{ , si l'actif est descendu} \\ H_n^{(0)}(1+r)^n + H_n u S_{n-1} = c(n, uS_{n-1}) \text{ , si l'actif est monté} \end{cases}$$

$$\Leftrightarrow \begin{cases} H_n d S_{n-1} - H_n u S_{n-1} = c(n, dS_{n-1}) - c(n, uS_{n-1}) \\ H_n^{(0)}(1+r)^n + H_n u S_{n-1} = c(n, uS_{n-1}) \end{cases}$$

$$\Leftrightarrow \begin{cases} H_n (d-u) S_{n-1} = c(n, dS_{n-1}) - c(n, uS_{n-1}) \\ H_n^{(0)}(1+r)^n + H_n u S_{n-1} = c(n, uS_{n-1}) \end{cases}$$

$$\Leftrightarrow \begin{cases} H_n = \dfrac{c(n, dS_{n-1}) - c(n, uS_{n-1})}{(d-u)S_{n-1}} \\ H_n^{(0)}(1+r)^n + H_n u S_{n-1} = c(n, uS_{n-1}) \end{cases}$$

$$\Leftrightarrow \begin{cases} H_n = \dfrac{c(n, dS_{n-1}) - c(n, uS_{n-1})}{(d-u)S_{n-1}} \\ H_n^{(0)} = \dfrac{c(n, uS_{n-1})}{(1+r)^n} - \dfrac{H_n u S_{n-1}}{(1+r)^n} \end{cases}$$

$$\Leftrightarrow \begin{cases} H_n = \dfrac{c(n, dS_{n-1}) - c(n, uS_{n-1})}{(d-u)S_{n-1}} \\ H_n^{(0)} = \dfrac{c(n, uS_{n-1})}{(1+r)^n} - \dfrac{u}{d-u} \times \dfrac{c(n, dS_{n-1}) - c(n, uS_{n-1})}{(1+r)^n} \end{cases}$$

$$\Leftrightarrow \begin{cases} H_n = \dfrac{c(n, dS_{n-1}) - c(n, uS_{n-1})}{(d-u)S_{n-1}} \\ H_n^{(0)} = \dfrac{(d-u)c(n, uS_{n-1}) - u(c(n, dS_{n-1}) - c(n, uS_{n-1}))}{(d-u)(1+r)^n} \end{cases}$$

$$\Leftrightarrow \begin{cases} H_n = \dfrac{c(n, dS_{n-1}) - c(n, uS_{n-1})}{(d-u)S_{n-1}} \\ H_n^{(0)} = \dfrac{dc(n, uS_{n-1}) - uc(n, dS_{n-1})}{(d-u)(1+r)^n} \end{cases}$$

À l'instant n, H_n (resp. $H_n^{(0)}$) nous donne la quantité d'actifs risqués (resp. non-risqués) à détenir afin d'avoir la couverture appropriée pour le Call.

De plus, la relation (9) étant vrai pour toute option, elle l'est également pour le Put européen, on à donc les relations :

$$\text{Dans le cas du Call : } \begin{cases} H_n = \dfrac{c(n, dS_{n-1}) - c(n, uS_{n-1})}{(d-u)S_{n-1}} \\ H_n^{(0)} = \dfrac{dc(n, uS_{n-1}) - uc(n, dS_{n-1})}{(d-u)(1+r)^n} \end{cases} \tag{10}$$

$$\text{Dans le cas du Put : } \begin{cases} J_n = \dfrac{p(n, dS_{n-1}) - p(n, uS_{n-1})}{(d-u)S_{n-1}} = H_n - 1 \\ J_n^{(0)} = \dfrac{dp(n, uS_{n-1}) - up(n, dS_{n-1})}{(d-u)(1+r)^n} = H_n^{(0)} + \dfrac{K}{(1+r)^N} \end{cases} \tag{11}$$

Voici les codes implémentés pour la mise en place de notre pricer d'option simple.

```
def pos(a,b):
    return max(a-b,0)

def binom(n,k):
    #retourne le coeficient binomial de k parmis n
    return math.factorial(n)/(math.factorial(n-k)*math.factorial(k))

def Value_Call(N,n,r,p,u,d,Sn,K):
    #donne la valeur d'un Call de prix d'exercice K et d'echeance T=1 a l'instant n
    Cn=0
    for k in range(N-n+1):
        Cn=Cn+binom(N-n,k)*pos(Sn*math.pow(u,k)*math.pow(d,N-n-k),K)*math.pow(p,k)*math.
    pow(1-p,N-n-k)
    return Cn/math.pow(1+r,N-n)
```

```python
def Value_Put(Cn,K,N,r,n,Sn):
    #donne la valeur d'un Put a l'instant n par la relation Call-Put
    return Cn-Sn+K/math.pow(1+r,N-n)

def qteActifs(N,n,r,p,u,d,Sn_1,K):
    #fonction qui donne la quantite d'actif risque et non rique a detenir a l'instant n
    #afin d'avoir la couverture appropriee
    # Hn : la qte d'actif risque pour le Call
    # H0 : la qte d'actif non-risque pour le Call
    # Jn : la qte d'actif risque pour le Put
    # J0 : la qte d'actif non-risque pour le Put
    C_u=Value_Call(N,n,r,p,u,d,u*Sn_1,K)
    C_d=Value_Call(N,n,r,p,u,d,d*Sn_1,K)
    A=math.pow(1+r,n)
    Hn=(C_d-C_u)/((d-u)*Sn_1)
    H0=(d*C_u-u*C_d)/((d-u)*A)
    Jn=Hn-1
    J0=H0+K/math.pow(1+r,N)
    return Hn, H0, Jn, J0
```

Voici le code implémenté pour le pricer d'option simple,

```python
def pricerOption(S_0,r,s,K,N):
    # Cette fonction est un pricer d'option (Call et Put europeen)
    # qui donne aussi a chaque etape la quantitee d'actifs risque et non-risque
    #  a detenir afin de fournir la couverture approprie

    #Ce pricer d'option n'evalue qu'un seul chemin qui est "tire" au hasard.

    #C_n : vecteur qui contient le prix du Call europeen c(n,S_n) a chaque instant
    #P_n : vecteur qui contient le prix du Put a chaque instant
    #Sn : vecteur qui contient la valeur de l'option a chaque instant
    #UD : vecteur de taille N qui contient l'information pour si S_n est monte ou
    descendu
    #     UD[i]=1 => S_n[i+1]=u*S_n[i] , i \in [0,N-1]

    # H_n : la qte d'actif risque pour le Call
    # H0 : la qte d'actif non-risque pour le Call
    # J_n : la qte d'actif risque pour le Put
    # J0 : la qte d'actif non-risque pour le Put

    dt=1/N
    u=np.exp(s*np.sqrt(dt))
    d=np.exp(-s*np.sqrt(dt))
    p=(1+r-d)/(u-d)

    UD=np.zeros(N)
    C_n=np.zeros(N)
    P_n=np.zeros(N)
    Sn=np.zeros(N+1)
    H_n=np.zeros(N)
    H0=np.zeros(N)
    J_n=np.zeros(N)
    J0=np.zeros(N)
    Sn[0]=S_0

    for i in range(N):
        if rnd.rand()<p:
            Sn[i+1]=u*Sn[i]
            UD[i]=1
        else :
            Sn[i+1]=d*Sn[i]
            UD[i]=-1
        C_n[i]=Value_Call(N,i+1,r,p,u,d,Sn[i+1],K)
        P_n[i]=Value_Put(C_n[i],K,N,r,i+1,Sn[i+1])
        A,B,C,D=qteActifs(N,i+1,r,p,u,d,Sn[i],K)
        H_n[i]=A
        H0[i]=B
        J_n[i]=C
        J0[i]=D
    return Sn,C_n,P_n,H_n,H0,J_n,J0,UD,p,u,d
```

Ce pricer d'option simple évalue la valeur du Call, du Put ainsi que les quantités d'actifs risqués et non-risqués à détenir à chaque instant que pour un seul chemin de l'arbre qui est tiré au hasard.

Voici un exemple d'execution d'un script qui créer un fichier texte afin de ranger de façon ordonné les différentes valeur du Call, du Put et des différentes quantités d'actifs risqués et non-risqués à détenier afin d'avoir la couverture appropriée.

```
Fichier qui contient l'évolution à chaque instant des différentes valeurs du Call,
 du Put, de la quantité d'actifs risqués et non risqués à détenir et
 qui décris aussi l'état de l'actif risqué (s'il est monté ou descendu).

s=3.30
r=2.50
K=20.00
S0=100.00
N=4.00

p=0.66
u=5.21
1+r=3.5
d=0.19

S_n : valeur à l'instant i de l'actif risqué
C_n : valeur à l'instant i du Call Européen
P_n : valeur à l'instant i du Put Européen
H0 : la quantité d'actifs non risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Call
H_n : la quantité d'actifs risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Call
J0 : la quantité d'actifs non risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Put
J_n : la quantité d'actifs risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Put


i | S_n | C_n | P_n | H0 | H_n | J0 | J_n
1 | S_0*u^0*d^1=19.20 | 18.84 | 0.11 | -0.10 | 1.00 | 0.03 | -0.00
2 | S_0*u^0*d^2=3.69 | 2.84 | 0.79 | -0.07 | 0.99 | 0.07 | -0.01
3 | S_0*u^1*d^2=19.20 | 15.08 | 1.59 | -0.01 | 0.82 | 0.12 | -0.18
4 | S_0*u^2*d^2=100.00 | 80.00 | 0.00 | -0.02 | 0.83 | 0.11 | -0.17
```

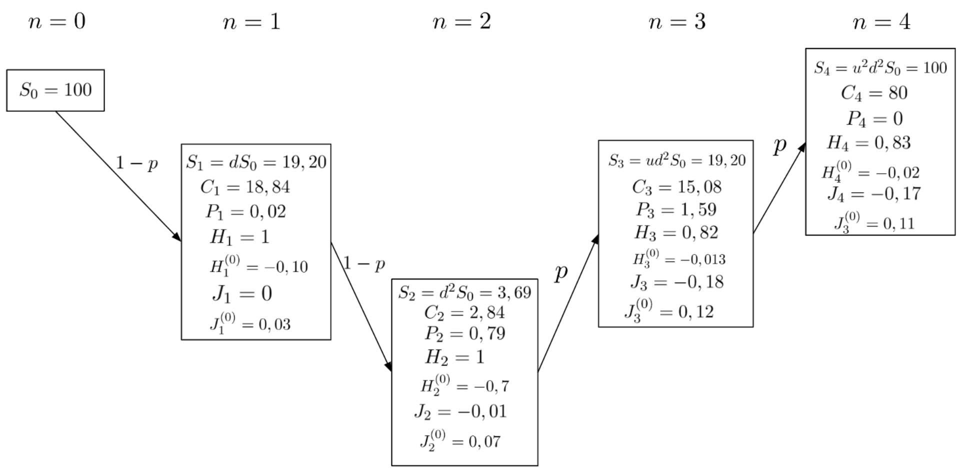


FIGURE 6 – exemple d'execution du code pour $N = 4$, $S_0 = 100$, $\sigma = 3,3$, $r = 2,5$ et $K = 20$ ainsi qu'une représentation graphique à l'aide d'un arbre de probabilité des résultats obtenue.

Commentaire : À la date d'exercice du Call européen, l'actif risqué $S = 100$, le Call $C = 80$ et le Put $P = 0$. Les valeurs de H et $H^{(0)}$ nous donne la stratégie suivante : à la date d'exercice du Call, on achète $0,83$ actifs risqués et on vend $0,02$ actifs non-risqués afin de compenser l'exercice du Call. De plus, les valeurs de J et $J^{(0)}$ nous donne la stratégie suivante : à la date d'exercice du Put, on vend $0,17$ actifs risqués et on achète $0,11$ actifs non-risqués.

Voici deux autres execution de notre fonction *pricerOption* pour différentes valeurs numériques :

```
Fichier qui contient l'évolution à chaque instant des différentes valeurs du Call,
 du Put, de la quantité d'actifs risqués et non risqués à détenir et
 qui décris aussi l'état de l'actif risqué (s'il est monté ou descendu).

s=3.30
r=0.89
K=20.00
S0=55.00
N=13.00

p=0.71
u=2.50
1+r=1.8900000000000001
d=0.40

S_n : valeur à l'instant i de l'actif risqué
C_n : valeur à l'instant i du Call Européen
P_n : valeur à l'instant i du Put Européen
H0 : la quantité d'actifs non risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Call
H_n : la quantité d'actifs risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Call
J0 : la quantité d'actifs non risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Put
J_n : la quantité d'actifs risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Put


i | S_n | C_n | P_n | H0 | H_n | J0 | J_n
1 | S_0*u^1*d^0=137.36 | 137.35 | 0.00 | -0.00 | 1.00 | 0.00 | -0.00
2 | S_0*u^1*d^1=55.00 | 54.98 | 0.00 | -0.01 | 1.00 | 0.00 | -0.00
3 | S_0*u^2*d^1=137.36 | 137.32 | 0.00 | -0.00 | 1.00 | 0.00 | -0.00
4 | S_0*u^3*d^1=343.04 | 342.97 | 0.00 | -0.00 | 1.00 | 0.00 | -0.00
5 | S_0*u^4*d^1=856.71 | 856.59 | 0.00 | -0.01 | 1.00 | 0.00 | -0.00
6 | S_0*u^4*d^2=343.04 | 342.81 | 0.00 | -0.01 | 1.00 | 0.00 | -0.00
7 | S_0*u^4*d^3=137.36 | 136.92 | 0.00 | -0.01 | 1.00 | 0.00 | -0.00
8 | S_0*u^4*d^4=55.00 | 54.19 | 0.02 | -0.00 | 1.00 | 0.00 | -0.00
9 | S_0*u^5*d^4=137.36 | 135.80 | 0.01 | -0.00 | 1.00 | 0.00 | -0.00
10 | S_0*u^6*d^4=343.04 | 340.08 | -0.00 | -0.00 | 1.00 | 0.00 | -0.00
11 | S_0*u^6*d^5=137.36 | 131.76 | -0.00 | -0.01 | 1.00 | 0.00 | 0.00
12 | S_0*u^7*d^5=343.04 | 332.46 | -0.00 | -0.01 | 1.00 | 0.00 | -0.00
13 | S_0*u^8*d^5=856.71 | 836.71 | 0.00 | -0.01 | 1.00 | 0.00 | 0.00
```

```
Fichier qui contient l'évolution à chaque instant des différentes valeurs du Call,
 du Put, de la quantité d'actifs risqués et non risqués à détenir et
 qui décris aussi l'état de l'actif risqué (s'il est monté ou descendu).

s=4.40
r=0.95
K=66.00
S0=86.00
N=10.00

p=0.45
u=4.02
1+r=1.95
d=0.25

S_n : valeur à l'instant i de l'actif risqué
C_n : valeur à l'instant i du Call Européen
P_n : valeur à l'instant i du Put Européen
H0 : la quantité d'actifs non risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Call
H_n : la quantité d'actifs risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Call
J0 : la quantité d'actifs non risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Put
J_n : la quantité d'actifs risqués à détenir à l'instant i pour avoir la couverture appropriée dans le cas d'un Put


i | S_n | C_n | P_n | H0 | H_n | J0 | J_n
1 | S_0*u^0*d^1=21.39 | 21.33 | 0.10 | -0.03 | 1.00 | 0.05 | -0.00
2 | S_0*u^1*d^1=86.00 | 85.83 | 0.14 | -0.02 | 1.00 | 0.06 | -0.00
3 | S_0*u^1*d^2=21.39 | 21.13 | 0.36 | -0.03 | 1.00 | 0.05 | -0.00
4 | S_0*u^1*d^3=5.32 | 4.98 | 0.86 | -0.02 | 1.00 | 0.06 | -0.00
5 | S_0*u^1*d^4=1.32 | 0.96 | 1.97 | -0.01 | 0.97 | 0.07 | -0.03
6 | S_0*u^1*d^5=0.33 | 0.06 | 4.29 | -0.00 | 0.80 | 0.08 | -0.20
7 | S_0*u^1*d^6=0.08 | 0.00 | 8.82 | -0.00 | 0.20 | 0.08 | -0.80
8 | S_0*u^1*d^7=0.02 | 0.00 | 17.34 | -0.00 | -0.00 | 0.08 | -1.00
9 | S_0*u^1*d^8=0.01 | 0.00 | 33.84 | -0.00 | -0.00 | 0.08 | -1.00
10 | S_0*u^2*d^8=0.02 | 0.00 | 65.98 | -0.00 | -0.00 | 0.08 | -1.00
```

3 EXERCICE 3

3.1 Question 1 :

Soit X la v.a qui désigne le nombre de vélos loués par un vendeur un jour donné, par l'énoncé, on a : $X \in L^2(\Omega, \mathbb{P})$ avec $\Omega = \{0, 1, 2, 3, 4\}$ et est de loi :

$$\begin{cases} \mathbb{P}(X = 0) = 0.1 \\ \mathbb{P}(X = 1) = 0.15 \\ \mathbb{P}(X = 2) = 0.3 \\ \mathbb{P}(X = 3) = 0.25 \\ \mathbb{P}(X = 4) = 0.2 \end{cases} \tag{12}$$

Si on note s le nombre de vélos présents en stock le matin à l'ouverture et X la v.a qui caractérise le nombre de vélos loués dans la journée, la fonction profit noté $P_s(X)$ est solution de profits=recettes-coûts , et se définie par

$$\begin{aligned} P_s(X) &= 130X \cdot \mathbb{1}_{\{X \leqslant s\}} - (20s + 30X \cdot \mathbb{1}_{\{X \leqslant s\}}) && (13) \\ &= 100X \cdot \mathbb{1}_{\{X \leqslant s\}} - 20s && (14) \end{aligned}$$

Hypothèse : Pour calculer le profit je suppose que si j'ai un stock de $s = 2$ vélos et que $X > 2$ alors le loueur ne loue en faite pas les X vélos et essuie donc une perte de $20 \times s$, je suppose donc que le loueur ne peut pas louer plus de vélos que de vélos présents en stock. Ainsi, pour $s = 1$, on à :

$$\begin{cases} P_1(0) = -20 \\ P_1(1) = 80 \\ P_1(2) = -20 \\ P_1(3) = -20 \\ P_1(4) = -20 \end{cases}$$

i) trouvons s qui maximise le profit :

On veut trouver le stock s qui maximise $P_s(X)$, on cherche donc :

$$\underset{s \in \Omega}{\text{Max}}\ \mathbb{E}[P_s(X)] \tag{15}$$

Avec $\mathbb{E}[P_s(X)] = 100\mathbb{E}[X \cdot \mathbb{1}_{\{X \leqslant s\}}] - 20s$, on veut donc maximiser la fonctionnelle :

$$\begin{aligned} \mathbb{E}[P_s(X)] &= 100\mathbb{E}[X \cdot \mathbb{1}_{\{X \leqslant s\}}] - 20s \\ &= 100\left(\sum_{i=0}^{4} i \times \mathbb{1}_{\{i \leqslant s\}} \times \mathbb{P}(X = i)\right) - 20s \\ &= 100(0,15 \cdot \mathbb{1}_{\{1 \leqslant s\}} + 2 \times 0,3 \cdot \mathbb{1}_{\{2 \leqslant s\}} + 3 \times 0,25 \cdot \mathbb{1}_{\{3 \leqslant s\}} + 4 \times 0,2 \cdot \mathbb{1}_{\{4 \leqslant s\}}) - 20s = f(s) \end{aligned}$$

On veut donc trouver la solution du problème :

$$\underset{s \in \Omega}{\text{Max}}\ f(s) \tag{16}$$

ii) trouvons s qui minimise le risque :

Le risque est définie par l'écart-type du profit, on le note σ_s et on à :

$$\sigma_s = \sqrt{\mathbb{V}ar(P_s(X))}$$

On veut donc trouver le stock s qui minimise σ_s, on cherche donc :

$$\underset{s \in \Omega}{\text{Min}}\ \sigma_s \tag{17}$$

Or,

$$\begin{aligned}\sqrt{\mathbb{V}ar(P_s(X))} &= 100\sqrt{\mathbb{V}ar(X \cdot \mathbb{1}_{\{X \leqslant s\}})} \\ &= 100\sqrt{(\mathbb{E}[X^2 \cdot \mathbb{1}_{\{X \leqslant s\}}] - \mathbb{E}[X \cdot \mathbb{1}_{\{X \leqslant s\}}]^2)} = g(s)\end{aligned}$$

On veut donc trouver la solution du problème :

$$\underset{s \in \Omega}{\text{Min}}\ g(s) \tag{18}$$

Ici, $s \in \{0, ..., 4\}$ ainsi on peut faire un petit code python qui calculera les différentes valeurs de $f(s)$ et $g(s)$ puis nous ferons le ratio $r = \frac{f(s)}{g(s)}$ et nous prendrons la valeur de s qui minimise $\left|\frac{f(s)}{g(s)} - 1\right|$. ce ration signifie que minimiser g à le même poids que maximiser f.

Pour le caluc‌l de f et g dans le cas (3.1) nous avons les codes python suivant :

```python
def espX(p,s):
    #cette fonction nous donne E[X*indicatrice(X<=s)]
    esp=0
    i=1
    while i<=s:
        esp=esp+i*p[i]
        i=i+1
    return esp

def espX2(p,s):
    #cette fonction nous donne E[X^2*indicatrice(X<=s)]
    esp=0
    i=1
    while i<=s:
        esp=esp+i*i*p[i]
        i=i+1
    return esp

def fg(p,s):
    #cette fonction nous donne la valeur de f(s) et g(s)
    esp=espX(p,s)
    f=100*esp-20*s
    g=100*np.sqrt(espX2(p,s)-esp*esp)
    return f,g

def Calcul_fg(p):
    #Cette fonction retourne des vecteurs de taille 5
    #tq: F[i]=f(i) et G[i]=g(i)
    F=np.zeros(5)
    G=np.zeros(5)
    for s in range(5):
        F[s],G[s]=fg(p,s)

    return F,G
```

Voici les résultats obtenus ainsi que la valeur de s qui maximise le profit tout en minimisant les risques :

```
Voici un tableau contenant l'ensemble des profits réalisable aisni que leurs risques en fonction du stock s

Ici, les probas sont P=[ 0.1 | 0.15 | 0.3 | 0.25 | 0.2 ]


s | f(s) | g(s) | ratio
0 | 0.00 | 0.00 | nan
1 | -5.00 | 35.71 | -0.14
2 | 35.00 | 88.74 | 0.39
3 | 90.00 | 116.19 | 0.77
4 | 150.00 | 122.88 | 1.22


Ainsi la valeur de s qui maximise le profit tout en minimisant les risques est s= 4
```

3.2 Question 2 :

Soit maintenant,

Nombre d'unités louées	0	1	2	3	4
Probabilités	p_0	p_1	p_2	p_3	p_4

— a : les frais d'exploitation.
— b : les frais fixes.
— r : le prix de la location journalière.

On suppose que a, b et $r \geqslant 0$.

On a notre nouvelle fonction profit, construit par le même principe que en (3.1) :

$$P_s(X) = (r - a)X \cdot \mathbb{1}_{\{X \leqslant s\}} - bs \tag{19}$$

3.2.1 les conditions $a < r$ et $0 < b < r - a$

i) la condition $a < r$

Si cette condition n'est pas respecter alors $\forall s \in \Omega$, le profit sera toujours négatif. En effet, soit $r \geqslant a$, on a $P_s(X) = (r-a)X \cdot \mathbb{1}_{\{X \leqslant s\}} - bs \leqslant -\text{Min}(a-r, s) \times (X \cdot \mathbb{1}_{\{X \leqslant s\}} + s)$ avec $X \cdot \mathbb{1}_{\{X \leqslant s\}} + s \geqslant 0$, $r - a \geqslant 0$ et $s \geqslant 0$ donc $P_s(X) = (r-a)X \cdot \mathbb{1}_{\{X \leqslant s\}} - bs \leqslant -\text{Min}(a-r, s) \times (X \cdot \mathbb{1}_{\{X \leqslant s\}} + s) \leqslant 0$.
En d'autres termes, les frais d'exploitation ne peuvent excéder les frais de locations.

ii) la condition $0 < b < r - a$

Cette condition signifie que les frais fixes font toujours parties des coûts d'une entreprise, en effet, d'après wikipédia les frais fixes sont "les dépenses d'une entreprise qui ne dépendent pas de sa production". De plus, la condition $b < r - a$ implique que les frais fixes doivent toujours être inférieur au bénéfice sinon l'entreprise serait la encore en déficit.

3.2.2 les problèmes de Min et Max pour des valeurs de a, b et r qui respecte les condition (3.2.1)

D'après (3.1), nous avons les deux problèmes de maximisation/minimisation :

$$\underset{s \in \Omega}{\text{Max}}\ \mathbb{E}[P_s(X)] \tag{20}$$

$$\underset{s \in \Omega}{\text{Min}}\ \sqrt{\mathbb{V}ar(P_s(X))} \tag{21}$$

Avec,

$$\mathbb{E}[P_s(X)] = (r - a)\mathbb{E}[X \cdot \mathbb{1}_{\{X \leqslant s\}}] - bs = f(s)$$

$$\begin{aligned} \text{Et, } \sqrt{\mathbb{V}ar(P_s(X))} &= (r-a)\sqrt{\mathbb{V}ar(X \cdot \mathbb{1}_{\{X \leqslant 0\}})} \\ &= (r-a)\sqrt{(\mathbb{E}[X^2 \cdot \mathbb{1}_{\{X \leqslant s\}}] - \mathbb{E}[X \cdot \mathbb{1}_{\{X \leqslant s\}}]^2)} = g(s) \end{aligned}$$

Ainsi les problèmes (21) et (20) se réécrivent :

$$\underset{s \in \Omega}{\text{Max}}\ f(s) \tag{22}$$

$$\underset{s \in \Omega}{\text{Min}}\ g(s) \tag{23}$$

Nous allons résoudre ces problèmes avec des codes python et afficher les différentes valeurs ainsi que la valeurs de s qui maximise f tout en minimisant g dans un document texte.

3.2.3 les codes et script python

```python
def fg(a,b,r,p,s):
    #cette fonction retourne la valeur de f(s) et g(s)
    #r : le prix de la location journaliere
    #a : les frais d'exploitation
    #b : les frais fixe
    esp=q1.espX(p,s)
    f=(r-a)*esp-b*s
    g=(r-a)*np.sqrt(q1.espX2(p,s)-esp*esp)
    return f,g

def Calcul_fg(p,a,b,r):
    #Cette fonction retourne des vecteurs de taille 5
    #tq: F[i]=f(i) et G[i]=g(i)
    #r : le prix de la location journaliere
    #a : les frais d'exploitation
    #b : les frais fixe
    F=np.zeros(5)
    G=np.zeros(5)
    for s in range(5):
        F[s],G[s]=fg(a,b,r,p,s)
    return F,G
```

```python
#######################
# #question 1
# P = [0.1, 0.15, 0.3, 0.25, 0.2]
# F,G=q1.Calcul_fg(P)
########################

###########################
#question 2
P=[0.1,0.08,0.5,0.1,0.22]
a=40
b=60
r=220
F,G=q2.Calcul_fg(P,a,b,r)
##########################

n=len(P)
rt=np.zeros(n)
rt[0]=nan
for i in range(1,n):
    rt[i]=F[i]/G[i]
abs_r=[abs(x-1) for x in rt[1:]]

#ecriture dans un fichier texte "result2" du tableau des differents profits
fichier=open("result2.txt","w")
fichier.write("Voici un tableau contenant l'ensemble des profits realisable aisni que
    leurs risques en fonction du stock s\n")
fichier.write("\n")
fichier.write(f"Ici, les probas sont P=[ {P[0]} | {P[1]} | {P[2]} | {P[3]} | {P[4]} ]\n"
    )
fichier.write("\n")
##########################
fichier.write(f"les frais d'exploitation a={a}\n")
fichier.write(f"les frais fixes b={b}\n")
fichier.write(f"le prix d'une location r={r}\n")
#########################
fichier.write("\n")
fichier.write("s | f(s) | g(s) | ratio \n")
for i in range(n):
    fichier.write(f"{i} | {F[i]:.2f} | {G[i]:.2f} | {rt[i]:.2f} \n")
fichier.write("\n")
fichier.write("\n")
fichier.write(f"Ainsi la valeur de s qui maximise le profit tout en minimisant les
    risques est s= {abs_r.index(min(abs_r))+1}\n")
fichier.close()
```

Voici les fichiers texte obtenus :

```
Voici un tableau contenant l'ensemble des profits réalisable aisni que leurs risques en fonction du stock s

Ici, les probas sont P=[ 0.2 | 0.4 | 0.15 | 0.1 | 0.15 ]

les frais d'exploitation a=10
les frais fixes b=5
le prix d'une location r=140

s | f(s) | g(s) | ratio
0 | 0.00 | 0.00 | nan
1 | 47.00 | 63.69 | 0.74
2 | 81.00 | 92.84 | 0.87
3 | 115.00 | 123.33 | 0.93
4 | 188.00 | 171.48 | 1.10


Ainsi la valeur de s qui maximise le profit tout en minimisant les risques est s= 3
```

```
Voici un tableau contenant l'ensemble des profits réalisable aisni que leurs risques en fonction du stock s

Ici, les probas sont P=[ 0.1 | 0.08 | 0.5 | 0.1 | 0.22 ]

les frais d'exploitation a=40
les frais fixes b=60
le prix d'une location r=220

s | f(s) | g(s) | ratio
0 | 0.00 | 0.00 | nan
1 | -45.60 | 48.83 | -0.93
2 | 74.40 | 172.05 | 0.43
3 | 68.40 | 186.68 | 0.37
4 | 166.80 | 212.40 | 0.79


Ainsi la valeur de s qui maximise le profit tout en minimisant les risques est s= 4
```

4 ANNEXE : liens vers les codes utilisés

Voici un lien .git qui contient l'ensemble des codes utilisés dans ce DM :
`https://github.com/enzoben/Benbalit_Enzo_MACS_2_printemps_2023`.