Évaluation des options américaines par simulation Longstaff-Schwartz

Amin Jellali
Hatem Bouguila

Encadré par
M.Mohammed Anis Ben Lasmar

ESPRIT
2018-2019

Contents

1 Introduction

Ce projet est basé sur l'étude de l'article "Valuing american Options by simulation : A simple Least Sqaures Approach" qui se base sur une approche d'évaluation des options américaines par simulation.

Le cadre général de ce projet est l'évaluation du prix d'un produit financier dont le "pay-off" dépend des valeurs futures d'un actif sous-jacent. Donc il s'agit de modéliser l'évolution de la valeur du sous-jacent. Le modèle de Black-Scholes est l'un de ces modèles ; la valeur su sous-jacent suit un mouvement brownien géométrique.

Un des problèmes du "Pricing" des options américaines c'est l'évaluation et la détermination de la date optimale de l'exercice pour s'assurer d'un maximum de profit.

2 Définition des termes

2.1 Option

Une option est un produit dérivé qui établit un contrat entre un acheteur et un vendeur.L'acheteur d'une option a le droit mais non l'obligation d'acheter (Call) ou de vendre(Put) une quantité donnée d'un actif sous-jacent à un prix déterminé à une date prédéterminé ou avant cette date(selon le type de l'option). On distingue deux types des options : L'option américaine qui peut être exécuté à n'importe quel moment avant la date d'échéance et l'option européenne qui ne peut etre exécuté qu'à la date de l'échéance.

2.2 Call

Option d'achat; le droit d'acheter l'actif sous-jacent.

2.3 Put

Option de vente; le droit de vendre l'actif sous-jacent.

2.4 La prime

C'est la somme versée par l'acheteur

2.5 Strike Price

C'est le prix de l'exercice; le prix déterminé à l'avance.

2.6 Date d'exercice

C'est la date d'échéance, après cette date l'option cesse d'exister.

2.7 Mouvement Brownien

On appelle mouvement brownien standard $\mathrm{B} = \mathrm{B}_t$ un processus stochastique à trajectoires continus vérifiant:

- $\mathrm{B}_0 = 0$
- $\forall \mathrm{t}_1, \mathrm{t}_2$ tel que $0 \leqslant \mathrm{t}_1 \leqslant \mathrm{t}_2$; avec $(\mathrm{B}_{\mathrm{t}_2} - \mathrm{B}_{\mathrm{t}_1}) \sim \mathcal{N}(0, \mathrm{t}_2 - \mathrm{t}_1)$
- $\forall \mathrm{K} \geqslant 2$, $0 = \mathrm{t}_0 \leqslant \mathrm{t}_1 \leqslant ... \leqslant \mathrm{t}_K$, les accroissements $(\mathrm{B}_{\mathrm{t}_i} - \mathrm{B}_{\mathrm{t}_{i-1}})$ sont indépendants.

Alors on pourra affirmer que le mouvement Brownien est un processus stochastique gaussien, à accroissement indépendants et stationnaires.
Cela signifie que l'accroissement $(\mathrm{B}_t - \mathrm{B}_s)$ avec $s < t$, est indépendant des processus précédents (B_u) avec $u < s$ et aussi qu'il est une variable aléatoire normale centrée en 0 et de variance $\sigma = t - s$.

L'étude du mouvement Brownien est toujours d'actualité comme est applicable dans plusieurs domaine comme la finance pour notre cas.

3 Formulation du problème du Pricing

3.1 Hypothèse

L'objet de ce projet est de déterminer la valeur d'un "Put" américain ainsi que le meilleur temps d'exercice pour maximiser le gain dans un intervalle de temps fini [0,T], on propose alors de se placer dans un espace probabilisé $(\Omega, \mathcal{F}, \mathbb{R})$ où Ω est l'ensemble de toutes les réalisations possibles du prix du sous-jacent stochastique de 0 et T.
On pose alors le vecteur $S_t = (S_t^1, S_t^2, .., S_t^d)$ où $t < T$ et d est le nombre des trajectoires du prix du sous-jacent à la date t, un processus de Markov a valeur dans $\mathbb{R}^d$,on note que toutes les trajectoires commencent par une valeur déterministe qu'on note S_0 qui représente le prix spot du sous-jacent à la date t=0, On muni l'espace d'une filtration $(\mathcal{F})$ engendrée par le processus S, On suppose encore qu'il existe une mesure martingale $\mathbb{Q}$ et on la note B_t la valeur en t de la monnaie investie a t=0 aux taux sans risque r constant $B_t = e^{rt}$.

Donc le problème de pricing consiste a déterminer :

$$\mathbb{Q}_0 = \sup_{t \in \tau} \mathbb{E}_{\mathbb{Q}}[\frac{P_\tau}{B_\tau}] \qquad (1)$$

où t est un temps d'arrêt qui appartient à l'ensemble $\{t_1, ..., t_N\}$ et $\frac{P_t}{B_t}$ est le payoff à la date t de l'option actualisé a la date 0.

4 Résolution par la méthode LSM

4.1 Simulation des trajectoires

La méthode LSM est basée sur la simulation Monte-Carlo du prix du sous-jacent sous la mesure risque neutre, à fin de le faire nous utiliserons le modéle de Black & Scholes. La dynamique du sous-jacent sous la mesure risque neutre s'écrit :

$$\frac{dS_t}{S_t} = rdt + \sigma dW_t \qquad (2)$$

S_t: est le prix du sous-jacent.
r: est le taux sans risque
σ: est la volatilité du sous-jacent.
W_t: est le mouvement brownien de dimension 1.

$$S_{t_{k+1}} = S_{t_k} \exp^{(r - \frac{\sigma^2}{2})t + \sigma\varepsilon\sqrt{\Delta t}} \qquad (3)\ (\textit{voir annexe})$$

avec ε est une variable aléatoire indépendante qui suit la loi normale centrée réduite.
si nous écrivons l'équation (1) autrement:

$$\mathbb{Q}_0 = sup\ \mathbb{E}_{\mathbb{Q}}[\exp^{-rt}[K - S_t]^+] \qquad (4)$$

S_t: Prix du sous-jacent.
r: Taux sans risque.
t: Temps d'exercice.
K: Prix d'exercice.

ce qui nous ramène a résoudre le problème rétrograde (”valuation framework”)

$$\mathbb{Q}_{t_k} = max\{[K - S_t]^+ \ , \ \mathbb{E}_{\mathbb{Q}}[\exp^{-r\Delta t}\mathbb{Q}_{t_k+1}|\mathcal{F}_{t_k}]\} \qquad (5)$$

avec l'espérance conditionnelle dans l'équation(4) est la valeur estimée des pay-off actualisés à la date t_{k+1} sachant le prix du sous-jacent simulé à la date t_k, l'objectif alors est de déterminer s'il existe un temps d'exercice qui génère le pay-off maximale sur chaque trajectoire pour finalement obtenir le prix.

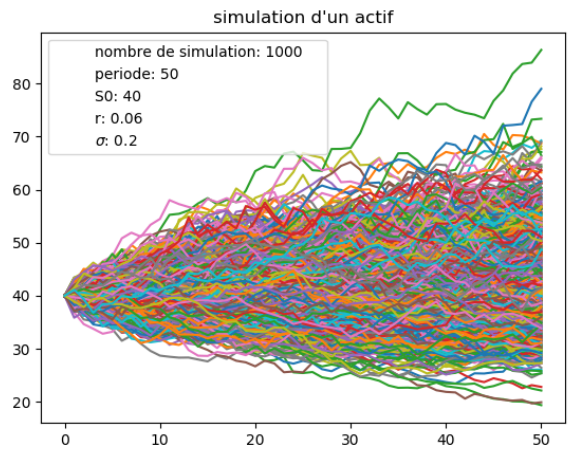

Figure 1: Exemple de génération de trajectoires

4.2 Détermination de la date d'exercice

$t = t_1$	$t = t_2$	..	$t = t_N = T$
$S^1_{t_1}$	$S^2_{t_2}$	..	$S^1_{t_N}$
$S^1_{t_1}$	$S^2_{t_2}$	..	$S^2_{t_N}$
$\vdots$	$\vdots$		$\vdots$
$S^M_{t_1}$	$S^M_{t_2}$	..	$S^M_{t_N}$

En partant du tableau ci-dessus, notre objectif est de construire un autre avec la même taille pour préciser sur chaque trajectoire la date d'exercice de l'option.

Chaque ligne du deuxième tableau sera construite par des 0 et une valeur positive qu'on note C ou seulement des 0 (C : correspond à la valeur de l'exercice réalisé a une date t).

Note :
Une ligne ne peut contenir qu'une seule valeur positive C au maximum; car une option ne peut être exécute qu'une seul fois. Dans l'autre cas on peut avoir une ligne qui contient que des 0 (l'option ne sera pas exerçais).

4.3 Initialisation à la date finale

Partant de la relation suivante ;

$$\mathbb{Q}_{t_N} = [K - S_t]^+ \qquad (6)$$

Sur chaque trajectoire, on met au croisement de la ligne correspondante et la dernière colonne la valeur de l'exercice $\mathbb{Q}_{t_k}$ si on $K > S_t^m$ et 0 si on a $K < S_t^m$; avec m correspond au numéro de la ligne.

4.4 Détermination de la valeur de continuation à une date t_k:

Nous souhaitons remplir la date d'exercice à la date t_k après le remplissage d'une manière rétrograde de la date t_N à la date t_{k+1}.
La méthode se base sur la comparaison entre la valeur de l'exercice à t_k et la valeur de continuation, pour choisir le maximum.

La détermination de la valeur de continuation se base sur la caractérisation de l'espérance conditionnelle.

$$\mathbb{E}[\exp^{-r\Delta t}\mathbb{Q}_{t_k+1}|\mathcal{F}_{t_k}] \qquad (7)$$

En plus on utilise le caractère Markovien qui nous permet que l'espérance conditionnelle s'écrit comme une fonction mesurable de S_{t_k}:

$$\mathbb{E}[.|\mathcal{F}_{t_k}] = \mathbb{E}[.|\mathcal{S}_{t_k}] \qquad (8)$$

Ces deux derniers points (7) et (8), nous permettent d'écrire cette égalité entre l'espérance conditionnelle et une fonction mesurable de $\mathcal{S}_{t_k}$:

$$f(\mathcal{S}_{t_k}) = \mathbb{E}[\exp^{-r\Delta t}\mathbb{Q}_{t_k+1}|\mathcal{F}_{t_k}] \qquad (9)$$

Essayons maintenant de chercher une approximation de la fonction f en minimisant:

$$\sum(\exp^{-r\Delta t}\mathbb{Q}^m_{t_k+1} - f(\mathcal{S}^m_{t_k}))^2 \qquad (10)$$

La fonction f sera écrite sous forme de combinaison linéaire des fonctions de base p_j (dans notre cas nous choisissons après la base de Laguerre)
f s'écrit sous cette forme :

$$f(\mathcal{S}^m_{t_k}) = \sum_{j=1}^{\mathrm{L}} \alpha_{j,k}\mathsf{p}_j(\mathcal{S}^m_{t_k}) \qquad (11)$$

Pour les valeurs $\alpha_{j,k}$, nous cherchons à les trouver en résolvant le problème des moindres carrés.
Si on trouve les coefficients optimaux qu'on note ($\alpha^*_{j,k}$), on utilise l'équation suivante pour avoir la valeur de continuation sur chaque trajectoire (avec m le numéro de la trajectoire) :

$$\mathbb{E}^m[\exp^{-r\Delta t}\mathbb{Q}_{t_k+1}|\mathcal{F}_{t_k}] = \sum_{j=1}^{\mathrm{L}} \alpha^*_{j,k}\mathsf{p}_j(\mathcal{S}^m_{t_k})$$

Note :
Afin de réduire les temps de calculs et augmenter l'efficacité, Longstaff et Schwartz incluent que les trajectoires pour lesquelles l'option est dans la monnaie.

4.5 Méthodologie du boucle rétrograde

Après le remplissage des dates d'exercices à $t = t_N$, nous essayons de compléter les autres dates avec une manière rétrograde.
Pour cela on doit utiliser un ensemble fini des fonctions des base; polynômes de Laguerre dans notre cas.
Supposons dans cette section que L = 3, les expressions des 3 premiers polynômes de Laguerre :

- $L_0(x) = 1$
- $L_1(x) = 1 - x$
- $L_2(x) = \left(\frac{1}{2}\right)(x^2 - 4x + 2)$

Le boucle rétrograde s'effectue de $K = N - 1$ jusqu'à $K = 1$

4.5.1 Première étape

On procède par parcourir les m trajectoires pour récuperer seulement les n_k trajectoires qui sont à la monnaie, ensuite on instance deux vecteurs qu'on note X et Y ayant la même taille n_k

- X contient les $S_{t_k}^m$ avec $k > S_{t_k}$
- Y contient les valeurs actualisés des cash-flows sur chaque trajectoire entre t_{k+1} et t_N

Les valeurs de X et Y sont dans le même ordre.

Prenons un cas pour comprendre comment on procède dans le remplissage des vecteurs X et Y :
Soit m_i une trajectoire à la monnaie, t_{k+k_i} la date sur la trajectoire où on a une valeur d'exercice positive dans le tableau des dates d'exercices alors le remplissage se fait de cette manière :

- X = $S_{t_k}^{m_i}$
- Y = $e^{-r(t_{k+k_i}-t_k)}[K - S_{t_{k+k_i}}]^+$

Note :
Dans le cas où on n'a aucune valeur positive, c'est à dire l'option n'a pas été exercé sur cette trajectoire, alors dans ce cas Y_i reçoit la valeur nulle.

4.5.2 Deuxième étape

En utilisant les fonctions de bases, on construit la matrice de régression M_k en utilisant les composantes de X ; c'est à dire les S_{t_k} où l'option est dans la monnaie.

$$M_k = \begin{pmatrix} P_0(X_1) & P_1(X_1) & P_2(X_1) \\ . & . & . \\ . & . & . \\ . & . & . \\ P_0(X_{n_k}) & P_1(X_{n_k}) & P_2(X_{n_k}) \end{pmatrix} \text{ ; avec } X_1 = S_{t_k}^{m_1} \text{ et } X_{n_k} = S_{t_k}^{m_{n_k}}$$

La régression se fait sur les colonnes de M_k

4.5.3 Troisième étape

Comme énoncé dans l'étape précédente, la régression de Y s'effectue sur les colonnes de M_k , Ce qui nous permet d'approximer l'espérance conditionnelle :

$$\mathbb{E}[Y|X] \simeq \alpha^*_{0,k}P_0(X) + \alpha^*_{1,k}P_1(X) + \alpha^*_{2,k}P_2(X)$$

Cette approximation nous permet de calculer la valeur de continuation pour la i-éme trajectoire :

$$\mathbb{E}[e^{-r\Delta t}\mathbb{Q}_{t_k+1}|\mathcal{F}_{t_k}]^{m_i} = \sum_{j=0}^{2} \alpha^*_{j,k}\mathsf{p}_j(\mathcal{S}^{m_i}_{t_k})$$

4.5.4 Quatrième étape

Après les trois étapes précédentes, nous disposons des valeurs de continuation sur chaque trajectoire à la monnaie, donc nous effectuons une mise à jour sur le tableau des dates d'exercices en suivant la méthodologie suivante:

1er Cas :

Si l'option est hors la monnaie, on met un 0 au croisement de la ligne m et la trajectoire t_k.

2ème Cas :

Si l'option est dans la monnaie, deux scénarios possibles se présentent :

- $\mathbb{E}[e^{-r\Delta t}\mathbb{Q}_{t_k+1}|\mathcal{F}_{t_k}]^{m_i} > [K - S^{m_i}_{t_k}]$; dans cette situation on met un 0 au croisement de m et t_k ,c'est à dire on garde l'option au lieu de l'exercer.
- $\mathbb{E}[e^{-r\Delta t}\mathbb{Q}_{t_k+1}|\mathcal{F}_{t_k}]^{m_i} < [K - S^{m_i}_{t_k}]$; dans cette situation on exerce l'option; c'est à dire on la valeur positive de l'exercice au croisement de m et t_k. En plus, sur la même trajectoire on doit remplacer s'il existe des valeurs positives par un 0 pour $t > t_k$; de cette manière on essaye d'exercer l'option au maximum.

4.6 Cash-flows et calcul de Q_0

En utilisant le tableau des dates de l'exercice, on pourra calculer le prix de l'option qui correspond à la moyenne des cash-flows actualisés sur chaque trajectoire.

$$Q_0 = \frac{\sum_{m=1}^{M} e^{-r\tau_m}(K - S^m_{\tau_m})}{M}$$

Avec τ_m ; représente la date d'exercice de l'option sur la trajectoire m.

4.7 Résolution du problème des moindres carrés

Un problème des moindres carrés s'écrit de la manière suivante:

Soit le système matriciel d'équations $X\alpha = y$

$$\text{X=} \begin{pmatrix} X_{11} & X_{12} & . & . & . & X_{1L} \\ X_{21} & X_{22} & . & . & . & X_{2L} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ X_{M1} & X_{M2} & . & . & . & X_{ML} \end{pmatrix} ; \ \alpha = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ . \\ . \\ . \\ \alpha_L \end{pmatrix} \quad \text{et Y=} \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_M \end{pmatrix}$$

Ce système n'a pas de solution, donc notre objectif est de déterminer le vecteur de coefficients α qui nous approche au mieux à cette égalité.

Essayons donc de déterminer :

$$\alpha^* = \arg inf \sum_{m=1}^{M} (y_i - \sum_{j=1}^{L} X_{mj}\alpha_j)^2 \text{ ; avec } \alpha \in \mathbb{R}^L$$

Ce problème mentionné possède une solution unique lorsque les L colonnes de X sont linéairement indépendantes. Cette solution est donnée par l'égalité suivante :

$$(X^t X)\alpha^* = X^t y$$

5 Algorithme

5.1 Schéma Représentatif

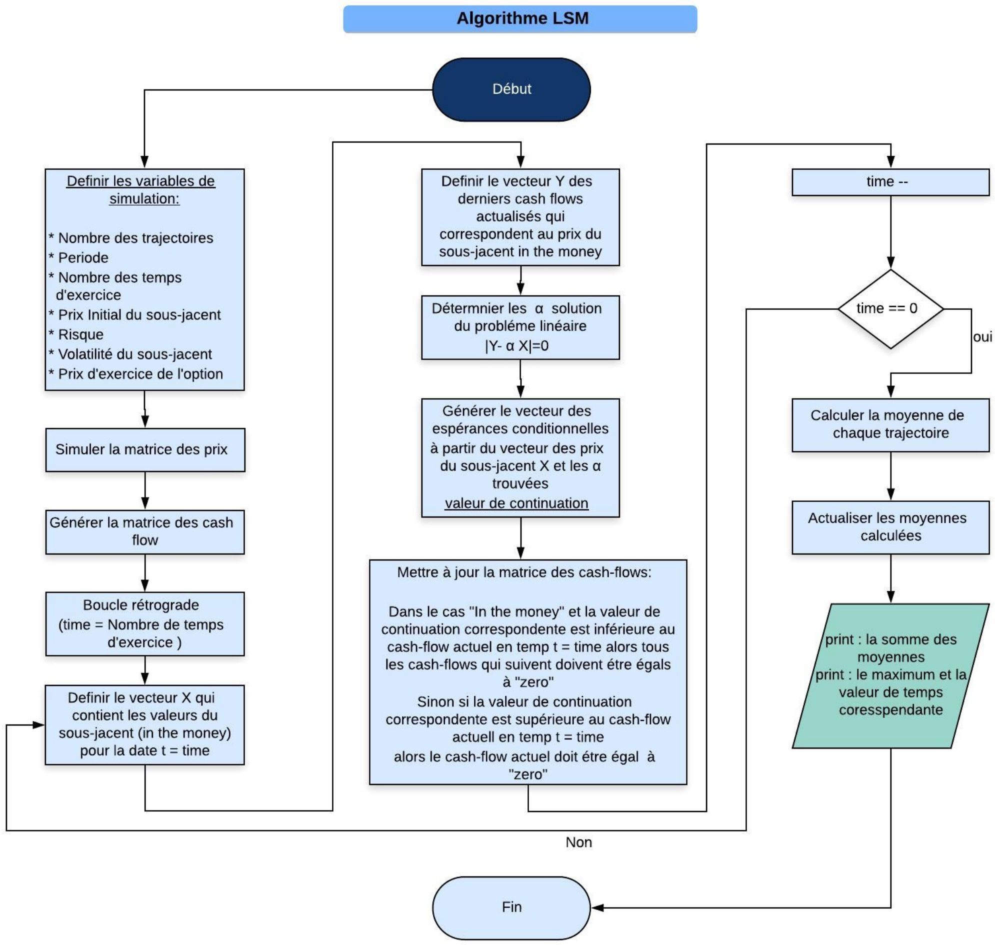

Figure 2: logigramme de l'implémentation de l'algorithme

6 Résultats et Interprétation

Dans cette partie, on présente les résultats de nos différents simulations; on fait fixer le nombre de trajectoires qu'on note M avec M=100000 et aussi le nombre des dates d'exercices à 50, on a appliqué la méthode LSM sur la famille de polynômes de Laguerre $\{L_0, L_1, L_2\}$.

On fait varier le prix initial du sous-jacent qu'on note S_0 , σ et la date d'échéance T.

Le tableau ci dessous (figure 3) présente la même démarche faite par Longstaff Schwartz où la valeur finite est la résultante d'une simulation faite par Longstaff Schwartz qui satisfait l'équation différentielle (8) de l'article d'où on a pris ses valeurs, de même pour les valeurs de Black & Scholes.
On compare les valeurs "Our Early ex" qui est la différence entre les valeurs de notre simulation et les valeurs de Black & Scholes avec "Early ex finite" qui présente la différence entre "Finite difference American" et Black & Scholes. On indique la différence entre ces deux dernières valeurs par "Differnece early ex".

S0	Sigma	T	Black Scholes	Finite Valeur	Notre Valeur (Sim)	Temps (seconds)	Standard error	Early ex Finite	Our Early ex	Difference early ex
36	0.2	1	3.844	4.478	4.464	102.81	0.015	0.634	0.62	0.014
36	0.2	2	3.763	4.84	4.826	93.91	0.023	1.077	1.063	0.014
36	0.4	1	6.711	7.101	7.134	99.77	0.031	0.39	0.423	-0.033
36	0.4	2	7.7	8.508	8.482	100.19	0.048	0.808	0.782	0.026
40	0.2	1	2.066	2.314	2.298	85.76	0.017	0.248	0.232	0.016
40	0.2	2	2.356	2.885	2.869	71.86	0.025	0.529	0.513	0.016
40	0.4	1	5.06	5.312	5.317	85.66	0.035	0.252	0.257	-0.005
40	0.4	2	6.326	6.92	6.896	82.4	0.053	0.594	0.57	0.024
44	0.2	1	1.017	1.11	1.093	46.04	0.019	0.093	0.076	0.017
44	0.2	2	1.429	1.69	1.678	61.74	0.028	0.261	0.249	0.012
44	0.4	1	3.783	3.948	3.925	88.72	0.039	0.165	0.142	0.023
44	0.4	2	5.202	5.647	5.632	101.99	0.059	0.445	0.43	0.015

Figure 3: Tableau comparatif des résultats de la simulation

Constations

On constate que lorsqu'on augmente σ la valeur simulée augmente pour le même prix du sous-jacent et la même période (On pourrait dire dans ce cas que σ n'a pas une grande influence sur le temps de l'exécution); ce résultat est logique puisque l'augmentation de la volatilité du sous-jacent engendre l'augmentation du prix de l'option vu qu'elle représente une bonne stratégie de couverture contre cette volatilité.

L'augmentation du temps T, toute chose étant égale par ailleurs, On constate que le prix simulé de l 'option augmente.

On constate que la diminution de S par rapport à K engendre une augmentation du prix simulé, et inversement lorsque S augmente par rapport à K le prix simulé diminue; ce qui vérifie les caractéristiques d'un PUT.

On compare nos résultats de " Difference in early exercice value" avec celle de Longstaff-Schwartz dans le tableau suivant (figure 4) ; en termes de moyenne et écart type , On peut dire que nos valeurs sont proches de celles de Longstaff-Schwartz.

	Notre simulation	Simulation Longstaff Schawrtz
Moyenne de la difference de "early exercice"	0.011583333	0.007333333
Ecart type de la difference de "early exercice"	0.016087592	0.010873933

Figure 4: Comparaison des résultats

Nombre de bases

On a constaté que lorsque on augmente le nombre des bases, l'écart entre les valeurs simulés est dans l'ordre de 10^{-2} , mais la différence du temps d'exécution est grande, ce qui est logique car on augmentant le nombre des bases plus de calcul est exigé.

On présente ci-dessous les résultats obtenus :

```
######################################################################
############################# Input Data #############################
######################################################################
           number of paths is:  50000
           time periode is:  1
           number of exercice is:  50
           step is :  0.02
           strike price is:  40
           spot price is:  40
           intrest rate is:  0.06
           volatility is:  0.2
           number of polynoms is:  3
######################################################################
############################ Final Values ############################
######################################################################
           max value is:  0.15525376246753503  at time:  49
           finale value is:  2.316336435422731
           execution time:  32.47592902183533  s
######################################################################
######################################################################
######################################################################
```

Figure 5: Résultats de simulation

```
######################################################################
############################ Input Data ##############################
######################################################################
           number of paths is:  50000
           time periode is:  1
           number of exercice is:  50
           step is :  0.02
           strike price is:  40
           spot price is:  40
           intrest rate is:  0.06
           volatility is:  0.2
           number of polynoms is:  4
######################################################################
########################### Final Values #############################
######################################################################
           max value is:  0.10120444237271377  at time:  10
           finale value is:  2.308642613326476
           execution time:  35.66938591003418  s
######################################################################
######################################################################
######################################################################
```

Figure 6: Résultats de simulation

Nombre de trajectoires

On a constaté que lorsqu'on fait augmenter le nombre des trajectoires le temps d'exécution augmente ce qui est explicable par des opérations de calculs supplémentaires.

On présente ci-dessous les résultats obtenus :

```
######################################################################
############################ Input Data ##############################
######################################################################
           number of paths is:  50000
           time periode is:  1
           number of exercice is:  50
           step is :  0.02
           strike price is:  40
           spot price is:  40
           intrest rate is:  0.06
           volatility is:  0.2
           number of polynoms is:  3
######################################################################
########################### Final Values #############################
######################################################################
           max value is:  0.15525376246753503  at time:  49
           finale value is:  2.316336435422731
           execution time:  32.47592902183533  s
######################################################################
######################################################################
######################################################################
```

Figure 7: Résultats de simulation

```
######################################################################
########################### Input Data ###########################
######################################################################
           number of paths is:  100000
           time periode is:  1
           number of exercice is:  50
           step is :  0.02
           strike price is:  40
           spot price is:  40
           intrest rate is:  0.06
           volatility is:  0.2
           number of polynoms is:  3
######################################################################
########################## Final Values ##########################
######################################################################
           max value is:  0.15429077509553224  at time:  49
           finale value is:  2.308286209389306
           execution time:  67.39123177528381  s
######################################################################
######################################################################
######################################################################
```

Figure 8: Résultats de simulation

Temps d'exécution

Lorsque on a augmenté le nombre de trajectoires à M = 100 000 on constate que notre algorithme prend beaucoup plus de temps pour terminer le calcul et c'est du principalement a deux facteur majeurs:

- **Le choix de la technologie :**
  La majorité des algorithmes itératifs et qui nécessite faire des calculs mathématiques sont implémentés avec le langage C++ et comme l'indique cette comparaison de IBM on constate que la courbe de python et c qui donne le temps d'exécution en fonction du taille de la matrice sont parallèle avec une gape plus que 10^2

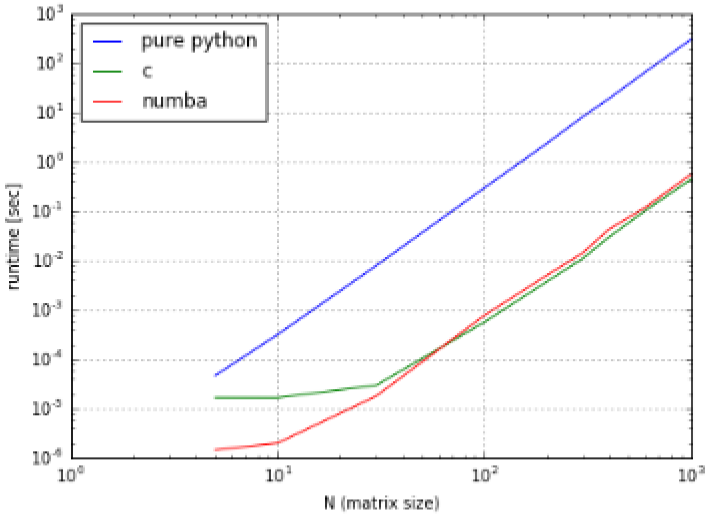

Figure 9: Comparaison du temps d'exécution du c et pure python, IBM

- **La complexité du code :**
  Comme les résultats l'indiquent notre code n'est pas parfaitement optimisé,par exemple d'autres variables peuvent être ajouté pour minimiser le temps du détermination du dernier cash-flow.

7 Avantages et Inconvénients de L'algorithme

7.1 Avantages

- Le faible nombre de scénarios nécessaires au calcul ; on a testé le code sur un nombre de trajectoires égale à 5000 et on obtient presque les memes résultats avec un écart dans l'ordre de 10^{-1}
- La diminution du nombre de trajectoires permet d'accélérer considérablement la vitesse de calcul donc la diminution du temps des simulations (5 secondes pour 5000 trajectoires et 50 secondes pour 50000 trajectoires).
- Le cadre mathématique de la méthode de Monte-Carlo repose fondamentalement sur la loi des grands nombres qui assure une convergence en probabilité vers l'espérance à calculer. La régression peut être complètement automatisée et effectuée en un temps très court de l'ordre de quelques minutes à quelques heures.
- La méthode, est facilement réalisable et ne nécessite pas des ressources informatiques importantes.
- LSM est applicable même pour un nombre N de sous jacents : gestion de portefeuille
- Cette méthode peut être aussi applicable dans les cadres de l'assurance et la gestion des risques.

7.2 Inconvénients

- l'application de la méthode Monte-Carlo nécessite l'utilisation de scénarios stochastiques issus d'un générateur préalablement calibré pour prendre en compte un certain spectre de scénarios. De ce fait, toutes les éventuelles erreurs ou imprécisions de modélisation contenues dans le générateur seront répercutées dans les projections futures.
- La simulation en plus n'est pas suffisamment efficiente pour couvrir les risques économique et culturelles.

8 Code en Python

8.1 functions

```
import numpy as np
'''#### Laguerre base ####'''
def l_0():
    return 1


def l_1(x):
    return 1-x


def l_2(x):
    return (1/2)*(x**2-4*x+2)

def l_3(x):
    return (1/6)*(-x**3+9*x**2-18*x+6)

# used to simulate stock price paths
def simulate_paths(numberOfPaths, timePeriod,
                   numberOftimePeriods, sigma,
                   r, S0):
    deltaT = timePeriod / numberOftimePeriods
    price = np.zeros((numberOftimePeriods + 1,
                      numberOfPaths), np.float64)
    price[0] = S0
    for t in range(1, numberOftimePeriods + 1):
     rand = np.random.standard_normal(numberOfPaths)
     price[t] = (price[t - 1] * np.exp((r - 0.5 *
                sigma ** 2) * deltaT + sigma *
                np.sqrt(deltaT) * rand))
    return price

# renders the solution of |alpha X - Y| = 0
# 3 polynoms
def renderContinuationValues(X, Y):
    # we apply the Lageurre-base polynom on the stock price
    # vector at a
    # given time t to obtain a matrix P_X
    plynomMatrix = np.array([[l_0(), l_1(x), l_2(x)]
                             for x in X])
    # multiplie the stock prices matrix with it's transpose
    A = np.matmul(np.transpose(plynomMatrix), plynomMatrix)
    # multuply the discounted cashFlows vector at times of
    # last one with
    # the transpose of P_X
    B = np.matmul(np.transpose(plynomMatrix), Y)
    # get the alpha* that give us the minimal distance:
    # minimal[(X*alpha-Y)^2]
    alphaStar = np.linalg.solve(A, B)
    # continuation values equation(6) of UCLA article
```

```
    conts = [alphaStar[0] * l_0() +
             alphaStar[1] * l_1(x) +
             alphaStar[2] * l_2(x)
             for x in X]
    return np.array(conts)


# renders the solution of |alpha X - Y| = 0
# 2 polynoms
def renderContinuationValues_2_poly_base(X, Y):
    # we apply the Lageurre-base polynom on the stock price
    # vector at a
    # given time t to obtain a matrix P_X
    plynomMatrix = np.array([[l_0(), l_1(x)] for x in X])
    # multyply the stock prices matrix with it's transpose
    A = np.matmul(np.transpose(plynomMatrix), plynomMatrix)
    # multuply the discounted cashFlows vector at times of
    # last one with
    # the transpose of P_X
    B = np.matmul(np.transpose(plynomMatrix), Y)
    # get the alpha* that give us the minimal distance:
    # minimal[(X*alpha-Y)^2]
    alphaStar = np.linalg.solve(A, B)
    # continuation values equation(6) of UCLA article
    conts = [alphaStar[0] * l_0() +
             alphaStar[1] * l_1(x)
             for x in X]
    return np.array(conts)


# renders the solution of |alpha X - Y| = 0
# 3 polynoms
def renderContinuationValues_4_poly_base(X, Y):
    # we apply the Lageurre-base polynom on the stock price
    # vector at a
    # given time t to obtain a matrix P_X
    plynomMatrix = np.array([[l_0(), l_1(x), l_2(x),
                              l_3(x)] for x in X])
    # multyply the stock prices matrix with it's transpose
    A = np.matmul(np.transpose(plynomMatrix), plynomMatrix)
    # multuply the discounted cashFlows vector at times of
    # last one with
    # the transpose of P_X
    B = np.matmul(np.transpose(plynomMatrix), Y)
    # get the alpha* that give us the minimal distance:
    # minimal[(X*alpha-Y)^2]
    alphaStar = np.linalg.solve(A, B)
    # continuation values equation(6) of UCLA article
    conts = [alphaStar[0] * l_0() +
             alphaStar[1] * l_1(x) +
             alphaStar[2] * l_2(x) +
             alphaStar[3] * l_3(x)
             for x in X]
    return np.array(conts)
```

8.2 LSM

```
import numpy as np
from functions import simulate_paths
'''
# one wants to test the LSM algorithm on the evolution of
# Laguerre base
# just change this import to the corresponding number of
# polynoms function in
# the functions file (includes only 2pol, 3pol, 4pol)
'''
from functions import renderContinuationValues
import time as time_clock
begin = time_clock.time()
# Inputs
numberOfPaths =  50000
timePeriod = 1
numberOftimePeriods = 50
deltaT = timePeriod / numberOftimePeriods
S0 = 40
r = 0.06
sigma = 0.2
K = 40
print('Generating stock price matrix ...')
stock_price = simulate_paths(numberOfPaths, timePeriod,
                             numberOftimePeriods, sigma,
                             r, S0)
print ('Calculating cash flow matrix...')
# At time t = N
cash_flow_matrix = np.zeros_like(stock_price)
for time in range (0,numberOftimePeriods + 1):
    for path_in_time in range (0, numberOfPaths):
        if K - stock_price[time][path_in_time] > 0:
            cash_flow_matrix[time][path_in_time] =(
            K - stock_price[time][path_in_time])
print('Entering loop...')
# stating the loop
for time in range (numberOftimePeriods-1, 0, -1):
    print('Remaining calculations: ',time)
    # fetch the last cash flow for in the money paths
    X = []
    Y = []
    for stock_p in range(0, numberOfPaths):
        if stock_price[time][stock_p] < K:
            X.append(stock_price[time][stock_p])
            for cash_flow_fetcher in range (time + 1,
                                  numberOftimePeriods + 1):
                if (cash_flow_matrix[cash_flow_fetcher]
                                    [stock_p] > 0):
                    Y.append(cash_flow_matrix
                            [cash_flow_fetcher][stock_p]*
                             np.exp(-r * deltaT *
                             (cash_flow_fetcher - time)))
                    break
```

```
                elif (cash_flow_matrix[cash_flow_fetcher]
                     [stock_p] == 0 and cash_flow_fetcher
                     == numberOftimePeriods) :
                    Y.append(0)
        # calculate the continuation values
        if len(X) > 1 and len(Y) > 1:
            continuation_values = renderContinuationValues(X,Y)
            # print(cash_flow_matrix)
            # generate the cash flow vector for time t
            # initialize the continuation values counter
            continuation_values_counter = -1
            cash_flow_vector = np.array([K - x
                                        if x < K else 0 for x in
                                        stock_price[time]])
            for cash_flow_index in range(0, numberOfPaths):
                if cash_flow_vector[cash_flow_index] > 0 :
                    continuation_values_counter += 1
                    # have an in the money path thus we campare
                    #continuation values
                    # to current vector
                    if(cash_flow_vector[cash_flow_index] >
                       continuation_values
                       [continuation_values_counter]):
                        # we exercice immedietly thus we change
                        # all future values
                        # to zero
                        for sub_time in range(time + 1,
                                             numberOftimePeriods + 1 ):
                                cash_flow_matrix[sub_time]
                                [cash_flow_index] = 0
                    else :
                        cash_flow_matrix[time][cash_flow_index]
                         = 0

# calculate mean values
cashFlowMeanVector = [np.mean(x) for x in cash_flow_matrix]
# discount mean values
DiscountedCashFlowVector = [cashFlowMeanVector[i] *
                           np.exp(-r* deltaT * i)
                           for i in range(1,
                            len(cashFlowMeanVector))]
# determine the value
summ=np.sum(DiscountedCashFlowVector)
# determine best stoping time
maxCashFlow = np.max(DiscountedCashFlowVector)
end = time_clock.time()

standard_error_vector = np.matrix(stock_price)
standard_error = np.mean(
       standard_error_vector.std(1)/np.sqrt(numberOfPaths))
print('################################################')
print('################# Input Data ###################')
print('################################################')
print('          ',"number of paths is: ",numberOfPaths)
```

```
print('            ',"time periode is: ",timePeriod)
print('            ',"number of exercice is: ",
                     numberOftimePeriods)
print('            ',"step is : ",deltaT)
print('            ',"strike price is: ",K)
print('            ',"spot price is: ",S0)
print('            ',"intrest rate is: ",r)
print('            ',"volatility is: ",sigma)
print('            ',"number of polynoms is: ",3)
print('##################################################')
print('################# Final Values ###################')
print('##################################################')
print('               ', " error is : ", standard_error )
print('            ',"max value is: ",maxCashFlow,
      " at time: ",
      DiscountedCashFlowVector.index(maxCashFlow))
print('            ',"finale value is: ",summ)
print('            ',"execution time: ", end - begin, ' s')
print('##################################################')
print('##################################################')
print('##################################################')
```

9 Annexe

Équation (3)

on propose la fonction Z avec $Z = \ln(S_t)$
d'après la formule d'Itô on a:

$$\begin{aligned} dZ &= \frac{d\ln(S_t)}{dt} dS_t + \frac{d\ln(S_t)}{dS_t} dS_t + \frac{1}{2}\frac{d^2\ln(S_t)}{d^2 S_t} d < S_t, S_t >_t \\ &= (r - \frac{\sigma^2}{2})dt + \sigma dWt \end{aligned}$$

après intégration sur $[t_k, t_{k+1}]$ on obtient le modèle du prix du sous-jacent :

$$S_{t_{k+1}} = S_{t_k} \exp^{(r-\frac{\sigma^2}{2})t+\sigma\varepsilon\sqrt{\Delta t}}$$

avec ε est une variable aléatoire indépendante qui suit la loi normale centrée réduite.

10 Bibliographie

(01) Chapitre 4 Mouvement Brownien et modèle de Black-Scholes Master Modélisation Statistique M2 Finance - : Clément Dombry, Laboratoire de Mathématiques de Besançon, Université de Franche-Comté.

(02) La modélisation brownienne — Maths et finance ; site web : mines-paritech.fr

(03) Les options :définition, usage, négociation, calcul de la valeur ; site web : fimarkets.com

(04) Article : Valuing American Options by Simulation : A Simple least Square Approch (Francis A.Longstaff , Eduardo S.Schwartz)

(05) Chapitre 2 : L'algorithme de Longstaff et Schwartz

(06) Mémoire d'Actuariat : Application de la méthode Least-Square Monte-Carlo pour la mise en place de l'ORSA en Assurance vie par Tsanta RA-MANAMPISOA

(07) A Speed Comparison Of C, Julia, Python, Numba, and Cython on LU Factorization ; site web : ibm.com

(08) Simulating Stock Prices Using Geometric Brownian Motion: Evidence from Australian Companies Krishna Reddy The University of Waikato, New Zealand.