

La simulation de Monte Carlo : forces et faiblesses

**Avec applications Visual Basic et Matlab
&
présentation d'une nouvelle méthode QMC**

François-Éric Racicot*

Département des sciences administratives
Université du Québec, Outaouais et LRSP

Raymond Théoret

Département Stratégie des Affaires
Université du Québec, Montréal

RePAd Working Paper No. 052006

* Adresse postale : François-Éric Racicot, Département des sciences administratives, Université du Québec en Outaouais, Pavillon Lucien Brault, 101 rue Saint Jean Bosco, Gatineau, Québec, Canada, J8Y 3J5.

Correspondance : francoiseric.racicot@uqo.ca.

Raymond Théoret, Département stratégie des affaires, Université du Québec à Montréal, 315 est, Ste-Catherine, Montréal, H2X-3X2. Correspondance : theoret.raymond@uqam.ca.

Ce papier est l'un des chapitres de notre prochain ouvrage intitulé : *Finance computationnelle et gestion des risques*.

La simulation de Monte Carlo : forces et faiblesses

Avec applications Visual Basic et Matlab & présentation d'une nouvelle méthode QMC

Résumé

La simulation de Monte Carlo comporte un avantage certain sur la technique de l'arbre binomial puisqu'elle est en mesure d'intégrer facilement les diverses dimensions d'un problème. Cependant, sa vitesse de convergence est plus lente. Dans cet article, nous montrons comment cette méthode peut être améliorée en recourant à diverses techniques : méthode des variables antithétiques, méthode des variables de contrôle et séquences de nombres déterministes : séquences de Fauré, de Sobol et d'Halton. Nous montrons comment calculer l'écart-type d'une simulation de Monte Carlo lorsque les payoffs d'un titre, tel un titre contingent, s'avèrent non linéaires. Il faut alors calculer cet écart-type non pas sur une seule simulation mais sur un grand nombre de simulations répétées de manière à ce que la distribution des résultats soit normale. La moyenne des moyennes de ces simulations est alors un bon estimateur du prix recherché. Nous montrons également comment combiner la séquence des nombres d'Halton avec des variables antithétiques pour améliorer la convergence d'une QMC (quasi Monte Carlo). Ce qui constitue notre nouvelle version de ladite méthode qui porte alors bien son nom car le résultat obtenu varie alors d'une simulation à l'autre alors que dans une QMC classique, le résultat, à l'instar de l'arbre binomial, demeure irrémédiablement fixe (non aléatoire).

Abstract

Monte Carlo simulation has an advantage upon the binomial tree as it can take into account the multi-dimensions of a problem. However its convergence speed is slower. In this article, we show how this method may be improved by various means: antithetic variables, control variates and low discrepancy sequences: Faure, Sobol and Halton sequences. We show how to compute the standard deviation of a Monte Carlo simulation when the payoffs of a claim, like a contingent claim, are nonlinear. In this case, we must compute this standard deviation by doing a great number of repeated simulations such that we arrive at a normal distribution of the results. The mean of the means of these simulations is then a good estimator of the wanted price. We also show how to combine Halton numbers with antithetic variables to improve the convergence of a QMC. That is our new version of QMC which is then well named because the result varies from one simulation to the other in our version of the QMC while the result is fixed (not random) in a classical QMC, like in the binomial tree.

Mots-clés : ingénierie financière; produits dérivés; simulation de Monte Carlo ; séquences de nombres quasi-aléatoires.

Keywords : financial engineering; derivatives; Monte Carlo simulation; low discrepancy sequences.

JEL : G12; G13; G33.

Depuis son incorporation par Boyle (1977) dans la panoplie des outils de la finance computationnelle, la simulation de Monte Carlo n'a eu de cesse de gagner en popularité comme méthode de calcul de prix d'options de plus en plus sophistiquées et comme instrument de gestion des risques. Elle se caractérise par sa très grande souplesse et par sa capacité de traiter un problème en plusieurs dimensions.

Le calcul du prix d'une option revient à la solution d'une équation différentielle. Mais comme le stipule le théorème de Feynman-Kac, une équation différentielle peut être représentée par une espérance mathématique. Or, qui dit espérance mathématique dit intégrale. Et c'est justement l'une des principaux objectifs de la simulation de Monte Carlo que d'estimer une intégrale. On comprend dès lors le lien très étroit qui existe entre les prix des produits dérivés et la simulation de Monte Carlo.

Le but du présent article est d'examiner de façon détaillée les principaux aspects de l'utilisation de la simulation de Monte Carlo en finance computationnelle et de fournir des programmes écrits en Matlab et en Visual Basic conçus de manière à maîtriser ces aspects. Nous introduirons d'abord ces méthodes en suivant la démarche adoptée par Boyle dans son article de 1977. Puis nous verrons comment la performance de la simulation de Monte Carlo peut être améliorée en recourant à plusieurs techniques : variables antithétiques, variables de contrôle et séquences de nombres pseudo-aléatoires.

1. Les aspects généraux de la simulation de Monte Carlo

Nous savons que la solution de Black et Scholes au calcul de prix d'une option d'achat européenne passe par la solution de l'équation différentielle suivante :

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS \frac{\partial C}{\partial S} - rC = 0$$

avec comme condition terminale :

$$C(T, S) = (S(T) - K)^+$$

Selon le théorème de représentation de Feynman-Kac qui établit l'équivalence entre une équation différentielle et une espérance mathématique, si C est une solution à ce système d'équations, on peut alors exprimer cette solution comme une espérance du payoff final de l'option, c'est-à-dire :

$$C = e^{r(T-t)} E^Q[C(T, S)]$$

avec E^Q , l'opérateur d'espérance dans un univers risque-neutre. $C(T, S)$, le *payoff*¹ de l'option d'achat, est une variable aléatoire. Ce sont Cox et Ross (1976) qui furent les premiers à faire le lien entre le théorème de représentation de Feynman-Kac et le prix d'un call vu comme une espérance.

Or, une espérance est une intégrale. Pour une variable aléatoire X , son espérance se calcule comme suit :

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx$$

Par ailleurs, on calcule l'espérance d'une fonction $g(y)$ comme suit :

$$E(g(y)) = \int_{-\infty}^{\infty} g(y)f(y)dy$$

où : $\int_{-\infty}^{\infty} f(y)dy = 1$.

Soit $E[g(y)] = \bar{g}$. On veut trouver un estimé \hat{g} de \bar{g} , ce dernier étant inconnu. On recourt pour ce faire à la simulation de Monte Carlo. L'évaluation d'intégrales est la principale utilisation de la simulation de Monte Carlo en finance. En effet, on peut représenter les prix d'options comme suit :

$$\text{prix d'options} = E^Q(.) = \int(.)$$

En temps continu, une espérance est une intégrale. En temps discret, c'est une moyenne. C'est cette moyenne que vise à calculer la simulation de Monte Carlo. Elle deviendra un estimateur du prix d'une option. Au tableau 1, on identifie les principales étapes du calcul de \hat{g} , soit l'estimateur de \bar{g} , à l'aide de la simulation de Monte Carlo.

¹ Rappelons que le terme «payoff» désigne le cash-flow de l'option à l'échéance de celle-ci. Comme nous ne connaissons pas le prix de l'action à l'échéance de l'option, le payoff est donc une variable aléatoire.

Tableau 1

Calcul de \hat{g} par la simulation de Monte Carlo.

Étape 1	Par la simulation de MC, on génère au hasard des réalisations de $y : y_1, y_2, \dots, y_n$.
Étape 2	Chacune de ces réalisations donne une valeur à $g(y)$.
Étape 3	L'estimé de l'intégrale qui résulte de la simulation est de : $\hat{g} = \frac{1}{n} \sum_{i=1}^n g(y_i)$
Étape 4	On peut évaluer la précision de cet estimé en calculant \hat{s} (déviation standard) : $\hat{s} = \frac{1}{\sqrt{n-1}} \sqrt{\sum_{i=1}^n \left[\underbrace{g(y_i)}_{\text{une itération}} - \underbrace{\hat{g}}_{\text{n itérations}} \right]^2}$

Asymptotiquement, c'est-à-dire quand $n \rightarrow \infty$, la distribution suivante tend vers une normale :

$$\frac{\hat{g} - \bar{g}}{\hat{s}/\sqrt{n-1}} \underset{\sim}{\sim} aN(0,1)$$

L'intervalle de confiance de l'estimateur \hat{g} est donc de :

$$\hat{g} \pm \alpha_c \frac{\hat{s}}{\sqrt{n-1}}$$

avec α_c , la valeur critique.

En ce qui nous concerne, \hat{g} est un estimateur du prix d'une option.

Pour illustrer les développements précédents, nous allons évaluer le prix d'un call européen asiatique en recourant à la simulation de Monte Carlo. Le *payoff* d'une telle option est de: $(\bar{S} - X)^+$, où \bar{S} est la moyenne arithmétique du prix du sous-jacent calculée sur la durée de vie de l'option. L'option asiatique dépend donc du sentier suivi par l'action jusqu'à l'échéance de l'option. On dit en anglais qu'elle est : *path dependent*.

Le prix de l'option asiatique est égal à l'espérance risque-neutre suivante :

$$C(t,s) = e^{-r(T-t)} E^Q \left[(\bar{S} - X)^+ \right]$$

E^Q étant une intégrale, on l'évalue par la simulation de Monte Carlo. En termes discrets, le prix de l'option se ramène à l'expression suivante, qui est directement calculable par la simulation de Monte Carlo :

$$\hat{C} = \left[\sum_{i=1}^n \frac{CF_i(T)}{n} \right] \times e^{-r(T-t)}$$

avec n le nombre d'itérations de la simulation et $CF_i(T) : [\bar{S}_i(T) - K]^+$, où $\bar{S}_i(T)$: moyenne des prix de l'action lors de l'itération i , qui est une variable aléatoire.

Pour effectuer la simulation, on se donne d'abord une représentation lognormale du prix de l'action :

$$S_{t+1} = S_t e^{\left(r - \frac{1}{2}\sigma^2 \right) \Delta t + \sigma \sqrt{\Delta t} \varepsilon} \quad (1)$$

avec $\varepsilon \sim N(0,1)$. L'équation (1) est la solution du mouvement brownien géométrique suivant supposé pour le prix de l'action dans l'univers risque-neutre:

$$dS = rSdt + \sigma Sdz$$

avec $dz = \varepsilon \sqrt{dt}$

Les étapes du calcul du prix d'un call européen asiatique par la méthode de la simulation de Monte Carlo sont les suivantes :

1. Diviser la durée T de l'option en m pas $\Delta t = \frac{T}{m}$. Δt est défini sur une base annuelle puisque tous les paramètres du call sont exprimés sur une base annuelle.
2. Générer un premier nombre aléatoire ε et calculer S_1 à partir de S_0 (connu).
3. Générer de façon séquentielle d'autres variables aléatoires et substituer dans (1).
4. On obtient en bout de piste une trajectoire (*path*) de S .

On calcule alors sa moyenne sur la trajectoire :

$$\bar{S} = \frac{\sum S_j}{m}$$

5. On calcule le cash-flow final de l'option asiatique pour cette trajectoire :

$$CF_i = (\bar{S}_i - K)^+$$

On effectue N itérations de la sorte (N doit être important pour réduire autant que possible \hat{s}) et on obtient un cash-flow respectif pour chaque itération.

6. On obtient finalement le prix du call asiatique :

$$\hat{C} = e^{-r(T-t)} \sum_{i=1}^N \frac{CF_i}{N}$$

L'écart-type de cette estimation est le suivant.

$$\hat{s} = \frac{1}{\sqrt{n-1}} \sqrt{\sum (C_i - \hat{C})^2}$$

avec : $C_i = e^{-r(T-t)} CF_i$, i symbolisation l'itération.

Cet écart-type est inapproprié pour évaluer la performance d'une simulation de Monte Carlo orientée vers le calcul du prix d'une option. En effet, les payoffs d'une option n'obéissent pas à une distribution normale. Pour pallier à ce problème, nous reprendrons un grand nombre de fois la même simulation. La distribution des résultats devrait tendre vers la normale. Nous calculerons la moyenne des résultats, qui devrait être centrée sur le prix effectif de l'option, de même que l'écart-type. Nous serons alors à même de juger de la performance de la simulation effectuée.

Nous voulons valoriser un call asiatique européen dont les spécifications se retrouvent au tableau 2. Le tableau 3 fournit un programme écrit en Visual Basic de nature à évaluer le prix d'une option asiatique (call ou put). Quand il s'agit d'un call, comme dans le cas présent, la variable *iopt* prend la valeur 1. Et lorsqu'il s'agit d'un put, la valeur (-1) lui est attribuée. Les paramètres du call asiatique se retrouvent au tableau 2.

Tableau 2

S	80
X	85
T	1 an
Rf	0,05
Q	0
σ	0,2

Tableau 3 Simulation du prix d'une option asiatique (moyenne arithmétique)

Sub simopt()

'Simulation de Monte Carlo pour calculer le prix d'une option asiatique

Range("d4:iv6000").ClearContents

Range("option1").ClearContents

Dim iopt1, s1, x1, rf1, q1, t1, sigma1, nsimg1, pas

Dim rnmulg, sigtg, sumg, randnsg, S1g, payoff1g, sigsum, sigmoyenne

Dim i As Integer

Dim j As Integer

iopt1 = 1

s1 = 80

x1 = 85

rf1 = 0.05

q1 = 0

t1 = 1

sigma1 = 0.2

pas = 100

nsimg1 = 100

rnmulg = (rf1 - q1 - 0.5 * sigma1 ^ 2) * (t1 / pas)

sigtg = sigma1 * Sqr(t1 / pas)

sumg = 0

For i = 1 To nsimg1

S1g = s1

sigsum = 0

For j = 1 To pas

Randomize

randnsg = Application.NormSInv(Rnd)

S1g = S1g * Exp(rnmulg + randnsg * sigtg)

Range("prix1").Offset(j - 1, i - 1) = S1g

sigsum = sigsum + S1g

Next j

sigmoyenne = sigsum / pas

Range("prix").Offset(i - 1, 0) = sigmoyenne

payoff1g = Application.Max(iopt1 * (sigmoyenne - x1), 0)

Range("cash").Offset(i - 1, 0) = payoff1g

```

sumg = sumg + payoff1g
Next i

```

```

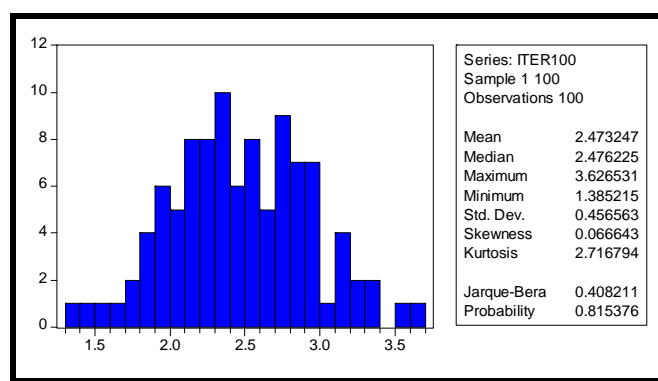
option1 = Exp(-rf1 * t1) * sumg / nsimg1
Range("option1").Value = option1

```

End Sub

Nous fixons dans un premier temps le nombre de pas à 100 et le nombre d'itérations, également à 100. Nous obtenons un prix de 1,47 \$ pour le call européen asiatique dont les spécifications apparaissent au tableau 2. Nous effectuons, à l'aide d'une boucle ajoutée au programme du tableau 3, 100 simulations additionnelles. L'histogramme des simulations, de même que les principales statistiques, sont répertoriés à la figure 1.

Figure 1



Source : EViews

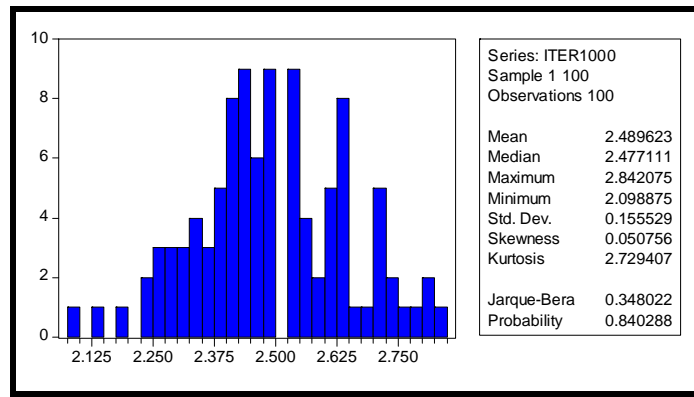
Comme on peut le constater à la figure 1, la moyenne des simulations est de 2,47, ce qui est très rapproché du prix effectif du call asiatique, c'est-à-dire 2,48 \$. La simulation que nous avons effectuée antérieurement qui lui assignait un prix de 1,47 \$ était donc bien loin de la marque. A hauteur de 0,45, l'écart-type des simulations est important. La statistique Bera-Jarque indiquant que la distribution des prix de l'option s'avère normale, l'intervalle pour un niveau de confiance de 95% du prix de l'option est de :

$$2,47 \pm (1,96)(0,45) = 2,47 \pm 0,88$$

L'intervalle de confiance du prix de l'option est déraisonnablement élevé puisqu'il s'étire de 1,59 \$ à 3,35\$. Le résultat obtenu d'une simulation n'est donc pas fiable pour un nombre d'itérations de 100.

Une façon de réduire l'intervalle de confiance du prix de l'option est d'accroître le nombre d'itérations. Nous avons refait les 100 simulations en augmentant le nombre d'itérations de 100 à 1000. Les résultats des simulations sont compilés à la figure 2.

Figure 2



Source : EViews

On remarque à la figure 2 que la distribution des simulations avec 1000 itérations est mieux centrée sur le prix théorique du call asiatique, à hauteur de 2,48\$, que la distribution qui correspond à 100 itérations (figure 1) . On note également à la figure 2 que l'écart-type des simulations s'est beaucoup réduit en augmentant le nombre de simulations de 100 à 1000 puisqu'il est passé de 0,45 à 0,15. Cette réduction était anticipée car lorsque le nombre d'itérations est de 100, la racine carrée de la somme des erreurs au carré est divisée par $\sqrt{100}$ lors du calcul de l'écart-type, tandis que lorsque le nombre d'itérations est de 1000, le diviseur est de $\sqrt{1000}$. Mais, encore là, l'intervalle de confiance du prix de l'option se révèle trop important. Pour le réduire davantage, on peut encore une fois augmenter le nombre d'itérations mais ces opérations finissent par consommer beaucoup de temps. Les sections qui suivent envisagent diverses techniques aptes à réduire cet intervalle dans un laps de temps raisonnable.

Nous pouvons également transposer les programmes précédents en langage Matlab. Pour ce faire, nous emprunterons nos fonctions à Brandimarte (2002). Ces fonctions apparaissent au tableau 4. La fonction Asia calcule le prix d'une option asiatique. Elle fait appel à la fonction *SentierBrownien* qui génère les scénarios du prix du sous-jacent.

Tableau 4 Fonctions Matlab du calcul du prix d'une option asiatique

```
function [prix,c]=Asia(S0,X,r,t,sigma,Npas,Niter)
profit=zeros(Niter,1);
for i=1:Niter
    Sentier=SentierBrownien(S0,r,t,sigma,Npas,1);
    profit(i)=max(0,mean(Sentier(2:(Npas+1)))-X);
end
[prix,a,c]=normfit(exp(-r*t)*profit);
```

```
function Sentier=SentierBrownien(S0,r,t,sigma,Npas,Niter)
dt=t/Npas;
rdt=(r-0.5*sigma^2)*dt;
sidt=sigma*sqrt(dt);
Increments=rdt+sidt*randn(Niter,Npas);
LogSentier=cumsum([log(S0)*ones(Niter,1),Increments],2);
Sentier=exp(LogSentier);
```

Calculons la valeur du call asiatique précédent en nous servant des commandes Matlab avec dans un premier temps 1000 itérations. Nous écrivons dans la fenêtre des commandes du logiciel Matlab :

```
tic,P=Asia(80,85,0.05,1,0.2,100,1000),toc,
```

P =

2.5260

elapsed_time =

0.3010

La valeur calculée pour le prix du call asiatique est donc de 2,52 \$. Nous savons que sa vraie valeur est de 2,48\$. Nous pouvons nous conforter en effectuant 10 million d'itérations, ce qui dépasse certes les capacités de Visual Basic (Excel). Le résultat correspond à nos attentes :

```
tic,P=Asia(80,85,0.05,1,0.2,100,10000000),toc,
```

P =

2.4847

elapsed_time =

860.1670

Le résultat s'est affiché après 14 minutes, ce qui est relativement rapide étant donné le nombre d'itérations demandé.

2. Les variables antithétiques²

La technique des variables antithétiques est, parmi la panoplie des techniques de réduction de la variance, la plus simple. Définissons des variables aléatoires dites antithétiques (opposées) dont la corrélation est de -1. C'est-à-dire que pour chaque scénario du prix de l'action, on aura les deux équations suivantes:

$$S_{t+\Delta t} = S_t e^{\left(r - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\varepsilon\sqrt{\Delta t}}$$

$$S_{t+\Delta t} = S_t e^{\left(r - \frac{1}{2}\sigma^2\right)\Delta t - \sigma\varepsilon\sqrt{\Delta t}}$$

La même variable aléatoire ε est d'abord introduite avec un signe positif pour calculer S puis avec un signe négatif. Il en résulte une corrélation de (-1) entre les deux mouvements browniens, ce qui accélère, pour un nombre d'itérations donné, la convergence vers le vrai prix de l'option. Le tableau 5 fait état de k simulations de Monte Carlo pour un call européen avec variables antithétiques.

Tableau 5 Programme Visual Basic de k simulations de Monte Carlo du prix d'un call asiatique avec variables antithétiques

Sub simopt()

'Simulation de Monte Carlo pour calculer prix d'une option asiatique

'Range("d4:iv6000").ClearContents

'Range("option1").ClearContents

Dim iopt1, s1, x1, rfl, q1, t1, sigma1, nsim1, pas

Dim rmutg, sigtg, sumg, randnsg, S1g, payoff1g, sigsum, sigmoyenne

Dim i As Integer

Dim j As Integer

'Randomize

iopt1 = 1

s1 = 80

x1 = 85

² Cette section s'inspire directement de : Racicot F. et R. Théoret (2004), chapitre 1.

```

rf1 = 0.05
q1 = 0
t1 = 1
sigma1 = 0.2
pas = 100
nsimg1 = 100

```

```

rnmulg = (rf1 - q1 - 0.5 * sigma1 ^ 2) * (t1 / pas)
sigtg = sigma1 * Sqr(t1 / pas)

```

```

For k = 1 To 100
sumg = 0

```

```

For i = 1 To nsimg1

```

```

S1g = s1
S2g = s1
sigsum1 = 0
sigsum2 = 0
For j = 1 To pas

```

```

randnsg = Application.NormSInv(Rnd)
S1g = S1g * Exp(rnmulg + randnsg * sigtg)
'Range("prix1").Offset(j - 1, i - 1) = S1g
sigsum1 = sigsum1 + S1g
S2g = S2g * Exp(rnmulg - randnsg * sigtg)
sigsum2 = sigsum2 + S2g

```

```

Next j
sigmoyenne1 = sigsum1 / pas
'Range("prix").Offset(i - 1, 0) = sigmoyenne
sigmoyenne2 = sigsum2 / pas

```

```

payoff1g = 0.5 * Application.Max(iopt1 * (sigmoyenne1 - x1), 0) + 0.5 * Application.Max(iopt1 *
(sigmoyenne2 - x1), 0)
'Range("cash").Offset(i - 1, 0) = payoff1g

```

```

sumg = sumg + payoff1g
Next i

```

```

option1 = Exp(-rf1 * t1) * sumg / nsimg1
Range("histo5").Offset(k, 0) = option1

```

```

Next k

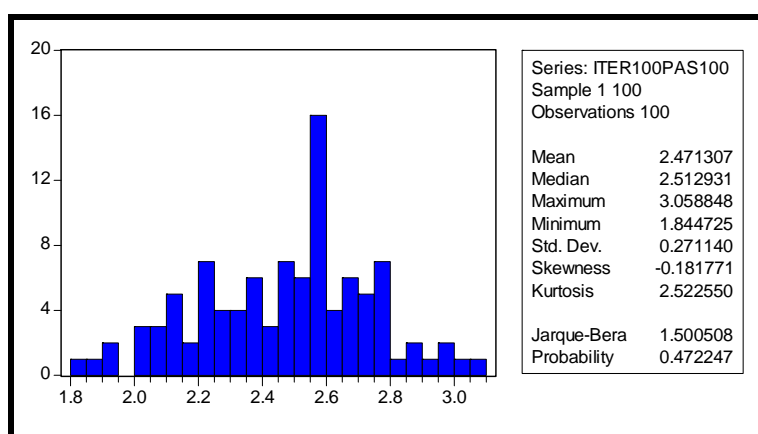
```

End Sub

Comme on le remarque à la lecture du tableau 5, à chaque itération, on calcule la moyenne du prix de l'action des scénarios obtenus à partir des deux variables antithétiques. Le payoff de l'option d'une itération est la moyenne des deux payoffs obtenus à partir des deux variables antithétiques. Cela accélère la convergence vers le vrai prix de l'action.

La figure 3 relate les résultats de 100 simulations du prix du call asiatique lorsque l'on recourt aux variables antithétiques. Chaque simulation comporte 100 itérations et 100 pas.

Figure 3

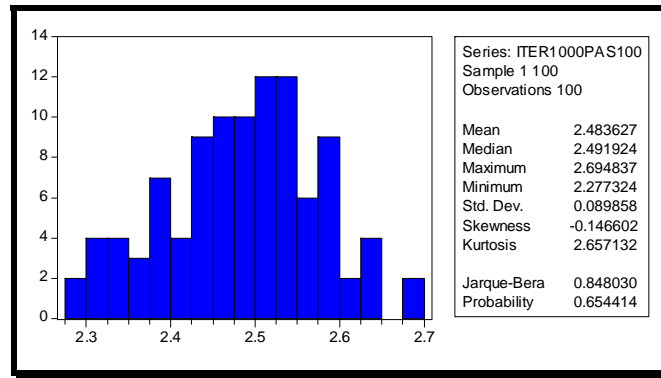


Source : EViews

On constate que, bien que la moyenne soit la même que celle obtenue sans variables antithétiques, l'écart-type des simulations s'est abaissé de 0,45 à 0,27 lorsque l'on passe de simulations sans variables antithétiques à des simulations avec variables antithétiques pour un même nombre d'itérations de 100, ce qui prend acte de la performance de la méthode des variables antithétiques. Mais un nombre d'itérations de 100 n'est pas encore suffisant, l'intervalle de confiance du prix de l'option étant encore trop élevé.

Nous augmentons donc le nombre d'itérations pour chaque simulation à 1000. La figure 4 fait état des résultats obtenus.

Figure 4



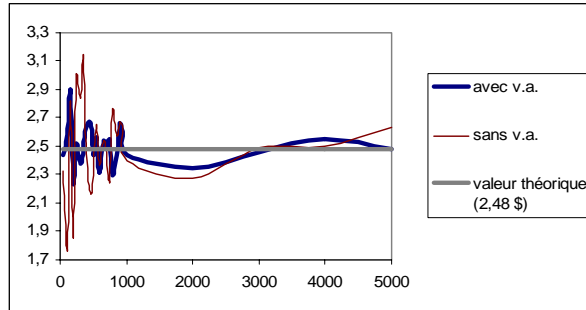
Source : EViews

Comme on le note à la figure 4, l'écart-type des simulations, à hauteur de 0,08, est pratiquement réduit de moitié en regard des simulations qui ne font pas appel aux variables antithétiques, cela pour le même nombre d'itérations, soit 1000. Augmenter par ailleurs le nombre de pas n'améliore pas les résultats.

La figure 5 montre comment le prix du call asiatique converge vers sa valeur théorique en fonction du nombre d'itérations dans deux situations : l'une avec variables antithétiques et l'autre, sans variables antithétiques. La figure révèle que les fluctuations du prix de l'option en regard de son prix théorique sont beaucoup plus faibles lorsque l'on recourt à des variables antithétiques. Mais même après 5000 itérations, le risque d'erreur n'est pas négligeable même si l'on fait appel à des variables antithétiques pour réduire la variance des simulations.

Figure 5

Convergence du prix de l'option en fonction du nombre d'itérations avec et sans variables antithétiques



3. La technique des variables de contrôle³.

La technique des variables de contrôle⁴ est une autre méthode pour réduire l'écart type d'une simulation de Monte Carlo. Formellement, une variable de contrôle⁵, que nous désignons par cv , doit être corrélée positivement avec V , la valeur de l'actif contingent que nous voulons estimer. Cependant, $E(cv)$ doit être égal à zéro. Construisons le portefeuille suivant V' , où V est le prix de l'actif contingent que nous voulons évaluer :

$$V' = V - \beta cv + \xi$$

V' est assimilable à un portefeuille couvert au sein duquel cv sert d'instrument de couverture pour V . Mais cette couverture est imparfaite, de telle sorte que l'équation incorpore un terme d'erreur ξ . Plutôt que de simuler directement la valeur de V , nous simulons la valeur de V' pour calculer la valeur de V . Nous avons :

$$E(V') = E(V)$$

$E(cv)$ étant égal à 0, par hypothèse. Nous pouvons par conséquent approcher la valeur de V en recourant au portefeuille V' . La variance de V' est de:

$$\sigma_{V'}^2 = \sigma_V^2 - 2\beta\sigma_{V,cv} + \beta^2\sigma_{cv}^2 \quad (2)$$

Étant donné que l'utilisation de cv est censée réduire la variance de la simulation, il est approprié de rechercher le β qui minimise $\sigma_{V'}^2$. En dérivant la dernière équation par rapport à β et en égalant le résultant à 0, on obtient :

$$\hat{\beta} = \frac{\sigma_{V,cv}}{\sigma_{cv}^2} \quad (3)$$

³ Cette section s'inspire directement de : Racicot F. et R. Théoret (2004), chapitre 1.

⁴ *Control variate*, en anglais.

⁵ A ce sujet, on consultera : James et Webber (2000).

A l'évidence, ce bêta s'assimile à l'estimateur des moindres carrés ordinaires (MCO) d'une régression de V sur cv . Clewlow et Strickland (1997)⁶ recourent à cette méthode d'estimation pour évaluer la sensibilité de V à cv . En substituant l'équation (3) dans l'équation (2) et en écrivant le résultat en termes de la corrélation ρ entre V et cv , on a :

$$\sigma_{V'}^2 = \sigma_V^2 (1 - \rho^2)$$

En vertu de cette équation, la variance de la simulation est reliée négativement à la corrélation entre la valeur du titre contingent et la variable de contrôle. A la limite, cette corrélation est de 1. Nous sommes alors en présence d'une couverture parfaite et l'erreur standard de la simulation est nulle. Une bonne variable de contrôle doit par conséquent entretenir une forte corrélation avec le titre contingent que nous voulons évaluer.

Le concept de variable de contrôle est si important en finance que nous nous devons de l'illustrer par quelques exemples⁷. Le premier concerne la détermination du prix d'une option d'achat asiatique arithmétique dont il a été question dans la section précédente. Le flux monétaire à l'échéance (*payoff*) de cette option est le suivant:

$$\text{payoff} = (\bar{S}_{aT} - X)^+$$

\bar{S}_{aT} étant la moyenne arithmétique du sous-jacent calculée sur la durée de vie de l'option selon une règle spécifique. Il n'existe aucune solution analytique pour le call asiatique arithmétique mais il en existe une pour sa version géométrique dont le prix est évidemment très corrélé avec celui de l'option arithmétique. Nous pouvons exploiter cette relation pour créer une variable de contrôle. N'oublions pas qu'une variable de contrôle est un portefeuille couvert en finance. Ce portefeuille est constitué d'une position en compte (*long*) dans le call arithmétique et d'une position à découvert (*short*) dans le call géométrique. Nous avons:

$$\tilde{A} = \hat{A} - (\beta \times cv)$$

où \tilde{A} est le portefeuille qui sert à calculer le prix du call arithmétique et où \hat{A} est la valeur simulée du call arithmétique. La variable de contrôle choisie cv est de $(\hat{G} - G)$, où \hat{G} est la valeur simulée du call géométrique calculée en recourant aux mêmes variables aléatoires que celles qui sont utilisées dans le cas du calcul du call arithmétique et où G est la valeur analytique du call géométrique, c'est-à-dire sa "vraie valeur". En vertu de la définition de la variable de contrôle, l'équation précédente devient:

$$\tilde{A} = \hat{A} - (\hat{G} - G)$$

⁶ Clewlow, L. et Strickland, C. (1997), Monte Carlo Valuation of Interest Rate Derivatives under Stochastic Volatility, *Journal of Fixed Income*, pp. 35-45.

⁷ A ce sujet, on consultera : Clewlow et Caverhill (1994), Boyle(1977), James et Webber (2000), Clewlow et Strickland (1998) et Clewlow et Strickland (1997).

Cette variable de contrôle satisfait aux deux exigences d'une bonne variable de contrôle. Premièrement, la corrélation entre \hat{A} et $(\hat{G} - G)$ ⁸ est évidemment très élevée. A ce point que nous fixons β à 1. Deuxièmement, $E(cv) = 0$, parce que $E(\hat{G}) = G$. Nous pouvons réécrire la dernière équation comme suit pour les besoins de la simulation:

$$\tilde{A} = G + (\hat{A} - \hat{G}) \quad (4)$$

A l'aide d'une simulation de Monte Carlo, on obtient la valeur du portefeuille $(\hat{A} - \hat{G})$ en calculant la moyenne actualisée des flux monétaires à l'échéance (*payoffs*) de ce portefeuille sur l'ensemble des M simulations, c'est-à-dire:

$$\hat{A} - \hat{G} = \frac{1}{M} \sum_{j=1}^M \left[(\bar{S}_{aT}^j - X)^+ - (\bar{S}_{gT}^j - X)^+ \right] \times e^{-rT}$$

Conformément à l'équation (4), nous ajouterons cette valeur estimée à G à la fin des simulations de façon à obtenir une approximation du prix du call arithmétique. En agissant de la sorte, nous aurons grandement réduit l'écart type de la simulation entre raison de la corrélation très élevée des prix des calls arithmétique et géométrique.

Une autre façon de construire des variables de contrôle est de recourir aux "grecques"⁹ pour couvrir un portefeuille. On peut ainsi construire un portefeuille qui est couvert par le delta¹⁰, par le delta-gamma¹¹, etc.. Clewlow et Strickland (1997,1998) et Clewlow et Carverhill (1994) donnent plusieurs exemples de ces types de couvertures. Leur méthode recourt à l'économétrie pour estimer les sensibilités des prix des options évaluées aux variables de contrôle.

Considérons un cas de couverture avec les grecques. Nous voulons évaluer par simulation le prix d'une option d'achat européenne standard¹². Pour y arriver, nous simulons le portefeuille suivant:

$$\text{portfolio} = C - \Delta S$$

c'est-à-dire l'écriture d'un call qui est couvert par une quantité Δ du sous-jacent S ¹³. Cette couverture en est une du type dynamique parce que le Δ réagit aux changements de S qui se produisent durant la durée de vie de l'option. Celui qui écrit l'option déposera la prime résultant de la vente du call dans un compte bancaire et vendra ou achètera le sous-jacent

⁸ N'oublions pas que G est une constante, étant la solution analytique du call géométrique.

⁹ Rappelons que les "grecques" sont les diverses dérivées du prix d'une option par rapport à ses variables explicatives.

¹⁰ Un *delta-hedged portfolio*, en anglais.

¹¹ Un *delta-gamma hedged portfolio*, en anglais.

¹² *Plain vanilla*, en anglais.

¹³ Cette équation est écrite en termes de flux monétaires. L'encaissement de la prime de l'option donne lieu à un flux monétaire positif. L'achat de l'action donne lieu à un flux monétaire négatif.

pour réajuster la couverture à la suite du changement du delta. Cela donne lieu à des mouvements de fonds positifs ou négatifs dans le compte bancaire. La contrepartie dynamique de cette dernière équation est:

$$C_0 e^{r(T)} - \left[\sum_{i=0}^{N-1} \left(\frac{\partial C_{ti}}{\partial S} - \frac{\partial C_{t(i-1)}}{\partial S} \right) S_{ti} e^{r(T-ti)} \right] = C_T + \eta \quad (5)$$

Le premier terme de l'équation (5) est le prix à terme (*forward price*) de l'option, c'est-à-dire $C_0 e^{rT}$, où C_0 est la prime ou le prix de l'option. Le terme entre crochets, qui est la variable de contrôle cv , représente les réajustements du portefeuille à la suite des changements du delta $\frac{\partial C_{ti}}{\partial S}$ survenus durant la simulation du sentier temporel du sous-jacent. Le delta a une solution analytique, soit le $N(d_1)$ de l'équation de Black et Scholes. Le prix à terme de l'option additionné des réaménagements de portefeuille reproduit les flux monétaires de l'option avec une erreur égale à η puisque les calculs s'effectuent en temps discret.

En remplaçant le terme entre crochets par cv dans l'équation (5), on a:

$$C_0^j e^{rT} - cv^j = C_T^j + \eta^j \quad (6)$$

où l'indice j désigne un résultat de la simulation j parmi M simulations. On notera que le bêta de cv est de 1 puisqu'il est question d'une couverture par le delta. En prenant l'espérance de (6), on obtient:

$$C_0 = e^{-rT} E(C_T)$$

Par cette équation, on se rend compte que la méthode des variables de contrôle est entièrement compatible avec la technique de la détermination des prix dans un univers neutre au risque.

L'équation (6) peut être réécrite en ignorant η^j :

$$C_0^j = e^{-rT} (C_T^j + cv^j)$$

C'est là le résultat d'une simulation pour le prix du call avec variable de contrôle. Après M simulations, on obtient l'estimé final du prix du call:

$$\hat{C}_0 = \frac{1}{M} \sum_{j=1}^M C_0^j$$

Dans le cas de la couverture par le delta, nous avons fixé à 1 le coefficient de cv , soit β . Supposons que nous n'ayons aucune idée a priori sur la valeur de ce coefficient. Pour le calculer, nous réécrivons comme suit l'équation :

$$e^{-rT} C_T^j = C_0^j - \beta cv^j e^{-rT} - \eta^j e^{-rT} \quad (7)$$

Or, avec une notation évidente:

$$C_T^{j*} = \beta_0 + \beta_1 cv^{j*} + n^{j*} \quad j = 1, \dots, M \quad (8)$$

Dans cette équation, C_T^{j*} et cv^{j*} sont connus. Chaque variable est un vecteur de M observations, résultant des M simulations. Si $n^{j*} \sim N(0,1)$, on peut régresser, par la méthode des moindres carrés ordinaires, C_T^{j*} sur cv^{j*} pour obtenir un estimé de $\hat{\beta}_1$ pour le coefficient de cv , soit la variable de contrôle¹⁴. On note, en comparant les équations (7) et (8), que $\hat{\beta}_0$ est un estimé du prix du call, qui fait l'objet de la simulation. Cet estimé peut être comparé avec le résultat de la simulation pour plus d'assurance.

Après quelques opérations élémentaires, le terme entre crochets de l'équation (5), qui fait figure de variable de contrôle, peut être exprimé comme suit¹⁵:

$$cv = \sum_{i=0}^{N-1} \frac{\partial C_{t_i}}{\partial S} [S_{t_{i+1}} - E(S_{t_i})] e^{r(T-t_{i+1})}$$

Le terme entre crochets dans cette équation représente évidemment une martingale à base delta.

3. Les nombres quasi-aléatoires et la simulation de Monte Carlo¹⁶

L'introduction des nombres quasi aléatoires dans la simulation de Monte Carlo vise à éviter que les nombres aléatoires générés ne se présentent en grappes, c'est-à-dire en séries de nombres rapprochés les uns des autres, ce qui nuit à l'efficacité de la simulation de Monte Carlo. Au lieu de faire appel à la distribution uniforme pour générer des variables aléatoires normales, on fait appel à des séquences qui balaient plus rapidement l'espace compris entre 0 et 1. On recourt ensuite à une fonction cumulative inverse qui nous régurgite des variables normales. Dans ce qui suit, nous inverserons la fonction cumulative normale pour y arriver. Mais il existe des techniques plus sophistiquées pour générer des variables normales¹⁷.

¹⁴ Certes, des techniques de régression plus robustes peuvent être utilisées, tel le GMM.

¹⁵ Pour cette transformation du terme entre crochets, voir: Clewlow et Strickland (1998), p. 93.

¹⁶ Nous nous inspirons fortement de Brandimarte (2002) pour cette section.

¹⁷ A titre d'exemple, Jackson et Staunton (2001) utilisent la technique de Moro pour générer des variables normales à partir de la séquence de Faure. Par ailleurs, Brandimarte (2002) fait appel à l'algorithme de Box-Muller pour générer des variables normales à partir des nombres d'Halton.

Pour introduire les nombres quasi-aléatoires, nous faisons appel à la séquence de Fauré qui convertit des nombres en base 10 en nombres à base 2¹⁸. Le tableau 6, emprunté à Jackson et Staunton (2001), fournit un programme en langage Visual Basic de nature à générer la séquence de Fauré.

Tableau 6 Programme Visual Basic du calcul d'une séquence de Fauré

Function FaureBase2(n) As Double

```

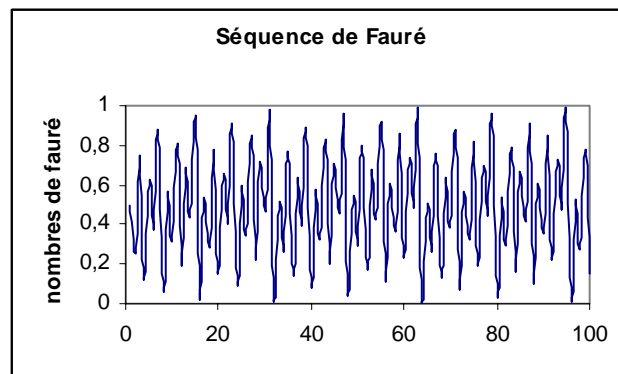
Dim f As Double, sb As Double
Dim i As Integer, n1 As Integer, n2 As Integer
n1 = n
f = 0
sb = 1 / 2
Do While n1 > 0
    n2 = Int(n1 / 2)
    i = n1 - n2 * 2
    f = f + sb * i
    sb = sb / 2
    n1 = n2
Loop
FaureBase2 = f
End Function

```

A partir de la fonction FaureBase2, nous générons 100 nombres aléatoires compris entre 0 et 1. Le résultat apparaît à la figure 6. Il faut noter que la séquence des nombres de Fauré, comme celle des autres catégories de séquences de nombres pseudo-aléatoires, est complètement déterministe. C'est-à-dire qu'en reprenant le calcul précédent pour n variant de 1 à 100, nous obtiendrons exactement les mêmes nombres. Pour un nombre d'itérations donné, la simulation de Monte Carlo du prix d'une option qui fait appel aux nombres de Fauré donnera donc toujours le même résultat. C'est pourquoi l'on parle dans ce cas de «Quasi Monte Carlo» (QMC).

¹⁸ On consultera par exemple le site de Mathworld pour la notion de base : <http://mathworld.wolfram.com>.

Figure 6



Comme on peut le constater à la figure 6, la séquence couvre bien la surface comprise entre 0 et 1. Nous nous servons de ces nombres pour calculer le prix d'un call européen classique dont les paramètres apparaissent au tableau 2. Nous faisons dans un premier temps appel à la simulation de Monte Carlo classique pour calculer ce prix, puis à la QMC basée sur la séquence de Fauré. Les programmes Visual Basic respectifs apparaissent au tableau 7.

Tableau 7 Fonctions Visual Basic du calcul du prix d'un call européen classique par la méthode de Monte Carlo classique et par la QMC faisant appel aux nombres de Fauré.

Function *MCOption*(*iopt*, *S*, *X*, *r*, *q*, *t*, *sigma*, *nsim*)

Dim i As Integer

rnmult = (*r* - *q* - 0.5 * *sigma* ^ 2) * *t*

sigt = *sigma* * Sqr(*t*)

sum = 0

For i = 1 To nsim

aleatoire = Application.NormSInv(Rnd)

s1 = *S* * Exp(*rnmult* + *aleatoire* * *sigt*)

sum = *sum* + Application.Max(*iopt* * (*s1* - *X*), 0)

Next i

MCOption = Exp(-*r* * *t*) * *sum* / *nsim*

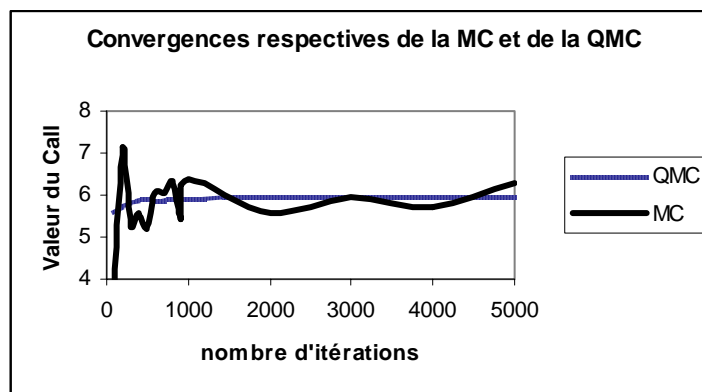
End Function

Function QMCOptionFaure(iopt, S, X, r, q, t, sigma, nsim)

```
Dim i As Integer, iskip As Integer
rnmult = (r - q - 0.5 * sigma ^ 2) * t
sigt = sigma * Sqr(t)
iskip = (2 ^ 4) - 1
sum = 0
For i = 1 To nsim
    aleatoire = Application.NormSInv(FaureBase2(i + iskip))
    s1 = S * Exp(rnmult + aleatoire * sigt)
    sum = sum + Application.Max(iopt * (s1 - X), 0)
Next i
QMCOptionFaure = Exp(-r * t) * sum / nsim
End Function
```

A la figure 7, nous étudions les convergences respectives des deux méthodes de simulation. Comme on peut le constater, la vitesse de convergence de la QMC est beaucoup plus rapide que celle de la MC classique. Le prix théorique du call tel que calculé par la formule de Black et Scholes est ici de 5,98 \$. On est à même de constater que la méthode du QMC atteint presque asymptotiquement sa cible tandis que la MC classique fluctue beaucoup avec l'augmentation du nombre d'itérations et est encore loin de sa cible après 5000 itérations tandis que la QMC l'a alors presque atteinte.

Figure 7



Les nombres quasi-aléatoires sont donc une autre technique pour accélérer la convergence d'une simulation de Monte Carlo. Pour fixer les idées, considérons l'exemple suivant qui

représente une application des nombres quasi-aléatoires de Sobol¹⁹. Quelques mots auparavant sur les nombres de Sobol avant de formuler l'application qui nous intéresse. Les nombres de Sobol sont initialement construits à partir d'un ensemble d'entiers dans l'intervalle $[1, 2^b - 1]$, où b est un paramètre représentant le nombre de bits : *binary digit* et est généralement fixé à 32. Nous nommons le $n^{\text{ième}}$ tirage d'un tel nombre entier de Sobol dans la dimension k par x_{nk} . La conversion finale à une variable uniforme y_{nk} appartenant à l'intervalle $(0, 1)$ est effectuée par l'opération:

$$y_{nk} = \frac{x_{nk}}{2^b}, \quad y_{nk} \in (0, 1), \quad x_{nk} \in Z[1, 2^b - 1]$$

Par construction, la seule variable de Sobol qui peut être nulle correspond au 0^{ième} tirage et cela pour toute dimension. Pour chaque dimension d , la base pour générer les nombres est donnée par un ensemble nommé «entiers» de direction «direction integers». Il en existe un pour la représentation binaire de chaque b bits de l'entier. En effet, la clef pour générer des nombres de Sobol réside dans le calcul des entiers de direction. Ce calcul implique des coefficients binaires d'un polynôme primaire de module 2 pour chaque dimension. Ce polynôme, de degré g_k , prend la forme suivante :

$$p_k(z) = \sum_{j=0}^{g_k} a_{kj} z^{g_k - j}$$

où les a sont les coefficients en question. Finalement, les nombres de Sobol peuvent être obtenus en effectuant l'opération:

$$x_{nk} = \sum_{j=1}^d v_{kj} 1$$

Considérons l'exemple suivant²⁰ concernant la génération d'une séquence de Sobol en une dimension. Nous voulons intégrer la fonction:

$$\int_0^1 \int_0^1 f(x, y) dx dy = \int_0^1 \int_0^1 e^{-xy} (\sin 6\pi x + \cos 8\pi y) dx dy$$

Nous savons qu'une intégrale d'une fonction du type $I = \int_0^1 f(x) dx$ peut être vue comme une espérance. En effet, il s'agit d'approximer l'espérance $E[f(U)]$, où U représente la loi uniforme sur l'intervalle d'intégration $(0, 1)$. Plus précisément, l'intégrale I peut être approximée

¹⁹ Sur les nombres de Sobol, on consultera Brandimarte (2002), Jäckel (2002) et Glasserman (2003).

²⁰ Cet exemple est une adaptation de Brandimarte (2003).

par : $\hat{I}_n = \frac{A}{n} \sum_{i=1}^n f(x_i)$ où A^{21} est l'aire (surface) sous la courbe dans la région d'intégration. En général,

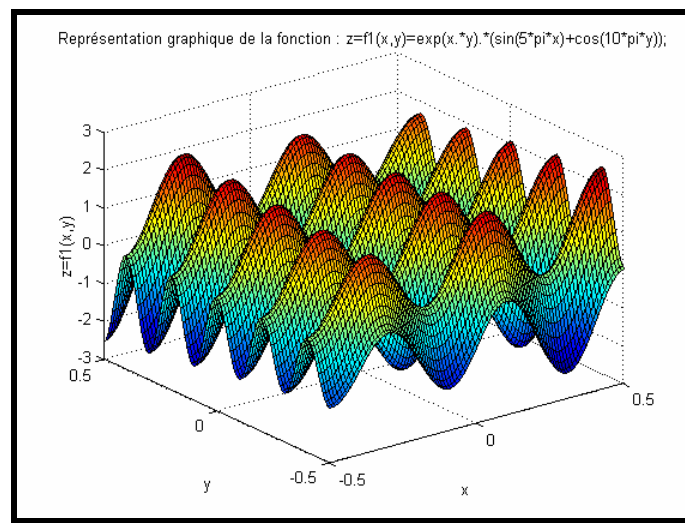
la surface ou le volume utilisés sont unitaires de sorte que $A = 1$. Cette intégrale peut être alors approximée par la moyenne de f évaluée aux points x_i en générant ces valeurs à partir de la loi U sur l'intervalle $(0,1)$. Donc, pour résoudre notre problème, il suffit simplement de générer des x et des y entre 0 et 1, les bornes d'intégration, et calculer la valeur moyenne de la fonction à ces divers points. Le code Matlab de la fonction est présenté au tableau 8.

Tableau 8 Code Matlab de la fonction à intégrer

```
function fl=f(x,y)
fl=exp(-x.*y).*(sin(5*pi*x)+cos(10*pi*y));
```

Pour tracer le graphique de $f(x,y)$, on utilise la commande Matlab : `surf(x,y,z)` où $z=f(x,y)$ et où les x et y sont générés par la commande `[x,y]=meshgrid(-0.5:0.01:0.5,-0.5:0.01:0.5)`. Cette fonction est représentée à la figure 8.

Figure 8



Il suffit maintenant de générer les x et les y que nous nommerons $S1$ et $S2$. Il faut également définir le polynôme ' p ', donner des valeurs de départ ' $m0$ ' et identifier le nombre ' n ' de nombres de direction pour enclencher l'algorithme utilisé lors de cet exercice.

```
P=[1 0 1 1 1 1 1 1 1 1 1];
m0=[ 1 3 5 9 11 13 15 17 19 21];
```

²¹ Il faut générer des points afin de couvrir l'ensemble de la surface A . Si le problème à résoudre est un volume, alors il faut générer des points afin de couvrir ce volume correctement.

```

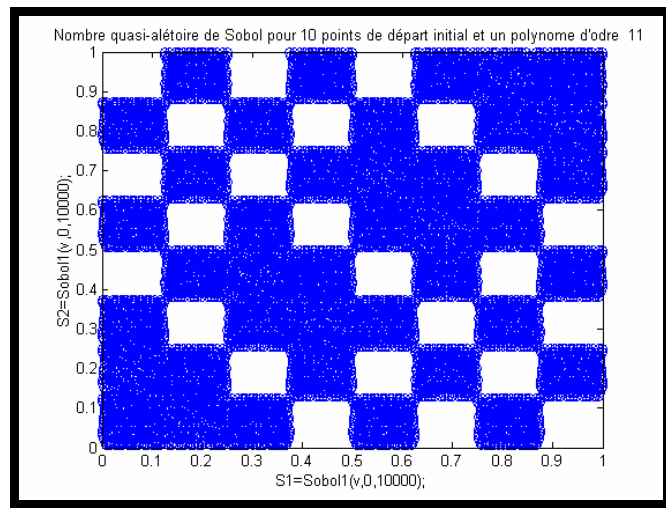
[v,m]=nombredirection(p,m0,500);
S1=SobolI(v,0,10000);

p=[1 0 0 0 1 1 1 1 1 0 0 0 1 0 1];
m0=[ 1 3 5 9 11 13 15];
[v,m]=nombredirection(p,m0,500);
S2=SobolI(v,0.233,10000);
plot(S1,S2,'o')

```

A la figure 9 se retrouve une représentation bidimensionnelle de nombres Sobol. On voit que la surface comprise entre 0 et 1 est assez bien couverte. Les figures 10 et 11 montrent une moins bonne couverture.

Figure 9



Dans Matlab, on écrit la fonction suivante et on tape *enter*.

```
Mean(f1(S1,S2))
```

```
ans =
```

```
0.0234
```

Cette valeur est rapprochée de celle obtenue pour une quadrature habituelle qui est de 0.0199. Les figures 10, 11 et 12 illustrent un mauvais remplissage de l'espace à couvrir. En effet, nous pourrions calculer l'intégrale précédente et nous serions à même de constater que la valeur obtenue est éloignée du résultat recherché. Par cet exemple, on comprend l'importance des conditions initiales sur l'échantillonnage des nombres quasi-aléatoires à l'aide de la méthode de Sobol. On prend ainsi acte de

la faiblesse de ladite méthode. En effet, une simple modification des vecteurs initiaux donne lieu à un résultat erroné. Nous allons refaire le même exercice avec les nombres d'Halton afin de bien illustrer le sujet moderne des simulations Quasi-Monte Carlo.

Figure 10

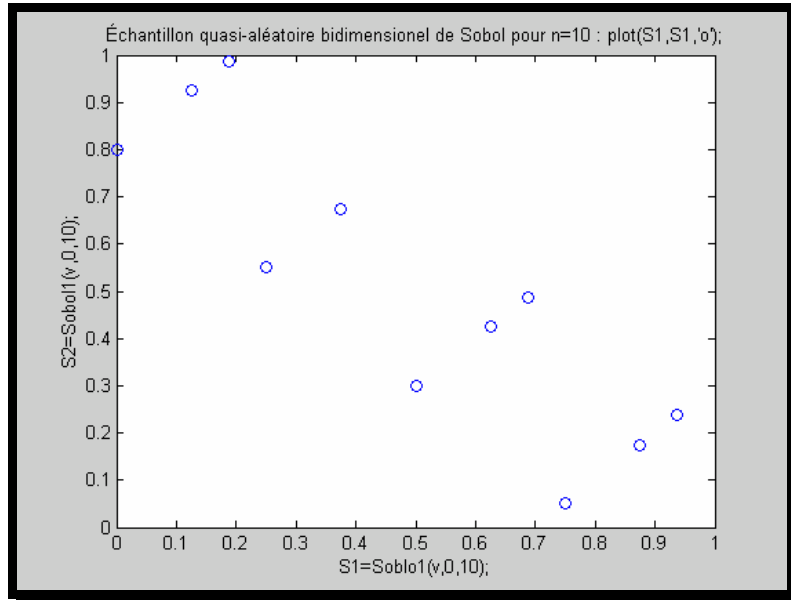


Figure 11

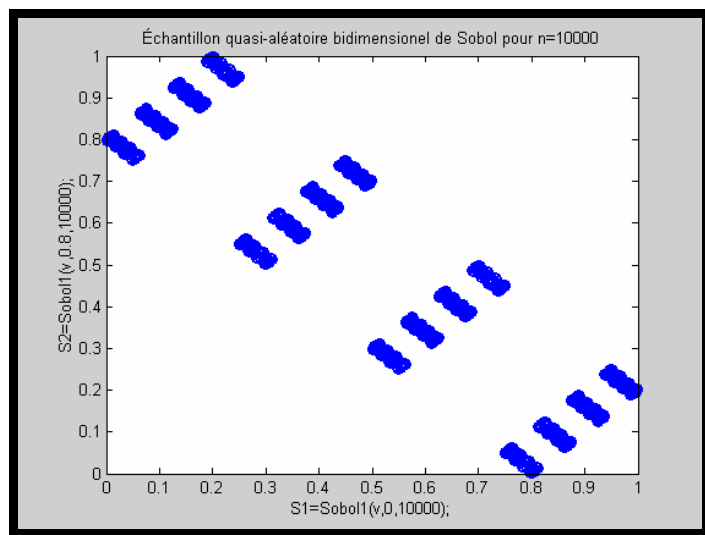
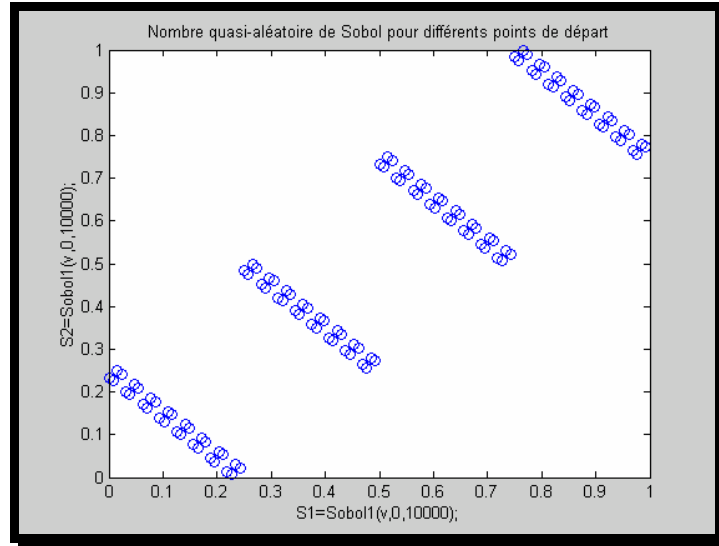


Figure 12



Au tableau 9, on retrouve une fonction Matlab empruntée à Brandimarte (2002) pour générer des nombres d'Halton et à la figure 13, une représentation graphique de ces nombres. Il est à remarquer que la base doit correspondre à un nombre premier²².

Tableau 9 Fonction Matlab pour générer des nombres d'Halton

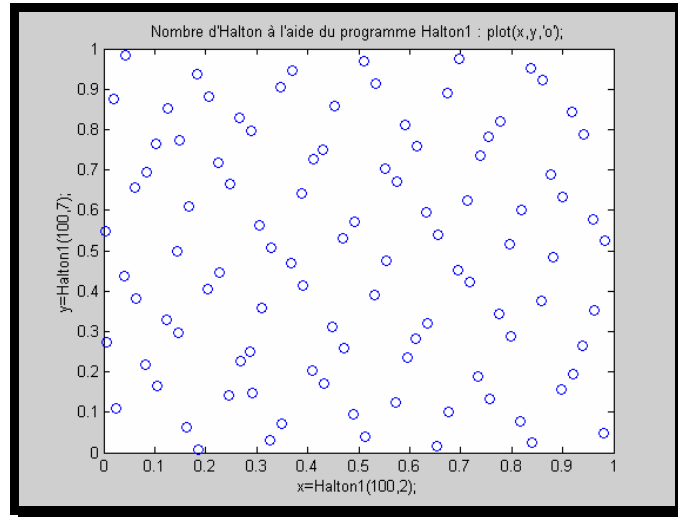
function Halton=Halton1(n,b)

```
Halton=zeros(n,1);
nbits=1+ceil(log(n)/log(b));
vb=b.^(-(1:nbits));
wv=zeros(1,nbits);
for i=1 :n
    % incrémentation de la dernière 'bit : binary digit'
    j=1;
    ok=0;
    while ok==0;
        wv(j)=wv(j)+1;
        if wv(j)<b
            ok=1;
        else
            wv(j)=0;
            j=j+1;
        end
    end
    Halton(i)=wv./vb;
end
```

²² Un nombre premier est un entier supérieur à 1 qui n'est divisible que par 1 et par lui-même.

```
end
end
Halton(i)=dot(wv,vb);
End
```

Figure 13



Nous appliquons maintenant la technique des nombres d'Halton au calcul du prix d'une option asiatique. Le tableau 10 fournit des programmes écrits en langage Matlab pour calculer le prix d'une option asiatique à partir de la méthode du quasi-Monte Carlo qui intègre des sentiers de prix générés par des nombres d'Halton. Ces fonctions sont également empruntées à Brandimarte (2003). Nous avons cependant modifié sa fonction pour générer des sentiers d'Halton du prix du sous-jacent. Nous avons eu recours à l'inversion de la fonction normale cumulative pour générer des variables aléatoires normales à partir des nombres d'Halton et non à l'algorithme de Box-Muller comme c'est le cas chez Brandimarte. Nous avons également combiné la technique des variables antithétiques à celle des nombres d'Halton comme techniques de réduction de la variance.

Tableau 10 Fonctions Matlab du calcul du prix d'une option asiatique par la méthode du QMS et des nombres d'Halton.

```

function P=AsianHalton3(S0,X,r,T,sigma,Npas,Niter) ;
Payoff=zeros(Niter,1) ; % Calcul du 'Payoff'
Path=HaltonPaths3(S0,r,sigma,T,Npas,Niter) ;
Payoff=max(0,mean(Path(:,2 :Npas+1)),2)-X) ; % Calcul du 'Payoff'
P=mean(exp(-r*T)*Payoff) ;

```

```

function Spaths=HaltonPaths3(S0,r,sigma,T,Npas,Niter)
dt=T/Npas ;

```

```

nudt=(r-0.5*sigma^2)*dt ;
sidt=sigma*sqrt(dt) ;
Niter=2*ceil(Niter/2) ;

```

```

RandMat=zeros(Niter,Npas) ;
seeds=myprimes(2*Npas) ;
Base1=seeds(1 :Npas) ;
Base2=seeds((Npas+1) :2*Npas) ;
for i=1 :Npas
    H1=GetHalton(Niter/2,Base1(i)) ;

```

```

    Norm1=norminv(H1) ;
    Norm2=-norminv(H1) ;
    RandMat(:,i)=[Norm1 ;Norm2] ;

```

```

end
Increments=nudt+sidt*RandMat ;
LogPaths=cumsum([log(S0)*ones(Niter,1),Increments],2) ;
Spaths=exp(LogPaths) ;

```

```

function Seq=GetHalton1(HowMany,Base)

```

```

Seq=zeros(HowMany,1) ;
NumBits=1+ceil(log(HowMany)/log(Base)) ;
VetBase=Base.^(-(1 :NumBits)) ;
WorkVet=zeros(1,NumBits) ;
for i=1 :HowMany
    % incrémentation de la dernière 'bit : binary digit'
    j=1 ;
    ok=0 ;
    while ok==0 ;
        WorkVet(j)=WorkVet(j)+1 ;
        if WorkVet(j)<Base
            ok=1 ;
        else
            WorkVet(j)=0 ;
            j=j+1 ;
        end
    end
    Seq(i)=dot(WorkVet,VetBase) ;
end

```

```

function p=myprimes(N)

```

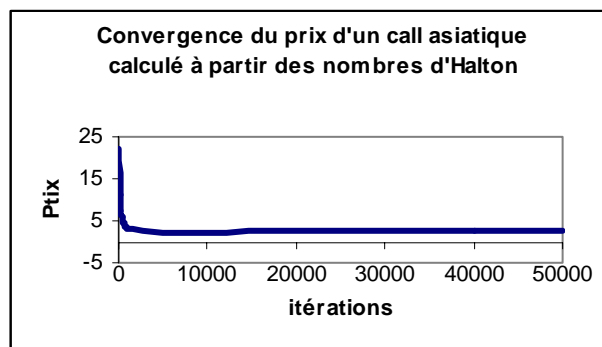
```

found=0 ;
trynumber=2 ;
p=[] ;
while (found<N)
    if isprime(trynumber)
        p=[p,trynumber] ;
        found=found+1 ;
    end
    trynumber=trynumber+1 ;
end

```

Nous avons repris donc le même calcul que précédemment à partir des fonctions du tableau 10, c'est-à-dire calculer le prix d'un call européen asiatique dont les paramètres sont compilés au tableau 2. Nous savons que le prix effectif de ce call est de 2,48\$. La figure 14 montre comment s'effectue la convergence lorsque le nombre de pas est fixé à 50. Comme on peut le constater, la convergence est somme toute régulière.

Figure 14



Conclusion

Une simulation de Monte Carlo se révèle très utile lorsque le problème à solutionner comporte plusieurs dimensions. Par exemple, l'option asiatique comporte une dimension de plus que l'option classique de Black et Scholes puisqu'elle dépend du sentier suivi par son sous-jacent (*path-dependent*). L'équation différentielle d'une option asiatique comporte donc un terme additionnel en regard de l'option classique, terme qui est relié à sa dépendance du sentier suivi par le sous-jacent. Comme nous le verrons dans cet article, la simulation de Monte Carlo est en mesure de s'attaquer à cette nouvelle dimension de l'option.

Toutefois, une simulation de Monte Carlo qui n'est pas soumise à une technique de réduction de la variance peut donner des résultats insatisfaisants même si le nombre d'itérations est poussé jusqu'au million. Nous avons examiné dans cet article comment diverses techniques de réduction de la variance permettaient de converger beaucoup plus rapidement vers le prix de l'option asiatique étudiée. Nous avons alors distingué la simulation de Monte Carlo proprement dite, dont les résultats se modifient d'une simulation à l'autre, et la Quasi Monte Carlo qui fait appel aux nombres pseudo aléatoires et qui donne toujours le même résultat pour un nombre d'itérations donné. Nous avons constaté que les nombres pseudo-aléatoires comportaient des forces et des faiblesses. Leur objectif est de couvrir plus rapidement la surface d'intégration, objectif qu'ils ne réalisent pas toujours. Ces nombres doivent donc être utilisés avec doigté et les bases utilisées pour concocter ces nombres doivent se plier également à certaines règles.

Bibliographie

- Boyle, P.P. (1977), Options: A Monte Carlo Approach, *Journal of Financial Economics*, p. 323-338.
- Brandimarte, P.(2002), Numerical Methods in Finance : A MATLAB-Based Introduction, Wiley.
- Clelow, L. et Caverhill, A.(1994), On the Simulation of Contingent Claims, *Journal of derivatives*, pp. 66-74.
- Clelow,L. & Strickland, C.(1998), Implementing Derivatives Models, John Wiley and Sons, Chichester.
- Clelow, L. et Strickland, C. (1997), Monte Carlo Valuation of Interest Rate Derivatives under Stochastic Volatility, *Journal of Fixed Income*, pp. 35-45.
- Cox, J.C. et S.A. Ross (1976), The Valuation of Options for Alternative Stochastic Processes, *Journal of Financial Economics*, 3, p. 145-166.
- Glasserman, P. (2003), Monte Carlo Methods in Financial Engineering, Springer-Verlag.
- Jäckel, P.(2002), Monte Carlo Methods in Finance, Chichester, Wiley.
- James, J. et Webber, N.(2000), Interest Rate Modelling, New York, John Wiley & Sons.
- Jackson, M. et Staunton, M. (2001), Advanced Modelling in Finance using Excel and VBA, Wiley.
- Racicot, F.E. et R. Théoret (2004), Le calcul numérique en finance empirique et quantitative : ingénierie financière et Excel (Visual basic), PUQ.