université de BORDEAUX

TP3 - Calcul d'option par la méthode de Monte Carlo

Mots clés :
- **Scilab**
- **Théorème central limite**
- **Marge d'erreur**
- **Intervalle de confiance**
- **Réduction de variance**
- **Variable de contrôle**
- **Variable antithétique**
- **Échantillonnage préférentiel**
- **Modèle de Black-Scholes**
- **Option vanille (européenne)**
- **Call & Put**

Élève : Boudjeltia Reda
M2 MIMSE Spé 3 AD
Date : 9 Novembre 2015

Table des matières

1 Description du protocole

Dans le cadre de ce TP nous avons à étudier la méthode de `Monte Carlo` dans le cas d'un calcul de `Put` d'une option européenne. Pour ce faire, nous devons comparer la méthode dite "classique" avec des méthodes de `Réduction de variance` qui sont :

— Variable de contrôle
— Variables antithétique
— Échantillonnage préférentiel

Ainsi, le protocole est quasi identique pour les quatre méthodes :

1. On définit une fonction g(X) (X étant une v.a dont on précisera la loi)
2. On calcul numériquement $\mathbb{E}(g(X))$ par la méthode de Monte Carlo : $\sum_{n=1}^{\infty} \frac{g(X)}{n} = \mathbb{E}(g(X))$
3. On évalue la variance de façon empirique avec l'estimateur sans biais$S_n = \frac{1}{n-1} \sum_{i=1}^{n} (g(X_i) - I_n)^2$
4. Puis évalue l'intervalle de confiance $[I_n - 1.96\sqrt{\frac{S_n}{n}}; I_n + 1.96\sqrt{\frac{S_n}{n}}]$

D'autre part, dans un premier temps nous ferons une étude en présentant les courbes pour finir par une étude comparatif des différentes méthodes. Ainsi, les méthodes seront comparé sur les critères suivant.

— Variance (réduction)
— Précision des bornes de l'intervalle de confiance
— Première valeur pour laquelle ont reste définitivement dans l'intervalle de confiance

2 Valeur exacte et Monte Carlo Simple

Ici, on a X qui suit une loi normale centrée réduite

$$g(X) = (Ke^{-rT} - xe^{\sigma X\sqrt{T} - \frac{\sigma^2 T}{2}})_+ \tag{1}$$

Voici, les évolutions des évaluations en fonction de n.

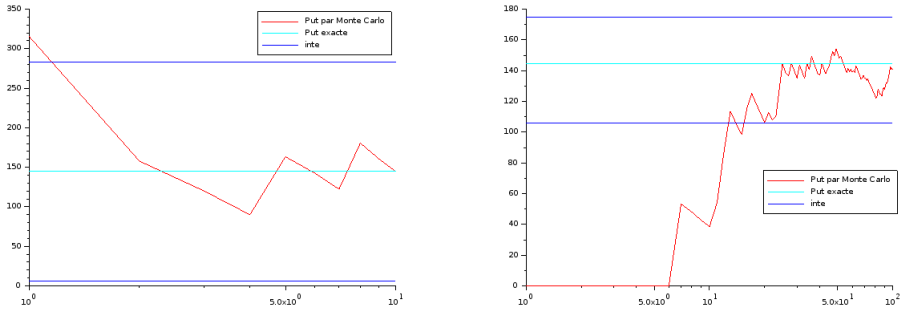

FIGURE 1 – n = 10 et n = 100

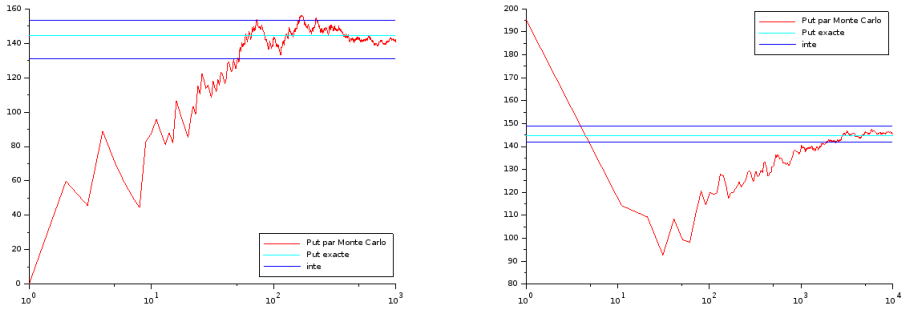

FIGURE 2 – n = 1 000 et n = 10 000

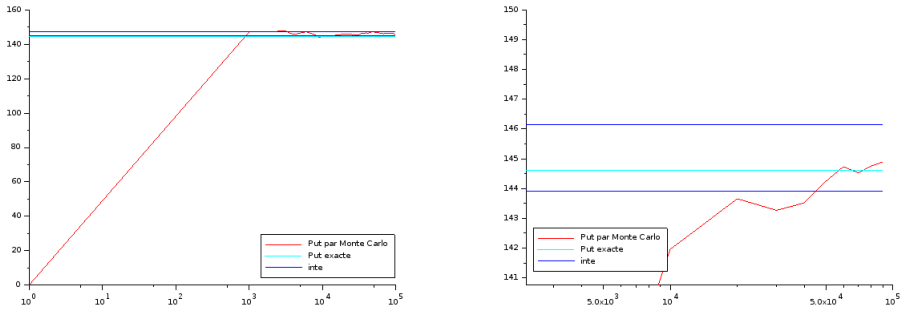

FIGURE 3 – n = 100 000 avec zoom

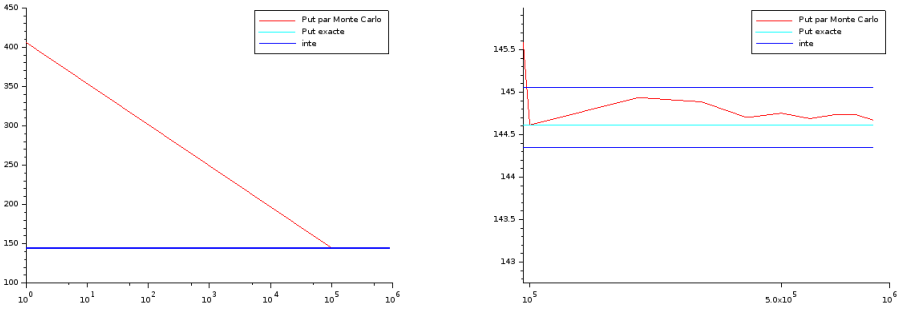

FIGURE 4 – n = 1 000 000 avec zoom

3 Variable de Contrôle

La méthode consiste à poser $\mathbb{E}[g(X)] = \mathbb{E}[g(X) - h(X)] + \mathbb{E}[g(X)]$ sous réserve que $var(g(X) - h(X)) < var(g(X))$ et que l'on sache calculer explicitement $\mathbb{E}[h(X)]$.

On a

$$C - P = x(1 - e^{-rT}) \tag{2}$$

$$\text{en posant } C = \mathbb{E}[h(X)] \tag{3}$$

$$\text{avec } h(X) = xe^{\sigma X\sqrt{T} - \frac{\sigma^2 T}{2}} - Ke^{-rT} \tag{4}$$

Ainsi

$$\begin{aligned} \mathbb{E}[g(X)] &= \mathbb{E}[g(X) - h(X)] + \mathbb{E}[h(X)] && (5)\\ &= \mathbb{E}[g(X)] - \mathbb{E}[h(X)] + \mathbb{E}[h(X)] && (6)\\ &= C - P + \mathbb{E}[h(X)] && (7) \end{aligned}$$

avec X qui suit une loi normale centré et réduite.

Néanmoins, pour le modèle de `Black & Scholes`, la variance du `Put` est souvent (mais pas systématiquement) inférieur à celle du `Call`.

Tout d'abord, nous avons fait le travail précédent mais pour un `call`. Par conséquent, on a posé

$$h(X) = (xe^{\sigma X\sqrt{T} - \frac{\sigma^2 T}{2}} - Ke^{-rT})_+ \tag{8}$$

On obtient les courbes suivant qui vérifie que nous n'avons pas fait d'erreurs.

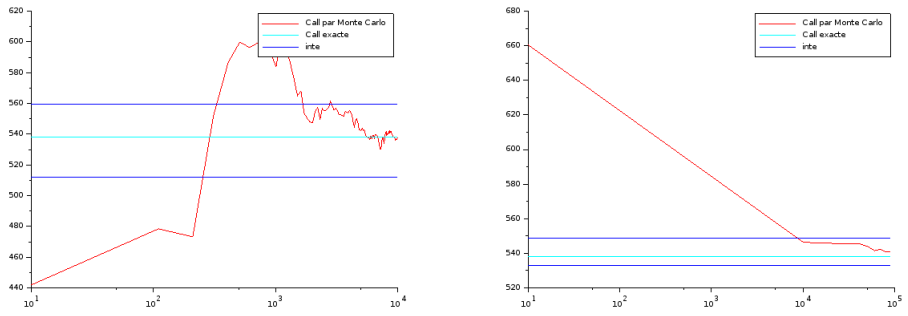

FIGURE 5 – n = 10 000 et 100 000

Ensuite, nous avons exécute nos simulations et obtenu les courbes suivant

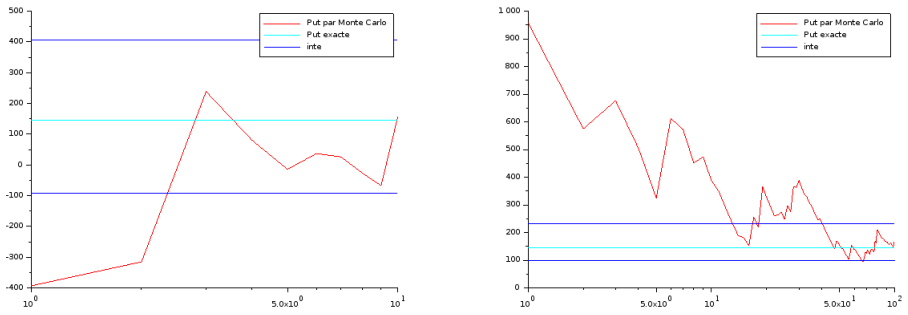

FIGURE 6 – n = 10 et 100

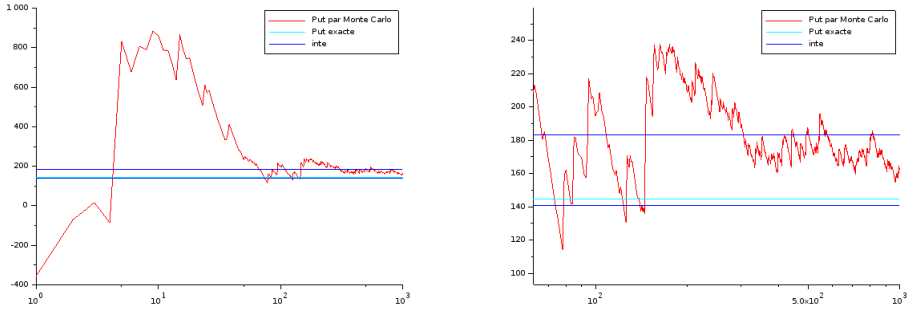

FIGURE 7 – n = 1000 avec zoom

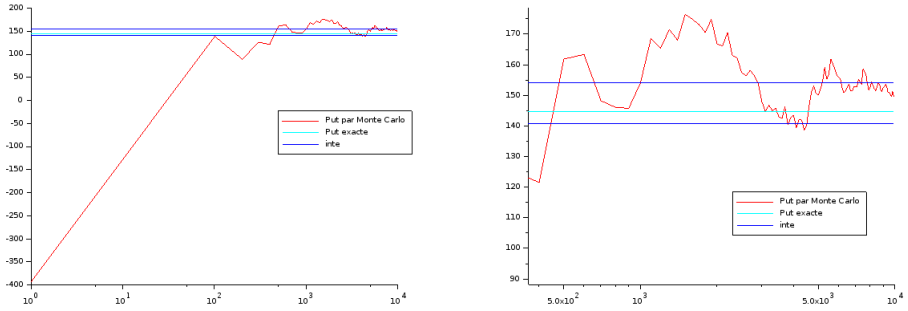

FIGURE 8 – n = 10 000 avec zoom

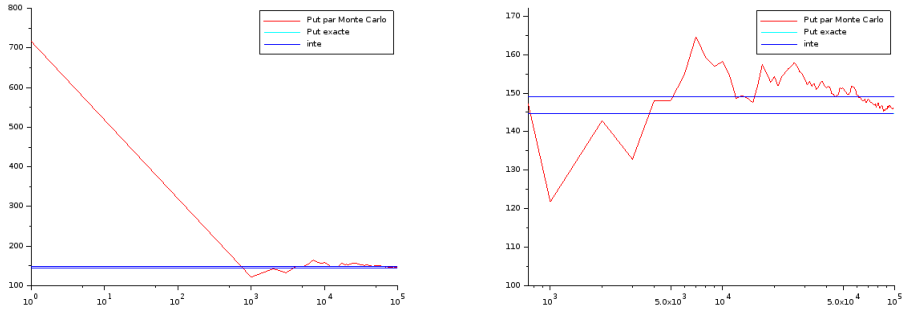

FIGURE 9 – n = 100 000 avec zoom

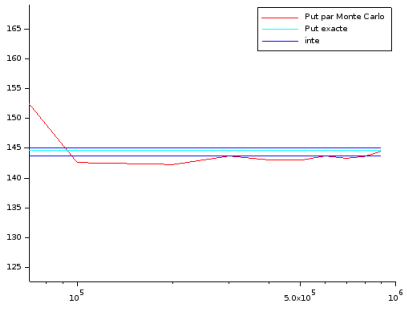

Figure 10 – n = 1 000 000 avec zoom

4 Variables antithétiques

La méthode consiste à poser $\mathbb{E}[g(X)] = \mathbb{E}[\frac{g(X)-g(h(X))}{2}]$. Ici, l'inégalité de Hölder garanti la diminution de variance.

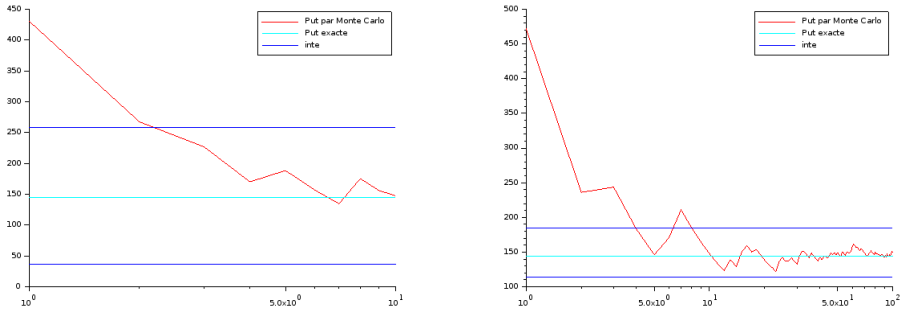

FIGURE 11 – n = 10 et 100

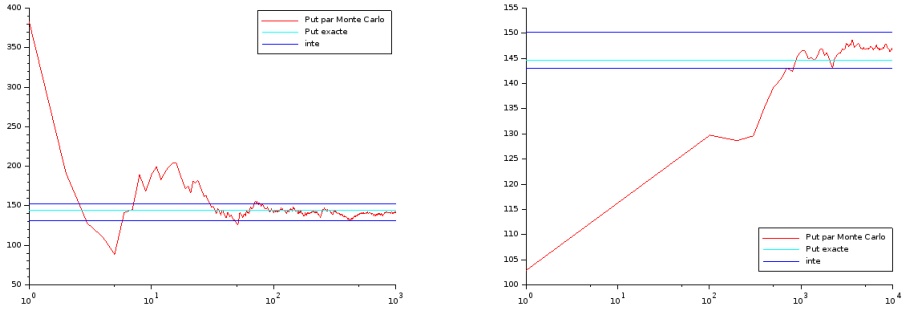

FIGURE 12 – n = 1 000 et 10 000

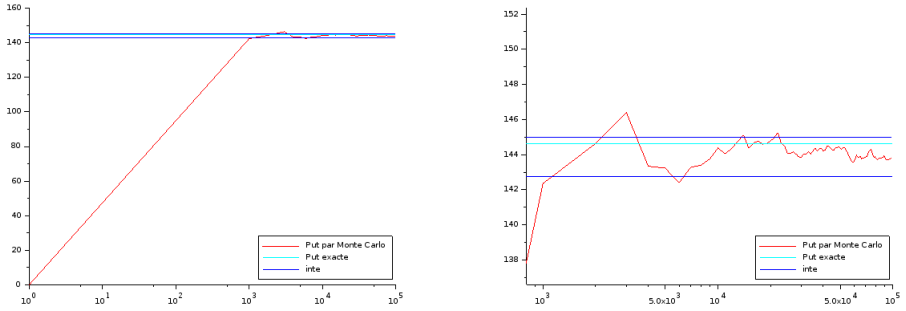

FIGURE 13 – n = 100 000 avec zoom

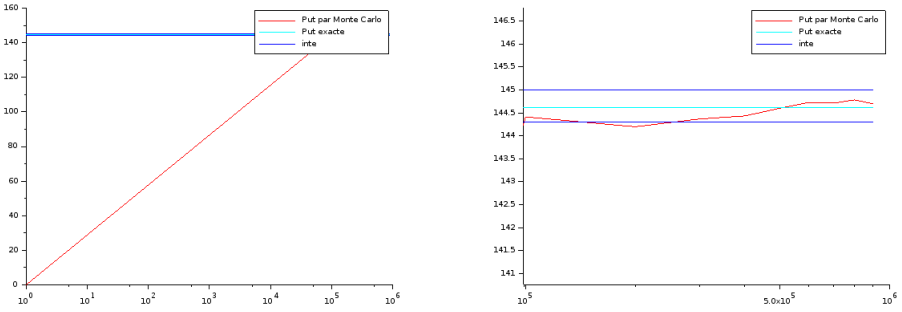

Figure 14 – n = 1 000 000 avec zoom

5 Échantillonnage préférentiel

Le but de la méthode est d'écrire I sous la forme suivante

$$I = \int g(x)f(x)dx = \int \frac{g(x)f(x)}{h(x)}h(x)dx = \mathbb{E}[\frac{\eth(\mathbb{Y})\mho(\mathbb{Y}}{\eqsim(\mathbb{Y})}] \tag{9}$$

Sous réserve que l'on sache simuler la v.a Y de densité h. La méthode s'avère intéressante si $var(\frac{gf}{h}(Y)) < var(g(X))$

Dans notre exemple on a $x = K$ et $r = \frac{\sigma^2}{2}$ alors $d_1 = \sigma\sqrt{T}$ et $d - 2 = 0$. On Obtient

$$\frac{P}{x} = \mathbb{E}[e^{-rT}(1 - e^{-\sigma\sqrt{T}X})\nVdash_{X\geq 0}] \tag{10}$$

$$= \frac{e^{-rt}}{\sqrt{2\pi}}\int_0^\infty (1 - e^{-\sigma\sqrt{T}y})e^{\frac{-y^2}{2}}dy \tag{11}$$

Or, pour x petit $e^x - 1 = x$. On pose $u = y^2/2$ et on obtient

$$P = x\frac{e^{-rt}}{\sqrt{2\pi}}\int_0^\infty \frac{1 - e^{-\sigma\sqrt{T}\sqrt{2u}}}{\sqrt{2u}}e^{-u}dy \tag{12}$$

$$= x\frac{e^{-rt}}{\sqrt{2\pi}}\mathbb{E}[\frac{1 - e^{-\sigma\sqrt{2TU}}}{\sqrt{2U}}] \tag{13}$$

avec T qui suit une loi exponentielle de paramètre 1.

De plus, on peut remarquer sur les graphiques suivants que la méthode donne des résultats précis très vite.

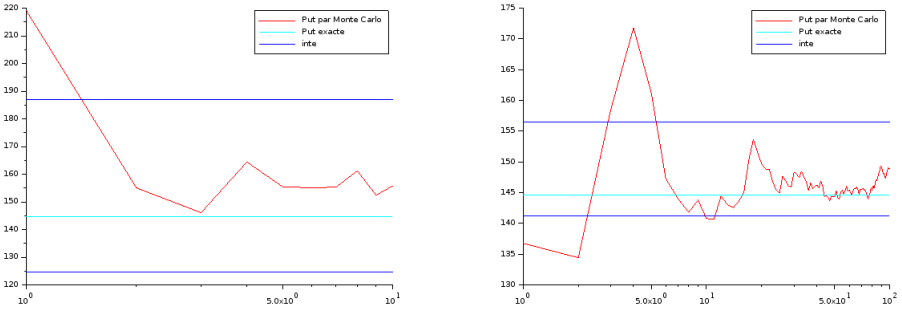

FIGURE 15 – n = 10 et n = 100

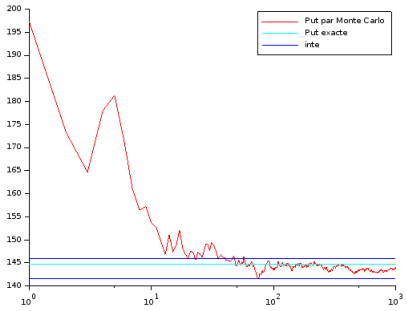

FIGURE 16 – n = 1 000

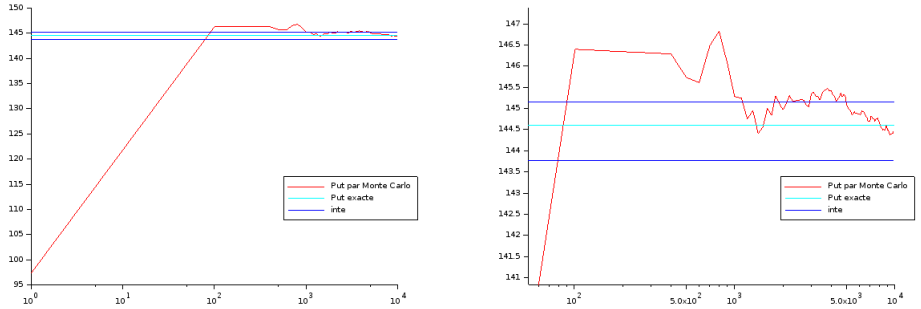

FIGURE 17 – n = 10 000 avec zoom

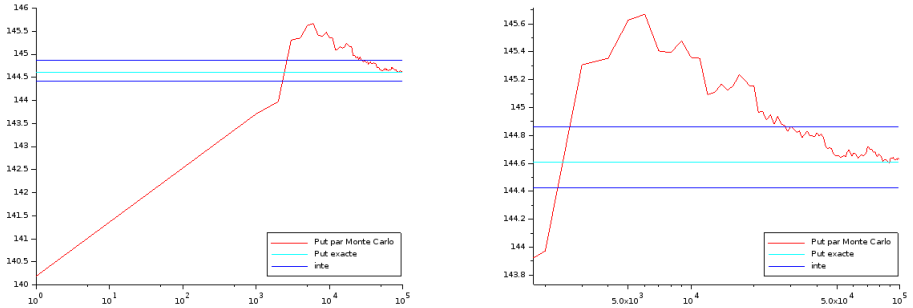

FIGURE 18 – n = 100 000 avec zoom

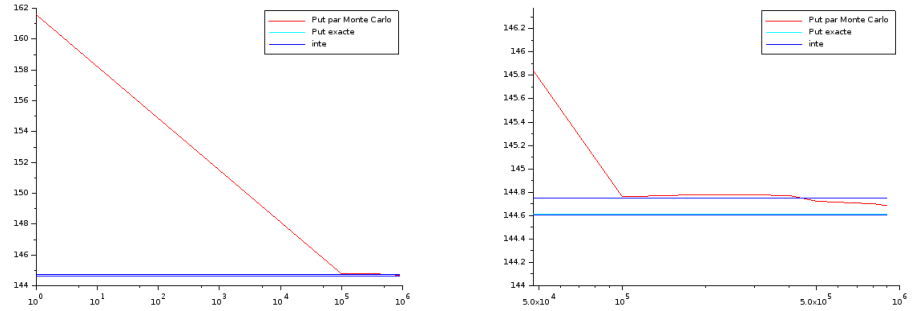

FIGURE 19 – n = 1 000 000 avec zoom

6 Bilan

Pour chaque simulation nous avons relevé les évaluation suivante
— Variance (réduction)
— Précision des bornes de l'intervalle de confiance
— première valeur pour laquelle ont reste dans l'intervalle de confiance
Notons, ensuite le nom des méthodes
— Méthode 1 : Monte Carlo simple
— Méthode 2 : Variable de contrôle
— Méthode 3 : Variable antithétique
— Méthode 4 : Échantillonnage préférentiel

Méthode 1

N	Variance	Borne Sup	Borne Inf	Entrée
10	30238,741	320	66	2
100	26802,24	165,75	98,89	12
1000	32623,83	153,43	131,095	52
10000	32516,76	148,95	141,88	1928
100000	32710,03	146,15	143,91	45482
1000000	32431,89	145,05	144,38	101026

Méthode 2

N	Variance	Borne Sup	Borne Inf	Entrée
10	161834,69	406,25	-92	3
100	117046,22	231,62	97,51	41
1000	118341,29	183,41	140,75	579
10000	11825,32	154,21	140,73	7887
100000	117356,54	148,96	144,71	63940
1000000	116029,7	145,01	143,67	794690

Méthode 3

N	Variance	Borne Sup	Borne Inf	Entrée
10	31799,25	257,97	36,92	2
100	32462,65	184,57	113,94	8
1000	31814,306	152,93	130,82	152
10000	32766,74	150,21	143,11	780
100000	32329,29	144,9	142,76	6274
1000000	32414,46	145,01	144,29	261155

Méthode 4

N	Variance	Borne Sup	Borne Inf	Entrée
10	2543,1	187,07	124,55	2
100	1528,39	156,52	141,2	6
1000	1235,63	145,94	141,58	58
10000	1235,19	145,15	143,77	4911
100000	1256,08	144,86	144,42	29065
1000000	1255,94	144,74	144,61	429909

FIGURE 20 – Relevé de nos tests

Par conséquent, on constate que la méthode de variable de contrôle donne une grande variance (dont la variance me parait trop grande, dû peut être une erreur de calcul) et que celle de l'échantillonnage préférentiel une très réduite comparé aux autres.

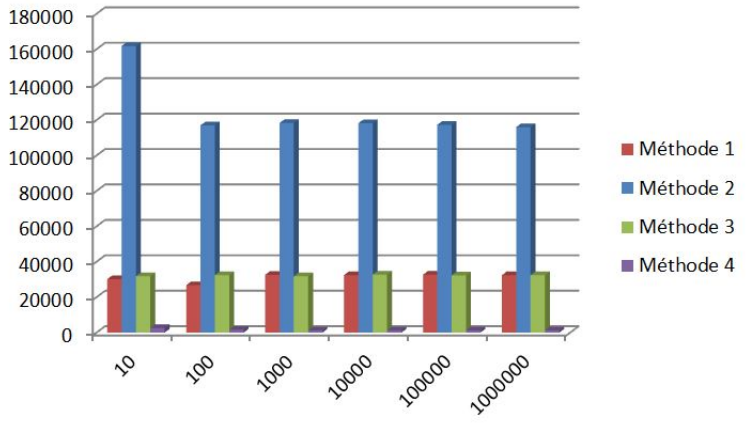

FIGURE 21 – Variances des méthodes

De plus, cela se confirme quand on observe les comparaisons de convergence des bornes de l'intervalle de confiance vers la vrai valeur du Put (rappelons qui est à 144.61).

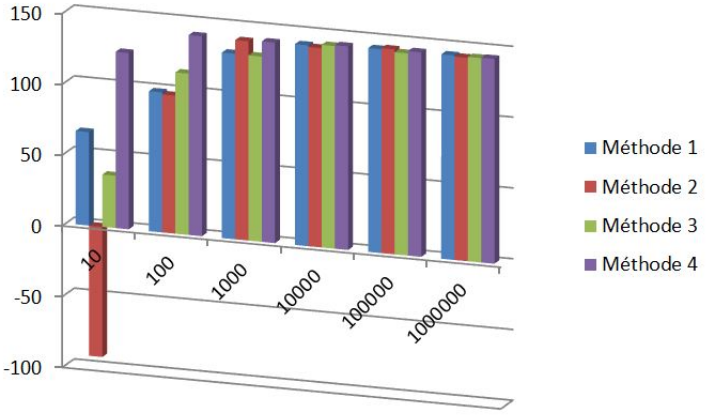

FIGURE 22 – valeur de la borne inférieur l'intervalle de confiance

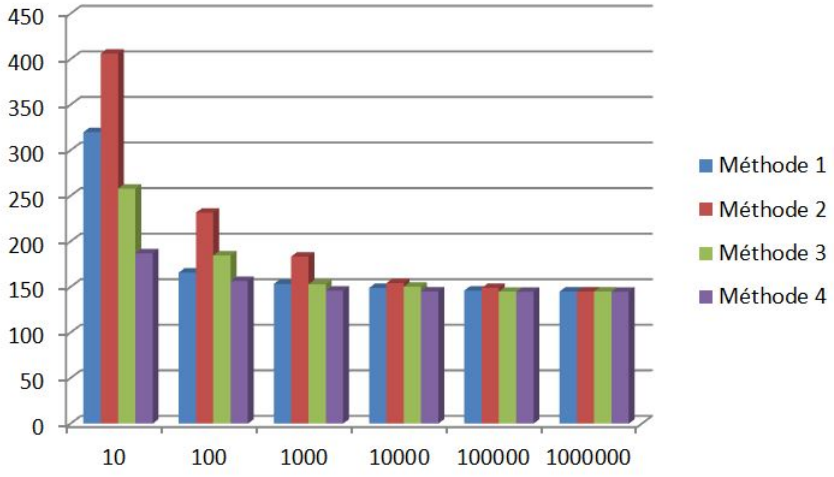

FIGURE 23 – valeur de la borne supérieur de l'intervalle de confiance

7 Code complet

```
//Rappel
function [p] = d1(x,t,K,T,r,sigma)
    a = log(x/K)+(r+sigma**2/2)*(T-t)
    p = a /(sigma*sqrt(T-t))
endfunction

test = d1(10,100,100,30,0.05,0.1)

function [p] = d2(x,t,K,T,r,sigma)
    p =  d1(x,t,K,T,r,sigma) - sigma*sqrt(T-t)
endfunction

test = d2(10,100,100,30,0.05,0.1)

function [p] = Call(x, t, T, K, r, sigma)
    first = x*cdfnor("PQ",d1(x,t,K,T,r,sigma),0,1);
    second = K*exp(-r*(T-t))*cdfnor("PQ",d2(x,t,K,T,r,sigma),0,1);
    p = first - second;
endfunction


function [p] = Put(x, t, T, K, r, sigma)
    first = x*cdfnor("PQ",-d1(x,t,K,T,r,sigma),0,1);
    second = K*exp(-r*(T-t))*cdfnor("PQ",-d2(x,t,K,T,r,sigma),0,1);
    p = -first + second;
endfunction

//-----------------------------------------------------------
t = 0;
x = 1000;
T = 100;
K = 1000;
r = 0.005;
sigma = 0.1;

n = 1000;
X = grand(n,1,"nor",0,1);

function [p] = g(X, x, t, T, K, r, sigma)
    first = K*exp(-r*(T-t));
    second = x*exp(sigma*X*sqrt(T-t)) *exp(-((T-t)*sigma**2)/2)
    p = max(0, first - second);
endfunction
```

Figure 24 – Simple simulation du Put 1/3

```
test = g(X(1), x, t, T, K, r, sigma);

//evaluation with Monte Carlo method
function [p] = eval_Put(X, x, t, T, K, r, sigma, n)
    tmp = 0;
    for i = 1:n
        tmp = tmp + g(X(i), x, t, T, K, r, sigma)/n;
    end
    p = tmp
endfunction


function [p]= error_Rate(X, x, t, T, K, r, sigma, n)
    tmp = 0;
    I = eval_Put(X, x, t, T, K, r, sigma, n);
    for i = 1:n
        tmp = tmp + (g(X(i), x, t, T, K, r) - I)**2/(n-1);
    end
    p = tmp;
endfunction


function [p] = boundary(X, x, t, T, K, r, sigma, n)
        I = eval_Put(X, x, t, T, K, r, sigma, n);
        A = error_Rate(X, x, t, T, K, r, sigma, n)

        p = [I - 1.96*sqrt(A/n), I + 1.96*sqrt(A/n)];
endfunction

function [p] = test1(X, x, t, T, K, r, sigma, y)
    for i = 1:length(y)
        a(i) = eval_Put(X, x, t, T, K, r, sigma, y(i));
    end
    p = [a];
endfunction


function [p] = test2(X, x, t, T, K, r, sigma, y)
    for i = 1:length(y)
        b(i) = Put(x, t, T, K, r, sigma);
    end
    p = [b];
endfunction


function [p] = test3(X, x, t, T, K, r, sigma, y,n)
    b = boundary(X, x, t, T, K, r, sigma, n)
    for i = 1:length(y)
        c(i) = b(1);
    end
    p = [c];
endfunction
```

Figure 25 – Simple simulation du Put 2/3

```
function [p] = test4(X, x, t, T, K, r, sigma, y, n)
    b = boundary(X, x, t, T, K, r, sigma, n)
    for i = 1:length(y)
        c(i) = b(2);
    end
    p = [c];
endfunction

//Simulation
n = 10
y = [1:1:n];
X = grand(n,1,"nor",0,1);

//clf();
//plot2d(y, test1(X, x, t, T, K, r, sigma, y), logflag = "ln", style=5);
//plot2d(y, test2(X, x, t, T, K, r, sigma, y), logflag = "ln", style=4);
//plot2d(y, test3(X, x, t, T, K, r, sigma, y, n), logflag = "ln", style=2);
//plot2d(y, test4(X, x, t, T, K, r, sigma, y, n), logflag = "ln", style=2);
//legend(["Put par Monte Carlo";"Put exacte";"inte"])
```

Figure 26 – Simple simulation du Put 3/3

```
//Rappel
function [p] = d1(x,t,K,T,r,sigma)
    a = log(x/K)+(r+sigma**2/2)*(T-t)
    p = a /(sigma*sqrt(T-t))
endfunction

test = d1(10,100,100,30,0.05,0.1)

function [p] = d2(x,t,K,T,r,sigma)
    p =  d1(x,t,K,T,r,sigma) - sigma*sqrt(T-t)
endfunction

test = d2(10,100,100,30,0.05,0.1)

function [p] = Call(x, t, T, K, r, sigma)
    first = x*cdfnor("PQ",d1(x,t,K,T,r,sigma),0,1);
    second = K*exp(-r*(T-t))*cdfnor("PQ",d2(x,t,K,T,r,sigma),0,1);
    p = first - second;
endfunction


function [p] = Put(x, t, T, K, r, sigma)
    first = x*cdfnor("PQ",-d1(x,t,K,T,r,sigma),0,1);
    second = K*exp(-r*(T-t))*cdfnor("PQ",-d2(x,t,K,T,r,sigma),0,1);
    p = -first + second;
endfunction

//-----------------------------------------------------------

x = 1000;
T = 100;
K = 1000;
r = 0.005;
sigma = 0.1;
t=0;

n = 1000;
X = grand(n,1,"nor",0,1);

//For a Call
function [p] = gc(X, x, t, T, K, r, sigma)
    first = K*exp(-r*(T-t));
    second = x*exp(sigma*X*sqrt(T-t)) *exp(-((T-t)*sigma**2)/2)
    p = max(0, -first + second);
endfunction
```

Figure 27 – Simple simulation du Call 1/3

```
test = g(X(1), x, t, T, K, r, sigma);

//evaluation of Call with Monte Carlo method
function [p] = eval_Call(X, x, t, T, K, r, sigma, n)
    tmp = 0;
    for i = 1:n
        tmp = tmp + gc(X(i), x, t, T, K, r, sigma)/n;
    end
    p = tmp
endfunction


function [p]= error_Rate(X, x, t, T, K, r, sigma, n)
    tmp = 0;
    I = eval_Call(X, x, t, T, K, r, sigma, n);
    for i = 1:n
        tmp = tmp + (gc(X(i), x, t, T, K, r) - I)**2/(n-1);
    end
    p = tmp;
endfunction


function [p] = boundary(X, x, t, T, K, r, sigma, n)
        I = eval_Call(X, x, t, T, K, r, sigma, n);
        A = error_Rate(X, x, t, T, K, r, sigma, n)

        p = [I - 1.96*sqrt(A/n), I + 1.96*sqrt(A/n)];
endfunction

function [p] = test1(X, x, t, T, K, r, sigma, y)
    for i = 1:length(y)
        a(i) = eval_Call(X, x, t, T, K, r, sigma, y(i));
    end
    p = [a];
endfunction


function [p] = test2(X, x, t, T, K, r, sigma, y)
    for i = 1:length(y)
        b(i) = Call(x, t, T, K, r, sigma);
    end
    p = [b];
endfunction
```

Figure 28 – Simple simulation du Call 2/3

```
function [p] = test3(X, x, t, T, K, r, sigma, y,n)
    b = boundary(X, x, t, T, K, r, sigma, n)
    for i = 1:length(y)
        c(i) = b(1);
    end
    p = [c];
endfunction


function [p] = test4(X, x, t, T, K, r, sigma, y, n)
    b = boundary(X, x, t, T, K, r, sigma, n)
    for i = 1:length(y)
        c(i) = b(2);
    end
    p = [c];
endfunction


n = 10000
y = [10:100:n];
X = grand(n,1,"nor",0,1);
//clf();
//plot2d(y, test1(X, x, t, T, K, r, sigma, y), logflag = "ln", style=5);
//plot2d(y, test2(X, x, t, T, K, r, sigma, y), logflag = "ln", style=4);
//plot2d(y, test3(X, x, t, T, K, r, sigma, y, n), logflag = "ln", style=2);
//plot2d(y, test4(X, x, t, T, K, r, sigma, y, n), logflag = "ln", style=2);
//legend(["Call par Monte Carlo";"Call exacte";"inte"])
```

Figure 29 – Simple simulation du Call 3/3

```
//Exercice 3

t = 0;
x = 1000;
T = 100;
K = 1000;
r = 0.005;
sigma = 0.1;
n = 10000
y = [10:100:n];
X = grand(n,1,"nor",0,1);

//We use the previous Call compute with Monte Carlo
function [p] = eval_PutVC(X, x, t, T, K, r, sigma, n)
    I = x*(1-exp(-r*T));
    p = I - eval_Call(X, x, t, T, K, r, sigma, n)
endfunction


function [p]= error_Rate(X, x, t, T, K, r, sigma, n)
    tmp = 0;
    I = eval_PutVC(X, x, t, T, K, r, sigma, n);
    for i = 1:n
        tmp = tmp + (g(X(i), x, t, T, K, r) - I)**2/(n-1);
    end
    p = tmp;
endfunction


function [p] = boundary(X, x, t, T, K, r, sigma, n)
        I = eval_PutVC(X, x, t, T, K, r, sigma, n);
        A = error_Rate(X, x, t, T, K, r, sigma, n)

        p = [I - 1.96*sqrt(A/n), I + 1.96*sqrt(A/n)];
endfunction
```

Figure 30 – Simple simulation du Put par Variable de contrôle 1/3

```
function [p] = test1(X, x, t, T, K, r, sigma, y)
    for i = 1:length(y)
        a(i) = -eval_PutVC(X, x, t, T, K, r, sigma, y(i));
    end
    p = [a];
endfunction


function [p] = test2(X, x, t, T, K, r, sigma, y)
    for i = 1:length(y)
        b(i) = Put(x, t, T, K, r, sigma);
    end
    p = [b];
endfunction


function [p] = test3(X, x, t, T, K, r, sigma, y,n)
    b = boundary(X, x, t, T, K, r, sigma, n)
    for i = 1:length(y)
        c(i) = -b(1);
    end
    p = [c];
endfunction


function [p] = test4(X, x, t, T, K, r, sigma, y, n)
    b = boundary(X, x, t, T, K, r, sigma, n)
    for i = 1:length(y)
        c(i) = -b(2);
    end
    p = [c];
endfunction


n = 1000
y = [1:10:n];
X = grand(n,1,"nor",0,1);
clf();
plot2d(y, test1(X, x, t, T, K, r, sigma, y), logflag = "ln", style=5);
plot2d(y, test2(X, x, t, T, K, r, sigma, y), logflag = "ln", style=4);
plot2d(y, test3(X, x, t, T, K, r, sigma, y, n), logflag = "ln", style=2);
plot2d(y, test4(X, x, t, T, K, r, sigma, y, n), logflag = "ln", style=2);
legend(["Put␣par␣Monte␣Carlo";"Put␣exacte";"inte"])
```

Figure 31 – Simple simulation du Put par Variable de contrôle 2/3

```
//Rappel
function [p] = d1(x,t,K,T,r,sigma)
    a = log(x/K)+(r+sigma**2/2)*(T-t)
    p = a /(sigma*sqrt(T-t))
endfunction

test = d1(10,100,100,30,0.05,0.1)

function [p] = d2(x,t,K,T,r,sigma)
    p =  d1(x,t,K,T,r,sigma) - sigma*sqrt(T-t)
endfunction

test = d2(10,100,100,30,0.05,0.1)

function [p] = Call(x, t, T, K, r, sigma)
    first = x*cdfnor("PQ",d1(x,t,K,T,r,sigma),0,1);
    second = K*exp(-r*(T-t))*cdfnor("PQ",d2(x,t,K,T,r,sigma),0,1);
    p = first - second;
endfunction


function [p] = Put(x, t, T, K, r, sigma)
    first = x*cdfnor("PQ",-d1(x,t,K,T,r,sigma),0,1);
    second = K*exp(-r*(T-t))*cdfnor("PQ",-d2(x,t,K,T,r,sigma),0,1);
    p = -first + second;
endfunction

//-------------------------------------------------------

x = 1000;
T = 100;
K = 1000;
r = 0.005;
sigma = 0.1;

n = 1000;
X = grand(n,1,"nor",0,1);

function [p] = g2(X, x, t, T, K, r, sigma)
    first = K*exp(-r*(T-t));
    second = x*exp(sigma*X*sqrt(T-t)) *exp(-((T-t)*sigma**2)/2)
    p = max(0, first - second);
endfunction


function [p] = g(X, x, t, T, K, r, sigma)
    p = (g2(X, x, t, T, K, r, sigma) + g2(X, x, t, T, K, r, sigma))/2
endfunction


test = g(X(1), x, t, T, K, r, sigma);
```

FIGURE 32 – Simple simulation du Put par Variable antithétique 1/3

```
//evaluation with Monte Carlo method
function [p] = eval_Put(X, x, t, T, K, r, sigma, n)
    tmp = 0;
    for i = 1:n
        tmp = tmp + g(X(i), x, t, T, K, r, sigma)/n;
    end
    p = tmp
endfunction


function [p]= error_Rate(X, x, t, T, K, r, sigma, n)
    tmp = 0;
    I = eval_Put(X, x, t, T, K, r, sigma, n);
    for i = 1:n
        tmp = tmp + (g(X(i), x, t, T, K, r) - I)**2/(n-1);
    end
    p = tmp;
endfunction


function [p] = boundary(X, x, t, T, K, r, sigma, n)
        I = eval_Put(X, x, t, T, K, r, sigma, n);
        A = error_Rate(X, x, t, T, K, r, sigma, n)

        p = [I - 1.96*sqrt(A/n), I + 1.96*sqrt(A/n)];
endfunction

function [p] = test1(X, x, t, T, K, r, sigma, y)
    for i = 1:length(y)
        a(i) = eval_Put(X, x, t, T, K, r, sigma, y(i));
    end
    p = [a];
endfunction


function [p] = test2(X, x, t, T, K, r, sigma, y)
    for i = 1:length(y)
        b(i) = Put(x, t, T, K, r, sigma);
    end
    p = [b];
endfunction


function [p] = test3(X, x, t, T, K, r, sigma, y,n)
    b = boundary(X, x, t, T, K, r, sigma, n)
    for i = 1:length(y)
        c(i) = b(1);
    end
    p = [c];
endfunction
```

FIGURE 33 – Simple simulation du Put par Variable antithétique 2/3

```
function [p] = test4(X, x, t, T, K, r, sigma, y, n)
    b = boundary(X, x, t, T, K, r, sigma, n)
    for i = 1:length(y)
        c(i) = b(2);
    end
    p = [c];
endfunction


n = 10000
y = [1:100:n];
X = grand(n,1,"nor",0,1);
clf();
plot2d(y, test1(X, x, t, T, K, r, sigma, y), logflag = "ln", style=5);
plot2d(y, test2(X, x, t, T, K, r, sigma, y), logflag = "ln", style=4);
plot2d(y, test3(X, x, t, T, K, r, sigma, y, n), logflag = "ln", style=2);
plot2d(y, test4(X, x, t, T, K, r, sigma, y, n), logflag = "ln", style=2);
legend(["Put␣par␣Monte␣Carlo";"Put␣exacte";"inte"])
```

Figure 34 – Simple simulation du Put par Variable antithétique 3/3

```
//Rappel
function [p] = d1(x,t,K,T,r,sigma)
    a = log(x/K)+(r+sigma**2/2)*(T-t)
    p = a /(sigma*sqrt(T-t))
endfunction

test = d1(10,100,100,30,0.05,0.1)

function [p] = d2(x,t,K,T,r,sigma)
    p =  d1(x,t,K,T,r,sigma) - sigma*sqrt(T-t)
endfunction

test = d2(10,100,100,30,0.05,0.1)

function [p] = Call(x, t, T, K, r, sigma)
    first = x*cdfnor("PQ",d1(x,t,K,T,r,sigma),0,1);
    second = K*exp(-r*(T-t))*cdfnor("PQ",d2(x,t,K,T,r,sigma),0,1);
    p = first - second;
endfunction


function [p] = Put(x, t, T, K, r, sigma)
    first = x*cdfnor("PQ",-d1(x,t,K,T,r,sigma),0,1);
    second = K*exp(-r*(T-t))*cdfnor("PQ",-d2(x,t,K,T,r,sigma),0,1);
    p = -first + second;
endfunction

//-------------------------------------------------------

x = 1000;
T = 100;
K = 1000;
r = 0.005;
sigma = 0.1;

n = 1000;
X = grand(n,1,"exp",1);

function [p] = g3(X, x, t, T, K, r, sigma)
    first = K*exp(-r*(T-t));
    second = x*exp(sigma*X*sqrt(T-t)) *exp(-((T-t)*sigma**2)/2)
    p = max(0, first - second);
endfunction
```

Figure 35 – Simple simulation du Put par échantillonnage préférentiel 1/3

```
//X is expenontielle randomized variable
function [p] = g(X, x, t, T, K, r, sigma)
    f = x*exp(-r*T)/sqrt(2*%pi);
    h = (1-exp(-sigma*sqrt(2*T*X)))/sqrt(2*X);
    p = f*h;
endfunction


test = g(X(1), x, t, T, K, r, sigma);

//evaluation with Monte Carlo method
function [p] = eval_Put(X, x, t, T, K, r, sigma, n)
    tmp = 0;
    for i = 1:n
        tmp = tmp + g(X(i), x, t, T, K, r, sigma)/n;
    end
    p = tmp
endfunction


function [p]= error_Rate(X, x, t, T, K, r, sigma, n)
    tmp = 0;
    I = eval_Put(X, x, t, T, K, r, sigma, n);
    for i = 1:n
        tmp = tmp + (g(X(i), x, t, T, K, r) - I)**2/(n-1);
    end
    p = tmp;
endfunction


function [p] = boundary(X, x, t, T, K, r, sigma, n)
        I = eval_Put(X, x, t, T, K, r, sigma, n);
        A = error_Rate(X, x, t, T, K, r, sigma, n)

        p = [I - 1.96*sqrt(A/n), I + 1.96*sqrt(A/n)];
endfunction

function [p] = test1(X, x, t, T, K, r, sigma, y)
    for i = 1:length(y)
        a(i) = eval_Put(X, x, t, T, K, r, sigma, y(i));
    end
    p = [a];
endfunction


function [p] = test2(X, x, t, T, K, r, sigma, y)
    for i = 1:length(y)
        b(i) = Put(x, t, T, K, r, sigma);
    end
    p = [b];
endfunction


function [p] = test3(X, x, t, T, K, r, sigma, y,n)
    b = boundary(X, x, t, T, K, r, sigma, n)
    for i = 1:length(y)
        c(i) = b(1);
    end
    p = [c];
endfunction
```

```
function [p] = test4(X, x, t, T, K, r, sigma, y, n)
    b = boundary(X, x, t, T, K, r, sigma, n)
    for i = 1:length(y)
        c(i) = b(2);
    end
    p = [c];
endfunction


n = 1000000
y = [1:100000:n];
X = grand(n,1,"exp",1);
clf();
plot2d(y, test1(X, x, t, T, K, r, sigma, y), logflag = "ln", style=5);
plot2d(y, test2(X, x, t, T, K, r, sigma, y), logflag = "ln", style=4);
plot2d(y, test3(X, x, t, T, K, r, sigma, y, n), logflag = "ln", style=2);
plot2d(y, test4(X, x, t, T, K, r, sigma, y, n), logflag = "ln", style=2);
legend(["Put␣par␣Monte␣Carlo";"Put␣exacte";"inte"])
```

Figure 37 – Simple simulation du Put par échantillonnage préférentiel 3/3