

# Méthodes de Monte Carlo et techniques de réduction de variance, applications au pricing d'options

Arouna GANOU

M2 IMOI MF, Université de Lorraine, France

ISN 3A, ENSEM, France

## I. Introduction

L'objectif du TP est d'utiliser la méthode de Monte Carlo pour approcher les prix des options européennes. Nous allons utiliser plusieurs approches et ensuite les comparer.

## II. Notation

Nous allons introduire les notations<sup>1</sup>.

$C_t$  = Le prix de l'option d'achat

$S_t$  = Le prix du sous-jacent à l'instant  $t$

$K$  = Le strike (prix d'exercice)

$r$  = Le taux d'intérêt pour un placement

$\sigma$  = La volatilité du prix du sous-jacent

$T$  = La maturité de l'option.

$N$  = la fonction de répartition de la loi normale centrée réduite

## III. Exercice 1

Nous avons montré les expressions exactes du call et du put sous forme analytique exacte à partir de  $C = \mathbb{E}[(\exp(\beta G) - K)_+]$  (\*) et  $P = \mathbb{E}[(K - \exp(\beta G))_+]$  (\*\*), elles permettront de tester les sorties des simulations. Nous avons trouvé ainsi :

$$C = \exp\left(\frac{\beta^2}{2}\right) N\left(\beta - \frac{\log(K)}{\beta}\right) - KN\left(-\frac{\log(K)}{\beta}\right) \quad (1)$$

$$P = KN\left(\frac{\log(K)}{\beta}\right) - \exp\left(\frac{\beta^2}{2}\right) N\left(\frac{\log(K)}{\beta} - \beta\right) \quad (2)$$

## IV. Exercice 2

1. A partir de la relation (\*) nous pouvons écrire un programme pour approcher le prix du call.

### CODE MATLAB POUR LE CALL EXERCICE 2

```
function [I_hat, err_std] = monteCarloCall(N)
% Cette fonction fait une simulation de Monte
% en
% N echantillons
% ENTREE : N: Le nombre de simulation
%
% SORTIE : I_hat: La valeur approchée de la
% quantite
% que nous voulons évaluer.
% err_std: Erreur standard de la
% simulation realisee
% parametre du call
beta=1;K=1;
X= randn(); % simulation d'une variable de loi
% normale
% centree reduite
Y=max(exp(beta*X)-K,0); % on évalue ensuite Y
S1=Y; % somme partielle des Yi
S2=Y^2; % somme partielle des Yi^2
for n=1:N
X=randn(); % simulation d'une variable de
% loi normale
Y=max(exp(beta*X)-K,0); % on évalue ensuite
% Y
S1=S1+Y; % mise-a-jour de S1
S2=S2+Y^2; % mise-a-jour de S2
end
% on estime la variance par son estimateur sans
% biais
s=sqrt((S2-N*(S1/N)^2)/(N-1));
% on retourne l'estimation obtenue.
I_hat=S1/N;
% on retourne l'erreur standard de cette
% simulation
err_std=s/sqrt(N);
end
```

L'application de la méthode de Monte Carlo se justifie juste par les

Courriel: arouna.ganou5@etu.univ-lorraine.fr

1. Ces notations sont celles de l'énoncé du TP

hypothèses de la Loi Forte des Grands Nombres (LFGN) pour la convergence et le Théorème Central Limit (TCL) pour l'utilisation des intervalles de confiance. On a un générateur de variables aléatoires (v.a) indépendantes et identiquement distribuées (i.i.d) de loi normale centrée réduite. Posons  $X = \exp(\beta G)$ . Il reste donc à justifier que la v.a  $Z_{call}^1 = (X - K)_+$  est  $\mathbb{L}^1$  pour la LFGN et  $\mathbb{L}^2$  pour le TCL. Or si  $Z_{call}^1$  est  $\mathbb{L}^2$  alors  $Z_{call}^1$  est aussi  $\mathbb{L}^1$ . Donc il suffit de montrer que  $Z_{call}^1$  est  $\mathbb{L}^2$ . On a :

$$|Z_{call}^1| = Z_{call}^1 = (X - K)_+ \leq |X - K|$$

et

$$|X - K|^2 = X^2 - 2KX + K^2 \quad (K \text{ constante})$$

$X$  suit la loi log-normale de paramètre  $(0, \beta^2)$  donc  $X$  est  $\mathbb{L}^2$ . On en déduit que  $|Z_{call}^1| = Z_{call}^1$  est  $\mathbb{L}^2$ . Donc l'application de la méthode de Monte Carlo est justifiée.

2. A partir de la relation (\*\*) nous pouvons écrire un programme pour approcher le prix du put.

### CODE MATLAB POUR LE PUT EXERCICE 2

```
function [I_hat, err_std] = monteCarloPut(N)
% Cette fonction fait une simulation de Monte
% en
% N echantillons
% ENTREE : N: Le nombre de simulation
%
% SORTIE : I_hat: La valeur approchée de la
% quantite
% que nous voulons évaluer.
% err_std: Erreur standard de la
% simulation realisee
% parametre du call
beta=1;
K=1;
X= randn(); % simulation d'une variable de loi
% normale
% centree reduite
Y=max(K-exp(beta*X),0); % on évalue ensuite Y
S1=Y; % somme partielle des Yi
S2=Y^2; % somme partielle des Yi^2
n=1;
while (n<N)
X=randn(); % simulation d'une variable de loi
% normale
% centree reduite
Y=max(K-exp(beta*X),0); % on évalue ensuite
% Y
S1=S1+Y; % mise-a-jour de S1
S2=S2+Y^2; % mise-a-jour de S2
n=n+1;
end
% on estime la variance par son estimateur sans
% biais
s=sqrt((S2-N*(S1/N)^2)/(N-1));
% on retourne l'estimation obtenue.
I_hat=S1/N;
% on retourne l'erreur standard de cette
% simulation
err_std=s/sqrt(N);
end
```

La justification est exactement la même chose que le cas du call. Posons  $Z_{put}^1 = (K - X)_+$ . Il s'agit de justifier dans ce cas que  $Z_{put}^1$  est  $\mathbb{L}^2$  et on a :

$$|Z_{put}^1| = Z_{put}^1 = (K - X)_+ \leq |K - X| = |X - K|$$

Or on a déjà justifié que  $|X - K|$  est  $\mathbb{L}^2$ . Ainsi  $Z_{put}^1$  est  $\mathbb{L}^2$ .

3. Commentaires : Sur la figure 1 ci-dessous, nous avons les résultats des simulations en fonction du nombre d'échantillons pris. Nous constatons que l'erreur pour l'estimation du put est très petite par rapport à celle du call.

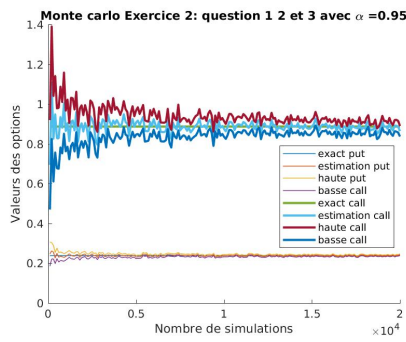


FIGURE 1. L' estimation et de l'erreur en fonction du nombre de simulations.

#### CODE MATLAB DE L'UTILISATION DES FONCTIONS PRÉCÉDENTES

```
1 %% monte carlo pour l'exercice 1
2 clc
3 close all
4 % le quantile d'ordre (alpha+1)/2 de la loi
   normale
5 % centree reduite.
6 alpha=.95;
7 Z=norminv((alpha+1)/2,0,1);
8 % le nombre d'estimations des options.
9 Nsim=200;
10 % les pas entre la taille des echantillons
11 pas=100;
12 % on initialise le vecteur des estimations
13 % et des erreurs. Pour trouver l'intervalle
14 % de confiance plus tard il suffirait d'utili-
15 % liser IC=[I-Z*s/sqrt(n),I+Z*s/sqrt(n)]
16 I_put=zeros(1,Nsim);
17 Err_put=L_put;
18 I_call=zeros(1,Nsim);
19 Err_call=L_call;
20 % parametres des options
21 K=1;beta=1;
22 % les valeurs exactes des options
23 C_ex=(exp(beta^2/2)*normcdf(beta-log(K)/beta,0,1)
   -...
   K*normcdf(-log(K)/beta,0,1));
24 C=C_ex*ones(1,Nsim);
25 P_ex=(-exp(beta^2/2)*normcdf(-beta+log(K)/beta
   ,0,1)+...
   K*normcdf(log(K)/beta,0,1));
26 P=P_ex*ones(1,Nsim);
27 % on augmente le nombre d'echantillons de pas a
   chaque
28 % fois pour obtenir Nsim simulation de chaque
   option
29 % debut de la boucle for pour les Nsim
   estimations
30 for n=1:Nsim
31 % estimation du call
32 [I_hat_call,err_std_call]=monteCarloCall(n*
   pas);
33 I_call(n)=I_hat_call;
34 Err_call(n)=err_std_call;
35 % estimation du put
36 [I_hat_put,err_std_put]=monteCarloPut(n*pas);
37 I_put(n)=I_hat_put;
38 Err_put(n)=err_std_put;
39 end
40 % fin de la boucle for pour les Nsim estimations
41 % affichage des resultats
42 % preparation de la figure
43 N=pas*(1:Nsim); % vecteur du nombre de simulation
44 fig_exo1=figure();
45 title('Monte carlo Exercice 1: question 1 2 et 3
   avec \alpha =0.95');
46 xlabel('Nombre de simulations');
47 ylabel('Valeurs des options');
48 hold on
49 % affichage du put
50 figEx_put=plot(N,P,'LineWidth',.74); % valeur
   exacte
51 figI_put=plot(N,I_put,'LineWidth',.74); % l'
   estimation
52 figHaut_put=plot(N,I_put+Z*Err_put,'LineWidth'
   ,.74); % borne sup de l'IC
53 figBas_put=plot(N,I_put-Z*Err_put,'LineWidth'
   ,.74); % borne inf de l'IC
54 % affichage du call
55 figEx_call=plot(N,C,'LineWidth',2); % valeur
   exacte
56 figI_call=plot(N,I_call,'LineWidth',2); % l'
   estimation
57 figHaut_call=plot(N,I_call+Z*Err_call,'LineWidth'
   ,2); % borne sup de l'IC
58 figBas_call=plot(N,I_call-Z*Err_call,'LineWidth'
   ,2); % borne inf de l'IC
59 % ajout des legendes a la figure
60 legend([figEx_put, figI_put, figHaut_put,
   figBas_put,...
   figEx_call, figI_call, figHaut_call,
   figBas_call],...
   'exact put', 'estimation put', 'haute put'
   , 'basse put',...
   'exact call', 'estimation call', 'haute
   call', 'basse call');
61 % on enregistre la figure sous format eps
62 chem='images';chem=strcat(chem,'/exo1');
63 print(fig_exo1,chem,'-djpeg')
```

#### V. Exercice 3

Dans cet exercice, nous allons utiliser l'échantillonnage préférentiel. Nous essayerons d simuler dans le domaine où la valeur du call est élevé(en l' occurrence ici non-nul) donc nous transformons l'expression(\*) dépendant d'une loi normale centrée réduite(dont les valeurs négatives ne sont pas utiles mais augmentent la variance de la simulation) en une autre forme dépendant d'une loi exponentielle.

1. On a (avec  $K = 1$  et  $\beta = 1$ ) :

$$C = \mathbb{E} \left[ Z_{call}^1 \right] = \mathbb{E} \left[ (\exp(\beta G) - K)_+ \right] = \mathbb{E} \left[ (\exp(G) - 1)_+ \right]$$

Par croissance de l'exponentielle :

$$C = \frac{1}{\sqrt{2\pi}} \int_0^\infty (e^g - 1) e^{-\frac{g^2}{2}} dg$$

On effectue ensuite le changement de variable croissante  $y = \frac{g^2}{2}$  donc comme  $g \geq 0$  on a  $g = \sqrt{2y}$  et  $dg = \frac{dy}{\sqrt{2y}}$ . Ainsi

$$C = \frac{1}{\sqrt{2\pi}} \int_0^\infty \frac{e^{\sqrt{2y}} - 1}{\sqrt{2y}} e^{-y} dy$$

On reconnaît l'espérance de la v.a  $Z_{call}^2 = \frac{e^{\sqrt{2Y}} - 1}{\sqrt{2Y}}$  avec  $Y$  une v.a de loi exponentielle de paramètre 1. Cette v.a est bien  $\mathbb{L}^2$  donc on peut appliquer la méthode de Monte Carlo et utiliser les intervalles de confiance. En effet,

$$y \mapsto \left[ \frac{e^{\sqrt{2y}} - 1}{\sqrt{2y}} \right]^2 e^{-y}$$

sur  $]32, +\infty[$  On a :

$$\left[ \frac{e^{\sqrt{2y}} - 1}{\sqrt{2y}} \right]^2 e^{-y} \leq \frac{e^{2\sqrt{2y}}}{2y} e^{-y} \leq e^{2\sqrt{2y}} e^{-y} = e^{-y(1 - \frac{2\sqrt{2}}{\sqrt{y}})} \leq e^{-\frac{y}{2}}$$

donc est intégrable sur  $]32, +\infty[$  car  $y \mapsto e^{-\frac{y}{2}}$  l'est.

$$y \mapsto \left[ \frac{e^{\sqrt{2y}} - 1}{\sqrt{2y}} \right]^2 e^{-y}$$

sur  $]0, 32[$  est intégrable sur  $]0, 32[$  car continue et la limite est finie à droite de 0. En effet, pour  $y$  petit, on a :

$$\left[ \frac{e^{\sqrt{2y}} - 1}{\sqrt{2y}} \right]^2 e^{-y} \approx_{y \rightarrow 0^+} \left[ \frac{\sqrt{2y}}{\sqrt{2y}} \right]^2 e^{-y} = e^{-y} \rightarrow 1$$

Finalement

$$y \mapsto \left[ \frac{e^{\sqrt{2y}} - 1}{\sqrt{2y}} \right]^2 e^{-y}$$

sur  $]0, +\infty[$  est intégrable. Donc on peut bien mettre  $C$  sous la forme :

$$C = \frac{1}{\sqrt{2\pi}} \mathbb{E} \left( \frac{e^{\sqrt{2Y}} - 1}{\sqrt{2Y}} \right)$$

avec  $Y$  une v.a de loi exponentielle de paramètre 1.

2. Nous avons écrit une nouvelle méthode de simulation du call : le code est à la page suivante.

3. Nous comparons cette méthode avec l'approche de la question précédente dans le cas du call uniquement. Nous constatons que la deuxième méthode est meilleure car on a un bon contrôle de l'erreur : la variance est beaucoup plus réduite. En effet, les largeurs des intervalles de confiance sont très petites par rapport à la première méthode. Par contre la deuxième méthode exige beaucoup plus de calculs comme nous l'avons indiqué sur la figure 2(à la page suivante)de l'ordre de 100 fois plus de temps de calculs. Ceci est dû à la loi exponentielle qui prend plus de temps à simuler sous Matlab que la loi normale.

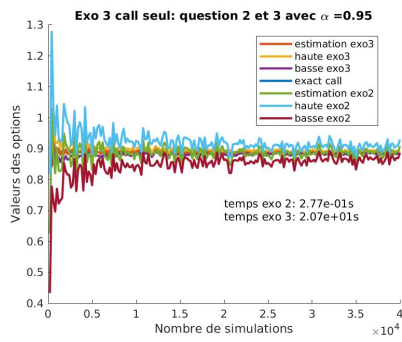


FIGURE 2. Comparaison entre les estimations et les erreurs.

### CODE MATLAB POUR LE CALL EXERCICE 3

```
function [I_hat, err_std]=monteCarloCallExo3(N)
% Cette fonction fait une simulation de Monte
% en
% N echantillons en utilisant la deuxieme
% expression
% du call et on simule une loi exponentielle
% ENTREE : N: Le nombre de simulation
% SORTIE : I_hat: La valeur approchee de la
% quantite
% que nous voulons evaluer.
% err_std: Erreur standard de la
% simulation realisee
% parametre du call
X= exprnd(1); % simulation d'une variable loi
% exponentielle
% de parametre 1
Y=(exp(sqrt(2*X))-1)/sqrt(2*X);
S1=Y; % somme partielle des Yi
S2=Y^2; % somme partielle des Yi^2
n=1;
while (n<N)
X=exprnd(1); % on une variable de loi
% exponentielle de parametre 1
Y=(exp(sqrt(2*X))-1)/(sqrt(2*X)*sqrt(2*pi));
% on evalue ensuite Y
S1=S1+Y; % mise-a-jour de S1
S2=S2+Y^2; % mise-a-jour de S2
n=n+1;
end
% on estime la variance par son estimateur sans
% biais
s=sqrt((S2-N*(S1/N)^2)/(N-1));
% on retourne l'estimation obtenue.
I_hat=S1/N;
% on retourne l'erreur standard de cette
% simulation
err_std=s/sqrt(N);
end
```

### CODE MATLAB UTILISANT LA MÉTHODE DE L'EXERCICE 3

```
% monte carlo pour l'exercice 3
clc; close all
% le quantile d'ordre (alpha+1)/2 de la loi
% normale
alpha=.95;
Z=norminv((alpha+1)/2,0,1);
% pas et nombre d'estimations des options.
Nsim=200; pas=200;
% on initialise le vecteur des estimations
% et des erreurs. Pour trouver l'intervalle
% de confiance plus tard il suffirait d'uti-
% liser IC=[I-Z*s/sqrt(n), I+Z*s/sqrt(n)]
I_call_1=zeros(1,Nsim);
Err_call_1=I_call_1;
I_call_2=zeros(1,Nsim);
Err_call_2=I_call_2;
% parametres des options
K=1; beta=1;
% les valeurs exactes des options
C_ex=(exp(beta^2/2)*normcdf(beta-log(K)/beta,0,1))-
...
K*normcdf(-log(K)/beta,0,1));
C=C_ex*ones(1,Nsim);
% on augmente le nombre d'echantillons de pas a
% chaque
% fois pour obtenir Nsim simulation de chaque
% option
```

```
% debut de la boucle for pour les Nsim
% estimations
% estimation du call avec la methode de l'exo2
start=tic;
for n=1:Nsim
[I_hat_call_1, err_std_call_1]=monteCarloCall(
n*pas);
I_call_1(n)=I_hat_call_1;
Err_call_1(n)=err_std_call_1;
end
t_exo2=toc(start);
% fin de la boucle for pour les Nsim estimations
% exo2
% debut de la boucle for pour les Nsim
% estimations
% estimation du call avec la methode de l'exo3
start=tic;
for n=1:Nsim
[I_hat_call_2, err_std_call_2]=
monteCarloCallExo3(n*pas);
I_call_2(n)=I_hat_call_2; Err_call_2(n)=
err_std_call_2;
end
t_exo3=toc(start);
% fin de la boucle for pour les Nsim estimations
% exo3
% affichage des resultats et preparation de la
% figure
N=pas*(1:Nsim); % vecteur du nombre de simulation
fig_exo2=figure();
title('Exo 3 call seul: question 2 et 3 avec \
alpha =0.95')
xlabel('Nombre de simulations')
ylabel('Valeurs des options'); hold on
% ajout des temps de calculs
str = {sprintf('temps exo 2: %0.2es',t_exo2), ...
sprintf('temps exo 3: %0.2es',t_exo3)};
text(pas*Nsim*7/14,.5,str)
% affichage du call exact
figEx_call=plot(N,C,'LineWidth',2); % valeur
% exacte
% affichages pour la methode de l'exo2
figI_call_2=plot(N,I_call_2,'LineWidth',2); % l'
% estimation
figHaut_call_2=plot(N,I_call_2+Z*Err_call_2, '
LineWidth',2); % sup de l'IC
figBas_call_2=plot(N,I_call_2-Z*Err_call_2, '
LineWidth',2); % inf de l'IC
% affichage du call par l'exo 1
figI_call_1=plot(N,I_call_1,'LineWidth',2); % l'
% estimation
figHaut_call_1=plot(N,I_call_1+Z*Err_call_1, '
LineWidth',2); % borne sup de l'IC
figBas_call_1=plot(N,I_call_1-Z*Err_call_1, '
LineWidth',2); % borne inf de l'IC
% ajout des legendes a la figure
legend([ figI_call_2, figHaut_call_2,
figBas_call_2, ...
figEx_call, figI_call_1, figHaut_call_1,
figBas_call_1 ], ...
'estimation exo3','haute exo3','basse
exo3',...
'exact call', 'estimation exo2','haute
exo2','basse exo2');
% on enregistre la figure sous format jpg
chem='images'; chem=strcat(chem,'/exo3');
print(fig_exo2,chem,'-djpeg')
```

### VI. Exercice 4

Comme nous l'avons vu dans l'exercice 2 lors de l'estimation que l'erreur de l'estimation du call est meilleure que celle du call. L'idée dans cet exercice c'est d'utiliser la relation entre le put et le call pour contrôler l'erreur du call. Nous allons naturellement utiliser la parité call-put pour estimer le call. Nous allons estimer le call indirectement en estimant le put.

1. On a :

$$\forall t \in [0, T] C_t - P_t = S_t - Ke^{-r(T-t)} \text{ donc } t = T, C_T - P_T = S_T - K$$

$$\text{donc } C_T - P_T = C - P = S_T - K \text{ donc } C = P - K + \mathbb{E}[S_T].$$

$K$  étant constante et en utilisant la propriété de martingale de  $C_t, P_t$  et  $S_t$ , On a en plus du put l'espérance d'une variable aléatoire de loi log-normale de paramètre  $(0, \beta^2)$  à estimer. Ici, nous allons utiliser nos connaissances sur la loi log-normale pour avoir la valeur  $\mathbb{E}[S_T] = e^{\frac{\beta^2}{2}}$ . En fait, nous pouvons effectivement simuler les deux à la fois, mais nous n'aurons aucune réduction de variance dans ce cas. Ce qui est clairement logique car cette relation n'est pas obtenue dans le cadre d'une réduction de variance.

2. Nous constatons sans surprise (puisque nous avons vu dans l'exercice 2 que le put avait une erreur plus faible) que l'erreur de l'estimation du call est très fortement améliorée par rapport à

l'exercice 1. Notre estimation dépendait uniquement du put et de deux autres constantes "connues". Mais le plus important ici, c'est de remarquer ici que nous avons réduit la variance en faisant clairement moins de calcul. Par rapport à la méthode de l'exercice 3, nous avons fait 100 fois moins de calcul pour aboutir à des erreurs de qualités proches.

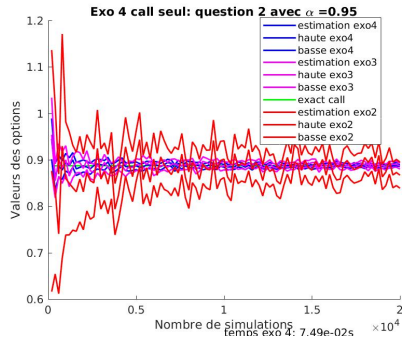


FIGURE 3. Comparaison entre les estimations et les erreurs.

#### CODE MATLAB CALL EXERCICE 4

```
function [I_hat, err_std]=monteCarloCallExo4(N)
% Cette fonction fait une simulation de Monte
% en
% N echantillons en utilisant la relation de
% parite call-put
% ENTREE : N: Le nombre de simulation
% SORTIE : I_hat: La valeur approchée du call
% err_std: Erreur standard de la
% simulation realisee
% parametre du call
beta=1;
K=1;
X= randn(); % simulation d'une variable de loi
% normale
% centree reduite
Y=max(K-exp(beta*X),0)-K*exp(beta^2/2);
S1=Y; % somme partielle des Yi
S2=Y^2; % somme patielle des Yi^2
n=1;
while (n<N)
X=randn(); % on une normale centree reduite
Y=max(K-exp(beta*X),0)-K*exp(beta^2/2);
S1=S1+Y; % mise-a-jour de S1
S2=S2+Y^2; % mise-a-jour de S2
n=n+1;
end
% on estime la variance par son estimateur sans
% biais
s=sqrt((S2-N*(S1/N)^2)/(N-1));
% on retourne l'estimation obtenue.
I_hat=S1/N;
% on retourne l'erreur standard de cette
% simulation
err_std=s/sqrt(N);
end
```

#### CODE MATLAB COMPARAISON EXERCICE 4

```
% monte carlo pour l'exercice 4
clc;close all
% le quantile d'ordre (alpha+1)/2 de la loi
% normale
% centree reduite.
alpha=.95;Z=norminv((alpha+1)/2,0,1);
% pas et nombre d'estimations des options.
Nsim=100;pas=200;
% on initialise le vecteur des estimations
% et des erreurs. Pour trouver l'intervalle
% de confiance plus tard il suffirait d'uti-
% liser IC=[I-Z*s/sqrt(n),I+Z*s/sqrt(n)]
I_call_1=zeros(1,Nsim);
Err_call_1=I_call_1;
I_call_2=zeros(1,Nsim);
Err_call_2=I_call_2;
I_call_3=zeros(1,Nsim);
Err_call_3=I_call_3;
% parametres des options
K=1;beta=1;
% les valeurs exactes des options
C_ex=(exp(beta^2/2)*normcdf(beta-log(K)/beta,0,1)-
...
K*normcdf(-log(K)/beta,0,1));
C=C_ex*ones(1,Nsim);
% on augmente le nombre d'echantillons de pas a
% chaque
% fois pour obtenir Nsim simulation de chaque
% option
```

```
% debut de la boucle for pour les Nsim
% estimations
% estimation du call avec la methode de l'exo2
start=tic;
for n=1:Nsim
[I_hat_call_1,err_std_call_1]=monteCarloCall(
n*pas);
I_call_1(n)=I_hat_call_1;
Err_call_1(n)=err_std_call_1;
end
t_exo2=toc(start);
% fin de la boucle for pour les Nsim estimations
% exo2
% debut de la boucle for pour les Nsim
% estimations
% estimation du call avec la methode de l'exo3
start=tic;
for n=1:Nsim
[I_hat_call_2,err_std_call_2]=
monteCarloCallExo3(n*pas);
I_call_2(n)=I_hat_call_2;Err_call_2(n)=
err_std_call_2;
end
t_exo3=toc(start);
% fin de la boucle for pour les Nsim estimations
% exo3
% debut de la boucle for pour les Nsim
% estimations
% estimation du call avec la methode de l'exo4
start=tic;
for n=1:Nsim
[I_hat_call_3,err_std_call_3]=
monteCarloCallExo4(n*pas);
I_call_3(n)=I_hat_call_3;Err_call_3(n)=
err_std_call_3;
end
t_exo4=toc(start);
% fin de la boucle for pour les Nsim estimations
% exo4
% affichage des resultats et preparation de la
% figure
N=pas*(1:Nsim); % vecteur du nombre de simulation
fig_exo4=figure();
title('Exo 4 call seul: question 2 avec \alpha
=.95')
xlabel('Nombre de simulations')
ylabel('Valeurs des options');hold on
% ajout des temps de calculs
str = {sprintf('temps exo 4: %0.2es',t_exo4),...
sprintf('temps exo 3: %0.2es',t_exo3),...
sprintf('temps exo 2: %0.2es',t_exo2)};
text(pas*Nsim*7/14,.5,str)
% affichage du call exact
figEx_call=plot(N,C,'LineWidth',1.4,'Color','g');
% valeur exacte
% affichages pour la methode de l'exo3
figI_call_3=plot(N,I_call_3,'LineWidth',1.4,'
Color','b');
figHaut_call_3=plot(N,I_call_3+Z*Err_call_3,'
LineWidth',1.4,'Color','b');
figBas_call_3=plot(N,I_call_3-Z*Err_call_3,'
LineWidth',1.4,'Color','b');
% affichages pour la methode de l'exo2
figI_call_2=plot(N,I_call_2,'LineWidth',1.4,'
Color','m');
figHaut_call_2=plot(N,I_call_2+Z*Err_call_2,'
LineWidth',1.4,'Color','m');
figBas_call_2=plot(N,I_call_2-Z*Err_call_2,'
LineWidth',1.4,'Color','m');
% affichage du call par l'exo 1
figI_call_1=plot(N,I_call_1,'LineWidth',1.4,'
Color','r');
figHaut_call_1=plot(N,I_call_1+Z*Err_call_1,'
LineWidth',1.4,'Color','r');
figBas_call_1=plot(N,I_call_1-Z*Err_call_1,'
LineWidth',1.4,'Color','r');
% ajout des legendes a la figure
legend([figI_call_3,figHaut_call_3,
figBas_call_3,...
figI_call_2,figHaut_call_2,
figBas_call_2,...
figEx_call,figI_call_1,figHaut_call_1,
figBas_call_1],...
'estimation exo4','haute exo4','basse
exo4',...
'estimation exo3','haute exo3','basse
exo3',...
'exact call','estimation exo2','haute
exo2','basse exo2');
% on enregistre la figure sous format jpg
chem='images';chem=strcat(chem,'/exo4');
print(fig_exo4,chem,'-djpeg')
```

#### VII. Exercice 5

- On a :  $C = C^+ = \mathbb{E} \left[ (e^G - 1)_+ \right]$  il est simple de remarquer que  $C = C^- = \mathbb{E} \left[ (e^{-G} - 1)_+ \right]$  puisque les moments d'une variable



aléatoire ne dépend que de sa loi et  $G$  et  $-G$  ont la même loi car la loi normale centrée réduite est symétrique. Ainsi, on a :

$$C = \frac{C^+ + C^-}{2}$$

On peut utiliser la méthode de Monte Carlo pour estimer  $C^+$  et  $C^-$ . On a déjà justifié l'application de la méthode pour  $C^+$  dans l'exercice 2. L'application de la méthode à  $C^-$  est immédiatement justifiée par l'exercice 2 puisque l'intégrabilité ne dépend que de la loi de la variable aléatoire.

La garantie de meilleure variance que celle de l'approche de l'exercice 2 est justifiée par le fait que  $x \mapsto [e^x - 1]_+$  est monotone sur  $\mathbb{R}$

- Sur la figure 4 ci-dessous, nous avons les valeurs approchées du prix du call en fonction du nombre de simulations.

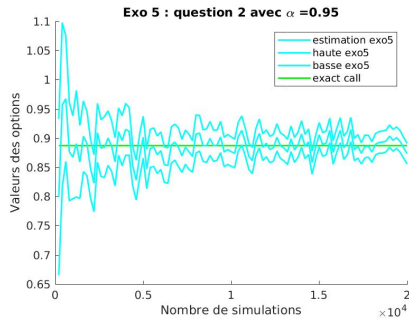


FIGURE 4. Simulation par variables antithétiques

- Sur la figure 5, comme nous l'avons dit dans la question 1. l'erreur est mieux contrôlée que par la méthode de l'exercice 1. Nous faisons également le même ordre de temps de calculs avec ces deux méthodes. Mais cette méthode ne réduit pas énormément la variance par rapport aux autres méthodes de l'exercice 3 et 4.

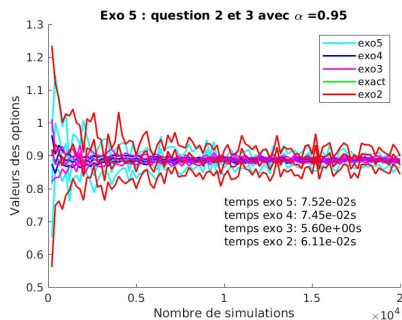


FIGURE 5. Comparaison graphique des méthodes

#### CODE MATLAB MÉTHODE EXERCICE 5

```
function [I_hat, err_std]=monteCarloCallExo5(N)
% Cette fonction fait une simulation de Monte
% en
% N echantillons en utilisant la symetrie de la
% loi normale centree reduite pour faire des v.
% a
% antithetiques
% ENTREE : N: Le nombre de simulation
% SORTIE : I_hat: La valeur approchée du call
% err_std: Erreur standard de la
% simulation realisee
% parametre du call
beta=1;K=1;
X=randn(); % simulation d'une variable de loi
% normale
% centree reduite
Y1=max(exp(beta*X)-K,0); % antithetique 1
Y2=max(exp(-beta*X)-K,0); % antithetique 2
S1_1=Y1; % somme partielle des Y1i
S2_1=Y1^2; % somme partielle des Y1i^2
S1_2=Y2; % somme partielle des Y2i
S2_2=Y2^2; % somme partielle des Y2i^2
m_y1y2=Y1*Y2; % la somme des produits des Y1i
% Y2i
% utile pour calculer la covariance
```

```
n=1;
while(n<N)
X=randn(); % on une normale centree reduite
Y1=max(exp(beta*X)-K,0); % antithetique 1
Y2=max(exp(-beta*X)-K,0); % antithetique 2
S1_1=S1_1+Y1; % mise-a-jour de S1_1
S2_1=S2_1+Y1^2; % mise-a-jour de S2_1
S1_2=S1_2+Y2; % mise-a-jour de S1_2
S2_2=S2_2+Y2^2; % mise-a-jour de S2_2
m_y1y2= m_y1y2+Y1*Y2; % mise-a-jour de
m_y1y2
n=n+1;
end
% on estime la variance par son estimateur sans
% biais
% pour la v.a Y1
v_1=(S2_1-N*(S1_1/N)^2)/(N-1);
% on estime la variance par son estimateur sans
% biais
% pour la v.a Y2
v_2=(S2_2-N*(S1_2/N)^2)/(N-1);
% la covariance
cov1_2=(m_y1y2-(S1_1*S1_2)/N)/(N-1);
% variance et ecart-type finaux est
variance=v_1/4+v_2/4+cov1_2/2;
s=sqrt(variance);
% on retourne l'estimation obtenue.
I_hat=(S1_1+S1_2)/(2*N);
% on retourne l'erreur standard de cette
% simulation
err_std=s/sqrt(N);
end
```

#### CODE MATLAB COMPARAISON EXERCICE 5

```
% monte carlo pour l'exercice 4
clc;close all
% le quantile d'ordre (alpha+1)/2 de la loi
% normale
% centree reduite.
alpha=.95;
Z=norminv((alpha+1)/2,0,1);
% pas et nombre d'estimations des options.
Nsim=100;pas=200;
% on initialise le vecteur des estimations
% et des erreurs. Pour trouver l'intervalle
% de confiance plus tard il suffirait d'uti-
% liser IC=[I-Z*s/sqrt(n),I+Z*s/sqrt(n)]
I_put=zeros(1,Nsim);
Err_put=I_put;
I_call_1=zeros(1,Nsim);
Err_call_1=I_call_1;
I_call_2=zeros(1,Nsim);
Err_call_2=I_call_2;
I_call_3=zeros(1,Nsim);
Err_call_3=I_call_3;
I_call_4=zeros(1,Nsim);
Err_call_4=I_call_4;
% parametres des options
K=1;beta=1;
% les valeurs exactes des options
C_ex=(exp(beta^2/2)*normcdf(beta-log(K)/beta,0,1)-
...
K*normcdf(-log(K)/beta,0,1));
C=C_ex*ones(1,Nsim);
% on augmente le nombre d'echantillons de pas a
% chaque
% fois pour obtenir Nsim simulation de chaque
% option
% debut de la boucle for pour les Nsim
% estimations
% estimation du call avec la methode de l'exo2
start=tic;
for n=1:Nsim
[I_hat_call_1, err_std_call_1]=monteCarloCall(
n*pas);
I_call_1(n)=I_hat_call_1;
Err_call_1(n)=err_std_call_1;
end
t_exo2=toc(start);
% fin de la boucle for pour les Nsim estimations
% exo2
% debut de la boucle for pour les Nsim
% estimations
% estimation du call avec la methode de l'exo3
start=tic;
for n=1:Nsim
[I_hat_call_2, err_std_call_2]=
monteCarloCallExo2(n*pas);
I_call_2(n)=I_hat_call_2; Err_call_2(n)=
err_std_call_2;
end
t_exo3=toc(start);
% fin de la boucle for pour les Nsim estimations
% exo3
```

```

1 % debut de la boucle for pour les Nsim
   estimations
2 % estimation du call avec la methode de l'exo4
3 start=tic;
4 for n=1:Nsim
5     [I_hat_call_3 , err_std_call_3]=
       monteCarloCallExo3(n*pas);
6     I_call_3(n)=I_hat_call_3 ; Err_call_3(n)=
       err_std_call_3 ;
7 end
8 t_exo4=toc(start);
9 % fin de la boucle for pour les Nsim estimations
   exo4
10 % debut de la boucle for pour les Nsim
    estimations
11 % estimation du call avec la methode de l'exo5
12 start=tic;
13 for n=1:Nsim
14     [I_hat_call_4 , err_std_call_4]=
       monteCarloCallExo5(n*pas);
15     I_call_4(n)=I_hat_call_4 ; Err_call_4(n)=
       err_std_call_4 ;
16 end
17 t_exo5=toc(start);
18 % fin de la boucle for pour les Nsim estimations
    exo5
19 % affichage des resultats et preparation de la
    figure
20 N=pas*(1:Nsim); % vecteur du nombre de simulation
21 fig_exo5=figure();
22 title('Exo 5 : question 2 avec \alpha =0.95')
23 xlabel('Nombre de simulations')
24 ylabel('Valeurs des options');hold on
25 % ajout des temps de calculs
26 str = { sprintf('temps exo 5: %0.2es',t_exo5) ,...
27         sprintf('temps exo 4: %0.2es',t_exo4) ,...
28         sprintf('temps exo 3: %0.2es',t_exo3) ,...
29         sprintf('temps exo 2: %0.2es',t_exo2) };
30 text(pas*Nsim*7/14,.,7,str)
31 % affichage du call exact
32 figEx_call=plot(N,C,'LineWidth',1.4,'Color','g');
    % valeur exacte
33 % affichages pour la methode de l'exo5
34 figI_call_4=plot(N,I_call_4,'LineWidth',1.4,'
    Color','c');
35 figHaut_call_4=plot(N,I_call_4+Z*Err_call_4,'
    LineWidth',1.4,'Color','c');
36 figBas_call_4=plot(N,I_call_4-Z*Err_call_4,'
    LineWidth',1.4,'Color','c');
37 % affichages pour la methode de l'exo4
38 figI_call_3=plot(N,I_call_3,'LineWidth',1.4,'
    Color','b');
39 figHaut_call_3=plot(N,I_call_3+Z*Err_call_3,'
    LineWidth',1.4,'Color','b');
40 figBas_call_3=plot(N,I_call_3-Z*Err_call_3,'
    LineWidth',1.4,'Color','b');
41 % affichages pour la methode de l'exo3
42 figI_call_2=plot(N,I_call_2,'LineWidth',1.4,'
    Color','m');
43 figHaut_call_2=plot(N,I_call_2+Z*Err_call_2,'
    LineWidth',1.4,'Color','m');
44 figBas_call_2=plot(N,I_call_2-Z*Err_call_2,'
    LineWidth',1.4,'Color','m');
45 % affichage du call par l'exo2
46 figI_call_1=plot(N,I_call_1,'LineWidth',1.4,'
    Color','r');
47 figHaut_call_1=plot(N,I_call_1+Z*Err_call_1,'
    LineWidth',1.4,'Color','r');
48 figBas_call_1=plot(N,I_call_1-Z*Err_call_1,'
    LineWidth',1.4,'Color','r');
49 ajout des legendes a la figure
50 legend([ figI_call_4 , figHaut_call_4 ,
    figBas_call_4 , figEx_call ],...
51 % 'estimation exo5', 'haute exo5', 'basse
    exo5', 'exact call');
52 legend([ figI_call_4 , figI_call_3 , figI_call_2 ,
    figEx_call , figI_call_1 ],...
53 % 'exo5', 'exo4', 'exo3', 'exact', 'exo2');
54 % on enregistre la figure sous format jpg
55 chem='images';chem=strcat(chem,'/exo5');
56 print(fig_exo5,chem,'-djpeg')
57 %% enregistrement en fichier CSV des
    caracteristiques des methodes
58 A=cell(2,5);
59 A(1,:)= { sprintf('Estim N= %d',Nsim*pas) , I_call_1 (
    Nsim) ,...
60           I_call_2 (Nsim) , I_call_3 (Nsim) , I_call_4 (
    Nsim) };
61 A(2,:)= { sprintf('Var N=%d',Nsim*pas) , Err_call_1 (
    Nsim) ^2 ,...
62           Err_call_2 (Nsim) ^2 , Err_call_3 (Nsim) ^2 ,
    Err_call_4 (Nsim) ^2 };
63 fid = fopen('comparaison.csv','w');
64 fprintf(fid,'%s %f %f %f %f\n',A{1,:});
65 fprintf(fid,'%s %f %f %f %f\n',A{2,:});
66 fclose(fid);
67 type comparaison.csv

```

## VIII. Exercice 6

1. Nous résumons dans le tableau ci-dessous les sorties des différentes méthodes dans ce TP.

Vals	Simple	Imp.Sampl	Controle	Ant. v.a
Estim N= 20000	0.8844178	0.8904634	0.8844906	0.8733639
Var N=20000	0.0001902	0.0000108	0.0000044	0.0000812

2. D'après le tableau de la question 1., nous constatons que la méthode de la variable de contrôle a la plus petite variance, donc c'est la meilleure méthode. La méthode des variables antithétiques est assez bonne et elle n'exige pas d'utiliser une relation entre le call et le put.