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 [1].

$$
\begin{aligned}
C_t &= \text{Le prix de l'option d'achat} \\
S_t &= \text{Le prix du sous-jacent à l'instant t} \\
K &= \text{Le strike(prix d'exercice)} \\
r &= \text{Le taux d'intérêt pour un placement} \\
\sigma &= \text{La volatilité du prix du sous-jacent} \\
T &= \text{La maturité de l'option.} \\
N &= \text{la fonction de répartition de la loi normale centrée réduite}
\end{aligned}
$$

III. Exercice 1

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

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

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

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

```matlab
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 approchee de la quantite
 %                      que nous voulons evaluer.
 %               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 evalue ensuite Y
 S1=Y; % somme partielle des Yi
 S2=Y^2; % somme patielle des Yi^2
 for n=1:N
     X=randn();% simulation d'une variable de loi normale
     Y=max(exp(beta*X)-K,0);% on evalue 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 hypothèses de la Loi Forte des Grands Nombres(LFGN) pour la convergence et le Théorème Centrale 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 $Z^1_{call} = (X-K)_+$ est $\mathbb{L}^1$ pour la LFGN et $\mathbb{L}^2$ pour le TCL. Or si Z^1_{call} est $\mathbb{L}^2$ alors Z^1_{call} est aussi $\mathbb{L}^1$. Donc il suffit de montrer que Z^1_{call} est $\mathbb{L}^2$. On a :

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

et

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

X suit la loi log-normale de paramètre $(0, \beta^2)$ donc X est $\mathbb{L}^2$. On en déduit que $\left|Z^1_{call}\right| = Z^1_{call}$ 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

```matlab
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 approchee de la quantite
  %                      que nous voulons evaluer.
  %               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 evalue ensuite Y
  S1=Y; % somme partielle des Yi
  S2=Y^2; % somme patielle 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 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
```

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

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

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

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

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

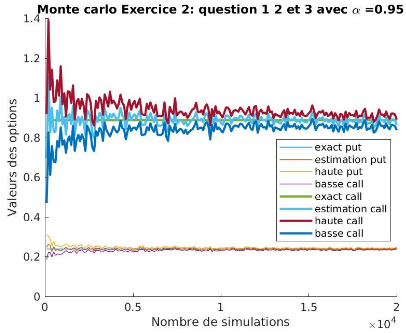

FIGURE 1. L' estimation et de l'erreur en fonction du nombre de simulations.

CODE MATLAB DE L'UTILISATION DES FONCTIONS PRÉCÉDENTES

```
%% monte carlo pour l'exercice 1
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);
% le nombre d'estimations des options.
Nsim=200;
% les pas entre la taille des echantillons
pas=100;
% on initialise le vecteur des estimations
% et des erreurs. Pour trouver l'intervalle
% de confiance plus tard il suffirat 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=zeros(1,Nsim);
Err_call=I_call;
% 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);
P_ex=(-exp(beta^2/2)*normcdf(-beta+log(K)/beta
    ,0,1)+...
    K*normcdf(log(K)/beta,0,1));
P=P_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
for n=1:Nsim
    % estimation du call
    [I_hat_call,err_std_call]=monteCarloCall(n*
        pas);
    I_call(n)=I_hat_call;
    Err_call(n)=err_std_call;
    % estimation du put
    [I_hat_put,err_std_put]=monteCarloPut(n*pas);
    I_put(n)=I_hat_put;
    Err_put(n)=err_std_put;
end
%fin de la bourcle for pour les Nsim estimations
% affichage des resultats
% preparation de la figure
N=pas*(1:Nsim); % vecteur du nombre de simulation
fig_exo1=figure();
title('Monte carlo Exercice 1: question 1 2 et 3
    avec \alpha =0.95')
xlabel('Nombre de simulations')
ylabel('Valeurs des options')
hold on
% affichage du put
figEx_put=plot(N,P,'LineWidth',.74);% valeur
    exacte
figI_put=plot(N,I_put,'LineWidth',.74);% l'
    estimation
figHaut_put=plot(N,I_put+Z*Err_put,'LineWidth'
    ,.74);% borne sup de l'IC
figBas_put=plot(N,I_put-Z*Err_put,'LineWidth'
    ,.74);% borne inf de l'IC
% affichage du call
figEx_call=plot(N,C,'LineWidth',2);% valeur
    exacte
figI_call=plot(N,I_call,'LineWidth',2);% l'
    estimation
figHaut_call=plot(N,I_call+Z*Err_call,'LineWidth'
    ,2);% borne sup de l'IC
figBas_call=plot(N,I_call-Z*Err_call,'LineWidth'
    ,2);% borne inf de l'IC
% ajout des legendes a la figure
legend([figEx_put, figI_put, figHaut_put,
    figBas_put,...
        figEx_call, figI_call, figHaut_call,
            figBas_call],...
        'exact put', 'estimation put','haute put'
            ,'basse call',...
        'exact call', 'estimation call','haute
            call','basse call');
 % on enregistre la figure sous format eps
chem='images';chem=strcat(chem,'/exo1');
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 $\beta = 1$) :

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

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 $Z^2_{call} = \frac{e^{\sqrt{2Y}}-1}{\sqrt{2Y}}$ 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 :

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

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.

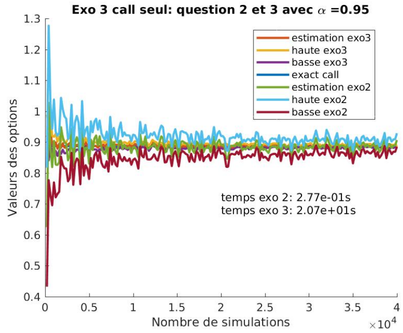

FIGURE 2. Comparaison entre les estimations et les erreurs.

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 patielle 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
% centree reduite.
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 suffirat 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 :
$$\forall t \in [0,T]\; C_t - P_t = S_t - Ke^{-r(T-t)} \text{donc} \quad t = T, C_T - P_T = S_T - K$$
donc $C_T - P_T = C - P = S_T - K$ 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 connaissance 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 mois 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.

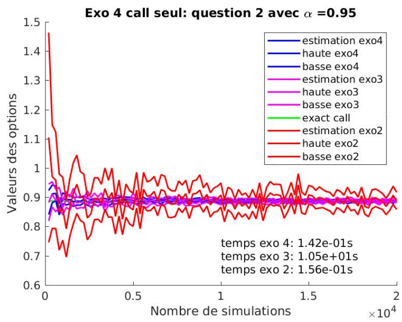

FIGURE 3. Comparaison entre les estimations et les erreurs.

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 approchee 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 suffirat 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 =0.95')
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,.7,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

1. On a : $C = C^{+} = \mathbb{E}\left[\left(e^{G}-1\right)_{+}\right]$ il est simple de remarquer que $C = C^{-} = \mathbb{E}\left[\left(e^{-G}-1\right)_{+}\right]$ 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 $\mathbb{R}$

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

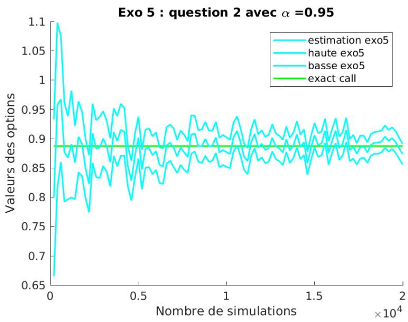

FIGURE 4. Simulation par variables antithétiques

3. 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.

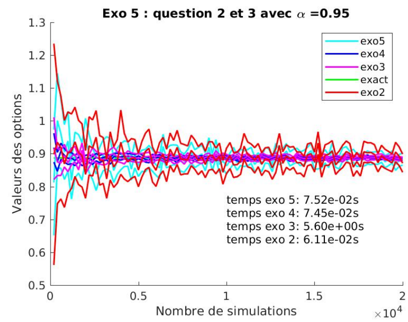

FIGURE 5. Comparaison graphique des méthodes

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 approchee 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_1=max(exp(beta*X)-K,0);% antithetique 1
 Y_2=max(exp(-beta*X)-K,0);% antithetique 2
 S1_1=Y_1; % somme partielle des Y1i
 S2_1=Y_1^2; % somme patielle des Y1i^2
 S1_2=Y_2; % somme partielle des Y2i
 S2_2=Y_2^2; % somme patielle des Y2i^2
 m_y1y2=Y_1*Y_2;% la somme des produits des Y1i*Y2i
 % utile pour calculer la covariance
 n=1;
 while(n<N)
     X=randn();% on une normale centree reduite
     Y_1=max(exp(beta*X)-K,0);% antithetique 1
     Y_2=max(exp(-beta*X)-K,0);% antithetique 2
     S1_1=S1_1+Y_1;    % mise-a-jour de S1_1
     S2_1=S2_1+Y_1^2;  % mise-a-jour de S2_1
     S1_2=S1_2+Y_2;    % mise-a-jour de S1_2
     S2_2=S2_2+Y_2^2;  % mise-a-jour de S2_2
     m_y1y2= m_y1y2+Y_1*Y_2;% 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 suffirat 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
```

```
% 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]=monteCarloCallExo3(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
% debut de la boucle for pour les Nsim estimations
% estimation du call avec la methode de l'exo5
start=tic;
for n=1:Nsim
    [I_hat_call_4,err_std_call_4]=monteCarloCallExo5(n*pas);
    I_call_4(n)=I_hat_call_4;Err_call_4(n)=err_std_call_4;
end
t_exo5=toc(start);
% fin de la boucle for pour les Nsim estimations exo5
% affichage des resultats et preparation de la figure
N=pas*(1:Nsim); % vecteur du nombre de simulation
fig_exo5=figure();
title('Exo 5 : question 2 avec \alpha =0.95')
xlabel('Nombre de simulations')
ylabel('Valeurs des options');hold on
% ajout des temps de calculs
str = {sprintf('temps exo 5: %0.2es',t_exo5),....
       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,.7,str)
% affichage du call exact
figEx_call=plot(N,C,'LineWidth',1.4,'Color','g'); % valeur exacte
% affichages pour la methode de l'exo5
figI_call_4=plot(N,I_call_4,'LineWidth',1.4,'Color','c');
figHaut_call_4=plot(N,I_call_4+Z*Err_call_4,'LineWidth',1.4,'Color','c');
figBas_call_4=plot(N,I_call_4-Z*Err_call_4,'LineWidth',1.4,'Color','c');
% affichages pour la methode de l'exo4
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'exo3
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'exo2
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_4, figHaut_call_4,figBas_call_4,figEx_call],...
%          'estimation exo5','haute exo5','basse exo5','exact call');
% legend([figI_call_4,figI_call_3,figI_call_2,figEx_call, figI_call_1],....
%          'exo5','exo4','exo3','exact', 'exo2');
 %% on enregistre la figure sous format jpg
chem='images';chem=strcat(chem,'/exo5');
print(fig_exo5,chem,'-djpeg')
%% enregistrement en fichier CSV des caracteritistiques des methodes
A=cell(2,5);
A(1,:)={sprintf('Estim N= %d',Nsim*pas),I_call_1(Nsim),...
        I_call_2(Nsim),I_call_3(Nsim),I_call_4(Nsim)};
A(2,:)={sprintf('Var N=%d',Nsim*pas),Err_call_1(Nsim)^2,...
        Err_call_2(Nsim)^2,Err_call_3(Nsim)^2,Err_call_4(Nsim)^2};
fid = fopen('comparaison.csv','w');
fprintf(fid,'%s, %f, %f,%f, %f\n',A{1,:});
fprintf(fid,'%s, %f,%f, %f, %f\n',A{2,:});
fclose(fid);
type comparaison.csv
```

VIII. Exercice 6

1. Nous résumons dans le tableau ci-dessous les sorties des differentes 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.