Mini projet 1: Calcul du prix d'une option asiatique

Valentin DE CRESPIN DE BILLY

Matthias LANG

09.01.2022

N. d'étudiant : 247067 et 313411

Université Catholique de l'Ouest

Mathématiques financières

Table des matières

1. Calculer le prix du sous-jacent

Nous avons essayé d'atteindre une équation qui ne dèpend que des variables connues comme la formule de Black-Scholes. Cela n'a pas fonctionné.

$$dS_t = S_t(rdt + \sigma\sqrt{S_t}dW_t) \tag{1}$$

$$\iff \frac{dS_t}{S_t} = rdt + \sigma\sqrt{S_t}dW_t \tag{2}$$

On prend l'équation 1 :

$$= dS_t \ = \ S_t rdt + \sigma S_t^{1.5} dW_t \quad \text{; Puis}$$

$$\begin{aligned} & d\langle S_t,\ S_t\rangle \\ & = \langle dS_t,\ dS_t\rangle \ = \\ & = \langle S_t rdt + \sigma S_t^{1.5} dW_t,\ S_t rdt + \sigma S_t^{1.5} dW_t\rangle \ = \\ & = \langle \sigma S_t^{1.5} dW_t,\ \sigma S_t^{1.5} dW_t\rangle \ = \\ & = S_t^3 \sigma^2 \langle dW_t,\ dW_t\rangle \ = \\ & = S_t^3 \sigma^2 dt \end{aligned}$$

On pose : $X_t = ln(S_t)$

$$\text{Formule d'Ito : } dln(S_t) = \frac{dS_t}{S_t} + \frac{1}{2}\frac{-1}{S_t^2} d\langle S_t,\ S_t\rangle \tag{3}$$

$$\tag{4}$$

Avec les équations 2 et 3 :

$$dln(S_t) = rdt + \sigma\sqrt{S_t}dW_t - \frac{1}{2}S_t\sigma^2 dt = (r - \frac{1}{2}S_t\sigma^2)dt + \sigma\sqrt{S_t}dW_t \tag{5}$$

$$\begin{aligned} ln(\frac{S_t}{S_0}) &= ln(S_t) - ln(S_0) = \int_0^t dln(S_u) = \\ &= \int_0^t (r - \frac{1}{2}S_t\sigma^2)du \ + \ \int_0^t \sigma\sqrt{S_t}dW_t \end{aligned}$$

$$= \dots$$

Donc on ne peut pas facilement dériver une formule pour le prix comme ça, qui dépend que des variables fixées, mais on peut le simuler pas à pas en utilisant (1) (méthode d'Euler) :

$$\begin{aligned} S_0 &\text{ soit connu} \\ dS_0 &= S_0(rdt + \sigma\sqrt{S_0}dW_0) \\ S_1 &\approx S_0 + dS_0 \\ dS_1 &= S_1(rdt + \sigma\sqrt{S_1}dW_1) \\ S_2 &\approx S_1 + dS_1 \\ &\dots \end{aligned} \tag{6}$$

1.1. Calculer le prix de l'action

K (soit choisi, soit calculé)

méthode de trapezes

calculer pay-off et actualiser

1.2. exo. 4) Calculer le prix d'exercice avec des intervalles

X_N will give a different estimator. This could be shown analytically, as even for $\sigma = 0$ the values will be different. X_{inf} uses the method of trapezes, giving the area under the whole graph, whereas X_N averages some values starting from T/Nd to T.

true values : $C_N \neq C$.

1.3. Réduction de la variance du éstimateur

Les éstimateurs ont une variance telle que : $\hat{Var}(C) = \hat{\sigma_i^2}/n_t$, oú n_t est le nombre des observations et $\hat{\sigma_i^2}$ est la variance estimée de la population, qui est égal à la variance de l'échantillon.

Supposons que nous ne connaissions ni les paramètres ni la règle à partir desquels les prix sont établis. Nous ne pouvons donc pas augmenter le nombre d'observations pour améliorer l'estimateur. Quelle autre possibilité existe-t-il pour réduire sa variance ?

Avec les techniques de bootstrap on pourrait répliquer les données. Mais on risque de introduir un biais. Si on utilise une variable de contrôle on n'invente pas des nouvelles données, ni risque-t-on de changer l'ésperance.

2. Réalisation numerique

Les algorithmes sont réalisées avec deux langues de programmation : Matlab et Visual Basic for Applications. Plusieurs graphiques sont y générés, vous les trouverez dans l'annexe A. En plus, avec le logiciel Excel nous avons crée un dashboard, voir une capture d'figure refX. Vous trouverez les scriptes et les images dans l'annexe, et avec la fiche de dashboard également dans le repository.

2.1. Variable antithétique

2.2. Variables de contrôle

Afin de réaliser une réduction de variance de l'éstimateur, sans être contraint des ressources de calculation, on peut se servire des variables des contrôle. Soit X la variable aléatoire dont on veut obtenir l'éstimateur Monte Carlo. Pour chaque X_i on peut simuler une autre variable Y_i, la variable de contrôle.

Cette variable est indépendante, mais correlée avec X. D'ailleurs on sait son ésperance. Avec les deux valeurs on creer une autre série de données ainsi : $Z_i = X_i - \lambda(Y_i - E(Y_i))$ avec $\lambda = sigma_X / sigma_Y * \rho$.

2.2.1. Variable de contrôle 1

La première variable est construite avec des bruits blancs, plus ou moins la même manière que le prix d'exercice d'avant. Problematique : Les X et Y ne sont pas independant en utilisant le meme processus ? Avec $W_{t_0} = S_0$:

$$X_i = \frac{1}{n} \int_0^i W_{t_i} dt$$

Comme pour la simulation des prix on calcule cette valeur avec la mèthode de trapezes.

C'est facile de démontrer l'espérance :.

$$\begin{aligned}
\mathbb{E}[X_i] &= \\
\mathbb{E}[\frac{1}{n}\int_0^i W_{t_i} dt] &= \\
\frac{1}{n}\mathbb{E}[\int_0^i W_{t_i} dt] &= \\
\frac{1}{n}\mathbb{E}[\int_0^i S_0 + dW_{t_i} dt] &= \\
\frac{1}{n}\mathbb{E}[\sum_0^i S_0 + dW_{t_i}] &= \\
\frac{1}{n}\mathbb{E}[iS_0 + \sum_0^i dW_{t_i}] &= \\
\frac{1}{n}(\mathbb{E}[iS_0] + \mathbb{E}[\sum_0^i dW_{t_i}]) &= \\
\frac{iS_0}{n}(1 + \mathbb{E}[\sum_0^i dW_{t_i}]) &= \\
\frac{iS_0}{n}(1 + \sum_0^i \mathbb{E}[dW_{t_i}]) = \frac{iS_0}{n}
\end{aligned}$$

Alors $\mathbb{E}[X_T] = S_0$.

Deux temporaires $[t, s]$ et $[v, u]$, sans chevauche $([t, s] \wedge [v, u] = \emptyset)$ ont valeurs de W qui sont indépendant. La variance, sachant que $\mathbb{E}[W_1 * W_2] = \mathbb{E}[W_{t_1}] * \mathbb{E}[W_{t_2}] = 0$ et $Var(W_{t_1}) = t_1$, donc $cov(W_{t_1}, W_{t_2}) = t_1$:

$$\begin{aligned}
var(X_i) &= var(\frac{1}{n}(S_0 + \sum_{i=1}^n W_{t_i})) \\
&= \frac{1}{n^2} var(\sum_{i=1}^n W_{t_i}) \\
&= \frac{1}{n^2}\mathbb{E}[(\sum_{i=1}^n W_{t_i})^2] \\
&= \frac{1}{n^2} var(\sum_{i=1}^n W_{t_i}) \\
&= \frac{1}{n^2}\sum_{i=1}^n cov(W_{t_i}, W_{t_i}) \\
&= \frac{1}{n^2}\sum_{i=1}^n \sum_{j=1}^n cov(W_{t_i}, W_{t_i})
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{n^2}(\sum_{i=1}^{n}\sum_{j=1}^{i} cov(W_{t_i}, W_{t_j}) + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} cov(W_{t_i}, W_{t_j})) \\
&= \frac{1}{n^2}(\sum_{i=1}^{n}\sum_{j=1}^{i} t_k + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} t_i) \\
&= \frac{1}{n^2}(T(\sum_{i=1}^{n}\sum_{j=1}^{i} \frac{k}{n} + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{i}{n})) \\
&= \frac{1}{n^2}(\frac{T}{n}(\sum_{i=1}^{n} \frac{i(i+1)}{2} + \sum_{i=1}^{n-1} i(n-i))) \\
&= \frac{1}{n^2}\frac{T}{n}(\frac{1}{6}n(n+1)(n+2) + \frac{1}{6}n(n-1)(n+1)) \\
&= \frac{1}{n^2}\frac{T}{6}(2n^2 + 3n + 1)
\end{aligned}
$$

Avec cela on calcule λ. λ peut-être calculé avec des données différentes. De toute façon le temps de calcul de λ n'est pas pris compte dans le temps de calcul affiché.

2.2.2. Variable de contrôle 2

Comme deuxième variable de contrôle sert un mouvement brownien des même pas que les trajectoires et avec $W_{t_0} = S_0$. Ce-ci suit que son ésperance égal S_0 et sa variance T. La covariance doit être calculée empiriquement. Avec un échantillon de $n = 200$ on calcule ainsi le λ avant qu'on simule les Z_i. D'où on n'utilize pas trop de calculations, le temps de calcul ne sera pas biais, et, le plus important, Z_i et λ seront indépendant de l'autre.

2.2.3. Variable de contrôle 3

En surplus, ces variables ne sont réalisées qu'en matlab.

Pour la troisième variable de contrôle on fait pareil que la deuxième, et on simule un prix d'une action décalée. Mais ici nous prenons en compte le taux d'interêt. Ça ne veut nécessairement dire le taux d'interêt, qui peut-être ne soit pas connu, mais le taux d'actualization. Avec ces trajectoires simulées on garde les dernier valeurs à temps T.

Les trajetoires seront simulées ensuite :

soit connu S_0, et soit $dS_0 = S_0 * r * dt$

$$\text{for} \quad i \;=\; 1:n$$
$$dS_i = S_{i-1}(r * dt + \frac{dW_i}{10})$$
$$S_i = S_{i-1} + dS_i$$

Cette algorithme peut être également exprimée comme somme :

$$S_i = S_0 + \sum_0^i dS_i$$

En espérance :

$$\mathbb{E}[S_t] = \mathbb{E}[S_0 + \sum_0^t dS_i] =$$

$$\mathbb{E}[S_0 + S_0 * r * dt + \sum_1^t dS_i] =$$
$$= S_0 * (1 + rdt) + \sum_1^t \mathbb{E}[dS_i] =$$
$$= S_0 * (1 + rdt) + \sum_1^t \mathbb{E}[S_{i-1}(r * dt + 0.1dW_i)] =$$
$$= S_0 * (1 + rdt) + \sum_1^t \mathbb{E}[S_{i-1}] * \mathbb{E}[r * dt + 0.1dW_i] =$$
$$= S_0 * (1 + rdt) + \sum_1^t \mathbb{E}[S_{i-1}] * \mathbb{E}[r * dt] + 0.1\mathbb{E}[dW_i] =$$
$$= S_0 * (1 + rdt) + \sum_1^t \mathbb{E}[S_{i-1}] * \mathbb{E}[r * dt] =$$
$$= S_0 * (1 + rdt) + \sum_0^{t-1} \mathbb{E}[S_i] * r * dt$$
$$= S_0 + \sum_1^t \mathbb{E}[S_{i-1}] * r * dt$$

Cela va donner au-peu-près la valeur d'une obligation à valeur faciale de S_0 et coupons au taux r ($\mathbb{E}[S_T] \approx S_0 * (1 + r)^T$), mais son espérance peut être calculée pas à pas :

$$\mathbb{E}[S_i|\mathbb{F}_{i-1}] = S_0 * (1 + rdt) + \sum_1^i S_{i-1} * r * dt$$

C'est pratique de remplacer la variance par sa contrepartie empirique.

2.2.4. Variable de contrôle 4

Là il faut calculer l'espérance, car le suivant est apparemment faux :

```
E_C_VC = S0;
for i=2:n
    E_C_VC = E_C_VC*(1 + r*dt);
end
E_C_VC = max(double(E_C_VC-K),0);
```

Voir les captures d'écran dans le dossier ou voir les notes du cours.

2.2.5. Variable de contrôle 5

$= exp(W_T/S_0)$

Espérance à démontrer : $exp(T/(2 * S_0^2))$

Appendices

Toutes les fiches se trouvent dans le repository en ligne :

`https://github.com/matthias-10/UCO_actuariat_mini-projet`

A. Graphiques

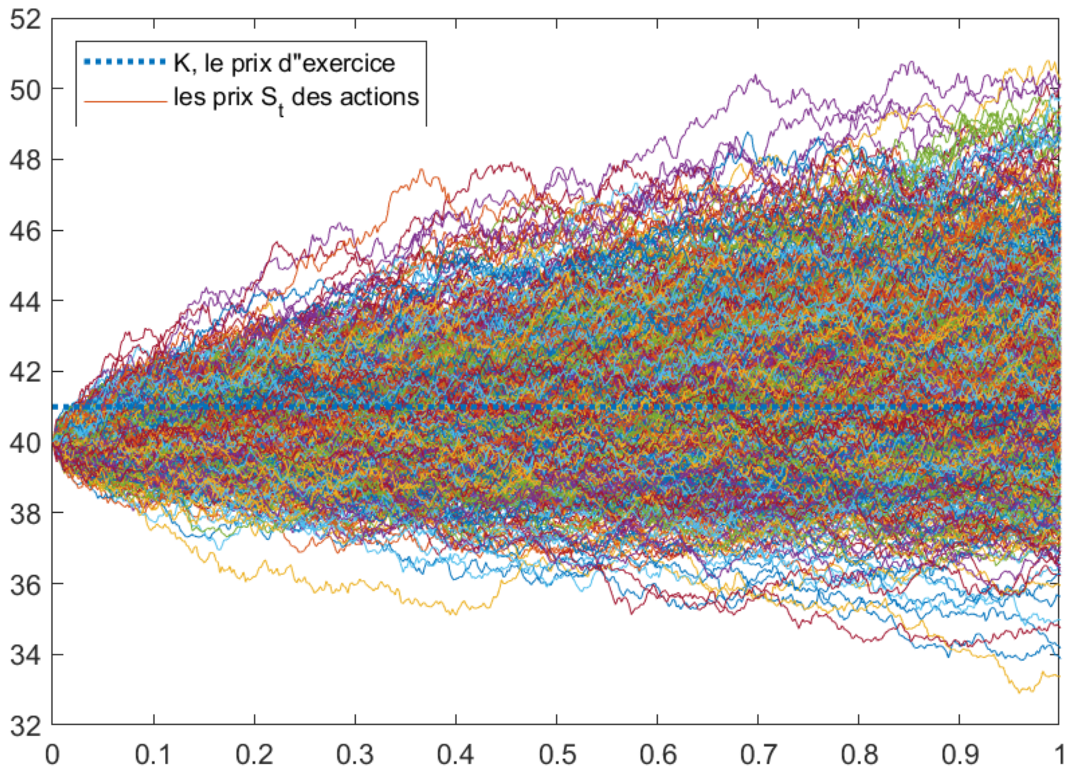

FIGURE 1 – Les graphes de tous trajectoires, plotés avec matlab

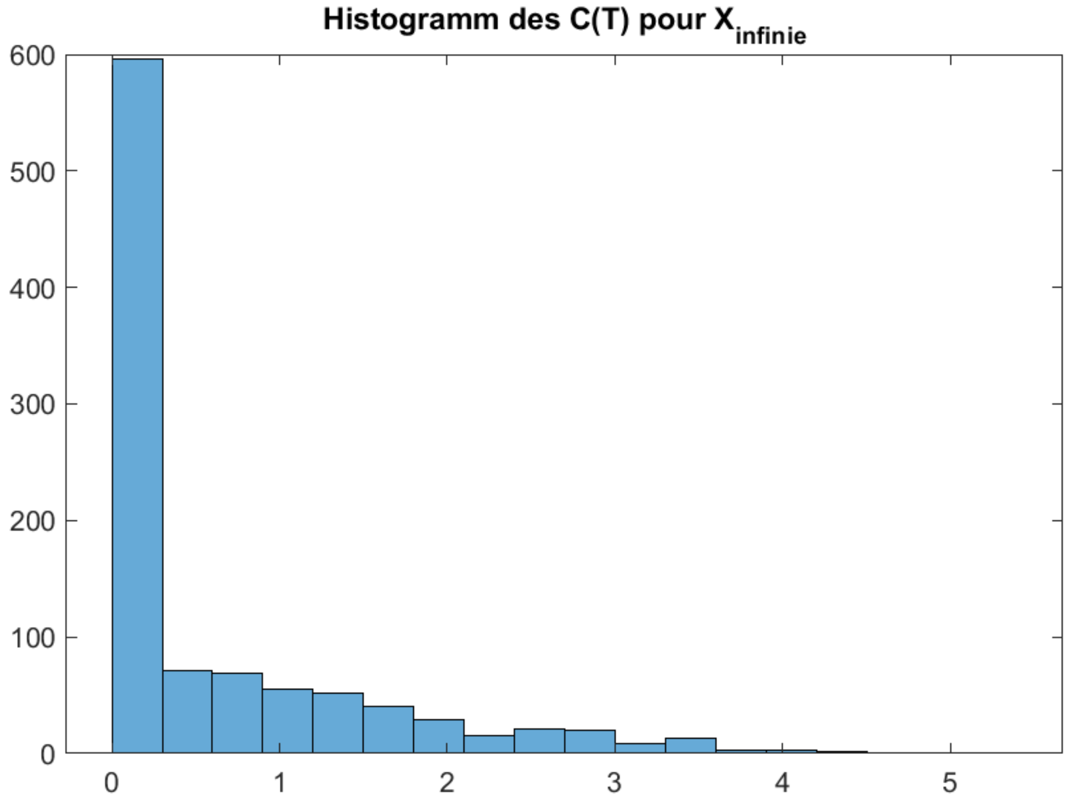

Figure 2 – Histogramme des simulations pour C_∞

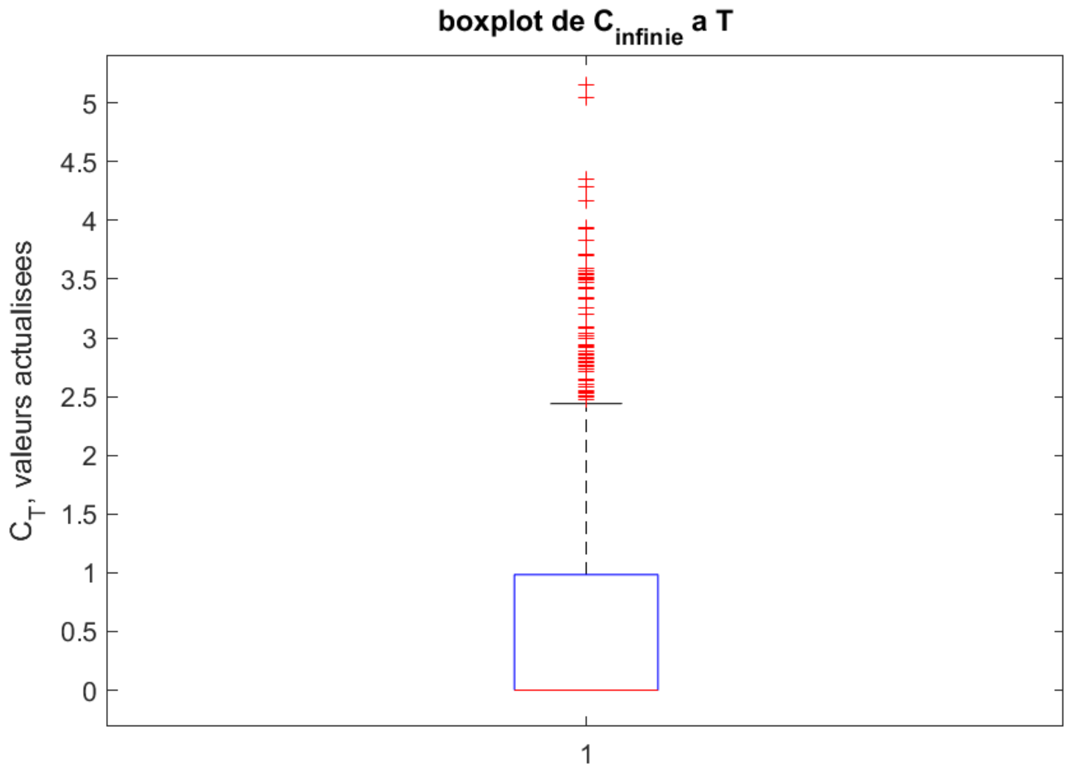

Figure 3 – Boxplot des simulations pour C_∞

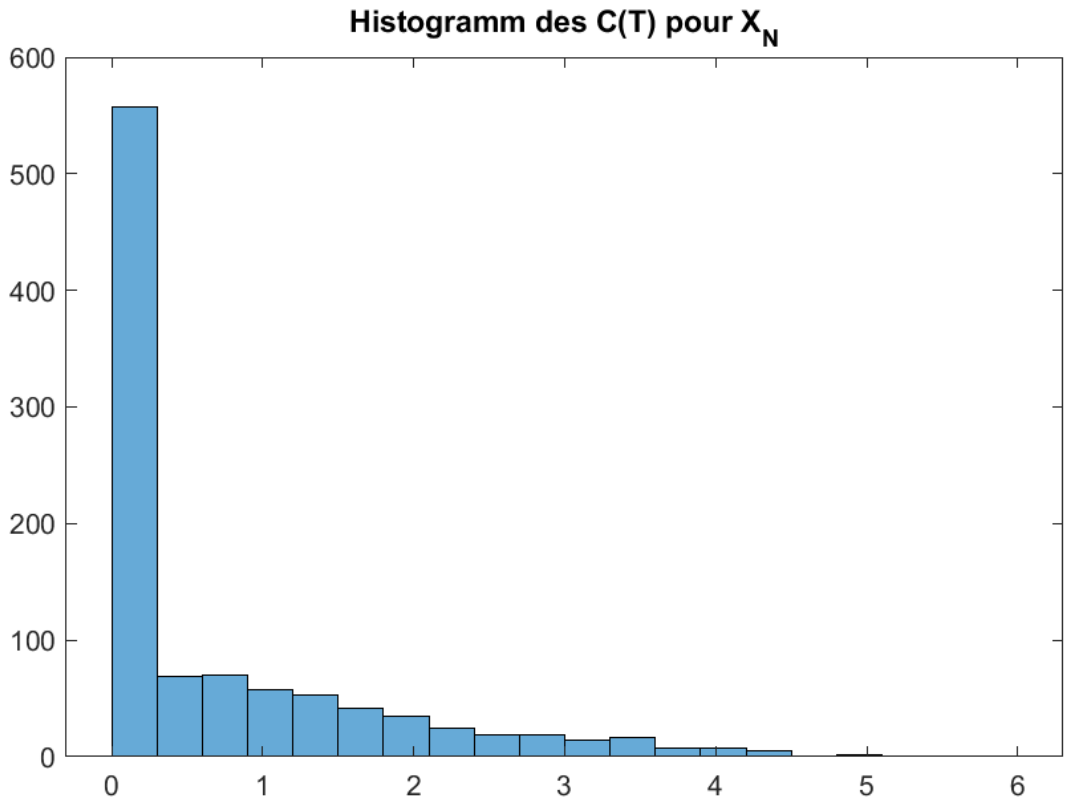

FIGURE 4 – Histogramme des simulations pour C_N

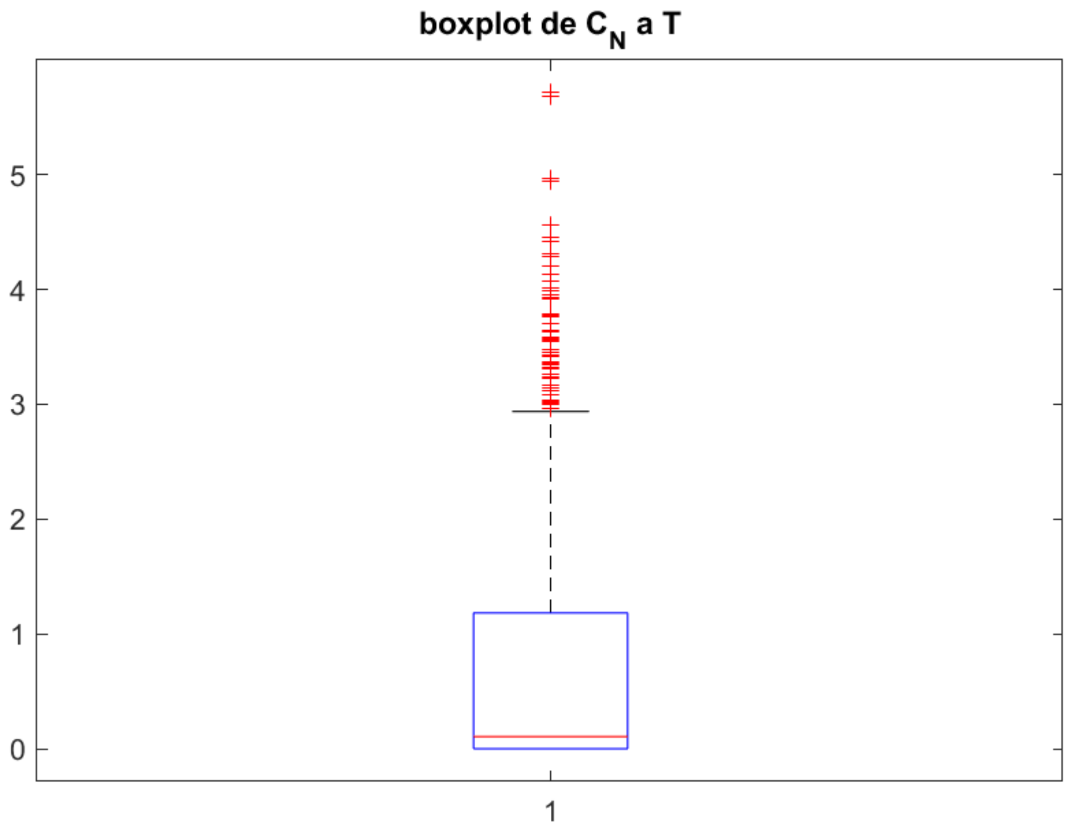

FIGURE 5 – Boxplot des simulations pour C_N

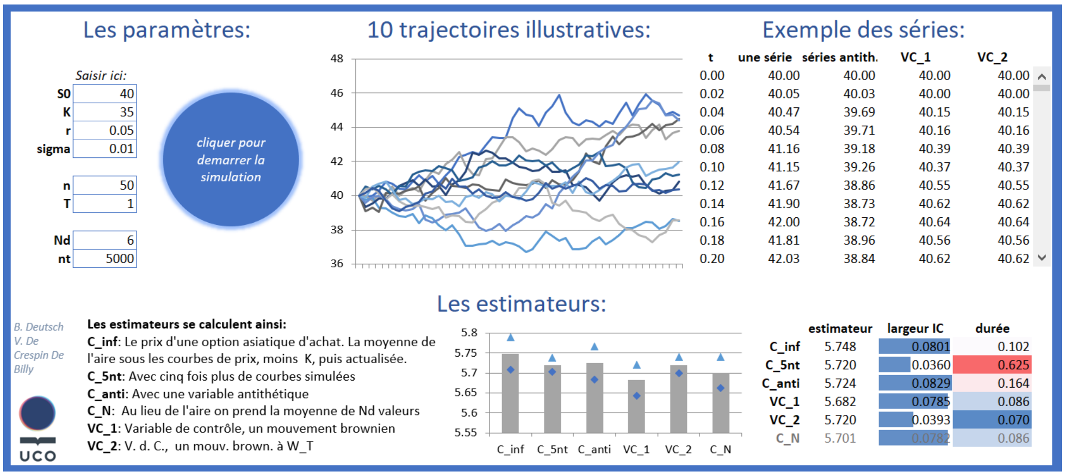

FIGURE 6 – le Excel dashboard

B. Code Matlab

```matlab
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ %
% Valentin DE CRESPIN DE BILLY                      UTF-8 %
% Matthias LANG                                30.11.2021 %
% exige:                                                  %
% - Statistics and Machine Learning Toolbox               %
% - Symbolic Math Toolbox                                 %
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ %

% ~~~~~~ Mathematiques financieres: Mini-projet 1 ~~~~~~~ %

%% ~~~~~~~~~~~~~~~~~~~~ Parametres ~~~~~~~~~~~~~~~~~~~~~ %%

S0 = 40;                % Prix initial du sous jacent
r = 0.05;               % Taux d'interet sous risque neutre
sigma = 0.01;           % Variance partie fixe

n = 2^6;                % Nombre de intervalles
T = 5;                  % Fin de la periode/exercice = tau
Nd = 8;                 % Nombre des sous-intervalles

nt = 10000;            % Nombre de trajectoires

alpha = 0.05;           % niveau au risque


%% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ %%

nlambda = 500; % sim.s pour determiner le lambda des VC

if Nd > n/2-1
    warning("Le nombre de sous-intervalles est tres petit")
    fprintf('Il fallait Nd << n')
end
```

```

fprintf('\n ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ \n');
fprintf('La programme a demarre a %s \n', datetime('now'));

% K base sur le prix moyen d'une obligation sans risque
syms func(x)
obligation(x) = S0*(1+r)^x;
K = double( int(obligation,0,T)/T);
bonds_T = obligation(T);

fprintf('%d -> Prix initial du sous jacent \n', S0)
fprintf(['%0.5g -> Prix univers risque neutre'...
        'et continue a T\n'], bonds_T)
fprintf('%0.5g -> Prix d''exercice de l''option \n', K);
fprintf('calculation en cours . . .\n')

dt = T/n;
t = 0:dt:T;


%% ~~~~~~~~~~~~~~~~~ Monte-Carlo pur ~~~~~~~~~~~~~~~~~~~ %%

tic

S = S0 * ones(nt, 1);
X = S/2;

for i = 2:(n+1)
    dW_t = normrnd(0, sqrt(dt), nt, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    X = X+S;
end

X = (X - S/2)/n;
```

```
C = exp(-r*T) * max(X-K,0);

% ~ Estimateur ~
% C_inf * exp(-rT) est une martingale donc
% E[exp(-rT)*C_inf]= C_inf(S_0)

% C
C_est = mean(C);
C_est_var = var(C)/nt; %/nt ?

C_IC_inf = C_est + sqrt(C_est_var)*norminv(alpha/2);
C_IC_sup = C_est + sqrt(C_est_var)*norminv(1-alpha/2);
L = C_IC_sup-C_IC_inf;
tps = toc;

% fonction d'affichage


fprintf('\n')
fprintf('%d trajectoires simules\n', nt);

fprintf('\n')
fprintf('estimateur Monte-Carlo: \n');

disp(strcat(...
{' C = '},sprintf('%05.3f',C_est),...
{' IC = ['},sprintf('%05.3f',C_IC_inf),...
{' , '},sprintf('%05.3f',C_IC_sup),...
{'] '},...
{' largeur = '},sprintf('%05.3f',L),...
{' t = '},sprintf('%05.3f',tps),...
{' eff = '},sprintf('%05.3f',L * sqrt(tps))));
```

```
%% ~~~~~~~~~~~~ C_N: calcul avec X_T_prim ~~~~~~~~~~~~~~ %%

tic

S = S0 * ones(nt, 1);
X = zeros(nt, 1);
l=1;

for i = 2:(n+1)
    dW_t = normrnd(0, sqrt(dt), nt, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    if (i/n) > (l/Nd)
        X = X+S;
        l = l+1;
    end
end

X = (X+S)/Nd;
C = exp(-r*T) * max(X-K,0);

% C
C_N_est = mean(C);
C_N_est_var = var(C)/nt; %/nt ?

C_N_IC_inf = C_N_est + sqrt(C_N_est_var)*norminv(alpha/2);
C_N_IC_sup = C_N_est + sqrt(C_N_est_var)*norminv(1-alpha/2);
L = C_N_IC_sup-C_N_IC_inf;
tps = toc;

% fonction d'affichage

fprintf('pour X_prime: \n');

disp(strcat(...
```

```
{' C = '},sprintf('%05.3f',C_N_est),...
{' IC = ['},sprintf('%05.3f',C_N_IC_inf),...
{' , '},sprintf('%05.3f',C_N_IC_sup),...
{'] '},...
{' largeur = '},sprintf('%05.3f',L),...
{' t = '},sprintf('%05.3f',tps),...
{' eff = '},sprintf('%05.3f',L * sqrt(tps))));


%% ~~~~~~~~~~~~~~~ variable antithetique ~~~~~~~~~~~~~~~ %%

tic

S = S0 * ones(nt, 1);
Sa = S;
X = S/2;
Xa = X;

% Simulation pas a pas
for i = 1:n
    dW_t = normrnd(0, sqrt(dt), nt, 1);
    S = S .* (1 + r*dt + sigma*sqrt(abs(S)).*dW_t );
    Sa = Sa .* (1 + r*dt - sigma*sqrt(abs(Sa)).*dW_t );
    X = X + S;
    Xa = Xa + Sa;
end

X = (X - S/2)/n;
Xa = (Xa - Sa/2)/n;

Ca = exp(-r*T)*(max(Xa - K,0) + max(X-K,0))/2;
Ca_est = mean(Ca);
Ca_est_var = var(Ca)/(2*nt);

```

```
Ca_IC_inf = Ca_est + sqrt(Ca_est_var)*norminv(alpha/2);
Ca_IC_sup = Ca_est + sqrt(Ca_est_var)*norminv(1-alpha/2);

L = Ca_IC_sup-Ca_IC_inf;
tps = toc;

% fonction d'affichage

fprintf('avec une variable antithetique: \n');

disp(strcat(...
{' C = '},sprintf('%05.3f',Ca_est),...
{' IC = ['},sprintf('%05.3f',Ca_IC_inf),...
{' , '},sprintf('%05.3f',Ca_IC_sup),...
{'] '},...
{' largeur = '},sprintf('%05.3f',L),...
{' t = '},sprintf('%05.3f',tps),...
{' eff = '},sprintf('%05.3f',L * sqrt(tps))));



% efficace ?
%co = cov([X Xa]);
%fprintf(['La covariance entre X et la variable '...
%         'antithetique est: %0.5g \n'], co(2,2))



%% ~~~~~~~~~~~~~ variable de controle 1 ~~~~~~~~~~~~~~~~ %%

% Variable de controle - aire sous un mouvement brownien
% ~~~~~~~~~~~~~~~~ Calculer lambda ~~~~~~~~~~~~~~~~~~~~~~ %

S = S0 * ones(nlambda,1);
VC = S0 * ones(nlambda, 1);
X = S/2;
```

```
VC_aire = S/2;
for i = 2:(n+1)
    dW_t = normrnd(0, sqrt(dt), nlambda, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    X = X+S;
    VC = VC + dW_t;
    VC_aire = VC_aire + VC;
    % succesion des petits accroissements du brownien
    % => VC et S sont correles
end
X = (X - S/2)/n;
VC_aire = (VC_aire - VC/2)/n;

C = exp(-r*T) * max(X-K,0);
A = cov(C, VC_aire);
sigma2 = (T/6*(2*n^2+3*n+1))/n^2;
lambda = A(1,2)/sigma2;

% en utilisant les est. empiriques
% comparer avec les moments analytiques? -> voir pdf/VBA

% ~~~~~~~~~~~~~~~~~~~~~ simuler ~~~~~~~~~~~~~~~~~~~~~~~~~ %

tic;

S = S0 * ones(nt,1);
VC = S0 * ones(nt, 1);
X = S/2;
VC_aire = X;
for i = 2:(n+1)
    dW_t = normrnd(0, sqrt(dt), nt, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    X = X+S;
    VC = VC + dW_t;
```

```
    VC_aire = VC_aire + VC;
    % succesion des petits accroissements du brownien
    % => VC et S sont correles
end
X = (X - S/2)/n;
VC_aire = (VC_aire - VC/2)/n;

C = exp(-r*T) * max(X-K,0);

Z = C - lambda * (VC_aire - S0);

C_1_est = mean(Z);
C_1_est_var = var(Z)/nt;

C_1_IC_inf = C_1_est + sqrt(C_1_est_var)*norminv(alpha/2);
C_1_IC_sup = C_1_est + sqrt(C_1_est_var)*norminv(1-alpha/2);
L = C_1_IC_sup - C_1_IC_inf;

tps = toc;

% fonction d'affichage

fprintf('Variable de controle 1- aire sous W_t: \n');

disp(strcat(...
{' C = '},sprintf('%05.3f',C_1_est),...
{' IC = ['},sprintf('%05.3f',C_1_IC_inf),...
{' , '},sprintf('%05.3f',C_1_IC_sup),...
{'] '},...
{' largeur = '},sprintf('%05.3f',L),...
{' t = '},sprintf('%05.3f',tps),...
{' eff = '},sprintf('%05.3f',L * sqrt(tps))));

p = corr(X, VC_aire);
```

```
% optimum: lambda =~ corr(X,Y)*(Var(X)/Var(Y))^.5
% efficace ?
fprintf(['La correlation entre X et la variable '...
        'de controle est: %0.5g\n'], p)
fprintf(['La covariance empirique = %0.5g ; \n' ...
         'La variance calculee = %0.5g\n'], A(2,2), sigma2)



%% ~~~~~~~~~~~~~ variable de controle 2 ~~~~~~~~~~~~~~~~ %%

% Variable de controle - somme de dW_t:
% correspond a un mouvement brownien
% ~~~~~~~~~~~~~~~~ Calculer lambda ~~~~~~~~~~~~~~~~~~~~~~ %

S = S0 * ones(nlambda,1);
VC = S0 * ones(nlambda, 1);
X = S/2;
for i = 2:(n+1)
    dW_t = normrnd(0, sqrt(dt), nlambda, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    X = X+S;
    VC = VC + dW_t;
    % succesion des petits accroissements du brownien
    % => VC et S sont correles
end
X = (X - S/2)/n;

C = exp(-r*T) * max(X-K,0);
A = cov(C, VC);
lambda = A(1,2)/A(2,2);

% en utilisant les est. empiriques
% comparer avec les moments analytiques? -> voir pdf/VBA

```

```
% ~~~~~~~~~~~~~~~~~~~~~ simuler ~~~~~~~~~~~~~~~~~~~~~~~~~ %

tic;

S = S0 * ones(nt,1);
VC = S0 * ones(nt, 1);
X = S/2;
for i = 2:(n+1)
    dW_t = normrnd(0, sqrt(dt), nt, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    X = X+S;
    VC = VC + dW_t;
end
X = (X - S/2)/n;
C = exp(-r*T) * max(X-K,0);

Z = C - lambda * (VC - S0);

C_2_est = mean(Z);
C_2_est_var = var(Z)/nt;

C_2_IC_inf = C_2_est + sqrt(C_2_est_var)*norminv(alpha/2);
C_2_IC_sup = C_2_est + sqrt(C_2_est_var)*norminv(1-alpha/2);
L = C_2_IC_sup - C_2_IC_inf;

tps = toc;

% fonction d'affichage

fprintf('\n')
fprintf('Variable de controle 2 - somme de dW_t: \n');

disp(strcat(...
{' C = '},sprintf('%05.3f',C_2_est),...
```

```
{' IC = ['},sprintf('%05.3f',C_2_IC_inf),...
{' , '},sprintf('%05.3f',C_2_IC_sup),...
{'] '},...
{' largeur = '},sprintf('%05.3f',L),...
{' t = '},sprintf('%05.3f',tps),...
{' eff = '},sprintf('%05.3f',L * sqrt(tps))));


%p = corr(X, VC);
% optimum: lambda =~ corr(X,Y)*(Var(X)/Var(Y))^.5
% efficace ?
%fprintf(['La correlation entre X et la variable '...
%        'de controle est: %0.5g\n'], p)



%% ~~~~~~~~~~~~~ variable de controle 3 ~~~~~~~~~~~~~~~~ %%

% mouvement pareil au prix des actions decale
% ~~~~~~~~~~~~~~~~ Calculer lambda ~~~~~~~~~~~~~~~~~~~~~~ %
% S = S0*ones(n+1,1);
% VC = S0*ones(n+1,1);
%
% i = 2;
% dW_t = normrnd(0, sqrt(dt), 1, 1);
% S(i) = S(i-1) .*(1 +r*dt + sigma*sqrt(abs(S(i-1))).*dW_t );
% VC(i) = S0 * (1+ r*dt);
%
% for i = 3:(n+1)
%     j = mod(i,2);
%     dW_t = normrnd(0, sqrt(dt), 1, 1);
%     S(i) = S(i-1) .*(1 +r*dt + sigma*sqrt(abs(S(i-1))).*dW_t );
%
%     VC(i) = VC(i-1) .*(1 +r*dt + dW_t/10 );
%     VC(i) = VC(i);
% end
```

```
% plot(S)
% hold on
% plot(VC)
% hold off

S = S0 * ones(nlambda,1);
VC = S0 * ones(nlambda, 1);

X = S/2;
for i = 2:(n+1)
    %j = mod(i,2);
    dW_t = normrnd(0, sqrt(dt), nlambda, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    X = X+S;

    %VC(:,3) = VC(:,j+1) .*(1 +r*dt + dW_t/10 );
    %VC(:,~j+1) = VC(:,3);
    VC = VC .* (1 +r*dt + dW_t/10 );
end
X = (X - S/2)/n;

C = exp(-r*T) * max(X-K,0);
A = cov(C, VC);
lambda = A(1,2)/A(2,2);



% E(VC)
E_VC = S0;
for i = 2:(n+1)
    E_VC = E_VC * (1 + r*dt);
end

% pour comparaison
double(bonds_T);
```

```
% en utilisant les est. empiriques
% comparer avec les moments analytiques? -> voir pdf/VBA

% ~~~~~~~~~~~~~~~~~~~~~ simuler ~~~~~~~~~~~~~~~~~~~~~~~~~ %

tic;
S = S0 * ones(nt,1);
VC = S0 * ones(nt, 1);

X = S/2;
for i = 2:(n+1)
    %j = mod(i,2);
    dW_t = normrnd(0, sqrt(dt), nt, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    X = X+S;

    %VC(:,3) = VC(:,j+1) .*(1 +r*dt + dW_t/10 );
    %VC(:,~j+1) = VC(:,3);
    VC = VC .* (1 +r*dt + dW_t/10 );
end
X = (X - S/2)/n;


C = exp(-r*T) * max(X-K,0);

Z = C - lambda * (VC - E_VC);

C_3_est = mean(Z);
C_3_est_var = var(Z)/nt;

C_3_IC_inf = C_3_est + sqrt(C_3_est_var)*norminv(alpha/2);
C_3_IC_sup = C_3_est + sqrt(C_3_est_var)*norminv(1-alpha/2);
L = C_3_IC_sup - C_3_IC_inf;
```

```
tps = toc;

% fonction d'affichage

fprintf('Variable de controle 3 - prix decales: \n');

disp(strcat(...
{' C = '},sprintf('%05.3f',C_3_est),...
{' IC = ['},sprintf('%05.3f',C_3_IC_inf),...
{' , '},sprintf('%05.3f',C_3_IC_sup),...
{'] '},...
{' largeur = '},sprintf('%05.3f',L),...
{' t = '},sprintf('%05.3f',tps),...
{' eff = '},sprintf('%05.3f',L * sqrt(tps))));

p = corr(X, VC);
% optimum: lambda =~ corr(X,Y)*(Var(X)/Var(Y))^.5
% efficace ?
fprintf(['La correlation entre X et la variable '...
        'de controle est: %0.5g\n'], p)

%% ~~~~~~~~~~~~~ variable de controle 4 ~~~~~~~~~~~~~~~~ %%

% calculer des option europeenes, mais avec des actions:
% S = S .*(1 +r*dt + sigma*sqrt(S0)*dW_t );
% sigma et r soient connu ici.
% alors qu'on puisse calculer leur payoff avec Balckes-Sch.
% ~~~~~~~~~~~~~~~~ Calculer lambda ~~~~~~~~~~~~~~~~~~~~~~ %

S = S0 * ones(nlambda,1);
VC = S;
X = S/2;
% VC = zeros(nlambda,1);
```

```
for i = 2:(n+1)
    dW_t = normrnd(0, sqrt(dt), nlambda, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    X = X+S;

    VC = VC .*(1 +r*dt + sigma*sqrt(S0).*dW_t );
end
X = (X - S/2)/n;
%Black-scholes: P_T, w_t = W_T
% sig_vc = (sigma*sqrt(S0));
% a=S0*exp(T*(r-0.5*sig_vc^2) + sig_vc*VC);
% b=VC+S0;
% apres B-S a=b?

d1 = log(VC/K);
d2 = d1; % a temps T
C_VC = S.*normcdf(d1) - K*exp(-r*T).*normcdf(d2);

C = exp(-r*T) * max(X-K,0);
A = cov(C, C_VC);
lambda = A(1,2)/A(2,2);

E_C_VC = S0;
for i=2:n
    E_C_VC = E_C_VC*(1 + r*dt);
end
E_C_VC = max(double(E_C_VC-K),0);

% ~~~~~~~~~~~~~~~~~~~~~ simuler ~~~~~~~~~~~~~~~~~~~~~~~~~ %

tic;

S = S0 * ones(nt,1);
X = S/2;
```

```
VC = S;

for i = 2:(n+1)
    dW_t = normrnd(0, sqrt(dt), nt, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    X = X+S;

    VC = VC .*(1 +r*dt + sigma*sqrt(S0).*dW_t );
end
X = (X - S/2)/n;

d1 = log(VC/K);
d2 = d1; % a temps T
C_VC = S.*normcdf(d1) - K*exp(-r*T).*normcdf(d2);
C_VC = max(C_VC, 0);
C = exp(-r*T) * max(X-K,0);

%Z = C - lambda * (C_VC - E_C_VC);
Z = C - lambda * (C_VC -mean(C_VC));

C_4_est = mean(Z);
C_4_est_var = var(Z)/nt;

C_4_IC_inf = C_4_est + sqrt(C_4_est_var)*norminv(alpha/2);
C_4_IC_sup = C_4_est + sqrt(C_4_est_var)*norminv(1-alpha/2);
L = C_4_IC_sup - C_4_IC_inf;

tps = toc;

% fonction d'affichage

fprintf('\n')
fprintf('Variable de controle 4 - option europeenne: \n');
```

```
disp(strcat(...
{' C = '},sprintf('%05.3f',C_4_est),...
{' IC = ['},sprintf('%05.3f',C_4_IC_inf),...
{' , '},sprintf('%05.3f',C_4_IC_sup),...
{'] '},...
{' largeur = '},sprintf('%05.3f',L),...
{' t = '},sprintf('%05.3f',tps),...
{' eff = '},sprintf('%05.3f',L * sqrt(tps))));

p = corr(C, C_VC);
% optimum: lambda =~ corr(X,Y)*(Var(X)/Var(Y))^.5
% efficace ?
fprintf(['La correlation entre X et la variable '...
        'de controle est: %0.5g\n'], p)


%% ~~~~~~~~~~~~~ variable de controle 5 ~~~~~~~~~~~~~~~~ %%


% comme VC2 mais exp(W_T); W_0 = 0
% ~~~~~~~~~~~~~~~~ Calculer lambda ~~~~~~~~~~~~~~~~~~~~~~ %

S = S0 * ones(nlambda,1);
VC = zeros(nlambda, 1);
X = S/2;
for i = 2:(n+1)
    dW_t = normrnd(0, sqrt(dt), nlambda, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    X = X+S;
    VC = VC + dW_t;
    % succesion des petits accroissements du brownien
    % => VC et S sont correles
end
X = (X - S/2)/n;
%VC = sign(VC).*exp(abs(VC));
VC = exp(VC/S0);
```

```
C = exp(-r*T) * max(X-K,0);
A = cov(C, VC);
lambda = A(1,2)/A(2,2);

% en utilisant les est. empiriques
% comparer avec les moments analytiques? -> voir pdf/VBA

% ~~~~~~~~~~~~~~~~~~~~~ simuler ~~~~~~~~~~~~~~~~~~~~~~~~~ %

tic;

S = S0 * ones(nt,1);
VC = zeros(nt, 1);
X = S/2;
for i = 2:(n+1)
    dW_t = normrnd(0, sqrt(dt), nt, 1);
    S = S .*(1 +r*dt + sigma*sqrt(abs(S)).*dW_t );
    X = X+S;
    VC = VC + dW_t;
    % succesion des petits accroissements du brownien
    % => VC et S sont correles
end
X = (X - S/2)/n;
VC = exp(VC/S0);
%VC = sign(VC).*exp(abs(VC));
C = exp(-r*T) * max(X-K,0);

Z = C - lambda * (VC - exp(T/(2*S0^2)));

C_5_est = mean(Z);
C_5_est_var = var(Z)/nt;

C_5_IC_inf = C_5_est + sqrt(C_5_est_var)*norminv(alpha/2);
C_5_IC_sup = C_5_est + sqrt(C_5_est_var)*norminv(1-alpha/2);
```

```
L = C_5_IC_sup - C_5_IC_inf;

tps = toc;

% fonction d'affichage

fprintf('\n')
fprintf('Variable de controle 5 - exp(W_T): \n');

disp(strcat(...
{' C = '},sprintf('%05.3f',C_5_est),...
{' IC = ['},sprintf('%05.3f',C_5_IC_inf),...
{' , '},sprintf('%05.3f',C_5_IC_sup),...
{'] '},...
{' largeur = '},sprintf('%05.3f',L),...
{' t = '},sprintf('%05.3f',tps),...
{' eff = '},sprintf('%05.3f',L * sqrt(tps))));

p = corr(X, VC);
% optimum: lambda =~ corr(X,Y)*(Var(X)/Var(Y))^.5
% efficace ?
fprintf(['La correlation entre X et la variable '...
        'de controle est: %0.5g\n'], p)


%% ~~~~~~~~~~~~~ variable de controle 6 ~~~~~~~~~~~~~~~~ %%

% calculer des option asiatiques, mais avec des actions:
% S = S .*(1 +r*dt + sigma*sqrt(S0)*dW_t );
% sigma et r soient connu ici.
% alors qu'on puisse calculer leur payoff avec Balckes-Sch.



%% ~~~~~~~~~~~~~~~~~~~~~ graphes ~~~~~~~~~~~~~~~~~~~~~~~ %%
```

```
% 1:   graphes de S;
% 2-3: ecdf de C_inf et C_N;
% 4-5: boxplot des estimateurs
% 6:   deux graphiques qui demontrent une problematique
% 7:   intervalles de confiance

% simuler n_aff graphes pour affichage
n_aff = 10;
S_aff = S0*ones(n_aff, n+1);
for i = 2:(n+1)
    dWt = normrnd(zeros(n_aff, 1),sqrt(dt));
    S_aff(:,i) = S_aff(:,i-1).*...
                (1 + r*dt+sigma*sqrt(S_aff(:,i-1)).*dWt);
end



G = "g";
%G = "q" %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%P = input(['\n' ...
%    'Pour afficher n''importe quel graphique, tapez ' ...
%    'son numero <1-8> ou [Enter]. \n' ...
%    'Pour quitter tapez plusieures fois [Enter]:\n'] );
P=8;

if isstring(P) || isempty(P)
    P = 1;
else
    if ~ismember(P,1:8)
        P = 1;
    end
end

while G~="q"
```

```
    disp("[Enter] pour continuer")
    switch P
    case 1
        fprintf('< 1: quelques premiers graphes de S >')
        figure(1)
        plot([0 T],[K K], ':k', 'LineWidth',2)
        hold on
        plot(t, obligation(t))
        plot(t, S_aff,'b')
        %plot(t, S_anti_aff, 'r:')
        %plot(t, VC_aff(1:nt_a,:),'g--')
        % probleme si nt < nt_a
        hold off
        % pour comparison, si j'epargne pour le taux r:
        %plot([0 T], [S0 S0*(1+r)^T],"--k"); %obl.
        %1% fplot(obligation, [0 T], "-k");
        xlabel("t")
        legend("K, le prix d''exercice", ...
               "obligation (sans risque)", ...
               "les prix S_t des actions",...
               "Location","northwest");

        if n*nt > 100*1000000; P=7; end
        P=P+1; input('\n\n');

    case 2
        if n*nt > 100*1000000; G="q"; end

        fprintf(['< 2: fonction de distribution ' ...
            'cumulative estime'  ...
            '\n C(T) pour X_{infinie} de C_infinie >'])
        figure(1)
        % E_\pi (e^-rT (X_T - K)^+ / F_0) ~ 1/nt \sum{C(T)}
        %histogram( C_inf );
```

```
        ecdf( X );
        hold on
        plot([K K],[0 1], 'k')
        plot([min(X) max(X)], [.5 .5],':b')
        x_ax = min(X):.1:max(X);
        % probleme si max-min < .1
        nor = normcdf(x_ax,X_mu,sqrt(v));
        plot(x_ax,nor,':r')
        hold off
        legend("ecdf", "K", "P=50%", "cdf normal")
        title("ecdf X(T) pour X_{infinie}");

        P=P+1; input('\n\n');

    case 3
        if n*nt > 100*1000000; G="q"; end

        fprintf(['< 3: fonction de distribution ' ...
            'cumulative estime' ...
            '\n C(T) pour X_{infinie} de C_N >'])
        figure(1)
        ecdf( X_prim );
        hold on
        plot([K K],[0 1], 'k')
        plot([min(X_prim) max(X_prim)], [.5 .5],':b')
        hold off
        legend("ecdf", "K", "P=50%")
        title("ecdf X(T) pour X_{N}");

        P=P+1; input('\n\n');

    case 4
        if n*nt > 100*1000000; G="q"; end
```

```
        fprintf(['< 4: boxplot de l''estimateur ' ...
                 'C_{infinie} >'])
        figure(1)
        boxplot( C_0 );
        xticks({})
        title('boxplot de C_{infinie} a T')
        ylabel('C_T, valeurs actualisees')

        P=P+1; input('\n\n');

    case 5
        if n*nt > 100*1000000; G="q"; end

        fprintf('< 5: boxplot de l''estimateur C_{N} >')
        figure(1)
        boxplot ( C_0_prim );
        xticks({})
        title('boxplot de C_{N} a T')

        P=P+1; input('\n\n');

    case 6
        if n*nt > 100*1000000; G="q"; end

        fprintf('< 6: L''IC de la variable de controle')
        fprintf('\n suivant pour Z a aide de VC')

        plot(sort(Z_vc))
        hold on
        plot(sort(X))
        plot([1 na],[K K], '--k', 'LineWidth',1)
        hold off
        title("X vs variable de controle Z")
        xlabel("nt")
```

```
        legend("Z","X","K")

        input('\n... 6.5 < scatter >');

        scatter(X,X_vc);
        hold on;
        plot([min(X) max(X)],[min(X) max(X)],'-k');
        plot(X_mu,EY_vc,'*r','LineWidth',2);
        legend("X-X_{vc} en pair",...
               "X=X_{vc}",...
               "les moyennes");
        hold off
        xlabel("X")
        ylabel("X_{vc}")

        P=P+1; input('\n');

    case 7
        if n*nt > 100*1000000; G="q"; end

        fprintf('< 7: L''IC de la variable de controle ')
        fprintf('\n suivant pour Z a aide de X_a')

        plot(sort(Z_a))
        hold on
        plot(sort(X))
        plot([1 na],[K K], '--k', 'LineWidth',1)
        hold off
        title("X vs variable de controle Z")
        xlabel("nt")
        legend("Z","X","K")

        input('\n... 7.5 < scatter >');

```

```
        scatter(X,Y);
        hold on;
        plot([min(X) max(X)],[min(X) max(X)],'-k');
        plot(X_mu,EY_a,'*r','LineWidth',2);
        legend("X-Y en pair","X=Y","les moyennes");
        hold off
        xlabel("X")
        ylabel("Y avec laquelle la v.c. est construite")

        P=P+1; input('\n');
    case 8

        w = 7; % nombre des IC affichees
        %fprintf('< 8: ICs (normales) >')
        plot([C_est Ca_est C_1_est C_2_est ...
              C_3_est C_4_est C_5_est], ...
            1:w, 'x')
        %line([K K],[0 5],'Color','green','LineStyle','--')

        line([C_IC_inf C_IC_sup],[1 1])
        line([Ca_IC_inf Ca_IC_sup],[2 2])
        line([C_1_IC_inf C_1_IC_sup],[3 3])
        line([C_2_IC_inf C_2_IC_sup],[4 4])
        line([C_3_IC_inf C_3_IC_sup],[5 5])
        line([C_4_IC_inf C_4_IC_sup],[6 6])
        line([C_5_IC_inf C_5_IC_sup],[7 7])

        legend("estimateurs",...
            'C','C_a','C_{VC1}','C_{VC2}',...
            'C_{VC3}','C_{VC4}','C_{VC5}')
        L1 = C_IC_sup - C_IC_inf;
        L2 = C_est - K;
        limf = [C_IC_inf C_IC_sup] + max(L1,L2)*[-1 1];
        %xlim(limf)
```

```
        ylim([0 w+1])
        yticks(1:w)

        yticklabels({'C','C_a','C_{VC1}','C_{VC2}',...
                     'C_{VC3}','C_{VC4}','C_{VC5}'})
        %title('Intervalles de confiance (sauf C)')

        %P=P+1;
        G="q";
    case 9
        if n*nt > 100*1000000; G="q"; end

        P=input(['\n ' ...
            'Pour afficher n''importe quel graphique, ' ...
            'tapez son numero <1-8> \n']);
        if ismember(P, 1:8)
            fprintf("Vous avez choisi: ")
        else
            G="q";
        end
    otherwise
        G="q";
    end
end

if n*nt > 100*1000000
    warning("Donnees trop grandes pour affichage");
end

%fprintf("\n ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~\n")
%fprintf(  " ~   MERCI POUR VOTRE ATTENTION    ~")
fprintf("\n ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~\n")
```

C. Code VBA

```
Option Explicit

Public n, nt, Nd As Integer
Public S0, k, r, sigma, T As Double

Sub Macro1()


' worksheets
Dim sh_dash, sh_traj, sh_ic As String
Dim sh_dash_o As Worksheet
sh_dash = "Dashboard"
sh_traj = "trajectoires"
sh_ic = "IC"
Set sh_dash_o = Worksheets(sh_dash)


' ~~~~~~~~~~ parametres ~~~~~~~~~~



S0 = Range("S0_").Value
k = Range("K_").Value

r = Range("r_").Value
sigma = Range("sigma_").Value

n = Range("n_").Value
T = Range("T_").Value
Nd = Range("Nd_").Value

nt = Range("nt_").Value
```

```
Dim dt As Double
dt = T / n


Dim tps_debut, tps_passe As Double
Dim n_aff As Integer
n_aff = 10 'nombre des trajectoires affichees dans le
    graphique

' premier cellule de la table de trajectoires ~ t0
Dim Srow, Scol As Integer
Dim Scol_abc As String
Srow = 3
Scol_abc = "I"
Scol = 9

' iteratives
Dim i, j, l As Integer

' array pour enregistrer les simulations
Dim X() As Double 'for some reason double arrays must be
    defined one by one, else they are variant
Dim X_N() As Double
Dim C() As Double

Dim dW As Double
Dim a, b, e, f As Double

'effacer t et S() anciens
With Worksheets(sh_traj)
    Range(.Cells(3, 1), _
          .Cells(Srow + 10000, Scol + 10000)).Delete
End With
```

```
' afficher t
Dim temps() As Double
ReDim temps(n)
temps(0) = 0
For j = 1 To n
    temps(j) = temps(j - 1) + dt
Next

'MsgBox temps(0)
'MsgBox temps(1)
'MsgBox temps(n)
'MsgBox Application.CountA(temps)
'MsgBox UBound(temps)

Sheets(sh_traj).Range("A3:A" & UBound(temps) + 3) = _
    WorksheetFunction.Transpose(temps)


'~~~~~~~~~~~~~~~~~~ series illustratives pour l'affichage
   ~~~~~~~~~~~~
'~~~~~~~~~~~~~~~~~~
' pour le graphique
'~~~~~~~~~~~~~~~~~~

For j = 1 To n_aff
    i = 1
    a = S0


    'If j <= n_aff Then
        Sheets(sh_traj).Cells(Srow - 1 + i, Scol - 1 + j).
           Value = a
```

```
    'End If

    For i = 2 To n + 1
        'simuler une v.a. normale
        dW = Sqr(-2 * Log(Rnd())) * _
             Cos(6.283185307 * Rnd()) * Sqr(dt)
        a = a * (1 + (r * dt + sigma * Sqr(a) * dW))

        'If j <= n_aff Then
            Sheets(sh_traj).Cells(Srow - 1 + i, Scol - 1 +
                j).Value = a
        'End If

    Next
Next

'~~~~~~~~~~~~~~~~~~~~~~~~~
' pour l'exemple de series
'~~~~~~~~~~~~~~~~~~~~~~~~~

a = S0
b = S0
e = S0
f = S0


i = 1
Sheets(sh_traj).Cells(Srow - 1 + i, 2).Value = a
Sheets(sh_traj).Cells(Srow - 1 + i, 3).Value = b
Sheets(sh_traj).Cells(Srow - 1 + i, 4).Value = e
Sheets(sh_traj).Cells(Srow - 1 + i, 5).Value = f

For i = 2 To n + 1
    'simuler une v.a. normale
```

```
        dW = Sqr(-2 * Log(Rnd())) * _
             Cos(6.283185307 * Rnd()) * Sqr(dt)

        'le prix de l'action
        a = a * (1 + (r * dt + sigma * Sqr(a) * dW))

        'la variable antithetique
        b = b * (1 + (r * dt - sigma * Sqr(b) * dW))

        'VC_1
        e = e + dW

        'VC_2
        f = f + dW ' * Exp(r * (i - 1) * dt)

        Sheets(sh_traj).Cells(Srow - 1 + i, 2).Value = a
        Sheets(sh_traj).Cells(Srow - 1 + i, 3).Value = b
        Sheets(sh_traj).Cells(Srow - 1 + i, 4).Value = e
        Sheets(sh_traj).Cells(Srow - 1 + i, 5).Value = f

Next



'~~~~~~~~~~~~~~~~ X_inf ~~~~~~~~~~~~~~~~

tps_debut = Timer

ReDim X(1 To nt)

'simuler S pas a pas, afficher n_aff S dans sh_traj

For j = 1 To nt
    i = 1
    a = S0
```

```
    X(j) = 0.5 * a / n


    For i = 2 To n + 1
        'simuler une v.a. normale
        dW = Sqr(-2 * Log(Rnd())) * _
             Cos(6.283185307 * Rnd()) * Sqr(dt)
        a = a * (1 + (r * dt + sigma * Sqr(a) * dW))
        X(j) = X(j) + a / n

    Next

    X(j) = X(j) - 0.5 * a / n

Next

'Calculer C a temps 0 avec X et K
C = payoff(X)

' ~~~~~~~~~~~~~~~~~
' estimateurs X_inf
'~~~~~~~~~~~~~~~~~~

Dim C_mu, C_std, C_mu_b, C_mu_h As Double

C_mu = Mean(C)
C_std = StdDev(C)

C_mu_b = C_mu - 1.96 * C_std / Sqr(nt)
C_mu_h = C_mu + 1.96 * C_std / Sqr(nt)

tps_passe = Round(Timer - tps_debut, 3)

'affichage
```

```
Sheets(sh_ic).Range("A3").Value = "C_inf"
Sheets(sh_ic).Range("B3").Value = C_mu_b
Sheets(sh_ic).Range("C3").Value = C_mu
Sheets(sh_ic).Range("D3").Value = C_mu_h
Sheets(sh_ic).Range("E3").Value = tps_passe


'~~~~~~~~~~~~~~~~ X_N ~~~~~~~~~~~~~~~~

tps_debut = Timer

' pour X_N
ReDim X_N(1 To nt)

'simuler S pas a pas, afficher n_aff S dans sh_traj

For j = 1 To nt
    l = 1
    i = 1
    a = S0
    X_N(j) = 0.5 * a / Nd 'c'est faux, non?

    For i = 1 To n
        'simuler une v.a. normale
        dW = Sqr(-2 * Log(Rnd())) * _
             Cos(6.283185307 * Rnd()) * Sqr(dt)
        a = a * (1 + (r * dt + sigma * Sqr(a) * dW))

        If (i / n) > (l / Nd) Then
            X_N(j) = X_N(j) + a / Nd
            l = l + 1
        End If

    Next
```

```
    X_N(j) = X_N(j) + 0.5 * a / Nd 'c'est faux, non?
    ' pas pour S0, seulement a la fin:  X_N(j) = X_N(j) +
       a/Nd

Next

'Sheets(sh_ic).Range("G1:G" & UBound(X_N)) = _
'     WorksheetFunction.Transpose(X_N)
'MsgBox X_N(Nd - 1)
'Calculer C_N a temps 0 avec X et K
C = payoff(X_N)

' ~~~~~~~~~~~~~~~~~
' estimateurs X_N
'~~~~~~~~~~~~~~~~~~

C_mu = Mean(C)
C_std = StdDev(C)

C_mu_b = C_mu - 1.96 * C_std / Sqr(nt)
C_mu_h = C_mu + 1.96 * C_std / Sqr(nt)

tps_passe = Round(Timer - tps_debut , 3)
'affichage
Sheets(sh_ic).Range("A8").Value = "C_N"
Sheets(sh_ic).Range("B8").Value = C_mu_b
Sheets(sh_ic).Range("C8").Value = C_mu
Sheets(sh_ic).Range("D8").Value = C_mu_h
Sheets(sh_ic).Range("E8").Value = tps_passe



'~~~~~~~~~~~~~~~~~ X_anti ~~~~~~~~~~~~~~~~~

```

```
tps_debut = Timer
Dim X_anti() As Double
ReDim X_anti(1 To nt)

'simuler S pas a pas, afficher n_aff S dans sh_traj

For j = 1 To nt
    a = S0
    b = S0
    X(j) = 0.5 * a / n
    X_anti(j) = 0.5 * b / n
    i = 1

    For i = 2 To n + 1
        'simuler une v.a. normale
        dW = Sqr(-2 * Log(Rnd())) * _
             Cos(6.283185307 * Rnd()) * Sqr(dt)

        a = a * (1 + (r * dt + sigma * Sqr(a) * dW))
        X(j) = X(j) + a / n

        b = b * (1 + (r * dt - sigma * Sqr(b) * dW))
        X_anti(j) = X_anti(j) + b / n

    Next

    X(j) = X(j) - 0.5 * a / n
    X_anti(j) = X_anti(j) - 0.5 * b / n

Next


' ~~~~~~~~~~~
' estimateurs
```

```
'~~~~~~~~~~~~~

Dim Z_mu, rho, Z_std, Z_mu_b, Z_mu_h As Double

Dim C_anti() As Double
ReDim C(1 To nt)

C_anti = payoff(X_anti)
C = payoff(X)

Z_mu = (Mean(C_anti) + Mean(C)) / 2

Dim SumSq As Double
SumSq = 0
For i = 1 To nt
        SumSq = SumSq + (C_anti(i) - Z_mu) ^ 2 + (C(i) -
           Z_mu) ^ 2
Next i
Z_std = SumSq / (2 * nt - 1)

Z_mu_b = Z_mu - 1.96 * Z_std / Sqr(2 * nt)
Z_mu_h = Z_mu + 1.96 * Z_std / Sqr(2 * nt)

tps_passe = Round(Timer - tps_debut, 3)

'affichage
Sheets(sh_ic).Range("A5").Value = "C_anti"
Sheets(sh_ic).Range("B5").Value = Z_mu_b
Sheets(sh_ic).Range("C5").Value = Z_mu
Sheets(sh_ic).Range("D5").Value = Z_mu_h
Sheets(sh_ic).Range("E5").Value = tps_passe

'~~~~~~~~~~~~~~~~ X_5nt ~~~~~~~~~~~~~~~~
```

```
nt = 5 * nt
tps_debut = Timer
ReDim X(1 To nt)
'ReDim C(1 To nt)

'simuler S pas a pas, afficher n_aff S dans sh_traj

For j = 1 To nt
    a = S0
    X(j) = 0.5 * a / n
    i = 1

    For i = 2 To n + 1
        'simuler une v.a. normale
        dW = Sqr(-2 * Log(Rnd())) * _
             Cos(6.283185307 * Rnd()) * Sqr(dt)
        a = a * (1 + (r * dt + sigma * Sqr(a) * dW))
        X(j) = X(j) + a / n

    Next

    X(j) = X(j) - 0.5 * a / n

Next

'Calculer C a temps 0 avec X et K
C = payoff(X)

' ~~~~~~~~~~~
' estimateurs
'~~~~~~~~~~~~

C_mu = Mean(C)
C_std = StdDev(C)
```

```
C_mu_b = C_mu - 1.96 * C_std / Sqr(nt)
C_mu_h = C_mu + 1.96 * C_std / Sqr(nt)

tps_passe = Round(Timer - tps_debut, 3)

'affichage
Sheets(sh_ic).Range("A4").Value = "C_5nt"
Sheets(sh_ic).Range("B4").Value = C_mu_b
Sheets(sh_ic).Range("C4").Value = C_mu
Sheets(sh_ic).Range("D4").Value = C_mu_h
Sheets(sh_ic).Range("E4").Value = tps_passe

nt = nt / 5
ReDim X(1 To nt)
ReDim C(1 To nt)

'~~~~~~~~~~~~~~~~ X_VC_1 ~~~~~~~~~~~~~~~~

Dim Y() As Double
ReDim Y(1 To nt)
Dim Z() As Double
ReDim Z(1 To nt)

tps_debut = Timer

For j = 1 To nt
    a = S0
    e = S0
    X(j) = 0.5 * a / n
    Y(j) = 0.5 * e / n
    i = 1

    For i = 2 To n + 1
```

```
        'simuler une v.a. normale
        dW = Sqr(-2 * Log(Rnd())) * _
             Cos(6.283185307 * Rnd()) * Sqr(dt)

        'comme X_inf
        a = a * (1 + (r * dt + sigma * Sqr(a) * dW))
        X(j) = X(j) + a / n

        'VC
        e = e + dW
        Y(j) = Y(j) + e / n


    Next

    X(j) = X(j) - 0.5 * a / n
    Y(j) = Y(j) - 0.5 * e / n
Next

'Problematique: Les X et Y ne sont pas independant en utilisant le meme processus?

' ~~~~~~~~~~~
' estimateurs
'~~~~~~~~~~~~~
'Non overlapping increments Wt-Ws and Wv-Wu for any 0=s<t=>u<v are independent of each other.

Dim lambda As Double

'E(Y) = S0
' lambda = Cov/sigma_Y^2
' Cov estime
' sigma_Y ?
```

```
' Var(Y) ~ Var(somme de mouv. brown) = E(somme(dw)^2) - E^2(somme(dw)) ? E[ int(dW^2) ]
'=T/6(2n^2+3n+1)

'je suppose que ca dure tres longtemps, calculer lambda
' idee: calculer lambda, puis le fixer

tps_passe = Timer - tps_debut
Dim sigma2_Y As Double
sigma2_Y = (T / 6 * (2 * (n * n) + 3 * n + 1))
lambda = Cov(X, Y) / Sqr(sigma2_Y)
MsgBox "le lambda de VC_1 = " & lambda
MsgBox "le rho_XY =" & Cov(X, Y) / (StdDev(X) * Sqr(sigma2_Y))
tps_debut = Timer

For i = LBound(Z) To UBound(Z)
    Z(i) = X(i) - lambda * (Y(i) - S0 * T)
Next

C = payoff(Z)

C_mu = Mean(C)
C_std = StdDev(C)

C_mu_b = C_mu - 1.96 * C_std / Sqr(nt)
C_mu_h = C_mu + 1.96 * C_std / Sqr(nt)

tps_passe = Round(tps_passe + tps_debut - Timer, 3)

'affichage
Sheets(sh_ic).Range("A6").Value = "VC_1"
Sheets(sh_ic).Range("B6").Value = C_mu_b
Sheets(sh_ic).Range("C6").Value = C_mu
```

```
Sheets(sh_ic).Range("D6").Value = C_mu_h
Sheets(sh_ic).Range("E6").Value = tps_passe



'~~~~~~~~~~~~~~~~ X_VC_2 ~~~~~~~~~~~~~~~~

'calculer lambda avec n=200
For j = 1 To 200
    a = S0
    f = S0
    X(j) = 0.5 * a / n

    i = 1
    For i = 1 To n
        'simuler une v.a. normale
        dW = Sqr(-2 * Log(Rnd())) * _
             Cos(6.283185307 * Rnd()) * Sqr(dt)

        'comme X_inf
        a = a * (1 + (r * dt + sigma * Sqr(a) * dW))
        X(j) = X(j) + a / n

        'VC
        f = f + dW

    Next

    X(j) = X(j) - 0.5 * a / n
    Y(j) = f
Next

Dim EY As Double
EY = S0
```

```
sigma2_Y = T
lambda = Cov(X, Y) / sigma2_Y

MsgBox "le lambda de VC_2 = " & lambda
MsgBox "le rho_XY =" & Cov(X, Y) / (StdDev(X) * Sqr(
   sigma2_Y))

'Puis simuler les trajectoires et la variable de controle
tps_debut = Timer

For j = 1 To nt
    a = S0
    f = S0
    X(j) = 0.5 * a / n
    'Y(j) = 0.5 * e / n
    i = 1

    For i = 1 To n
        'simuler une v.a. normale
        dW = Sqr(-2 * Log(Rnd())) * _
             Cos(6.283185307 * Rnd()) * Sqr(dt)

        'comme X_inf
        a = a * (1 + (r * dt + sigma * Sqr(a) * dW))
        X(j) = X(j) + a / n

        'VC
        f = f + dW
        'Y(j) = Y(j) + f / n



    Next

    X(j) = X(j) - 0.5 * a / n
```

```
    'Y(j) = Y(j) - 0.5 * f / n
    Y(j) = f
Next


' ~~~~~~~~~~~
' estimateurs
'~~~~~~~~~~~~


For i = LBound(Z) To UBound(Z)
    Z(i) = X(i) - lambda * (Y(i) - EY)
Next

C = payoff(Z)

C_mu = Mean(C)
C_std = StdDev(C)

C_mu_b = C_mu - 1.96 * C_std / Sqr(nt)
C_mu_h = C_mu + 1.96 * C_std / Sqr(nt)

tps_passe = Round(Timer - tps_debut, 3)

'affichage
Sheets(sh_ic).Range("A7").Value = "VC_2"
Sheets(sh_ic).Range("B7").Value = C_mu_b
Sheets(sh_ic).Range("C7").Value = C_mu
Sheets(sh_ic).Range("D7").Value = C_mu_h
Sheets(sh_ic).Range("E7").Value = tps_passe



'~~~~~~~~~~~~~~~~ affichage ~~~~~~~~~~~~~~~~~

```

```
' ~~~~~~~~~~~~~~~~~~~
' faire defiler le tableau
' ~~~~~~~~~~~~~~~~~~~

Sheets(sh_dash).Shapes.Range(Array("Scroll Bar 2")).Select
' problematique: le baton pour defiler reste selecter
' d'ailleurs, si quelque chose d'autre est selecte, il y a
    de problemes
' pour le moment: faut demarrer la programme seulement par
    le bouton !

With Selection
'Range(Array("Scroll Bar 2"))
.Value = 0
'Range(Array("Scroll Bar 2"))
.Min = 0
'Range(Array("Scroll Bar 2"))
.Max = n - 10
'Range(Array("Scroll Bar 2"))
.SmallChange = 1
'Range(Array("Scroll Bar 2"))
.LargeChange = 20
'Range(Array("Scroll Bar 2"))
.LinkedCell = "trajectoires!F2"
'Range (Array("Scroll Bar 2"))
.Display3DShading = True
End With



Range("M4").FormulaR1C1 = _
    "=OFFSET(trajectoires!R[-1]C[-12], trajectoires!R2C6
        ,0)"
```

```
Range("M4").AutoFill Destination:=Range("M4:M14"), Type:=xlFillDefault
Range("M4:M14").AutoFill Destination:=Range("M4:Q14"), Type:=xlFillDefault

' ~~~~~~~~~~~~~~~~
' affichage graphe
' ~~~~~~~~~~~~~~~~

Dim chartrange As Range
Set chartrange = Sheets(sh_traj).Cells(Srow, Scol)
Set chartrange = chartrange.Resize(n + 1, n_aff)

Worksheets(sh_dash).Activate
Dim Graphe As Object

'effacer graphes aines
For Each Graphe In ActiveSheet.ChartObjects
  Graphe.Delete
Next Graphe

Set Graphe = sh_dash_o.ChartObjects.Add( _
    Left:=Range("G3").Left, Width:=360 - 70, _
    Top:=Range("G3").Top, Height:=185)
With Graphe.Chart
    .SetSourceData chartrange
    .PlotBy = xlColumns 'echanger x et y axes
    .ChartType = xlLine
    '.HasTitle = True
    '.ChartTitle.Text = "Prix des actions"
    .FullSeriesCollection(1).XValues = _
        Range(Scol_abc & Srow & ":" _
              & Scol_abc & UBound(temps) + 1)
    .Legend.Delete
```

```
    '.Axes(xlCategory).HasTitle = True
    '.Axes(xlCategory).AxisTitle.Text = "t"
    .Axes(xlValue).MinimumScale = Application.
       WorksheetFunction.RoundDown(WorksheetFunction.Min(
       chartrange), 0)
    .ChartColor = 11
End With

'If False Then
' ~~~~~~~~~~~~~~~~
' affichage IC
' ~~~~~~~~~~~~~~~~
Dim Graphe_IC As Object
Set Graphe_IC = sh_dash_o.ChartObjects.Add( _
    Left:=Range("I17").Left, Width:=300 - 60, _
    Top:=Range("I17").Top, Height:=115)

With Graphe_IC.Chart
'Shapes.AddChart2(201, xlColumnClustered).Select
    .SetSourceData Source:=Range("IC!$A$2:$D$8")
    .FullSeriesCollection(1).ChartType = xlColumnClustered
    .FullSeriesCollection(1).AxisGroup = 1
    .FullSeriesCollection(2).ChartType = xlColumnClustered
    .FullSeriesCollection(2).AxisGroup = 1
    .FullSeriesCollection(3).ChartType = xlLine
    .FullSeriesCollection(3).AxisGroup = 1
    .FullSeriesCollection(1).ChartType = xlXYScatter
    .FullSeriesCollection(3).ChartType = xlXYScatter
    .FullSeriesCollection(3).AxisGroup = 1
    .FullSeriesCollection(1).AxisGroup = 1
    .Legend.Delete
    .ChartColor = 11
End With
```

```
Dim chart_shp As Shape
For Each chart_shp In ActiveSheet.Shapes
    chart_shp.Line.Visible = msoFalse
Next chart_shp

Range("D17").Select
MsgBox "Simulation finie pour " & nt & " trajectoires.",
   vbInformation
'MsgBox chartrange.Address

End Sub


Function Mean(Arr() As Double)
    Dim Sum As Double
    Dim i, k As Integer
    k = Application.CountA(Arr)
    Sum = 0
    For i = 1 To k
        Sum = Sum + Arr(i)
    Next i

    Mean = Sum / k

End Function

Function StdDev(Arr() As Double)
    Dim i, k As Integer
    Dim avg As Double, SumSq As Double
    k = Application.CountA(Arr)
    avg = Mean(Arr)
    For i = 1 To k
        SumSq = SumSq + (Arr(i) - avg) ^ 2
    Next i
```

```
    StdDev = Sqr(SumSq / (k - 1))

End Function

Function Cov(Arr1() As Double, Arr2() As Double)
    ' faudra verifier que les deux Arr ont les meme dimensions
    Dim i As Integer
    Dim avg1, avg2, SumSq As Double
    'k = Application.CountA(Arr1)
    avg1 = Mean(Arr1)
    avg2 = Mean(Arr2)
    For i = 1 To nt
        SumSq = SumSq + (Arr1(i) - avg1) * (Arr2(i) - avg2)
    Next i
    Cov = SumSq / (nt - 1)

End Function

Function payoff(a() As Double) ', Optional k As Double = k, Optional r As Double = r, Optional T As Double = T)
    Dim l As Integer
    l = Application.CountA(a)
    Dim i As Integer
    Dim C() As Double
    ReDim C(1 To l)
    For i = 1 To l
        If a(i) > k Then
            C(i) = Exp(-r * T) * (a(i) - k)
        Else
            C(i) = 0
        End If
    payoff = C
```

```
Next

End Function
```