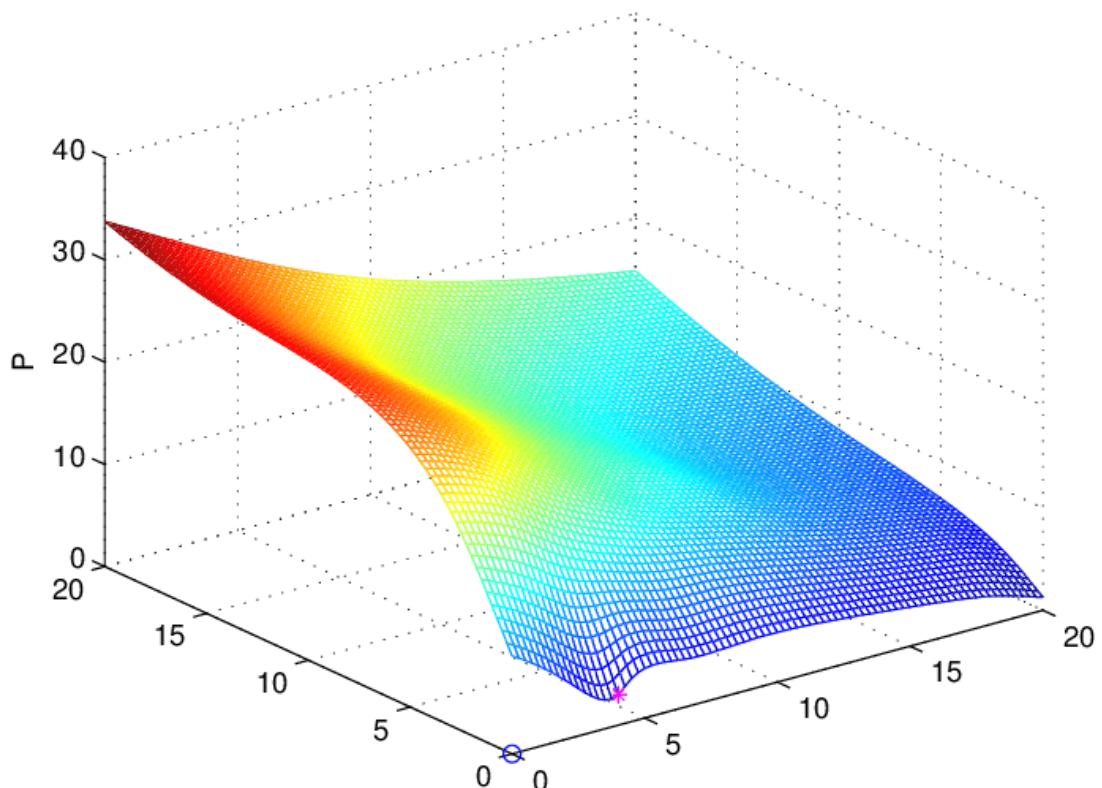


# Méthodes Numériques Avancées

## Pour la Finance : TP1 et TP2



Adnane EL KASMI (ING2 MF01)  
à l'attention de : Irina Kortchemski

# Table des matières

I	Introduction . . . . .	2
1	Présentation . . . . .	2
2	Modèle de Black et Scholes . . . . .	2
3	Simulation d'évolution du prix de l'actif sous-jacent par la méthode aux Différences Finies . . . . .	3
II	TP1 : Résolution numérique de l'équation de Black et scholes par la méthode explicite d'Euler . . . . .	4
1	Partie I : Vanilla. Call. Conditions aux limites de Dirichlet . . . . .	4
2	Partie II : Vanilla. Call. Conditions aux limites de Neumann . . . . .	14
3	Partie III : Vanilla. Put. Conditions aux limites de Neumann . . . . .	17
4	Partie IV : Put Américain . . . . .	20
5	Partie V : Option de Butterfly . . . . .	22
6	Partie VI : Comparaison entre les solutions numérique et analytique . . .	25
7	Partie VII : Volatilité locale . . . . .	33
III	TP2 : Résolution numérique de l'équation de Black et Scholes par les methodes de Monte-Carlo. Réduction de la variance. . . . .	36
1	Partie I : Théorie . . . . .	36
2	Partie II : Implémentation . . . . .	37
3	Partie III : Le prix de l'option Européene pour une valeur $S_t$ et un temps $t$ quelconques . . . . .	42
4	Partie IV : Variance. Intervalle de confiance . . . . .	46
5	Partie V : Réduction de la variance. Variables Antithétiques. . . . .	49
6	Partie VI : Réduction de la variance. Variables de Contrôle. . . . .	62
7	Partie VII. Le lien entre les méthodes déterministes (Différences Finies ) et stochastiques (Monte-Carlo) . . . . .	67
IV	Conclusion . . . . .	70

## I Introduction

### 1 Présentation

Les méthodes numériques fondées sur les équations aux dérivées partielles n'étaient pas très populaires Jusqu'à récemment. Les modèles obtenus par des arguments probabilistes et simulés par les méthodes de Monte-Carlo sont en effet plus naturelles et il est plus facile à implémenter des méthodes stochastiques que les algorithmes utilisés pour les EDP. Cependant, quand il est possible discréteriser les EDP, les algorithmes pour la résolution des équations discréétisées sont très efficaces. Les solutions, numériques des EDP donnent plus d'informations. Elles donnent, par exemple, les prix d'une option pour toutes les valeurs initiales de l'actif sous-jacent et pour toutes les valeurs du temps d'exercice, tandis que les méthodes de Monte-Carlo les donnent pour une seule valeur de l'actif et du temps d'exercice. Les EDP sont aussi très efficaces pour le calcul des Greeks.

Les PDF en finance possèdent plusieurs caractéristiques. On travaille sur des domaines temporels et spatiaux finis. On doit impérativement imposer des conditions aux limites et des conditions initiales. Les équations sont souvent paraboliques, mais elles peuvent aussi contenir les termes hyperboliques, comme par exemple l'EDP pour les options asiatiques.

Des modèles de stratégie de hedging du marché non liquide ont permis d'arriver à une équation de Black et Scholes non linéaire. La resolution numérique des ces équations sont indispensables.

### 2 Modèle de Black et Scholes

L'actif sous-jacent  $S$  est un processus stochastique  $S(t)$ ,  $t \in [t, T]$  qui vérifie l'équation différentielle stochastique suivante :

$$dS_t = rS_t dt + \sigma S_t dW_t \text{ avec } S_{t=0} = S_0$$

On note  $\sigma$  la volatilité de l'action,  $r$  le taux d'intérêt. La solution de cette équation est donnée par la formule :

$$S(t) = S_0 \exp \left[ \left( r - \frac{\sigma^2}{2} \right) t + \sigma W(t) \right]$$

Le prix d'une Option Européenne au moment de temps  $t$  est donnée par l'espérance conditionnelle d'une fonction Pay-Off :

$$V(S_0, t) = e^{-r(T-t)} \mathbb{E}[\max(S(T) - K, 0) | S_t = S_0]$$

On peut expliquer ce résultat de façon suivante : on simule un grand nombre de chemins d'évolution de l'actif  $S(t)$  sur l'intervalle de temps  $[t, T]$ , en partant toujours de  $S_0$ . Pour chaque chemin on cherche la valeur finale  $S(T)$ . Puis on calcule la moyenne arithmétique de gains, c'est-à-dire de  $\max(S(T)K, 0)$ . Le facteur  $e^{-r(T-t)}$  exprime le fait qu'une banque rembourse les intérêts sur l'intervalle du temps  $[t, T]$ .

## 3 Simulation d'évolution du prix de l'actif sous-jacent par la méthode aux Différences Finies

L'évolution de l'actif sous-jacent  $S$  est un processus stochastique  $S(t)$ ,  $t \in [0, T]$  qui vérifie l'équation différentielle stochastique suivante:

$$dS_t = rS_t dt + \sigma S_t dW_t, \quad S(t=0) = S_0$$

On note  $\sigma$  la volatilité de l'actif,  $r$  le taux d'intérêt.

**La méthode 1** consiste à discrétiser l'équation stochastique par les Différences Finies, puis la simuler.

- Discrétisons l'intervalle  $[0, T]$  sur  $N$  parties: :  $(\Delta t = T/N, \quad t_i = i \Delta t)$ .
- Discrétisons le différentiel

$$dS(t_i) \sim S(t_i + \Delta t) - S(t_i) \equiv S_{t_{i+1}} - S_i$$

- Discrétisons le différentiel

$$dW(t_i) \sim W(t_i + \Delta t) - W(t_i) \equiv W_{t_{i+1}} - W_{t_i}$$

- Discrétisons l'équation différentielle

$$S_{t_{i+1}} - S_{t_i} = rS_{t_i} \Delta t + S_{t_i} \sigma (W_{t_{i+1}} - W_{t_i})$$

soit

$$S(t_{i+1}) = S_{t_i} ((1 + r\Delta t) + \sigma (W_{t_{i+1}} - W_{t_i}))$$

- Simulons pour un chemin du mouvement Brownien:

$$\left\{ \begin{array}{l} W_0 = 0 \\ W_{t_1} = g_1 \sqrt{\Delta t} \\ W_{t_2} = W_{t_1} + g_2 \sqrt{\Delta t} \\ \vdots \\ W_{t_N} = W_{t_{N-1}} + g_N \sqrt{\Delta t}, \end{array} \right.$$

où les nombres  $\{g_i\}$  suivent la loi Normale  $\mathbb{N}(0, 1)$ .

- Pour ce chemin du mouvement Brownien simulons le chemin correspondant à l'évolution de l'actif

$$S(t_{i+1}) = S(t_i) ((1 + r\Delta t) + \sigma (W_{t_{i+1}} - W_{t_i}))$$

$$\left\{ \begin{array}{l} S_0 = S_{t_0} \\ S_{t_1} = S_{t_0} ((1 + r\Delta t) + \sigma (W_{t_1} - W_{t_0})) \equiv S_{t_0} ((1 + r\Delta t) + \sigma \sqrt{\Delta t} g_1) \\ S_{t_2} = S_{t_1} ((1 + r\Delta t) + \sigma (W_{t_2} - W_{t_1})) \equiv S_{t_1} ((1 + r\Delta t) + \sigma \sqrt{\Delta t} g_2) \\ \vdots \\ S_{t_N} = S(t_{N-1}) ((1 + r\Delta t) + \sigma (W_{t_N} - W_{t_{N-1}})) \equiv S_{t_{N-1}} ((1 + r\Delta t) + \sigma \sqrt{\Delta t} g_N) \end{array} \right.$$

## II TP1 : Résolution numérique de l'équation de Black et Scholes par la méthode explicite d'Euler

Le but de ce TP 1 est l'évaluation numérique du prix des options Européenne et Américaine par la méthode de Différences Finies.

### 1 Partie I : Vanilla. Call. Conditions aux limites de Dirichlet

L'équation de Black et Scholes avec la condition finale et les conditions aux limites de Dirichlet s'écrit de la forme :

$$\begin{cases} \frac{\partial V}{\partial t} - rV + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} = 0 \\ V(t = T, S) = \max(S - K, 0) \\ V(t, S = 0) = 0 \\ V(t, S = L) = L - Ke^{-r(T-t)} \end{cases}$$

Cette équation permet de trouver le prix  $V(S, t)$  de l'option d'achat (ou call) d'une action qui vaut initialement  $S$  et qu'on pourra acheter au prix  $K$  dans un temps ultérieur  $T$ .  $V(0, S)$  est le prix au temps  $t = 0$  de l'option d'achat de prix d'exercice  $K$  à l'échéance  $T > 0$ , et d'actif  $S$  en  $t = 0$ . On note  $\sigma$  la volatilité de l'action et  $r$  le taux d'intérêt.

→ Tracer la condition finale

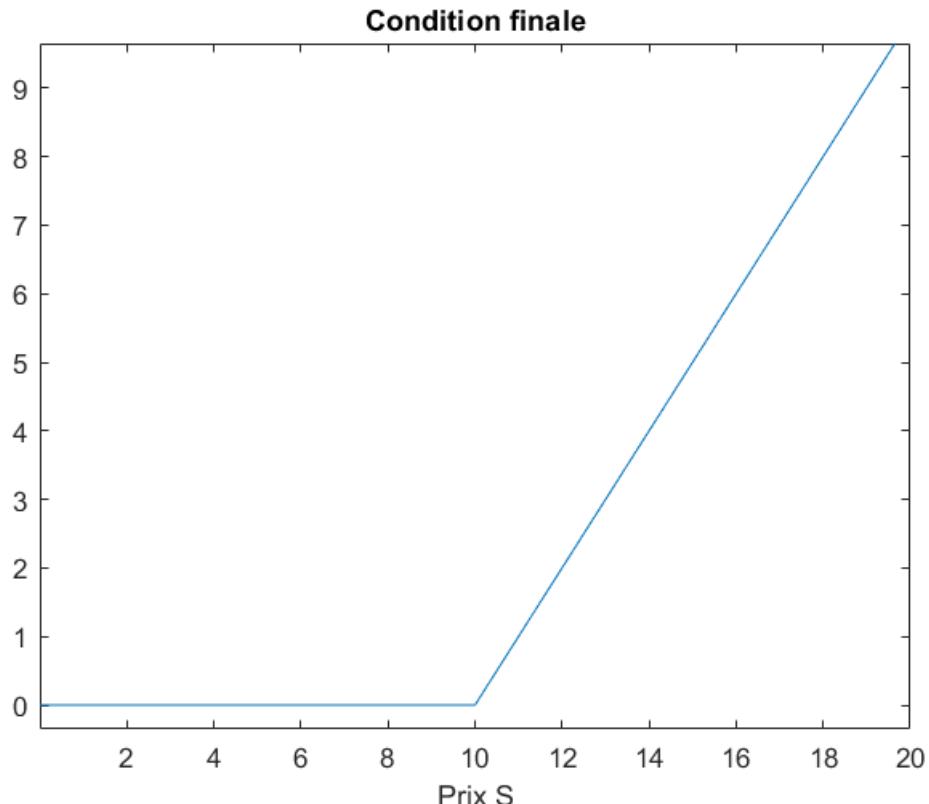


FIGURE 1 – Graphe de la condition finale

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab pour tracer la condition finale :

```
3  [-] function [] = tracer_condition_finale()
4
5      % Définition des constantes %
6
7      L=20;
8      K=10;
9      T=0.5;
10     r=0.1;
11     sigma=0.5;
12     N=99;
13     M=4999;
14
15     % Définition des vecteurs %
16
17     S=linspace(0,L,N+2);
18     t=linspace(0,T,M+2);
19     dt=T/(M+2);
20     ds=L/(N+2);
21     V=zeros(M+2,N+2);
22
23     % Implementation de la condition finale %
24
25     [-] for i=1:N+2
26         V(M+2,i)=condition_finale(S(i),K);
27     end
28
29     figure;
30     plot(S,V(M+2,:))
31     xlabel('Prix S')
32     title('Condition finale')
33
34     end
35
36     % la condition finale %
37
38     [-] function [f] = condition_finale(S,K)
39         f = max(S-K,0);
40     end
```

FIGURE 2 – Code Sous MATLAB du fichier Code1.m

# TP: Méthodes numériques avancées appliquées à la finance

## → Tracer la condition aux limites de Dirichlet

En mathématiques, une condition aux limites de Dirichlet (nommée d'après Johann Dirichlet) est imposée à une équation différentielle ou à une équation aux dérivées partielles lorsque l'on spécifie les valeurs que la solution doit vérifier sur les frontières/limites du domaine.

Il existe d'autres conditions possibles. Par exemple la condition aux limites de Neumann, ou la condition aux limites de Robin, qui est une combinaison des conditions de Dirichlet et Neumann.

Les conditions aux limites de Dirichlet s'écrivent sous forme :

$$\begin{cases} V_0^n = 0 \\ V_{N+1}^n = L - K e^{-r(T-t(n))} \\ n = M, \dots, 1, 0 \end{cases}$$

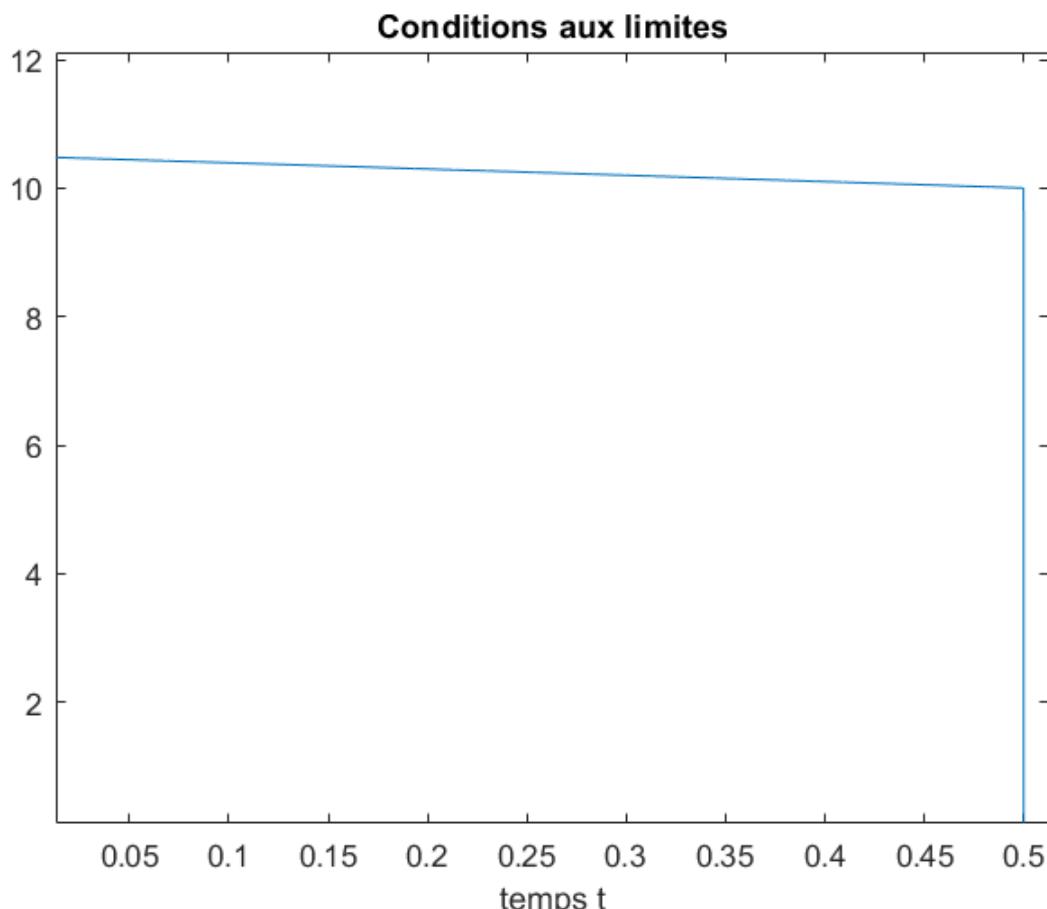


FIGURE 3 – Graphe des conditions aux limites de Dirichlet

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab pour tracer les conditions aux limites de Dirichlet :

```
3  [-] function [] = tracer_condition_limites()
4
5      % Définition des constantes %
6
7      L=20;
8      K=10;
9      T=0.5;
10     r=0.1;
11     sigma=0.5;
12     N=99;
13     M=4999;
14
15     % Définition des vecteurs %
16
17     S=linspace(0,L,N+2);
18     t=linspace(0,T,M+2);
19     dt=T/(M+2);
20     ds=L/(N+2);
21     V=zeros(M+2,N+2);
22
23     % Implementation des conditions aux limites %
24
25     [-] for n=1:M+1
26         V(n,1)=0;
27         V(n,N+2)=L-K*exp(-r*(T-t(n)));
28     end
29
30     figure;
31     plot(t,V(:,1));
32     plot(t,V(:,N+2));
33     xlabel('temps t')
34     title('Conditions aux limites')
35
36     end
```

FIGURE 4 – Code Sous MATLAB du fichier Code2.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Discrétisation de V (le programme principale) :

Pour discréteriser l'équation on utilise:

pour  $\frac{\partial V}{\partial t}$  la dérivée retrograde,

pour  $\frac{\partial^2 V}{\partial S^2}$  la seconde dérivée, centrée,

pour  $\frac{\partial V}{\partial S}$  la dérivée centrée.

On obtient

$$V_i^{n-1} = V_{i+1}^n \frac{\Delta t}{2} \left( \sigma^2 \frac{S(i)^2}{(\Delta S)^2} + r \frac{S(i)}{(\Delta S)} \right) + V_i^n \left( 1 - \Delta t \left( \sigma^2 \frac{S(i)^2}{(\Delta S)^2} + r \right) \right) + V_{i-1}^n \frac{\Delta t}{2} \left( \sigma^2 \frac{S(i)^2}{(\Delta S)^2} - r \frac{S(i)}{(\Delta S)} \right)$$

```
for n=M+2:-1:2
for i=2:N+1

    % Discrétisation de l'équation de Black et Scholes %

    V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
end
end
```

FIGURE 5 – Code sous Matlab de la discrétisation de V

→ Visualiser la fonction  $V(S, t)$  aux instants  $t = T$  ,  $t = \frac{T}{2}$  et  $t = 0$

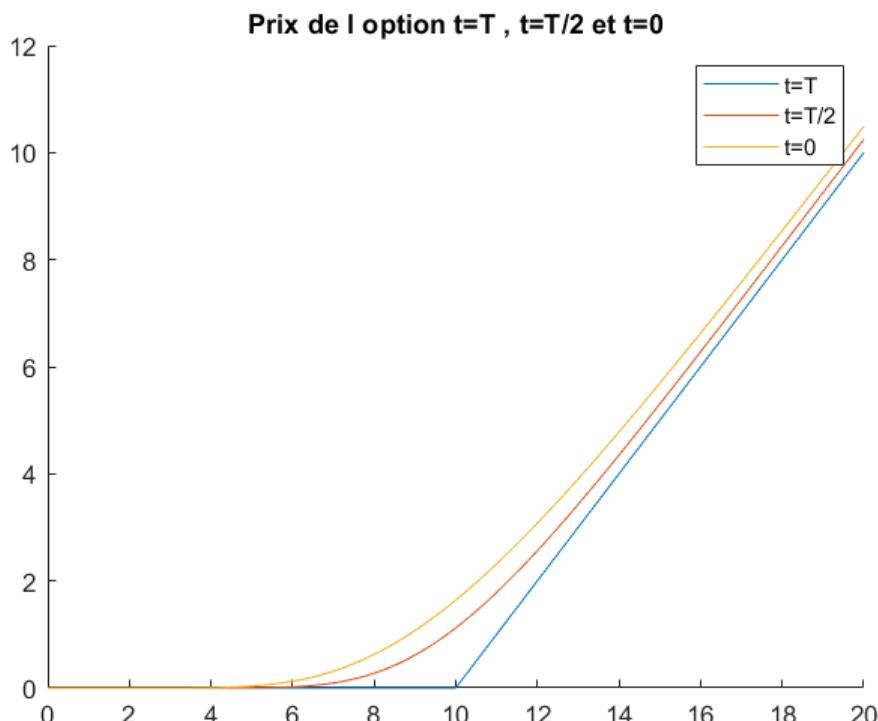


FIGURE 6 – Graphe des conditions aux limites à t différents

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab pour visualiser la fonction  $V(S, t)$  aux instants  $t = T$ ,  $t = \frac{T}{2}$  et  $t = 0$ :

```
3 % function [] = visualiser_fonction_V()
4
5 % Définition des constantes %
6
7 L=20;
8 K=10;
9 T=0.5;
10 r=0.1;
11 sigma=0.5;
12 N=99;
13 M=4999;
14
15 % Définition des vecteurs %
16
17 S=linspace(0,L,N+2);
18 t=linspace(0,T,M+2);
19 dt=T/(M+2);
20 ds=L/(N+2);
21 V=zeros(M+2,N+2);
22
23 % Implementation de la condition finale %
24
25 for j=1:N+2
26     V(M+2,j)=condition_finale(S(j),K);
27 end
28
29 % Implementation des conditions aux limites %
30
31 for k=1:M+1
32     V(k,1)=0;
33     V(k,N+2)=L-K*exp(-r*(T-t(k)));
34 end
35
36 % Discrétisation de l'équation de Black et Scholes %
37
38 for n=M+2:-1:2
39     for i=2:N+1
40         V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
41     end
42 end
43
44 figure;
45 hold;
46 plot(S,V(M+2,:));
47 plot(S,V(floor(T/(2*dt))+1,:));
48 plot(S,V(1,:));
49 legend('t=T','t=T/2','t=0');
50 title('Prix de l option t=T , t=T/2 et t=0')
51
52 end
53
54 % la condition finale %
55
56 function [f] = condition_finale(S,K)
57 f = max(S-K,0);
58 end
```

FIGURE 7 – Code Sous MATLAB du fichier Code3.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Tracer la fonction solution  $V(S, t)$

On trace la surface de  $(S, t, V)$  pour visualiser la solution :

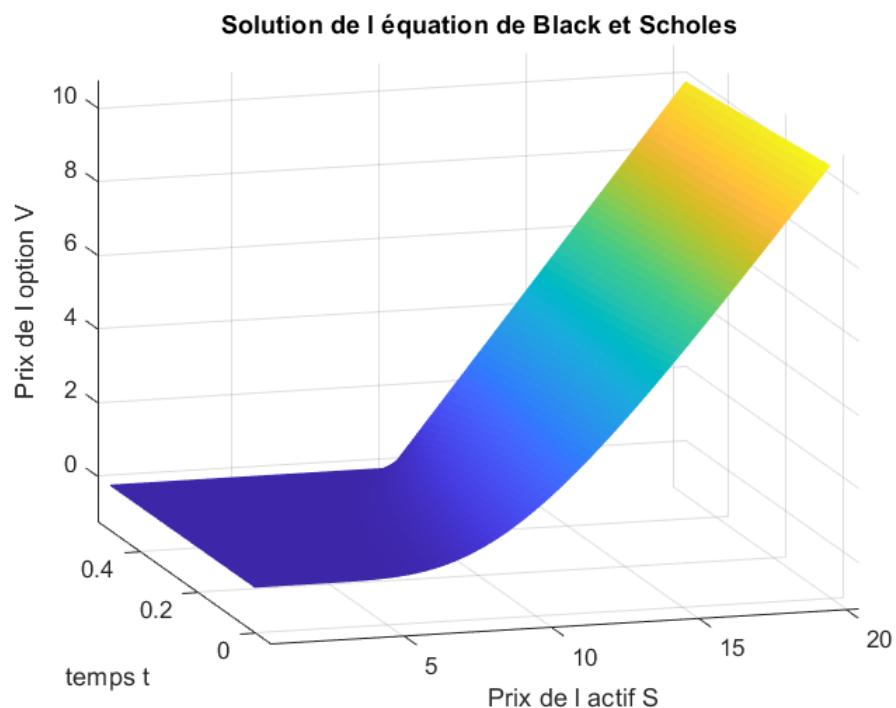


FIGURE 8 – Surface de la fonction solution (Conditions de Dirichlet)

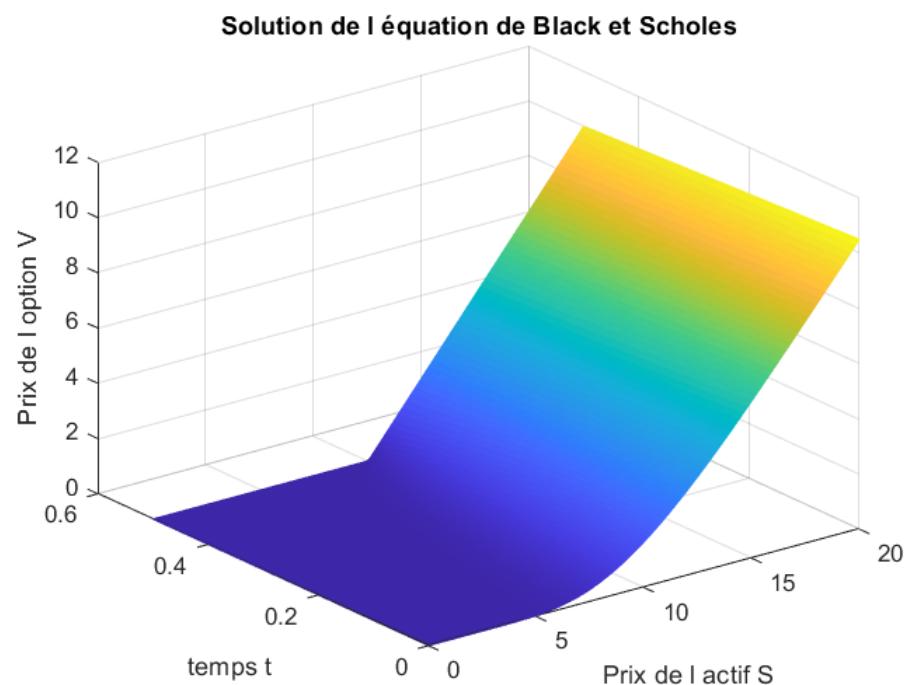


FIGURE 9 – Surface de la fonction solution (Conditions de Dirichlet)

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab pour la surface de  $(S,t,V)$  pour visualiser la solution :

```
3 % function [] = visualiser_fonction_V()
4
5 % Définition des constantes %
6
7 L=20;
8 K=10;
9 T=0.5;
10 r=0.1;
11 sigma=0.5;
12 N=99;
13 M=4999;
14
15 % Définition des vecteurs %
16
17 S=linspace(0,L,N+2);
18 t=linspace(0,T,M+2);
19 dt=T/(M+2);
20 ds=L/(N+2);
21 V=zeros(M+2,N+2);
22
23 % Implementation de la condition finale %
24
25 for j=1:N+2
26     V(M+2,j)=condition_finale(S(j),K);
27 end
28
29 % Implementation des conditions aux limites %
30
31 for k=1:M+1
32     V(k,1)=0;
33     V(k,N+2)=L-K*exp(-r*(T-t(k)));
34 end
35
36 % Discréttisation de l'équation de Black et Scholes %
37
38 for n=M+2:-1:2
39     for i=2:N+1
40         V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
41     end
42 end
43
44 figure;
45 mesh(S,t,V)
46 xlabel('Prix de l actif S')
47 ylabel('temps t')
48 zlabel('Prix de l option V')
49 title('Solution de l équation de Black et Scholes')
50
51 end
52
53 % la condition finale %
54
55 function [f] = condition_finale(S,K)
56 f = max(S-K,0);
57 end
```

FIGURE 10 – Code Sous MATLAB du fichier Code4.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Divergence de l'algorithme explicite

On trace la surface de  $(S,t,V)$  pour visualiser la solution à l'instant  $\Delta t = 10^{-3}$  :

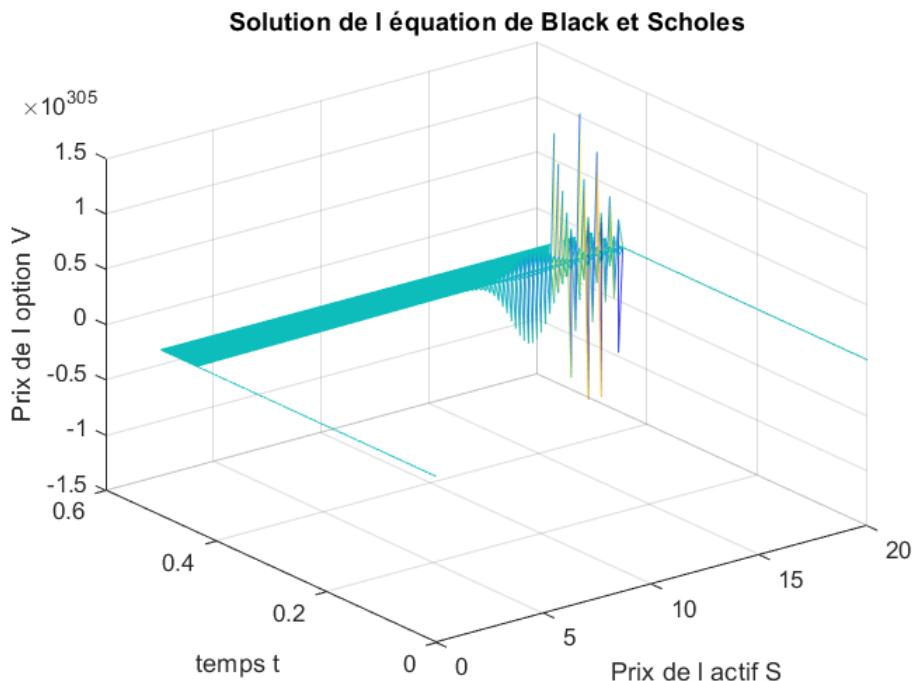


FIGURE 11 – Divergence de l'algorithme explicite

On trace la surface de  $(S,t,V)$  pour visualiser la solution à l'instant  $\Delta t = 10^{-2}$  :

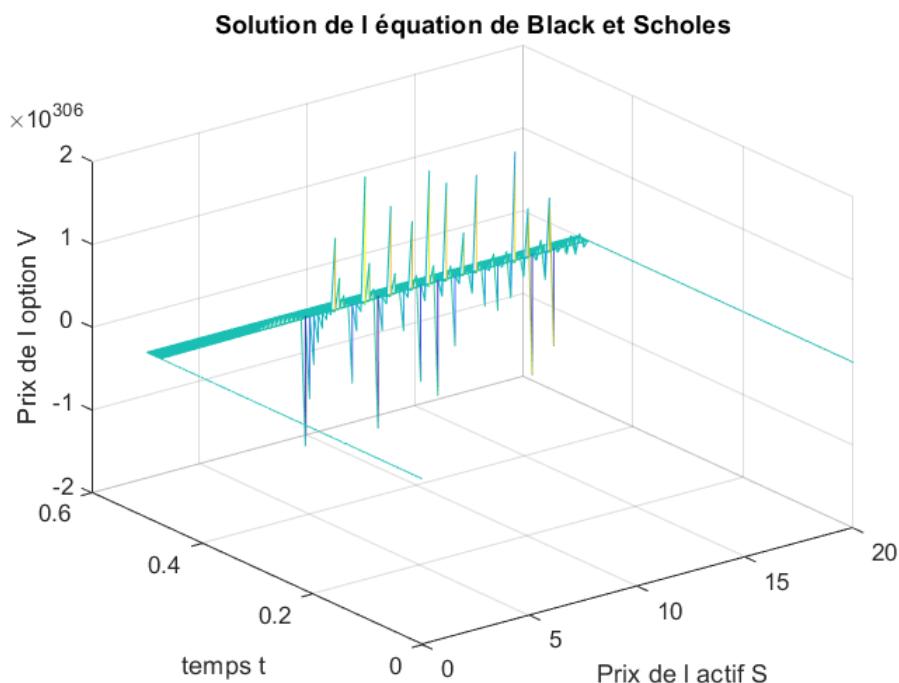


FIGURE 12 – Divergence de l'algorithme explicite

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab pour Divergence de l'algorithme explicite :

```
3  function [] = visualiser_fonction_V()
4
5      % Définition des constantes %
6
7      L=20;
8      K=10;
9      T=0.5;
10     r=0.1;
11     sigma=0.5;
12     N=99;
13     M=4999;
14
15     % Définition des vecteurs %
16
17     S=linspace(0,L,N+2);
18     t=linspace(0,T,M+2);
19     dt=T/(M+2);
20     dt=10^(-3);
21     V=zeros(M+2,N+2);
22
23     % Implementation de la condition finale %
24
25     for j=1:N+2
26         V(M+2,j)=condition_finale(S(j),K);
27     end
28
29     % Implementation des conditions aux limites %
30
31     for k=1:M+1
32         V(k,1)=0;
33         V(k,N+2)=L-K*exp(-r*(T-t(k)));
34     end
35
36     % Discrétisation de l'équation de Black et Scholes %
37
38     for n=M+2:-1:2
39         for i=2:N+1
40             V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*dt)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(dt^2))-r*V(n,i));
41         end
42     end
43
44     figure;
45     mesh(S,t,V)
46     xlabel('Prix de l actif S')
47     ylabel('temps t')
48     zlabel('Prix de l option V')
49     title('Solution de l équation de Black et Scholes')
50
51     end
52
53     % la condition finale %
54
55     function [f] = condition_finale(S,K)
56         f = max(S-K,0);
57     end
```

FIGURE 13 – Code Sous MATLAB du fichier Code5.m

## 2 Partie II : Vanilla. Call. Conditions aux limites de Neumann

L'équation de Black et Scholes avec la condition finale et les conditions aux limites de Neumann s'écrit de la forme :

$$\left\{ \begin{array}{l} \frac{\partial V}{\partial t} - rV + rS \frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} = 0 \\ V(t = T, S) = \max(S - K, 0) \\ \frac{\partial V}{\partial S}(t, S = 0) = 0 \\ \frac{\partial V}{\partial S}(t, S = L) = 1 \end{array} \right.$$

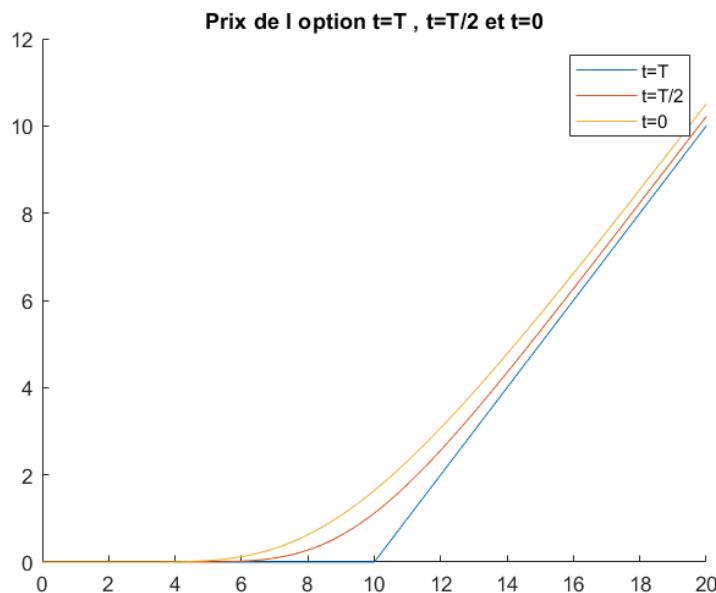


FIGURE 14 – Graphe des conditions aux limites de Neumann

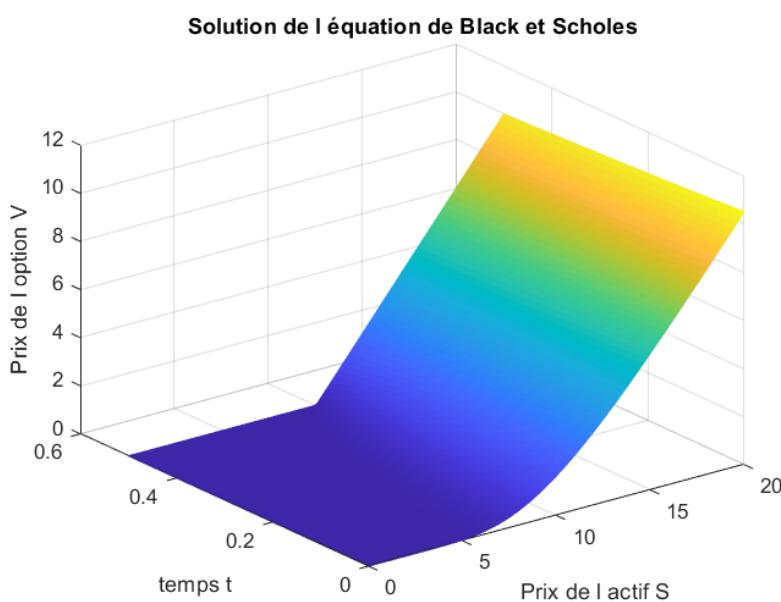


FIGURE 15 – Surface de la fonction solution (Neumann)

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab du graphe des conditions aux limites de Neumann :

```
1 -     visualiser_fonction_V()
2
3 - function [] = visualiser_fonction_V()
4
5 - % Définition des constantes %
6
7 - L=20;
8 - K=10;
9 - T=0.5;
10 - r=0.1;
11 - sigma=0.5;
12 - N=99;
13 - M=4999;
14
15 - % Définition des vecteurs %
16
17 - S=linspace(0,L,N+2);
18 - t=linspace(0,T,M+2);
19 - dt=T/(M+2);
20 - ds=L/(N+2);
21 - V=zeros(M+2,N+2);
22
23 - % Implementation de la condition finale %
24
25 - for j=1:N+2
26 -     V(M+2,j)=condition_finale(S(j),K);
27 - end
28
29 - % Discrétisation de l'équation de Black et Scholes %
30
31 - for n=M+2:-1:2
32 -     for i=2:N+1
33 -         V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
34 -     end
35
36 - % Implementation des conditions aux limites de Neumann %
37
38 -     V(n-1,1)=V(n-1,2);
39 -     V(n-1,N+2)=V(n-1,N+1)+ds;
40 - end
41
42 - figure;
43 - hold;
44 - plot(S,V(M+2,:));
45 - plot(S,V(floor(T/(2*dt))+1,:));
46 - plot(S,V(1,:));
47 - legend('t=T','t=T/2','t=0');
48 - title('Prix de l option t=T , t=T/2 et t=0')
49
50 - end
51
52 - % la condition finale %
53
54 - function [f] = condition_finale(S,K)
55 - f = max(S-K,0);
56 - end
```

FIGURE 16 – Code Sous MATLAB du fichier Code6.m

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab de la surface de la fonction solution sous conditions aux limites de Neumann :

```
3  function [] = visualiser_fonction_V()
4
5      % Définition des constantes %
6
7      L=20;
8      K=10;
9      T=0.5;
10     r=0.1;
11     sigma=0.5;
12     N=99;
13     M=4999;
14
15     % Définition des vecteurs %
16
17     S=linspace(0,L,N+2);
18     t=linspace(0,T,M+2);
19     dt=T/(M+2);
20     ds=L/(N+2);
21     V=zeros(M+2,N+2);
22
23     % Implementation de la condition finale %
24
25     for j=1:N+2
26         V(M+2,j)=condition_finale(S(j),K);
27     end
28
29     % Discrétisation de l'équation de Black et Scholes %
30
31     for n=M+2:-1:2
32         for i=2:N+1
33             V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
34         end
35
36     % Implementation des conditions aux limites de Neumann %
37
38     V(n-1,1)=V(n-1,2);
39     V(n-1,N+2)=V(n-1,N+1)+ds;
40     end
41
42     figure;
43     mesh(S,t,V)
44     xlabel('Prix de l actif S')
45     ylabel('temps t')
46     zlabel('Prix de l option V')
47     title('Solution de l équation de Black et Scholes')
48
49     end
50
51     % la condition finale %
52
53     function [f] = condition_finale(S,K)
54         f = max(S-K,0);
55     end
```

FIGURE 17 – Code Sous MATLAB du fichier Code7.m

## 3 Partie III : Vanilla. Put. Conditions aux limites de Neumann

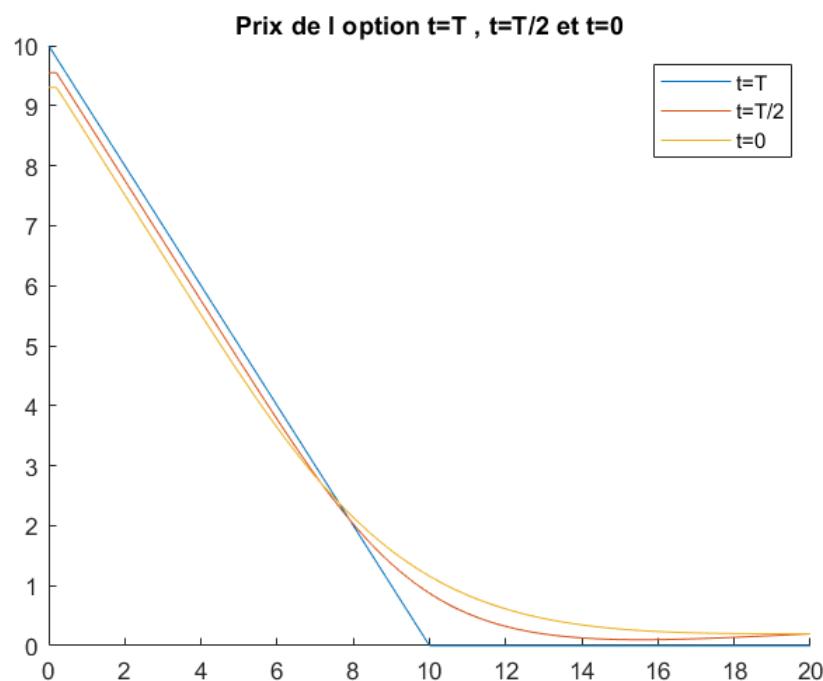


FIGURE 18 – Graphe des conditions aux limites de Neumann

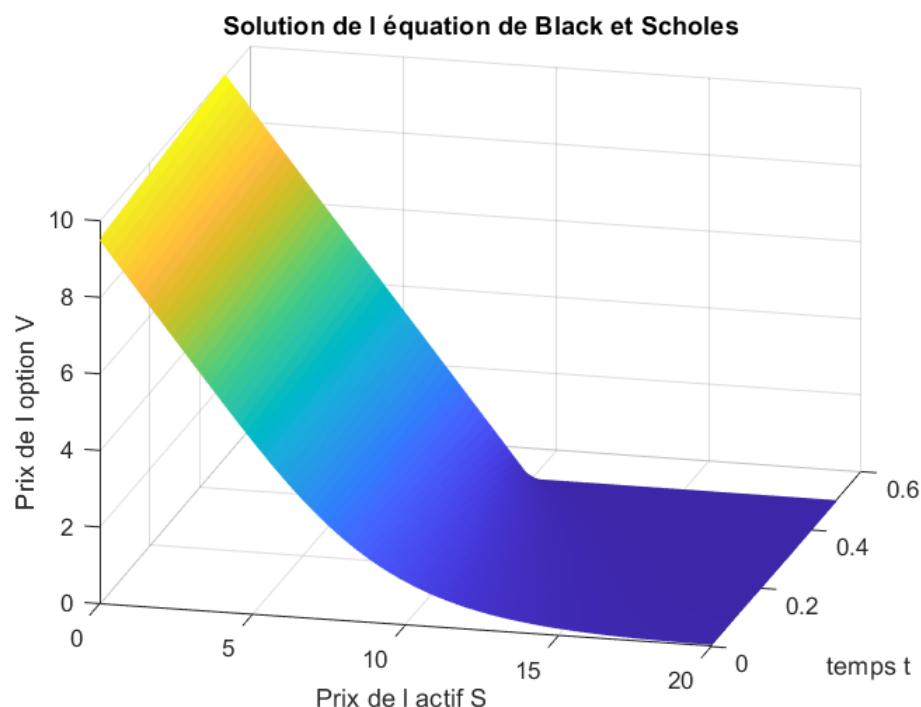


FIGURE 19 – Surface de la fonction solution (Neumann)

# TP: Méthodes numériques avancées appliquées à la finance

Le code sous Matlab du graphe des conditions aux limites de Neumann :

```
1 - visualiser_fonction_V()
2
3 - function [] = visualiser_fonction_V()
4
5 - % Définition des constantes %
6
7 - L=20;
8 - K=10;
9 - T=0.5;
10 - r=0.1;
11 - sigma=0.5;
12 - N=99;
13 - M=4999;
14
15 - % Définition des vecteurs %
16
17 - S=linspace(0,L,N+2);
18 - t=linspace(0,T,M+2);
19 - dt=T/(M+2);
20 - ds=L/(N+2);
21 - V=zeros(M+2,N+2);
22
23 - % Implementation de la condition finale %
24
25 - for j=1:N+2
26 -     V(M+2,j)=condition_finale(S(j),K);
27 - end
28
29 - for n=M+2:-1:2
30 -     for i=2:N+1
31 -         % Implementation des conditions aux limites de Neumann %
32 -
33 -         V(n-1,1)=V(n-1,2);
34 -         V(n-1,N+2)=V(n-1,N+1)+ds;
35
36 -         % Discrétisation de l'équation de Black et Scholes %
37
38 -         V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
39 -     end
40 - end
41
42 - figure;
43 - hold;
44 - plot(S,V(M+2,:));
45 - plot(S,V(floor(T/(2*dt))+1,:));
46 - plot(S,V(1,:));
47 - legend('t=T','t=T/2','t=0');
48 - title('Prix de l option t=T , t=T/2 et t=0')
49
50 - end
51
52 - % la condition finale %
53
54 - function [f] = condition_finale(S,K)
55 - f = max(0,K-S);
56 - end
```

FIGURE 20 – Code Sous MATLAB du fichier Code8.m

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab de la surface de la fonction solution sous conditions aux limites de Neumann :

```
3  function [] = visualiser_fonction_V()
4
5      % Définition des constantes %
6
7      L=20;
8      K=10;
9      T=0.5;
10     r=0.1;
11     sigma=0.5;
12     N=99;
13     M=4999;
14
15     % Définition des vecteurs %
16
17     S=linspace(0,L,N+2);
18     t=linspace(0,T,M+2);
19     dt=T/(M+2);
20     ds=L/(N+2);
21     V=zeros(M+2,N+2);
22
23     % Implementation de la condition finale %
24
25     for j=1:N+2
26         V(M+2,j)=condition_finale(S(j),K);
27     end
28
29     for n=M+2:-1:2
30         for i=2:N+1
31             % Implementation des conditions aux limites de Neumann %
32
33             V(n-1,1)=V(n-1,2);
34             V(n-1,N+2)=V(n-1,N+1)+ds;
35
36             % Discrétisation de l'équation de Black et Scholes %
37
38             V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
39         end
40     end
41
42     figure;
43     mesh(S,t,V)
44     xlabel('Prix de l actif S')
45     ylabel('temps t')
46     zlabel('Prix de l option V')
47     title('Solution de l équation de Black et Scholes')
48
49 end
50
51 % la condition finale %
52
53 function [f] = condition_finale(S,K)
54     f = max(0,K-S);
55 end
```

FIGURE 21 – Code Sous MATLAB du fichier Code9.m

## 4 Partie IV : Put Américain

Une personne qui possède une option américaine peut l'exercer à n'importe quel moment  $t_n \in [0, T]$ . La fonction Pay-Off s'écrit de la forme :  $Put_{Pay-Off} = \max(K - S, 0)$

### → Justification de l'algorithme utilisé

Grâce à la méthode de discrétisation on obtient l'algorithme (Cf Cours 3) :

$$V_i^{n-1} = \max\left(V_{i+1}^n \frac{\Delta t}{2} \left(\sigma^2 \frac{S(i)^2}{(\Delta S)^2} + r \frac{S(i)}{(\Delta S)}\right) + V_i^n \left(1 - \Delta t \left(\sigma^2 \frac{S(i)^2}{(\Delta S)^2} + r\right)\right) + V_{i-1}^n \frac{\Delta t}{2} \left(\sigma^2 \frac{S(i)^2}{(\Delta S)^2} - r \frac{S(i)}{(\Delta S)}\right), \max(K - S(i), 0)\right)$$

### → Tracer les graphes du Put Européen et Pur Américain dans le même graphe

Graphes du Put européen et Américain dans le même système de coordonnées :

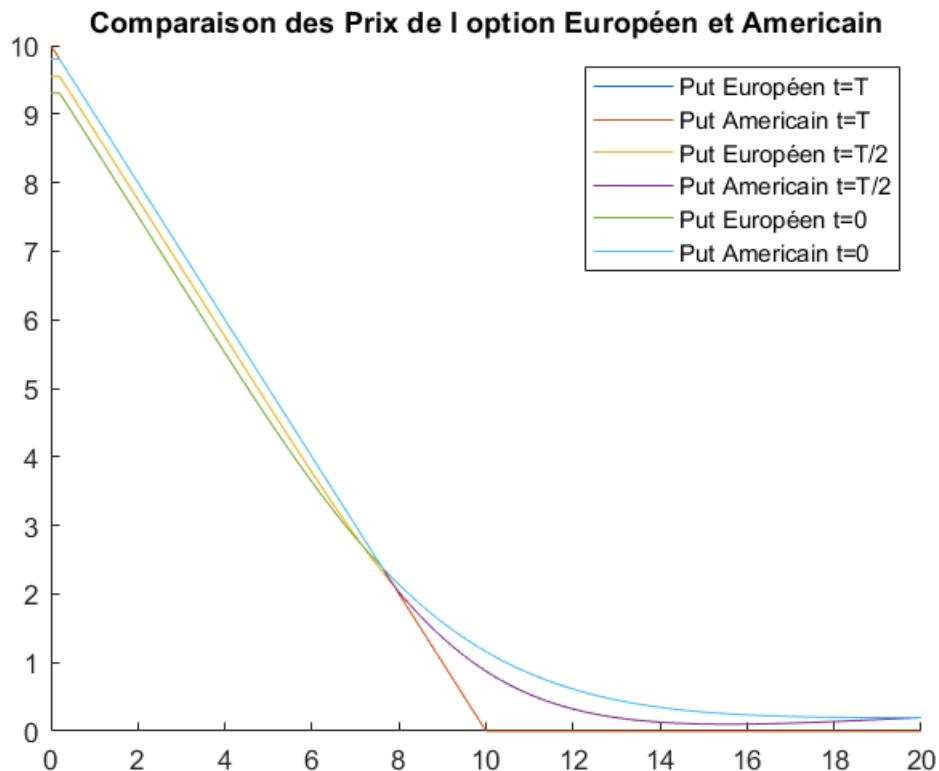


FIGURE 22 – Graphe de Put Européen et Américain

# TP: Méthodes numériques avancées appliquées à la finance

Le code sous Matlab des graphes du Put européen et Américain dans le même système de coordonnées :

```
3  function [] = visualiser_fonction_V()
4
5      % Définition des constantes %
6
7      L=20;
8      K=10;
9      T=0.5;
10     r=0.1;
11     sigma=0.5;
12     N=99;
13     M=4999;
14
15     % Définition des vecteurs %
16
17     S=linspace(0,L,N+2);
18     t=linspace(0,T,M+2);
19     dt=T/(M+2);
20     ds=L/(N+2);
21     V=zeros(M+2,N+2);
22     Veu=zeros(M+2,N+2);
23     Vam=zeros(M+2,N+2);
24
25
26     % Implementation de la condition finale %
27
28     for j=1:N+2
29         V(M+2,j)=condition_finale(S(j),K);
30         Veu(M+2,j)=condition_finale(S(j),K);
31         Vam(M+2,j)=condition_finale(S(j),K);
32     ,
33     for n=M+2:-1:2
34         for i=2:N+1
35
36             % Implementation des conditions aux limites de Neumann %
37
38             V(n-1,1)=V(n-1,2);
39             V(n-1,N+2)=V(n-1,N+1)+ds;
40             Veu(n-1,1)=Veu(n-1,2); % Européen
41             Veu(n-1,N+2)=Veu(n-1,N+1)+ds; % Européen
42             Vam(n-1,1)=Vam(n-1,2); % Americain
43             Vam(n-1,N+2)=Vam(n-1,N+1)+ds; % Americain
44
45             % Discrétisation de l'équation de Black et Scholes %
46
47             V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
48             Veu(n-1,i)=V(n-1,i); % Européen
49             Vam(n-1,i)=max(V(n-1,i),max(K-S(i),0)); % Americain
50         end
51     end
52
53     figure;
54     hold;
55     plot(S,Veu(M+2,:));
56     plot(S,Vam(M+2,:));
57     plot(S,Veu(floor(T/(2*dt))+1,:));
58     plot(S,Vam(floor(T/(2*dt))+1,:));
59     plot(S,Veu(1,:));
60     plot(S,Vam(1,:));
61     legend('Put Européen t=T','Put Americain t=T','Put Européen t=T/2','Put Americain t=T/2','Put Européen t=0','Put Americain t=0');
62     title('Comparaison des Prix de l option Européen et Americain')
63
64     end
65
66     % la condition finale %
67
68     function [f] = condition_finale(S,K)
69         f = max(0,K-S);
70     end
```

FIGURE 23 – Code Sous MATLAB du fichier Code10.m

## 5 Partie V : Option de Butterfly

Le prix de l'option de Butterfly :

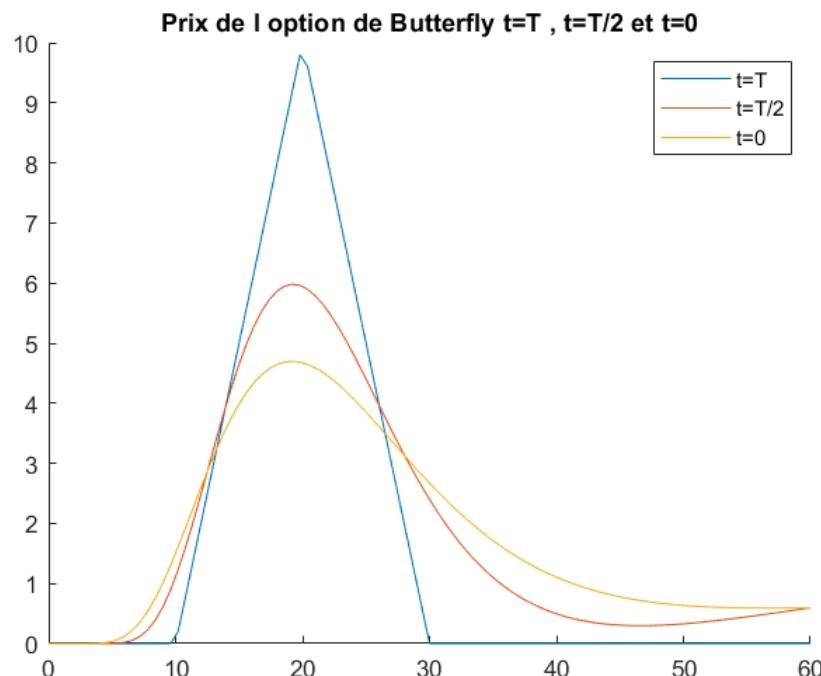


FIGURE 24 – Option butterfly

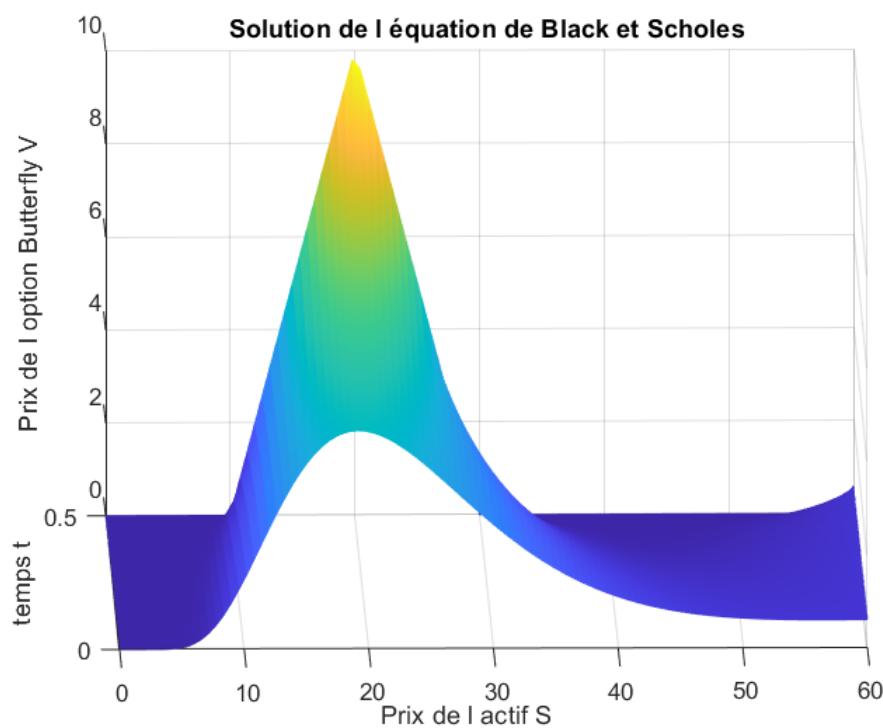


FIGURE 25 – Surface de la fonction solution option Butterfly

# TP: Méthodes numériques avancées appliquées à la finance

Le code sous Matlab du graphe de l'option Butterfly sous conditions aux limites de Neumann :

```
1 -    visualiser_fonction_V()
2
3 -    function [] = visualiser_fonction_V()
4
5 -        % Définition des constantes %
6
7 -        L=60;
8 -        K=10;
9 -        T=0.5;
10 -       r=0.1;
11 -       sigma=0.5;
12 -       N=99;
13 -       M=4999;
14
15 -       % Définition des vecteurs %
16
17 -       S=linspace(0,L,N+2);
18 -       t=linspace(0,T,M+2);
19 -       dt=T/(M+2);
20 -       ds=L/(N+2);
21 -       V=zeros(M+2,N+2);
22
23 -       % Implementation de la condition finale %
24
25 -       for j=1:N+2
26 -           V(M+2,j)=PayOff_Butterfly(S(j),K);
27 -       end
28
29 -       for n=M+2:-1:2
30 -           for i=2:N+1
31
32 -               % Implementation des conditions aux limites de Neumann %
33 -               V(n-1,1)=V(n-1,2);
34 -               V(n-1,N+2)=V(n-1,N+1)+ds;
35
36 -               % Discrétisation de l'équation de Black et Scholes %
37
38 -               V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
39 -           end
40 -       end
41
42 -       figure;
43 -       hold;
44 -       plot(S,V(M+2,:));
45 -       plot(S,V(floor(T/(2*dt))+1,:));
46 -       plot(S,V(1,:));
47 -       legend('t=T','t=T/2','t=0');
48 -       title('Prix de l option de Butterfly t=T , t=T/2 et t=0')
49 -   end
50
51 -   % la condition finale %
52
53 -   function [f] = PayOff_Butterfly(S,K)
54 -       f = max(S-K,0)+max(S-3*K,0)-2*max(S-2*K,0);
55 -   end
```

FIGURE 26 – Code Sous MATLAB du fichier Code12.m

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab du programme complet de la fonction solution option Butterfly sous conditions aux limites de Neumann :

```
3  function [] = visualiser_fonction_V()
4
5      % Définition des constantes %
6
7      L=60;
8      K=10;
9      T=0.5;
10     r=0.1;
11     sigma=0.5;
12     N=99;
13     M=4999;
14
15     % Définition des vecteurs %
16
17     S=linspace(0,L,N+2);
18     t=linspace(0,T,M+2);
19     dt=T/(M+2);
20     ds=L/(N+2);
21     V=zeros(M+2,N+2);
22
23     % Implementation de la condition finale %
24
25     for j=1:N+2
26         V(M+2,j)=PayOff_Butterfly(S(j),K);
27     end
28
29     for n=M+2:-1:2
30         for i=2:N+1
31
32             % Implementation des conditions aux limites de Neumann %
33
34             V(n-1,1)=V(n-1,2);
35             V(n-1,N+2)=V(n-1,N+1)+ds;
36
37             % Discrétisation de l'équation de Black et Scholes %
38
39             V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
40
41         end
42     end
43
44     figure;
45     mesh(S,t,V)
46     xlabel('Prix de l actif S')
47     ylabel('temps t')
48     zlabel('Prix de l option Butterfly V')
49     title('Solution de l équation de Black et Scholes')
50
51     % la condition finale %
52
53     function [f] = PayOff_Butterfly(S,K)
54         f = max(S-K,0)+max(S-3*K,0)-2*max(S-2*K,0);
55     end
```

FIGURE 27 – Code Sous MATLAB du fichier Code13.m

## 6 Partie VI : Comparaison entre les solutions numérique et analytique

→ Tracer le graphe  $S_i \rightarrow V_{analytique}(t = 0, S)$

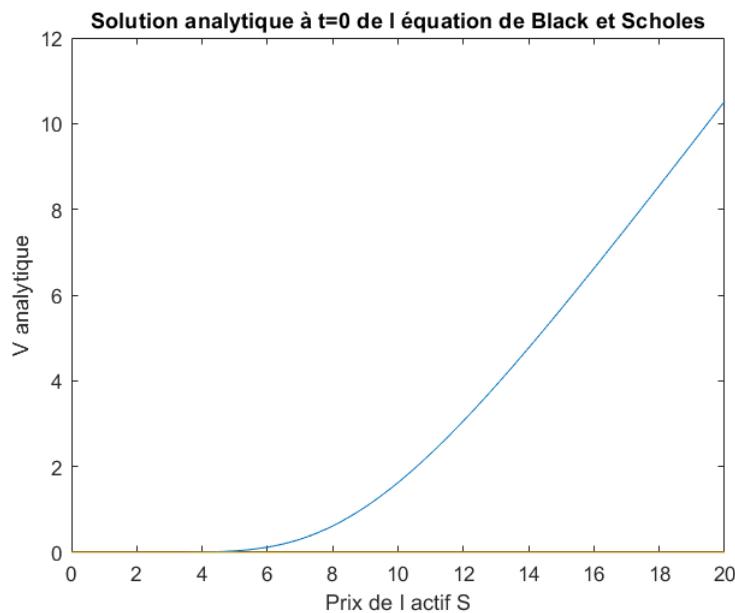


FIGURE 28 – Graphe de la solution analytique  $V$  à  $t=0$

→ Tracer la surface de la solution analytique

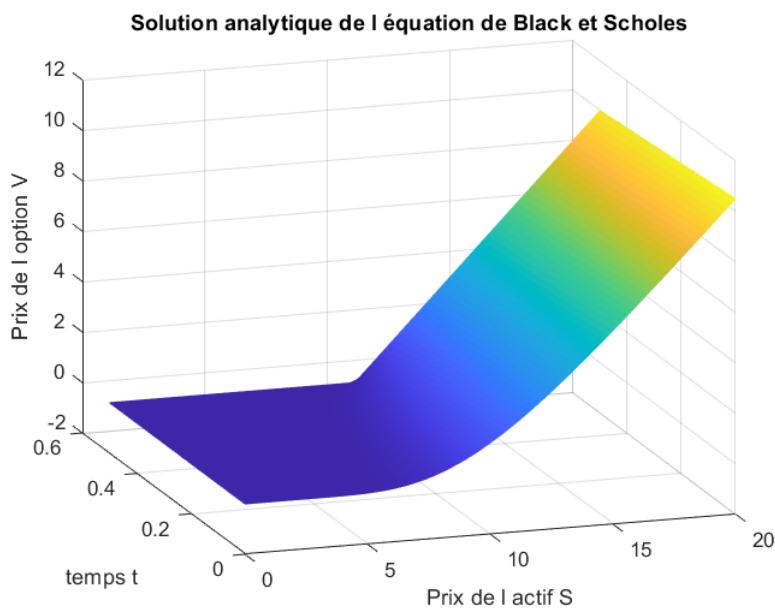


FIGURE 29 – Surface de la solution analytique

# TP: Méthodes numériques avancées appliquées à la finance

Le code sous Matlab de graphe de la solution analytique à l'instant t=0 :

```
4     function [] = visualiser_fonction_V_theorique()
5
6         % Définition des constantes %
7
8         L=20;
9         K=10;
10        T=0.5;
11        r=0.1;
12        sigma=0.5;
13        N=99;
14        M=4999;
15
16        % Définition des vecteurs %
17
18        S=linspace(0,L,N+2);
19        t=linspace(0,T,M+2);
20        V=zeros(N+2);
21
22    for i=1:N+2
23        if i==1
24            V(i)=0;
25        else
26            V(i)=BS_theorie(S(i),K,r,sigma,0,T);
27        end
28    end
29
30    figure;
31    plot(S,V)
32    xlabel('Prix de l actif S')
33    ylabel('V analytique')
34    title('Solution analytique à t=0 de l équation de Black et Scholes')
35    end
36
37    function [f] = BS_theorie(S,K,r,sigma,t,T)
38        if (t==T)
39            f=max(S-K,0);
40        else
41            f=S*N(d1(S,K,r,sigma,t,T))-K*exp(-r*(T-t))*N(d2(S,K,r,sigma,t,T));
42        end
43    end
44
45    function [f] = d1(S,K,r,sigma,t,T)
46        f = (log(S/K)+(r+(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
47    end
48
49    function [f] = d2(S,K,r,sigma,t,T)
50        f = (log(S/K)+(r-(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
51    end
52
53    function [f] = N(x)
54        f = 1/2*(1+erf(x/sqrt(2)));
55    end
```

FIGURE 30 – Code Sous MATLAB du fichier Code14.m

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab de la surface de la solution analytique :

```
4  function [] = visualiser_fonction_V_theorique()
5
6      % Définition des constantes %
7
8      L=20;
9      K=10;
10     T=0.5;
11     r=0.1;
12     sigma=0.5;
13     N=99;
14     M=4999;
15
16     % Définition des vecteurs %
17
18     S=linspace(0,L,N+2);
19     t=linspace(0,T,M+2);
20
21     V=zeros(M+2,N+2);
22     for n=1:M+2
23         for i=1:N+2
24             if i==1
25                 V(n,i)=0;
26             else
27                 V(n,i)=BS_theorie(S(i),K,r,sigma,t(n),T);
28             end
29         end
30     end
31
32     figure;
33     mesh(S,t,V)
34     xlabel('Prix de l actif s')
35     ylabel('temps t')
36     zlabel('Prix de l option V')
37     title('Solution analytique de l équation de Black et Scholes')
38
39 end
40
41 function [f] = BS_theorie(S,K,r,sigma,t,T)
42 if (t==T)
43     f=max(S-K,0);
44 else
45     f=S*N(d1(S,K,r,sigma,t,T))-K*exp(-r*(T-t))*N(d2(S,K,r,sigma,t,T));
46 end
47 end
48
49 function [f] = d1(S,K,r,sigma,t,T)
50     f = (log(S/K)+(r+(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
51 end
52
53 function [f] = d2(S,K,r,sigma,t,T)
54     f = (log(S/K)+(r-(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
55 end
56
57 function [f] = N(x)
58     f = 1/2*(1+erf(x/sqrt(2)));
59 end
```

FIGURE 31 – Code Sous MATLAB du fichier Code15.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Tracer le graphe  $S_i \rightarrow |V_{analytique}(t = 0, S_i) - V_{condition\_Neumann}(t = 0, S_i)|$

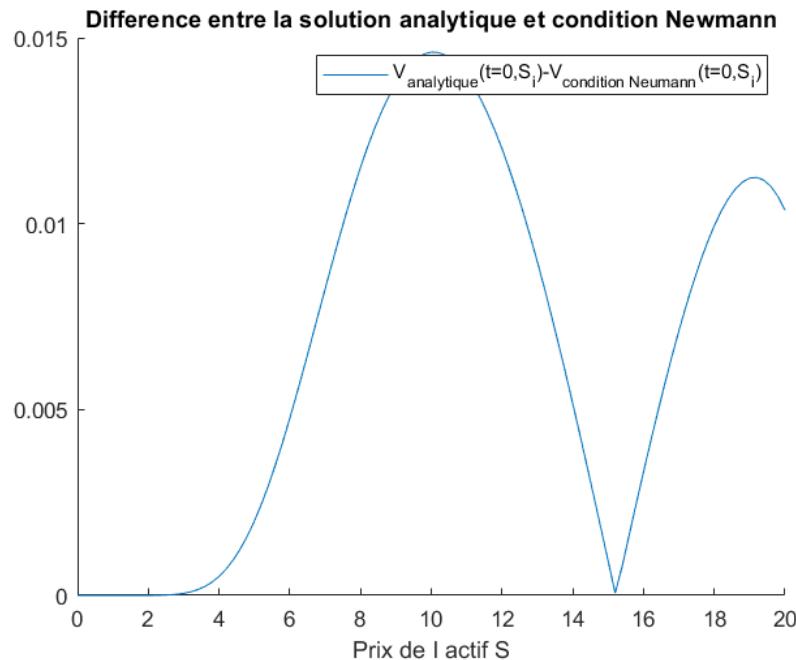


FIGURE 32 – Graphe de  $S_i \rightarrow |V_{analytique}(t = 0, S_i) - V_{condition\_Neumann}(t = 0, S_i)|$

→ Tracer le graphe  $S_i \rightarrow |V_{analytique}(t = 0, S_i) - V_{condition\_Dirichlet}(t = 0, S_i)|$

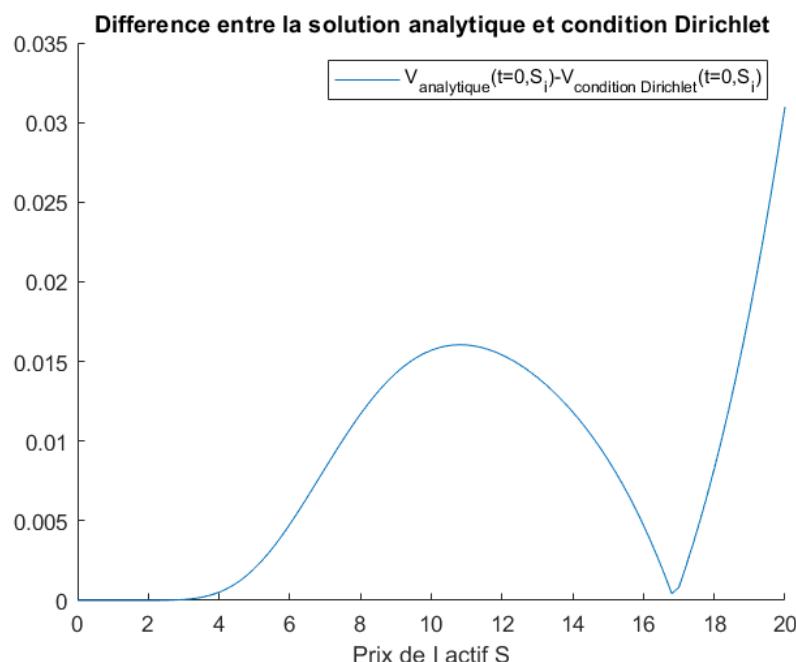


FIGURE 33 – Graphe de  $S_i \rightarrow |V_{analytique}(t = 0, S_i) - V_{condition\_Dirichlet}(t = 0, S_i)|$

## → Comparaison entre les deux graphes

On peut calculer les erreurs numériques imposés par les deux conditions aux limites. Pour cela on utilise la solution analytique de l'équation de Black et Scholes en points discrètes. Puis on calcule en mêmes points la différence entre les valeurs analytiques et celles de numériques calculées à l'aide de conditions aux limites de Dirichlet et de Neumann.

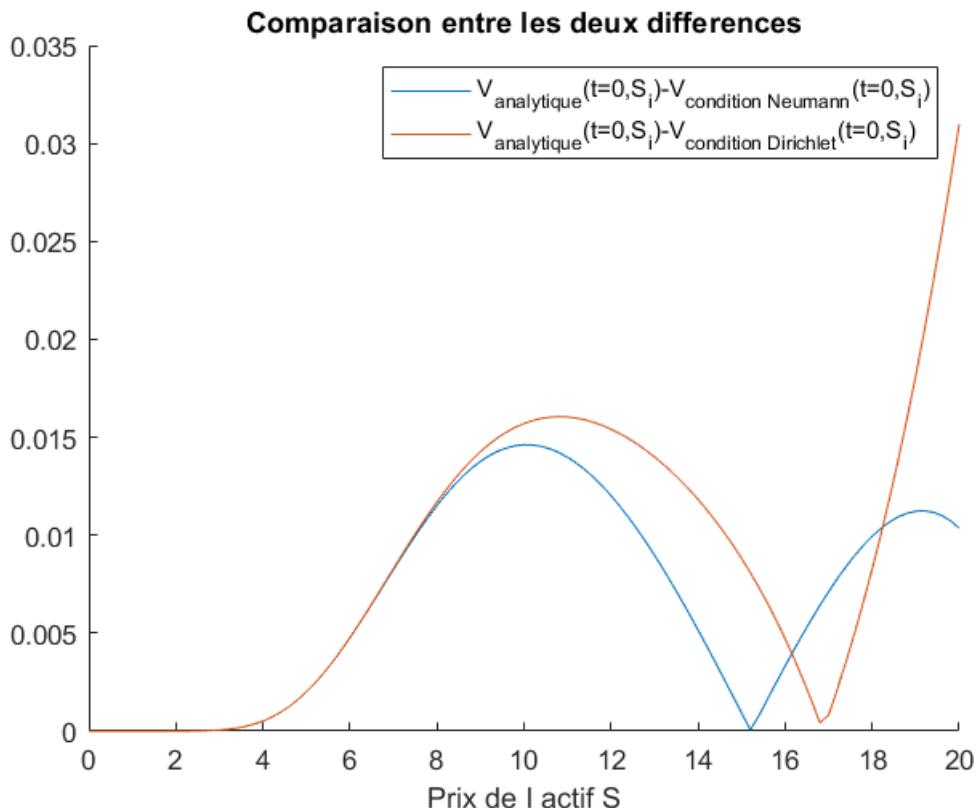


FIGURE 34 – Graphe de  $S_i \rightarrow |V_{analytique}(t = 0, S_i) - V_{condition\ Dirichlet}(t = 0, S_i)|$

On constate que l'ordre de grandeur de l'erreur imposé par les deux conditions aux limites est le même, mais les signes en  $S = L$  sont opposés.

L'ordre de grandeur de l'erreur imposé par les conditions de Neumann diminue si  $S$  augmente.

L'ordre de grandeur de l'erreur imposé par les conditions de Dirichlet augmente si  $S$  augmente.

En général, on observe l'ordre de grandeur de l'erreur est petit d'où l'approximation de Neumann et Dirichlet pour résoudre l'équation de Black et Scholes.

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab du graphe de  $S_i \rightarrow |V_{analytique}(t = 0, S_i) - V_{condition\_Neumann}(t = 0, S_i)|$ :

```

3 % fonctionnalite-graphes_errerui()
4 %
5 % Définition des constantes %
6
7 L=20;
8 K=10;
9 T=0.5;
10 r=0.1;
11 sigma=0.5;
12 N=59;
13 M=4999;
14
15 % Définition des vecteurs %
16
17 S=linspace(0,L,M*2);
18 t=linspace(0,T,M*2);
19 Vanav_thorique();
20 Vdir=V_Dirichlet();
21 Vneut=V_Neumann();
22
23 figure;
24 plot(S,abs(Vneut-V));
25 plot(S,abs(Vanav_t-thorique()));
26 xlabel('Prix de l actif $t$')
27 legend('V$_{analytique}$(t=0,S_i)-V$_{condition Neumann}$(t=0,S_i)')
28 title('Difference entre la solution analytique et condition Neumann')
29
30 end
31
32 % fonctionnalite [Vanav] = V_theorique()
33 %
34 % Définition des constantes %
35
36 L=20;
37 K=10;
38 T=0.5;
39 r=0.1;
40 sigma=0.5;
41 N=59;
42 M=4999;
43
44 % Définition des vecteurs %
45
46 S=linspace(0,L,M*2);
47 t=linspace(0,T,M*2);
48
49 % Vzeros(N*2,N*2);
50 for i=1:N*2
51 for j=1:N*2
52 if i==1
53 Vn(i,j)=0;
54 else
55 Vn(i,j)=BS_theorie(S(j),K,r,sigma,t(i),T);
56 end
57 end
58 end
59
60 Vanav;
61
62 end
63
64 % fonctionnalite [f] = BS_theorie(S,K,r,sigma,t,T)
65 if (t>=T)
66 f=max(S-K,0);
67 else
68 f=BS*N(d1(S,K,r,sigma,t,T))-K*exp(-r*(T-t))*N(d2(S,K,r,sigma,t,T));
69 end
70 end
71
72 % fonctionnalite [f] = d1(S,K,r,sigma,t,T)
73 f = (log(S/K)+(r+(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
74
75 % fonctionnalite [f] = d2(S,K,r,sigma,t,T)
76 f = (log(S/K)+(r-(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
77
78 % fonctionnalite [f] = K(x)
79 f = 1/2*(1+erf((x-sigma*sqrt(2))/sigma));
80
81 % fonctionnalite [Vdir] = V_Dirichlet()
82 %
83 % Définition des constantes %
84
85 L=20;
86 K=10;
87 T=0.5;
88 r=0.1;
89 sigma=0.5;
90 N=59;
91 M=4999;
92
93 % Définition des vecteurs %
94
95 S=linspace(0,L,M*2);
96 t=linspace(0,T,M*2);
97 dt=t/M*2;
98 ds=L/M*2;
99 Vdir=zeros(M*2,N*2);
100
101 % Implementation de la condition finale %
102 for j=1:N*2
103 Vdir(2,j)=condition_finale(S(j),R);
104 end
105
106 % Implementation des conditions aux limites %
107 for k=1:N*1
108 Vdir(1,k)=0;
109 Vdir(1,N*2+k)=L-K*exp(-r*(T-t(k)));
110 end
111
112 % Discréttisation de l'équation de Black et Scholes %
113 for n=N*2:-1:2
114 for i=2:N*1
115 V(n,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1)) / (2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i)) / (ds^2))-r*V(n,i));
116 end
117 end
118 Vdir=Vdir;
119
120 % fonctionnalite [Vneut] = V_Neumann()
121 %
122 % Définition des constantes %
123
124 L=20;
125 K=10;
126 T=0.5;
127 r=0.1;
128 sigma=0.5;
129 N=59;
130 M=4999;
131
132 % Définition des vecteurs %
133
134 S=linspace(0,L,M*2);
135 t=linspace(0,T,M*2);
136 dt=T/(M*2);
137 ds=L/(M*2);
138 Vzeros=zeros(M*2,N*2);
139
140 % Implementation de la condition finale %
141 for j=1:N*2
142 Vzeros(2,j)=condition_finale(S(j),R);
143 end
144
145 % Discréttisation de l'équation de Black et Scholes %
146 for n=N*2:-1:2
147 for i=2:N*1
148 V(n,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1)) / (2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i)) / (ds^2))-r*V(n,i));
149 end
150 end
151 % Implementation des conditions aux limites de Neumann %
152 for j=1:N*2
153 Vzeros(1,j)=0;
154 end
155
156 % Implementation des conditions aux limites de Neumann %
157 for i=1:N*2
158 Vzeros(1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1)) / (2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i)) / (ds^2))-r*V(n,i));
159 end
160
161 % Implementation des conditions aux limites de Neumann %
162
163 end

```

FIGURE 35 – Code Sous MATLAB du fichier Code16.m

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab du graphe de  $S_i \rightarrow |V_{analytique}(t=0, S_i) - V_{condition\_Dirichlet}(t=0, S_i)|$  :

```

3 % function () = graphe_errre2()
4 %
5 % Définition des constantes %
6
7 t=20;
8 K=10;
9 T=0.5;
10 r=0.1;
11 sigma=0.5;
12 ds=0.01;
13 M=4999;
14
15 % Définition des vecteurs %
16
17 S=linspace(0,L,M*2);
18 t=linspace(0,T,M*2);
19 Vanav_theorique();
20 Vdir=V_Dirichlet();
21 Vneu=V_Neumann();
22
23 figure
24 plot(t,abs(Vneu(1,:)-Vdir(1,:)))
25 xlabel("Prix de l actif $t$")
26 legend("V$_{analytique}$(t=0,S_i)-V$_{condition Dirichlet}$(t=0,S_i)")
27 title("Difference entre la solution analytique et condition Dirichlet")
28
29 end
30
31 % function [Vanav] = V_theorique()
32 %
33 % Définition des constantes %
34
35 L=20;
36 K=10;
37 T=0.5;
38 r=0.1;
39 sigma=0.5;
40 q=0.1;
41 N=4999;
42
43 % Définition des vecteurs %
44
45 S=linspace(0,L,M*2);
46 t=linspace(0,T,M*2);
47
48 % Voption(N*2,N*2);
49 % for i=1:N*2
50 % for j=1:N*2
51 % if i==1
52 %   Vn(1,j)=0;
53 % else
54 %   Vn(j)=
55 %     Vanav(i)*BS_theorie(S(i),K,r,sigma,t(i),T);
56 % end
57 % end
58 % end
59
60 Vanav;
61
62 end
63
64 % function [f] = BS_theorie(S,K,r,sigma,t,T)
65 if (t>T)
66   f=max(S-K,0);
67 else
68   f=BS*N(d1(S,K,r,sigma,t,T))-K*exp(-r*(T-t))*N(d2(S,K,r,sigma,t,T));
69 end
70
71
72 % function [f] = d1(S,K,r,sigma,t,T)
73 f = (log(S/K)+(r+(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
74
75 % function [f] = d2(S,K,r,sigma,t,T)
76 f = (log(S/K)+(r-(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
77
78 % function [f] = K(x)
79 f = 1/2*(1+erf(x/sqrt(2)));
80
81 end
82
83
84 % function [Vdir] = V_Dirichlet()
85 %
86 % Définition des constantes %
87
88 L=20;
89 K=10;
90 T=0.5;
91 r=0.1;
92 sigma=0.5;
93 ds=0.01;
94 M=4999;
95
96 % Définition des vecteurs %
97
98 S=linspace(0,L,M*2);
99 t=linspace(0,T,M*2);
100 dt=t/M*2;
101 ds=L/M*2;
102 V=zeros(M*2,N*2);
103
104 % Implementation de la condition finale %
105
106 for j=1:N*2
107   V(M+2,j)=condition_finale(S(j),R);
108 end
109
110 % Implementation des conditions aux limites %
111
112 for k=1:M*2
113   V(1,k)=0;
114   V(L,k)=0;
115   V(K,R+2)=L-K*exp(-r*(T-t(k)));
116 end
117
118 % Discréttisation de l'équation de Black et Scholes %
119
120 for n=M*2:-1:2
121 for i=2:N*2
122   V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1)) / (2*ds)+(1/2)*(sigma^2)*(S(i)^2)*(V(n,i+1)+V(n,i-1)-2*V(n,i)) / (ds^2))-r*V(n,i));
123 end
124
125 Vdir=V;
126
127 end
128
129 % function [Vneu] = V_Neumann()
130 %
131 % Définition des constantes %
132
133 L=20;
134 K=10;
135 T=0.5;
136 r=0.1;
137 sigma=0.5;
138 N=59;
139 M=4999;
140
141 % Définition des vecteurs %
142
143 S=linspace(0,L,M*2);
144 t=linspace(0,T,M*2);
145 dt=T/(M*2);
146 ds=L/(M*2);
147 V=zeros(M*2,N*2);
148
149 % Implementation de la condition finale %
150
151 for j=1:N*2
152   V(M+2,j)=condition_finale(S(j),R);
153 end
154
155 % Discréttisation de l'équation de Black et Scholes %
156
157 for n=M*2:-1:2
158 for i=2:N*2
159   V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1)) / (2*ds)+(1/2)*(sigma^2)*(S(i)^2)*(V(n,i+1)+V(n,i-1)-2*V(n,i)) / (ds^2))-r*V(n,i));
160 end
161
162 % Implementation des conditions aux limites de Neumann %
163
164 end

```

FIGURE 36 – Code Sous MATLAB du fichier Code17.m

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab des deux graphes dans les mêmes coordonnées :

```

3 % function () = graphe_erreur()
4
5 % Définition des constantes %
6
7 L=20;
8 K=10;
9 T=0.5;
10 r=0.1;
11 sigma=0.5;
12 R=59;
13 M=4999;
14
15 % Définition des vecteurs %
16 S=linspace(0,L,M*2);
17 t=linspace(0,T,M*2);
18 Vanalytique(t);
19 Vanalytique();
20 Vdirvt_Dirichlet();
21 Vneut_V_Neumann();
22
23 figure;
24 hold;
25 plot(S,abs(Vanalytique(1,:)-Vneut(1,:)));
26 plot(S,abs(Vanalytique(1,:)-Vdirvt(1,:)));
27 axis([0,20,-0.5,1]);
28 legend('analytique(t=0,S_i)', 'V_(analytique)(t=0,S_i) - V_(condition Neumann)(t=0,S_i)');
29 title('Comparaison entre les deux différences');
30 end
31
32 %function (Vana) = V_theorique()
33
34 % Définition des constantes %
35
36 L=20;
37 K=10;
38 T=0.5;
39 r=0.1;
40 sigma=0.5;
41 R=59;
42 M=4999;
43
44 % Définition des vecteurs %
45 S=linspace(0,L,M*2);
46 t=linspace(0,T,M*2);
47
48 Vzeroc(N=2,N=2);
49 for n=1:N=2
50 for i=1:N=2
51 if i==1
52 if i==n
53 V(n,i)=0;
54 else
55 V(n,i)=R3_theorie(S(i),K,r,sigma,t(n),T);
56 end
57 end
58 end
59
60 Vana=V;
61
62 end
63
64 %function [f] = R3_theorie(S,K,r,sigma,t,T)
65 if S>T
66 f=max(S-T,0);
67 else
68 f=8*(di(S,K,r,sigma,t,T))-8*exp(-r*(T-t))*N(d2(S,K,r,sigma,t,T));
69 end
70
71
72 %function [f] = di(S,K,r,sigma,t,T)
73 f = (Log(S/N)+(r+(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
74
75
76 %function [f] = d2(S,K,r,sigma,t,T)
77 f = (Log(S/N)+(r-(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
78
79
80 %function [f] = r(x)
81 f = 1/2*(1+erfx(sqrt(2)));
82
83
84 %function (Vdir) = V_Dirichlet()
85
86 % Définition des constantes %
87
88 L=20;
89 K=10;
90 T=0.5;
91 r=0.1;
92 sigma=0.5;
93 R=59;
94 M=4999;
95
96 % Définition des vecteurs %
97 S=linspace(0,L,M*2);
98 t=linspace(0,T,M*2);
99 ds=dt/(M*2);
100 dsL=ds/L;
101 Vzeroc(N=2,N=2);
102
103
104 % Implementation de la condition finale %
105 for j=1:N=2
106 V(M+2,j)=condition_finale(S(j),R);
107 end
108
109 % Implementation des conditions aux limites %
110
111 for k=1:M*2
112 V(1,k)=0;
113 V(K,N=2)=R*exp(-r*(T-k));
114 end
115
116 % Discrétilisation de l'équation de Black et Scholes %
117
118 for n=M+2:-1:2
119 for i=1:N=1
120 V(i-1,i)=V(i,i)+dt*(r*S(i)*(V(i,i+1)-V(i,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*(V(i,i+1)+V(i,i-1)-2*V(i,i))/(ds^2))-r*T*V(i,i));
121 end
122 end
123
124 Vdir=V;
125
126 end
127
128
129 %function (Vneu) = V_Neumann()
130
131 % Définition des constantes %
132
133 L=20;
134 K=10;
135 T=0.5;
136 r=0.1;
137 sigma=0.5;
138 R=59;
139 M=4999;
140
141 % Définition des vecteurs %
142 S=linspace(0,L,M*2);
143 t=linspace(0,T,M*2);
144 dt=T/(M*2);
145 ds=dt/L;
146 Vzeroc(R=2,N=2);
147 Vzeroc(R=2,N=2);
148
149 % Implementation de la condition finale %
150
151 for i=1:N=2
152 V(M+2,i)=condition_finale(S(i),R);
153 end
154
155 % Discrétilisation de l'équation de Black et Scholes %
156
157 for n=M+2:-1:2
158 for i=1:N=1
159 V(i-1,i)=V(i,i)+dt*(r*S(i)*(V(i,i+1)-V(i,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*(V(i,i+1)+V(i,i-1)-2*V(i,i))/(ds^2))-r*T*V(i,i));
160 end
161 end
162 % Implementation des conditions aux limites de Neumann %

```

FIGURE 37 – Code Sous MATLAB du fichier Code18.m

## 7 Partie VII : Volatilité locale

Nous utilisons maintenant "Constant Elasticity of Variance Model" (CEV). La volatilité n'est pas une constante :

$$\sigma(t, S) = \frac{1}{\sqrt{S}} \exp(-t/T)$$

→ Tracer le graphe  $S_i \rightarrow V_{analytique}(t = 0, S)$  en utilisant (CEV)

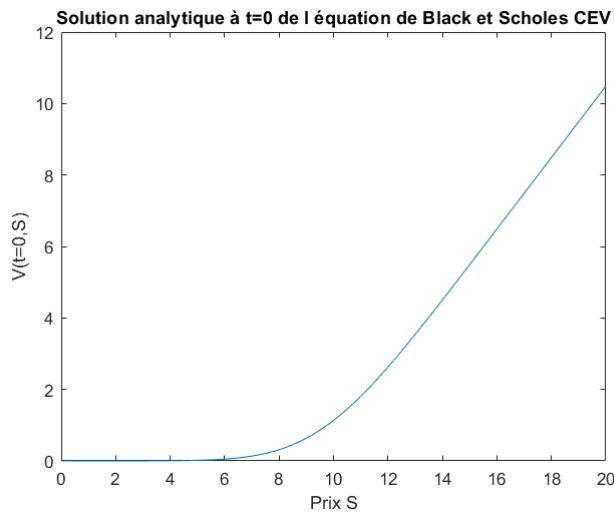


FIGURE 38 – Graphe de la solution analytique  $V$  à  $t=0$  en utilisant (CEV)

→ Tracer la surface de la solution analytique en utilisant (CEV)

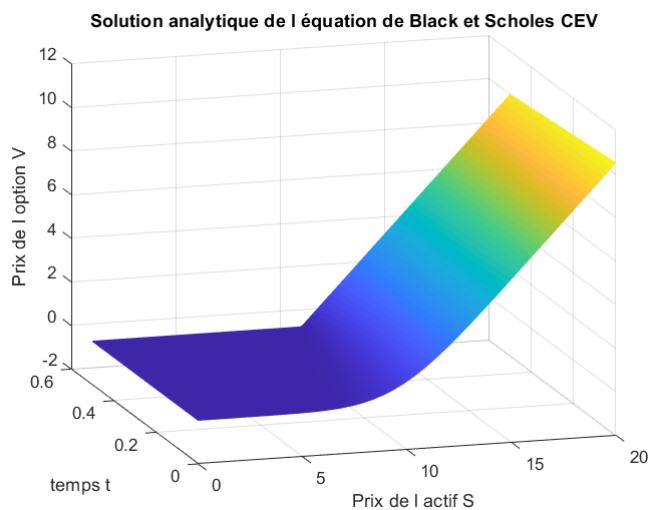


FIGURE 39 – Surface de la solution analytique en utilisant (CEV)

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab du Graphe de la solution analytique V à t=0 en utilisant (CEV) :

```
4 %function [] = visualiser_V_theorique_CEV()
5
6 % Définition des constantes %
7
8 L=20;
9 K=10;
10 T=0.5;
11 r=0.1;
12 N=99;
13 M=4999;
14
15 % Définition des vecteurs %
16
17 S=linspace(0,L,N+2);
18 t=linspace(0,T,M+2);
19 V=zeros(M+2,N+2);
20 sigma=zeros(M+2,N+2);
21
22 % Volatilité locale %
23
24 for p=1:M+2
25 for k=1:N+2
26 sigma(p,k)=(1/sqrt(S(k)))*exp(-t(p)/T);
27 end
28 end
29
30 % Solution analytique %
31
32 for n=1:M+2
33 for i=1:N+2
34 if i==1
35 V(n,i)=0;
36 else
37 V(n,i)=BS_theorie(S(i),K,r,sigma(n,i),t(n),T);
38 end
39 end
40 end
41
42 figure;
43 plot(S,V(1,:))
44 xlabel('Prix S')
45 ylabel('V(t=0,S)')
46 title('Solution analytique à t=0 de l'équation de Black et Scholes CEV')
47
48
49 end
50
51 function [f] = BS_theorie(S,K,r,sigma,t,T)
52 if (t==T)
53 f=max(S-K,0);
54 else
55 f=S*N(d1(S,K,r,sigma,t,T))-K*exp(-r*(T-t))*N(d2(S,K,r,sigma,t,T))
56 end
57 end
58
59 function [f] = d1(S,K,r,sigma,t,T)
60 f = (log(S/K)+(r+(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
61 end
62
63 function [f] = d2(S,K,r,sigma,t,T)
64 f = (log(S/K)+(r-(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
65 end
66
67 function [f] = N(x)
68 f = 1/2*(1+erf(x/sqrt(2)));
69 end
```

FIGURE 40 – Code Sous MATLAB du fichier Code20.m

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab de la surface de la solution analytique en utilisant (CEV) :

```
4  function [] = visualiser_V_theorique_CEV()
5
6      % Définition des constantes %
7
8      L=20;
9      K=10;
10     T=0.5;
11     r=0.1;
12     N=99;
13     M=4999;
14
15     % Définition des vecteurs %
16
17     S=linspace(0,L,N+2);
18     t=linspace(0,T,M+2);
19     V=zeros(M+2,N+2);
20     sigma=zeros(M+2,N+2);
21
22     % Volatilité locale %
23
24     for p=1:M+2
25         for k=1:N+2
26             sigma(p,k)=(1/sqrt(S(k)))*exp(-t(p)/T);
27         end
28     end
29
30     % Solution analytique %
31
32     for n=1:M+2
33         for i=1:N+2
34             if i==1
35                 V(n,i)=0;
36             else
37                 V(n,i)=BS_theorie(S(i),K,r,sigma(n,i),t(n),T);
38             end
39         end
40     end
41
42     figure;
43     mesh(S,t,V)
44     xlabel('Prix de l actif S')
45     ylabel('temps t')
46     zlabel('Prix de l option V')
47     title('Solution analytique de l équation de Black et Scholes CEV')
48
49 end
50
51 function [f] = BS_theorie(S,K,r,sigma,t,T)
52 if (t==T)
53     f=max(S-K, 0);
54 else
55     f=S*N(d1(S,K,r,sigma,t,T))-K*exp(-r*(T-t))*N(d2(S,K,r,sigma,t,T))
56 end
57 end
58
59 function [f] = d1(S,K,r,sigma,t,T)
60     f = (log(S/K)+(r+(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
61 end
62
63 function [f] = d2(S,K,r,sigma,t,T)
64     f = (log(S/K)+(r-(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
65 end
66
67 function [f] = N(x)
68     f = 1/2*(1+erf(x/sqrt(2)));
69 end
```

FIGURE 41 – Code Sous MATLAB du fichier Code19.m

## III TP2 : Résolution numérique de l'équation de Black et Scholes par les méthodes de Monte-Carlo. Réduction de la variance.

Le but de ce TP 2 est la simulation numérique du prix de l'option Vanilla par la méthode de Monte-Carlo.

### 1 Partie I : Théorie

L'évolution de l'actif sous-jacent  $S$  est un processus stochastique  $S(t)$ ,  $t \in [0, T]$  qui vérifie l'équation différentielle stochastique suivante :

$$dS_t = rS_t dt + \sigma S_t dW_t \text{ avec } S(t=0) = S_0$$

On note  $\sigma$  la volatilité de l'actif,  $r$  le taux d'intérêt. La solution de cette équation est donnée par la formule :

On note  $\sigma$  la volatilité de l'action,  $r$  le taux d'intérêt. La solution de cette équation est donnée par la formule :

$$S(t) = S_0 \exp [(r - \frac{\sigma^2}{2})t + \sigma W(t)]$$

Le prix d'une option européenne au moment de temps  $t = 0$  est donnée par :

$$V(S_0, t) = e^{-r(T-t)} \mathbb{E}[\max(S(T) - K, 0) | S_t = S_0]$$

On utilise l'expression pour  $S(T)$  et on obtient :

$$V(S_0, t) = e^{-r(T-t)} \mathbb{E}[\max(S_0 \exp [(r - \frac{\sigma^2}{2})T + \sigma W(T)] - K, 0) | S_t = S_0]$$

$W(T)$  est la valeur du mouvement Brownien à l'instant  $t = T$ . C'est une variable aléatoire qui suit la loi  $N(0, T)$ .

On peut donc modéliser la valeur finale du mouvement Brownien par la variable aléatoire :  $W(T) = \sqrt{T}N(0, 1)$ .

En conclusion le prix de l'option européenne au moment  $t = 0$  au point  $S = S_0$  est donnée par la moyenne arithmétique :

$$V(S_0, t)_{estim} = e^{-r(T-t)} \sum_{n=1}^{N_{mc}} (\max(S_0 \exp [(r - \frac{\sigma^2}{2})T + \sigma \sqrt{T}N^{(n)}(0, 1)] - K, 0)) / N_{mc}$$

# TP: Méthodes numériques avancées appliquées à la finance

## 2 Partie II : Implémentation

→ Calculer le prix du call pour  $S_0 = 10$  avec la méthode de Monte-carlo :

Le prix du Call pour  $S_0 = 10$  par Monte-Carlo pour  $N_{mc} = 10000$  est : 1.616673.

D'après le TP1 si on compare le prix du Call pour  $S_0 = 10$  par Monte-Carlo pour  $N_{mc} = 10000$  et par Différences Finies, on trouve un résultat très proche, on deduit alors que la simulation de Monte-carlo est validé.

```
>> Code21
Le prix du Call pour s0=10 par Monte-Carlo est: 1.654695
```

FIGURE 42 – Le prix du Call pour  $S_0 = 10$  par Monte-Carlo pour  $N_{mc} = 10000$

Le code sous Matlab du prix du call pour  $S_0 = 10$  par Monte-Carlo pour  $N_{mc} = 10000$  :

```
1 % Prix du Call pour S0=10 par Monte-Carlo %
2
3 - s0=10;
4
5 - fprintf("Le prix du Call pour S0=10 par Monte-Carlo est: %f\n",Prix_Europ_S0_fixe_MC(S0));
6
7 - function [prix] = Prix_Europ_S0_fixe_MC(S0)
8
9 - % Définition des constantes %
10
11 - K=10;
12 - T=0.5;
13 - r=0.1;
14 - sigma=0.5;
15 - Nmc=10000;
16 - sum=0;
17
18 - % Methode Monte-Carlo pour calculer S %
19
20 - for n=1:Nmc
21 -     S(n)=S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*randn);
22 -     sum=sum+Payoff_Europ_Call(S(n),K);
23 - end
24
25 - prix=exp(-r*T)*sum/Nmc;
26 - end
27
28 - % la condition finale %
29
30 - function [f] = Payoff_Europ_Call(S,K)
31 - f=max(S-K,0);
32 - end
```

FIGURE 43 – Code Sous MATLAB du fichier Code21.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Tracer le graphe du prix du Call de l'option Européenne :  $S_0 \rightarrow V(t = 0, S_0)$  pour  $N_{mc} = 100$  et  $N_{mc} = 1000$  et comparaison avec la méthode aux Diférences Finies :

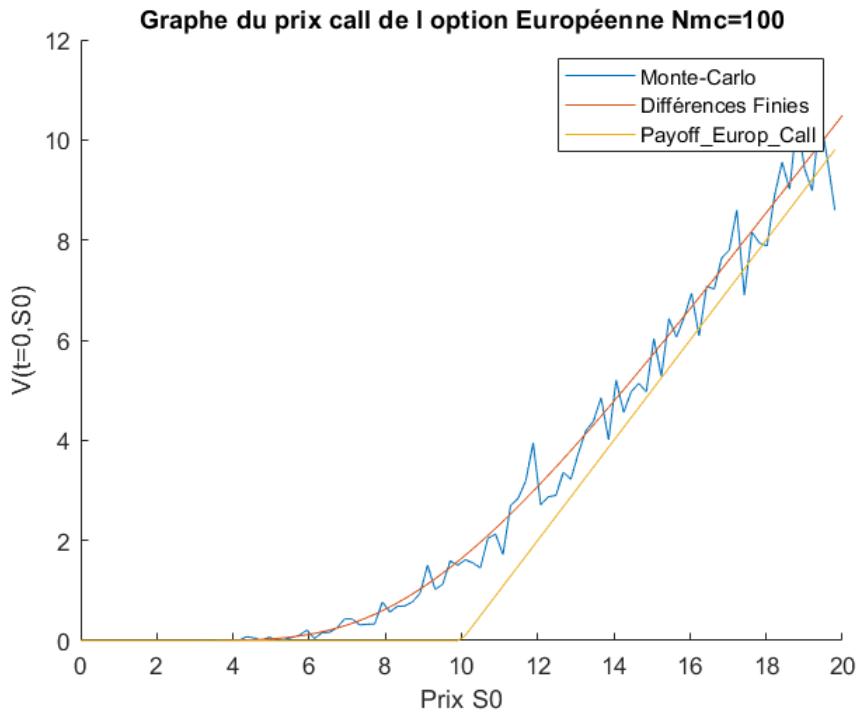


FIGURE 44 – Graphe du prix du Call pour  $S_0$  quelconque par MC pour  $N_{mc} = 100$  et par DF

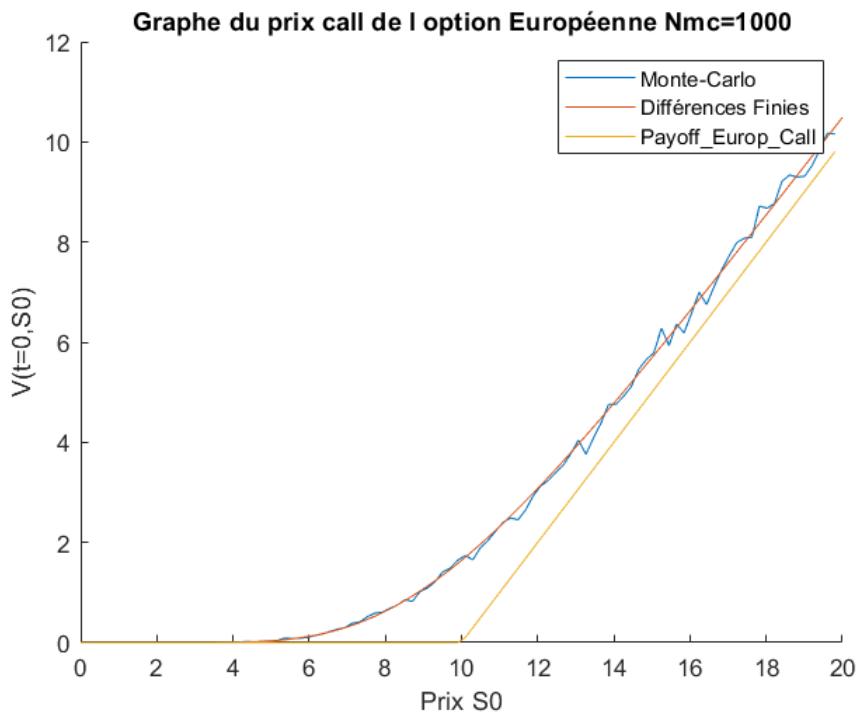


FIGURE 45 – Graphe du prix du Call pour  $S_0$  quelconque par MC pour  $N_{mc} = 1000$  et par DF

# TP: Méthodes numériques avancées appliquées à la finance

Le code sous Matlab du graphe du prix du Call pour  $S_0$  quelconque par Monte-carlo pour  $N_{mc} = 100$  (Code22.m) et  $N_{mc} = 1000$  (Code23.m) et par méthodes aux Différences Finies :

```
4 % function [] = Graphe_Call_Europ()
5 % Définition des constantes %
6 K=10;
7 L=20;
8 T=0.5;
9 r=0.1;
10 sigma=0.5;
11 N=99;
12 M=4999;
13 %
14 % Définition des vecteurs %
15 S=linspace(0,L,N+2);
16 t=linspace(0,T,M+2);
17 dt=T/(M+2);
18 ds=L/(N+2);
19 V=zeros(M+2,N+2);
20 %
21 % Implementation de la condition finale %
22 %
23 %
24 %
25 for j=1:N+2
26     V(M+2,j)=Payoff_Europ_Call(S(j),K);
27 end
28 %
29 % Implementation des conditions aux limites Dirichlet %
30 %
31 for k=1:M+1
32     V(k,1)=0;
33     V(k,N+2)=L-K*exp(-r*(T-t(k)));
34 end
35 %
36 %
37 % Discrétisation de l'équation de Black et Scholes %
38 %
39 for n=M+2:-1:2
40     for i=2:N+1
41         V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
42     end
43 end
44 %
45 % Methode Monte-Carlo pour calculer S quelconque %
46 %
47 for l=1:N+2
48     S0(l)=(l-1)*(L/(N+2));
49     prix_option(l)=Prix_Europ_S0_fixe_MC(S0(l));
50 end
51 %
52 figure;
53 hold;
54 plot(S0,prix_option)
55 plot(S,V(1,:));
56 plot(S0,Payoff_Europ_Call(S0,K))
57 xlabel('Prix S0')
58 ylabel('V(t=0,S0)')
59 legend('Monte-Carlo','Différences Finies','Payoff_Europ_Call')
60 title('Graphe du prix call de l option Européenne Nmc=1000')
61 %
62 end
63 %
64 function [prix] = Prix_Europ_S0_fixe_MC(S0)
65 %
66 % Définition des constantes %
67 K=10;
68 T=0.5;
69 r=0.1;
70 sigma=0.5;
71 Nmc=1000;
72 sum=0;
73 %
74 % Methode Monte-Carlo pour calculer S %
75 %
76 for n=1:Nmc
77     S(n)=S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*randn);
78     sum=sum+Payoff_Europ_Call(S(n),K);
79 end
80 %
81 prix=exp(-r*T)*sum/Nmc;
82 %
83 %
84 % la condition finale %
85 %
86 function [f] = Payoff_Europ_Call(S,K)
87 f=max(S-K,0);
88 end
```

FIGURE 46 – Code Sous MATLAB du fichier Code23.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Calculer le prix de l'option de Butterfly par Monte-Carlo à  $t=0$  :

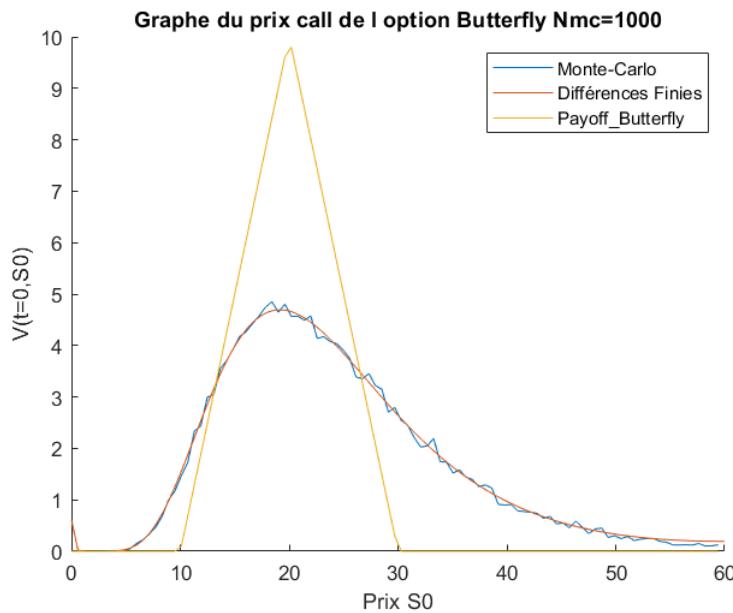


FIGURE 47 – Graphe du prix de l'option Butterfly pour  $S_0$  quelconque par MC pour  $N_{mc} = 1000$  et par DF

On remarque qu'on obtient un prix par Monte-Carlo pour  $N_{mc} = 1000$  très proche du prix par Méthodes aux Différences Finies. Par exemple pour  $N_{mc} = 100000$  on retrouve que les courbes sont identiques :

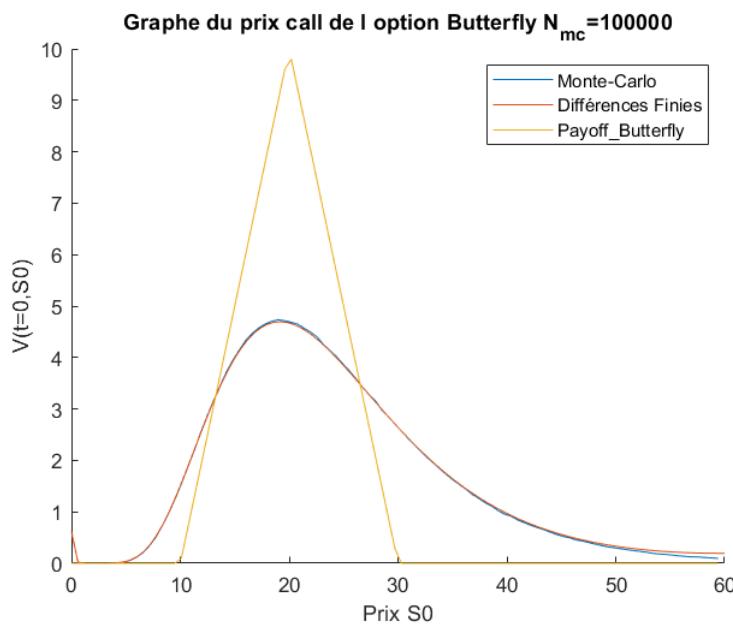


FIGURE 48 – Graphe du prix de l'option Butterfly pour  $S_0$  quelconque par MC pour  $N_{mc} = 100000$  et par DF

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab du graphe du prix de l'option Butterfly pour  $S_0$  quelconque par Monte-carlo pour  $N_{mc} = 1000$  (Code24.m) ,  $N_{mc} = 100000$  (Code25.m) et par méthodes aux Différences Finies :

```
4 %function [] = Graphe_MC_DF_Butterfly()
5 %
6 % Définition des constantes %
7
8 K=10;
9 L=60;
10 T=0.5;
11 r=0.1;
12 sigma=0.5;
13 N=99;
14 M=4999;
15
16 % Définition des vecteurs %
17
18 S=linspace(0,L,N+2);
19 t=linspace(0,T,M+2);
20 dt=T/(M+2);
21 ds=L/(N+2);
22 V=zeros(M+2,N+2);
23
24 % Implementation de la condition finale %
25
26 for j=1:N+2
27     V(M+2,j)=PayOff_Butterfly(S(j),K);
28 end
29
30 % Discrétisation de l'équation de Black et Scholes %
31
32 for n=M+2:-1:2
33     for i=2:N+1
34         V(n-1,i)=V(n,i)+dt*(r*S(i)*(V(n,i+1)-V(n,i-1))/(2*ds)+(1/2)*(sigma^2)*(S(i)^2)*((V(n,i+1)+V(n,i-1)-2*V(n,i))/(ds^2))-r*V(n,i));
35     end
36
37 % Implementation des conditions aux limites de Neumann %
38
39 V(n-1,1)=V(n-1,2)+ds;
40 V(n-1,N+2)=V(n-1,N+1);
41
42 % Methode Monte-Carlo pour calculer S quelconque %
43
44 for l=1:N+2
45     S0(l)=(l-1)*(L/(N+2));
46     prix_option(l)=Prix_Europ_S0_fixe_MC(S0(l));
47 end
48
49 figure;
50 hold;
51 plot(S0,prix_option)
52 plot(S,V(1,:));
53 plot(S0,Payoff_Butterfly(S0,K))
54 xlabel('Prix S0')
55 ylabel('V(t=0,S0)')
56 legend('Monte-Carlo','Différences Finies','Payoff\Butterfly')
57 title('Graphe du prix call de l option Butterfly Nmc=1000')
58
59 end
60
61 function [prix] = Prix_Europ_S0_fixe_MC(S0)
62 %
63 % Définition des constantes %
64
65 K=10;
66 T=0.5;
67 r=0.1;
68 sigma=0.5;
69 Nmc=1000;
70 sum=0;
71
72 % Methode Monte-Carlo pour calculer S %
73
74 for n=1:Nmc
75     S(n)=S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*randn);
76     sum=sum+PayOff_Butterfly(S(n),K);
77 end
78
79
80 prix=exp(-r*T)*sum/Nmc;
81
82
83 % la condition finale %
84
85 function [f] = PayOff_Butterfly(s,K)
86 f = max(s-K,0)+max(s-3*K,0)-2*max(s-2*K,0);
87 end
```

FIGURE 49 – Code Sous MATLAB du fichier Code25.m

### 3 Partie III : Le prix de l'option Européene pour une valeur $S_t$ et un temps t quelconques

→ Tracer en 2 dimension le graphe  $S_t \rightarrow V(t = T/2, S_t)$

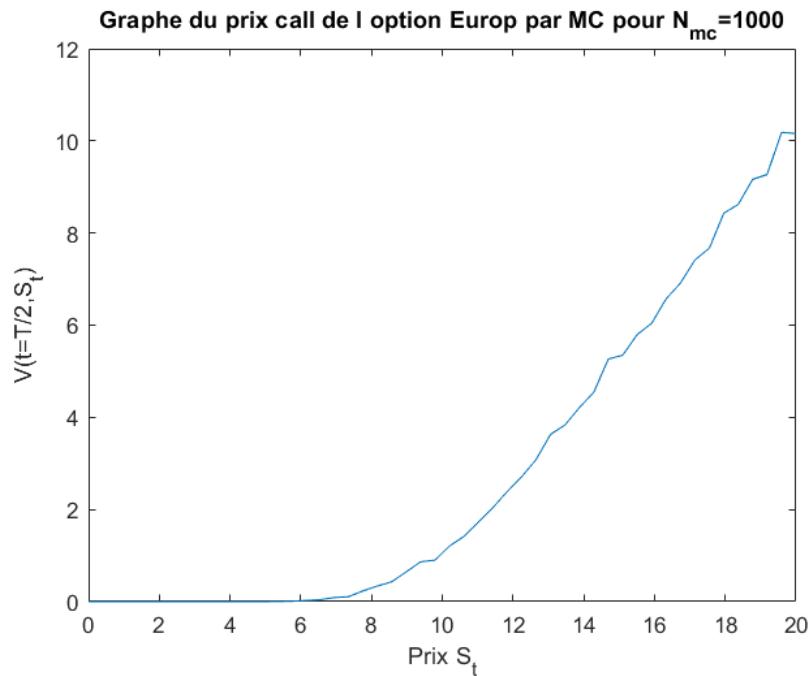


FIGURE 50 – Graphe du prix Call de l'option Europ pour  $S_t$  par MC pour  $N_{mc} = 1000$

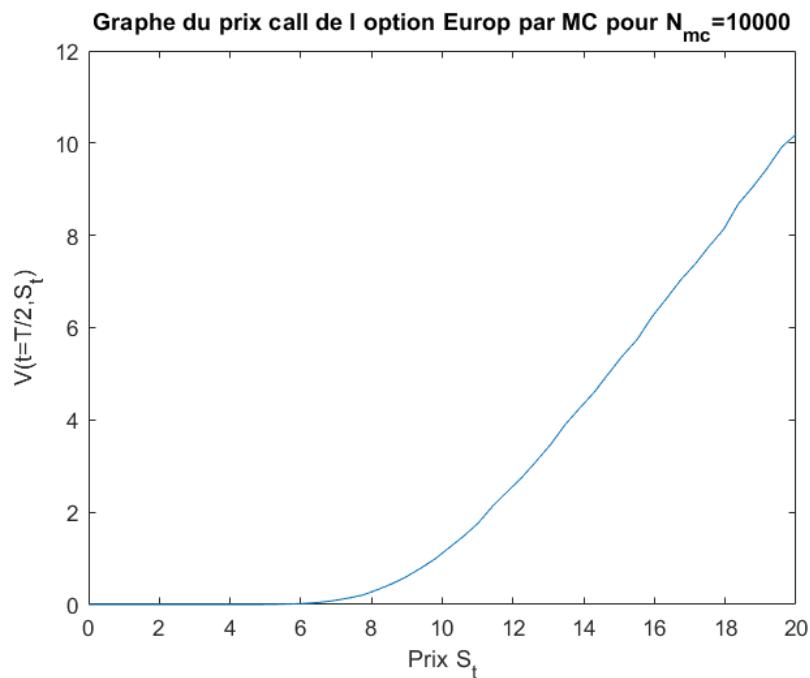


FIGURE 51 – Graphe du prix Call de l'option Europ pour  $S_t$  par MC pour  $N_{mc} = 10000$

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab du graphe du prix Call de l'option Europe pour  $S_t$  quelconque par Monte-carlo pour  $N_{mc} = 1000$  :

```
2 -     Graphe_Call()
3
4 - function [] = Graphe_Call()
5
6     % Définition des constantes et vecteurs %
7
8 - L=20;
9 - T=0.5;
10 - Ns=49;
11 - t=T/2;
12
13 - for j=1:(Ns+1)
14 -     St(j)=(L/Ns)*(j-1);
15 -     prix_graphe(j)=Prix_Call_St_fixe_t_fixe(t,st(j));
16 - end
17
18 - figure;
19 - plot(St,prix_graphe)
20 - xlabel('Prix S_t')
21 - ylabel('V(t=T/2,S_t)')
22 - title('Graphe du prix call de l option Europ par MC pour N_mc=1000')
23
24 - end
25
26 - function [prix] = Prix_Call_St_fixe_t_fixe(t,St)
27
28     % Définition des constantes et vecteurs %
29
30 - K=10;
31 - T=0.5;
32 - r=0.1;
33 - sigma=0.5;
34 - Nmc=1000;
35
36 - for n=1:Nmc
37 -     ST(n)=St*exp((r-(sigma^2)/2)*(T-t)+sigma*sqrt(T-t)*randn);
38 -     gain(n)=Payoff_Europ_Call(ST(n),K);
39 - end
40
41 - prix=exp(-r*(T-t))*mean(gain);
42
43 - end
44
45 - function [f] = Payoff_Europ_Call(S,K)
46 - f=max(S-K,0);
47 - end
```

FIGURE 52 – Code Sous MATLAB du fichier Code26.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Tracer la surface de prix de l'option Call Europe

Pour  $N_s = 49$ ,  $N_t = 10$  on obtient le graphe suivant :

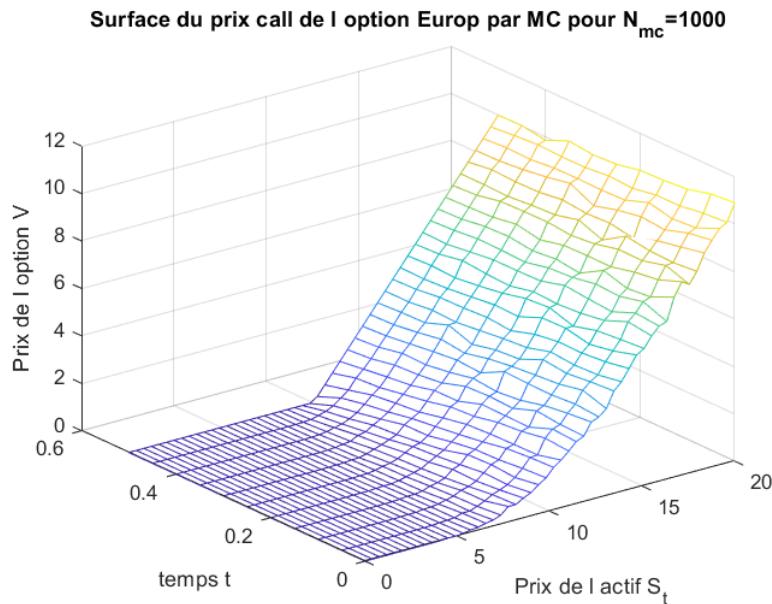


FIGURE 53 – Surface du prix Call de l'option Europe pour  $S_t$  par MC pour  $N_{mc} = 1000$

Pour  $N_s = 499$ ,  $N_t = 100$  on obtient le graphe suivant :

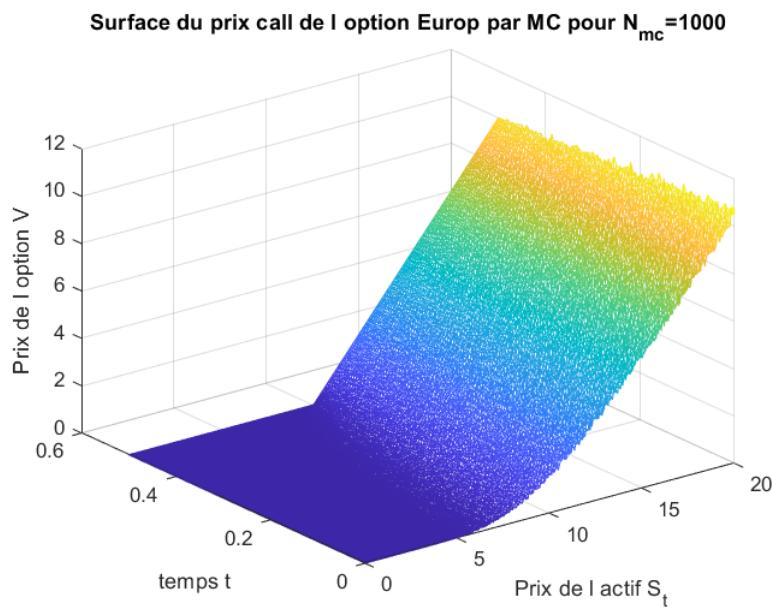


FIGURE 54 – Surface du prix Call de l'option Europe pour  $S_t$  par MC pour  $N_{mc} = 1000$

Comparaison avec la surface du prix Call Europe obtenu par les Méthodes aux différences finies : on remarque qu'on obtient une surface par Monte-carlo proche de celle obtenu par Différences Finies si on augmente  $N_s$  et  $N_t$  et quand le Nombre de Monte-carlo augmente.

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab de la surface du prix Call de l'option Europe pour  $S_t$  par MC pour  $N_{mc} = 1000$  ,  $N_s = 49$  et  $N_t = 10$  :

```
4  function [] = Surface_Call_Europe()
5
6      % Définition des constantes et vecteurs %
7
8      L=20;
9      T=0.5;
10     Ns=49;
11     Nt=10;
12
13     for j=1:(Ns+1)
14         for k=1:(Nt+1)
15             St(j)=(L/Ns)*(j-1);
16             t(k)=(T/Nt)*(k-1);
17             prix_graphe(k,j)=Prix_Call_St_fixe(t(k),St(j));
18         end
19     end
20
21     figure;
22     mesh(St,t,prix_graphe)
23     xlabel('Prix de l actif s_t')
24     ylabel('temps t')
25     zlabel('Prix de l option V')
26     title('Surface du prix call de l option Europ par MC pour N_mc=1000')
27
28     end
29
30     function [prix] = Prix_Call_St_fixe(t,St)
31
32     % Définition des constantes et vecteurs %
33
34     K=10;
35     T=0.5;
36     r=0.1;
37     sigma=0.5;
38     Nmc=1000;
39
40     for n=1:Nmc
41         ST(n)=St*exp((r-(sigma^2)/2)*(T-t)+sigma*sqrt(T-t)*randn);
42         gain(n)=Payoff_Europ_Call(ST(n),K);
43     end
44
45     prix=exp(-r*(T-t))*mean(gain);
46
47     end
48
49     function [f] = Payoff_Europ_Call(S,K)
50     f=max(S-K,0);
51     end
```

FIGURE 55 – Code Sous MATLAB du fichier Code27.m

# TP: Méthodes numériques avancées appliquées à la finance

---

## 4 Partie IV : Variance. Intervalle de confiance

Introduisons une fonction  $\Phi(W_T)$  qui représente un des prix actualisé du Call à la maturité.

$$\Phi(W_T) = e^{-rT} \max(S_0 \exp [(r - \frac{\sigma^2}{2})T + \sigma W(T)] - K, 0)$$

Le prix de l'option est donnée par la formule :

$$V(S_0, 0) = \mathbb{E}[\Phi(W_T)]$$

- Premier estimateur de l'option :

$$V(S_0, 0)_{\text{estime1}} = \frac{1}{N_{mc}} \sum_{n=1}^{N_{mc}} [\Phi(W_T^{(n)})]$$

- Calculons la variance de  $\Phi$ :

$$\text{Var}[\Phi] = \mathbb{E}[\Phi^2] - (\mathbb{E}[\Phi])^2$$

$$\text{Var}[\Phi]_{\text{estime}} = \frac{1}{N_{mc}} \sum_{n=1}^{N_{mc}} (\Phi(W_T^{(n)}))^2 - \left( \frac{1}{N_{mc}} \sum_{n=1}^{N_{mc}} \Phi(W_T^{(n)}) \right)^2.$$

- Le vrai prix du Call se trouve dans l'intervalle de confiance

$$V(S_0, 0) \in \left[ \sum_{n=1}^{N_{mc}} \Phi(W_T^{(n)}) / N_{mc} - \frac{1.96 \cdot \sqrt{\text{Var}[\Phi]}}{\sqrt{N_{mc}}}, \sum_{n=1}^{N_{mc}} \Phi(W_T^{(n)}) / N_{mc} + \frac{1.96 \cdot \sqrt{\text{Var}[\Phi]}}{\sqrt{N_{mc}}} \right]$$

avec la probabilité 0.95. Ici

$$\sum_{n=1}^{N_{mc}} \Phi(W_T^{(n)}) / N_{mc} = V(S_0, 0)_{\text{estime}}$$

est le prix du Call qu'on a estimé par la méthode de Monté-Carlo. On ne connaît pas la vraie variance donc on la remplace par la variance estimée.

- Le vrai prix du Call se trouve dans l'intervalle de confiance

$$V(S_0, 0) \in \left[ \sum_{n=1}^{N_{mc}} \Phi(W_T^{(n)}) / N_{mc} - \frac{1.96 \cdot \sqrt{\text{Var}[\Phi]_{\text{estime}}}}{\sqrt{N_{mc}}}, \sum_{n=1}^{N_{mc}} \Phi(W_T^{(n)}) / N_{mc} + \frac{1.96 \cdot \sqrt{\text{Var}[\Phi]_{\text{estime}}}}{\sqrt{N_{mc}}} \right]$$

avec la probabilité 0.95.

# TP: Méthodes numériques avancées appliquées à la finance

→ Calculez la variance de la fonction pay-off actualisée pour  $S_0 = 10$  ( $N_{mc} = 1000$ )

```
>> Code28
La variance de fonction Pay-Off actualisé pour s0=10 est: 7.576325
```

FIGURE 56 – La variance de la fonction pay-off actualisée pour  $S_0 = 10$  ( $N_{mc} = 1000$ )

Le code sous Matlab de la variance de la fonction pay-off actualisée pour  $S_0 = 10$  ( $N_{mc} = 1000$ ) :

```
2 - S0=10;
3
4 - fprintf("La variance de fonction Pay-Off actualisé pour s0=10 est: %f\n",Variance(S0));
5
6
7 - function [V_estime1] = Variance(S0)
8
9 - % Définition des constantes %
10-
11- K=10;
12- T=0.5;
13- r=0.1;
14- sigma=0.5;
15- Nmc=1000;
16- esperance=0;
17- esperance_carre=0;
18-
19- for n=1:Nmc
20-     g=randn;
21-     esperance=esperance+Phi (K,r,T,sigma,S0,g);
22-     esperance_carre=esperance_carre+Phi (K,r,T,sigma,S0,g)^2;
23- end
24-
25- V_estime1=(esperance_carre/Nmc)-(esperance/Nmc)^2;
26-
27- end
28
29 - function [f] = Phi(K,r,T,sigma,S0,g)
30-
31- f=exp(-r*T)*max(S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*g)-K,0);
32-
33- end
```

FIGURE 57 – Code Sous MATLAB du fichier Code28.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Calculer l'intervalle de confiance pour Call pour  $S_0 = 10$  ( $N_{mc} = 1000$ )

```
>> Code29
Intervalle de confiance pour Call à S0=10 est: [1.491466,1.805958]
Le prix de l option estimé pour Call à S0=10 est: 1.648712 dans [1.491466,1.805958]
```

FIGURE 58 – L'intervalle de confiance pour Call pour  $S_0 = 10$  ( $N_{mc} = 1000$ )

Le code sous Matlab de l'intervalle de confiance pour Call pour  $S_0 = 10$  ( $N_{mc} = 1000$ ) :

```
3 -     S0=10;
4 -     Intervalle_de_confiance(S0)
5 -
6 - function [] = Intervalle_de_confiance(S0)
7 -
8 - % Définition des constantes %
9 -
10 - Nmc=1000;
11 - [prix,V_estime1] = Variance(S0);
12 - a= prix-1.96*sqrt(V_estime1/Nmc);
13 - b= prix+1.96*sqrt(V_estime1/Nmc);
14 -
15 - % Afficher l'intervalle de confiance avec probabilité 95% %
16 -
17 - fprintf("Intervalle de confiance pour Call à S0=10 est: [%f,%f]\n",a,b);
18 - fprintf("Le prix de l option estimé pour Call à S0=10 est: %f dans [%f,%f]\n",prix,a,b);
19 -
20 - end
21
22
23 function [prix,V_estime1] = Variance(S0)
24
25 % Définition des constantes %
26
27 K=10;
28 T=0.5;
29 r=0.1;
30 sigma=0.5;
31 Nmc=1000;
32 esperance=0;
33 esperance_carre=0;
34
35 for n=1:Nmc
36     g=randn;
37     esperance=esperance+Phi(K,r,T,sigma,S0,g);
38     esperance_carre=esperance_carre+Phi(K,r,T,sigma,S0,g)^2;
39 end
40
41 V_estime1=(esperance_carre/Nmc)-(esperance/Nmc)^2;
42 prix=esperance/Nmc;
43
44 end
45
46 function [f] = Phi(K,r,T,sigma,S0,g)
47
48 f=exp(-r*T)*max(S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*g)-K,0);
49
50 end
```

FIGURE 59 – Code Sous MATLAB du fichier Code29.m

## 5 Partie V : Réduction de la variance. Variables Antithétiques.

Pour diminuer l'intervalle de confiance soit on augmente le nombre  $N_{mc}$  soit on diminue la variance  $Var[\Phi]$ .

L'idée du contrôle antithétique est très simple. Elle est basée sur la propriété de symétrie du mouvement brouwnien  $W_t$  et  $-W_t$ . Donc pour une fonction  $F$  on a

$$\mathbb{E}\left[\frac{F(W_t) + F(-W_t)}{2}\right] = \mathbb{E}[F(W_t)].$$

On utilise l'identité

$$Var\left[\frac{F(W_t) + F(-W_t)}{2}\right] = \frac{1}{4}(Var[F(W_t)] + Var[F(-W_t)] + 2Cov(F(W_t), F(-W_t)))$$

on obtient

$$Var\left[\frac{F(W_t) + F(-W_t)}{2}\right] = \frac{1}{2}(Var[F(W_t)] + Cov(F(W_t), F(-W_t)))$$

On peut montrer que si  $F : x \rightarrow F(x)$  et  $G : x \rightarrow G(x)$  sont monotones (toutes les deux croissantes ou toutes les deux décroissantes) alors

$$Cov(F(W_t), G(W_t)) > 0.$$

### Réduction de la variance. Application à l'évaluation du prix d'une Option

Dans notre cas prenons:

$$F(W_t) = \Phi(W_T), \quad G(W_t) = -\Phi(-W_T)$$

Donc

$$Cov(\Phi(W_T), \Phi(-W_T)) < 0$$

$$Var\left[\frac{\Phi(W_T) + \Phi(-W_T)}{2}\right] <= \frac{1}{2}Var[\Phi(W_T)]$$

et on a réduit la variance de  $\Phi(W_T)$ .

- Nous avons montré que

$$V(S_0, 0) = \mathbb{E}[\Phi(W_T)] = \mathbb{E}[\Phi(-W_T)]$$

On peut donc présenter le prix de l'option par la formule:

$$V(S_0, 0) = \mathbb{E}\left[\frac{\Phi(W_T) + \Phi(-W_T)}{2}\right]$$

### • Deuxième estimateur de l'option :

$$V(S_0, 0)_{estime2} = \frac{1}{2N_{mc}} \sum_{n=1}^{N_{mc}} [\Phi(W_T^{(n)}) + \Phi(-W_T^{(n)})]$$

# TP: Méthodes numériques avancées appliquées à la finance

→ Calculer le prix pour  $S_0 = 10$  ( $N_{mc} = 1000$ )

```
>> Code31
Le prix pour S0=10 est: 1.645802
>>
```

FIGURE 60 – le prix pour  $S_0 = 10$  ( $N_{mc} = 1000$ )

On a trouvé grâce au deuxième estimateur un prix pour  $S_0 = 10$  et ( $N_{mc} = 1000$ ) la valeur 1.645802 tandis qu' avec le premier estimateur pour  $S_0 = 10$  et ( $N_{mc} = 1000$ ) on a trouver un prix de 1.648712 (cf page 41). On deduit qu'on obtient presque le même prix avec les deux estimateurs.

Le code sous Matlab du prix pour  $S_0 = 10$  ( $N_{mc} = 1000$ ) :

```
2 -     S0=10;
3 -
4 -     [prix,V_estime2] = Variance_2(S0);
5 -
6 -     fprintf("Le prix pour S0=10 est: %f\n",prix);
7 -
8 -
9 - function [prix,V_estime2] = Variance_2(S0)
10 -
11    % Définition des constantes %
12 -
13    K=10;
14    T=0.5;
15    r=0.1;
16    sigma=0.5;
17    Nmc=1000;
18    esperance=0;
19    esperance_carre=0;
20 -
21 - for n=1:Nmc
22 -     g=randn;
23 -     esperance=esperance+((Phi(K,r,T,sigma,S0,g)+Phi_Sym(K,r,T,sigma,S0,g))/2);
24 -     esperance_carre=esperance_carre+((Phi(K,r,T,sigma,S0,g)+Phi_Sym(K,r,T,sigma,S0,g))/2)^2;
25 - end
26 -
27 - V_estime2=(esperance_carre/Nmc)-(esperance/Nmc)^2;
28 - prix=esperance/Nmc;
29 -
30 - end
31 -
32 - function [f] = Phi(K,r,T,sigma,S0,g)
33 -
34 - f=exp(-r*T)*max(S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*g)-K,0);
35 -
36 - end
37 -
38 - function [f] = Phi_Sym(K,r,T,sigma,S0,g)
39 -
40 - f=exp(-r*T)*max(S0*exp((r-(sigma^2)/2)*T-sigma*sqrt(T)*g)-K,0);
41 -
42 - end
```

FIGURE 61 – Code Sous MATLAB du fichier Code31.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Calculer la variance de  $\frac{1}{2}(\Phi(W_t) + \Phi(-W_t))$

```
>> Code32
La variance à S0=10 est: 2.634489
Intervalle de confiance pour Call à S0=10 est: [1.543318,1.744521]
Le prix de 1 option estimé pour Call à S0=10 est: 1.643920 dans [1.543318,1.744521]
```

FIGURE 62 – Variance et intervalle de confiance estimateur 2 pour  $S_0 = 10$  ( $N_{mc} = 1000$ )

On remarque que l'intervalle de confiance est réduit et la variance du deuxième estimateur est plus petite que la variance du premier estimateur.

Comparaison de l'intervalle de confiance avec celui du premier estimateur :

→ Intervalle de confiance pour le premier estimateur (Code30.m) :

```
>> Code30
La variance à S0=10 est: 7.013225
Intervalle de confiance pour Call à S0=10 est: [1.476013,1.804293]
Le prix de 1 option estimé pour Call à S0=10 est: 1.640153 dans [1.476013,1.804293]
```

FIGURE 63 – Variance et intervalle de confiance estimateur 1 pour  $S_0 = 10$  ( $N_{mc} = 1000$ )

→ Intervalle de confiance pour le deuxième estimateur (Code32.m) :

```
>> Code32
La variance à S0=10 est: 2.634489
Intervalle de confiance pour Call à S0=10 est: [1.543318,1.744521]
Le prix de 1 option estimé pour Call à S0=10 est: 1.643920 dans [1.543318,1.744521]
```

FIGURE 64 – Variance et intervalle de confiance estimateur 2 pour  $S_0 = 10$  ( $N_{mc} = 1000$ )

# TP: Méthodes numériques avancées appliquées à la finance

Le code sous Matlab de la variance  $S_0 = 10$  ( $N_{mc} = 1000$ ) :

```
2 -     S0=10;
3 -     Intervalle_de_confiance_2(S0)
4 -
5 - function [] = Intervalle_de_confiance_2(S0)
6 -
7 % Définition des constantes %
8 -
9 - Nmc=1000;
10 - [prix,V_estime2] = Variance_2(S0);
11 - a= prix-1.96*sqrt(V_estime2/Nmc);
12 - b= prix+1.96*sqrt(V_estime2/Nmc);
13 -
14 % Afficher l'intervalle de confiance avec probabilité 95%
15 -
16 - fprintf("La variance à S0=10 est: %f\n",V_estime2);
17 - fprintf("Intervalle de confiance pour Call à S0=10 est: [%f,%f]\n",a,b);
18 - fprintf("Le prix de l'option estimé pour Call à S0=10 est: %f dans [%f,%f]\n",prix,a,b);
19 -
20 - end
21 -
22 -
23 function [prix,V_estime2] = Variance_2(S0)
24 -
25 % Définition des constantes %
26 -
27 - K=10;
28 - T=0.5;
29 - r=0.1;
30 - sigma=0.5;
31 - Nmc=1000;
32 - esperance=0;
33 - esperance_carre=0;
34 -
35 - for n=1:Nmc
36 -     g=randn;
37 -     esperance=esperance+((Phi(K,r,T,sigma,S0,g)+Phi_Sym(K,r,T,sigma,S0,g))/2);
38 -     esperance_carre=esperance_carre+((Phi(K,r,T,sigma,S0,g)+Phi_Sym(K,r,T,sigma,S0,g))/2)^2;
39 - end
40 -
41 - V_estime2=(esperance_carre/Nmc)-(esperance/Nmc)^2;
42 - prix=esperance/Nmc;
43 -
44 - end
45 -
46 function [f] = Phi(K,r,T,sigma,S0,g)
47 -
48 f=exp(-r*T)*max(S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*g)-K,0);
49 -
50 end
51 -
52 function [f] = Phi_Sym(K,r,T,sigma,S0,g)
53 -
54 f=exp(-r*T)*max(S0*exp((r-(sigma^2)/2)*T-sigma*sqrt(T)*g)-K,0);
55 -
56 end
```

FIGURE 65 – Code Sous MATLAB du fichier Code32.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Tracer le graphe  $S_0 \rightarrow V(S_0, 0)_{\text{estime1}}$

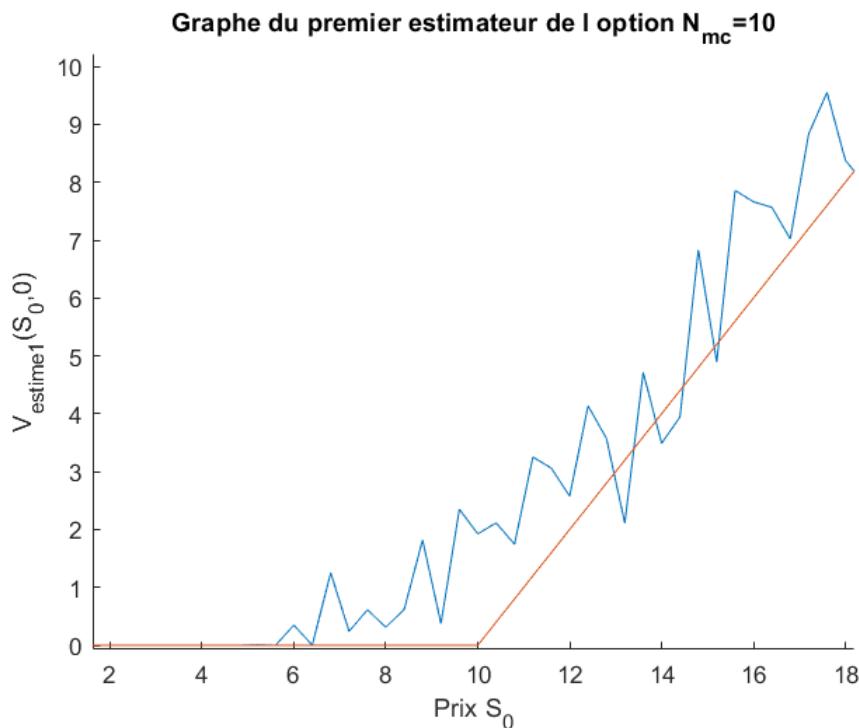


FIGURE 66 – Graphe de l'estimateur 1 pour ( $N_{mc} = 10$ )

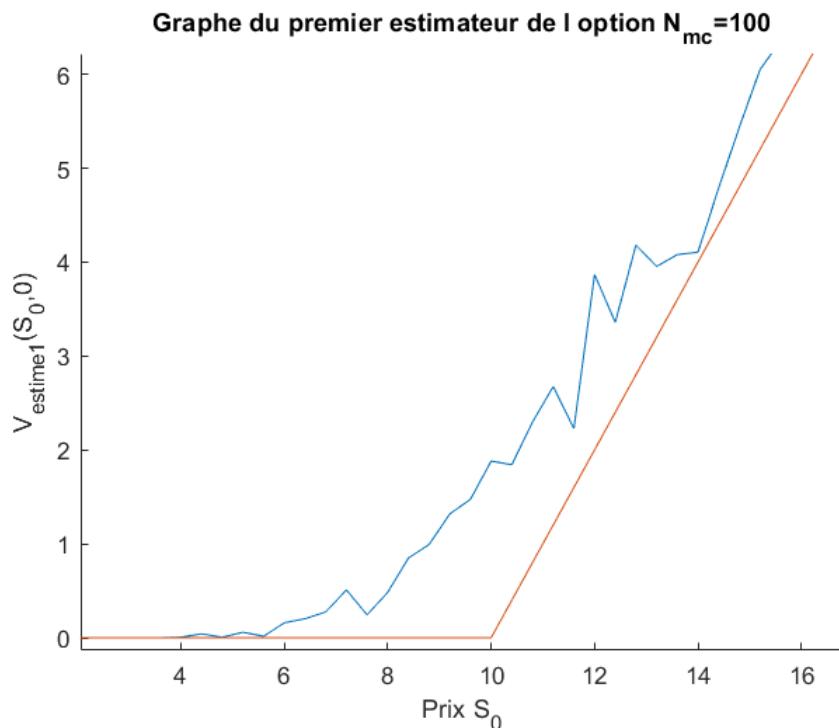


FIGURE 67 – Graphe de l'estimateur 1 pour ( $N_{mc} = 100$ )

# TP: Méthodes numériques avancées appliquées à la finance

---

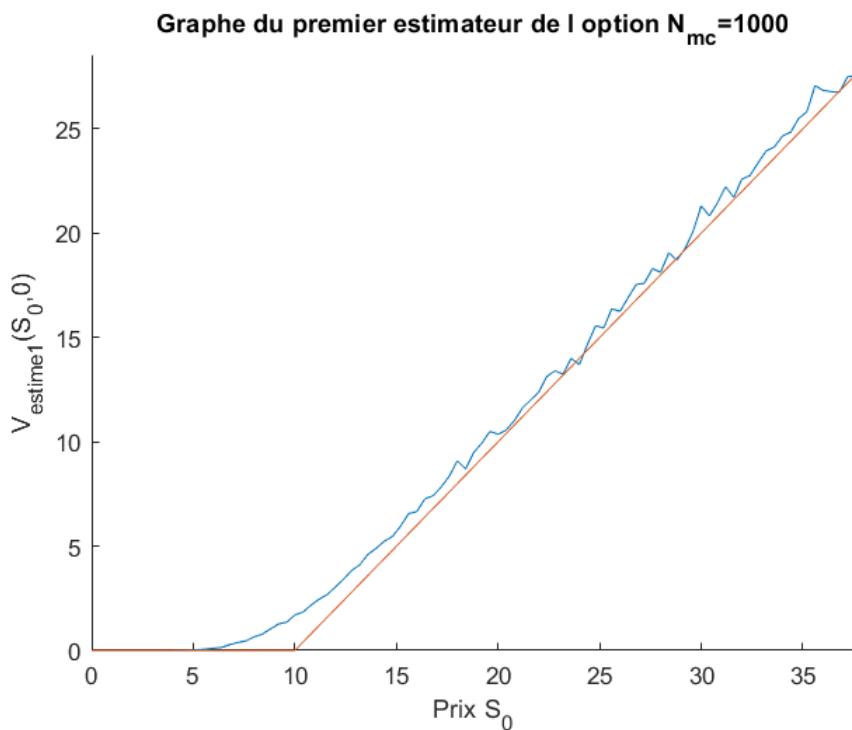


FIGURE 68 – Graphe de l'estimateur 1 pour ( $N_{mc} = 1000$ )

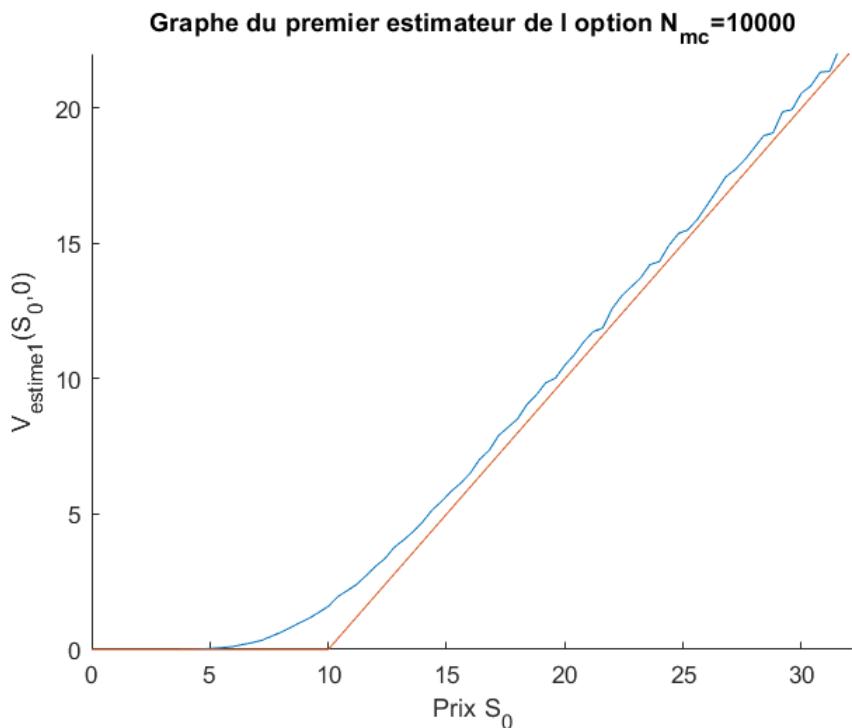


FIGURE 69 – Graphe de l'estimateur 1 pour ( $N_{mc} = 10000$ )

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab du graphe de l'estimateur 1 ( $N_{mc} = 1000$ ) :

```
1 - Nmc=10000;
2 -
3 - graph_V_estime1(Nmc)
4 -
5 - function [] = graph_V_estime1(Nmc)
6 -
7 - K=10;
8 -
9 - for j=1:500
10 -    S0(j)=0.4*(j-1);
11 -    prix_graphe(j)=V_estime1(S0(j),Nmc);
12 - end
13 -
14 - figure;
15 - hold;
16 - plot(S0,prix_graphe)
17 - plot(S0,max(S0-K,0))
18 - xlabel('Prix S_0')
19 - ylabel('V_{estime1}(S_0,0)')
20 - title('Graphe du premier estimateur de l option N_{mc}=10000')
21 -
22 - end
23 -
24 -
25 - function [prix] = V_estime1(S0,Nmc)
26 -
27 - % Définition des constantes %
28 -
29 - K=10;
30 - T=0.5;
31 - r=0.1;
32 - sigma=0.5;
33 - esperance=0;
34 -
35 - for n=1:Nmc
36 -    g=randn;
37 -    esperance=esperance+Phi(K,r,T,sigma,S0,g);
38 - end
39 -
40 - prix=esperance/Nmc;
41 -
42 - end
43 -
44 - function [f] = Phi(K,r,T,sigma,S0,g)
45 -
46 - f=exp(-r*T)*max(S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*g)-K,0);
47 -
48 - end
```

FIGURE 70 – Code Sous MATLAB du fichier Code34.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Tracer le graphe  $S_0 \rightarrow V(S_0, 0)_{estime2}$

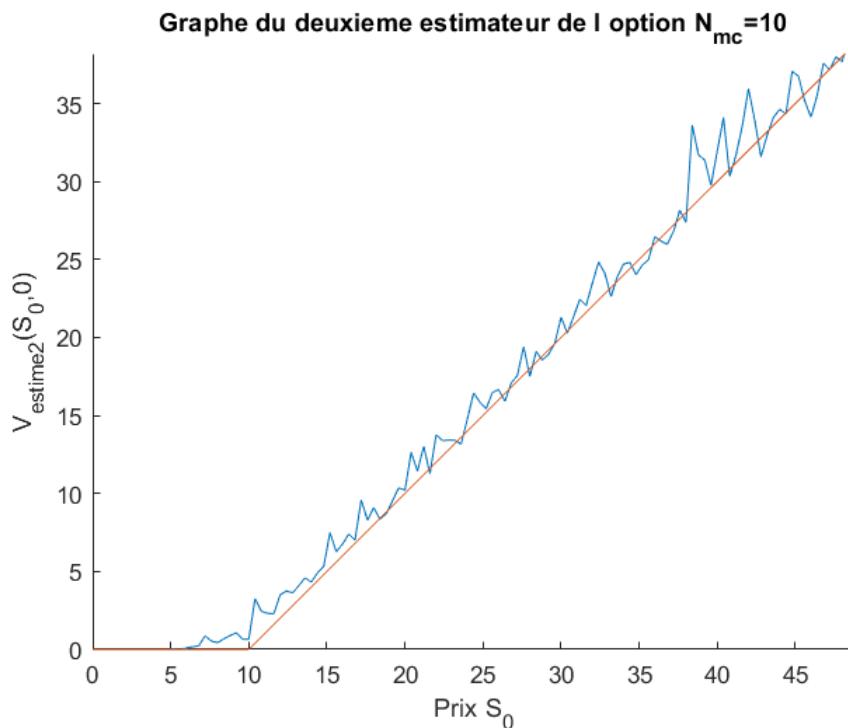


FIGURE 71 – Graphe de l'estimateur 2 pour ( $N_{mc} = 10$ )

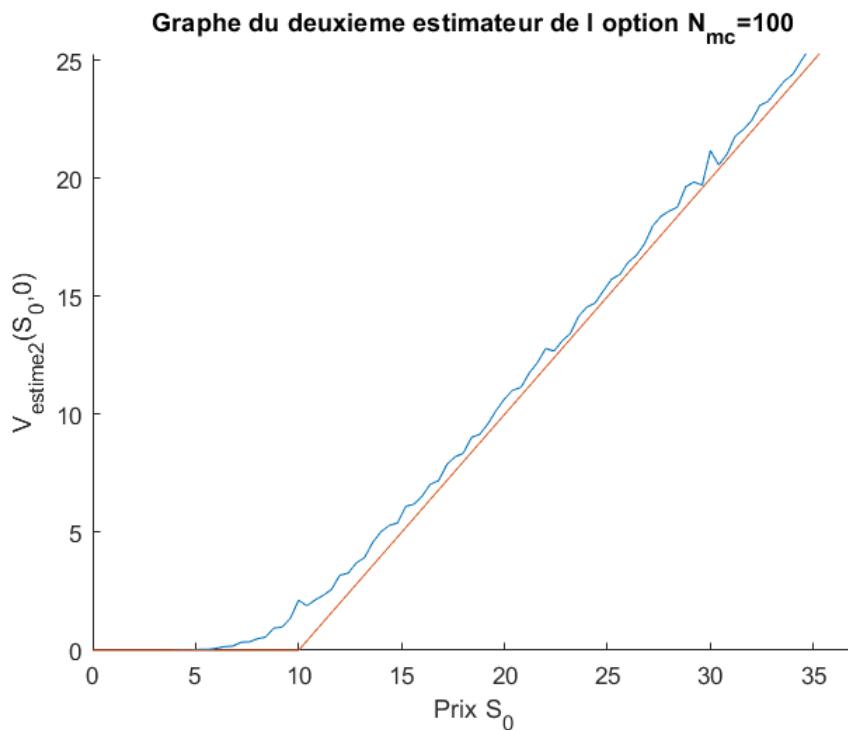


FIGURE 72 – Graphe de l'estimateur 2 pour ( $N_{mc} = 100$ )

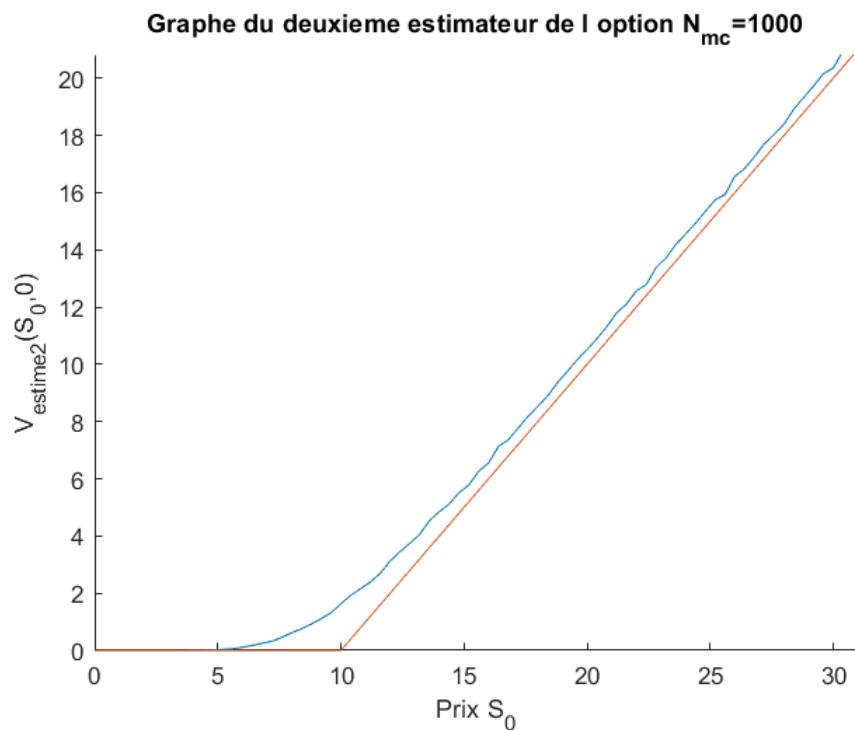


FIGURE 73 – Graphe de l'estimateur 2 pour ( $N_{mc} = 1000$ )

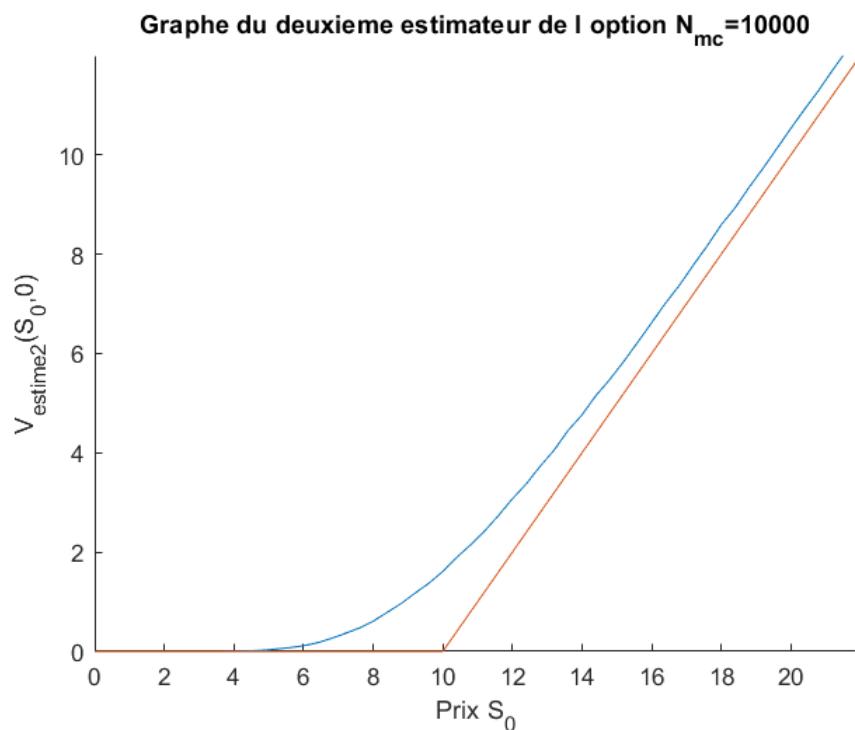


FIGURE 74 – Graphe de l'estimateur 2 pour ( $N_{mc} = 10000$ )

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab du graphe de l'estimateur 2 ( $N_{mc} = 1000$ ) :

```
1 - Nmc=10000;
2 -
3 - graphe_V_estime2(Nmc)
4 -
5 - function [] = graphe_V_estime2(Nmc)
6 -
7 - K=10;
8 -
9 - for j=1:500
10 -    S0(j)=0.4*(j-1);
11 -    prix_graphe(j)=V_estime2(S0(j),Nmc);
12 - end
13 -
14 - figure;
15 - hold;
16 - plot(S0,prix_graphe)
17 - plot(S0,max(S0-K,0))
18 - xlabel('Prix S_0')
19 - ylabel('V_{estime2}(S_0,0)')
20 - title('Graphe du deuxieme estimateur de l option N_{mc}=10000')
21 -
22 - end
23 -
24 -
25 - function [prix] = V_estime2(S0,Nmc)
26 -
27 - % Définition des constantes %
28 -
29 - K=10;
30 - T=0.5;
31 - r=0.1;
32 - sigma=0.5;
33 - esperance=0;
34 -
35 - for n=1:Nmc
36 -    g=randn;
37 -    esperance=esperance+((Phi(K,r,T,sigma,S0,g)+Phi_Sym(K,r,T,sigma,S0,g))/2);
38 - end
39 -
40 - prix=esperance/Nmc;
41 -
42 - end
43 -
44 -
45 - function [f] = Phi(K,r,T,sigma,S0,g)
46 -
47 - f=exp(-r*T)*max(S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*g)-K,0);
48 -
49 - end
50 -
51 - function [f] = Phi_Sym(K,r,T,sigma,S0,g)
52 -
53 - f=exp(-r*T)*max(S0*exp((r-(sigma^2)/2)*T-sigma*sqrt(T)*g)-K,0);
54 -
55 - end
```

FIGURE 75 – Code Sous MATLAB du fichier Code35.m

# TP: Méthodes numériques avancées appliquées à la finance

→ Tracer le graphe  $S_0 \rightarrow V(S_0, 0)_{\text{estime1}}$  et  $S_0 \rightarrow V(S_0, 0)_{\text{estime2}}$  dans le même graphe

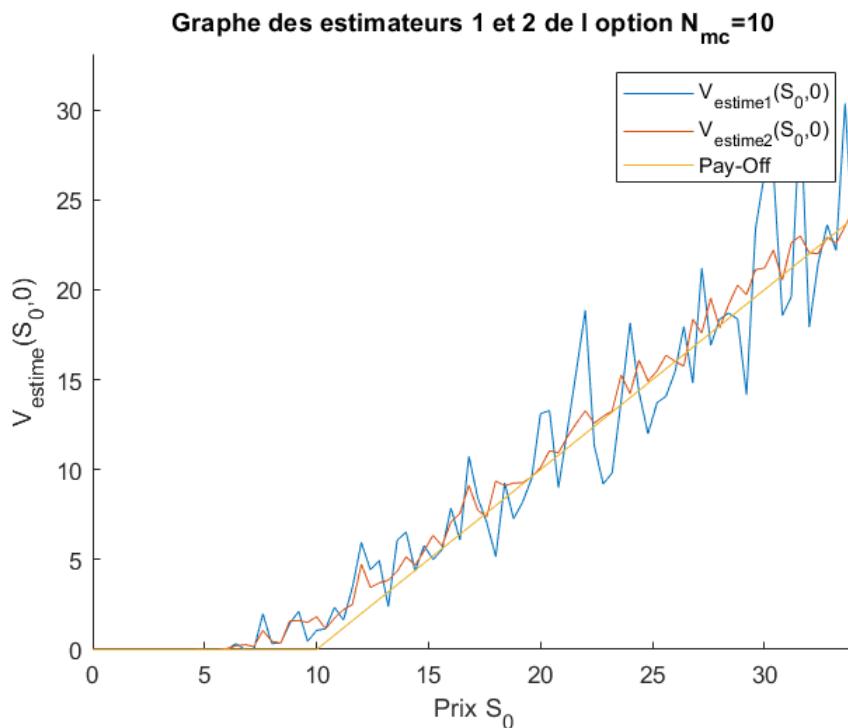


FIGURE 76 – Graphe des estimateurs pour ( $N_{mc} = 10$ )

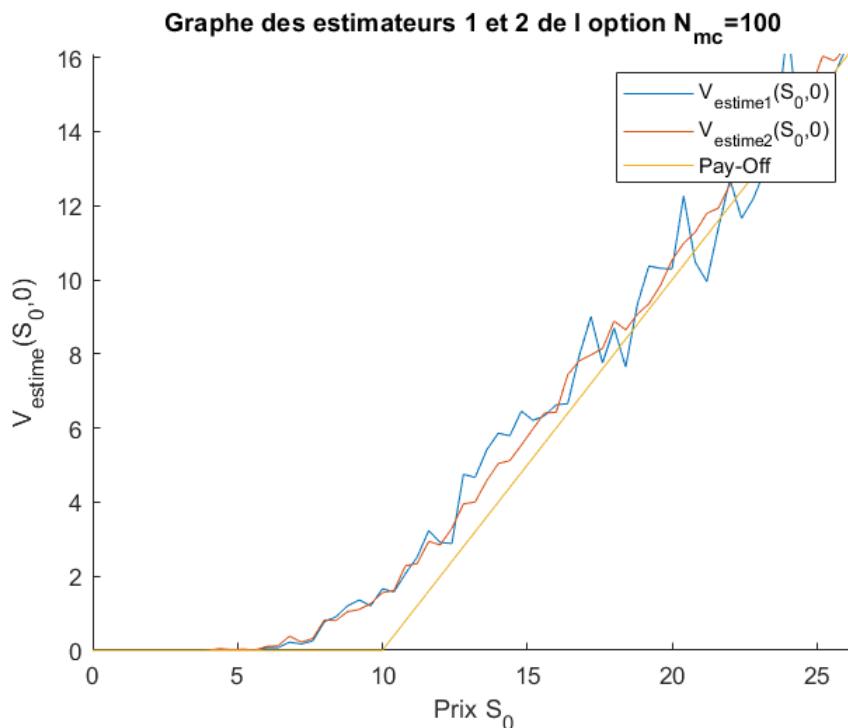


FIGURE 77 – Graphe des estimateurs pour ( $N_{mc} = 100$ )

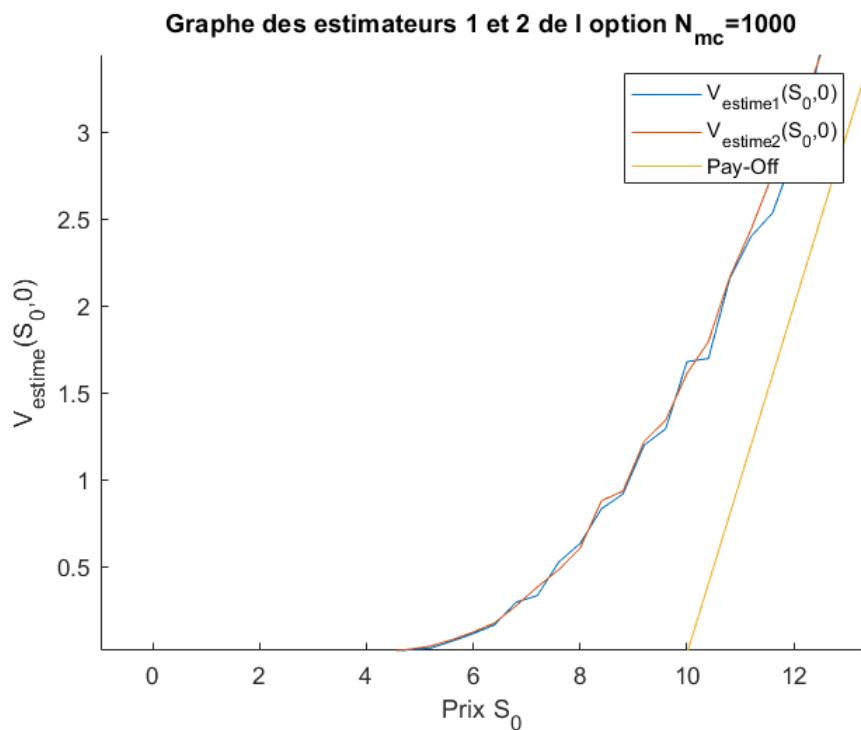


FIGURE 78 – Graphe des estimateurs pour ( $N_{mc} = 1000$ )

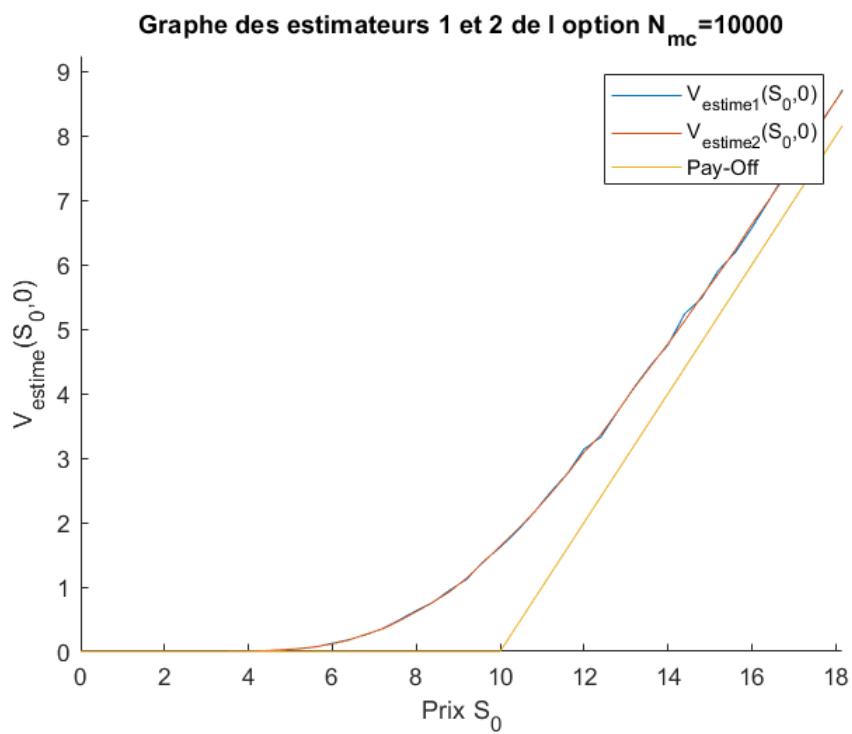


FIGURE 79 – Graphe des estimateurs pour ( $N_{mc} = 10000$ )

# TP: Méthodes numériques avancées appliquées à la finance

Le code sous Matlab du graphe des estimateurs ( $N_{mc} = 10000$ ) :

```
1 -      Nmc=10000;
2 -
3 -      graphe_V_estime(Nmc)
4 -
5 -      function [] = graphe_V_estime(Nmc)
6 -
7 -          K=10;
8 -
9 -          for j=1:500
10 -              S0(j)=0.4*(j-1);
11 -              [prix_graphe1(j),prix_graphe2(j)]=V_estime(S0(j),Nmc);
12 -          end
13 -
14 -          figure;
15 -          hold;
16 -          plot(S0,prix_graphe1)
17 -          plot(S0,prix_graphe2)
18 -          plot(S0,max(S0-K,0))
19 -          xlabel('Prix S_0')
20 -          ylabel('V_{estime}(S_0,0)')
21 -          legend('V_{estime1}(S_0,0)', 'V_{estime2}(S_0,0)', 'Pay-Off');
22 -          title('Graphe des estimateurs 1 et 2 de l option N_mc=10000')
23 -
24 -      end
25 -
26 -
27 -      function [prix1,prix2] = V_estime(S0,Nmc)
28 -
29 -          % Définition des constantes %
30 -
31 -          K=10;
32 -          T=0.5;
33 -          r=0.1;
34 -          sigma=0.5;
35 -          esperance1=0;
36 -          esperance2=0;
37 -
38 -          for n=1:Nmc
39 -              g=randn;
40 -              esperance1=esperance1+Phi(K,r,T,sigma,S0,g);
41 -              esperance2=esperance2+((Phi(K,r,T,sigma,S0,g)+Phi_Sym(K,r,T,sigma,S0,g))/2);
42 -          end
43 -
44 -          prix1=esperance1/Nmc;
45 -          prix2=esperance2/Nmc;
46 -
47 -      end
48 -
49 -
50 -      function [f] = Phi(K,r,T,sigma,S0,g)
51 -
52 -          f=exp(-r*T)*max(S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*g)-K,0);
53 -
54 -      end
55 -
56 -      function [f] = Phi_Sym(K,r,T,sigma,S0,g)
57 -
58 -          f=exp(-r*T)*max(S0*exp((r-(sigma^2)/2)*T-sigma*sqrt(T)*g)-K,0);
59 -
60 -      end
```

FIGURE 80 – Code Sous MATLAB du fichier Code36.m

## 6 Partie VI : Réduction de la variance. Variables de Contrôle.

### Idée des Variables de Control.

Au lieu de calculer l'espérance de la variable aléatoire  $X$  on va calculer l'espérance de la variable aléatoire

$$Z = X - b(Y - \mathbb{E}[Y])$$

Ici  $b$  est un nombre,  $Y$  est une variable aléatoire dont l'espérance est connue. On va montrer que  $\text{Var}[Z] < \text{Var}[X]$ .

Il est évidente que

$$\mathbb{E}[X] = \mathbb{E}[Z]$$

Calculons la variance de  $Z$

$$\begin{aligned} \text{Var}[Z] &= \mathbb{E}[(Z - \mathbb{E}[Z])^2] = \mathbb{E}[(X - b(Y - \mathbb{E}[Y])) - \mathbb{E}[X]]^2 = \\ &= \mathbb{E}[(X - \mathbb{E}[X])^2] - 2b\mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] + b^2\mathbb{E}[(Y - \mathbb{E}[Y])^2] = \\ &= \text{Var}[X] - 2b\text{Cov}[XY] + b^2\text{Var}[Y] \end{aligned}$$

La variance de  $Z$  est minimale si  $\frac{\partial Z}{\partial b} = 0$ . Donc

$$b = \frac{\text{Cov}[XY]}{\text{Var}[Y]}, \quad \text{Var}[Z] = \text{Var}[X] - \frac{(\text{Cov}[XY])^2}{\text{Var}[Y]}$$

### Variables de Control. Application à l'évaluation du prix d'une Option.

On applique maintenant cette idée au calcul du prix de l'option Européenne. Choisissons pour  $X$  et  $Y$  les variables aléatoires suivantes:

$$X = e^{-rT} \max(S_T - K, 0), \quad Y = S_T, \quad \mathbb{E}[Y] = S_0 e^{rT}$$

Le prix de l'option est égale à

$$V(S_0, 0) = \mathbb{E}[Z] = \mathbb{E}[X - b(Y - \mathbb{E}[Y])]$$

- Troisième estimateur du prix d'une option.

$$V(S_0, 0)_{\text{estime3}} = \frac{1}{N_{mc}} \sum_{n=1}^{N_{mc}} [e^{-rT} \max(S_T^{(n)} - K, 0) - b \cdot (S_T^{(n)} - S_0 e^{rT})]$$

# TP: Méthodes numériques avancées appliquées à la finance

→ On calcule b pour  $S_0 = 10$ ,  $N_{mc} = 10$  et  $N_{mc} = 100$  indépendamment de  $V(S_0, 0)_{estime}$

```
>> Code37
Pour Nmc=10, b= 0.877447
Pour Nmc=100, b= 0.691676
```

FIGURE 81 – Valeur de b pour  $N_{mc} = 10$  et  $N_{mc} = 100$

Le code sous Matlab du calcul de b :

```
1 - S0=10;
2 -
3 - fprintf("Pour Nmc=10, b= %f\n",b_estime(S0,10));
4 - fprintf("Pour Nmc=100, b= %f\n",b_estime(S0,100));
5 -
6 - function [b] = b_estime(S0,Nmc)
7 -
8 - % Définition des constantes et vecteurs %
9 -
10 - K=10;
11 - T=0.5;
12 - r=0.1;
13 - sigma=0.5;
14 - Nominateur=0;
15 - Denominateur=0;
16 -
17 - for n=1:Nmc
18 -     ST(n)=S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*randn);
19 -     Nominateur=Nominateur+(exp(-r*T)*Payoff_Europ_Call(ST(n),K)-Esperance_estime(S0,Nmc))*(ST(n)-S0*exp(r*T))^2;
20 -     Denominateur=Denominateur+(ST(n)-S0*exp(r*T))^2;
21 - end
22 -
23 - b=Nominateur/Denominateur;
24 -
25 - end
26 -
27 - function [prix] = Esperance_estime(S0,Nmc)
28 -
29 - % Définition des constantes et vecteurs %
30 -
31 - K=10;
32 - T=0.5;
33 - r=0.1;
34 - sigma=0.5;
35 -
36 - for n=1:Nmc
37 -     ST(n)=S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*randn);
38 -     gain(n)=Payoff_Europ_Call(ST(n),K);
39 - end
40 -
41 - prix=exp(-r*T)*mean(gain);
42 -
43 - end
44 -
45 - function [f] = Payoff_Europ_Call(S,K)
46 - f=max(S-K,0);
47 - end
48 -
```

FIGURE 82 – Code Sous MATLAB du fichier Code37.m

# TP: Méthodes numériques avancées appliquées à la finance

→ On calcule le prix du Call  $V(S_0, 0)_{estime3}$  pour  $S_0 = 10$ ,  $N_{mc} = 10$  et  $N_{mc} = 100$

```
>> Code38
Pour S0=10 et Nmc=10, V(S0,0)_estime3= 1.620357
Pour S0=10 et Nmc=100, V(S0,0)_estime3= 1.766832
```

FIGURE 83 – Le prix du Call  $V(S_0, 0)_{estime3}$  pour  $N_{mc} = 10$  et  $N_{mc} = 100$

Le code sous Matlab pour le prix du call  $V(S_0, 0)_{estime3}$  :

```
1 -      S0=10;
2 -
3 -      fprintf("Pour S0=10 et Nmc=10, V(S0,0)_estime3= %f\n",V_estime3(S0,10));
4 -      fprintf("Pour S0=10 et Nmc=100, V(S0,0)_estime3= %f\n",V_estime3(S0,100));
5 -
6 -      function [Prix_Call] = V_estime3(S0,Nmc)
7 -
8 -      % Définition des constantes et vecteurs %
9 -
10 -     K=10;
11 -     T=0.5;
12 -     r=0.1;
13 -     sigma=0.5;
14 -     valeur=0;
15 -
16 -     for n=1:Nmc
17 -         ST(n)=S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*randn);
18 -         valeur=valeur+(exp(-r*T)*Payoff_Europ_Call(ST(n),K)-b_estime(S0,Nmc)*(ST(n)-S0*exp(r*T)));
19 -     end
20 -
21 -     Prix_Call=valeur/Nmc;
22 -     end
23 -
24 -
25 -     function [b] = b_estime(S0,Nmc)
26 -
27 -     % Définition des constantes et vecteurs %
28 -
29 -     K=10;
30 -     T=0.5;
31 -     r=0.1;
32 -     sigma=0.5;
33 -     Nominateur=0;
34 -     Denominateur=0;
35 -
36 -     for n=1:Nmc
37 -         ST(n)=S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*randn);
38 -         Nominateur=Nominateur+(exp(-r*T)*Payoff_Europ_Call(ST(n),K)-Esperance_estime(S0,Nmc))*(ST(n)-S0*exp(r*T));
39 -         Denominateur=Denominateur+(ST(n)-S0*exp(r*T))^2;
40 -     end
41 -
42 -     b=Nominateur/Denominateur;
43 -
44 -     end
45 -
46 -
47 -     function [prix] = Esperance_estime(S0,Nmc)
48 -
49 -     % Définition des constantes et vecteurs %
50 -
51 -     K=10;
52 -     T=0.5;
53 -     r=0.1;
54 -     sigma=0.5;
55 -
56 -     for n=1:Nmc
57 -         ST(n)=S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*randn);
58 -         gain(n)=Payoff_Europ_Call(ST(n),K);
59 -     end
60 -
61 -     prix=exp(-r*T)*mean(gain);
62 -
63 -     end
64 -
65 -     function [f] = Payoff_Europ_Call(S,K)
66 -     f=max(S-K, 0);
67 -     end
68 -
```

FIGURE 84 – Code Sous MATLAB du fichier Code38.m

# TP: Méthodes numériques avancées appliquées à la finance

---

→ Tracer le graphe  $S_0 \rightarrow V(S_0, 0)_{estime3}$  pour  $N_{mc} = 10$  et  $N_{mc} = 100$

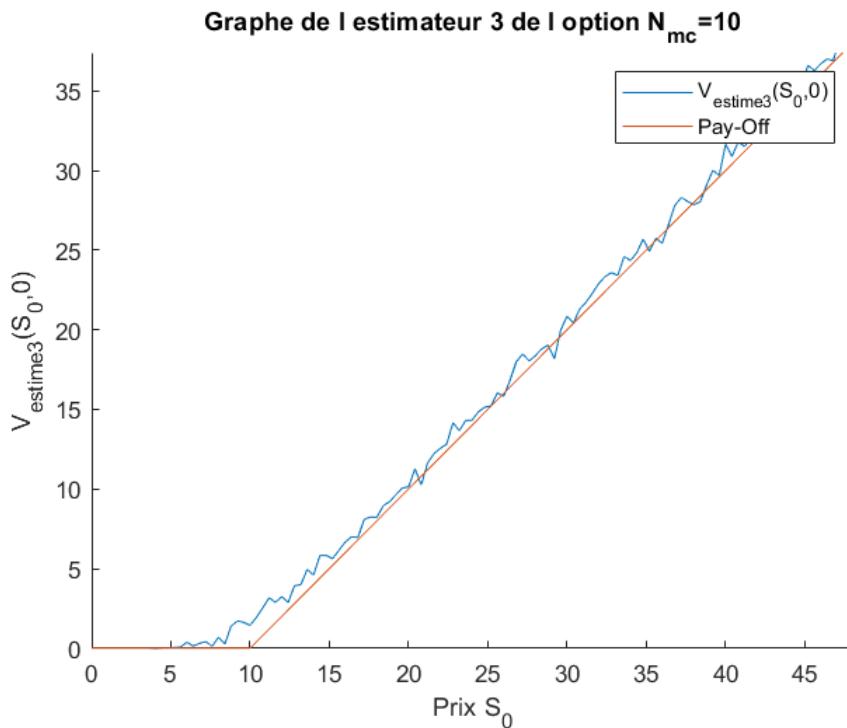


FIGURE 85 – Graphe de l'estimateur 3 pour ( $N_{mc} = 10$ )

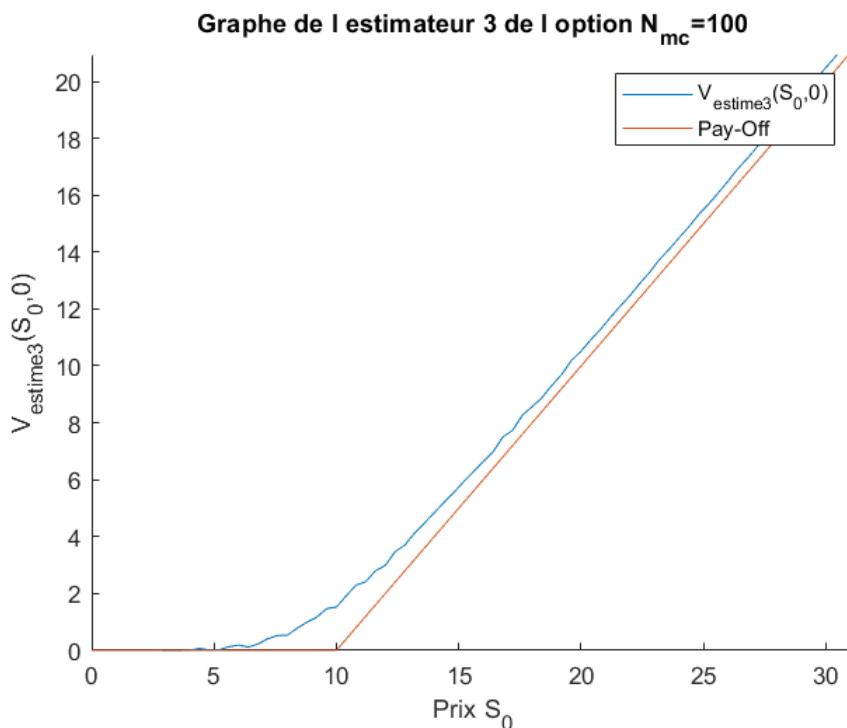


FIGURE 86 – Graphe de l'estimateur 3 pour ( $N_{mc} = 100$ )

On observe que la courbe  $S_0 \rightarrow V(S_0, 0)_{estime3}$  de converge rapidement donc un intervalle de confiance réduit. Cette méthode de variable de contrôle est efficace aussi.

# TP: Méthodes numériques avancées appliquées à la finance

---

Le code sous Matlab du Graphe de l'estimateur 3 pour ( $N_{mc} = 100$ ) :

```
2 -      Nmc=100;
3 -
4 -      graphe_V_estime3(Nmc)
5 -
6 -  function [] = graphe_V_estime3(Nmc)
7 -      K=10;
8 -
9 -      for j=1:500
10 -        S0(j)=0.4*(j-1);
11 -        prix_graphe(j)=V_estime3(S0(j),Nmc);
12 -      end
13 -
14 -      figure;
15 -      hold;
16 -      plot(S0,prix_graphe)
17 -      plot(S0,max(S0-K,0))
18 -      xlabel('Prix S_0')
19 -      ylabel('V_{estime3}(S_0,0)')
20 -      legend('V_{estime3}(S_0,0)', 'Pay-Off');
21 -      title('Graphe de l estimateur 3 de l option N_{mc}=100')
22 -    end
23 -
24 -  end
25 -
26 -
27 -
28 -  function [Prix_Call] = V_estime3(S0,Nmc)
29 -  % Définition des constantes et vecteurs %
30 -
31 -  K=10;
32 -  T=0.5;
33 -  r=0.1;
34 -  sigma=0.5;
35 -  valeur=0;
36 -
37 -  for n=1:Nmc
38 -    ST(n)=S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*randn);
39 -    valeur=valeur+(exp(-r*T)*Payoff_Europ_Call(ST(n),K)-b_estime(S0,Nmc)*(ST(n)-S0*exp(r*T)));
40 -  end
41 -
42 -  Prix_Call=valeur/Nmc;
43 -  end
44 -
45 -  end
46 -
47 -
48 -  function [b] = b_estime(S0,Nmc)
49 -  % Définition des constantes et vecteurs %
50 -
51 -  K=10;
52 -  T=0.5;
53 -  r=0.1;
54 -  sigma=0.5;
55 -  Nominateur=0;
56 -  Denominateur=0;
57 -
58 -  for n=1:Nmc
59 -    ST(n)=S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*randn);
60 -    Nominateur=Nominateur+(exp(-r*T)*Payoff_Europ_Call(ST(n),K)-Esperance_estime(S0,Nmc))*(ST(n)-S0*exp(r*T));
61 -    Denominateur=Denominateur+(ST(n)-S0*exp(r*T))^2;
62 -  end
63 -
64 -  b=Nominateur/Denominateur;
65 -  end
66 -
67 -  end
68 -
69 -  function [prix] = Esperance_estime(S0,Nmc)
70 -  % Définition des constantes et vecteurs %
71 -
72 -  K=10;
73 -  T=0.5;
74 -  r=0.1;
75 -  sigma=0.5;
76 -
77 -  for n=1:Nmc
78 -    ST(n)=S0*exp((r-(sigma^2)/2)*T+sigma*sqrt(T)*randn);
79 -    gain(n)=Payoff_Europ_Call(ST(n),K);
80 -  end
81 -
82 -  prix=exp(-r*T)*mean(gain);
83 -  end
84 -
85 -  end
86 -
87 -  function [f] = Payoff_Europ_Call(S,K)
88 -  f=max(S-K,0);
89 -  end
```

FIGURE 87 – Code Sous MATLAB du fichier Code39.m

## 7 Partie VII. Le lien entre les méthodes déterministes (Différences Finies) et stochastiques (Monte-Carlo)

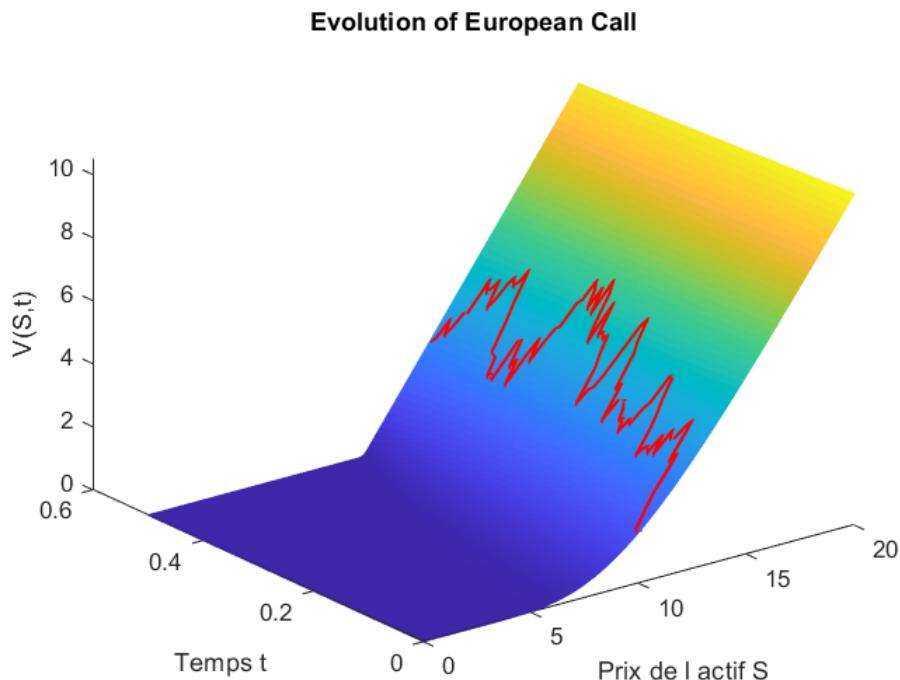


FIGURE 88 – Evolution of European Call

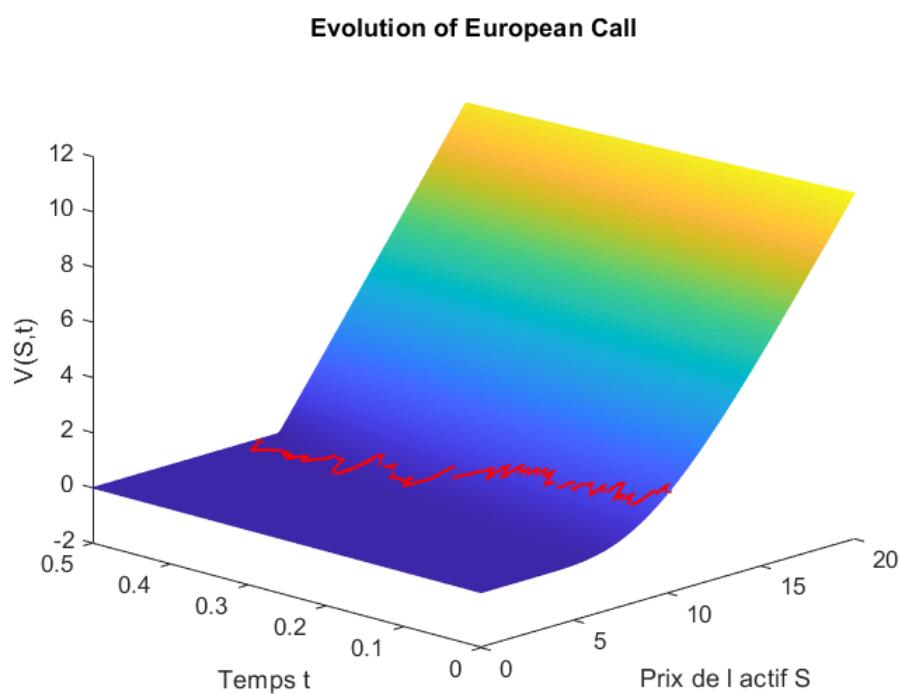


FIGURE 89 – Evolution of European Call

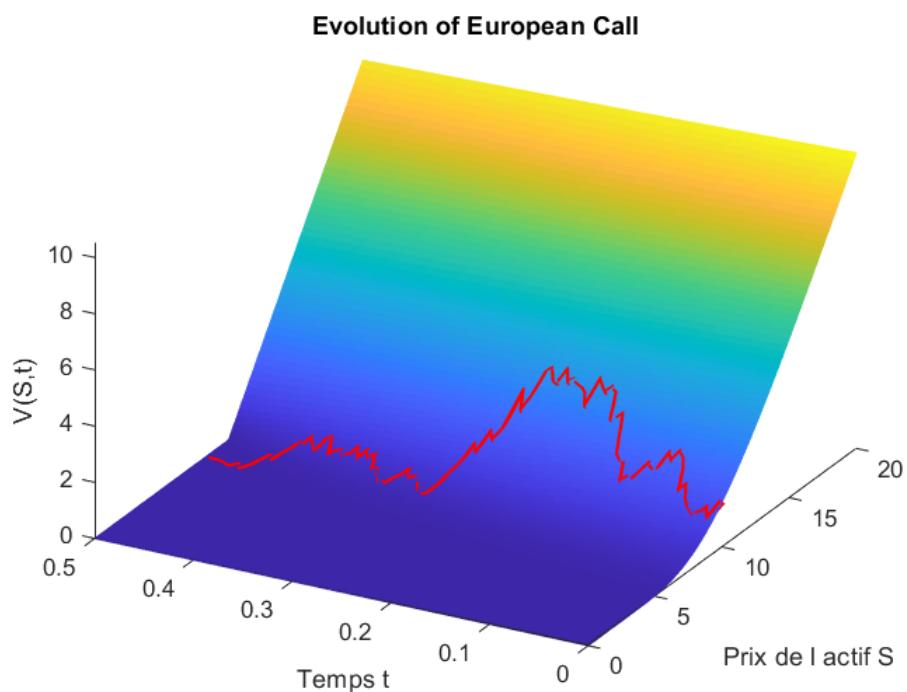


FIGURE 90 – Evolution of European Call

Si on réduit le nombre de discréétisation de  $S$  on obtient :

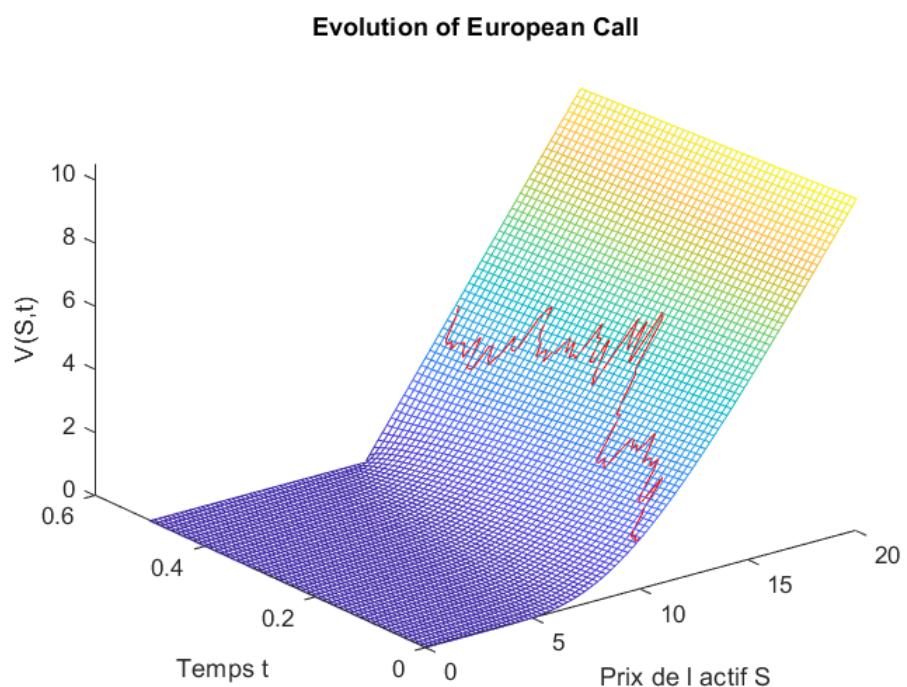


FIGURE 91 – Evolution of European Call

# TP: Méthodes numériques avancées appliquées à la finance

Le code sous Matlab pour Evolution of European Call :

```
2 -      visualiser_fonction_V()
3 -
4 -  function [] = visualiser_fonction_V()
5 -
6 -    % Définition des constantes %
7 -
8 -    L=20;
9 -    K=10;
10 -   T=0.5;
11 -   r=0.1;
12 -   sigma=0.5;
13 -   N=99;
14 -   M=4999;
15 -   S(1)=10;
16 -   t(1)=0;
17 -   delta_t=T/(N+2);
18 -   S1=linspace(0,L,N+2);
19 -   t1=linspace(0,T,M+2);
20 -
21 -   for i=1:N+1
22 -     S(i+1)=S(i)*exp((r-(sigma^2)/2)*delta_t+sigma*sqrt(delta_t)*randn);
23 -     t(i+1)=t(i)+delta_t;
24 -   end
25 -
26 -   for j=1:N+2
27 -     price_option(j)=BS_theorie(S(j),K,r,sigma,t(j),T);
28 -   end
29 -
30 -   for k=1:M+2
31 -     for l=1:N+2
32 -       price_BS(k,l)=BS_theorie(S1(l),K,r,sigma,t1(k),T);
33 -     end
34 -   end
35 -
36 -   figure;
37 -   plot3(S,t,price_option,'r','LineWidth',3)
38 -   hold on;
39 -   mesh(S1,t1,price_BS)
40 -   xlabel('Prix de l actif S')
41 -   ylabel('Temps t')
42 -   zlabel('V(S,t)')
43 -   title('Evolution of European Call')
44 -   end
45 -
46 - function [f] = BS_theorie(S,K,r,sigma,t,T)
47 - if (t==T)
48 -   f=max(S-K,0);
49 - else
50 -   f=S*N(d1(S,K,r,sigma,t,T))-K*exp(-r*(T-t))*N(d2(S,K,r,sigma,t,T));
51 - end
52 - end
53 -
54 - function [f] = d1(S,K,r,sigma,t,T)
55 -   f = (log(S/K)+(r+(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
56 - end
57 -
58 - function [f] = d2(S,K,r,sigma,t,T)
59 -   f = (log(S/K)+(r-(sigma^2)/2)*(T-t))/(sigma*sqrt(T-t));
60 - end
61 -
62 - function [f] = N(x)
63 -   f = 1/2*(1+erf(x/sqrt(2)));
64 - end
```

FIGURE 92 – Code Sous MATLAB du fichier Code40.m

## IV Conclusion

Je tiens tout d'abord à remercier mon professeur Irina Kortchemski pour ces efforts agréables malgré l'état sanitaire et l'enseignement à distance.

Le cours Méthodes numériques avancée appliquée à la finance m'a permis de bien comprendre les notions et les techniques fondamentales de la méthode de simulation Monte Carlo grâce à des algorithmes en Matlab et les liens de celle-ci avec les principales méthodes de résolution numérique des Équations Différentielles Stochastiques ainsi que la modélisation et estimation par simulation des différents paramètres intervenant aux problèmes des finances quantitatives.

Grâce au module "Méthodes numériques avancée appliquée à la finance" pour l'année 2020-2021, j'ai accumulé de nombreuses connaissances :

- Résolution numérique de l'équation de Black et Scholes.
- Option Call Europ / Butterfly / Americ.
- Options Call, Butterfly par Monte-Carlo.
- Méthode de la réduction de la variance.
- Méthode aux Différences Finies.
- Conditions aux limites de Neumann.
- Conditions aux limites de Dirichlet.
- Programmation des solutions analytiques de Black et Sholes.
- Volatilité locale.

