université de BORDEAUX

TP2 - Mouvement Brownien et Modèle de Black-Scholes

Mots clés :
- **Scilab**
- **Marche aléatoire**
- **Mouvement brownien**
- **Modèle de Black-Scholes**
- **Option vanille (européenne)**
- **Call & Put**

Élève : Boudjeltia Reda
M2 MIMSE Spé 3 AD
Date : 11 Octobre 2015

Table des matières

1 Marches aléatoires et trajectoires Browniennes

1.1 La marche aléatoire

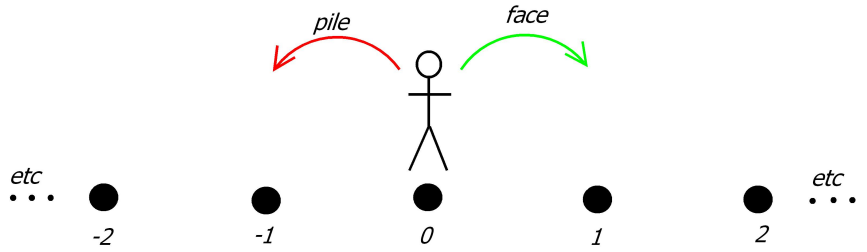

FIGURE 1 – Illustration de la marche aléatoire

On part d'un nombre de pas : $P_a = D + G$ et d'une position $P_o = D - G$ avec D le nombre de pas à droite et G à gauche (Pile ou face).
On pose le système suivant :

$$\begin{cases} P_a = D + G & (1.1) \\ P_o = D - G & (1.2) \end{cases}$$

On en déduit en fixant le nombre de pas à 1

$$\begin{cases} G = P_a - D & (1.1) \\ P_o = 2D - 1 & (1.2) \end{cases}$$

Pour la suite, on notera X_i la variable aléatoire qui est associé à la position au temps i, D_i le i^{eme} pas. Ainsi, si $D_i = 0$ alors $X_i = -1$ et si $D_i = 1$ alors $X_i = 1$.

Or, D_i suit une loi binomiale $B(n, p)$ avec $p = \frac{1}{2}$ et n le nombre de pas. Car la suite D_n est une suite de variable de Bernouilli de paramètre p.
Par Conséquent, pour un pas on a :

$$\begin{aligned} \mathbb{P}(X_i = -1) &= \mathbb{P}(2D_i - 1 = -1) && (1) \\ &= \mathbb{P}(D_i = 0) && (2) \\ &= \binom{n}{1} p^0 (1-p)^{n-0} && (3) \\ &= \binom{1}{1} p^0 (1-p)^{1-0} && (4) \\ &= 1 - p = \frac{1}{2} && (5) \end{aligned}$$

De même,

$$\begin{aligned} \mathbb{P}(X_i = 1) &= \mathbb{P}(2D_i - 1 = 1) && (6) \\ &= \mathbb{P}(D_i = 1) && (7) \\ &= \binom{n}{1} p^1 (1-p)^{n-1} && (8) \\ &= \binom{1}{1} p^1 (1-p)^{1-1} && (9) \\ &= p = \frac{1}{2} && (10) \end{aligned}$$

Au final, on obtient bien $\mathbb{P}(X_i = 1) = \mathbb{P}(X_i = -1) = \frac{1}{2}$.

Ensuite, on a modélisé sur `scilab` le modèle.

```
function [P]=Pos(n)
    D = grand(1,1,"bin",n, 0.5);
    P = 2*D-1;
endfunction

function [p] = RandWalk(n)
    p(1) = 0;
    for i=2:n
        p(i) = p(i-1) + Pos(1)
    end
endfunction
```

FIGURE 2 – Code pour une simulation

Puis, on a tracé 10 marches. On s'aperçoit que les courbes ne sont pas corrélées. C'est bien ce qu'on cherche à avoir.

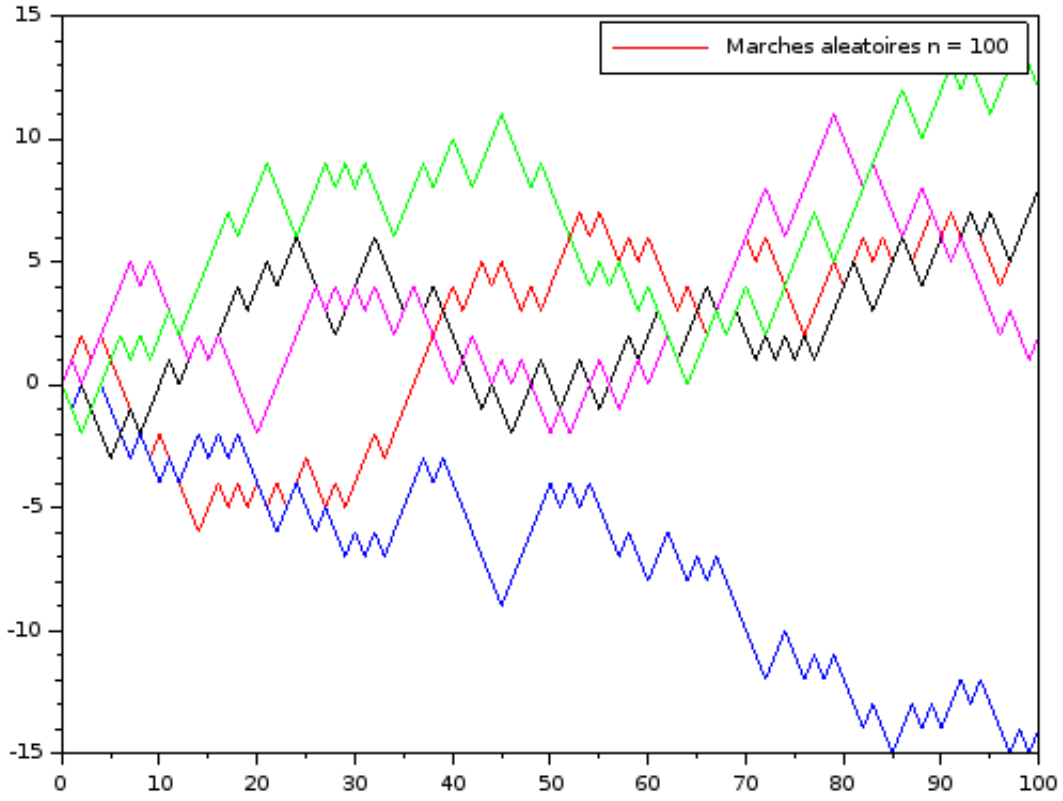

FIGURE 3 – 10 marches aléatoires

1.2 Le mouvement Brownien

Après avoir simulé plusieurs $B^N(\frac{k}{N}) = \frac{M_k}{\sqrt{N}}$, on s'aperçoit que la courbe se limite à rester que dans l'intervalle $[0, 1]$ à mesure que N devient grand (illustration du théorème de Donsker).

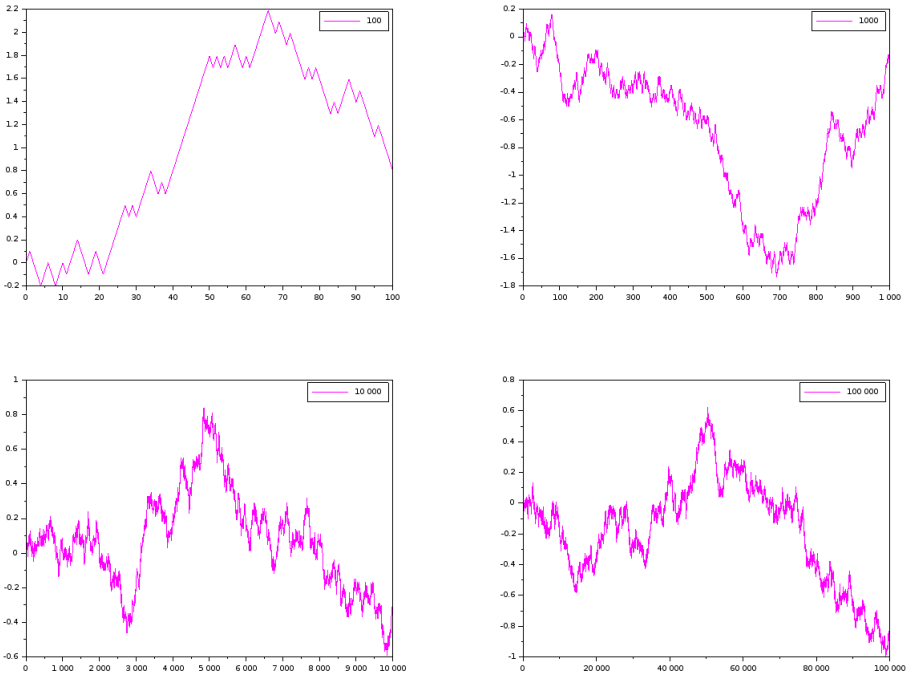

FIGURE 4 – Tracés de B^N

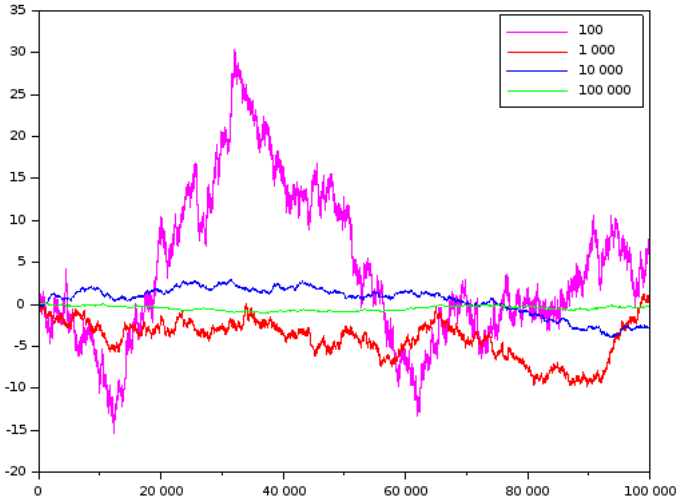

FIGURE 5 – Vue générale des différents B^N

L'implémentation `scilab` du mouvement Brownien est la suivante.

```
function [p]=brown(k,n)
    V = RandWalk(k);
    for i=1:n
        p(i) = V(i)/sqrt(n);
    end
endfunction
```

FIGURE 6 – Implémentation du mouvement pseudo mouvement Brownien

2 Modèle de Black et Scholes

On donne $(S_t)_{0\leq t\leq 1}$ la trajectoire d'un actifs risqué suivant le modèle de Black et Scholes de dynamique.

$$S_t = exp(\mu.t - \frac{\sigma^2}{2}.t + \sigma.B_t) \tag{11}$$

avec μ la dérive, ρ la volatilité, et B_t un mouvement brownien standard.
La dérive à tendance à faire lever la courbe vers des objectifs plus hauts d'une part.

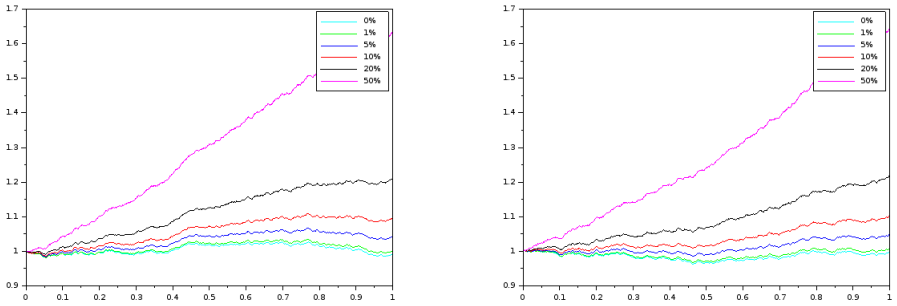

FIGURE 7 – En faisant varier la dérive pour une volatilité de 10%

D'autre part, la volatilité à tendance à amplifier le bruit de la courbe. Ainsi plus elle est grande plus le bruit prend de grande amplitude. Cela s'explique par fait que μ est le coefficient de B_t ce qui a pour effet de mettre de mettre en avant l'aspect risqué de l'actif.
En effet, en cherchant un peu l'origine de la formule (11), on voit qu'elle est solution de l'équation différentielle stochastique suivante.

$$dS_t = S_t(\mu.dt + \sigma.dB_t) \tag{12}$$

or si $\sigma = 0$, on retombe sur la forme d'une équation d'un actif sans risque.

$$dS_t = S_t(\mu.dt) \tag{13}$$

où μ serait le taux d'intérêt de l'actif. Ce qui démontre la figure 6. Puisque plus l'intérêt est élevé plus les profits augmentent.

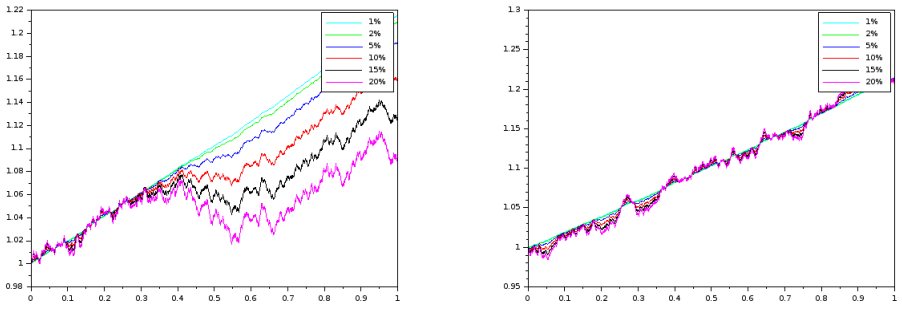

FIGURE 8 – Deux simulations en faisant varier la volatilité pour une dérive de 20%

L'implémentation `Scilab` est la suivante.

```
function [p] = BS(sigma, mu, b)
    t = [0: 0.001: 1];
    for i=1:length(t)
        p(i) = exp(mu*t(i) - rho**2*t(i)/2 + sigma*b(i));
    end
endfunction
```

FIGURE 9 – Modèle de black & Scholes pour la trajectoire d'un actifs risqué

3 Évaluation d'une option européenne

3.1 Pour un Call

On donne

$$C(t,x) \quad = \quad xN(d_1) - Ke^{-r(T-t)}N(d_2) \tag{14}$$

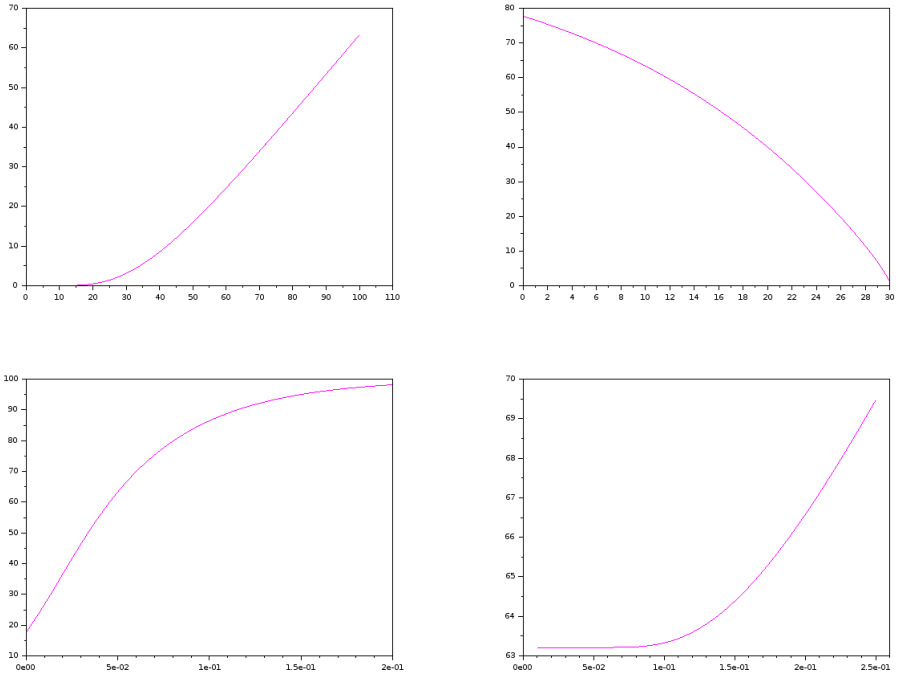

Figure 10 – Prix du call en faisant varier des données d'entrée pour un prix (Stock Price) de 100 unités

La première figure en haut à gauche représente la courbe quand on fait varier le prix actuel (ou `Stock price`) x de 0.01 à 100. Ainsi, elle croit quasiment linéairement à partir d'environ $x = 50$.

La seconde en haut à droite est quand on fait varier la date de l'exercice t de 0 à 29.99 années avant la date d'échéance qui est fixée à 30. Plus on s'approche de la date d'échéance plus le prix décroit. Cela est un comportement attendu car plus on avance à l'échéance moins le risque est élevé car on a plus d'information sur l'avenir, moins l'option rapportera.

La troisième en bas à gauche est quand on fait varier le taux d'intérêt sans risque r de 0% à 20%. Le prix croit rapidement jusqu'à ralentir à partir de 15%.

La dernière est quand on fait varier la volatilité σ de 0.01 à 0.25. On s'aperçoit que la volatilité influence le prix de l'option surtout après le dépassement de 10% où l'exponentielle se développe. Il est parfois dit qu'acheter une option revient à acheter de la volatilité. La figure en est l'illustration même.

L'implémentation `Scilab` du modèle est la suivante.

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

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

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

Figure 11 – Implémentation du Call-price

3.2 Pour un Put

Même si cette partie n'est pas demandé il m'a semblé pertinent de l'étudié à titre personnel afin de vérifié la validité des courbes obtenu avant.
on a

$$P(t,x) \quad = \quad -xN(-d_1) + Ke^{-r(T-t)}N(-d_2) \tag{15}$$

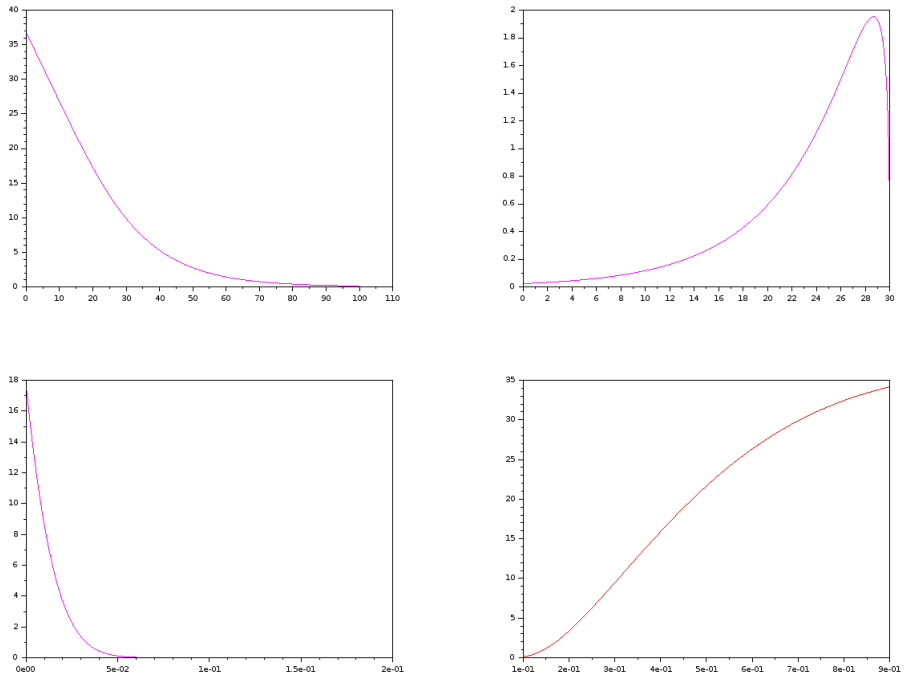

Figure 12 – Prix d'un Put en faisant varier des données d'entrée pour un prix (Stock Price) de 100 unités

Remarquons une diminution brutale du prix dépassé le 28^{me} jour sur la seconde courbe, cela parait logique car on aura $T-t$ va tendre vers 0. De plus, qu'on soit dans un `Put` ou un `Call` la volatilité fait toujours augmenté le prix de l'option.

L'implémentation `Scilab` du modèle est la suivante. avec d1 et d2 qui sont les mêmes que précédemment.

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

Figure 13 – Implémentation du Call-price

4 Code complet

.

```
//Exercice 1

n = 10;
p0 = 1/2;
p1 = 1/2;

D = grand(1,1,"bin",n, p0);
G = grand(1,1,"bin",n,p1)

P = D-G;
Pa = D+G

function [P]=Pos(n)
    D = grand(1,1,"bin",n, 0.5);
    P = 2*D-1;
endfunction


function [p] = RandWalk(n)
    p(1) = 0;
    for i=2:n
        p(i) = p(i-1) + Pos(1)
    end
endfunction

clf();
x=[0:100]
    plot(x, RandWalk(101), "r-");
    plot(x, RandWalk(101), "b-");
    plot(x, RandWalk(101), "k-");
    plot(x, RandWalk(101), "m-");
    plot(x, RandWalk(101), "g-");
legend(["Marches␣aleatoires␣n␣=␣100"])


function [p]=brown(k,n)
    V = RandWalk(k);
    for i=1:n
        p(i) = V(i)/sqrt(n);
    end
endfunction

 B = brown(100000,100)

//clf();
//x=[0:100]
//plot(x, brown(100001,101), "m-")
//legend(["100"])
```

Figure 14 – Code pour une simulation

```
//clf();
//x=[0:1000]
//plot(x, brown(100001,1001),"m-")
//legend(["1000"])

//clf();
//x=[0:10000]
//plot(x, brown(100001,10001),"m-")
//legend(["10 000"])


//clf();
//x=[0:100000]
//plot(x, brown(100001,100001),"m-")
//legend(["100 000"])



//Exercice 2


function [p] = BS(sigma, mu, b)
    t = [0: 0.001: 1];
    for i=1:length(t)
        p(i) = exp(mu*t(i) - rho**2*t(i)/2 + sigma*b(i));
    end
endfunction


function [p] = simuBS()
    k = 10000;
    N = 10000;
    b = brown(k, N);
    rho = 0.1;
    mu = [0, 0.01, 0.05, 0.1, 0.2, 0.5];
    for i =1:6
        p(i) = BS(rho, mu(i), b(i));
    end

endfunction

//Cas 1

k = 10000;
N = 10000;
b = brown(k, N);
sigma = 0.1;
mu = [0, 0.01, 0.05, 0.1, 0.2, 0.5]
```

Figure 15 – Code pour une simulation

```
//Variation de la dérive
//clf();
//x= [0: 0.001: 1];
//plot(x, BS(0.1, 0, b),"c-")
//plot(x, BS(0.1, 0.01, b),"g-")
//plot(x, BS(0.1, 0.05, b),"b-")
//plot(x, BS(0.1, 0.1, b),"r-")
//plot(x, BS(0.1, 0.2, b),"k-")
//plot(x, BS(0.1, 0.5, b),"m-")
//legend(["0%"; "1%"; "5%";"10%"; "20%"; "50%"]);



//Variation de la volatilité
//clf();
//x= [0: 0.001: 1];
//plot(x, BS(0.01, 0.2, b),"c-")
//plot(x, BS(0.02, 0.2, b),"g-")
//plot(x, BS(0.05, 0.2, b),"b-")
//plot(x, BS(0.1 ,0.2, b),"r-")
//plot(x, BS(0.15, 0.2, b),"k-")
//plot(x, BS(0.2, 0.2, b),"m-")
//legend(["1%"; "2%"; "5%";"10%"; "15%"; "20%"]);



//Exercice 4

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

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

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

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

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

Figure 16 – Code pour une simulation

```
//Strike price
clf();
x= [0.01: 0.001: 100];
function [p] = test1(x)
    for i = 1:length(x)
        p(i) = Call(x(i),10,30,100,0.05,0.1);
    end
endfunction

plot(x, test1(x),"m-")


//Temps avant echeance
x= [0:0.1:29.99];
function [p] = test2(x)
    for i = 1:length(x)
        p(i) = Call(100, x(i),30,100,0.05,0.1);
    end
endfunction

//plot(x, test2(x),"m-")


//taux interet
x= [0:0.0001:0.2];
function [p] = test3(x)
    for i = 1:length(x)
        p(i) = Call(100, 10,30,100,x(i),0.1);
    end
endfunction

//plot(x, test3(x),"m-")


//volatilité
x= [0.01:0.0001:0.25];
function [p] = test4(x)
    for i = 1:length(x);
        p(i) = Call(100,10,30,100,0.05,x(i));
    end
endfunction

//plot(x, test4(x),"m-")
```

Figure 17 – Code pour une simulation

```
//Put

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


//clf();
x= [0.01: 0.001: 100];
function [p] = test1(x)
    for i = 1:length(x)
        p(i) = Put(x(i),10,30,100,0.05,0.1);
    end
endfunction


//plot(x, test1(x),"m-")

x = [0:0.01:29.999];
function [p] = test2(x)
    for i = 1:length(x)
        p(i) = Put(100, x(i),30,100,0.05,0.1);
    end
endfunction
//plot(x, test2(x),"m-")


x= [0:0.0001:0.2];
function [p] = test3(x)
    for i = 1:length(x)
        p(i) = Put(100, 10,30,100,x(i),0.1);
    end
endfunction
//plot(x, test3(x),"m-")

x= [0.1:0.001:0.9];
function [p] = test4(x)
    for i = 1:length(x)
        p(i) = Put(100, 10,30,100,0.05,x(i));
    end
endfunction
//plot(x, test4(x),"r-")
```

Figure 18 – Code pour une simulation