Simulation d'un marché financier

Sercan Sen

Introduction

- Comment une approche mathématique et informatique permet-elle de simuler l'évolution des prix dans un marché financier ainsi que les risques engendrés par les fluctuations ?

Deux approches :

- L'équation de Black-Scholes
- Le jeu de la minorité

Définition : Call Européen/Put Européen

Option d'achat/de vente : Droit d'acheter/vendre à un prix E au temps T(fixés dans le contrat)

- Problème : Le vendeur peut théoriquement avoir une perte infinie

L'équation de Black-Scholes

$$\forall (t, S) \in [0, T] x \mathbb{R}^+, \frac{\partial C}{\partial t}(S, t) + rS\frac{\partial C}{\partial t} + \frac{\sigma}{2}S^2\frac{\partial^2 C}{\partial S^2} - rC(S, t) = 0$$

$$\forall S \in \mathbb{R}^+, C(S, T) = (S - E)^+$$

- C est la valeur de l'option d'achat
- S est la valeur du sous-jacent
- E est le strike ie le prix du sous-jacent au temps T
- T est la maturité ie date fixée pour l'échange
- $t \in [0, T]$ est le temps
- σ est la volatilité constante
- r est le taux d'intérêt constant

Définition : Volatilité

Amplitude de la variation du prix d'un actif i.e. mesure de l'instabilité du prix d'un actif financier

Pour une volatilité élevée, le risque de perte est important mais le risque de gain l'est aussi.

Il est possible de résoudre l'équation de Black-Scholes en connaissant S, T, σ, r et E.
S, T, r et E sont imposé par le marché. Il faut donc déterminer σ.
En résolvant analytiquement l'EDP, on peut déterminer la volatilité par dichotomie. Dans la littérature, on trouve :

$$C(S,t) = \frac{S}{2} erfc(\frac{-d1}{\sqrt{2}}) - \frac{E}{2} e^{-r(T-t)} erfc(\frac{-d2}{\sqrt{2}})$$

où $erfc : x \mapsto \frac{2}{\sqrt{\pi}} \int_x^{+\infty} e^{-t^2} dt$

$d_1 = \frac{(log(\frac{S}{E}) + (r + \frac{\sigma^2}{2})T)}{\sigma\sqrt{T}}$

$d_2 = d_1 - \sigma\sqrt{T}$ (Algorithme en annexe)

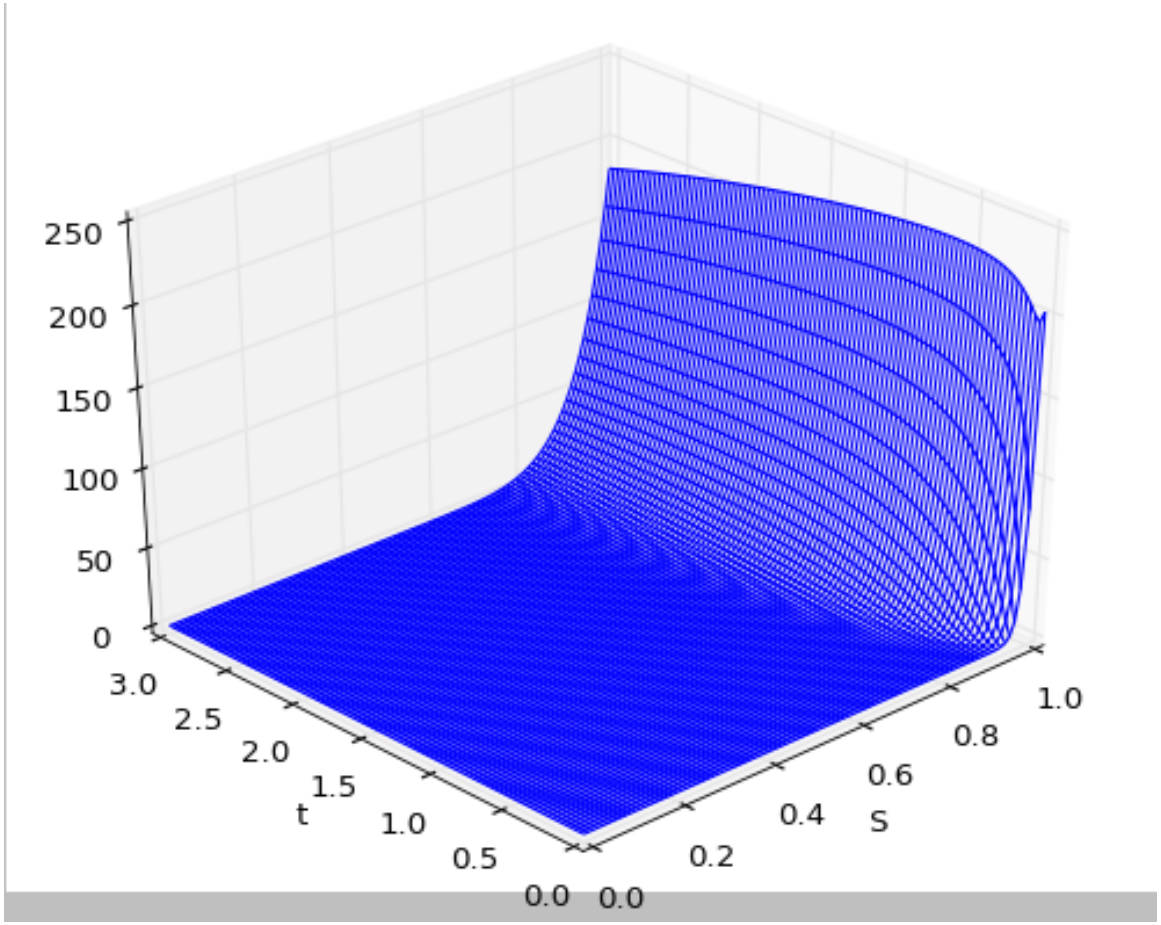
- On utilise la méthode des différences finies pour trouver la valeur de l'option d'achat pour tout prix de sous jacent dans $[0, R]$ et tout temps dans $[0, T]$
- On choisit h le pas du prix et k le pas de temps.
- $\frac{\partial C}{\partial t}(S_i, t_j) \approx \frac{C(S_i,t_{j+1})-C(S_i,t_j)}{k}$
- $\frac{\partial C}{\partial S}(S_i, t_j) \approx \frac{C(S_{i+1},t_j)-C(S_i,t_j)}{h}$
- $\frac{\partial^2 C}{\partial S^2}(S_i, t_j) \approx \frac{C(S_{i+1},t_j)-2C(S_i,t_j)+C(S_{i-1},t_j)}{h^2}$
- En reportant dans l'EDP,
- $\frac{C(S_i,t_{j+1})-C(S_i,t_j)}{k} + rS_i\frac{C(S_{i+1},t_j)-C(S_i,t_j)}{h} + \frac{\sigma^2}{2}S_i\frac{C(S_{i+1},t_j)-C(S_i,t_j)}{h} - rC(S_i, t_j) = 0$

- On se ramène alors à un système tri-diagonal tel que :

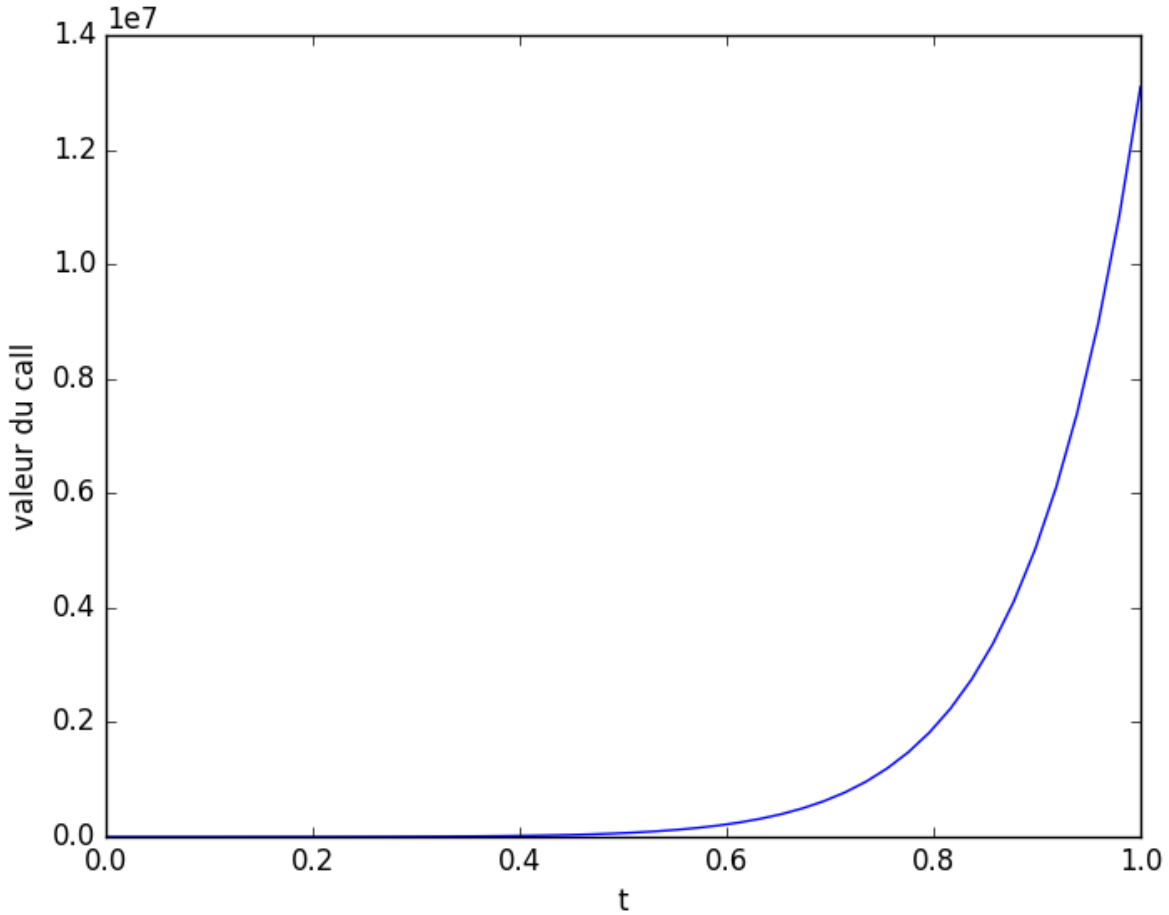
$$AC_{n+1} = C_n + B_n \text{ où } A = \begin{pmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ c_1 & a_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & b_N \\ 0 & \cdots & 0 & c_N & a_N \end{pmatrix} \text{ et } B_n = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ B(M) \end{pmatrix}$$

où pour tout $j \in [|1, N|]$:

- $a_j = 1 + k(\frac{\sigma S_j}{h})^2 + rk$
- $b_j = \frac{-rkS_j}{2h} - \frac{k}{2}(\frac{\sigma S_j}{h})^2$
- $c_j = \frac{rkS_j}{2h} - \frac{k}{2}(\frac{\sigma S_j}{h})^2$


250
200
150
100
50
0
3.0
2.5
2.0
1.5
t
1.0
0.5
0.0
0.0
0.2
0.4
S
0.6
0.8
1.0

En pratique, le prix actuel de l'option d'achat étant connu, on utilise l'équation de Black-Scholes pour déterminer la volatilité. Ce qui donne l'information au trader sur le risque de gain/perte.



Hypothèses non-réalistes :

- La volatilité est considérée constante
- Le taux d'intérêt est considéré constant
- Les évènements improbables sont sous-estimés

Cependant, ce modèle étant simple, il reste très utilisé.

Le jeu de la minorité

On considère un modèle avec moins d'hypothèses dans lequel les acteurs interagissent entre eux.

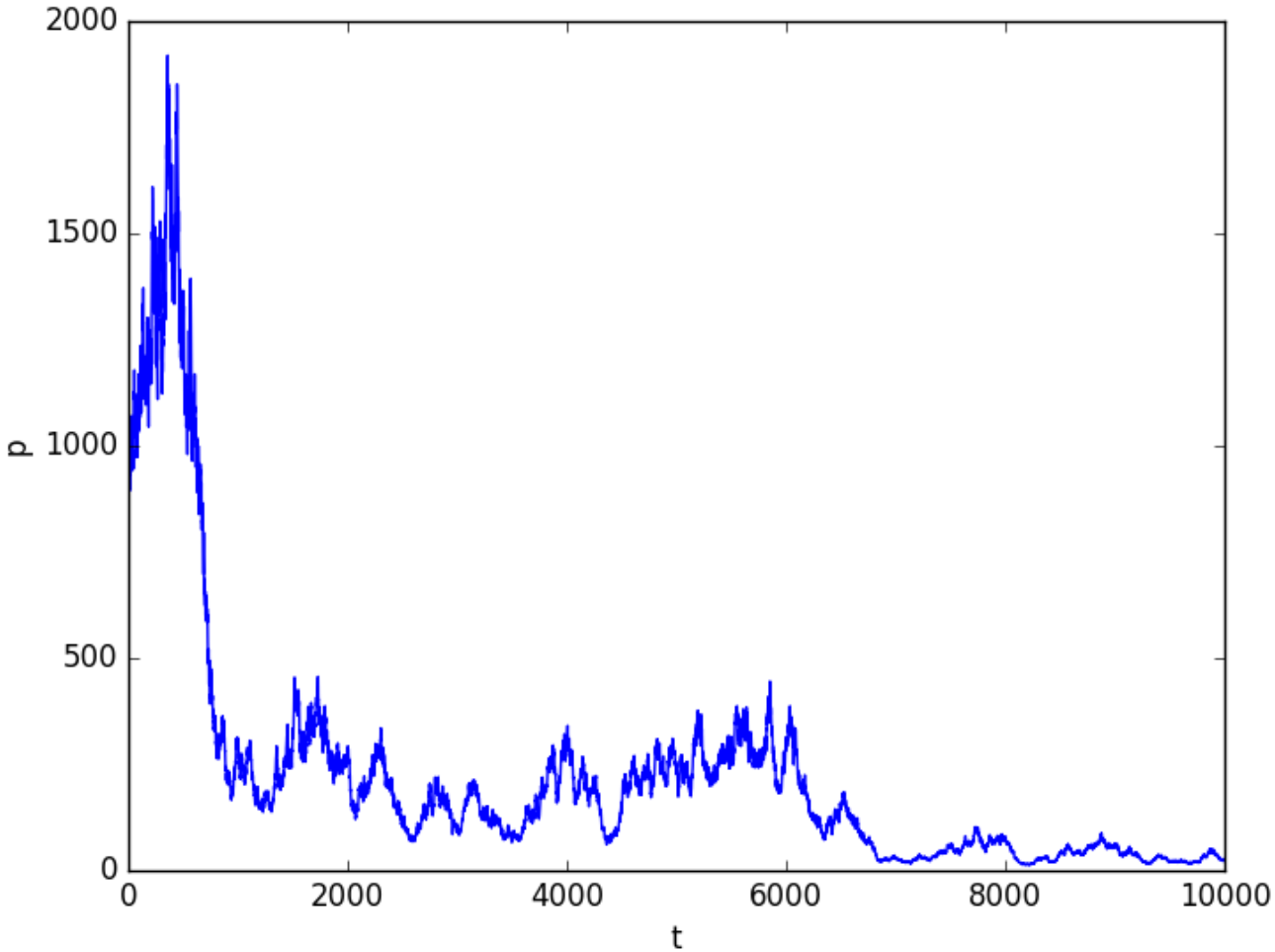
- N agents : producteurs ou spéculateurs
- Chaque agent i a le choix de faire un échange : $a_i(t) = 1$ si c'est un achat, $a_i(t) = -1$ si c'est une vente
- $A(t) = \sum a_i(t)$ donne l'information sur l'offre et la demande et le signe de $g(t) = -a_i(t)A(t)$ définit si i a gagné ou perdu avec cet échange
- $\mu(t) \in [|1, P|]$ représente l'état du monde et est tiré au hasard pour tout t.

- On note N_p leur nombre
- Action déterministe qui suit $\mu(t)$
- Pour chaque agent i et chaque état du monde $\mu(t)$, on tire au hasard $a_i(t) = \pm 1$

- On note N_s leur nombre
- Action dépandant d'une stratégie que choisit chaque spéculateur.
- On considère S stratégies et on choisit l'ensemble $[|0, N|]$ pour les représenter.
- La stratégie 0 est la stratégie ne rien faire. Donc pour la stratégie 0, $a_i(t) = 0$
- Pour une stratégie non-nulle s, $a_i(t) = \sigma_{i,s}^{\mu(t)}$ où $\sigma_{i,s}^{\mu(t)} = \pm 1$
- On assigne un score $U_{i,s}(t)$ à chaque stratégie. Le spéculateur choisit la stratégie ayant le score le plus important
- Le score de la stratégie 0 est augmenté d'une valeur constante ϵ i.e. $U_{i,s}(t+1) = U_{i,s}(t) + \epsilon$
- Le score des autres stratégies est augmenté au temps $t+1$ s'il aurait permis un gain au temps t et diminué sinon i.e. $U_{i,s}(t+1) = U_{i,s}(t) - a_i(t)A(t)$

- On note N_s leur nombre
- Action depandant d'une stratégie que choisit chaque spéculateur.
- On considère S stratégies et on choisit l'ensemble $[|0, N|]$ pour les représenter.
- La stratégie 0 est la stratégie ne rien faire. Donc pour la stratégie 0, $a_i(t) = 0$
- Pour une stratégie non-nulle s, $a_i(t) = \sigma_{i,s}^{\mu(t)}$ où $\sigma_{i,s}^{\mu(t)} = \pm 1$

On définit la dynamique des prix par :
$log(p(t+1)) = log(p(t)) + \frac{A(t)}{\lambda}$



```
def BScall(S,E,r,T,sigma):
    d1 = (np.log(S/E) + (r + sigma ** 2 / 2) * T) / (sigma * T**0.5)
    d2 = d1 - sigma * (T ** 0.5)
    Ee = E * np.exp(-r * T)
    N1 = 0.5 * erfc(-d1/2**0.5)
    N2 = 0.5 * erfc(-d2/2**0.5)
    return S * N1 - Ee * N2
```

```
def volatilite(T, S, E, r, C0,nb_iter):
    d1 = lambda sigma : (np.log(S/E) + (r + sigma ** 2 / 2) * T) / (sigma * T**0.5)
    d2 = lambda sigma : d1(sigma) - sigma * (T ** 0.5)
    Ee = E * np.exp(-r * T)
    N1 = lambda sigma : 0.5 * erfc(-d1(sigma)/2**0.5)
    N2 = lambda sigma : 0.5 * erfc(-d2(sigma)/2**0.5)
    C = lambda sigma : S * N1(sigma) - Ee * N2(sigma)
    a = 0
    b = 1
    c = (a+b)/2
    i = 0
    while i < nb_iter:
        if C0 - C(c) > 0:
            a = c
        else:
            b = c
        c = (a+b)/2
        i += 1
    return c
```

```
def BlackScholes2(E, r, T, R, sigma, N, e):
    """ k est le pas d'espace -- h est le pas en temps -- """
    k = R / (N - 1)
    h = T / (N - 1)
    ub = lambda t : BScall(E*R, E, r, t,sigma)
    t = np.linspace(e,T-e,N)
    tau = T - t
    print(tau)
    x = np.linspace(e, R, N)
    C = np.zeros((N,N))
    A = np.zeros((N,N))
    B = np.zeros((N,N))
    for temps in range(N):
        B[-1, temps]=(r * k * x[N - 1] / (2 * h) + (k / 2)*(sigma * x[N - 1] / h) ** 2) * ub(tau[temps])
    for i in range(N):
        A[i, i] = 1 + k * (sigma * x[i] / h) ** 2 + r * k
        if i != N - 1:
            A[i,i + 1] = - r * k * x[i] / 2 * h - k / 2 * (sigma * x[i] / h) ** 2
            A[i+1, i] = r * k * x[i] / 2 * h - k / 2 * (sigma * x[i] / h) ** 2
    print(A)

    for temps in range(N):
        C[temps, -1] = ub(tau[temps])

    for temps in range(N-1):
        Cold = C[temps, :]
        X = np.linalg.solve(A, Cold + B[:, temps])
        C[temps + 1, :] = X
    return B, C
```

```
for i in range(Np):
    for j in range(duree):
        h = rd.random()
        if h < 0.5:
            for strat in range(S):
                a[i][j][strat] = -1
        else:
            for strat in range(S):
                a[i][j][strat] = 1
        A[j] += a[i][j][0]
```

```
for i in range(Np, N):
    for j in range(1,duree):
        a[i][j][0] = 0
        u[i][j][0] = u[i][j-1][0] + epsilon
        for strat in range(1,S):
            u[i][j][strat] = u[i][j-1][strat] - a[i][j-1][strat] * A[j-1]
            strat_choisie = 0
            if u[i][j][strat] > u[i][j][strat_choisie]:
                strat_choisie = strat
                s[i][j] = strat
        A[j] += a[i][j][s[i][j]]
for i in range(1, duree):
    p[i] = p[i-1]*np.exp(A[i-1]/500)
```