Projet : Volatilité implicite : Méthode Newton-Raphson

Baptiste VERMEERSCH

Ce projet a pour objectif, à partir du modèle de Black-Scholes, de calculer la volatilité implicite d'une option européenne en utilisant la méthode de Newton-Raphson afin de résoudre numériquement l'équation reliant le prix de marché de l'option à son prix théorique. Cela met en lumière la puissance des algorithmes itératifs pour l'estimation de paramètres financiers non observables.

La volatilité implicite ne peut pas être directement observée à partir du prix d'une option, elle est "cachée" derière le prix. Il va falloir résourdre cette équation :

$$C_{mrk} - C_{BS}(\sigma) = 0$$

- C_{mrk} : prix de l'option observé sur le marché
- $C_{BS}(\sigma)$: prix de l'option obtenu par Black-Scholes pour un niveau de volatilité σ donné

La méthode de Newton-Raphson consiste à approcher la racine de cette équation par l'itération :

$$\sigma_{n+1} = \sigma_n - \frac{C_{BS}(\sigma_n) - C_{mkt}}{\frac{\partial C_{BS}}{\partial \sigma}(\sigma_n)}$$

où le dénominateur représente Vega, qui correspond à la sensibilité du prix de l'option à la volatilité. Il est donné par :

$$\text{Vega} = \frac{\partial C_{BS}}{\partial \sigma} = S_0 \, \phi(d_1) \, \sqrt{T}$$

- S_0 : prix de l'actif sous-jacent en $t = 0$
- $\phi(d_1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}d_1^2}$: densité de probabilité de la loi normale standard en d1
- T : maturité de l'option

Commençons par importer les packages nécessaires :

In [2]:
```
import numpy as np
from scipy.stats import norm
```

Fixons maintenant les paramètres qui nous premettrons de calculer le prix de l'option par Black-Scholes :

In [4]:
```
S0 = 100
K = 100
r = 0.05
T = 1
sigma = 0.2 # on fixe une valeur arbitraire
C_obs = 10 # on fixe une valeur arbitraire
```

Une volatilité fixée à 20% permet généralement à l'algorithme de converger assez rapidement.

Voici le code qui implémente la méthode Newton-Raphson et qui donne directement la volatilité implicite de l'option :

In [5]:
```
max_iter = 100
epsilon = 1 * np.exp(-6)
i = 0

while i < max_iter:

    d1 = (np.log(S0/K)+(r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)

    Nd1 = norm.cdf(d1)
    Nd2 = norm.cdf(d2)
    C_bs = S0 * Nd1 - K * np.exp(-r * T) * Nd2

    theta_d1 = 1/np.sqrt(2 * np.pi) * np.exp((-d1**2)/2)
    Vega = S0 * np.sqrt(T) * theta_d1

    if abs(C_bs - C_obs) < epsilon:
```

```
        break

    sigma = sigma - ((C_bs - C_obs) / Vega)
    i = i + 1

print("Volatilité implicite : ", sigma)
```

```
Volatilité implicite :  0.187992134217771
```

Nous avons fixé le nombre maximum d'itération à 100 afin de s'assurer que l'algorithme aura le temps de converger sans problème.

L'obtention de la volatilité implicite est un point essentiel en finance quantitative car :

- les traders utilisent d'avantage la volatilité implicite d'une option plutôt que son prix pour la trater ;
- elle permet de traduire les anticipations du marché sur la volatilité future de l'actif sous-jacent ;
- elle constitue un outil central pour analyser et comparer les prix d'options ;
- elle permet de représenter et d'interpréter le volatility smile, reflet des imperfections du modèle de Black–Scholes et des préférences des investisseurs.

Ainsi, le calcul de la volatilité implicite fournit une information plus riche que la volatilité historique et constitue un indicateur incontournable dans la gestion des risques, le trading d'options et le calibrage de modèles plus sophistiqués qui prennent en compte le caractère stocastique de la volatilité (Heston, SABR...).