Université Lyon 1 - ISFA

Finance quantitative

TP - Simulation du prix d'un Call

Élèves :
Valentin Messina
Eloi Munoz

Enseignant :
Ying Jiao

3 décembre 2023

Introduction

On s'intéresse au cours de l'action LVMH entre le 23 novembre 2022 et le 23 novembre 2023. On récupère ces données journalières sur Boursorama et on peut tracer le cours de cette action sur cette période d'1 an avec $\Delta T = 1$ jour.

On cherche alors à évaluer le prix d'une option Call de maturité T, de strike K et de sous-jacent S : l'action LVMH présentée ci-dessus. On considère le taux-sans risque actuel r fourni par la BCE, considéré constant $r = 4,5\%$.

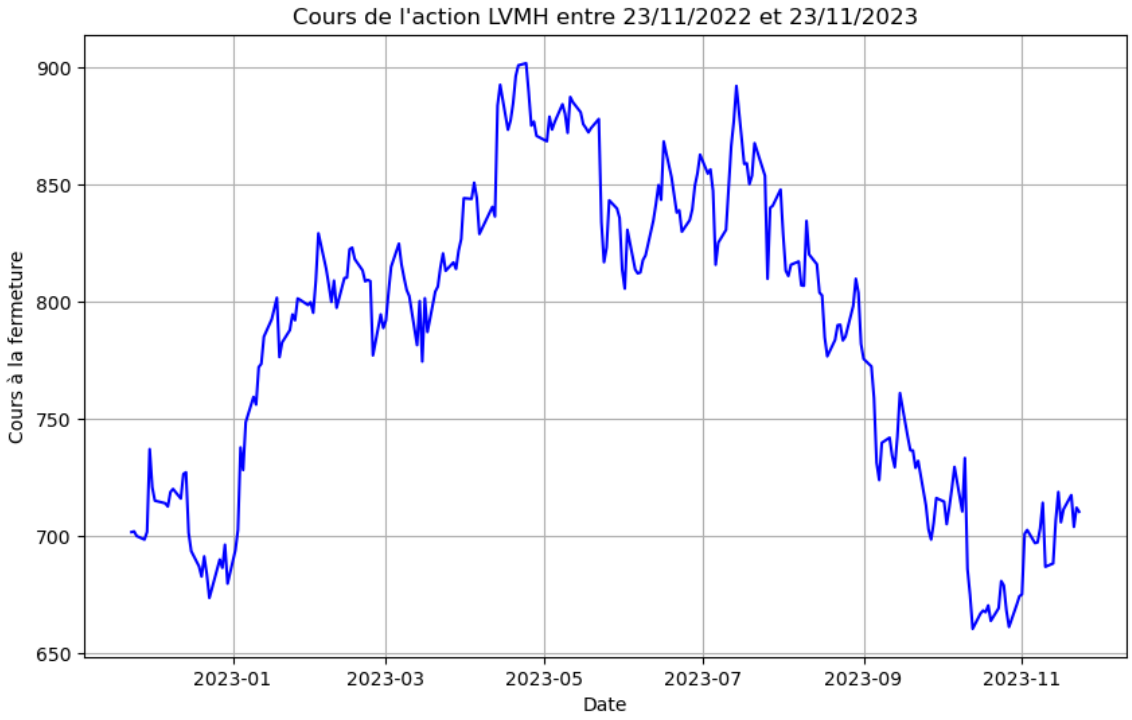

FIGURE 1 – Cours de l'action LVMH

1 Simulation du prix d'un Call par B&S

Les codes utilisés pour obtenir les figures ci-dessous sont tous disponibles en annexes.

1.1 Estimation des paramètres ν et σ

Tout d'abord, nous avons besoin d'estimer le rendement historique ν et la volatilité historique σ de ce sous-jacent. Pour ce faire, on utilise les estimateurs statistiques de la moyenne et de la variance : pour une famille de variables aléatoires indépendantes et indentiquement distribuées par une loi $\mathcal{N}(\nu\Delta t, \sigma^2\Delta t)$, on a :

$$\bar{\nu} = \frac{1}{n\Delta t}\sum_{k=1}^{n} Y_k \tag{1}$$

$$\bar{\sigma}^2 = \frac{1}{n\Delta t}\sum_{k=1}^{n}(Y_k - \bar{\nu}\Delta t)^2 \tag{2}$$

Numériquement, on obtient $\nu \approx 0.00477\%$ et $\sigma \approx 1.67\%$.

1.2 Méthode de Monte-Carlo

On choisit un strike de $K = 700$, une maturité $T = 20$ jours et on réalise une simulation de Monte-Carlo pour $N = 5000$ simulations et on obtient la figure ci-dessous.

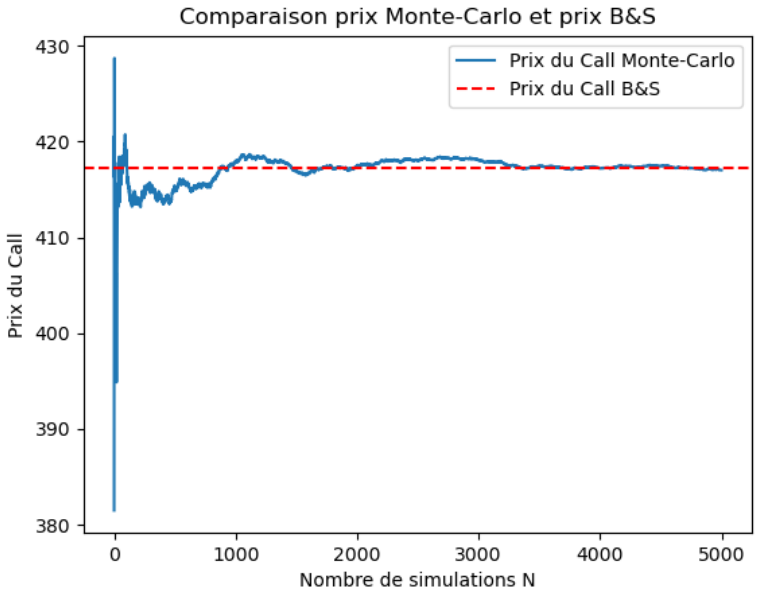

FIGURE 2 – Prix Monte-Carlo du Call $K = 700, T = 20$ j pour $N = 5000$ simulations

1.3 Comparaison formule B&S

On observe que le prix de Monte-Carlo converge bien vers le prix théorique donné par la formule de Black & Scholes. Cependant, la convergence est lente car on observe que la solution Monte-Carlo commence à être correcte au bout de $N = 4000$ simulations. Pour s'en rendre mieux compte on peut ajouter un intervalle de confiance à 5% sur le graphe.

1.4 Comportement asymptotique

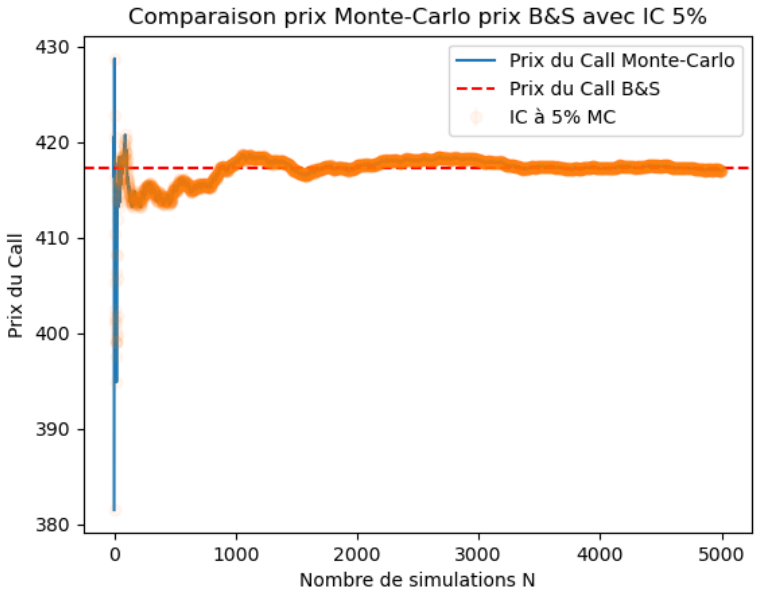

FIGURE 3 – Prix Monte-Carlo du Call $K = 700, T = 20$ j pour $N = 5000$ simulations

L'intervalle de confiance à 5% sur le graphe permet de voir que l'erreur entre le prix Monte-Carlo et théorique B&S reste bien inférieure à 5% à partir d'environ $N = 4000$ simulations.

1.5 Réduction de variance : méthode de la variable de contrôle

On va réduire la variance en appliquant la méthode de la variable de contrôle dans la formule de Black & Scholes. On cherche alors une variable aléatoire Y telle que :

$$\mathbb{E}(Y) = \mathbb{E}\Big(Y - \beta\big(X - \mathbb{E}(X)\big)\Big)$$

et telle que :

$$\mathbb{V}ar(Y) \geqslant \mathbb{V}ar\Big(Y - \beta\big(X - \mathbb{E}(X)\big)\Big)$$

On calcule alors :

$$\begin{aligned}\mathbb{V}ar\Big(Y - \beta\big(X - \mathbb{E}(X)\big)\Big) &= \mathbb{V}ar(Y) + \mathbb{V}ar\Big(\beta\big(X - \mathbb{E}(X)\big)\Big) - 2\beta\mathrm{Cov}(Y, X - \mathbb{E}(X)) \\ &= \mathbb{V}ar(X)\beta^2 - 2\beta\mathrm{Cov}(Y, X) + \mathbb{V}ar(Y)\end{aligned}$$

On cherche la valeur $\hat{\beta}$ qui minimise cette dernière quantité. Étant un polynôme du degré 2, on obtient facilement :

$$\hat{\beta} = \frac{\mathrm{Cov}(X, Y)}{\mathbb{V}ar(X)}$$

Finalement,

$$\mathbb{V}ar(X)\beta^2 - 2\beta\mathrm{Cov}(Y,X) + \mathbb{V}ar(Y) = \left(\beta - \frac{\mathrm{Cov}(X,Y)}{\mathbb{V}ar(X)}\right)^2$$

Dans le modèle de Black & Scholes, $Y = (S_T - K)^+$ et $X = S_T$. On a alors :

$$\hat{\beta_{BS}} = \frac{\mathrm{Cov}\big((S_T - K)^+, S_T\big)}{\mathbb{V}ar(S_T)}$$

où $S_T = S_0 \exp\left((r - \frac{1}{2}\sigma^2)T + \sigma W_T\right)$ et $\mathbb{V}ar(S_T) = S_0 \mathrm{e}^{2rT}(\mathrm{e}^{\sigma^2 T} - 1)$.

Finalement, on calcule $\mathrm{Cov}\big((S_T - K)^+, S_T\big)$ par simulation de Monte-Carlo avec la formule :

$$\mathrm{Cov}(X,Y) = \mathbb{E}\Big(\big(X - \mathbb{E}(X)\big)\big(Y - \mathbb{E}(Y)\big)\Big)$$

Avec ces résultats, On peut calculer C_0 avec les deux méthodes : Monte-Carlo simple et Monte-Carlo avec variable de contrôle.

Méthode simple de Monte-Carlo :

$$C_0 = \frac{1}{N}\sum_{i=1}^{n}(S_T^{(i)} - K)^+ \mathrm{e}^{-rT} \tag{3}$$

Méthode avec la variable de contrôle :

$$\tilde{C_0} = \frac{1}{N}\sum_{i=1}^{n}\mathrm{e}^{-rT}(S_T^{(i)} - K)^+ - \hat{\beta}(S_T^{(i)} - S_0\mathrm{e}^{2rT}) \tag{4}$$

On peut alors faire la simulation numérique et on obtient la Figure 4 ci-dessous.

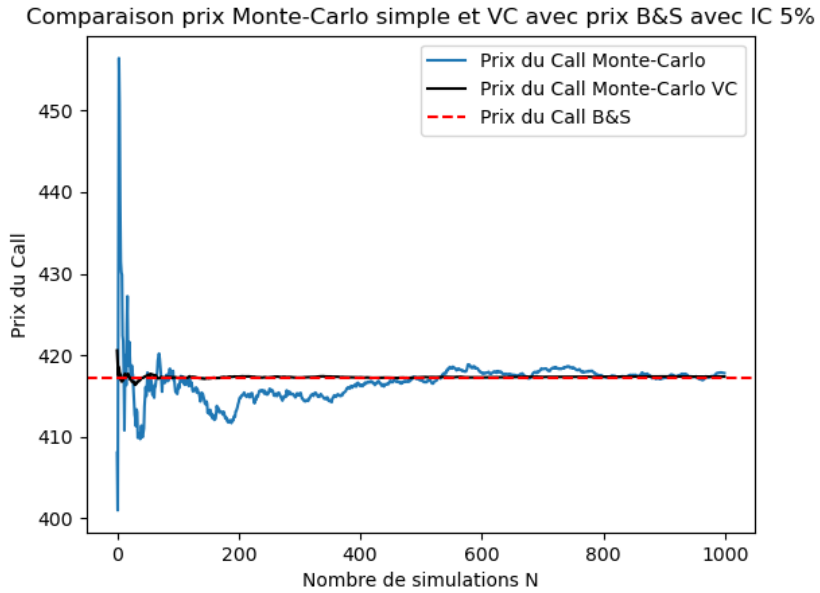

FIGURE 4 – Comparaison prix Monte-Carlo et prix Monte-Carlo avec variable de contrôle (VC) du Call $K = 700, T = 20$ j pour $N = 1000$ simulations

On observe que la convergence dans la simulation avec la variable de contrôle semble bien plus rapide. Pour s'en rendre compte, on trace uniquement la solution Monte-Carlo à variance réduite et son intervalle de confiance à 5%.

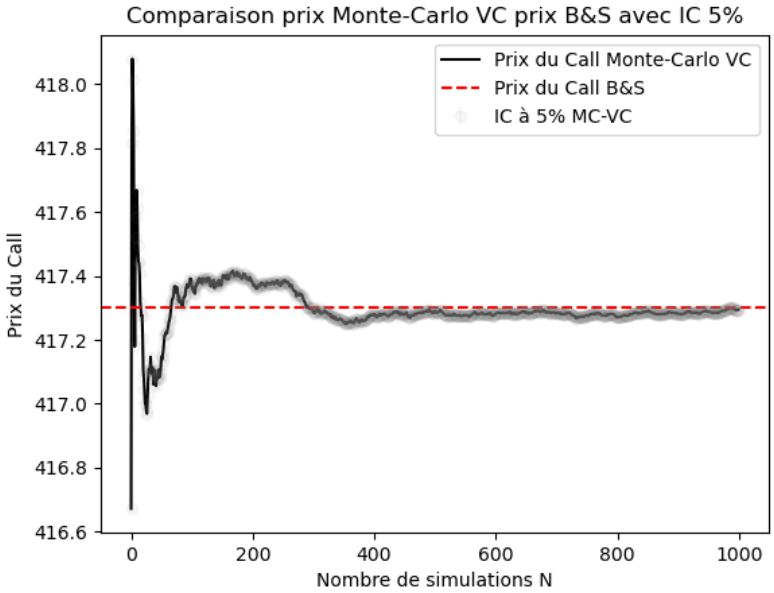

FIGURE 5 – Prix Monte-Carlo avec variable de contrôle (VC) du Call $K = 700, T = 20$ j pour $N = 100$ simulations

Ainsi, on voit qu'on rentre (et on ne sort plus) de la bande d'erreur de 5% vers environ $N = 400$ simulations : la méthode de la variable de contrôle a permis de réduire le temps de convergence d'un facteur 10.

Conclusion

En conclusion, nous avons réussi à déterminer le prix d'un Call de sous-jacent l'action LVMH grâce à la méthode de Monte-Carlo. Nous avons vu que ce prix convergeait bien vers le prix théorique donné par la formule de Black & Scholes mais nécessitait un grand nombre de simulations.

Enfin, nous avons appliqués à notre algorithme de Monte-Carlo une méthode de réduction de variance (méthode de la variable de contrôle) qui nous a permis d'obtenir une convergence 10 fois plus rapide de notre prix.

Annexes

```
r= 0.045
T= 20
deltaT= 1 #1jour

# Chemin vers le fichier texte
chemin_fichier = 'LVMH_2023-11-23.txt'

# Lire le fichier en tant que DataFrame
df = pd.read_csv(chemin_fichier, delimiter='\t')

# Extraire les colonnes de volatilit
cours = df[['clot']]
Y = np.log(cours['clot'] / cours['clot'].shift(1))

# Convertir la colonne 'clot' en un numpy array pour le tra age
closing_prices = cours['clot'].values

# Cr er un axe temporel pour l'index du DataFrame
dates = pd.to_datetime(df['date'], format='%d/%m/%Y %H:%M')

plt.figure(figsize=(10, 6))
plt.plot(dates, closing_prices, linestyle='-', color='b')
plt.title("Cours de l'action LVMH entre 23/11/2022 et 23/11/2023")
plt.xlabel("Date")
plt.ylabel("Cours    la fermeture")
plt.grid(True)
plt.show()
```

Listing 1 – Cours LVMH

```
nu=0
for i in range(1,len(cours)):
    nu+=Y[i]

nu=nu/(deltaT*len(cours))

sigma2=0
for i in range(1,len(cours)):
    sigma2+=(Y[i]-nu*deltaT)**2

sigma2 = sigma2/(deltaT*(len(cours)-1))

sigma=np.sqrt(sigma2)
```

Listing 2 – Estimation des paramètres

```
def black_scholes_call(S, K, r, T, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T
    ))
    d2 = d1 - sigma * np.sqrt(T)

    call_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return call_price[0]
```

Listing 3 – Prix théorique BS

```python
def S(T):
    return S0[0]*np.exp((r-0.5*sigma2)*T+sigma*np.sqrt(T)*stats.norm.rvs(loc=0,scale=1, size=1)[0])

def prixCallMC(K,N):
    C=[]
    prix=[]
    for i in range(N):
        C.append(np.maximum(0,S(T)-K)*np.exp(-r*T))
        prix.append(np.mean(C))
    return prix
```

Listing 4 – Prix Monte-Carlo

```python
def prixCallMC_VarContr(K, N):
    C_vc = []
    prix_vc = []
    C0 = prixCallMC(K, N)[N-1]
    betaL=[]
    beta=0

    for i in range(N):
        ST = S(T)
        varianceST = S0*S0*np.exp(2*r*T)*(np.exp(sigma2*T)-1)
        betaL.append((ST - S0 * np.exp(r * T)) * (np.exp(-r * T) *np.maximum(0, ST - K) - C0 )  )
        beta=np.mean(betaL)/varianceST

    for i in range(N):
        ST = S(T)
        C_vc.append( np.exp(-r * T)*np.maximum(0, ST - K)- beta*(ST-S0*np.exp(r * T)) )
        prix_vc.append(np.mean(C_vc))

    return prix_vc
```

Listing 5 – Prix Monte-Carlo variable de contrôle

```python
def prixCallMC_IC(K, N):
    C=[]
    prix=[]
    for i in range(N):
        C.append(np.maximum(0,S(T)-K)*np.exp(-r*T))
        prix.append(np.mean(C))

    # Calculate mean and standard deviation for confidence intervals
    mean_prix = np.mean(prix)
    std_prix = np.std(prix, ddof=1)  # Use ddof=1 for sample standard deviation

    # Calculate confidence intervals
    confidence_interval = 1.96 * (std_prix / np.sqrt(N))

    return prix, confidence_interval

def prixCallMC_VarContr_IC(K, N):
    C_vc = []
    prix_vc = []
```

```
    C0 = prixCallMC(K, N)[N-1]
    betaL=[]
    beta=0

    for i in range(N):
        ST = S(T)
        varianceST = S0*S0*np.exp(2*r*T)*(np.exp(sigma2*T)-1)
        betaL.append((ST - S0 * np.exp(r * T)) * (np.exp(-r * T) *np.maximum(0, ST - K) - C0 )  )
        beta=np.mean(betaL)/varianceST

    for i in range(N):
        ST = S(T)
        C_vc.append( np.exp(-r * T)*np.maximum(0, ST - K)- beta*(ST-S0*np.exp(r * T)) )
        prix_vc.append(np.mean(C_vc))

    # Calculate mean and standard deviation for confidence intervals
    mean_prix_vc = np.mean(prix_vc)
    std_prix_vc = np.std(prix_vc, ddof=1)  # Use ddof=1 for sample standard deviation

    # Calculate confidence intervals
    confidence_interval_vc = 1.96 * (std_prix_vc / np.sqrt(N))

    return prix_vc, confidence_interval_vc
```

Listing 6 – Intervalles de confiance et prix Monte-Carlo et Monte-Carlo VC