Université Paris 1 Panthéon Sorbonne

UFR27- Malliavin Calculus

Projet B

Zhen MENG, Ndoumbé BAYO

Mars 2025

DM pour Monte Carlo

Zhen MENG et Ndoumbé BAYO

March 2025

1 Part A : An integration by part formula

(a)

D'après le cours 4, page 11.

nous avons **Proposition 2:** Let $\mathbf{F} = (F_1, \ldots, F_n) \in \mathbb{D}^n$ and $\varphi : \mathbb{R}^n \to \mathbb{R}$ of class $\mathcal{C}^1$ and Lipschitz then

$$\varphi(\mathbf{F}) \in \mathbb{D} \text{ and } D_t\varphi(\mathbf{F}) = \sum_{i=1}^{n} \frac{\partial \varphi}{\partial x_i}(F) D_t F_i.$$

Nous voulons prouver que

$$E\left[\langle DZ, u\rangle_{L^2([0,T])}\right] = E\left[\Phi'(X)Y\right],$$

où $Z = \Phi(X)$.

1er étape: Règle de la chaîne pour la dérivée de Malliavin :

Puisque $X \in \mathbb{D}$, $\Phi \in \mathcal{C}^1 \cap \mathrm{Lip}(\mathbb{R}, \mathbb{R})$ ($\mathcal{C}^1$ and Lipschitz), on peut appliquer la proposition 2 du cours:

$$Z = \Phi(X) \in \mathbb{D}$$

la dérivée de Malliavin de Z est donnée par:

$$D_t\Phi(X) = D_tZ = \Phi'(X)D_tX.$$

2ème étape: :

$$\langle DZ, u\rangle = \int_0^T D_tZ \cdot u(t)\, dt.$$

En substituant $D_tZ = \Phi'(X)D_tX$ et $u(t) = \frac{h(t)Y}{\int_0^T h(s)D_sX\, ds}$, nous obtenons

$$\langle DZ, u\rangle = \int_0^T \Phi'(X)D_tX \cdot \frac{h(t)Y}{\int_0^T h(s)D_sX\, ds}\, dt.$$

Le terme $\int_0^T h(s)D_sX\,ds$ est une constante par rapport à t, donc il peut être factorisé hors de l'intégrale :

$$\langle DZ, u\rangle = \Phi'(X)Y \cdot \frac{\int_0^T h(t)D_tX\,dt}{\int_0^T h(s)D_sX\,ds}.$$

Puisque le numérateur et le dénominateur sont identiques, ils s'annulent, ce qui donne

$$\langle DZ, u\rangle = \Phi'(X)Y.$$

3ème étape: :

Nous avons établi l'égalité suivante :

$$\langle DZ, u\rangle_{L^2([0,T])} = \Phi'(X)Y.$$

En prenant l'espérance des deux côtés, nous obtenons :

$$E\big[\langle DZ, u\rangle_{L^2([0,T])}\big] = E\big[\Phi'(X)Y\big].$$

(b)

D'après le cours 5, page 3: The Skorohod Integral

Definition [4] We denote by δ the adjoint operator of $D : \mathbb{D} \to L^2(\Omega \times [0,T])$:

δ is a (linear) operator from $\mathrm{dom}(\delta) \subset L^2(\Omega \times [0,T])$ with values in $L^2(\Omega)$ such that:

(i) $u \in \mathrm{dom}(\delta)$, if $\exists C \in \mathbb{R}_+$, $\forall F \in \mathbb{D}$,

$$\big|E[\langle DF, u\rangle_{L^2([0,T])}]\big| \leq C\|F\|_{L^2(\Omega)}$$

(ii) When $u \in \mathrm{dom}(\delta)$, $\delta(u)$ is the unique element of $L^2(\Omega)$ such that $\forall F \in \mathbb{D}$,

$$E[\langle DF, u\rangle_{L^2([0,T])}] = E[F\delta(u)].$$

D'après l'énoncé, on trouve

$$u(t) = \frac{h(t)Y}{\int_0^T h(t)D_tX\,dt} \in dom(\delta). \tag{2}$$

donc on en déduit que

$\forall Z \in \mathbb{D}$,

$$E[\langle DZ, u\rangle_{L^2([0,T])}] = E[Z\delta(u)].$$

A l'aide du résultat de la question a): on a aussi

$$E\left[\langle DZ, u\rangle_{L^2([0,T])}\right] = E\left[\Phi'(X)Y\right],$$

donc on en déduit que

$$E[\Phi(X)\delta(u)] = E[\Phi'(X)Y].$$

2 Part B : Delta for asian options

Dans les modèles financiers classiques, les prix sont des espérances sous la mesure martingale équivalente, à une constante près. On appelle Greeks les dérivées des prix. Pour calculer le prix, on peut utiliser la méthode de Monte Carlo, et pour les Greeks, on peut également utiliser la méthode de Monte Carlo ainsi que la méthode des différences finies.

Cependant, nous avons vu en cours qu'une utilisation combinée des deux méthodes soulève un dilemme. En effet, pour réduire l'erreur due au schéma de différences finies, il faut supposer que soit petit. Mais si nous voulons estimer des quantités avec la méthode de Monte Carlo, cela entraîne une forte augmentation de la variance de ce qui est à l'intérieur de l'espérance concernée, ce qui accroît l'erreur provenant de la méthode de Monte Carlo.

Notre objectif est d'exprimer le delta sans utiliser les dérivées, en l'écrivant comme une espérance d'une quantité qui ressemble aux prix mais multipliée par une variable aléatoire que nous appellerons le poids.

1. On sait que

$$\begin{aligned}\Delta_0(x) =& \frac{dP(x)}{dx} \\ =& e^{-rT}\frac{d}{dx}\mathbb{E}\left[\Phi\left(\int_0^T X_u^x du\right)\right]\end{aligned}$$

La fonction est de classe $C^1(\mathbb{R},\mathbb{R})$, ce qui nous permet d'utiliser le théorème de dérivation de Lebesgue. Ce théorème nous autorise à inverser l'espérance et la dérivation.

$$\Delta_0(x) = e^{-rT}\mathbb{E}\left[\frac{d}{dx}\Phi\left(\int_0^T X_u^x du\right)\right]$$

Proposition 1 (Chain Rule (proposition 2 du cours))**.**

Let $F = (F_1, \ldots, F_n) \in \mathbb{D}^n$ and $\varphi : \mathbb{R}^n \to \mathbb{R}$ of class C^1 and Lipschitz then

$$\varphi(F) \in \mathbb{D} \text{ and } D_t\varphi(F) = \sum_{i=1}^{n} \frac{\partial \varphi}{\partial x_i}(F) D_t F_i.$$

Nous utilisons la proposition ci-dessus du cours qui nous dit que, puisque Φ est de classe $C^1(\mathbb{R},\mathbb{R})$ et Lipschitz, nous pouvons appliquer la chain rule. De plus, nous écrivons f' puisque la fonction est régulière.

$$\Delta_0(x) = e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_u^x du\right)\int_0^T Y_u^x du\right]$$

où $Y_u^x = \frac{dX_u^x}{dx}$ est le *first variation process.*
Nous avons défini l'actif risqué comme suit :

$$dX_t^x = rX_t^x dt + \sigma X_t^x dB_t, \quad X_0^x = x$$

Nous avons vu au premier semestre que la solution de cette SDE est:

$$X_t^x = xe^{(r-\frac{\sigma^2}{2})t+\sigma Bt}$$

Ainsi,

$$Y_t^x = \frac{X_t^x}{dx} = e^{(r-\frac{\sigma^2}{2})t+\sigma Bt} = \frac{X_t^x}{x}$$

Par conséquent,

$$\Delta_0(x) = \frac{e^{-rT}}{x}\mathbb{E}\left[\Phi'\left(\int_0^T X_u^x du\right)\int_0^T X_u^x du\right]$$

2. Nous avons prouvé dans la partie précédente du devoir que

$$\mathbb{E}[\Phi(X)\delta(u)] = \mathbb{E}[\Phi'(X)Y]$$

où

$$u(t) = \frac{h(t)Y}{\int_0^T h(t)D_t X\, dt} \in \mathrm{dom}(\delta)$$

Nous allons utiliser cette formule pour réexprimer $\Delta(x)$.

$$\Delta(x) = \frac{e^{-rT}}{x}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x\, ds\right)\int_0^T X_s^x\, ds\right] = \frac{e^{-rT}}{x}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x\, ds\right)\delta(u)\right]$$

Dans notre cas, on a

$$X = \int_0^T X_s^x\, ds \quad ; \quad Y = \int_0^T X_s^x\, ds \quad ; \quad u(t) = \frac{Y}{\int_0^T D_t X\, dt} = \frac{\int_0^T X_s^x\, ds}{\int_0^T D_t X\, dt}$$

On doit maintenant simplifier au maximum notre calcul

$$\begin{aligned}\int_0^T D_t X\, dt =& \int_0^T \left(\int_t^T \sigma X_s^x\, ds\right) dt \\ =& \sigma \int_0^T \int_t^T X_s^x\, dsdt\end{aligned}$$

On utilise Fubini pour inverser l'ordre d'intégration et on obtient

$$\begin{aligned}\int_0^T D_t X\,dt =& \sigma \int_0^T \int_0^s X_s^x\,dt\,ds \\ =& \sigma \int_0^T sX_s^x\,ds\end{aligned}$$

De ce fait, on a finalement

$$u(t) = \frac{\int_0^T X_s^x\,ds}{\sigma \int_0^T sX_s^x\,ds}$$

Proposition 2 (Proposition 7 du cours)**.**

If $u \in dom(\delta)$ and $F \in \mathcal{D}$ such that $Fu \in dom(\delta)$ then

$$\delta(Fu) = F\delta(u) - \int_0^T D_t F\,u_t\,dt.$$

On remarque que $u(t) = Fv$

$$\begin{aligned}F =& \frac{1}{\sigma \int_0^T sX_s^x\,ds} \\ v =& \int_0^T X_s^x\,ds\end{aligned}$$

On applique donc la proposition précédente et on obtient

$$\begin{aligned}\delta(u) =& F \cdot \delta(v) - v \cdot \int_0^T D_t F\,dt \\ =& F \cdot v \cdot B_T - v \cdot \int_0^T D_t F\,dt \\ =& \frac{\int_0^T X_s^x\,ds}{\sigma \int_0^T sX_s^x\,ds} \cdot B_T - v \cdot \int_0^T D_t F\,dt\end{aligned}$$

On a

$$D_t F = -\frac{1}{\sigma} \cdot \frac{D_t(\int_0^T sX_s^x\,ds)}{(\int_0^T sX_s^x\,ds)^2}$$

et

$$D_t\left(\int_0^T sX_s^x\,ds\right) = \int_t^T sD_t(X_s^x)\,ds = \sigma\int_t^T sX_s^x\,ds$$

ce qui nous donne

$$\int_0^T D_tF\,dt = -\frac{1}{(\int_0^T sX_s^x\,ds)^2}\int_0^T\int_t^T sX_x^s\,ds\,dt$$

On utilise une fois encore le théorème de Fubni pour inverser le sens des intégrales

$$\begin{aligned}\int_0^T D_tF\,dt &= -\frac{1}{(\int_0^T sX_s^x\,ds)^2}\int_0^T\left(\int_0^s dt\right) sX_x^s\,ds\\ &= -\frac{1}{(\int_0^T sX_s^x\,ds)^2}\int_0^T s^2X_x^s\,ds\end{aligned}$$

Il suit que

$$\int_0^T D_tF\,dt = -\frac{\int_0^T s^2X_x^s\,ds}{\left(\int_0^T sX_x^s\,ds\right)^2}.$$

Par conséquent,

$$\delta(u) = \frac{\int_0^T X_x^s\,ds}{\int_0^T sX_x^s\,ds}\left[\frac{B_T}{\sigma}+\frac{\int_0^T s^2X_x^s\,ds}{\int_0^T sX_x^s\,ds}\right]$$

3. On suppose comme précédemment que l'on peut appliquer les résultats de la partie A et on pose. En appliquant Fubini, on obtient $h(t) = S_t = X_s^x$

$$\begin{aligned}u(t) =&\frac{X_x^t\cdot\int_0^T X_s^x\,ds}{\int_0^T X_t^x\cdot D_t(\int_0^T X_x^s)\,dt}\\ =&\frac{X_x^t\cdot\int_0^T X_s^x\,ds}{\sigma\int_0^T X_t^x\left(\int_t^T X_x^s\,ds\right)\,dt}\\ =&\frac{X_x^t\cdot\int_0^T X_s^x\,ds}{\sigma\int_0^T X_x^s\left(\int_0^s X_x^t\,dt\right)\,ds}\end{aligned}$$

Comme dans le cours, on pose

$$g(s) = \int_0^s X_x^t \, dt \quad ; \quad g'(s) = X_x^s ds$$

On réalise une intégration par parties

$$\int_0^T \left(\int_0^s X_x^t \, dt \right) X_x^s \, ds = \left(\int_0^s X_t^x dt \right)^2 \Bigg|_0^T - \int_0^T \left(\int_0^s X_t^x dt \right) X_s^x ds$$

ce qui implique que

$$2 \cdot \int_0^T \left(\int_0^s X_t^x dt \right) X_s^x ds = \left(\int_0^s X_t^x dt \right)^2 \Bigg|_0^T$$

Par conséquent,

$$u(t) = \frac{X_t^x \cdot \int_0^T X_t^x \, dt}{\frac{\sigma}{2} (\int_0^T X_t^x \, dt)^2} = \frac{2X_t^x}{\sigma \int_0^T X_t^x \, dt}$$

Comme la question précédente, on utilise la proposition 7 du cours pour calculer $\delta(Fu)$

$$\delta(Fu) = F \cdot \delta(u) - \int_0^T D_t F \cdot u(t) \, dt$$

On a

$$\begin{aligned} \delta(u) =& \frac{2\delta(X_t^x)}{\sigma \int_0^T X_t^x \, dt} \\ =& \frac{2 \int_0^T X_t^x \, dBt}{\sigma \int_0^T X_t^x \, dt} \end{aligned}$$

On a défini la dynamique de l'actif risqué comme

$$dX_t^x = rX_t^x dt + \sigma X_t^x dB_t \implies \sigma X_t^x dB_t = dX_t^x - rX_t^x dt$$

D'après ce qui précéde, on peut écrire

$$\int_0^T X_t^x dB_t = \frac{1}{\sigma} \left(X_T^x - x - r \int_0^T X_t^x dt \right)$$

On prend

$$F = \frac{1}{(\int_0^T X_t^x \, dt)^2}$$

On a

$$D_t F = -\frac{D_t \left(\int_0^T X_t^x \, dt \right)}{\left(\int_0^T X_t^x \, dt \right)^2} = -\frac{\sigma \int_t^T X_s^x \, ds}{(\int_0^T X_s^x \, ds)^2}$$

Ainsi, en se basant sur ce qui précéde

$$\int_0^T D_t F \cdot X_t^x \, dt = -\frac{\sigma}{(\int_0^T X_t^x \, dt)^2} \int_0^T X_t^x \left(\int_t^T X_s^x \, ds \right) dt = -\frac{\sigma}{2}$$

On a donc

$$\delta(Fu) = \frac{1}{\int_0^T X_t^x \, dt} \cdot \frac{X_T^x - x - r \cdot (\int_0^T X_t^x \, dt)}{\sigma} + \frac{\sigma}{2}$$

On multiplie par $\frac{2}{\sigma}$ et on obtient

$$\Pi_2 = \frac{2}{\sigma^2} \left(\frac{X_T^x - x}{\int_0^T X_t^x \, dt} - r \right) + 1$$

ce qu'il fallait démontrer.

4. On sait que à $t = 0$

$$\text{Greek} = \mathbb{E}^* \left[\Phi \left(\int_0^T X_s^x \, ds \right) \Pi \right]$$

Il serait intéressant de savoir ce qui se passe lorsque nous avons plusieurs candidats. En effet, comme nous l'avons vu précédemment, il peut exister différents poids satisfaisant cette propriété. Une façon d'aborder ce problème est de résoudre le problème suivant.

$$\min_{\Pi \text{ tel que } \text{Greek } = \mathbb{E}^*[\Phi(\int_0^T X_s^x \, ds)\Pi]} \Pi$$

En effet, la qualité d'une approximation de Monte-Carlo dépendra de la variance de ce que nous avons à l'intérieur de l'espérance. Pour un Π arbitraire, le poids optimal est donné par.

$$\Pi_0 = \mathbb{E}^* \left[\Pi | \sigma \left(\int_0^T X_s^x \, ds \right) \right]$$

Prouvons le. On définit $\tilde{\Pi}$ tel que Greek= $\mathbb{E}^*[\Phi(\int_0^T X_s^x\,ds)\tilde{\Pi}]$

$$
\begin{aligned}
var\left(\Phi(\int_0^T X_s^x\,ds)\Pi\right) =&\mathbb{E}^*\left[\left(\Phi(\int_0^T X_s^x\,ds)\tilde{\Pi} - \text{ Greek }\right)^2\right] \\
&= \mathbb{E}^*\left[\Phi(\int_0^T X_s^x\,ds)(\tilde{\Pi}-\Pi_0)^2\right] + \mathbb{E}\left[\left(\Pi_0\Phi(\int_0^T X_s^x\,ds) - \text{Greek}\right)^2\right] + \\
&2\mathbb{E}^*\left[\mathbb{E}^*\left[\Phi(\int_0^T X_s^x\,ds)(\Pi_0\Phi(\int_0^T X_s^x\,ds) - \text{ Greek })(\tilde{\Pi}-\Pi_0)|\sigma(\int_0^T X_s^x\,ds)\right]\right] \\
&= \mathbb{E}^*\left[\Phi(\int_0^T X_s^x\,ds)^2(\tilde{\Pi}-\Pi_0)^2\right] + \mathbb{E}\left[\left(\Pi_0\Phi(\int_0^T X_s^x\,ds) - \text{Greek}\right)^2\right] + \\
&2\mathbb{E}^*\left[\Phi(\int_0^T X_s^x\,ds)(\Pi_0\Phi(\int_0^T X_s^x\,ds) - \text{ Greek })\mathbb{E}^*\left[(\tilde{\Pi}-\Pi_0)|\ \sigma\left(\int_0^T X_s^x\,ds\right)\right]\right] \\
&= \mathbb{E}^*\left[\Phi(\int_0^T X_s^x\,ds)(\tilde{\Pi}-\Pi_0)^2\right] + \mathbb{E}^*\left[\Phi(\int_0^T X_s^x\,ds)(\Pi-\Pi_0)^2\right] \\
=&\mathbb{E}^*\left[\Phi(\int_0^T X_s^x\,ds)^2(\tilde{\Pi}-\Pi_0)^2\right] + var\left(\Pi_0\Phi(\int_0^T X_s^x\,ds)\right)]
\end{aligned}
$$

La variance est minimal quand

$$
\mathbb{E}^*\left[\Phi(\int_0^T X_s^x\,ds)^2(\tilde{\Pi}-\Pi_0)^2\right] = 0
$$

En d'autres termes, elle est minimal quand $\tilde{\Pi} = \Pi_0$

5. Π_0 est indépendant du choix de Π. En effet, si on suppose que $\tilde{\Pi_1}$ et $\tilde{\Pi_2}$ satisfont Greek= $\mathbb{E}^*[\Phi(\int_0^T X_s^x\,ds)\tilde{\Pi}]$. Alors, pour tout f,

$$
\text{Greek } = \mathbb{E}^*[\Phi(\int_0^T X_s^x\,ds)\tilde{\Pi_1}] = \mathbb{E}^*[\Phi(\int_0^T X_s^x\,ds)\tilde{\Pi_2}]
$$

Nous avons, par définition de l'espérance conditionnelle,

$$
\mathbb{E}^*\left[\tilde{\Pi_1}|\ \sigma\left(\int_0^T X_s^x\,ds\right)\right] = \mathbb{E}^*\left[\tilde{\Pi_2}|\ \sigma\left(\int_0^T X_s^x\,ds\right)\right]
$$

Π_0 est ainsi indépendant du choix d'un tel $\tilde{\Pi}$. Les poids Π_1 et Π_2 sont optimaux dans ce sens. Il est inutile de chercher d'autres candidats car ils minimisent la variance.

3 Part C : Numerical part

a)

Nous voulons utiliser la méthode de simulation de Monte-Carlo afin d'évaluer le prix d'une option asiatique de type call et d'une option asiatique à barrière. En simulant les trajectoires du prix de l'actif sous-jacent, il calcule l'intégrale de la valeur moyenne au fil du temps, puis utilise cette information pour estimer le prix de l'option. Et puis etudions leurs intervalle de confiance et la convergence.

Idée: Simulation de Monte Carlo

- 1er étape: Génération de N trajectoires d'actifs, calcul de l'intégrale et des paiements pour chaque trajectoire.
  - Paramètres Initiaux
    - Prix initial de l'actif : $x = 100$
    - Taux sans risque : $r = 0.03$
    - Volatilité : $\sigma = 0.2$
    - Maturité : $T = 1$ an
    - Strike de l'option call : $K_1 = 100$
    - Niveau de la barrière : $K_2 = 110$
    - Nombre de pas de temps : $M = [50, 150, 250]$
    - Nombre de simulations : $N = [1000, 3000, \dots, 51000]$
  - Simulation des Trajectoires

    Dans le modèle du Black-Scholes sous la probabilité Q, Le prix de l'actif sous-jacent X_t suit un mouvement brownien géométrique :

    $$dX_t = rX_t\,dt + \sigma X_t\,dW_t$$

    où W_t est un mouvement brownien standard.

    D'après lemme d'ito , on trouve la solution analytique:

    $$X_T = X_t \exp\left(\left(r - \frac{1}{2}\sigma^2\right)(T-t) + \sigma(W_T - W_t)\right)$$

    avec $(W_T - W_t) \sim \mathcal{N}(0, T-t)$

    On définit la variable Z comme suit :

    $$Z = \frac{W_T - W_t}{\sqrt{T-t}} \sim \mathcal{N}(0,1)$$

    On reécrit:

    $$X_T = X_t \exp\left(\left(r - \frac{1}{2}\sigma^2\right)(T-t) + \sigma\sqrt{T-t}\,Z\right)$$

Pour simuler les trajectoires, on discrétise l'intervalle de temps $[0, T]$ en M pas de temps, notamment $\Delta t = \frac{T}{M}$. À chaque pas de temps, le prix de l'actif est mis à jour selon :

$$X_{t+\Delta t} = X_t \exp\left(\left(r - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}\, Z\right)$$

```python
import numpy as np
import matplotlib.pyplot as plt

# Paramètres initiaux
x = 100          # Prix initial de l'actif
r = 0.03         # Taux sans risque
sigma = 0.2      # Volatilité
T = 1            # Temps jusqu'à l'échéance (années)
K1 = 100         # Prix d'exercice du call
K2 = 110         # Barrière supérieure
N = 51000        # Nombre de simulations

# Définir la graine aléatoire pour la reproductibilité
np.random.seed(42)


def geo_brownian(M ,N ,T,x,r,sigma):

    dt = T / M
    S_path = np.zeros((M+1,N))
    S_path[0] = x

    for t in range(1,M+1):
        Z = np.random.standard_normal(N)
        S_path[t] = S_path[t - 1] * np.exp((r-0.5*sigma**2)*dt +sigma*np.sqrt(dt)*Z)
    return S_path

```

– 2ème étape: Calcul des Prix des Options

Nous avons cette formule d'approximation de la méthode de Monte-Carlo:

$$\int_0^T X_s\, ds \approx \frac{T}{M}\sum_{k=0}^{M} X_{\frac{kT}{M}} \approx \frac{T}{M}\left(X_0 + X_1 + \cdots + X_T\right).$$

où le pas de temps est $M \in \{50, 150, 250\}$.

Option Call

$$\text{Payoff}_{\text{call}} = \left(\int_0^T X_s\, ds - K_1\right)_+ = \max\left(\int_0^T X_s\, ds - K_1,\ 0\right).$$

$$\text{Price} = e^{-rT} \cdot \mathbb{E}^*[\text{Payoff}_{\text{call}}] \;\approx\; e^{-rT}\, \frac{1}{N} \sum_{k=1}^{N} Payoff_i,$$

où $\mathbb{E}^*$ désigne l'espérance sous la mesure risque-neutre, et Payoff_i est le gain de la i-ième trajectoire.

```python

def simulate_asian_Call(T, M, r, sigma, x, N, K1):
    S_path = geo_brownian(M, N ,T,x,r,sigma)
    S_int = (T/M) * S_path.sum(axis=0)
    payoff = np.maximum(S_int - K1, 0)
    Call_price = np.exp(-r*T) * payoff.mean()
    return Call_price

# Nombre de pas de temps
M1 = 50
M2 = 150
M3 = 250

print(simulate_asian_Call(T, M1, r, sigma, x, N, K1))
print(simulate_asian_Call(T, M2, r, sigma, x, N, K1))
print(simulate_asian_Call(T, M3, r, sigma, x, N, K1))


# Résultats
6.424558752704621
5.666959019074447
5.526127495150566
```

Option Barrière

$$\text{Payoff}_{\text{barrière}} = 1_{K_1 < \int_0^T X_s\,ds < K_2} = \begin{cases} 1 & \text{si } K_1 < \displaystyle\int_0^T X_s\,ds < K_2, \\ 0 & \text{sinon.} \end{cases}$$

$$\text{Price} = e^{-rT} \cdot \mathbb{E}^*[\text{Payoff}_{\text{barrière}}] \;\approx\; e^{-rT}\, \frac{1}{N} \sum_{k=1}^{N} Payoff_i,$$

```python

def simulate_asian_barriere(T, M, r, sigma, x, N, K1, K2):
    S_path = geo_brownian(M, N ,T,x,r,sigma)
    S_int = (T/M) * S_path.sum(axis=0)
    condition = (S_int > K1) & (S_int < K2)
    barriere_price = np.exp(-r*T)* condition.mean()
    return barriere_price


```

```
M1 = 50
M2 = 150
M3 = 250

print(simulate_asian_barriere(T, M1, r, sigma, x, N, K1, K2))
print(simulate_asian_barriere(T, M2, r, sigma, x, N, K1, K2))
print(simulate_asian_barriere(T, M3, r, sigma, x, N, K1, K2))

# Résultats
0.30814499941734397
0.2968421632030731
0.29665187976512236

```

– 3ème étape: Variance empirique :

Espérance :

$$\mathbb{E}[\text{Price}] = e^{-rT} \cdot \mathbb{E}\left[\frac{1}{N}\sum_{k=1}^{N} \text{Payoff}_i\right] = e^{-rT}\mathbb{E}[\text{Payoff}]$$

Variance :

$$\text{Var}(\text{Price}) = e^{-2rT} \cdot \text{Var}\left(\frac{1}{N}\sum_{i=1}^{N} \text{Payoff}_i\right) = e^{-2rT} \cdot \frac{\sigma^2_{\text{Payoff}}}{N},$$

où $\sigma^2_{\text{Payoff}} = \text{Var}(\text{Payoff})$ est la variance des gains d'une seule trajectoire.

– 4ème étape: Intervalle de confiance de 95%:

Nous appliquons le Théorème Central Limite (CLT) pour déduire l'intervalle de confiance:

Lorsque $N \to \infty$, la moyenne empirique suit une distribution normale :

$$\frac{1}{N}\sum_{i=1}^{N} \text{Payoff}_i \sim \mathcal{N}\left(\mathbb{E}[\text{Payoff}], \frac{\sigma^2_{\text{Payoff}}}{N}\right).$$

Par conséquent, la distribution de l'estimateur du prix est donnée par :

$$\text{Price} \sim \mathcal{N}\left(e^{-rT}\mathbb{E}[\text{Payoff}], \frac{e^{-2rT}\sigma^2_{\text{Payoff}}}{N}\right).$$

Pour un niveau de confiance **95%**, nous utilisons la propriété suivante de la loi normale :

$$\frac{price - e^{-rT}\mathbb{E}[\text{Payoff}]}{\frac{\sqrt{e^{-2rT}} * \sigma_{\text{Payoff}}}{\sqrt{N}}} \sim \mathcal{N}(0,1)$$

Grace au cours 2, page 52. Nous trouvons cette formule:

$$\mathit{price} \in \left[e^{-rT}\frac{1}{N}\sum_{i=1}^{N}\text{Payoff}_i - \frac{1.96e^{-2rT}\Sigma}{\sqrt{N}}, e^{-rT}\frac{1}{N}\sum_{i=1}^{N}\text{Payoff}_i + \frac{1.96e^{-2rT}\Sigma}{\sqrt{N}} \right]$$

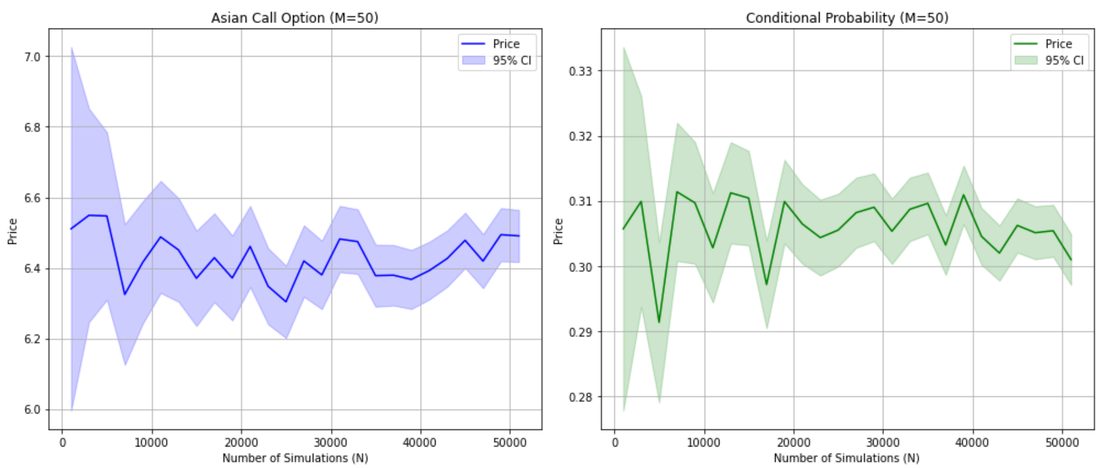
Avec

$$\Sigma^2 \quad \text{being the (empirical) variance of } \mathit{price}.$$

```

#Calculer la moyenne, la variance et l'intervalle de
↪  confiance à 95 %
def compute_statistics(payoffs, discount_factor):

    mean = np.mean(payoffs) * discount_factor
    variance = np.var(payoffs, ddof=1) * (discount_factor**2)
    std_err = np.std(payoffs, ddof=1) * discount_factor
    ↪  \np.sqrt(len(payoffs))
    ci_low = mean - 1.96 * std_err
    ci_high = mean + 1.96 * std_err
    return mean, variance, (ci_low, ci_high)



def analyze_convergence(M_fixed, N_steps,
↪  payoff_type='call'):

    discount = np.exp(-r*T)
    means, variances, ci_lows, ci_highs = [], [], [], []

    for n in N_steps:
        S_path = geo_brownian(M_fixed, n, T, x, r, sigma)

        if payoff_type == 'call':
            S_int = (T/M_fixed) * S_path.sum(axis=0)
            payoffs = np.maximum(S_int - K1, 0)
        else:
            S_int = (T/M_fixed) * S_path.sum(axis=0)
            payoffs = ((S_int > K1) & (S_int <
            ↪  K2)).astype(float)

        mean, var, ci = compute_statistics(payoffs, discount)
        means.append(mean)
        variances.append(var)
        ci_lows.append(ci[0])
        ci_highs.append(ci[1])

    return means, variances, ci_lows, ci_highs


```

```
N_steps = np.arange(1000, 51001, 2000)

#Exemples M = 50
call_means, call_vars, call_ci_low, call_ci_high = analyze_convergence(M_fixed=50, N_steps=N_steps, payoff_type='call')
cond_means, cond_vars, cond_ci_low, cond_ci_high = analyze_convergence(M_fixed=50, N_steps=N_steps, payoff_type='barrier')


```

– Convergence



Nous observons que :
1) À mesure que le nombre de simulations (N) augmente de 0 à 51 000, le prix de l'option (Prix) converge progressivement. Au stade initial (lorsque N est petit), le prix peut fluctuer considérablement ; à mesure que N augmente, le prix a tendance à se stabiliser, indiquant que l'estimation de Monte Carlo est proche du véritable prix théorique. C'est le cœur de la loi des grands nombres : lorsque le nombre d'expériences répétées indépendantes est suffisamment grand, la moyenne de l'échantillon converge vers la valeur théorique attendue.
2) L'intervalle de confiance à 95% indiqué dans l'image devient progressivement plus étroit à mesure que N augmente.
Cela est dû au fait que l'erreur de la méthode de Monte-Carlo suit généralement une convergence de l'ordre de :

$$\frac{1}{\sqrt{N}}$$

Plus le nombre de simulations est élevé, plus l'incertitude de l'estimation est faible.

3) Lorsque N dépasse une certaine valeur (par exemple 30 000 ou 40 000), le prix et l'intervalle de confiance peuvent ne plus montrer de changements significatifs. Cela signifie qu'à ce stade, augmenter davantage le nombre de simulations a un effet limité sur l'amélioration de la précision.

b)

- Δ mesure la sensibilité du prix par rapport à l'actif sous-jacent.

$$\Delta_t(S_t) = \frac{\partial F}{\partial x}(t, S_t).$$

- la méthode des differences finie :

$$\Delta_t(x) \approx \frac{F(t, x+h) - F(t, x-h)}{2h}.$$

$\mathbb{E}^*$ représente l'espérance sous la mesure risque-neutre.

avec

$$F(t, x{+}h) = e^{-r(T-t)}\mathbb{E}^*\left[f(S_{T-t}^{x+h})\right] \approx e^{-r(T-t)}\frac{1}{N}\sum_{i=1}^{N} f(S_{T-t}^{x+h,i}) \quad \text{(MONTE CARLO)}$$

$$F(t, x{-}h) = e^{-r(T-t)}\mathbb{E}^*\left[f(S_{T-t}^{x-h})\right] \approx e^{-r(T-t)}\frac{1}{N}\sum_{i=1}^{N} f(S_{T-t}^{x-h,i}) \quad \text{(MONTE CARLO)}$$

Ainsi,

$$\frac{F(t, x+h) - F(t, x-h)}{2h} \approx \frac{e^{-r(T-t)}}{2hN}\sum_{i=1}^{N}\left(f(S_{T-t}^{x+h,i}) - f(S_{T-t}^{x-h,i})\right)$$

Variance

La précision de cette approximation par Monte Carlo dépend de la variance indiquée ci-dessous. Si cette variance est plus faible, nous obtenons une meilleure approximation, car l'intervalle de confiance peut alors être réduit.

$$\mathrm{Var}\left(f(S_{T-t}^{x+h,i}) - f(S_{T-t}^{x-h,i})\right) = \mathrm{Var}(f(S_{T-t}^{x+h,i})) + \mathrm{Var}(f(S_{T-t}^{x-h,i})) - 2\,\mathrm{Cov}\left(f(S_{T-t}^{x+h,i}), f(S_{T-t}^{x-h,i})\right)$$

on ne veut pas mettre Cov $= 0$, c'est-à-dire que $f(S_{T-t}^{x+h,i})$ est **indépendant** de $f(S_{T-t}^{x-h,i})$.

Donc **On utilise le même échantillon aléatoire pour les deux simulations Monte Carlo :**

```python

import numpy as np

# Paramètres initiaux
x = 100          # Prix initial de l'actif
r = 0.03         # Taux sans risque
sigma = 0.2      # Volatilité
T = 1            # Temps
K1 = 100         # Prix d'exercice 1
K2 = 110         #Prix d'exercice 2
N = 51000        # Nombre de simulations Monte Carlo
M = 50           # Nombre de pas de temps
epsilon_values = [0.1, 0.5, 1.0, 2.0, 5.0]
np.random.seed(42)  # Graine aléatoire pour la reproductibilité

def geo_brownian_common(M, N, T, x, r, sigma, Z):

    dt = T / M
    S_path = np.zeros((M + 1, N))
    S_path[0] = x
    for t in range(1, M + 1):
        S_path[t] = S_path[t - 1] * np.exp((r - 0.5 * sigma ** 2)
         ↪  * dt + sigma * np.sqrt(dt) * Z[t - 1])
    return S_path


def prix_call_asiatique(S_path, T, M, K1):
    S_int = (T / M) * S_path.sum(axis=0)
    return np.maximum(S_int - K1, 0)


def prix_barrière(S_path, T, M, K1, K2):
    S_int = (T / M) * S_path.sum(axis=0)
    return ((S_int > K1) & (S_int < K2)).astype(float)

```

Et puis, j'ai utilisé deux différentes dénominateur pour calculer Delta par différences finies qui dépend de la régularité du payoff de l'option.

D'après le cours:

Nous savons pour Call Option:

$$\mathbb{E}^*\left[\left|f\left(S_T^{x+h}\right) - f\left(S_T^{x}\right)\right|^2\right] \;\leq\; \mathbb{E}^*\left[\left(S_T^{x+h} - S_T^{x}\right)^2\right] \;=\; h^2\,\mathbb{E}^*\Big[e^{\,2\left(r-\frac{1}{2}\sigma^2\right)T+2\,\sigma\,B_T}\Big] \;=\; O\left(h^2\right).$$

donc nous choisissons le dénominateur est $4\epsilon^2$.

Nous savons pour Barrier Option:

$$\mathbb{E}^*\big[\big|f\big(S_T^{x+h}\big) - f\big(S_T^{x}\big)\big|^2\big] \;=\; P^*\Big(\frac{M}{x+h} \;\leq\; e^{\left(r-\frac{1}{2}\sigma^2\right)T+\sigma\, B_T} \;<\; \frac{M}{x}\Big) \;=\; O(h).$$

donc nous choisissons le dénominateur est 4ϵ.

```python

def delta(M, N, x, sigma, T, r, epsilon, option_type):
    Z = np.random.standard_normal((M, N))

    # trajectoire
    S_plus = geo_brownian_common(M, N, T, x + epsilon, r, sigma,
    ↪  Z)
    S_moins = geo_brownian_common(M, N, T, x - epsilon, r, sigma,
    ↪  Z)


    if option_type == 'call':
        payoff_plus = prix_call_asiatique(S_plus, T, M, K1)
        payoff_moins = prix_call_asiatique(S_moins, T, M, K1)
    elif option_type == 'barrier':
        payoff_plus = prix_barrière(S_plus, T, M, K1, K2)
        payoff_moins = prix_barrière(S_moins, T, M, K1, K2)

    # payoff
    actualisation = np.exp(-r * T)
    V_plus = actualisation * payoff_plus.mean()
    V_moins = actualisation * payoff_moins.mean()

    #Delta
    delta_value = (V_plus - V_moins) / (2 * epsilon)

    # variance et intervalle de confiance
    actualisation_squared = np.exp(-2 * r * T)
    covariance = np.cov(payoff_plus, payoff_moins)[0, 1]
    var_plus = payoff_plus.var(ddof=1)
    var_moins = payoff_moins.var(ddof=1)


    if option_type == 'call':
        denominator = 4 * epsilon**2 * N
    elif option_type == 'barrier':
        denominator = 4 * epsilon * N

    var_delta = (actualisation_squared * (var_plus + var_moins - 2
    ↪  * covariance)) / denominator
```

```python
    erreur_type = np.sqrt(var_delta) if var_delta >= 0 else 0.0
    ci_bas = delta_value - 1.96 * erreur_type
    ci_haut = delta_value + 1.96 * erreur_type

    return delta_value, var_delta, (ci_bas, ci_haut)
```

```
Analyse du Delta pour l'option d'achat asiatique:
ε=0.1: Delta = 0.6375, Variance = 0.000006, IC 95% = [0.6329, 0.6422]
ε=0.5: Delta = 0.6391, Variance = 0.000005, IC 95% = [0.6345, 0.6437]
ε=1.0: Delta = 0.6385, Variance = 0.000005, IC 95% = [0.6339, 0.6430]
ε=2.0: Delta = 0.6354, Variance = 0.000005, IC 95% = [0.6309, 0.6399]
ε=5.0: Delta = 0.6311, Variance = 0.000005, IC 95% = [0.6269, 0.6353]

Analyse du Delta pour l'option barrière:
ε=0.1: Delta = 0.0019, Variance = 0.000001, IC 95% = [0.0004, 0.0034]
ε=0.5: Delta = 0.0045, Variance = 0.000001, IC 95% = [0.0029, 0.0060]
ε=1.0: Delta = 0.0039, Variance = 0.000001, IC 95% = [0.0024, 0.0054]
ε=2.0: Delta = 0.0043, Variance = 0.000001, IC 95% = [0.0029, 0.0058]
ε=5.0: Delta = 0.0055, Variance = 0.000001, IC 95% = [0.0041, 0.0070]
```

Contrairement aux prix, pour Delta, il existe deux facteurs d'approximation:

$$\underbrace{\text{Erreur due aux différences finies}}_{E_1} + \underbrace{\text{Erreur due à la Monte Carlo}}_{E_2}$$

- **Le choix de ϵ peut être difficile** (cf: Broadie-Glasserman) :
  - Si ϵ est trop grand, E_1 peut fortement augmenter.
  - Si ϵ est trop petit, la variance de l'estimateur Monte Carlo peut exploser.

  Option d'achat : Comme la fonction de payoff est lisse, la variation de Δ par rapport à ε est relativement stable. C'est à dire la méthode des différences finies est un choix efficace et fiable pour calculer le delta des options d'achat

  Option barrière : Comme la fonction de payoff n'est pas continue, Δ est plus sensible à ε, il faut choisir prudemment la taille du pas.

c) La méthode Intégrale par partie pour calculer Delta:

Dans la partie B, nous avons deux fonctions weight π_1 et π_2:

$$\Delta(x) = \frac{dP(x)}{dx} = \frac{e^{-rT}}{x}\mathbb{E}\left[\Phi\left(\int_0^T X_s^x ds\right)\Pi\right]; \quad \forall \Phi \in C^1 \cap Lip(\mathbb{R},\mathbb{R}) \tag{1}$$

where

$$\Pi_1 = \frac{\int_0^T X_s^x ds}{\int_0^T sX_s^x ds}\left[\frac{B_T}{\sigma} + \frac{\int_0^T s^2 X_s^x ds}{x\int_0^T sX_s^x ds}\right]. \tag{2}$$

$$\Pi_2 = \frac{2}{\sigma^2}\left[\frac{X_T^x - x}{\int_0^T X_s^x ds} - r\right] + 1. \tag{3}$$

Call asiatique option

```python

def geo_brownian_ipp(M, N, T, x, r, sigma):
    dt = T / M
    S_path = np.zeros((M+1, N))
    S_path[0] = x
    B_increments = np.zeros((M, N))
    for t in range(1, M+1):
        Z = np.random.standard_normal(N)
        S_path[t] = S_path[t-1] * np.exp((r - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt) * Z)
        B_increments[t-1] = Z * np.sqrt(dt)
    B_total = np.sum(B_increments, axis=0)
    return S_path, B_total, B_increments

def simulate_asian_call(T, M, r, sigma, x, N, K1):
    S_path, _, _ = geo_brownian_ipp(M, N, T, x, r, sigma)
    S_int = (T/M) * S_path.sum(axis=0)
    payoff = np.maximum(S_int - K1, 0)
    call_price = np.exp(-r*T) * payoff.mean()
    return call_price

def delta_asian_call_ibp(T, M, r, sigma, x, N, K):
    S_path, B_total, B_increments = geo_brownian_ipp(M, N, T, x, r, sigma)
    time_points = np.linspace(0, T, M+1)

    S_int = (T/M) * S_path.sum(axis=0)
    integral_sX = np.sum(S_path * time_points.reshape(-1, 1), axis=0) * (T/M)
    integral_s2X = np.sum(S_path * (time_points**2).reshape(-1, 1), axis=0) * (T/M)

    # weight 1 et 2
    Pi1 = (S_int / integral_sX) * (B_total / sigma + integral_s2X / (x * integral_sX))
```

```python
    Pi2 = (2 / sigma**2) * ((S_path[-1] - x) / S_int - r) + 1

    # Payoff Call
    payoff = np.maximum(S_int - K, 0)

    # Delta
    delta_Pi1 = (np.exp(-r * T) / x * np.mean(payoff * Pi1))
    delta_Pi2 = (np.exp(-r * T) / x * np.mean(payoff * Pi2))

    # variance et intervalle de confiance
    samples_Pi1 = np.exp(-r * T) * payoff * Pi1 / x
    samples_Pi2 = np.exp(-r * T) * payoff * Pi2 / x
    var_Pi1 = np.var(samples_Pi1) / N
    var_Pi2 = np.var(samples_Pi2) / N
    ci_Pi1 = (delta_Pi1 - 1.96 * np.sqrt(var_Pi1), delta_Pi1 + 1.96 * np.sqrt(var_Pi1))
    ci_Pi2 = (delta_Pi2 - 1.96 * np.sqrt(var_Pi2), delta_Pi2 + 1.96 * np.sqrt(var_Pi2))

    return delta_Pi1, delta_Pi2, var_Pi1, var_Pi2, ci_Pi1, ci_Pi2

if __name__ == "__main__":

    call_price = simulate_asian_call(T, M, r, sigma, x, N, K1)
    delta_Pi1, delta_Pi2, var_Pi1, var_Pi2, ci_Pi1, ci_Pi2 = delta_asian_call_ibp(T, M, r, sigma, x, N, K1)

    print("Asian Call Option Results:")
    print(f"Price = {call_price:.4f}")
    print(f"Delta (1 Method) = {delta_Pi1:.6f}, Variance = {var_Pi1:.6f}")
    print(f"95% CI (1): [{ci_Pi1[0]:.6f}, {ci_Pi1[1]:.6f}]")
    print(f"Delta (2 Method) = {delta_Pi2:.6f}, Variance = {var_Pi2:.6f}")
    print(f"95% CI (2): [{ci_Pi2[0]:.6f}, {ci_Pi2[1]:.6f}]")


#résultats

Asian Call Option Results:
Price = 6.4246

Delta (1 Method) = 0.620071, Variance = 0.000040
95% CI (1): [0.607674, 0.632467]

Delta (2 Method) = 0.628298, Variance = 0.000042
95% CI (2): [0.615619, 0.640976]
```

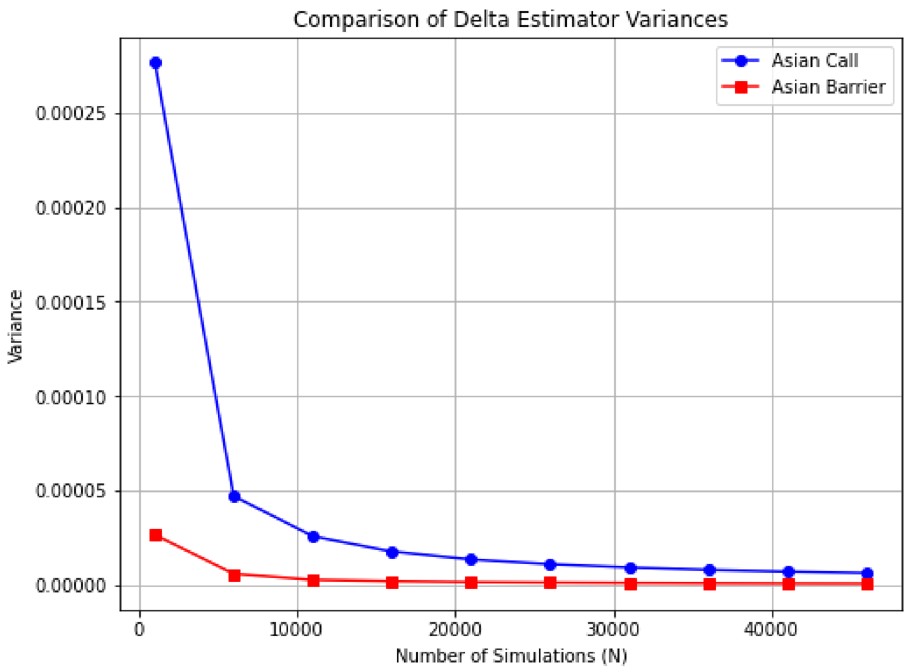
Barrier asiatique option

```python
def simulate_asian_barriere(T, M, r, sigma, x, N, K1, K2):
    S_path,_,_ = geo_brownian_ipp(M, N ,T,x,r,sigma)
    S_int = (T/M) * S_path.sum(axis=0)
    condition = (S_int > K1) & (S_int < K2)
    barriere_price = np.exp(-r*T)* condition.mean()
    return barriere_price


def delta_asian_barrier(T, M, r, sigma, x, N, K1,K2):
    S_path, B_total, B_increments = geo_brownian_ipp(M, N, T, x, r, sigma)
    time_points = np.linspace(0, T, M+1)

    S_int = (T/M) * S_path.sum(axis=0)
    integral_sX = np.sum(S_path * time_points.reshape(-1, 1), axis=0) * (T/M)
    integral_s2X = np.sum(S_path * (time_points**2).reshape(-1, 1), axis=0) * (T/M)

    # weight 1 et 2
    Pi1 = (S_int / integral_sX) * (B_total / sigma + integral_s2X / (x * integral_sX))
    Pi2 = (2 / sigma**2) * ((S_path[-1] - x) / S_int - r) + 1

    payoff_b = ((S_int > K1) & (S_int < K2)).astype(float)

    # Delta
    delta_Pi1 = (np.exp(-r * T) / x) * np.mean(payoff_b * Pi1)
    delta_Pi2 = (np.exp(-r * T) / x) * np.mean(payoff_b * Pi2)

    # variance et intervalle de confiance
    samples_Pi1 = np.exp(-r * T) * payoff_b * Pi1 / x
    samples_Pi2 = np.exp(-r * T) * payoff_b * Pi2 / x
    var_Pi1 = np.var(samples_Pi1) / N
    var_Pi2 = np.var(samples_Pi2) / N
    ci_Pi1 = (delta_Pi1 - 1.96 * np.sqrt(var_Pi1), delta_Pi1 + 1.96 * np.sqrt(var_Pi1))
    ci_Pi2 = (delta_Pi2 - 1.96 * np.sqrt(var_Pi2), delta_Pi2 + 1.96 * np.sqrt(var_Pi2))

    return delta_Pi1, delta_Pi2, var_Pi1, var_Pi2, ci_Pi1, ci_Pi2


if __name__ == "__main__":

    barrier_price = simulate_asian_barriere(T, M, r, sigma, x, N, K1, K2)
```

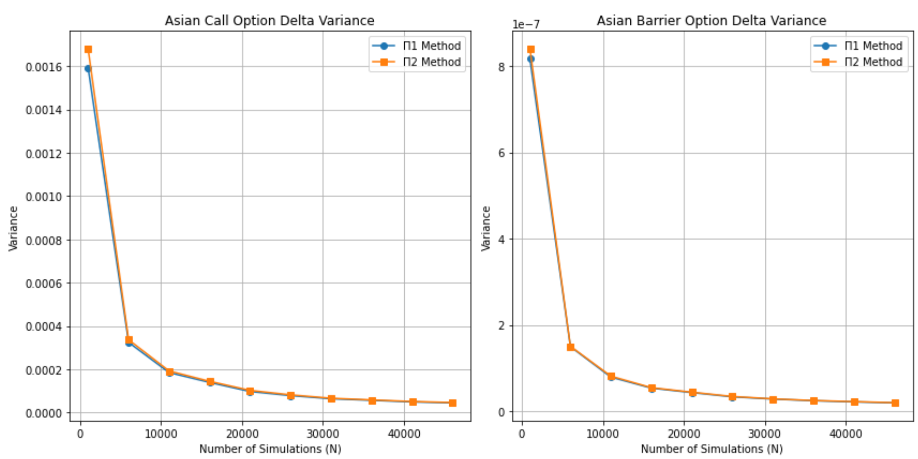
```
    delta_Pi1_barrier, delta_Pi2_barrier, var_Pi1_barrier,
    ↪ var_Pi2_barrier, ci_Pi1_barrier, ci_Pi2_barrier =
    ↪ delta_asian_barrier(T, M, r, sigma, x, N, K1, K2)

    print("\nAsian Barrier Option Results:")
    print(f"Price = {barrier_price:.4f}")
    print(f"Delta (1 Method) = {delta_Pi1_barrier:.6f}, Variance =
    ↪ {var_Pi1_barrier:.6f}")
    print(f"95% CI (1): [{ci_Pi1_barrier[0]:.6f},
    ↪ {ci_Pi1_barrier[1]:.6f}]")
    print(f"Delta (2 Method) = {delta_Pi2_barrier:.6f}, Variance =
    ↪ {var_Pi2_barrier:.6f}")
    print(f"95% CI (2): [{ci_Pi2_barrier[0]:.6f},
    ↪ {ci_Pi2_barrier[1]:.6f}]")

#résultats

Asian Barrier Option Results:
Price = 0.3059
Delta (pi_1 Method) = 0.003837, Variance = 0.000000
95% CI (pi_1): [0.003583, 0.004091]
Delta (pi_2 Method) = 0.004655, Variance = 0.000000
95% CI (pi_2): [0.004398, 0.004911]
```

3.1 d)

1) La méthode différence finie



2) La méthode d'intégrale par partie avec deux fonctions weight



e) Conclusion

1. **Comparaison des méthodes Π_1 et Π_2**
   Leurs variances étant similaires sur toute la plage de N, le choix entre

les deux fonctions Π_1 et Π_2 n'a qu'un impact négligeable sur la précision. Toutefois, Π_1 donne des résultats légèrement meilleurs que Π_2 pour les deux types d'options.

2. **Règle empirique**
   L'efficacité de l'intégration par parties est corrélée à la régularité du payoff : meilleure pour des payoffs lisses (*Asian Call*), L'intégration par parties réduit efficacement la variance. moins pour des payoffs discontinus (*Asian Barrier*).

   Donc, nous choisissons la méthode d'intégration par parties pour l'option asiatique de type call, et la méthode des différences finies pour l'option asiatique à barrière, car elle converge mieux (sa variance est stable et faible).