Cutting Edge Project

A Multilevel Monte Carlo Implementation

Alaa Bouattour Mahdi Ben Ayed

M2QF – Université Paris-Saclay

18 juin 2025

Plan

Méthode Monte Carlo : principe et limites

Pour estimer une espérance $\mathbb{E}[P]$ à partir d'un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$, la méthode de Monte Carlo consiste à :

- Générer N réalisations indépendantes $\omega^{(n)}$.
- Calculer l'estimateur suivant :

$$\hat{P}_N = \frac{1}{N} \sum_{n=1}^{N} P(\omega^{(n)})$$

Erreur et coût :

- Variance de l'estimateur : $\frac{1}{N}\mathbb{V}[P]$
- Erreur quadratique moyenne (r.m.s) : $\mathcal{O}(N^{-1/2})$
- Pour une précision ε, on a besoin de $N = \mathcal{O}(\varepsilon^{-2})$
- Le coût devient très élevé si chaque $P(\omega)$ implique la résolution numérique d'une EDP ou une simulation coûteuse avec de nombreux pas de temps.

Variable de contrôle : réduction de variance

Soit f une fonction d'intérêt et g une variable de contrôle telle que :

- g est corrélée à f,
- $\mathbb{E}[g]$ est connue.

Estimateur sans biais de $\mathbb{E}[f]$:

$$\frac{1}{N}\sum_{n=1}^{N}\left(f(\omega^{(n)}) - \lambda\left(g(\omega^{(n)}) - \mathbb{E}[g]\right)\right)$$

Valeur optimale de λ :

$$\lambda^* = \rho\sqrt{\frac{\mathbb{V}[f]}{\mathbb{V}[g]}} \quad \text{où } \rho = \text{corr}(f, g)$$

Gain : la variance est réduite d'un facteur $1 - \rho^2$ par rapport à l'estimateur standard.

Estimateur à deux niveaux : idée clé

Objectif : estimer $\mathbb{E}[P_1]$ avec un coût réduit, où P_1 est coûteux à simuler.

On introduit une approximation bon marché $P_0 \approx P_1$.

$$\mathbb{E}[P_1] = \mathbb{E}[P_0] + \mathbb{E}[P_1 - P_0]$$

Estimateur sans biais :

$$\frac{1}{N_0} \sum_{n=1}^{N_0} P_0^{(n)} + \frac{1}{N_1} \sum_{n=1}^{N_1} \left(P_1^{(n)} - P_0^{(n)} \right)$$

- $P_1^{(n)}$ et $P_0^{(n)}$ sont simulés à partir du **même** échantillon $\omega^{(n)}$.
- $P_1 - P_0$ a une variance faible : on peut le simuler avec moins d'échantillons.

Optimisation du coût pour une variance fixée

On définit :

- C_0, C_1 : coût d'un échantillon de P_0, $P_1 - P_0$
- V_0, V_1 : variance de P_0, $P_1 - P_0$

Coût total :

$$\text{Coût} = N_0 C_0 + N_1 C_1$$

Variance totale :

$$\text{Var} = \frac{V_0}{N_0} + \frac{V_1}{N_1}$$

Optimisation : minimiser le coût pour une variance fixée via un multiplicateur de Lagrange.

Choix optimal des tailles d'échantillons :

$$\frac{N_1}{N_0} = \sqrt{\frac{V_1 C_0}{V_0 C_1}}$$

$\rightarrow$ *Plus de simulations sur le niveau le moins coûteux et le plus bruité.*

Estimateur MLMC à plusieurs niveaux

On considère une suite de raffinement : $P_0, P_1, \ldots, P_L$ telle que :

$$\mathbb{E}[P_L] = \mathbb{E}[P_0] + \sum_{\ell=1}^{L} \mathbb{E}[P_\ell - P_{\ell-1}]$$

Estimateur sans biais :

$$\frac{1}{N_0} \sum_{n=1}^{N_0} P_0^{(0,n)} + \sum_{\ell=1}^{L} \left(\frac{1}{N_\ell} \sum_{n=1}^{N_\ell} \left(P_\ell^{(\ell,n)} - P_{\ell-1}^{(\ell,n)} \right) \right)$$

- Chaque niveau affine l'approximation du niveau précédent.
- Les échantillons sont indépendants entre niveaux.

Minimisation du coût pour une variance fixée

On note :

- C_ℓ : coût d'un échantillon au niveau ℓ
- V_ℓ : variance de $P_\ell - P_{\ell-1}$

Coût total :

$$\text{Coût} = \sum_{\ell=0}^{L} N_\ell C_\ell$$

Variance totale :

$$\sum_{\ell=0}^{L} \frac{V_\ell}{N_\ell} = \varepsilon^2$$

Optimisation (Lagrange) :

$$N_\ell = \mu \sqrt{\frac{V_\ell}{C_\ell}} \quad \Rightarrow \quad \text{Coût} = \varepsilon^{-2} \left(\sum_{\ell=0}^{L} \sqrt{V_\ell C_\ell} \right)^2$$

Coût MLMC et comparaison avec MC standard

Coût MLMC : $C = \varepsilon^{-2} \left(\sum_{\ell=0}^{L} \sqrt{V_\ell C_\ell} \right)^2$

Selon le comportement de $V_\ell C_\ell$ **:**

- **Croissant :** $C \approx \varepsilon^{-2} V_L C_L$
- **Décroissant :** $C \approx \varepsilon^{-2} V_0 C_0$
- **Constant :** $C = \varepsilon^{-2} L^2 V_0 C_0 = \varepsilon^{-2} L^2 V_L C_L$

MC standard (Giles, 2008) : $C_{\text{MC}} \approx \varepsilon^{-2} V_0 C_L$ (avec $C_L \approx$ coût de $P_L - P_{L-1}$ et $\mathbb{V}[P_L] \approx \mathbb{V}[P_0]$)

Conclusion :

- Si $V_L \ll V_0$, gain $\sim V_L / V_0$
- Si $C_0 \ll C_L$, gain $\sim C_0 / C_L$

Formulation théorique du MLMC

On approxime une variable aléatoire P par P_L à différents niveaux de résolution. On souhaite estimer :

$$\text{MSE} = \mathbb{E}\left[(Y - \mathbb{E}[P])^2\right] = \mathbb{V}[Y] + (\mathbb{E}[Y] - \mathbb{E}[P])^2$$

Avec l'estimateur MLMC :

$$Y = \sum_{\ell=0}^{L} Y_\ell, \quad Y_\ell = \frac{1}{N_\ell} \sum_{n=1}^{N_\ell} \left(P_\ell^{(n)} - P_{\ell-1}^{(n)}\right), \quad P_{-1} := 0$$

Alors :

$$\mathbb{E}[Y] = \mathbb{E}[P_L], \quad \mathbb{V}[Y] = \sum_{\ell=0}^{L} \frac{V_\ell}{N_\ell}, \quad V_\ell = \mathbb{V}[P_\ell - P_{\ell-1}]$$

Objectif : garantir MSE $< \varepsilon^2$ en combinant :

- $\mathbb{V}[Y] < \frac{\varepsilon^2}{2}$ (variance)
- $(\mathbb{E}[P_L] - \mathbb{E}[P])^2 < \frac{\varepsilon^2}{2}$ (biais)

Théorème de complexité (Giles, 2008)

Hypothèses : il existe des constantes $\alpha, \beta, \gamma > 0$ telles que :

i) $|\mathbb{E}[P_L - P]| \leq c_1 \cdot 2^{-\alpha L}$ (biais)

ii) $V_\ell \leq c_2 \cdot 2^{-\beta \ell}$ (décroissance de variance)

iii) $C_\ell \leq c_3 \cdot 2^{\gamma \ell}$ (coût croissant)

Alors il existe une constante c_4 telle que pour tout $\varepsilon < 1$:

$$\text{MSE} = \mathbb{E}[(Y - \mathbb{E}[P])^2] < \varepsilon^2$$

avec un coût :

$$\mathbb{E}[C] \leq \begin{cases} c_4 \cdot \varepsilon^{-2}, & \beta > \gamma \\ c_4 \cdot \varepsilon^{-2} (\log \varepsilon)^2, & \beta = \gamma \\ c_4 \cdot \varepsilon^{-2-(\gamma-\beta)/\alpha}, & \beta < \gamma \end{cases}$$

Interprétation :

- Si $\beta > \gamma$: complexité optimale $O(\varepsilon^{-2})$
- Si $\beta = \gamma$: pénalité logarithmique
- Si $\beta < \gamma$: coût exponentiellement plus élevé

Intuition de la preuve du théorème

L'analyse repose sur les ordres :

$$V_\ell = O(2^{-\beta\ell}), \quad C_\ell = O(2^{\gamma\ell})$$

Nombre optimal d'échantillons au niveau ℓ :

$$N_\ell \propto 2^{(\gamma-\beta)\ell/2} \Rightarrow \text{coût au niveau } \ell : N_\ell C_\ell \propto 2^{(\gamma-\beta)\ell/2} \cdot 2^{\gamma\ell} = 2^{(3\gamma-\beta)\ell/2}$$

Conclusion :

- La répartition des coûts dépend de la position de β par rapport à γ.
- Le niveau final L est choisi pour que le biais vérifie $(\mathbb{E}[P_L] - \mathbb{E}[P])^2 < \varepsilon^2/2$.
- Le nombre de niveaux et la répartition des N_ℓ sont alors optimisés pour contrôler la variance.

Cas clé : $\beta = \gamma$ correspond à un équilibre parfait entre variance et coût — les contributions sont réparties également sur tous les niveaux.

Vers le C-MLMC

- Le **MLMC classique** repose sur une allocation optimale entre les niveaux, basée sur la variance moyenne par niveau.
- Cela peut conduire à un **gaspillage d'échantillons** dans des zones peu informatives.
- La **variance intra-niveau** peut être fortement hétérogène.
- **Idée clé :** exploiter des **caractéristiques** (*features*) des trajectoires pour identifier les zones à forte contribution à la variance.

Objectif

Réduire la **variance totale à coût équivalent** ou réduire le **coût** pour une précision donnée.

Théorie : rappel sur le MLMC

- Estimateur MLMC :

$$\hat{Y}_{\text{MLMC}} = \sum_{\ell=0}^{L} \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} \left(Y_\ell^{(i)} - Y_{\ell-1}^{(i)} \right)$$

- Allocation optimale du nombre d'échantillons :

$$N_\ell \propto \sqrt{\frac{V_\ell}{C_\ell}}$$

où V_ℓ est la variance du niveau ℓ et C_ℓ son coût moyen.

- Cette stratégie dépend uniquement des **variances globales** V_ℓ

Théorie : intuition derrière le C-MLMC

- On extrait des **features** $\varphi(path)$ pour chaque trajectoire simulée
- On applique un **clustering** (ex. KMeans) pour regrouper les trajectoires en **strates à variance homogène**
- On effectue un **échantillonnage stratifié par cluster** à chaque niveau

Objectif

Réduire la **variance intra-niveau** grâce à une **allocation intra-cluster optimale**

Algorithme du C-MLMC

1. **Phase pilote** : simuler N_0 trajectoires pour chaque niveau ℓ
2. Extraire les **features** $\varphi(path)$ pour chaque trajectoire
3. Effectuer un **clustering** (ex. KMeans) pour former C clusters
4. Estimer **variance** $V_{\ell,c}$ et **coût** $C_{\ell,c}$ pour chaque niveau ℓ et cluster c
5. Allouer les échantillons selon :

$$N_{\ell,c} \propto p_{\ell,c} \cdot \sqrt{\frac{V_{\ell,c}}{C_{\ell,c}}}$$

6. Simuler de manière **stratifiée** par cluster et agréger les contributions

Agrégation finale

Pondération basée sur les proportions estimées $p_{\ell,c}$ de chaque cluster

C-MLMC : Allocation optimale (1/2)

Objectif : estimer l'espérance

$$\mathbb{E}[Y] = \sum_{\ell=0}^{L} \mathbb{E}[Z_\ell] \quad \text{où} \quad Z_\ell = Y_\ell - Y_{\ell-1}$$

Estimateur stratifié par cluster :

$$\hat{Y}_{\text{CMLMC}} = \sum_{\ell=0}^{L} \sum_{c=1}^{C} \frac{1}{N_{\ell,c}} \sum_{i=1}^{N_{\ell,c}} Z_{\ell,c}^{(i)} \cdot p_{\ell,c}$$

- $N_{\ell,c}$: nombre de trajectoires dans le cluster c au niveau ℓ
- $p_{\ell,c}$: proportion estimée de trajectoires dans le cluster c
- $\text{Var}(Z_{\ell,c}) = V_{\ell,c}$, coût moyen : $C_{\ell,c}$

Problème d'optimisation :

$$\min_{N_{\ell,c}} \sum_{\ell,c} N_{\ell,c} C_{\ell,c} \quad \text{s.t.} \quad \sum_{\ell,c} \frac{p_{\ell,c}^2 V_{\ell,c}}{N_{\ell,c}} = \varepsilon^2$$

C-MLMC : Allocation optimale (2/2)

Méthode : Lagrangien

$$\mathcal{L}(N_{\ell,c}, \lambda) = \sum_{\ell,c} N_{\ell,c} C_{\ell,c} + \lambda \left(\sum_{\ell,c} \frac{p_{\ell,c}^2 V_{\ell,c}}{N_{\ell,c}} - \varepsilon^2 \right)$$

Condition du premier ordre :

$$\frac{\partial \mathcal{L}}{\partial N_{\ell,c}} = C_{\ell,c} - \lambda \frac{p_{\ell,c}^2 V_{\ell,c}}{N_{\ell,c}^2} = 0 \quad \Rightarrow \quad N_{\ell,c} = \sqrt{\lambda \cdot \frac{p_{\ell,c}^2 V_{\ell,c}}{C_{\ell,c}}}$$

Donc, allocation optimale :

$$N_{\ell,c} \propto p_{\ell,c} \cdot \sqrt{\frac{V_{\ell,c}}{C_{\ell,c}}}$$

Interprétation

- Plus de trajectoires dans les clusters à variance élevée et coût faible.

Structure du code : classes et fonctions clés

1. **Classe BSLevelFunction (classe mère)**
   - Interface pour simuler trajectoires sous Black-Scholes à un niveau ℓ donné
   - Méthode `simulate(l, N)` à implémenter
   - Retourne payoff et coût simulation

2. **Sous-classe MilsteinBSLevelFunction**
   - Hérite de BSLevelFunction
   - Implémente `simulate` avec la méthode de Milstein
   - Simule trajectoires couplées fine/coarse
   - Calcule différence de payoff $Y_\ell = P_\ell - P_{\ell-1}$

3. **Fonction `make_level_fn`**
   - Combine paramètres modèles + payoff + classe simulation
   - Produit la fonction `level_fn(l, N)` utilisable par C-MLMC

Fonction feature et clustering

Extraction des features :

- À partir des détails simulés (valeur finale fine/coarse)
- Exemples : S_{fine}, $S_{\text{fine}} - S_{\text{coarse}}$
- Ces features décrivent la variance locale des trajectoires

Clustering intra-niveau :

- Regroupe trajectoires avec comportements similaires
- Utilisation d'algorithme KMeans sur les features
- Permet d'identifier des clusters homogènes en variance

Avantage : meilleure allocation des échantillons selon clusters pour réduire coût total

Classe C_MLMC : algorithme et workflow

Entrées :

- Fonction simulation par niveau `sde_step_fn` (ex : `level_fn`)
- Fonction extraction features `feature_fn`
- Paramètres MLMC : L_{min}, L_{max}, taille pilote N_0, nombre de clusters K

Processus :

1. Simulation pilote à chaque niveau ℓ
2. Extraction des features et clustering en K clusters
3. Estimation variance et coût par cluster
4. Calcul allocation optimale $N_{\ell,c}$ par niveau/cluster
5. Simulation finale avec allocation optimisée

Sorties :

- Estimation du prix
- Allocation par niveau et cluster
- Coût total et variance finale maîtrisée

Exemple d'utilisation du C-MLMC

```
payoff = lambda S: call_payoff(S, K)
level_fn = MilsteinBSLevelFunction(S0, r, sigma, T, payoff,

            verbose=False).simulate

def feature_fn(detail):
    S_fine = detail["S_fine"]
    if "S_coarse" in detail:
        S_coarse = detail["S_coarse"]
        return np.array([S_fine, S_fine - S_coarse])
    else:
        return np.array([S_fine])

cmlmc_obj = C_MLMC(sde_step_fn=level_fn, feature_fn=feature_fn,
            Lmin=2, Lmax=20, n_clusters=3, N0=200,

            scale_features=True)

params = S0, K, r, sigma, T = 100.0, 100.0, 0.05, 0.20, 1.0
print("Exact price of this option: ", 10.450583572185565)  #euro call
```

Prix estimé en fonction de la tolérance ε

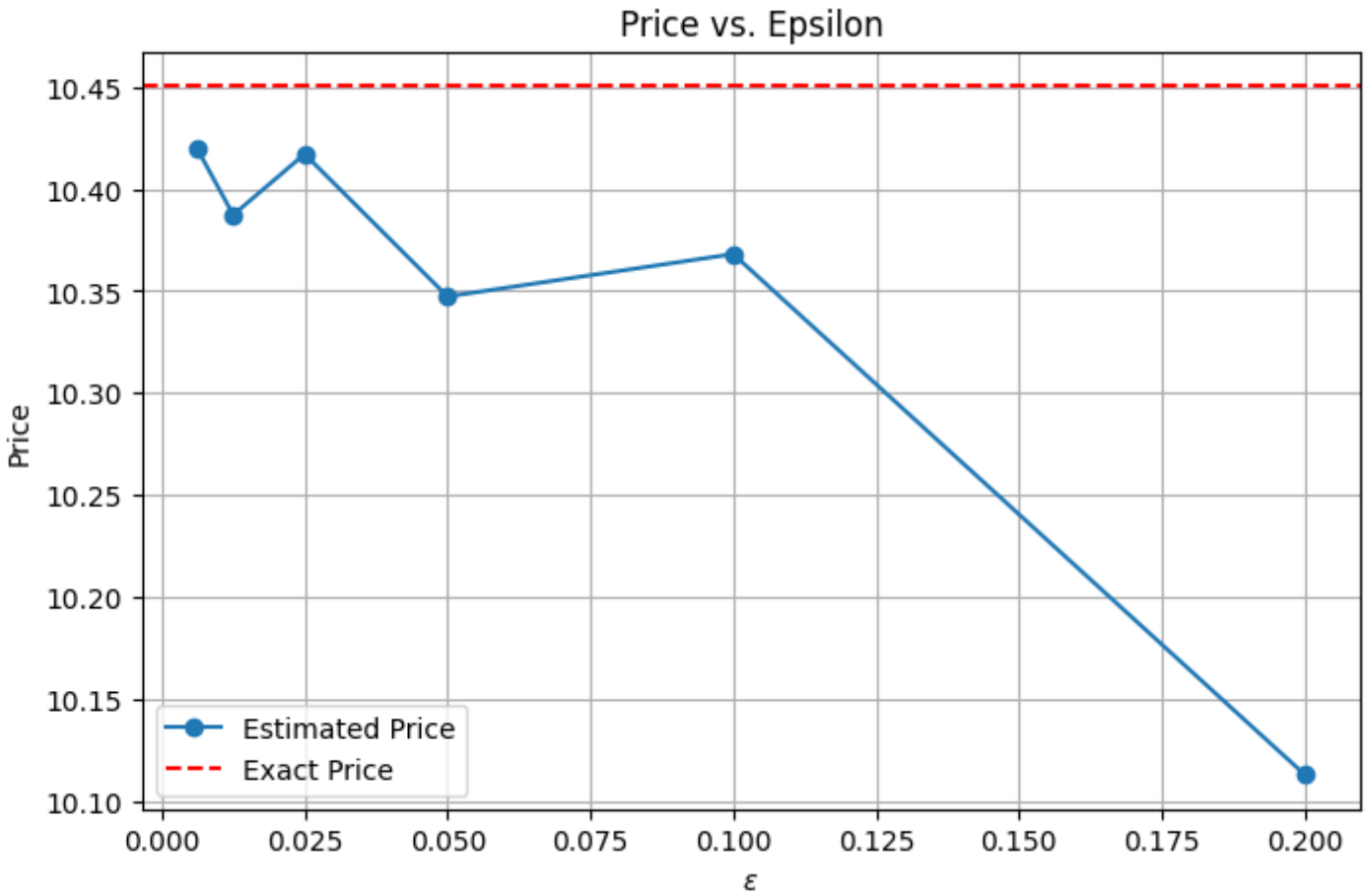

Figure – Évolution du prix estimé par C-MLMC en fonction de la tolérance ε. La ligne rouge en pointillés représente le prix exact de l'option call européenne.

Convergence en erreur et coût

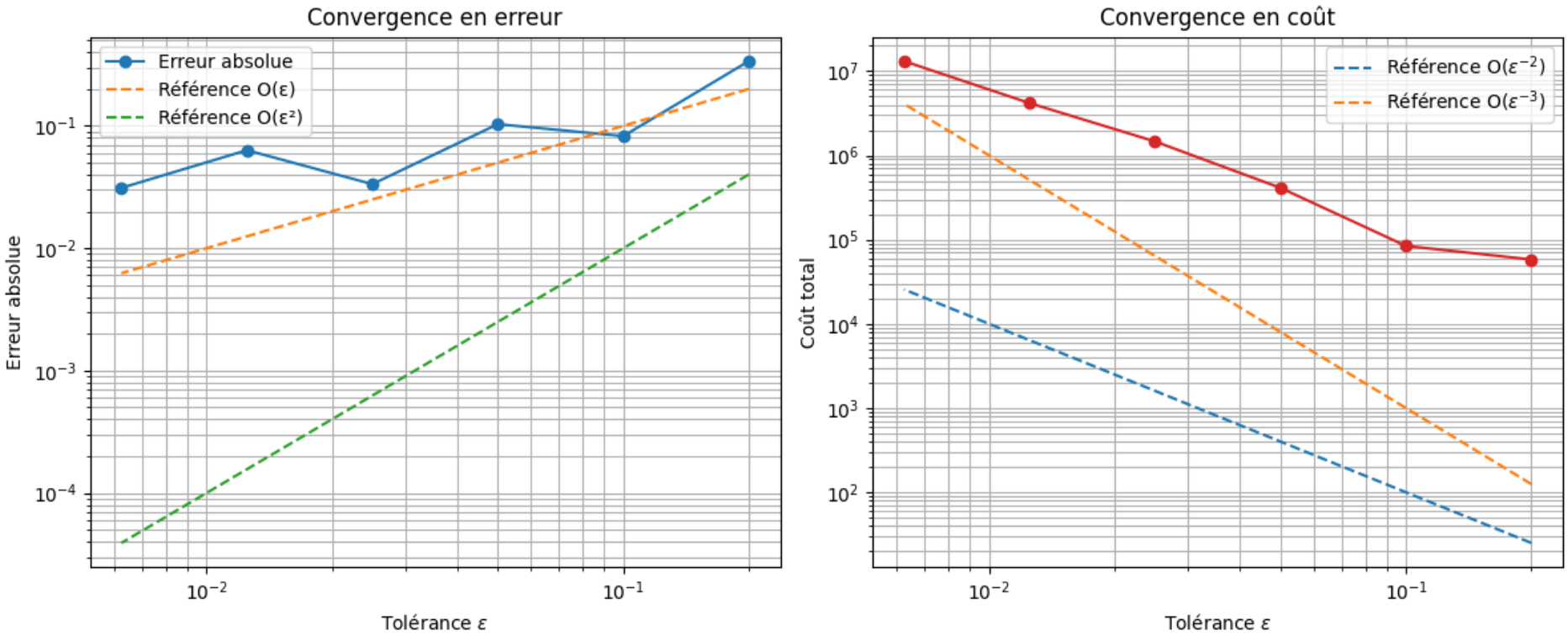

Figure – Convergence du C-MLMC en fonction de ε en erreur absolue et en coût

Option digitale : limite du MLMC classique

- **Discontinuité du payoff** $P(S_T) = 1_{\{S_T > K\}} \rightarrow$ presque toutes les différences de niveau ΔP_ℓ sont nulles sauf sur les trajectoires rares proches du seuil.
- **Variance pilote plate** : la variance estimée V_ℓ est quasi constante et très faible, ne révélant pas les zones à forte variance.
- **Allocation aveugle** : la règle $N_\ell \propto \sqrt{V_\ell / C_\ell}$ ne détecte aucun signal et répartit uniformément les échantillons, gaspillant les ressources.
- **Conséquence** : inefficacité du MLMC classique face aux payoffs discontinus.

Extraction de la feature utilisée pour C-MLMC

Feature définie :

$$\varphi(\text{trajectory}) = |S_{\text{fine}} - K|$$

- S_{fine} : valeur finale de la trajectoire fine simulée
- K : prix d'exercice de l'option digitale

Paramètres de la simulation :

- Nombre de clusters : 3
- Taille pilote $N_0 = 5000$
- Niveaux considérés : de $L_{\text{min}} = 2$ à $L_{\text{max}} = 20$
- Tolérances testées :

$$\varepsilon \in \{0.2, 0.1, 0.05, 0.025, 0.0125, 0.00625\}$$

Prix exact de l'option digitale :

$$0.02849536146532932$$

MLMC (classique) : Allocation des Nl par ε

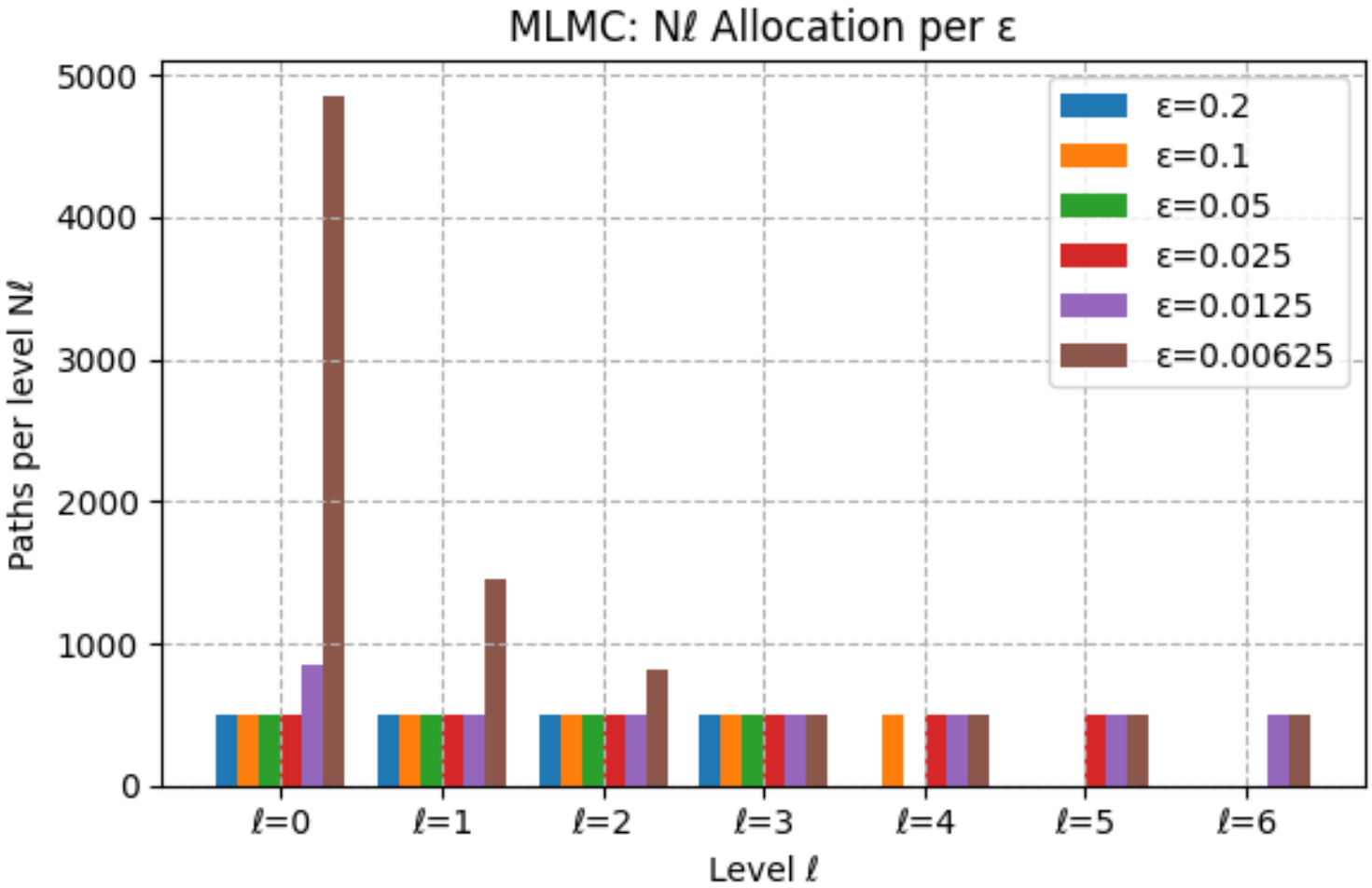

Figure – Allocation des Nl pour des valeurs disctinctes de ε pour le MLMC

C-MLMC : Allocation des Nl par ε

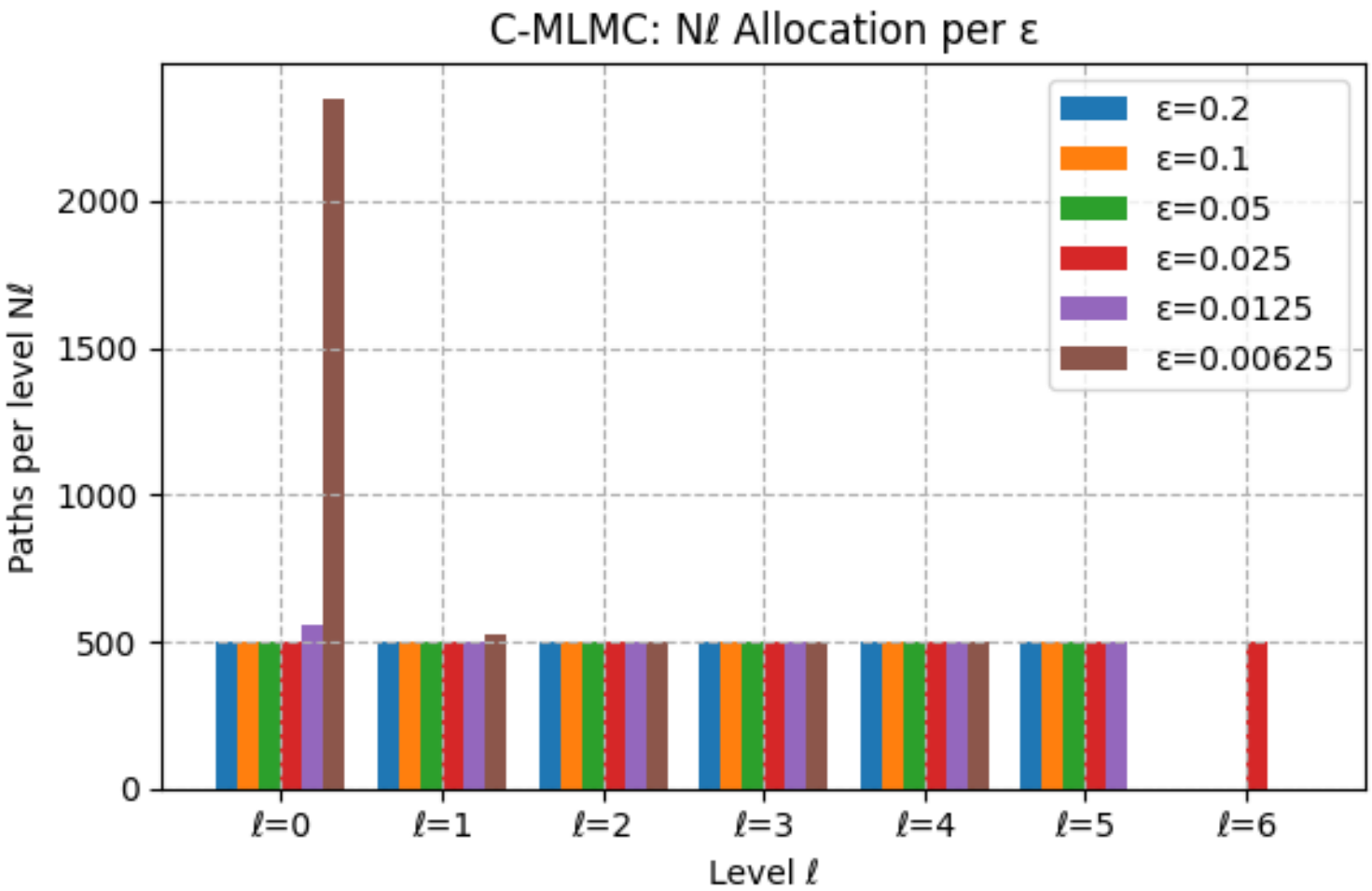

Figure – Allocation des Nl pour des valeurs disctinctes de ε pour le C-MLMC

Convergence en erreur et coût : MLMC vs C-MLMC

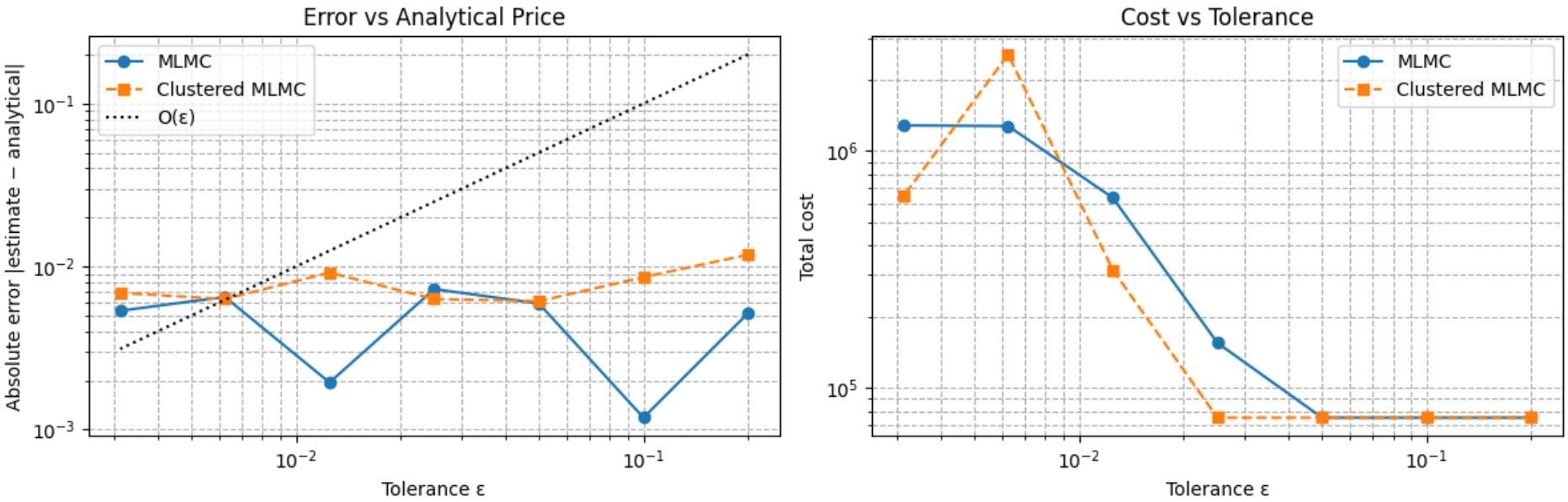

Figure – Convergence en fonction de ε en erreur absolue et en coût : MLMC vs C-MLMC

Option asiatique : test de performance C-MLMC

- **Contexte :** option asiatique call, payoff dépend de la moyenne du sous-jacent.
- **Caractéristique du problème :** plus lissée que l'option digitale, mais les trajectoires peuvent varier significativement selon la forme (minima, moyenne, fin de trajectoire).
- **Objectif :** comparer les performances de MLMC et C-MLMC sur ce cas "intermédiaire", non pathologique.
- **Résultat de référence (Monte Carlo standard) :**

$$\mathbb{E}[P] \approx 1.176119$$

Feature utilisée pour C-MLMC : option asiatique

Feature vector :

$$\varphi(\text{trajectory}) = \begin{cases} \text{Fine avg}, \ \text{Fine avg - Coarse avg}, \\ \text{Fine min - Coarse min}, \ \text{Fine end - Coarse end} \end{cases}$$

Interprétation :

- Moyenne fine : contribue directement au payoff.
- Différences fine/coarse : mesures d'écarts de trajectoire (forme, terminal).

Paramètres de simulation :

- $S_0 = 95$, $K = 100$, $r = 0.05$, $\sigma = 0.1$, $T = 1$
- Niveaux : $L_{\text{min}} = 2$, $L_{\text{max}} = 20$
- Nombre de clusters : 3, taille pilote $N_0 = 5000$
- Tolérances testées :

$$\varepsilon \in \{0.2, \ 0.1, \ 0.05, \ 0.025, \ 0.0125\}$$

MLMC (classique) : Allocation des Nl par ε

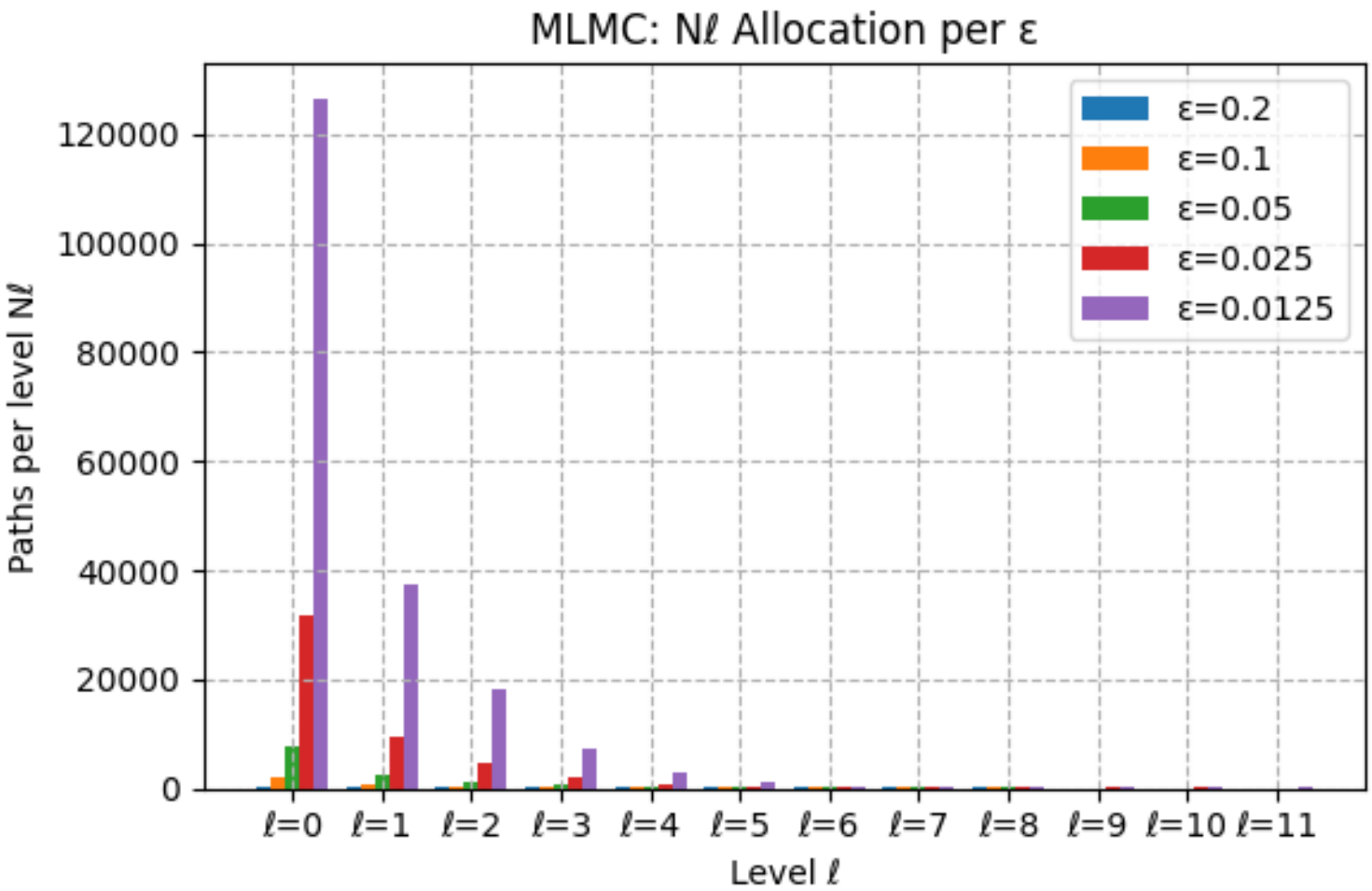

Figure – Allocation des Nl pour des valeurs disctinctes de ε pour le MLMC

C-MLMC : Allocation des Nl par ε

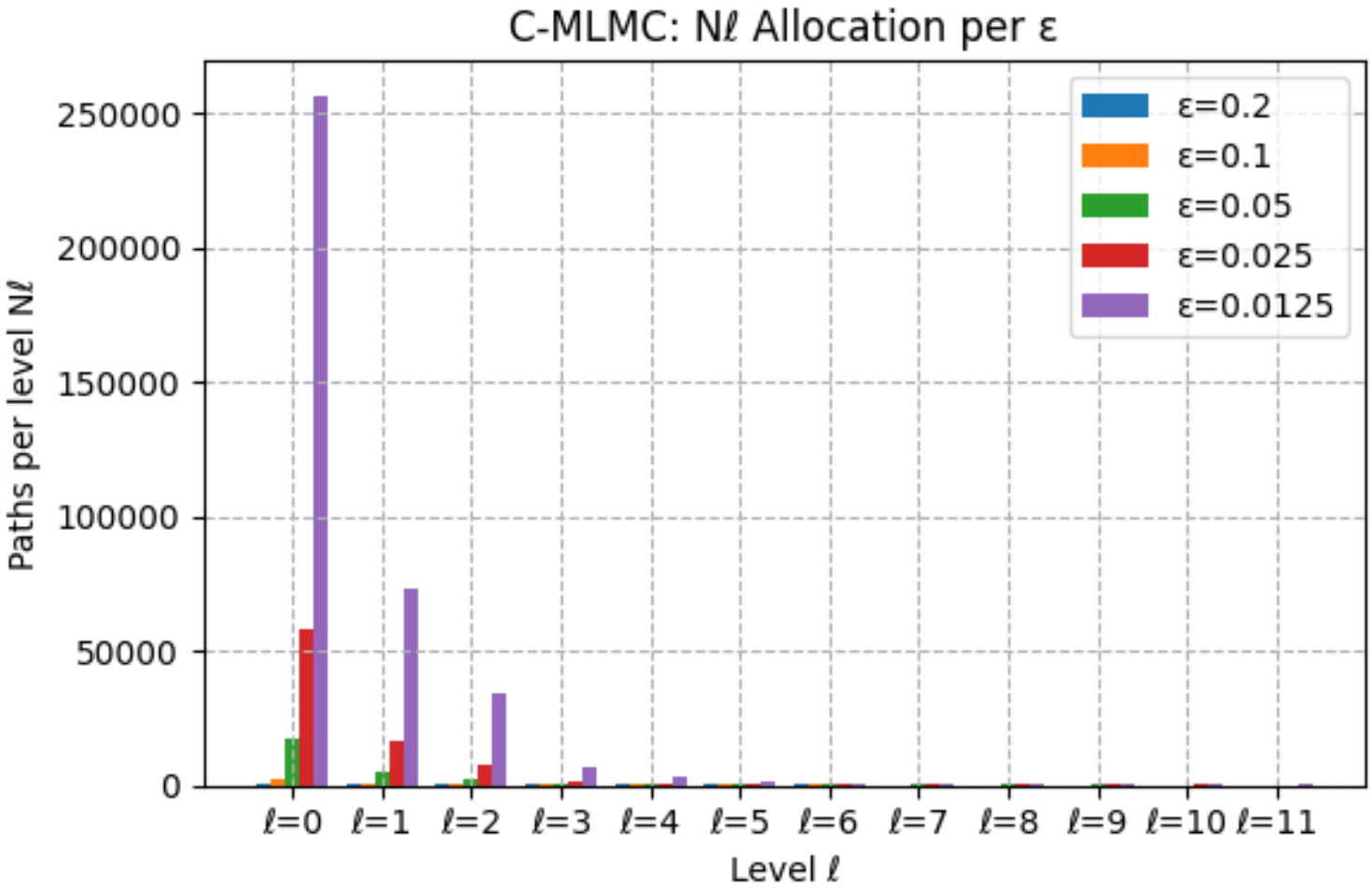

Figure – Allocation des Nl pour des valeurs disctinctes de ε pour le C-MLMC

Convergence en erreur et coût : MLMC vs C-MLMC

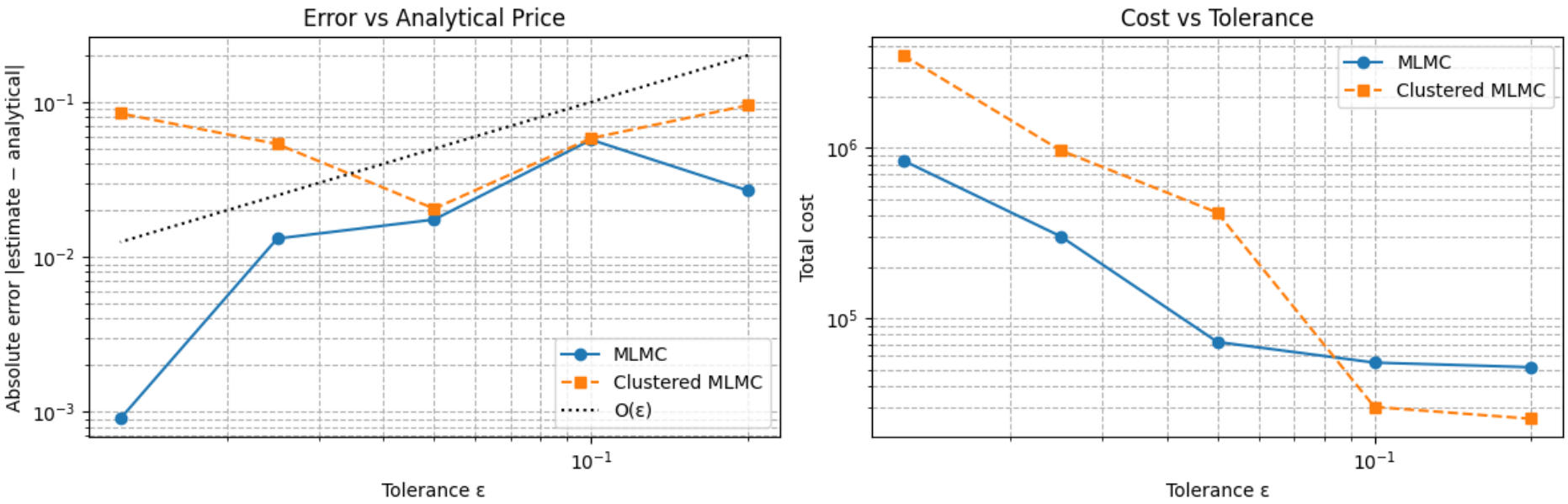

Figure – Convergence en fonction de ε en erreur absolue et en coût : MLMC vs C-MLMC