L'IA avant le Deep Learning : Évaluation d'Options Américaines avec l'Algorithme de Longstaff-Schwartz

Projet P1RV – Centrale Nantes

F. Barbe, N. El Manssouri

Janvier 2026

Lancer le Diaporama

Sommaire

1. L'IA avant le Deep Learning : Pourquoi ce choix ?

Positionnement du projet :

- **LSMC** = *Machine Learning* Supervisé (Régression).
- Avant l'ère du Deep Learning (Réseaux de Neurones).

Comparaison :

- **Approche Classique (LSMC)** :
  - Feature Engineering explicite (Polynômes).
  - Modèle linéaire, interprétable.
  - Solution analytique (Moindres Carrés).
- **Deep Learning** :
  - "Black Box", apprentissage de caractéristiques.
  - Optimisation stochastique (Descente de gradient).

Pourquoi ce retour aux sources ?

Comprendre les fondements mathématiques de l'approximation avant d'utiliser des boîtes noires.

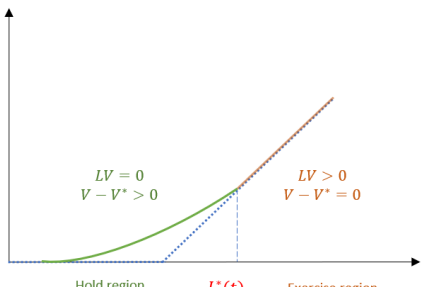
2.1. Options Américaines : Définition

Qu'est-ce qu'une option américaine ?

- Droit d'acheter (Call) ou vendre (Put) un actif
- À un prix fixé **(Strike K)**
- **À tout moment** avant maturité T

Différence avec les options européennes :

- Européenne : exercice uniquement à T
- Américaine : flexibilité $\rightarrow$ prime supplémentaire



Frontière d'exercice optimal

(Axe X : Prix Sous-jacent S_t | Axe Y : Valeur Option V_t)

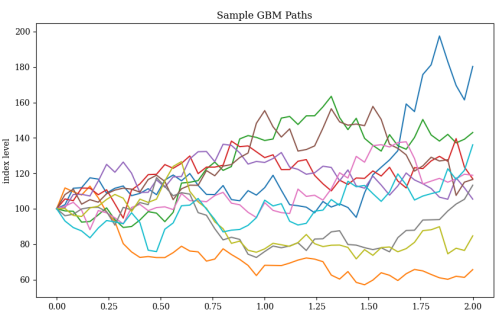
2.2. Modélisation : Mouvement Brownien Géométrique

Mouvement Brownien Géométrique (GBM) :

$$\mathrm{d}S_t = rS_t\,\mathrm{d}t + \sigma S_t\,\mathrm{d}W_t$$

Paramètres :

- S_0 : Prix initial
- r : Taux sans risque
- σ : Volatilité
- W_t : Processus de Wiener



2.3. Le Problème Mathématique : Exercice Optimal

Problème : À chaque instant, faut-il exercer l'option ou attendre ?

Valeur intrinsèque

$$\Phi(S_t) = \max(K - S_t, 0)$$

(pour un Put)

Valeur de continuation

$$C(t, S_t) = \mathbb{E}[e^{-r\Delta t} V_{t+1} | S_t]$$

(espérance conditionnelle)

Décision : Exercer si $\Phi(S_t) \geq C(t, S_t)$

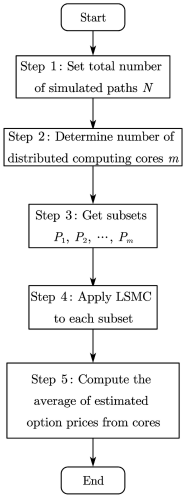
$\rightarrow$ **Pas de solution analytique !** $\rightarrow$ Méthodes numériques

3.1. Principe de l'Algorithme LSMC

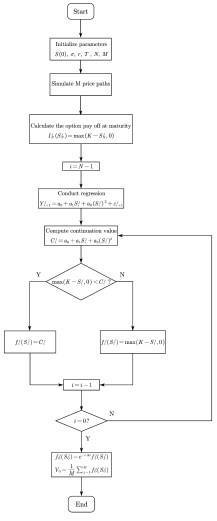
Approche hybride Simulation / Régression :

1. **Phase Forward (Simulation)** :
   On lance des milliers de trajectoires aléatoires du prix de l'actif. C'est la phase *"Monte Carlo"*.

2. **Phase Backward (Apprentissage)** :
   C'est là que l'intelligence opère. On remonte le temps, de la fin vers le début.
   - **Dilemme** : Comparer gain immédiat (connu) vs espérance de gain futur (inconnue).
   - **Régression** : On utilise toutes les trajectoires pour construire une courbe (polynôme) qui prédit cette espérance.
   - **Décision** : Si Gain immédiat $>$ Courbe $\rightarrow$ On exerce.



3.2. Illustration Visuelle du LSMC

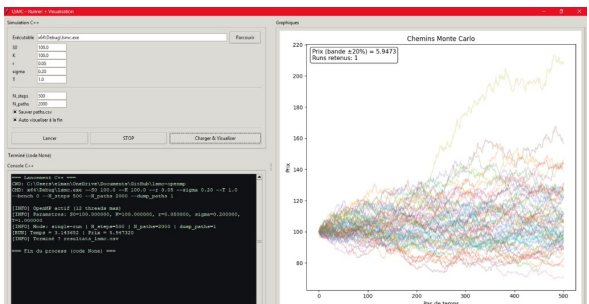


Source : Visualisation des trajectoires et de la frontière d'exercice

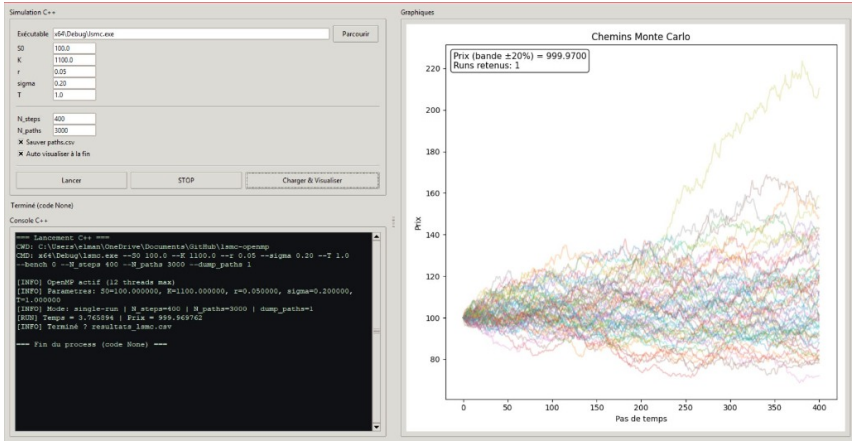
4. L'Interface Utilisateur (GUI)

Fonctionnalités principales :

- Paramétrage complet (S_0, K, r, σ, T)
- Console intégrée avec logs
- Visualisation intuitive (100 trajectoires)



4. L'Interface Utilisateur (GUI)



Test de charge : Visualisation fluide de 3000 trajectoires simultanées.

5.1. Du CPU Séquentiel au GPU Massif

Approche incrémentale :

1. **CPU C++** : Valider la logique ("Vérité Terrain")
2. **OpenMP** : Parallélisme multi-cœurs
3. **CUDA (GPU)** : Parallélisme massif

Pourquoi le GPU ?

- Monte Carlo = *"Embarrassingly Parallel"*
- Chaque trajectoire est indépendante
- 3072 cœurs CUDA (RTX 4060)

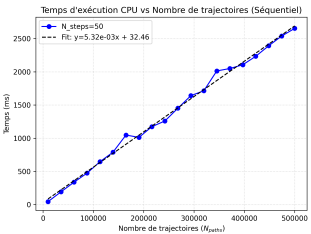
Architecture logicielle :

- Backend : C++/CUDA
- Build : CMake
- Frontend : GUI Python
- Visualisation en temps réel

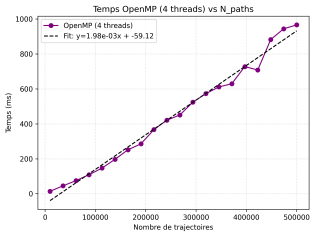
5.2. Validation de la Linéarité (Cohérence)

Vérification de la complexité algorithmique $O(N)$:

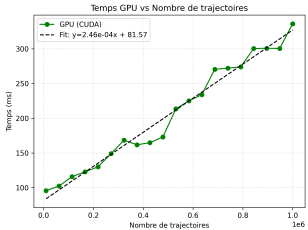
CPU Séquentiel



OpenMP



GPU CUDA

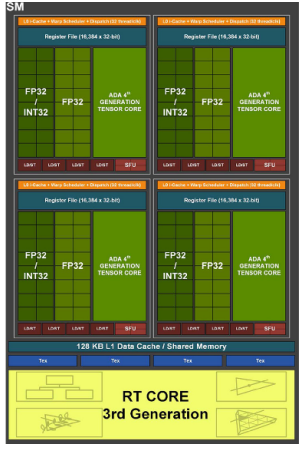


- **Cohérence** : Temps d'exécution proportionnel au nombre de trajectoires (lignes droites).
- Validation des 3 implémentations avant optimisation poussée.
- **Amélioration** : Le coefficient directeur est de plus en plus petit → On valide l'accélération effective du calcul.

6.1. Architecture GPU (Ada Lovelace)

Streaming Multiprocessor (SM) :

- **Unités de Calcul :**
  - Cœurs FP32/INT32.
  - Tensor Cores (N/A).
- **Ordonnancement :**
  - **Warps** (32 threads).
  - SIMT.
- **Mémoire :**
  - Registres (16K).
  - Cache L1 (128 KB).



6.2. Le Défi du GPU (Difficultés rencontrées)

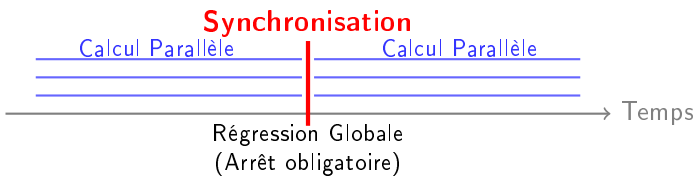
1. Le Paradoxe Algorithmique :

Parallélisme Spatial (OK)

- Simulation "Embarrassingly Parallel"
- 1M de chemins simultanés (Force du GPU)

Dépendance Temporelle (Bloquant)

- Backward Induction séquentiel ($t \to t-1$)
- Impossible de traiter $t-1$ avant la fin de t



2. Difficulté Technique (Outillage) :

- Échec intégration CUDA dans Visual Studio → Migration vers **CMake** (Plus robuste).

7.1. Comparaison des Performances

Mode	Pas (N)	Trajectoires (M)	Prix (€)	Temps (ms)	Écart/FDM
FDM Implicite	1000	-	6.067	0.67	Ref
FDM Explicite	1000	-	6.079	60.18	+0.012
FDM RK4	1000	-	6.079	231.88	+0.012
CPU Séquentiel	50	100,000	6.057	559.91	-0.010
OpenMP	50	100,000	6.057	540.23	-0.010
GPU CUDA	50	100,000	6.070	**41.96**	+0.003
CPU Séquentiel	50	1,000,000	6.059	6914.19	-0.008
OpenMP	50	1,000,000	6.059	6562.99	-0.008
GPU CUDA	50	1,000,000	6.047	**455.63**	-0.020
CPU Séquentiel	50	5,000,000	6.057	35352.25	-0.010

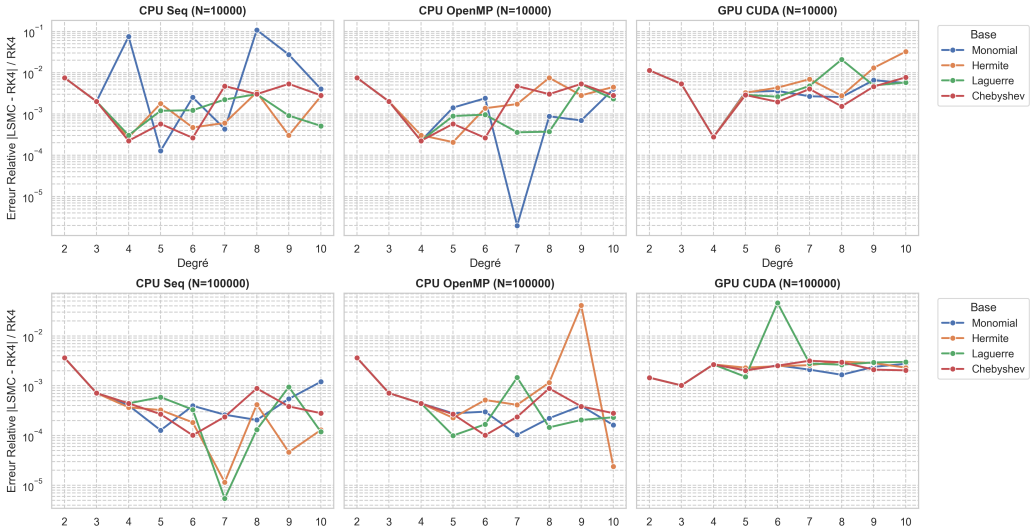
Speedup GPU : $\times 15$ pour 1M trajectoires (455ms vs 6914ms)

7.2. Analyse des Résultats

1. **Validité (Cohérence Mathématique)** :
   - Convergence vers les prix déterministes (Différences Finies).
   - Écart minime (quelques centimes), validant la justesse de l'algorithme.

2. **Performance (Scalabilité)** :
   - **Faible charge** : CPU suffisant pour peu de trajectoires.
   - **Haute charge (1M trajectoires)** :
     - CPU : ≈ 7 secondes (Lent).
     - GPU : 0.45 **seconde** (Quasi-instantané).
   - **Facteur d'accélération :** $\times 15$.

7.3. Influence du choix de la base de régression linéaire (Graphique)



7.4. Influence du choix de la base de régression linéaire (Analyse)

- **Comparaison des Bases** : Test de 4 familles de polynômes (Monomiale, Laguerre, Hermite, Chebyshev).
- **Impact du Degré (2 à 9)** :
  - **Stabilité** : Le prix calculé reste cohérent quel que soit le degré choisi.
  - Pas de sur-apprentissage notable ("Overfitting") observé ici.
- **Conclusion** : La méthode est robuste au choix des fonctions de base.
- **Contradiction Théorique** : *"Le temps devrait décroître en allure avec le degré"* (Théorie). Ce n'est pas observé, signalant une anomalie.

7.5. Synthèse des Difficultés Rencontrées

- **Complexité Théorique** :
  - Compréhension fine de l'*Arrêt Optimal* (Exercice Anticipé).
  - Choix délicat de la base de régression (Compromis Biais-Variance, Phénomène de Runge).

- **Défis Techniques** :
  - **Intégration CUDA** : Échec sous Visual Studio → Migration complète vers **CMake**.
  - **Synchronisation GPU** : Barrière temporelle inévitable de la *Backward Induction* (limite de la loi d'Amdahl).

- **Méthodologie & Apprentissage** :
  - **Documentation Technique** : Investissement temps important pour maîtriser la documentation NVIDIA (Modèle mémoire, Warps) et CMake.
  - Refonte de l'architecture logicielle pour séparer proprement CPU (Logique) et GPU (Calcul).

7.6. Organisation & Bilan Personnel

Chronologie du Projet (5 Phases) :

1. **Octobre** : Étude théorique LSMC & Prototype CPU Séquentiel.
2. **Novembre** : Restructuration (Classes), Parallélisme OpenMP.
3. **Décembre** : Interface Python (GUI), Validation, Migration CMake, Implémentation CUDA ("Cœur Calcul").
4. **Janvier** : Optimisation fine, Benchmarks & Rédaction.

Investissement Personnel :

- **Florian Barbe ($\approx$ 40h) :**
  - Architecture Technique (C++/CUDA).
  - Optimisation Mémoire & GPU.
  - Gestion Build (CMake) & Git.
- **Narjisse El Manssouri ($\approx$ 35h) :**
  - Analyse Mathématique & Modèle.
  - Conception Interface Utilisateur (Python).
  - Validation Théorique & Rédaction.

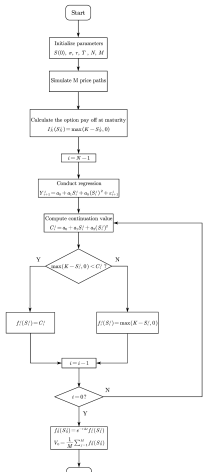
8. Apports et Perspectives

- **Bilan Performance** :
  - **Succès** : Validation d'un pricer performant ($\times 15$ sur GPU).
  - **Constat** : Gains OpenMP limités (Bande passante mémoire, cf. Annexes).
- **Conclusion Structurelle (LSMC)** :
  - **Paradoxe** : Parallélisme spatial excellent vs Blocage temporel inévitable.
  - L'accélération est intrinsèquement bornée par la *backward induction* séquentielle.
- **Perspectives d'Avenir** :
  - **HPC** : Approches hybrides (Batching d'options) pour saturer le GPU et contourner la barrière séquentielle.
  - **Comparaison** : Confrontation avec d'autres méthodes (Arbres de décision, Transformées de Fourier).
  - **IA Haute Dimension** : Remplacement des polynômes par des Réseaux de Neurones (Deep Learning) pour gérer les paniers d'actifs (50+).

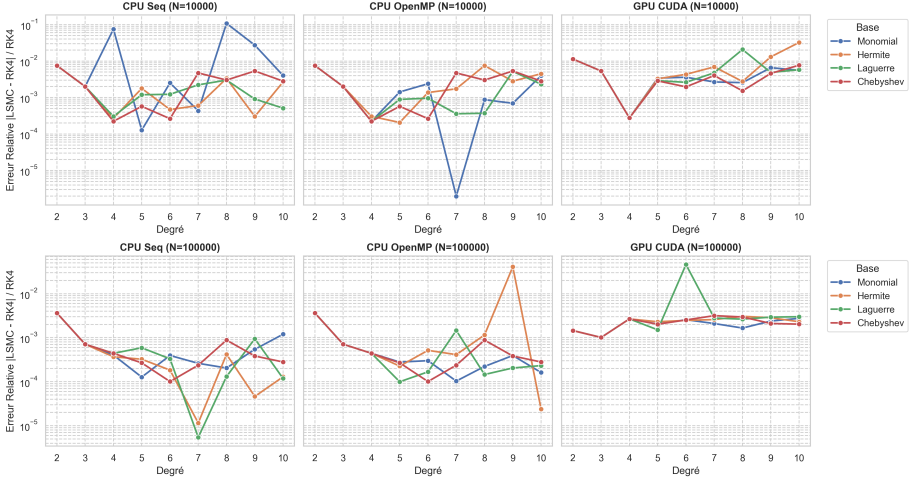
Merci de votre attention !

Avez-vous des questions ?

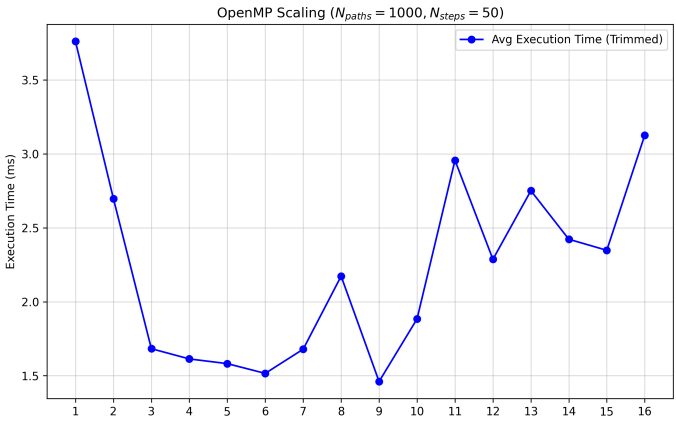
Annexe : Illustration LSMC



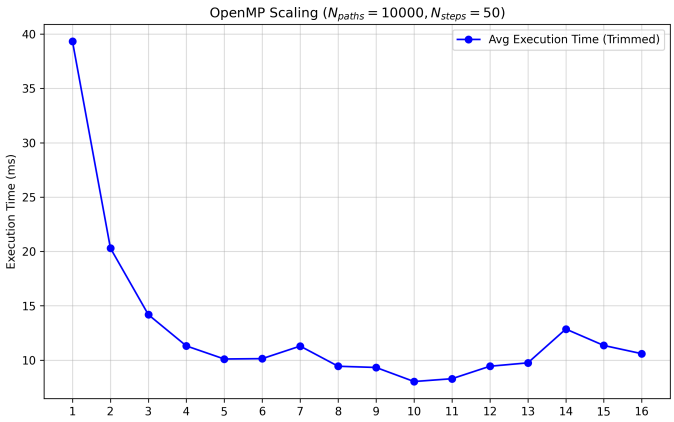
Annexe : Convergence du Prix



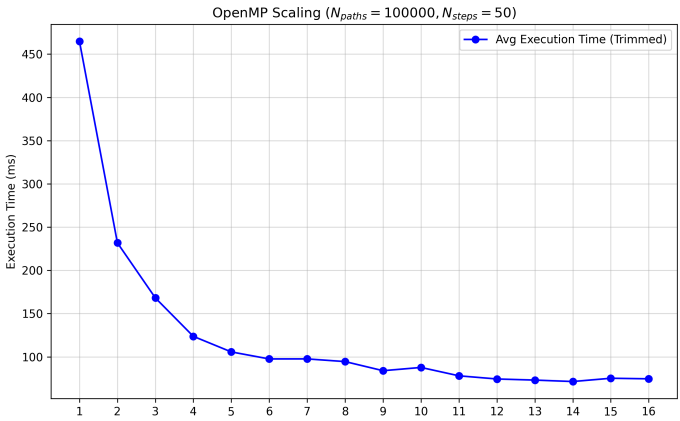
Annexe : Scalabilité OpenMP (N=1000)



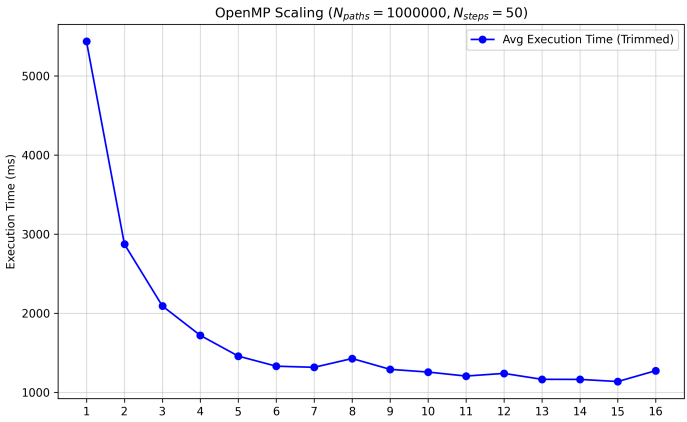
Annexe : Scalabilité OpenMP (N=10000)



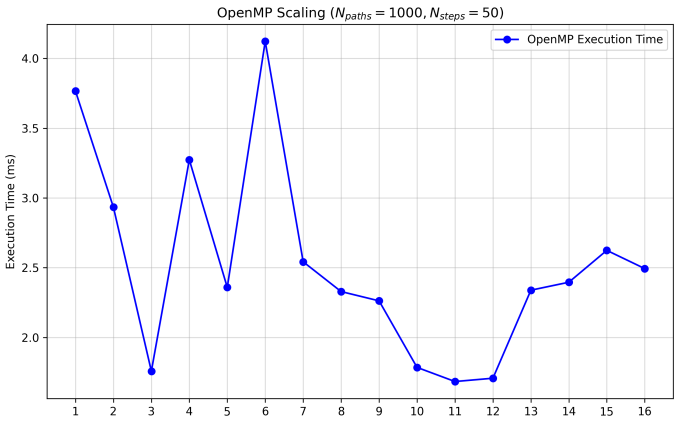
Annexe : Scalabilité OpenMP (N=100000)



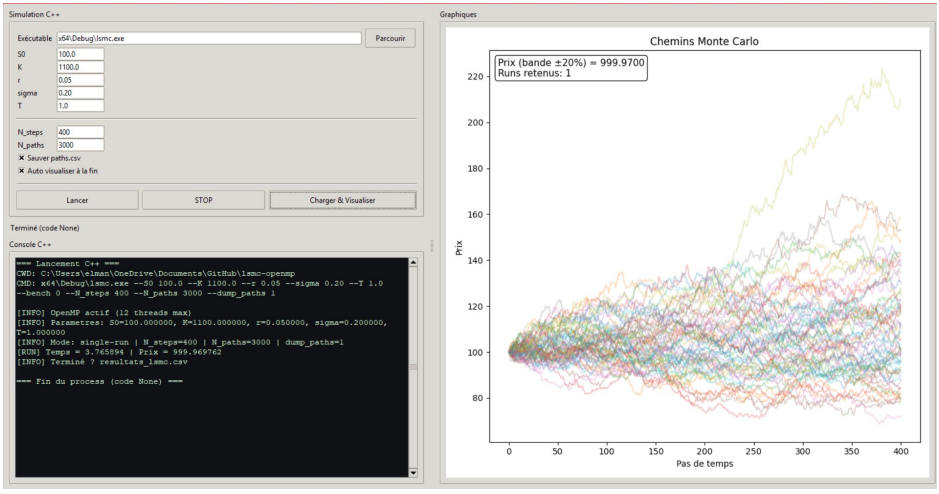
Annexe : Scalabilité OpenMP (N=1000000)



Annexe : Scalabilité OpenMP Raw (N=1000)



Annexe : Photo Projet



Annexe A. Diagramme de Classes Simplifié

(Voir fichier `diagramme_classes.md` pour le diagramme de classes)