

Questions Réponses Anticipées – Soutenance P1RV

F. Barbe – N. El Manssouri

Janvier 2026

1. Méthode LSMC & Mathématiques

Q: Pourquoi utiliser une régression (Moindres Carrés) ?

R: L'objectif est d'estimer l'**espérance conditionnelle** de la valeur de continuation $E[V_{t+1}|S_t]$. Comme on ne connaît pas la forme analytique de cette fonction, on l'approxime par une projection sur une base de fonctions simples (polynômes) en utilisant les milliers de trajectoires simulées comme nuage de points.

Q: Pourquoi l'algorithme "remonte-t-il le temps" (Backward Induction) ?

R: C'est propre aux options américaines (problème d'arrêt optimal). Pour savoir si je dois exercer en t , je dois comparer le gain immédiat avec ce que je gagnerais en attendant. Or, "ce que je gagne en attendant" dépend de ce qui se passera en $t + 1$. Il faut donc avoir résolu le problème en $t + 1$ (et donc $t + 2$, etc.) avant de pouvoir le résoudre en t .

Q: Comment avez-vous choisi les polynômes (Laguerre, Hermite...) ?

R: Nous avons testé plusieurs bases. Théoriquement, Laguerre est adapté aux variables positives ($[0, \infty[$) et Hermite aux variables gaussiennes ($]-\infty, \infty[$). En pratique, sur nos tests, la base Monomiale simple suffisait pour converger, mais Hermite donnait une matrice un peu mieux conditionnée. Le gain de précision n'était pas flagrant par rapport au bruit de Monte Carlo.

Q: Pourquoi ne pas utiliser plus de fonctions de base (Degré > 10) ?

R: Cela mènerait au **sur-apprentissage** (overfitting) : la régression capturerait le bruit aléatoire des trajectoires au lieu de la tendance générale. De plus, les matrices deviendraient mal conditionnées (instables numériquement).

2. GPU & Performance (CUDA)

Q: Vous avez 3000 cœurs, pourquoi "seulement" x15 et pas x3000 en speedup ?

R: C'est la **loi d'Amdahl**. L'algorithme a une partie séquentielle incompressible : la boucle temporelle (Backward Induction). On ne peut pas paralléliser le temps. Le GPU n'accélère que

la partie spatiale (simulation des trajectoires). De plus, il y a des coûts fixes de transfert mémoire et de latence de lancement des kernels qui "mangent" le gain sur les petits calculs.

Q: Quel est le goulot d'étranglement (Bottleneck) actuel ?

R: La **latence de la régression** à chaque pas de temps. À chaque date t , le GPU doit attendre que l'on ait construit la matrice et résolu le système linéaire (souvent fait sur CPU ou avec des réductions coûteuses) avant de passer à $t - 1$. C'est une barrière de synchronisation.

Q: Avez-vous utilisé la mémoire partagée (Shared Memory) ?

R: Oui, pour les réductions (calcul des sommes X , X^2 , XY pour la régression). Chaque bloc calcule une somme partielle en mémoire partagée rapide, puis on agrège le tout. Sans ça, les accès à la mémoire globale seraient trop lents.

3. Finance & Modélisation

Q: Quelle est la différence fondamentale entre option Américaine et Européenne ?

R: L'**exercice anticipé**. Une option européenne ne s'exerce qu'à la fin (T). Une américaine peut s'exercer n'importe quand. Cela ajoute de la valeur (prime d'illiquidité/flexibilité) et rend le pricing beaucoup plus dur (pas de formule fermée simple type Black-Scholes).

Q: Votre modèle assume une volatilité constante, est-ce réaliste ?

R: Non, c'est le modèle de Black-Scholes classique. En réalité, il y a un "smile" de volatilité. Pour plus de réalisme, il faudrait utiliser un modèle à volatilité locale (Dupire) ou stochastique (Heston), ce qui est tout à fait faisable avec LSMC (il suffit de changer la formule de simulation des trajectoires).

Q: Votre prix converge-t-il vers la "vraie" valeur ?

R: Il converge vers la valeur théorique du modèle (validé par FDM). Cependant, c'est un estimateur **biaisé bas** (Low biased) car on utilise une approximation sous-optimale de la frontière d'exercice (on exerce "moins bien" que la théorie parfaite, donc on sous-estime légèrement la valeur de l'option).

4. Gestion de Projet & Difficultés

Q: Quelle a été une difficulté majeure inattendue ?

R: La **longue lecture de la documentation** technique. Avant de produire du code efficace, il a fallu passer beaucoup de temps à décrypter la documentation officielle de NVIDIA (pour comprendre le modèle mémoire unifié, les warps) et de CMake (pour la compilation hybride C++/CUDA). C'est un investissement invisible mais crucial.