



Rapport du projet de C++

Ralph NADER et Benjamin BENISTI

24 décembre 2025

Introduction

Pour ce projet, notre choix s'est porté sur la réalisation d'un **pricer de produits dérivés par résolution d'EDP** (ou équations aux dérivées partielles). Cette décision s'inscrit dans la continuité de ce qui nous a motivé à suivre cet électif de programmation en C++; à savoir la volonté de poursuivre une carrière dans le monde de la finance. Étant fréquemment utilisé dans les systèmes de trading, et d'autant plus au sein du trading de haute fréquence (ou HFT), ce premier travail d'application en C++ constitue une amorce cohérente quant aux champs d'utilisation de ce langage.

La variété des options traitées nous a poussé à implémenter deux méthodes de valorisation: d'une part, celle basée sur la résolution du modèle de Black-Scholes-Merton et de l'autre, une alternative définie suivant la méthode de Monte Carlo. Notre analyse est complétée par la considération du temps d'exécution (illustrant à la fois les différences entre modèles mais aussi l'intérêt du C++ de façon générale) ainsi que d'outils de visualisation quand cela est possible.

Comme indiqué dans la table des matières ci-dessous, ce rapport vise à être un livrable tout-en-un relatif à ce projet. Il comportera notamment un guide d'utilisation (dont les commandes permettant de reproduire l'ensemble des graphes et/ou tables présentés), une description étaillée des modèles utilisés et les résultats principaux en découlant. Une dernière partie s'attardera davantage sur les difficultés rencontrées, nos solutions ainsi qu'une prise de recul sur notre projet et ses limitations.

| | |
|--|-----------|
| 1 Présentation et architecture du programme | 2 |
| 1.1 Objectifs et cadre du projet | 2 |
| 1.2 Fonctionnalités | 2 |
| 1.3 Organisation du dossier "Code" | 3 |
| 1.4 Architecture générale et flux de données | 4 |
| 2 Guide d'utilisation | 5 |
| 2.1 Prérequis et environnement | 5 |
| 2.2 Compilation | 5 |
| 2.3 Paramètres personnalisables | 5 |
| 2.4 Scénarios d'usage typiques | 7 |
| 3 Contexte théorique | 8 |
| 3.1 Produits dérivés utilisés | 8 |
| 3.2 Modèle de Black-Scholes-Merton et schémas numériques EDP | 9 |
| 3.3 Méthode de Monte Carlo | 10 |
| 4 Résultats et validation | 12 |
| 4.1 Présentation des résultats par type d'option | 12 |
| 4.2 Comparaison des modèles et des performances | 15 |
| 5 Commentaires | 17 |
| 5.1 Problèmes rencontrés et solutions adoptées | 17 |
| 5.2 Limitations et pistes d'amélioration | 17 |
| 5.3 Conclusion | 18 |
| Annexes | 19 |

1 Présentation et architecture du programme

1.1 Objectifs et cadre du projet

Lorsqu'il s'agit de déterminer la valeur d'un bien et plus spécifiquement dans notre cas, d'un produit dérivé, il importe tout d'abord de définir le cadre dans lequel est conçu notre pricer. Le marché dans lequel sont échangées nos options est décrit par les caractéristiques suivantes.

Hypothèses :

- **Parfaite information** : tous les paramètres (S_0 , K , r , σ , T) sont connus.
- **Absence d'arbitrage** : il n'existe pas de stratégie auto-financante permettant un gain certain sans risque initial.
- **Actifs parfaitement fongibles et liquides** : possibilité d'acheter ou vendre n'importe quelle quantité instantanément au prix du marché, sans impacter le prix.
- **Absence de coûts de transaction** : la couverture n'engendre aucun coût ici.
- **Possibilité d'emprunter et/ou de prêter au taux sans risque**.
- **Trading continu** : le portefeuille peut être rééquilibré en continu dans le temps, sans délai ni latence.
- **Volatilité ' σ ' constante** : ne dépend ni du temps ni de la valeur du spot. On ne considérera donc pas le sourire de volatilité.
- **Taux sans risque ' r ' constant** : il est fixe sur l'horizon considéré, et il n'y aura donc pas de courbe de taux multiple.
- **Absence de dividendes** : le sous-jacent ne verse pas de flux (par souci de simplification).

L'ensemble des caractéristiques énoncées ci-dessus, *ainsi que la nécessité pour le sous-jacent de suivre une loi lognormale*, correspond aux hypothèses nécessaires à l'application du modèle de Black-Scholes que nous présenterons par la suite. Comme mentionné brièvement durant l'introduction, les objectifs du projet sont les suivants:

- Implémenter des solveurs EDP pour résoudre l'équation de Black-Scholes à l'aide de schémas de différences finies.
- Traiter l'exercice anticipé des options américaines via un algorithme de frontière libre.
- Proposer une approche suivant la méthode de Monte Carlo pour non seulement apporter une validation croisée mais aussi pricer des options plus complexes.
- Comparer la performance et la précision des différentes méthodes utilisées ainsi que fournir un outil de visualisation en vue d'enrichir notre analyse.

1.2 Fonctionnalités

Le programme offre plusieurs modes de pricing et types de produits dérivés dont le récapitulatif se trouve sur la figure 1. Cette dernière indique pour chaque option quels sont les algorithmes pouvant déterminer sa valorisation. En parallèle du prix, le programme renverra le temps de calcul pris par l'ordinateur à l'aide de la bibliothèque <chrono>. Enfin, nous proposons aussi des visualisations 3D de l'évolution du prix de certaines options. Même si cela sera discuté plus en détails ultérieurement (dans la section Limitations et pistes d'amélioration), seuls les produits dérivés pouvant être résolus par EDP disposent du support graphique.

| Type d'option | Explicite | Implicite | Crank–Nicolson | Solveur américain | Monte Carlo |
|---------------|-----------|-----------|----------------|-------------------|-------------|
| European | ✓ | ✓ | ✓* | ✗ | ✓ |
| American | ✗ | ✗ | ✗ | ✓ | ✗ |
| Barrier | ✓ | ✓ | ✓* | ✗ | ✓ |
| Digital | ✓ | ✓ | ✓* | ✗ | ✗ |
| Asian | ✗ | ✗ | ✗ | ✗ | ✓ |
| Lookback | ✗ | ✗ | ✗ | ✗ | ✓ |
| Chooser | ✗ | ✗ | ✗ | ✗ | ✓ |

Compatibilité : ✓ = supporté, ✗ = non implémenté, * = recommandé (si choix multiple)

Figure 1 Support des options par les différents solveurs

1.3 Organisation du dossier "Code"

Voici l’arborescence du programme ci-dessous. Comme on peut le noter ce dernier suit une structure classique avec d’une part **les déclarations** (qui correspondent aux fichiers .hpp) et **leurs implémentations** (.cpp).

```
/Code
|-- include/
|   |-- AmericanOption.hpp          # Headers (.hpp)
|   |-- AmericanSolver.hpp         # Classe option américaine
|   |-- CrankNicolsonSolver.hpp    # Solveur spécialisé (frontière libre)
|   |-- ExoticOption.hpp           # Classes options exotiques
|   |-- ExplicitSolver.hpp         # Schéma explicite
|   |-- Grid.hpp                   # Grille (spatiale + temps)
|   |-- ImplicitSolver.hpp        # Schéma implicite
|   |-- MarketData.hpp            # Paramètres du marché
|   |-- MonteCarlo.hpp             # Modèle Monte Carlo
|   |-- PDESolver.hpp              # Classe abstraite - solveurs EDP
|   `-- VanillaOption.hpp          # Classe option vanille européenne

|-- src/                           # Implementations (.cpp)
|   |-- AmericanOption.cpp
|   |-- AmericanSolver.cpp
|   |-- CrankNicolsonSolver.cpp
|   |-- ExplicitSolver.cpp
|   |-- Grid.cpp
|   |-- ImplicitSolver.cpp
|   |-- main.cpp
|   |-- MonteCarlo.cpp
|   |-- PDESolver.cpp
|   `-- VanillaOption.cpp

|-- option_viz                     # Binaire executable
|-- output_grid.csv                # Grille complète prix V(S,t)
|-- upin_call_B120.csv              # Exemple pour un barrière call
|-- readme.md                       # Documentation utilisateur
`-- option_viz.dSYM/
    |-- Contents/
    |   |-- Info.plist
    |   |-- Resources/
```

Figure 2 Arborescence du programme

Le choix de cette structure suit les notions vues en cours, permettant de préserver la clarté des contrats ainsi que d'optimiser le processus de compilation (uniquement sur les fichiers .cpp concernés). Cela se retrouve notamment dans le choix de reprendre les principes d'**héritage et de polymorphisme**. La classe abstraite *PDESolver* en est l'illustration en jouant à la fois le rôle d'interface pour l'ensemble des solveurs d'EDP mais aussi en les manipulant comme pointeur unique. De son côté, le binaire exécutable *option_viz* est obtenu après linkage et conclut donc le fonctionnement d'un programme codé en C++.

1.4 Architecture générale et flux de données

Pour revenir plus en détail sur l'architecture, cette dernière suit une structure orientée objet. La distinction est claire avec une séparation entre les classes d'**options** (*VanillaOption*, *AmericanOption*, et *ExoticOption*) et de l'autre celles des **solveurs** (*PDESolver*, *ExplicitSolver*, *ImplicitSolver*, *CrankNicolsonSolver*, et *AmericanSolver*). Les autres modules correspondent aux **données de marché** (*MarketData*), à la **grille stockant les valeurs spatiotemporelles** (*Grid*) et enfin celui de **Monte Carlo** (*MonteCarlo*). Enfin le fichier **main.cpp** constitue le point d'entrée du programme et vient coordonner l'ensemble des modules cités juste avant.

La présence des fichiers **.csv** réside dans le processus de visualisation de données au sein de ce projet. En effet, cette dernière contient les valeurs de prix sur la grille spatiotemporelle discrétisée et via Gnuplot, va permettre la génération de surfaces 3D.

La figure suivante permet d'avoir une vision schématisée du flux de données. Pour chaque option, on définit ses caractéristiques auxquelles on ajoute les propriétés du marché en entrée du solveur. Ce dernier renvoie les prix obtenus aux différents points de la grille.

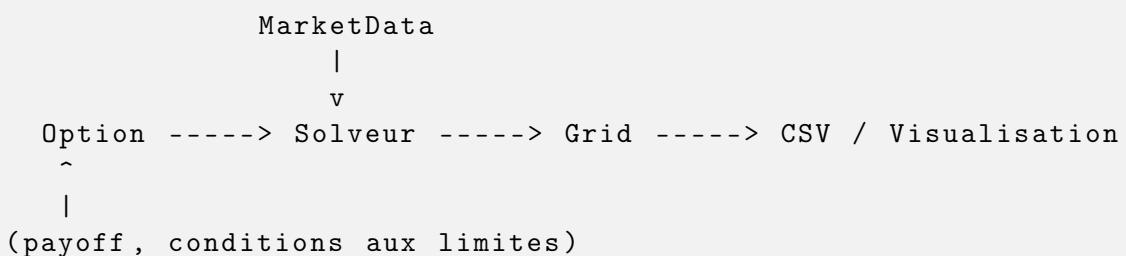


Figure 3 Flux de données

2 Guide d'utilisation

2.1 Prérequis et environnement

Le programme nécessite un environnement minimal pour fonctionner correctement, mis à part l'aspect graphique qui demande l'installation d'un logiciel. Le choix de Gnuplot comme outil de visualisation est justifié à plusieurs niveaux : étant à la fois l'option la plus accessible pour générer les graphes tout en restant en C++ mais aussi du fait de sa forte compatibilité avec LaTeX.

Prérequis :

- **Compilateur C++17** : clang++ (macOS/Linux) ou g++ version 7+
- **Système d'exploitation** : macOS, Linux ou Windows
- **Bibliothèques** : uniquement la bibliothèque standard C++.
- **Gnuplot** version 5.0+ pour la visualisation 3D.

Installation de Gnuplot :

Sur macOS via Homebrew:

```
brew install gnuplot
```

Puis vérifier l'installation avec :

```
gnuplot --version
```

Sur Linux via le gestionnaire de paquets APT:

```
sudo apt update
sudo apt install gnuplot
```

Sur Windows en téléchargeant directement depuis <http://www.gnuplot.info/download.html> et en ajoutant le répertoire d'installation au PATH système.

2.2 Compilation

Le programme se compile à l'aide d'une seule commande réalisée depuis le répertoire "Code", à savoir :

```
clang++ -std=c++17 -O2 -I./include src/*.cpp -o option_viz
```

Si la compilation est faite avec GCC, il suffit de remplacer *clang++* par *g++* comme ceci :

```
g++ -std=c++17 -O2 -I./include src/*.cpp -o option_viz
```

2.3 Paramètres personnalisables

Dans cette section, vous trouverez l'ensemble des paramètres qu'il est possible de modifier avec leur description ainsi qu'une idée des ordres de grandeur à utiliser. La liste peut sembler conséquente à première vue mais des exemples plus concrets seront donnés dans la section suivante de sorte à rendre la prise en main plus intuitive. Pour rappel, **chaque résultat présenté dans ce rapport sera accompagné de la commande qui a permis de l'obtenir**.

| Paramètre | Valeur par défaut | Fourchette réaliste | Remarques |
|-----------|-------------------|---------------------|--|
| -strike | 100 | 50–500 | Dépend du sous-jacent. |
| -spot | 100 | 50–500 | Généralement proche du strike pour ATM. |
| -T | 1.0 | 0.08–5.0 | 0.08 correspond à 1 mois, 0.25 à 3 mois, 1.0 = 1 an, ... |
| -r | 0.05 | 0.0–0.10 | 0.0 = taux zéro, 0.03–0.05 = normal, 0.08–0.10 = élevé |
| -sigma | 0.2 | 0.10–0.80 | 0.10–0.15 = faible, 0.20–0.30 = moyen, 0.40–0.80 = élevé |

Figure 4 Paramètres financiers

| Paramètre | Valeur par défaut | Fourchette usuelle | Contraintes |
|-----------|-------------------|--------------------|---|
| -M | 200 | 50–2000 | Min : 50, rec. : 200–500 |
| -N | 2000 | 500–20000 | Dépend de CFL pour explicite. |
| -Smax | 150 | 2K–5K | $S_{\text{max}} \geq 2\text{--}3 \times K$ recommandé |

Figure 5 Paramètres de discréétisation EDP**Règles pratiques :**

■ **Ratio M/N** : $M/N \approx 10 \times N$ pour Crank-Nicolson. Cela offre généralement un bon équilibre entre précision et vitesse.

■ **Exemples de configurations :**

Rapide : $M = 100, N = 1000$ ($\sim 0.05\text{s}$, précision 1%)

Standard : $M = 200, N = 2000$ ($\sim 0.15\text{s}$, précision 0.1%)

Précis : $M = 500, N = 5000$ ($\sim 1\text{s}$, précision 0.01%)

Haute précision : $M = 1000, N = 10000$ ($\sim 6\text{s}$, précision 0.001%)

| Solveur | Valeur | Stabilité | Usage recommandé |
|----------------|----------|------------------------------------|--------------------------------------|
| Explicite | explicit | Conditionnelle (CFL ¹) | Utile que pour debug |
| Implicite | implicit | Inconditionnelle | Stabilité critique |
| Crank-Nicolson | cn | Inconditionnelle | À choisir par défaut. |
| Américain | american | Inconditionnelle | Options américaines |
| Monte Carlo | mc | N/A | Pour validation ou options exotiques |

Figure 6 Choix du solveur

| Paramètre | Typique | Fourchette | Remarques |
|---------------|---------|---------------|---|
| -barrierType | upout | downout/upout | Upout=désactivé si $S \geq B$, downout=si $S \leq B$ |
| -barrierLevel | 120 | 110–150 | Niveau de la barrière B. |
| -digitalCash | 1.0 | 0.1–100 | Payout fixe indépendant de S . |
| -lookbackType | max | min/max | call sur $\max(S)$, put sur $\min(S)$ |
| -chooseTime | 0.5 | 0.1–0.9 | Fraction de T : 0.5=mi-maturité. |

Figure 7 Paramètres d'options exotiques

¹CFL = Condition de Courant–Friedrichs–Lewy. Elle impose $\Delta t \leq \Delta S^2 / (\sigma^2 S_{\text{max}}^2)$.

| Paramètre | Typique | Fourchette | Remarques |
|------------------|-----------------|---------------|---|
| -mcPaths | 100000 | 10000–5000000 | Nombre de trajectoires simulée (erreur $\propto 1/\sqrt{N}$). |
| -mcSteps | 252 ou 365 | 50–1000 | Pas de temps par trajectoire (rappel : 252 = jours de trading). |
| -mcSeed | 0 | 0 | Graine pseudo-aléatoire. |
| -mcAntithetic | on | on/off | Réduit variance de 20–40%, jamais désactivée. |
| -mcJumpIntensity | 0.0 (désactivé) | 0.0–10.0 | Intensité de sauts ' λ ' (processus de Poisson). |
| -mcJumpMean | -0.05 | -0.20 à 0.10 | Moyenne log des tailles de saut. |
| -mcJumpVol | 0.10 | 0.05–0.30 | Volatilité des amplitudes de sauts. |

Figure 8 Paramètres Monte Carlo

| Paramètre | Défaut | Alternatives | Usage |
|-------------------|-----------------|------------------|--|
| -out | output_grid.csv | Tout chemin .csv | Export grille complète. |
| -earlyBoundaryOut | (vide) | frontier.csv | Export $S^*(\tau)$ américaines uniquement. |
| -plot | on | on/off | Gnuplot requis si on |
| -spin | off | on/off | Rotation 3D automatique. |
| -clampK | -1.0 (auto) | 0–500 | Fixe échelle z/couleur de [0:K] |

Figure 9 Paramètres de sortie

2.4 Scénarios d'usage typiques

Put européen basique avec rotation :

```
./option_viz --solver mc --option european --type put \
--spot 100 --strike 100 --T 1.0 --r 0.05 --sigma 0.2 \
--mcPaths 100000 --mcSteps 252 --plot off --spin on
```

Call avec modèle à saut de Merton :

```
./option_viz --solver mc --option european --type call \
--spot 100 --strike 105 --T 1.0 --r 0.05 --sigma 0.15 \
--mcPaths 200000 --mcSteps 365 \
--mcJumpIntensity 2.0 --mcJumpMean -0.05 --mcJumpVol 0.1 \
--plot off
```

Option exotique barrière avec méthode de Monte Carlo :

```
./option_viz --solver mc --option european --type call \
--spot 100 --strike 100 --barrierType upout --barrierLevel
120 \
--T 1.0 --r 0.05 --sigma 0.3 \
--mcPaths 150000 --mcSteps 500 --plot off
```

3 Contexte théorique

3.1 Produits dérivés utilisés

Rappels :

- **Produit dérivé** : instrument financier dont la valeur dépend (dérive) d'un actif sous-jacent qui doit être mesurable.
- **Option** : contrat donnant à l'acheteur le droit, mais pas l'obligation, d'acheter (call) ou vendre (put) un actif sous-jacent à un prix fixé (strike K) à une date donnée (maturité T) ou avant cette date (suivant si type européen ou américain).

Terminologie :

- **Call** : a pour payoff $\max(S_T - K, 0)$.
- **Put** : a pour payoff $\max(K - S_T, 0)$.

Moneyness :

- **In-the-money (ITM)** : exercice profitable immédiatement.
- **At-the-money (ATM)** : $S \approx K$.
- **Out-of-the-money (OTM)** : exercice non profitable.

Options vanilles :

Européenne :

Ne peut être exercée qu'à maturité T. Il s'agit du type d'option le plus simple (et dispose de la formule analytique de Black-Scholes).

Américaine :

Peut être exercée à tout instant $t \in [0, T]$. Cela rajoute un degré de complexité avec une optimisation de formes (frontières libres).

Propriété importante : sa valeur est toujours supérieure à une option européenne équivalente (du fait de la flexibilité supplémentaire accordée).

Options exotiques :

Barrière :

Activation/désactivation **si le sous-jacent atteint un niveau "barrière"**.

Up-and-in (barrière haute): active si $S \geq B$.

Down-and-out (barrière basse): désactive si $S \leq B$.

Asiatique :

Payoff dépend de la **moyenne du sous-jacent sur $[0, T]$** .

Pour un call, la formule devient $\max(\bar{S} - K, 0)$ où $\bar{S} = \frac{1}{T} \int_0^T S_t dt$

Lookback :

Payoff basé sur **max/min historique**.

Pour un put sur un maximum, la formule devient : $\max(S_{\max} - S_T, 0)$ où $S_{\max} = \max_{t \in [0, T]} S_t$.

Pour un call sur un minimum, la formule devient : $\max(S_T - S_{\min}, 0)$ où $S_{\min} = \min_{t \in [0, T]} S_t$.

Digitale (ou binaire) :

Payoff **si condition remplie**.

Matérialisé par une indicatrice : $\mathbb{1}_{S_T > K}$ pour un call.

Chooser :

Possibilité de choisir à une date $t \in [0, T]$ si l'option est un call ou un put.

Au temps de choix : valeur = $\max(V_{\text{call}}(S_t, T-t), V_{\text{put}}(S_t, T-t))$.

À l'échéance, on retrouve les payoffs classiques pour un call/put.

3.2 Modèle de Black-Scholes-Merton et schémas numériques EDP

Équation de Black–Scholes–Merton :

Pour une option sur un actif non-dividende, le prix $V(S, t)$ satisfait l'EDP suivante :

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV = 0,$$

avec les conditions terminales $V(S, T)$ et aux bornes adaptées au type d'option. Elle se décompose de la sorte:

- un terme de diffusion (impact volatilité): $\frac{1}{2}\sigma^2 S^2 V_{SS}$
- un terme de convection : rSV_S
- un terme discount (actualisation continue): $-rV$

Le modèle de Black–Scholes–Merton est résolu numériquement en discréteisant le domaine du sous-jacent $[0, S_{\max}]$ en M points et l'intervalle de temps $[0, T]$ en N points. On se propose ici d'effectuer cette résolution suivant 3 schémas de différence finie. Tout d'abord, ceux explicites qui avancent la solution dans le temps à l'aide d'un pas de temps Δt , mais qui nécessitent de respecter une condition de stabilité (CFL) pour éviter les oscillations numériques. Dans un second temps, ceux implicites, qui quant à eux, sont inconditionnellement stables et consistent à résoudre un système tridiagonal à chaque pas de temps, typiquement via l'algorithme de Thomas. Enfin, le schéma de Crank–Nicolson combine les avantages des schémas explicite et implicite, offrant un ordre 2 en temps et en espace impliquant une meilleure précision. Néanmoins, ce dernier nécessite également la résolution d'un système tridiagonal.

Pour les options américaines, la flexibilité d'exercice anticipé est prise en compte en comparant, à chaque pas de temps, la valeur obtenue par le schéma implicite avec le payoff immédiat de l'option, selon la formule **max(payoff, valeur implicite)**. Cette projection assure que la solution respecte la condition d'inégalité caractéristique des options américaines, ce qui rend leur valorisation plus complexe que celle des options européennes.

Remarque : Afin d'éviter de surcharger le rapport de formules techniques, les conditions aux limites ainsi que les formules et méthodes de résolution des différents solveurs d'EDP seront mis en annexe.

3.3 Méthode de Monte Carlo

La méthode de Monte Carlo consiste à estimer le prix d'une option comme l'espérance sous la mesure risque-neutre des flux futurs actualisés :

$$V_0 = e^{-rT} \mathbb{E}[\text{payoff}(S_T)],$$

où S_T suit un processus géométrique brownien défini par

$$dS_t = rS_t dt + \sigma S_t dW_t,$$

avec W_t un mouvement brownien standard.

Le problème étant qu'il ne nous est pas possible de calculer cette espérance analytiquement (sauf dans des cas simples), d'où la nécessité de l'approximer par la loi des grands nombres :

$$V_0 \approx \frac{1}{M} \sum_{m=1}^M e^{-rT} P(S_T^{(m)}),$$

où M est le nombre de trajectoires simulées.

La simulation consiste aussi à générer un grand nombre de trajectoires indépendantes du sous-jacent S_t en discrétisant le temps en pas de taille $\Delta t = T/N$. À chaque pas :

$$S_{k+1} = S_k \exp\left((r - \frac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t} Z_k\right),$$

avec $Z_k \sim \mathcal{N}(0, 1)$ i.i.d. (ce qui correspond à une erreur de discréétisation : $\mathcal{O}(\Delta t)$). La valeur de l'option est ensuite estimée par la moyenne empirique des payoffs actualisés à $t = 0$.

Cette approche est particulièrement flexible pour les options exotiques, mais sa convergence est lente, avec une erreur statistique d'ordre $\mathcal{O}(1/\sqrt{M})$, nécessitant un nombre important de simulations pour obtenir une précision satisfaisante.

Générateur de nombres aléatoires : PCG32

Pour améliorer la qualité des trajectoires simulées, on s'est penché sur le générateur PCG32 (Permuted Congruential Generator, 32 bits) a été développé par Melissa O'Neill en 2014. Cette génération est réalisée en 3 étapes :

- **Mise à jour de l'état** à l'aide d'un générateur congruentiel linéaire (LGC) : $state \leftarrow state \times c + inc$.
- **Permutation XOR-shift** : suivie d'une rotation pour produire un entier 32 bits de haute qualité.
- **Conversion en nombre uniforme (0,1)**.

Il s'agit donc d'un générateur rapide et fiable afin de produire des nombres qui paraissent complètement aléatoires. Cela permet des simulations de Monte Carlo aboutissant à de meilleures estimations des prix. À ce premier niveau d'amélioration, on a décidé d'intégrer un degré de précision supplémentaire au travers de deux méthodes activables ou non.

Sauts de Merton

L'idée est de rajouter des sauts ponctuels aux rendements du sous-jacent afin de capturer les mouvements brusques. Si les sauts sont activés (intensité λ) :

$$\text{drift} = (r - q - \frac{1}{2}\sigma^2 - \lambda\kappa)\Delta t, \quad \kappa = e^{\mu_J + \frac{1}{2}\sigma_J^2} - 1,$$

$$S_{k+1} = S_k \exp\left(\text{drift} + \sigma\sqrt{\Delta t} Z_k + \text{jumpShock}\right),$$

où $\text{jumpShock} = N_k \mu_J + \sqrt{N_k} \sigma_J Z_{\text{jump}}$, avec $N_k \sim \text{Poisson}(\lambda \Delta t)$.

On notera que l'on reste en accord avec les hypothèses citées en section 1.1 puisque le retrait de $\lambda\kappa$ empêche toute opportunité d'arbitrage et le "q" vaut bien 0 étant donné qu'il n'y a pas de versement de dividende.

Variables antithétiques

L'idée cette fois-ci est que pour chaque suite de bruits Z_k , on simule simultanément la trajectoire miroir avec $-Z_k$. On réalise par la suite la moyenne des deux payoffs. L'estimateur restant de fait non biaisé et égal à :

$$\hat{V}_0 = \frac{1}{2M} \sum_{m=1}^M e^{-rT} \left(P(S_T^{(m)}) + P(\tilde{S}_T^{(m)}) \right),$$

La contribution linéaire du bruit se compense ainsi entre les deux trajectoires, induisant une corrélation négative des payoffs et donc une variance plus faible. Le gain empirique observé est estimé entre 20 et 40%.

4 Résultats et validation

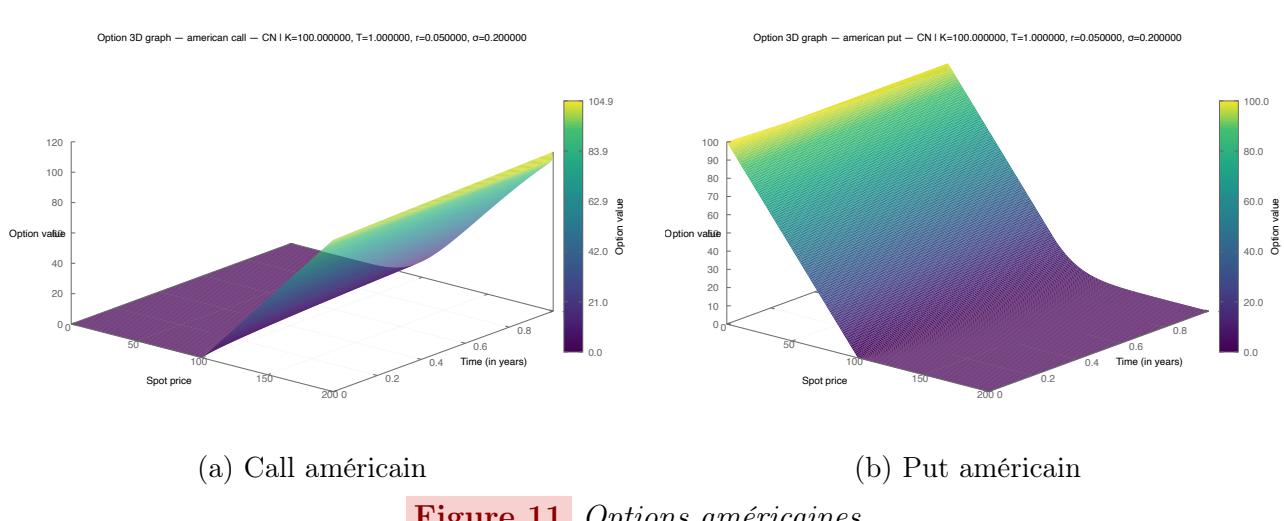
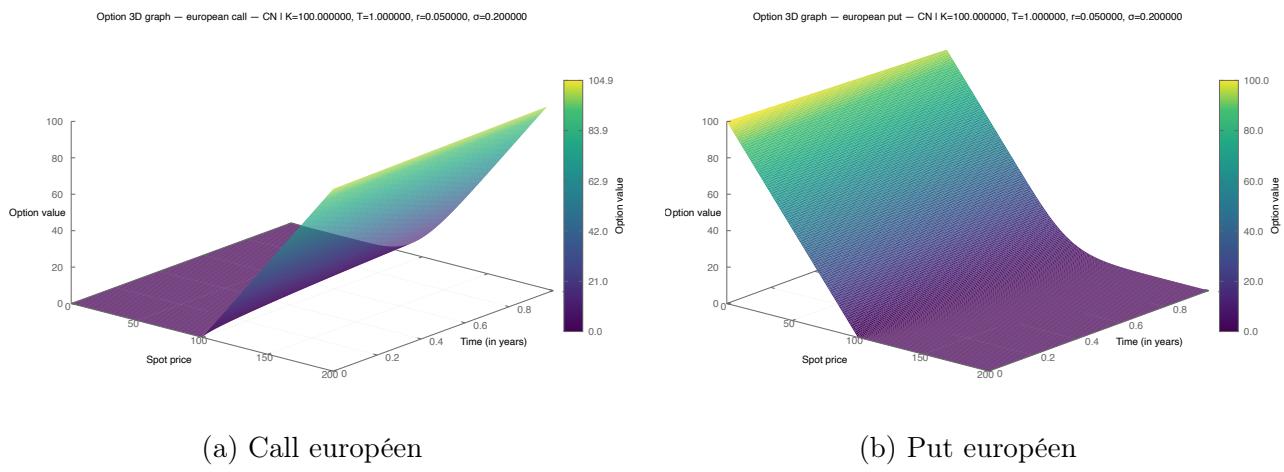
4.1 Présentation des résultats par type d'option

Cette première sous-partie vise à retrouver les propriétés caractéristiques des différentes options étudiées et de fait à fournir un premier niveau de vérification sur la pertinence des pricers. On notera que, sauf indication contraire, les figures et tables seront calculés avec les paramètres par défaut.

Paramètres par défaut:

- **Paramètres financiers :** $K = 100$, $S_{max} = 200$, $T = 1$ an, $r = 5\%$, $\sigma = 20$
- **Discrétisation de la grille :** $M = 200$, $N = 2000$

Commençons avec les options européennes et américaines ci-dessous, on note que les modélisations 3D respectent bien la structure de rampes ascendante/descendante propres aux call/put. Même si l'analyse visuelle des graphes est quelque peu imprécise, on observe que plus on est éloigné de la maturité ($T = 1$), plus la valeur enregistrée par les options américaines vire au jaune bien plus que leurs homologues européennes. Ce constat appuie bien la compensation supplémentaire justifiée par la possibilité d'exercer l'option américaine plus tôt et qui s'amenuise à mesure que le temps avance.



La commande permettant de tracer les différents graphes ci-dessus (en remplaçant call/put, european/american) peut être trouvée dans les annexes et il en est de même pour celle permettant de réaliser la table suivante.

| Type d'option | Call | Put |
|---------------|---------|---------|
| Européenne | 10.2917 | 5.6646 |
| Americaine | 10.2056 | 6.19259 |

Figure 12 Prix d'options à l'exercice ($S=K=100$, $T=1an$, $r=5\%$, $\sigma=20\%$)

Le choix du prix ITM permet d'annuler la valeur intrinsèque et de ce fait, le prix vient entièrement de la probabilité de finir ITM à l'expiration. On note que la valeur des calls américain et européen sont pratiquement équivalentes. Le fait que le call américain soit légèrement en-dessous provient d'un léger manque de précision de l'ordre de 0.8%. Ce résultat était attendu, étant donné que pour des actifs sans dividende, exercer l'option de façon anticipée fait perdre de la valeur temps. Néanmoins, la valeur du put américain excède son équivalente européenne avec une prime de 9.4%. Il est plus rentable d'exercer plus tôt et d'investir au taux sans risques r .

En ce qui concerne les options digitales, on retrouve la logique binaire propre à l'option. La modélisation nous donne une fonction sigmoïde qui, à mesure que l'on s'approche de la maturité ($T \rightarrow 0$), va afficher des pentes de plus en plus raides décrivant cette dynamique "cash-or-nothing" couplée à une incertitude au temps décroissante.

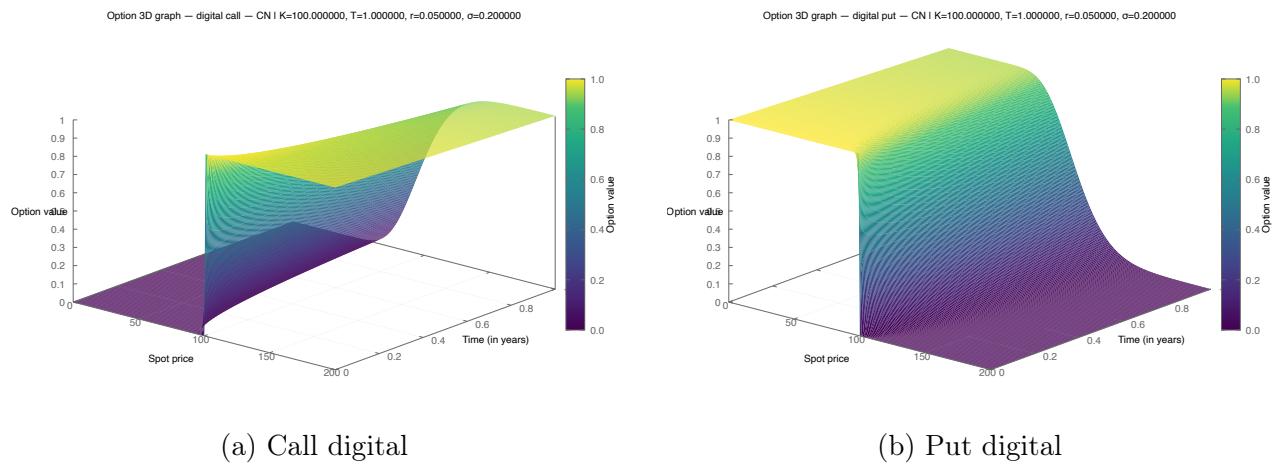


Figure 13 Options digitales (ou binaires)

Cela nous amène à présenter le dernier type d'options exotiques dont nous disposons de modélisation 3D à savoir celles de type barrière. Au-delà de la distinction classique call/put, ces dernières viennent décrire 4 scénarios sous la terminologie "Up/Down-and-In/Out". Up/down signifie simplement qu'on place la barrière au-dessus/en-dessous du cours actuel et In/out indique l'effet lorsque la barrière est atteinte à savoir activer/désactiver l'option. Ainsi la **somme d'une barrière In et d'une Out (du même type Up ou down) donne une option vanille classique**. Cette propriété implique non seulement que les options barrières sont moins chères que leurs équivalentes vanilles mais aussi qu'elles peuvent avoir des comportements similaires suivant la situation. Ici nous faisons le choix de montrer uniquement des scénarios où l'option accomplit cet effet "barrière" comme on peut le voir ci-dessous. Les niveaux de barrière ont été placé à 30% du prix d'exercice afin de mieux rendre compte de l'effet barrière. On note par ailleurs que comme pour les options précédentes, à mesure que l'on

s'approche de la maturité, la valeur du call barrière décroît.

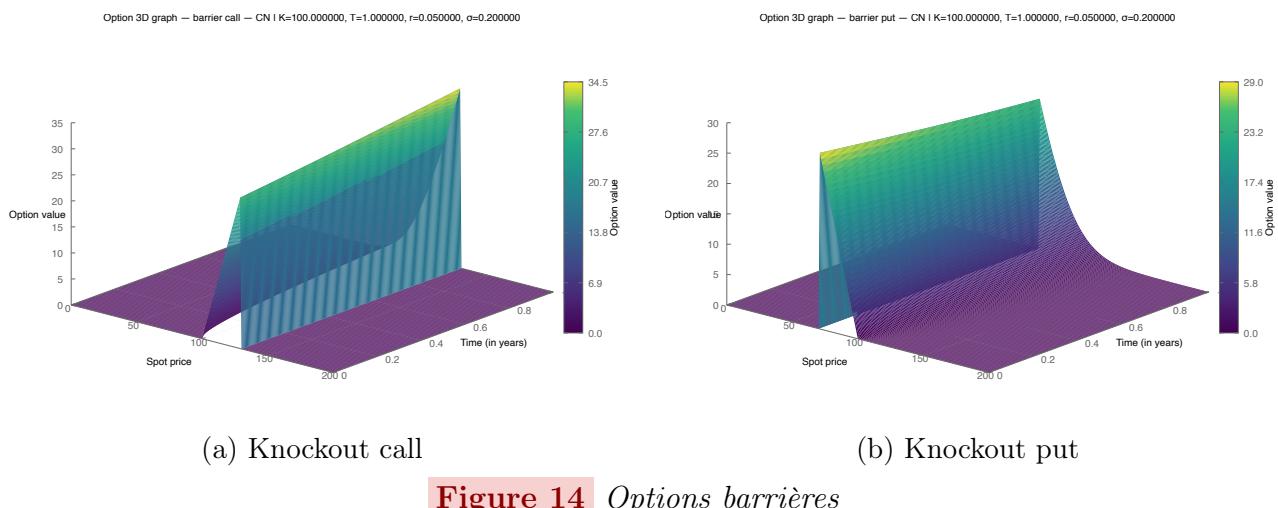


Figure 14 Options barrières

Pour les dernières options évaluées à l'aide de la méthode de Monte Carlo (à savoir celles asiatique, lookback, chooser), on peut noter dans la section Produits dérivés utilisés que les trois ont pour point commun une valuation fortement dépendante de la maturité. L'idée ici est donc de réaliser une table récapitulative des résultats obtenus pour chacune des options à des maturités différentes et en jauger les différentes évolutions. Les paramètres appliqués à la méthode de Monte Carlo y seront identiques et peuvent être retrouvés avec la commande correspondante dans la section Annexes.

| T (années) | Asiatique | Lookback | Chooser |
|------------|-----------|----------|---------|
| 0.25 | 2.61462 | 7.98644 | 6.87573 |
| 0.5 | 3.91438 | 11.4737 | 9.65704 |
| 1.0 | 5.7847 | 16.5227 | 13.8318 |
| 1.5 | 7.3714 | 20.58 | 17.2454 |
| 2.0 | 8.7571 | 24.025 | 19.9721 |
| 5.0 | 14.956 | 38.8805 | 32.8691 |
| 10.0 | 21.5975 | 55.4321 | 47.3132 |

| T (années) | Européen (référence) | Asiatique/Euro | Lookback/Euro | Chooser/Euro |
|------------|----------------------|----------------|---------------|--------------|
| 0.25 | 4.60906 | 0.57 | 1.73 | 1.49 |
| 0.5 | 6.90007 | 0.56 | 1.66 | 1.41 |
| 1.0 | 10.5269 | 0.55 | 1.58 | 1.31 |
| 1.5 | 13.3638 | 0.55 | 1.55 | 1.29 |
| 2.0 | 16.3102 | 0.53 | 1.48 | 1.23 |
| 5.0 | 29.2792 | 0.51 | 1.34 | 1.11 |
| 10.0 | 45.3229 | 0.48 | 1.23 | 1.04 |

Figure 15 Tables de comparaison des options exotiques (ici, des calls) suivant la maturité

Les résultats ci-dessus présentent deux niveaux d'information : le premier étant l'influence de la maturité sur ces dernières et la seconde leur rapport à une option vanille classique (et de ce fait l'évolution de ce rapport dans le temps). Concernant l'option asiatique, le fait de réaliser une moyenne arithmétique réduit la volatilité effective et donc amortissement implique une valeur plus faible (entre 43 et 52% ici). L'augmentation de la maturité T aura pour effet de lisser les valeurs extrêmes, ce qui explique une cote relative décroissante.

Les options lookback a contrario va ne conserver que la valeur extrême sur la période, ce qui explique ses valeurs plus élevées que son équivalente européenne. Si le ratio relatif décroît avec le temps, cela s'explique par le fait qu'à courte maturité, les extrêmes sont certes plus rares mais de ce fait plus "précieuses". Ceci se retranscrit avec une prime qui passe de 73% à 23%.

Enfin pour ce qu'il en est des options choisir, c'est la flexibilité de choisir la meilleure configuration (call ou put) qui explique le gain en valeur. Le temps intermédiaire de choix correspond ici à $T/2$. De ce fait, plus le temps entre la flexibilité et la maturité s'accroît augmente, plus l'incertitude face au temps est de plus en plus identique à celle rencontrée avec une option vanille classique. Cette dynamique se retrouve dans l'évolution de la prime qui baisse de 49% à 4%. Ainsi pour une distance conséquente entre le temps de décision et la maturité, le choisir se comporte de façon quasiment équivalente à une option européenne.

4.2 Comparaison des modèles et des performances

Cette sous-partie comparant les modèles au travers de leurs performances va se diviser en deux temps. Pour cette première partie, on cherche à comparer nos différents solveurs utilisant des EDP et utiliserons comme option de référence celle européenne. Cette dernière est la seule à pouvoir être évaluée par l'ensemble des modèles présentés dans ce projet, ce qui facilite le choix. Les comparaisons se feront à deux niveaux : sur les prix renvoyés (à l'exercice pour toujours annuler la valeur intrinsèque) et sur le temps nécessaire pour en réaliser le calcul. Les paramètres d'évaluation seront M (nombre de points spatiaux) et N (nombre de points temporels). En annexes seront mis les tableaux croisés entiers ainsi que la commande les ayant générés. On se contentera ici de réaliser notre analyse sur 4 scénarios de grille plus ou moins discrétisée.

| Solveur | Scénario 1 (100, 500) | Scénario 2 (200, 2000) | Scénario 3 (1000, 10000) | Scénario 4 (2000, 20000) | Précision |
|----------------|--------------------------|---------------------------|-----------------------------|-----------------------------|-----------------|
| Explicit | 9.82432 | 10.2922 | CFL | CFL | $O(dt, dS^2)$ |
| Implicit | 9.8202 | 10.2911 | 10.3868 | 10.4345 | $O(dt, dS^2)$ |
| Crank-Nicolson | 9.82226 | 10.2917 | 10.3869 | 10.4346 | $O(dt^2, dS^2)$ |

CFL : Condition CFL² non respectée.

Figure 16 Tables de comparaison des prix obtenus par solveur EDP (ici, avec call européen)

| Solveur | Scénario 1 (100, 500) | Scénario 2 (200, 2000) | Scénario 3 (1000, 10000) | Scénario 4 (2000, 20000) | Complexité |
|----------------|--------------------------|---------------------------|-----------------------------|-----------------------------|------------|
| Explicit | 0.15 | 2.12 | CFL | CFL | $O(MN)$ |
| Implicit | 0.46 | 4.53 | 128.90 | 602.39 | $O(MN)$ |
| Crank-Nicolson | 0.46 | 5.64 | 153.56 | 662.74 | $O(MN)$ |

CFL : Condition CFL non respectée.

Figure 17 Tables de comparaison des temps (en ms) obtenus par solveur EDP (ici, avec call européen)

Comme on peut le noter sur le graphe des prix, les trois solveurs dénote d'une bonne convergence en prix avec davantage de précision pour le schéma Crank-Nicolson du fait d'une dépendance temporelle d'ordre 2. De son côté, l'instabilité de modèle explicite se ressent suite à la nécessité de respecter la condition CFL. Cela étant dit, il réussit à donner des

²CFL $\lambda = \frac{\sigma^2 S_{\max}^2 \Delta t}{(\Delta S)^2} \leq 0.5 \Rightarrow N \geq 0.08M^2$

approximations correctes tant que l'équilibre entre M et N est respecté. En ce qui concerne la rapidité des modèles, on note que le modèle explicit délivre des résultats **2 à 3 fois plus vite** que les modèles Implicit/Crank-Nicolson. Néanmoins, cela se fait en contrepartie d'une perte de précision. De son côté, le Crank-Nicolson présente aussi une durée **10 à 20% plus importante** que le modèle implicite du fait de cette dépendance temporelle d'ordre deux.

Ces observations confirment le fait que le Crank-Nicolson reste le choix optimal quant au rapport précision/temps de calcul. Le modèle implicite s'affirme aussi être une alternative convenable tandis que le modèle explicite bien que rapide dispose de contraintes non négligeables (le manque de précision ainsi que le respect de la condition CFL).

Nous attaquons maintenant le deuxième niveau de comparaison avec la méthode de Monte Carlo. On se place un contexte similaire avec toujours un call européen sauf que cette fois-ci, la comparaison se fera sur le nombre de trajectoires et de pas de temps par trajectoires simulées. Les trois modèles étudiés ici seront :

- Modèle de Carlo standard (Modèle 1)
- Modèle de Carlo standard avec sauts de Merton (Modèle 2)
- Modèle de Carlo standard avec sauts de Merton et variables antithétiques (Modèle 3)

| (Trajectoires, pas) | Modèle 1 | | | Modèle 2 | | | Modèle 3 | | |
|---------------------|----------|--------|------------|----------|--------|------------|----------|--------|------------|
| | Prix | Stderr | Temps (ms) | Prix | Stderr | Temps (ms) | Prix | Stderr | Temps (ms) |
| (5K, 100) | 10.574 | 0.209 | 10.0 | 13.162 | 0.290 | 12.5 | 13.376 | 0.295 | 8.7 |
| (10K, 252) | 10.271 | 0.146 | 52.2 | 13.447 | 0.215 | 69.8 | 13.080 | 0.208 | 44.9 |
| (50K, 252) | 10.470 | 0.066 | 306.6 | 13.287 | 0.097 | 346.1 | 13.295 | 0.095 | 245.4 |
| (100K, 500) | 10.371 | 0.046 | 1263.4 | 13.275 | 0.068 | 1591.5 | 13.238 | 0.067 | 940.3 |

Figure 18 Tables de comparaison des temps (en ms), prix et erreurs standards obtenus par les modèles Monte Carlo (ici, avec call européen)

Avant de comparer les trois modèles, on notera que pour un niveau d'erreur similaire le modèle de Monte Carlo est **8 à 10 plus lent** que les solveurs EDP. Néanmoins, l'utilité de ce dernier est d'autant plus justifiée dans le cas d'options ne disposant pas d'EDP.

Pour revenir sur la comparaison, les trois modèles présentent bien des signes de convergence. En ce qui concerne les performances temporelles, l'ajout des sauts de Merton accroît le temps nécessaire de **13 à 34% supplémentaires** tandis que l'ajout des variables antithétiques accélèrent le temps de calcul car on traîte deux fois moins de trajectoires en prenant les opposés à chaque fois (**13 à 25% plus lent** que le modèle de base).

La différence de valeur d'environ 25% liée aux sauts de Merton sont représentatifs d'une réalité financière différente ce qui empêche la comparaison directe des prix. Néanmoins, on peut se pencher sur les erreurs standards propres à chaque modèle. Tous présentent un affaiblissement conséquent de l'erreur standard à mesure que les paramètres de simulation augmentent. L'ajout des sauts entraîne une erreur systématique expliquant cette augmentation de **quasiment 50% comparé au modèle de base**. Assez surprenamment l'ajout des variables antithétiques ne permet pas de réduire cette erreur comme espéré dans la partie théorique. La raison principale serait qu'on ne se place pas dans une situation qui le permette étant donné que l'on réalise la simulation dans la zone ATM qui manque de linéarité : le cadre théorique des variables antithétiques n'est pas respecté.

5 Commentaires

5.1 Problèmes rencontrés et solutions adoptées

Au cours du développement des différentes solveurs C++ pour la valorisation des options, de nombreuses difficultés techniques ont été rencontrées. L'un des premiers problème portait sur la gestion des indices dans la grille. En effet, les boucles pour parcourir les dimensions à la fois en espace mais aussi en temps devaient être soigneusement contrôlées pour éviter les dépassements de tableau et garantir que les valeurs aux frontières soient correctement initialisées. Pour résoudre ce problème, nous avons introduit des fonctions d'accès get et set dans la classe Grid, ce qui a permis de centraliser et sécuriser la manipulation des valeurs et de limiter les erreurs d'indices.

La gestion des héritages et des différents types d'options a elle-aussi posé quelques problèmes. Les classes dérivées **EuropeanOption** et **AmericanOption** devaient permettre de redéfinir la fonction payoff et le code devait rester suffisamment générique pour permettre l'utilisation de différents solveurs. Pour éviter des problèmes de compatibilité ou de slicing d'objets, nous nous sommes assurés d'utiliser des références constantes vers les options dans les solveurs et à déclarer les fonctions virtuelles correctement, ce qui nous a permis de nous assurer de la cohérence et la flexibilité de l'architecture orientée objet.

5.2 Limitations et pistes d'amélioration

Les schémas numériques pour la valorisation d'options via l'équation de Black–Scholes présentent plusieurs limites intrinsèques liées à la discrétisation. La précision des méthodes explicite, implicite ou Crank–Nicolson dépend fortement du nombre de points en espace et en temps. Une grille trop grossière peut introduire des erreurs numériques importantes, en particulier pour les options américaines, où l'approximation de la frontière libre d'exercice anticipé est cruciale. Le schéma explicite est conditionnellement stable, ce qui impose de choisir des pas de temps très petits pour éviter les oscillations numériques, augmentant ainsi le temps de calcul pour des options à maturité longue ou sous forte volatilité. Le schéma Crank–Nicolson, bien que plus précis, peut également produire des oscillations si la condition d'exercice anticipé n'est pas correctement traitée, et la localisation exacte de la frontière libre reste approximative.

La méthode de Monte Carlo, utilisée pour les options dépendant du chemin ou exotiques, souffre d'une convergence relativement lente, avec une erreur empirique décroissant comme $\frac{1}{\sqrt{N}}$. Pour atteindre une précision satisfaisante, un nombre très élevé de trajectoires est nécessaire, ce qui peut rendre le calcul coûteux en temps. De plus, la variance des estimations peut être importante d'une simulation à l'autre, ce qui rend indispensable l'utilisation de techniques de réduction de variance, telles que les antithetic variates ou les control variates. Les options asiatiques, lookback ou autres exotiques augmentent encore la complexité et le coût computationnel de cette approche.

De nombreuses pistes d'améliorations sont néanmoins envisageables. Un raffinement adaptatif de la grille spatiale permettrait de concentrer les points autour du strike ou de la frontière libre, améliorant ainsi la précision locale tout en limitant le coût global. Les méthodes Monte Carlo avancées, comme le quasi-Monte Carlo ou le stratified sampling, offrent une meilleure convergence, tandis que l'importance sampling permet d'estimer plus efficacement des événements à très faible probabilité, utiles pour les options fortement out-of-the-money. Sur le plan numérique, l'utilisation de schémas implicites avancés, comme la méthode ADI (Alternating Direction Implicit) pour des modèles multidimensionnels, ou des techniques de type Front-Fixing et penalty pour les options américaines, peut améliorer la précision et la stabilité. Enfin, l'optimisation du code via la parallélisation des boucles temporelles et des

simulations Monte Carlo, ainsi que l'exploitation de bibliothèques linéaires optimisées pour résoudre les systèmes tridiagonaux, permet de réduire considérablement le temps de calcul.

5.3 Conclusion

Ce projet de programmation en C++ nous a permis de concevoir un environnement complet pour la valorisation d'options en alliant algorithmes **mathématiques** avancés et une architecture logicielle orientée objet. L'objectif principal, qui consistait à confronter deux approches de pricing (résolution déterministe par équations aux dérivées partielles et simulation stochastique de Monte Carlo), a été atteint pour une large gamme d'instruments financiers.

Nos travaux sur l'équation de Black-Scholes ont mis en évidence l'importance du choix des schémas numériques. Nous avons pu constater que l'implémentation du schéma de Crank-Nicolson s'est avérée être la plus performante, offrant une précision d'ordre 2 tout en garantissant une stabilité inconditionnelle, contrairement au schéma explicite. De plus, le traitement des options américaines via un algorithme de frontière libre a permis d'illustrer la complexité liée à l'exercice anticipé, se traduisant notamment par une prime de valeur nette par rapport aux options américaines, en particulier pour les puts.

La méthode de Monte Carlo, bien que présentant une convergence plus lente en $O(1/\sqrt{M})$, a démontré une flexibilité fondamentale pour le pricing d'options exotiques dépendant du chemin, telles que les options asiatiques ou lookback. L'intégration de techniques plus avancées, notamment le générateur de nombres aléatoires PCG32 et les sauts de Merton, a permis d'optimiser la fiabilité des estimations et réduire les erreurs standards respectivement à leur contexte financier.

Enfin, l'architecture en C++ reposant sur le polymorphisme et l'utilisation de pointeurs uniques a offert une séparation claire entre les modèles de marché, les types d'options et les solveurs. Cette structure garantit la modularité du programme et facilite l'intégration future de modèles plus complexes, comme ceux à volatilité locale ou le support du multithreading. Ce travail constitue ainsi une base solide pour appréhender les enjeux du trading de haute fréquence et de la gestion des risques, où l'efficience algorithmique est primordiale.

Annexes

I - Commandes générant les résultats

Pour les options européennes et américaines des figures 10 et 11 de la section 4.1.

```
cd "/Users/ralphnader/Desktop/Option-Pricing---C- copy/Code" && ./option_viz --solver cn --option american --type put --strike 100 --T 1 --r 0.05 --sigma 0.2 --M 200 --N 2000 --Smax 200 --plot on --plotFormat pdf --plotOut put_am.pdf
```

Il en est de même pour la commande de la table 12 de la section 4.1.

```
cd "/Users/ralphnader/Desktop/Option-Pricing---C- copy/Code" && echo "European call" && ./option_viz --option european --type call --strike 100 --T 1 --r 0.05 --sigma 0.2 --M 200 --N 2000 --plot off 2>&1 | grep "Price at strike" && echo "European put" && ./option_viz --option european --type put --strike 100 --T 1 --r 0.05 --sigma 0.2 --M 200 --N 2000 --plot off 2>&1 | grep "Price at strike" && echo "American call" && ./option_viz --option american --type call --strike 100 --T 1 --r 0.05 --sigma 0.2 --M 200 --N 2000 --plot off 2>&1 | grep "Price at strike" && echo "American put" && ./option_viz --option american --type put --strike 100 --T 1 --r 0.05 --sigma 0.2 --M 200 --N 2000 --plot off 2>&1 | grep "Price at strike"
```

Voici la commande pour les options digitales (figure 13) de la section 4.1. Il suffit de changer entre call et put.

```
cd "/Users/ralphnader/Desktop/Option-Pricing---C- copy/Code" && ./option_viz --solver cn --option digital --type put --strike 100 --T 1 --r 0.05 --sigma 0.2 --M 200 --N 2000 --Smax 200 --plot on --plotFormat pdf --plotOut put_digital.pdf
```

En ce qui concerne les deux commandes permettant d'obtenir les figures barrières présentées sur la figure 14.

```
cd "/Users/ralphnader/Desktop/Option-Pricing---C- copy/Code" && ./option_viz --solver cn --option barrier --type put --strike 100 --barrierType downout --barrierLevel 70 --T 1 --r 0.05 --sigma 0.2 --M 200 --N 2000 --Smax 200 --plot on --plotFormat pdf --plotOut put_downout.pdf && echo "Price at strike (S=100):" && ./option_viz --solver cn --option barrier --type put --strike 100 --barrierType downout --barrierLevel 70 --T 1 --r 0.05 --sigma 0.2 --M 200 --N 2000 --Smax 200 --plot off 2>&1 | grep "Price at strike"

cd "/Users/ralphnader/Desktop/Option-Pricing---C- copy/Code" && ./option_viz --solver cn --option barrier --type call --strike 100 --barrierType upout --barrierLevel 130 --T 1 --r 0.05 --sigma 0.2 --M 200 --N 2000 --Smax 200 --plot on --plotFormat pdf --plotOut call_knockout.pdf && echo "Price at strike (S=100):" && ./option_viz --solver cn --option barrier --type call --strike 100 --barrierType upout --barrierLevel 130 --T 1 --r 0.05 --sigma 0.2 --M 200 --N 2000 --Smax 200 --plot off 2>&1 | grep "Price at strike"
```

Voici la table donnant les résultats comparatifs entre les différents types d'options utilisant Monte carlo que l'on retrouve sur les tables figure 15.

```

cd "/Users/ralphnader/Desktop/Option-Pricing---C- copy/Code" && cat <<
'SCRIPT' | bash
BASE="--solver mc --spot 100 --strike 100 --r 0.05 --sigma 0.2 --
mcPaths 30000 --mcSteps 252 --type call"

printf "\n%-8s | %-12s | %-12s | %-12s\n" "T (ann es)" "Asian" "
Lookback" "Chooser"
printf "%s\n" "-----|-----|-----|-----"
for T in 0.25 0.5 1.0 1.5 2.0 5.0 10.0; do
    asian=$(./option_viz $BASE --option asian --T $T 2>&1 | awk '/MC
Price:/ {print $3}')
    lookback=$(./option_viz $BASE --option lookback --lookbackType min --
T $T 2>&1 | awk '/MC Price:/ {print $3}')
    chooser=$(./option_viz $BASE --option chooser --chooseTime 0.5 --T $T
2>&1 | awk '/MC Price:/ {print $3}')

    printf "%-8s | %-12s | %-12s | %-12s\n" "$T" "$asian" "$lookback" "
$chooser"
done

printf "\n%-8s | %-12s | %-12s | %-12s | %-12s\n" "T (ann es)" "Euro (
r ference)" "Asian/Euro" "Look/Euro" "Chooser/Euro"
printf "%s\n" "-----|-----|-----|-----|-----"
for T in 0.25 0.5 1.0 1.5 2.0 5.0 10.0; do
    euro=$(./option_viz $BASE --option european --T $T 2>&1 | awk '/MC
Price:/ {print $3}')
    asian=$(./option_viz $BASE --option asian --T $T 2>&1 | awk '/MC
Price:/ {print $3}')
    lookback=$(./option_viz $BASE --option lookback --lookbackType min --
T $T 2>&1 | awk '/MC Price:/ {print $3}')
    chooser=$(./option_viz $BASE --option chooser --chooseTime 0.5 --T $T
2>&1 | awk '/MC Price:/ {print $3}')

    ratio_asian=$(awk "BEGIN {printf \"% .2f\", $asian/$euro}")
    ratio_look=$(awk "BEGIN {printf \"% .2f\", $lookback/$euro}")
    ratio_choose=$(awk "BEGIN {printf \"% .2f\", $chooser/$euro}")

    printf "%-8s | %-12s | %-12s | %-12s | %-12s\n" "$T" "$euro" "
$ratio_asian" "$ratio_look" "$ratio_choose"
done
SCRIPT

```

Ci-dessous se trouve la commande permettant de générer les tableaux croisés montrant les performances des solveurs EDP (présents dans la section des annexes).

```

cd "/Users/ralphnader/Desktop/Option-Pricing---C- copy/Code" && for
SOLVER in explicit implicit cn; do echo ""; echo "$SOLVER"; for M in
100 200 500 1000 2000; do for N in 500 1000 2000 10000 20000; do
out=$(./option_viz --solver $SOLVER --option european --type call --
strike 100 --T 1 --r 0.05 --sigma 0.2 --M $M --N $N --plot off 2>&1)

```

```
; p=$(echo "$out" | awk '/Price at strike:/ {print $4}'); t=$(echo "$out" | awk '/^Time:/ {print $2}'); e=$(echo "$out" | grep "Erreur CFL"); if [ -z "$e" ]; then echo "M=$M N=$N | Prix=$p | Temps=$t ms"; else echo "M=$M N=$N | CFL | CFL"; fi; done; done
```

Voici la commande pour la comparaison de méthodes à base de Monte Carlo (il faut la relancer en changeant les valeurs de mcSteps et mcPaths à chaque fois).

```
cd "/Users/ralphnader/Desktop/Option-Pricing---C- copy/Code" && echo "(100K, 500)" && echo "Base:" && ./option_viz --solver mc --option european --type call --strike 100 --T 1 --r 0.05 --sigma 0.2 --plot off --mcPaths 100000 --mcSteps 500 --mcJumpIntensity 0 --mcAntithetic off 2>&1 && echo "Merton:" && ./option_viz --solver mc --option european --type call --strike 100 --T 1 --r 0.05 --sigma 0.2 --plot off --mcPaths 100000 --mcSteps 500 --mcJumpIntensity 0.5 --mcJumpMean -0.1 --mcJumpVol 0.3 --mcAntithetic off 2>&1 && echo "Merton+Anti:" && ./option_viz --solver mc --option european --type call --strike 100 --T 1 --r 0.05 --sigma 0.2 --plot off --mcPaths 100000 --mcSteps 500 --mcJumpIntensity 0.5 --mcJumpMean -0.1 --mcJumpVol 0.3 --mcAntithetic on 2>&1
```

II - Autres figures

| M \ N | 500 | 1000 | 2000 | 10000 | 20000 |
|-------|---------|---------|---------|---------|---------|
| 100 | 9.82432 | 9.82329 | 9.82278 | 9.82236 | 9.82231 |
| 200 | CFL | CFL | 10.2922 | 10.2918 | 10.2917 |
| 500 | CFL | CFL | CFL | CFL | 10.3869 |
| 1000 | CFL | CFL | CFL | CFL | CFL |
| 2000 | CFL | CFL | CFL | CFL | CFL |

| M \ N | 500 | 1000 | 2000 | 10000 | 20000 |
|-------|---------|---------|---------|---------|---------|
| 100 | 9.8202 | 9.82123 | 9.82175 | 9.82216 | 9.82221 |
| 200 | 10.2896 | 10.2906 | 10.2911 | 10.2915 | 10.2916 |
| 500 | 10.3848 | 10.3858 | 10.3863 | 10.3868 | 10.3868 |
| 1000 | 10.3848 | 10.3858 | 10.3864 | 10.3868 | 10.3868 |
| 2000 | 10.4325 | 10.4335 | 10.434 | 10.4345 | 10.4345 |

| M \ N | 500 | 1000 | 2000 | 10000 | 20000 |
|-------|---------|---------|---------|---------|---------|
| 100 | 9.82226 | 9.82226 | 9.82226 | 9.82226 | 9.82226 |
| 200 | 10.2917 | 10.2917 | 10.2917 | 10.2917 | 10.2917 |
| 500 | 10.3869 | 10.3869 | 10.3869 | 10.3869 | 10.3869 |
| 1000 | 10.3869 | 10.3869 | 10.3869 | 10.3869 | 10.3869 |
| 2000 | 10.4346 | 10.4346 | 10.4346 | 10.4346 | 10.4346 |

Notation : CFL : le couple (M, N) ne respecte pas les conditions de stabilité du schéma explicite

Figure Tables de comparaison des solveurs EDP (ici, avec call européen) pour les prix

| M \ N | 500 | 1000 | 2000 | 10000 | 20000 |
|-------|------|------|------|-------|-------|
| 100 | 0.15 | 0.39 | 1.00 | 5.15 | 11.28 |
| 200 | CFL | CFL | 2.12 | 11.22 | 22.77 |
| 500 | CFL | CFL | CFL | CFL | 53.64 |
| 1000 | CFL | CFL | CFL | CFL | CFL |
| 2000 | CFL | CFL | CFL | CFL | CFL |

| M \ N | 500 | 1000 | 2000 | 10000 | 20000 |
|-------|------|-------|-------|--------|--------|
| 100 | 0.46 | 1.04 | 2.13 | 11.46 | 22.81 |
| 200 | 0.94 | 2.04 | 4.53 | 22.54 | 44.83 |
| 500 | 2.23 | 5.00 | 11.31 | 56.53 | 114.45 |
| 1000 | 4.56 | 10.33 | 23.23 | 128.90 | 254.80 |
| 2000 | 9.47 | 20.64 | 53.16 | 312.78 | 602.39 |

| M \ N | 500 | 1000 | 2000 | 10000 | 20000 |
|-------|-------|-------|-------|--------|--------|
| 100 | 0.46 | 1.04 | 2.37 | 12.64 | 24.89 |
| 200 | 0.96 | 2.46 | 5.64 | 27.24 | 53.29 |
| 500 | 2.53 | 5.60 | 13.59 | 64.87 | 169.15 |
| 1000 | 11.04 | 17.16 | 29.67 | 153.56 | 315.79 |
| 2000 | 10.62 | 23.51 | 59.40 | 333.90 | 662.74 |

Notation : CFL : le couple (M, N) ne respecte pas les conditions de stabilité du schéma explicite

Figure Tables de comparaison des solveurs EDP (ici, avec call européen) pour le temps