Rapport de résolution numérique de l'équation de Black–Scholes

Par différences finies

Module : Programmation avancée et projet
ENSIIE

Étudiants : Mouhamed CISSE, Mohammed-Yahya DAMI

Année universitaire : 2025–2026

3 janvier 2026

Résumé

Ce rapport présente une résolution numérique de l'équation de Black–Scholes pour la valorisation d'options européennes, en adoptant les bonnes pratiques de la programmation avancée et orientée objet. Deux schémas de différences finies sont étudiés et implémentés en C++ : le schéma implicite et la méthode de Crank–Nicholson. Une étude théorique est menée, incluant la transformation de l'équation de Black–Scholes en une équation de diffusion. Une discrétisation spatio-temporelle est ensuite introduite. Les systèmes linéaires tridiagonaux obtenus sont résolus à l'aide de l'algorithme de Thomas. Les résultats numériques sont visualisés à l'aide de la bibliothèque SDL pour les options Call et Put, ainsi que pour l'erreur entre les deux méthodes.

Table des matières

1 Introduction et contexte **4**

2 Étude théorique **4**
2.1 Équation de Black–Scholes 4
2.2 Conditions terminales et aux limites 4
2.3 Équation réduite 5
2.4 Discrétisation du domaine 6
2.5 Approximation des dérivées 6
2.6 Méthode de Crank–Nicholson 7
2.7 Schéma implicite 8
2.8 Résolution des systèmes linéaires 9

3 Implémentation en C++ **11**
3.1 Architecture logicielle 11
3.1.1 Classes pour les options 11
3.1.2 Classes pour les EDP 11
3.1.3 Classes pour les méthodes des différences finies 11
3.1.4 Classe pour l'affichage 12

4 Choix techniques **13**
4.1 Architecture orientée objet 13
4.2 Options financières 13
4.3 Actif sous-jacent 13
4.4 Équations aux dérivées partielles 13
4.5 Méthodes numériques 13
4.6 Gestion des conditions aux limites du Call 13
4.7 Affichage des résultats 14

5 Difficultés rencontrées et solutions apportées **15**
5.1 Modélisation des options 15
5.2 Schémas numériques et discrétisation 15
5.3 Affichage SDL et gestion des fenêtres 15

6 Affichage et interprétation des résultats **16**
6.1 Option Call 16
6.2 Erreur entre les deux méthodes pour une option Call 17
6.3 Option Put 18
6.4 Erreur entre les deux méthodes pour une option Put 18

7 Limites du modèle et de l'approche numérique **19**

8 Perspectives et améliorations possibles **20**
8.1 Amélioration numérique 20
8.2 Calcul des Grecs 20
8.3 Extension du cadre 20

9 Conclusion **20**

10 Annexes **21**
10.1 Diagramme UML des classes 21

1 Introduction et contexte

Le modèle de Black–Scholes, développé par Black, Scholes et Merton en 1973, est une référence pour la valorisation des options européennes. Il repose sur des hypothèses classiques telles que l'efficience des marchés financiers, la constance de la volatilité du sous-jacent et l'existence d'un taux d'intérêt sans risque connus à l'avance. Sous ces hypothèses, le prix d'une option vérifie une équation aux dérivées partielles.

La résolution analytique de cette équation n'est possible que dans des cas particuliers, ce qui justifie le recours à des méthodes numériques. Dans ce projet, l'équation de Black–Scholes est résolue numériquement à l'aide de méthodes de différences finies, notamment les schémas implicites et de Crank–Nicholson, puis implémentée en langage C++ dans une architecture logicielle structurée.

2 Étude théorique

2.1 Équation de Black–Scholes

On note $C(t, S)$ le prix d'une option européenne en fonction du temps $t \in [0, T]$ et du prix du sous-jacent $S \in [0, L]$. La fonction C vérifie l'équation aux dérivées partielles suivante :

$$\frac{\partial C}{\partial t} + rS\frac{\partial C}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} = rC \tag{1}$$

où :
— T est la maturité de l'option,
— L est une borne supérieure du prix du sous-jacent,
— r est le taux d'intérêt sans risque,
— σ est la volatilité du sous-jacent.

2.2 Conditions terminales et aux limites

La condition terminale correspond au payoff à maturité $t = T$. On considère deux types d'options européennes de strike $K > 0$.

Option Put européenne

$$C(T, S) = \max(K - S, 0), \quad S \in [0, L]$$

$$C(t, 0) = Ke^{-r(T-t)}, \quad t \in [0, T]$$

$$C(t, L) = 0, \quad t \in [0, T]$$

Option Call européenne

$$C(T, S) = \max(S - K, 0), \quad S \in [0, L]$$

$$C(t, 0) = 0, \quad t \in [0, T]$$

$$C(t, L) = L - Ke^{-r(T-t)}, \quad t \in [0, T]$$

2.3 Équation réduite

L'équation de Black–Scholes peut être transformée par un changement de variables approprié en une équation de diffusion classique. Cette transformation permet de simplifier l'étude numérique tout en conservant les propriétés essentielles du modèle.

On part de l'équation de Black–Scholes écrite sous la forme :

$$\frac{\partial C}{\partial t} + rS\frac{\partial C}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} - rC = 0. \tag{2}$$

Changement de variable spatial

On introduit la variable logarithmique :

$$x = \ln\left(\frac{S}{K}\right),$$

où K est le strike de l'option. On définit ensuite une nouvelle fonction $u(t, x)$ telle que :

$$C(t, S) = K\, u(t, x).$$

Les dérivées par rapport à S s'expriment alors en fonction de x :

$$\frac{\partial C}{\partial S} = \frac{1}{S} K \frac{\partial u}{\partial x},$$

$$\frac{\partial^2 C}{\partial S^2} = \frac{K}{S^2}\left(\frac{\partial^2 u}{\partial x^2} - \frac{\partial u}{\partial x}\right).$$

Changement de variable temporel

On effectue un renversement du temps afin de transformer la condition terminale en condition initiale :

$$\tau = T - t.$$

La dérivée temporelle devient alors :

$$\frac{\partial C}{\partial t} = -K \frac{\partial u}{\partial \tau}.$$

Équation transformée

En injectant ces expressions dans l'équation de Black–Scholes, on obtient :

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial x^2} + \left(r - \frac{1}{2}\sigma^2\right)\frac{\partial u}{\partial x} - ru. \tag{3}$$

Élimination du terme de dérivée première

On cherche à éliminer le terme en dérivée première en introduisant le changement de fonction :

$$u(\tau, x) = e^{\alpha x + \beta \tau} v(\tau, x),$$

où les constantes α et β sont choisies de manière appropriée.

En imposant l'annulation du terme en $\partial v/\partial x$, on obtient :

$$\alpha = -\frac{1}{2}\left(\frac{2r}{\sigma^2} - 1\right).$$

On définit alors le paramètre :

$$\mu = \frac{2r}{\sigma^2} - 1.$$

Équation réduite de diffusion

Avec ces choix, la fonction $v(\tau, x)$ vérifie l'équation de diffusion classique :

$$\frac{\partial v}{\partial \tau} = \frac{\sigma^2}{2}\frac{\partial^2 v}{\partial x^2}. \tag{4}$$

Cette équation est une équation de la chaleur, bien connue en analyse numérique. Elle constitue l'équation réduite de Black–Scholes et sert de base à la mise en œuvre de schémas numériques simples et efficaces.

2.4 Discrétisation du domaine

Pour réaliser la résolution numérique de l'équation aux dérivées partielles de Black–Scholes, on procède tout d'abord à une discrétisation du domaine temporel et du domaine spatial.

L'intervalle temporel $[0, T]$ est découpé en M sous-intervalles $[t_i, t_{i+1}]$ de pas constant

$$\Delta t = t_{i+1} - t_i.$$

De même, l'intervalle spatial $[0, L]$ correspondant au prix du sous-jacent est découpé en N sous-intervalles $[S_j, S_{j+1}]$ de pas

$$\Delta S = S_{j+1} - S_j.$$

Les points de discrétisation sont définis par :

$$t_n = n\Delta t, \quad n = 0, \ldots, M,$$

$$S_j = j\Delta S, \quad j = 0, \ldots, N.$$

La solution discrète est alors notée :

$$C_j^n \approx C(t_n, S_j),$$

où C_j^n représente une approximation numérique du prix de l'option au temps t_n et au prix du sous-jacent S_j.

2.5 Approximation des dérivées

Les dérivées spatiales sont approchées par des différences finies centrées, tandis que la dérivée temporelle est approchée par une différence arrière ou centrée selon le schéma utilisé.

2.6 Méthode de Crank–Nicholson

La méthode de Crank–Nicholson est un schéma implicite d'ordre deux en temps, obtenu par moyennage entre un schéma explicite et un schéma implicite. Elle présente un bon compromis entre précision et stabilité numérique.

On considère l'équation de Black–Scholes sous la forme :

$$\frac{\partial C}{\partial t} + rS\frac{\partial C}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} - rC = 0. \tag{5}$$

Discrétisation temporelle

La dérivée temporelle est approchée par une différence centrée :

$$\frac{\partial C}{\partial t}(t_n, S_j) \approx \frac{C_j^{n+1} - C_j^n}{\Delta t}.$$

Discrétisation spatiale

Les dérivées spatiales sont approchées par des différences finies centrées :

$$\frac{\partial C}{\partial S}(t_n, S_j) \approx \frac{C_{j+1}^n - C_{j-1}^n}{2\Delta S},$$

$$\frac{\partial^2 C}{\partial S^2}(t_n, S_j) \approx \frac{C_{j+1}^n - 2C_j^n + C_{j-1}^n}{(\Delta S)^2}.$$

Dans la méthode de Crank–Nicholson, les termes spatiaux sont évalués à la moyenne des instants t_n et t_{n+1}.

Formulation du schéma

On obtient alors, pour tout $j = 1, \ldots, N-1$:

$$\frac{C_j^{n+1} - C_j^n}{\Delta t} + \frac{1}{2}\Big[\mathcal{L}C_j^{n+1} + \mathcal{L}C_j^n\Big] = 0, \tag{6}$$

où l'opérateur spatial $\mathcal{L}$ est défini par :

$$\mathcal{L}C_j = rS_j\frac{C_{j+1} - C_{j-1}}{2\Delta S} + \frac{1}{2}\sigma^2 S_j^2 \frac{C_{j+1} - 2C_j + C_{j-1}}{(\Delta S)^2} - rC_j.$$

Mise sous forme algébrique

Après réorganisation des termes, on obtient un système linéaire de la forme :

$$a_j C_{j-1}^{n+1} + b_j C_j^{n+1} + c_j C_{j+1}^{n+1} = d_j,$$

où les coefficients sont donnés par :

$$a_j = -\frac{\Delta t}{4}\left(\frac{\sigma^2 S_j^2}{(\Delta S)^2} - \frac{rS_j}{\Delta S}\right),$$

$$b_j = 1 + \frac{\Delta t}{2}\left(\frac{\sigma^2 S_j^2}{(\Delta S)^2} + r\right),$$

$$c_j = -\frac{\Delta t}{4}\left(\frac{\sigma^2 S_j^2}{(\Delta S)^2} + \frac{r S_j}{\Delta S}\right),$$

et le second membre :

$$d_j = \frac{\Delta t}{4}\left(\frac{\sigma^2 S_j^2}{(\Delta S)^2} - \frac{r S_j}{\Delta S}\right) C_{j-1}^n \tag{7}$$

$$+ \left(1 - \frac{\Delta t}{2}\left(\frac{\sigma^2 S_j^2}{(\Delta S)^2} + r\right)\right) C_j^n \tag{8}$$

$$+ \frac{\Delta t}{4}\left(\frac{\sigma^2 S_j^2}{(\Delta S)^2} + \frac{r S_j}{\Delta S}\right) C_{j+1}^n. \tag{9}$$

Système tridiagonal

À chaque pas de temps, le schéma de Crank–Nicholson conduit donc à la résolution d'un système linéaire tridiagonal :

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & a_{N-2} & b_{N-2} & c_{N-2} \\ 0 & \cdots & 0 & a_{N-1} & b_{N-1} \end{pmatrix} \begin{pmatrix} C_1^{n+1} \\ C_2^{n+1} \\ \vdots \\ C_{N-2}^{n+1} \\ C_{N-1}^{n+1} \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N-2} \\ d_{N-1} \end{pmatrix}.$$

2.7 Schéma implicite

Le schéma implicite est utilisé pour sa robustesse et sa stabilité numérique, notamment lorsque les pas de discrétisation sont relativement grands. Il consiste à évaluer l'ensemble des termes spatiaux au temps futur t_{n+1}.

On considère l'équation de Black–Scholes sous la forme :

$$\frac{\partial C}{\partial t} + rS\frac{\partial C}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} - rC = 0. \tag{10}$$

Discrétisation temporelle

La dérivée temporelle est approchée par une différence arrière :

$$\frac{\partial C}{\partial t}(t_{n+1}, S_j) \approx \frac{C_j^{n+1} - C_j^n}{\Delta t}.$$

Discrétisation spatiale

Les dérivées spatiales sont approchées par des différences finies centrées, évaluées à l'instant t_{n+1} :

$$\frac{\partial C}{\partial S}(t_{n+1}, S_j) \approx \frac{C_{j+1}^{n+1} - C_{j-1}^{n+1}}{2\Delta S},$$

$$\frac{\partial^2 C}{\partial S^2}(t_{n+1}, S_j) \approx \frac{C_{j+1}^{n+1} - 2C_j^{n+1} + C_{j-1}^{n+1}}{(\Delta S)^2}.$$

Formulation du schéma implicite

En remplaçant les dérivées par leurs approximations, on obtient pour $j = 1, \ldots, N-1$:

$$\frac{C_j^{n+1} - C_j^n}{\Delta t} + rS_j \frac{C_{j+1}^{n+1} - C_{j-1}^{n+1}}{2\Delta S} \tag{11}$$

$$+ \frac{1}{2}\sigma^2 S_j^2 \frac{C_{j+1}^{n+1} - 2C_j^{n+1} + C_{j-1}^{n+1}}{(\Delta S)^2} - rC_j^{n+1} = 0. \tag{12}$$

Mise sous forme algébrique

En regroupant les termes en C^{n+1}, on obtient un système linéaire de la forme :

$$a_j C_{j-1}^{n+1} + b_j C_j^{n+1} + c_j C_{j+1}^{n+1} = C_j^n,$$

avec les coefficients :

$$a_j = -\Delta t \left(\frac{1}{2} \frac{\sigma^2 S_j^2}{(\Delta S)^2} - \frac{rS_j}{2\Delta S} \right),$$

$$b_j = 1 + \Delta t \left(\frac{\sigma^2 S_j^2}{(\Delta S)^2} + r \right),$$

$$c_j = -\Delta t \left(\frac{1}{2} \frac{\sigma^2 S_j^2}{(\Delta S)^2} + \frac{rS_j}{2\Delta S} \right).$$

Système matriciel tridiagonal

À chaque pas de temps, le schéma implicite conduit à la résolution du système tridiagonal suivant :

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & a_{N-2} & b_{N-2} & c_{N-2} \\ 0 & \cdots & 0 & a_{N-1} & b_{N-1} \end{pmatrix} \begin{pmatrix} C_1^{n+1} \\ C_2^{n+1} \\ \vdots \\ C_{N-2}^{n+1} \\ C_{N-1}^{n+1} \end{pmatrix} = \begin{pmatrix} C_1^n \\ C_2^n \\ \vdots \\ C_{N-2}^n \\ C_{N-1}^n \end{pmatrix}.$$

2.8 Résolution des systèmes linéaires

Les schémas implicite et Crank–Nicholson conduisent à la résolution à chaque pas de temps d'un système linéaire tridiagonal de la forme :

$$a_j C_{j-1}^{n+1} + b_j C_j^{n+1} + c_j C_{j+1}^{n+1} = d_j, \quad j = 1, \ldots, N-1.$$

Choix de la méthode

Plutôt que d'utiliser une factorisation LU classique, dont la complexité est en $O(N^3)$ pour un système de taille N, on utilise l'algorithme de Thomas. Cet algorithme est spécifiquement adapté aux matrices tridiagonales et résout le système en complexité linéaire $O(N)$.

Algorithme de Thomas

L'algorithme de Thomas consiste en deux étapes principales :

1. **Elimination en avant** : on modifie les coefficients pour annuler les éléments inférieurs a_j de la matrice, transformant la matrice originale en une matrice triangulaire supérieure.

$$\tilde{b}_j = b_j - a_j \frac{c_{j-1}}{\tilde{b}_{j-1}}, \quad \tilde{d}_j = d_j - a_j \frac{\tilde{d}_{j-1}}{\tilde{b}_{j-1}}, \quad j = 2, \ldots, N-1$$

2. **Substitution en arrière** : on calcule la solution C_j^{n+1} à partir de la dernière équation et en remontant :

$$C_{N-1}^{n+1} = \frac{\tilde{d}_{N-1}}{\tilde{b}_{N-1}}, \quad C_j^{n+1} = \frac{\tilde{d}_j - c_j C_{j+1}^{n+1}}{\tilde{b}_j}, \quad j = N-2, \ldots, 1.$$

Avantages

— Complexité linéaire $O(N)$, beaucoup plus rapide que LU classique pour les grands systèmes.
— Méthode stable et adaptée aux matrices tridiagonales issues des schémas numériques de Black–Scholes.
— Facile à implémenter en C++ et mémoire optimisée.

3 Implémentation en C++

3.1 Architecture logicielle

L'implémentation repose sur une architecture orientée objet structurée autour de classes représentant les options, les EDP et les méthodes numériques.

3.1.1 Classes pour les options

Classe `Option` : classe abstraite servant de base pour les classes `Put` et `Call`.
Attributs :
— `K_` : prix d'exercice (Strike)
— `T_` : date d'échéance
Méthodes :
— Getters : `getK()`, `getT()`
— Méthodes virtuelles pures : `payoff(double S)`, `lowerBoundary(double t, double r)`, `upperBoundary(double S_max, double t, double r)`
Classes `Call` **et** `Put` : héritent de `Option` et implémentent les méthodes virtuelles pures.

3.1.2 Classes pour les EDP

Classe `EDP` : classe abstraite représentant une équation aux dérivées partielles.
Attributs :
— `option_` : référence vers l'option associée
— `actif_` : référence vers l'actif sous-jacent
Méthodes :
— Constructeur : initialise `option_` et `actif_`
— Getters : `getOption()`, `getActif()`
Classes `EDPComplete` **et** `EDPReduite` : héritent de `EDP` et représentent respectivement l'équation complète et l'équation réduite de Black–Scholes.

3.1.3 Classes pour les méthodes des différences finies

Classe `DifferenceFinie` : classe abstraite représentant une méthode des différences finies pour résoudre une EDP.
Attributs :
— `edp_` : référence vers l'EDP à résoudre
— `N_`, `M_` : nombre de pas en espace et en temps
— `dt_`, `dS_` : pas de temps et d'espace
— `L_`, `t_` : grilles des prix et des temps
Méthodes :
— Constructeur : initialise tous les attributs
— Getters pour chaque attribut
— Méthode virtuelle pure : `solve()` pour résoudre l'EDP
Classes `Crank_Nicholson` **et** `Implicite` : héritent de `DifferenceFinie`.
— `Crank_Nicholson` : méthode de Crank–Nicholson
— `Implicite` : méthode implicite
— La fonction `ThomasAlgo` permet de résoudre efficacement les systèmes tridiagonaux.

3.1.4 Classe pour l'affichage

Classe `Sdl` : permet d'afficher les courbes des prix d'options dans une fenêtre SDL.
Attributs :
— `window_` : pointeur vers la fenêtre SDL
— `renderer_` : pointeur vers le renderer SDL
Méthodes :
— `drawCurve()` : dessine une courbe
— `present()` : affiche le renderer
— `clear()` : nettoie la fenêtre
— `isRunning()` : vérifie si la fenêtre est ouverte

4 Choix techniques

4.1 Architecture orientée objet

Nous avons opté pour une approche rigoureuse de programmation orientée objet, définissant des classes abstraites comme cadres génériques, puis des classes concrètes spécialisées. Cette architecture s'applique uniformément aux options financières, aux équations aux dérivées partielles et aux méthodes numériques.

4.2 Options financières

Pour les options, la classe abstraite `Option` est héritée par `Call` et `Put`, permettant de factoriser les éléments communs tout en imposant l'implémentation des payoffs et des conditions aux limites spécifiques. Les méthodes `lowerBoundary` et `upperBoundary` assurent la cohérence des conditions aux limites.

4.3 Actif sous-jacent

L'option est séparée de l'actif sous-jacent, modélisé par une structure indépendante contenant le prix du sous-jacent, la volatilité et le taux d'intérêt. Le lien entre l'option et l'actif est établi au niveau de la classe `EDP`, conformément à la théorie de Black et Scholes.

4.4 Équations aux dérivées partielles

Les équations aux dérivées partielles sont représentées par la classe `EDP`, dont dérivent `EDPComplete` et `EDPReduite`. Cette structure facilite le passage entre l'équation complète et l'équation réduite de diffusion sans modifier les solveurs numériques, tout en conservant une interface unifiée.

4.5 Méthodes numériques

Les méthodes numériques reposent sur la classe `DifferenceFinie`, héritée par `Implicite` et `Crank_Nicholson`. Cette organisation unifie la gestion des grilles et des structures de données tout en laissant chaque schéma gérer sa formulation spécifique.

4.6 Gestion des conditions aux limites du Call

Lors de l'implémentation, nous avons relevé une divergence entre l'énoncé du projet et la théorie financière classique concernant la condition aux limites supérieure du Call ($S \to L$). L'énoncé préconisait la formule $C(t, L) = Ke^{-r(t-T)}$, tandis que la théorie de Black-Scholes établit que pour un prix de sous-jacent très élevé, la valeur du Call tend vers $L - Ke^{-r(T-t)}$.

Nous avons fait le choix délibéré d'utiliser la condition asymptotique standard, soit $L - Ke^{-r(T-t)}$. En effet, l'utilisation de la formule suggérée par l'énoncé aurait introduit une discontinuité majeure avec le payoff à l'échéance $(L - K)$, empêchant ainsi la convergence du schéma numérique et produisant des résultats financiers incohérents.

4.7 Affichage des résultats

Pour l'affichage, nous avons créé quatre fenêtres SDL distinctes, chacune encapsulée dans la classe `Sdl`, afin de séparer le calcul numérique de la visualisation. Chaque fenêtre correspond à un graphique spécifique : le prix du call, l'erreur du call, le prix du put et l'erreur du put. Cette approche garantit un rendu lisible, modulable et extensible.

Pour simplifier la gestion des événements et résoudre le problème de fermeture simultanée des fenêtres, nous avons choisi de **centraliser l'interaction utilisateur** : l'appui sur la touche `Échap` ferme toutes les fenêtres simultanément. Cela permet d'éviter de devoir gérer indépendamment la fermeture de chaque fenêtre et assure une sortie propre du programme.

5 Difficultés rencontrées et solutions apportées

5.1 Modélisation des options

La première difficulté concernait la modélisation des options. Traduire la notion financière d'option européenne en architecture orientée objet claire et extensible a nécessité de définir clairement les responsabilités de chaque classe. La solution a été la création de la classe abstraite `Option` et l'ajout des méthodes `lowerBoundary` et `upperBoundary` pour gérer correctement les conditions aux limites.

5.2 Schémas numériques et discrétisation

Une seconde difficulté a porté sur la compréhension des schémas numériques et la discrétisation de l'équation de Black–Scholes. La gestion des indices dans les vecteurs et matrices a été source d'erreurs fréquentes, notamment pour l'assemblage des systèmes tridiagonaux. Ces problèmes ont été résolus par un débogage systématique et la vérification progressive des résultats intermédiaires.

5.3 Affichage SDL et gestion des fenêtres

L'utilisation de la bibliothèque SDL a posé des problèmes : la gestion de plusieurs fenêtres simultanément et le tracé manuel des courbes étaient complexes et fastidieux. La solution adoptée a été de créer quatre fenêtres distinctes, chacune encapsulée dans la classe `Sdl`, qui centralise les appels bas niveau et fournit une interface simplifiée pour visualiser les courbes et les erreurs.

Pour résoudre le problème de fermeture des fenêtres, nous avons implémenté un mécanisme unique : **l'appui sur la touche `Échap` ferme toutes les fenêtres en même temps**, garantissant ainsi un rendu lisible, modulable et cohérent pour chaque type d'option tout en assurant une sortie propre du programme.

6 Affichage et interprétation des résultats

Les résultats numériques sont visualisés à l'aide de la bibliothèque SDL. Pour chaque option, deux courbes sont affichées dans une même fenêtre : la solution obtenue par le schéma implicite $C(0, S)$ et celle obtenue par la méthode de Crank–Nicholson $\tilde{C}(0, S)$. Une seconde fenêtre est dédiée à la courbe d'erreur définie comme la différence point à point entre les deux solutions.

L'analyse conjointe des courbes de prix et des courbes d'erreur permet d'évaluer à la fois la cohérence mathématique des schémas numériques et leur précision relative.

6.1 Option Call

La figure 1 présente l'évolution du prix de l'option Call à l'instant initial en fonction du prix du sous-jacent. Les solutions issues du schéma implicite et de la méthode de Crank–Nicholson sont superposées, respectivement en rouge et en bleu. Les erreurs numériques étant très faibles, les deux courbes se confondent visuellement et une seule courbe est observable.

Du point de vue financier, la courbe obtenue respecte les propriétés théoriques d'un Call européen : le prix de l'option est nul pour les faibles valeurs du sous-jacent et croît de manière quasi linéaire lorsque S devient supérieur au strike. La monotonie croissante de la courbe est correctement reproduite par les deux schémas, ce qui valide la mise en œuvre des conditions aux limites et du payoff.

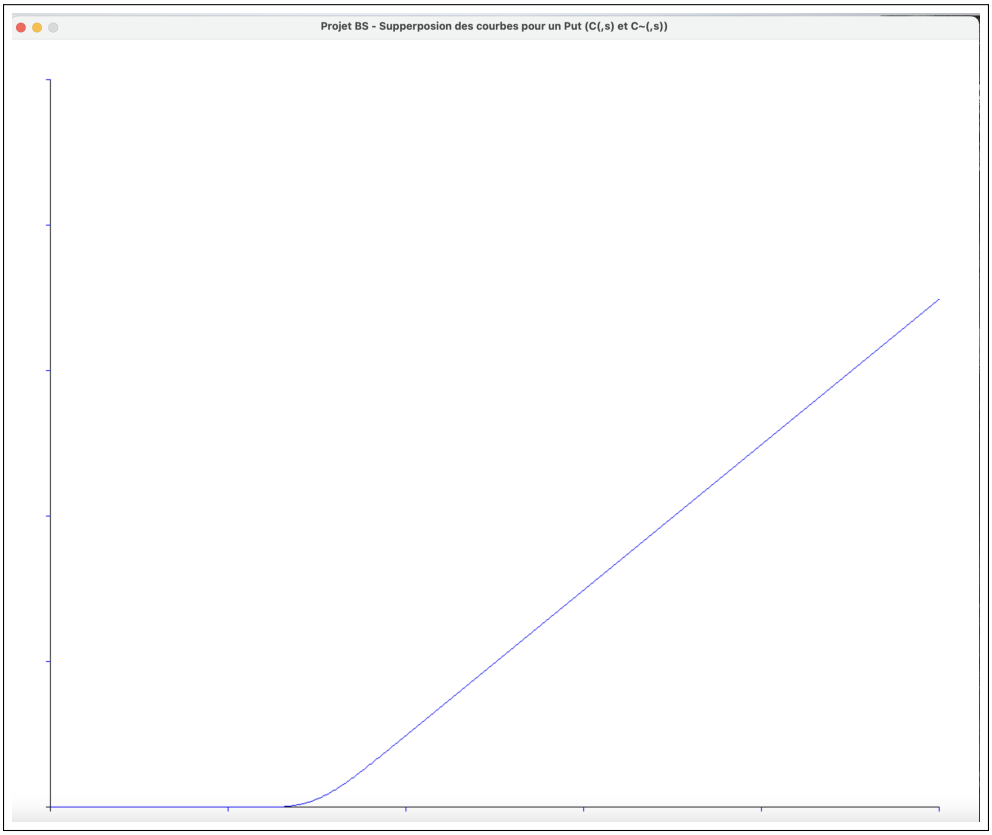

FIGURE 1 – Prix de l'option Call à $t = 0$ obtenu par les méthodes implicite et Crank–Nicholson

L'erreur maximale entre les deux méthodes pour l'option Call est donnée par

$$\max_S \left| C(0,S) - \tilde{C}(0,S) \right|$$

6.2 Erreur entre les deux méthodes pour une option Call

La figure 2 représente la courbe de l'erreur. Elle est nulle au voisinage de $S = 0$, ce qui s'explique par le fait que, pour les deux méthodes, la valeur de l'option est imposée par la même condition aux limites. Les faibles oscillations observées au centre du domaine sont dues à l'approximation numérique et à la discrétisation spatiale, mais leur amplitude reste très limitée.

Cette courbe confirme que la méthode de Crank–Nicholson et le schéma implicite fournissent des résultats très proches, avec un écart numérique négligeable à l'échelle des prix de l'option.

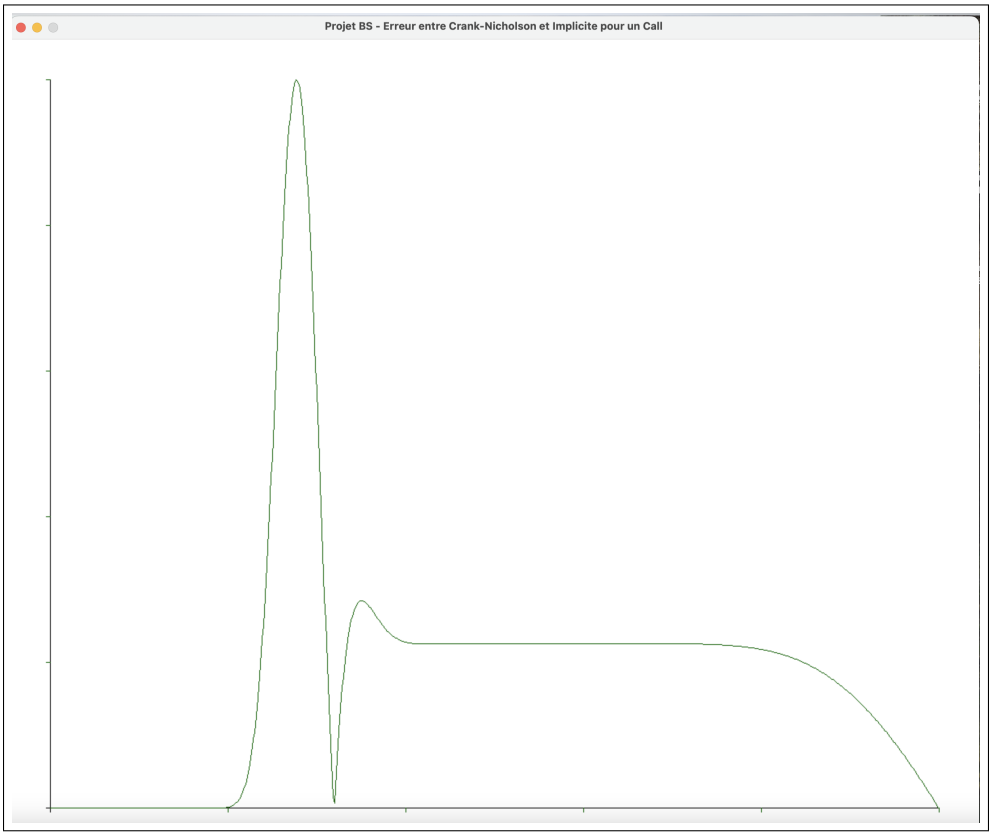

FIGURE 2 – Erreur entre les méthodes Crank–Nicholson et implicite pour l'option Call

6.3 Option Put

La figure 3 illustre le prix de l'option Put à l'instant initial. Comme pour le Call, les courbes $C(0, S)$ et $\tilde{C}(0, S)$ obtenues par les deux méthodes sont superposées et indiscernables visuellement.

La forme de la courbe est conforme aux propriétés théoriques d'un Put européen : le prix de l'option est élevé lorsque le sous-jacent est faible et décroît lorsque S augmente. Les deux schémas numériques reproduisent correctement cette décroissance, ce qui valide à la fois la modélisation du payoff et le traitement des conditions aux limites.

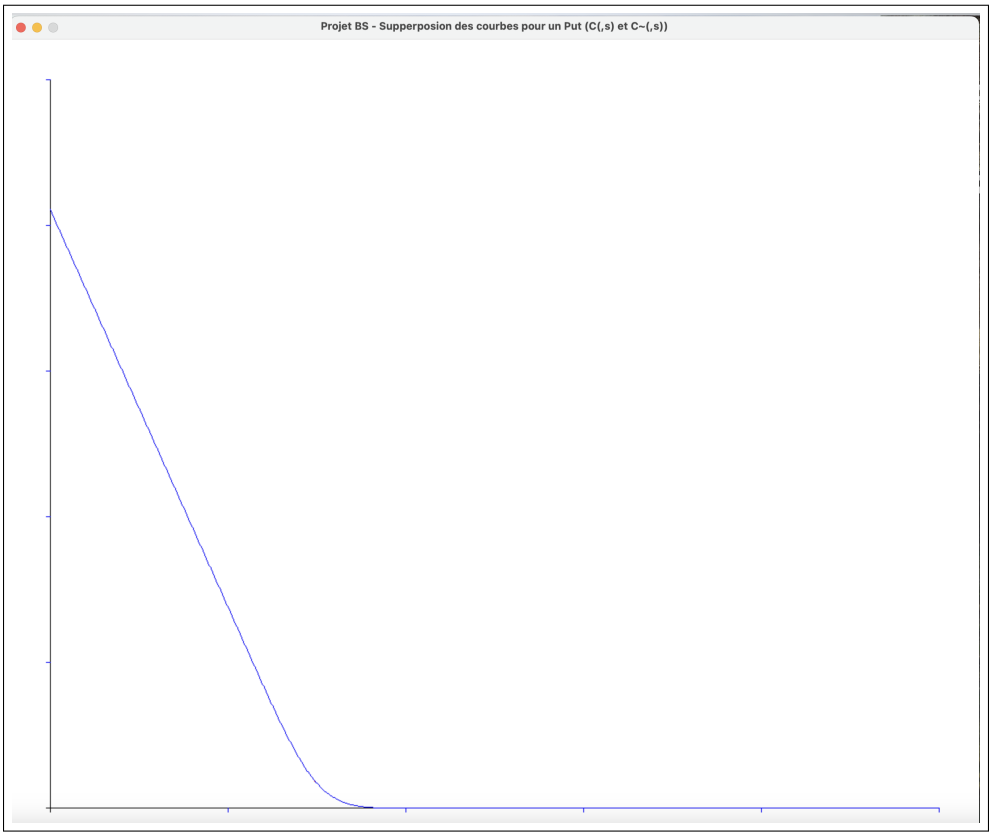

FIGURE 3 – Prix de l'option Put à $t = 0$ obtenu par les méthodes implicite et Crank–Nicholson

L'erreur maximale observée pour l'option Put est

$$\max_S \left| C(0, S) - \tilde{C}(0, S) \right|$$

6.4 Erreur entre les deux méthodes pour une option Put

La figure 4 montre la courbe d'erreur correspondante. Comme pour l'option Call, l'erreur est nulle aux bornes du domaine, où les valeurs de l'option sont imposées analytiquement. À l'intérieur du domaine, les variations restent faibles et régulières, ce qui traduit une bonne stabilité numérique.

L'amplitude légèrement supérieure de l'erreur par rapport au Call s'explique par la forme du payoff du Put, qui présente une pente plus marquée près du strike. Malgré cela, l'erreur reste de l'ordre de 10^{-3}, confirmant la précision et la robustesse des deux méthodes de résolution.

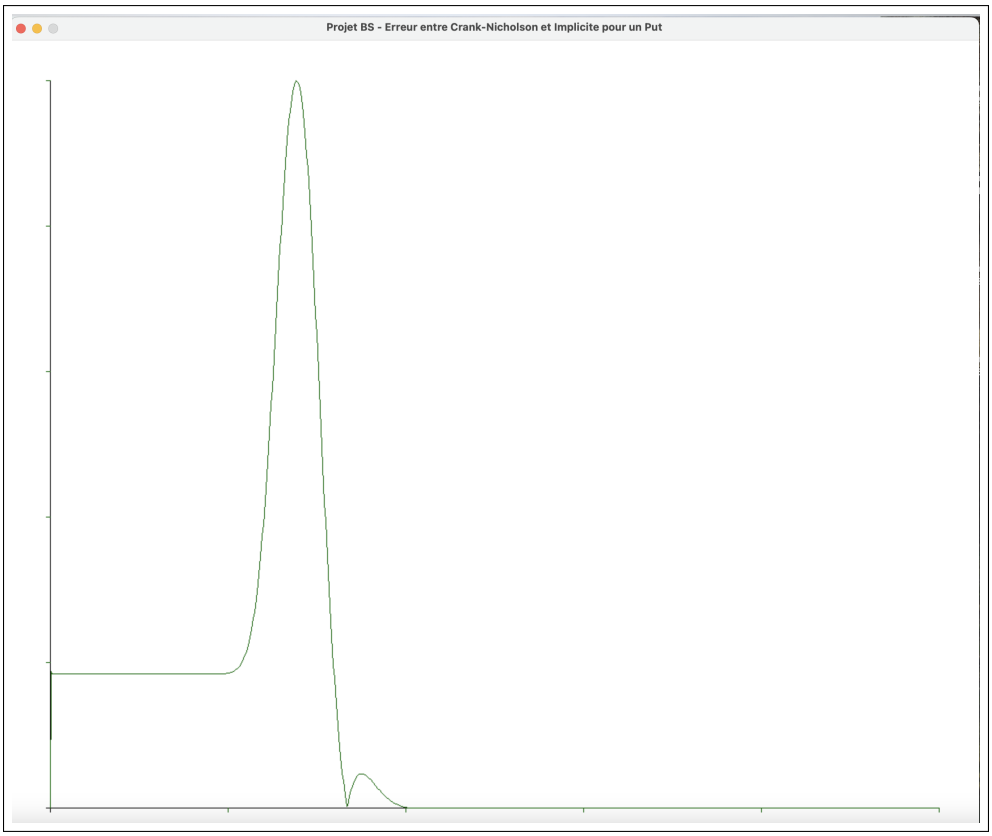

FIGURE 4 – Erreur entre les méthodes Crank–Nicholson et implicite pour l'option Put

7 Limites du modèle et de l'approche numérique

Ce projet présente certaines limites liées à son périmètre et à ses choix techniques.

L'affichage des résultats repose sur la bibliothèque SDL, ce qui conduit à une visualisation volontairement simple. Les courbes sont tracées sans légendes détaillées ni annotations complètes, et la superposition de courbes proches rend parfois la lecture visuelle difficile.

L'analyse numérique reste également limitée. La comparaison entre les méthodes implicite et Crank–Nicholson se concentre sur l'erreur maximale à l'instant initial, sans étude approfondie de la convergence ou de l'influence des paramètres de discrétisation.

Enfin, les tests numériques sont réalisés pour un ensemble restreint de paramètres, ce qui limite la portée expérimentale des résultats.

8 Perspectives et améliorations possibles

Plusieurs prolongements naturels peuvent être envisagés pour enrichir ce travail.

8.1 Amélioration numérique

La précision peut être améliorée par l'utilisation de grilles non uniformes, plus fines autour du strike. Une étude de convergence en fonction de Δt et ΔS permettrait également de valider expérimentalement l'ordre des schémas utilisés. D'autres approches numériques, comme les méthodes de Monte Carlo ou les éléments finis, peuvent aussi être envisagées.

8.2 Calcul des Grecs

Les sensibilités financières peuvent être obtenues par différentiation numérique des solutions discrètes. Le Delta, le Gamma et le Theta peuvent être approchés par des schémas de différences finies, tandis que le Vega peut être estimé en recalculant la solution pour différentes valeurs de la volatilité. Ces quantités, plus sensibles numériquement que le prix, permettraient une analyse plus complète des résultats.

8.3 Extension du cadre

L'architecture orientée objet facilite l'extension à des produits plus complexes. Les options barrières, asiatiques ou américaines peuvent être intégrées moyennant des adaptations des conditions aux limites ou des méthodes de résolution. Enfin, le modèle peut être enrichi pour prendre en compte des dynamiques plus réalistes du sous-jacent.

9 Conclusion

Ce projet a permis de mettre en œuvre une résolution numérique complète de l'équation de Black–Scholes par différences finies, en comparant un schéma implicite et un schéma de Crank–Nicholson. Les résultats obtenus montrent une très bonne cohérence entre les deux approches, confirmant la validité des méthodes employées. L'architecture logicielle proposée offre une base solide pour des extensions futures vers des modèles financiers plus complexes et des produits dérivés plus élaborés.

10 Annexes

10.1 Diagramme UML des classes

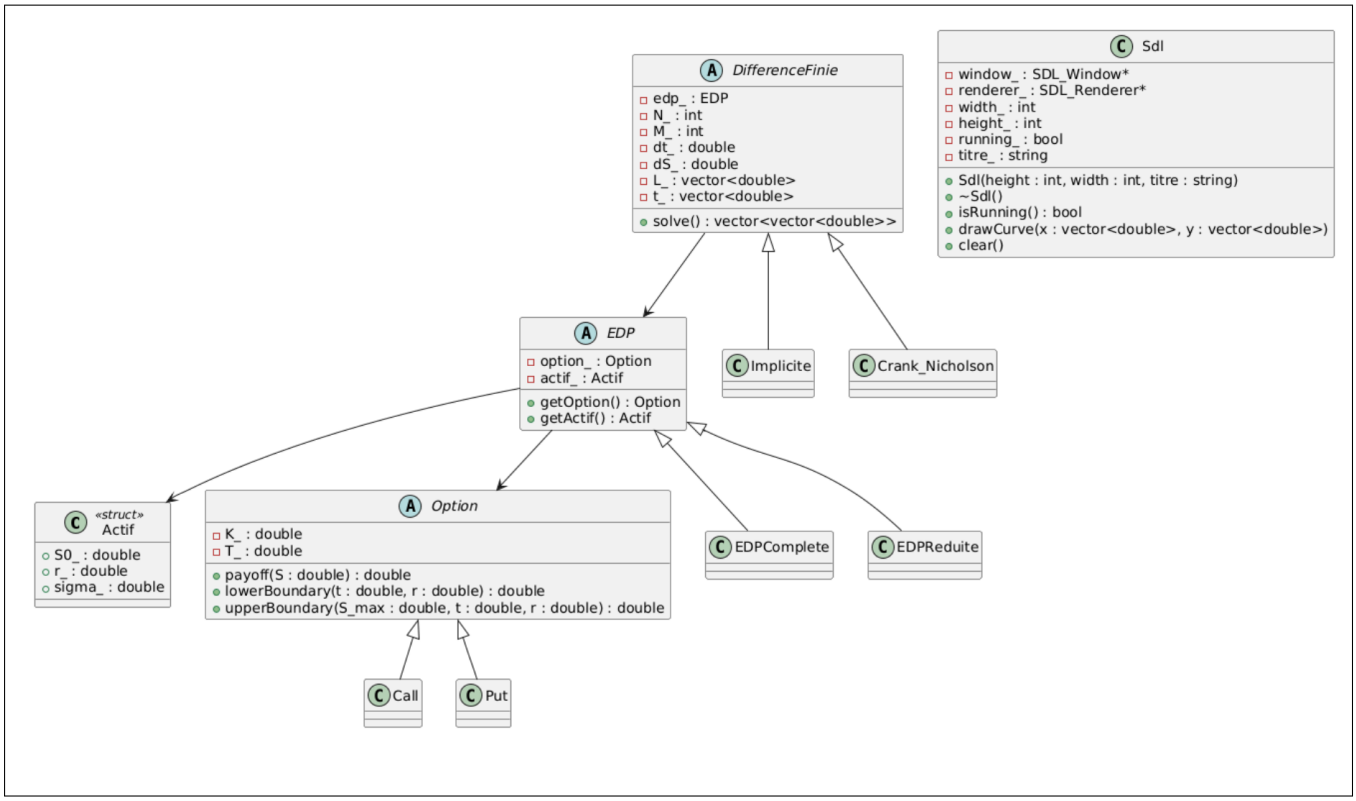

FIGURE 5 – Diagramme UML des classes