

## Equation de Black et Scholes

### 1 Informations pratiques

Ce projet est à effectuer en binômes. Le code sera écrit en C++ et devra être compilable en ligne de commande ainsi :

```
g++ -g -Wall -Wextra -o prog *.cpp $(pkg-config --cflags --libs sdl2)
```

Pas de Makefile, de projet Eclipse, de projet Visua Studio, de projet Xcode ou quoi que ce soit d'autre.

#### Instructions

- Le code devra être indenté de manière uniforme.
- La définition des méthodes et des fonctions ne devront pas dépasser un nombre de lignes raisonnable (au plus 50 lignes).
- Le code devra être commenté en utilisant la syntaxe de Doxygen.
- Le rapport sera écrit en L<sup>A</sup>T<sub>E</sub>X et inclura les choix, les problèmes techniques qui se posent et les solutions trouvées, (la conception (dont un diagramme UML complet) et réalisation). Le soin apporté à la grammaire et à l'orthographe sera largement pris en compte.
- Le projet sera créé sous forme d'archive tar.xz ainsi :
  1. On vérifie que le code compile.
  2. On crée le répertoire *NOM1\_NOM2\_projet\_bs* où *NOM1* et *NOM2* sont les 2 noms de famille en majuscule. PAS de sous-répertoires.
  3. On y met **UNIQUEMENT** les fichiers source, fichiers d'en-tête, les fichiers .tex et .pdf du rapport et le fichier Doxyfile (qui crée la documentation)
  4. On compresse ce répertoire en ligne de commande ainsi :
    - *tar cvf NOM1\_NOM2\_projet\_bs.tar NOM1\_NOM2\_projet\_bs*
    - *xz -9 NOM1\_NOM2\_projet\_bs.tar*
  5. L'archive sera envoyée à [vincent.torri@gmail.com](mailto:vincent.torri@gmail.com) avec pour sujet du mail *NOM1\_NOM2\_projet\_bs* avant le dimanche 4 janvier 2026 à 23h59. Un mail de confirmation sera envoyé dans les 24h. Si ce n'est pas le cas, relancez-moi.

**IMPORTANT :**

- Ne pas attendre le dernier moment pour envoyer l'archive.
- Si plusieurs binômes ont des codes trop similaires, leur note sera divisée par le nombre de binômes impliqués.
- Tout projet rendu en retard se verra attribuer la note de 0.
- Si le projet ne compile pas, la note sera 0. Je n'essaierai pas de corriger les erreurs.
- Non respect du sujet du mail : note diminuée de 1.
- Non respect du nom de l'archive : note diminuée de 1.
- Non respect du nom du sous-répertoire : note diminuée de 1.

**Suggestions fortement recommandées :**

- Créer un dépôt *Github* pour un travail collaboratif.
- Ajouter une intégration continue pour Linux et Windows (avec MinGW), pour s'assurer que le code compile sur ces deux plateformes.

## 2 Sujet

En 1973, Fischer Black et Myron Scholes ont développé un modèle pour calculer le prix d'une option, dite européenne , liant le prix de cette option aux variations de l'actif sous-jacent (que l'on peut considérer comme une action). Leur modélisation du prix de l'actif sous-jacent  $S_t$  vérifie l'équation différentielle stochastique suivante :

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

où  $\sigma$  est la volatilité,  $\mu$  une dérive, et  $W_t$  est un processus de Wiener.

### 2.1 Equation aux dérivées partielles

En utilisant la formule d'Ito, on se ramène à un problème déterministe sous la forme d'une équation aux dérivées partielles (EDP) vérifiée par la fonction  $C(t, S)$  définie sur  $[0, T] \times [0, L]$  :

$$\frac{\partial C}{\partial t} + rS \frac{\partial C}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} = rC \quad \text{sur } [0, T] \times [0, L], \quad (1)$$

avec  $T$  le temps terminal,  $r$  le taux d'intérêt du marché et  $\sigma$  à nouveau la volatilité de l'actif. Il est à noter que la condition est terminale (et non initiale), on part du temps  $T$  et non de 0 comme habituellement.

Pour résoudre une EDP, la condition initiale (ici terminale) et les conditions

aux bords sont nécessaires :

$$C(T, s) \quad \text{condition initiale donnée ci-dessous} \quad (2)$$

$$C(t, 0) \quad \text{condition au bord donnée ci-dessous} \quad (3)$$

$$C(t, L) \quad \text{condition au bord donnée ci-dessous} \quad (4)$$

On utilisera 2 types de condition terminale pour  $C(T, s)$ , nommées **payoff**, dépendant d'un paramètre  $K > 0$  appelé le **strike** :

1. Un *put* :  $C(T, s) = \max(0, K - s)$  pour  $s \in [0, L]$  et dans ce cas :
  - $C(t, 0) = Ke^{-r(T-t)}$  pour tout  $t \in [0, T]$ ,
  - $C(t, L) = 0$  pour tout  $t \in [0, T]$ .
2. Un *call* :  $C(T, s) = \max(0, s - K)$  pour  $s \in [0, L]$  et dans ce cas :
  - $C(t, 0) = 0$  pour tout  $t \in [0, T]$ ,
  - $C(t, L) = Ke^{-r(t-T)}$  pour tout  $t \in [0, T]$ .

## 2.2 EDP réduite

On peut aussi effectuer un changement de variable de telle sorte que l'équation (1) devienne :

$$\frac{\partial \tilde{C}}{\partial \tilde{t}} = \mu \frac{\partial^2 \tilde{C}}{\partial \tilde{s}^2}. \quad (5)$$

**Le but est de calculer  $C(0, s)$  pour tout  $s \in [0, L]$  par les deux méthodes ci-dessus.**

## 3 Travail

Ecrire un programme C++ qui résoud (1) et (5) de manière structurée, sans l'aide de bibliothèque extérieure. On utilisera une classe abstraite pour gérer les deux **payoff** considérés ci-dessus.

Pour les applications numériques, on utilisera les valeurs suivantes :

- $T = 1$
- $r = 0.1$
- $\sigma = 0.1$
- $K = 100$
- $L = 300$

### 3.1 EDP complète

On utilisera le schéma de Cranck-Nicholson (implicite pour  $\frac{\partial^2 C}{\partial S^2}$ ) pour résoudre (1). Une partie du travail consiste à se documenter sur ce schéma numérique.

On fera une discrétisation de  $[0, T]$  en  $M = 1000$  intervalles réguliers et une discrétisation de  $[0, L]$  en  $N = 1000$  intervalles réguliers.

### 3.2 EDP réduite

On utilisera le schéma aux différences finies implicite pour résoudre (5). Une partie du travail consiste à se documenter sur ce schéma numérique. On fera aussi une discrétisation de  $[0, T]$  en  $M = 1000$  intervalles réguliers et une discrétisation de  $[0, L]$  en  $N = 1000$  intervalles réguliers.

### 3.3 Affichage des courbes $C(0, .)$

1. Pour un *put*, afficher dans une fenêtre la courbe qui approche  $C(0, s)$  ainsi que la courbe qui approche  $\gamma(0, s)$  qui lui est superposée, et dans une autre fenêtre, l'erreur entre les deux. Pour afficher une courbe, on utilisera pour cela la bibliothèque SDL, dont l'utilisation sera faite grâce à une classe **Sdl** facilitant son utilisation.
2. Faire de même pour un *call*.