Université d'Évry-Val d'Essonne

Options lookback par la méthode de Monte-Carlo

Rapport de projet M2QF

Auteur : Daniel Badaire - Manuel Griseri
Encadrant : Vincent Torri - Pedro Ferreira

7 février 2026

Remerciements

Nous tenons à exprimer notre profonde gratitude à Mr.Pedro Ferreira et Mr.Vincent Torri pour nous avoir offert l'opportunité de travailler sur ce sujet de recherche particulièrement intéressant.

Nous souhaitons adresser un remerciement à M. Arnaud Gloter et M. Ahmed Kebaier pour leurs enseignements, ainsi qu'à l'ensemble du corps enseignant du Master M2QF pour la qualité de la formation académique qui nous a permis de mener à bien ce projet.

Enfin, nous remercions chaleureusement nos familles pour leur soutien constant, ainsi que pour la relecture et la correction attentive de ce mémoire, qui nous ont permis de l'améliorer et de le finaliser dans les meilleures conditions.

Table des matières

Listings

1. Introduction

Avec l'essor des marchés financiers contemporains, l'évaluation des options est devenue un enjeu central, tant sur le plan théorique que pratique. Les Produits dérivés occupent aujourd'hui une place majeure dans les stratégies de couverture et de gestion du risque, et leur étude mobilise des outils issus des mathématiques, des probabilités et de l'informatique financière, traduisant ainsi la complexité croissante des marchés.

Dans ce contexte, le modèle de Black–Scholes constitue un pilier de la finance moderne. Introduit en 1973 par Fischer Black et Myron Scholes, il fournit un cadre probabiliste rigoureux permettant de valoriser certaines options européennes à partir de la dynamique stochastique de leur actif sous-jacent, s'appuyant notamment sur le mouvement brownien et le calcul stochastique d'Itô.

Cependant, la diversification des instruments financiers a rapidement mis en évidence les limites des modèles analytiques classiques. En particulier, les produits path-dependent, dont le payoff dépend de l'ensemble de la trajectoire du sous-jacent, échappent le plus souvent à toute formule fermée. C'est notamment le cas des option lookback, pour lesquelles la valeur dépend du maximum ou du minimum atteint par le sous-jacent sur la durée de vie du contrat.

Dès lors, les méthodes numériques se sont imposées comme des outils incontournables pour la valorisation de ces produits complexes. Parmi elles, la méthode de Monte-Carlo permet d'approcher efficacement le prix des options path-dependent par simulation de trajectoires sous la mesure risque-neutre, offrant ainsi une grande flexibilité dans le cadre du modèle de Black–Scholes.

L'objectif de ce projet est d'étudier la valorisation d'options lookback par la méthode de Monte-Carlo. Après avoir rappelé le cadre théorique du modèle de Black–Scholes, nous présenterons le principe de la méthode de simulation retenue, le calcul numérique des sensibilités de l'option, ainsi que l'implémentation en langage C++ et son exploitation via une interface Excel.

2. Modélisation mathématique

Dans ce chapitre, nous présentons le cadre mathématique retenu pour l'étude de la valorisation des options lookback. Nous rappelons tout d'abord les hypothèses du modèle de Black–Scholes, qui fournit une description probabiliste standard de la dynamique de l'actif sous-jacent et constitue le cadre de référence de notre étude.

Nous présentons ensuite le principe de valorisation risque-neutre, permettant d'exprimer le prix d'un produit dérivé comme l'espérance actualisée de son payoff sous une mesure de probabilité appropriée. Cette formulation met en évidence le rôle central joué par les espérances conditionnelles dans le calcul des prix théoriques.

Enfin, nous définissons mathématiquement les options lookback et soulignons leur caractère path-dependent.

2.1 Modélisation des marchés financiers

Afin de modéliser le prix d'un produit dérivé, il est nécessaire d'introduire un modèle stochastique décrivant la dynamique incertaine de l'actif sous-jacent.

Espace de probabilité filtrée. On se place sur un espace de probabilité filtré $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{P})$, où :

- Ω est l'ensemble des états possibles du monde,
- $\mathcal{F}$ est une tribu (ou σ-algèbre) décrivant les événements observables,
- $(\mathcal{F}_t)_{t\geq 0}$ est une filtration représentant l'information disponible jusqu'au temps t,
- $\mathbb{P}$ est une mesure de probabilité.

Mouvement brownien. La source d'incertitude est modélisée par un Mouvement brownien standard $(B_t)_{t\geq 0}$, qui est un processus stochastique.

Le cadre mathématique étant posé, nous introduisons dans la section suivante le modèle de Black–Scholes, qui fournit une description probabiliste standard de la dynamique du sous-jacent et servira de base à la simulation des trajectoires utilisées pour la valorisation des options lookback.

2.2 Modèle de Black–Scholes

On considère un marché composé d'un actif sans risque et d'un actif risqué.

— L'actif sans risque, noté S_t^0, évolue selon

$$dS_t^0 = rS_t^0\,dt,$$

dont la solution explicite est

$$S_t^0 = S_0^0 e^{rt}.$$

Ainsi, le rendement de cet actif est parfaitement déterministe et ne comporte aucune incertitude.

— L'actif risqué, noté S_t, suit sous la mesure risque-neutre $\mathbb{Q}$ la dynamique

$$dS_t = rS_t\,dt + \sigma S_t\,dB_t^{\mathbb{Q}},$$

où $(B_t^{\mathbb{Q}})_{t\geq 0}$ est un mouvement brownien standard sous $\mathbb{Q}$.

Le prix du sous-jacent S_t suit ainsi un mouvement brownien géométrique sous la mesure risque-neutre.

Remarque Sous la mesure risque-neutre, le rendement espéré de l'actif risqué est égal au taux sans risque r, ce qui traduit l'absence d'opportunité d'arbitrage dans un marché parfait, conformément aux hypothèses du modèle.

Notations

— S_t^0 : prix de l'actif sans risque à la date t
— S_0 : prix initial de l'actif sous-jacent
— S_t : prix de l'actif sous-jacent à la date t
— T : maturité de l'option
— $r > 0$: taux d'intérêt sans risque constant
— $\sigma > 0$: volatilité constante de l'actif sous-jacent

Hypothèses

— Les marchés sont parfaits et efficients
— Absence d'opportunités d'arbitrage
— Le taux sans risque r et la volatilité σ sont constants
— Absence de dividendes pendant la durée de vie de l'option
— Absence de frais de transaction
— Les prix suivent une loi lognormale, conséquence directe du mouvement brownien géométrique

La solution explicite de cette équation différentielle stochastique est donnée par :

$$S_t = S_0 \exp\Big(\Big(r - \tfrac{1}{2}\sigma^2\Big)t + \sigma B_t^{\mathbb{Q}}\Big).$$

Cette expression sera utilisée dans la suite pour simuler numériquement des trajectoires du sous-jacent, qui constituent la base de la méthode de Monte-Carlo appliquée à la valorisation des options lookback.

Remarque Le caractère géométrique de cette dynamique garantit que le prix de l'actif reste strictement positif et traduit une évolution proportionnelle à sa valeur.

2.3 Valorisation risque-neutre

2.3.1 Principe fondamental

Sous les hypothèses citées précédemment, les prix actualisés des actifs sont des martingales.

Ainsi, le prix à la date $t = 0$ d'un produit dérivé de maturité T et de payoff H est donné par l'espérance actualisée de ce payoff sous la mesure $\mathbb{Q}$:

$$V_0 = \mathbb{E}^{\mathbb{Q}}\Big[e^{-rT} H\Big],$$

où r désigne le taux d'intérêt sans risque supposé constant.

Cette relation constitue le principe fondamental de la valorisation en finance mathématique. Elle permet ainsi de ramener le problème de la valorisation d'un produit dérivé au calcul d'une espérance sous la mesure risque-neutre, indépendamment de la dynamique réelle du marché.

2.3.2 Options lookback

Call lookback Une option lookback de type call à strike flottant et de maturité T est un produit dérivé dont le payoff dépend du minimum atteint par le sous-jacent sur l'intervalle $[0, T]$. Son payoff est défini par :

$$H_{\text{call}} = S_T - \min_{0 \le t \le T} S_t,$$

où $(S_t)_{0 \le t \le T}$ désigne le prix du sous-jacent.

Put lookback De manière symétrique, une option lookback de type put à strike flottant et de maturité T est définie par le payoff :

$$H_{\text{put}} = \max_{0\leq t\leq T} S_t - S_T.$$

2.3.3 Conséquences mathématiques

Caractère path-dependent Les options lookback appartiennent à la classe des produits dérivés dits *path-dependent*. En effet, leur payoff ne dépend pas uniquement de la valeur finale du sous-jacent S_T, mais de l'ensemble de la trajectoire du processus $(S_t)_{0\leq t\leq T}$.

Formellement, le payoff peut s'écrire comme une fonctionnelle du chemin, ce qui les distingue des options européennes standards pour lesquelles le payoff est de la forme $f(S_T)$.

$$H = f\Big((S_t)_{0\leq t\leq T}\Big).$$

Absence de formule fermée Bien que la loi marginale du sous-jacent S_T soit connue explicitement dans le cadre du modèle de Black–Scholes, la valorisation des options lookback nécessite la connaissance de la loi jointe du couple $(S_T, \min_{0\leq t\leq T} S_t)$ ou $(S_T, \max_{0\leq t\leq T} S_t)$.

La complexité de cette loi jointe empêche, dans la plupart des cas, l'obtention d'une formule fermée simple et exploitable pour le prix des options lookback. Ainsi, les méthodes analytiques classiques utilisées pour les options européennes ne sont pas directement applicables.

Motivation du recours à la méthode de Monte-Carlo D'après le principe de valorisation risque-neutre, le prix d'une option lookback s'écrit sous la forme :

$$V_0 = \mathbb{E}^{\mathbb{Q}}\Big[e^{-rT}\, f\Big((S_t)_{0\leq t\leq T}\Big)\Big],$$

où f désigne la fonctionnelle définissant le payoff.

La méthode permet d'approximer cette espérance en simulant un grand nombre de trajectoires du sous-jacent selon la dynamique du modèle de Black–Scholes, en calculant le payoff associé à chaque trajectoire, puis en prenant la moyenne empirique des valeurs obtenues.

3. Méthode Monte-Carlo

3.1 Notions générales

Les méthodes de Monte-Carlo reposent sur un principe probabiliste fondamental : la loi des grands nombres. L'idée consiste à approximer l'espérance d'une variable aléatoire intégrable par une moyenne empirique construite à partir d'un grand nombre de réalisations indépendantes.

Plus précisément, on considère une suite de variables aléatoires indépendantes et identiquement distribuées, de même loi qu'une variable aléatoire X. L'espérance de X est alors estimée numériquement à partir de la moyenne empirique associée. La justification théorique de cette approximation est fournie par le théorème classique de la loi des grands nombres, rappelé ci-dessous.

Théorème : (Loi des grands nombres). Soit $(X_n)_{n\geq 1}$ une suite de variables aléatoires réelles intégrables, indépendantes et identiquement distribuées. Alors,

$$\frac{1}{N}\sum_{n=1}^{N} X_n \longrightarrow \mathbb{E}[X_1],$$

presque sûrement et dans L^1.

On considère pour la preuve et la suite que $E[X] = 0$ et $E[X^4] < \infty$
Démonstration : A.1

La loi des grands nombres garantit ainsi la convergence presque sûre de la moyenne empirique vers l'espérance.

Le problème se pose maintenant d'évaluer théoriquement l'efficacité de cette méthode. Pour cela, il existe des théorèmes qui permettent de calculer la vitesse de convergence de la méthode de Monte Carlo.

Pour commencer, le théorème central limite précise la vitesse de convergence et décrit les fluctuations asymptotiques autour de la valeur limite.

Théorème Central Limite. Soit $(X_n)_{n\geq 1}$ une suite de variables aléatoires indépendantes et de même loi que X. Si $\mathbb{E}(|X|^3) < +\infty$, alors

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}[X]\right) \xrightarrow[n\to\infty]{\mathcal{L}} \mathcal{N}(0,\sigma^2),$$

où $\sigma^2 = \mathrm{Var}(X)$.
Demonstration : A.2

Ce résultat fondamentale justifie d'une part, l'approximation asymptotiquement gaussienne de l'erreur de Monte-Carlo et d'autre part, elle permet la construction d'intervalles de confiance pour les estimateurs numériques :

Erreur statistique Monte-Carlo.
L'erreur statistique de l'estimateur Monte-Carlo est d'ordre

$$\mathcal{O}\left(\frac{1}{\sqrt{N}}\right),$$

indépendamment de la dimension du problème.

Intervalle de confiance asymptotique.
Un intervalle de confiance asymptotique à 95% pour $\mathbb{E}[X]$ est donné par

$$\widehat{I}_N \pm 1.96\,\frac{\widehat{\sigma}}{\sqrt{N}},$$

où $\widehat{\sigma}^2$ désigne la variance empirique de l'échantillon.

Remarques :

1. la méthode de Monte-Carlo ne requiert pas de conditions de régularité particulières sur la fonction considérée. Cette propriété constitue un avantage majeur, notamment dans des contextes où les intégrandes présentent des discontinuités ou une dépendance complexe au chemin.
2. la largeur des intervalles de confiance dépend à la fois du nombre de simulations N et de la variance σ^2, ainsi il est nécessaire de trouver un compromis entre coût et calcul de précision.
3. Dans la pratique la variance n'est pas connue, d'où la necessité de l'estimer par les données simulées en considérant l'estimateur non biaisé suivant : (Le calcul de cet estimateur est explicité dans le chapitre **Implémentation C++**)
   $\widehat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}_n\right)^2$

3.2 Génération des variables aléatoires

La mise en œuvre de la méthode de Monte–Carlo nécessite la génération de variables aléatoires suivant des lois données, en particulier des lois gaussiennes pour la simulation des incréments du mouvement brownien dans le cas de Black–Scholes. D'un point de vue théorique, cette génération repose sur le principe de la transformation, largement utilisé en simulation numérique.

Méthode de la transformation inverse
Soit Y une variable aléatoire réelle de fonction de répartition F_Y, et soit $U \sim \mathcal{U}([0,1])$. On définit l'inverse généralisé de F_Y par

$$F_Y^{-1}(u) := \inf\{x \in \mathbb{R} \mid F_Y(x) \geq u\}, \quad u \in [0,1].$$

Alors la variable aléatoire

$$X = F_Y^{-1}(U)$$

suit la même loi que Y.
Remarque. Si F_Y est continue, alors F_Y^{-1} coïncide avec l'inverse usuel de F_Y et l'on a $F_Y(F_Y^{-1}(u)) = u$ pour tout $u \in [0,1]$.

Ce théorème permet de générer des variables aléatoires suivant une loi donnée à partir de variables uniformes. Dans le cas particulier de la loi normale, on peut citer par exemple l'algorithme de Box–Muller, ou d'autres méthodes reposant sur des transformations équivalentes, largement utilisées dans les générateurs pseudo–aléatoires standards.

Dans notre cadre de simulation, les incréments du mouvement brownien seront simulés sous la forme suivante :

$$\Delta W_t = \sqrt{\Delta t}\, Z, \qquad Z \sim \mathcal{N}(0,1),$$

où Δt désigne le pas de discrétisation temporelle que nous détaillerons par la suite.

3.3 Simulation des trajectoires du modèle de Black–Scholes

Rappelons tout d'abord la solution explicite du modèle de Black–Scholes.

$$S_t = S_0 \exp\Big(\Big(r - \tfrac{1}{2}\sigma^2\Big)\, t + \sigma W_t\Big)\,.$$

Pour la simulation numérique, l'intervalle de temps $[0,T]$ est discrétisé selon une grille uniforme $(t_j)_{j=0,\dots,N}$, avec $\Delta t = T/N$.

Par conséquent, la trajectoire du sous–jacent est simulée de manière exacte sur chaque pas de temps par :

$$S_{t_{j+1}} = S_{t_j} \exp\Big(\Big(r - \tfrac{1}{2}\sigma^2\Big)\,\Delta t + \sigma\sqrt{\Delta t}\,Z_j\Big)\,,$$

où $(Z_j)_{j=0,\dots,N-1}$ est une suite de variables aléatoires indépendantes suivant une loi normale centrée réduite.

Ainsi, cette discrétisation exponentielle correspond à la solution exacte du mouvement brownien géométrique sur chaque intervalle.

Remarque : En effet, dans le modèle de Black–Scholes, la trajectoire du sous-jacent peut être simulée exactement à des dates discrètes, grâce à l'expression fermée de la solution de l'équation différentielle stochastique associée. Il n'est donc pas nécessaire d'avoir recours à un schéma de discrétisation de type Euler pour la simulation du processus lui-même.

3.4 Valorisation Monte-Carlo des options lookback

Comme précisé dans le chapitre précédent, les options lookback appartiennent à la classe des options dites path–dependent.

Ainsi, la méthode de Monte–Carlo constitue une approche naturelle, permettant de simuler directement les trajectoires du sous-jacent et d'en déduire les payoffs associés.

Dans le cadre de ce projet, le minimum ou le maximum du sous-jacent est approché de manière discrète, aux dates de la grille temporelle. Pour chaque trajectoire simulée, une statistique de trajectoire (minimum ou maximum discret) est mise à jour itérativement au cours de la simulation. Le payoff est ensuite évalué à partir de cette statistique, puis actualisé au taux sans risque.

Le prix de l'option est enfin approché par la moyenne empirique des payoffs actualisés :

$$V_0 \approx e^{-rT}\frac{1}{N}\sum_{k=1}^{N} \text{Payoff}^{(k)},$$

où N désigne le nombre de trajectoires simulées.

A noter que cette approximation discrète des extrema introduit toutefois un biais numérique, lié à l'observation du processus à un nombre fini de dates.

3.5 Biais de discrétisation des extrema et correction par pont brownien

3.5.1 Motivation

Dans le cadre de la valorisation des options lookback dans l'estimation du minimum ou du maximum continu atteint par celui-ci sur l'intervalle de temps $[0, T]$.

En effet, l'extrémum continu du processus n'est pas observé directement : il peut être atteint entre deux dates de la grille de discrétisation. En conséquence, une approximation naïve consistant à remplacer l'extrémum continu par le minimum (ou le maximum) discret introduit un biais de discrétisation. Ce constat justifie l'introduction de méthodes correctives, telles que la correction par pont brownien, afin de mieux approcher l'extrémum continu du sous-jacent.

$$\min_{0\le t\le T} S_t \;\approx\; \min_{0\le j\le N} S_{t_j}, \qquad \max_{0\le t\le T} S_t \;\approx\; \max_{0\le j\le N} S_{t_j}.$$

Remarques :

— Le minimum discret surestime en moyenne le minimum continu, tandis que le maximum discret sous-estime le maximum continu.

— Lorsque le nombre de pas de temps N augmente, le pas Δt tend vers zéro et l'approximation discrète converge vers l'extremum continu.

3.5.2 Pont brownien

Le pont brownien permet d'exploiter la loi conditionnelle du processus entre deux instants de discrétisation et d'obtenir une approximation plus fidèle des extrema continus.

Pour commencer, on considère des payoffs du type

$$f(X_T, M_T),$$

où $X_t = \log S_t$, solution du modèle de Black–Scholes, et $M_T = \max_{0\le s\le T} X_s$.

Dans le cadre du modèle de Black–Scholes, le processus (X_t) est un mouvement brownien à dérive. Il peut donc être simulé exactement sur une grille temporelle uniforme $(t_k)_{k=0,\dots,N}$ de pas $h := \Delta = T/N$ selon

$$\bar{X}^n_{t_{k+1}} = \bar{X}^n_{t_k} + \left(r - \tfrac{1}{2}\sigma^2\right)\Delta + \sigma\sqrt{\Delta}\, Z_k, \qquad Z_k \sim \mathcal{N}(0,1).$$

L'idée de base consiste alors à exploiter la loi conditionnelle du processus continu entre deux dates successives t_k et t_{k+1}, sachant les valeurs $\bar{X}^n_{t_k}$ et $\bar{X}^n_{t_{k+1}}$. En effet, conditionnellement à ces valeurs aux bornes, le processus interpolé sur l'intervalle $[t_k, t_{k+1}]$ a la loi d'un pont brownien, ce qui permet de caractériser explicitement la loi de son maximum sur cet

intervalle.

Afin d'introduire la méthode, nous rappelons ci-dessous certaines propriétés essentielles.

Proposition 1.
Soit $(W_t)_{t\geq 0}$ un mouvement brownien standard et $T > 0$. Le processus $(Z_t)_{0\leq t\leq T}$ défini par

$$Z_t = W_t - \frac{t}{T}W_T$$

est un processus gaussien, indépendant de W_T. Il vérifie

$$\mathbb{E}[Z_t] = 0, \qquad \mathbb{E}[Z_t Z_s] = s \wedge t - \frac{st}{T}, \quad \forall (s,t) \in [0,T]^2.$$

Enfin, le mouvement brownien conditionné par $W_T = y$ a même loi que

$$Z_t^y = W_t - \frac{t}{T}\Big(W_T - y\Big).$$

Démonstration : A.4

Proposition 2.
On suppose que $\sigma \neq 0$. Conditionnellement à

$$\bar{X}^n_{t_k} = x_k \qquad \text{et} \qquad \bar{X}^n_{t_{k+1}} = x_{k+1},$$

le processus $(\bar{X}^n_t)_{t\in[t_k,t_{k+1}]}$ a la loi de

$$x_k + \sigma\, Z_{t-t_k},$$

où Z est un pont brownien reliant 0 à $\frac{x_{k+1}-x_k}{\sigma}$ sur l'intervalle $[0,h]$.
Il s'agit d'un processus gaussien d'espérance

$$x_k \frac{t_{k+1}-t}{t_{k+1}-t_k} + x_{k+1}\frac{t-t_k}{t_{k+1}-t_k},$$

et de variance

$$\frac{(t-t_k)(t_{k+1}-t)}{t_{k+1}-t_k}\,\sigma^2.$$

Proposition 3.
Soit

$$Z_t = W_t - \frac{t}{T}\Big(W_T - y\Big),$$

un pont brownien tel que $Z_T = y$. Alors, pour tout $a > y$,

$$\mathbb{P}\left(\max_{t\in[0,h]} Z_t \leq a\right) = 1 - \exp\Big(-\frac{2}{h}a(a-y)\Big)\,.$$

Démonstration : A.5

En couplant les propositions précédentes, on obtient que la loi du maximum du processus interpolé entre t_k et t_{k+1}, conditionnellement aux valeurs aux bornes, admet pour fonction de répartition :

$$\mathbb{P}\left[\max_{t_k\le t\le t_{k+1}} \bar{X}_t^n \le a \;\middle|\; \bar{X}_{t_k}^n = x_k,\ \bar{X}_{t_{k+1}}^n = x_{k+1}\right] = 1-\exp\left(-\frac{2}{h}\frac{(a-x_k)(a-x_{k+1})}{\sigma^2}\right) =: F_h(a,x_k,x_{k+1}).$$

La fonction de répartition inverse s'écrit donc :

$$F_h^{-1}(U,x_k,x_{k+1}) = \frac{1}{2}\left(x_k + x_{k+1} + \sqrt{(x_{k+1}-x_k)^2 - 2\sigma^2 h\ln(U)}\right),$$

où U est une variable aléatoire de loi uniforme sur $[0,1]$.

On peut ainsi simuler un maximum continu conditionnel sur chaque intervalle $[t_k, t_{k+1}]$ par :

$$\hat{m}_k = F_h^{-1}(U_k, x_k, x_{k+1}),$$

où les (U_k) sont indépendantes et uniformes sur $[0,1]$.

Pour finir, une approximation du maximum global est alors donnée par

$$\widehat{M}_T = \max_{1\le k\le N} \hat{m}_k.$$

Cas du minimum. La loi du minimum continu s'obtient par symétrie à partir de celle du maximum. En effet, pour tout processus continu (X_t), on a :

$$\min_{t_k\le t\le t_{k+1}} X_t = -\max_{t_k\le t\le t_{k+1}} (-X_t).$$

Le processus opposé $(-X_t)$ étant encore un pont brownien conditionnellement aux valeurs aux bornes, on applique directement les résultats précédents.

Ainsi, conditionnellement à $X_{t_k} = x_k$ et $X_{t_{k+1}} = x_{k+1}$, la loi du minimum sur $[t_k, t_{k+1}]$ est donnée par

$$\mathbb{P}\left[\min_{t_k\le t\le t_{k+1}} X_t \ge b\right] = 1 - \exp\left(-\frac{2}{h}\frac{(x_k-b)(x_{k+1}-b)}{\sigma^2}\right),$$

et sa simulation s'obtient en remplaçant le signe "+" par "−" devant la racine dans la formule inverse.

Remarque. Dans ce projet, la correction par pont brownien n'est pas utilisée comme méthode principale de valorisation, mais comme un outil de référence numérique. Elle permet d'obtenir une valeur asymptotique servant de benchmark pour l'étude de convergence des estimateurs Monte-Carlo basés sur une discrétisation classique.

4. Calcul des sensibilités (Greeks)

Dans le cadre de la valorisation et de la couverture d'options, il est courant d'introduire les *Greeks*, c'est-à-dire les sensibilités du prix de l'option par rapport à différents paramètres du modèle. Ces quantités jouent un rôle essentiel tant du point de vue de l'analyse du risque que dans la mise en place de stratégies de couverture.

Dans ce chapitre, nous nous intéressons au calcul des principales sensibilités du prix d'une option lookback.

4.1 Définition des sensibilités

$$\Delta = \frac{\partial V}{\partial S}, \qquad \Gamma = \frac{\partial^2 V}{\partial S^2}, \qquad \Theta = \frac{\partial V}{\partial t}, \qquad \rho = \frac{\partial V}{\partial r}, \qquad \nu = \frac{\partial V}{\partial \sigma}.$$

Ici, Δ mesure l'exposition instantanée au sous-jacent, Γ la convexité, Θ l'effet du passage du temps, ρ la sensibilité au taux sans risque, et ν (vega) la sensibilité à la volatilité.

Nous rappelons l'expression du prix de l'option sous la mesure risque–neutre $\mathbb{Q}$:

$$V_0 = V(0, S_0; r, \sigma) = e^{-rT}\, \mathbb{E}^{\mathbb{Q}}[f(\{S_t\}_{0 \le t \le T})]\,,$$

où f désigne le payoff, dépendant de l'ensemble de la trajectoire du sous-jacent, typiquement via un maximum ou un minimum.

Dans ce projet, les Greeks sont estimés *numériquement* en combinant :
— la simulation Monte–Carlo du prix de l'option,
— des schémas de différences finies (*bump and revalue*) appliqués à l'estimateur Monte–Carlo.

4.2 Estimation des Greeks par Monte–Carlo

4.2.1 Principe de la méthode *bump and revalue*

Soit $\widehat{V}_N(\theta)$ l'estimateur Monte–Carlo du prix, où θ représente un paramètre (par exemple S_0, r ou σ). Le principe *bump and revalue* consiste à évaluer le prix pour des valeurs perturbées de θ puis à approximer la dérivée par un quotient de différences finies.

On distingue :
— différences avant : ordre $O(h)$,
— différences centrées : ordre $O(h^2)$,

Remarque importante : Pour réduire l'erreur statistique, il est standard d'utiliser des *nombres aléatoires communs* (*common random numbers*) : on réutilise les mêmes réalisations gaussiennes Z dans les évaluations perturbées. Ceci induit d'une part, une forte corrélation entre estimateurs et, d'autre part, diminue la variance du quotient de différences.

4.2.2 Estimation du Delta

On approche $\Delta = \partial V/\partial S$ par une différence finie centrée :

$$\widehat{\Delta} = \frac{\widehat{V}_N(S_0 + h) - \widehat{V}_N(S_0 - h)}{2h}.$$

Le choix du pas h réalise un compromis :
— si h est trop grand, le biais de discrétisation (erreur de troncature) augmente ;
— si h est trop petit, la variance du quotient augmente et les erreurs d'arrondi peuvent devenir visibles.

Dans la pratique, on retient typiquement un pas relatif $h = \varepsilon S_0$ avec $\varepsilon \in [10^{-4}, 10^{-2}]$ selon l'échelle du problème.

4.2.3 Estimation du Gamma

On estime $\Gamma = \partial^2 V/\partial S^2$ via la formule centrée :

$$\widehat{\Gamma} = \frac{\widehat{V}_N(S_0 + h) - 2\widehat{V}_N(S_0) + \widehat{V}_N(S_0 - h)}{h^2}.$$

L'estimation de Γ est généralement plus instable que celle de Δ, en raison de l'amplification de la variance induite par la division par h^2. De ce fait, l'usage de nombres aléatoires communs est ici particulièrement important pour obtenir une estimation exploitable.

4.2.4 Estimation du Theta

On définit $\Theta = \partial V/\partial t$.

On utilise alors une différence finie (souvent avant, car $T - h$ doit rester positif) :

$$\widehat{\Theta} = \frac{\widehat{V}_N(t + h_t) - \widehat{V}_N(t)}{h_t},$$

ou bien une centrée si $t - h_t > 0$:

$$\widehat{\Theta} = \frac{\widehat{V}_N(t + h_t) - \widehat{V}_N(t - h_t)}{2h_t}.$$

Ici encore, les mêmes trajectoires (mêmes Z) peuvent être réutilisées pour réduire la variance.

4.2.5 Estimation du rho

On rappelle $\rho = \partial V/\partial r$. On l'estime par différence finie centrée :

$$\widehat{\rho} = \frac{\widehat{V}_N(r + h_r) - \widehat{V}_N(r - h_r)}{2h_r}.$$

Notons que r intervient à la fois :

— dans la dynamique du sous-jacent simulé sous la mesure risque–neutre,

— dans le facteur d'actualisation e^{-rT}.

La perturbation de r doit donc être appliquée de manière cohérente à ces deux endroits.

4.2.6 Estimation du vega

On définit le vega $\nu = \partial V/\partial \sigma$. Une approximation centrée est donnée par

$$\widehat{\nu} = \frac{\widehat{V}_N(\sigma + h_\sigma) - \widehat{V}_N(\sigma - h_\sigma)}{2h_\sigma}.$$

La volatilité σ intervient directement dans la loi du sous-jacent (terme $\sigma\sqrt{\Delta t}\, Z$), de sorte que le vega peut être estimé efficacement en réutilisant les mêmes réalisations Z pour les deux évaluations perturbées.

4.3 Limites numériques et sources d'erreur

Les estimations Monte–Carlo des Greeks sont affectées par plusieurs sources d'erreur :

1. **Erreur statistique (Monte–Carlo).** Elle décroît en $O(N^{-1/2})$ et dépend de la variance du payoff.

2. **Biais de discrétisation temporelle.** Le payoff lookback dépend d'extrema ; l'utilisation d'un maximum/minimum discret induit un biais qui décroît lorsque N augmente.
3. **Biais de différence finie.** Les dérivées sont approchées avec un pas h : biais en $O(h)$ (avant) ou $O(h^2)$ (centrée).
4. **Variance amplifiée pour les dérivées d'ordre 2.** Le calcul de Γ est particulièrement sensible car il divise par h^2.

Remarque importante : En pratique, on choisit conjointement N (nombre de trajectoires), N_{steps} (pas de temps) et les pas de différence finie (h, h_r, h_σ, h_T) afin d'obtenir un compromis satisfaisant entre précision et coût de calcul. L'utilisation de nombres aléatoires communs est recommandée pour stabiliser les estimations.

5. Méthode de Réduction de la variance

On se place dans le cadre du modèle de Black–Scholes en dimension 1.

Même si l'estimateur de Monte-Carlo est convergent, il demeure une variable aléatoire et possède une variance propre. Celle ci peut être élevée dans le cas d'options *path-dependent*, telles que les options lookback, ce qui dégrade la précision numérique pour un nombre raisonnable de simulations.

C'est pourquoi nous avons décidé d'utiliser la **méthode des variables antithétiques** afin de réduire la variance et ainsi améliorer l'efficacité de l'estimation par Monte-Carlo.

Cette méthode repose sur l'exploitation de la symétrie du mouvement brownien :

$$(W_t)_{t\geq 0} \overset{\mathcal{L}}{=} (-W_t)_{t\geq 0}.$$

De ce fait, cela implique que les variables aléatoires suivantes ont la même loi :

$$S_T = S_0 \exp\Big(\Big(r - \tfrac{\sigma^2}{2}\Big)\, T + \sigma W_T\Big) \quad \text{et} \quad S_T^- = S_0 \exp\Big(\Big(r - \tfrac{\sigma^2}{2}\Big)\, T - \sigma W_T\Big)$$

Ainsi, pour tout payoff mesurable f, on obtient alors l'expression suivante qui montre que l'estimateur antithétique est non biaisé :

$$\mathbb{E}\left[\frac{f(S_T) + f(S_T^-)}{2}\right] = \mathbb{E}[f(S_T)],$$

Le théorème suivant formalise et fournit une condition suffisante garantissant une réduction effective de la variance.

Théorème (Réduction de variance par variables antithétiques)
Si

$$\mathrm{Cov}\Big(f(S_T), f(S_T^-)\Big) \leq 0,$$

alors l'estimateur antithétique

$$\hat{V}^{\text{anti}} = \frac{1}{N/2} \sum_{i=1}^{N/2} \frac{f(S_T^{(i)}) + f((S_T^-)^{(i)})}{2}$$

admet une variance strictement inférieure à celle de l'estimateur Monte-Carlo standard basé sur N trajectoires indépendantes.

Démonstration : A.3

Remarque : Dans le cas des options lookback, le payoff f dépend de la trajectoire du sous-jacent et est généralement monotone par rapport au mouvement brownien, ce qui assure en pratique une covariance négative et rend la méthode des variables antithétiques particulièrement efficace.

6. Implémentation C++

6.1 Introduction

L'implémentation numérique des méthodes de valorisation développées dans ce mémoire a été réalisée en langage `C++`. Ce choix a été motivé par les performances élevées du langage, sa gestion précise de la mémoire et son usage largement répandu dans les domaines du calcul scientifique et de la finance quantitative.

L'héritage est largement utilisé dans notre code pour définir des interfaces génériques, notamment pour les payoffs et les modèles, tout en permettant des implémentations spécifiques au sein des classes dérivées grâce aux méthodes virtuelles.

En complément, l'utilisation de *templates* nous a permis de favoriser la réutilisabilité du code tout en conservant une efficacité optimale.

6.2 `Payoff.h` : interface des payoffs

Ce fichier définit l'interface `Payoff` et plusieurs payoffs concrets (call, put, digitaux...). Le but est de séparer clairement la *forme du payoff* du reste du moteur (simulation, valorisation, etc.).

On obtient ainsi une conception extensible : ajouter un nouveau payoff revient tout simplement à créer une nouvelle classe dérivée et à surcharger **operator()**.

```
1 namespace opt {
2 class Payoff {
3 public:
4     virtual double operator()(double S, double K) const = 0;
5 };}
```

Listing 6.1 – Interface Payoff

Remarque : Les classes filles **PayoffCall** et **PayoffPut** implémentent les payoffs d'options européennes (call et put), qui servent de briques de base pour la construction de nos options path-dépendantes. En effet, dans la suite, le paramètre K n'est plus interprété comme un strike constant : il sera remplacé par une quantité **agg**, issue d'une agrégation de

la trajectoire (par exemple un maximum ou un minimum), que nous définirons précisément dans les sections suivantes.

```
    class PayoffCall : public Payoff {
    public:
        double operator()(double S, double K) const override;
    };
    class PayoffPut : public Payoff {
    public:
        double operator()(double S, double K) const override;
    };
```

Listing 6.2 – Payoff Call et Put

6.3 `Aggregator.h` : agrégation path-dépendante

Ce fichier introduit une interface `Aggregator` dont le rôle est de construire, pas à pas, une quantité *agrégée* à partir d'une trajectoire du sous-jacent.

Cette quantité, notée `agg`, résume ainsi l'information path-dépendante utile (moyenne, maximum, minimum, etc.) et sera ensuite utilisée dans le payoff à la place d'un paramètre constant (comme un strike).

L'interface impose une mise à jour récursive via :
— `agg` est la valeur agrégée à l'étape précédente.
— `price` le prix courant.
— `step` l'indice du pas.

```
namespace opt {
class Aggregator {
public:
    virtual double operator()(double agg, double price, double step)
       const = 0;
};}
```

Listing 6.3 – Interface `Aggregator` pour l'agrégation path-dépendante (`Aggregator.h`)

Remarque. Le paramètre `step` permet d'écrire des mises à jour *en ligne* (sans stocker toute la trajectoire), en particulier pour les moyennes : on met à jour la moyenne à l'étape n à partir de la moyenne à l'étape $n-1$. Pour le maximum/minimum, `step` n'intervient pas mais l'interface est conservée pour homogénéiser l'API.

Agrégateurs concrets. Le fichier propose ensuite quatre implémentations prêtes à l'emploi dont deux qui nous serviront dans notre projet :
— moyenne arithmétique

— moyenne géométrique
— maximum (lookback max)
— minimum (lookback min)

```
...
class LookMax : public Aggregator {
public:
    double operator()(double agg, double price, double step) const
        override {
         return std::max<double>(agg, price);
    }};

class LookMin : public Aggregator {
public:
    double operator()(double agg, double price, double step) const
        override {
         return std::min<double>(agg, price);
    }};
```

Listing 6.4 – Agrégateurs : moyenne arithmétique, moyenne géométrique, maximum et minimum (`Aggregator.h`)

Remarque : Cette méthode est particulièrement efficace car on évite ainsi de conserver la trajectoire complète $(S_{t_i})_i$, ce qui ramène la mémoire de $O(N)$ à $O(1)$ par trajectoire, point crucial en Monte-Carlo où nous pouvons avoir beaucoup de chemins et de pas.

6.4 `Option.h` : socle commun et statistiques Monte–Carlo

Ce fichier définit une structure **MCStats** et une classe abstraite **Option**. L'objectif est de centraliser les éléments communs à toutes les options simulées sous Black–Scholes.

Structure MCStats

Cette structure encapsule les informations essentielles renvoyées par une estimation Monte–Carlo : moyenne empirique, erreur standard, et bornes d'un intervalle de confiance à 95%.

```
struct MCStats {
    double estimate = std::numeric_limits<double>::quiet_NaN();
    double stdError = std::numeric_limits<double>::quiet_NaN();
    double ciLow = std::numeric_limits<double>::quiet_NaN();
    double ciHigh = std::numeric_limits<double>::quiet_NaN();
};
```

Listing 6.5 – Structure `MCStats` : estimateur, erreur standard et IC 95% (`Option.h`)

Classe abstraite Option

Cette classe stocke les paramètres du modèle de Black–Scholes `(S0, R, sigma, T0, T)` et fournit des utilitaires partagés :

— `validate()` pour contrôler la cohérence des paramètres,

— `makeCI95` renvoie un objet de type **MCStats** en construisant un intervalle de confiance à 95% à partir de la moyenne et de l'erreur standard.

```
class Option {
protected:
    double S0_, R_, sigma_, T0_, T_;
    void validate() const;
    static MCStats makeCI95(double mean, double stdError);

public:
    Option(double S0, double R, double sigma, double T0, double T);
    virtual ~Option() = default;
    ...
    virtual MCStats priceMC(int paths, int steps, std::uint64_t seed,
       bool antithetic) const = 0;
};
```

Listing 6.6 – Interface `Option` : paramètres Black–Scholes et méthode virtuelle `priceMC` (`Option.h`)

Remarque. La méthode `priceMC` est purement virtuelle et sera explicitée dans la section suivante.

6.5 `Asian.h` : option path-dépendante, Monte–Carlo et grecques

Ce fichier implémente une option *path-dépendante* sous Black–Scholes valorisée par Monte–Carlo. La classe `Asian` hérite de `Option` et est **générique** : elle est paramétrée par un type de payoff `TPayoff` et un type d'agrégateur `TAggregator`.

```
template <typename TPayoff, typename TAggregator>
class Asian : public Option {
private:
    TPayoff payoff_;
    TAggregator aggregator_;
    // ...
public:
    Asian(double S0, double R, double sigma, double T0, double T,
          const TPayoff& payoff, const TAggregator& aggregator);
    MCStats priceMC(int paths, int steps, std::uint64_t seed, bool
       antithetic) const override;
    // greeks : deltaMC, gammaMC, thetaMC, rhoMC, vegaMC
};
```

Listing 6.7 – Déclaration template de la classe `Asian<TPayoff,TAggregator>` (`Asian.h`)

Simulation d'une trajectoire et payoff actualisé

La méthode `discountedPayoffFromZ` simule une trajectoire discrète entre T_0 et T à partir d'un vecteur gaussien `Zs`. Ainsi, à chaque pas, le prix `St` est mis à jour via la discrétisation log-normale du modèle de Black–Scholes, puis l'agrégat `agg` est mis à jour grâce à `aggregator_`. Enfin, la valeur renvoyée est le payoff multiplié par le facteur d'actualisation $\exp(-R(T-T_0))$.

```
double discountedPayoffFromZ(double S0, double R, double sigma, double
    T0, double T, int steps,
     const std::vector<double>& Zs, bool flip) const
{
    double Tau = T - T0;
    double dt = Tau / static_cast<double>(steps);
    double disc = std::exp(-R * Tau);
    double St = S0;
    double agg = S0;
    for (int j = 0; j < steps; ++j) {
        double Z = flip ? -Zs[j] : Zs[j];
        St *= std::exp((R - 0.5 * sigma * sigma) * dt + sigma * std::
            sqrt(dt) * Z);
        agg = aggregator_(agg, St, static_cast<double>(j + 1));
    }
    return disc * payoff_(St, agg);
}
```

Listing 6.8 – Payoff actualisé sur une trajectoire discrète (`discountedPayoffFromZ`)

Moteur Monte–Carlo générique et variables antithétiques

La fonction `runMC` encapsule le calcul Monte–Carlo : elle génère les gaussiennes, évalue un estimateur *par trajectoire* via une fonction `sampleFn`, puis calcule moyenne, variance, erreur standard et IC 95%.
La moyenne et la variance sont mises à jour de manière incrémentale au fil des trajectoires, ce qui évite de stocker tous les tirages et permet un calcul en une seule passe.
Lorsque `antithetic=true`, on réutilise les mêmes `Zs` en remplaçant Z par $-Z$ afin de réduire la variance.

```
template <typename SampleFn>
MCStats runMC(int paths, int steps, std::uint64_t seed, bool antithetic,
    SampleFn&& sampleFn) const
{
    std::mt19937_64 rng(seed);
```

```
    std::normal_distribution<double> nd(0.0, 1.0);

    int m = 0;
    double mean = 0.0;
    double M2 = 0.0;

    auto pushSample = [&](double x) {
        ++m;
        double d = x - mean;
        mean += d / m;
        double d2 = x - mean;
        M2 += d * d2;
    };

    if (antithetic) {
        int pairs = (paths + 1) / 2;
        for (int i = 0; i < pairs; ++i) {
            std::vector<double> Zs(steps);
            for (int j = 0; j < steps; ++j) Zs[j] = nd(rng);
            double s1 = sampleFn(Zs, false);
            double s2 = sampleFn(Zs, true);
            pushSample(0.5 * (s1 + s2));
        }
    } else {
        for (int i = 0; i < paths; ++i) {
            std::vector<double> Zs(steps);
            for (int j = 0; j < steps; ++j) Zs[j] = nd(rng);
            pushSample(sampleFn(Zs, false));
        }
    }

    double var = (m > 1) ? (M2 / (m - 1)) : 0.0;
    double se = (m > 0) ? std::sqrt(var / m) : std::numeric_limits<
       double>::quiet_NaN();
    return Option::makeCI95(mean, se);
}
```

Listing 6.9 – Moteur `runMC` : Welford + antithétiques + IC 95% (`Asian.h`)

Pricing et greeks par bump-and-reprice

La méthode `priceMC` appelle `runMC` avec un estimateur par trajectoire égal au payoff actualisé. Les grecques sont calculées par **différences finies centrées** (bump-and-reprice) en réutilisant les mêmes vecteurs gaussiens `Zs`, ce qui stabilise l'estimation.

```
MCStats priceMC(int paths, int steps, std::uint64_t seed, bool
    antithetic) const override
{
```

```
    auto samplePrice = [&](const std::vector<double>& Zs, bool flip) -> double {
        return discountedPayoffFromZ(S0_, R_, sigma_, T0_, T_, steps, Zs, flip);
    };
    return runMC(paths, steps, seed, antithetic, samplePrice);
}

MCStats deltaMC(int paths, int steps, std::uint64_t seed, bool antithetic, double relEps = 1e-4) const
{
    double eps = relEps * S0_;
    auto sampleDelta = [&](const std::vector<double>& Zs, bool flip) -> double {
        double Pu = discountedPayoffFromZ(S0_ + eps, R_, sigma_, T0_, T_, steps, Zs, flip);
        double Pd = discountedPayoffFromZ(S0_ - eps, R_, sigma_, T0_, T_, steps, Zs, flip);
        return (Pu - Pd) / (2.0 * eps);
    };
    return runMC(paths, steps, seed, antithetic, sampleDelta);
}
```

Listing 6.10 – Pricing Monte–Carlo et delta par différence centrée (`Asian.h`)

Référence asymptotique Brownian Bridge (lookback uniquement)

Le fichier contient enfin une méthode `priceMC_BrownianBridge_Asymptotic` destinée à produire une **valeur numérique de référence** pour l'étude de convergence des options *lookback*.

Elle utilise une correction de type pont brownien, explicitée dans les sections précédentes, pour approcher l'extrémum continu (minimum ou maximum) entre deux dates discrètes.

6.6 Relation de récurrence pour la moyenne et la variance

Dans le cadre de la méthode de Monte-Carlo, l'estimation du prix et de l'erreur statistique repose sur le calcul de la moyenne empirique et de la variance des payoffs simulés. Lorsque le nombre de trajectoires est élevé, il est donc numériquement inefficace de stocker l'ensemble des réalisations afin de calculer ces quantités a posteriori.

C'est pourquoi nous avons décidé de mettre à jour la moyenne et la variance de manière récursive à chaque nouvelle observation.

6.6.1 Moyenne empirique

Soit $(X_n)_{n\geq 1}$ une suite de réalisations indépendantes d'une variable aléatoire intégrable. La moyenne empirique après n observations est définie par

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

Cette quantité peut être mise à jour récursivement selon la relation suivante avec la condition initiale $\overline{X}_0 = 0$:

$$\overline{X}_n = \overline{X}_{n-1} + \frac{X_n - \overline{X}_{n-1}}{n}, \qquad n \geq 1,$$

Cette formule permet de calculer la moyenne empirique en une seule passe, sans conserver l'ensemble des observations.

6.6.2 Variance empirique

La variance empirique est donnée par

$$\widehat{\sigma}_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X}_n)^2, \qquad n \geq 2.$$

On introduit la quantité

$$\sigma_n = \sum_{i=1}^{n}(X_i - \overline{X}_n)^2.$$

En posant

$$\delta = X_n - \overline{X}_{n-1},$$

La relation de récurrence pour σ_n est alors donnée par

$$\sigma_n = \sigma_{n-1} + \delta\,(X_n - \overline{X}_n)$$

La variance empirique non biaisée est finalement obtenue par

$$\widehat{\sigma}_n^2 = \frac{\sigma_n}{n-1}.$$

La démonstration est triviale.

7. Interface Excel – VBA

7.1 Principe général de l'interface Excel–C++

Afin de rendre les méthodes de valorisation accessibles à un utilisateur non spécialiste du langage C++, une interface Excel a été développée. Cette interface repose sur l'utilisation d'une *bibliothèque dynamique* (DLL) écrite en C++, qui encapsule l'ensemble des calculs numériques liés à la simulation Monte–Carlo.

Le principe général est le suivant : Excel sert d'interface utilisateur pour la saisie des paramètres et le lancement des calculs, tandis que le code C++ assure l'exécution des algorithmes de valorisation de manière performante. Les résultats sont ensuite renvoyés vers Excel pour être affichés et analysés.

Cette approche permet de combiner la performance du C++ pour les calculs intensifs avec la souplesse et l'ergonomie d'Excel pour l'expérimentation numérique.

7.2 Création de la DLL et intégration dans Excel

Le code C++ est compilé sous la forme d'une DLL exposant plusieurs fonctions via le mot-clé `__declspec(dllexport)`. Ces fonctions sont déclarées avec une interface en langage C (`extern "C"`) afin d'assurer une compatibilité directe avec Excel et d'éviter certains problèmes.

Chaque fonction exportée correspond à une opération précise. Nous pouvons avoir, par exemple, le calcul du prix, des sensibilités ou encore la récupération des statistiques associées. Depuis Excel, ces fonctions sont appelées via des macros VBA ou directement depuis la feuille, selon une logique simple : paramètres en entrée, calcul dans la DLL, résultats en sortie.

Cette architecture assure une séparation claire entre la logique de calcul (C++) et l'interface utilisateur (Excel).

7.3 Paramétrage des simulations dans la feuille Excel

La feuille Excel permet de paramétrer entièrement les simulations sans modification du code source. Les paramètres sont organisés en plusieurs catégories :

- paramètres de marché : prix initial S_0, taux sans risque r, volatilité σ, maturité T ;
- paramètres numériques : nombre de trajectoires Monte–Carlo, nombre de pas de temps, graine du générateur aléatoire ;
- choix du produit : type d'option lookback (minimum ou maximum), call ou put ;
- options de calcul : activation des variables antithétiques et de la correction par pont brownien.

Chaque cellule de paramétrage correspond directement à un argument transmis à la DLL. Cette organisation permet de comparer facilement différentes configurations numériques et d'étudier l'impact de chaque paramètre sur les résultats.

7.4 Résultats retournés et interprétation

À l'issue du calcul, la DLL renvoie à Excel les résultats suivants, avec et sans réduction de variance :

- le prix estimé de l'option ;
- les sensibilités (Greeks).
- l'erreur standard associée à l'estimateur Monte–Carlo ;
- un intervalle de confiance à 95% ;

Les résultats sont affichés directement dans la feuille Excel, ce qui permet une analyse visuelle et quantitative des sorties du modèle. En particulier, plusieurs graphiques sont utilisés : l'évolution du prix de l'option en fonction du sous-jacent, la comparaison du prix et du delta, ainsi qu'un graphique tridimensionnel représentant le prix et le delta en fonction du temps et du sous-jacent.

Ces représentations facilitent l'étude de la convergence du prix, la comparaison avec les méthodes de réduction de variance et l'interprétation de l'impact des différents paramètres de marché.

L'interface Excel constitue ainsi un outil expérimental complet, facilitant la validation et l'exploitation des méthodes développées.

8. Conclusion et perspectives

8.1 Bilan du projet

Ce projet avait pour objectif principal la mise en œuvre et l'analyse de méthodes numériques de type Monte–Carlo pour la valorisation d'options path-dépendantes, en particulier les options lookback, dans le cadre du modèle de Black–Scholes.

Une attention particulière a été portée à la structuration du code C++, fondée sur les principes de la programmation orientée objet, afin de construire un moteur de valorisation modulaire, extensible et performant. L'utilisation conjointe de l'héritage, du polymorphisme et des templates a permis de séparer clairement les notions de payoff, d'agrégation path-dépendante et de moteur Monte–Carlo.

Sur le plan numérique, le projet a permis d'étudier le comportement statistique des estimateurs Monte–Carlo, l'impact du biais de discrétisation des extrema dans le cas des options lookback, ainsi que l'apport de techniques de réduction de variance, notamment la méthode des variables antithétiques. La correction par pont brownien a été utilisée comme référence numérique pour analyser la convergence des estimateurs.

Enfin, le développement d'une interface Excel connectée à une DLL C++ a rendu l'outil facilement exploitable, en offrant un environnement interactif pour le paramétrage des simulations et l'analyse des résultats. Ce projet constitue ainsi une synthèse cohérente entre théorie financière, méthodes numériques et implémentation logicielle, et fournit une base solide pour l'étude de produits dérivés plus complexes.

8.2 Ouvertures possibles

Dans ce projet, nous avons mis en œuvre une méthode de réduction de variance par variables antithétiques afin d'améliorer l'efficacité numérique. Cette technique présente l'avantage d'être simple à implémenter, peu coûteuse en calcul et directement compatible avec la simulation exacte du modèle de Black–Scholes.

Cependant, d'autres méthodes de réduction de variance auraient également pu être envisagées. Parmi celles-ci, nous pouvons citer la technique de *importance sampling* qui

constitue une approche particulièrement pertinente dans le cadre de payoffs fortement dépendants des événements extrêmes, comme c'est le cas pour les options lookback. En effet, le principe consiste à modifier la mesure de probabilité afin de favoriser la simulation des trajectoires les plus contributives au payoff, puis à corriger ce biais par un poids de vraisemblance.

L'avantage principal de l'importance sampling réside donc dans sa capacité à réduire drastiquement la variance lorsque les événements rares jouent un rôle dominant dans la valorisation. En revanche, cette méthode présente des inconvénients notables : le choix optimal du changement de mesure n'est pas trivial, dépend fortement du produit étudié, et une mauvaise calibration peut au contraire dégrader les performances numériques. De plus, son implémentation est généralement plus complexe que celle des variables antithétiques.

Enfin, ce travail pourrait être étendu à des cadres plus généraux. On pourrait notamment considérer des modèles de volatilité locale ou stochastique, pour lesquels la simulation exacte du sous-jacent n'est plus disponible, rendant alors nécessaires des schémas de discrétisation tels que le schéma d'Euler ou de Milstein. Une autre extension naturelle consisterait à étudier la valorisation et la couverture des options lookback dans un cadre multi-actifs ou en présence de sauts, ce qui soulèverait de nouveaux enjeux numériques et théoriques.

Ces différentes perspectives montrent que la valorisation des options exotiques par Monte-Carlo constitue un domaine riche, dans lequel le choix des méthodes numériques reste étroitement lié à la structure du produit étudié et aux compromis entre précision, complexité et coût de calcul.

A. Démonstrations détaillées

A.1 Preuve de la loi des grands nombres (via moment d'ordre 4)

Énoncé. Soit $(X_n)_{n\geq 1}$ une suite de variables aléatoires réelles i.i.d. telle que

$$\mathbb{E}[X_1] = \mu \qquad \text{et} \qquad \mathbb{E}\Big[|X_1|^4\Big] < +\infty.$$

Alors,

$$\frac{1}{n}\sum_{k=1}^{n} X_k \xrightarrow[n\to\infty]{a.s.} \mu.$$

Réduction au cas centré. Posons $Y_k := X_k - \mu$. Alors $(Y_k)_{k\geq 1}$ est i.i.d., $\mathbb{E}[Y_1] = 0$ et $\mathbb{E}[|Y_1|^4] < \infty$. Il suffit donc de montrer que, en notant

$$S_n := \frac{1}{n}\sum_{k=1}^{n} Y_k,$$

on a $S_n \to 0$ presque sûrement.

Calcul de $\mathbb{E}[S_n^4]$. On écrit

$$\mathbb{E}[S_n^4] = \frac{1}{n^4}\,\mathbb{E}\left[\left(\sum_{k=1}^{n} Y_k\right)^4\right].$$

En développant $\left(\sum_{k=1}^{n} Y_k\right)^4$ et en utilisant l'indépendance ainsi que $\mathbb{E}[Y_1] = 0$, les seuls termes non nuls dans l'espérance sont : (i) les termes Y_i^4, et (ii) les termes $Y_i^2 Y_j^2$ pour $i \neq j$. On obtient donc

$$\mathbb{E}\left[\left(\sum_{k=1}^{n} Y_k\right)^4\right] = n\,\mathbb{E}[Y_1^4] + 6\sum_{1\leq i<j\leq n} \mathbb{E}[Y_i^2 Y_j^2].$$

Par indépendance,

$$\mathbb{E}[Y_i^2 Y_j^2] = \mathbb{E}[Y_1^2]\,\mathbb{E}[Y_1^2] = \Big(\mathbb{E}[Y_1^2]\Big)^2,$$

et comme $\sum_{1\le i<j\le n} 1 = \frac{n(n-1)}{2}$, il vient

$$\mathbb{E}\left[\left(\sum_{k=1}^{n} Y_k\right)^4\right] = n\,\mathbb{E}[Y_1^4] + 3n(n-1)\Big(\mathbb{E}[Y_1^2]\Big)^2.$$

Ainsi,

$$\mathbb{E}[S_n^4] = \frac{1}{n^4}\left(n\,\mathbb{E}[Y_1^4] + 3n(n-1)\Big(\mathbb{E}[Y_1^2]\Big)^2\right).$$

Majorisation en $1/n^2$. Par l'inégalité de Jensen appliquée à la fonction convexe $x \mapsto x^2$, on a

$$\Big(\mathbb{E}[Y_1^2]\Big)^2 \le \mathbb{E}[Y_1^4].$$

Donc

$$\mathbb{E}[S_n^4] \le \frac{1}{n^4}\Big(n\,\mathbb{E}[Y_1^4] + 3n(n-1)\mathbb{E}[Y_1^4]\Big) = \frac{1}{n^4}\Big((3n^2-2n)\mathbb{E}[Y_1^4]\Big) \le \frac{3\,\mathbb{E}[Y_1^4]}{n^2}.$$

En particulier,

$$\sum_{n=1}^{\infty} \mathbb{E}[S_n^4] < +\infty.$$

Contrôle des grandes déviations (Markov) et Borel–Cantelli. Soit $\varepsilon > 0$. Par l'inégalité de Markov,

$$\mathbb{P}(|S_n| > \varepsilon) = \mathbb{P}\Big(|S_n|^4 > \varepsilon^4\Big) \le \frac{\mathbb{E}[S_n^4]}{\varepsilon^4} \le \frac{3\,\mathbb{E}[Y_1^4]}{\varepsilon^4}\cdot\frac{1}{n^2}.$$

La série $\sum_{n\ge1} \mathbb{P}(|S_n| > \varepsilon)$ converge donc, puisque $\sum_{n\ge1} 1/n^2 < \infty$. Par le lemme de Borel–Cantelli,

$$\mathbb{P}\Big(|S_n| > \varepsilon \text{ i.o.}\Big) = 0,$$

c'est-à-dire : presque sûrement, il existe $N(\omega)$ tel que pour tout $n \ge N(\omega)$, $|S_n(\omega)| \le \varepsilon$.

Conclusion. Comme ceci est vrai pour tout $\varepsilon > 0$ (par exemple $\varepsilon = 1/m$ et intersection dénombrable), on obtient

$$S_n \xrightarrow[n\to\infty]{a.s.} 0.$$

En revenant à $X_k = Y_k + \mu$, on conclut finalement

$$\frac{1}{n}\sum_{k=1}^{n} X_k \xrightarrow[n\to\infty]{a.s.} \mu = \mathbb{E}[X_1],$$

ce qui démontre la loi des grands nombres sous l'hypothèse $\mathbb{E}[|X_1|^4] < \infty$.

A.2 Preuve du théorème central limite (par fonctions caractéristiques)

Énoncé. Soit $(X_n)_{n\geq 1}$ une suite de variables aléatoires i.i.d. de même loi que X. Si $\mathbb{E}(|X|^3) < \infty$, alors

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}[X]\right) \xrightarrow[n\to\infty]{\mathcal{L}} \mathcal{N}(0,\sigma^2), \qquad \text{où } \sigma^2 = \mathrm{Var}(X).$$

Réduction sans perte de généralité. Sans perte de généralité, on se ramène au cas $\mathbb{E}[X] = 0$ et (en dimension 1) à une variable réelle. Alors

$$\sigma^2 = \mathrm{Var}(X) = \mathbb{E}[X^2].$$

On pose

$$S_n := \frac{1}{n}\sum_{i=1}^{n} X_i, \qquad \text{et l'on étudie } \sqrt{n}S_n.$$

Fonction caractéristique de $\sqrt{n}S_n$. Pour tout $u \in \mathbb{R}$, on note

$$\psi_{\sqrt{n}S_n}(u) := \mathbb{E}\Big[\exp(iu\sqrt{n}S_n)\Big].$$

Par indépendance des (X_i),

$$\psi_{\sqrt{n}S_n}(u) = \mathbb{E}\left[\exp\left(\frac{iu}{\sqrt{n}}\sum_{i=1}^{n} X_i\right)\right] = \prod_{i=1}^{n}\mathbb{E}\left[\exp\left(\frac{iuX_i}{\sqrt{n}}\right)\right] = \left(\mathbb{E}\left[\exp\left(\frac{iuX}{\sqrt{n}}\right)\right]\right)^n.$$

Développement de Taylor et contrôle du reste. On utilise l'inégalité (issue d'un développement de Taylor) valable pour tout $y \in \mathbb{R}$:

$$\left|e^{iy} - 1 - iy + \frac{y^2}{2}\right| \leq \min\left(\frac{|y|^3}{6},\, y^2\right).$$

On applique cette inégalité à $y = \dfrac{uX}{\sqrt{n}}$. Il existe alors une variable aléatoire $h_n(X)$ telle que

$$\exp\left(\frac{iuX}{\sqrt{n}}\right) = 1 + \frac{iuX}{\sqrt{n}} - \frac{u^2X^2}{2n} + h_n(X),$$

avec le contrôle

$$|h_n(X)| \leq \min\left(\frac{|u|^3|X|^3}{6n^{3/2}},\, \frac{u^2X^2}{n}\right) = \frac{u^2}{n}\min\left(\frac{|u||X|^3}{6\sqrt{n}},\, X^2\right).$$

En particulier, on en déduit que la suite $(n\,h_n(X))_{n\geq 1}$ est dominée uniformément par u^2X^2 :

$$|n\,h_n(X)| \leq u^2X^2, \qquad \forall n \geq 1.$$

Passage à l'espérance. En prenant l'espérance dans l'identité précédente et en utilisant $\mathbb{E}[X] = 0$, on obtient

$$\mathbb{E}\left[\exp\left(\frac{iuX}{\sqrt{n}}\right)\right] = 1 - \frac{u^2}{2n}\mathbb{E}[X^2] + \mathbb{E}[h_n(X)] = 1 - \frac{u^2\sigma^2}{2n} + \mathbb{E}[h_n(X)].$$

Convergence de $\mathbb{E}[nh_n(X)]$ (domination). On montre maintenant que

$$\lim_{n\to\infty} \mathbb{E}\Big[n\, h_n(X)\Big] = 0.$$

En effet, d'après le contrôle précédent,

$$|n\, h_n(X)| \leq u^2X^2 \quad \text{et} \quad n\, h_n(X) \xrightarrow[n\to\infty]{} 0 \text{ p.s.},$$

car $\min\Big(\frac{|u||X|^3}{6\sqrt{n}}, X^2\Big) \to 0$ lorsque $n \to \infty$. De plus, $\mathbb{E}[u^2X^2] < \infty$ car $\mathbb{E}[X^2] < \infty$ (puisque $\mathbb{E}|X|^3 < \infty$). Le théorème de convergence dominée donne donc

$$\mathbb{E}\Big[n\, h_n(X)\Big] \xrightarrow[n\to\infty]{} 0,$$

c'est-à-dire $\mathbb{E}[h_n(X)] = o\Big(\frac{1}{n}\Big)$.

Limite de la fonction caractéristique. On obtient ainsi

$$\mathbb{E}\left[\exp\left(\frac{iuX}{\sqrt{n}}\right)\right] = 1 - \frac{u^2\sigma^2}{2n} + o\Big(\frac{1}{n}\Big)\,,$$

et donc

$$\psi_{\sqrt{n}S_n}(u) = \left(1 - \frac{u^2\sigma^2}{2n} + o\Big(\frac{1}{n}\Big)\right)^n.$$

En utilisant le fait que si $a_n = \frac{x}{n} + o(1/n)$ alors $(1 + a_n)^n \to e^x$, on déduit

$$\lim_{n\to\infty} \psi_{\sqrt{n}S_n}(u) = \exp\left(-\frac{u^2\sigma^2}{2}\right).$$

Or, si $G \sim \mathcal{N}(0,1)$, alors

$$\mathbb{E}\Big[e^{iu\sigma G}\Big] = \exp\left(-\frac{u^2\sigma^2}{2}\right).$$

Ainsi, pour tout $u \in \mathbb{R}$,

$$\psi_{\sqrt{n}S_n}(u) \xrightarrow[n\to\infty]{} \mathbb{E}\Big[e^{iu\sigma G}\Big],$$

ce qui implique, par le théorème de Lévy, que

$$\sqrt{n}\, S_n \xrightarrow[n\to\infty]{\mathcal{L}} \sigma G \sim \mathcal{N}(0, \sigma^2).$$

Retour au cas général. Si $\mathbb{E}[X] = \mu$, on applique le résultat à $X - \mu$ et l'on obtient finalement

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right) \xrightarrow[n\to\infty]{\mathcal{L}} \mathcal{N}(0, \sigma^2), \qquad \sigma^2 = \mathrm{Var}(X).$$

A.3 Preuve du théorème de réduction de variance pour le contrôle antithétique

On pose $Y = g(S_T)$ et $Y^- = g(S_T^-)$. On a

$$\mathrm{Var}\left(\frac{Y + Y^-}{2}\right) = \frac{1}{4}\Big(\mathrm{Var}(Y) + \mathrm{Var}(Y^-) + 2\,\mathrm{Cov}(Y, Y^-)\Big).$$

Comme Y et Y^- ont même loi, on a $\mathrm{Var}(Y) = \mathrm{Var}(Y^-)$, d'où

$$\mathrm{Var}\left(\frac{Y + Y^-}{2}\right) = \frac{1}{2}\mathrm{Var}(Y) + \frac{1}{2}\mathrm{Cov}(Y, Y^-).$$

Sous l'hypothèse $\mathrm{Cov}(Y, Y^-) \leq 0$, il vient

$$\mathrm{Var}\left(\frac{Y + Y^-}{2}\right) \leq \frac{1}{2}\mathrm{Var}(Y),$$

ce qui établit le gain de variance.

A.4 Preuve de la proposition 1 du pont brownien

Soient $0 \leq t_1 < \cdots < t_k < T$. Le vecteur aléatoire $(Z_{t_1}, \ldots, Z_{t_k}, W_T)$ est gaussien centré en tant qu'image linéaire d'un vecteur gaussien centré. En particulier, le processus $(Z_t)_{t\in[0,T]}$ est un processus gaussien centré.

De plus, pour tout $t_i \in [0, T]$, on a

$$\mathrm{Cov}(Z_{t_i}, W_T) = \mathbb{E}[W_{t_i} W_T] - \frac{t_i}{T}\mathbb{E}[W_T^2] = t_i - \frac{t_i}{T}T = 0.$$

Il en résulte que le vecteur $(Z_{t_1}, \ldots, Z_{t_k})$ est indépendant de W_T. Par un argument de classe monotone, on obtient l'indépendance du processus $(Z_t)_{t\in[0,T]}$ et de la variable aléatoire W_T.

Par ailleurs, le processus Z étant centré, pour $s \leq t$,

$$\begin{aligned}\mathbb{E}[Z_s Z_t] &= \mathbb{E}[W_s W_t] - \frac{t}{T}\mathbb{E}[W_s W_T] - \frac{s}{T}\mathbb{E}[W_t W_T] + \frac{st}{T^2}\mathbb{E}[W_T^2] \\ &= s - \frac{st}{T}.\end{aligned}$$

En symétrisant, on obtient finalement la covariance d'un pont brownien sur $[0, T]$.

$$\mathbb{E}[Z_s Z_t] = s \wedge t - \frac{st}{T},$$

A.5 Preuve de la proposition 3 du pont brownien

D'après les propositions précédentes de la section pont brownien, le processus $(Z_t)_{t\in[0,h]}$ a même loi que $(W_t \mid W_h = y)$. Soit $\tau_a := \inf\{t \geq 0 : W_t = a\}$, le temps d'atteinte du niveau a par le mouvement brownien W.

On a alors

$$\mathbb{P}\left(\max_{t\in[0,h]} W_t \geq a,\ W_h \leq y\right) = \mathbb{P}(\tau_a \leq h,\ W_h \leq y) = \mathbb{P}(\tau_a \leq h,\ W_h - W_{\tau_a} \leq y - a).$$

Comme τ_a est $\mathcal{F}^W_{\tau_a}$-mesurable et que $W_h - W_{\tau_a}$ est indépendant de $\mathcal{F}^W_{\tau_a}$ par la propriété de Markov forte du mouvement brownien, on obtient

$$\begin{aligned}\mathbb{P}\left(\max_{t\in[0,h]} W_t \geq a,\ W_h \leq y\right) &= \mathbb{P}(\tau_a \leq h,\ W_h - W_{\tau_a} \geq a - y)\\ &= \mathbb{P}(\tau_a \leq h,\ W_h \geq 2a - y),\end{aligned}$$

car $W_h - W_{\tau_a}$ et $W_{\tau_a} - W_h$ ont même loi par symétrie.

Comme $2a - y \geq a$, on en déduit

$$\mathbb{P}\left(\max_{t\in[0,h]} W_t \geq a,\ W_h \leq y\right) = \mathbb{P}(W_h \geq 2a - y).$$

Finalement, en utilisant la formule de conditionnement,

$$\mathbb{P}\left(\max_{t\in[0,h]} W_t \leq a \mid W_h = y\right) = 1 - \frac{\frac{\partial}{\partial y}\mathbb{P}\left(\max_{t\in[0,h]} W_t \geq a,\ W_h \leq y\right)}{\frac{\partial}{\partial y}\mathbb{P}(W_h \leq y)},$$

et un calcul direct permet d'obtenir le résultat annoncé.

Bibliographie

[1] M. J. Capinski et T. Zastawniak, *Numerical Methods in Finance with C++*, Cambridge University Press, 2003.

[2] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, Springer, volume 53, 2003.

[3] M. Broadie, P. Glasserman et S. G. Kou, *A continuity correction for discrete barrier options*, Mathematical Finance, volume 7, numéro 4, pages 325–349, 1997.

[4] M. Jeanblanc, *Cours de Mathématiques Financières*, Université d'Évry / Université Paris-Saclay, support de cours.

[5] V. Torri, *Cours de Programmation C++*, Université d'Évry, support de cours.

[6] E. Chevalier, *Cours de Finance Mathématique*, Université d'Évry–Courcouronnes, support de cours.

Glossaire

Mouvement brownien Processus stochastique $(B_t)_{t\geq 0}$ servant de modèle d'aléa en finance. Il est défini par les propriétés suivantes :

- $B_0 = 0$ presque sûrement,
- les accroissements sont indépendants : $B_t - B_s$ est indépendant de $\mathcal{F}_s$ pour $t > s$,
- les accroissements sont stationnaires et suivent une loi normale : $B_t - B_s \sim \mathcal{N}(0, t-s)$,
- les trajectoires $t \mapsto B_t$ sont continues presque sûrement.

.

méthode de Monte-Carlo Méthode numérique fondée sur la simulation de trajectoires aléatoires permettant d'approximer des espérances. Elle est particulièrement privilégiée pour la valorisation de produits dérivés complexes, notamment lorsque le payoff dépend de l'ensemble de la trajectoire du sous-jacent (produits path-dependent), lorsque la dimension du problème est élevée, ou lorsqu'aucune formule analytique explicite n'est disponible, ce qui est fréquemment le cas en finance sous la mesure risque-neutre.

option lookback Option path-dependent dont le payoff dépend du maximum ou du minimum atteint par le prix du sous-jacent sur la période de vie du contrat. Un call lookback paie la différence entre le prix final et le minimum observé, tandis qu'un put lookback dépend du maximum observé.

path-dependent Qualifie un produit dérivé dont le payoff dépend de l'ensemble de la trajectoire du sous-jacent sur un intervalle de temps donné, et non uniquement de sa valeur finale à maturité.

Produits dérivés Instruments financiers dont la valeur dépend (ou « dérive ») de l'évolution d'un actif sous-jacent. Ce sous-jacent peut être de nature diverse : actions, indices boursiers, taux d'intérêt, devises, matières premières, etc. Les principaux produits dérivés sont les *options*, les *futures*, les *forwards* et les *swaps*. Ils sont utilisés à la fois pour la couverture contre le risque, la spéculation et l'arbitrage..