Quantitative Finance

Lecture Notes

Master M2 — 2025–2026

Yaë Ulrich Gaba

"In finance, the point is not to predict the future but to prepare for it."
— Pierre-Simon Laplace

March 25, 2026

Contents

Preface **1**

1 Financial Markets — Modelling **5**

1.1 Introduction to Financial Markets . . . 5

1.1.1 Asset Classes . . . 5

1.1.2 Fundamental Derivatives . . . 6

1.2 Discrete-Time Modelling . . . 6

1.2.1 One-Period Model . . . 6

1.2.2 Risk-Neutral Probability . . . 7

1.3 Multi-Period Binomial Model (CRR) . . . 7

1.3.1 CRR Parameter Calibration . . . 8

1.4 Fundamental Theorems of Asset Pricing . . . 8

1.5 Replication and Hedging . . . 8

1.6 Python Implementation . . . 8

1.7 Put-Call Parity . . . 9

1.8 Market Hypotheses . . . 10

1.9 Stylised Facts of Financial Returns . . . 10

1.10 Exercises . . . 11

2 The Black–Scholes Model **13**

2.1 Introduction . . . 13

2.2 Geometric Brownian Motion . . . 13

2.3 Black–Scholes Model Assumptions . . . 14

2.4 The Black–Scholes Equation . . . 14

2.5 Black–Scholes Formulae . . . 14

2.6 The Greeks . . . 15

2.7 Implied Volatility . . . 16

2.8 Python Implementation . . . 16

2.9 Model Extensions . . . 17

2.9.1 Continuous Dividends . . . 17

2.9.2 Options on Futures (Black's Formula) . . . 18

2.10 Model Limitations . . . 18

2.11 Exercises . . . 18

3 Option Pricing **19**

3.1 Introduction . . . 19

3.2 Binomial Trees . . . 19

3.2.1 CRR Algorithm . . . 19

3.2.2 Trinomial Tree . . . 20

3.3 Monte Carlo Methods . . . 20
3.3.1 General Principle . . . 20
3.3.2 GBM Simulation . . . 20
3.3.3 Variance Reduction . . . 21
3.4 Finite Differences . . . 22
3.4.1 Transformation and Discretisation . . . 22
3.5 Method Comparison . . . 24
3.6 Longstaff–Schwartz Method (American Options) . . . 24
3.7 Exercises . . . 25

4 Delta Hedging and Risk Management **27**
4.1 Introduction . . . 27
4.2 Delta Hedging . . . 27
4.2.1 Principle . . . 27
4.2.2 Discrete-Time Hedging Error . . . 28
4.3 Gamma Hedging . . . 29
4.4 Vega Hedging . . . 29
4.5 P&L Decomposition . . . 30
4.6 Transaction Costs and Optimal Hedging . . . 30
4.7 Hedging in Incomplete Markets . . . 31
4.8 Exercises . . . 31

5 Stochastic Calculus in Finance **33**
5.1 Brownian Motion . . . 33
5.2 The Itô Integral . . . 34
5.2.1 Construction . . . 34
5.2.2 Calculus Rules . . . 35
5.3 Itô's Formula . . . 35
5.4 Girsanov's Theorem . . . 36
5.5 Martingale Representation Theorem . . . 36
5.6 Stochastic Differential Equations . . . 36
5.7 Stochastic Exponential . . . 37
5.8 Implementation: Discretisation Schemes . . . 37
5.9 Exercises . . . 38

6 Stochastic Volatility Models **39**
6.1 Motivation: The Volatility Smile . . . 39
6.2 Local Volatility (Dupire) . . . 39
6.3 The Heston Model . . . 40
6.3.1 Pricing via Characteristic Function . . . 40
6.4 The SABR Model . . . 41
6.5 Calibration . . . 42
6.6 Local vs Stochastic Volatility . . . 43
6.7 Stochastic Local Volatility (SLV) Models . . . 43
6.8 Exercises . . . 43

7 Interest Rate Models 45
7.1 Term Structure of Interest Rates . . . 45
7.2 Short-Rate Models . . . 46
7.2.1 Vasicek Model (1977) . . . 46
7.2.2 Cox–Ingersoll–Ross Model (CIR, 1985) . . . 46
7.2.3 Hull–White Model (1990) . . . 46
7.3 Heath–Jarrow–Morton (HJM) Framework . . . 47
7.4 Interest Rate Product Pricing . . . 48
7.4.1 Caps and Floors . . . 48
7.4.2 Swaptions . . . 48
7.5 LIBOR Market Model (BGM/LMM) . . . 49
7.6 Exercises . . . 49

8 Portfolio Optimisation 51
8.1 Markowitz Theory . . . 51
8.1.1 Mean-Variance Framework . . . 51
8.1.2 Optimisation Problem . . . 51
8.2 The CAPM . . . 53
8.3 Black–Litterman Model . . . 54
8.4 Practical Constraints . . . 54
8.5 Robust Optimisation . . . 55
8.6 Parameter Estimation . . . 55
8.7 Exercises . . . 55

9 Risk Measures 57
9.1 Introduction . . . 57
9.2 Value at Risk (VaR) . . . 57
9.3 Expected Shortfall (CVaR) . . . 58
9.4 Coherent Risk Measures . . . 58
9.5 Stress Testing . . . 59
9.6 Copulas and Dependence Modelling . . . 59
9.7 Exercises . . . 60

10 Exotic Derivatives 63
10.1 Introduction . . . 63
10.2 Barrier Options . . . 63
10.3 Asian Options . . . 64
10.4 Lookback Options . . . 64
10.5 Binary (Digital) Options . . . 65
10.6 Monte Carlo Pricing . . . 65
10.7 Tree Methods for Exotics . . . 65
10.8 Exercises . . . 66

11 Machine Learning in Finance 67
11.1 Introduction . . . 67
11.2 Algorithmic Trading . . . 67
11.3 Neural Networks for Option Pricing . . . 68
11.4 Reinforcement Learning for Portfolio Management . . . 68
11.5 NLP and Sentiment Analysis . . . 69

11.6 Risks and Regulation 69
11.7 Exercises 70

Preface

Course Objectives

Quantitative finance lies at the intersection of mathematics, statistics, and financial economics. This Master-level course aims to provide students with a rigorous training covering the theoretical foundations and numerical methods essential for understanding and practising modern financial engineering.

Contemporary financial markets are characterised by increasing complexity: exotic derivatives, high-frequency algorithmic trading, quantitative portfolio management, and more recently the emergence of machine learning in financial modelling. This course equips students with the mathematical and computational tools necessary to navigate this environment.

The approach adopted is resolutely **mathematical and computational**: each theoretical concept is accompanied by a Python implementation that allows verification of analytical results and develops numerical intuition.

Prerequisites

This course assumes that students possess solid knowledge in the following areas:

- **Mathematical analysis**: differential and integral calculus, series, function spaces, ordinary differential equations.
- **Linear algebra**: vector spaces, matrices, eigenvalues, matrix decompositions (Cholesky, SVD).
- **Probability theory**: probability spaces, random variables, classical distributions, conditional expectation, limit theorems (LLN, CLT).
- **Statistics**: parametric estimation, hypothesis testing, linear regression, maximum likelihood estimation.
- **Programming**: proficiency in Python (or an equivalent scientific language), basic object-oriented programming.

Course Organisation

The course is organised into eleven chapters, each designed to be covered in one to two weeks of lectures accompanied by tutorials and computer laboratory sessions.

Chapter 1 — Financial Markets and Modelling. Introduction to markets, financial assets, modelling hypotheses, no-arbitrage principle, discrete-time models (Cox–Ross–Rubinstein).

Chapter 2 — The Black–Scholes Model. Model derivation, the Black–Scholes PDE, closed-form formulae for European options, computation and interpretation of the Greeks.

Chapter 3 — Option Pricing Methods. Binomial and trinomial trees, Monte Carlo methods, finite differences, variance reduction techniques.

Chapter 4 — Hedging and Risk Management. Delta hedging, dynamic hedging, gamma hedging, hedging costs, practical limitations of the Black–Scholes model.

Chapter 5 — Stochastic Calculus in Finance. Brownian motion, Itô integral, Itô's formula, Girsanov's theorem, martingale representation theorem.

Chapter 6 — Stochastic Volatility. Volatility smile and surface, the Heston model, the SABR model, calibration, local vs stochastic volatility.

Chapter 7 — Interest Rate Models. Term structure, Vasicek, CIR, Hull–White models, HJM framework, bond pricing and caps/floors.

Chapter 8 — Portfolio Optimisation. Markowitz theory, efficient frontier, CAPM, Black–Litterman model, robust optimisation.

Chapter 9 — Risk Measures. Value-at-Risk (VaR), Expected Shortfall (CVaR), coherence properties, computation methods, backtesting, stress testing.

Chapter 10 — Exotic Derivatives. Barrier options, Asian options, lookback options, basket options, analytical and numerical pricing.

Chapter 11 — Machine Learning in Finance. Neural networks for pricing, reinforcement learning for trading, anomaly detection, overfitting risks.

Pedagogical Methodology

Each chapter follows a consistent structure:

1. **Theory**: rigorous presentation of mathematical concepts with complete or sketched proofs.

2. **Financial interpretation**: discussion of the economic significance of results and their limitations.

3. **Implementation**: Python code illustrating numerical methods using the `numpy`, `scipy`, and `matplotlib` libraries.

4. **Exercises**: theoretical and numerical problems of increasing difficulty to deepen understanding.

Notation Conventions

Throughout this course, we use the following notation:

Notation	**Meaning**
$(\Omega, \mathcal{F}, \mathbb{P})$	Probability space
$\mathbb{E}[X]$	Expectation of the random variable X
$\mathbb{E}^{\mathbb{Q}}[\cdot]$	Expectation under the risk-neutral measure
$\text{Var}(X)$	Variance of X
$\text{Cov}(X, Y)$	Covariance of X and Y
$\mathbb{R}, \mathbb{R}^+$	Real numbers, positive reals
$\mathbb{N}$	Natural numbers
W_t or B_t	Standard Brownian motion
$(\mathcal{F}_t)_{t\geq 0}$	Natural filtration
$\mathbb{Q}$	Risk-neutral probability measure
r	Risk-free interest rate (continuous)
σ	Volatility of the underlying asset
S_t	Price of the underlying asset at time t
$C(S,t), P(S,t)$	European call and put prices
K	Strike price
T	Option maturity
$\Delta, \Gamma, \Theta, \mathcal{V}, \rho$	Greek letters (sensitivities)

Software and Tools

The numerical implementations in this course use the scientific Python ecosystem:

- `numpy`: matrix computation, linear algebra, random number generation.
- `scipy`: optimisation (`scipy.optimize`), integration, special functions, normal distribution (`scipy.stats.norm`).
- `matplotlib`: graphical visualisation, volatility surface plots, efficient frontiers.
- `pandas`: manipulation and analysis of financial time series.
- `scikit-learn`: classical machine learning algorithms (Chapter 11).
- `tensorflow` / `pytorch`: deep neural networks (Chapter 11).

Cross-Cutting Themes

Several themes run through the entire course and deserve to be highlighted from this preface:

- **No-arbitrage**: the fundamental principle underpinning all pricing theory in quantitative finance.
- **Market completeness**: whether every contingent claim can be replicated by a self-financing trading strategy.

- **Model risk**: the consequences of using simplified mathematical models for complex financial realities.
- **Calibration vs estimation**: the distinction between calibrating a model to market prices and estimating its parameters from historical data.

References

This course draws upon the following reference texts:

1. Shreve, S.E. — *Stochastic Calculus for Finance I & II*, Springer, 2004.
2. Hull, J.C. — *Options, Futures, and Other Derivatives*, Pearson, 2017 (10th edition).
3. Björk, T. — *Arbitrage Theory in Continuous Time*, Oxford University Press, 2009 (3rd edition).
4. Glasserman, P. — *Monte Carlo Methods in Financial Engineering*, Springer, 2003.
5. Gatheral, J. — *The Volatility Surface: A Practitioner's Guide*, Wiley, 2006.
6. Brigo, D. & Mercurio, F. — *Interest Rate Models — Theory and Practice*, Springer, 2006.
7. McNeil, A.J., Frey, R. & Embrechts, P. — *Quantitative Risk Management: Concepts, Techniques and Tools*, Princeton University Press, 2015.
8. López de Prado, M. — *Advances in Financial Machine Learning*, Wiley, 2018.
9. Cont, R. & Tankov, P. — *Financial Modelling with Jump Processes*, Chapman & Hall/CRC, 2004.

Acknowledgements

This course has benefited from the contributions of numerous colleagues in mathematical finance. We are particularly grateful to students from previous cohorts whose questions and pertinent remarks have considerably improved the presentation of this material.

The Author
March 2026

Chapter 1

Financial Markets — Modelling

In 1900, Louis Bachelier defended his thesis at the Sorbonne under Henri Poincaré's supervision. His subject: the "theory of speculation," in which he proposed to model stock prices as a stochastic process. Poincaré, though visionary, judged the work "not without interest" but awarded it an honorable rather than a very honorable mention. It took sixty years, and the arrival of Samuelson, Black, Scholes, and Merton, for Bachelier's ideas to become the cornerstone of modern finance. Today, financial markets move trillions each day, and the mathematical models that describe them lie at the heart of the global economy.

1.1 Introduction to Financial Markets

Financial markets are venues for the exchange of assets whose value is uncertain. Understanding their functioning and modelling them mathematically is the starting point of all quantitative finance.

Definition 1.1 (Financial Market)**.** A **financial market** is an organised system enabling the exchange of financial assets between economic agents. We distinguish:

- The **primary market**: issuance of new securities.
- The **secondary market**: trading of previously issued securities.

1.1.1 Asset Classes

Definition 1.2 (Financial Asset)**.** A **financial asset** is a contract entitling its holder to future cash flows. The main classes are:

1. **Equities**: ownership shares in a company.
2. **Bonds**: fixed-income debt securities.
3. **Derivatives**: contracts whose value depends on an underlying asset.
4. **Currencies**: foreign exchange (forex) market.
5. **Commodities**: gold, oil, wheat, etc.

CHAPTER 1. FINANCIAL MARKETS — MODELLING

1.1.2 Fundamental Derivatives

Definition 1.3 (European Option)**.** A **European call option** with strike price K and maturity T on an underlying S gives its holder the right (but not the obligation) to buy S at price K at time T. Its payoff is:

$$(S_T - K)^+ = \max(S_T - K, 0).$$

Similarly, a **European put option** has payoff:

$$(K - S_T)^+ = \max(K - S_T, 0).$$

Definition 1.4 (Forward Contract)**.** A **forward contract** is an agreement to buy or sell an asset at a future date T at a price F fixed today. The buyer's payoff is $S_T - F$.

1.2 Discrete-Time Modelling

1.2.1 One-Period Model

Consider a market with two dates $t = 0$ and $t = 1$, with:

- A risk-free asset with price $B_0 = 1$, $B_1 = 1 + r$.
- A risky asset with price S_0 at $t = 0$, which can take values $S_1^u = uS_0$ (up) or $S_1^d = dS_0$ (down) with $0 < d < 1 + r < u$.

Definition 1.5 (Trading Strategy)**.** A **trading strategy** is a pair $(\phi^0, \phi^1) \in \mathbb{R}^2$ where ϕ^0 represents the quantity of risk-free asset and ϕ^1 the quantity of risky asset held. The portfolio value is:

$$V_t = \phi^0 B_t + \phi^1 S_t, \quad t \in \{0, 1\}.$$

Definition 1.6 (Arbitrage Opportunity)**.** An **arbitrage opportunity** is a strategy (ϕ^0, ϕ^1) such that:

$$V_0 \leq 0, \quad V_1 \geq 0 \text{ a.s.}, \quad \mathbb{P}(V_1 > 0) > 0.$$

That is, a possible gain without risk of loss and without initial investment.

Theorem 1.7 (No-Arbitrage — One-Period Binomial Model)**.** *The one-period binomial model is arbitrage-free if and only if* $d < 1 + r < u$.

Proof. Suppose there exists an arbitrage (ϕ^0, ϕ^1) with $V_0 = 0$. Then:

$$\phi^0 + \phi^1 S_0 = 0 \implies \phi^0 = -\phi^1 S_0.$$

The terminal values are:

$$V_1^u = \phi^1(uS_0 - (1 + r)S_0) = \phi^1 S_0(u - 1 - r),$$
$$V_1^d = \phi^1(dS_0 - (1 + r)S_0) = \phi^1 S_0(d - 1 - r).$$

For both $V_1^u \geq 0$ and $V_1^d \geq 0$ with at least one strict inequality, $u - 1 - r$ and $d - 1 - r$ must have the same non-zero sign, contradicting $d < 1 + r < u$. □

1.2.2 Risk-Neutral Probability

Theorem 1.8 (Risk-Neutral Measure)**.** *Under the no-arbitrage condition ($d < 1+r < u$), there exists a unique probability measure $\mathbb{Q}$ on $\{\omega_u, \omega_d\}$ such that the discounted risky asset price is a $\mathbb{Q}$-martingale:*

$$S_0 = \frac{1}{1+r}\mathbb{E}^{\mathbb{Q}}[S_1].$$

The risk-neutral probability is given by:

$$q = \frac{(1+r)-d}{u-d}, \quad 1-q = \frac{u-(1+r)}{u-d}.$$

Risk-Neutral Pricing of a Contingent Claim

The price at $t=0$ of a contingent claim with payoff $H = h(S_1)$ is:

$$V_0 = \frac{1}{1+r}\mathbb{E}^{\mathbb{Q}}[H] = \frac{1}{1+r}\big[q \cdot h(uS_0) + (1-q)\cdot h(dS_0)\big].$$

Example 1.9. Let $S_0 = 100$, $u = 1.2$, $d = 0.9$, $r = 0.05$, $K = 100$.

The risk-neutral probability is:

$$q = \frac{1.05-0.9}{1.2-0.9} = \frac{0.15}{0.30} = 0.5.$$

The call price is:

$$C_0 = \frac{1}{1.05}\big[0.5 \times (120-100)^+ + 0.5 \times (90-100)^+\big] = \frac{0.5 \times 20}{1.05} \approx 9.52.$$

1.3 Multi-Period Binomial Model (CRR)

Definition 1.10 (Cox–Ross–Rubinstein Model)**.** The **CRR model** with N periods is defined by:

- $S_{n+1} = S_n \cdot Y_{n+1}$ where $Y_{n+1} \in \{u, d\}$ are i.i.d. under $\mathbb{Q}$.
- The discounted price $\tilde{S}_n = S_n/(1+r)^n$ is a $\mathbb{Q}$-martingale.
- After n periods: $S_n = S_0 u^j d^{n-j}$ for j up-moves out of n.

Theorem 1.11 (CRR Pricing Formula)**.** *The price of a European call in the CRR model is:*

$$C_0 = \frac{1}{(1+r)^N}\sum_{j=0}^{N}\binom{N}{j} q^j (1-q)^{N-j}\big(S_0 u^j d^{N-j} - K\big)^+.$$

Convergence to Black–Scholes

As $N \to \infty$ with the parametrisation $u = e^{\sigma\sqrt{\Delta t}}$, $d = e^{-\sigma\sqrt{\Delta t}}$, $\Delta t = T/N$, the CRR formula converges to the Black–Scholes formula (Chapter 2). This is a consequence of the Central Limit Theorem.

CHAPTER 1. FINANCIAL MARKETS — MODELLING

1.3.1 CRR Parameter Calibration

For the discrete model to approximate the continuous model with volatility σ and rate r, we set:

$$u = e^{\sigma\sqrt{\Delta t}}, \quad d = e^{-\sigma\sqrt{\Delta t}} = 1/u, \tag{1.1}$$

$$q = \frac{e^{r\Delta t} - d}{u - d}. \tag{1.2}$$

1.4 Fundamental Theorems of Asset Pricing

Theorem 1.12 (First Fundamental Theorem of Asset Pricing (FTAP))**.** *A market is arbitrage-free if and only if there exists a probability measure $\mathbb{Q}$ equivalent to $\mathbb{P}$ under which discounted asset prices are martingales.*

Theorem 1.13 (Second Fundamental Theorem)**.** *An arbitrage-free market is **complete** (every contingent claim is replicable) if and only if the martingale measure $\mathbb{Q}$ is unique.*

Remark 1.14. In the binomial model, the market is complete because there are two states of nature and two assets. In contrast, in a trinomial model (three states), the market is incomplete and infinitely many martingale measures exist.

1.5 Replication and Hedging

Proposition 1.15 (Replicating Portfolio — One Period)**.** In the one-period binomial model, the contingent claim $H = h(S_1)$ is replicated by (ϕ^0, ϕ^1) with:

$$\phi^1 = \frac{h(uS_0) - h(dS_0)}{(u-d)S_0}, \quad \phi^0 = \frac{1}{1+r}\big[h(uS_0) - \phi^1 uS_0\big].$$

The quantity ϕ^1 is called the **delta** of the option.

1.6 Python Implementation

CRR Binomial Model in Python

```python
import numpy as np

def crr_european(S0, K, T, r, sigma, N, option='call'):
    """Price a European option using the CRR binomial model."""
    dt = T / N
    u = np.exp(sigma * np.sqrt(dt))
    d = 1 / u
    q = (np.exp(r * dt) - d) / (u - d)
    discount = np.exp(-r * dt)

    # Terminal prices
    j = np.arange(N + 1)
    ST = S0 * u**j * d**(N - j)
    if option == 'call':
        payoff = np.maximum(ST - K, 0)
    else:
        payoff = np.maximum(K - ST, 0)

    # Backward induction
    for n in range(N - 1, -1, -1):
        payoff = discount * (q * payoff[1:] + (1 - q) * payoff[:-1])

    return payoff[0]

# Example
S0, K, T, r, sigma = 100, 100, 1.0, 0.05, 0.20
for N in [10, 50, 100, 500, 1000]:
    price = crr_european(S0, K, T, r, sigma, N)
    print(f"N={N:4d}: C = {price:.4f}")
```

Put-Call Parity — Verification

```
def verify_put_call_parity(S0, K, T, r, sigma, N):
    """Verify put-call parity: C - P = S0 - K*exp(-rT)."""
    C = crr_european(S0, K, T, r, sigma, N, 'call')
    P = crr_european(S0, K, T, r, sigma, N, 'put')
    parity = S0 - K * np.exp(-r * T)
    print(f"C - P = {C - P:.6f}")
    print(f"S0 - K*exp(-rT) = {parity:.6f}")
    print(f"Error = {abs(C - P - parity):.2e}")

verify_put_call_parity(100, 100, 1.0, 0.05, 0.20, 500)
```

1.7 Put-Call Parity

Theorem 1.16 (Put-Call Parity). *For European options with the same strike K and maturity T:*

$$C - P = S_0 - Ke^{-rT}.$$

Proof. Consider the portfolio $\Pi = C - P - S_0 + Ke^{-rT}$. At maturity:

$$\Pi_T = (S_T - K)^+ - (K - S_T)^+ - S_T + K = S_T - K - S_T + K = 0.$$

By no-arbitrage, $\Pi_0 = 0$, yielding the result. □

1.8 Market Hypotheses

Simplifying Assumptions

Classical models rely on strong assumptions:

1. Frictionless market (no transaction costs).
2. Continuous trading is possible.
3. Short selling is allowed without restrictions.
4. Borrowing rate equals lending rate.
5. No counterparty risk.

Relaxing these assumptions leads to more realistic but significantly more complex models.

1.9 Stylised Facts of Financial Returns

Log-returns $r_t = \ln(S_t/S_{t-1})$ exhibit well-documented empirical properties:

1. **Heavy tails**: the return distribution has heavier tails than the normal distribution (excess kurtosis).
2. **Asymmetry**: slight negative skewness.
3. **Aggregational Gaussianity**: returns over longer horizons approach normality.
4. **Absence of autocorrelation**: returns are nearly uncorrelated.
5. **Volatility clustering**: periods of high volatility are clustered together.
6. **Leverage effect**: negative correlation between returns and volatility.

Stylised Facts Analysis

```python
import numpy as np
from scipy import stats

def stylized_facts(returns):
    """Analyse stylised facts of returns."""
    print(f"Mean:       {np.mean(returns):.6f}")
    print(f"Std dev:    {np.std(returns):.6f}")
    print(f"Skewness:   {stats.skew(returns):.4f}")
    print(f"Kurtosis:   {stats.kurtosis(returns):.4f}")

    # Normality test (Jarque-Bera)
    jb_stat, jb_pval = stats.jarque_bera(returns)
    print(f"Jarque-Bera: stat={jb_stat:.2f}, p={jb_pval:.4e}")
    # Autocorrelation of returns and squared returns
    n = len(returns)
    r_centered = returns - np.mean(returns)
    acf1 = np.correlate(r_centered[:-1], r_centered[1:]) / (n *
    ↪  np.var(returns))
    r2 = r_centered**2
    acf1_sq = np.correlate(r2[:-1], r2[1:]) / (n * np.var(r2))
    print(f"ACF(1) returns:  {acf1[0]:.4f}")
    print(f"ACF(1) squared:  {acf1_sq[0]:.4f}")

# Simulated GBM for comparison
np.random.seed(42)
S0, mu, sigma, dt, N = 100, 0.08, 0.20, 1/252, 2520
Z = np.random.standard_normal(N)
log_returns = (mu - 0.5*sigma**2)*dt + sigma*np.sqrt(dt)*Z
stylized_facts(log_returns)
```

1.10 Exercises

Exercise 1.1 (One-Period Binomial Model)**.** Let $S_0 = 50$, $u = 1.3$, $d = 0.8$, $r = 0.04$.

1. Verify the no-arbitrage condition.
2. Compute the risk-neutral probability q.
3. Compute the price of a call with strike $K = 55$.
4. Determine the replicating portfolio (ϕ^0, ϕ^1).
5. Verify that the put price satisfies put-call parity.

Exercise 1.2 (CRR Convergence)**.** 1. Implement the CRR model and plot the call price as a function of N for $N = 1, 2, \ldots, 200$.

2. Observe the convergence and even/odd oscillations.
3. Compare with the Black–Scholes formula (Chapter 2).

Exercise 1.3 (Incomplete Market)**.** Consider a trinomial model: $S_1 \in \{uS_0, mS_0, dS_0\}$ with $d < m < u$.

1. Show that the market is incomplete.
2. Characterise the set of martingale measures.
3. Deduce the no-arbitrage bounds for the call price.

Exercise 1.4 (American Options in the CRR Model)**.** 1. Modify the CRR algorithm to price an American put by incorporating the possibility of early exercise.

2. Compare American and European put prices. Verify that the American put is always at least as expensive.
3. Determine the optimal exercise boundary.

CHAPTER 1. FINANCIAL MARKETS — MODELLING

Chapter 2

The Black–Scholes Model

2.1 Introduction

On an ordinary spring day in 1973, two papers appeared that would reshape the financial world. Fischer Black and Myron Scholes published their option pricing formula in the *Journal of Political Economy*, while Robert Merton, in a companion paper in the *Bell Journal of Economics*, provided a rigorous derivation using Itô calculus and the principle of no-arbitrage. The result was electrifying: for the first time, there existed an explicit, closed-form formula for the price of a European option—a formula that depended not on investors' risk preferences, but only on observable quantities and one crucial parameter, the volatility.

The impact was immediate. The Chicago Board Options Exchange had opened just weeks earlier, and traders quickly adopted the formula. Scholes and Merton received the 1997 Nobel Prize in Economics for this work (Black having passed away in 1995). But the Black–Scholes model is more than a formula: it is a way of thinking. It introduces the paradigm of *dynamic hedging*—the idea that risk can be eliminated by continuously adjusting a portfolio—and connects option pricing to the theory of stochastic differential equations. Understanding this model is understanding the mathematical engine that drives modern finance.

2.2 Geometric Brownian Motion

Definition 2.1 (Geometric Brownian Motion (GBM))**.** A process $(S_t)_{t\geq 0}$ follows a **geometric Brownian motion** if it satisfies the stochastic differential equation:

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

where $\mu \in \mathbb{R}$ is the expected return (drift), $\sigma > 0$ the volatility, and $(W_t)_{t\geq 0}$ a standard Brownian motion.

Theorem 2.2 (GBM Solution)**.** *The solution to the above SDE is:*

$$S_t = S_0 \exp\left[\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right].$$

In particular, $\ln(S_t/S_0) \sim \mathcal{N}((\mu - \sigma^2/2)t,\ \sigma^2 t)$.

CHAPTER 2. THE BLACK–SCHOLES MODEL

Proof. Set $X_t = \ln S_t$. By Itô's formula (cf. Chapter 5):

$$dX_t = \frac{1}{S_t} dS_t - \frac{1}{2}\frac{1}{S_t^2}(dS_t)^2 = \left(\mu - \frac{\sigma^2}{2}\right) dt + \sigma\, dW_t.$$

Integrating from 0 to t: $X_t = X_0 + (\mu - \sigma^2/2)t + \sigma W_t$, yielding the result. $\square$

2.3 Black–Scholes Model Assumptions

Fundamental Assumptions

1. The underlying follows a GBM with constant volatility σ.
2. The risk-free rate r is constant.
3. No dividends (generalisation possible).
4. Frictionless market (no transaction costs).
5. Continuous trading is possible.
6. Short selling is allowed.

2.4 The Black–Scholes Equation

Theorem 2.3 (Black–Scholes PDE)**.** *Under the model assumptions, the price $V(S,t)$ of a European derivative satisfies the partial differential equation:*

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

with terminal condition $V(S,T) = h(S)$ where h is the payoff function.

Proof sketch via replication. Consider a portfolio $\Pi = V - \Delta S$ where $\Delta = \partial V / \partial S$. By Itô's formula:

$$\begin{aligned} d\Pi &= dV - \Delta\, dS \\ &= \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}\right) dt. \end{aligned}$$

This portfolio is riskless (no dW_t term), so by no-arbitrage: $d\Pi = r\Pi\, dt = r(V - \Delta S)\, dt$. Equating yields the PDE. $\square$

2.5 Black–Scholes Formulae

Theorem 2.4 (Black–Scholes Closed-Form Formulae)**.** *The European call price is:*

$$C(S,t) = S\,\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2),$$

and the European put price:

$$P(S,t) = Ke^{-r(T-t)}\Phi(-d_2) - S\,\Phi(-d_1),$$

where Φ denotes the standard normal CDF and:

$$d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 = d_1 - \sigma\sqrt{T-t}.$$

Black–Scholes Formulae — Summary

$$\begin{aligned} C &= S\Phi(d_1) - Ke^{-r\tau}\Phi(d_2), \\ P &= Ke^{-r\tau}\Phi(-d_2) - S\Phi(-d_1), \\ d_1 &= \frac{\ln(S/K) + (r + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}, \quad d_2 = d_1 - \sigma\sqrt{\tau}, \quad \tau = T - t. \end{aligned}$$

Proof via risk-neutral pricing. Under the risk-neutral measure $\mathbb{Q}$:

$$C = e^{-r\tau}\mathbb{E}^{\mathbb{Q}}[(S_T - K)^+].$$

Now $S_T = S\exp[(r - \sigma^2/2)\tau + \sigma\sqrt{\tau}\,Z]$ with $Z \sim \mathcal{N}(0,1)$ under $\mathbb{Q}$. The payoff is positive when $Z > -d_2$. Computing:

$$\begin{aligned} C &= e^{-r\tau}\int_{-d_2}^{+\infty}\left(Se^{(r-\sigma^2/2)\tau+\sigma\sqrt{\tau}\,z} - K\right)\frac{e^{-z^2/2}}{\sqrt{2\pi}}\,dz \\ &= S\Phi(d_1) - Ke^{-r\tau}\Phi(d_2). \end{aligned}$$

The shift from d_2 to d_1 in the first term results from completing the square in the exponential. □

2.6 The Greeks

The **Greeks** measure the sensitivity of the option price to the various model parameters.

Definition 2.5 (Main Greeks)**.** For a European call:

$$\begin{aligned} \Delta &= \frac{\partial C}{\partial S} = \Phi(d_1), \\ \Gamma &= \frac{\partial^2 C}{\partial S^2} = \frac{\phi(d_1)}{S\sigma\sqrt{\tau}}, \\ \Theta &= \frac{\partial C}{\partial t} = -\frac{S\sigma\phi(d_1)}{2\sqrt{\tau}} - rKe^{-r\tau}\Phi(d_2), \\ \mathcal{V} &= \frac{\partial C}{\partial \sigma} = S\sqrt{\tau}\,\phi(d_1), \\ \rho &= \frac{\partial C}{\partial r} = K\tau e^{-r\tau}\Phi(d_2), \end{aligned}$$

where $\phi(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2}$ is the standard normal density.

Interpreting the Greeks

- Δ: number of underlying units to hedge the option.
- Γ: convexity; measures delta stability.
- Θ: time decay of the option value.
- $\mathcal{V}$ (vega): sensitivity to volatility.
- ρ: sensitivity to the interest rate.

Proposition 2.6 (Fundamental Greeks Relation)**.** The delta, gamma, and theta of a derivative satisfy the Black–Scholes PDE in the form:

$$\Theta + rS\Delta + \frac{1}{2}\sigma^2 S^2\Gamma = rV.$$

2.7 Implied Volatility

Definition 2.7 (Implied Volatility)**.** The **implied volatility** σ_{impl} is the value of σ that, when inserted into the Black–Scholes formula, reproduces the observed market price C_{mkt}:

$$C_{\text{BS}}(S, K, T, r, \sigma_{\text{impl}}) = C_{\text{mkt}}.$$

Remark 2.8. The existence and uniqueness of σ_{impl} follow from the fact that $\mathcal{V} = \partial C/\partial\sigma > 0$ (the call price is strictly increasing in volatility).

2.8 Python Implementation

Black–Scholes Formulae and Greeks

```python
import numpy as np
from scipy.stats import norm

def bs_price(S, K, T, r, sigma, option='call'):
    """Black-Scholes price for a European option."""
    d1 = (np.log(S/K) + (r + 0.5*sigma**2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma*np.sqrt(T)
    if option == 'call':
        return S*norm.cdf(d1) - K*np.exp(-r*T)*norm.cdf(d2)
    else:
        return K*np.exp(-r*T)*norm.cdf(-d2) - S*norm.cdf(-d1)

def bs_greeks(S, K, T, r, sigma):
    """Compute Greeks for a European call."""
    d1 = (np.log(S/K) + (r + 0.5*sigma**2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma*np.sqrt(T)
    delta = norm.cdf(d1)
    gamma = norm.pdf(d1) / (S * sigma * np.sqrt(T))
    theta = (-S*sigma*norm.pdf(d1)/(2*np.sqrt(T))
              - r*K*np.exp(-r*T)*norm.cdf(d2))
    vega  = S * np.sqrt(T) * norm.pdf(d1)
    rho   = K * T * np.exp(-r*T) * norm.cdf(d2)
    return {'delta': delta, 'gamma': gamma, 'theta': theta,
            'vega': vega, 'rho': rho}

# Example
S, K, T, r, sigma = 100, 100, 1.0, 0.05, 0.20
print(f"Call = {bs_price(S,K,T,r,sigma,'call'):.4f}")
print(f"Put  = {bs_price(S,K,T,r,sigma,'put'):.4f}")
greeks = bs_greeks(S, K, T, r, sigma)
for name, val in greeks.items():
    print(f"{name:>6s} = {val:.6f}")
```

Implied Volatility (Newton-Raphson)

```
def implied_vol(market_price, S, K, T, r, option='call',
                tol=1e-8, max_iter=100):
    """Implied volatility via Newton-Raphson."""
    sigma = 0.20  # initial guess
    for _ in range(max_iter):
        price = bs_price(S, K, T, r, sigma, option)
        d1 = (np.log(S/K) + (r + 0.5*sigma**2)*T) / (sigma*np.sqrt(T))
        vega = S * np.sqrt(T) * norm.pdf(d1)
        if vega < 1e-12:
            break
        sigma -= (price - market_price) / vega
        if abs(price - market_price) < tol:
            return sigma
    return sigma

# Example: recover sigma from a price
C_mkt = bs_price(100, 100, 1.0, 0.05, 0.25, 'call')
iv = implied_vol(C_mkt, 100, 100, 1.0, 0.05)
print(f"Recovered implied vol: {iv:.6f}")
```

2.9 Model Extensions

2.9.1 Continuous Dividends

For an underlying paying a continuous dividend yield q, the Black–Scholes formula modifies to:

$$C = Se^{-q\tau}\Phi(d_1) - Ke^{-r\tau}\Phi(d_2),$$

with $d_1 = \frac{\ln(S/K)+(r-q+\sigma^2/2)\tau}{\sigma\sqrt{\tau}}$.

CHAPTER 2. THE BLACK–SCHOLES MODEL

2.9.2 Options on Futures (Black's Formula)

Theorem 2.9 (Black's Formula (1976))**.** *For a European option on a futures contract with price F:*

$$C = e^{-r\tau}\left[F\Phi(d_1) - K\Phi(d_2)\right],$$

with $d_1 = \frac{\ln(F/K)+\sigma^2\tau/2}{\sigma\sqrt{\tau}}$, $d_2 = d_1 - \sigma\sqrt{\tau}$.

2.10 Model Limitations

Known Limitations of Black–Scholes

1. Volatility is not constant in practice (volatility smile, cf. Chapter 6).
2. Returns are not Gaussian (heavy tails).
3. Continuous trading is an idealisation.
4. Transaction costs are not zero.
5. The model underestimates crash risk.

2.11 Exercises

Exercise 2.1 (Verification of the Black–Scholes Formula)**.** 1. Verify by direct substitution that the call formula satisfies the Black–Scholes PDE.

2. Verify the terminal condition $C(S,T) = (S-K)^+$.
3. Derive put-call parity from the formulae.

Exercise 2.2 (Limiting Behaviour)**.** Study the following limits:

1. $C(S,t)$ as $\sigma \to 0^+$.
2. $C(S,t)$ as $\sigma \to +\infty$.
3. $C(S,t)$ as $T-t \to 0^+$ for $S > K$, $S = K$, $S < K$.
4. Δ as $T-t \to 0^+$.

Exercise 2.3 (Numerical Greeks)**.** 1. Compute delta and gamma by finite differences and compare with analytical formulae.

2. Plot Δ, Γ, Θ, $\mathcal{V}$ as functions of S for different values of τ.
3. Verify numerically that $\Theta + rS\Delta + \frac{1}{2}\sigma^2 S^2\Gamma = rC$.

Exercise 2.4 (Implied Volatility Surface)**.** 1. Generate option prices from a Heston model (Chapter 6) for different strikes and maturities.

2. Compute Black–Scholes implied volatility for each price.
3. Plot the implied volatility surface $(K,T) \mapsto \sigma_{\text{impl}}(K,T)$.

Chapter 3

Option Pricing

3.1 Introduction

The Black-Scholes formula is elegant, but it only applies to European options on an underlying following a geometric Brownian motion. What about American options (early exercise), Asian options (depending on the average price), or exotic barrier products? Closed-form formulae no longer exist, and one must turn to numerical computation. Three great families of methods share the field: binomial trees, inherited from the Cox-Ross-Rubinstein model (1979), which discretise time and build the option price by backward recursion; Monte Carlo methods, which simulate thousands of underlying trajectories and average discounted payoffs; and finite differences, which solve the Black-Scholes PDE numerically on a grid.

Each has its strengths and weaknesses, and the choice depends on the product, the dimension, and the required precision.

3.2 Binomial Trees

3.2.1 CRR Algorithm

Recall the Cox–Ross–Rubinstein model (Chapter 1). The algorithm proceeds in two phases:

1. **Forward phase**: build the price tree $S_{n,j} = S_0 u^j d^{n-j}$ for $n = 0, \ldots, N$ and $j = 0, \ldots, n$.

2. **Backward phase**: recursive computation of option values from $n = N$ to $n = 0$.

Backward Recursion

For a European option:

$$V_{n,j} = e^{-r\Delta t}\big[q \cdot V_{n+1,j+1} + (1-q) \cdot V_{n+1,j}\big],$$

with terminal condition $V_{N,j} = h(S_{N,j})$.
For an American option, we add the possibility of early exercise:

$$V_{n,j} = \max\!\big(h(S_{n,j}),\ e^{-r\Delta t}[qV_{n+1,j+1} + (1-q)V_{n+1,j}]\big).$$

3.2.2 Trinomial Tree

Definition 3.1 (Trinomial Tree)**.** The trinomial tree uses three possible movements:

$$S_{n+1} \in \{uS_n,\ S_n,\ dS_n\}$$

with probabilities (p_u, p_m, p_d) chosen to match the first two moments:

$$p_u = \frac{1}{2}\left(\frac{\sigma^2\Delta t + \nu^2\Delta t^2}{(\Delta x)^2} + \frac{\nu\Delta t}{\Delta x}\right), \tag{3.1}$$

$$p_d = \frac{1}{2}\left(\frac{\sigma^2\Delta t + \nu^2\Delta t^2}{(\Delta x)^2} - \frac{\nu\Delta t}{\Delta x}\right), \tag{3.2}$$

$$p_m = 1 - p_u - p_d, \tag{3.3}$$

where $\nu = r - \sigma^2/2$ and $\Delta x = \sigma\sqrt{3\Delta t}$.

Remark 3.2. The trinomial tree converges faster than the binomial tree and exhibits fewer oscillations in the price as a function of N.

3.3 Monte Carlo Methods

3.3.1 General Principle

Theorem 3.3 (Monte Carlo Pricing)**.** *The price of a European derivative with payoff $h(S_T)$ is:*

$$V_0 = e^{-rT}\mathbb{E}^{\mathbb{Q}}[h(S_T)] \approx \frac{e^{-rT}}{M}\sum_{m=1}^{M} h(S_T^{(m)}),$$

where the $S_T^{(m)}$ are independent realisations of S_T under $\mathbb{Q}$.

Convergence is $O(1/\sqrt{M})$ by the Central Limit Theorem, regardless of the problem dimension. This is the major advantage of Monte Carlo for high-dimensional problems.

Proposition 3.4 (Confidence Interval)**.** The Monte Carlo estimator $\hat{V}_M$ satisfies:

$$\mathbb{P}\left(\left|\hat{V}_M - V_0\right| \leq z_{\alpha/2}\frac{\hat{\sigma}_{\mathrm{MC}}}{\sqrt{M}}\right) \approx 1 - \alpha,$$

where $\hat{\sigma}_{\mathrm{MC}}$ is the empirical standard deviation of discounted payoffs and $z_{\alpha/2}$ the normal quantile.

3.3.2 GBM Simulation

Under the risk-neutral measure:

$$S_T = S_0 \exp\left[\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}\,Z\right], \quad Z \sim \mathcal{N}(0,1).$$

For simulation on a grid $0 = t_0 < t_1 < \cdots < t_N = T$ (necessary for path-dependent options):

$$S_{t_{i+1}} = S_{t_i} \exp\left[\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t}\,Z_i\right].$$

Monte Carlo for a European Call

```python
import numpy as np

def mc_european(S0, K, T, r, sigma, M=100000, option='call'):
    """Monte Carlo price for a European option."""
    Z = np.random.standard_normal(M)
    ST = S0 * np.exp((r - 0.5*sigma**2)*T + sigma*np.sqrt(T)*Z)

    if option == 'call':
        payoffs = np.maximum(ST - K, 0)
    else:
        payoffs = np.maximum(K - ST, 0)

    disc_payoffs = np.exp(-r*T) * payoffs
    price = np.mean(disc_payoffs)
    std_err = np.std(disc_payoffs) / np.sqrt(M)
    ci_95 = (price - 1.96*std_err, price + 1.96*std_err)
    return price, std_err, ci_95

np.random.seed(42)
price, se, ci = mc_european(100, 100, 1.0, 0.05, 0.20)
print(f"MC price: {price:.4f} +/- {se:.4f}")
print(f"95% CI: [{ci[0]:.4f}, {ci[1]:.4f}]")
```

3.3.3 Variance Reduction

Definition 3.5 (Antithetic Variates)**.** For each Z_i, we also compute the payoff with $-Z_i$, then average:

$$\hat{V} = \frac{e^{-rT}}{M}\sum_{m=1}^{M}\frac{h(S_T^{(Z_m)}) + h(S_T^{(-Z_m)})}{2}.$$

Definition 3.6 (Control Variate)**.** We use an estimator Y with known mean $\mathbb{E}[Y]$ correlated with the payoff $X = h(S_T)$:

$$\hat{V}_{\text{CV}} = \frac{1}{M}\sum_{m=1}^{M}\left[X^{(m)} - \beta(Y^{(m)} - \mathbb{E}[Y])\right],$$

where $\beta = \text{Cov}(X,Y)/\text{Var}(Y)$ is the optimal coefficient. A classical choice is $Y = S_T$ with $\mathbb{E}^{\mathbb{Q}}[S_T] = S_0e^{rT}$.

Monte Carlo with Variance Reduction

```python
def mc_antithetic(S0, K, T, r, sigma, M=100000):
    """Monte Carlo with antithetic variates."""
    Z = np.random.standard_normal(M)
    ST_plus = S0 * np.exp((r - 0.5*sigma**2)*T + sigma*np.sqrt(T)*Z)
    ST_minus = S0 * np.exp((r - 0.5*sigma**2)*T - sigma*np.sqrt(T)*Z)
    payoffs = 0.5*(np.maximum(ST_plus - K, 0)
                      + np.maximum(ST_minus - K, 0))
    disc = np.exp(-r*T) * payoffs
    return np.mean(disc), np.std(disc)/np.sqrt(M)

def mc_control_variate(S0, K, T, r, sigma, M=100000):
    """Monte Carlo with control variate (S_T)."""
    Z = np.random.standard_normal(M)
    ST = S0 * np.exp((r - 0.5*sigma**2)*T + sigma*np.sqrt(T)*Z)
    X = np.exp(-r*T) * np.maximum(ST - K, 0)
    Y = np.exp(-r*T) * ST
    EY = S0  # E^Q[exp(-rT)*S_T] = S_0

    beta = np.cov(X, Y)[0, 1] / np.var(Y)
    X_cv = X - beta * (Y - EY)
    return np.mean(X_cv), np.std(X_cv)/np.sqrt(M)

np.random.seed(42)
p1, s1 = mc_antithetic(100, 100, 1.0, 0.05, 0.20)
p2, s2 = mc_control_variate(100, 100, 1.0, 0.05, 0.20)
print(f"Antithetic:      {p1:.4f} (se={s1:.4f})")
print(f"Control variate: {p2:.4f} (se={s2:.4f})")
```

3.4 Finite Differences

The Black–Scholes PDE can be solved numerically by the finite difference method.

3.4.1 Transformation and Discretisation

Setting $x = \ln(S/K)$, $\tau = T - t$, the PDE transforms to:

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

Definition 3.7 (Explicit Scheme)**.** Discretising on a grid (x_i, τ_j):

$$\frac{u_i^{j+1} - u_i^j}{\Delta\tau} = \frac{\sigma^2}{2}\frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{(\Delta x)^2} + \nu\frac{u_{i+1}^j - u_{i-1}^j}{2\Delta x} - ru_i^j.$$

This scheme is **explicit** (the next step is computed directly) but subject to a CFL stability condition.

Definition 3.8 (Implicit Scheme (Crank–Nicolson))**.** The Crank–Nicolson scheme averages explicit and implicit steps:

$$\frac{u_i^{j+1} - u_i^j}{\Delta\tau} = \frac{1}{2}\left[\mathcal{L}u_i^j + \mathcal{L}u_i^{j+1}\right],$$

where $\mathcal{L}$ is the spatial differential operator. This scheme is unconditionally stable and second-order in both time and space.

Finite Differences — Crank–Nicolson

```python
import numpy as np
from scipy.linalg import solve_banded

def fd_crank_nicolson(S0, K, T, r, sigma, Nx=200, Nt=500):
    """European call price via Crank-Nicolson."""
    x_max = 5 * sigma * np.sqrt(T)
    dx = 2 * x_max / Nx
    dt = T / Nt
    x = np.linspace(-x_max, x_max, Nx + 1)

    nu = r - 0.5 * sigma**2
    alpha = 0.5 * dt * sigma**2 / dx**2
    beta_val = 0.5 * dt * nu / (2 * dx)

    # Terminal condition (call)
    u = np.maximum(K * (np.exp(x) - 1), 0)

    # Tridiagonal coefficients
    a = -0.5 * (alpha - beta_val)
    b = 1 + alpha + 0.5 * r * dt
    c = -0.5 * (alpha + beta_val)

    a_e = 0.5 * (alpha - beta_val)
    b_e = 1 - alpha - 0.5 * r * dt
    c_e = 0.5 * (alpha + beta_val)

    AB = np.zeros((3, Nx + 1))
    AB[0, 1:] = c
    AB[1, :] = b
    AB[2, :-1] = a

    for j in range(Nt):
        rhs = a_e * np.roll(u, 1) + b_e * u + c_e * np.roll(u, -1)
        rhs[0] = 0
        rhs[-1] = K * (np.exp(x[-1]) - np.exp(-r * (j+1)*dt))
        AB[1, 0] = 1; AB[0, 1] = 0
        AB[1, -1] = 1; AB[2, -2] = 0
        u = solve_banded((1, 1), AB, rhs)
        AB[1, 0] = b; AB[0, 1] = c
        AB[1, -1] = b; AB[2, -2] = a

    x0 = np.log(S0 / K)
    return np.interp(x0, x, u)

price_fd = fd_crank_nicolson(100, 100, 1.0, 0.05, 0.20)
print(f"Crank-Nicolson price: {price_fd:.4f}")
```

3.5 Method Comparison

Criterion	Trees	Monte Carlo	Finite Diff.
Dimension	low	high	low
American	yes	difficult	yes
Path-dependent	no	yes	difficult
Convergence	$O(1/N)$	$O(1/\sqrt{M})$	$O(h^2)$
Greeks	finite diff.	bump-and-reprice	direct

3.6 Longstaff–Schwartz Method (American Options)

Theorem 3.9 (LSM Algorithm). *For American options, the Longstaff–Schwartz algorithm (2001) estimates the continuation value by regression on polynomials of simulated prices. At each time step t_n in backward:*

1. *Compute the immediate exercise payoff $h(S_{t_n}^{(m)})$.*
2. *Regress discounted future cash flows on basis functions $\psi_k(S_{t_n}^{(m)})$ (e.g., Laguerre polynomials).*
3. *Exercise if $h(S_{t_n}^{(m)}) > \hat{C}(S_{t_n}^{(m)})$ (estimated continuation).*

Longstaff–Schwartz for American Put

```
def lsm_american_put(S0, K, T, r, sigma, M=50000, N=100):
    """American put via Longstaff-Schwartz."""
    dt = T / N
    discount = np.exp(-r * dt)

    # Simulate paths
    Z = np.random.standard_normal((N, M))
    S = np.zeros((N + 1, M))
    S[0] = S0
    for i in range(N):
        S[i+1] = S[i] * np.exp((r - 0.5*sigma**2)*dt
                                + sigma*np.sqrt(dt)*Z[i])

    # Terminal payoff
    cashflow = np.maximum(K - S[N], 0)
    stopping = np.full(M, N)

    # Backward induction
    for n in range(N - 1, 0, -1):
        exercise = np.maximum(K - S[n], 0)
        itm = exercise > 0

        if np.sum(itm) > 0:
            X = S[n, itm]
            Y = cashflow[itm] * discount**(stopping[itm] - n)
            A = np.column_stack([np.ones_like(X), X, X**2, X**3])
```

```
                coeffs = np.linalg.lstsq(A, Y, rcond=None)[0]
                continuation = A @ coeffs

                exercise_now = exercise[itm] > continuation
                idx = np.where(itm)[0][exercise_now]
                cashflow[idx] = exercise[idx]
                stopping[idx] = n

        price = np.mean(cashflow * discount**stopping)
        return price

np.random.seed(42)
p_am = lsm_american_put(100, 100, 1.0, 0.05, 0.20)
print(f"American put (LSM): {p_am:.4f}")
```

3.7 Exercises

Exercise 3.1 (Convergence of Methods).

1. Compare CRR, Monte Carlo, and Crank–Nicolson prices for a European call with $S_0 = 100$, $K = 100$, $T = 1$, $r = 0.05$, $\sigma = 0.20$.
2. Plot the error relative to Black–Scholes as a function of the discretisation parameter (N or M).
3. Measure computation time for each method.

Exercise 3.2 (Variance Reduction).

1. Implement antithetic variates and control variates.
2. Compare standard errors for $M = 10^4, 10^5, 10^6$.
3. Compute the variance reduction factor for each technique.

Exercise 3.3 (American Options).

1. Compare American put prices from CRR, Crank–Nicolson with early exercise, and LSM.
2. Determine the early exercise premium $P_{\text{am}} - P_{\text{eu}}$ as a function of r and σ.

Exercise 3.4 (Monte Carlo Greeks).

1. Compute Δ and Γ by bump-and-reprice: $\Delta \approx (V(S+h) - V(S-h))/(2h)$.
2. Compute Δ by the likelihood ratio method: $\Delta = e^{-rT}\mathbb{E}\big[h(S_T)\frac{Z}{\sigma S_0\sqrt{T}}\big]$.
3. Compare the standard deviations of the two estimators.

Chapter 4

Delta Hedging and Risk Management

The Black-Scholes model does not merely give a price: it tells you how to *hedge*. The central idea is that the risk of an option can be eliminated by continuously holding the right quantity of the underlying asset—the *delta*. By adjusting this position at every instant, the portfolio becomes insensitive to market fluctuations. But perfect hedging is an ideal: in practice, one can only rebalance at discrete intervals, transaction costs accumulate, and volatility is never constant. Understanding these limitations is as important as understanding the theory.

4.1 Introduction

Hedging is the activity of reducing or eliminating risk associated with a derivatives position. This chapter develops the theory and practice of delta hedging, gamma hedging, and analyses the costs and limitations of dynamic hedging.

4.2 Delta Hedging

4.2.1 Principle

Definition 4.1 (Delta Hedging)**. Delta hedging** consists of maintaining a portfolio $\Pi_t = V_t - \Delta_t S_t$ locally insensitive to changes in S, where $\Delta_t = \frac{\partial V}{\partial S}(S_t, t)$.

Theorem 4.2 (Self-Financing Delta-Hedged Portfolio)**.** *Under the Black–Scholes framework with continuous trading, the delta-hedged portfolio is riskless and perfectly replicates the derivative. Its value satisfies:*

$$d\Pi_t = r\Pi_t \, dt.$$

Discrete-Time Hedging

In practice, rebalancing is discrete (every Δt time units). The hedging error arises because Δ changes between rebalancings. Higher frequency improves the hedge but increases transaction costs.

4.2.2 Discrete-Time Hedging Error

Proposition 4.3 (Hedging Error)**.** For a European call hedged with delta rebalancing at dates $t_0, t_1, \ldots, t_N$, the hedging error is:

$$\epsilon = \sum_{i=0}^{N-1} \frac{1}{2}\Gamma(S_{t_i}, t_i) S_{t_i}^2 \left[(\Delta S_{t_i})^2 - \sigma^2 S_{t_i}^2 \Delta t\right] \cdot e^{r(T-t_{i+1})},$$

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

The hedging error has zero expectation under $\mathbb{Q}$ but non-zero variance, proportional to Δt.

Discrete Delta-Hedging Simulation

```python
import numpy as np
from scipy.stats import norm

def bs_delta(S, K, T, r, sigma):
    """Delta of a European call."""
    if T < 1e-10:
        return 1.0 if S > K else 0.0
    d1 = (np.log(S/K) + (r + 0.5*sigma**2)*T) / (sigma*np.sqrt(T))
    return norm.cdf(d1)

def bs_price(S, K, T, r, sigma):
    """Black-Scholes call price."""
    if T < 1e-10:
        return max(S - K, 0)
    d1 = (np.log(S/K) + (r + 0.5*sigma**2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma*np.sqrt(T)
    return S*norm.cdf(d1) - K*np.exp(-r*T)*norm.cdf(d2)

def simulate_delta_hedge(S0, K, T, r, sigma, N_rebal, M=10000):
    """Simulate delta hedging and compute hedging errors."""
    dt = T / N_rebal
    errors = np.zeros(M)

    for m in range(M):
        S = S0
        V0 = bs_price(S, K, T, r, sigma)
        delta = bs_delta(S, K, T, r, sigma)
        cash = V0 - delta * S

        for i in range(N_rebal):
            tau = T - (i+1)*dt
            Z = np.random.standard_normal()
            S_new = S * np.exp((r - 0.5*sigma**2)*dt
                               + sigma*np.sqrt(dt)*Z)
            cash *= np.exp(r * dt)
            if tau > 1e-10:
                delta_new = bs_delta(S_new, K, tau, r, sigma)
            else:
                delta_new = 1.0 if S_new > K else 0.0
            cash -= (delta_new - delta) * S_new
            delta = delta_new
            S = S_new

        portfolio = cash + delta * S
        payoff = max(S - K, 0)
        errors[m] = portfolio - payoff

    return errors

np.random.seed(42)
for N in [10, 50, 252, 1000]:
    errs = simulate_delta_hedge(100, 100, 1.0, 0.05, 0.20, N)
    print(f"N={N:4d}: mean={np.mean(errs):.4f}, "
          f"std={np.std(errs):.4f}")
```

4.3 Gamma Hedging

Definition 4.4 (Gamma Hedging)**. Gamma hedging** makes the portfolio insensitive not only to small changes in S (delta neutral) but also to larger changes (gamma neutral). A second instrument (typically another option) is used to neutralise gamma.

Proposition 4.5 (Constructing a Gamma-Neutral Portfolio)**.** Let V be the derivative to hedge and O a hedging instrument with gamma Γ_O. The position in O needed to neutralise gamma is:

$$w_O = -\frac{\Gamma_V}{\Gamma_O}.$$

After this hedge, the position in the underlying is adjusted to restore delta neutrality:

$$\Delta_{\text{total}} = \Delta_V + w_O \Delta_O.$$

Example 4.6. A trader is short a call ($\Delta = 0.55$, $\Gamma = 0.03$). They have access to a call with different maturity ($\Delta_O = 0.40$, $\Gamma_O = 0.05$).

Gamma-neutral position: $w_O = -(-0.03)/0.05 = 0.60$.

New delta: $-0.55 + 0.60 \times 0.40 = -0.31$.

Buy 0.31 units of the underlying to become delta neutral.

4.4 Vega Hedging

Definition 4.7 (Vega Hedging)**. Vega hedging** aims to neutralise the portfolio's sensitivity to volatility. Since the underlying's vega is zero, options must be used.

Remark 4.8. In practice, vega hedging is more important than gamma hedging because implied volatility movements can generate substantial profit and loss (P&L).

CHAPTER 4. DELTA HEDGING AND RISK MANAGEMENT

4.5 P&L Decomposition

Theorem 4.9 (P&L Decomposition of a Delta-Hedged Portfolio)**.** *The instantaneous P&L of a delta-hedged portfolio is:*

$$dP\&L = \frac{1}{2}\Gamma S^2\left[(dS/S)^2 - \sigma^2\,dt\right] + \mathcal{V}\,d\sigma_{impl} + \cdots$$

The first term is the ***gamma P&L*** *(profitable when realised volatility exceeds implied volatility), the second is the* ***vega P&L****.*

Taylor Approximation of P&L

For a general portfolio, the Taylor approximation gives:

$$\Delta\text{P\&L} \approx \Delta\cdot\delta S + \frac{1}{2}\Gamma(\delta S)^2 + \Theta\,\delta t + \mathcal{V}\,\delta\sigma + \rho\,\delta r.$$

4.6 Transaction Costs and Optimal Hedging

Theorem 4.10 (Optimal Rebalancing Frequency (Leland))**.** *In the presence of proportional transaction costs $k\,|\delta S|$, Leland (1985) shows that optimal hedging uses a modified volatility:*

$$\hat{\sigma}^2 = \sigma^2\left(1 + \sqrt{\frac{2}{\pi}}\frac{k}{\sigma\sqrt{\Delta t}}\right).$$

Proposition 4.11 (Whalley–Wilmott Model)**.** An alternative approach rebalances only when delta deviates sufficiently from its target. The optimal no-trade band is:

$$\Delta_{\text{target}} \pm \left(\frac{3k\,e^{-r\tau}\Gamma^2S^2}{2\lambda}\right)^{1/3},$$

where λ represents the trader's risk aversion.

Hedging with Transaction Costs

```
def hedge_with_costs(S0, K, T, r, sigma, N_rebal, k=0.001, M=5000):
    """Delta hedging with proportional transaction costs."""
    dt = T / N_rebal
    total_costs = np.zeros(M)
    hedge_errors = np.zeros(M)

    for m in range(M):
        S = S0
        V0 = bs_price(S, K, T, r, sigma)
        delta = bs_delta(S, K, T, r, sigma)
        cash = V0 - delta * S
        cost = k * abs(delta) * S

        for i in range(N_rebal):
            tau = T - (i+1)*dt
            Z = np.random.standard_normal()
            S_new = S * np.exp((r-0.5*sigma**2)*dt
                               + sigma*np.sqrt(dt)*Z)
            cash *= np.exp(r*dt)

            delta_new = bs_delta(S_new, K, max(tau, 1e-10), r, sigma)
            trade = delta_new - delta
            cost += k * abs(trade) * S_new
            cash -= trade * S_new
            delta = delta_new
            S = S_new

        portfolio = cash + delta * S
        payoff = max(S - K, 0)
        hedge_errors[m] = portfolio - payoff
        total_costs[m] = cost

    print(f"Average cost: {np.mean(total_costs):.4f}")
    print(f"Error std:    {np.std(hedge_errors):.4f}")
    return hedge_errors, total_costs

np.random.seed(42)
hedge_with_costs(100, 100, 1.0, 0.05, 0.20, 252, k=0.001)
```

4.7 Hedging in Incomplete Markets

Remark 4.12. When the market is incomplete (stochastic volatility, jumps), perfect hedging is impossible. Alternative approaches include:

- **Minimum variance hedging**: minimise $\mathrm{Var}(\Pi_T - H)$ over admissible strategies.
- **Super-replication**: find the minimal cost of a portfolio that dominates the payoff.
- **Quantile hedging**: control the probability of loss.

4.8 Exercises

Exercise 4.1 (Delta-Hedging Simulation). 1. Simulate delta hedging of a European call for $N = 10, 50, 252, 1000$ rebalancings.

2. Plot the histogram of hedging errors for each N.

3. Verify that the error variance decreases as $O(1/N)$.

Exercise 4.2 (Gamma Hedging). 1. Construct a delta-gamma neutral portfolio from a short ATM call, a long OTM call, and the underlying.

2. Simulate the P&L over one week and compare with the delta-only hedged portfolio.

Exercise 4.3 (P&L Analysis). 1. Decompose the daily P&L of a delta-hedged portfolio into delta, gamma, theta, and vega components.

CHAPTER 4. DELTA HEDGING AND RISK MANAGEMENT

2. Show that for a delta-neutral portfolio, P&L $\approx \frac{1}{2}\Gamma S^2(\sigma_r^2 - \sigma_i^2)\, dt$ where σ_r is realised volatility and σ_i is implied volatility.

Exercise 4.4 (Transaction Costs).

1. Implement Leland hedging with modified volatility.
2. Compare total cost and hedging error with the standard method for different values of k.
3. Implement the Whalley–Wilmott no-trade band and analyse the cost/risk trade-off.

Chapter 5

Stochastic Calculus in Finance

Brownian motion is the fundamental building block of quantitative finance—but it is a mathematically monstrous object: its paths are continuous yet *nowhere differentiable.* One cannot differentiate in the classical sense, and the ordinary integral breaks down. It was Kiyosi Itô who, in the 1940s in Japan, constructed the differential calculus adapted to this situation: the *Itô integral* and the celebrated *Itô formula*, which is to stochastic calculus what the chain rule is to classical calculus—with an additional corrective term, the $\frac{1}{2}\sigma^2 f''$ term, that changes everything. This chapter builds these tools and shows how they apply to financial modelling.

5.1 Brownian Motion

Definition 5.1 (Standard Brownian Motion). A **standard Brownian motion** (or Wiener process) $(W_t)_{t\geq 0}$ is a continuous stochastic process such that:

1. $W_0 = 0$ a.s.
2. The paths $t \mapsto W_t(\omega)$ are continuous a.s.
3. Increments are independent: for $0 \leq s < t \leq u < v$, $W_t - W_s$ and $W_v - W_u$ are independent.
4. Increments are Gaussian: $W_t - W_s \sim \mathcal{N}(0, t - s)$.

Proposition 5.2 (Properties of Brownian Motion).

1. $\mathbb{E}[W_t] = 0$ and $\mathrm{Var}(W_t) = t$ for all $t \geq 0$.
2. $\mathrm{Cov}(W_s, W_t) = \min(s, t)$.
3. W_t is a martingale with respect to its natural filtration.
4. $W_t^2 - t$ is a martingale.
5. Paths are a.s. nowhere differentiable.
6. The quadratic variation satisfies $\langle W \rangle_t = t$.

Simulating Brownian Paths

```python
import numpy as np
import matplotlib.pyplot as plt

def brownian_paths(T, N, M):
    """Simulate M Brownian paths on [0, T] with N steps."""
    dt = T / N
    dW = np.sqrt(dt) * np.random.standard_normal((M, N))
    W = np.zeros((M, N + 1))
    W[:, 1:] = np.cumsum(dW, axis=1)
    t = np.linspace(0, T, N + 1)
    return t, W

np.random.seed(42)
t, W = brownian_paths(1.0, 1000, 5)
for i in range(5):
    plt.plot(t, W[i])
plt.xlabel('t'); plt.ylabel('$W_t$')
plt.title('Brownian Paths')
plt.grid(True)
plt.show()
```

5.2 The Itô Integral

5.2.1 Construction

The stochastic integral extends the classical integral to random integrands with respect to Brownian motion.

Definition 5.3 (Itô Integral)**.** For an adapted process $(H_t)_{t\geq 0}$ satisfying $\mathbb{E}\left[\int_0^T H_t^2\,dt\right] < \infty$, the **Itô integral** is defined as:

$$I_T(H) = \int_0^T H_t\,dW_t = \lim_{n\to\infty} \sum_{i=0}^{n-1} H_{t_i}(W_{t_{i+1}} - W_{t_i}).$$

The limit is taken in mean square (L^2).

Theorem 5.4 (Properties of the Itô Integral)**.** *Let $I_t = \int_0^t H_s\,dW_s$. Then:*

1. *I_t is a martingale: $\mathbb{E}[I_t \mid \mathcal{F}_s] = I_s$ for $s \leq t$.*
2. ***Itô isometry**: $\mathbb{E}[I_T^2] = \mathbb{E}\left[\int_0^T H_t^2\,dt\right]$.*
3. *$\mathbb{E}[I_T] = 0$.*
4. *The quadratic variation is $\langle I\rangle_t = \int_0^t H_s^2\,ds$.*

Itô vs Stratonovich

The Itô integral evaluates the integrand at the **left endpoint** of each interval: H_{t_i} rather than $H_{t_{i+1}}$ or $\frac{1}{2}(H_{t_i} + H_{t_{i+1}})$. The Stratonovich choice (midpoint evaluation) gives a different result and satisfies classical calculus rules, but the Itô integral is preferred in finance because it preserves the martingale property (and hence risk-neutral pricing).

5.2.2 Calculus Rules

The rules of calculus differ from the classical case. The key identities are:

$$(dW_t)^2 = dt, \quad (dt)^2 = 0, \quad dW_t \cdot dt = 0.$$

5.3 Itô's Formula

Theorem 5.5 (Itô's Formula — One-Dimensional Case)**.** *Let (X_t) be an Itô process satisfying $dX_t = \mu_t\, dt + \sigma_t\, dW_t$ and $f \in C^{1,2}(\mathbb{R}^+ \times \mathbb{R})$. Then:*

$$df(t, X_t) = \frac{\partial f}{\partial t}\, dt + \frac{\partial f}{\partial x}\, dX_t + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}\sigma_t^2\, dt.$$

In expanded form:

$$df(t, X_t) = \left(\frac{\partial f}{\partial t} + \mu_t \frac{\partial f}{\partial x} + \frac{1}{2}\sigma_t^2 \frac{\partial^2 f}{\partial x^2}\right) dt + \sigma_t \frac{\partial f}{\partial x}\, dW_t.$$

Itô's Formula — Compact Form

$$df = \left(\partial_t f + \mu\, \partial_x f + \tfrac{1}{2}\sigma^2 \partial_{xx} f\right) dt + \sigma\, \partial_x f\, dW_t.$$

The extra term $\frac{1}{2}\sigma^2\partial_{xx} f\, dt$ has no analogue in deterministic calculus; it arises from the non-zero quadratic variation of Brownian motion.

Example 5.6. Apply Itô's formula to $f(x) = x^2$ with $X_t = W_t$:

$$d(W_t^2) = 2W_t\, dW_t + \frac{1}{2} \cdot 2\, dt = 2W_t\, dW_t + dt.$$

Integrating: $W_T^2 = 2\int_0^T W_t\, dW_t + T$, hence:

$$\int_0^T W_t\, dW_t = \frac{W_T^2 - T}{2}.$$

Note the difference from classical calculus, which would give $W_T^2/2$.

Theorem 5.7 (Multidimensional Itô's Formula)**.** *Let $dX_t^i = \mu_t^i\, dt + \sum_j \sigma_t^{ij}\, dW_t^j$ and $f \in C^{1,2}$. Then:*

$$df = \partial_t f\, dt + \sum_i \partial_{x_i} f\, dX_t^i + \frac{1}{2}\sum_{i,j} \partial_{x_i x_j} f\, d\langle X^i, X^j\rangle_t.$$

CHAPTER 5. STOCHASTIC CALCULUS IN FINANCE

5.4 Girsanov's Theorem

Theorem 5.8 (Girsanov). *Let (W_t) be a $\mathbb{P}$-Brownian motion and (θ_t) an adapted process satisfying the Novikov condition:*

$$\mathbb{E}^{\mathbb{P}}\left[\exp\left(\frac{1}{2}\int_0^T \theta_t^2\,dt\right)\right] < \infty.$$

Define the Radon–Nikodym density:

$$\left.\frac{d\mathbb{Q}}{d\mathbb{P}}\right|_{\mathcal{F}_T} = \mathcal{E}_T(\theta) = \exp\left(-\int_0^T \theta_t\,dW_t - \frac{1}{2}\int_0^T \theta_t^2\,dt\right).$$

Then under $\mathbb{Q}$, the process $\tilde{W}_t = W_t + \int_0^t \theta_s\,ds$ is a standard Brownian motion.

Application to Finance

Girsanov's theorem enables the passage from the historical measure $\mathbb{P}$ (under which $dS_t/S_t = \mu\,dt + \sigma\,dW_t$) to the risk-neutral measure $\mathbb{Q}$ (under which $dS_t/S_t = r\,dt + \sigma\,d\tilde{W}_t$) by setting $\theta = (\mu - r)/\sigma$. The quantity θ is the **market price of risk**.

5.5 Martingale Representation Theorem

Theorem 5.9 (Martingale Representation). *Every square-integrable martingale $(M_t)_{0\le t\le T}$ adapted to the Brownian filtration can be written as:*

$$M_t = M_0 + \int_0^t H_s\,dW_s,$$

for a unique predictable process (H_t) satisfying $\mathbb{E}\left[\int_0^T H_t^2\,dt\right] < \infty$.

Remark 5.10. This theorem guarantees that in the Black–Scholes model (complete market), every square-integrable contingent claim is replicable. The process H_t provides the hedging strategy.

5.6 Stochastic Differential Equations

Definition 5.11 (General SDE). A **stochastic differential equation** (SDE) takes the form:

$$dX_t = b(t, X_t)\,dt + \sigma(t, X_t)\,dW_t, \quad X_0 = x_0,$$

where b is the drift coefficient and σ the diffusion coefficient.

Theorem 5.12 (Existence and Uniqueness). *If b and σ are Lipschitz continuous in x uniformly in t and have linear growth, then the SDE admits a unique strong solution $(X_t)_{0\le t\le T}$.*

Theorem 5.13 (PDE–SDE Link (Feynman–Kac))**.** *Let $u(t,x)$ be a solution of:*

$$\partial_t u + b(t,x)\partial_x u + \frac{1}{2}\sigma^2(t,x)\partial_{xx}u - r\,u = 0, \quad u(T,x) = g(x).$$

Then:

$$u(t,x) = \mathbb{E}\left[e^{-r(T-t)}g(X_T) \mid X_t = x\right],$$

where $dX_s = b(s,X_s)\,ds + \sigma(s,X_s)\,dW_s$.

Remark 5.14. The Feynman–Kac theorem establishes the fundamental link between PDE-based pricing (Black–Scholes) and risk-neutral expectation pricing (Monte Carlo).

5.7 Stochastic Exponential

Definition 5.15 (Stochastic Exponential)**.** The **stochastic exponential** (or Doléans-Dade exponential) of an Itô process X_t is:

$$\mathcal{E}_t(X) = \exp\left(X_t - X_0 - \frac{1}{2}\langle X\rangle_t\right).$$

It satisfies $d\mathcal{E}_t = \mathcal{E}_t\,dX_t$, $\mathcal{E}_0 = 1$.

Proposition 5.16. The GBM $S_t = S_0\exp[(\mu - \sigma^2/2)t + \sigma W_t]$ is the stochastic exponential of $\mu t + \sigma W_t$:

$$S_t = S_0\,\mathcal{E}_t(\mu\,\cdot + \sigma W).$$

5.8 Implementation: Discretisation Schemes

Euler–Maruyama and Milstein Schemes

```python
import numpy as np

def euler_maruyama(x0, b, sigma, T, N, M):
    """Euler-Maruyama scheme for dX = b(X)dt + sigma(X)dW."""
    dt = T / N
    X = np.full(M, x0, dtype=float)
    for i in range(N):
        dW = np.sqrt(dt) * np.random.standard_normal(M)
        X = X + b(X)*dt + sigma(X)*dW
    return X

def milstein(x0, b, sigma, sigma_prime, T, N, M):
    """Milstein scheme (strong order 1.0)."""
    dt = T / N
    X = np.full(M, x0, dtype=float)
    for i in range(N):
        dW = np.sqrt(dt) * np.random.standard_normal(M)
        X = (X + b(X)*dt + sigma(X)*dW
             + 0.5*sigma(X)*sigma_prime(X)*(dW**2 - dt))
    return X
```

```python
# Application to GBM: dS = r*S*dt + sig*S*dW
r_val, sig = 0.05, 0.20
S0, T, N, M = 100, 1.0, 100, 100000

np.random.seed(42)
ST_euler = euler_maruyama(S0, lambda S: r_val*S, lambda S: sig*S,
                          T, N, M)
ST_milstein = milstein(S0, lambda S: r_val*S, lambda S: sig*S,
                       lambda S: sig, T, N, M)
Z = np.random.standard_normal(M)
ST_exact = S0 * np.exp((r_val - 0.5*sig**2)*T + sig*np.sqrt(T)*Z)

print(f"Euler:    E[S_T] = {np.mean(ST_euler):.2f}")
print(f"Milstein: E[S_T] = {np.mean(ST_milstein):.2f}")
print(f"Exact:    E[S_T] = {S0*np.exp(r_val*T):.2f}")
```

5.9 Exercises

Exercise 5.1 (Itô's Formula). 1. Compute $d(e^{W_t})$ and $d(e^{\alpha W_t - \alpha^2 t/2})$.

2. Show that $\mathcal{E}_t = e^{\sigma W_t - \sigma^2 t/2}$ is a martingale.

3. Apply Itô's formula to $f(t,x) = e^{-rt}x$ to verify that $e^{-rt}S_t$ is a $\mathbb{Q}$-martingale.

Exercise 5.2 (Girsanov's Theorem). 1. Verify the change of measure explicitly for GBM: write S_t under $\mathbb{P}$ then under $\mathbb{Q}$.

2. Compute the market price of risk $\theta = (\mu - r)/\sigma$ for $\mu = 0.10$, $r = 0.03$, $\sigma = 0.25$.

3. Show that the Radon–Nikodym density $\mathcal{E}_T(\theta)$ is a $\mathbb{P}$-martingale.

Exercise 5.3 (Convergence of Schemes). 1. Compare the strong error $\mathbb{E}[|S_T^N - S_T|]$ of Euler and Milstein schemes as a function of N.

2. Verify that the strong convergence order is 1/2 for Euler and 1 for Milstein.

3. Implement a weak order 2 scheme (Platen) and compare.

Exercise 5.4 (Feynman–Kac). 1. Verify the Feynman–Kac theorem for the European call by comparing the Black–Scholes PDE solution with the Monte Carlo expectation.

2. Apply Feynman–Kac to the heat equation $\partial_t u = \frac{1}{2}\partial_{xx} u$ and interpret.

Chapter 6

Stochastic Volatility Models

The Black-Scholes model assumes constant volatility. But markets tell a different story: when one computes implied volatility from observed option prices, it varies with strike and maturity, drawing the celebrated “volatility smile.” This phenomenon, which became glaring after the 1987 crash, motivated a new generation of models where volatility itself is a stochastic process. The Heston model (1993), the SABR model (Hagan et al., 2002), and Dupire’s local volatility models (1994) each attempt to capture this dynamic.

6.1 Motivation: The Volatility Smile

The Black–Scholes model assumes constant volatility σ. In practice, however, the implied volatility $\sigma_{\text{impl}}(K,T)$ depends on the strike K and maturity T, forming a **volatility surface** with a characteristic *smile* or *skew*.

Definition 6.1 (Volatility Smile)**.** The **volatility smile** is the curve $K \mapsto \sigma_{\text{impl}}(K,T)$ for a fixed maturity. In equity markets, one typically observes a negative *skew*: implied volatility is higher for OTM puts (low strikes) than for OTM calls (high strikes).

Remark 6.2. The smile is incompatible with the Black–Scholes model and motivates the introduction of stochastic volatility or jump-diffusion models.

6.2 Local Volatility (Dupire)

Theorem 6.3 (Dupire’s Equation (1994))**.** *If European option prices $C(K,T)$ are known for all (K,T), the* ***local volatility*** *$\sigma_{loc}(K,T)$ is defined by:*

$$\sigma_{loc}^2(K,T) = \frac{\partial_T C + rK\partial_K C}{\frac{1}{2}K^2\partial_{KK}C}.$$

The underlying then satisfies: $dS_t = rS_t\,dt + \sigma_{loc}(S_t,t)S_t\,dW_t^{\mathbb{Q}}$.

Limitations of Local Volatility

Dupire’s model reproduces today’s volatility surface exactly, but its predictions of future smile dynamics are often unrealistic (the smile flattens too quickly). Stochastic volatility models offer better dynamics.

CHAPTER 6. STOCHASTIC VOLATILITY MODELS

6.3 The Heston Model

Definition 6.4 (Heston Model (1993))**.** The **Heston model** describes the underlying and its variance by the system:

$$dS_t = rS_t\,dt + \sqrt{v_t}\,S_t\,dW_t^1, \tag{6.1}$$
$$dv_t = \kappa(\bar{v} - v_t)\,dt + \xi\sqrt{v_t}\,dW_t^2, \tag{6.2}$$

with $d\langle W^1, W^2\rangle_t = \rho\,dt$ and parameters:

- $v_0 > 0$: initial variance,
- $\kappa > 0$: mean-reversion speed,
- $\bar{v} > 0$: long-run variance,
- $\xi > 0$: volatility of variance (vol of vol),
- $\rho \in [-1, 1]$: correlation ($\rho < 0$ generates skew).

Theorem 6.5 (Feller Condition)**.** *The variance v_t remains strictly positive if and only if the **Feller condition** is satisfied:*

$$2\kappa\bar{v} \geq \xi^2.$$

6.3.1 Pricing via Characteristic Function

Theorem 6.6 (Heston Semi-Analytical Formula)**.** *The European call price in the Heston model is:*

$$C = S_0P_1 - Ke^{-rT}P_2,$$

where the probabilities P_j are obtained by Fourier inversion:

$$P_j = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-iu\ln K}\varphi_j(u)}{iu}\right]du,$$

and $\varphi_j(u)$ is the characteristic function of the log-price under the associated measures.

Heston Pricing via Fourier Transform

```python
import numpy as np
from scipy.integrate import quad

def heston_char_func(u, T, r, v0, kappa, vbar, xi, rho):
    """Characteristic function of the log-price under Heston."""
    d = np.sqrt((rho*xi*1j*u - kappa)**2 + xi**2*(1j*u + u**2))
    g = (kappa - rho*xi*1j*u - d) / (kappa - rho*xi*1j*u + d)

    C = r*1j*u*T + (kappa*vbar/xi**2) * (
        (kappa - rho*xi*1j*u - d)*T
        - 2*np.log((1 - g*np.exp(-d*T))/(1 - g))
    )
    D = ((kappa - rho*xi*1j*u - d)/xi**2) * (
        (1 - np.exp(-d*T))/(1 - g*np.exp(-d*T))
    )
    return np.exp(C + D*v0)

def heston_call(S0, K, T, r, v0, kappa, vbar, xi, rho):
    """European call price under the Heston model."""
    ln_K = np.log(K)

    def integrand(u):
        phi = heston_char_func(u - 0.5j, T, r, v0,
                               kappa, vbar, xi, rho)
        return np.real(np.exp(-1j*u*ln_K) * phi / (u**2 + 0.25))

    integral, _ = quad(integrand, 0, 200, limit=200)
    return S0 - np.sqrt(S0*K)*np.exp(-r*T)/np.pi * integral

# Heston parameters
S0, K, T, r = 100, 100, 1.0, 0.05
v0, kappa, vbar, xi, rho = 0.04, 2.0, 0.04, 0.3, -0.7

price = heston_call(S0, K, T, r, v0, kappa, vbar, xi, rho)
print(f"Heston price: {price:.4f}")
```

6.4 The SABR Model

Definition 6.7 (SABR Model)**.** The **SABR** model (Stochastic Alpha Beta Rho, Hagan et al., 2002) is defined by:

$$dF_t = \alpha_t F_t^{\beta}\, dW_t^1, \tag{6.3}$$
$$d\alpha_t = \nu \alpha_t\, dW_t^2, \tag{6.4}$$

with $d\langle W^1, W^2\rangle_t = \rho\, dt$, $F_0 = f$ (forward rate), $\alpha_0 = \alpha$, and $\beta \in [0,1]$.

Theorem 6.8 (Hagan's Approximation for Implied Volatility)**.** *The Black implied volatility in the SABR model is approximated by:*

$$\sigma_B(K,f) \approx \frac{\alpha}{(fK)^{(1-\beta)/2}\left[1 + \frac{(1-\beta)^2}{24}\ln^2\frac{f}{K} + \cdots\right]} \cdot \frac{z}{x(z)} \cdot \left[1 + \left(\frac{(1-\beta)^2}{24}\frac{\alpha^2}{(fK)^{1-\beta}} + \frac{1}{4}\frac{\rho\beta\nu\alpha}{(fK)^{(1-\beta)/2}} + \frac{2-3}{24}\right.\right.$$

where $z = \frac{\nu}{\alpha}(fK)^{(1-\beta)/2}\ln\frac{f}{K}$ *and* $x(z) = \ln\frac{\sqrt{1-2\rho z+z^2}+z-\rho}{1-\rho}$.

SABR Implied Volatility

```
import numpy as np

def sabr_vol(K, f, T, alpha, beta, rho, nu):
    """Hagan approximation for SABR implied volatility."""
    if abs(f - K) < 1e-10:
        # ATM case
        fk = f**(1 - beta)
        vol = (alpha / fk) * (
            1 + ((1-beta)**2/24 * alpha**2/fk**2
                 + 0.25*rho*beta*nu*alpha/fk
                 + (2-3*rho**2)/24 * nu**2) * T
        )
        return vol

    fk_mid = (f * K)**((1 - beta)/2)
    ln_fk = np.log(f / K)
    z = nu / alpha * fk_mid * ln_fk
    x_z = np.log((np.sqrt(1 - 2*rho*z + z**2) + z - rho)
                 / (1 - rho))

    prefix = alpha / (fk_mid * (1 + (1-beta)**2/24 * ln_fk**2
                                + (1-beta)**4/1920 * ln_fk**4))
    correction = 1 + ((1-beta)**2/24 * alpha**2 / (f*K)**(1-beta)
                      + 0.25*rho*beta*nu*alpha / fk_mid
                      + (2-3*rho**2)/24 * nu**2) * T

    return prefix * z / x_z * correction

# Example: SABR smile
f, T = 0.03, 5.0
alpha, beta, rho, nu = 0.02, 0.5, -0.3, 0.4
strikes = np.linspace(0.01, 0.06, 50)
vols = [sabr_vol(K, f, T, alpha, beta, rho, nu) for K in strikes]
```

6.5 Calibration

Definition 6.9 (Calibration Problem). **Calibration** consists of determining model parameters θ^* that minimise the discrepancy with market prices (or implied volatilities):

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} w_i \big(\sigma_{\text{mod}}(K_i, T_i; \theta) - \sigma_{\text{mkt}}(K_i, T_i)\big)^2.$$

Heston Model Calibration

```
from scipy.optimize import minimize

def calibrate_heston(market_data, S0, r):
    """Calibrate Heston to market implied volatilities.

    market_data: list of (K, T, sigma_mkt)
    """
    def objective(params):
        v0, kappa, vbar, xi, rho = params
        if v0<0 or kappa<0 or vbar<0 or xi<0 or abs(rho)>1:
            return 1e10
        total = 0
        for K, T, sigma_mkt in market_data:
            try:
                C_mod = heston_call(S0, K, T, r, v0, kappa,
                                    vbar, xi, rho)
                sigma_mod = implied_vol_from_price(
                    C_mod, S0, K, T, r)
                total += (sigma_mod - sigma_mkt)**2
            except:
                total += 1e6
        return total

    x0 = [0.04, 1.5, 0.04, 0.3, -0.5]
    bounds = [(0.001,1), (0.01,10), (0.001,1),
              (0.01,2), (-0.99,0.99)]
    result = minimize(objective, x0, method='L-BFGS-B',
                      bounds=bounds)
    return result.x
```

6.6 Local vs Stochastic Volatility

Property	**Local (Dupire)**	**Stochastic (Heston)**
Exact calibration	Yes	Approximate
Smile dynamics	Unrealistic	Realistic
Number of factors	1	2
Market	Complete	Incomplete
Greeks	Simple	Complex

6.7 Stochastic Local Volatility (SLV) Models

Definition 6.10 (SLV Model)**. SLV models** combine local and stochastic volatility:

$$dS_t = rS_t\,dt + L(S_t,t)\sqrt{v_t}\,S_t\,dW_t^1,$$

where $L(S,t)$ is the leverage function calibrated to reproduce the volatility surface exactly, while the dynamics of v_t control the future behaviour of the smile.

6.8 Exercises

Exercise 6.1 (Volatility Surface)**.** 1. Generate option prices in the Heston model for $K/S_0 \in \{0.8, 0.9, 1.0, 1.1, 1.2\}$ and $T \in \{0.25, 0.5, 1, 2\}$.

2. Compute Black–Scholes implied volatility for each price.

3. Plot the smile for each maturity and comment on the impact of ρ.

Exercise 6.2 (Feller Condition)**.** 1. For $\kappa = 2$, $\bar{v} = 0.04$, determine the maximum value of ξ satisfying Feller.

CHAPTER 6. STOCHASTIC VOLATILITY MODELS

2. Simulate paths of v_t with and without Feller and observe behaviour near zero.

Exercise 6.3 (SABR Calibration)**.**

1. Calibrate the SABR model to swaption data (implied volatilities for different strikes).
2. Analyse the stability of parameters (α, ρ, ν) as a function of maturity.
3. Compare with Heston calibration on the same data.

Exercise 6.4 (Monte Carlo for Heston)**.**

1. Implement the truncated Euler scheme for Heston: $v_{t+\Delta t} = \max(v_t + \kappa(\bar{v} - v_t)\Delta t + \xi\sqrt{v_t^+}\Delta W^2, 0)$.
2. Compare with the characteristic function method.
3. Study the impact of ρ on the smile skew.

Chapter 7

Interest Rate Models

Until now, we have treated the interest rate as a constant. But anyone who has observed bond markets knows that rates are anything but constant: they form a *curve* — the term structure — that evolves stochastically over time. Modelling this evolution is one of the richest challenges in quantitative finance. Oldrich Vasicek, in 1977, proposed the first stochastic short-rate model with mean reversion, drawing inspiration from the Ornstein–Uhlenbeck process. John Cox, Jonathan Ingersoll, and Stephen Ross (1985) corrected Vasicek's flaw — the possibility of negative rates — by introducing a volatility proportional to $\sqrt{r}$. Then David Heath, Robert Jarrow, and Andrew Morton (1992) reversed the perspective entirely: instead of modelling the short rate, they modelled the entire forward curve directly. Each approach serves different needs, and understanding them is understanding how markets price time and uncertainty.

7.1 Term Structure of Interest Rates

Definition 7.1 (Zero-Coupon Bond and Yield Curve)**.** A **zero-coupon bond** with maturity T is a contract paying 1 unit at time T. Its price at time t is denoted $P(t,T)$. The **continuously compounded zero rate** is:

$$R(t,T) = -\frac{\ln P(t,T)}{T-t}.$$

The curve $T \mapsto R(t,T)$ is the **term structure** of interest rates at time t.

Definition 7.2 (Instantaneous Forward Rate)**.** The **instantaneous forward rate** is:

$$f(t,T) = -\frac{\partial \ln P(t,T)}{\partial T},$$

so that $P(t,T) = \exp\Big(-\int_t^T f(t,u)\,du\Big)$.

Definition 7.3 (Short Rate)**.** The **short rate** (or instantaneous rate) is: $r_t = f(t,t) = \lim_{T\to t^+} R(t,T)$.

7.2 Short-Rate Models

7.2.1 Vasicek Model (1977)

Definition 7.4 (Vasicek Model)**.** The short rate follows an Ornstein–Uhlenbeck process:

$$dr_t = a(b - r_t)\,dt + \sigma\,dW_t,$$

with $a > 0$ (mean-reversion speed), b (long-term rate), $\sigma > 0$ (volatility).

Theorem 7.5 (Zero-Coupon Price — Vasicek)**.** *The zero-coupon bond price has the affine form:*

$$P(t,T) = A(t,T)\exp\bigl(-B(t,T)\,r_t\bigr),$$

where, with $\tau = T - t$:

$$B(\tau) = \frac{1 - e^{-a\tau}}{a}, \tag{7.1}$$

$$A(\tau) = \exp\left[\left(b - \frac{\sigma^2}{2a^2}\right)\bigl(B(\tau) - \tau\bigr) - \frac{\sigma^2}{4a}B(\tau)^2\right]. \tag{7.2}$$

Remark 7.6. The Vasicek model allows negative interest rates, which was considered a defect until the 2010s but proved realistic in the European negative rate environment.

7.2.2 Cox–Ingersoll–Ross Model (CIR, 1985)

Definition 7.7 (CIR Model)**.**

$$dr_t = a(b - r_t)\,dt + \sigma\sqrt{r_t}\,dW_t.$$

The $\sqrt{r_t}$ term in the diffusion ensures $r_t \geq 0$ under the Feller condition $2ab \geq \sigma^2$.

Theorem 7.8 (Zero-Coupon Price — CIR)**.** *The price is affine in r_t: $P(t,T) = A(\tau)\exp(-B(\tau)r_t)$ with:*

$$B(\tau) = \frac{2(e^{\gamma\tau} - 1)}{(\gamma + a)(e^{\gamma\tau} - 1) + 2\gamma}, \tag{7.3}$$

$$A(\tau) = \left[\frac{2\gamma e^{(a+\gamma)\tau/2}}{(\gamma + a)(e^{\gamma\tau} - 1) + 2\gamma}\right]^{2ab/\sigma^2}, \tag{7.4}$$

where $\gamma = \sqrt{a^2 + 2\sigma^2}$.

7.2.3 Hull–White Model (1990)

Definition 7.9 (Hull–White Model)**.** An extension of Vasicek with time-dependent parameters:

$$dr_t = \bigl[\theta(t) - a\,r_t\bigr]\,dt + \sigma\,dW_t,$$

where $\theta(t)$ is chosen to calibrate the initial yield curve exactly:

$$\theta(t) = \frac{\partial f^M(0,t)}{\partial t} + a\,f^M(0,t) + \frac{\sigma^2}{2a}(1 - e^{-2at}),$$

with $f^M(0,t)$ the market forward rate.

Vasicek and CIR Models

```python
import numpy as np

def vasicek_bond(r, t, T, a, b, sigma):
    """Vasicek zero-coupon price."""
    tau = T - t
    B = (1 - np.exp(-a * tau)) / a
    A = np.exp((b - sigma**2/(2*a**2)) * (B - tau)
               - sigma**2/(4*a) * B**2)
    return A * np.exp(-B * r)

def cir_bond(r, t, T, a, b, sigma):
    """CIR zero-coupon price."""
    tau = T - t
    gamma = np.sqrt(a**2 + 2*sigma**2)
    eg = np.exp(gamma * tau)
    denom = (gamma + a)*(eg - 1) + 2*gamma
    B = 2*(eg - 1) / denom
    A = (2*gamma * np.exp((a + gamma)*tau/2) / denom)**(2*a*b/sigma**2)
    return A * np.exp(-B * r)

# Yield curve
r0 = 0.03
maturities = np.linspace(0.25, 30, 100)
P_vas = [vasicek_bond(r0, 0, T, 0.5, 0.04, 0.01) for T in maturities]
P_cir = [cir_bond(r0, 0, T, 0.5, 0.04, 0.06) for T in maturities]
R_vas = -np.log(P_vas) / maturities
R_cir = -np.log(P_cir) / maturities

print(f"Vasicek R(10y): {R_vas[39]:.4f}")
print(f"CIR     R(10y): {R_cir[39]:.4f}")
```

7.3 Heath–Jarrow–Morton (HJM) Framework

Theorem 7.10 (HJM Framework (1992))**.** *The HJM framework models instantaneous forward rates directly:*

$$df(t,T) = \alpha(t,T)\,dt + \sigma(t,T)\,dW_t.$$

The no-arbitrage condition imposes the ***HJM drift condition****:*

$$\alpha(t,T) = \sigma(t,T)\int_t^T \sigma(t,u)\,du.$$

Remark 7.11. The HJM framework is very general but typically leads to non-Markovian models (except for specific choices of σ). The Vasicek and Hull–White models are special cases of HJM.

CHAPTER 7. INTEREST RATE MODELS

7.4 Interest Rate Product Pricing

7.4.1 Caps and Floors

Definition 7.12 (Caplet)**.** A **caplet** on the period $[T_i, T_{i+1}]$ with strike rate K and notional N pays:

$$N\,\delta_i\,(L(T_i, T_i, T_{i+1}) - K)^+,$$

where $L(T_i, T_i, T_{i+1})$ is the LIBOR rate and $\delta_i = T_{i+1} - T_i$.

Theorem 7.13 (Black's Formula for Caplets)**.** *Under the Black model, the caplet price is:*

$$Caplet = N\,\delta_i\,P(0, T_{i+1})\big[L_0\Phi(d_1) - K\Phi(d_2)\big],$$

with $d_1 = \frac{\ln(L_0/K)+\sigma_i^2 T_i/2}{\sigma_i\sqrt{T_i}}$, $L_0 = L(0, T_i, T_{i+1})$.

7.4.2 Swaptions

Definition 7.14 (Swaption)**.** A **swaption** gives the right to enter into an interest rate swap at a future date. A payer swaption with strike K and maturity T_0 on a swap with dates $T_0, \dots, T_n$ has payoff:

$$\left(\sum_{i=0}^{n-1} \delta_i P(T_0, T_{i+1})\right)(S_{T_0} - K)^+,$$

where S_{T_0} is the par swap rate.

Short Rate Path Simulation (Vasicek)

```
def vasicek_paths(r0, a, b, sigma, T, N, M):
    """Simulate M short rate paths under Vasicek."""
    dt = T / N
    r = np.zeros((M, N + 1))
    r[:, 0] = r0
    for i in range(N):
        dW = np.sqrt(dt) * np.random.standard_normal(M)
        r[:, i+1] = r[:, i] + a*(b - r[:, i])*dt + sigma*dW
    return r

def cir_paths(r0, a, b, sigma, T, N, M):
    """Simulate M short rate paths under CIR (truncated Euler)."""
    dt = T / N
    r = np.zeros((M, N + 1))
    r[:, 0] = r0
    for i in range(N):
        dW = np.sqrt(dt) * np.random.standard_normal(M)
        r_pos = np.maximum(r[:, i], 0)
        r[:, i+1] = (r[:, i] + a*(b - r_pos)*dt
                     + sigma*np.sqrt(r_pos)*dW)
        r[:, i+1] = np.maximum(r[:, i+1], 0)
    return r
```

```
np.random.seed(42)
r_vas = vasicek_paths(0.03, 0.5, 0.04, 0.01, 10, 2520, 5)
r_cir = cir_paths(0.03, 0.5, 0.04, 0.06, 10, 2520, 5)
```

7.5 LIBOR Market Model (BGM/LMM)

Definition 7.15 (BGM/LMM Model)**.** The **LIBOR Market Model** (Brace–Gatarek–Musiela) models forward LIBOR rates $L_i(t) = L(t, T_i, T_{i+1})$ directly under the T_{i+1} forward measure:

$$dL_i(t) = \sigma_i(t)L_i(t)\, dW_t^{T_{i+1}}.$$

Under other measures, a drift term appears depending on correlations between forward rates.

7.6 Exercises

Exercise 7.1 (Yield Curve)**.**

1. Compute and plot the yield curve $T \mapsto R(0,T)$ for Vasicek and CIR with $r_0 = 0.03$, $a = 0.5$, $b = 0.04$, $\sigma_V = 0.01$, $\sigma_C = 0.06$.
2. Compare curve shapes (upward-sloping, downward-sloping, humped).
3. Study the impact of parameters a, b, σ on the shape.

Exercise 7.2 (Monte Carlo Bond Pricing)**.**

1. Price a 5-year zero-coupon bond by Monte Carlo under Vasicek: $P(0,T) = \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_0^T r_t\, dt}\right]$.
2. Compare with the analytical formula.
3. Repeat for the CIR model.

Exercise 7.3 (Caplet Pricing)**.**

1. Price a 1-year caplet using Black's formula.
2. Price the same caplet by Monte Carlo under the Hull–White model.
3. Compare the results.

Exercise 7.4 (Hull–White Calibration)**.**

1. Calibrate the Hull–White model to a given yield curve by determining $\theta(t)$.
2. Verify that the model zero-coupon prices match market prices.

Chapter 8

Portfolio Optimisation

In 1952, Harry Markowitz, then a doctoral student at the University of Chicago, published a fourteen-page paper that would revolutionize finance: “Portfolio Selection.” His idea is simple but profound: an investor should not choose assets individually, but optimize the *portfolio* as a whole, taking into account the correlations between returns. Diversification — “don’t put all your eggs in one basket” — thus received a rigorous mathematical foundation, expressed as a quadratic optimization problem under constraints. Markowitz would receive the Nobel Prize in Economics in 1990 for this work. This chapter develops mean-variance theory, the efficient frontier, Sharpe’s CAPM, and modern extensions including CVaR risk and robust optimization.

8.1 Markowitz Theory

8.1.1 Mean-Variance Framework

Definition 8.1 (Portfolio)**.** Consider n assets with random returns $R = (R_1, \dots, R_n)^\top$, expected return vector $\mu = \mathbb{E}[R] \in \mathbb{R}^n$, and covariance matrix $\Sigma = \operatorname{Cov}(R) \in \mathbb{R}^{n\times n}$ (symmetric positive definite).

A **portfolio** is defined by a weight vector $w = (w_1, \dots, w_n)^\top$ with $\mathbf{1}^\top w = 1$. The portfolio return is:

$$R_p = w^\top R, \quad \mathbb{E}[R_p] = w^\top \mu, \quad \operatorname{Var}(R_p) = w^\top \Sigma w.$$

8.1.2 Optimisation Problem

Definition 8.2 (Markowitz Problem)**.** The **Markowitz problem** (1952) seeks the minimum variance portfolio for a target return μ_p:

$$\min_w \frac{1}{2} w^\top \Sigma w \quad \text{s.t.} \quad w^\top \mu = \mu_p, \quad \mathbf{1}^\top w = 1.$$

Theorem 8.3 (Analytical Solution)**.** *The solution to the Markowitz problem is:*

$$w^* = \Sigma^{-1}\big[\lambda_1 \mu + \lambda_2 \mathbf{1}\big],$$

where the Lagrange multipliers λ_1, λ_2 are determined by the constraints. Setting:

$$A = \mathbf{1}^\top \Sigma^{-1} \mu, \quad B = \mu^\top \Sigma^{-1} \mu, \quad C = \mathbf{1}^\top \Sigma^{-1} \mathbf{1}, \quad D = BC - A^2, \tag{8.1}$$

we obtain:

$$\lambda_1 = \frac{C\mu_p - A}{D}, \quad \lambda_2 = \frac{B - A\mu_p}{D}.$$

Minimum Variance of the Optimal Portfolio

$$\sigma_p^2 = \frac{C\mu_p^2 - 2A\mu_p + B}{D}.$$

This is a parabola in μ_p in the (σ_p^2, μ_p) plane, or a hyperbola in the (σ_p, μ_p) plane.

Definition 8.4 (Efficient Frontier)**.** The **efficient frontier** is the upper branch of the hyperbola: the set of optimal portfolios for which it is impossible to increase return without increasing risk.

Definition 8.5 (Global Minimum Variance Portfolio (GMV))**.** The **GMV portfolio** minimises variance without a return constraint:

$$w_{\text{GMV}} = \frac{\Sigma^{-1}\mathbf{1}}{\mathbf{1}^\top\Sigma^{-1}\mathbf{1}}.$$

Its expected return is $\mu_{\text{GMV}} = A/C$ and its variance $\sigma_{\text{GMV}}^2 = 1/C$.

Markowitz Efficient Frontier

```python
import numpy as np
from scipy.optimize import minimize

def efficient_frontier(mu, Sigma, n_points=100):
    """Compute the efficient frontier."""
    n = len(mu)
    Sigma_inv = np.linalg.inv(Sigma)
    ones = np.ones(n)
    w_gmv = Sigma_inv @ ones / (ones @ Sigma_inv @ ones)
    mu_gmv = w_gmv @ mu

    mu_range = np.linspace(mu_gmv - 0.02, max(mu) + 0.05, n_points)
    sigmas = []
    weights = []

    for mu_target in mu_range:
        constraints = [
            {'type': 'eq', 'fun': lambda w: np.sum(w) - 1},
            {'type': 'eq', 'fun': lambda w, m=mu_target: w @ mu - m}
        ]
        result = minimize(
            lambda w: 0.5 * w @ Sigma @ w,
            np.ones(n)/n, method='SLSQP',
            constraints=constraints
        )
        sigmas.append(np.sqrt(result.fun * 2))
        weights.append(result.x)
    return mu_range, np.array(sigmas), np.array(weights)

# Example with 3 assets
mu = np.array([0.08, 0.12, 0.15])
Sigma = np.array([[0.04, 0.006, 0.002],
                  [0.006, 0.09, 0.009],
                  [0.002, 0.009, 0.16]])

mu_range, sigmas, weights = efficient_frontier(mu, Sigma)
print(f"GMV: sigma={sigmas.min():.4f}")
```

8.2 The CAPM

Theorem 8.6 (Capital Asset Pricing Model (Sharpe, 1964)). *At equilibrium, the expected return of any asset i satisfies:*

$$\mathbb{E}[R_i] - r = \beta_i(\mathbb{E}[R_M] - r),$$

*where $\beta_i = \mathrm{Cov}(R_i, R_M)/\mathrm{Var}(R_M)$ is the asset's **beta** relative to the market portfolio M, and r is the risk-free rate.*

Definition 8.7 (Sharpe Ratio). The **Sharpe ratio** of portfolio p is:

$$\mathrm{SR}_p = \frac{\mathbb{E}[R_p] - r}{\sigma_p}.$$

The tangency portfolio (intersection of the CML with the efficient frontier) maximises the Sharpe ratio.

Tangency Portfolio (Maximum Sharpe)

```python
def tangent_portfolio(mu, Sigma, rf):
    """Tangency portfolio maximising the Sharpe ratio."""
    n = len(mu)
    Sigma_inv = np.linalg.inv(Sigma)
    excess = mu - rf
    w_tan = Sigma_inv @ excess / (np.ones(n) @ Sigma_inv @ excess)
    mu_tan = w_tan @ mu
    sigma_tan = np.sqrt(w_tan @ Sigma @ w_tan)
    sharpe = (mu_tan - rf) / sigma_tan
    return w_tan, mu_tan, sigma_tan, sharpe

rf = 0.03
w_tan, mu_tan, sig_tan, sr = tangent_portfolio(mu, Sigma, rf)
print(f"Tangency portfolio: w = {w_tan}")
print(f"Return: {mu_tan:.4f}, Risk: {sig_tan:.4f}")
print(f"Sharpe ratio: {sr:.4f}")
```

CHAPTER 8. PORTFOLIO OPTIMISATION

8.3 Black–Litterman Model

Definition 8.8 (Black–Litterman Model (1992))**.** The model combines equilibrium returns (CAPM) $\Pi = \delta\Sigma w_M$ with investor **views** $P\mu = Q + \epsilon$, $\epsilon \sim \mathcal{N}(0, \Omega)$, to obtain posterior returns:

$$\hat{\mu} = \left[(\tau\Sigma)^{-1} + P^\top\Omega^{-1}P\right]^{-1}\left[(\tau\Sigma)^{-1}\Pi + P^\top\Omega^{-1}Q\right].$$

The optimal portfolio then uses $\hat{\mu}$ in the Markowitz optimisation.

Advantage of Black–Litterman

Black–Litterman solves two classical Markowitz problems: (1) extreme weights due to sensitivity to estimated returns, and (2) lack of connection to market equilibrium. Views are expressed intuitively and the model generates more stable portfolios.

8.4 Practical Constraints

Remark 8.9. In practice, additional constraints are often imposed:

- No short selling: $w_i \geq 0$.
- Concentration limits: $w_i \leq w_{\max}$.
- Sector or liquidity constraints.
- Transaction costs (dynamic optimisation).

Optimisation with Constraints

```python
def efficient_frontier_constrained(mu, Sigma, rf, n_points=50):
    """Efficient frontier with w >= 0 (no short selling)."""
    n = len(mu)
    bounds = [(0, 1) for _ in range(n)]
    mu_range = np.linspace(min(mu), max(mu), n_points)
    results = []

    for mu_target in mu_range:
        constraints = [
            {'type': 'eq', 'fun': lambda w: np.sum(w) - 1},
            {'type': 'eq', 'fun': lambda w, m=mu_target: w @ mu - m}
        ]
        res = minimize(lambda w: w @ Sigma @ w, np.ones(n)/n,
                       method='SLSQP', bounds=bounds,
                       constraints=constraints)
        if res.success:
            results.append((mu_target, np.sqrt(res.fun), res.x))

    return results
```

8.5 Robust Optimisation

Definition 8.10 (Robust Optimisation)**.** **Robust optimisation** accounts for uncertainty in estimated parameters. We solve:

$$\min_{w} \max_{\mu \in \mathcal{U}} \left\{ -w^\top \mu + \lambda\, w^\top \Sigma w \right\},$$

where $\mathcal{U}$ is an uncertainty set around $\hat{\mu}$. For $\mathcal{U} = \{\mu : \|\mu - \hat{\mu}\|_{\Sigma^{-1}} \le \kappa\}$, the problem reduces to:

$$\min_{w} \left\{ -w^\top \hat{\mu} + \kappa \sqrt{w^\top \Sigma w} + \lambda\, w^\top \Sigma w \right\}.$$

8.6 Parameter Estimation

Estimation Errors

Expected returns μ are notoriously difficult to estimate. With T observations, the standard error of $\hat{\mu}_i$ is $\sigma_i/\sqrt{T}$, often of the same order as the signal. The covariance matrix Σ is better estimated but can be singular or ill-conditioned when $n \approx T$.

Proposition 8.11 (Ledoit–Wolf Estimator)**.** The Ledoit–Wolf (2004) **shrinkage estimator** regularises the covariance matrix:

$$\hat{\Sigma}_{\text{LW}} = (1 - \alpha)\hat{\Sigma} + \alpha\, \nu I,$$

where $\alpha \in [0, 1]$ and $\nu = \operatorname{tr}(\hat{\Sigma})/n$ are determined analytically to minimise the quadratic loss.

8.7 Exercises

Exercise 8.1 (Efficient Frontier)**.**

1. Compute and plot the efficient frontier for 5 assets.
2. Identify the GMV portfolio and the tangency portfolio.
3. Plot the Capital Market Line.
4. Compare frontiers with and without no-short-selling constraints.

Exercise 8.2 (CAPM and Beta)**.**

1. Estimate betas of 5 stocks by linear regression against a market index.
2. Verify the CAPM relation by plotting average historical return against beta.
3. Compute Jensen's alpha for each stock.

Exercise 8.3 (Black–Litterman)**.**

1. Implement the Black–Litterman model.
2. Starting from equilibrium returns, add the view "asset 1 will outperform asset 2 by 2%".
3. Compare weights before and after incorporating the view.

Exercise 8.4 (Robust Estimation).

1. Simulate 250 days of returns for 10 assets and estimate $\hat{\mu}$ and $\hat{\Sigma}$.
2. Compare optimal portfolios using $\hat{\Sigma}$ and $\hat{\Sigma}_{\mathrm{LW}}$.
3. Perform a backtest and measure out-of-sample performance.

Chapter 9

Risk Measures

9.1 Introduction

After the financial crises of the 1990s — the Orange County debacle (1994), the Barings collapse (1995), the LTCM crisis (1998) — the question of measuring risk became central. J.P. Morgan popularized *Value at Risk* (VaR) in 1994 with its RiskMetrics system: a single number summarizing the maximum probable loss over a given horizon at a given confidence level. But Philippe Artzner and colleagues showed in 1999 that VaR is not *coherent*: it violates subadditivity, meaning that diversifying a portfolio can, paradoxically, increase measured VaR. *Expected Shortfall* (CVaR) corrects this flaw. This chapter traces this history, from empirical measures to theoretical axioms, through copulas and stress testing.

Intuition

A risk measure answers the question: "How much can I lose at most, with a given probability, over a given horizon?" VaR answers with a quantile, but says nothing about the severity of losses beyond that threshold. Expected Shortfall fills this gap by averaging losses in the tail of the distribution.

9.2 Value at Risk (VaR)

Definition 9.1 (Value at Risk)**.** Let L be the random variable representing the portfolio loss over a horizon Δt. The **Value at Risk** at confidence level $\alpha \in (0,1)$ is

$$\mathrm{VaR}_\alpha(L) = \inf\{l \in \mathbb{R} : \mathbb{P}(L > l) \leq 1-\alpha\} = F_L^{-1}(\alpha),$$

where F_L^{-1} denotes the generalised quantile of L.

Example 9.2 (Gaussian VaR)**.** If $L \sim \mathcal{N}(\mu, \sigma^2)$, then $\mathrm{VaR}_\alpha = \mu + \sigma\,\Phi^{-1}(\alpha)$, where Φ is the standard normal CDF. For $\alpha = 0.99$: $\mathrm{VaR}_{99\%} = \mu + 2.326\,\sigma$.

Proposition 9.3 (VaR computation methods)**.**

1. **Historical**: $\mathrm{VaR}_\alpha =$ empirical quantile of past returns.
2. **Parametric (variance-covariance)**: assumes a distribution (often Gaussian) and uses estimated μ, σ.
3. **Monte Carlo**: simulate scenarios and compute the quantile of simulated losses.

Limitations of VaR

VaR does not satisfy sub-additivity in general: one can have $\mathrm{VaR}_\alpha(L_1 + L_2) > \mathrm{VaR}_\alpha(L_1) + \mathrm{VaR}_\alpha(L_2)$. This means diversification can *appear* to increase risk, which is a major conceptual flaw.

9.3 Expected Shortfall (CVaR)

Definition 9.4 (Expected Shortfall)**.** The **Expected Shortfall** (or CVaR, Conditional Value at Risk) at level α is

$$\mathrm{ES}_\alpha(L) = \frac{1}{1-\alpha}\int_\alpha^1 \mathrm{VaR}_u(L)\,\mathrm{d}u.$$

If F_L is continuous, $\mathrm{ES}_\alpha(L) = \mathbb{E}[L \mid L \geq \mathrm{VaR}_\alpha(L)]$.

Example 9.5 (Gaussian ES)**.** If $L \sim \mathcal{N}(\mu, \sigma^2)$:

$$\mathrm{ES}_\alpha = \mu + \sigma\,\frac{\phi(\Phi^{-1}(\alpha))}{1-\alpha},$$

where ϕ is the standard normal density. For $\alpha = 0.99$: $\mathrm{ES}_{99\%} \approx \mu + 2.665\,\sigma$.

Theorem 9.6 (Dual representation of ES)**.**

$$\mathrm{ES}_\alpha(L) = \sup_{Q\in\mathcal{Q}_\alpha} \mathbb{E}_Q[L],$$

where $\mathcal{Q}_\alpha = \{Q \ll \mathbb{P} : \mathrm{d}Q/\mathrm{d}\mathbb{P} \leq 1/(1-\alpha)\}$. *This shows that ES is the worst-case expectation over an ambiguity set.*

9.4 Coherent Risk Measures

Definition 9.7 (Coherent risk measure)**.** A functional $\rho : \mathcal{X} \to \mathbb{R}$ is a **coherent risk measure** (Artzner, Delbaen, Eber, Heath, 1999) if it satisfies:

1. **Monotonicity**: $L_1 \leq L_2$ a.s. $\implies \rho(L_1) \leq \rho(L_2)$.
2. **Translation invariance**: $\rho(L + c) = \rho(L) + c$ for all $c \in \mathbb{R}$.
3. **Positive homogeneity**: $\rho(\lambda L) = \lambda\rho(L)$ for all $\lambda > 0$.
4. **Sub-additivity**: $\rho(L_1 + L_2) \leq \rho(L_1) + \rho(L_2)$.

Theorem 9.8 (ADEH, 1999)**.** 1. *Expected Shortfall is a* ***coherent*** *risk measure.*

2. *VaR is* ***not*** *coherent (it violates sub-additivity).*
3. *Every coherent risk measure admits a representation* $\rho(L) = \sup_{Q\in\mathcal{Q}} \mathbb{E}_Q[L]$ *for a closed convex set* $\mathcal{Q}$ *of probability measures.*

VaR vs ES Comparison

Property	VaR	ES
Sub-additivity	No	Yes
Coherence	No	Yes
Tail sensitivity	No	Yes
Basel III+ regulation	No	Yes
Ease of backtesting	Yes	Difficult

9.5 Stress Testing

Definition 9.9 (Stress test)**.** A **stress test** evaluates portfolio losses under predetermined or historical extreme scenarios:

$$L_{\text{stress}} = \sum_{i=1}^{n} \Delta_i \cdot S_i^{\text{stress}},$$

where Δ_i are sensitivities and S_i^{stress} are shocks applied to risk factors.

Proposition 9.10 (Types of stress tests)**.**

1. **Historical**: reproduce past crises (2008, 2020).
2. **Hypothetical**: constructed scenarios (300 bp rate shock, sector collapse).
3. **Reverse stress tests**: find the scenario leading to a given critical loss.

9.6 Copulas and Dependence Modelling

Definition 9.11 (Copula)**.** A **copula** $C : [0,1]^d \to [0,1]$ is a joint CDF with uniform marginals on $[0,1]$. It captures the dependence structure independently of the marginal distributions.

Theorem 9.12 (Sklar, 1959)**.** *For any joint distribution F with marginals $F_1, \dots, F_d$, there exists a copula C such that*

$$F(x_1, \dots, x_d) = C(F_1(x_1), \dots, F_d(x_d)).$$

If the F_i are continuous, C is unique.

Example 9.13 (Gaussian copula)**.** The **Gaussian copula** with correlation matrix Σ is

$$C_\Sigma^{\text{Ga}}(u_1, \dots, u_d) = \Phi_\Sigma(\Phi^{-1}(u_1), \dots, \Phi^{-1}(u_d)),$$

where Φ_Σ is the multivariate normal CDF.

Definition 9.14 (Clayton copula)**.** The **Clayton copula** (bivariate) is

$$C_\theta(u, v) = (u^{-\theta} + v^{-\theta} - 1)^{-1/\theta}, \quad \theta > 0.$$

It models strong lower tail dependence (co-crash risk).

Definition 9.15 (Tail dependence)**.** The **lower tail dependence coefficient** is

$$\lambda_L = \lim_{u \to 0^+} \frac{C(u,u)}{u}.$$

For the Gaussian copula, $\lambda_L = 0$ (unless $\rho = 1$). For the Clayton copula, $\lambda_L = 2^{-1/\theta} > 0$.

Remark 9.16. The Gaussian copula underestimates tail dependence. This was one of the causes of the 2008 subprime crisis, where CDO models used Li's (2000) Gaussian copula without accounting for extreme dependence.

VaR and ES computation via Monte Carlo

```
import numpy as np
from scipy.stats import norm

np.random.seed(42)
n_sim = 100_000
mu, sigma = 0.0, 0.02
alpha = 0.99

returns = np.random.normal(mu, sigma, n_sim)
losses = -returns

var_99 = np.percentile(losses, 100 * alpha)
es_99 = losses[losses >= var_99].mean()

print(f"VaR 99%: {var_99:.4f}")
print(f"ES 99%:  {es_99:.4f}")
print(f"Analytical VaR: {-mu + sigma * norm.ppf(alpha):.4f}")
```

9.7 Exercises

Exercise 9.1 (Portfolio VaR)**.** A portfolio worth EUR 10M has daily returns $\mathcal{N}(0.05\%, (1.5\%)^2)$. Compute the 99% VaR at 1 day and 10 days (square root of time rule).

Exercise 9.2 (Sub-additivity violation)**.** Construct an explicit example of two random variables L_1, L_2 such that $\mathrm{VaR}_{0.95}(L_1 + L_2) > \mathrm{VaR}_{0.95}(L_1) + \mathrm{VaR}_{0.95}(L_2)$. *Hint*: use concentrated Bernoulli losses.

Exercise 9.3 (Clayton copula)**.** Simulate 10,000 pairs (U, V) from a Clayton copula with $\theta = 2$. Estimate the lower tail dependence coefficient and compare with the theoretical value $\lambda_L = 2^{-1/2}$.

Exercise 9.4 (Convex risk measures)**.** Show that every coherent risk measure is convex. Give an example of a convex risk measure that is not coherent (positive homogeneity violated).

Exercise 9.5 (Stress test)**.** Design a historical stress test based on the 2008 crisis for a portfolio containing 60% equities and 40% bonds. Compute the loss under this scenario and compare with the 99% VaR.

Exercise 9.6 (Backtesting)**.** Explain the Kupiec test for VaR backtesting. If over 250 trading days, 5 VaR exceedances are observed at the 99% level, is the VaR correctly calibrated at the 5% significance level?

Chapter 10

Exotic Derivatives

10.1 Introduction

Exotic options extend vanilla payoffs (European calls and puts) to meet specific hedging or speculation needs. This chapter presents the main families of exotics and their pricing methods.

Intuition

Vanilla options are elementary building blocks. Exotic options combine these blocks or add conditions (barriers, averages, extrema) to create tailor-made payoff profiles. Their pricing often requires advanced numerical methods since closed-form formulas are rare.

10.2 Barrier Options

Definition 10.1 (Barrier option)**.** A **barrier option** is an option whose existence or activation depends on the underlying S_t crossing a level B during the option's life:

- **Knock-out**: the option ceases to exist if S_t reaches B.
- **Knock-in**: the option only comes into existence if S_t reaches B.

Combined with direction (up/down), this gives 8 fundamental types.

Theorem 10.2 (In-out parity)**.** *For any barrier B, the price of a vanilla call (or put) satisfies:*

$$V_{\text{vanilla}} = V_{\text{knock-in}} + V_{\text{knock-out}}.$$

Theorem 10.3 (Merton's formula for a down-and-out call)**.** *In the Black-Scholes model, for a down-and-out call with barrier $B < S_0$ and $B < K$:*

$$C_{\text{do}} = C_{\text{BS}} - \left(\frac{B}{S_0}\right)^{2\lambda} C_{\text{BS}}\left(\frac{B^2}{S_0}, K, r, \sigma, T\right),$$

where $\lambda = \frac{r-\sigma^2/2}{\sigma^2} + \frac{1}{2}$ and C_{BS} is the Black-Scholes price.

Hedging risk of barriers

The delta of a barrier option diverges as the underlying approaches the barrier, making dynamic hedging extremely costly. In practice, discrete barriers (observed daily) are used with a Broadie-Glasserman-Kou correction.

10.3 Asian Options

Definition 10.4 (Asian option)**.** An **Asian option** has a payoff depending on the average of the underlying:

- **Average price**: payoff $= (\bar{S}-K)^+$ (call) where $\bar{S} = \frac{1}{T}\int_0^T S_t \, \mathrm{d}t$ (continuous average) or $\bar{S} = \frac{1}{n}\sum_{i=1}^n S_{t_i}$ (discrete average).

- **Average strike**: payoff $= (S_T - \bar{S})^+$.

Proposition 10.5 (Properties of Asian options)**.** 1. Averaging reduces effective volatility: $\mathrm{Var}(\bar{S}) < \mathrm{Var}(S_T)$, so Asian options are cheaper than the corresponding vanillas.

2. The distribution of $\bar{S}$ is not lognormal, even when S_t follows geometric Brownian motion.

3. No exact closed-form formula exists in Black-Scholes for the arithmetic average (unlike the geometric average).

Theorem 10.6 (Geometric Asian price)**.** *For the continuous geometric average $\bar{S}_G = \exp\left(\frac{1}{T}\int_0^T \ln S_t \, \mathrm{d}t\right)$, the call price is given by a modified Black-Scholes formula with*

$$\hat{\sigma} = \frac{\sigma}{\sqrt{3}}, \qquad \hat{r} = \frac{1}{2}\left(r - \frac{\sigma^2}{6}\right).$$

10.4 Lookback Options

Definition 10.7 (Lookback option)**.** A **lookback option** depends on the extremum of the underlying:

- **Floating strike lookback call**: payoff $= S_T - \min_{0\le t\le T} S_t$.

- **Fixed strike lookback call**: payoff $= (\max_{0\le t\le T} S_t - K)^+$.

Theorem 10.8 (Goldman, Sosin, Gatto (1979))**.** *In Black-Scholes, the floating strike lookback call price admits a closed-form expression involving S_0, $S_{\min}$, r, σ, T and the normal CDF Φ. In particular, the lookback call is always more expensive than the vanilla call since the holder benefits from the best possible exercise timing.*

10.5 Binary (Digital) Options

Definition 10.9 (Binary option)**.** A **binary** (or digital) option has a discontinuous payoff:

- **Cash-or-nothing call**: payoff $= Q \cdot \nVdash_{S_T>K}$ (pays Q if $S_T > K$).
- **Asset-or-nothing call**: payoff $= S_T \cdot \nVdash_{S_T>K}$.

Proposition 10.10 (Black-Scholes prices)**.**

$$V_{\text{cash}} = Q\, e^{-rT}\, \Phi(d_2),$$
$$V_{\text{asset}} = S_0\, \Phi(d_1),$$

where d_1, d_2 are the standard Black-Scholes quantities. A vanilla call decomposes as: $C = V_{\text{asset}} - K\, e^{-rT}\Phi(d_2)$.

Remark 10.11. The delta of a binary option diverges near the strike at expiry, making hedging very difficult. In practice, they are replicated by a tight call spread.

10.6 Monte Carlo Pricing

Definition 10.12 (Monte Carlo estimator)**.** The price of an exotic option with payoff $h(S_{t_1}, \dots, S_{t_n})$ is

$$V_0 = e^{-rT}\, \mathbb{E}^{\mathbb{Q}}[h(S_{t_1}, \dots, S_{t_n})] \approx \frac{e^{-rT}}{M} \sum_{j=1}^{M} h(S_{t_1}^{(j)}, \dots, S_{t_n}^{(j)}),$$

where paths $(S^{(j)})$ are simulated under the risk-neutral measure.

Proposition 10.13 (Variance reduction techniques)**.**

1. **Antithetic variates**: for each path W_t, also use $-W_t$.
2. **Control variate**: use the known geometric average price to correct the arithmetic average estimator.
3. **Importance sampling**: change measure to sample rare events more efficiently.

Path Simulation under $\mathbb{Q}$

In the Black-Scholes model:

$$S_{t_{i+1}} = S_{t_i} \exp\left[\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t}\, Z_i\right], \quad Z_i \sim \mathcal{N}(0,1).$$

10.7 Tree Methods for Exotics

Definition 10.14 (Trinomial tree)**.** A **trinomial tree** discretises the underlying with three moves per step: $S \to Su$, $S \to Sm$, $S \to Sd$. The risk-neutral probabilities (p_u, p_m, p_d) are calibrated to match the first two moments of $\ln S$.

Proposition 10.15 (Barrier tree adjustment)**.** For a barrier option, the tree is adjusted so that a node coincides exactly with the barrier B, eliminating specification errors due to discretisation.

Asian option pricing by Monte Carlo

```python
import numpy as np

S0, K, r, sigma, T = 100, 100, 0.05, 0.2, 1.0
n_steps, n_paths = 252, 100_000
dt = T / n_steps

np.random.seed(42)
Z = np.random.randn(n_paths, n_steps)
S = np.zeros((n_paths, n_steps + 1))
S[:, 0] = S0

for i in range(n_steps):
    S[:, i+1] = S[:, i] * np.exp((r - 0.5*sigma**2)*dt
                                  + sigma*np.sqrt(dt)*Z[:, i])

S_mean = S[:, 1:].mean(axis=1)
payoff = np.maximum(S_mean - K, 0)
price = np.exp(-r*T) * payoff.mean()
stderr = np.exp(-r*T) * payoff.std() / np.sqrt(n_paths)
print(f"Asian call price: {price:.4f} +/- {1.96*stderr:.4f}")
```

10.8 Exercises

Exercise 10.1 (In-out parity)**.** Prove the in-out parity relation for barrier options using the no-arbitrage principle.

Exercise 10.2 (Asian vs vanilla)**.** Show analytically that the price of an Asian call (average price) is less than the corresponding vanilla call price.

Exercise 10.3 (Lookback pricing)**.** Implement a Monte Carlo pricer for a floating strike lookback option and compare with the Goldman-Sosin-Gatto closed-form formula.

Exercise 10.4 (Digital replication)**.** Show that a cash-or-nothing call with payoff $Q\mathbb{1}_{S_T>K}$ can be approximated by a call spread: $(C(K) - C(K+\epsilon))/\epsilon \times Q$, and compute the limit as $\epsilon \to 0$.

Exercise 10.5 (Antithetic variates)**.** Show that the antithetic variate Monte Carlo estimator has lower variance than the standard estimator if and only if payoffs are negatively correlated under the inversion $W \to -W$.

Exercise 10.6 (Discrete barrier correction)**.** For a down-and-out call with a daily observed discrete barrier, apply the Broadie-Glasserman-Kou correction: $B_{\text{eff}} = B\, e^{-\beta\sigma\sqrt{\Delta t}}$ with $\beta \approx 0.5826$, and compare with the continuous barrier price.

Chapter 11

Machine Learning in Finance

11.1 Introduction

Machine learning (ML) is transforming quantitative finance by enabling the discovery of nonlinear structures in market data. This chapter presents the main applications: algorithmic trading, neural network option pricing, reinforcement learning for portfolio management, NLP-based sentiment analysis, and associated risks and regulations.

Intuition

Classical models (Black-Scholes, GARCH) impose parametric structure. ML allows the data to "speak for itself" by learning complex relationships directly, at the cost of increased overfitting risk and reduced interpretability.

11.2 Algorithmic Trading

Definition 11.1 (Algorithmic trading strategy)**.** An **algorithmic trading strategy** is a systematic rule $\delta_t = g(x_t; \theta)$ determining the position δ_t at time t as a function of observable features x_t and parameters θ optimised on historical data.

Proposition 11.2 (Classical features)**.** Typical predictors include:

1. **Technical indicators**: moving averages, RSI, Bollinger bands, MACD.
2. **Microstructure**: order flow imbalance, bid-ask spread, volume.
3. **Fundamental features**: financial ratios, earnings surprises.
4. **Alternative data**: satellite imagery, web traffic, social media sentiment.

Overfitting and data snooping

Extensive backtesting on historical data produces strategies that perform perfectly in-sample but fail out-of-sample. Remedies include: time-series cross-validation (walk-forward), Bonferroni correction for multiple testing, and holding out fresh data for the final test.

CHAPTER 11. MACHINE LEARNING IN FINANCE

Theorem 11.3 (Generalisation bound)**.** *For a hypothesis class $\mathcal{H}$ of VC dimension d_{VC}, with probability at least $1-\delta$ on a sample of size n:*

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{d_{\mathrm{VC}}(\ln(2n/d_{\mathrm{VC}})+1)+\ln(4/\delta)}{n}},$$

where $R(h)$ is the true risk and $\hat{R}(h)$ is the empirical risk.

11.3 Neural Networks for Option Pricing

Definition 11.4 (Feedforward neural network)**.** A **feedforward neural network** with L layers is a function

$$f_\theta(x) = W_L\,\sigma(W_{L-1}\,\sigma(\cdots\sigma(W_1x+b_1)\cdots)+b_{L-1})+b_L,$$

where σ is a nonlinear activation function (ReLU, tanh, etc.), W_ℓ are weight matrices, and b_ℓ are biases.

Theorem 11.5 (Universal approximation)**.** *For any continuous function $g : \mathbb{R}^d \to \mathbb{R}$ on a compact set K and any $\epsilon > 0$, there exists a single-hidden-layer feedforward network with sigmoid activation such that* $\sup_{x\in K}|f_\theta(x) - g(x)| < \epsilon$.

Proposition 11.6 (Neural network pricing)**.** The approach consists of:

1. Generate N parameter sets $(S_0^{(i)}, K^{(i)}, T^{(i)}, \sigma^{(i)}, r^{(i)})$ and compute prices $V^{(i)}$ via Black-Scholes or Monte Carlo.
2. Train a network f_θ minimising $\frac{1}{N}\sum_{i=1}^{N}(f_\theta(x^{(i)}) - V^{(i)})^2$.
3. At inference, f_θ yields prices and Greeks ($\Delta = \partial f_\theta/\partial S$) near-instantaneously.

Remark 11.7. Neural networks accelerate pricing by a factor of 10^3 to 10^5 compared to Monte Carlo, which is crucial for XVA computation (CVA, DVA, FVA) on portfolios containing thousands of trades.

11.4 Reinforcement Learning for Portfolio Management

Definition 11.8 (Markov Decision Process (MDP))**.** An **MDP** is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s,a)$ is the transition dynamics, $R(s,a)$ is the reward, and $\gamma \in [0,1)$ is the discount factor.

Definition 11.9 (Policy and value function)**.** A **policy** $\pi : \mathcal{S} \to \mathcal{A}$ specifies the action at each state. The **value function** is $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t) \mid s_0 = s]$. The optimal policy satisfies $V^*(s) = \max_\pi V^\pi(s)$.

Theorem 11.10 (Bellman equation)**.** *The optimal value function satisfies*

$$V^*(s) = \max_{a\in\mathcal{A}}\left[R(s,a) + \gamma\sum_{s'}P(s'|s,a)\,V^*(s')\right].$$

Proposition 11.11 (Application to portfolio management).

- **State**: $s_t = (w_t, S_t, \text{features}_t)$ (wealth, prices, indicators).
- **Action**: $a_t = (\delta_1, \ldots, \delta_d)$ (allocations across d assets).
- **Reward**: $R_t = \ln(w_{t+1}/w_t)$ (log return) or differential Sharpe ratio.
- **Algorithms**: DDPG, PPO, SAC for continuous action spaces.

11.5 NLP and Sentiment Analysis

Definition 11.12 (Financial sentiment analysis)**. Sentiment analysis** in finance consists of extracting a bullish/bearish signal from textual sources: press releases, analyst reports, tweets, earnings call transcripts.

Proposition 11.13 (NLP pipeline for finance).

1. **Collection**: Reuters/Bloomberg feeds, Twitter/Reddit APIs.
2. **Preprocessing**: tokenisation, stop word removal, lemmatisation.
3. **Model**: FinBERT (BERT fine-tuned on financial text) or specialised GPT.
4. **Aggregation**: weighted average sentiment score by source and recency.
5. **Trading signal**: combination with quantitative features.

ML Strategy Performance Metrics

$$\text{Sharpe Ratio} = \frac{\mathbb{E}[R_p] - r_f}{\sqrt{\text{Var}(R_p)}}$$
$$\text{Sortino Ratio} = \frac{\mathbb{E}[R_p] - r_f}{\sqrt{\mathbb{E}[\min(R_p - r_f, 0)^2]}}$$
$$\text{Maximum Drawdown} = \max_t \frac{\max_{s\leq t} W_s - W_t}{\max_{s\leq t} W_s}$$

11.6 Risks and Regulation

Definition 11.14 (Model risk)**. Model risk** in ML includes:

- **Overfitting**: the model memorises noise.
- **Distribution shift**: future data differs from training data (regime changes).
- **Black box**: inability to explain decisions (interpretability problem).
- **Adversarial attacks**: deliberate manipulation of inputs to deceive the model.

Remark 11.15. The EU AI Act (2024) classifies credit scoring and algorithmic trading systems as "high risk," imposing transparency, documentation, and human oversight requirements.

Option pricing with a neural network

```python
import numpy as np
import torch
import torch.nn as nn
from scipy.stats import norm

def bs_call(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + sigma**2/2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma*np.sqrt(T)
    return S*norm.cdf(d1) - K*np.exp(-r*T)*norm.cdf(d2)

n_train = 50_000
S = np.random.uniform(50, 150, n_train)
K = np.random.uniform(50, 150, n_train)
T = np.random.uniform(0.1, 2.0, n_train)
sigma = np.random.uniform(0.1, 0.5, n_train)
prices = bs_call(S, K, T, 0.05, sigma)

X = torch.tensor(np.column_stack([S, K, T, sigma]), dtype=torch.float32)
y = torch.tensor(prices, dtype=torch.float32).unsqueeze(1)

model = nn.Sequential(
    nn.Linear(4, 64), nn.ReLU(),
    nn.Linear(64, 64), nn.ReLU(),
    nn.Linear(64, 1))
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

for epoch in range(200):
    pred = model(X)
    loss = nn.MSELoss()(pred, y)
    optimizer.zero_grad(); loss.backward(); optimizer.step()
print(f"Final MSE: {loss.item():.6f}")
```

11.7 Exercises

Exercise 11.1 (Feature engineering)**.** For an S&P 500 trading strategy, propose 10 relevant features. Discuss look-ahead bias and multicollinearity issues.

Exercise 11.2 (Greeks via autodiff)**.** Train a neural network to price European calls. Compute delta via automatic differentiation and compare with the analytical formula.

Exercise 11.3 (RL for allocation)**.** Implement a simple DQN agent for allocation between a risky asset (return $\mathcal{N}(8\%, 20\%)$) and a risk-free asset (3%). Compare the learned policy with Merton's rule.

Exercise 11.4 (Sentiment analysis)**.** Collect 1000 financial news headlines and classify them as positive/negative/neutral. Measure the correlation between the aggregated sentiment score and next-day returns.

Exercise 11.5 (Temporal validation)**.** Explain why standard k-fold cross-validation is inappropriate in finance. Implement walk-forward validation with a 2-year training window and a 6-month test window.

Exercise 11.6 (Interpretability)**.** Apply Shapley values (SHAP) to an XGBoost model for return prediction. Identify the most important features and discuss the stability of explanations over time.

Bibliography

[1] Hull, J.C., *Options, Futures, and Other Derivatives*, 11th ed., Pearson, 2022.

[2] Shreve, S.E., *Stochastic Calculus for Finance II*, Springer, 2004.

[3] Brigo, D. and Mercurio, F., *Interest Rate Models*, 2nd ed., Springer, 2006.

[4] Björk, T., *Arbitrage Theory in Continuous Time*, 4th ed., Oxford, 2020.

[5] Lamberton, D. and Lapeyre, B., *Introduction au calcul stochastique appliqué à la finance*, 3rd ed., Ellipses, 2012.