uc3m | Universidad Carlos III de Madrid

Master Degree in Computational and Applied Mathematics

2024-2025

Master Thesis

Leveraging Physics-Informed Neural Networks for Option Pricing Problems

Jose Pedro Melo Olivares

Pedro Echeverria, Ph.D

Madrid, 2025

SUMMARY

This thesis investigates the application of Physics-Informed Neural Networks to the pricing of financial derivatives governed by partial differential equations. Traditional numerical methods—such as finite-difference schemes and Monte Carlo simulations— face limitations when confronted with high-dimensional models, early-exercise features and complex market dynamics. PINNs offer a mesh-free alternative: by embedding the pricing partial differential equation (PDE) directly into the training loss of a neural network, they deliver efficient and flexible approximations.

We employ PINNs to price a range of derivatives—including European and American options, basket options and swaptions—under both the classical Black-Scholes assumptions and extensions with stochastic volatility and stochastic interest rates. Empirical results show that PINNs recover accurate pricing surfaces with low relative error, scale to higher dimensions, and provide significant speed-ups at inference time versus Monte Carlo methods. Training time and accuracy near payoff discontinuities remain challenges. Overall, this work highlights both the potential and the limits of PINNs in quantitative finance and outlines directions for future research on scalable, data-driven pricing methodologies.

Keywords: Physics-Informed Neural Networks, quantitative finance, option pricing, derivatives.

DEDICATION

I would like to thank both Francisco Gomez and Pedro Echeverria from BBVA for their guidance and support throughout this project. Their insights and expertise have been invaluable in shaping the direction of this work and providing practical context to the theoretical concepts explored in this thesis. Also, I would like to express my gratitude to the UC3M math faculty, and in particual, Francisco Bernal, for their assistance and support during my studies. Their dedication to teaching and mentoring has greatly contributed to my academic growth and understanding of the subject matter. Finally, I would like to dedicate this work to my partner, Paula, whose unwavering support and encouragement have been a constant source of motivation throughout this journey.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation

The rapid progress in machine learning over the past decade has transformed fields such as speech recognition, image recognition and object detection [1]. The financial industry likewise seeks innovative, accurate computational methods to price complex products. Derivatives—whose value depends on underlying assets [2]—are central to risk management, speculative trading and hedging.

Derivatives reference a wide range of assets, including equities, bonds, commodities and indices [3]. Accurate pricing informs both investment decisions and risk control. Historically, practitioners relied on Monte Carlo (MC) simulation [4] or on partial differential equations (PDEs) such as Black-Scholes [5] for this task. While PDE-based methods yield theoretical insight, their application is curtailed by the curse of dimensionality [6].

The emergence of Physics-Informed Neural Networks (PINNs) [7] offers a novel approach. Exploiting the universal-approximation capability of neural networks, PINNs embed the governing PDE directly into the learning objective. This allows the network to learn from the mathematical structure of the problem, ensuring consistency with financial theory while reducing training time compared to traditional data-driven approaches. This method is particularly attractive to institutions that require real-time pricing for complex derivatives.

PINNs can also generate complete price surfaces instantaneously, enabling immediate sensitivity analysis and risk assessment across broad market scenarios—e.g. valuation adjustments (XVA) or internal-model frameworks.

Recent contributions have explored the use of PINNs in finance, notably for pricing European options under Black-Scholes dynamics, and for American options in both univariate and multivariate settings [8], [9]. These works represent important early steps in adapting PINNs to financial applications. However, they typically focus on specific models or contract types and often remain limited in dimensionality or scope.

This master's thesis explores and validates the use of PINNs in derivative pricing. Guided by professionals at BBVA, the research assesses whether PINNs can efficiently and accurately price complex derivatives through PDEs and thus be adopted in practice.

1.2. Objectives

The overarching aim of this thesis is two-fold. First, we investigate the suitability of PINNs for pricing financial derivatives with a variety of underlying dynamics and industry

products. Second, we evaluate whether PINNs can serve as a competitive alternative to traditional numerical procedures—most notably Monte-Carlo simulation and finite-difference schemes.

More concretely, the work pursues three specific objectives:

- **Theoretical review:** present the foundations of derivative pricing and derive the associated partial-differential equations (PDEs) for a representative set of products.
- **Classical benchmarks:** summarise the main numerical techniques used to solve these PDEs—finite differences and Monte-Carlo—and highlight their practical limitations.
- **PINN assessment:** measure the accuracy, computational efficiency and scalability of PINNs when pricing both single-asset and multi-asset derivatives.

Beyond these technical goals, the thesis seeks to provide insights of direct relevance to the financial industry—drawing on guidance from professionals at BBVA—and to contribute to the growing body of research that bridges deep learning and quantitative finance.

1.3. Thesis Structure

The manuscript is organised into six chapters, each addressing a distinct aspect of the study:

- **Chapter 1: Introduction**
  Sets out the motivation, states the objectives, and outlines the thesis structure.
- **Chapter 2: Theoretical Background**
  Reviews the fundamentals of derivative pricing, from the Black-Scholes model to common extensions, and surveys classical numerical methods together with their shortcomings.
- **Chapter 3: Physics-Informed Neural Networks**
  Introduces the neural-network concepts underpinning PINNs, details their architecture, and explains the training procedure used to enforce the governing PDEs and constraints.
- **Chapter 4: Implementation and Results**
  Describes the computational set-up and reports empirical results for the different PDEs and instruments considered, analysing the performance of PINNs in terms of accuracy and speed.
- **Chapter 5: Conclusions and Future Work**
  Summarises the key findings, discusses limitations, and proposes directions for future research on PINN-based pricing.

2. THEORETICAL BACKGROUND

This chapter provides the foundational concepts necessary for understanding how financial products are priced, as well as the mathematical tools traditionally used for their valuation. It begins by introducing derivatives and the types most commonly traded in financial markets, then discusses the theoretical models used to price them and the limitations of classical numerical methods.

2.1. Derivative Pricing Fundamentals

Financial derivatives are contracts whose cash flows, or payments, are linked to the value of one or several *underlyings*-such as shares, commodities, foreign exchange rates, or interest rates [2], [3]. These instruments enable market participants to insure portfolios, manage exposure to financial risks, or express directional views on the different risks and assets available in the market.

The starting point for almost every derivatives textbook is the European call or put option. A European call option gives, at a fixed future date T, the right-but not de obligation-to buy the underlying at the pre-agreed strike price K; a European put option grants the right to sell the underlying asset at the same strike. The payoff functions

$$\max(S_T - K, 0), \qquad \max(K - S_T, 0)$$

characterise this optionality and depend only on the future spot price S_T. Valuing the contract amounts to forecasting the *distribution* of S_T under a special probability measure that allows the counterparties to eliminate, or *hedge*, the risks associated with movements in the underlying asset. The Black-Scholes formula achieves this under a set of assumptions, for example that the underlying follows a stochastic process (geometric Brownian motion) with constant drift and volatility. Under these assumptions, the call price is describe by a PDE whose solution is given by the closed-form expression

$$C(S_t, K, T) = S_t N(d_1) - K e^{-r(T-t)} N(d_2),$$

where N is the cumulative distribution function of the standard normal distribution, r is the risk-free interest rate, and

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

Practitioners rarely rely on the price alone for risk-management tasks; they hedge with the option's *sensitivities*. Differentiating the Black-Scholes formula gives five core quantities: delta $N(d_1)$ for first-order underlying exposure; gamma $\partial^2 C/\partial S_t^2$ for the curvature of that exposure; vega $S_t\sqrt{T-t}N'(d_1)$ for implied-volatility risk; theta $\partial C/\partial t$ for daily

time-decay; and rho $\partial C/\partial r$ for interest-rate moves. Together, these numbers condense a non-linear payoff into a form that trading systems can aggregate and that dealers adjust through underlying trades or offsetting options.

The Black-Scholes model is a cornerstone of modern finance, but its assumption of constant volatility is often too simplistic for real-world markets. Empirical evidence shows that volatility frequently varies over time, prompting practitioners to enhance the basic model by treating volatility itself as random or dynamic [10]. Although these modifications make the pricing equations more realistic, they no longer yield simple closed-form solutions, thus requiring numerical methods-such as finite differences, Monte Carlo simulations, Fourier integrals, or, as this thesis investigates, Physics-Informed Neural Networks (PINNs).

However, not all financial derivatives fit neatly into this framework. For instance, American options differ from European ones by allowing the holder to exercise at any time up to expiry, adding complexity because the optimal exercise time becomes part of the pricing problem. Technically, valuation transforms into a free-boundary problem in which one must determine both the option price and the moving boundary that separates the hold and exercise regions. Traditional numerical approaches-tree methods, finite-difference schemes, or regression-based Monte Carlo-tend to suffer from slow convergence or high variance.

Similarly, basket type of options-which derive value from multiple underlying assets-introduce another layer of complexity. Such contracts, commonly found in structured products, depend heavily on the joint behaviour of the underlyings. The payoff may, for example, depend only on the best-performing asset in a basket, making valuation challenging because complexity grows exponentially as additional assets are included. This "curse of dimensionality" makes traditional finite-difference methods impractical and Monte Carlo simulations noisy, further motivating advanced techniques such as high-dimensional PINNs.

Complexity continues to rise with *long-dated equity or FX options*, which are sensitive not only to the underlying asset price but also to volatility dynamics and varying interest rates. One way to address this is to combine stochastic-volatility models, which capture realistic volatility dynamics, with interest-rate models that discount future cash flows accurately. Individually manageable, these factors together create intricate, multi-dimensional valuation problems without straightforward analytic solutions.

Unlike equity or FX derivatives that are linked to a single asset price, *interest-rate derivatives*—such as *payer swaptions*—depend on the behavior of an entire *yield curve*. The yield curve represents the relationship between interest rates and the length of time money is borrowed—typically showing the market interest rates for loans of different maturities (e.g., 1 year, 5 years, 10 years).

A *swap* is a financial agreement between two parties to exchange sets of future cash flows, usually one paying a fixed interest rate and the other paying a floating (market-

based) rate. A *payer swaption* gives the holder the right, but not the obligation, to enter such a swap at a future date T, where they pay the fixed rate K and receive the floating rate. If the market expects higher interest rates in the future, meaning the floating rate $L(T)$ is above the fixed rate K, the swaption has value and will be exercised. The payoff is based on how much higher the floating rate is than the fixed rate, scaled by the value of future cash flows. To price such options accurately, models must account for both today's interest rates and how the entire yield curve might evolve over time.

Simplified models such as the one-factor Hull-White approach partially achieve this balance, yet closed-form solutions disappear once real-world constraints (e.g., callability or collateral requirements) are introduced, forcing practitioners to rely on numerical integration or other flexible numerical methods. PINNs emerge as an attractive alternative, seamlessly integrating curve fitting and pricing within a single computational framework.

From the plain European call to complex structures such as American options, basket contracts, and swaptions, each derivative type adds incremental complexity: multiple underlyings, additional risk factors, early-exercise features, yield-curve dependencies, and more. Collectively, these complications erode the neat closed-form solutions of classical theory and push traditional numerical methods beyond practicality.

By contrast, PINNs offer a mesh-free, dimension-agnostic alternative that naturally incorporates boundary or inequality constraints through their loss functions. The remainder of this thesis systematically quantifies how these desirable properties translate into gains in computational speed, flexibility, and accuracy across five representative derivative contracts [11], [12].

2.2. Modelling Framework

Financial derivatives can be valued in two complementary ways. On the one hand, the price equals the discounted expectation of the payoff under a risk-neutral probability measure $\mathbb{Q}$; on the other, it is the unique solution of a specific partial differential equation (PDE). The Feynman-Kac theorem builds a bridge between these stochastic and analytic representations: once the appropriate state vector has been specified, it turns a conditional expectation into a PDE, and vice versa.

This equivalence is the backbone of the chapter. We first state the general Feynman-Kac theorem and then use it as a template to derive the PDEs associated with increasingly rich models: the Black-Scholes geometric Brownian motion, its correlated multi-asset extension, and further variants that incorporate stochastic volatility and stochastic interest rates.

2.2.1. Feynman-Kac Representation

The Feynman-Kac theorem, a fundamental result linking PDEs to stochastic processes, is presented here in a financial context. Under standard regularity assumptions, the value function $u : [0, T] \times \mathbb{R}^d \to \mathbb{R}$ that solves

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \rho_{ij} \sigma_i(t, \mathbf{x}) \sigma_j(t, \mathbf{x}) x_i x_j \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^{d} \mu_i(t, \mathbf{x}) x_i \frac{\partial u}{\partial x_i} - r_t u = 0, \\ u(T, \mathbf{x}) = \Phi(\mathbf{x}), \end{aligned} \tag{2.1}$$

admits the probabilistic representation

$$u(t, \mathbf{x}) = \mathbb{E}^{\mathbb{Q}}\Big[e^{-\int_t^T r_s \mathrm{d}s} \Phi(\mathbf{X}_T) \;\Big|\; \mathbf{X}_t = \mathbf{x} \Big], \tag{2.2}$$

where the state process $\mathbf{X}_t = (X_t^{(1)}, \ldots, X_t^{(d)})$ follows the correlated geometric Itô diffusion

$$\mathrm{d}X_t^{(i)} = \mu_i(t, \mathbf{X}_t) X_t^{(i)} \mathrm{d}t + \sigma_i(t, \mathbf{X}_t) X_t^{(i)} \mathrm{d}W_t^{(i)}, \qquad i = 1, \ldots, d, \tag{2.3}$$

with quadratic covariations $\mathrm{d}W_t^{(i)} \mathrm{d}W_t^{(j)} = \rho_{ij} \mathrm{d}t$. Here μ_i and σ_i are (possibly state- and time-dependent, Lipschitz continuous functions) drift and volatility functions, r_t is the short-rate process, ρ is the constant correlation matrix, and Φ specifies the terminal payoff at maturity T.

2.2.2. The Black-Scholes Model

We begin by briefly reviewing the framework proposed by Black and Scholes [13]. As alreadt stated in the introducction, they assumed that the underlying asset price $(S_t)_{t \geq 0}$ evolves according to a geometric Brownian motion described by the stochastic differential equation (SDE):

$$\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t^{\mathbb{P}}, \quad S_0 > 0, \tag{2.4}$$

where μ denotes the expected return under the real-world measure $\mathbb{P}$ and σ is the asset volatility.

The key insight behind derivative pricing is the construction of a hedged portfolio:

$$\Pi_t = V(t, S_t) - \Delta_t S_t,$$

where $V(t, S_t)$ is the derivative price and Δ_t denotes the quantity of the underlying asset held. By applying Ito's lemma, it can be shown that choosing

$$\Delta_t = \frac{\partial V}{\partial S}$$

makes the portfolio instantaneously risk-free. Consequently, it must earn the risk-free rate r, leading to the well-known *Black-Scholes PDE*:

$$\frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV = 0, \tag{2.5}$$

with terminal condition $V(T, S) = \Phi(S)$, where all partial derivatives are evaluated at (t, S_t). The Feynman-Kac theorem in the Black-Scholes setting reduces to

$$V(t, S) = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}[\Phi(S_T) \mid S_t = S],$$

where the risk-neutral dynamics simplify to

$$\mathrm{d}S_t = rS_t\mathrm{d}t + \sigma S_t \mathrm{d}W_t^{\mathbb{Q}}. \tag{2.6}$$

An important observation in (2.6) is that the drift term-in the risk-neutral measure $\mathbb{Q}$-equals the risk-free rate r. This result applies to any derivative we consider in this thesis, as it is a consequence of the *no-arbitrage principle* [14].

Boundary Conditions

To solve the PDE for specific derivatives, appropriate boundary conditions must be defined. Practitioners define some as follows:

- **Terminal condition:** At maturity T,

  $$V(T, S) = \Phi(S).$$

  This simply states that the option's value at expiry equals its payoff. For example, a European call option pays $\Phi(S) = \max(S - K, 0)$ at time T, since the right to buy at strike K is worth nothing if the market price S is below K, and worth $S - K$ otherwise.

- **Lower boundary ($S \to 0$):** As the underlying asset price approaches zero, it becomes increasingly unlikely that a call option will end up in the money. Thus, its value should tend toward zero:

  $$V(t, 0) = 0.$$

  More generally, when modeling under a risk-neutral measure, if the option becomes insensitive to further changes in the underlying (as it would be deep out of the money), the PDE simplifies to

  $$\frac{\partial V}{\partial t} - rV = 0,$$

  indicating that the value decays at the risk-free rate, independent of S.

- **Upper boundary ($S \to \infty$):** For very large values of the underlying, the payoff of a call option increases roughly linearly with S, since the strike becomes negligible in comparison. As a result, the curvature of the price function flattens:

  $$\frac{\partial^2 V}{\partial S^2} \to 0, \quad S \to \infty.$$

  This reflects the idea that in the far right tail of the asset price distribution, the option behaves like a linear contract with negligible convexity.

Boundary conditions for other derivatives can be found in standard references such as [3].

2.2.3. Multi-Asset Extension of the Black-Scholes Model

We now extend the Black-Scholes framework to the case of $d \geq 2$ underlying assets. Setting the appropriate hedging weights $\mathbf{\Delta}_t = (\Delta_t^1, \ldots, \Delta_t^d)^\top$ and repeating the replication argument from the single-asset case, one finds that, under the risk-neutral measure $\mathbb{Q}$, each asset earns the risk-free rate r. Hence the price vector $\mathbf{S}_t = (S_t^1, \ldots, S_t^d)^\top$ satisfies

$$\mathrm{d}S_t^i = rS_t^i \mathrm{d}t + \sigma_i S_t^i \mathrm{d}W_t^{\mathbb{Q},i}, \qquad \mathrm{d}W_t^{\mathbb{Q},i} \mathrm{d}W_t^{\mathbb{Q},j} = \rho_{ij} \mathrm{d}t, \quad 1 \leq i, j \leq d. \tag{2.7}$$

Let $V(t, \mathbf{S})$ denote the option value. The same hedging argument gives the PDE

$$\frac{\partial V}{\partial t} + \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \sigma_i \sigma_j \rho_{ij} S_i S_j \frac{\partial V}{\partial S_i S_j} + r \sum_{i=1}^{d} S_i \frac{\partial V}{\partial S_i} - rV = 0. \tag{2.8}$$

Dimensionless Backward Equation

Training PINNs require appropiate input scaling and loss transformation in order to avoid numerical stability issues. A wise choice is to transform the PDE (2.8) into a dimensionless form. Formally, for maturity T define

$$x_k = \ln \frac{S_k}{K}, \qquad \tau = T - t, \qquad u(\tau, \mathbf{x}) = \frac{V(t, \mathbf{S})}{K},$$

where $\mathbf{S} = (S_1, \ldots, S_d)^\top$ and K is the strike. Starting from the multi-asset Black-Scholes PDE and applying the above change of variables, the chain rule yields the *dimensionless backward* equation

$$\frac{\partial u}{\partial \tau} = \frac{1}{2} \sum_{i=1}^{d} \sigma_i^2 \Big(\frac{\partial^2 u}{\partial x_i^2} - \frac{\partial u}{\partial x_i} \Big) + \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \sigma_i \sigma_j \rho_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + r \sum_{i=1}^{d} \frac{\partial u}{\partial x_i} - ru. \tag{2.9}$$

to be solved for $\tau \in (0, T]$ and $\mathbf{x} \in \mathbb{R}^d$ with terminal condition $u(0, \mathbf{x}) = \Phi(\mathbf{x})/K$. Equation (2.9) will serve as the reference problem for PINN solver in Section 2.2.3.

Boundary Conditions

To solve the multi-asset case described adove via the PDE (2.9) is a significant challenge due to the complexity of the boundary conditions. This conditions appear as an extension of the single-asset case and are explained below.

- **Terminal condition:** At maturity T, the option value is given by the payoff function of the dimensionless problem:

  $$u(0, \mathbf{x}) = \Phi(\mathbf{x})/K,$$

  where Φ is the payoff function of the option in dimensionless form. In this thesis we will focus on the case of the *best-of* option, which has the dimensionless payoff

  $$\Phi(\mathbf{x}) = \max\big(\max_{1 \leq k \leq d} e^{x_k} - 1, 0\big).$$

- **Lower boundaries ($S_k \to 0$):** As the underlying asset price S_k approaches zero, the option behaves as the $d-1$-asset case. This is equivalent to solving (2.9) but removing the asset S_k from the problem. Mathematically, the boundary condition is given by:

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}\sum_{\substack{i=1\\ i\neq k}}^{d} \sigma_i^2\Big(\frac{\partial^2 u}{\partial x_i^2} - \frac{\partial u}{\partial x_i}\Big) + \frac{1}{2}\sum_{\substack{i=1\\ i\neq k}}^{d}\sum_{\substack{j=1\\ j\neq k}}^{d} \sigma_i\sigma_j\rho_{ij}\frac{\partial^2 u}{\partial x_i \partial x_j} + r\sum_{\substack{i=1\\ i\neq k}}^{d}\frac{\partial u}{\partial x_i} - ru.$$

- **Upper boundaries ($S_k \to \infty$):** In the limit where the underlying asset price S_k becomes very large, we also want the option to behave linearly with respect to the underlying asset price S_k. As now we are working with dimensionless variables, we need to ensure that $\frac{\partial^2 V}{\partial S_k^2} = 0$ is appropiately scaled. The resulting condition is

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}\sum_{\substack{i=1\\ i\neq k}}^{d} \sigma_i^2\left(\frac{\partial^2 u}{\partial x_i^2} - \frac{\partial u}{\partial x_i}\right) + \frac{1}{2}\sum_{\substack{i=1\\ i\neq k}}^{d}\sum_{\substack{j=1\\ j\neq k}}^{d} \sigma_i\sigma_j\rho_{ij}\frac{\partial^2 u}{\partial x_i \partial x_j} + r\sum_{i=1}^{d}\frac{\partial u}{\partial x_i} - ru.$$

2.2.4. Stochastic Volatility Extension

As noted earlier, the Black-Scholes assumption of constant volatility and deterministic interest rates is often inadequate, especially for longer-dated equity and foreign-exchange options. Market-implied volatilities vary by strike and maturity, while the yield curve itself fluctuates with economic conditions and monetary policy. To reflect these dynamics, practitioners introduce *stochastic volatility* and *stochastic interest rates*, allowing models to match observed prices and to capture the joint sensitivity of derivatives to the underlying level, volatility changes, and rate movements.

Several routes lead to the desired pricing PDE; here we follow the hedging argument, so that later sections can invoke the Feynman-Kac theorem to obtain the corresponding probabilistic representation.

Working under the physical measure $\mathbb{P}$, assume

$$\begin{aligned} \mathrm{d}S_t &= \mu S_t \mathrm{d}t + \sqrt{v_t} S_t \mathrm{d}W_t^{\mathbb{P},S}, \\ \mathrm{d}v_t &= \kappa(\theta - v_t)\mathrm{d}t + \sigma_v \sqrt{v_t}\mathrm{d}W_t^{\mathbb{P},v}, \\ \mathrm{d}W_t^{\mathbb{P},S}\,\mathrm{d}W_t^{\mathbb{P},v} &= \rho \mathrm{d}t, \end{aligned} \tag{2.10}$$

where μ is the physical drift of the stock, κ the mean-reversion speed of the variance, and θ its long-run level. Let $V(t,S,v)$ be the value of the contract to price, and let $\widehat{V}(t,S,v)$ be a second, traded option chosen for its sensitivity to v. Construct the self-financing portfolio

$$\Pi_t = V_t - \Delta_t S_t - \Phi_t \widehat{V}_t, \tag{2.11}$$

where Δ_t and Φ_t denote (short) positions in the underlying and in the auxiliary option, respectively. The diffusion of the portfolio is

$$\mathrm{d}\Pi_t = \mathrm{d}V_t - \Delta_t \mathrm{d}S_t - \Phi_t \mathrm{d}\widehat{V}_t, \tag{2.12}$$

and we analyse each component. Applying Ito's lemma to V and $\widehat{V}$ and substituting

$$\begin{aligned} \mathrm{d}V &= \Big(\frac{\partial V}{\partial t} + \mathcal{L}^{\mathbb{P}} V\Big)\mathrm{d}t + \frac{\partial V}{\partial S}\sqrt{v}S\,\mathrm{d}W_t^{\mathbb{P},S} + \frac{\partial V}{\partial v}\sigma_v\sqrt{v}\mathrm{d}W_t^{\mathbb{P},v}, \\ \mathrm{d}\widehat{V} &= \Big(\frac{\partial \widehat{V}}{\partial t} + \mathcal{L}^{\mathbb{P}} \widehat{V}\Big)\mathrm{d}t + \frac{\partial \widehat{V}}{\partial S}\sqrt{v}S\,\mathrm{d}W_t^{\mathbb{P},S} + \frac{\partial \widehat{V}}{\partial v}\sigma_v\sqrt{v}\mathrm{d}W_t^{\mathbb{P},v}, \end{aligned} \tag{2.13}$$

where

$$\mathcal{L}^{\mathbb{P}} = \mu S\frac{\partial}{\partial S} + \kappa(\theta - v)\frac{\partial}{\partial v} + \frac{1}{2}vS^2\frac{\partial^2}{\partial S^2} + \rho\sigma_v vS\frac{\partial^2}{\partial S\,\partial v} + \frac{1}{2}\sigma_v^2 v\frac{\partial^2}{\partial v^2}.$$

Substituting (2.13) into (2.12) and rearranging, the stochastic part of $\mathrm{d}\Pi_t$ becomes

$$[\frac{\partial V}{\partial S} - \Delta_t - \Phi_t\frac{\partial \widehat{V}}{\partial S}]\sqrt{v}S\,\mathrm{d}W_t^{\mathbb{P},S} + [\frac{\partial V}{\partial v} - \Phi_t\frac{\partial \widehat{V}}{\partial v}]\sigma_v\sqrt{v}\mathrm{d}W_t^{\mathbb{P},v}.$$

Setting each bracket to zero eliminates both Brownian terms. From the $\mathrm{d}W_t^{\mathbb{P},v}$ term we obtain

$$\Phi_t = \frac{\partial V/\partial v}{\partial \widehat{V}/\partial v}, \tag{2.14}$$

and from the $\mathrm{d}W_t^{\mathbb{P},S}$ term

$$\Delta_t = \frac{\partial V}{\partial S} - \Phi_t\frac{\partial \widehat{V}}{\partial S}. \tag{2.15}$$

With (2.14)-(2.15) the diffusion in Π_t is zero, so the drift of $\mathrm{d}\Pi_t$ must equal the risk-free rate:

$$\mathrm{d}\Pi_t = r(V_t - \Delta_t S_t - \Phi_t\widehat{V}_t)\mathrm{d}t. \tag{2.16}$$

Substituting (2.14), (2.15), and (2.13) into (2.12), and after some algebra, we obtain

$$\frac{\frac{\partial V}{\partial t} + \tilde{\mathcal{L}}V - rV}{\frac{\partial V}{\partial v}} = \frac{\frac{\partial \widehat{V}}{\partial t} + \tilde{\mathcal{L}}\widehat{V} - r\widehat{V}}{\frac{\partial \widehat{V}}{\partial v}}, \tag{2.17}$$

where

$$\tilde{\mathcal{L}} = rS\frac{\partial}{\partial S} + \kappa(\theta - v)\frac{\partial}{\partial v} + \frac{1}{2}vS^2\frac{\partial^2}{\partial S^2} + \rho\sigma_v vS\frac{\partial^2}{\partial S\,\partial v} + \frac{1}{2}\sigma_v^2 v\frac{\partial^2}{\partial v^2}.$$

The only difference between $\mathcal{L}^{\mathbb{P}}$ and $\tilde{\mathcal{L}}$ is that the drift μ has been replaced by the risk-free rate r.

Relation (2.17) must hold for *every* sufficiently smooth pair of payoffs $(V, \widehat{V})$, so the common value of the two quotients can depend *only* on the state variables (t, S, v). We therefore introduce the *instantaneous market price of volatility risk*

$$\lambda(t, S, v) := \frac{\frac{\partial \widehat{V}}{\partial t} + \tilde{\mathcal{L}}\widehat{V} - r\widehat{V}}{\frac{\partial \widehat{V}}{\partial v}}$$

Substituting this definition into the numerator of (2.17) yields

$$\frac{\partial V}{\partial t} + \tilde{\mathcal{L}}V - rV = \lambda(t, S, v)\frac{\partial V}{\partial v}.$$

Collecting terms, we obtain the *risk-neutral* generator

$$\mathcal{L}^{\mathbb{Q}} = rS\frac{\partial}{\partial S} + [\kappa(\theta - v) - \lambda(t,S,v)]\frac{\partial}{\partial v} + \frac{1}{2}vS^2\frac{\partial^2}{\partial S^2} + \rho\sigma_v vS\frac{\partial^2}{\partial S\,\partial v} + \frac{1}{2}\sigma_v^2 v\frac{\partial^2}{\partial v^2}.$$

Hence every European-style payoff $V(t,S,v)$ satisfies

$$\frac{\partial V}{\partial t} + \mathcal{L}^{\mathbb{Q}}V - rV = 0, \qquad V(T,S,v) = \Phi(S). \tag{2.18}$$

It is customary to assume that investors are *not* compensated for volatility risk, that is, $\lambda(t,S,v) \equiv 0$. The risk-neutral PDE then becomes

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \kappa(\theta - v)\frac{\partial V}{\partial v} + \frac{1}{2}vS^2\frac{\partial^2 V}{\partial S^2} + \rho\sigma_v vS\frac{\partial^2 V}{\partial S\,\partial v} + \frac{1}{2}\sigma_v^2 v\frac{\partial^2 V}{\partial v^2} - rV = 0. \tag{2.19}$$

By the Feynman-Kac theorem, the unique solution is

$$V(t,S,v) = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\Big[\Phi(S_T) \;\Big|\; S_t = S, v_t = v\Big], \tag{2.20}$$

where $(S_t, v_t)_{t\in[0,T]}$ evolves according to

$$\begin{aligned} \mathrm{d}S_t &= rS_t\mathrm{d}t + \sqrt{v_t}S_t\mathrm{d}W_t^{\mathbb{Q},S}, \\ \mathrm{d}v_t &= \kappa(\theta - v_t)\mathrm{d}t + \sigma_v\sqrt{v_t}\mathrm{d}W_t^{\mathbb{Q},v}, \\ \mathrm{d}W_t^{\mathbb{Q},S}\,\mathrm{d}W_t^{\mathbb{Q},v} &= \rho\mathrm{d}t, \end{aligned} \tag{2.21}$$

with parameters as defined above.

Because (S_t, v_t) is a two-dimensional Ito diffusion under $\mathbb{Q}$, the Feynman-Kac representation (2.20) holds without further hypotheses. In other words, once the risk-neutral SDEs (2.21) are fixed, the PDE follows automatically—a shortcut that will be invaluable for the extensions introduced later.

Boundary Conditions

The Heston model introduces an additional state variable v, representing the stochastic variance of the underlying asset. As a result, the pricing PDE must be complemented by appropriate boundary conditions at the extremal values of v. These conditions are crucial for the well-posedness and numerical stability of the solution. We follow the guidance from [15] and elaborate as follows:

- **Lower boundary $v \to 0$:** When the instantaneous variance v approaches zero, the asset price becomes effectively deterministic. In this regime, the uncertainty in future outcomes vanishes, and the asset evolves along a single path governed by the drift. Since the option is no longer exposed to volatility risk, its price converges to the discounted intrinsic value under a risk-free growth assumption. Mathematically, the solution to the PDE reduces to the payoff evaluated at the forward price:

  $$V(t,S,0) = e^{-r(T-t)}\Phi(S_T), \quad S_T = Se^{r(T-t)},$$

  where $\Phi(\cdot)$ denotes the option's payoff function. This reflects the idea that, in the absence of diffusion, the asset grows deterministically at rate r.

- **Upper boundary $v \to \infty$:** In the opposite regime, where v becomes very large, the volatility dominates the dynamics of the asset price. This implies an extreme level of randomness in the asset path, causing the option value to behave more like a function of the expected range of future outcomes rather than specific paths. While the option price itself remains bounded, its sensitivity to variance grows.

  In this high-volatility limit, the option price increases roughly proportionally with the underlying asset price S—this is particularly true for call options, which benefit from the convexity of the payoff. To express this, one imposes a Neumann boundary condition in the variance direction:

$$\frac{\partial V}{\partial v}(t, S, v) \to S \quad \text{as } v \to \infty.$$

  This derivative expresses how the option price responds to changes in variance. The condition indicates that, for very high volatility, this sensitivity saturates to the asset level—capturing the intuition that in such cases, the option can swing as wildly as the underlying asset itself.

2.2.5. Stochastic Interest Rates Extension

For maturities beyond a few months, or for products whose underlying is an interest rate, treating the short rate as a constant is inadequate [16]. A more realistic approach lets the short rate r_t evolve stochastically, reflecting macroeconomic factors, monetary policy, and other market forces. One widely used model is the *Hull-White* [17] model. It assumes an Ornstein-Uhlenbeck, mean-reverting process, that allows a time-dependent mean level, giving additional flexibility to fit the current yield curve.

Under the risk-neutral measure $\mathbb{Q}$ the model can be written as

$$\mathrm{d}r_t = \kappa(\theta(t) - r_t)\mathrm{d}t + \sigma_r \mathrm{d}W_t^{\mathbb{Q},r}, \tag{2.22}$$

where κ is the mean-reversion speed, $\theta(t)$ the long-run level, and σ_r the volatility. For a derivative whose price is $V(t, r)$, the corresponding PDE is

$$\frac{\partial V}{\partial t} + \kappa(\theta(t) - r)\frac{\partial V}{\partial r} + \frac{1}{2}\sigma_r^2 \frac{\partial^2 V}{\partial r^2} - rV = 0. \tag{2.23}$$

A distinguishing feature of these short-rate models is their *affine* structure in the state variable r_t. In an affine model the continuously-compounded spot rate for maturity T can be written as a linear function of the state variable [16]:

$$R(t, T) = \alpha(t, T) + \beta(t, T) r_t,$$

where α and β are deterministic functions of t and T. This structure allows us to express the price of a zero-coupon bond $P(t, T)$ as an exponential-affine function of the short rate:

$$P(t, T) = A(t, T) \exp\{-B(t, T) r_t\}. \tag{2.24}$$

In the Hull-White model, for $0 \le t \le T$, the functions $A(t,T)$ and $B(t,T)$ are given by [16]

$$B(t,T) = \frac{1 - e^{-\kappa(T-t)}}{\kappa}, \tag{2.25}$$

$$A(t,T) = \frac{P^M(0,T)}{P^M(0,t)} \exp\Big[-\frac{\sigma_r^2}{4\kappa}(1 - e^{-2\kappa t})B(t,T)^2\Big], \tag{2.26}$$

where $P^M(0,T)$ is the market price at time 0 of a zero-coupon bond maturing at T and

$$f^M(0,t) = -\frac{\partial \log P^M(0,t)}{\partial t}$$

is the corresponding instantaneous forward rate. Substituting (2.25)-(2.26) into (2.24) yields the calibrated bond-price formula

$$P(t,T) = \frac{P^M(0,T)}{P^M(0,t)} \exp\Big[-B(t,T)r_t - \frac{\sigma_r^2}{4\kappa}(1 - e^{-2\kappa t})B(t,T)^2\Big]. \tag{2.27}$$

The time-dependent mean-reversion level $\theta(t)$ required for exact curve fitting is recovered from the market instantaneous forward rate $f^M(0,t)$ via

$$\theta(t) = \frac{\partial f^M(0,t)}{\partial t} + \kappa f^M(0,t) + \frac{\sigma_r^2}{2\kappa^2}(1 - e^{-2\kappa t}). \tag{2.28}$$

Boundary Conditions

To ensure well-posedness and numerical stability, boundary conditions must also be specified in the r-direction. Following the guidance in [15], we describe the asymptotic behavior as $r \to \pm\infty$:

- **Lower and upper boundary $r \to \pm\infty$:** In the limiting regimes where the short rate becomes either extremely high or extremely low (even negative), the impact of further changes in r on the option price becomes negligible.

  For very large r, this can be seen in the discounting factor $e^{-r(T-t)}$ as it starts to dominate, rapidly driving the present value of any future payoff to zero. In this regime, the value of the derivative asymptotically approaches zero, and changes in r no longer significantly affect its price.

  For very negative r, the discounting factor grows, but this is typically bounded in practice by central bank policies and economic considerations. Moreover, the terminal payoff structure (especially for vanilla options) often dominates the pricing behavior in this regime, leading again to diminishing sensitivity to changes in r.

  As a result, the option value becomes asymptotically flat in the r-direction, and a natural boundary condition is to set the partial derivative of the option value with respect to the short rate to zero:

$$\frac{\partial V}{\partial r}(t,S,v,r) = 0, \quad \text{as } r \to \pm\infty.$$

This Neumann boundary condition reflects the assumption that outside a realistic band of short rate values, the solution surface is locally constant in r.

2.2.6. Joint Stochastic Volatility and Stochastic Rates

Finally, we drop both the assumption of constant volatility and the assumption of constant interest rates, and consider a model that combines both factors as source of randomness. In this case, the joint dynamics of the state vector (S_t, v_t, r_t) under the risk-neutral measure $\mathbb{Q}$ are given by the SDEs

$$\begin{aligned} \mathrm{d}S_t &= r_t S_t \mathrm{d}t + \sqrt{v_t} S_t \mathrm{d}W_t^{\mathbb{Q},S}, \\ \mathrm{d}v_t &= \kappa_v(\theta_v - v_t)\mathrm{d}t + \sigma_v \sqrt{v_t}\mathrm{d}W_t^{\mathbb{Q},v}, \\ \mathrm{d}r_t &= \kappa_r(\theta_r(t) - r_t)\mathrm{d}t + \sigma_r \mathrm{d}W_t^{\mathbb{Q},r}, \\ \mathrm{d}W_t^{\mathbb{Q},S}\mathrm{d}W_t^{\mathbb{Q},v} &= \rho_{SV}\mathrm{d}t, \\ \mathrm{d}W_t^{\mathbb{Q},S}\mathrm{d}W_t^{\mathbb{Q},r} &= \rho_{SR}\mathrm{d}t, \\ \mathrm{d}W_t^{\mathbb{Q},v}\mathrm{d}W_t^{\mathbb{Q},r} &= \rho_{VR}\mathrm{d}t, \end{aligned} \tag{2.29}$$

where $\rho_{SV}, \rho_{SR}, \rho_{VR} \in [-1, 1]$ are the instantaneous correlations between the Brownian motions driving the dynamics of the underlying asset, the volatility, and the short rate, respectively. The corresponding pricing PDE for a derivative whose value depends on the state vector (S_t, v_t, r_t) is

$$\begin{aligned} &\frac{\partial V}{\partial t} + \kappa_r(\theta_r(t) - r)\frac{\partial V}{\partial r} + \kappa_v(\theta_v - v)\frac{\partial V}{\partial v} + rS\frac{\partial V}{\partial S} - rV \\ &\quad + \rho_{SV}\sigma_v v S \frac{\partial^2 V}{\partial S\,\partial v} + \rho_{SR}\sigma_r S\sqrt{v}\frac{\partial^2 V}{\partial S\,\partial r} + \rho_{VR}\sigma_v\sigma_r\sqrt{v}\frac{\partial^2 V}{\partial v \partial r} \\ &\quad + \frac{1}{2}\sigma_r^2\frac{\partial^2 V}{\partial r^2} + \frac{1}{2}vS^2\frac{\partial^2 V}{\partial S^2} + \frac{1}{2}\sigma_v^2 v\frac{\partial^2 V}{\partial v^2} = 0. \end{aligned} \tag{2.30}$$

As no new dimensions are added compared to the previous sections, the boundary conditions are the same as in Section 2.2.4 and 2.2.5.

2.3. Traditional Numerical Approaches

Finite Difference Methods

As noted earlier, Finite Difference Methods (FDM) are among the most widely used techniques for numerically solving partial differential equations such as (2.5) and their multi-dimensional counterpart (2.8). The idea is to discretise the time and asset-price domains into a structured grid and replace the continuous derivatives in the PDE with finite-difference approximations.

For instance, in the single-asset case the time derivative can be approximated with a backward difference:

$$\frac{\partial V}{\partial t} \approx \frac{V^n - V^{n-1}}{\Delta t},$$

while the first- and second-order spatial derivatives are obtained with central differences:

$$\frac{\partial V}{\partial S} \approx \frac{V^n_{i+1} - V^n_{i-1}}{2\Delta S}, \qquad \frac{\partial^2 V}{\partial S^2} \approx \frac{V^n_{i+1} - 2V^n_i + V^n_{i-1}}{\Delta S^2}.$$

These substitutions turn the PDE into a system of algebraic equations that can be solved by standard linear-algebra techniques such as LU decomposition or iterative methods.

Different strategies can be used for the time discretisation, each with its own stability and accuracy characteristics:

- **Explicit Scheme:** The option value at the next time step is computed directly from known values at the current time. This results in a simple, forward-marching algorithm. However, the method is only conditionally stable—meaning that the time step Δt must be sufficiently small to satisfy the Courant-Friedrichs-Lewy (CFL) condition for stability. Otherwise, the numerical solution becomes unstable.
- **Implicit Scheme:** The option value at the next time step appears on both sides of the finite difference equation, resulting in a linear system that must be solved at each time step. This method is unconditionally stable, allowing for larger time steps, but requires more computational effort per step due to the matrix inversion or solution.
- **Crank-Nicolson Scheme:** This is a popular second-order accurate method that averages the explicit and implicit schemes. It offers better accuracy than the implicit method while retaining unconditional stability for linear problems. However, it may introduce oscillations near discontinuities (e.g., at option exercise boundaries) unless damping techniques are applied.

The choice among these schemes depends on the specific problem being solved. Explicit methods are attractive for their simplicity and ease of implementation but are impractical for stiff or high-dimensional PDEs. Implicit and Crank-Nicolson schemes are preferred for more stable and accurate solutions, especially in pricing derivatives with long maturities or complex boundary conditions.

Despite their simplicity and transparency, FDMs suffer from the curse of dimensionality: in d dimensions the number of grid points grows exponentially with d. Consequently, their use is limited to small d, or to cases where symmetry, dimensional reduction, or sparse grid techniques can reduce the effective dimensionality.

Monte Carlo Methods

Monte Carlo (MC) methods estimate prices by simulating many sample paths of the underlying assets under the risk-neutral measure $\mathbb{Q}$ and averaging the discounted payoff [4].

With several risk factors this means generating correlated geometric Brownian motions. An example of an MC pricing algorithm is given below:

Algorithm 1 MC pricing of a general payoff

1: **Input:** number of simulations N, horizon T, risk-free rate r, strike K, asset parameters $(\mu_i, \sigma_i, \rho_{ij})$
2: **for** $n = 1$ to N **do**
3: Simulate one terminal price S_T^i for each asset using correlated Brownian increments
4: Compute the payoff $\text{Payoff}^{(n)}$
5: **end for**
6: Estimate the option value

$$V(0, \mathbf{S}_0) \approx e^{-rT} \frac{1}{N} \sum_{n=1}^{N} \text{Payoff}^{(n)}$$

MC methods are attractive in high dimensions because their convergence rate does not deteriorate with the number of state variables. They converge only at $O(1/\sqrt{N})$, however, so variance-reduction techniques (control variates, antithetic sampling, and so on) are usually needed for efficiency.

Another limitation is that plain MC simulation is ill-suited to early-exercise or strongly path-dependent derivatives, which require decisions at multiple dates or knowledge of the full price path. Although extensions such as the Longstaff-Schwartz regression method [18] exist, they tend to be less accurate and less efficient than PDE-based approaches. In these situations, PINNs can offer a compelling alternative, as they tackle the pricing equation directly and naturally encode early-exercise or path-dependent features in the loss function.

3. PHYSICS-INFORMED MACHINE LEARNING

3.1. Introduction to Physics-Informed Neural Networks (PINNs)

As noted in earlier sections, traditional numerical techniques for solving pricing PDEs often suffer from severe computational burdens that stem from complex payoff features or high dimensionality. Physics-Informed Neural Networks, introduced by Raissi et al. [7], offer an alternative framework that embeds the governing PDE directly into the training process. Instead of relying solely on labelled data, PINNs enforce the physical laws and constraints intrinsic to the problem, thereby improving the accuracy, efficiency and robustness of the resulting approximation.

Consider a generic initial-boundary-value problem

$$\begin{aligned} \mathcal{N}_I[u(t,x)] &= 0, \quad x \in \Omega, \quad t \in [0,T], \\ \mathcal{N}_B[u(t,x)] &= 0, \quad x \in \partial\Omega, \quad t \in [0,T], \\ \mathcal{N}_0[u(t^*,x)] &= 0, \quad x \in \Omega, \quad t^* = 0, \end{aligned} \tag{3.1}$$

where $\mathcal{N}_I$ is the interior operator on the spatial domain Ω, $\mathcal{N}_B$ enforces the boundary conditions on $\partial\Omega$, and $\mathcal{N}_0$ specifies the initial data.

A PINN approximates the solution $u(t,x)$ by a neural network $u_\theta(t,x)$ with parameters θ. These parameters are determined by minimising the composite loss

$$\mathcal{L}(\theta) = \lambda_1 \mathcal{L}_{\text{PDE}}(\theta) + \lambda_2 \mathcal{L}_{\text{boundary}}(\theta) + \lambda_3 \mathcal{L}_{\text{initial}}(\theta), \tag{3.2}$$

where λ_i are predefined loss weights, and

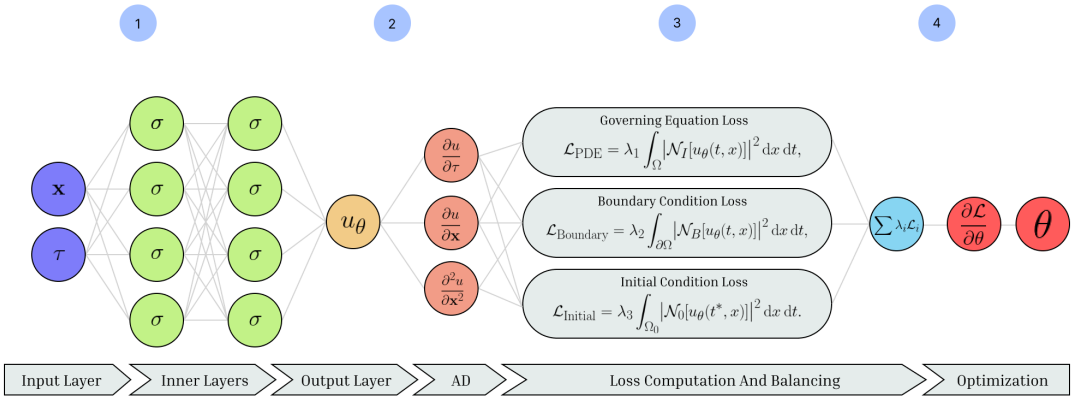
$$\begin{aligned} \mathcal{L}_{\text{PDE}}(\theta) &= \int_0^T \int_\Omega |\mathcal{N}_I[u_\theta(t,x)]|^2 \, \mathrm{d}x\mathrm{d}t, \\ \mathcal{L}_{\text{boundary}}(\theta) &= \int_0^T \int_{\partial\Omega} |\mathcal{N}_B[u_\theta(t,x)]|^2 \, \mathrm{d}s\mathrm{d}t, \\ \mathcal{L}_{\text{initial}}(\theta) &= \int_\Omega |\mathcal{N}_0[u_\theta(0,x)]|^2 \, \mathrm{d}x. \end{aligned} \tag{3.3}$$

In practice these integrals are approximated via Monte Carlo sampling by evaluating the residuals at sets of collocation points drawn inside the corresponding domains.

Figure 3.1 illustrates the overall training workflow of a PINN applied to a PDE problem like (3.1). Once the neural network architecture is defined, a set of collocation points is sampled across the interior, boundary, and initial domains. These points are fed through the network to generate predicted outputs, and automatic differentiation is used to compute the required spatial and temporal derivatives. The residuals of the governing equations are then evaluated at these points, forming the loss terms in (3.3). The total loss $\mathcal{L}(\theta)$ is used to iteratively update the network parameters via gradient descent. This cycle

Figure 3.1

Schematic of PINN training. (1) Collocation points are fed into the neural network. (2) Automatic differentiation yields spatial and temporal derivatives. (3) PDE residuals—and hence the loss terms—are evaluated at those points. (4) The resulting losses are combined to form $\mathcal{L}(\theta)$; its gradient drives an optimisation step on θ. The cycle repeats until convergence.



continues until the PDE, boundary, and initial conditions are satisfied within a desired tolerance.

3.2. Training PINNs

Training PINNs involves minimising the total loss $\mathcal{L}(\theta)$ with respect to the network parameters θ. This is typically accomplished using gradient-based optimisers such as stochastic gradient descent (SGD) and its variants (e.g., Adam [19], RMSProp), which iteratively update the parameters using the gradient of the loss. Alternatively, second-order quasi-Newton methods like L-BFGS are sometimes employed, particularly in the later stages of training due to their fast convergence in low-dimensional parameter spaces. However, these methods can suffer from ill-conditioning in the approximate Hessian, which may lead to slow convergence or divergence [20].

Gradients are computed via automatic differentiation, ensuring accurate and efficient calculation of sensitivities with respect to θ. Training proceeds until a convergence criterion is met, such as loss stabilization or a maximum number of iterations.

The total loss $\mathcal{L}(\theta)$ typically consists of multiple components: PDE residuals, initial and boundary condition mismatches, and any regularization terms. Proper loss balancing is critical—assigning appropriate weights λ_i to each term can significantly influence convergence and solution quality. Heuristics, fixed scaling, or dynamic reweighting schemes (e.g., learning loss weights during training or gradient-based adaptive balancing) are often used with gradient-descent methods to avoid domination of one component over others and to improve overall stability [21]

The choice of optimizer also plays a key role. First-order methods like Adam offer robustness and adaptability across a wide range of problems, especially during early training when gradients can be noisy. In contrast, second-order optimizers such as L-BFGS may provide superior local convergence but require careful tuning and are sensitive to scale and conditioning. Hybrid strategies that begin with Adam and switch to L-BFGS mid-training are sometimes used to combine the benefits of both. In larger models, computational cost and memory requirements can become prohibitive, making first-order methods more practical. In constrast, second-order methods could be prefered for smaller networks or when high precision is required.

The underlying neural architecture influences both expressiveness and trainability. Shallow feedforward networks are the default choice, but more expressive alternatives such as residual networks, Fourier feature embeddings, or adaptive basis networks can better capture complex solution behaviours or handle stiff PDEs. Architectural choices also affect the propagation of gradients and the network's ability to represent sharp transitions or singularities, which are common in financial PDEs.

PINNs rely on synthetically generated collocation points sampled throughout the domain: in the interior Ω, on the boundary $\partial\Omega$, and on the initial slice Ω_0. While uniform sampling is common, alternative sampling strategies can improve convergence and solution quality. These include importance sampling based on solution gradients, error indicators, or physics-informed heuristics that focus computational effort on high-error or high-variance regions. Stratified sampling or Latin Hypercube designs can help in covering the domain more uniformly when dimensions increase [22].

Before optimisation begins, the parameters θ are typically initialised randomly, commonly using Xavier or He initialisation, or drawn from a pretrained model. Good initialisation accelerates convergence and improves final accuracy. The dimensionless reformulation of the PDEs discussed earlier helps by keeping input scales comparable, which also stabilises training [23].

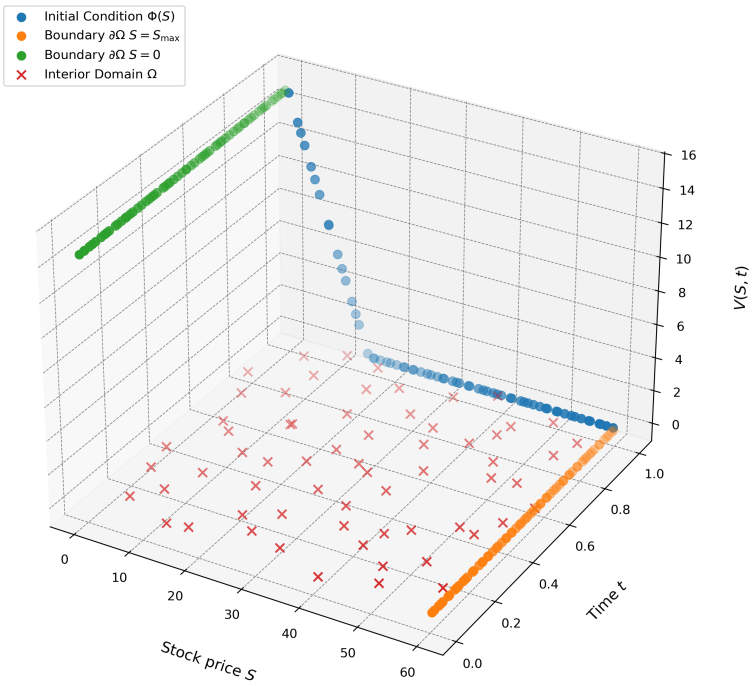
Algorithm 2 Training procedure for a PINN

1: **Input:** network architecture; PDE, boundary, and initial operators.
2: Initialise parameters θ.
3: Generate collocation points in Ω, $\partial\Omega$, and Ω_0.
4: **while** not converged **do**
5: (Optional) select a mini-batch of collocation points.
6: Evaluate the total loss $\mathcal{L}(\theta)$ from (3.2).
7: Compute $\nabla_\theta \mathcal{L}(\theta)$ via automatic differentiation.
8: Update θ with the chosen optimiser.
9: **end while**

Fine-tuning any component of this pipeline—whether loss composition, optimizer configuration, architecture design, or sampling strategy—can materially influence perfor-

Figure 3.2

Collocation points for a put option. Points are sampled inside the domain Ω, on the boundary $\partial\Omega$, and on the initial slice Ω_0.



mance. The next section outlines the configuration adopted in this thesis; implementation details follow in the subsequent chapter.

4. IMPLEMENTATION AND RESULTS

4.1. Implementation Details

Most of the results presented in this thesis use relatively small feedforward neural networks. These smaller architectures provided an optimal balance between computational efficiency and approximation accuracy. Regarding activation functions, we primarily employed the hyperbolic tangent (*tanh*), given its smooth and differentiable nature. Although we also experimented with the Softplus activation—which closely resembles typical payoff functions—we observed only marginal improvements in convergence speed, insufficient to justify its use over *tanh*.

In terms of optimization methods, quasi-Newton algorithms, particularly the L-BFGS solver implemented in PyTorch, consistently delivered the best outcomes. Other solvers such as BFGS and SSBroyden were also implemented and tested, and in some cases yielded better results. Gradient-based optimizers, notably Adam, were evaluated but generally failed to match the accuracy or convergence speed achieved by quasi-Newton approaches.

Sampling collocation points was predominantly carried out using a uniform distribution over the domain of interest to evaluate PDE residuals and boundary conditions. Alternative sampling strategies, including Sobol and Halton sequences, were tested but did not lead to significant improvements in accuracy or computational efficiency.

All models were implemented in Python using the PyTorch framework. The complete implementation is publicly available on GitHub [24] and a brief article on the topic can be found in [25]. Computations were performed on a personal computer equipped with an Intel i7-12700K CPU and an NVIDIA RTX 2070 GPU, facilitating efficient training of neural network models.

4.2. Call Option

As an initial test case for the methodology, we priced a standard European call option under the classical Black-Scholes model, whose PDE in Eq. (2.5) admits a closed-form solution that serves as a convenient benchmark. The numerical experiment used the parameters in Table 4.1.

Parameter	Value
Strike price, K	\$100
Maturity, T	1.0 year
Risk-free rate, r	0.05
Volatility, σ	0.20

Table 4.1
Call Option Parameters

The PINN consisted of two hidden layers with 20 neurons each and was trained with the BFGS optimizer on 75000 collocation points distributed across the interior domain Ω, the boundary $\partial\Omega$, and the initial hypersurface Ω_0. Despite its simple architecture, the network reproduced the analytical solution accurately, achieving an ℓ_2 error of 1.5054×10^{-5}.

Figure 4.1
Call option price under the Black-Scholes PDE. The PINN solution closely matches the analytical benchmark.

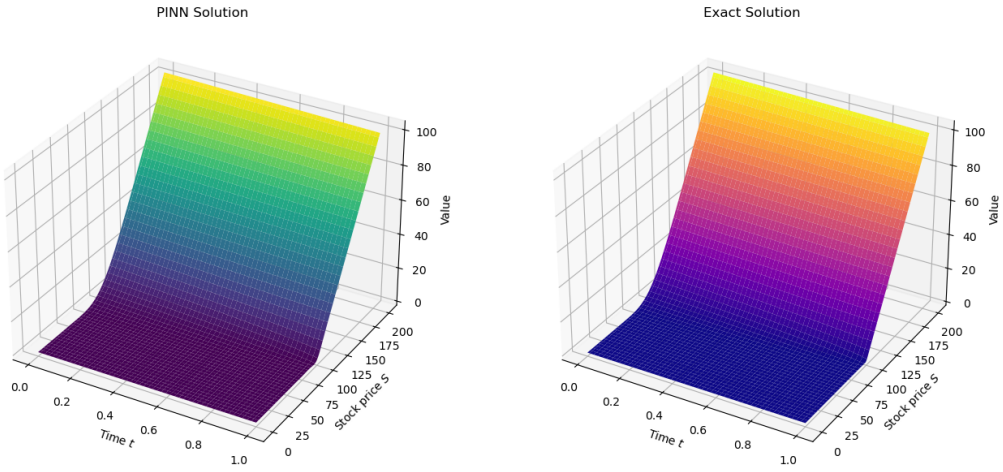
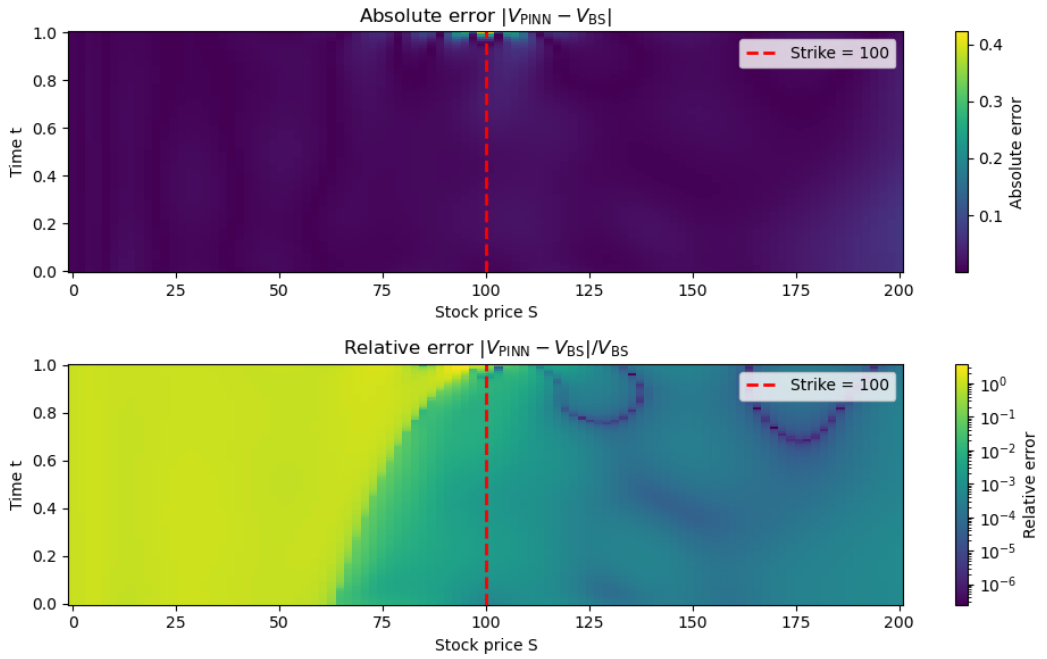


Figure 4.1 shows that the learned pricing surface is visually indistinguishable from the Black-Scholes benchmark, while Figure 4.2 depicts the absolute error, which is largest near maturity and the strike.

Figure 4.2

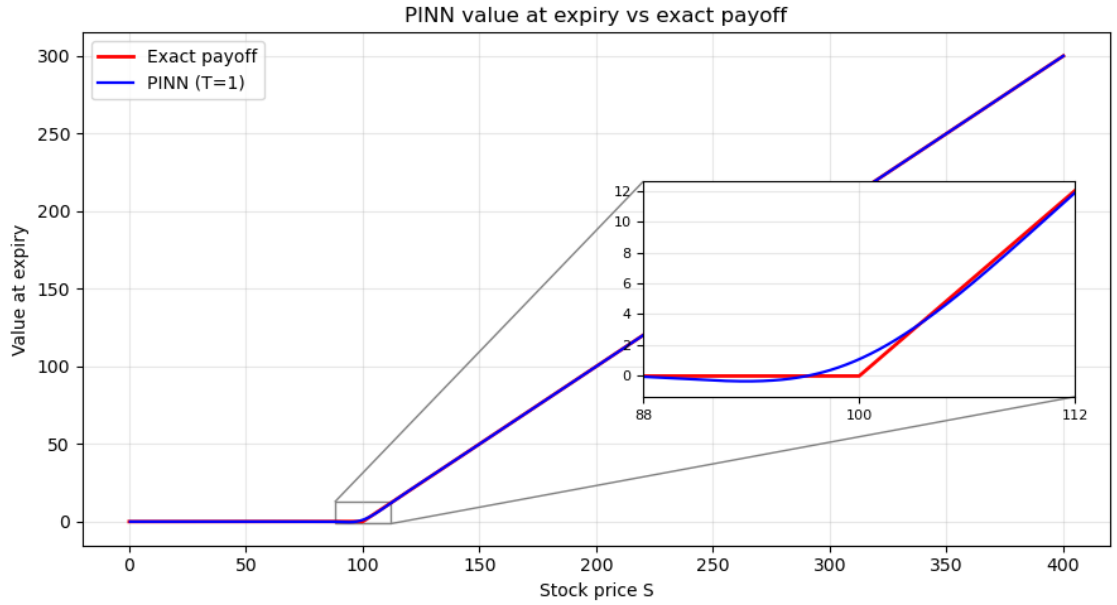
Absolute error of the PINN solution. Errors peak close to maturity and the strike price.



The pronounced spike in error at-the-money on the expiry surface stems from the kink in the option payoff. When a function has a discontinuous first derivative, any approximation built from smooth basis functions—including neural networks—develops oscillatory overshoots, an instance of the *Gibbs phenomenon*.

Figure 4.3

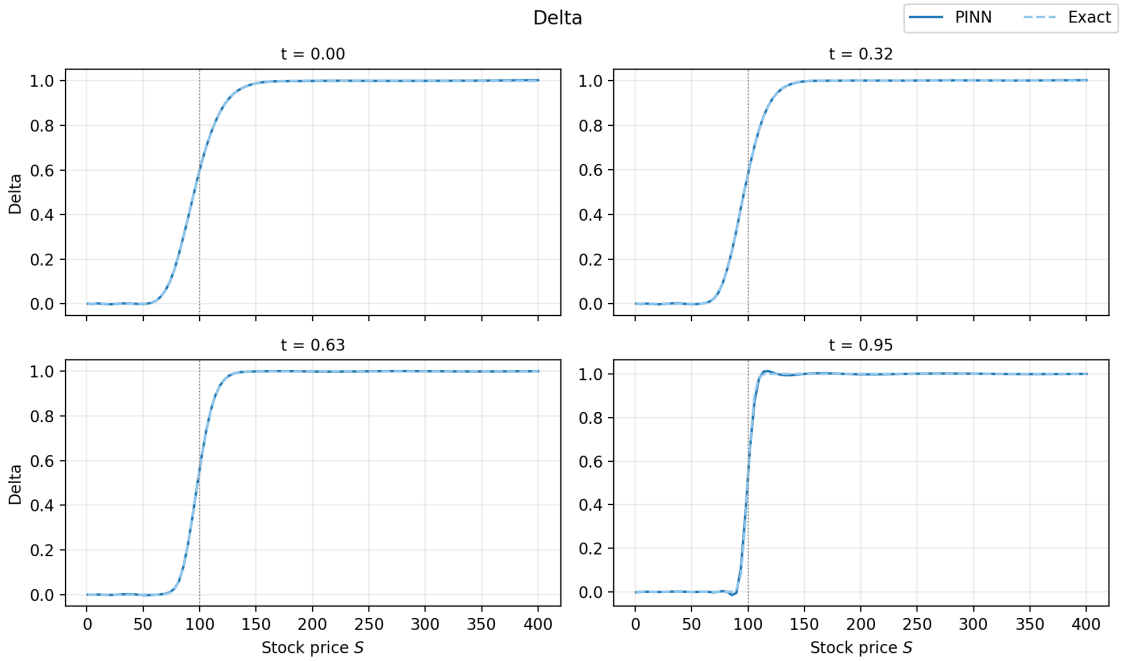
Option value at expiry. Away from the strike, the PINN matches the analytical solution; near the kink the absolute error peaks.



As already stated, a key advantage of training neural networks with automatic differentiation is that it furnishes *exact* derivatives of the network output with respect to its inputs. In quantitative finance these derivatives are the familiar *Greeks*, which risk desks use to monitor and hedge positions. Because the network's forward and backward passes are computed simultaneously, we obtain the entire vector of Greeks at virtually no extra cost, so their fidelity becomes a critical test of the PINN. The same effect propagates—more visibly—into the Greeks: Delta (Δ) (Figure 4.4) is recovered almost perfectly, except during the final moments before maturity, when the true delta collapses into a step function that is hard to emulate with smooth activations.

Figure 4.4

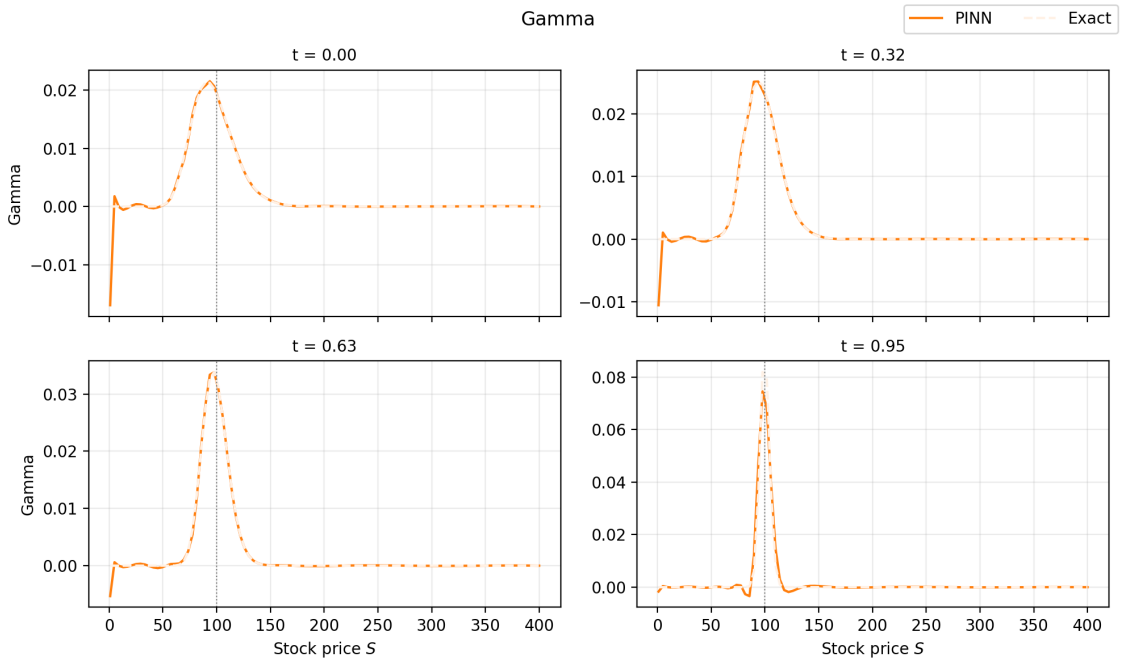
Δ *comparison. Small oscillations emerge only very close to* $t = T$.



Gamma (Γ) presents the greatest challenge (Figure 4.5). As a second spatial derivative it amplifies any ripple in the reconstructed price surface, producing the largest L_2 error in Table 4.2. Additional collocation points around the strike or an explicit curvature-penalty term in the loss could alleviate this issue.

Figure 4.5

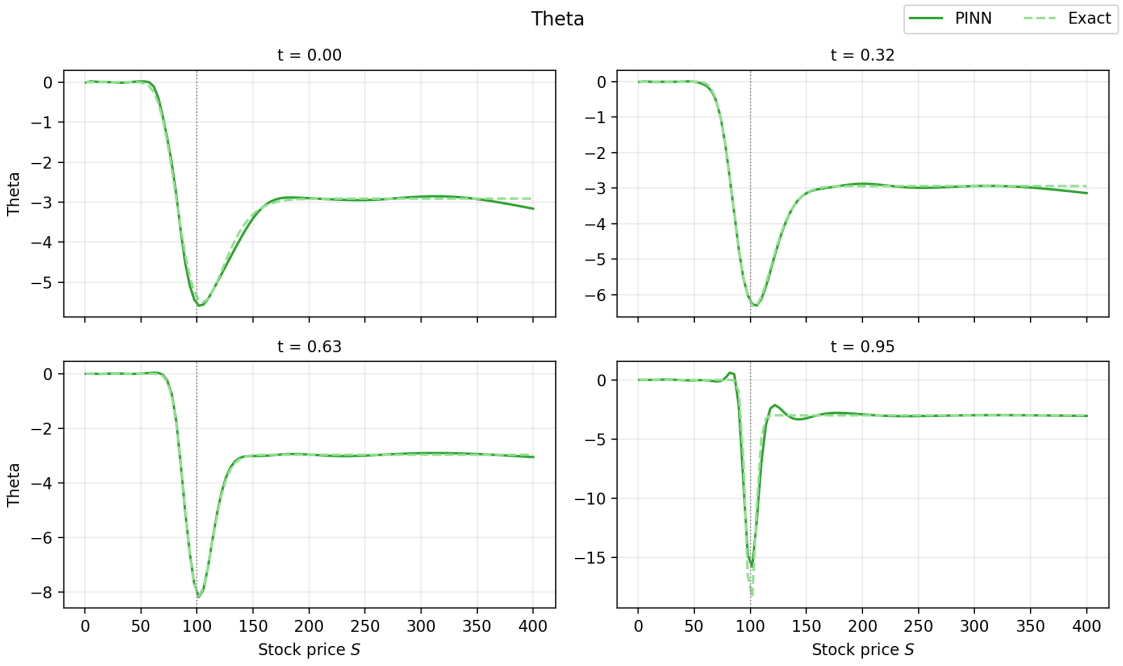
Γ *comparison. Oscillations near the strike dominate the error; a secondary mismatch appears for very small* S.



Theta (Θ) sits between the two extremes: its error (12%) is larger than delta's but well below gamma's 26% (see Table 4.2). In this case, the error can be explain by violations of the temporal dependence structure of the PDE, as reported in [26], [27].

Figure 4.6

Θ *comparison. The main discrepancies occur near maturity, consistent with the temporal derivative being the hardest term to satisfy in the PDE residual.*



$\|\Delta_{\text{PINN}} - \Delta_{\text{BS}}\|_2 / \|\Delta_{\text{BS}}\|_2$	$\|\Theta_{\text{PINN}} - \Theta_{\text{BS}}\|_2 / \|\Theta_{\text{BS}}\|_2$	$\|\Gamma_{\text{PINN}} - \Gamma_{\text{BS}}\|_2 / \|\Gamma_{\text{BS}}\|_2$
0.0066	0.1201	0.2553

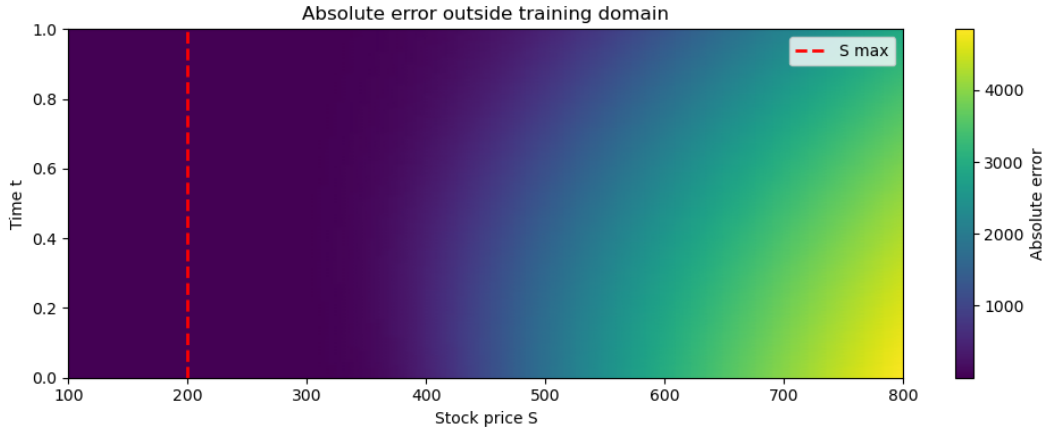
Table 4.2

Relative L_2-errors of the Greeks.

Finally, Figure 4.7 underscores a generic limitation of data-driven solvers: accuracy deteriorates outside the region spanned by training points. Within the calibrated domain, however, the errors remain well below thresholds used in practice.

Figure 4.7

Relative error of the PINN solution. The error is concentrated around the strike price and maturity.



For this straightforward example, training required about two minutes, underscoring the computational burden that PINNs impose. Once trained, evaluation is fast—in our test, roughly 1.45× faster than the closed-form Black-Scholes formula—but the initial training cost remains a practical consideration.

4.3. Multi-Asset Option

To test the ability of PINNs to solve higher-dimensional problems, we extend the single-asset experiment to multiple underlying assets. A compact network with three hidden layers of ten neurons each is first trained on the dimensionless form of the multi-asset Black-Scholes PDE in Eq. (2.9).

Because training costs increase significantly with dimensionality—due both to the complexity of the boundary conditions and the exponential growth in the volume of the domain—we employ a deliberately small set of collocation points: 500 in the interior domain Ω, 500 on the boundary $\partial\Omega$, and 500 on the initial hypersurface Ω_0. It is important to note that the number of required points on the boundary increases linearly with the number of assets, which further contributes to the computational burden as dimensionality grows. The specific parameters used in this test case are summarized in Table 4.3.

Parameter	Value
Strike price, K	\$100
Maturity, T	1.0 year
Risk-free rate, r	0.05
Volatilities, σ_i	0.20
Correlation, ρ_{ij}	0.25

Table 4.3
Best-Of Call Option Parameters

The surfaces obtained from the trained PINN, shown in Figures 4.8 and 4.9, correspond to the first two asset dimensions in the dimensionless and original coordinate spaces, respectively.

As in the single-asset case, the solution displays the expected features: values near zero when asset prices are far below the strike, approximately linear growth as asset prices increase well above the strike, and a ridge or kink near the strike region. These qualitative features are preserved as dimensionality increases, but the quality of the approximation deteriorates. The surfaces reveal increased irregularity and loss of precision, particularly in regions near the domain boundaries.

Figure 4.8
Dimensionless multi-asset call option price (first two asset axes).

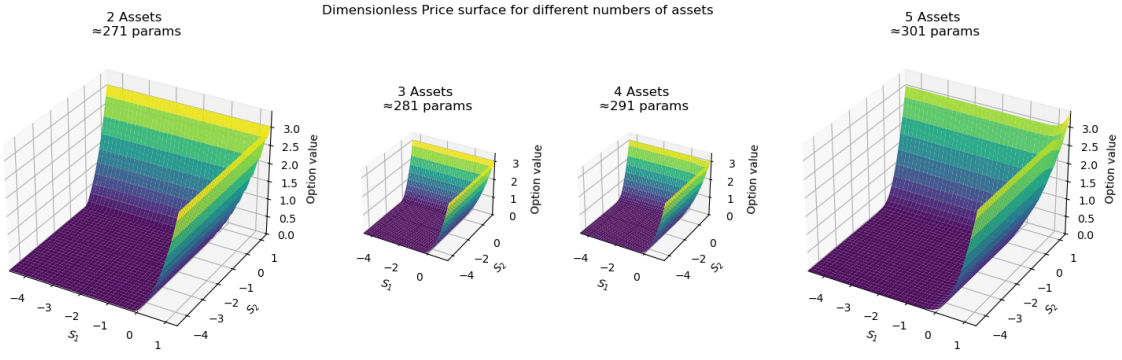
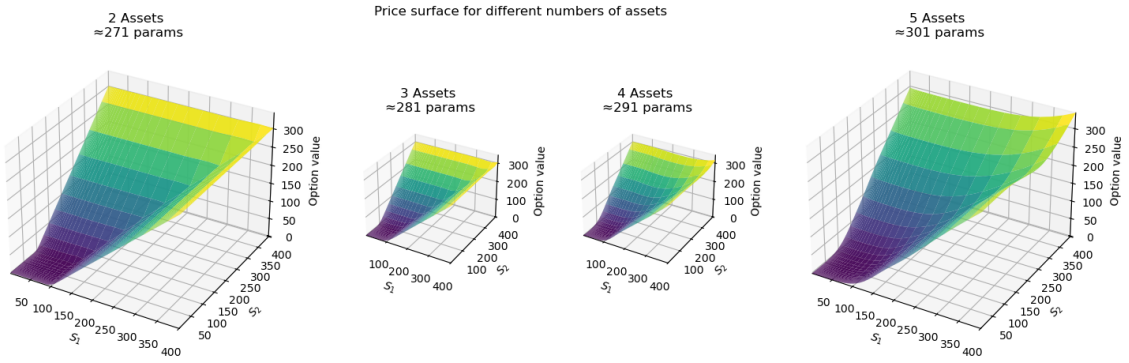


Figure 4.9

Multi-asset call option price in original coordinates (first two asset axes).

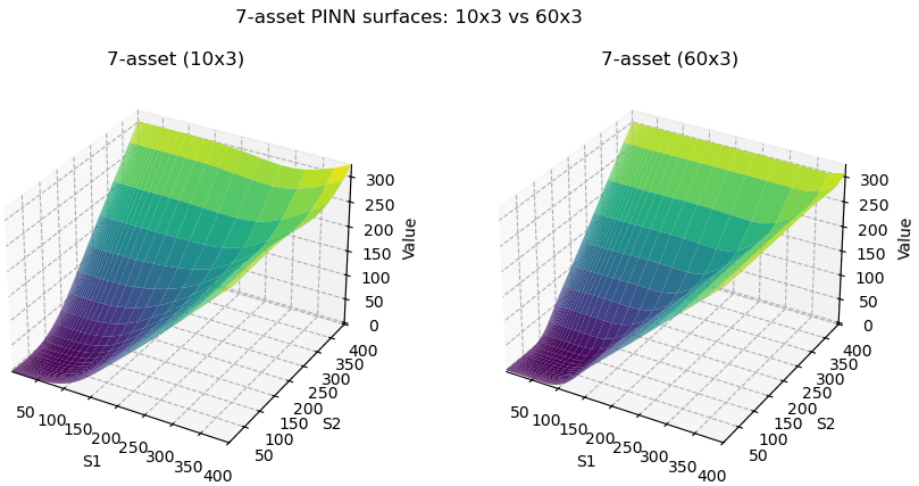


At the boundaries of the domain, especially in cases with more than three assets, the solution begins to exhibit an undesired curvature because the small network struggles to capture accurately the solution. In our experiments, increasing the number of boundary collocation points did not lead to significant improvements. Instead, better accuracy was achieved by increasing the expressiveness (width and depth) of the network.

By retaining the same number of layers but increasing the width to 60 neurons per layer, the network was able to represent the solution more faithfully, as demonstrated in Figure 4.10, which compares both architectures in a seven-asset configuration.

Figure 4.10

Seven-asset call option surfaces for two architectures. The network with 60 neurons per layer captures the solution more faithfully than the smaller 10-neuron model.

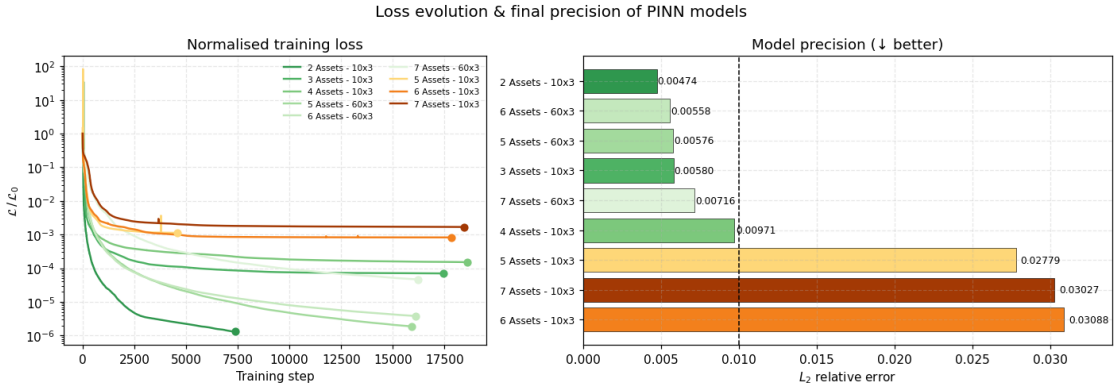


With the larger network and careful tuning of hyperparameters, it was possible to achieve a relative ℓ_2 error below 10^{-3} (or better, depending on the case), which is generally considered acceptable for most practical applications. However, it is worth noting that

such accuracy was not guaranteed across all dimensions and required repeated experimentation to achieve. Moreover, although the final error levels are satisfactory, there remains a lack of a comprehensive theoretical framework to rigorously bound the approximation error in higher dimensions. Figure 4.11 illustrates the convergence behaviour, where it is evident that larger networks converge more slowly but more reliably. In contrast, smaller networks occasionally terminate early due to reaching capacity limits, which may lead to misleadingly short training times but significantly worse solutions.

Figure 4.11

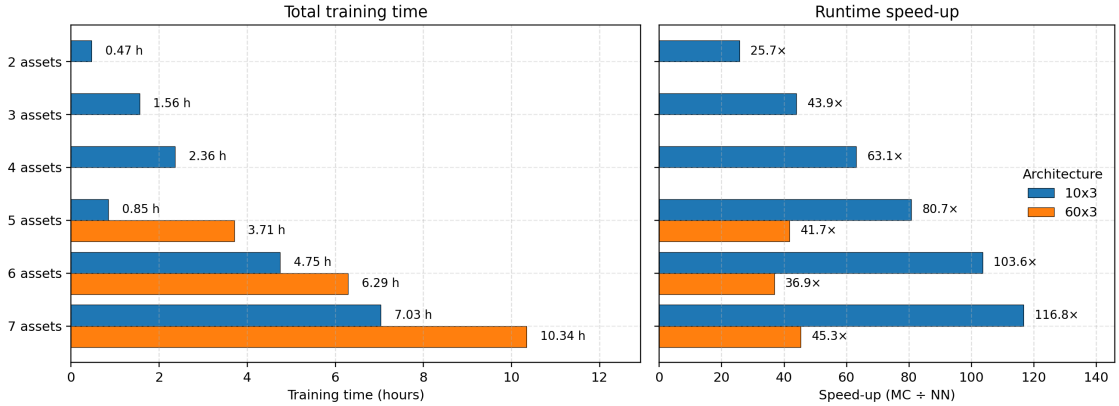
Training loss for multi-asset PINNs. Higher dimensions slow convergence and flatten the loss curve. The smaller network sometimes terminates early, yielding a shorter training time but a noticeably larger error.



Once trained, PINNs offer excellent performance in terms of computational speed. As shown in Figure 4.12, the inference time of a trained PINN is significantly faster than traditional Monte Carlo simulation methods, achieving substantial speedups even on a standard CPU. This computational advantage makes PINNs particularly attractive for real-time evaluations and repeated queries.

Figure 4.12

CPU timing comparison between PINN inference and Monte Carlo simulation (50000 simulations).



Despite the impressive inference speed, the long training times required for PINNs present a major limitation in professional financial settings. In practice, model parameters such as volatilities, correlations, and interest rates are frequently recalibrated to reflect changing market conditions.

Additionally, option-specific inputs such as strike prices and maturities may vary across instruments or need to be adjusted in scenario analyses. In such environments, it is common to perform rapid recalculations or sensitivity analyses, which demand flexible and responsive models. The requirement to retrain a PINN from scratch for each change in input parameters makes the method impractical for many real-time applications. To be viable in professional workflows, a PINN-based solution would either need to retrain extremely quickly, generalize across a wide range of parameter configurations, or be integrated selectively into systems that can tolerate delayed response times.

4.4. Call Option with Stochastic Volatility and Interest Rates

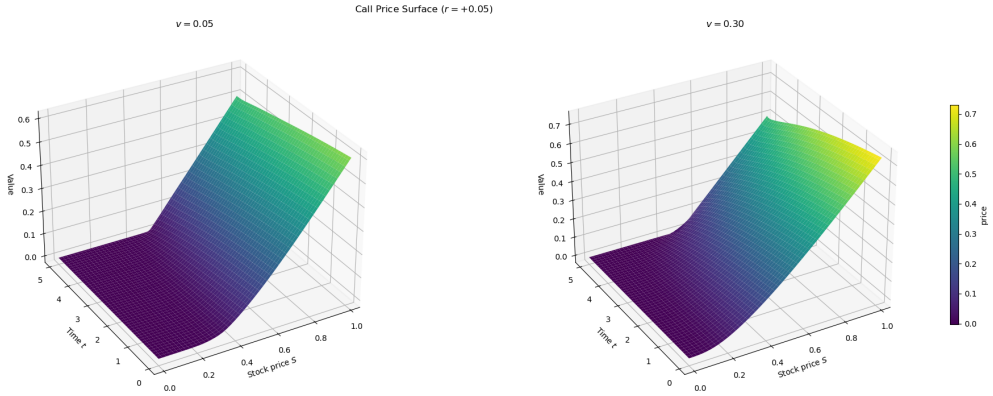
We now extend our previous examples by introducing stochastic volatility and stochastic interest rates using the Heston-Hull-White model. This combined framework allows us to capture realistic market features, including volatility smiles and interest rate fluctuations. Figure 4.13 illustrates the price surfaces produced by the PINN under various volatility (v) levels. The complete set of parameters used for this scenario is provided in Table 4.4.

Parameter	**Value**
Strike price, K	\$0.5
Maturity, T	5.0 years
Initial risk-free short rate, r_0	0.05
Hull-White mean reversion speed, a_r	0.1
Hull-White mean reversion level, θ_r	0.05
Hull-White interest rate volatility, σ_r	0.02
Heston initial volatility, v_0	0.04
Heston mean reversion speed, κ_v	1.5
Heston long-term volatility, θ_v	0.04
Heston volatility of volatility, σ_v	0.2
Correlation between asset and volatility, ρ_{Sv}	-0.7
Correlation between asset and interest rate, ρ_{Sr}	0.0
Correlation between volatility and interest rate, ρ_{vr}	0.0

Table 4.4
Parameters for the Call Option under the Heston-Hull-White Model

Figure 4.13

Call option pricing surfaces using the Heston-Hull-White model for different volatility levels.

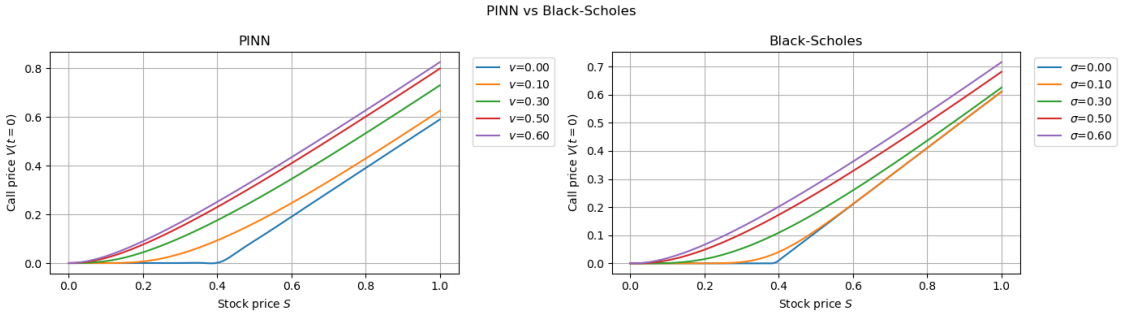


Using the experience from the case of Section 2.2.3, we used a neural network with three hidden layers, each containing 60 neurons, and trained it with the BFGS optimizer on 13 320 collocation points distributed equitatively across the different domains. Training took approximately 12 minutes, which contrast with the multi-dimensional case for the same number of dimensions.

Since no analytical solution exists for the combined Heston-Hull-White model, we qualitatively compare the PINN results to those obtained using the simpler Black-Scholes model (see Figure 4.14). It is possible to observe that for high underlying asset values, option prices behave nearly linearly with the underlying; at lower values, the price closely resembles the discounted payoff.

Figure 4.14

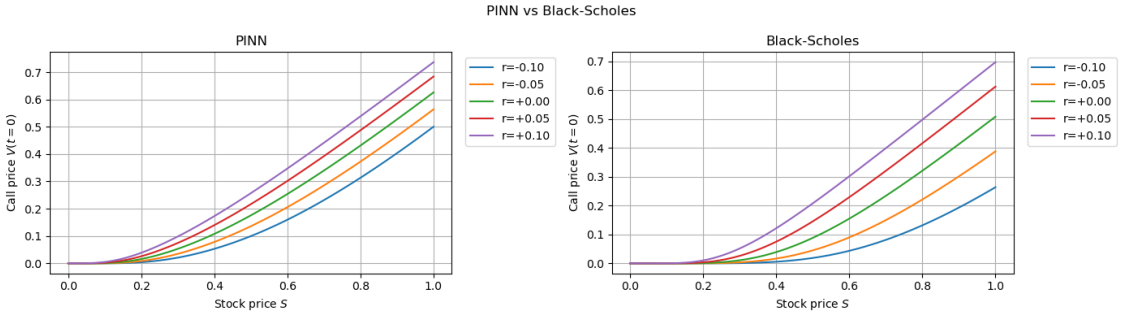
Price sensitivity to changes in volatility (v). Higher values decrease curvature in the price surface.



The effect of interest-rate dynamics on option pricing is shown in Figure 4.15. As short-term interest rates increase, option prices generally rise, reflecting the increased cost associated with hedging positions and financing derivatives contracts over longer horizons. This aligns with standard market expectations.

Figure 4.15

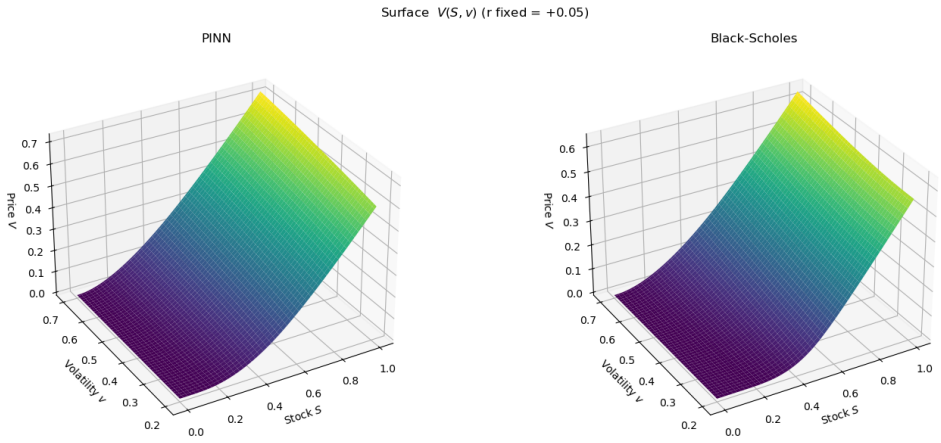
Price sensitivity to short rates in the Heston-Hull-White model. Higher interest rates raise the option's hedging cost, increasing the option price.



As the model incorporates stochastic volatility and interest rates, the resulting price surfaces differe from the Black-Scholes solution, both because additional stochastic factors are present and because the combined model incorporates correlations between the underlying asset, volatility, and interest rates. This can be seen in Figure 4.16 and Figure 4.17, where the price surfaces are shown at $t = 0$ for the stock and volatility dimensions, and the stock and interest rate dimensions, respectively.

Figure 4.16

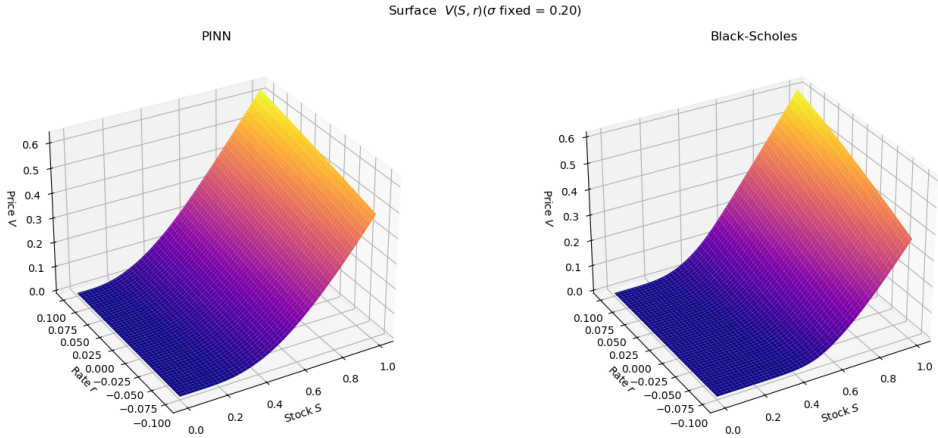
Price surface at $t = 0$ for the stock and volatility dimension.



Parameter	Value
Strike price, K	\$100
Maturity, T	1.0 year
Risk-free rate, r	0.05
Volatility, σ	0.20

Table 4.5
Parameters for the European Put Option

Figure 4.17
Price surface at $t = 0$ for the stock and volatility dimension.



This experiment demonstrates that PINNs can accommodate richer dynamics than the Black-Scholes setting while retaining rapid inference once training is complete—a property valuable in latency-sensitive contexts such as high-frequency trading, where models must reconcile speed with the need to reflect joint dynamics.

4.5. Put Option Pricing

In this section, we aim to analyze early excersice features, where the most common example is the American put option, but we will also consider European put options for comparison. Puts are characterized by their distinct payoff function

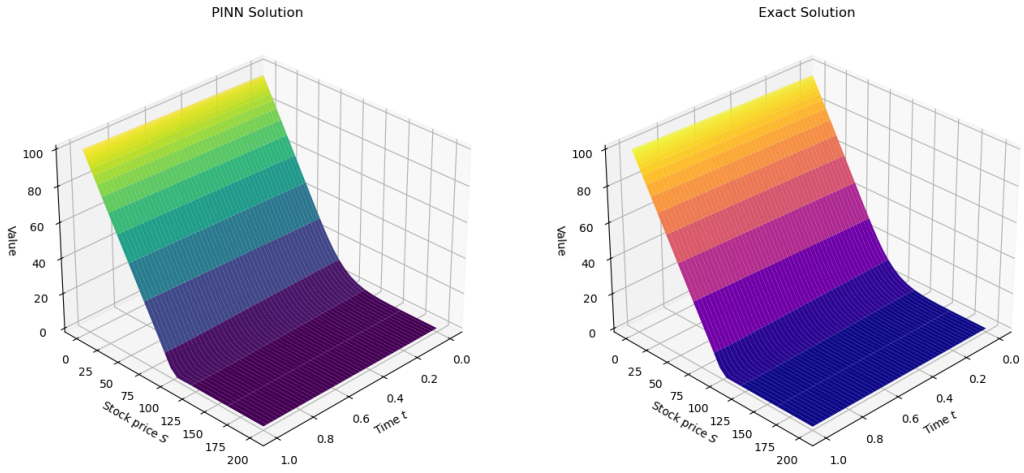
$$\Phi(S) = \max(K - S, 0).$$

For this section, the parameters used are summarized in Table 4.5.

The pricing surface obtained from the PINN compared to the exact Black-Scholes analytical solution is displayed in Figure 4.18. Unlike call options, put option prices increase as the underlying asset value decreases, reflecting their payoff structure. Therefore, we need to exchange the boundary conditions when training the neural network and comparing to the call option-when $S \to \infty$, the prices behaves linearly and when $S \to 0$.

Figure 4.18

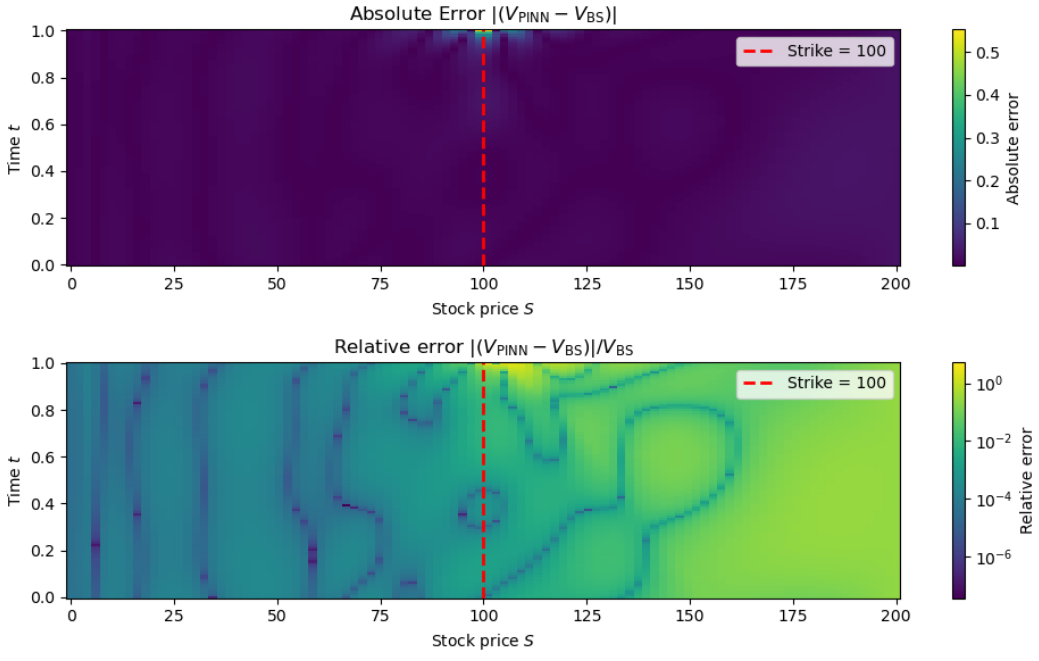
European put option pricing: comparison between PINN (left) and analytical Black-Scholes solution (right).



The absolute error in the put option pricing is shown in Figure 4.19, revealing error concentrations near maturity and around the strike price, consistent with the call option.

Figure 4.19

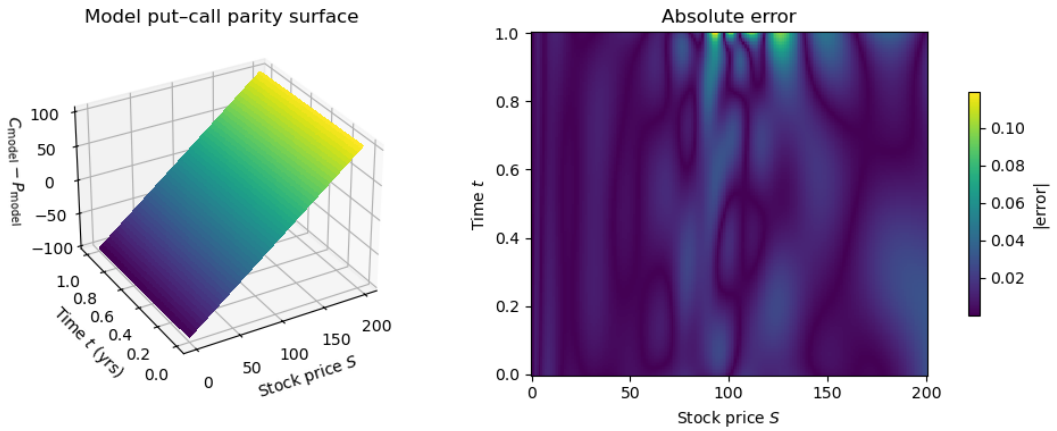
Absolute error distribution for the European put option. Errors peak around maturity near the strike price.



Now that we have the solution of both the put and call option, we can use the so-called put-call parity to compare the overall results. For European contracts the parity reads

$$C(S,K,t,T) - P(S,K,t,T) = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}[S_T - K] = S_t - Ke^{-r(T-t)},$$

Figure 4.20
Put-call parity verification: the difference between the PINN solutions for call and put options matches the expected relationship.



where C and P denote the call and put prices and S_t is the underlying price at evaluation time t. This relationship tells us that the difference between the call and put prices should equal the discounted expected value of the underlying asset at maturity minus the strike price where no stochasticity is present, producing a linear relationship between the underlying asset price and the difference between the call and put prices. Evaluating the left-hand side with the two PINN surfaces confirms the identity to high accuracy: the relative ℓ_2 error of the parity is 5.63×10^{-5}.

In future applications, this parity can be of use for adding additional constraints to the PINN, ensuring that the solutions for call and put options are consistent with the theoretical relationship.

In this particular case, we limit our selves to state the ℓ_2 error obtained by the PINN, 5.85×10^{-5}, as the case is very similar to the call option.

4.6. American Put Option Pricing

An American put option introduces complexity due to its early-exercise feature, allowing the holder to exercise anytime before maturity. Consequently, pricing requires solving a free-boundary problem.

There are various ways to incorporate into the modeling framework free-boundary restrictions. In this experiment we achieve this by instead of directly minimizing the PDE interior residual ($\mathcal{L}_{\text{PDE}}$), we minimize the following loss function:

$$\mathcal{L}_{\text{PDE}}(\theta) = \int_0^T \int_\Omega |\max\{\Phi(S) - u_\theta(t,x), \mathcal{N}_I[u_\theta(t,x)]\}|^2 \, \mathrm{d}x\mathrm{d}t. \tag{4.1}$$

More on this can be found in [28], [29]. The resulting PINN solution, compared

against a benchmark numerical solution (FDM), is shown in Figure 4.21. In this case, we used the same neural network architecture and parameters as the European put option.

Figure 4.21

American put option prices: PINN solution (left) compared to a numerical benchmark solution (right).

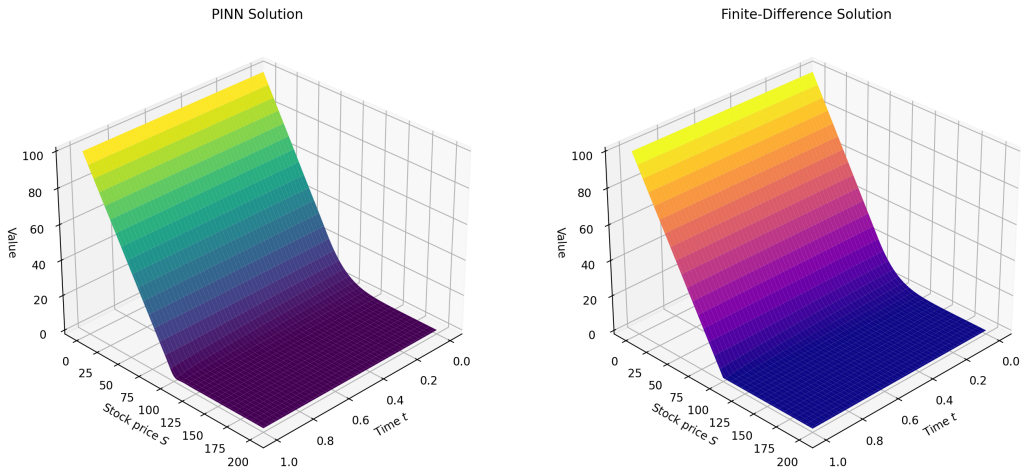
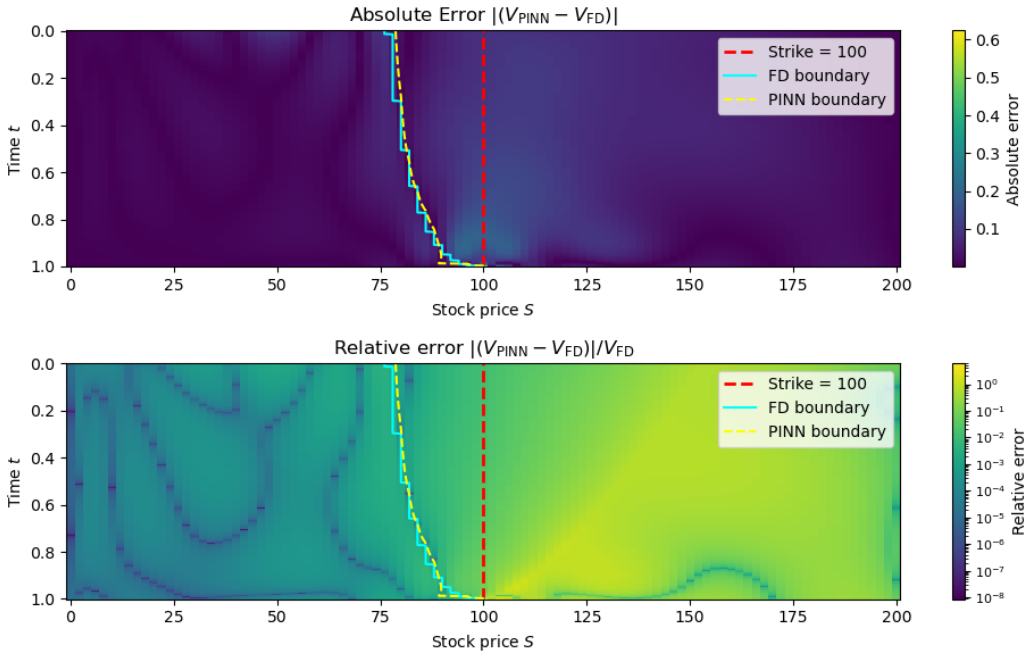


Figure 4.22 shows the absolute error in pricing the American put. The error is again concentrated around the strike price, though errors now extend across a wider region due to the early-exercise boundary's dynamic nature.

Figure 4.22

Errors for American put option pricing. Is possible to see that the PINN is able to infer correctly most of the free boundary.



From Figure 4.22, we can see that the PINN is able to achieve good prediction of

the free boundary, highlited by the blue and yellow lines,, which is the main challenge in American option pricing. Still, some artifacts are visible, especially near the strike price.

In terms of the accuracy of the obtained solution, the PINN achieved an ℓ_2 error of 1.30×10^{-3}, which is acceptable for practical applications, but worse than the European cases. This this can be attributed to the new loss function, which introduces a maximun operator inside the integrand of the loss, making the PDE residual less smooth and harder to approximate with a neural network.

4.7. Swaption Pricing

The final derivative instrument we evaluate is a European *payer swaption*, which gives the holder the right, but not the obligation, to enter into a fixed-for-floating interest rate swap at a specified future date. The underlying model used for pricing is the one-factor Hull-White model, as described in Section 2.2.5.

The semi-analytical benchmark formula in the one-factor Hull-White model, following the notation of [16] is presented below. For a European *payer* swaption with option maturity $T \equiv T_0$, payment dates $\mathcal{T} = \{T_1, \ldots, T_n\}(T_n > T)$ and fixed rate X, let $\tau_i = T_i - T_{i-1}$ and define the coupon weights

$$c_i := X\tau_i, \quad i = 1, \ldots, n-1, \qquad c_n := 1 + X\tau_n.$$

With $p(t,T) = A(t,T)e^{-B(t,T)r_t}$ denoting the Hull-White zero-coupon bond price, Jamshidian's decomposition expresses the swaption payoff as the sum of n call options on zero-coupon bonds. Let $r^\star$ be the unique root of

$$\sum_{i=1}^{n} c_i A(T,T_i)e^{-B(T,T_i)r^\star} = 1,$$

and set $X_i^\star = A(T,T_i)e^{-B(T,T_i)r^\star}$. Then, the pricing formula is given by

$$\mathrm{PS}_{\mathrm{pay}}(t,T,\mathcal{T},X) = \sum_{i=1}^{n} c_i \,\mathrm{ZBP}(t,T,T_i,X_i^\star),$$

where the Hull-White price of a put option with underlying a zero-coupon bond is

$$\mathrm{ZBP}(t,T,S,K) = P(t,S)^M\Phi(h) - XP(t,T)^M\Phi(h-\sigma_p),$$

$$\sigma_p = \sigma\sqrt{\frac{1-e^{-2\kappa(T-t)}}{2\kappa}}B(T,S),$$

$$h = \frac{1}{\sigma_p}\ln\Big(\frac{P(t,S)^M}{P(t,T)^M X}\Big) + \frac{\sigma_p}{2}.$$

Here, $\Phi(\cdot)$ denotes a standar normal CDF and the parameters are the sames as in Section 2.2.5.

In this framework, the short rate r_t evolves according to the stochastic differential equation defined in 2.23. The swaption price depends on the future evolution of the short rate and on the value of the underlying swap, which itself is the present value of fixed versus floating rate payments over the swap's tenor.

Figure 4.23

European payer swaption pricing surface. Left: PINN solution. Right: reference solution computed with an analytic approximation or semi-analytical method.

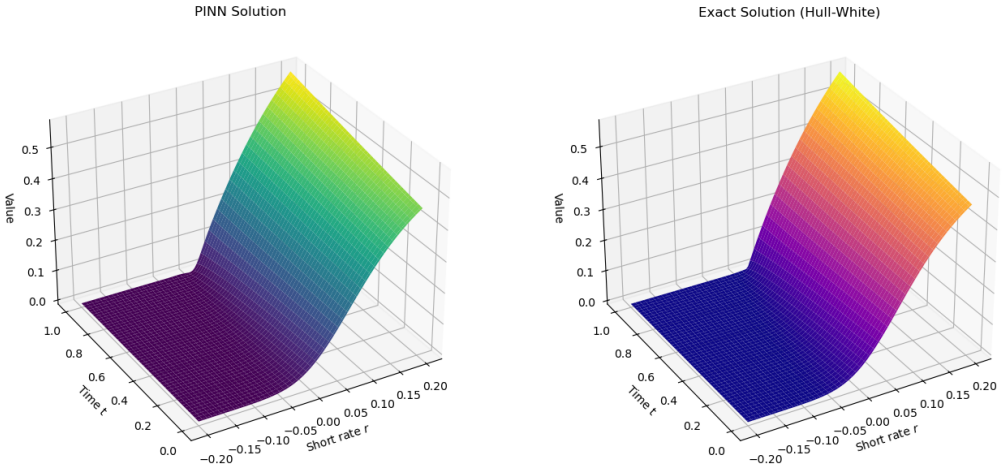
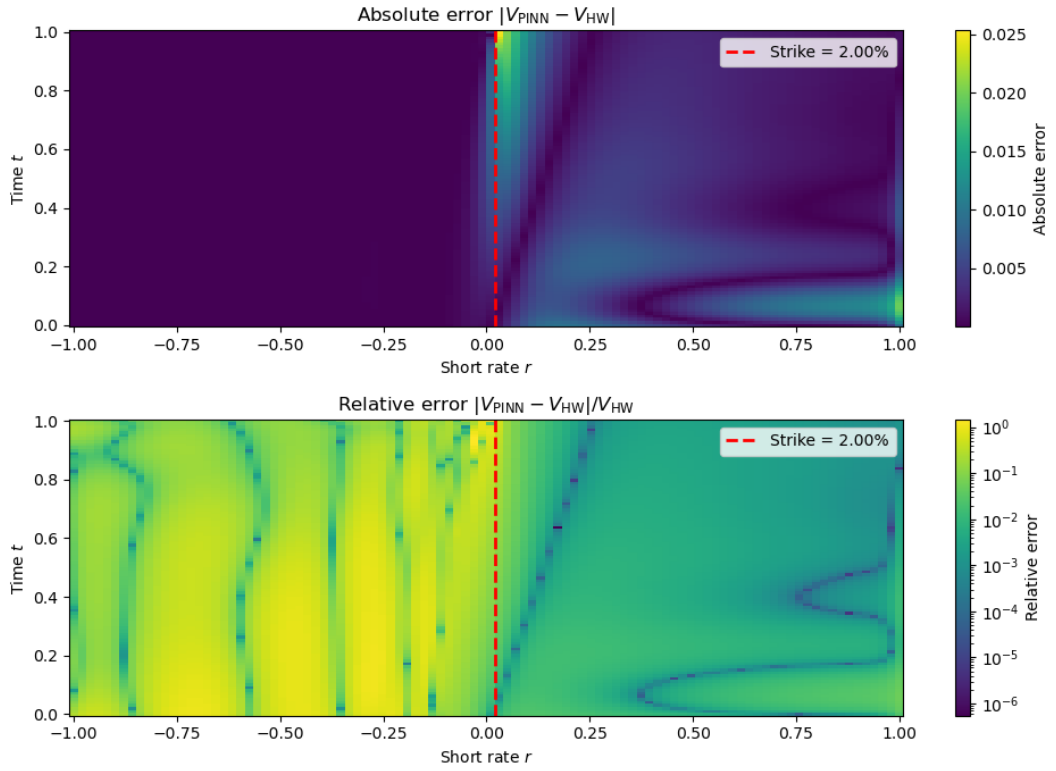


Figure 4.23 shows the pricing surface of a European swaption obtained using a PINN, compared against a benchmark solution from a conventional method. The neural network accurately captures the swaption price profile, particularly across key regions of the short-rate and strike axes. We deliverately choose a wide range for r to show the difference in the price surface, as it has non-linear features.

The absolute error of the PINN solution is shown in Figure 4.24. Most of the error is concentrated, again, near the strike region and close to expiry—areas where option payoffs exhibit low smoothness or where the valuation is highly sensitive to changes in rates.

Figure 4.24

Absolute error in pricing the European payer swaption. The error is primarily concentrated near expiry and around the strike.



For this particular case, the PINN achieved an ℓ_2 error of 5.7×10^{-3}. In general, the PINN demonstrates strong agreement with analytical benchmarks. This confirms its ability to handle the pricing of interest-rate derivatives with path-dependent structures and stochastic dynamics.

5. DISCUSSION, LIMITATIONS, AND OUTLOOK

5.1. Discussion of the Main Findings

The empirical evidence gathered in the preceding chapter confirms that PINNs can replicate benchmark prices for a broad family of derivative contracts while delivering near-instantaneous inference once training is complete.

d	Net Size	Instrument	Model	Training Time (hr)	Ω	$\partial\Omega$	Ω_0	ℓ_2 error (%)
2	2x20x20	Call	BS	0.12	25,000	25,000	25,000	0.023
2	2x20x20	Put	BS	0.07	25,000	25,000	25,000	0.006
2	2x20x20	American Put	BS	0.09	25,000	25,000	25,000	0.132
4	4x60x60	Call	Heston-HW	0.20	10,000	1,660	1,660	N/A
2	2x60x60	Swaption	HW	0.07	25,000	25,000	25,000	0.58
3	3x10x10	Best-Of	BS	0.47	500	2,000	500	0.474
4	4x10x10	Best-Of	BS	1.56	500	3,000	500	0.580
5	5x10x10	Best-Of	BS	2.36	500	4,000	500	0.971
6	6x10x10	Best-Of	BS	0.85	500	5,000	500	2.779
7	7x10x10	Best-Of	BS	4.75	500	6,000	500	3.088
8	8x10x10	Best-Of	BS	7.03	500	7,000	500	3.027
6	6x60x60	Best-Of	BS	3.71	500	5,000	500	0.576
7	7x60x60	Best-Of	BS	6.29	500	6,000	500	0.558
8	8x60x60	Best-Of	BS	10.34	500	7,000	500	0.716

Table 5.1
PINN training results across models and dimensions.

For low-dimensional problems—the European call and put under Black-Scholes, the one-factor Hull-White swaption, and the three-state Heston-Hull-White call—the trained networks attained relative ℓ_2 errors below 10^{-3} and reproduced the qualitative features of the true pricing surfaces, including the expected linear growth deep-in-the-money and the curvature induced by stochastic volatility and rates. In all those cases a single forward pass through the network was between one and two orders of magnitude faster than a finite-difference or Monte-Carlo solver executed on the same hardware.

This results are promising, as they suggest that PINNs can serve as viable surrogates for traditional numerical methods in latency-sensitive applications, such as high-frequency quoting engines, where the speed of inference is critical. Also, we can envision use cases where PINNs could be integrated into larger systems, such as XVA or initial margin calculators, where the ability to quickly evaluate a wide range of contracts is essential. Another use case, not explored in this document, is the use of PINNs to solve inverse problems, such as calibrating model parameters to market data.

Still, care must be taken when applying PINNs to real-world cases. As shown in this document, there are challanges and limitations that need to be addressed before PINNs can be considered a fully-fledged alternative to traditional numerical methods.

First, the training process is computationally expensive, often requiring several minutes or even hours to converge to reasonable precissions, depending on the complexity of the problem and the architecture of the neural network. This initial cost can be a significant barrier to adoption in environments where rapid recalibration is necessary.

Also, performance deteriorated as soon as one of three stress factors was introduced: an increase in state dimension, the presence of kinks or free boundaries, or the need to retrain for new market parameters. The multi-asset experiment illustrates the first effect and the swaption case the second. After widening the hidden layers to sixty neurons the network required almost the same number of iterations to converge, but the complexity of the problem made each iteration more expensive, leading to a significant increase in wall-clock time. In all the cases, the residual error concentrated near the payoff ridge, a manifestation of the Gibbs phenomenon that smooth activations cannot suppress.

The third limitation—lack of transferability—emerges whenever the strike, maturity or other model parameters changed. Because a classical PINN encodes one particular solution rather than the pricing operator itself, every change forced a full optimisation run whose wall-clock cost erodes the apparent speed-up achieved at inference.

5.2. Further Research

Some of the current limitations of PINNs are the subject of ongoing research, with several promising directions aimed at enhancing their performance in financial applications. Notable avenues include:

- *Operator-learning approaches*, such as Fourier Neural Operators [30] and DeepONets [31], integrate physics constraints into architectures designed to learn mappings from model parameters to solutions. These methods amortize training costs by producing models that generalize across a range of parameter values, eliminating the need to retrain for each scenario. A particularly promising example is [32], which demonstrates both improved accuracy on standard benchmarks and the potential of operator learning in tackling nonlinear PDEs.

- *Domain-decomposition strategies*, including XPINNs [33], divide the space-time domain into smaller sub-domains that can be trained in parallel. This approach mitigates the curse of dimensionality while managing memory requirements effectively. By decomposing the problem into smaller, tractable components, it enables more scalable training and faster inference, making it particularly appealing for large-scale or high-dimensional problems.

- *Transfer learning techniques*, such as those explored in [34], [35], aim to reuse learned representations to adapt to new problem instances without full retraining. This is especially useful in settings where training is computationally intensive, en-

abling PINNs to efficiently generalize to new market conditions or contract structures with minimal additional cost.

- *Hard constraints, adaptive sampling, and curriculum learning* are complementary strategies that improve training efficiency and accuracy. Hard constraints [36] enforce boundary conditions—such as payoff profiles at maturity—explicitly, reducing the burden on the network to learn them implicitly. Adaptive sampling [22] focuses training effort on regions of the domain with higher error or greater complexity, thereby improving overall accuracy without increasing the number of collocation points. Curriculum learning [37] introduces training examples in increasing order of complexity, allowing the network to first master simpler patterns before addressing more challenging ones, improving convergence and generalization.

Collectively, these developments suggest a promising future for PINNs in quantitative finance. For one- and two-factor models, PINNs already offer competitive accuracy and unmatched speed after training, making them strong candidates for deployment in latency-sensitive applications such as XVA computation or high-frequency pricing. However, scaling to higher-dimensional contracts or environments requiring frequent recalibration will demand architectural innovations that support parameter generalization, provide *a posteriori* error estimates, and ensure robust optimization dynamics. If these challenges are met, physics-informed learning could become a foundational tool in next-generation quantitative finance infrastructure, offering a compelling blend of theoretical rigor and data-driven adaptability.

BIBLIOGRAPHY

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, May 2015. DOI: 10.1038/nature14539.

[2] J. Hull, *Options, futures, and other derivatives*, eng, Ninth edition, Global edition. Boston: Pearson Education Limited, 2018, ISBN: 1292212896.

[3] P. Wilmott, "Paul wilmott on quantitative finance," 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:153668090.

[4] P. Glasserman, *Monte Carlo methods in financial engineering*. New York: Springer, 2004, ISBN: 0387004513 9780387004518 1441918221 9781441918222. [Online]. Available: http://www.amazon.com/Financial-Engineering-Stochastic-Modelling-Probability/dp/0387004513/ref=pd_sim_b_68?ie=UTF8&refRID=1AN8JXSDGMEV2RPHFC2A.

[5] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 637–654, 1973, ISSN: 00223808, 1537534X. [Online]. Available: http://www.jstor.org/stable/1831029 (visited on 04/25/2025).

[6] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[7] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019, ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[8] F. Gatta, V. S. Di Cola, F. Giampaolo, F. Piccialli, and S. Cuomo, "Meshless methods for american option pricing through physics-informed neural networks," *Engineering Analysis with Boundary Elements*, vol. 151, pp. 68–82, 2023, ISSN: 0955-7997. DOI: https://doi.org/10.1016/j.enganabound.2023.02.040. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0955799723000978.

[9] B. Salvador, C. W. Oosterlee, and R. van der Meer, "Financial option valuation by unsupervised learning with artificial neural networks," *Mathematics*, vol. 9, no. 1, 2021, ISSN: 2227-7390. DOI: 10.3390/math9010046. [Online]. Available: https://www.mdpi.com/2227-7390/9/1/46.

[10] S. L. Heston, "A closed-form solution for options with stochastic volatility with applications to bond and currency options," *The Review of Financial Studies*, vol. 6, no. 2, pp. 327–343, 1993, ISSN: 08939454, 14657368. [Online]. Available: http://www.jstor.org/stable/2962057 (visited on 07/06/2025).

[11] B. Huge and A. Savine, *Differential machine learning*, 2020. arXiv: 2005.02347 [q-fin.CP]. [Online]. Available: https://arxiv.org/abs/2005.02347.

[12] J. B. Heaton, N. G. Polson, and J. H. Witte, *Deep learning in finance*, 2018. arXiv: 1602.06561 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1602.06561.

[13] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 637–654, 1973, ISSN: 00223808, 1537534X. [Online]. Available: http://www.jstor.org/stable/1831029 (visited on 04/23/2025).

[14] T. Björk, *Arbitrage Theory in Continuous Time* (Oxford Finance Series). Oxford University Press, Incorporated, 2004, ISBN: 9780191533846. [Online]. Available: https://books.google.es/books?id=TJgjgJATeVIC.

[15] D. Duffy, *Numerical Methods in Computational Finance: A Partial Differential Equation (PDE/FDM) Approach* (Wiley Finance). Wiley, 2022, ISBN: 9781119719670. [Online]. Available: https://books.google.es/books?id=6cZ6EAAAQBAJ.

[16] D. Brigo and F. Mercurio, *Interest Rate Models Theory and Practice* (Springer Finance). Springer Berlin Heidelberg, 2013, ISBN: 9783662045534. [Online]. Available: https://books.google.es/books?id=USvrCAAAQBAJ.

[17] J. Hull and A. White, "Pricing interest-rate-derivative securities," *The Review of Financial Studies*, vol. 3, no. 4, pp. 573–592, Apr. 2015, ISSN: 0893-9454. DOI: 10.1093/rfs/3.4.573. eprint: https://academic.oup.com/rfs/article-pdf/3/4/573/24416170/030573.pdf. [Online]. Available: https://doi.org/10.1093/rfs/3.4.573.

[18] S. Longstaff, "Valuing american options by simulation: A simple least squares approach," *Review of Finance Studies*, vol. 14, pp. 113–147, Jan. 2001.

[19] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1412.6980.

[20] J. F. Urbán, P. Stefanou, and J. A. Pons, "Unveiling the optimization process of physics informed neural networks: How accurate and competitive can pinns be?" *Journal of Computational Physics*, vol. 523, p. 113 656, Feb. 2025, ISSN: 0021-9991. DOI: 10.1016/j.jcp.2024.113656. [Online]. Available: http://dx.doi.org/10.1016/j.jcp.2024.113656.

[21] Z. Xiang, W. Peng, X. Liu, and W. Yao, "Self-adaptive loss balanced physics-informed neural networks," *Neurocomputing*, vol. 496, pp. 11–34, 2022, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.05.015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523122200546X.

[22] C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu, "A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 403, p. 115 671, 2023, ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2022.115671. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782522006260.

[23] N. Malekjani, R. Kharaghani, and E. Tsotsas, "A comparative study of dimensional and non-dimensional inputs in physics-informed and data-driven neural networks for single-droplet evaporation," *Chemical Engineering Science*, vol. 306, p. 121 214, Jan. 2025. DOI: 10.1016/j.ces.2025.121214.

[24] J. Melo, *Leveraging physics-informed neural networks for option pricing problems*, https://github.com/jmelo11/pricing_pinns, GitHub repository, accessed July 2025, 2025.

[25] J. Melo, *Options-pinns: A physics-informed neural network approach to option pricing*, https://jmelo11.github.io/notebook/options-pinns/, Accessed: 2025-07-24, 2025.

[26] R. Mattey and S. Ghosh, "A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 390, p. 114 474, Feb. 2022, ISSN: 0045-7825. DOI: 10.1016/j.cma.2021.114474. [Online]. Available: http://dx.doi.org/10.1016/j.cma.2021.114474.

[27] J. Stiasny and S. Chatzivasileiadis, "Physics-informed neural networks for time-domain simulations: Accuracy, computational cost, and flexibility," *Electric Power Systems Research*, vol. 224, p. 109 748, 2023, ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2023.109748. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378779623006375.

[28] P. Wilmott, S. Howison, and J. Dewynne, *The Mathematics of Financial Derivatives: A Student Introduction* (The Mathematics of Financial Derivatives: A Student Introduction). Cambridge University Press, 1995, ISBN: 9780521497893. [Online]. Available: https://books.google.es/books?id=VYVhnC3fIVEC.

[29] S. Ikonen and J. Toivanen, "Operator splitting methods for american option pricing," *Applied Mathematics Letters*, vol. 17, no. 7, pp. 809–814, 2004, ISSN: 0893-9659. DOI: https://doi.org/10.1016/j.aml.2004.06.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893965904804968.

[30] Z. Li, N. Kovachki, K. Azizzadenesheli, *et al.*, *Fourier neural operator for parametric partial differential equations*, 2021. arXiv: 2010.08895 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2010.08895.

[31] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," *Nature Machine Intelligence*, vol. 3, no. 3, pp. 218–229, 2021. DOI: 10.1038/s42256-021-00302-5. [Online]. Available: https://doi.org/10.1038/s42256-021-00302-5.

[32] H. Li, Y. Miao, Z. S. Khodaei, and M. Aliabadi, "An architectural analysis of deeponet and a general extension of the physics-informed deeponet model on solving nonlinear parametric partial differential equations," *Neurocomputing*, vol. 611, p. 128 675, 2025, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2024.128675. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231224014462.

[33] A. Jagtap D. and E. Karniadakis George, "Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," *Communications in Computational Physics*, vol. 28, no. 5, pp. 2002–2041, 2020, ISSN: 1991-7120. DOI: https://doi.org/10.4208/cicp.OA-2020-0164. [Online]. Available: https://global-sci.com/article/79747/extended-physics-informed-neural-networks-xpinns-a-generalized-space-time-domain-decomposition-based-deep-learning-framework-for-nonlinear-partial-differential-equations.

[34] R. Pellegrin, B. Bullwinkel, M. Mattheakis, and P. Protopapas, *Transfer learning with physics-informed neural networks for efficient simulation of branched flows*, 2022. arXiv: 2211.00214 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2211.00214.

[35] Y. Gao, K. C. Cheung, and M. K. Ng, "Svd-pinns: Transfer learning of physics-informed neural networks via singular value decomposition," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, Dec. 2022, pp. 1443–1450. DOI: 10.1109/ssci51031.2022.10022281. [Online]. Available: http://dx.doi.org/10.1109/SSCI51031.2022.10022281.

[36] X. Li, J. Deng, J. Wu, S. Zhang, W. Li, and Y.-G. Wang, "Physical informed neural networks with soft and hard boundary constraints for solving advection-diffusion equations using fourier expansions," *Computers and Mathematics with Applications*, vol. 159, pp. 60–75, 2024, ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2024.01.021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0898122124000348.

[37] Y. W. Bekele, *Physics-informed neural networks with curriculum training for poroelastic flow and deformation processes*, 2024. arXiv: 2404.13909 [cs.CE]. [Online]. Available: https://arxiv.org/abs/2404.13909.