Fourier transform and machine learning methods for the pricing of discretely monitored barrier options

Jiaqi Liang

A dissertation submitted in fulfillment of the requirements
for the degree of
Doctor of Philosophy
of
University College London.

Department of Computer Science

Primary Supervisor: Prof. Guido Germano
Secondary Supervisor: Dr. Carolyn E. Phelan

October 2, 2025

I, Jiaqi Liang, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Discretely monitored barrier options, observed only at specific monitoring dates, are common in practice but pose significant challenges to pricing due to their path dependence and the complexity introduced by discrete monitoring.

This thesis develops efficient and accurate methods for pricing discretely monitored barrier options under the Heston stochastic volatility model, which captures important market features such as volatility clustering and the implied volatility smile. Leveraging the Heston model's tractable characteristic function, two novel Fourier-based methods are proposed: the Fourier z-transform method (FZ), based on an extension of the Wiener-Hopf technique, and the recursive Fourier-Hilbert transform method (FH).

To support these techniques, the thesis includes a detailed analysis of two formulations of the joint conditional characteristic function of the Heston model, resolving discontinuities and ensuring numerical stability. Extensive numerical experiments validate the robustness and accuracy of the proposed methods across a wide range of parameter regimes, including challenging cases.

The thesis also explores the use of deep learning to approximate option prices. A neural network model is trained on data generated by the FZ method and demonstrates strong predictive performance, offering a scalable, data-driven alternative for pricing and calibration.

By combining analytical tractability, numerical efficiency, and machine learning, this thesis provides practical and adaptable tools for pricing discretely monitored barrier options.

Impact statement

Barrier options are widely traded by both speculators and hedgers due to their lower premiums compared to standard options. Meanwhile, stochastic volatility models have gained increasing popularity, particularly since the financial crisis, as they allow parameters to be calibrated to fit the smile and skew patterns observed in market volatility surfaces. The Fourier-based numerical schemes tailored to the Heston model, including the Fourier z-transform and Fourier-Hilbert transform methods for discretely monitored barrier options developed in this thesis provides practitioners with greater flexibility in fitting pricing models to observed market data. This approach has been presented at several international conferences attended by both academic researchers and industry practitioners.

In recent years, machine learning techniques have become increasingly prominent due to advancements in computational power and the availability of large datasets. One project in this thesis applied a neural network-based regression technique to learn pricing behaviour of Heston-based Fourier z-transform method for discrete barrier options. This approach can be easily extended to other types of financial derivatives and has potential applications in industrial settings. Moreover, the analysis and discussion of hyperparameter tuning strategies conducted in this thesis provide practical insights that can be adapted and developed further in industry applications.

Finally, joint conditional characteristic functions are a crucial component of Fourier transform-based pricing methods widely used in the financial industry. This thesis validated two formulations of the Heston model JCCF and corrected the complex discontinuity present in one expression, thereby improving the reliability and accuracy of Fourier-based pricing implementations. A new algorithm is also proposed to correct complex discontinuities in a widely used formulation of the Heston characteristic function, enhancing numerical stability.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor, Prof. Guido Germano for his patient support and valuable guidance and feedback through out my Ph.D. study and research.

I would also like to thank my secondary supervisor Dr. Carolyn E. Phelan and Prof. Daniele Marazzina along with my supervisor for enlightening me and providing idea, comments and supports for my research.

Last but not the least, I would like to thank my parents, my grandparents, my boyfriend Dr. Zheng Gao, my friends Mengrui Li, Dr. Weiguan Wang, Dr. Kangjianan Xi, Dr. Yiran Cui, Dr. Yiwei Guan, and Dr. Yupeng Jiang for supporting me throughout writing this thesis and my life in general.

A summary of my research preprints, presentations, teaching, and professional activities during the PhD can be found in Appendix C.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Barrier options are a class of exotic derivatives whose payoff structure depends on whether the underlying asset price breaches a predetermined level during the contract's lifespan. Knock-in barrier options activate only if the underlying asset reaches a specified price, while knock-out options cease to exist if the asset crosses a given threshold. A key distinction lies in the monitoring method: whether the barrier is observed continuously or at discrete intervals. In practice, most traded barrier options are discretely monitored—typically at the close of each monitoring day—due to market conventions and operational constraints.

Discretely monitored barrier options are widely used in financial markets because they offer tailored risk management strategies at a lower cost compared to vanilla options. However, their path-dependent nature, combined with discrete monitoring, introduces additional pricing challenges, making their valuation an ongoing and dynamic area of research in quantitative finance, with significant theoretical and practical implications.

The primary objective of this thesis is to develop efficient and accurate pricing methodologies for discretely monitored barrier options under realistic market conditions. To achieve this, the Heston stochastic volatility model (Heston, 1993) is employed to model the underlying asset dynamics. The Heston model captures key empirical features observed in financial markets, such as volatility clustering, mean reversion, and the implied volatility smile. Additionally, it retains analytical tractability through a closed-form expression for the characteristic function, which facilitates the development of efficient Fourier-based pricing methods.

This thesis introduces two novel Fourier-based pricing methods under the Heston model, designed to address the challenges posed by path dependence and discrete monitoring, while maintaining both computational efficiency and analytical tractability. Although developed specifically for the Heston model, the proposed techniques are theoretically applicable to affine stochastic volatility models that admit closed-form joint characteristic functions for the log-price and variance, conditional on the current variance.

Additionally, the thesis explores the application of deep learning to barrier option pricing. Neural network architectures are investigated for their ability to learn the mapping from market and model parameters to option prices, providing fast and accurate approximations once trained. This data-driven approach complements traditional numerical methods, offering scalability, flexibility, and the potential to generalise across different market conditions and contract specifications. The deep learning approach is highly adaptable, making it suitable for pricing a broad range of exotic options beyond barrier contracts.

The literature on pricing barrier options under realistic asset dynamics is extensive. Methods for discretely monitored options under general Lévy processes are provided by Fusai et al. (2016) and Feng and Linetsky (2008), while stochastic volatility models like the Heston model (Heston, 1993), and jump-diffusion models like the Merton model (Merton, 1976), have become widely used for their ability to capture the empirical features of asset returns. Various numerical and semi-analytical techniques, including Monte Carlo simulation, Fourier-based methods, partial differential equation (PDE) approaches, and fast Hilbert transform methods, have been developed to price barrier options under these models. The COS method (Fang and Oosterlee, 2009a), for example, is a Fourier-based approach that has been successfully extended to discretely monitored barrier options. Each of these methods offers distinct trade-offs in computational efficiency, accuracy, and flexibility. While Monte Carlo methods are versatile, they can suffer from slow convergence when applied to path-dependent structures. Fourier-based methods provide computational efficiency but require careful handling of characteristic functions. PDE-based approaches offer high accuracy under continuous monitoring but are challenging for complex models.

These diverse methodologies highlight the ongoing need for robust and adaptable pricing frameworks, to which this thesis contributes.

This thesis is organised as follows:

Chapter 2 provides a comprehensive review of the literature on barrier options, surveying existing pricing methodologies and highlighting their strengths and limitations. It also presents the technical background for the various models and techniques employed later in the thesis, aiming to make the thesis more self-contained and serve as a refresher for readers.

Chapter 3 focuses on the Fourier transform pricing approach, with particular emphasis on the joint conditional characteristic function (JCCF) of the Heston model. Two distinct JCCF formulations are compared and validated, and complex discontinuity problem is addressed to ensure numerical stability and accuracy in option pricing.

Chapter 4 introduces two Fourier-based methods for pricing discretely monitored barrier options under the Heston model: the Fourier z-transform method (FZ), an extension of the Wiener-Hopf technique, and the recursive Fourier-Hilbert transform method (FH). Both incorporate spectral filtering and the fractional Fourier transform to improve accuracy. FH converges faster with fewer grid points, while FZ requires finer grids but is better suited for large-scale pricing tasks.

Chapter 5 explores recent advancements in deep learning for option pricing. A neural network model is trained on data generated by the FZ method to approximate option prices. The chapter discusses the model architecture, hyperparameter tuning, and performance evaluation against conventional pricing techniques, demonstrating the effectiveness of this method.

Finally, Chapter 6 concludes the thesis by summarising the key contributions and suggesting potential directions for future research in barrier option pricing.

Chapter 2

Literature review and technical background

Barrier options, as a subset of exotic options, have gained substantial attention in both academic research and financial practice due to their path-dependent nature and relatively lower premiums compared to vanilla options. Their payoffs depend not only on the terminal price of the underlying asset but also on whether the asset price breaches certain predetermined barrier levels during the option's lifetime.

Discretely monitored barrier options, where the barrier condition is evaluated at discrete time intervals—typically at the close of each monitoring day—are particularly relevant in real-world markets, as continuous monitoring is often impractical due to technical, operational, or contractual constraints.

The pricing of discretely monitored barrier options poses significant analytical and computational challenges. Unlike continuously monitored barriers, which sometimes admit closed-form solutions under specific assumptions, discrete monitoring breaks the Markov property in traditional models and introduces discontinuities in the option's path dependence. As a result, a wide range of numerical and semi-analytical methods has been developed to address the unique complexities these instruments present.

This chapter provides a comprehensive review of the existing literature on the pricing of discretely monitored barrier options. It begins by outlining foundational concepts in option pricing and the role of barrier options within the broader derivatives market. The discussion then delves into approaches for pricing these options, ranging from finite difference methods to more advanced stochastic volatility models, highlighting their respective advantages, limitations, and areas of applicability. The chapter concludes with a series of sections reviewing the analytical and numerical techniques that will be employed later in the thesis. These sections are intended to make the thesis more self-contained and to serve as a refresher for readers.

The objective of this chapter is to contextualise the current research by identifying gaps in the literature, particularly in terms of the accuracy, efficiency, and robustness of pricing methods for discretely monitored barrier options under realistic market conditions. This review serves as the foundation upon which the subsequent research contributions of this thesis are built.

2.1 Barrier options

Options are contingent claims that provide the holder with the right, but not the obligation, to buy or sell an underlying asset at a specified strike price on or before a given maturity date. These instruments are fundamental building blocks in financial markets and are often used for hedging, speculation, and arbitrage.

Barrier options extend the idea of plain vanilla options by introducing a *barrier* level, an additional price threshold that either activates or deactivates the option. For example, a knock-out barrier option expires worthless if the underlying touches a certain level during the life of the contract. In contrast, knock-in barrier options receive the payoff if the underlying touches a certain price. Barrier options are important in the Foreign Exchange markets; Dadachanji (2015) wrote, “In particular, barrier options are one of the most traded exotic derivatives in the forex market.”

The value of barrier options depends not only on the terminal asset price but also on the trajectory of the price over time, making them path-dependent. This feature makes barrier options cheaper than their vanilla counterparts and more sensitive to the path followed by the underlying asset, rendering their pricing considerably more complex.

2.1.1 Types and payoffs of barrier options

Barrier options are categorised based on their activation mechanism and direction of the barrier:

- **Up-and-out:** Deactivates if the underlying price rises above the barrier.
- **Down-and-out:** Deactivates if the price falls below the barrier.
- **Up-and-in:** Activates only if the price rises above the barrier.
- **Down-and-in:** Activates only if the price falls below the barrier.

A barrier option may also include a rebate feature, which provides a fixed payout if the option is knocked out. The payoff of an up-and-out call option with strike K, barrier $B > K$, and maturity T is:

$$\text{Payoff} = \begin{cases} (S_T - K)^+, & \text{if } S_t < B \quad \forall t \in \mathscr{T}, \\ 0, & \text{otherwise}, \end{cases}$$

where $\mathscr{T}$ denotes the set of monitoring times. The option pays off like a vanilla call only if the barrier is not breached during the life of the option.

2.1.2 Continuous and discrete monitoring

Barrier options can be monitored either *continuously* or *discretely*:

- **Continuously monitored** barrier options assume the underlying asset is observed at all times during the option's life.
- **Discretely monitored** barrier options check for barrier breaches only at specific observation times, typically aligned with trading days or fixings.

In practice, due to market conventions and data availability, barrier options are more commonly monitored at discrete intervals. Discrete monitoring introduces a nontrivial difference in valuation compared to continuous monitoring, even with frequent observations, because the asset may cross the barrier between observation dates without being detected.

This distinction is critical: analytical formulas that assume continuous monitoring often overestimate the probability of barrier breaches, leading to mispricing when applied to discretely monitored contracts. Consequently, accurate pricing of discretely monitored barrier options typically requires specialised numerical methods or analytical approximations.

2.1.3 No-arbitrage and risk-neutral valuation

The modern theory of option pricing is grounded in the *no-arbitrage principle*, which asserts that if two portfolios yield identical payoffs under all possible scenarios, they must have the same present value. This principle leads directly to the framework of *risk-neutral valuation*, where derivative prices are computed as discounted expectations under a risk-neutral probability measure $\mathbb{Q}$.

Let S_t denote the underlying asset price at time t, and let $\Pi(S_T)$ represent the option payoff at maturity T. Under the risk-neutral measure, the fair price V_0 of the option at time $t = 0$ is given by:

$$V_0 = e^{-rT}\mathbb{E}^{\mathbb{Q}}[\Pi(S_T)],$$

where r is the risk-free interest rate.

For barrier options, however, this expectation must incorporate the condition that the asset price has not crossed the barrier during the monitoring period. This dependence on the path, rather than just the terminal value, complicates the application of risk-neutral valuation and often necessitates numerical approaches.

2.2 Pricing methods for barrier options

Pricing formulas for single- and multi-asset European barrier options under Gaussian processes were developed by Rich (1994) and Wong and Kwok (2003), respectively. However, much of the early literature assumes continuous monitoring of the barrier—an assumption that is often impractical. As Kou (2007) noted, “due to regulatory and practical issues, most of the path-dependent options traded in markets are discrete path-dependent options... In practice, most, if not all, barrier options traded in markets are discretely monitored. In other words, they specify fixed time points for the monitoring of the barrier (typically daily closings).”

This discrete monitoring introduces several challenges. Theoretically, it precludes closed-form solutions under most realistic models. Numerically, it demands accurate approximation of asset paths between monitoring dates to detect possible barrier breaches. The discontinuous nature of barrier activation further complicates matters: small changes in the asset's path near the barrier can lead to abrupt shifts in option value, making both pricing and hedging more difficult.

To address these issues, a substantial body of research has developed pricing frameworks that incorporate more realistic models for both contract features and underlying dynamics. For example, Fusai et al. (2016) and Feng and Linetsky (2008) proposed methods for discretely monitored barrier options under general Lévy processes. More recently, stochastic volatility models have gained prominence for their ability to capture empirically observed phenomena such as volatility clustering and return distributions with fat tails and excess kurtosis. These models more accurately reflect market features like volatility smiles and skews, and better align with observed asset return dynamics than constant-volatility models.

In the following sections, we review analytical and numerical methods developed for pricing discretely monitored barrier options, with particular focus on those incorporating stochastic volatility.

2.2.1 Finite difference approach

Finite difference methods are a class of numerical techniques that approximate the solution of partial differential equations (PDEs). They have been widely adopted in option pricing since their introduction by Schwartz (1977). The key insight is that the evolution of an option's value can be described by a PDE, typically as a function of time and the price of the underlying asset. Finite difference methods discretise these continuous variables on a grid and iteratively solve the PDE backward in time, making them well-suited to handling early-exercise features and various boundary conditions.

This approach has been extensively applied to pricing discretely monitored barrier options, as explored by Boyle and Tian (1998) and Tagliani and Milev (2009). However, these methods face challenges in terms of computational efficiency. As noted by Tagliani and Milev (2009), although finite difference schemes can yield accurate results, they are often computationally intensive and highly sensitive to the choice of discretisation parameters. In the case of discretely monitored barrier options, the number of time steps must typically match or exceed the number of monitoring dates to achieve acceptable accuracy, which significantly increases computational cost.

Furthermore, incorporating more sophisticated asset price dynamics, such as stochastic volatility models, require the solution of higher-dimensional PDEs, which further exacerbates the computational burden. As a result, while finite difference methods remain a valuable tool in option pricing, their practical application to exotic options like discretely monitored barrier options often demands careful implementation and performance trade-offs.

2.2.2 Monte Carlo approach

Monte Carlo simulation has become one of the most powerful and versatile tools for option pricing, especially for exotic derivatives and options under complex underlying models. After Boyle (1977) first applied Monte Carlo simulation to option pricing, the method gained significant popularity due to its intuitive foundation and ease of implementation.

The core idea behind Monte Carlo simulation is the law of large numbers: by simulating a large number of independent realisations of the underlying asset price, one can approximate the expected payoff of the option. In practice, the method involves two main steps: first, simulate N underlying asset price paths by generating independent random variables; second, compute the option price as the average of the discounted payoff across all simulated paths. As N increases, the estimate converges to the true value.

However, the Monte Carlo method has a notable limitation: the convergence rate is only $1/\sqrt{N}$. This implies that to reduce the standard error by a factor of 10, one must increase the number of simulated paths by a factor of 100. Compared to the $1/N^4$ convergence rate achievable in one-dimensional numerical integration using Simpson's rule, Monte Carlo simulation converges rather slowly.

Monte Carlo methods are well-suited for pricing discretely monitored barrier options, because it naturally accommodates discrete monitoring by explicitly tracking the asset path at each observation time. Despite the slow convergence rate, the flexibility of the Monte Carlo approach often outweighs its drawbacks in these settings, especially when analytical or lattice-based methods become cumbersome or inapplicable.

2.2.3 Fourier transform based approach

The Fourier transform has emerged as a powerful technique in option pricing, particularly in models where the characteristic function of the log-asset price is known. Key foundational work includes Heston's semi-analytical solution for European options under stochastic volatility (Heston, 1993), and Carr and Madan's fast Fourier transform (FFT) framework for pricing a broad class of European-style options (Carr and Madan, 1999; Walker, 2017).

The pricing of discretely monitored barrier options using Fourier techniques is more complex due to the path-dependent nature of the payoff. Feng and Linetsky (2008) developed a backward induction scheme in the Fourier domain that computes the option price recursively across monitoring dates. While effective, the computational cost of this method grows linearly with the number of monitoring times, limiting its scalability.

To address this, Fusai et al. (2016) proposed a novel approach that applies the z-transform to the time dimension, effectively collapsing the recursive pricing structure into a single-stage computation in the frequency domain. This dramatically reduces computational complexity and makes the method more efficient for large numbers of monitoring dates. Phelan et al. (2019) further improved this method's numerical stability and convergence by incorporating spectral filtering techniques.

Beyond these, other authors have explored related Fourier-based pricing schemes in various settings. Lord et al. (2008) and Levendorskiĭ (2004) developed efficient Fourier inversion techniques for exotic options under Lévy and affine jump-diffusion models. These techniques, while not always specific to barrier options, lay the groundwork for extensions to discretely monitored contracts.

In Chapter 4, we extend the Fourier z-transform method to the Heston model and provide a comparative analysis with the recursive approach of Feng and Linetsky (2008).

2.2.3.1 COS method

The COS method, developed by Fang and Oosterlee (2009a), is an efficient Fourier-based technique originally proposed for pricing European-style options. It approximates the probability density function of the log-asset price using a truncated Fourier-cosine series expansion, leveraging the availability of characteristic functions in many models such as Black-Scholes, Heston, and Variance Gamma.

Initially introduced for vanilla options, the COS method has been extended to exotic options, including discretely monitored barrier options. Fang and Oosterlee (2009b) showed how to adapt the Fourier-cosine expansion to handle knock-in and knock-out conditions, achieving accurate and efficient pricing under the Black-Scholes model for both single and double barrier options.

The main advantages of the COS method are its rapid convergence and low computational cost. Compared to traditional numerical approaches like finite difference methods or Monte Carlo simulation, it achieves comparable or superior accuracy with fewer terms. Particularly for barrier options, where resolving discontinuities in the payoff is critical, the COS method's exponential convergence (under mild smoothness conditions) allows it to outperform other Fourier-based methods such as the Carr-Madan FFT approach (Carr and Madan, 1999).

Recent advances have further enhanced the COS framework. For example, Aimi et al. Aimi et al. (2023) coupled the COS method with the Boundary Element Method (BEM) to efficiently price barrier options under the Heston model, combining the COS method's efficiency with BEM's ability to handle free boundary problems.

Despite its strengths, the COS method requires careful selection of the truncation range $[a,b]$ and the number of expansion terms to balance efficiency and accuracy. In models with jumps or strong skewness, slower decay of the density function may necessitate more terms to maintain precision.

2.2.4 Deep learning approach

The application of neural networks in finance dates back to the late 1980s and early 1990s (Hutchinson et al., 1996; Malliaris and Salchenberger, 1993; White, 1988). However, the limited computational power and data availability at the time restricted their practical success. With the recent surge in computational capabilities and the availability of large datasets, deep learning has achieved remarkable breakthroughs in areas such as image recognition, natural language processing, and medicine — and is now increasingly influencing financial modelling.

In option pricing, deep learning methods have gained traction in two primary directions. The first involves function approximation, where neural networks are trained to learn a non-parametric mapping from inputs—such as the option's features and the state of the underlying asset—to option prices. This approach, exemplified by Liu et al. (2019), enables rapid pricing of options after a model is trained, offering a practical alternative to traditional pricing formulas.

The second direction enhances model-based methods, particularly PDE solvers, by leveraging neural networks to speed up computations. This has attracted growing interest in both finance and engineering domains, where PDEs are commonly encountered. For instance, Buehler et al. (2019) introduced the Deep Hedging framework, which learns hedging strategies directly from market data using deep reinforcement learning, bypassing the need for explicit pricing models.

Several studies have demonstrated the superior performance of neural networks compared to classical models. Malliaris and Salchenberger (1993) and Culkin and Das (2017) compared neural network-based models to the Black-Scholes model and found that deep neural networks can yield more accurate pricing, particularly in capturing nonlinearities and market anomalies. Mitra (2012) showed that neural networks trained on data from the 2008 financial crisis could effectively learn option pricing patterns during extreme market conditions.

Horvath et al. (2019) further extended deep learning approaches to complex models like the rough Bergomi model, demonstrating that neural networks can facilitate fast and accurate pricing in cases where traditional numerical methods are computationally expensive.

2.3 Foundations of stochastic processes

Stochastic processes are at the heart of modern mathematical finance. They are used to model the random behaviour of financial assets over time. This section provides a concise review of stochastic processes, intended as a refresher for readers of this thesis. For a more comprehensive treatment, refer to Shreve (2004).

A *stochastic process* is a collection of random variables $\{X(t) : t \in T\}$ defined on a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$, where Ω is the sample space, $\mathfrak{F}$ is the σ-algebra of events, and $\mathbb{P}$ is a probability measure. The index set T typically represents time. For each outcome $\omega \in \Omega$, the mapping $t \mapsto X(t, \omega)$ describes the evolution of the process.

An evocative description is offered in Joshi and Joshi (2003): “We can think of the goddess of probability living in eternity outside the ephemeral world of an option trader. She draws an entire stock path from a jar containing all the possible stock prices. The god of time stops us from looking into the future and slowly reveals the stock price path to us, second by second.” This highlights the interplay between randomness and temporal revelation inherent in stochastic processes.

The concept of *filtration* arises naturally to represent the information available up to a given time t. This is crucial for defining adapted processes and for the mathematical formulation of stochastic integrals.

2.3.1 Characteristic functions

The *characteristic function* of a probability distribution μ is the Fourier transform of its measure:

$$\phi_\mu(\xi) = \int e^{i\xi x}\mu(dx). \tag{2.1}$$

For a random variable X with distribution μ_X, the characteristic function is:

$$\phi_X(\xi) = \mathbb{E}[e^{i\xi X}]. \tag{2.2}$$

Characteristic functions play a vital role in deriving transition densities and option prices in models where closed-form probability density functions are not available.

2.3.2 Itô processes and Itô’s lemma

Most continuous-time models in finance assume that asset prices follow an Itô process:

$$X(t) = X(0) + \int_0^t \Delta(u)\,dW(u) + \int_0^t \Theta(u)\,du, \tag{2.3}$$

where $\Delta(u) : u \geq 0$ and $\Theta(u) : u \geq 0$ are adapted processes, and $W(t)$ denotes a standard Brownian motion.

A powerful result in stochastic calculus is *Itô's lemma*, the stochastic analogue of the chain rule. For a twice-differentiable function $f(t,X(t))$, we have:

$$df(t,X(t)) = \frac{\partial f}{\partial t}dt + \frac{\partial f}{\partial x}\Delta(t)\,dW(t) + \frac{\partial f}{\partial x}\Theta(t)\,dt + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}\Delta^2(t)\,dt. \tag{2.4}$$

2.3.3 Stochastic models

2.3.3.1 Geometric Brownian motion

A constant volatility approach assumes that the derivative's underlying asset price follows a geometric Brownian motion:

$$dS(t) = \mu S(t)\,dt + \sigma S(t)\,dW(t), \quad S(0) = S_0, \tag{2.5}$$

where μ is the constant drift (i.e. expected return) of the security price and σ is the constant volatility. Its closed-form solution is:

$$S(t) = S_0 e^{(\mu - \frac{1}{2}\sigma^2)t + \sigma W(t)}. \tag{2.6}$$

This basic model underpins classical frameworks such as Black and Scholes (1973); Cox et al. (1979); Merton (1973).

2.3.3.2 Stochastic volatility model

In a stochastic volatility (SV) model, the volatility is itself a random process. A general form is:

$$dS(t) = \mu S(t)\,dt + \sqrt{v(t)}S(t)\,dW_1(t), \tag{2.7}$$

$$dv(t) = \alpha(v,t)\,dt + \beta(v(t))\,dW_2(t), \tag{2.8}$$

$$dW_1(t)\,dW_2(t) = \rho\,dt, \tag{2.9}$$

where $W_1(t)$ and $W_2(t)$ are Wiener process with a constant correlation ρ with $|\rho| < 1$. Usually ρ is negative because of the leverage effect, i.e. the observed negative correlation between volatility and asset price. The variance process $v(t)$ is often modelled using Ornstein-Uhlenbeck or Cox-Ingersoll-Ross dynamics. Popular SV models include the Heston model (Heston, 1993), the constant elasticity of variance model (Cox, 1996), and the SABR model (Hagan et al., 2002).

2.3.3.3 The Heston model

In the Heston model, the variance follows a mean-reverting Cox-Ingersoll-Ross (CIR) process:

$$dv(t) = \kappa(\theta - v(t))\,dt + \sigma\sqrt{v(t)}\,dW_2(t), \tag{2.10}$$

where κ, θ, and σ are nonnegative constants. Together with the initial variance $v_0 = v(0)$ and the correlation ρ, the Heston model has five parameters, which are typically calibrated by fitting to the volatility surface (Cui et al., 2017).

Let $\tau = T - t$ denote the time to maturity, given a fixed contract expiry date T, and let $x(t)$ be the log-price defined as $x(t) = \log(S(t)/S_0)$, where $S(0) = S_0$ is the initial price of the underlying asset. Applying Itô's lemma, Eq. (2.7) can be rewritten as:

$$\begin{aligned} dx(t) &= \left(\mu - \frac{v(t)}{2}\right)dt + \sqrt{v(t)}\,dW_1(t) \\ &= \left(\mu - \frac{v(t)}{2}\right)dt + \rho\sqrt{v(t)}\,dW_2(t) + \sqrt{1-\rho^2}\sqrt{v(t)}\,dW_1'(t), \end{aligned} \tag{2.11}$$

where W_1' and W_2 are *independent* Wiener processes. Under the risk-neutral measure, the drift term μ is replaced by the risk-free rate r.

Due to the mean-reverting nature of the variance process, the log-price process does not have stationary increments. As a result, the two-dimensional transition density cannot be expressed solely in terms of increments in the log asset price and variance. However, the transition density can be expressed in terms of the increments in the log-price and the terminal variance, conditional on the current variance:

$$g(X(T), V(T) \mid X(t), V(t)) = g(X(T) - X(t), V(T) \mid V(t)). \tag{2.12}$$

The joint conditional probability density (also called the transition density) can be obtained by performing a two-dimensional inverse Fourier transform of the joint conditional characteristic function (JCCF). According to Griebsch (2013), the JCCF of $(X(T),V(T))$ given $X(t)=x(t)$ and $V(t)=v(t)$ is:

$$\mathbb{E}\left[e^{i\xi X(T)+i\omega V(T)} \mid X(t)=x(t),V(t)=v(t)\right]=e^{A(\xi,\omega,\tau)+B(\xi,\omega,\tau)x(t)+C(\xi,\omega,\tau)v(t)}, \tag{2.13}$$

where $\tau=T-t$ and A, B, and C are functions of ξ, ω, and τ. At $\tau=0$, the initial condition is:

$$\begin{aligned}\mathbb{E}\left[e^{i\xi X(T)+i\omega V(T)} \mid X(T)=x(t),V(T)=v(t)\right]&=e^{i\xi x(t)+i\omega v(t)}\\ &=e^{A(\xi,\omega,0)+B(\xi,\omega,0)x(t)+C(\xi,\omega,0)v(t)},\end{aligned} \tag{2.14}$$

which implies that $A(\xi,\omega,0)=0$, $B(\xi,\omega,0)=i\xi$, and $C(\xi,\omega,0)=i\omega$. Since $x(t)$ does not appear in the SDE for $v(t)$ (i.e. in neither the drift nor diffusion terms of Eq. (2.10)), it follows that $B(\xi,\omega,\tau)=i\xi$.

Performing a 2-dimensional inverse Fourier transform of the JCCF yields:

$$\begin{aligned}&g(x(T),v(T)\mid x(t),v(t))\\ &=\frac{1}{2\pi}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}e^{-i(\xi x(T)+\omega v(T))}\,e^{A(\xi,\omega,\tau)+i\xi x(t)+C(\xi,\omega,\tau)v(t)}\,d\xi\,d\omega\\ &=\frac{1}{2\pi}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}e^{-i\xi(x(T)-x(t))-i\omega v(T)+A(\xi,\omega,\tau)+C(\xi,\omega,\tau)v(t)}\,d\xi\,d\omega.\end{aligned} \tag{2.15}$$

From Eq. (2.15), it is evident that the transition density depends only on $(x(T)-x(t))$, $v(T)$, and $v(t)$, and can therefore be rewritten as:

$$\begin{aligned}g(x(T),v(T)\mid x(t),v(t))&=g(x(T)-x(t),v(T)\mid v(t))\\ &=f(x(T)-x(t)\mid v(T),v(t))\,p(v(T)\mid v(t)),\end{aligned} \tag{2.16}$$

where $p(v(T)\mid v(t))$ is the transition density of the variance. As the variance follows a Feller square-root process, this density is given by:

$$p(v(T)\mid v(t))=c\,e^{-u-\nu}\left(\frac{\nu}{u}\right)^{j/2}I_j(2\sqrt{u\nu}), \tag{2.17}$$

where

$$c = \frac{2\kappa}{(1-e^{-\kappa\tau})\sigma^2}, \quad j = \frac{2\kappa\theta}{\sigma^2} - 1, \quad u = c\,v(t)\,e^{-\kappa\tau}, \quad \nu = c\,v(T),$$

and $I_j(2\sqrt{u\nu})$ is the modified Bessel function of the first kind of order j.

The expression for the conditional characteristic function of $(x(T)-x(t))$ can be found in Appendix A.

2.4 Analytic background

The Fourier transform is a central mathematical tool used extensively throughout this thesis. This section begins by introducing its formal mathematical definition, along with other transforms that are relevant to the option pricing techniques discussed later in the thesis. Following this, we explore the Wiener-Hopf factorisation technique, which plays a crucial role in the pricing of discretely monitored barrier options, particularly under models with jumps or other complex dynamics. To conclude, we present a simple example illustrating how the Fourier transform can be applied to price European options, serving as a practical refresher and motivation for its use in more advanced settings.

2.4.1 Fourier transform

The Fourier transform decomposes a time-dependent function into its frequency components. For example, it can break down a piece of music into its constituent frequencies and their corresponding amplitudes (or volumes). Imagine a musical signal composed of several pure tones. To identify these tones, we apply the Fourier transform to the signal. Plotting the result—amplitude versus frequency—reveals peaks at certain frequencies. These peaks indicate the pure frequencies present in the signal, and their amplitudes represent the relative contribution, or weight, of each frequency.

Formally, the Fourier transform $\widehat{f}(\xi)$ of a square-integrable function $f(x)$ is a linear transform $\mathscr{F} : \mathbb{C} \mapsto \mathbb{C}$. There are several conventions for the Fourier transform.

This thesis adopts the usual convention in mathematical finance and defines

$$\mathscr{F}_{x\to\xi}\left[f(x)\right] = \widehat{f}(\xi) := \int_{-\infty}^{+\infty} f(x)e^{i\xi x}dx. \tag{2.18}$$

The corresponding inverse Fourier transform is

$$\mathscr{F}^{-1}_{\xi\to x}\left[\widehat{f}(\xi)\right] = f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \widehat{f}(\xi)e^{-ix\xi}d\xi. \tag{2.19}$$

Some important properties of the Fourier transform that we will need are:

1. **Differentiation**: $\mathscr{F}_{x\to\xi}[g'(x)](\xi) = -i\xi\widehat{g}(\xi)$, where $g'(x)$ is the derivative of $g(x)$.

2. **Shift**: $\mathscr{F}_{x\to\xi}[e^{iax}g(x)](\xi) = \widehat{g}(\xi+a)$, where a is a constant. This can be seen easily by writing the definition of the Fourier transform of $e^{\alpha x}g(x)$:

$$\begin{aligned}\mathscr{F}_{x\to\xi}(e^{iax}g(x))(\xi) &= \int_{-\infty}^{+\infty} e^{iax}g(x)e^{i\xi x}dx \\ &= \int_{-\infty}^{+\infty} e^{i(\xi+a)x}g(x)dx \\ &= \widehat{g}(\xi+a). \end{aligned} \tag{2.20}$$

3. **Damping**: Closely related to the shift property is the effect of applying an exponential damping factor $e^{\alpha x}$, i.e. $\mathscr{F}_{x\to\xi}[e^{\alpha x}g(x)](\xi) = \widehat{g}(\xi - i\alpha)$, where α is the damping parameter.

4. **Convolution**: The convolution of two functions $f(x)$ and $g(x)$ is written

$$(f*g)(x) = \int_{-\infty}^{+\infty} f(y)g(x-y)dy. \tag{2.21}$$

By the convolution theorem

$$\mathscr{F}_{x\to\xi}\left[(f*g)(x)\right] = \widehat{f}(\xi)\widehat{g}(\xi) \tag{2.22}$$

and also

$$\mathscr{F}^{-1}_{\xi\to x}\big[(\widehat{f} * \widehat{g})(\xi)\big] = f(x)g(x). \tag{2.23}$$

5. **Plancherel relation**: The Fourier transform is norm-preserving i.e.

$$\begin{aligned}\int_{-\infty}^{+\infty} f(x)g^*(x)dx &= \frac{1}{2\pi}\int_{-\infty}^{+\infty} \widehat{f}^*(\xi)\widehat{g}(\xi)d\xi \\ &= \mathscr{F}^{-1}_{\xi\to x}\big[\widehat{f}^*(\xi)\widehat{g}(\xi)\big].\end{aligned} \tag{2.24}$$

   where $g^*(x)$ is the complex conjugate of $g(x)$.

2.4.2 Fractional Fourier transform

The Fractional Fourier transform (FRFT) is a generalised version of Fourier transform.

Definition 2.4.1 *Fractional Fourier transform For any real α, the α-angle fractional Fourier transform of a function f is denoted by $\mathscr{F}_\alpha(\xi)$ and defined by*

$$\mathscr{F}_\alpha[f](\xi) = \sqrt{1 - i\cot(\alpha)}e^{i\pi\cot(\alpha)\xi^2}\int_{-\infty}^{\infty} e^{-i2\pi\left(\csc(\alpha)\xi x - \frac{\cot(\alpha)}{2}x^2\right)} f(x)\mathrm{d}x. \tag{2.25}$$

For $\alpha = \pi/2$, it becomes precisely the definition of the continuous Fourier transform, and for $\alpha = -\pi/2$, it is the definition of the inverse continuous Fourier transform.

We can also write the Eq. (2.25) as the form of integral transform with the fractional α-angle kernel $K_\alpha(\xi,x)$,

$$\mathscr{F}_\alpha f(\xi) = \int K_\alpha(\xi,x)f(x)\mathrm{d}x, \tag{2.26}$$

where the α-angle kernel is

$$K_\alpha(\xi,x) = \begin{cases} \frac{c(\alpha)}{\sqrt{2\pi}}\exp\left\{ia(\alpha)\left(x^2+\xi^2-2b(\alpha)x\xi\right)\right\} & \text{if } \alpha \text{ is not a multiple of } \pi \\ \delta(\xi - x) & \text{if } \alpha \text{ is a multiple of } 2\pi \\ \delta(\xi + x) & \text{if } \alpha + \pi \text{ is a multiple of } 2\pi \end{cases} \tag{2.27}$$

with

$$a(\alpha) = \frac{\cot\alpha}{2}, \quad b(\alpha) = \sec\alpha, \quad c(\alpha) = \sqrt{1 - i\cot\alpha}. \tag{2.28}$$

For simplicity, we denote $a(\alpha)$, $b(\alpha)$ and $c(\alpha)$ as a, b and c respectively.

2.4.3 Z-transform

The z-transform of a sequence of time-discrete functions $f_n(x) = f(x,n) = f(x,t_n)$ indexed with $n \in \mathbb{N}_{\not 0}$ (or a series of discretely sampled points from a time-continuous function) is

$$\widetilde{f}(x,q) = \sum_{n=0}^{\infty} q^n f_n(x), \tag{2.29}$$

with $q \in \mathbb{C}$. An example of the z-transform is the probability generating function, where the component $f_n(x)$ is the probability that a discrete random variable takes the value x.

The inverse z-transform is obtained with a Bromwich integral: choosing a circular integration contour C with radius ρ must be within radius convergence,

$$f_n(x) = \mathscr{Z}^{-1}_{q\to n}\left[\widetilde{f}(x,q)\right] = \frac{1}{2\pi\rho^n}\oint_C \widetilde{f}(x,\rho e^{i\theta})e^{-in\theta}d\theta. \tag{2.30}$$

Instead of solving analytically, this can be approximated numerically via Abate and Whitt (1992) and Fusai et al. (2016) successfully accelerated the Abate's method via Euler summation.

2.4.4 Hilbert transform

The Hilbert transform of a function in the Fourier domain $\widehat{f}(\xi)$ is

$$\begin{aligned}\mathscr{H}\left[f(\xi)\right] &= \text{p.v.}\frac{1}{\pi\xi} * \widehat{f}(\xi)\\ &= \text{p.v.}\frac{1}{\pi}\int_{-\infty}^{+\infty}\frac{\widehat{f}(\xi')}{\xi-\xi'}d\xi'\\ &= \lim_{\varepsilon\to 0}\left(\frac{1}{\pi}\int_{-1/\varepsilon}^{\xi-\varepsilon}\frac{\widehat{f}(\xi')}{\xi-\xi'}d\xi' + \frac{1}{\pi}\int_{\xi+\varepsilon}^{1/\varepsilon}\frac{\widehat{f}(\xi')}{\xi-\xi'}d\xi'\right),\end{aligned} \tag{2.31}$$

where $p.v.$ denotes the Cauchy principal value. The Fourier transform of the signum function $\text{sgn}(x) = \mathbf{1}_{x>0}(x) - \mathbf{1}_{x<0}(x)$ is $1/i\pi\xi$ and therefore, using the convolution theorem,

$$i\mathscr{H}\widehat{f}(\xi) = \mathscr{F}_{x\to\xi}[\text{sgn}(x)f(x)]. \tag{2.32}$$

2.4.5 Wiener-Hopf technique

A Wiener-Hopf equation of the second kind is an integral equation of the form

$$f(x) - \lambda \int_a^b k(x-x')f(x')dx' = g(x), \quad x \in (a,b), \tag{2.33}$$

where $f(x)$ is an unknown function, $k(x)$ is a given kernel, $g(x)$ is a known forcing function and λ is a constant. When $b = \infty$ or $a = -\infty$, a direct solution can be obtained using the Wiener-Hopf technique (Wiener and Hopf, 1931) which exploits that Eq. (2.33) has a convolutional form. In Section 2.4.5.1 we introduce two important techniques for the Wiener-Hopf method, decomposition and factorisation, before providing a brief overview of the Wiener-Hopf method in Section 2.4.5.2.

2.4.5.1 Decomposition and factorisation

Decomposition We decompose a function $f(x)$ as

$$f(x) = f_+(x) + f_-(x), \tag{2.34}$$

with

$$f_+(x) = \mathbf{1}_{\mathbb{R}_+}(x)f(x), \tag{2.35}$$

$$f_-(x) = \mathbf{1}_{\mathbb{R}_-}(x)f(x), \tag{2.36}$$

where $\mathbf{1}_A(x)$ is the indicator function of the set A. The Fourier transforms of $f_+(x)$ and $f_-(x)$ are

$$\widehat{f_+}(\xi) = \mathscr{F}_{x\to\xi}[f_+(x)](\xi) = \int_0^{+\infty} f(x)e^{i\xi x}dx, \tag{2.37}$$

$$\widehat{f_-}(\xi) = \mathscr{F}_{x\to\xi}[f_-(x)](\xi) = \int_{-\infty}^0 f(x)e^{i\xi x}dx. \tag{2.38}$$

There is a constant c such that $\widehat{f_\pm}(\xi)$ is analytic in the $\Im(\xi) \lesseqgtr c_\mp$ with $c_- < 0 < c_+$. The analytic regions of $\widehat{f_+}(\xi)$ and $\widehat{f_-}(\xi)$ overlap in a strip. Within the strip we have

$$\widehat{f_+}(\xi) + \widehat{f_-}(\xi) = \widehat{f}(\xi). \tag{2.39}$$

Please note, the strip includes the real line. There is an important relationship between the two half-range Fourier transforms and Hilbert transform:

$$\widehat{f_+}(\xi) - \widehat{f_-}(\xi) = i\mathscr{H}\widehat{f}(\xi). \tag{2.40}$$

With Eqs. (2.39) and (2.40) and the shift property we can obtain the generalised **Plemelj-Sokhotsky relations** with an arbitrary barrier l:

$$\widehat{f_+}(\xi) = \frac{1}{2}\{\widehat{f}(\xi) + ie^{il\xi}\mathscr{H}\left[e^{-il\xi}\widehat{f}(\xi)\right]\}, \tag{2.41}$$

$$\widehat{f_-}(\xi) = \frac{1}{2}\{\widehat{f}(\xi) - ie^{il\xi}\mathscr{H}\left[e^{-il\xi}\widehat{f}(\xi)\right]\}. \tag{2.42}$$

Factorisation The Wiener-Hopf technique also requires functions to be factorised, i.e. for $\widehat{f}(\xi) = \mathscr{F}_{x\to\xi}[f(x)]$ obtain $\widehat{f}_\oplus(\xi)$ and $\widehat{f}_\ominus(\xi)$ such that

$$\widehat{f}_\oplus(\xi)\widehat{f}_\ominus(\xi) = \widehat{f}(\xi). \tag{2.43}$$

We use the $_{\oplus\ominus}$ notation to distinguish factorised and decomposed functions. In order to find the factorisation of Eq. (2.43), we can take the logarithm on both sides

$$\log\widehat{f}(\xi) = \log\widehat{f}_\oplus(\xi) + \log\widehat{f}_\ominus(\xi) \tag{2.44}$$

Then we use the Plemelj-Sokhotsky relations obtaining

$$\log\widehat{f}_\oplus(\xi) = \frac{1}{2}(\log\widehat{f}(\xi) + i\mathscr{H}\log\widehat{f}(\xi)), \tag{2.45}$$

$$\log\widehat{f}_\ominus(\xi) = \frac{1}{2}(\log\widehat{f}(\xi) - i\mathscr{H}\log\widehat{f}(\xi)). \tag{2.46}$$

Last, we take the exponential of the above results.

2.4.5.2 Wiener-Hopf procedure

The similarity of Eq. (2.33) to the convolution in Eq. (2.21) is immediately apparent, but the limits on the integral of the former are infinite, so we cannot immediately apply the convolution theorem as described above. As this thesis is concerned with single-barrier options, we concentrate on the case with an upper integral limit $b = \infty$ and a lower integral limit $a > -\infty$ which can be shifted to $a = 0$ without loss of generality. Thus Eq. (2.33) becomes

$$f(x) - \lambda \int_0^\infty k(x-x')f(x')dx' = g(x), \quad x \in (0,\infty). \tag{2.47}$$

In order to apply the convolution theorem we must extend Eq. (2.47) over $\mathbb{R}$. First we can redefine $f(x)$ as $f_+(x) = f(x)\mathbf{1}_{\mathbb{R}^+}(x)$ to extend the range of the integral to $\pm\infty$. We also redefine $g(x)$ as $g_+(x) = g(x)\mathbf{1}_{\mathbb{R}^+}(x)$ and define a auxiliary function $f_-(x) = \int_{-\infty}^{+\infty} k(x-x')f_+(x')\mathbf{1}_{\mathbb{R}^-}(x)dx'$. We thus obtain the equivalent to Eq. (2.47) over the entire range of $\mathbb{R}$,

$$f_+(x) - \lambda \int_{-\infty}^\infty k(x-x')f_+(x')dx' + f_-(x) = g_+(x), \quad x \in \mathbb{R}. \tag{2.48}$$

We apply the Fourier transform to Eq. (2.48) and replace the convolution with a multiplication as described in Section 2.4.1, obtaining

$$\widehat{l}(\xi)\widehat{f_+}(\xi) + \widehat{f_-}(\xi) = \widehat{g_+}(\xi), \tag{2.49}$$

where $\widehat{l}(\xi) = 1 - \lambda\widehat{k}(\xi)$. Subject to various analyticity conditions (Noble and Weiss, 1959), we can factorise $\widehat{l}(\xi) = \widehat{l_\oplus}(\xi)\widehat{l_\ominus}(\xi)$ and divide both sides of Eq. (2.49) by $\widehat{l_\ominus}(\xi)$, yielding

$$\widehat{l_\oplus}(\xi)\widehat{f_+}(\xi) + \frac{\widehat{f_-}(\xi)}{\widehat{l_\ominus}(\xi)} = \frac{\widehat{g_+}(\xi)}{\widehat{l_\ominus}(\xi)}. \tag{2.50}$$

We notice that the left hand side of Eq. (2.50) is split into “plus" and “minus" parts. Therefore, if we decompose the right hand side we can discard the “minus" parts and equate the “plus" parts,

$$\widehat{l_\oplus}(\xi)\widehat{f_+}(\xi) = \left[\frac{\widehat{g_+}(\xi)}{\widehat{l_\ominus}(\xi)}\right]_+. \tag{2.51}$$

Rearranging, we obtain the Fourier transform of the solution of the Eq. (2.33).

$$\widehat{f_+}(\xi) = \frac{1}{\widehat{l_\oplus}(\xi)}\left[\frac{\widehat{g_+}(\xi)}{\widehat{l_\ominus}(\xi)}\right]_+. \tag{2.52}$$

2.4.6 Example: Fourier transform for European option pricing

Having reviewed Fourier techniques, we now illustrate their application in the simpler context of European option pricing. The initial value of the call price $C(k)$ can be found by

$$C(k) = \int_k^\infty e^{-rT}(e^x - e^k) f(x) dx, \tag{2.53}$$

where k denotes the logarithm of the strike price K and similarly x is the logarithm price with risk neutral density $f(x)$. Note $C(k)$ is not square integrable and in order to obtain a square integrable function for all real k we modified $C(k)$ to

$$c(k) = e^{\alpha k} C(k), \tag{2.54}$$

where α is positive.

Now the Fourier transform of $c(k)$ is

$$\begin{aligned}\psi(\xi) &= \int_{-\infty}^\infty e^{i\xi k} c(k) dk \\ &= \int_{-\infty}^\infty e^{i\xi k} \int_k^\infty e^{\alpha k} e^{-rT}(e^x - e^k) f(x) dx dk.\end{aligned} \tag{2.55}$$

Please note by computing the double integral, $\psi(\xi)$ can be expressed by the characteristic function of the log price density as

$$\psi(\xi) = \frac{e^{-rT}\phi(\xi - (\alpha+1)i)}{\alpha^2 + \alpha - \xi^2 + i(2\alpha+1)\xi}, \tag{2.56}$$

where $\phi(\cdot)$ is the characteristic function of $f(x)$:

$$\phi(u) = \int_{-\infty}^\infty e^{iux} f(x) dx. \tag{2.57}$$

Then the call option price $C(k)$ can be found by taking the inverse transform of $\psi(\xi)$ and times the factor $e^{-\alpha k}$.

2.5 Numerical methods

In this section, we review several numerical methods that are relevant to the development and implementation of the models and algorithms discussed later in this thesis. Although each of these techniques is well-established, we provide a concise refresher to ensure that the thesis remains self-contained and accessible to readers with diverse technical backgrounds.

We begin by summarising the discrete Fourier transform (DFT) and the discrete fractional Fourier transform (DFrFT), both of which play crucial roles in spectral methods and in the efficient evaluation of characteristic functions. We then discuss root-finding methods, which are frequently employed to solve nonlinear equations arising in calibration and pricing problems.

2.5.1 Discrete Fourier transform

The discrete Fourier transform (DFT) is a fundamental tool for analysing and manipulating signals in the frequency domain. In quantitative finance, it is particularly useful for efficiently computing option prices when characteristic functions are available. The DFT enables the evaluation of complex integrals and convolution operations by transforming problems from the time or spatial domain into the frequency domain, where they often become simpler to solve.

By Nyquist relation

$$\Delta\xi = \frac{2\pi}{N_x dx}, \tag{2.58}$$

we define the DFT as

$$\begin{aligned} \widehat{f}(\xi_k) &= \Delta x \sum_{n=-N/2}^{N/2-1} f(x_n) e^{ix_n\xi_k} \\ &= \Delta x \sum_{n=-N/2}^{N/2-1} f(x_n) e^{in\Delta x k\Delta\xi} \\ &= \Delta x \sum_{n=-N/2}^{N/2-1} f(x_n) e^{2\pi ikn/N}. \end{aligned} \tag{2.59}$$

Similarly, we define IDFT as

$$f(x_n) = \frac{\Delta\xi}{2\pi} \sum_{k=-N/2}^{N/2-1} \widehat{f}(\xi_k) e^{-2\pi ikn/N}, \tag{2.60}$$

where $\Delta\xi$ and Δx are the grid spacing of the sequences $\{x_n\}$ and $\{\xi_n\}$, N is the total number of the grid points, and both k and n are in the set $\left[\frac{-N}{2}, \frac{N}{2}-1\right]$ if N is even and $\left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$ if N is odd.

MATLAB provides the built-in functions `fft.m` and `ifft.m` to compute the DFT and IDFT, respectively. Unlike Eq. (2.59) and Eq. (2.60), MATLAB defines the DFT and IDFT as

$$\begin{aligned}\widehat{f_m}(\nu_k) &= \sum_{n=0}^{N-1} f_m(x_n) e^{-i2\pi kn/N} \\ &= \sum_{n=0}^{N-1} f_m(x_n) F(k,n), \end{aligned} \tag{2.61}$$

and

$$\begin{aligned} f_m(x_n) &= \frac{1}{N} \sum_{k=0}^{N-1} \widehat{f_m}(\nu_k) e^{i2\pi kn/N} \\ &= \sum_{k=0}^{N-1} \widehat{f_m}(\nu_k) F(k,n)^{-1}, \end{aligned} \tag{2.62}$$

where

$$F(k,n) = e^{-i2\pi kn/N}. \tag{2.63}$$

Thus, to correctly use `fft.m` and `ifft.m`, a few adjustments must be made:

First, the signs in the exponential functions in Eq. (2.61) and Eq. (2.62) are reversed compared to our definitions. As a result, we use `ifft.m` to compute the forward Fourier transform and `fft.m` to compute the inverse Fourier transform.

Second, both Eq. (2.61) and Eq. (2.62) contain a factor of 2π in the exponentials, whereas our definitions do not. This is because MATLAB uses the scaled variable $\nu := \xi/2\pi$ in the transformed space instead of ξ.

Third, MATLAB assumes that the x-grid spacing is 1, and the ν-grid spacing is $1/N$. To account for different spacings:

- For the forward Fourier transform, Eq. (2.62) must be multiplied by $N\Delta x$.

- For the inverse Fourier transform, Eq. (2.61) must be multiplied by $\Delta\xi/2\pi$, or equivalently $\frac{1}{N\Delta x}$, according to the Nyquist relation Eq. (2.58), since

$$\frac{\Delta\xi}{2\pi} = \frac{1}{N\Delta x}.$$

Additionally, MATLAB uses sequence indexing that starts from 0, whereas our sequence is centred around zero. Since both sequences are periodic with period N, we can reconcile the difference by applying `fftshift.m` before and after performing the Fourier transforms.

To summarise, when using MATLAB to compute forward and inverse Fourier transforms, the following adjustments are needed:

- The forward Fourier transform (i.e., $\widehat{f}(\xi)$) can be computed as

$$\widehat{f}(\xi) = \text{ifftshift}\left(\text{ifft}\left(\text{fftshift}(f(x))\right)N\Delta x\right),$$

- The inverse Fourier transform (i.e., $f(x)$) can be computed as

$$f(x) = \text{ifftshift}\left(\frac{\text{fft}\left(\text{fftshift}(\widehat{f}(\xi))\right)}{N\Delta x}\right).$$

2.5.2 Discrete fractional Fourier transform

In this section, we show how to use FRFT to calculate the inverse Fourier transform instead of using IFFT.

The integration in the IFFT (2.19) can be approximated using trapezoidal, Simpson's, Gaussian quadrature or other numerical integration rules. Here by choosing trapezoidal rule, we get

$$\int_{-\infty}^{+\infty} \widehat{f}(\xi)e^{-i\xi x}d\xi \approx \sum_{k=-N/2}^{N/2-1} \widehat{f}(\xi_k)e^{-\xi_k x_n}\Delta\xi w_k, \tag{2.64}$$

where $\Delta\xi$ is the grid spacing for the sequence ξ_k and w_k are trapezoidal weights. $\Delta\xi$ need to be sufficient small for the accuracy of the integration approximation. Whereas the value of $N\Delta\xi$, the truncation limit, need to be large enough to ensure the characteristic function is 0 for any $\xi > (N/2-1)\Delta\xi$ and $\xi < -N/\Delta\xi$.

The inverse discrete fractional Fourier transform of a vector $f(\xi_n)$ with $n=[0,N-1]$ is computed as

$$\widehat{f_a}(x_k) = \sum_{n=0}^{N-1} f(\xi_n) F^a(k,n), \tag{2.65}$$

where $a = \frac{2\alpha}{\pi}$ and $F(k,n)$ is defined with Eq. (2.63). As we discussed earlier, when $a=1$ (i.e. $\alpha = \pi/2$), we have the FFT formula.

We observe that the Eq. (2.64) can be rewritten in the form of Eq. (2.65) by taking $\Delta x \Delta\xi = \frac{2\pi a}{N}$. In addition to N, Δx and $\Delta\xi$ are determined by the range of x and ξ respectively and independently. Then the parameter a of the FRFT can be automatic calculated by

$$a = \frac{\Delta x \Delta\xi N}{2\pi}. \tag{2.66}$$

As discussed in Bailey and Swarztrauber (1991) and Chourdakis (2005), the FRFT can be computed by taking three FFT. 2N-point FFT is needed to calculate the FRFT of the N-point vector $f(\xi)$ with $\xi = \left(\xi_{-N/2}, ..., \xi_{N/2-1}\right)$. First, we define the following vectors

$$\mathbf{w} = \left(\frac{1}{2},\quad (1)_{n=-N/2+1}^{N/2-2},\quad \frac{1}{2}\right), \tag{2.67}$$

$$\mathbf{y_1} = \left((0)_{n=-N}^{-N/2-1},\quad (f(\xi_n)e^{-i\pi n^2\alpha}\Delta\xi w_n)_{n=-N/2}^{N/2-1},\quad (0)_{n=N/2}^{N-1}\right), \tag{2.68}$$

$$\mathbf{y_2} = \left((e^{i\pi n^2\alpha})_{n=-N}^{-N/2-1},\quad (e^{-i\pi n^2\alpha})_{n=-N/2}^{N/2-1},\quad (e^{i\pi n^2\alpha})_{n=N/2}^{N-1}\right), \tag{2.69}$$

The vector $\mathbf{w}$ contains the trapezoidal rule weights. Both ξ and $\mathbf{x}$ are defined symmetrically around zero, and the vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ are zero-padded to length $2N$ accordingly.

The inverse fractional Fourier transform is then computed as

$$\widehat{f_\alpha}(x) = \left(\mathscr{F}^{-1}(\widehat{\mathbf{y}}_1 \cdot \widehat{\mathbf{y}}_2)\right) \cdot \mathbf{y}_2, \tag{2.70}$$

where $\widehat{\mathbf{y}}_1 = \mathscr{F}(\mathbf{y}_1)$, $\widehat{\mathbf{y}}_2 = \mathscr{F}(\mathbf{y}_2)$, and all operations are elementwise. Only the central portion of the output, corresponding to indices $n = N/2+1, \dots, N+N/2$, is retained to form the final result $\widehat{f_\alpha}(x)$.

In the MATLAB implementation, we use the built-in `fft` and `ifft` functions. To align the output with centrally defined variables, we need to shift our centred vector before and after applying `fft` and `ifft` functions. We divide $\widehat{f_\alpha}(x)$ by 2π after the inverse transform to match the Fourier scaling convention used throughout this work.

2.5.3 Root finding methods

Given a nonlinear function $f(x)$, the root finding problem is to find the solution of $f(x) = 0$. There are two main groups of methods for solving this problem: closed-domain methods and open-domain methods.

Closed-domain methods, such as bisection and false position, start with two bounds that produce function values with different signs. The correct solution lies within these bounds, and we can successively reduce the interval to eventually reach the solution. Closed-domain methods are guaranteed to converge but can be slow.

In contrast, open-domain methods do not trap the solution within an interval. Famous examples of open-domain methods include Newton's method, fixed-point iteration, and the secant method. According to Press et al. (2007) and Hoffman and Frankel (2018), open-domain methods are considerably more efficient, even though they can diverge. Additionally, Newton's method and Halley's method require f to be differentiable, although they can achieve quadratic and cubic convergence rates, respectively.

In the following, we describe some root-finding methods relevant to this thesis. The first one is called *fixed-point iteration* or the *fixed-point algorithm*. The procedure starts by rearranging $f(x) = 0$ into $x = g(x)$, which has the same solution. Then, starting from an initial guess x_0, it updates x iteratively by setting $x_1 = g(x_0)$, until the solution no longer changes significantly. Upon reaching this stable point, the algorithm finds the solution that satisfies $x = g(x)$. The detailed algorithm is given in Algorithm 1.

In practice, we normally set a maximum number of iterations to prevent the program from running endlessly when the problem is divergent. If the method converges, the convergence rate is linear. This can be accelerated using acceleration methods, such as Anderson acceleration (Anderson, 1965).

Algorithm 1 Fixed-point iteration

```
1: Select an initial guess $x_0$, a tolerance error $\varepsilon > 0$ and maximum number of iterations $K$.
2: while $||x^{k+1} - x^k|| \geq \varepsilon$ and $k < K$ do
3:     $x^{k+1} = g(x^k)$
4:     $k = k + 1$
5: end while
6: Return $x^k$
```

Another approach to address slow convergence is to use a different root-finding method, such as the *secant method*, which is similar to Newton's method but approximates the derivative $f'(x)$ using finite differences. The recurrence relation for finding the solution is

$$x^{(n)} = x^{(n-1)} - f(x^{(n-1)}) \frac{x^{(n-1)} - x^{(n-2)}}{f(x^{(n-1)}) - f(x^{(n-2)})}, \tag{2.71}$$

and we require two initial values $x^{(0)}$ and $x^{(1)}$, which should be close to the correct solution. The order of convergence is approximately 1.618, meaning the convergence is super-linear but not quadratic.

Muller's method is a higher-order version of the secant method, with an order of convergence around 1.8. It is used less frequently, often because the secant method is already so efficient that the slightly more complicated logic of the higher-order method is not justified, according to Hoffman and Frankel (2018).

There are some challenges when applying root-finding methods. First, finding a good initial approximation is very important for convergence. A good initial guess should be close to the true solution. A poor initial guess can lead to divergence, slow convergence, or convergence to the wrong root. According to Hoffman and Frankel (2018); Press et al. (2007), this issue can be addressed either by plotting the function or by conducting a fine incremental search. Second, slow convergence can occur. As mentioned before, this can be mitigated by either finding a better initial guess or switching to a more efficient method.

Chapter 3

Expressions for the joint conditional characteristic function for the Heston model

The stochastic volatility model become increasingly popular since Heston introduce it in 1993 (Heston, 1993). because the Black-Scholes assumption of the geometric Brownian motion with the constant volatility is not consistent with what observed in the market. Whereas, the Heston return distribution has fatter tails and higher peak than the normal distribution. And by making the variance process stochastic, the Heston model enables practitioners and academics to fit the smile and skew patterns of volatility surface by adjusting the model parameters. After Carr and Madan (Carr and Madan, 1999) introduce the fast Fourier transform method for option pricing, the characteristic function has received an increasingly amount of attention, especially the characteristic function for the stochastic volatility model (Albrecher et al., 2007; Cui et al., 2017; Lord and Kahl, 2010). In the same time, there are applications of the fast Fourier transform method with stochastic volatility model in exotic options such as American-style options (Zhylyevskyy, 2010), compound options (Griebsch, 2013) and discrete barrier options (Cai et al., 2021). For pricing discrete barrier option, Bermudan option and other callable option, the payoff at any intermediate time before maturity is depend on the the log price and variance at that particular time. The option price can be expressed as an iterative relationship between the successive monitoring dates. Then in order to capture this iterative relationship we need to use joint conditional probability distribution. The general idea is to derive analytical expressions of the option price in the Fourier space and use the Fourier relationship between joint characteristic function and the joint probability distribution.

The calculation of the characteristic function involves taking the logarithm of complex variables. The characteristic function can be discontinuous due to the restriction of the logarithm to the principle branch as the most software packages do. Albrecher et al. (2007) showed that the original Heston characteristic function suffers from numerical discontinuity when reach a maturity threshold and the second Heston characteristic function formula used in Bakshi et al. (1997), Duffie et al. (2000) and Gatheral (2011) is continuous under almost all combination of parameters. The left missing gap was filled by Lord and Kahl (2010) who also showed how to avoid discontinuous problem in the exact simulation algorithm of the Heston model. Cui et al. (2017) also find a new representation of the characteristic function which is continuous and easy differentiable.

This chapter focuses on solving numerical instability issue that can happen when calculating the joint conditional characteristic function (JCCF). While the main goal is to show how to write the characteristic function in a way that avoids this problem, we start by explaining how the JCCF is derived, which usually involves solving a system of Riccati ordinary differential equations. We then look at two versions of the JCCF proposed by Zhylyevskyy and Griebsch. We find that Zhylyevskyy's version suffers from the so-called "little Heston trap," which causes numerical discontinuities, while the corrected and Griebsch versions stay stable. To fix this problem, we suggest a simple and general method that deals with how the software chooses the principal branch of complex logarithms. This solution is explained in Section 3.2, and we test the different expressions using Monte Carlo simulations.

3.1 Derivation of the joint conditional characteristic function

In this section, we first briefly revisit the model and derive its JCCF expression using a similar method to the one in Heston (1993), Duffie et al. (2000), Lord and Kahl (2010). Zhylyevskyy also used this derivation method, which led to the expression falling into the "little Heston trap" (Albrecher et al., 2007). We provide a correction for this. Following on from this, in Section 3.1.2, we show that Zhylyevskyy's formulation, the corrected version, and Griebsch's formulation are analytically equivalent.

3.1.1 New derivation removing the little Heston trap

Under the risk-neutral pricing measure, the underlying price process $S = \{S(t), t \geq 0\}$ under the Heston model is defined by the following system of stochastic differential equations (SDE)

$$dS(t) = r\,dt + \sqrt{v(t)}\,dW_1(t), \quad S(t) \geq 0, \tag{3.1}$$

$$dv(t) = \kappa(\theta - v(t))\,dt + \sigma\sqrt{v(t)}\,dW_2(t), \tag{3.2}$$

$$dW_1(t)\,dW_2(t) = \rho\,dt \tag{3.3}$$

where $r \geq 0$ is the constant risk-free interest rate, and $W_i = \{W_i(t), t \geq 0\}, i = 1,2$ are two standard Brownian motions. The correlation between them is defined by $\rho \in [-1,1]$. The instantaneous variance $v = \{v(t), t \geq 0\}$ is a mean-reverting process with non-negative parameters κ (mean-reversion speed), θ (long-term mean), and σ (volatility of variance). If the initial variance is greater than 0, the variance process is almost surely positive if it satisfies $2\kappa\theta \geq \sigma^2$, i.e., the Feller condition.

It is notable that a negative ρ is often used, since due to the leverage effect, a downward movement in stock price is typically associated with an upward movement in volatility.

We define $x(t)$ as the logarithm of the underlying price, $x(t) = \log\frac{S(t)}{S(0)}$. According to the definition of an affine process by Duffie et al. (2003), it has been observed that the Heston model is not affine when formulated in terms of (S, v) but becomes affine in $(\log S, v)$; see Duffie et al. (2000); Kallsen (2006); Lord and Kahl (2010). The general theory of affine processes and their analytical tractability in asset pricing is developed in Duffie et al. (2003, 2000). Kallsen (2006) provides a didactic overview of affine stochastic volatility models, while Kallsen et al. (2011) applies these models to the pricing of variance derivatives. A more recent summary of affine and polynomial processes in financial modelling is provided in the chapter *Affine and Polynomial Processes* by Eberlein and Kallsen (2019).

By applying Itô's lemma, we get the SDE for the log-price

$$dx(t) = \left(r - \frac{v(t)}{2}\right) dt + \sqrt{v(t)}\, dW_1(t). \tag{3.4}$$

Using Eq. (3.4) and Eq. (3.2), the log-price at time T becomes

$$\begin{aligned} x(T) = x(t) + r\tau - \frac{1}{2}\int_t^T v(s)\,ds + \frac{\rho}{\sigma}\left(v(T) - v(t) - \kappa\theta\tau + \kappa\int_t^T v(s)\,ds\right) \\ + \sqrt{1-\rho^2}\int_t^T \sqrt{v(s)}\,dW_2(s), \end{aligned} \tag{3.5}$$

where $\tau = T - t$ is time to maturity.

Since $x(t)$ follows an affine process, the JCCF has the form

$$\begin{aligned} \phi(\xi_x, T; \xi_v, T | x, t, v, t) &= \phi(\xi_x, \xi_v, \tau | x, v) \\ &= \mathbb{E}\left[\exp(i\xi_x x(T) + i\xi_v v(T)) \mid x(t), v(t)\right] \qquad (3.6) \\ &= \exp[p(\xi_x, \xi_v, \tau) + q(\xi_x, \xi_v, \tau)v(t) + i\xi_x x(t)], \qquad (3.7) \end{aligned}$$

where $p := p(\xi_x, \xi_v, \tau)$ and $q := q(\xi_x, \xi_v, \tau)$ are determined by solving ODEs. The full derivation is provided in Appendix B. Here, we state the closed-form solutions

$$q = \frac{1}{\sigma^2}\left[\kappa - \rho\sigma i\xi_x + \sqrt{A}\,\frac{Be^{-\tau\sqrt{A}} - 1}{Be^{-\tau\sqrt{A}} + 1}\right], \tag{3.8}$$

where

$$\begin{aligned} A &= (\kappa - \sigma\rho i\xi_x)^2 + \sigma^2(i\xi_x + \xi_x^2), \qquad (3.9) \\ B &= -\frac{\sigma\rho i\xi_x - \kappa + \sqrt{A} + \sigma^2 i\xi_v}{\sigma\rho i\xi_x - \kappa - \sqrt{A} + \sigma^2 i\xi_v}. \qquad (3.10) \end{aligned}$$

$$p = r\tau i\xi_x + \frac{\kappa\theta}{\sigma^2}\left[(\kappa - \sigma\rho i\xi_x)\tau - \tau\sqrt{A} + 2\log\frac{B+1}{Be^{-\tau\sqrt{A}} + 1}\right], \tag{3.11}$$

with $p_0 := p(\xi_x, \xi_v, 0) = 0$.

It should be noted that two values for $\sqrt{A}$ exist. While both are analytically valid, using $\sqrt{A}$ (positive root) leads to numerical instability, whereas $-\sqrt{A}$ avoids the issue (Albrecher et al., 2007; Cui et al., 2017). This instability underlies why Zhylyevskyy's formulation becomes undefined at $(\xi_x, \xi_v, \tau) = (0, 0, \tau)$, as discussed in Section 3.2.

3.1.2 Comparison

The first closed-form expression for the conditional characteristic function of the Heston model $\phi_{\mathsf{H}}(\xi_x, T|x,t;v,t)$ was derived by Heston (1993) using the Feynman-Kac theorem. Later, more numerically stable formulations were proposed (Bakshi et al., 1997; Cui et al., 2017; Duffie et al., 2000; Gatheral, 2011). However, to price discretely monitored path-dependent options, we require the JCCF.

Two expressions for the Heston model JCCF were given by Zhylyevskyy (2010) and Griebsch (2013). The formulation we derived in Section 3.1.1 is equivalent to both and corrects the numerical instability in Zhylyevskyy's version, as discussed below.

3.1.2.1 Zhylyevskyy formulation

Zhylyevskyy's JCCF takes the same form as Eq. (3.7)

$$\phi_{\mathsf{Z}}(\xi_x, \xi_v, \tau|x, v) = \exp[p_{\mathsf{Z}}(\xi_x, \xi_v, \tau) + q_{\mathsf{Z}}(\xi_x, \xi_v, \tau)v(t) + i\xi_x x(t)], \tag{3.12}$$

with

$$p_{\mathsf{Z}} = r\tau i\xi_x + \frac{\kappa\theta}{\sigma^2}\left[(\kappa - \sigma\rho i\xi_x)\tau + \tau\sqrt{A} + 2\log\frac{B_{\mathsf{Z}}+1}{B_{\mathsf{Z}}e^{\tau\sqrt{A}}+1}\right], \tag{3.13}$$

$$q_{\mathsf{Z}} = \frac{1}{\sigma^2}\left[\kappa - \rho\sigma i\xi_x - \sqrt{A}\,\frac{B_{\mathsf{Z}}e^{\tau\sqrt{A}}-1}{B_{\mathsf{Z}}e^{\tau\sqrt{A}}+1}\right], \tag{3.14}$$

where

$$B_{\mathsf{Z}} = -\frac{\sigma\rho i\xi_x - \kappa - \sqrt{A} + \sigma^2 i\xi_v}{\sigma\rho i\xi_x - \kappa + \sqrt{A} + \sigma^2 i\xi_v} = \frac{1}{B}. \tag{3.15}$$

Comparing p vs. p_{Z} and q vs. q_{Z}, we observe that the only difference lies in the sign of $\sqrt{A}$ and the definition of B vs. B_{Z}. Most software defaults to the principal branch $\sqrt{A}$, which introduces problems. For instance

$$\lim_{(\xi_x,\xi_v)\to(0,0)} B_{\mathsf{Z}}(\xi_x, \xi_v) = -\frac{-\kappa-\kappa}{-\kappa+\kappa} = \frac{2\kappa}{0} = \infty, \tag{3.16}$$

making $\phi_{\mathsf{Z}}(0,0)$ undefined. In contrast, using $-\sqrt{A}$ gives

$$\lim_{(\xi_x,\xi_v)\to(0,0)} B(\xi_x,\xi_v) = -\frac{-\kappa+\kappa}{-\kappa-\kappa} = 0. \tag{3.17}$$

Though Zhylyevskyy suggests manually setting $q_{\mathsf{Z}} = p_{\mathsf{Z}} = 0$ at $(\xi_x,\xi_v) = (0,0)$, this does not fix the discontinuity introduced by the complex logarithm. This is discussed further in Section 3.2.

Despite this, the two formulations are analytically equivalent. We observe

$$\begin{aligned} &-\tau\sqrt{A}+2\log\frac{B+1}{Be^{-\tau\sqrt{A}}+1} \\ &=\tau\sqrt{A}+2\log\frac{e^{-\tau\sqrt{A}}(1/B+1)}{e^{-\tau\sqrt{A}}/B+1} \\ &=\tau\sqrt{A}+2\log\frac{B_{\mathsf{Z}}+1}{B_{\mathsf{Z}}e^{\tau\sqrt{A}}+1}, \end{aligned} \tag{3.18}$$

and

$$\sqrt{A}\,\frac{Be^{-\tau\sqrt{A}}-1}{Be^{-\tau\sqrt{A}}+1} = -\sqrt{A}\,\frac{B_{\mathsf{Z}}e^{\tau\sqrt{A}}-1}{B_{\mathsf{Z}}e^{\tau\sqrt{A}}+1}. \tag{3.19}$$

3.1.2.2 Griebsch formulation

Griebsch derived the JCCF differently. She began with Eq. (3.5), and rewrote Eq. (3.6) as

$$\begin{aligned} \phi_{\mathsf{G}}(\xi_x,\xi_v,\tau|x,v) = {}& \exp\left(i\xi_x(x(t)+r\tau-\frac{\rho}{\sigma}v(t)-\frac{\rho}{\sigma}\kappa\theta\tau)\right) \\ & \times\mathbb{E}\left[\exp\left(-b\int_t^T v(u)\,du+av(T)\right)\mid x(t),v(t)\right], \end{aligned} \tag{3.20}$$

where

$$a = i\xi_v + i\xi_x\frac{\rho}{\sigma}, \tag{3.21}$$

$$b = -\frac{\kappa\rho}{\sigma}i\xi_x+\frac{1}{2}i\xi_x+\frac{1}{2}\xi_x^2(1-\rho^2). \tag{3.22}$$

Griebsch (2013) showed the expectation can be solved using the Feynman-Kac framework, giving

$$\begin{aligned}&\phi_{\mathsf{G}}(\xi_x, \xi_v, \tau | x, v)\\&= \exp\left(i\xi_x(x(t) + r\tau - \frac{\rho}{\sigma}v(t) - \frac{\rho}{\sigma}\kappa\theta\tau)\right)\exp\left(p_{\mathsf{G}}(a,b,\tau) + q_{\mathsf{G}}(a,b,\tau)v(t)\right), \end{aligned}\tag{3.23}$$

with

$$p_{\mathsf{G}}(a,b,\tau) = \frac{\kappa\theta}{\sigma^2}\left(\kappa\tau - d\tau + 2\log\frac{2d}{\gamma}\right), \tag{3.24}$$

$$q_{\mathsf{G}}(a,b,\tau) = \frac{-(1-e^{-d\tau})(2b+\kappa a) + da(1+e^{-d\tau})}{\gamma}, \tag{3.25}$$

where

$$d = \sqrt{\kappa^2 + 2\sigma^2 b}, \tag{3.26}$$

$$\gamma = 2de^{-d\tau} + (\kappa + d - \sigma^2 a)(1 - e^{-d\tau}). \tag{3.27}$$

Although this expression looks different from Zhylyevskyy's, they are algebraically equivalent. For instance, one can show $d^2 = A$, and Griebsch also used $-d$ to avoid numerical issues, consistent with our corrected formulation. We also observe

$$\begin{aligned}\log\frac{2d}{\gamma} &= \log\frac{2\sqrt{A}}{2\sqrt{A}e^{-\tau\sqrt{A}} + (\kappa + \sqrt{A} - \sigma^2 i\xi_v - \sigma\rho i\xi_x)(1 - e^{-\tau\sqrt{A}})}\\&= \log\frac{B+1}{Be^{-\tau\sqrt{A}} + 1},\end{aligned}\tag{3.28}$$

which will also be relevant in Section 3.2. Hence, we conclude

$$\phi_{\mathsf{Z}} = \phi = \phi_{\mathsf{G}}.$$

Despite the undefined $\phi_{\mathsf{Z}}(0,0,\tau)$, the three formulations are equivalent. However, Zhylyevskyy's use of the principal square root $\sqrt{A}$ introduces numerical problems, which will be discussed in detail in the next section.

3.2 Complex discontinuity

The previous three expressions of the closed-form Heston JCCF include the logarithm of a complex variable. Depending on which branch is taken in the calculation, this may cause discontinuities in the complex plane. Consider a complex variable $z = \omega e^{i\delta}$, where ω and δ are the modulus and argument of z. Then, by taking the logarithm of z, we get

$$\log(z) = \log|\omega| + i\delta. \tag{3.29}$$

Most programming languages evaluate complex logarithms using the *principal branch* of the argument, which restricts the imaginary part $\delta = \mathrm{Im}(\log z)$ to the interval $(-\pi, \pi]$. As a result, discontinuities naturally arise when the true value of δ falls outside this range. Specifically, if δ exceeds π or drops below $-\pi$, it is wrapped back into the principal branch by subtracting or adding 2π, respectively. This behavior is common across many programming environments such as PYTHON (`numpy.unwrap`), R, and MATLAB.

As an example, Fig. 3.1 illustrates this effect, the red dotted line represents the ideal (continuous) imaginary part of $\log z$ for $\delta \in [0, 5\pi]$, while the blue solid line shows the result returned by MATLAB, where clear discontinuities occur at multiples of 2π.

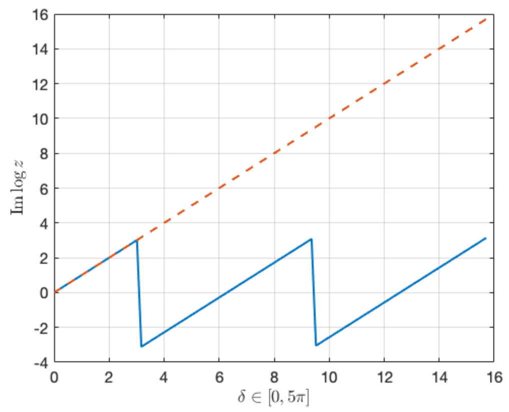


Figure 3.1: Example of discontinuity in the imaginary part of $\log z$ calculated by MATLAB.

In this section, we first test continuity in the complex plane for the JCCFs in Zhylyevskyy's and Griebsch's papers. Then we propose a general solution that addresses discontinuities not only in the Heston JCCF but also in any application where the programming language restricts the imaginary part of the logarithm.

3.2.1 Discontinuity in the Heston JCCF formulations

Table 3.1: Heston model and market parameters.

Model parameters		**Market parameters**	
κ	4	$x(0)$	100
θ	0.035	r	0.05
σ	0.15		
ρ	-0.6		
$v(0)$	0.04		

In Section 3.1.2, we proved that the three JCCF expressions are analytically equivalent. We can then define the following expressions

$$f_1 = \tau\sqrt{A} + 2\log\frac{B\text{Z}+1}{B\text{Z}e^{\tau\sqrt{A}}+1}, \tag{3.30}$$

$$f_2 = -\tau\sqrt{A} + 2\log\frac{B+1}{Be^{-\tau\sqrt{A}}+1}, \tag{3.31}$$

$$f_3 = -d\tau + 2\log\frac{2d}{\gamma}, \tag{3.32}$$

which correspond to Zhylyevskyy's, the corrected, and Griebsch's JCCF formulations, respectively. Although f_1, f_2, and f_3 are analytically equal, we can expect them to return different results for certain values of (ξ_x, ξ_v) in numerical implementations.

Fig. 3.2 illustrates this issue, when fixing ξ_v and plotting $\mathrm{Im}(\log f_i),\ i = 1,2,3$ over $\xi_x \in [0,10]$, the curve for f_1 (Zhylyevskyy's formulation) clearly differs from the others. It displays a zigzag pattern caused by the restriction of the logarithm to the principal branch, which is consistent with the default behaviour of most programming languages. In contrast, the corrected Zhylyevskyy and Griebsch formulations—f_2 and f_3—do not suffer from the branch switching problem as ξ_x increases.

We can understand why this only affects f_1 by inspecting the core structure. Consider the second term in Eq. (3.30)

$$F_1(\xi_x, \xi_v) = \frac{B\text{Z}+1}{B\text{Z}e^{\tau\sqrt{A}}+1}. \tag{3.33}$$

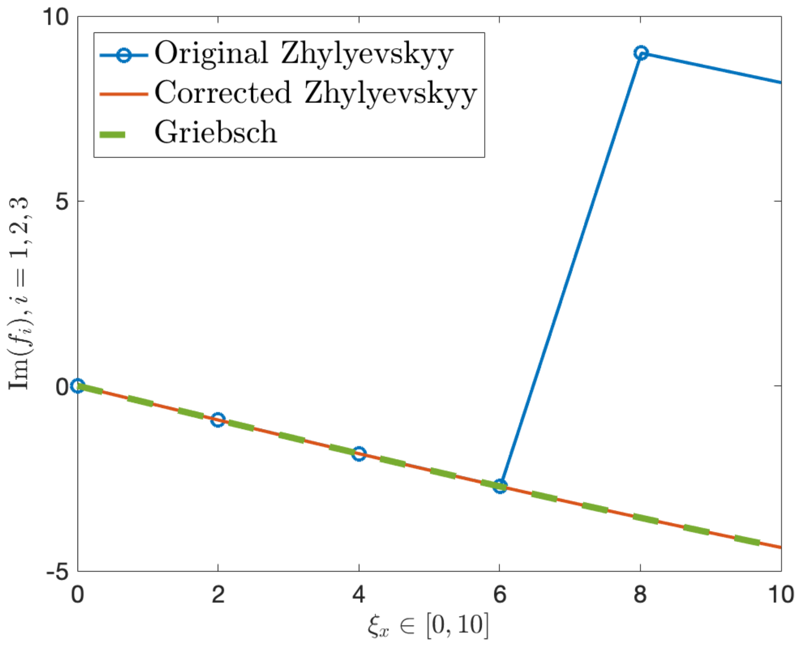


Figure 3.2: The imaginary part of three equivalent expressions $\mathrm{Im}(\log f_i), i = 1,2,3$. The curves are generated using the parameters in Table 3.1 with $\xi_x \in [0,10]$, $\tau = 5$, and fixed $\xi_v = 1.0127$.

For comparison, the similar term from the corrected Zhylyevskyy JCCF is

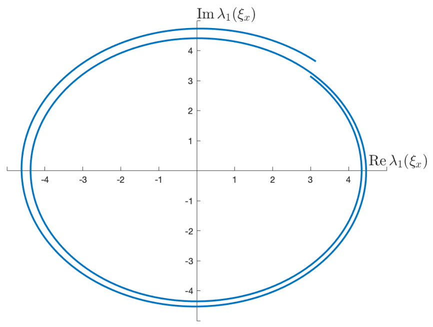
$$F_2(\xi_x, \xi_v) = \frac{B+1}{Be^{-\tau\sqrt{A}}+1}. \tag{3.34}$$

To visualise how the branch cut affects these two functions, we follow the idea in Albrecher et al. (2007); Kahl and Jäckel (2005) and plot the trajectory $\lambda(\xi_x, \xi_v)$ of F_1 and F_2 on the complex plane. It is calculated as

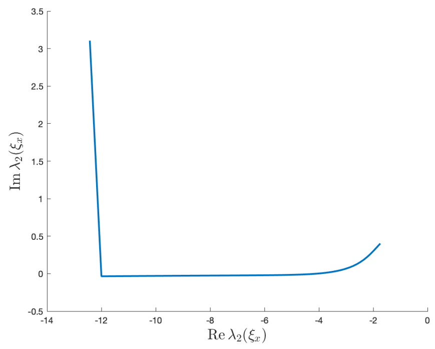
$$\lambda_i(\xi_x, \xi_v) = F_i(\xi_x, \xi_v)\frac{\log\log|F_i(\xi_x, \xi_v)|}{|F_i(\xi_x, \xi_v)|}, \quad i = 1,2, \tag{3.35}$$

where one logarithm comes from the use of $\log F_i$ in the JCCF, and the second compensates for the outward spiral due to growing modulus.

As shown in Fig. 3.3a, λ_1 exhibits a spiral pattern that crosses the negative real axis in the complex plane. In contrast, λ_2 (Fig. 3.3b) does not cross this branch cut. The repeated crossings of the negative real axis in λ_1 cause the discontinuity, as the imaginary part of the logarithm jumps between branches of $(-\pi, \pi]$. The exponential growth in the radius of F_1 results from $e^{\tau\sqrt{A}}$, which itself spirals outward



(a) Trajectory of λ_1.



(b) Trajectory of λ_2.

Figure 3.3: Trajectories of $\lambda_i(\xi_x, \xi_v)$, $i = 1,2$, where $\xi_x \in [0,50]$, $\xi_v = 1.0127$, and $\tau = 5$, generated using the parameters in Table 3.1.

when $\mathrm{Im}(\sqrt{A}) \neq 0$.

Here we can rewrite the logarithm of F_1 as

$$\log F_1 = -(-\log F_1) = -\log \frac{B_Z e^{\tau\sqrt{A}} + 1}{B_Z + 1}. \tag{3.36}$$

If τ is sufficiently large, F_1 is dominated by the term $e^{\tau\sqrt{A}} \frac{B_Z}{B_Z+1}$. In this case, the imaginary part of $\log F_1$ may exceed π, leading to a discontinuity due to branch switching.

3.2.2 Correcting the discontinuity

To avoid the branch cut problem when implementing the solution numerically, one approach is to use the JCCF formulations ϕ and ϕ_G, which do not involve branch switching in the complex logarithm. Another widely applicable method is to apply the unwrapping procedure described in Algorithm 2, which corrects discontinuities in the imaginary part of the logarithm. This procedure identifies jumps greater than or equal to π between consecutive values and adjusts them by adding or subtracting appropriate multiples of 2π to restore continuity.

Algorithm 2 Unwrapping Algorithm

```
Require: A sequence $P = [P_0, P_1, \ldots, P_n]$ representing $\mathrm{Im}(\log F_1)$
Ensure: A new sequence unwrapped with $2\pi$ discontinuities removed
 1: Initialise unwrapped[0] with $P_0$
 2: Set the variable correction to zero
 3: for each index $i$ from $1$ to $n$ do
 4:     Compute the difference $\Delta = P_i - P_{i-1}$
 5:     if $\Delta > \pi$ then
 6:         Subtract $2\pi$ from correction
 7:     else if $\Delta < -\pi$ then
 8:         Add $2\pi$ to correction
 9:     end if
10:     Set unwrapped[i] to $P_i$ plus correction
11: end for
12: return unwrapped
```

This procedure is implemented in high-level languages such as `unwrap` in both MATLAB and PYTHON (`numpy.unwrap`). In the R programming language, similar functionality is available through packages such as `signal` or through custom implementations. In lower-level languages like C++, this functionality is typically not provided by default and must be implemented manually using the logic shown in Algorithm 2.

An alternative technique to handle such discontinuities was introduced by Kahl and Jäckel (2005) and later verified by Lord and Kahl (2006). Instead of correcting each local jump, their method counts the total number of rotations a complex-valued function makes around the origin, and reconstructs the full argument by adding 2π times the rotation number to the wrapped phase.

In our case, let $P = \mathrm{Im}(\log F_1)$. Then the corrected expression for f_1 using the unwrapping method can be implemented as

```
ImagPart = unwrap(imag(log(F1)))
RealPart = real(log(F1))
f4 = tau * sqrt(A) + 2 * complex(RealPart, ImagPart).
```

The resulting expression f_4 avoids the discontinuities in the complex plane, as illustrated in Fig. 3.4.

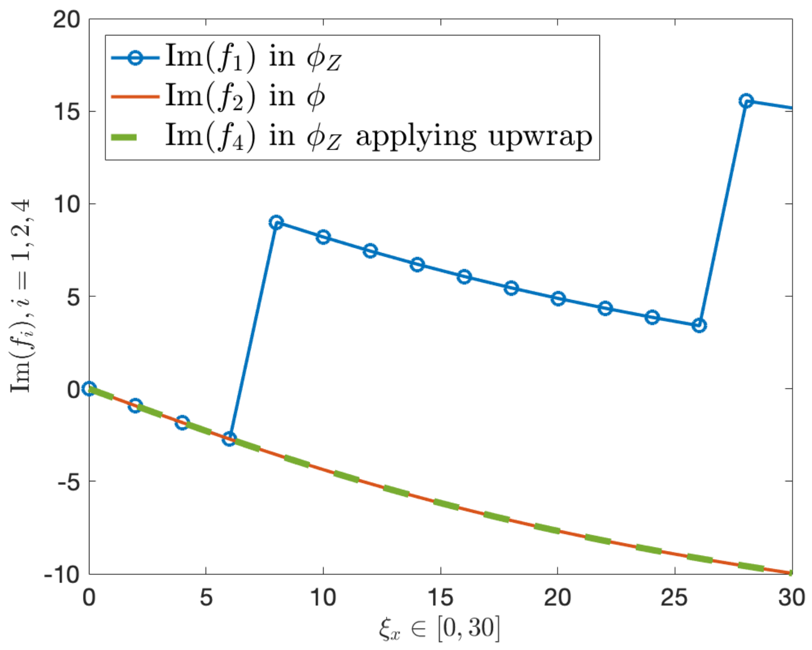


Figure 3.4: The imaginary part of f_1, f_2, and f_4, where f_4 is f_1 after applying the `unwrap` function in MATLAB. The curves are generated using the parameters in Table 3.1, with $\xi_x \in [0, 10]$, $\tau = 5$, and fixed $\xi_v = 1.0127$.

3.2.3 Application to exact simulation methods

The Unwrapping Algorithm 2 can also be applied to other cases involving branch switching on the complex plane. One such example is the exact simulation algorithm for the Heston model proposed by Broadie and Kaya (2006). This method generates exact samples of $S(T)$ from its distribution, conditioned on the terminal value of variance obtained from simulating the variance process.

The main advantage of this exact simulation over conventional Euler discretisation is its faster convergence rate, which leads to lower computational costs for achieving high accuracy. It can also be applied to price exotic path-dependent options, since the required transition probability density can be obtained by taking the inverse Fourier transform of the JCCF constructed during the simulation.

Although the full algorithm is beyond the scope of this paper, one important and relevant step involves simulating the integral $\int_t^T v(u)\,du$ given $v(t)$ and $v(T)$. Because the distribution of $\int_t^T v(u)\,du$ is not available in closed form, Broadie and Kaya derived its probability density by taking the inverse Fourier transform of the following characteristic function

$$\begin{aligned}\Phi(\xi) &= \mathbb{E}\left[\exp\left(i\xi\int_t^T v(u)\,du\right) \mid v(t), v(T)\right] \\ &= \frac{\eta(\xi)e^{-0.5(\eta(\xi)-\kappa)\tau}(1-e^{-\kappa\tau})}{\kappa(1-e^{-\eta(\xi)\tau})} \\ &\quad\times\exp\left\{\frac{v(t)+v(T)}{\sigma^2}\left[\frac{\kappa(1+e^{-\kappa\tau})}{1-e^{-\kappa\tau}}-\frac{\eta(\xi)(1+e^{-\eta(\xi)\tau})}{1-e^{-\eta(\xi)\tau}}\right]\right\} \\ &\quad\times\frac{I_\nu\left(\sqrt{v(t)v(T)}\frac{4\eta(\xi)e^{-0.5\eta(\xi)\tau}}{\sigma^2(1-e^{-\eta(\xi)\tau})}\right)}{I_\nu\left(\sqrt{v(t)v(T)}\frac{4\kappa e^{-0.5\kappa\tau}}{\sigma^2(1-e^{-\kappa\tau})}\right)},\end{aligned} \tag{3.37}$$

where $\eta(\xi) = \sqrt{\kappa^2 - 2\sigma^2 i\xi}$, and $\nu = 2\kappa\theta/\sigma^2 - 1$ is the number of degrees of freedom. Here, $I_\nu(\cdot)$ denotes the modified Bessel function of the first kind, which satisfies

$$I_\nu(z) = \left(\frac{1}{2}z\right)^\nu \sum_{j=0}^{\infty} \frac{\left(\frac{1}{4}\nu^2\right)^j}{j!\,\Gamma(\nu+j+1)}, \tag{3.38}$$

where z is a complex number and $\Gamma(\cdot)$ is the gamma function.

Broadie and Kaya found that $I_\nu(z)$ exhibits a branch switching problem which causes the characteristic function in Eq. (3.37) to become discontinuous. This issue arises because evaluating $I_\nu(z)$ requires computing $z^\nu = \exp(\nu\log z)$, and the complex logarithm $\log z$ introduces discontinuity when $\arg(z)$ crosses π.

In addition, to avoid numerical instability when the denominator

$$I_\nu\left(\sqrt{v(t)v(T)}\,\frac{4\kappa e^{-0.5\kappa\tau}}{\sigma^2(1-e^{-\kappa\tau})}\right)$$

becomes close to zero, we compute the ratio of Bessel functions in logarithmic form. The original expression is given in

$$\frac{I_\nu\left(\sqrt{v(t)v(T)}\frac{4\eta(\xi)e^{-0.5\eta(\xi)\tau}}{\sigma^2(1-e^{-\eta(\xi)\tau})}\right)}{I_\nu\left(\sqrt{v(t)v(T)}\frac{4\kappa e^{-0.5\kappa\tau}}{\sigma^2(1-e^{-\kappa\tau})}\right)}, \tag{3.39}$$

which can be rewritten as

$$\exp\left(\log\left[\sqrt{v(t)v(T)}\,\frac{4\eta(\xi)e^{-0.5\eta(\xi)\tau}}{\sigma^2(1-e^{-\eta(\xi)\tau})}\right]-\log\left[\sqrt{v(t)v(T)}\,\frac{4\kappa e^{-0.5\kappa\tau}}{\sigma^2(1-e^{-\kappa\tau})}\right]\right). \tag{3.40}$$

Lord and Kahl (2010) provide a solution that is to find an expression which equivalent to $\frac{\eta(\xi)e^{-0.5\eta(\xi)\tau}}{1-e^{-\eta(\xi)\tau}}$ but continuous on the complex plane. However, a practical limitation of their approach is that it requires deriving a mathematically equivalent but numerically stable expression for $I_v(z)$. In contrast, our proposed Unwrapping Algorithm 2 offers a programming-level solution that addresses the discontinuity directly and can be applied broadly without altering the mathematical form. Since the denominator of Eq. (3.39) is real and does not introduce branch cuts, we only need to unwrap the imaginary part of the numerator's logarithm. Define

$$h(\xi)=\log\left[\sqrt{v(t)v(T)}\,\frac{4\eta(\xi)e^{-0.5\eta(\xi)\tau}}{\sigma^2(1-e^{-\eta(\xi)\tau})}\right], \tag{3.41}$$

and apply the algorithm to $\mathrm{Im}(h(\xi))$.

In practice, this can be efficiently implemented using the built-in `unwrap` function in MATLAB, which follows the same logic as Algorithm 2.

Figure 3.5 shows the imaginary part of $h(\xi)$ before and after unwrapping. The discontinuities caused by the complex logarithm are effectively removed by this simple correction, demonstrating the practical utility of the algorithm.

3.3 Numerical results

In Section 3.1.2, we analytically demonstrated that the three joint characteristic function (JCCF) formulations, ϕ, ϕ_Z, and ϕ_G, are mathematically equivalent. However, as discussed in Section 3.2, the formulation ϕ_Z is suffers from numerical discontinuities due to complex logarithmic branch issues.

This section presents numerical evaluations to validate the practical performance of these three formulations.

We begin by testing ϕ, ϕ_Z, and ϕ_G using a representative set of model and market parameters, shown in Table 3.1. This parameter set is also used in the later

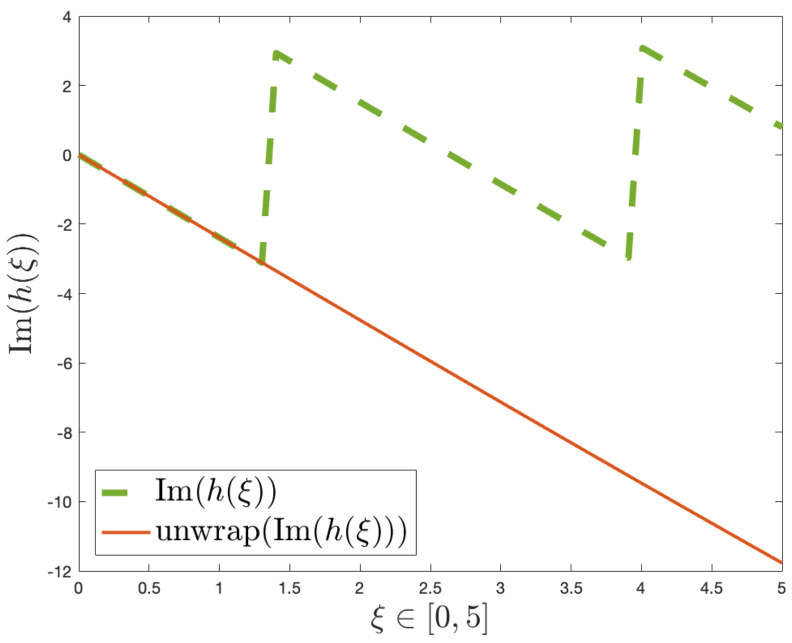


Figure 3.5: The imaginary part of $h(\xi)$ with and without the application of the `unwrap` function in MATLAB. The curves are generated using the parameters in Table 3.1 with $\xi_x \in [0,5]$, $v(t) = 0.04$, $v(T) = 0.045$, and $\tau = 5$.

option pricing analysis to ensure consistency. To validate the characteristic function formulations, we compare each computed JCCF against simulated JCCFs obtained from a Monte Carlo simulation using 10^6 paths, with maturity set to $T = 1$ and time step $\Delta t = 0.01$. The simulation of the Heston model paths is carried out using the Quadratic-Exponential (QE) discretisation scheme proposed by Andersen (2007), which is designed to ensure stability and accuracy across a wide range of parameter values. In parallel, we assess the accuracy of the joint conditional probability density functions (JCPDFs), derived from these JCCFs via inverse Fourier transform, by comparing them to histograms constructed from the same Monte Carlo simulation.

In addition to this baseline case, we include three challenging parameter test cases in Table 3.2 described in Andersen (2007); Glasserman and Kim (2011), which are designed to stress-test the formulations under extreme and realistic market conditions.

To evaluate robustness more broadly, we then extend the analysis by computing the root mean squared error (RMSE) and the maximum absolute error (MAE) across a wider range of model parameters, as detailed in Table 3.3. This extended param-

Table 3.2: Three challenging Heston parameter cases from Andersen (2007). These cases are selected to test numerical robustness under extreme but realistic market conditions.

	Case I	Case II	Case III
κ	0.5	0.3	1.0
θ	0.04	0.04	0.09
σ	1.0	0.9	1.0
ρ	-0.9	-0.5	-0.3

eters range is based on the setup from Cui et al. (2017), but further broadened to ensure coverage of the extreme scenarios presented in the three challenging cases.

3.3.1 Validation of JCCF: baseline case

We first validate the three JCCF analytical expressions ϕ, ϕ_{Z}, and ϕ_{G}, under a standard Heston parameter set defined in Table 3.1.

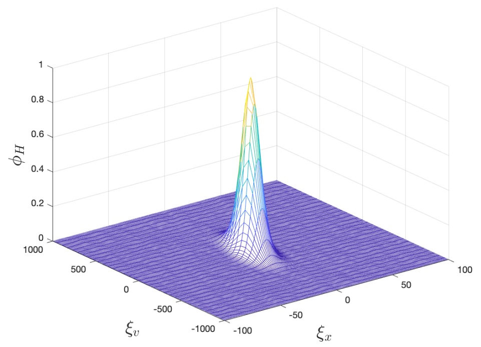
To construct numerical benchmarks, we evaluate the JCCF using two independent methods based on the Monte Carlo simulation with QE scheme. In the first method, we approximate the JCPDF using a histogram over the simulated log-price and variance. A two-dimensional discrete Fourier transform is then applied to recover the corresponding JCCF. In the second method, we compute the empirical JCCF (EJCCF) directly from the simulated data using the expression

$$\phi_{\mathrm{e}}(\xi_x,\xi_v)=\frac{1}{M^2}\left(\sum_{j=1}^{M}\exp(i\xi_x x_j)\times\sum_{j=1}^{M}\exp(i\xi_v v_j)\right), \tag{3.42}$$

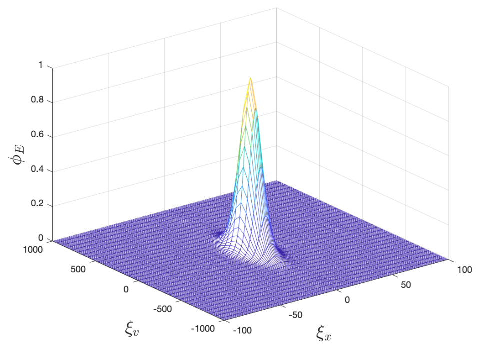
where x_j and v_j are the simulated log-price and variance at maturity.

These two approaches provide benchmark results, as shown in Fig. 3.6a and Fig. 3.6b, which we use to assess the numerical behaviour of the analytical JCCF formulations.

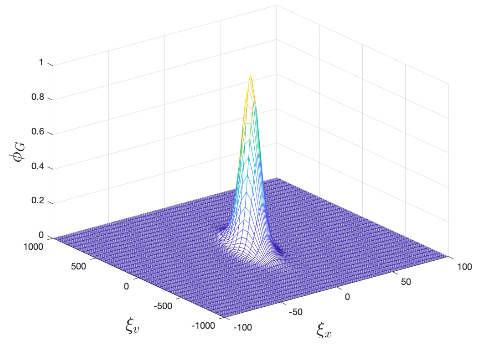
We can notice the benchmark JCCF plots from the two different methods are same. By representing the three JCCF formulations ϕ, ϕ_{Z} and ϕ_{G} in Figs. 3.6e, 3.6d, 3.6c respectively, we can observe ϕ and ϕ_{G} produce the plots matching the benchmarks. It is unsurprising that ϕ and ϕ_{G} have the same plot because we proved the two expressions are exactly identical in Section 3.1.2. In addition, by observing that Figs. 3.6a, 3.6b, 3.6c, 3.6e are identical we can validate the correctness of both



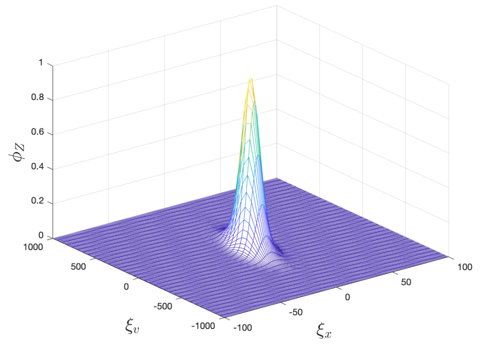
(a) Histogram with 2D Fourier transform



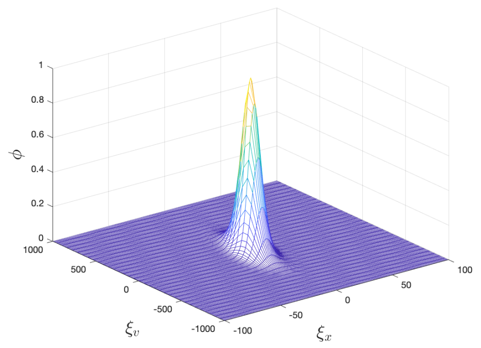
(b) EJCCF



(c) Griebsch

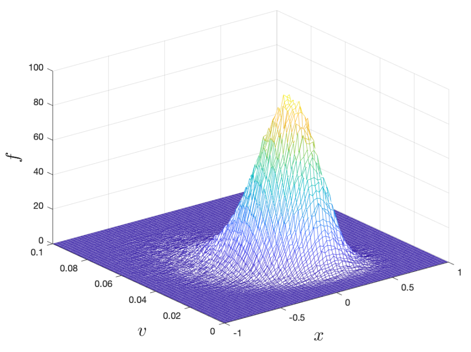


(d) Zhylyevskyy

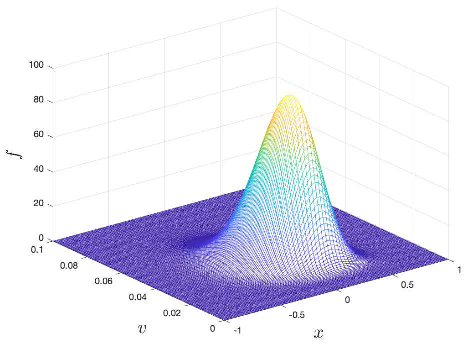


(e) Corrected Zhylyevskyy

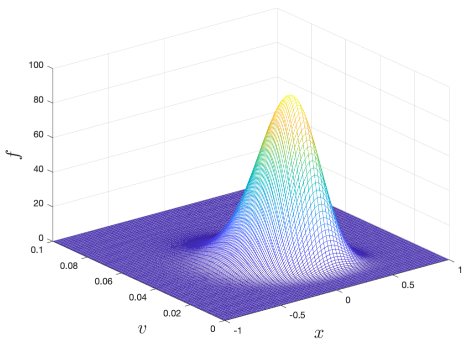
Figure 3.6: Joint conditional characteristic function (JCCF) at maturity T for the baseline Heston parameter set. The benchmark results are obtained from Monte Carlo simulation using the 2D Fourier transform of the joint histogram (3.6a) and the empirical JCCF computed via Eq. (3.42) (3.6b). The analytical JCCFs are shown for Griebsch (3.6c), Zhylyevskyy (3.6d), and corrected Zhylyevskyy (3.6e). Zhylyevskyy's expression is undefined at $(\xi_x, \xi_v) = (0,0)$.



(a) Monte Carlo



(b) Griebsch



(c) Corrected Zhylyevskyy

Figure 3.7: Joint conditional probability density function (JCPDF) at maturity T for the baseline Heston parameter set. The benchmark result is obtained from the Monte Carlo histogram (3.7a). The JCPDFs are computed via inverse 2D Fourier transform of the analytical JCCFs are shown for Griebsch (3.7b) and corrected Zhylyevskyy (3.7c). Due to the undefined value in the original Zhylyevskyy expression, the corresponding JCPDF cannot be recovered directly using standard FFT-based numerical inversion unless manual correction is applied.

ϕ and ϕ_{G}. In contrast, when we examine Fig. 3.6d, one can notice that although everything else matches, the function is undefined at the point $(\xi_x = 0, \xi_v = 0)$ due to the choice of using principal square root, as discussed in Section 3.1.2.

We can further validate the expressions ϕ and ϕ_{G} by applying a 2D inverse Fourier transform to obtain the JCPDFs which are compared with the 2D histogram. The results can be found in Fig. 3.7. Again we can verify both expressions ϕ and ϕ_{G} are correct as they match the result displayed on the histogram.

In addition, we verified the JCCF analytical expressions ϕ and ϕ_{G} by examining the relationship between the JCCF expression and the well-known conditional characteristic function (CCF) for Heston model. We noticed that although there are

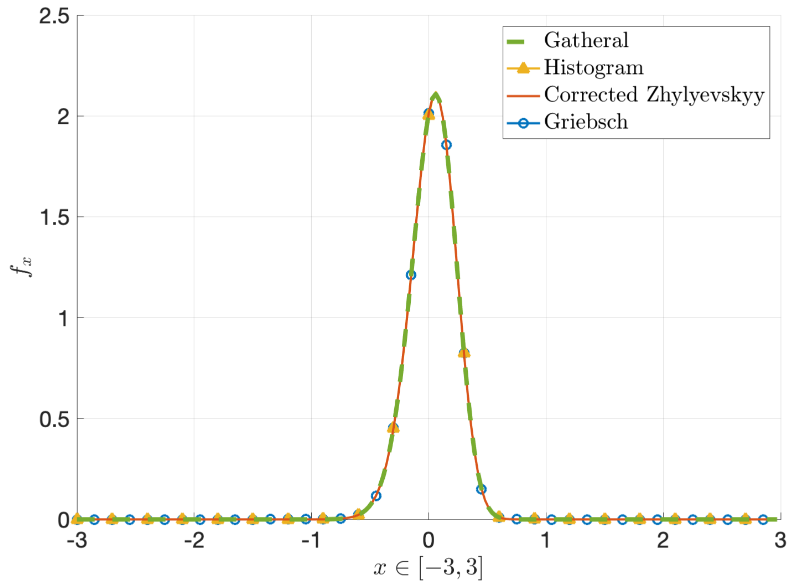


Figure 3.8: Marginal conditional probability density functions for log-price at maturity $T = 5$ under the baseline Heston parameter set (Table 3.1). The results are obtained from the Monte Carlo histogram, the Bakshi conditional characteristic function ϕ_{H}, and the analytical JCCF expressions from corrected Zhylyevskyy ϕ and Griebsch ϕ_{G}. All methods show excellent agreement, supporting the validity of the analytical formulations.

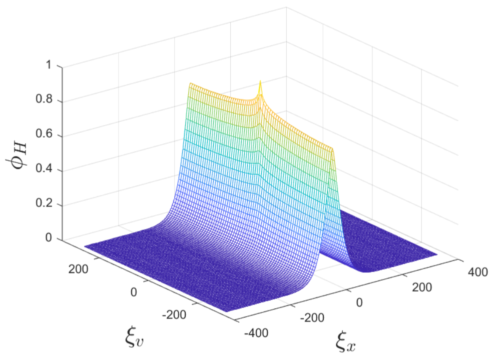
various ways of expressing the Heston CCF ϕ_{H}, these are based are two main formulations. The first of these can be found from the in the original paper of Heston. However as reported by Lord and Kahl (2010) and Albrecher et al. (2007), the original Heston CCF also suffers from the problem of numerical discontinuities in the complex plane, in the same way as we observed in Zhylyevskyy's expression for the JCCF. Therefore here we opt use the second conditional characteristic function from Bakshi et al. (1997) for the validation. The idea is to compute the log-price marginal conditional probability functions from different JCCF expressions ϕ and ϕ_{G} and compare these with the CCF ϕ_{H}. By applying an inverse Fourier transform to ϕ_{H} it is also straightforward to obtain the marginal conditional probability functions for log-price. In contrast, when we begin with the JCCF, ϕ or ϕ_{G}, we need an extra step which is taking the 2D inverse Fourier transform to ϕ or ϕ_{G} to obtain the JCPDF. We can then integrate the JCPDF with respect to variance to get the log-price marginal conditional probability functions. Furthermore, the marginal probability functions calculated from the analytical expressions can also be assessed by comparing them with the numerical marginal probability functions computed from numerically integrating the 2D histogram. Fig. 3.8 provides a summary of the resulting marginal probability functions derived from ϕ_{H}, ϕ, ϕ_{G} and the approximation from the histogram and it can be seen that the plots match each other. It is not unexpected that the probability functions computed from ϕ, ϕ_{G} and histogram are same as they have same JCPDF in Fig. 3.7, but by confirming that the Heston CCF gives the same result, we can verify there is a relationship between the above analytical JCCF expressions (i.e. ϕ and ϕ_{G}) and the CCF, thus providing additional validation of the correctness of the JCCF expressions.

3.3.2 Validation of JCCF: challenging cases

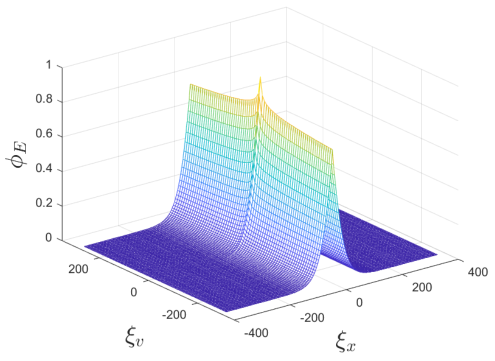
We now assess the performance of the three JCCF formulations under three challenging Heston parameter sets, listed in Table 3.2. These cases are used to test the robustness of numerical methods for the Heston model in Andersen (2007); Glasserman and Kim (2011). According to these sources, Case I corresponds to long-dated FX options, Case II to long-dated interest rate derivatives, and Case III to equity options. Andersen (2007) describes these cases (I–III) as both practically relevant and numerically difficult, due to combinations of high volatility of variance and low mean reversion speed.

To validate the three JCCF formulations in these stress-test scenarios, we repeat the same procedure used in the baseline case. This allows us to examine whether the numerical issues observed in the baseline case persist or become more severe under extreme market conditions, and whether the corrected Zhylyevskyy and Griebsch formulations remain stable and accurate in these more challenging settings.

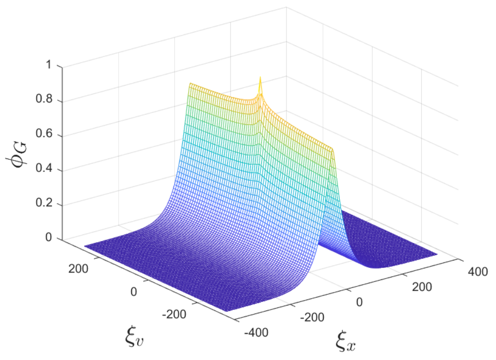
We can observe, these extreme parameter cases in Table 3.2 significantly influence the JCPDFs at maturity. The JCPDFs for Cases I–III, displayed in Figs. 3.12a, 3.13a, and 3.14a, exhibit noticeable skewness and are heavily concentrated near zero variance, in contrast to the more balanced distribution observed in the baseline case (Fig. 3.7a). While a high volatility of variance (σ) allows the variance process to swing widely in both directions, the combination of low mean-reversion speed (κ), small long-term mean (θ), and a low initial variance leads to a strong tendency for the



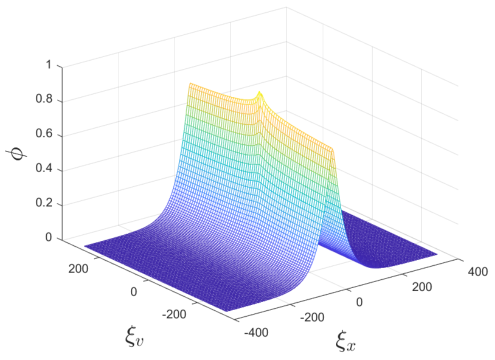
(a) Histogram with 2D Fourier transform



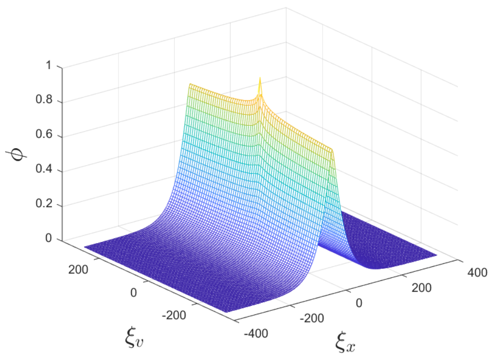
(b) EJCCF



(c) Griebsch

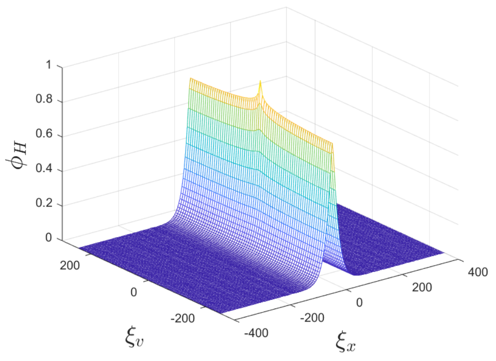


(d) Zhylyevskyy

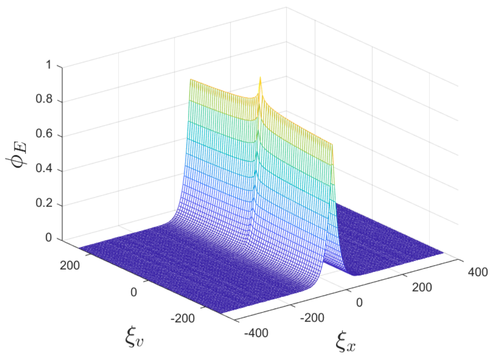


(e) Corrected Zhylyevskyy

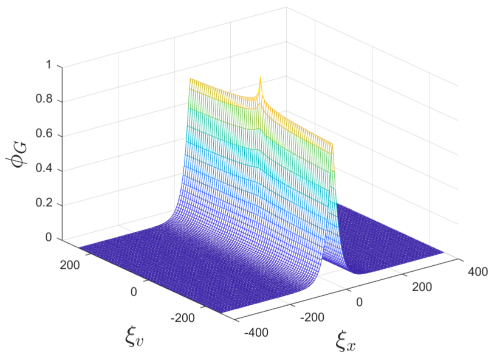
Figure 3.9: Joint conditional characteristic function (JCCF) at maturity T for Heston parameter Case I. The benchmark results are obtained from Monte Carlo simulation using the 2D Fourier transform of the joint histogram (3.9a) and the empirical JCCF computed via Eq. (3.42) (3.9b). The analytical JCCFs are shown for Griebsch (3.9c), Zhylyevskyy (3.9d), and corrected Zhylyevskyy (3.9e). Zhylyevskyy's expression is undefined at $(\xi_x, \xi_v) = (0, 0)$.



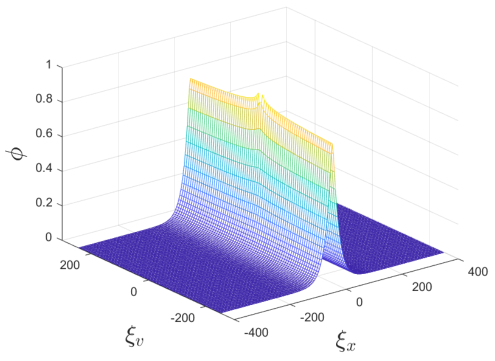
(a) Histogram with 2D Fourier transform



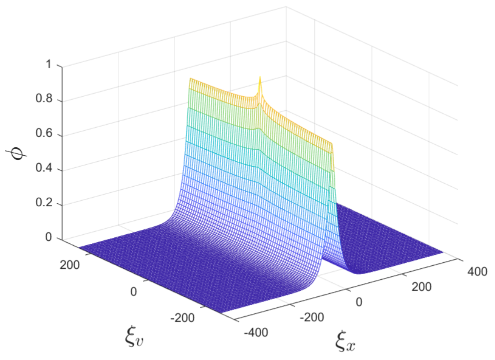
(b) EJCCF



(c) Griebsch

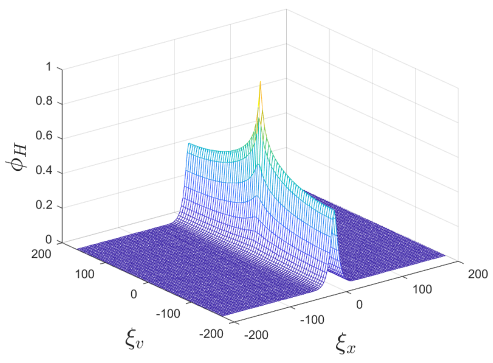


(d) Zhylyevskyy

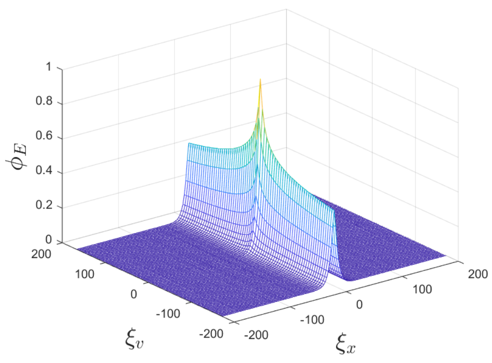


(e) Corrected Zhylyevskyy

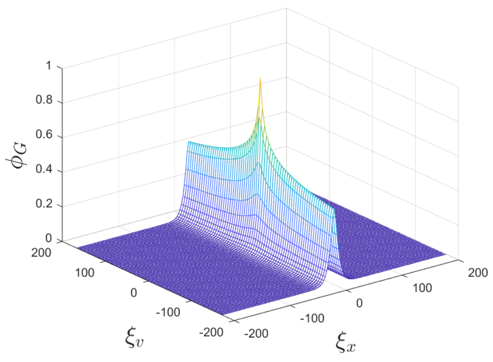
Figure 3.10: Joint conditional characteristic function (JCCF) at maturity T for Heston parameter Case II. The benchmark results are obtained from Monte Carlo simulation using the 2D Fourier transform of the joint histogram (3.10a) and the empirical JCCF computed via Eq. (3.42) (3.10b). The analytical JCCFs are shown for Griebsch (3.10c), Zhylyevskyy (3.10d), and corrected Zhylyevskyy (3.10e). Zhylyevskyy's expression is undefined at $(\xi_x, \xi_v) = (0,0)$.



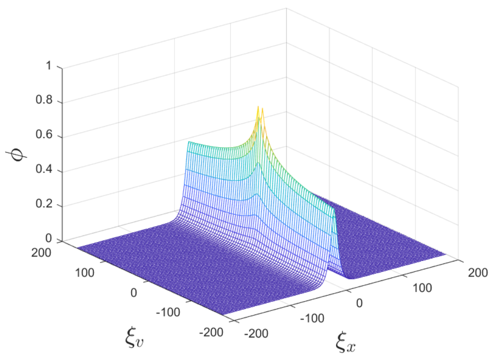
(a) Histogram with 2D Fourier transform



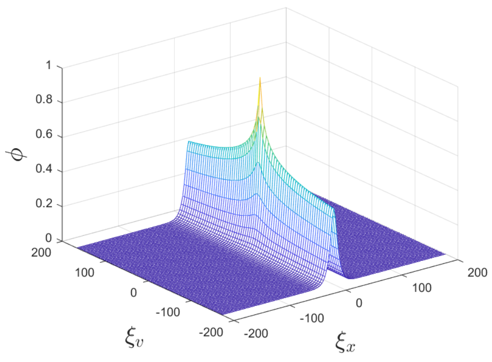
(b) EJCCF



(c) Griebsch

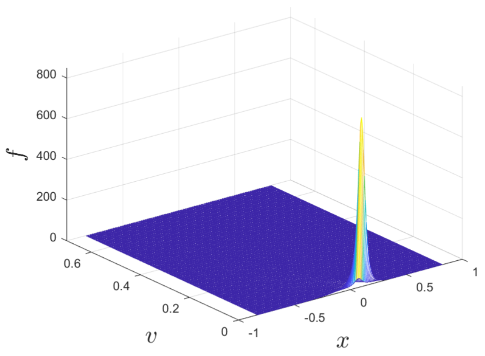


(d) Zhylyevskyy

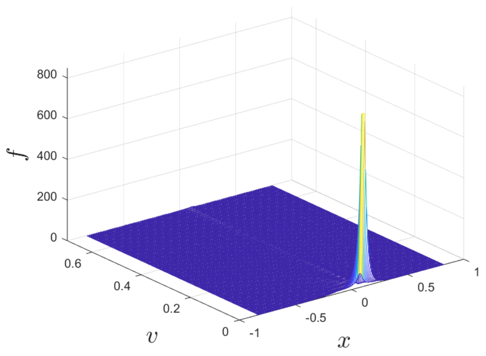


(e) Corrected Zhylyevskyy

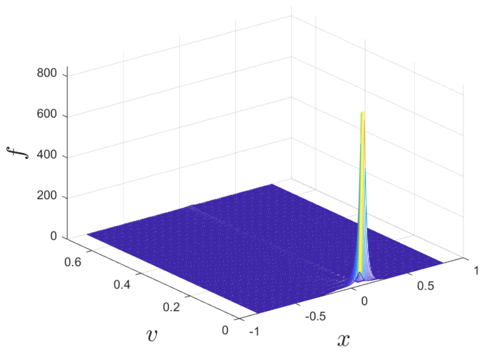
Figure 3.11: Joint conditional characteristic function (JCCF) at maturity T for Heston parameter Case III. The benchmark results are obtained from Monte Carlo simulation using the 2D Fourier transform of the joint histogram (3.11a) and the empirical JCCF computed via Eq. (3.42) (3.11b). The analytical JCCFs are shown for Griebsch (3.11c), Zhylyevskyy (3.11d), and corrected Zhylyevskyy (3.11e). Zhylyevskyy's expression is undefined at $(\xi_x, \xi_v) = (0,0)$.



(a) Monte Carlo



(b) Griebsch

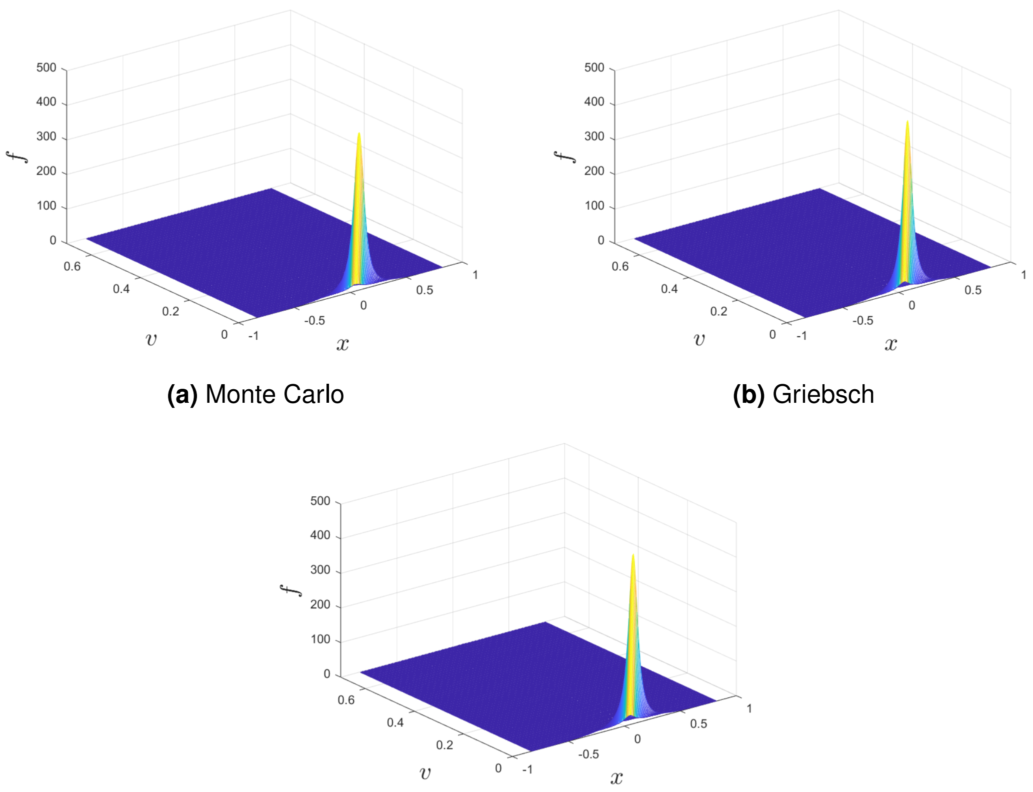


(c) Corrected Zhylyevskyy

Figure 3.12: Joint conditional probability density function (JCPDF) at maturity T for Heston parameter Case I. The benchmark result is obtained from the Monte Carlo histogram (3.12a). The JCPDFs derived from the inverse 2D Fourier transform of the analytical JCCFs are shown for Griebsch (3.12b) and corrected Zhylyevskyy (3.12c). Due to the undefined value in the original Zhylyevskyy expression, the corresponding JCPDF cannot be recovered directly using standard FFT-based numerical inversion unless manual correction is applied.

variance to dip toward zero and remain there. Moreover, the diffusion term ($\sigma\sqrt{v(t)}$) causes the process to diffuse more slowly when $v(t)$ near zero, making it effectively "sticky" once it drops. As a result, despite the potential for large variance excursions, the density in these cases clusters tightly around low variance values, producing a sharp peak in the JCPDF. This behaviour poses notable challenges for numerical methods that depend on the smoothness and spread of the target distribution.

Despite this extremeness, both the corrected Zhylyevskyy and Griebsch expressions show good agreement with the benchmark for the JCCF, as illustrated in Figs. 3.9, 3.10, and 3.11. Likewise, the corresponding JCPDFs, shown in Figs. 3.12, 3.13, and 3.14, closely match the histograms obtained from Monte Carlo simula-



(a) Monte Carlo

(b) Griebsch

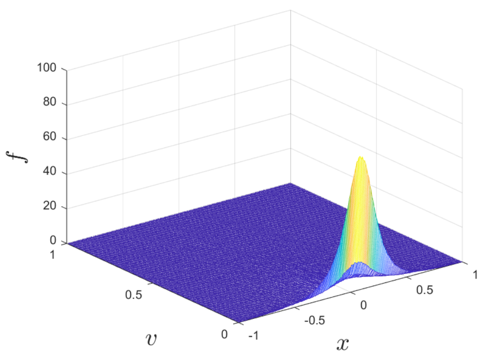
(c) Corrected Zhylyevskyy

Figure 3.13: Joint conditional probability density function (JCPDF) at maturity T for Heston parameter Case II. The benchmark result is obtained from the Monte Carlo histogram (3.13a). The JCPDFs derived from the inverse 2D Fourier transform of the analytical JCCFs are shown for Griebsch (3.13b) and corrected Zhylyevskyy (3.13c). Due to the undefined value in the original Zhylyevskyy expression, the corresponding JCPDF cannot be recovered directly using standard FFT-based numerical inversion unless manual correction is applied.

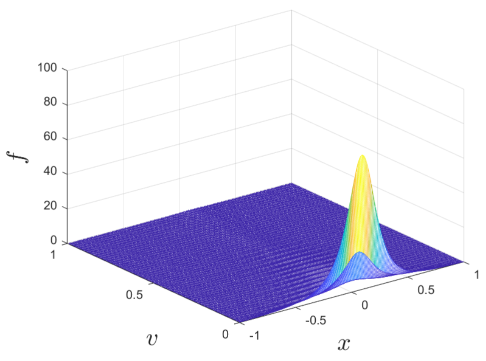
tions. For the original Zhylyevskyy formulation, we again observe a `NaN` value at the origin $(0, 0)$, consistent with the issue encountered in the baseline case. Overall, the results are qualitatively similar to those observed under the standard parameter set, indicating that both the corrected Zhylyevskyy and Griebsch expressions remain robust under extreme market conditions. A more quantitative evaluation, based on RMSE and MAE, will be presented in the next subsection.

3.3.3 Error analysis

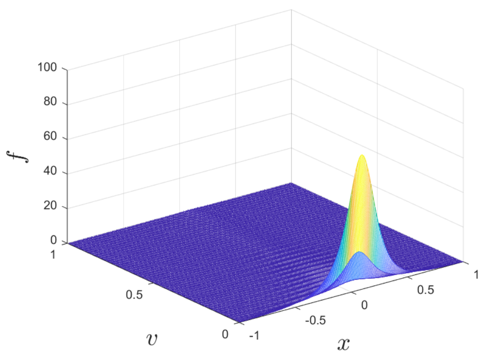
In Sects. 3.3.1 and 3.3.2, we presented visual comparisons between the analytical JCCF expressions: Griebsch, Zhylyevskyy, and the corrected Zhylyevskyy, and Monte Carlo benchmarks for both the baseline parameter set (Table 3.1) and the



(a) Monte Carlo



(b) Griebsch



(c) Corrected Zhylyevskyy

Figure 3.14: Joint conditional probability density function (JCPDF) at maturity T for Heston parameter Case III. The benchmark result is obtained from the Monte Carlo histogram (3.14a). The JCPDFs derived from the inverse 2D Fourier transform of the analytical JCCFs are shown for Griebsch (3.14b) and corrected Zhylyevskyy (3.14c). Due to the undefined value in the original Zhylyevskyy expression, the corresponding JCPDF cannot be recovered directly using standard FFT-based numerical inversion unless manual correction is applied.

Table 3.3: Reasonable ranges for randomly generating the Heston model parameters.

	Parameter range
κ	$(0.30, 5.00)$
θ	$(0.01, 0.95)$
σ	$(0.01, 1.00)$
ρ	$(-0.90, 0.10)$
$v(0)$	$(0.01, 0.95)$

three challenging scenarios (Table 3.2). While the qualitative agreement in those cases is encouraging, a more comprehensive validation requires testing across a broader range of model configurations. To this end, we conduct a systematic error analysis using a collection of parameter sets randomly sampled from the ranges

Table 3.4: RMSE and MAE comparison of the Zhylyevskyy JCCF expression ϕ_Z (with $(0,0)$ manually filled by 1) and the corrected Zhylyevskyy JCCF expression ϕ, relative to the Griebsch JCCF expression ϕ_G, based on 10,000 randomly sampled parameter sets from the range in Table 3.3.

Method	**RMSE**	**MAE**
Zhylyevskyy	2.0690×10^{-3}	3.3871×10^{-3}
Corrected Zhylyevskyy	1.0854×10^{-14}	6.2253×10^{-14}

specified in Table 3.3. These ranges are based on Cui et al. (2017), Andersen (2007), and Glasserman and Kim (2011), and are designed to cover both typical and extreme market conditions. For each sampled parameter set, we compute the root mean square error (RMSE) and maximum absolute error (MAE). The three analytical formulations are compared internally in Table 3.4, and their accuracy is further assessed against Monte Carlo results in Table 3.6. In addition, we report RMSE and MAE values for the three challenging cases specifically, in order to clearly highlight the behaviour of each formulation under extreme market conditions in Tables 3.5 and 3.7.

Please note that since the JCCFs are complex-valued, we compute the error using the complex modulus. By defining

$$\Delta\phi(\xi_x,\xi_v) := \phi_{\text{test}}(\xi_x,\xi_v) - \phi_{\text{ref}}(\xi_x,\xi_v),$$

the error is computed as

$$|\Delta\phi(\xi_x,\xi_v)| = \sqrt{\left(\Re[\Delta\phi(\xi_x,\xi_v)]\right)^2 + \left(\Im[\Delta\phi(\xi_x,\xi_v)]\right)^2}.$$

This captures discrepancies in both the real and imaginary parts and provides a unified and consistent measure of numerical accuracy.

Tables 3.4 and 3.5 report the numerical comparison between the analytical JCCF expressions. In both tables, the Griebsch expression ϕ_G is treated as the benchmark, while the corrected Zhylyevskyy expression ϕ and the original Zhylyevskyy expression ϕ_Z are used as test cases. Although we proved in Section 3.1.2 that all three expressions are analytically equivalent, the original Zhylyevskyy form ϕ_Z

Table 3.5: RMSE and MAE comparison of the Zhylyevskyy JCCF expression ϕ_{Z} (with $(0,0)$ manually filled by 1) and the corrected Zhylyevskyy JCCF expression ϕ, relative to the Griebsch JCCF expression ϕ_{G}, across the three challenging parameter cases in Table 3.2.

Case	Method	RMSE	MAE
I	Zhylyevskyy	2.8575×10^{-2}	1.7398×10^{-1}
	Corrected Zhylyevskyy	7.5490×10^{-17}	8.6736×10^{-16}
II	Zhylyevskyy	2.8230×10^{-17}	2.3714×10^{-16}
	Corrected Zhylyevskyy	3.6080×10^{-17}	8.4691×10^{-16}
III	Zhylyevskyy	1.8860×10^{-17}	5.1348×10^{-16}
	Corrected Zhylyevskyy	2.8895×10^{-17}	8.3831×10^{-16}

is undefined at $(\xi_x, \xi_v) = (0,0)$. To enable a fair numerical comparison, we manually assign $\phi_{\mathrm{Z}}(0,0) = \phi(0,0) = \phi_{\mathrm{G}}(0,0) = 1$ prior to computing the RMSE and MAE.

The results show that ϕ consistently yields RMSE and MAE values close to zero when compared with ϕ_{G}, supporting our theoretical claim of equivalence and demonstrating the reliability of the corrected expression in numerical implementations. In contrast, ϕ_{Z} exhibits noticeably larger errors, particularly in challenging Case I (Table 3.5). These discrepancies are attributable to numerical instabilities arising from branch discontinuities, as discussed in Section 3.2. Specifically, when the volatility of variance (σ) is large and the correlation (ρ) is strongly negative—as is the case in many of the randomly sampled sets and in Case I—these parameters contribute to large values of $\sqrt{A}$ in the characteristic function. This increases the likelihood of crossing a branch cut in the complex plane, which leads to discontinuities and numerical errors in ϕ_{Z}, even when the undefined origin is patched manually.

Since the corrected Zhylyevskyy and Griebsch expressions yield essentially indistinguishable numerical results, it is sufficient for the remaining validation to focus on comparing ϕ and ϕ_{Z} against benchmark values from Monte Carlo simulations.

Table 3.6 presents the RMSE and MAE results for both the JCCFs and JCPDFs computed from the analytical expressions ϕ_{Z} and ϕ_{G}, compared against numerical benchmarks obtained from Monte Carlo simulations. The JCCF benchmarks include both the histogram-based JCCF and EJCCF, while the JCPDF benchmarks are obtained by applying inverse FFT to EJCCF and the histogram-based approximation. As mentioned earlier, we fill the undefined value at $(\xi_x = 0, \xi_v = 0)$ in ϕ_{Z} with 1 to

Table 3.6: RMSE and MAE comparison of the Zhylyevskyy JCCF expression ϕ_Z (with the value at $(0,0)$ manually filled) and the Griebsch JCCF expression ϕ_G, relative to numerical benchmark results, across 10,000 test cases. Comparisons are reported for histogram-based JCCF, EJCCF, and histogram-based JCPDF.

Method	**Histogram JCCF**		**EJCCF**		**Histogram JCPDF**	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Zhylyevskyy	0.0368	1.4773	0.0075	0.1594	2.6173	3.5236
Griebsch	0.0368	1.4773	0.0073	0.1583	2.6016	3.4388

ensure consistency in the comparison.

We first focus on the JCCF results. The RMSE values are consistently very small across all benchmark types, confirming that both ϕ_Z and ϕ_G closely match the Monte Carlo-based numerical JCCFs on average. This reinforces the overall reliability of the analytical expressions in capturing the joint distribution behaviour. However, the MAE results reveal a clear difference between the two benchmark types. When comparing against the EJCCF, the MAEs remain small, indicating excellent pointwise agreement. In contrast, the MAEs are significantly larger when using the histogram-based JCCF as the benchmark. This could be attributed to numerical artifacts introduced by the 2D FFT used to recover the JCCF from the discretised histogram, such as frequency leakage and truncation effects. Although each JCCF is theoretically bounded in magnitude by 1, the pointwise norm $|\phi_Z - \phi_G|$ can still exceed 1 when phase mismatches occur—for example, when the real and imaginary parts have similar magnitudes but opposite signs. These discrepancies are not observed in the EJCCF, which is computed directly from Monte Carlo samples without a Fourier transform, and consistently yields lower MAE.

We now turn to the JCPDF results in Table 3.6. Although the RMSE and MAE values are noticeably higher than those for the JCCF, they remain small enough to be considered acceptable. This is expected, as probability densities are not bounded in the same way characteristic functions are. As observed in the JCPDF plots in Sects. 3.3.1 and 3.3.2, the density values can vary significantly across different model parameter sets. In particular, for extreme cases, the probability mass is often concentrated within a narrow region in both the log-price and variance axes, resulting in large peak values in the JCPDF, see Figs. 3.12, 3.13, and 3.14. These high-density

Table 3.7: RMSE and MAE comparison of the Zhylyevskyy JCCF expression ϕ_Z (with the value at $(0,0)$ manually filled) and the Griebsch JCCF expression ϕ_G, relative to numerical benchmarks under the three challenging parameter cases in Table 3.2. Comparisons are reported for the histogram-based JCCF, EJCCF, and histogram-based JCPDF.

Case	**Method**	**Histogram JCCF**		**EJCCF**		**JCPDF**	
		RMSE	MAE	RMSE	MAE	RMSE	MAE
I	Zhylyevskyy	0.2798	1.8207	0.0625	0.2209	4.5218	260.8515
	Griebsch	0.2799	1.8207	0.0532	0.1387	3.5831	235.7050
II	Zhylyevskyy	0.1922	1.8045	0.0188	0.0845	0.6836	33.5421
	Griebsch	0.1922	1.8045	0.0188	0.0845	0.6836	33.5421
III	Zhylyevskyy	0.0965	1.6624	0.0121	0.1042	0.3642	6.1504
	Griebsch	0.0965	1.6624	0.0121	0.1042	0.3642	6.1504

regions tend to amplify numerical noise introduced during the inverse Fourier transform, leading to greater pointwise discrepancies and, consequently, higher RMSE and MAE values for the JCPDF.

From Table 3.6, we observe that the Griebsch expression ϕ_G consistently achieves slightly lower RMSE and MAE values than the Zhylyevskyy expression ϕ_Z across all benchmark types for the randomly sampled test cases. This indicates that ϕ_G provides better numerical stability, benefiting from its formulation that avoids issues related to branch discontinuities and phase sensitivity.

Similar behaviour is observed in the challenging parameter cases presented in Table 3.7. The overall errors remain low for the JCCF and EJCCF comparisons, although we still observe elevated MAE in the histogram-based JCCF due to phase mismatches, as discussed earlier. The JCPDF errors are higher than those of the JCCF, again due to amplification effects in high-density regions. Notably, the JCPDF error values align with the structure of the densities observed in Figs. 3.12, 3.12 and 3.14: Case I, which has the sharpest peak, yields the highest RMSE and MAE, while Case III, with the broadest density, results in the lowest errors.

Overall, both analytical expressions perform reasonably well under extreme parameter settings, with the Griebsch expression demonstrating consistently superior numerical accuracy.

3.4 Conclusion

In this chapter, we conducted a detailed comparative study of two JCCF formulations for the Heston stochastic volatility model—Zhylyevskyy's and Griebsch's. We began by identifying a subtle numerical issue in Zhylyevskyy's original formulation, referred to as the "little Heston trap", and showed that despite being undefined at $(0, 0)$, the original Zhylyevskyy expression is analytically equivalent to both the corrected version and the Griebsch formulation. This equivalence was formally established through direct mathematical comparison.

However, unlike Griebsch's formulation, the original Zhylyevskyy expression suffers from complex discontinuities arising from branch switching in logarithmic and square root terms. To address this, we proposed an unwrapping algorithm that resolves the numerical discontinuity and ensures the function remains continuous across the complex plane. Details of this algorithm and the underlying numerical issues were discussed in Section 3.2.

To validate the correctness and practical robustness of the JCCF formulations, we performed a comprehensive set of numerical experiments across a wide range of Heston model parameter sets, including extreme cases reported in the literature. These validations included both visual comparisons through plots and quantitative comparisons using RMSE and MAE metrics. As detailed in Section 3.3, the results confirm the analytical equivalence of the three JCCFs, and demonstrate that the corrected Zhylyevskyy and Griebsch expressions produce numerically identical and highly accurate results across both standard and challenging regimes. In contrast, the original Zhylyevskyy formulation exhibits larger numerical errors due to its discontinuity.

Chapter 4

Pricing discretely monitored barrier options with Heston using the Wiener-Hopf method

In this chapter, we price discretely monitored single-barrier options under the Heston stochastic volatility model. We focus on the down-and-out put option as our working example, although the methods developed are easily adaptable to other single-barrier structures. This type of option becomes worthless if the underlying asset touches a lower barrier at any of the discrete monitoring dates. The price of the corresponding down-and-in put option can be computed as the difference between a standard European put and the down-and-out put. Call options can be priced by modifying the payoff function, and Haug (1999) provides an approximate call-put transformation applicable to barrier options.

We introduce two new Fourier-based pricing methods tailored specifically to the Heston model. The first extends the work of Fusai et al. (2016), originally developed for Lévy processes, to the Heston setting. This method applies the z-transform in the discrete-time domain to collapse the time dimension and reformulates the problem as a system of Wiener-Hopf equations, which are solved via fixed-point iteration. The second method is a recursive scheme that applies the Hilbert transform at each monitoring date to enforce the barrier condition in the Fourier domain. It extends the approach of Feng and Linetsky (2008), also originally designed for Lévy processes.

The choice of the Heston model is motivated by both theoretical tractability and practical relevance in financial markets. Practitioners widely use the Heston model because it captures key stylised facts observed in market option prices, such as the volatility smile and the term structure of implied volatility. It introduces stochastic volatility while retaining analytical tractability, making it a strong candidate for fast and robust pricing algorithms. Its ability to fit implied volatility surfaces with relatively few parameters also makes it particularly suitable for calibration and risk management.

Although our method is implemented using the Heston model, the underlying structure is not limited to it. More generally, the approach is compatible with any affine stochastic volatility model that admits a closed-form joint characteristic function of the log-return and future variance, conditional on the current variance. Our method also relies on the model being time-homogeneous—meaning that the conditional distribution of the log-price and variance depends only on their difference, not their absolute values. This property is naturally satisfied by affine stochastic volatility models, as established in Duffie et al. (2003, 2000). These models enable the required conditional transform—developed in detail in the first section—to be obtained via numerical inversion in the variance dimension without requiring an explicit transition density. A key example is the Bates model, which extends the Heston framework by incorporating jumps in the asset price while maintaining analytical tractability. The joint characteristic function of the log-price and future variance in the Bates model is derived in Zhylyevskyy (2012), making it a practical candidate for applying our method beyond the pure diffusion setting.

We also explore several numerical enhancements aimed at improving convergence and computational performance, including the fractional Fourier transform and Anderson acceleration for fixed-point iteration in the Wiener-Hopf solver.

In the final section, we present numerical results comparing the proposed methods—with and without these enhancements—against benchmark methods, including Monte Carlo simulation and the COS method from Fang and Oosterlee (2011). We assess accuracy across different grid sizes in both the log-price and variance dimensions, and test performance under varying numbers of monitoring dates and challenging Heston parameters, as defined in Table 3.2.

4.1 Fourier z-transform method

We now present the first of two Fourier-based methods for pricing discretely monitored barrier options under the Heston model. This method extends the z-transform approach introduced by Fusai et al. (2016)—originally developed for Lévy processes—to the stochastic volatility setting. By collapsing the time dimension via the z-transform and reformulating the pricing problem as a system of Wiener-Hopf equations, we obtain a numerically tractable scheme tailored for the Heston framework.

Let N denote the total number of discrete monitoring dates. We define the time-to-maturity index $n \in \{0, 1, \ldots, N\}$, where $n = 0$ corresponds to maturity and $n = N$ corresponds to the initial time. The associated monitoring times are given by $t_n = (N - n)\Delta t$, where $\Delta t = T/N$ is the constant time step, so that $t_0 = T$ and $t_N = 0$. That is,

$$t_N < \cdots < t_n < \cdots < t_0 = T.$$

Using the notation from Section 2.3.3.3 for the log-price and variance, we denote $x_n = x_{t_n}$ and $v_n = v_{t_n}$, i.e., the log-price and variance at time t_n.

The barrier condition is only monitored at these fixed dates. Let $V_n(x_n, v_n)$ denote the price of the option at time t_n, given log-price x_n and variance v_n. To remain consistent with the backward indexing of time, we define the initial asset price as S_N instead of S_0, so that the shifted log-price becomes

$$x_n = \log(S_n/K).$$

The log-barrier is then $l = \log(L/K)$. The option payoff at maturity $T = t_0$ is therefore given by

$$V_0(x_0, v_0) = K \max(1 - e^{x_0}, 0)\, \mathbf{1}_{(l,\infty)}(x_0). \tag{4.1}$$

Then, the option price at the initial time is expressed as

$$V_N(x_N, v_N) = e^{-rT}\mathbb{E}\left[V_0(x_0, v_0)\mathbf{1}_{(l,\infty)}(x_1)\cdots\mathbf{1}_{(l,\infty)}(x_{N-1}) \mid \mathscr{F}_N\right]. \tag{4.2}$$

To simplify the treatment of the barrier condition, we shift the log-price coordinate by subtracting the log-barrier level. Specifically, we define

$$V_{n,l}(x_n', v_n) := V_n(x_n' + l, v_n) = V_n(x_n, v_n), \tag{4.3}$$

where $x_n = x_n' + l$, and the barrier is relocated to zero in the shifted coordinate system. Under this transformation, the option becomes worthless if $x_n' < 0$ at any monitoring date.

Without loss of generality, and to simplify notation in this section, we drop the prime and treat x_n as the shifted log-price (i.e., the distance from the barrier). Accordingly, the barrier condition is applied at zero, and the option price becomes

$$V_{N,l}(x_N, v_N) = e^{-rT}\mathbb{E}\left[V_{0,l}(x_0, v_0)\mathbf{1}_{(0,\infty)}(x_1)\cdots\mathbf{1}_{(0,\infty)}(x_{N-1}) \mid \mathscr{F}_N\right], \tag{4.4}$$

where the terminal payoff is now expressed as

$$V_{0,l}(x_0, v_0) = \phi(x_0) = K\max(1 - e^{x_0+l}, 0)\,\mathbf{1}_{(0,\infty)}(x_0). \tag{4.5}$$

Using the tower property

$$\begin{aligned}
V_{N,l}(x_N, v_N) &= e^{-rT}\mathbb{E}\left[\mathbb{E}\big[V_{0,l}(x_0, v_0)\prod_{j=1}^{N-2}\mathbf{1}_{(0,\infty)}(x_j) \,\big|\, \mathscr{F}_{N-1}\big]\mathbf{1}_{(0,\infty)}(x_{N-1}) \,\big|\, \mathscr{F}_N\right] \\
&= e^{-r\Delta t}\mathbb{E}\left[e^{-r(N-1)\Delta t}\,\mathbb{E}\big[V_{0,l}(x_0, v_0)\prod_{j=1}^{N-2}\mathbf{1}_{(0,\infty)}(x_j) \,\big|\, \mathscr{F}_{N-1}\big]\,\mathbf{1}_{(0,\infty)}(x_{N-1}) \,\big|\, \mathscr{F}_N\right] \\
&= e^{-r\Delta t}\mathbb{E}\left[V_{N-1,l}(x_{N-1}, v_{N-1}) \mid \mathscr{F}_N\right]. \qquad (4.6)
\end{aligned}$$

Due to the Markov property of the Heston model, the option price at time t_N can be written as a double integral over the joint transition density $g(y, w \mid x_N, v_N)$, which gives the conditional probability of transitioning to log-price y and variance w over a single time step

$$V_{N,l}(x_N, v_N) = e^{-r\Delta t}\int_0^\infty\int_0^\infty V_{N-1,l}(y, w)\,g(y, w \mid x_N, v_N)\,dw\,dy, \quad x_N \geq 0,\ v_N > 0, \tag{4.7}$$

where $g(\cdot, \cdot \mid \cdot, \cdot)$ is the joint transition density of the Heston model, as constructed via inverse Fourier transform in Eq. (2.15) and further detailed in Section 2.3.3.3. Since the recursive relationship holds for all $n > 1$, we can generalise the expression for any monitoring date as

$$V_{n,l}(x_n, v_n) = e^{-r\Delta t} \int_0^\infty \int_0^\infty V_{n-1,l}(y,w)\, f(y - x_n \mid w, v_n)\, p(w \mid v_n)\, dw\, dy, \tag{4.8}$$

where $x_n \geq 0,\ v_n > 0$, and $f(\cdot \mid \cdot,\cdot)$ denotes the transition density of the log-price conditional on the variances. And $p(\cdot \mid \cdot)$ is the transition density of the variance process over one time step, which is defined in Eq. (2.17).

Following Fusai et al. (2006), we apply the z-transform to the time domain in order to convert the recursive pricing relation into a system of Wiener-Hopf equations in the Fourier-z domain. The z-transform of a time-indexed sequence $V_{n,l}(x,v)$ is defined as

$$\widetilde{V}_{q,l}(x,v,q) := \sum_{n=0}^{\infty} q^n V_{n,l}(x,v), \tag{4.9}$$

where $q \in \mathbb{C}$ is the transform variable.

In Eq. (4.8), the right-hand side involves the term $V_{n-1,l}$, which naturally suggests summing from $n = 1$. We therefore begin by applying the summation from $n = 1$ to both sides

$$\sum_{n=1}^{\infty} q^n V_{n,l}(x_n, v_n) = \sum_{n=1}^{\infty} q^n e^{-r\Delta t} \int_0^\infty \int_0^\infty V_{n-1,l}(y,w) f(y - x_n \mid w, v_n) p(w \mid v_n)\, dw\, dy, \tag{4.10}$$

where $x_n \geq 0,\ v_n > 0$. By Fubini's theorem, and because the joint density $f(y - x_n \mid w, v_n)p(w \mid v_n)$ is time-homogeneous, we fix $x_n = x$ and $v_n = v$, and interchange the sum and integral. This gives

$$\begin{aligned}
&\sum_{n=0}^{\infty} q^n V_{n,l}(x,v) - V_{0,l}(x_0, v_0) \\
&= q e^{-r\Delta t} \int_0^\infty \int_0^\infty \sum_{n=0}^{\infty} \left[q^n V_{n,l}(y,w) \right] f(y - x \mid w, v) p(w \mid v)\, dw\, dy,
\end{aligned} \tag{4.11}$$

where $x \geq 0,\ v > 0$. By definition of the z-transform, this can be written as

$$\widetilde{V}_{q,l}(x,v,q) - \phi(x_0) = q e^{-r\Delta t} \int_0^\infty \int_0^\infty \widetilde{V}_{q,l}(y,w,q) f(y - x \mid w, v) p(w \mid v)\, dw\, dy, \tag{4.12}$$

where $x \geq 0,\ v > 0$.

Due to the convolution-like form of the integral over the log-price in Eq. (4.12), we follow the method of Fusai et al. (2006) and apply the Wiener-Hopf technique in the Fourier domain. However, the integral over volatility does not lend itself to a Fourier treatment. To address this, we apply Gauss quadrature to approximate the integration over variance.

Let N_v denote the number of quadrature nodes. Then, for a discretised volatility grid with nodes v_i and associated weights τ_i, we approximate the integral in Eq. (4.12) as

$$\widetilde{V}_{q,l}(x,v_i,q) - \phi(x_0) = qe^{-r\Delta t}\sum_{j=1}^{N_v}\tau_j\int_0^{\infty}\widetilde{V}_{q,l}(y,w_j,q)f(y-x \mid w_j,v_i)p(w_j \mid v_i)\,dy, \quad (4.13)$$

where $x \geq 0$, v_i and τ_i denote the Gauss quadrature nodes and weights, respectively, for $i = 1,\dots,N_v$. This formulation yields a system of Wiener-Hopf equations.

In practice, the computational complexity for handling the variance dimension is $\mathscr{O}(N_v^2)$, as each volatility node interacts with all others in the grid. Since this component significantly impacts overall computational time, it is desirable to keep N_v as small as possible, while still achieving a sufficiently accurate approximation. More detailed numerical results and parameter sensitivity analyses are provided in Section 4.3.

In a similar manner to solving a single Wiener-Hopf equation (Wiener and Hopf, 1931), we now extend both the integral and the domain of the function over the entire real line. We define

$$\widetilde{V}_{q,l,+}(x,v_i,q) = \begin{cases} \widetilde{V}_{q,l}(x,v_i,q), & x \geq 0, \\ 0, & x < 0, \end{cases}$$

$$\widetilde{V}_{q,l,-}(x,v_i,q) = \begin{cases} 0, & x \geq 0, \\ qe^{-r\Delta t}\sum_{j=1}^{N_v}\tau_j\int_{-\infty}^{+\infty}\widetilde{V}_{q,l,+}(y,w_j,q)f(y-x \mid w_j,v_i)p(w_j \mid v_i)\,dy, & x < 0. \end{cases}$$

The use of these extensions allows the full integral equation to be rewritten over the entire real line

$$\widetilde{V}_{q,l,+}(x,v_i,q) - qe^{-r\Delta t}\sum_{j=1}^{N_v}\tau_j\int_{-\infty}^{+\infty}\widetilde{V}_{q,l,+}(y,w_j,q)f(y-x\mid w_j,v_i)p(w_j\mid v_i)\,dy$$
$$+\widetilde{V}_{q,l,-}(x,v_i,q) = \phi(x_0), \qquad x\in\mathbb{R}. \qquad (4.14)$$

According to the risk-neutral valuation framework, the option price is the expectation of the discounted payoff. As first shown by Lewis (2001), and subsequently applied throughout the option pricing literature (e.g., Phelan et al. (2019)), this expectation can be computed efficiently in the Fourier domain by leveraging the Plancherel theorem.

However, as explained in detail by Feng and Linetsky (2008), the Fourier transform of a call payoff is not defined on the real line due to the unbounded nature of the function as $x\to\infty$. To ensure convergence of the Fourier integral, a damping factor is introduced to the payoff in Eq. (4.5). We therefore define the damped put option payoff function as

$$\phi_d(x_0) = e^{\alpha x_0}K\max(1-e^{x_0+l},0)\,\mathbf{1}_{(0,\infty)}(x_0)\,\vartheta \qquad (4.15)$$

where $\vartheta=-1$ and $\alpha<0$ for a call option, and $\vartheta=1$, $\alpha>0$ for a put option.

In practice, since our pricing scheme is numerical, the integration is performed only over a finite domain, i.e., $x\in[x_{\min},x_{\max}]$. While this makes the FFT numerically tractable, it does not address the integrability issues of unbounded payoffs. The damping factor is therefore retained to ensure that the Fourier transform of the payoff is well-defined by enforcing exponential decay at infinity (see, e.g., Feng and Linetsky (2008)).

Multiplying Eq. (4.14) by $e^{\alpha x}$, we obtain

$$\begin{aligned}\phi_d(x_0) =& e^{\alpha x}\widetilde{V}_{q,l,+}(x,v_i,q) + e^{\alpha x}\widetilde{V}_{q,l,-}(x,v_i,q)\\ &- qe^{-r\Delta t}e^{\alpha x}\sum_{j=1}^{N_v}\tau_j\int_{-\infty}^{+\infty}\widetilde{V}_{q,l,+}(y,w_j,q)f(y-x\mid w_j,v_i)p(w_j\mid v_i)\,dy. \qquad (4.16)\end{aligned}$$

Here we will apply the one-dimensional Fourier transform in the log-price domain to Eq. (4.16), for all $i = 1, 2, \ldots, N_v$. First we define

$$\int_{-\infty}^{+\infty} e^{i(\xi - i\alpha)x} \widetilde{V}_{q,l}(x, v_i, q)\, dx := \widehat{\widetilde{V}}_{q,l,+}(\xi - i\alpha, v_i, q).$$

Next, consider the Fourier transformation of integral $\int_{-\infty}^{+\infty} e^{\alpha x} \widetilde{V}_{q,l}(y, w_j, q) f(y - x \mid w_j, v_i)\, dy$ we have,

$$\begin{aligned}
&\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{i(\xi - i\alpha)x} \widetilde{V}_{q,l}(y, w_j, q) f(y - x \mid w_j, v_i)\, dy\, dx \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{i(\xi - i\alpha)(y-z)} \widetilde{V}_{q,l}(y, w_j, q) f(z \mid w_j, v_i)\, dz\, dy \\
&= \left(\int_{-\infty}^{+\infty} e^{i(\xi - i\alpha)y} \widetilde{V}_{q,l}(y, w_j, q)\, dy\right)\left(\int_{-\infty}^{+\infty} e^{i(-\xi + i\alpha)z} f(z \mid w_j, v_i)\, dz\right) \\
&= \widehat{\widetilde{V}}_{q,l,+}(\xi - i\alpha, w_j, q)\, \Psi(-[\xi - i\alpha] \mid w_j, v_i).
\end{aligned}$$

Therefore, we arrive at

$$\begin{aligned}
\widehat{\phi_d}(\xi) =& \widehat{\widetilde{V}}_{q,l,+}(\xi - i\alpha, v_i, q) + \widehat{\widetilde{V}}_{q,l,-}(\xi - i\alpha, v_i, q) \\
& - q e^{-r\Delta t} \sum_{j=1}^{N_v} \tau_j \widehat{\widetilde{V}}_{q,l,+}(\xi - i\alpha, w_j, q)\, \Psi(-[\xi - i\alpha] \mid w_j, v_i)\, p(w_j \mid v_i). \qquad (4.17)
\end{aligned}$$

Here, $\Psi(\cdot \mid \cdot, \cdot)$ denotes the conditional characteristic function of the log-price increment, and $\Phi(\cdot)$ is defined in Eq. (3.37)

$$\Psi(\xi \mid v_{t+\Delta t}, v_t) = e^{i\xi\left[r\Delta t + \frac{\rho}{\sigma}(v_{t+\Delta t} - v_t - \kappa\theta\Delta t)\right]} \Phi\left(\xi\left(\frac{\kappa\rho}{\sigma} - \frac{1}{2}\right) + \frac{1}{2} i\xi^2 (1 - \rho^2)\right). \qquad (4.18)$$

The derivation of $\Psi(\cdot \mid \cdot, \cdot)$ is provided in Appendix A. And $\widehat{\phi_d}(\xi)$ is the Fourier transform of the damped payoff with shifted log-price

$$\widehat{\phi_d}(\xi) = K\left(\frac{e^{-l(\alpha + i\xi)} - 1}{\alpha + i\xi} - \frac{e^{-l(\alpha + i\xi)} - e^{l}}{1 + \alpha + i\xi}\right), \qquad (4.19)$$

The product $\Psi(-[\xi - i\alpha] \mid w_j, v_i)\, p(w_j \mid v_i)$ in Eq. (4.17) can be interpreted as the inverse Fourier transform (in the variance-frequency domain) of the joint characteristic function of the log-return and future variance, conditional on the current variance

$$\Psi(-[\xi - i\alpha] \mid w_j, v_i)\, p(w_j \mid v_i) = \mathscr{F}_\omega^{-1}\left[\Psi(\xi, \omega \mid v_i)\right](w_j),$$

where $\Psi(\xi, \omega \mid v_i)$ denotes the joint conditional characteristic function, and $\mathscr{F}_\omega^{-1}$ is defined in Eq. (2.19).

This allows Eq. (4.17) to be rewritten in a more general form applicable to models that admit a closed-form joint characteristic function

$$\begin{aligned}\widehat{\phi}_d(\xi) =& \widehat{\widetilde{V}}_{q,l,+}(\xi - i\alpha, v_i, q) + \widehat{\widetilde{V}}_{q,l,-}(\xi - i\alpha, v_i, q)\\ &- qe^{-r\Delta t} \sum_{j=1}^{N_v} \tau_j \widehat{\widetilde{V}}_{q,l,+}(\xi - i\alpha, w_j, q)\, \mathscr{F}_\omega^{-1}\left[\phi(\xi, \omega \mid v_i)\right](w_j).\end{aligned} \tag{4.20}$$

This formulation enables the method to accommodate models like the Bates model, which extends Heston by including jumps in the asset price. As demonstrated in Zhylyevskyy (2012), the JCCF for the Bates model is available in closed form, and can be substituted directly into Eq. (4.20), allowing our numerical scheme to be applied without modification to its structure. However, in this chapter, we focus on the Heston model and retain Eq. (4.17), which provides the required quantity directly and therefore avoids the need for an additional numerical inverse Fourier transform in the variance dimension.

The system of equations described by Eq. (4.17) can be written in matrix form as

$$\widehat{\mathbf{L}}(\xi)\, \widehat{\widetilde{\mathbf{V}}}_+(\xi) + \widehat{\widetilde{\mathbf{V}}}_-(\xi) = \widehat{\mathbf{V}}_0(\xi). \tag{4.21}$$

Here we define $\widehat{\mathbf{L}}(\xi)$ as

$$\begin{pmatrix} 1 - q_t\tau_1 p_{11}\Psi_{11}(i\alpha - \xi) & -q_t\tau_2 p_{12}\Psi_{12}(i\alpha - \xi) & \cdots & -q_t\tau_{N_v} p_{1N_v}\Psi_{1N_v}(i\alpha - \xi) \\ -q_t\tau_1 p_{21}\Psi_{21}(i\alpha - \xi) & 1 - q_t\tau_2 p_{22}\Psi_{22}(i\alpha - \xi) & \cdots & -q_t\tau_{N_v} p_{2N_v}\Psi_{2N_v}(i\alpha - \xi) \\ -q_t\tau_1 p_{31}\Psi_{31}(i\alpha - \xi) & -q_t\tau_2 p_{32}\Psi_{32}(i\alpha - \xi) & \cdots & -q_t\tau_{N_v} p_{3N_v}\Psi_{3N_v}(i\alpha - \xi) \\ \vdots & \vdots & \ddots & \vdots \\ -q_t\tau_1 p_{N_v1}\Psi_{N_v1}(i\alpha - \xi) & -q_t\tau_2 p_{N_v2}\Psi_{N_v2}(i\alpha - \xi) & \cdots & 1 - q_t\tau_{N_v} p_{N_vN_v}\Psi_{N_vN_v}(i\alpha - \xi) \end{pmatrix},$$

where $q_t = qe^{-r\Delta t}$, $\Psi_{ij}(\cdot) = \Psi(\cdot \mid w_j, v_i)$, and $p_{ij} = p(w_j \mid v_i)$. And we also define

$$\widehat{\widehat{\mathbf{V}}}_+(\xi) = \begin{pmatrix} \widehat{\widehat{V}}_{q,l,+}(\xi - i\alpha, v_1, q) \\ \widehat{\widehat{V}}_{q,l,+}(\xi - i\alpha, v_2, q) \\ \vdots \\ \widehat{\widehat{V}}_{q,l,+}(\xi - i\alpha, v_{N_v}, q) \end{pmatrix}, \quad \widehat{\widehat{\mathbf{V}}}_-(\xi) = \begin{pmatrix} \widehat{\widehat{V}}_{q,l,-}(\xi - i\alpha, v_1, q) \\ \widehat{\widehat{V}}_{q,l,-}(\xi - i\alpha, v_2, q) \\ \vdots \\ \widehat{\widehat{V}}_{q,l,-}(\xi - i\alpha, v_{N_v}, q) \end{pmatrix}. \tag{4.22}$$

And define the payoff term as

$$\widehat{\mathbf{V}}_0(\xi) = \begin{pmatrix} \widehat{\phi}_d(\xi) \\ \widehat{\phi}_d(\xi) \\ \vdots \\ \widehat{\phi}_d(\xi) \end{pmatrix}. \tag{4.23}$$

To solve this system, we follow the classical Wiener–Hopf factorisation approach. That is, we seek matrix factors $\widehat{\mathbf{L}}_-(\xi)$ and $\widehat{\mathbf{L}}_+(\xi)$ such that

$$\widehat{\mathbf{L}}_-(\xi)\,\widehat{\mathbf{L}}_+(\xi) = \widehat{\mathbf{L}}(\xi),$$

and the inverses retain their analyticity in respective half-planes. Dividing Eq. (4.21) from the left by $\widehat{\mathbf{L}}_-(\xi)$, we get

$$\widehat{\mathbf{L}}_+(\xi)\,\widehat{\widehat{\mathbf{V}}}_+(\xi) + \widehat{\mathbf{L}}_-^{-1}(\xi)\,\widehat{\widehat{\mathbf{V}}}_-(\xi) = \widehat{\mathbf{L}}_-^{-1}(\xi)\,\widehat{\mathbf{V}}_0(\xi). \tag{4.24}$$

The term $\widehat{\mathbf{L}}_+(\xi)\,\widehat{\widehat{\mathbf{V}}}_+(\xi)$ on the left-hand side of Eq. (4.24) is supported only on the positive half-line $x \geq 0$, while $\widehat{\mathbf{L}}_-^{-1}(\xi)\,\widehat{\widehat{\mathbf{V}}}_-(\xi)$ is supported on $x < 0$. Therefore, we decompose the right-hand side of Eq. (4.24) as

$$\mathbf{M}(\xi) = \widehat{\mathbf{L}}_-^{-1}(\xi)\,\widehat{\mathbf{V}}_0(\xi) = \mathbf{M}_+(\xi) + \mathbf{M}_-(\xi), \tag{4.25}$$

with $\mathbf{M}_+(\xi)$ and $\mathbf{M}_-(\xi)$ analytic in the upper and lower complex half-planes, respectively.

We then obtain the solution to the original matrix Wiener–Hopf system

$$\widehat{\widehat{\mathbf{V}}}_{+}(\xi) = \widehat{\mathbf{L}}_{+}^{-1}(\xi)\mathbf{M}_{+}(\xi), \tag{4.26}$$

$$\widehat{\widehat{\mathbf{V}}}_{-}(\xi) = \widehat{\mathbf{L}}_{-}(\xi)\mathbf{M}_{-}(\xi), \tag{4.27}$$

with $\widehat{\widehat{\mathbf{V}}}_{+}(\xi)$ containing the desired option value components in the half-space $x \geq 0$.

While the vector decomposition $\mathbf{M}(\xi) = \mathbf{M}_{+}(\xi) + \mathbf{M}_{-}(\xi)$ is straightforward, which performed elementwise using the Plemelj–Sokhotsky relations in Eqs. (2.41) and (2.42). The matrix factorisation of $\widehat{\mathbf{L}}(\xi)$ into $\widehat{\mathbf{L}}_{-}(\xi)\widehat{\mathbf{L}}_{+}(\xi)$ is considerably more difficult. This longstanding open problem has been widely studied Abrahams (1997); Jones (1984a,b, 1991); Lawrie and Abrahams (2007); Veitch and Abrahams (2007), but a general closed-form solution is not yet available. We therefore follow Fusai et al. (2016) and employ a fixed-point iteration scheme to approximate the matrix factor numerically.

4.1.1 Fixed-point algorithm

To implement the fixed-point algorithm, we first rewrite Eq. (4.17) for each $i = 1, 2, \ldots, N_v$ as

$$\begin{aligned}\widehat{\phi}_d(\xi) =& \widehat{\widehat{V}}_{q,l,+}(\xi - i\alpha, v_i, q)\,[1 - q_t \tau_i \Psi_{ii}(i\alpha - \xi) p_{ii}] \\ & - q_t \sum_{\substack{j \neq i \\ j \in J_{N_v}}} \tau_j \widehat{\widehat{V}}_{q,l,+}(\xi - i\alpha, w_j, q)\,\Psi_{ij}(i\alpha - \xi) p_{ij} + \widehat{\widehat{V}}_{q,l,-}(\xi - i\alpha, v_i, q),\end{aligned} \tag{4.28}$$

where $J_{N_v} = \{1, 2, \ldots, N_v\}$.

Define $\widehat{l}_i(\xi) := 1 - q_t \tau_i \Psi(i\alpha - \xi \mid w_i, v_i) p(w_i \mid v_i)$, and factorise it as $\widehat{l}_i(\xi) = \widehat{l}_{i\oplus}(\xi)\widehat{l}_{i\ominus}(\xi)$, as discussed in Section 2.4.5.1. Dividing the equation through by $\widehat{l}_{i\ominus}(\xi)$ and taking only the positive part, we get

$$\begin{aligned}&\widehat{l}_{i\oplus}(\xi)\,\widehat{\widehat{V}}_{q,l,+}(\xi - i\alpha, v_i, q) \\ &= \left[\widehat{l}_{i\ominus}^{-1}(\xi) q_t \sum_{\substack{j \neq i \\ j \in J_{N_v}}} \tau_j \widehat{\widehat{V}}_{q,l,+}(\xi - i\alpha, w_j, q)\,\Psi_{ij}(i\alpha - \xi) p_{ij} + \widehat{l}_{i\ominus}^{-1}(\xi)\,\widehat{\phi}_d(\xi)\right]_{+},\end{aligned} \tag{4.29}$$

where $[\cdot]_{+}$ denotes the positive part from the decomposition in Section 2.4.5.1.

Dividing both sides by $\widehat{l}_{i\oplus}(\xi)$, we obtain the core of the fixed-point algorithm

$$\begin{aligned}&\widehat{\widehat{V}}_{q,l,+}(\xi - i\alpha, v_i, q)\\ &= \widehat{l}_{i\oplus}^{-1}(\xi)\left[\widehat{l}_{i\ominus}^{-1}(\xi) q_t \sum_{\substack{j\neq i\\ j\in J_{N_v}}} \tau_j \widehat{\widehat{V}}_{q,l,+}(\xi - i\alpha, w_j, q)\,\Psi_{ij}(i\alpha - \xi) p_{ij} + \widehat{l}_{i\ominus}^{-1}(\xi)\,\widehat{\phi}_d(\xi)\right]_+ .\end{aligned} \tag{4.30}$$

Let $\widehat{\widehat{V}}_{i+}(\xi) := \widehat{\widehat{V}}_{q,l,+}(\xi - i\alpha, v_i, q)$ for convenience. The algorithm proceeds as follows

1. Define $m_{\max}$ as the maximum number of iterations and ε as the convergence tolerance.

2. Initialize $\widehat{\widehat{V}}_{i+}^{(0)}(\xi) = 0$ for all i.

3. Set iteration counter $m = 1$.

4. Factorise $\widehat{l}_i(\xi) = 1 - q_t\tau_i\Psi_{ii}(i\alpha - \xi)p_{ii}$ for each $i \in J_{N_v}$.

5. Update

$$\widehat{\widehat{V}}_{i+}^{(m)}(\xi) = \widehat{l}_{i\oplus}^{-1}(\xi)\left[\widehat{l}_{i\ominus}^{-1}(\xi) q_t \sum_{\substack{j\neq i\\ j\in J_{N_v}}} \tau_j \widehat{\widehat{V}}_{j+}^{(m-1)}(\xi)\,\Psi_{ij}(i\alpha - \xi) p_{ij} + \widehat{l}_{i\ominus}^{-1}(\xi)\,\widehat{\phi}_d(\xi)\right]_+ . \tag{4.31}$$

   for all $i \in J_{N_v}$.

6. If

$$\max_{i\in J_{N_v}}\left|\widehat{\widehat{V}}_{i+}^{(m)}(\xi) - \widehat{\widehat{V}}_{i+}^{(m-1)}(\xi)\right| < \varepsilon \quad \text{or} \quad m = m_{\max},$$

   then stop. Otherwise, increment $m \leftarrow m + 1$ and return to Step 5.

7. Compute

$$\mathscr{Z}_{q\to N-1}\left[\mathscr{F}^{-1}_{\xi\to x}\left[\widehat{\widehat{V}}_{i+}^{(m)}(\xi)\right]\right] e^{-\alpha x}, \tag{4.32}$$

   where i corresponds to the current variance level.

Although this algorithm is easy to implement, its convergence is linear, which can lead to long runtimes. To accelerate this process, we apply Anderson acceleration (AA), a method that improves convergence by leveraging previous residuals. AA algorithm is originally introduced by Anderson (1965) and later extended by Walker and Ni (2011), is a technique for accelerating the convergence of fixed-point iterations. Instead of using only the most recent iterate to update the solution, AA constructs each new iterate as a least-squares combination of several past residuals, allowing it to incorporate curvature information implicitly. This can lead to significantly faster convergence, particularly in slowly converging linear systems.

We present the general AA procedure in Algorithm 3.

Algorithm 3 Anderson-accelerated fixed-point iteration

```
1: Choose initial guess $\mathbf{u}^{(0)}$, history depth $d \geq 1$, tolerance $\varepsilon > 0$, and maximum
   iterations $K$.
2: Compute $\mathbf{u}^{(1)} = \mathscr{G}(\mathbf{u}^{(0)})$.
3: while $\|\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}\| \geq \varepsilon$ and $k < K$ do
4:     $m_k = \min\{d, k\}$
5:     Form residual matrix $\mathbf{R}^{(k)} = [\mathbf{r}_{k-m_k}, \dots, \mathbf{r}_k]$, where $\mathbf{r}_i = \mathscr{G}(\mathbf{u}^{(i)}) - \mathbf{u}^{(i)}$
6:     Solve for weights
           $\alpha^{(k)} = \arg\min \|\mathbf{R}^{(k)}\alpha^{(k)}\|_2 \quad \text{s.t.} \sum_{i=0}^{m_k} \alpha_i^{(k)} = 1$
7:     Update
           $\mathbf{u}^{(k+1)} = \sum_{i=0}^{m_k} \alpha_i^{(k)} \mathscr{G}(\mathbf{u}^{(k-m_k+i)})$
8:     $k \leftarrow k+1$
9: end while
10: Return $\mathbf{u}^{(k)}$
```

In our pricing algorithm, we apply AA to accelerate the fixed-point iteration defined in Eq. (4.31). In this context, the mapping $\mathscr{G}(\cdot)$ corresponds to the right-hand side of the equation Eq. (4.31), and the vector $\mathbf{u}^{(k)}$ represents the stacked collection of all values $\widehat{\widehat{V}}_{i+}^{(k)}(\xi)$ for $i \in J_{N_v}$ at iteration k. When $m_k = 0$, the AA algorithm reduces to the basic fixed-point iteration. The constrained least-squares problem ensures the update is a convex combination of past values. In our implementation, we set $d \leq 5$ to balance memory usage and acceleration. The effectiveness of AA in this context is demonstrated and discussed in Section 4.3.

4.2 Fourier-Hilbert transform method

As a second pricing method, we implement the Fourier–Hilbert (FH) approach, which offers an efficient alternative for handling the barrier condition in the Fourier domain. Originally proposed by Feng and Linetsky (2008) and later refined by Fusai et al. (2016), the FH method applies a sinc-based fast Hilbert transform to enforce the barrier constraint directly in the Fourier space.

This technique enables recursive computation of the option value at each monitoring date while maintaining computational efficiency. In our implementation, the lower barrier remains fixed at $l = \log(L/K)$, and the recursion takes the following form

$$V_0(x_0, v_0) = \phi_f(x_0) = K\max(1 - e^{x_0}, 0)\,\mathbf{1}_{(l,\infty)}(x_0), \tag{4.33}$$

$$V_n(x_n, v_n) = e^{-r\Delta t}\,\mathbb{E}[V_{n-1}(x_{n-1}, v_{n-1}) \mid x_n = x, v_n = v]\,\mathbf{1}_{(l,\infty)}(x), \tag{4.34}$$

$$V_N(x_N, v_N) = e^{-r\Delta t}\,\mathbb{E}[V_{N-1}(x_{N-1}, v_{N-1}) \mid x_N = x, v_N = v], \tag{4.35}$$

where $n = N-1, N-2, \ldots, 1$, and the final option price is given by $V_N(x_N, v_N)$.

The one-step forward value $V_n(x_n, v_n)$ can be computed explicitly as

$$\begin{aligned} V_n(x_n, v_n) &= e^{-r\Delta t}\,\mathbb{E}[V_{n-1}(x_{n-1}, v_{n-1}) \mid x_n = x, v_n = v]\,\mathbf{1}_{(l,\infty)}(x) \\ &= e^{-r\Delta t}\int_0^{+\infty}\int_l^{+\infty} V_{n-1}(x_{n-1}, v_{n-1})\,f(x_{n-1} - x \mid v_{n-1}, v) \\ &\qquad p(v_{n-1} \mid v)\,dx_{n-1}\,dv_{n-1}\,\mathbf{1}_{(l,\infty)}(x), \end{aligned} \tag{4.36}$$

where same as Section 4.1, $f(\cdot \mid \cdot, \cdot)$ and $p(\cdot \mid \cdot)$ denote the transition densities of the log-price and variance, respectively.

As in Section 4.1, we apply a damping factor $e^{\alpha x}$ before moving to the Fourier domain. Following Feng and Linetsky (2008), the Hilbert transform can be used to enforce the barrier in Fourier space. However, the inner volatility integral must be evaluated numerically; we use Gauss quadrature for this purpose. Applying quadrature to variance domain of Eq. (4.36) and multiplying by $e^{\alpha x}$, we obtain:

$$\begin{aligned}&e^{\alpha x}V_n(x_n,v_n)\\ &= e^{-r\Delta t}e^{\alpha x}\int_l^{+\infty}\sum_{j=1}^{N_v}\tau_j V_{n-1}(x_{n-1},v_j)f(x_{n-1}-x \mid v_j,v)p(v_j \mid v)\,dx_{n-1}\,\mathbf{1}_{(l,\infty)}(x),\end{aligned} \tag{4.37}$$

where N_v is the number of volatility quadrature nodes, and τ_j are the corresponding weights.

By Fubini's theorem, we can interchange summation and integration. Applying the Fourier transform to Eq. (4.37) and using the generalized Plemelj–Sokhotsky relation Eq. (2.41), we obtain

$$\widehat{V}_n(\xi-i\alpha,v_n) = \frac{1}{2}\widehat{h}(\xi)+\frac{i}{2}e^{i\xi l}\mathscr{H}\left(e^{-i\xi l}\widehat{h}(\xi)\right), \tag{4.38}$$

where

$$\widehat{h}(\xi) = \sum_{j=1}^{N_v}\tau_j\widehat{V}_{n-1}(\xi-i\alpha,v_j)\Psi(-\xi+i\alpha \mid v_j,v_n)p(v_j \mid v_n), \tag{4.39}$$

encodes the recursive structure in Fourier space. The initial transform is given by the damped payoff function

$$\widehat{V}_0(\xi-i\alpha,v_0) = \widehat{\phi}_f(\xi) = K\left(\frac{1-e^{l(\alpha+i\xi)}}{\alpha+i\xi}-\frac{1-e^{l(1+\alpha+i\xi)}}{1+\alpha+i\xi}\right). \tag{4.40}$$

And $\Psi(\cdot|\cdot,\cdot)$ is the characteristic function conditional on the volatility, as before.

As in Section 4.1, the function $\widehat{h}(\xi)$ can alternatively be expressed using the inverse Fourier transform in the variance dimension

$$\widehat{h}(\xi) = \sum_{j=1}^{N_v}\tau_j\widehat{V}_{n-1}(\xi-i\alpha,v_j)\mathscr{F}_\omega^{-1}\left[\Psi(\xi,\omega \mid v_n)\right](v_j), \tag{4.41}$$

where $\Psi(\xi,\omega \mid v_n)$ is the joint characteristic function of the log-return increment and prior variance, conditional on v_n. This representation highlights that any affine stochastic volatility model admitting a closed-form JCCF can be incorporated into our method. In this chapter, however, we focus on the Heston model, for which the product $\Psi(-\xi+i\alpha \mid v_j,v_n)\,p(v_j \mid v_n)$ is known in closed form and does not require numerical inversion.

To price a discretely monitored single barrier option under the Heston model, we compute Eq. (4.38) for each monitoring date up to the final step N, and then perform the inverse Fourier transform

$$V(x_N, v_N) = e^{-\alpha x} \mathscr{F}^{-1}_{\xi \to x_N} \left[\widehat{V}_N(\xi - i\alpha, v_N) \right] \tag{4.42}$$

to obtain the final option price.

4.2.1 Fractional Fourier transform

The fast Fourier transform (FFT) is the standard method for computing the discrete Fourier transform (DFT). In the DFT framework, there is a fixed inverse relationship between the grid spacings Δx and $\Delta \xi$ of the spatial and Fourier grids

$$\Delta \xi = \frac{2\pi}{N_x \Delta x}. \tag{4.43}$$

This constraint means that among the three parameters $\Delta \xi, \Delta x, N_x$, only two can be chosen freely. If $\Delta \xi$ and N_x are selected to ensure accurate numerical integration, Δx is automatically determined by Eq. (4.43). In contrast, the fractional Fourier transform (FRFT) allows $\Delta \xi$ and Δx to be chosen independently.

The FRFT is a generalization of the DFT, and we refer to Section 2.4.2 for its formal definition. The FRFT algorithm can produce results equivalent to FFT with the same computational complexity of $\mathscr{O}(N_x \log N_x)$, and enables independent tuning of resolution in both spatial and Fourier domains. The FRFT order α corresponding to given Δx and $\Delta \xi$ is computed as

$$\alpha = \frac{\Delta \xi \Delta x}{2\pi}. \tag{4.44}$$

To improve numerical accuracy in the inverse transform, we can use the FRFT in place of the standard FFT. This approach leads to better resolution and reduced quadrature error, as discussed by Chourdakis (2005). Specifically, we apply the FRFT to compute the inverse Fourier transforms required in Eq. (4.32) and Eq. (4.42).

Model parameters	κ	θ	σ	v_0	ρ
	4	0.035	0.15	0.04	-0.6
Market parameters	S_0	r			
	100	0.05			
Contract parameters	T	L	K		
	1	80	100		

Table 4.1: Parameters for the Down-and-out discretely monitored barrier put option.

The implementation details of the inverse FRFT are provided in Section 2.5.2. The numerical behaviour of the FRFT will be tested and compared with the FFT in Section 4.3.

4.3 Numerical results

In this section, we present a numerical comparison of the Fourier z-transform method (FZ) and the Fourier-Hilbert transform method (FH) against two benchmark approaches: the Monte Carlo (MC) simulation method and the COS method introduced by Fang and Oosterlee (2011). All methods are used to price discretely monitored down-and-out put barrier options under the Heston model with the parameter sets listed in Table 4.1.

In addition to baseline comparisons, we examine the impact of several numerical enhancements. Specifically, we apply a spectral filter (Planck-taper window) in the ξ_x domain to smooth the characteristic function, following the approach of Phelan et al. (2019); replace the standard Fourier transform with the fractional Fourier transform; and incorporate acceleration techniques into the fixed-point iteration.

To assess the robustness of the proposed methods, we also report results under extreme Heston parameter settings, as summarised in Table 3.2.

One of our benchmark methods is MC simulation. There is an extensive body of literature on MC simulation for the Heston model, largely due to the challenges introduced by the square-root diffusion in the variance process. Standard time-discretisation methods such as Euler and Milstein are easy to implement but are well known to perform poorly in the Heston setting, often producing negative variances or significant bias (Kahl and Jäckel, 2006; Lord et al., 2010). To address these issues, a number of more advanced simulation schemes have been proposed. The exact simulation method by Broadie and Kaya (2006), as well as the Gamma Expansion (GE) method by Glasserman and Kim (2011), provide accurate results but are computationally intensive due to the need to simulate conditional distributions or evaluate special function-based expansions. A more efficient low-bias alternative is the inverse gamma (IG) scheme developed by Tse and Wan (2013). Among these methods, the Quadratic-Exponential (QE) scheme by Andersen (2007) has become one of the most widely adopted due to its balance between accuracy and computational speed. van Haastrecht and Pelsser (2010) demonstrate that QE performs favourably in comparison to other discretisation schemes. Furthermore, Choi and Kwok (2024) emphasise that in high-monitoring-frequency scenarios, time-discretisation-based schemes such as QE are often preferred due to their lower per-step cost. For this reason, we adopt the QE scheme in our simulations of discretely monitored barrier options. The scheme involves switching between two simulation regimes based on a threshold parameter that controls the shape of the conditional variance distribution. We set this threshold to 2, meaning that when the shape parameter exceeds this value, the variance is simulated using an exponential distribution; otherwise, a quadratic transformation is applied. In our MC setup, we use 1,000,000 simulation paths. And to avoid exceeding memory limits in our simulation environment (MATLAB) while maintaining consistent temporal resolution, we cap the total number of time steps at 800, resulting in a step size of approximately 0.00125 across different monitoring frequencies, consistent with common practice in the literature.

Another benchmark method is the COS method, which is known for its high efficiency and accuracy in European-style option pricing, and we adapt it to the barrier setting following the implementation guidelines in Fang and Oosterlee (2009b, 2011). In the COS method, the Fourier-cosine expansion is applied in the log-price dimension to approximate the conditional density, while the integration over the log-variance is performed using numerical quadrature. The pricing is carried out through recursive backward induction across the monitoring dates. In our implementation of the COS method, we use $N_x = 2^8$ and $N_v = 2^7$, where N_x and N_v represent the number of grid points in the log-price and variance dimensions (so same as log-variance), respectively. This resolution aligns with the recommendation in Fang and

Oosterlee (2011), where the authors state that choosing approximately 128 points in both dimensions is usually sufficient. While we find this to be true for standard Heston parameters, we observe that the COS method becomes unstable and fails to converge under more extreme parameters and higher monitoring frequencies (e.g., more than 12 monitoring dates with $T = 1$ in Case I of the challenging cases defined in Table 3.2). In such cases, we adopt Monte Carlo simulation as the primary benchmark.

4.3.1 Accuracy evaluation

In this subsection, we evaluate the accuracy of the FZ method and the FH method in pricing discretely monitored down-and-out put options under the Heston model. We compare the results against two benchmark methods: Monte Carlo simulation and the COS method. The comparison is performed under the standard Heston parameter set listed in Table 4.1 together with market and contract parameters, across different numbers of monitoring dates, denoted by N. Accuracy is assessed by examining the absolute pricing errors of the proposed methods with respect to the benchmarks.

To achieve accurate pricing results, it is essential to control two key sources of numerical error, the quadrature error from approximating the variance integral, and the approximation error from the Fourier transform, which is itself a numerical integration. Both types of errors are sensitive to the choice of integration domain. In particular, the integration ranges for both the log-price and the variance must be carefully chosen, if the domain is too narrow, truncation errors will dominate; if it is too wide, a prohibitively large number of grid points may be required to maintain accuracy.

To balance this trade-off, we determine the integration ranges based on the distributional behaviour of the log-price and variance under the Heston model. The log-price, conditional on the variance path, is approximately normally distributed, as discussed in Gatheral (2011). For the variance process, we use its analytical conditional mean and variance to approximate its distribution and define the integration domain accordingly. Specifically, the ranges are set to span four standard deviations around the mean for both variables. This choice provides sufficient coverage of the probability mass under the Heston dynamics and has been empirically tested to yield accurate results without requiring excessive resolution.

Table 4.2: Down-and-out discretely monitored barrier put option prices, absolute errors, and computation times for FZ and FH methods compared with the COS benchmark. FZ and FH use $N_v = 80$ and $N_x = 2^8$; COS uses $N_v = 2^7$ and $N_x = 2^8$.

N	COS	Price		Abs. Error		Time (s)		
		FZ	FH	FZ	FH	FZ	FH	COS
4	2.2790	2.2705	2.2790	0.0085	0.0000	1.4289	3.0196	7.0257
12	1.9750	1.8347	1.9749	0.1403	0.0001	1.7316	9.3599	23.7750
52	1.7122	1.5130	1.7122	0.1993	0.0001	9.6174	24.9856	51.5659
100	1.6389	1.4483	1.6388	0.1906	0.0001	10.8427	45.7705	74.4087
150	1.6033	1.4197	1.6032	0.1836	0.0001	14.8289	67.5516	108.1521
252	1.5663	1.3939	1.5666	0.1724	0.0003	40.5067	99.7157	179.0571

Table 4.3: Down-and-out discretely monitored barrier put option prices, absolute errors, and computation times for FZ and FH methods compared with MC benchmark. MC prices include standard deviation and 95% confidence intervals based on 10^6 simulated paths, using a fixed time step to target 800 steps per simulation, with the final step aligned to the last monitoring date. FZ and FH use $N_v = 80$ and $N_x = 2^8$.

N	MC Price (SD)	95% CI	Abs. Error		Time (s)		
			FZ	FH	FZ	FH	MC
4	2.2763 (0.0045)	[2.2674, 2.2852]	0.0058	0.0027	1.4289	3.0196	650.4998
12	1.9725 (0.0042)	[1.9643, 1.9807]	0.1378	0.0024	1.7316	9.3599	633.0507
52	1.7113 (0.0038)	[1.7038, 1.7189]	0.1984	0.0008	9.6174	24.9856	758.4885
100	1.6366 (0.0037)	[1.6293, 1.6439]	0.1883	0.0022	10.8427	45.7705	641.6834
150	1.6069 (0.0037)	[1.5996, 1.6141]	0.1872	0.0037	14.8289	67.5516	625.7235
252	1.5700 (0.0036)	[1.5628, 1.5771]	0.1761	0.0034	40.5067	99.7157	656.9654

With carefully chosen integration ranges for both the log-price and variance dimensions, we use $N_x = 2^8$ grid points for log-price and $N_v = 80$ for variance, which we find to be sufficient to ensure accurate and stable results across different monitoring dates. Tables 4.2 and 4.3 summarise the pricing accuracy and computational time of the FZ and FH methods compared with the COS and MC benchmarks, respectively. The FH method consistently delivers highly accurate results, closely matching both benchmarks across all monitoring frequencies. The FZ method, while generally less accurate, is significantly faster. This is expected from its use of the z-transform, which collapses the time dimension and avoids repeated computation over monitoring dates. The computational cost of FZ grows primarily due to the fixed-point iterations rather than the number of monitoring dates directly. However, the FZ method requires additional numerical procedures, such as Wiener–Hopf factorisation and decomposition (which involves Hilbert transforms) and inverse z-transforms, which may introduce further sources of numerical error. In contrast, the FH method applies the barrier sequentially in time, which helps maintain numerical stability at the cost of increased runtime. In the next subsection, we will explore various numerical enhancements aimed at numerical error of both methods, with particular focus on mitigating the error sources present in the FZ approach.

4.3.2 Numerical enhancements

In this section, we discuss several techniques used to improve the numerical performance of both the FZ and FH methods. One key source of numerical instability arises when computing the Hilbert transform via FFT, as required in both methods. The numerical computation of the Hilbert transform using FFT can suffer from oscillations and instability due to the Gibbs phenomenon at the sign-function discontinuity and the sensitivity to edge effects introduced by FFT's periodic extension. Spectral filtering has been shown to mitigate these issues by suppressing high-frequency components that contribute to numerical artifacts; see Phelan et al. (2019). We apply a smooth spectral filter in the frequency domain. Specifically, we use the Planck-taper window with a tapering parameter of 0.02, which falls within the reasonable range suggested in Phelan et al. (2019), and which we find to be robust across a range of model parameters and option configurations (to be verified under extreme cases). This choice balances accuracy and filtering strength, effectively suppressing high-frequency noise while preserving the essential structure of the transformed function.

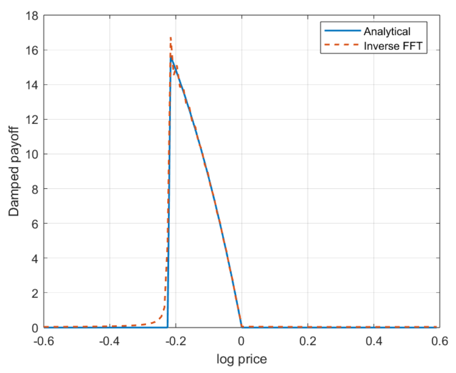
Another important source of numerical error arises from the approximation of numerical integration, particularly in the inverse Fourier transform and in the variance dimension. For the variance integration, the error can be directly reduced by increasing the number of grid points, i.e., increasing N_v. While this improves accuracy, it also increases computational cost, reinforcing the importance of selecting a compact yet sufficient integration domain as discussed earlier.

In contrast, reducing numerical error in the inverse Fourier transform is less straightforward due to the trade-off governed by the Nyquist relation in Eq. (4.43), which links the integration domain in the log-price space (Δx) to that in the Fourier space ($\Delta\xi$). Specifically, decreasing $\Delta\xi$ by increasing N_x also reduces Δx, potentially offsetting the benefit of a finer frequency resolution. To overcome this limitation, we employ the fractional Fourier transform (FrFT), which decouples this relationship and allows us to reduce both Δx and $\Delta\xi$ without increasing the number of grid points. See Section 4.2.1 for more details on its formulation and implementation.

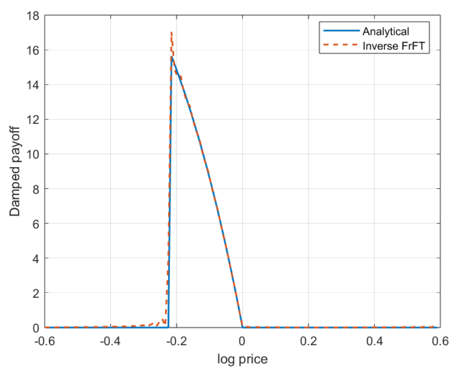
To evaluate the effectiveness of the FrFT, we compare the numerical inverse transforms of the damped payoff function in Eq. (4.19) using both FFT and FrFT, against the analytical expression in Eq. (4.15). For better visibility of the differences, we use a relatively small number of grid points for the log-price domain. The results are presented in Figure 4.1, which includes three panels: the FFT-based reconstruction, the FrFT-based reconstruction, and the absolute error for both methods. As shown in Figures 4.1a and 4.1b, the FFT introduces a noticeable spike near the boundary, which is a typical artifact of spectral leakage and the Gibbs phenomenon. The FrFT slightly improves the reconstruction by avoiding this spike, although it introduces its own mild oscillations. This modest improvement is also reflected in Figure 4.1c, where the absolute error from FrFT is generally lower than that of FFT across the domain.

We also investigate the use of Anderson acceleration (AA) to improve the efficiency of the FZ method. As previously discussed, the computational cost of FZ increases with the number of monitoring dates due to the growing number of fixed-point iterations required for convergence. To address this, we apply AA (see Algorithm 3) to accelerate the convergence of the fixed-point algorithm and evaluate its effectiveness. In the final part of this section, we compare the numerical performance of all method variants under two scenarios. First, we fix the number of monitoring dates at $N = 100$, which we find to be numerically challenging enough to reveal differences in method behaviour, and vary the number of grid points N_x and N_v to assess convergence and accuracy. Second, we fix N_x and N_v at sufficiently high resolution and vary the number of monitoring dates to evaluate how each method scales with monitoring frequency.

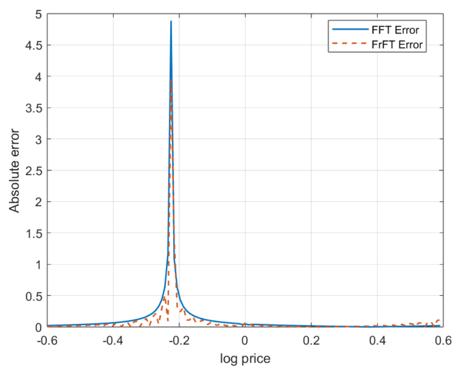
Table 4.4 illustrates how various FZ-based and FH-based method variants converge as the number of grid points in the log-price domain N_x increases. From the



(a) Inverse FFT vs. analytical payoff



(b) Inverse FrFT vs. analytical payoff



(c) Absolute error: FFT and FrFT

Figure 4.1: Comparison of numerical reconstruction of the damped payoff using FFT and FrFT. Subfigures 4.1a and 4.1b show the analytical payoff (solid blue) compared with the inverse FFT and FrFT results (dotted red), respectively. Subfigure 4.1c presents the absolute error for each method. Here, the number of log-price grid points is fixed at $N_x = 2^7$.

results, we observe that the FH-based methods converge rapidly, achieving stable and accurate option prices by $N_x = 2^7$. In contrast, the FZ-based methods require significantly finer grids to reach comparable accuracy. Among them, only those variants with filtering applied show convergence by $N_x = 2^{13}$, while the baseline and unfiltered variants require up to $N_x = 2^{15}$ to produce accurate results. This demonstrates the importance of the spectral filter in stabilizing the Wiener–Hopf-based FZ method. By smoothing the payoff function in the Fourier domain, the filter effectively reduces aliasing and numerical oscillations, enabling faster and more reliable convergence. FrFT also provides moderate improvement, particularly when N_x is relatively small, by enhancing resolution in the Fourier domain. However, Anderson acceleration, which is designed to speed up the convergence of the fixed-point iteration, does

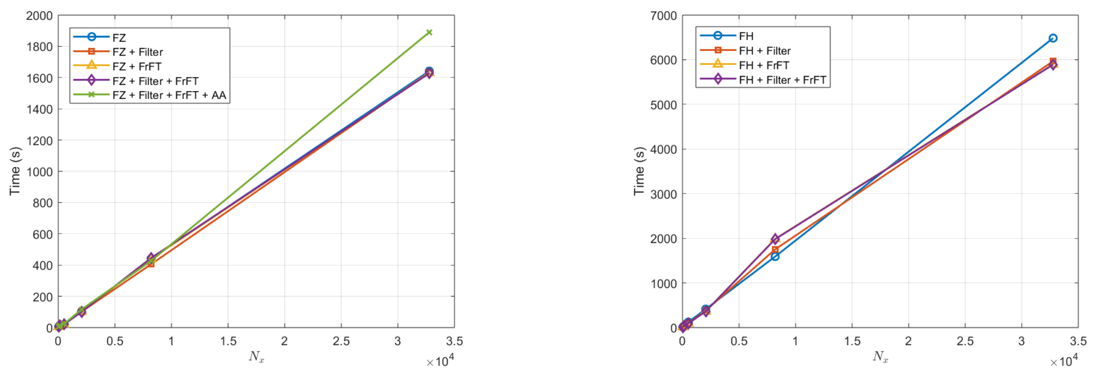
Table 4.4: Down-and-out discretely monitored barrier put option prices computed using different FZ-based and FH-based method variants by varying the number of grid points in the log-price domain N_x. The number of monitoring dates is fixed at $N = 100$, and the variance grid size is fixed at $N_v = 80$. Benchmark prices are COS: 1.6389 and MC: 1.6366 with 95% confidence interval $[1.6293,\ 1.6439]$. Method headers: B = base method, F = apply spectral filter, T = use FrFT instead of FFT, A = apply Anderson acceleration.

N_x	FZ-based methods					FH-based methods			
	B	F	T	F+T	F+T+A	B	F	T	F+T
2^5	-17.2492	52.6636	39.8520	76.7741	76.7741	2.0929	1.8137	1.7786	1.6945
2^7	0.8748	1.3950	1.0063	1.6234	1.6234	1.6388	1.6388	1.6387	1.6387
2^9	1.5909	1.6415	1.5983	1.6390	1.6390	1.6388	1.6388	1.6387	1.6387
2^{11}	1.6357	1.6386	1.6361	1.6386	1.6386	1.6388	1.6388	1.6387	1.6387
2^{13}	1.6385	1.6387	1.6386	1.6387	1.6387	1.6388	1.6388	1.6387	1.6387
2^{15}	1.6387	1.6387	1.6387	1.6387	1.6387	1.6388	1.6388	1.6387	1.6387

not affect the accuracy or convergence behaviour.

Interestingly, in the FH-based methods, we observe two slightly different clusters of converged values, one shared by the base and filtered methods, and the other by the FrFT and FrFT + filter variants. While the differences are numerically small, they are consistent across all grid sizes. This observation suggests that further investigation is needed to understand how FFT and FrFT handle the discontinuity introduced by the barrier. As shown in Figure 4.1c, the FFT-based reconstruction produces a noticeable spike near the barrier, while the FrFT result shows more oscillatory behaviour spread across the domain.

Figure 4.2 compares the computational time of FZ-based and FH-based methods as the number of log-price grid points N_x increases, with $N = 100$ and $N_v = 80$ held fixed. As expected, the FZ-based methods are significantly faster than their FH counterparts for the same value of N_x, due to the time-collapsing nature of the z-transform and the absence of time stepping. However, as shown in Table 4.4, the FH-based methods reach stable and accurate prices much earlier — with convergence observed at $N_x = 2^7$ — while the FZ-based methods typically require much finer grids (e.g., $N_x = 2^{13}$ or 2^{15} depending on the variant) to achieve similar levels of accuracy. This illustrates an trade-off: although FZ methods are computationally cheaper per grid size, they demand more refined grids to stabilise, which can offset their raw speed advantage. FH methods, despite being slower per iteration, offer better convergence behaviour and require fewer grid points overall to produce reliable



(a) FZ-based methods

(b) FH-based methods

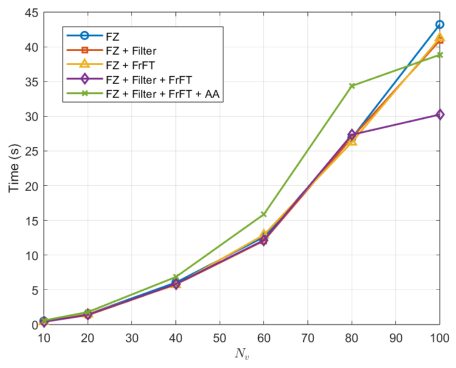
Figure 4.2: Computational time (in seconds) for pricing down-and-out discretely monitored barrier put options using multiple FZ-based and FH-based method variants over increasing numbers of log-price grid points $N_x = [2^5, 2^7, 2^9, 2^{11}, 2^{13}, 2^{15}]$. Subfigures 4.2a and 4.2b show results for each variant, including configurations with spectral filtering, fractional Fourier transforms (FrFT), and Anderson acceleration (AA). The number of monitoring dates is fixed at $N = 100$, and the variance grid size is fixed at $N_v = 80$. Model and contract parameters are provided in Table 4.1.

results, making them more efficient in practice when high accuracy is required.

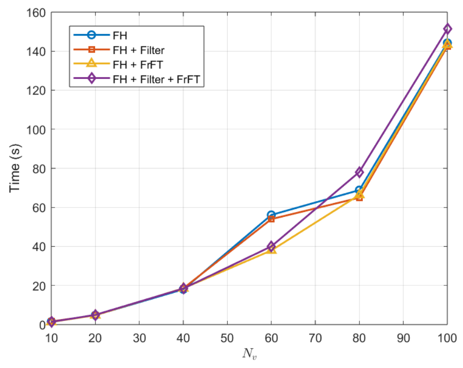
Table 4.5: Down-and-out discretely monitored barrier put option prices computed using various FZ-based and FH-based method variants over increasing numbers of variance grid points N_v. The number of log-price grid points is fixed at $N_x = 2^8$, and the number of monitoring dates is fixed at $N = 100$. Benchmark prices are COS: 1.6389 and MC: 1.6366 with 95% confidence interval $[1.6293,\ 1.6439]$. Method headers: B = base method, F = apply spectral filter, T = use FrFT instead of FFT, A = apply Anderson acceleration.

N_v	FZ-based methods					FH-based methods			
	B	F	T	F+T	F+T+A	B	F	T	F+T
10	-4.7e+4	-4.7e+4	-1.2e+5	-1.2e+5	-1.2e+5	4.2e+27	4.2e+27	4.2e+27	4.2e+27
20	5.6e+3	5.6e+3	5.6e+3	5.6e+3	5.6e+3	5.6e+3	5.6e+3	5.6e+3	5.6e+3
40	1.4784	1.6640	1.5016	1.6683	1.6683	1.6408	1.6408	1.6408	1.6408
60	1.4571	1.6678	1.4826	1.6705	1.6705	1.6388	1.6388	1.6387	1.6387
80	1.4483	1.6705	1.4749	1.6724	1.6724	1.6388	1.6388	1.6387	1.6387
100	1.4432	1.6720	1.4704	1.6735	1.6735	1.6388	1.6388	1.6387	1.6387

Table 4.5 and Figure 4.3 present the option prices and corresponding computational times for various FZ-based and FH-based methods as the number of grid points in the variance dimension N_v increases. The trends observed here are consistent with those discussed earlier for the log-price dimension in Table 4.4 and Figure 4.2. For FH-based methods, accurate and stable option prices are obtained with relatively small values of N_v, around 40 to 60, beyond which all variants converge tightly to the benchmark price. In contrast, FZ-based methods show slower con-



(a) FZ-based methods



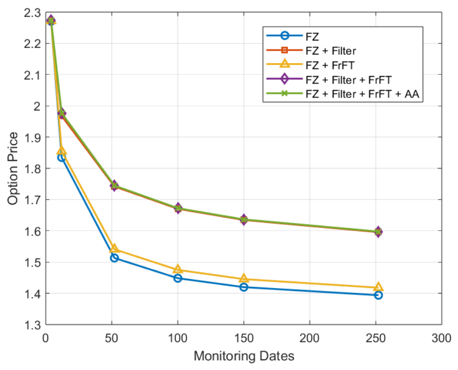
(b) FH-based methods

Figure 4.3: Computational time (in seconds) for pricing down-and-out discretely monitored barrier put options using multiple FZ-based and FH-based method variants over increasing numbers of variance grid points $N_v = [10, 20, 40, 60, 80, 100]$. Subfigures 4.3a and 4.3b show results for each variant, including configurations with spectral filtering, fractional Fourier transforms (FrFT), and Anderson acceleration (AA). The number of monitoring dates is fixed at $N = 100$, and the log-price grid size is fixed at $N_x = 2^8$. Model and contract parameters are provided in Table 4.1.

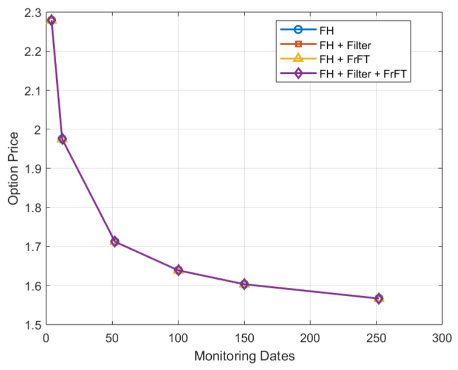
vergence with respect to N_v, and even at $N_v = 100$, some variants—particularly the unfiltered and FrFT-only ones—have not yet reached a stable price. This is consistent with the findings in Table 4.4, where FZ-based methods typically required much finer resolution in the log-price dimension (e.g., $N_x = 2^{13}$ or 2^{15}) to achieve convergence. Thus, the lack of full convergence with respect to N_v may reflect residual numerical errors from insufficient resolution in N_x rather than variance discretisation alone.

As before, the spectral filter proves to be the most impactful enhancement for the FZ method, improving both stability and convergence. FrFT again shows moderate benefit when used without filtering. Anderson acceleration (AA), however, offers no observable improvement in either convergence or computational efficiency. As shown in both Table 4.4 and Table 4.5, as well as the timing plots in Figures 4.2 and 4.3, AA not only fails to reduce runtime but actually increases it. This makes AA practically ineffective in our current pricing framework.

Figure 4.4 presents the down-and-out discretely monitored barrier option prices computed by various FZ-based and FH-based methods across a range of monitoring dates. All methods produce results closely aligned with the benchmark prices shown in Tables 4.2 and 4.3, except for the FZ methods without filtering—specifically, the baseline FZ and the variant using only FrFT. As seen in Figure 4.4a, these un-

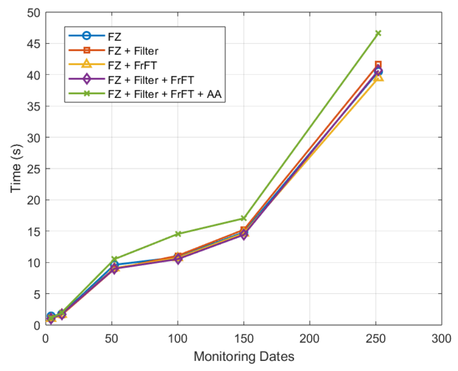


(a) Option prices using FZ-based methods

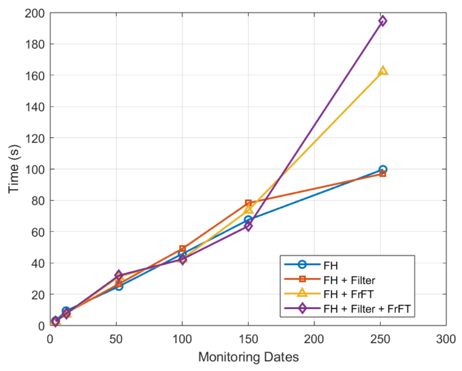


(b) Option prices using FH-based methods

Figure 4.4: Down-and-out discretely monitored barrier put option prices computed using multiple FZ-based and FH-based method variants over increasing monitoring dates $[4, 12, 52, 100, 150, 252]$. Subfigures 4.4a and 4.4b display results for each variant, including versions incorporating spectral filters, fractional Fourier transforms (FrFT), and Anderson acceleration (AA). Model and contract parameters are listed in Table 4.1, with $N_v = 80$ and $N_x = 2^8$.



(a) Computational time for FZ-based methods



(b) Computational time for FH-based methods

Figure 4.5: Computational time (in seconds) for pricing down-and-out discretely monitored barrier put options using multiple FZ-based and FH-based method variants over increasing monitoring dates $N = [4, 12, 52, 100, 150, 252]$. Subfigures 4.5a and 4.5b display results for each variant, including versions incorporating spectral filters, fractional Fourier transforms (FrFT), and Anderson acceleration (AA). Model and contract parameters are listed in Table 4.1, with $N_v = 80$ and $N_x = 2^8$.

filtered FZ methods begin to deviate significantly when the number of monitoring dates exceeds 12. This highlights the critical role of the spectral filter in smoothing the log-price domain in Fourier space before applying the Hilbert transform in the Wiener–Hopf technique. The filter effectively suppresses oscillations that would otherwise distort the pricing kernel, particularly when N is large. While FrFT and Anderson acceleration (AA) are designed to improve frequency resolution and convergence, our results show no noticeable difference in pricing accuracy when these enhancements are applied—both in FZ-based and FH-based methods. This may be due to the fact that the number of log-price grid points N_x is already sufficiently large. Under such conditions, the additional gains from FrFT and AA appear moderate and do not lead to observable improvements. This suggests that beyond a certain resolution threshold, these enhancements may offer limited practical benefit.

For the FH-based methods (Figure 4.4b), no noticeable difference is observed between the variants, indicating that with adequately chosen N_x and N_v, the numerical integration is already stable enough without additional smoothing or FrFT. This may be attributed to the more localised time-stepping structure of the FH method, where errors do not accumulate as globally as in FZ. However, as shown in Figure 4.5b, applying FrFT significantly increases the computational cost for FH, particularly at higher monitoring frequencies.

Lastly, we note that again the FZ method is substantially faster than FH across all test cases. From the runtime data in Figure 4.5, the FZ variants are typically between 3 to 8 times faster than their FH counterparts, depending on the number of monitoring dates and the enhancements applied. This efficiency advantage is especially valuable when pricing needs to be repeated across multiple strikes or calibration iterations.

In this subsection, we introduced two numerical strategies aimed at reducing integration-related errors—spectral filtering and the fractional Fourier transform. As well as Anderson acceleration to improve the convergence rate of the fixed-point algorithm used in the FZ method. We assessed the performance of these enhancements by analyzing the convergence behaviour of option prices with respect to increasing numbers of grid points in both the log-price (N_x) and variance (N_v) dimensions. Across all FZ-based variants, we observed that spectral filtering substantially improves stability and accuracy, particularly by suppressing high-frequency oscillations that arise in the Fourier domain due to discontinuities such as the barrier and truncation. The FrFT method showed moderate improvements, especially at lower grid resolutions, by enabling finer frequency control. In contrast, Anderson acceleration did not provide any meaningful benefit in our setup—it neither reduced computational time nor improved convergence, and in fact slightly increased runtime in all cases tested.

Compared with FZ, the FH method consistently achieved faster convergence with respect to both N_x and N_v, reaching stable prices with fewer grid points. However, this came at the cost of longer computational time per configuration, making FH more expensive when executed repeatedly. We also validated all methods under varying numbers of monitoring dates using fixed, sufficiently large values of N_x and N_v, and found the same performance patterns held.

In summary, if high accuracy is the priority, especially for a single or small batch of valuations, the FH method is recommended due to its superior convergence behaviour. On the other hand, for applications that require repeated pricing across a large number of parameter configurations—such as model calibration or generating training data for machine learning—the FZ method, particularly when combined with spectral filtering, offers a significantly more efficient alternative with acceptable accuracy.

4.3.3 Robustness under challenging Heston parameters

In this subsection, we test the robustness of our proposed methods under challenging Heston model parameter settings, as listed in Table 3.2. These cases are designed to push the limits of numerical stability and convergence, particularly in settings with strong mean reversion, high volatility of volatility, or extreme correlation. As discussed in Section 4.3.2, we apply our FH method using $N_x = 2^8$ without any numerical enhancements, and our FZ-based method using $N_x = 2^{13}$ with spectral filtering applied. For both methods, the variance grid size is set to $N_v = 250$ to ensure sufficient resolution and convergence under the challenging parameter settings.

The choice of $N_v = 250$ is motivated by the variance distributions observed in the challenging cases, as illustrated in Figures 3.12, 3.13, and 3.14. These results show that the variance transition densities are highly left-skewed, particularly under conditions of high volatility of volatility and low mean reversion. To better handle this, we reduced the variance domain from four standard deviations around the mean to 1.5 standard deviations. In addition, following the approach proposed in Fang and

Oosterlee (2011), we perform all computations in the log-variance domain. This transformation reduces skewness and improves quadrature accuracy. The variance-to-log-variance conversion is applied consistently throughout the computation. In particular, all density-related evaluations—such as the transition density and the quadrature over the variance dimension—are performed in the log-variance domain. The range of the log-variance domain is set by taking the logarithm of the original variance bounds, with a minimum value capped at 10^{-50} to avoid negative infinity when computing the logarithm of zero.

While we tested a few representative cases comparing variance vs. log-variance discretisation, and found that log-variance yields more stable and accurate results under challenging parameter regimes, we omit detailed results here for brevity. For benchmarking, we rely solely on Monte Carlo simulation, using 10^6 simulated paths and a time step small enough to target approximately 800 steps per simulation, as described in the accuracy subsection. The COS method from Fang and Oosterlee (2011) is excluded as a benchmark in this subsection, since it fails to converge under the challenging Heston parameter settings considered here. In our test, under challenging Heston parameter settings, using a wide log-price range can lead to numerical overflow during the characteristic function evaluation, resulting in infinities or unstable behaviour. To prevent this, the integration range must be tightened; however, doing so risks truncating significant probability mass—particularly problematic when the underlying distributions are heavy-tailed or highly skewed. This instability was not reported in the original paper, possibly because they only tested a single extreme case with $T = 10$ and 12 monitoring dates—less frequent than the cases we consider here.

The results across the three challenging Heston parameter cases reveal several important observations. Although the FZ and FH methods produce relatively close option prices to each other when the number of monitoring dates N is small, both methods show noticeable errors compared to the MC benchmark even at low monitoring frequencies, with prices often lying outside the MC 95% confidence intervals.

The FZ method maintains consistent convergence behavior across all monitoring frequencies tested, although its prices remain biased compared to MC values.

Table 4.6: Down-and-out discretely monitored barrier put option prices, absolute errors, and computation times for FZ and FH methods compared with MC benchmark under challenging Case 1 in Table 3.2. MC prices include standard deviation and 95% confidence intervals based on 10^6 simulated paths, using a fixed time step to target 800 steps per simulation. FZ uses $N_x = 2^{13}$, FH uses $N_x = 2^8$, and both use $N_v = 250$.

N	MC Price (SD)	95% CI	Price		Abs. Error		Time (s)		
			FZ	FH	FZ	FH	FZ	FH	MC
4	0.3564 (0.0018)	[0.3529, 0.3600]	0.3214	0.3215	0.0350	0.0349	191.48	13.38	682.11
12	0.2538 (0.0015)	[0.2508, 0.2567]	0.2154	0.2154	0.0384	0.0384	412.11	47.33	721.23
52	0.1879 (0.0012)	[0.1855, 0.1904]	0.1225	0.1300	0.0654	0.0579	2197.51	217.70	661.37
100	0.1691 (0.0012)	[0.1668, 0.1714]	0.0838	0.1088	0.0852	0.0603	4178.07	419.04	754.19
150	0.1617 (0.0011)	[0.1594, 0.1639]	0.0605	0.1396	0.1012	0.0221	8588.80	974.78	742.69
252	0.1546 (0.0011)	[0.1524, 0.1567]	0.0261	17.2387	0.1285	17.0841	8892.83	1240.99	825.09

Table 4.7: Down-and-out discretely monitored barrier put option prices, absolute errors, and computation times for FZ and FH methods compared with MC benchmark under challenging Case 2 in Table 3.2. MC prices include standard deviation and 95% confidence intervals based on 10^6 simulated paths, using a fixed time step to target 800 steps per simulation. FZ uses $N_x = 2^{13}$, FH uses $N_x = 2^8$, and both use $N_v = 250$.

N	MC Price (SD)	95% CI	Price		Abs. Error		Time (s)		
			FZ	FH	FZ	FH	FZ	FH	MC
4	0.7243 (0.0025)	[0.7195, 0.7292]	0.5958	0.5958	0.1285	0.1285	188.15	14.24	733.33
12	0.6044 (0.0022)	[0.6001, 0.6088]	0.4139	0.4140	0.1905	0.1904	427.73	54.90	734.65
52	0.4998 (0.0019)	[0.4960, 0.5036]	0.1461	0.1508	0.3537	0.3489	2053.27	251.75	727.91
100	0.4731 (0.0018)	[0.4695, 0.4768]	0.0579	0.0647	0.4152	0.4084	6337.60	494.71	743.67
150	0.4642 (0.0018)	[0.4607, 0.4678]	0.0287	0.0400	0.4356	0.4242	6161.33	969.13	665.91
252	0.4503 (0.0018)	[0.4468, 0.4538]	0.0128	0.2762	0.4375	0.1739	8390.70	1626.36	681.92

In contrast, the FH method, while initially producing stable prices at small N, begins to show clear signs of divergence when N becomes large. For Cases 1 and 3, the FH method fails to maintain the expected decreasing trend in option prices beyond $N = 100$, and for Case 2, this breakdown occurs beyond $N = 150$. This loss of monotonicity suggests that the FH method struggles to converge under high monitoring frequency in extreme parameter regimes. The better stability of the FZ method compared to FH likely stems from the much finer log-price resolution used in FZ ($N_x = 2^{13}$) relative to FH ($N_x = 2^8$).

Both methods exhibit systematic biases relative to MC across all N, indicating that further improvements are needed. Analysis of the density plots in Figures 3.12, 3.13, and 3.14 reveals that the effective support of the log-price distribution becomes very narrow under these challenging parameters. We tested reducing the integration domain from ± 4 to ± 2 standard deviations around the mean to better capture the density mass. The corresponding prices with 252 monitoring dates un-

Table 4.8: Down-and-out discretely monitored barrier put option prices, absolute errors, and computation times for FZ and FH methods compared with MC benchmark under challenging Case 3 in Table 3.2. MC prices include standard deviation and 95% confidence intervals based on 10^6 simulated paths, using a fixed time step to target 800 steps per simulation. FZ uses $N_x = 2^{13}$, FH uses $N_x = 2^8$, and both use $N_v = 250$.

N	MC Price (SD)	95% CI	Price		Abs. Error		Time (s)		
			FZ	FH	FZ	FH	FZ	FH	MC
4	1.3971 (0.0036)	[1.3900, 1.4042]	1.3105	1.3105	0.0866	0.0866	230.06	20.09	547.53
12	1.1544 (0.0032)	[1.1481, 1.1608]	1.0737	1.0738	0.0807	0.0806	518.01	73.75	509.27
52	0.9408 (0.0028)	[0.9352, 0.9464]	0.8806	0.8863	0.0602	0.0545	2546.42	341.50	482.11
100	0.8825 (0.0027)	[0.8771, 0.8878]	0.8279	0.8510	0.0546	0.0315	5482.52	661.72	537.65
150	0.8598 (0.0027)	[0.8545, 0.8651]	0.8026	0.8668	0.0572	0.0070	7479.63	1089.52	502.97
252	0.8339 (0.0026)	[0.8288, 0.8391]	0.7304	2.2906	0.1035	1.4567	9273.00	1781.15	542.04

der three challenging parameter sets are summarised in Table 3.2. Narrowing the log-price domain significantly improves FH convergence by focusing on the relevant region of the distribution. In contrast, the FZ method is less affected by domain adjustment, as it already employs a high number of grid points to ensure resolution. Nevertheless, even after domain refinement, both FH and FZ results show deviations from benchmark Monte Carlo values. This suggests that further adjustment of the variance integration domain may be necessary, particularly ensuring that the left boundary is sufficiently small to accurately capture the heavy left tail of the distribution with good resolution.

Table 4.9: Down-and-out discretely monitored barrier put option prices for challenging cases in Table 3.2. Calculations are performed with a 2-standard-deviation log-price grid span and 252 monitoring dates. FZ uses $N_x = 2^{13}$, FH uses $N_x = 2^8$, and both use $N_v = 250$.

Case	**FH price**	**FZ price**
Case 1	0.04926971	0.02601989
Case 2	0.01206138	0.01125164
Case 3	0.79256056	0.72735784

The challenging cases tested indeed pose significant difficulties for Fourier-based pricing methods. Although using finer grid resolution and narrowing both the log-price and variance domains improves convergence — and all methods eventually converge under these settings — noticeable pricing gaps compared to Monte Carlo benchmarks persist. Furthermore, the computational time required to achieve convergence with finer grids becomes substantial, causing the FZ method to lose much of its initial speed advantage over the FH method. Finally, the observed discrepancies suggest that further refinement of the variance integration domain may still be necessary, particularly ensuring that the left boundary is sufficiently small to capture the heavy left tail of the underlying distribution with adequate resolution.

4.4 Conclusion

In this chapter, we developed and evaluated two Fourier-based numerical methods for pricing discretely monitored down-and-out barrier options under the Heston stochastic volatility model. The first method, referred to as FZ, is an extension of the Wiener–Hopf technique using the z-transform to collapse the time dimension, and the second method, FH, is a recursive Hilbert transform-based approach that sequentially applies the barrier condition at each monitoring date. Both methods are constructed to work within the characteristic function framework and leverage the tractability of the Heston model.

To improve numerical stability and accuracy, we incorporated spectral filtering and the fractional Fourier transform into both methods. We also tested the impact of Anderson acceleration for enhancing the fixed-point iteration in the FZ scheme. Through a series of numerical experiments, we evaluated the convergence behaviour of each method by varying the number of grid points in both the log-price and variance dimensions. The results showed that FH converges more rapidly with fewer grid points, while FZ benefits significantly from spectral filtering but requires finer grids to reach stability. FrFT provided moderate improvements at lower resolutions, whereas AA showed no significant benefit in either convergence or computational time in our setting.

We further tested the robustness of both methods under challenging Heston parameter regimes, characterised by high volatility of volatility, strong correlations, and slow mean reversion. Under these extreme conditions, the COS method from Fang and Oosterlee (2011) failed to converge, primarily due to characteristic function instability and high sensitivity to domain truncation. Narrowing the integration domains and increasing grid resolution enabled both the FZ and FH methods to converge; however, noticeable pricing gaps compared to Monte Carlo benchmarks persisted.

Furthermore, the computational burden associated with achieving convergence became substantial, particularly for the FZ method, eroding its initial computational advantage over FH. These findings suggest that further refinement of the variance integration domain, especially careful adjustment of the left boundary to capture the heavy left tail, may be necessary to improve accuracy in extreme parameter regimes.

In summary, our results indicate that the FH method is well-suited for applications requiring high accuracy at moderate monitoring frequencies and reasonable computational budgets. The FZ method—especially when combined with spectral filtering—offers a practical alternative for large-scale pricing tasks, such as model calibration or data generation for machine learning, under standard parameter settings. Nevertheless, under highly volatile conditions with very high monitoring frequencies, FZ gradually loses its computational time advantage due to the need for finer grids and more fixed-point iterations. While both Fourier-based methods demonstrate strong robustness under standard settings, achieving MC-level accuracy efficiently under extreme market conditions will likely require further refinements, particularly through adaptive domain optimisation strategies.

Chapter 5

Deep learning to price the discretely monitored barrier option

Machine learning, especially artificial neural networks (ANNs), has shown growing promise in addressing the complexity of option pricing. When provided with sufficiently large training datasets, neural networks can learn the mapping from contract terms and model parameters to option prices, without the need for explicit pricing formulas. In practice, one can treat contract terms as input and the corresponding market prices as output, and let the neural network learn this mapping through training. This idea shifts the burden from analytical tractability to data-driven learning. Neural networks aim to minimize a loss function—typically the mean squared error between predicted and actual prices—to produce robust out-of-sample performance. Notably, certain ANN architectures are universal approximators, meaning they can approximate any function to arbitrary precision given sufficient complexity and training data, as shown by Cybenko (1989) and Hornik et al. (1989).

Although neural networks can operate as black-box predictors, incorporating financial knowledge into the feature design or training framework often improves their performance. While training is computationally demanding, the pricing process after training is extremely fast—offering an efficient alternative to traditional numerical pricing tools. This has motivated a growing body of research exploring machine learning techniques for option pricing across various financial models. The use of machine learning in option pricing dates back to the early 1990s. Malliaris and Salchenberger (1993) and Hutchinson et al. (1996) applied neural networks to price S&P100 and

S&P500 index options. With the development of modern software tools, this approach has become more accessible. Liu et al. (2019) trained optimized ANNs using synthetic data from classical pricing models, such as Black-Scholes and Heston, and deployed them as fast pricing agents. McGhee (2018) and Horvath et al. (2019) extended this framework to SABR and rough Bergomi models, respectively. McGhee (2018) also explored ANN pricing under local stochastic volatility models.

This chapter focuses on applying ANNs to price the same discretely monitored down-and-out put barrier options previously studied in Chapter 4. Specifically, we use the Fourier–Z-transform (FZ) method, in combination with the Heston stochastic volatility model, to generate the training data used for supervised learning. The Heston model (Heston, 1993), a widely used extension of Black-Scholes, captures volatility smiles by modelling stochastic variance. While more accurate than Black-Scholes, it remains computationally intensive and lacks flexibility to incorporate jump features. By using the FZ method to generate prices under the Heston model and training ANNs on this data, we aim to build a fast and reliable approximation model. This allows us to retain the modelling richness of Heston while avoiding the runtime complexity of repeated simulations.

In the remainder of this chapter, we first review the background of artificial neural networks and introduce the training methodology. We then describe how the network is constructed and optimized for pricing discretely monitored down-and-out put barrier options. Following this, we explore different neuron configurations across hidden layers, test the generalisability of the model on alternative datasets involving different option types and underlying asset models, and evaluate performance across varying training set sizes to understand the model's convergence behaviour and generalisation capacity.

5.1 Artificial neural networks for option pricing

Artificial Neural Networks (ANNs) are flexible, data-driven models capable of capturing complex nonlinear relationships between inputs and outputs. As shown in Figure 5.1, a typical ANN consists of three main components, an input layer, one or more hidden layers, and an output layer. Each layer contains a set of neurons connected through weighted links. These weights are the model's parameters and are updated during training to improve prediction accuracy.

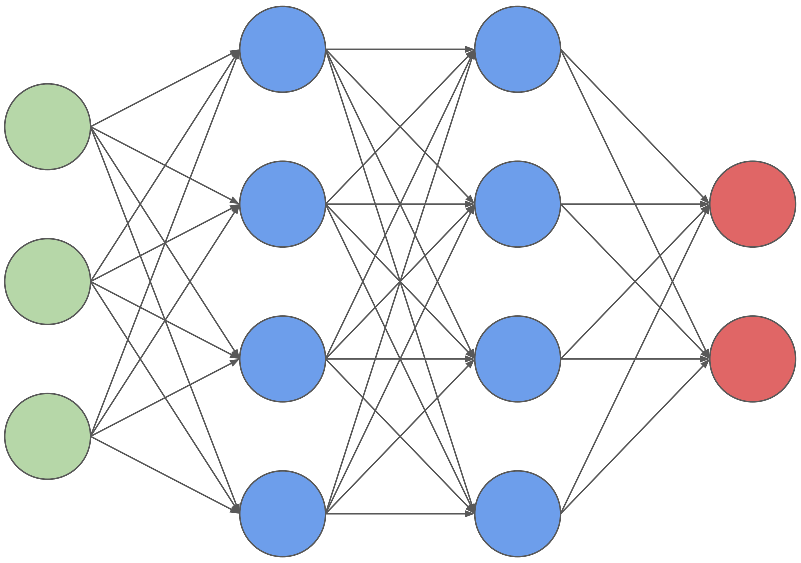


Figure 5.1: An example artificial neural network. The green dots represent input layer, blue dots forms two hidden layers and red dots represent output layer.

Each neuron (Figure 5.2) computes a weighted sum of its inputs, applies a non-linear activation function, and passes the result to the next layer. By stacking multiple layers, ANNs can learn high-level feature representations from raw input data. The process of training involves two key steps, *forward propagation*, where inputs pass through the network to produce an output, and *backward propagation*, where errors are propagated backward to adjust the weights using gradient descent or its variants. This iterative process continues until the model converges to a set of weights that minimizes a chosen loss function—typically mean squared error for regression tasks like option pricing.

To avoid overfitting and select an optimal model architecture, we employ cross-validation. As illustrated in Figure 5.3, K-fold cross-validation involves splitting the training data into K subsets, training the model on $K-1$ parts and validating it on the remaining part in a rotating fashion. This helps ensure the model generalizes well to unseen data.

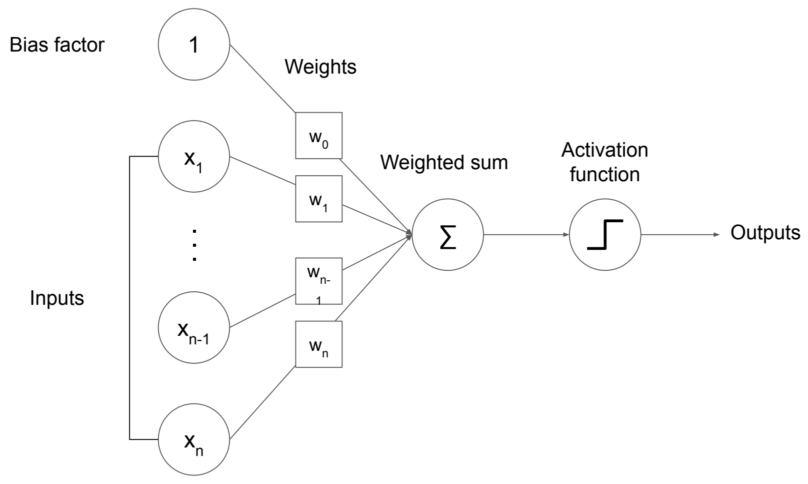


Figure 5.2: Neuron, a basic unit of the neural network, accepts an input and generates a prediction.

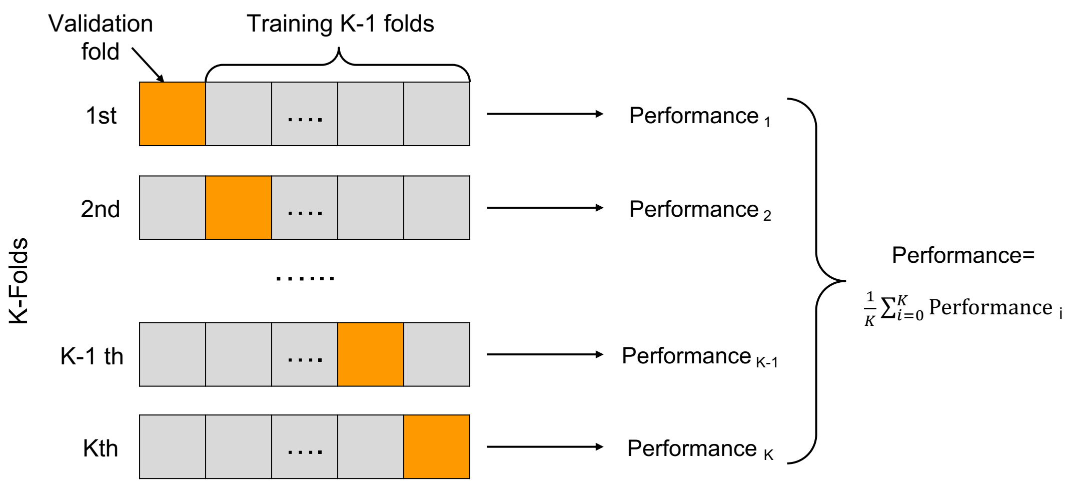


Figure 5.3: K-fold cross-validation.

5.1.1 Training and prediction workflow

In this work, ANNs are used to price discretely monitored barrier options. The core idea is to replace computationally expensive numerical pricing methods with a fast and accurate ANN trained on synthetic data generated using the Fourier–Z-transform (FZ) method under the Heston stochastic volatility model.

The modelling process consists of the following stages,

1. **Data Generation:** Generate a large dataset of input features (e.g., Heston parameters, barrier level, strike, maturity) and corresponding option prices using

the FZ method.

2. **Data Splitting:** Divide the dataset into training and validation subsets.

3. **Hyperparameter Tuning:** Use K-fold cross-validation (Figure 5.3) to select the optimal network architecture and training parameters, including number of layers, neurons, learning rate, activation function, batch size, and optimizer.

4. **Model Training:** Train the ANN using the selected hyperparameters and minimize the prediction error over the training set.

5. **Validation and Evaluation:** Evaluate the trained model on the validation set to ensure generalisation and detect any overfitting.

Once trained, the ANN provides immediate price predictions for new input data without the need for numerical simulations. This results in a powerful pricing tool that combines the flexibility of machine learning with the modelling richness of the Heston framework.

5.2 Data set generation

Following the pricing tasks described in the previous chapter, we continue to focus on pricing discretely monitored down-and-out barrier put options under the Heston stochastic volatility model. The parameters generated for each option include the initial stock price (S_0), strike price (K), lower barrier level (L), risk-free interest rate (r), Heston model parameters (κ, θ, σ, ρ, v_0), and the number of monitoring dates (M), all sampled uniformly from the ranges specified in Table 5.1. Since the relative ratios S_0/K and L/K are the quantities that primarily determine the option's behaviour, the input feature matrix for the neural network is constructed as $[M, S_0/K, L/K, r, \kappa, \theta, \sigma, \rho, v_0]$.

The ranges are selected carefully to ensure both practical relevance and modelling challenges. The initial stock price S_0 and strike price K are generated from the same interval [90, 130]. Since the contracts are down-and-out barrier options, the barrier level L is generated dynamically in the range $[0.7S_0, 0.95S_0]$ for each sampled S_0. The ranges for the Heston parameters are chosen to cover challenging cases previously identified (Table 3.2), as well as to align with settings used in Liu et al. (2019).

Option prices corresponding to each parameter set are computed using the Fourier-Z transform (FZ) method, employing $N_x = 2^8$ log-price grid points and $N_v = 80$ variance grid points. Although the FZ method sacrifices a small amount of pricing accuracy compared to methods like Fourier-Heston (FH), it offers a significant gain in computational speed, which is essential for generating large datasets. Alternative methods such as the COS method or Monte Carlo simulations are either unreliable under extreme Heston parameter configurations or prohibitively slow.

The number of monitoring dates M is randomly selected between 2 and 100. This range is chosen to balance computational feasibility and pricing accuracy, based on observations from previous chapters that a combination of large M and extreme parameter settings can compromise numerical stability.

In total, we generate 120,000 samples, each consisting of uniformly sampled parameters from the specified ranges and corresponding option prices computed via the FZ method. Having generated the dataset, the next step is to design and optimize the ANN model via hyperparameter tuning.

Table 5.1: Parameter ranges used for data generation under the Heston model for down-and-out barrier option.

Parameter	**Range**
Initial stock price S_0	$[90, 130]$
Lower barrier L	$[0.7S_0, 0.95S_0]$
Strike price K	$[90, 130]$
Risk-free rate r	$[0.0, 0.10]$
Mean reversion rate κ	$[0.3, 2]$
Long-term variance θ	$[0.01, 0.95]$
Volatility of variance σ	$[0.01, 1]$
Correlation ρ	$[-0.9, 0.1]$
Initial variance v_0	$[0.01, 0.95]$
Number of monitoring dates M	$\{2, \ldots, 100\}$

5.3 Hyperparameters optimisation

In neural networks, parameters can be broadly classified into two categories: model parameters and hyperparameters. Model parameters, such as neuron weights and biases, are internal to the model and are adjusted automatically during the training process. In contrast, hyperparameters are external settings defined before training begins, such as the number of hidden layers, the type of activation functions, the batch size, and learning rates. Tuning hyperparameters is essentially an optimisation problem, where the goal is to find the combination of values that results in the best model performance.

Hyperparameter tuning is known to be a challenging task (Claesen and De Moor, 2015), largely because optimal hyperparameter values can vary significantly across different problems and datasets. As a result, there is no universal set of hyperparameters that guarantees good performance across all tasks. This difficulty has led practitioners to often describe hyperparameter tuning as “more of an art than a science.” In the following, we introduce the specific hyperparameters considered in this study and explain how we define reasonable search spaces for each. For some hyperparameters, we adopt standard values recommended in the literature as starting points during random search. If the resulting model performance is not satisfactory, we expand the search space or adjust these values accordingly. Our hyperparameter selection strategy is informed by previous research in applying ANNs to option pricing problems (Horvath et al., 2019; Karataş et al., 2019; Liu et al., 2019; McGhee, 2018).

We begin by considering the *number of hidden layers* and the *number of neurons* per layer, as these are fundamental architectural choices when designing a neural network. Although there is theoretical evidence that deeper networks can be more powerful than shallower ones (Eldan and Shamir, 2016), empirical studies suggest that increasing depth beyond a certain point does not always lead to improved performance. In particular, Bayer et al. (2019) found that adding more than four hidden layers did not consistently reduce validation errors. Several studies on ANN-based option pricing have found four hidden layers to work well: Liu et al. (2019) used 400 neurons per layer, Karataş et al. (2019) used 120 neurons per layer, and Horvath et al. (2019) used 30 neurons per layer. In contrast, McGhee (2018) employed a single hidden layer with a relatively large number of neurons.

In our initial experiments, we tested architectures with both four and five hidden layers. While adding a fifth layer slightly increased the model complexity, it did not lead to significant improvements in predictive performance compared to the four-layer architecture. Considering the additional computational cost associated with deeper networks and the risk of overfitting with limited training data, we chose to adopt four hidden layers for the final model configuration. Regarding the number of neurons per layer, there is no universally accepted guidance in the literature. Therefore, we treat the number of neurons as a hyperparameter and consider candidate values of 100, 200, 300, 500, 700, and 900.

Following the network architecture design, we next discuss the choice of *optimiser* and *activation function*, which are closely linked in neural network training. The activation function determines the nonlinearity of the neuron outputs, and its properties directly influence the gradient flow during backpropagation. Consequently, the optimiser, which updates the model weights based on these gradients, must interact effectively with the activation function to ensure stable and efficient training.

Popular optimisers for neural networks include stochastic gradient descent (SGD), RMSProp, and Adam, all of which are gradient-based optimisation methods. Common activation functions include the softplus activation, the rectified linear unit (ReLU), and its variants such as leaky ReLU and the exponential linear unit (ELU). ReLU and its variants have become widely used in deep learning applications due to their simplicity and ability to mitigate the vanishing gradient problem. There is extensive literature discussing the performance of different optimiser-activation function combinations. Kingma and Ba (2014) demonstrated that Adam often achieves better convergence compared to SGD and RMSProp when paired with ReLU-based activations. In addition, Kawaguchi et al. (2018); Li and Yuan (2017) analysed the convergence behaviour of networks trained with SGD and ReLU activations. In the context of option pricing, Karataş et al. (2019); Liu et al. (2019); McGhee (2018) employed Adam as the optimiser, using ReLU, leaky ReLU, and softplus activations respectively, while Horvath et al. (2019) used gradient-based optimisers together with ELU activation functions. In this work, we test different combinations of SGD, RMSProp, and Adam optimisers with softplus, ReLU, leaky ReLU, and ELU activation functions to determine the best-performing configurations for our pricing problem.

Weight initialisation is another key hyperparameter that must be set before training. Unlike in linear or logistic regression, weights in an ANN cannot be initialised to zero or identical values across neurons in the same layer. This is because identical initialisation would cause neurons to produce identical outputs and gradients, effectively making them redundant during training. Standard practice, based on Glorot and Bengio (2010); Goodfellow et al. (2016); He et al. (2015), is to initialise weights randomly, using either Gaussian or uniform distributions, while biases are typically set to zero. The initial scale of the weights is important: overly large weights can lead to large activations, causing small gradients for saturating activation functions (e.g., sigmoid, tanh) and oscillations for non-saturating ones (e.g., ReLU).

Different activation functions motivate different initialisation schemes. For sigmoid or tanh activations, the "Xavier" or "Glorot" initialisation (Glorot and Bengio, 2010) is commonly used. For ReLU and its variants, He et al. (2015) proposed the "He" initialisation, which adjusts for the asymmetry of rectified activations e.g. ReLU, ELU. In the He initialisation scheme, weights are drawn independently either from a normal distribution $N(0, 2/n_{\text{in}})$ or from a uniform distribution $U\left[-\sqrt{6/n_{\text{in}}}, \sqrt{6/n_{\text{in}}}\right]$, where n_{in} denotes the number of input neurons. Following the approach in Liu et al. (2019), we adopt the "He-uniform" initialisation in our hyperparameter tuning, as they tested and recommended uniform-based variants for option pricing models.

One of other key hyperparameter in training deep neural networks is the *batch size*, which determines the number of samples passed through the network during each forward and backward propagation cycle. After each batch, the internal parameters, such as the network weights, are updated based on the computed gradients. The choice of batch size can significantly influence both training stability and final model performance. Liu et al. (2019) employed a relatively large batch size of 1024 in their pricing model, whereas Goodfellow et al. (2016) suggested that mini-batch sizes are typically chosen between 10 and a few hundred examples, depending on the application and hardware constraints. Similarly, Horvath et al. (2019) used a batch size of 32 in their deep learning framework for option pricing. Empirically, we found that batch sizes below 100 generally led to better model performance in our setting. Furthermore, as discussed in Hoffer et al. (2017); Smith et al. (2017), batch size and optimiser choice are interdependent, and different combinations can yield different convergence behaviours. Therefore, in our hyperparameter search, we jointly tune batch size together with optimiser choice. Based on both literature and experimental results, we select batch sizes from the range $\{2^3, 2^4, 2^5, 50, 100\}$ for testing.

The *learning rate* is also a key hyperparameter in training neural networks. It controls the size of the steps taken during optimisation when updating model parameters based on the computed gradients. If the learning rate is set too large, the optimisation process may oscillate around minima and fail to converge, or even diverge entirely. Conversely, if the learning rate is too small, convergence can become extremely slow, leading to extended training times and potentially getting stuck in suboptimal local minima. Common practice is to select the learning rate from a relatively small range, typically between 10^{-2} and 10^{-5}, depending on the model architecture, data complexity, and optimiser used. Following the approach in Liu et al. (2019), we test learning rates from the set $\{10^{-2}, 10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$ as candidates in our hyperparameter tuning process.

We also specify the *number of epochs*, which defines how many times the learning algorithm passes through the entire training dataset. After each epoch, every training sample has contributed once to updating the network's internal parameters. For hyperparameter tuning, we use a relatively small number of epochs (50) to reduce computational cost, whereas during final training we adopt a larger number (200 epochs), following the setups in Liu et al. (2019) and Horvath et al. (2019). Additionally, we apply early stopping, terminating the training if the validation loss does not improve after 25 consecutive evaluations, to prevent unnecessary computations.

Two additional hyperparameters, *batch normalisation* and *dropout rate*, are also considered based on findings from the literature. Batch normalisation (Ioffe and Szegedy, 2015) rescales the inputs to each layer by subtracting the mini-batch mean and dividing by the mini-batch standard deviation, aiming to accelerate and stabilise training. However, based on the results reported by Liu et al. (2019) in the context of option pricing, we disable batch normalisation in our model. Similarly, dropout (Srivastava et al., 2014) randomly deactivates a proportion of neurons during training to prevent overfitting. Since Liu et al. (2019) observed that the optimal dropout rate was effectively zero in their experiments, and given that overfitting was not initially detected in our tests, we set the dropout rate to zero as well.

Based on the discussion and initial tests above, we summarise our hyperparameter choices as follows. Certain hyperparameters are directly set based on previous literature or empirical observations, and these settings are listed in Table 5.2. For the remaining hyperparameters, we conduct more rigorous tests by randomly searching over a specified range of values, as detailed in Table 5.3.

Table 5.2: Fixed hyperparameters set directly based on literature or preliminary tests.

Parameter	Setting
Number of hidden layers	4
Number of epochs (for training)	200
Weight initialisation	He-uniform
Dropout rate	0
Batch normalisation	Off

Table 5.3: Hyperparameters and corresponding test ranges used for random search during model tuning. Fixed hyperparameters are listed in Table 5.2. A smaller number of epochs (50) is used during hyperparameter search.

Parameter	Test Range
Number of neurons	100, 200, 300, 500, 700, 900
Activation function	softplus, ReLU, leaky ReLU, ELU
Optimiser	SGD, Adam, RMSprop
Batch size	2^3, 2^4, 2^5, 2^6, 2^7
Learning rate	10^{-2}, 10^{-3}, 5×10^{-4}, 10^{-4}

5.3.1 Hyperparameter search and validation

We employ random search, specifically the `RandomizedSearchCV` module from Scikit-learn (Pedregosa et al., 2011), to optimise hyperparameters. Random search has the advantage of exploring a wider range of hyperparameter combinations more efficiently than grid search, especially when the number of hyperparameters is large or the search spaces are broad. In such cases, grid search becomes computationally expensive, while random search can still sample diverse configurations effectively.

Specifically, we randomly select 150 hyperparameter combinations from the ranges specified in Table 5.3, with fixed settings given in Table 5.2. These combinations are ranked based on their mean-squared error (MSE) performance. MSE is chosen as the evaluation metric because it directly measures absolute prediction errors, which are critical in pricing tasks. To further mitigate overfitting during hyperparameter search, we apply 2-fold cross-validation. Using a small number of folds provides a good balance between time efficiency and overfitting control at the initial stage of hyperparameter tuning. The overall procedure for hyperparameter tuning is summarised in Algorithm 4. Here, X_t denotes the input features used during hyperparameter search, and y_t denotes the corresponding target outputs.

Algorithm 4 Hyperparameter Optimisation with K-fold Cross-Validation

```
1: procedure HyperparameterOptimisation($X_t$, $y_t$)
2:     Split the training samples into $K$ disjoint subsets
3:     for each hyperparameter combination do
4:         for each selection of $K-1$ training subsets do
5:             for each epoch do
6:                 for each mini-batch do
7:                     Update model parameters using the current batch
8:                 end for
9:             end for
10:            Evaluate the model on the remaining validation subset
11:        end for
12:        Average the validation metrics over the $K$ folds
13:    end for
14:    Compare and rank all hyperparameter combinations based on the averaged
       metrics
15:    return the best hyperparameter combination
16: end procedure
```

Following the approach in Liu et al. (2019), we use a subset of the available data for hyperparameter tuning and reserve the full dataset for final model training and prediction. Specifically, we randomly select 20,000 samples out of the full 120,000 for hyperparameter search. After completing the random search, we select the top five hyperparameter combinations to mitigate the risk of choosing a suboptimal local minimum. The final model configuration is obtained by averaging the corresponding hyperparameter values across these top combinations.

The final selected hyperparameters, based on the search ranges in Table 5.3 and the fixed settings in Table 5.2, are summarised in Table 5.4. To further vali-

Table 5.4: Final ANN model configuration for barrier option pricing.

Parameter	**Specification**
Number of hidden layers	4
Number of neurons per layer	380
Activation function	softplus
Optimiser	Adam
Batch size	20
Learning rate	0.001
Weight initialisation	He-uniform
Dropout rate	0
Batch normalisation	Off
Number of epochs (final training)	200

date the selected hyperparameters, we train the model using 80,000 samples and evaluate its performance on a separate, non-overlapping test set of 20,000 samples. The training is conducted over 200 epochs. The resulting performance metrics are summarised in Table 5.5.

Table 5.5: Evaluation of the final ANN model predictions on the training and test sets using mean-squared error and R^2 score. The model is trained (80,000 samples) using the configuration summarised in Table 5.4 and tested on separate 20,000 samples.

	Mean-squared error	R^2 score
Training set	0.02126451	0.98593515
Test set	0.09788835	0.93795592

The final ANN model demonstrates strong predictive performance. The training R^2 score of 0.9859 and test R^2 score of 0.9380 indicate that the model captures the underlying pricing dynamics effectively without significant overfitting. The relatively low mean-squared errors on both training and testing sets further confirm the model's generalisation ability. Overall, the selected hyperparameters yield a stable and accurate model for pricing discretely monitored barrier options.

5.3.2 Hyperparameter validation using an alternative dataset

To further assess the robustness of our hyperparameter selection, we apply the model configuration summarised in Table 5.4 to an alternative dataset consisting of double barrier call option prices generated under the NIG model. This dataset, provided by a colleague, serves solely to illustrate the sensitivity of hyperparameter selection across different option types and stochastic models; our main focus remains on the Heston-based down-and-out barrier options discussed earlier. The model is trained with the NIG-based data over 200 epochs, using the previously selected hyperparameters. We train the model using 80,000 samples and evaluate its performance on a separate, non-overlapping test set of 20,000 samples. The evaluation results on the training and test sets are summarised in Table 5.6.

Table 5.6: Evaluation of the ANN model trained with NIG-based double barrier option prices on the training and test sets using mean-squared error and R^2 score. The model is trained (80,000 samples) using the configuration summarised in Table 5.4 and tested on separate 20,000 samples.

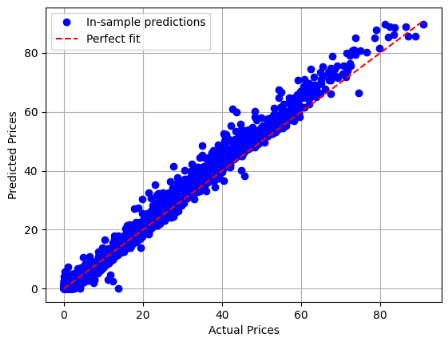
	Mean-squared error	R^2 score
Training set	0.37484177	0.99155681
Test set	0.42165059	0.99041205

The final model also performs very well on the NIG-based double barrier option dataset. As shown in Table 5.6, the model achieves a training R^2 score of 0.9916 and a test R^2 score of 0.9904, with corresponding mean squared errors of 0.3748 and 0.4217, respectively. Figure 5.4 shows the predicted versus actual price plots for both the in-sample and out-of-sample sets, demonstrating good alignment along the perfect prediction line and further confirming the model's predictive accuracy.

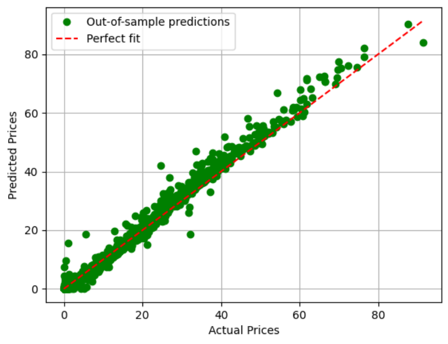
Compared with the Heston dataset (Table 5.5), the NIG dataset yields higher R^2 scores, primarily due to its significantly larger price range (NIG price mean: 2.27, standard deviation: 6.66) compared to the Heston data (mean: 0.60, standard deviation: 1.24). The larger spread in the NIG prices results in a greater total variance, which naturally leads to higher R^2 values even though the absolute prediction errors are somewhat larger. These results confirm that the hyperparameters tuned on the Heston dataset generalise well to alternative option types and stochastic models.

5.3.3 Exploration of neuron configurations across hidden layers

As discussed earlier, previous studies such as Horvath et al. (2019); Karataş et al. (2019); Liu et al. (2019) typically use the same number of neurons for each hidden layer. In this section, we explore varying the number of neurons across different hidden layers, and evaluate the impact on model performance using two distinct



(a) Training (in-sample) set



(b) Test (out-of-sample) set

Figure 5.4: Predicted versus actual price plots for both the in-sample (80,000 samples) and out-of-sample (20,000 samples) sets for the NIG-based double barrier option dataset. Predictions are generated using the final ANN model configuration summarised in Table 5.4.

datasets: the Heston-based single barrier option dataset and the NIG-based double barrier option dataset.

To conduct this investigation, we perform a new hyperparameter search using 20,000 training samples for each dataset, exploring the configurations summarised in Table 5.7. Given that our previous experiments indicated that default learning rates for each optimiser already worked well, we omit learning rate tuning in this analysis.

Table 5.7: Hyperparameters and corresponding test ranges used for random search during model tuning, allowing for different numbers of neurons across hidden layers. Fixed hyperparameters are listed in Table 5.2 with default learning rate for different optimiser. A smaller number of epochs (50) is used during hyperparameter search to improve computational efficiency.

Parameter	**Test Range**
Number of neurons in 1st hidden layer	20, 50, 100, 400, 800
Number of neurons in 2nd hidden layer	20, 50, 100, 400, 800
Number of neurons in 3rd hidden layer	10, 50, 100, 200, 400
Number of neurons in 4th hidden layer	10, 50, 100, 200, 400
Activation function	softplus, ReLU, leaky ReLU, ELU
Optimiser	SGD, Adam, RMSprop
Batch size	2^3, 2^4, 2^5, 2^7, 2^8

The final model configurations summarised in Table 5.8 are obtained by randomly evaluating 1,500 hyperparameter combinations and averaging the top five configurations with the lowest validation MSE. While both datasets select the same

Table 5.8: Best hyperparameter configurations for the Heston-based and NIG-based datasets, obtained via random search using the ranges specified in Table 5.7.

Parameter	**Specification**	
	Heston-based	NIG-based
Number of neurons in 1st hidden layer	210	210
Number of neurons in 2nd hidden layer	484	194
Number of neurons in 3rd hidden layer	250	162
Number of neurons in 4th hidden layer	104	112
Activation function	softplus	softplus
Optimiser	RMSprop	Adam
Batch size	22	8

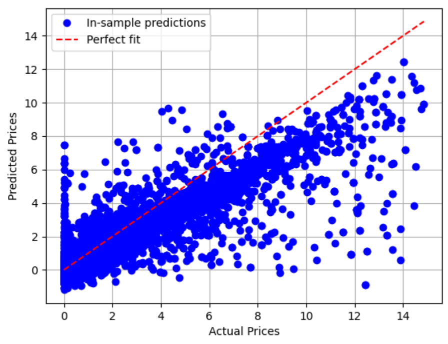
activation function (softplus) and utilise mini-batch training. Moreover, the NIG-based dataset selected Adam as the optimiser, whereas the Heston-based dataset selected RMSprop, although Adam appeared in two of the top five configurations for the Heston-based search as well.

These observations confirm that finer tuning of the number of neurons across hidden layers leads to different optimal configurations for different datasets, which is expected given the differences in option type, underlying asset model, and data distribution. We assess the performance of the final configurations by training each model on 80,000 samples and evaluating it on a separate, non-overlapping test set of 20,000 samples. The resulting MSE and R^2 scores are reported in Table 5.9, and the predicted versus actual price plots are presented in Figure 5.5, for both the Heston-based and NIG-based datasets.

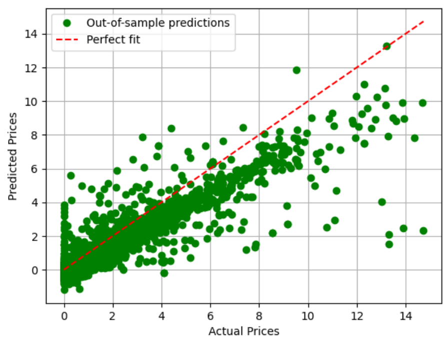
Table 5.9: Performance of the final models with different neuron configurations across hidden layers, as summarised in Table 5.8, for the Heston-based and NIG-based datasets. The models are trained with 80,000 samples and evaluated on a separate test set of 20,000 samples.

	Mean-squared error		**R^2 score**	
	Heston	NIG	Heston	NIG
Training set	0.25251697	0.15506260	0.83321162	0.99650727
Test set	0.23375391	0.19683336	0.84205809	0.99552419

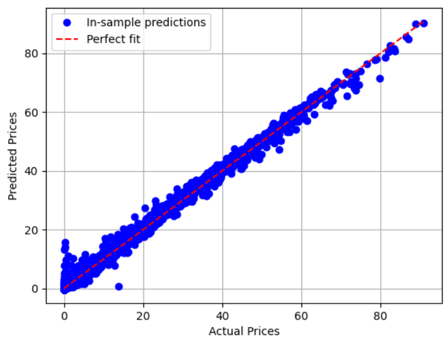
The results summarised in Table 5.9 and Figure 5.5 indicate that the models with varying numbers of neurons across hidden layers perform very well on the NIG-based dataset. Compared to previous results, we observe a slight reduction in MSE



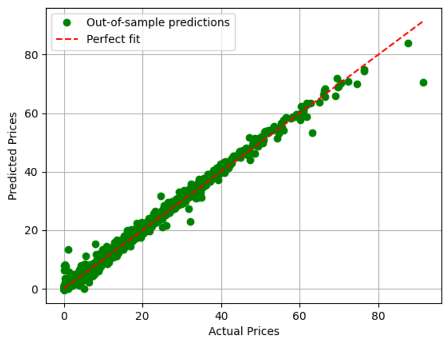
(a) Training (in-sample) set



(b) Test (out-of-sample) set



(c) Training (in-sample) set



(d) Test (out-of-sample) set

Figure 5.5: Predicted versus actual price plots for both the in-sample (80,000 samples) and out-of-sample (20,000 samples) sets. Subfigures 5.5a and 5.5b correspond to the Heston-based single barrier option dataset, while Subfigures 5.5c and 5.5d correspond to the NIG-based double barrier option dataset. Predictions are generated using the final model configurations summarised in Table 5.8.

and an increase in R^2 score, suggesting that the NIG-based double barrier option pricing task benefits from a more flexible ANN structure.

In contrast, the results for the Heston-based dataset show a deterioration in performance compared to Table 5.5. As shown in Figures 5.5a and 5.5b, both the training and test sets exhibit weaker fits, with greater deviation from the perfect prediction line. Additionally, the hyperparameter search process required substantially more computational time, as three additional hyperparameters were introduced into the search space. Despite this increased complexity, no clear improvement in model accuracy was achieved, aligning with prior literature findings that using the same number of neurons across hidden layers is generally sufficient for option pricing tasks.

Therefore, in the following analyses, we continue to use the configuration with the same number of neurons across hidden layers, as it offers a good balance between model complexity, computational efficiency, and predictive performance for the Heston-based single barrier option pricing tasks.

5.4 Numerical results

To evaluate the impact of training sample size on the predictive performance of the final ANN model for pricing Heston-based down-and-out put barrier options, we conduct experiments using varying training set sizes while fixing the out-of-sample test set size at 20,000 samples. Specifically, the training sizes considered are 20,000, 40,000, 60,000, 80,000, and 100,000 samples. The model configuration remains the same as summarised in Table 5.4. All models are trained over 200 epochs, and their performances are evaluated using mean squared error (MSE) and R^2 score on the fixed test set.

Table 5.10: Performance of the final ANN model (configuration summarised in Table 5.4) trained with different training set sizes and evaluated on a fixed out-of-sample test set of 20,000 samples.

Training size	**Mean-squared error**		**R^2 score**	
	Training set	Test set	Training set	Test set
20,000	0.04301476	0.18976083	0.97068628	0.87106386
40,000	0.02402977	0.15146867	0.98365558	0.89869903
60,000	0.02636204	0.15027335	0.98254345	0.89711840
80,000	0.04517221	0.10878742	0.97016359	0.92649495
100,000	0.02805878	0.08101602	0.98142239	0.94789864

Table 5.10 summarises the performance of the final ANN model trained with different training set sizes, while keeping the out-of-sample test set fixed at 20,000 samples. In general, larger training sets lead to improved test set performance, with test R^2 scores increasing steadily from 0.8711 (for 20,000 training samples) to 0.9479 (for 100,000 training samples), and corresponding decreases in test MSE.

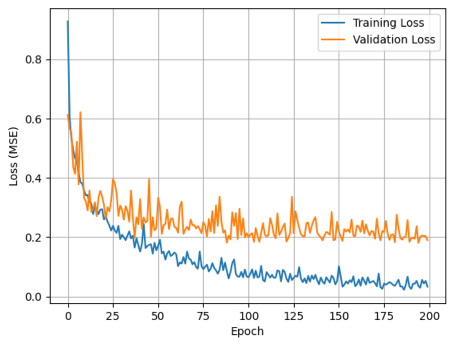
A slight fluctuation is observed between 40,000 and 60,000 test samples, where the test R^2 score shows a small dip. However, the overall trend remains clear, increasing the training data size enhances the model's generalisation ability, leading to higher predictive accuracy on unseen data.

To better understand the convergence behaviour at different training set sizes, Figure 5.6 presents the training and validation loss curves for models trained with 20,000, 40,000, 60,000, 80,000, and 100,000 samples, respectively. Overall, larger training sets lead to smoother and more stable convergence. For all training sizes, both training and validation losses decrease sharply within the first 25 epochs and then gradually flatten, with minimal further improvements after approximately 100 epochs. Increasing the training set size leads to progressively lower final validation losses at the end of training, whereas the final training losses remain relatively similar across different training sizes beyond 20,000 samples. This pattern suggests that providing more training data mainly enhances the model's generalisation ability, as reflected by improved validation performance, rather than significantly lowering the training loss. Interestingly, higher oscillations are observed in the validation loss curve for the model trained with 100,000 samples. This behaviour may be attributed to increased training data variability and noisier gradient updates due to small batch sizes relative to the dataset size.

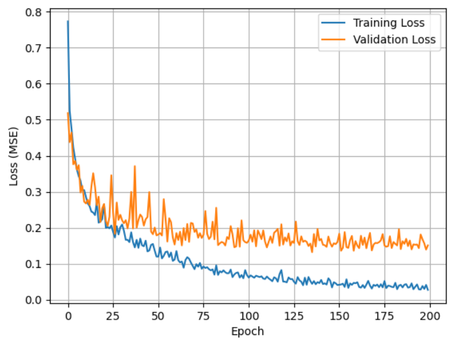
The predicted versus actual price plots across different training set sizes, shown in Figures 5.7–5.11, confirm that increasing the training data primarily improves the model's generalisation ability rather than its training set fitting performance. As training size increases, the test set predictions show tighter clustering around the perfect prediction line, particularly at higher training sizes (80,000 and 100,000 samples). However, the improvement is gradual rather than dramatic, and the in-sample prediction quality remains relatively stable beyond 20,000 samples. These observations are consistent with the behaviour seen in the validation loss curves and performance metrics.

5.5 Conclusion

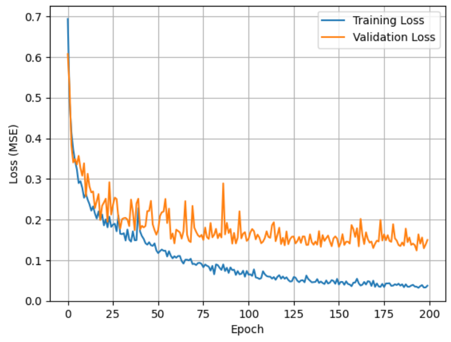
In this chapter, we developed and evaluated artificial neural network models for pricing discretely monitored down-and-out put barrier options under the Heston stochastic volatility model. We first introduced the network architecture and hyperparameter tuning methodology, and validated our final model by testing it on out-of-sample data. We further explored the effects of using different numbers of neurons across hidden layers, tested the model's generalisability by applying it to an alternative dataset in-



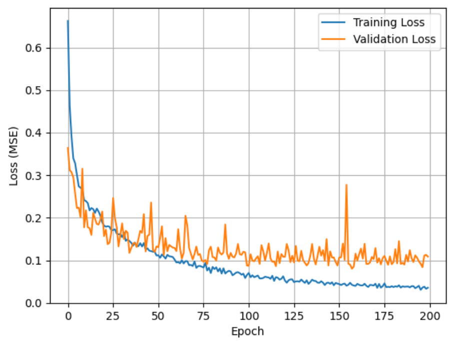
(a) Training set size: 20,000



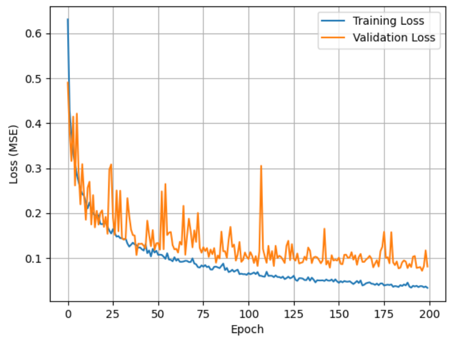
(b) Training set size: 40,000



(c) Training set size: 60,000



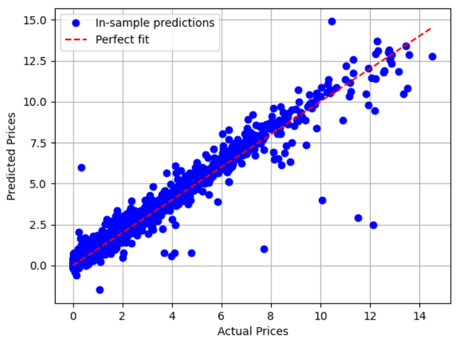
(d) Training set size: 80,000



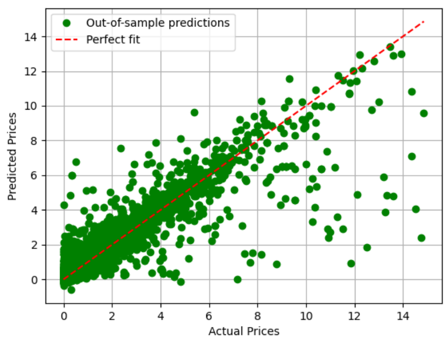
(e) Training set size: 100,000

Figure 5.6: Training and validation loss curves over 200 epochs for different training set sizes (20,000; 40,000; 60,000; 80,000; and 100,000 samples) using the final ANN model configuration from Table 5.4. Each subfigure shows the evolution of MSE across epochs, illustrating convergence behaviour for increasing amounts of training data.

volving NIG-based double barrier options, and analysed the impact of training set size on model performance and convergence. Our results show that using the same number of neurons across layers provides a good trade-off between performance and training efficiency, especially for the Heston-based problem. The ANN model demon-

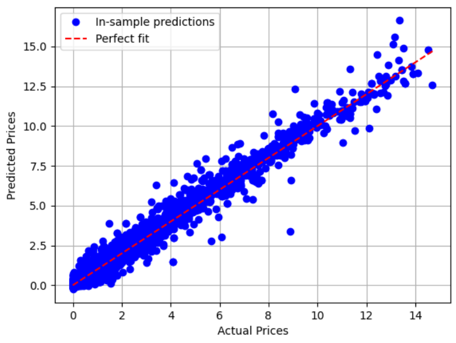


(a) Training (in-sample) set

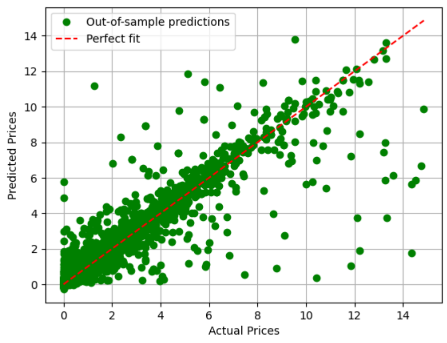


(b) Test (out-of-sample) set

Figure 5.7: Predicted versus actual price plots for the Heston-based single barrier option dataset with 20,000 training samples and 20,000 out-of-sample test samples. Predictions are generated using the final ANN model configuration summarised in Table 5.4.



(a) Training (in-sample) set

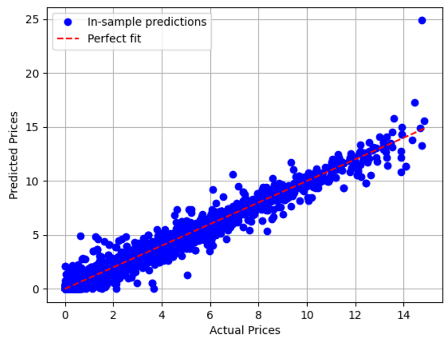


(b) Test (out-of-sample) set

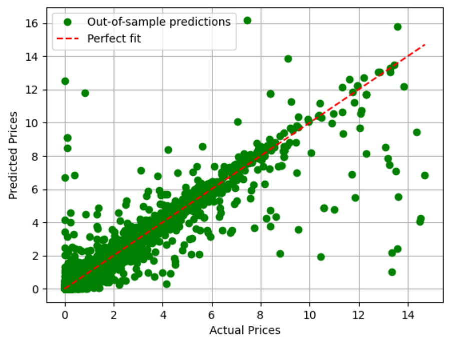
Figure 5.8: Predicted versus actual price plots for the Heston-based single barrier option dataset with 40,000 training samples and 20,000 out-of-sample test samples. Predictions are generated using the final ANN model configuration summarised in Table 5.4.

strated strong predictive accuracy and robustness across different training regimes, with larger training sets leading to improved generalisation and lower validation errors.

It is important to note that in this chapter, the fitness of the predicted option prices is assessed purely based on the generated training data. The correctness of the predicted prices relative to true theoretical values ultimately depends on the accuracy of the training data generated by the FZ method under the Heston model. Overall, this chapter demonstrates the strong predictive capabilities of deep learning models in replicating option prices generated by the FZ method for discretely

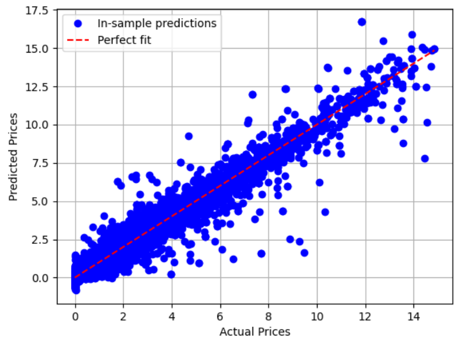


(a) Training (in-sample) set

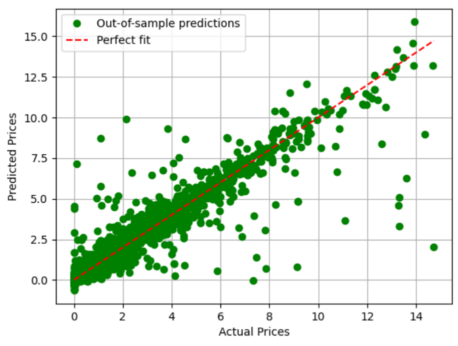


(b) Test (out-of-sample) set

Figure 5.9: Predicted versus actual price plots for the Heston-based single barrier option dataset with 60,000 training samples and 20,000 out-of-sample test samples. Predictions are generated using the final ANN model configuration summarised in Table 5.4.



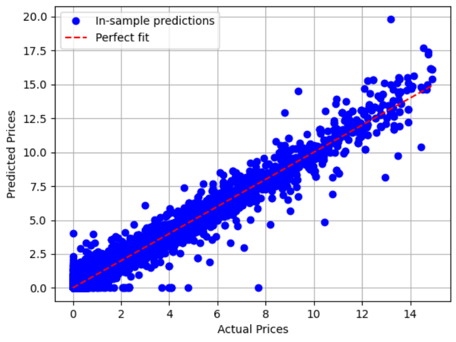
(a) Training (in-sample) set



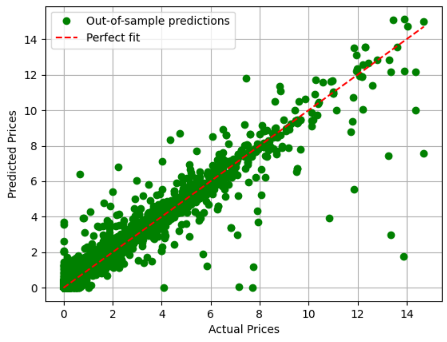
(b) Test (out-of-sample) set

Figure 5.10: Predicted versus actual price plots for the Heston-based single barrier option dataset with 80,000 training samples and 20,000 out-of-sample test samples. Predictions are generated using the final ANN model configuration summarised in Table 5.4.

monitored barrier options.



(a) Training (in-sample) set



(b) Test (out-of-sample) set

Figure 5.11: Predicted versus actual price plots for the Heston-based single barrier option dataset with 100,000 training samples and 20,000 out-of-sample test samples. Predictions are generated using the final ANN model configuration summarised in Table 5.4.

Chapter 6

Conclusions

This thesis has presented a comprehensive study on the pricing of discretely monitored barrier options under stochastic volatility, combining semi-analytical and machine learning techniques to advance current methodologies in both theory and practice.

Chapter 2 reviewed the literature on barrier options and various pricing methodologies, including Monte Carlo simulation, Fourier-based techniques, and machine learning approaches, highlighting their respective strengths and limitations. It also provided an overview of key stochastic processes, Fourier transforms, and several numerical methods, such as discrete Fourier transforms and fixed-point algorithms. This review serves to establish the necessary background for the models and techniques discussed later in the thesis, ensuring the content is more self-contained.

Chapter 3 focused on the joint conditional characteristic function (JCCF) of the Heston model, which plays a central role in Fourier-based pricing approaches. A detailed comparative analysis was conducted between two known formulations—Zhylyevskyy's and Griebsch's. While both are analytically equivalent, we identified a numerical discontinuity in the original Zhylyevskyy expression caused by branch switching in complex logarithmic. To address this, a novel unwrapping algorithm was introduced to ensure continuity and stability across the complex domain. Numerical experiments confirmed the equivalence of the corrected expressions and highlighted the improved robustness of the proposed fix.

Chapter 4 introduced two novel Fourier-based numerical schemes for pricing down-and-out put barrier options under the Heston model. The first, the Fourier

z-transform method (FZ), builds on the Wiener–Hopf technique and employs the z-transform to collapse the time dimension. The second, the Fourier–Hilbert transform method (FH), applies the barrier condition recursively via the Hilbert transform. Possible numerical enhancements such as spectral filtering and the fractional Fourier transform are explored. Numerical tests revealed that FH converges faster with fewer grid points, while FZ performs more robustly when the number of monitoring dates is large. Additional tests under extreme Heston parameter sets demonstrated that both methods remain operational, although both methods require finer grids large enough log-variance domain relative to Monte Carlo benchmarks.

Chapter 5 explored a data-driven alternative to traditional numerical methods by training artificial neural networks (ANNs) to approximate the prices of discretely monitored barrier options. Using training data generated by the FZ method, we demonstrated that deep learning models can learn the complex relationship between Heston model parameters and option prices with high accuracy. Sensitivity analyses were conducted on network architecture, training set size, and generalisation to different option structures. While the ANN model's predictive accuracy depends on the quality of the training data, the results of Chapter 5 show that, once properly trained, neural networks provide highly efficient and flexible tools for pricing barrier options. In particular, the models demonstrated strong generalization capabilities across different option structures and parameter regimes, offering practical advantages for fast pricing tasks, calibration workflows, and real-time risk management applications.

Future work could explore further improving the numerical robustness and efficiency of the FZ and FH methods under extreme parameter regimes. Potential directions include developing adaptive grid strategies to better capture narrow log-price and variance distributions, optimizing domain truncation dynamically based on density concentration, and applying multi-level or variance reduction techniques to enhance computational speed. Extending the methods to incorporate jumps or other affine stochastic volatility models beyond the Heston framework also presents promising avenues for research.

Appendix A

Derivation of the conditional characteristic function of Heston

In this chapter, we will derive the the characteristic function of the transition of the logarithm of the stock price given the initial variance and variance at maturity. The two conditioned variance can be generalised to two successive variance. Under risk neural measure, the solution of log price and variance can be found from Eq. (2.11) and Eq. (2.10)

$$x_T = x_0 + r(T - t_0) - \frac{1}{2}\int_{t_0}^{T} v(s)ds + \rho \int_{t_0}^{T} \sqrt{v(s)}dW_2(s) + \sqrt{1-\rho^2}\int_{t_0}^{T} \sqrt{v(s)}dW_1'(s), \tag{A.1}$$

$$v_T = v_0 + \kappa\theta(T - t_0) - \kappa \int_{t_0}^{T} v(s)ds + \sigma \int_{t_0}^{T} \sqrt{v(s)}dW_2(s). \tag{A.2}$$

Rewrite Eq. (A.2) as $\sigma \int_{t_0}^{T} \sqrt{v(s)}dW_2(s) = v_T - v_0 - \kappa\theta(T - t_0) + \kappa \int_{t_0}^{T} v(s)ds$ and substitute $\int_{t_0}^{T} \sqrt{v(s)}dW_2(s)$ to Eq. (A.1) and get

$$\begin{aligned} x_T - x_0 = & r(T - t_0) + \frac{\rho}{\sigma}(v_T - v_0 - \kappa\theta(T - t_0)) + (\frac{\kappa\rho}{\sigma} - \frac{1}{2})\int_{t_0}^{T} v(s)ds \\ & + \sqrt{1-\rho^2}\int_{t_0}^{T} \sqrt{v(s)}dW_1'(s). \end{aligned}$$

Note that $\int_{t_0}^{T} \sqrt{v(s)}dW_1'(s)$ is normally distributed with zero mean and variance equals to $\int_{t_0}^{T} v(s)ds$ via Ito isometry. With denoting z as standard normal random variable, then the characteristic function of $x_t - x_0$ given v_T and v_0 can be calculated as

$$E[e^{i\xi(x_T-x_0)}|v_T,v_0] = E\left[\exp\left(i\xi\left[r(T-t_0)+\frac{\rho}{\sigma}(v_T-v_0-\kappa\theta(T-t_0))\right.\right.\right.$$
$$\left.\left.\left.+(\frac{\kappa\rho}{\sigma}-\frac{1}{2})\int_{t_0}^{T}v(s)ds+\sqrt{1-\rho^2}\sqrt{\int_{t_0}^{T}v(s)ds}z\right]\right)|v_T,v_0\right].$$

Denoting $V := \int_{t_0}^{T} v(s)ds$ and using tower property we can obtain

$$\begin{aligned}
E[e^{i\xi(x_T-x_0)}|v_T,v_0] &= \exp\left(i\xi\left[r(T-t_0)+\frac{\rho}{\sigma}(v_T-v_0-\kappa\theta(T-t_0))\right]\right)\\
&\quad\times E\left[E\left[\exp(i\xi(\frac{\kappa\rho}{\sigma}-\frac{1}{2})V+i\xi\sqrt{1-\rho^2}\sqrt{V}z)|v_T,V\right]\right]\\
&= \exp\left(i\xi\left[r(T-t_0)+\frac{\rho}{\sigma}(v_T-v_0-\kappa\theta(T-t_0))\right]\right)\\
&\quad\times E\left[\exp(i\xi(\frac{\kappa\rho}{\sigma}-\frac{1}{2})V-\frac{1}{2}\xi^2(1-\rho^2)V)|v_T\right]\\
&= \exp\left(i\xi\left[r(T-t_0)+\frac{\rho}{\sigma}(v_T-v_0-\kappa\theta(T-t_0))\right]\right)\\
&\quad\times\Phi(\xi(\frac{\kappa\rho}{\sigma}-\frac{1}{2})+\frac{1}{2}i\xi^2(1-\rho^2)),
\end{aligned}$$

where $\Phi(\cdot)$ is defined in Eq. (3.37).

Appendix B

Derivation of the joint conditional characteristic function of Heston

In this chapter, we continue the derivation of the JCCF for the Heston model from Section 3.1, following the Riccati-based approach of Heston (1993), Zhylyevskyy (2010), and Griebsch (2013). Section 3.1 $x(T)$ is expressed as

$$\begin{aligned} x(T) =& x(t) + r\tau - \frac{1}{2}\int_t^T v(s)ds + \frac{\rho}{\sigma}\left(v(T) - v(t) - \kappa\theta\tau + \kappa\int_t^T v(s)ds\right) \\ &+ \sqrt{1-\rho^2}\int_t^T \sqrt{v(s)}dW_2(s), \end{aligned} \tag{B.1}$$

where T is maturity and the time to maturity $\tau = T - t$. Then as $x(t)$ is an affine process, the JCCF has the form

$$\begin{aligned} \phi(\xi_x, T; \xi_v, T|x, t, v, t) = \phi(\xi_x, \xi_v, \tau|x, v) &= E\left[\exp(i\xi_x x(T) + i\xi_v v(T))|x(t), v(t)\right] \\ &= \exp[p(\xi_x, \xi_v, \tau) + q(\xi_x, \xi_v, \tau)v(t) + i\xi_x x(t)], \end{aligned} \tag{B.2}$$

where $p := p(\xi_x, \xi_v, \tau)$ and $q := q(\xi_x, \xi_v, \tau)$ can be obtained in closed form solving a set of ordinary differential equations. The derivation of the JCCF starts with the application of Itō's formula to $\phi(\xi_x, \xi_v, \tau|x, v)$, which yields

$$-\frac{\partial\phi}{\partial\tau}dt + \frac{\partial\phi}{\partial x}dx + \frac{\partial\phi}{\partial v}dv + \frac{1}{2}\left(\frac{\partial^2\phi}{\partial x^2}dx^2 + \frac{\partial^2\phi}{\partial v^2}dv^2\right) + \frac{\partial^2\phi}{\partial x\partial v}dxdv. \tag{B.3}$$

Substituting dx (3.4) and dv (3.2) and combining like terms

$$d\phi = \frac{\partial \phi}{\partial x}\sqrt{v}dW_1 + \frac{\partial \phi}{\partial v}\sigma\sqrt{v}dW_2 + \mathscr{D}\phi, \tag{B.4}$$

where the differential operator $\mathscr{D}$ is defined as

$$\mathscr{D}\phi = \left[-\frac{\partial \phi}{\partial \tau} + \frac{\partial \phi}{\partial x}\left(r - \frac{v}{2}\right) + \frac{\partial \phi}{\partial v}\kappa(\theta - v) + \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial v^2}\sigma^2\right)\frac{v}{2} + \frac{\partial^2 \phi}{\partial x \partial v}\rho\sigma v\right] dt. \tag{B.5}$$

The Feynman-Kac theorem implies the conditional expectation in Eq. (B.2) is the unique bounded solution for the partial differential equation (PDE) $\mathscr{D}\phi(\xi_x, \xi_v, \tau | x, v) = 0$, i.e.

$$\frac{\partial \phi}{\partial \tau} = \frac{\partial \phi}{\partial x}\left(r - \frac{v}{2}\right) + \frac{\partial \phi}{\partial v}\kappa(\theta - v) + \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial v^2}\sigma^2\right)\frac{v}{2} + \frac{\partial^2 \phi}{\partial x \partial v}\rho\sigma v, \tag{B.6}$$

where $\tau \in [0,T], x \in \mathbb{R}, v \in \mathbb{R}^+$, and subject to the terminal condition

$$\phi(\xi_x, T; \xi_v, T | x, T; v, T) = \phi(\xi_x, \xi_v, 0 | x, v) = \exp(i\xi_x x + i\xi_v v), \tag{B.7}$$

where we have simplified the notation in the second expression for ϕ Partially differentiating Eq. (3.7) with respect to τ, x and v we obtain

$$\begin{aligned} &\frac{\partial \phi}{\partial \tau} = \left(\frac{\partial p}{\partial \tau} + \frac{\partial p}{\partial \tau}v(t)\right)\phi, \quad \frac{\partial \phi}{\partial x} = i\xi_x \phi, \quad \frac{\partial \phi}{\partial v} = q\phi, \\ &\frac{\partial^2 \phi}{\partial x^2} = (i\xi_x)^2 \phi, \quad \frac{\partial^2 \phi}{\partial v^2} = q^2 \phi, \quad \frac{\partial^2 \phi}{\partial x \partial v} = i\xi_x q \phi. \end{aligned}$$

After substituting these back into the PDE $\mathscr{D}\phi(\xi_x, \xi_v, \tau | x, v) = 0$, grouping the terms and and dividing by the common factor ϕ, we get

$$-\frac{\partial p}{\partial \tau} + i\xi_x r + \kappa\theta q + v(t)\left[-\frac{\partial q}{\partial \tau} - \frac{1}{2}(\xi_x^2 + i\xi_x) + (i\xi_x\rho\sigma - \kappa)q + \frac{1}{2}\sigma^2 q^2\right] = 0, \tag{B.8}$$

subject to the initial conditions $q(\xi_x, \xi_v, 0) = i\xi_v$ and $p(\xi_x, \xi_v, 0) = 0$.

As the Eq. (B.8) must be true for all $\tau \in [0,T]$, $x \in \mathbb{R}$ and $v \in \mathbb{R}^+$, both of the coefficient of v and the constant term must consequently be equal to 0. So we must have the following system of differential equations

$$\frac{dq}{d\tau}+Cq^2+Dq+E=0 \tag{B.9}$$

$$\frac{dp}{d\tau}-i\xi_x r-\kappa\theta q=0, \tag{B.10}$$

where $C=-\frac{1}{2}\sigma^2$, $D=-i\xi_x\rho\sigma+\kappa$ and $E=\frac{1}{2}(\xi_x^2+i\xi_x)$.

Eq. (B.9) is a Riccati equation which can be solved by defining $y:=q-q_1$ and making the change of variable $q=y+q_1$, where q_1 is a known solution of Eq. (B.9); then the Riccati equation reduces to the Bernoulli equation

$$\frac{dy}{d\tau}=-(2Cp_1+D)y-Cy^2. \tag{B.11}$$

Then change the variable again $w:=\frac{1}{y}$ and get

$$\frac{dw}{d\tau}=-Zw+C, \tag{B.12}$$

where $Z=-(2Cp_1+D)$. Multiplying both sides of Eq. (B.12) by $e^{Z\tau}$ and using the product rule, the result is

$$\frac{d(we^{Z\tau})}{d\tau}=Ce^{Z\tau}. \tag{B.13}$$

Integrating both sides with respect to τ, we get the general solution

$$w=\frac{C}{Z}+C_0e^{-Z\tau}, \tag{B.14}$$

where $C_0=w(0)-\frac{C}{Z}=\frac{1}{i\xi_v-q_1}-\frac{C}{Z}$. Changing back the variable q, we get

$$q=\frac{1+\frac{C}{Z}+\left(\frac{1}{i\xi_v-q_1}-\frac{C}{Z}\right)e^{-Z\tau}}{\frac{C}{Z}+\left(\frac{1}{i\xi_v-q_1}-\frac{C}{Z}\right)e^{-Z\tau}}. \tag{B.15}$$

q_1 can be found by solving

$$Cq_1^2+Dq_1+E=0. \tag{B.16}$$

Then there are two solutions,

$$q_{1,\pm} = \frac{-D \pm \sqrt{D^2 - 4CE}}{2C}. \tag{B.17}$$

Both solutions are analytically equivalent, but the positive square root solution yields numerical problems which the negative one avoids (Albrecher et al., 2007; Cui et al., 2017). This is why the characteristic function cannot be defined when $(\xi_x, \xi_v, \tau \neq 0) = (0, 0, \tau)$ in Zhylyevskyy's paper. We will discuss the problem in detail in Section 3.2.

After manipulations we get

$$q = \frac{1}{\sigma^2}\left[\kappa - \rho\sigma i\xi_x + \sqrt{A}\frac{Be^{-\tau\sqrt{A}} - 1}{Be^{-\tau\sqrt{A}} + 1}\right], \tag{B.18}$$

where

$$A = D^2 - 4CE = (\kappa - \sigma\rho i\xi_x)^2 + \sigma^2(i\xi_x + \xi_x^2) \tag{B.19}$$

$$B = -\frac{\sigma\rho i\xi_x - \kappa + \sqrt{A} + \sigma^2 i\xi_v}{\sigma\rho i\xi_x - \kappa - \sqrt{A} + \sigma^2 i\xi_v}. \tag{B.20}$$

We substitute Eq. (B.18) into Eq. (B.10) and similarly, with a change of variable and an integration, we obtain the solution

$$p = r\tau i\xi_x + \frac{\kappa\theta}{\sigma^2}\left[(\kappa - \sigma\rho i\xi_x)\tau - \tau\sqrt{A} + 2\log\frac{B+1}{Be^{-\tau\sqrt{A}} + 1}\right] + p_0, \tag{B.21}$$

where $p_0 = p(\xi_x, \xi_v, 0) = 0$.

Appendix C

Academic and professional activities

C.1 Preprints

The following preprints were prepared during my PhD research. They reflect the core theoretical and numerical contributions developed across the chapters of this thesis.

- **Expressions for the joint conditional characteristic function of the Heston stochastic volatility model**
  Authors: Jiaqi Liang, Carolyn E. Phelan, Guido Germano
  32 pages. Manuscript in preparation for submission: *Review of Derivatives Research*.

- **Pricing of discretely monitored barrier options with the Heston model using the Wiener-Hopf method**
  Authors: Jiaqi Liang, Carolyn E. Phelan, Daniele Marazzina, Guido Germano
  35 pages. Manuscript in preparation.

- **Pricing of discretely monitored barrier options with deep learning**
  Authors: Jiaqi Liang, Carolyn E. Phelan, Guido Germano
  20 pages. Manuscript in preparation for submission: *Quantitative Finance*.

C.2 Conference Presentations

During my PhD, I presented my research at several international conferences, workshops, and academic seminars in the form of both talks and poster sessions. These presentations primarily focused on the pricing of discretely monitored barrier options under stochastic volatility models—especially via the Wiener-Hopf method—as well as on the development of characteristic functions for the Heston model. The list below includes both invited and contributed contributions, which provided valuable opportunities to engage with researchers from both academia and industry.

- Jiaqi Liang, Carolyn E. Phelan, Daniele Marazzina, Guido Germano,
  "Application of the Wiener-Hopf technique to the pricing of discretely monitored barrier options with stochastic volatility",
  invited, Section: Numerical methods and quantitative finance: new perspectives and applications (NM1),
  42nd Annual Meeting of the Association for Mathematics Applied to Social and Economic Sciences (AMASES),
  Università di Napoli Parthenope, 13–15 September 2018,
  http://amases2018.uniparthenope.it/.

- Jiaqi Liang, Carolyn E. Phelan, Daniele Marazzina, Guido Germano,
  "Application of the Wiener-Hopf technique to the pricing of discretely monitored barrier options with stochastic volatility",
  Financial Computing and Analytics (FCA) Group Seminar,
  University College London, 14 November 2018.

- Jiaqi Liang, Carolyn E. Phelan, Daniele Marazzina, Guido Germano,
  "Pricing of discretely monitored barrier options with stochastic volatility using the Wiener-Hopf method",
  XX Workshop on Quantitative Finance (QFW 2019),
  ETH Zürich, 23–25 January 2019,
  https://people.math.ethz.ch/ jteichma/index.php?content=qfw2019.

- Jiaqi Liang, Carolyn E. Phelan, Daniele Marazzina, Guido Germano,
  "Pricing of discretely monitored barrier options with stochastic volatility using the Wiener-Hopf method",
  24th Annual Workshop on Economic Science with Heterogeneous Interacting Agents (WEHIA 2019),

City, University of London, 24–26 June 2019,
https://www.city.ac.uk/news-and-events/events/2019/06/workshop-on-economic-science-with-heterogeneous-interacting-agents.

- Jiaqi Liang, Carolyn E. Phelan, Guido Germano,
  "Expressions for the joint conditional characteristic function of the Heston stochastic volatility model",
  4th International Conference on Computational Finance (ICCF 2022),
  Bergische Universität Wuppertal, 6–10 June 2022,
  https://iccf2022.uni-wuppertal.de.

C.3 Teaching Assistantships

During my PhD at University College London, I contributed as a teaching assistant in several modules related to probability and financial mathematics. These roles involved assisting with coursework and assessments.

- **2018**: TA for *COMP0045: Probability Theory and Stochastic Processes* and *COMP0048: Financial Engineering*.
- **2019**: TA for *COMP0045: Probability Theory and Stochastic Processes*.
- **2020**: TA for *COMP0045: Probability Theory and Stochastic Processes*.

C.4 Professional Career

In addition to academic research, I gained industry experience through internships and full-time positions in quantitative finance. The following roles reflect my transition from doctoral study to professional practice.

- **June 2021 – August 2021**: Quantitative Analysis Intern, *JP Morgan*, London. Participated in the reviewing and validating of pricing models in the XVA area.
- **September 2021 – August 2022**: Quantitative Analyst, *JP Morgan*, London. Conducted model review and validation in the XVA team, including independent implementation of pricing models for benchmarking and testing purposes.

- **August 2022 – Present**: Quantitative Analyst, QIS Team, *Barclays*, London. Focused on implementing, and maintaining systematic index strategies across multiple asset classes.

Bibliography

Abate, J. and Whitt, W. (1992). Numerical inversion of probability generating functions. *Operations Research Letters*, 12(4):245–251. doi:10.1016/0167-6377(92)90050-D.

Abrahams, I. D. (1997). On the solution of Wiener–Hopf problems involving noncommutative matrix kernel decompositions. *SIAM Journal on Applied Mathematics*, 57(2):541–567. doi:10.1137/S0036139995287673.

Aimi, A., Guardasoni, C., Ortiz-Gracia, L., and Sanfelici, S. (2023). Fast barrier option pricing by the COS BEM method in Heston model (with Matlab code). *Computational Methods in Applied Mathematics*, 23(2):301–331. doi:10.1515/cmam-2022-0088.

Albrecher, H., Mayer, P., Schoutens, W., and Tistaert, J. (2007). The little Heston trap. *Wilmott Magazine*, pages 83–92. https://perswww.kuleuven.be/~u0009713/HestonTrap.pdf.

Andersen, L. B. G. (2007). Efficient simulation of the Heston stochastic volatility model. doi:10.2139/ssrn.946405.

Anderson, D. G. (1965). Iterative procedures for nonlinear integral equations. *Journal of the Association for Computing Machinery*, 12(4):547–560. doi:10.1145/321296.321305.

Bailey, D. H. and Swarztrauber, P. N. (1991). The fractional Fourier transform and applications. *SIAM Review*, 33(3):389–404. doi:10.1137/1033097.

Bakshi, G., Cao, C., and Chen, Z. (1997). Empirical performance of alternative option pricing models. *Journal of Finance*, 52(5):2003–2049. doi:10.1111/j.1540-6261.1997.tb02749.x.

Bayer, C., Horvath, B., Muguruza, A., Stemper, B., and Tomas, M. (2019). On deep calibration of (rough) stochastic volatility models. *arXiv preprint*. doi:10.48550/arXiv.1908.08806.

Black, F. and Scholes, M. (1973). Pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654. doi:10.1086/260062.

Boyle, P. P. (1977). Options: A Monte Carlo approach. *Journal of Financial Economics*, 4(3):323–338. doi:10.1016/0304-405X(77)90005-8.

Boyle, P. P. and Tian, Y. (1998). An explicit finite difference approach to the pricing of barrier options. *Applied Mathematical Finance*, 5(1):17–43. doi:10.1080/135048698334718.

Broadie, M. and Kaya, Ö. (2006). Exact simulation of stochastic volatility and other affine jump diffusion processes. *Operations Research*, 54(2):217–231. doi:10.1287/opre.1050.0247.

Buehler, H., Gonon, L., Teichmann, J., and Wood, B. (2019). Deep hedging. *Quantitative Finance*, 19(8):1271–1291. doi:10.1080/14697688.2019.1571683.

Cai, N., Li, C., and Shi, C. (2021). Pricing discretely monitored barrier options: When Malliavin calculus expansions meet Hilbert transforms. *Journal of Economic Dynamics and Control*, 127. doi:10.1016/j.jedc.2021.104113.

Carr, P. and Madan, D. (1999). Option valuation using the fast Fourier transform. *Journal of Computational Finance*, 2:61–73. doi:10.21314/JCF.1999.043.

Choi, J. and Kwok, Y. K. (2024). Simulation schemes for the Heston model with Poisson conditioning. *European Journal of Operational Research*, 314(1):363–376. doi:10.1016/j.ejor.2023.10.048.

Chourdakis, K. (2005). Option pricing using the fractional FFT. *Journal of Computational Finance*, 8(2):1–18. doi:10.21314/JCF.2005.137.

Claesen, M. and De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint*. doi:10.48550/arXiv.1502.02127.

Cox, J. C. (1996). The constant elasticity of variance option pricing model. *The Journal of Portfolio Management*, 23(5):15–17. doi:10.3905/jpm.1996.015.

Cox, J. C., Ross, S. A., and Rubinstein, M. (1979). Option pricing: A simplified approach. *Journal of Financial Economics*, 7(3):229–263. doi:10.1016/0304-405X(79)90015-1.

Cui, Y., del Baño Rollin, S., and Germano, G. (2017). Full and fast calibration of the Heston stochastic volatility model. *European Journal of Operational Research*, 263(2):625–638. doi:10.1016/j.ejor.2017.05.018.

Culkin, R. and Das, S. R. (2017). Machine learning in finance: The case of deep learning for option pricing. *Journal of Investment Management*, 15(4):92–100. https://api.semanticscholar.org/CorpusID:92991481.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314. doi:10.1007/BF02551274.

Dadachanji, Z. (2015). *FX barrier options: A comprehensive guide for industry quants*. Applied Quantitative Finance. Palgrave Macmillan. doi:10.1057/9781137462756.

Duffie, D., Filipović, D., Schachermayer, W., et al. (2003). Affine processes and applications in finance. *Annals of Applied Probability*, 13(3):984–1053. doi:10.1214/aoap/1060202833.

Duffie, D., Pan, J., and Singleton, K. (2000). Transform analysis and asset pricing for affine jump-diffusions. *Econometrica*, 68(6):1343–1376. doi:10.1111/1468-0262.00164.

Eberlein, E. and Kallsen, J. (2019). Affine and polynomial processes. In *Mathematical Finance*, pages 365–398. Springer. doi:10.1007/978-3-030-26106-1_6.

Eldan, R. and Shamir, O. (2016). The power of depth for feedforward neural networks. In *Conference on Learning Theory*, pages 907–940. doi:10.48550/arXiv.1512.03965.

Fang, F. and Oosterlee, C. W. (2009a). A novel pricing method for European options based on Fourier–Cosine series expansions. *SIAM Journal on Scientific Computing*, 31(2):826–848. doi:10.1137/080718061.

Fang, F. and Oosterlee, C. W. (2009b). Pricing early-exercise and discrete barrier options by Fourier–Cosine series expansions. *Numerische Mathematik*, 114(1):27–62. doi:10.1007/s00211-009-0252-4.

Fang, F. and Oosterlee, C. W. (2011). A Fourier-based valuation method for Bermudan and barrier options under Heston's model. *SIAM Journal on Financial Mathematics*, 2(1):439–463. doi:10.1137/100794158.

Feng, L. and Linetsky, V. (2008). Pricing discretely monitored barrier options and defaultable bonds in Lévy process models: A Hilbert transform approach. *Mathematical Finance*, 18(3):337–384. doi:10.1111/j.1467-9965.2008.00338.x.

Fusai, G., Abrahams, I. D., and Sgarra, C. (2006). An exact analytical solution for discrete barrier options. *Finance and Stochastics*, 10:1–26. doi:10.1007/s00780-005-0170-y.

Fusai, G., Germano, G., and Marazzina, D. (2016). Spitzer identity, Wiener–Hopf factorisation, and pricing of discretely monitored exotic options. *European Journal of Operational Research*, 251(1):124–134. doi:10.1016/j.ejor.2015.11.027.

Gatheral, J. (2011). *The volatility surface: A practitioner's guide*, volume 357. John Wiley & Sons. doi:10.1002/9781119202073.

Glasserman, P. and Kim, K. K. (2011). Gamma expansion of the Heston stochastic volatility model. *Finance and Stochastics*, 15:267–296. doi:10.1007/s00780-009-0115-y.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256. PMLR. http://proceedings.mlr.press/v9/glorot10a.html.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org/.

Griebsch, S. A. (2013). The evaluation of European compound option prices under stochastic volatility using Fourier transform techniques. *Review of Derivatives Research*, 16(2):135–165. doi:10.1007/s11147-012-9083-z.

Hagan, P. S., Kumar, D., Lesniewski, A. S., and Woodward, D. E. (2002). Managing smile risk. *Wilmott Magazine*, pages 84–108. http://web.math.ku.dk/~rolf/SABR.pdf.

Haug, E. G. (1999). Barrier put–call transformations. *Social Science Research Network*. doi:10.2139/ssrn.150156.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034. doi:10.1109/ICCV.2015.123.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6(2):327–343. doi:10.1093/rfs/6.2.327.

Hoffer, E., Hubara, I., and Soudry, D. (2017). Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. *arXiv preprint*. doi:10.48550/arXiv.1705.08741.

Hoffman, J. D. and Frankel, S. (2018). *Numerical methods for engineers and scientists*. CRC Press. doi:10.1201/9781315274508.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366. doi:10.1016/0893-6080(89)90020-8.

Horvath, B., Muguruza, A., and Tomas, M. (2019). Deep learning volatility. *Social Science Research Network*. https://ssrn.com/abstract=3322085.

Hutchinson, J. M., Lo, A. W., and Poggio, T. (1996). A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance*, 49(3):851–889. https://ssrn.com/abstract=236673.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR. doi:10.48550/arXiv.1502.03167.

Jones, D. S. (1984a). Commutative Wiener–Hopf factorization of a matrix. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 393(1804):185–192. doi:10.1098/rspa.1984.0053.

Jones, D. S. (1984b). Factorization of a Wiener–Hopf matrix. *IMA Journal of Applied Mathematics*, 32:211–220. doi:10.1093/imamat/32.1-3.211.

Jones, D. S. (1991). Wiener–Hopf splitting of a 2×2 matrix. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 434:419–433. doi:10.1098/rspa.1991.0101.

Joshi, M. S. and Joshi, M. S. (2003). *The concepts and practice of mathematical finance*, volume 1. Cambridge University Press. https://api.semanticscholar.org/CorpusID:152761915.

Kahl, C. and Jäckel, P. (2005). Not-so-complex logarithms in the Heston model. *Wilmott Magazine*, 19(9):94–103. https://wilmott.com/not-so-complex-logarithms-in-the-heston-model/.

Kahl, C. and Jäckel, P. (2006). Fast strong approximation Monte Carlo schemes for stochastic volatility models. *Quantitative Finance*, 6:513–536. `doi:10.1080/14697680600841108`.

Kallsen, J. (2006). A didactic note on affine stochastic volatility models. In Kabanov, Y., Liptser, R., and Stoyanov, J., editors, *From Stochastic Calculus to Mathematical Finance*, pages 343–368. Springer, Berlin, Heidelberg. `doi:10.1007/978-3-540-30788-4_18`.

Kallsen, J., Muhle-Karbe, J., and Voß, M. (2011). Pricing options on variance in affine stochastic volatility models. *Mathematical Finance*, 21(4):627–641. `doi:10.1111/j.1467-9965.2010.00447.x`.

Karataş, T., Oskoui, A., and Hirsa, A. (2019). Supervised deep neural networks (DNNs) for pricing/calibration of vanilla/exotic options under various different processes. *arXiv preprint*, arXiv:1902.05810. `doi:10.48550/arXiv.1902.05810`.

Kawaguchi, K., Huang, J., and Kaelbling, L. P. (2018). Effect of depth and width on local minima in deep learning. *arXiv preprint*. `doi:10.48550/arXiv.1811.08150`.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint*. `doi:10.48550/arXiv.1412.6980`.

Kou, S. G. (2007). Discrete barrier and lookback options. In *Handbooks in Operations Research and Management Science*, volume 15, pages 343–373. Elsevier. `doi:10.1016/S0927-0507(07)15008-8`.

Lawrie, J. B. and Abrahams, I. D. (2007). A brief historical perspective of the Wiener–Hopf technique. *Journal of Engineering Mathematics*, 59(4):351–358. `doi:10.1007/s10665-007-9195-x`.

Levendorskiĭ, S. Z. (2004). Pricing of the American put under Lévy processes. *International Journal of Theoretical and Applied Finance*, 7(3):303–335. `doi:10.1142/S0219024904002446`.

Lewis, A. (2001). A simple option formula for general jump-diffusion and other exponential Lévy processes. doi:10.2139/ssrn.282110.

Li, Y. and Yuan, Y. (2017). Convergence analysis of two-layer neural networks with ReLU activation. *arXiv preprint*. doi:10.48550/arXiv.1705.09886.

Liu, S., Oosterlee, C. W., and Bohte, S. M. (2019). Pricing options and computing implied volatilities using neural networks. *Risks*, 7(1):16. doi:10.3390/risks7010016.

Lord, R., Fang, F., Bervoets, F., and Oosterlee, C. W. (2008). A fast and accurate FFT-based method for pricing early-exercise options under Lévy processes. *SIAM Journal on Scientific Computing*, 30(4):1678–1705. doi:10.1137/070683878.

Lord, R. and Kahl, C. (2006). Why the rotation count algorithm works. *Tinbergen Institute Discussion Paper*, 2006-065/2. doi:10.2139/ssrn.921335.

Lord, R. and Kahl, C. (2010). Complex logarithms in Heston-like models. *Mathematical Finance*, 20(4):671–694. doi:10.1111/j.1467-9965.2010.00416.x.

Lord, R., Koekkoek, R., and van Dijk, D. (2010). A comparison of biased simulation schemes for stochastic volatility models. *Quantitative Finance*, 10(2):177–194. doi:10.1080/14697680802392496.

Malliaris, M. and Salchenberger, L. M. (1993). A neural network model for estimating option prices. *Applied Intelligence*, 3:193–206. doi:10.1007/BF00871937.

McGhee, W. A. (2018). An artificial neural network representation of the SABR stochastic volatility model. *Social Science Research Network*. https://ssrn.com/abstract=3288882.

Merton, R. C. (1973). Theory of rational option pricing. *Bell Journal of Economics and Management Science*, 4(1):141–183. doi:10.2307/3003143.

Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1-2):125–144. doi:10.1016/0304-405X(76)90022-2.

Mitra, S. K. (2012). An option pricing model that combines neural network approach and Black–Scholes formula. *Global Journal of Computer Science and Technology*, 12(4). `https://computerresearch.org/index.php/computer/article/view/455`.

Noble, B. and Weiss, G. (1959). Methods based on the Wiener–Hopf technique for the solution of partial differential equations. *Physics Today*, 12(9):50–50. `doi:10.1063/1.3060973`.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. `doi:10.48550/arXiv.1201.0490`.

Phelan, C. E., Marazzina, D., Fusai, G., and Germano, G. (2019). Hilbert transform, spectral filters, and option pricing. *Annals of Operations Research*, 282(1-2):273–298. `doi:10.1007/s10479-018-2881-4`.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge University Press. `https://dl.acm.org/doi/10.5555/1403886`.

Rich, D. R. (1994). The mathematical foundation of barrier option-pricing theory. *Advances in Futures and Options Research*, 7:267–311. `https://ssrn.com/abstract=5730`.

Schwartz, E. S. (1977). The valuation of warrants: Implementing a new approach. *Journal of Financial Economics*, 4(1):79–93. `doi:10.1016/0304-405X(77)90037-X`.

Shreve, S. E. (2004). *Stochastic calculus for finance II: Continuous-time models*. Springer. `https://link.springer.com/book/9780387401010`.

Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint*. `doi:10.48550/arXiv.1711.00489`.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958. `http://jmlr.org/papers/v15/srivastava14a.html`.

Tagliani, A. and Milev, M. (2009). Discrete monitored barrier options by finite difference schemes. *Mathematics and Education in Mathematics*, 38:81–89. `https://www.researchgate.net/publication/241128381_Discrete_Monitored_Barrier_Options_by_Finite_Difference_Schemes`.

Tse, S. T. and Wan, J. W. L. (2013). Low-bias simulation scheme for the Heston model by inverse Gaussian approximation. *Quantitative Finance*, 13:919–937. `doi:10.1080/14697688.2012.696678`.

van Haastrecht, A. and Pelsser, A. (2010). Efficient, almost exact simulation of the Heston stochastic volatility model. *International Journal of Theoretical and Applied Finance*, 13:1–43. `doi:10.1142/S0219024910005668`.

Veitch, B. H. and Abrahams, I. D. (2007). On the commutative factorization of $n \times n$ matrix Wiener–Hopf kernels with distinct eigenvalues. volume 463, pages 613–639. `doi:10.1098/rspa.2006.1780`.

Walker, H. F. and Ni, P. (2011). Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735. `doi:10.1137/10078356X`.

Walker, J. S. (2017). *Fast Fourier transforms*. CRC Press. `doi:10.1201/9780203756188`.

White, H. (1988). Economic prediction using neural networks: The case of IBM daily stock returns. *IEEE International Conference on Neural Networks*, 2:451–458. `doi:10.1109/ICNN.1988.23959`.

Wiener, N. and Hopf, E. (1931). Über eine Klasse singulärer Integralgleichungen. *Sitzungsberichte der Preußischen Akademie der Wissenschaften, Mathematisch-Physikalische Klasse*, 1:696–706.

Wong, H. Y. and Kwok, Y.-K. (2003). Multi-asset barrier options and occupation time derivatives. *Applied Mathematical Finance*, 10(3):245–266. `doi:10.1080/1350486032000107352`.

Zhylyevskyy, O. (2010). A fast Fourier transform technique for pricing American options under stochastic volatility. *Review of Derivatives Research*, 3(1):1–24. `doi:10.1007/s11147-009-9041-6`.

Zhylyevskyy, O. (2012). Joint characteristic function of stock log-price and squared volatility in the Bates model and its asset pricing applications. *Theoretical Economics Letters*, 2(4):400–407. `doi:10.4236/tel.2012.24074`.