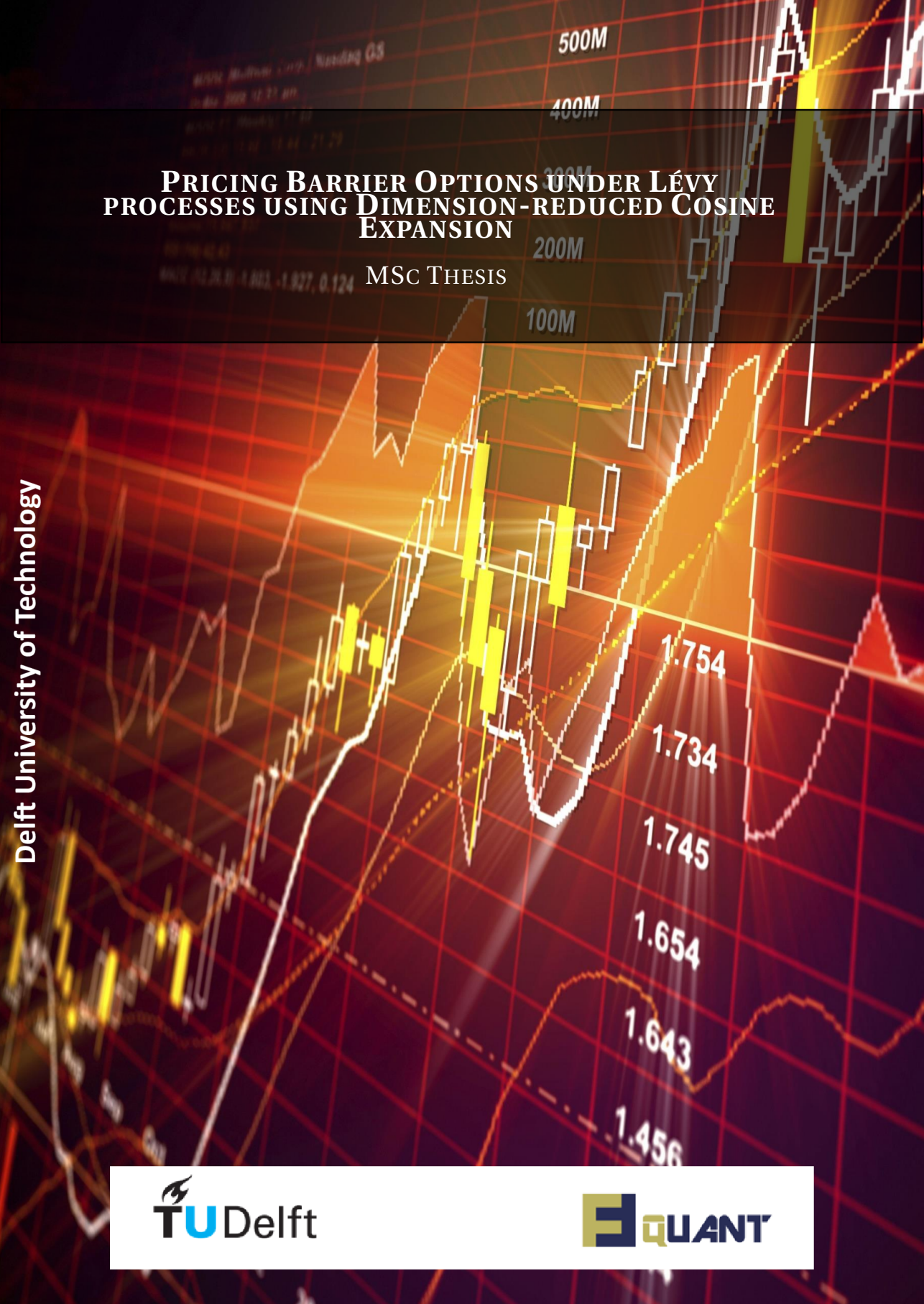


Pricing Barrier Options under Lévy processes using Dimension-reduced Cosine Expansion

MSc Thesis

Delft University of Technology

TU Delft

QUANT

PRICING BARRIER OPTIONS UNDER LÉVY PROCESSES USING DIMENSION-REDUCED COSINE EXPANSION

MSc THESIS

by

Levi KLOMP

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on August 30, 2023 at 11:00.

Student number:	4908775	
Project duration:	December 10, 2022 - August 30, 2023	
Thesis committee:	Dr. F. Fang,	TU Delft, supervisor
	Prof.Dr.ir. C. Vuik,	TU Delft
	Dr. ir. J. Bierkens,	TU Delft
	Dr. X. Shen,	FF Quant, supervisor
Institution:	Delft University of Technology, Faculty of Electrical Engineering, Mathematics & Computer Science (EEMCS)	
Master programme:	Applied Mathematics	
Specialisation:	Financial Engineering	

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

ABSTRACT

Barrier options are fundamental financial tools that give rise to pricing challenges, particularly when embedded within stochastic models. This study directs its focus towards Lévy processes as a strategic approach to navigate and resolve these intricate complexities.

The model assumption adopted in this thesis is that the underlying log-asset price follows a Levy process. This assumption, in combination with the COS method, enables us to reduce the dimensionality of the pricing problem for barrier options. To be more precise, the strike price and the log-asset price are both factored out from the calculation formula.

The traditional COS method, known for rapid European option valuation, depends on the knowledge of the ch.f.. The COS method has been extended to pricing barrier options in [1] and [2]. However, both methods depend on a recursive calculation formula whereby the number of recursion equal to the number of monitoring dates, and thus, the calculation speed lags behind pricing European options , especially with quite a number of monitoring dates.

Our key insight lies in the potential of using the traditional COS method for pricing barrier options. As the first step, we integrate the barrier hitting probability into the ch.f., thereby transforming it into a survival ch.f. Although the exact function of the survival ch.f. is unknown, its values on the grid used by the COS method can be accurately solved via Singular Value Decomposition (SVD). Testing results robustly reinforces our findings.

Building on this, we work out two techniques to approximate the characteristic function through supervised learning. The model takes Lévy process model parameters and yields approximation function of the ch.f..

The first method, COS-GPR, combines Gaussian Process Regression (GPR) with the traditional COS method for ch.f. estimation. Assuming mutual independence among multivariate model outputs, COS-GPR addresses GPR's limitations at boundaries with an additional Fourier expansion. Notably, COS-GPR demonstrates significant significantly faster calculation than the existing COS Barrier by seven times while maintaining heightened accuracy, with occasional outliers.

The second method, the COS Fourier CPD (CFC) method, replaces the ch.f. in the COS method by a Fourier-series expansion of the ch.f., after which the dimension is strategically reduced using Canonical Polyadic Decomposition (CPD). CFC achieves superior overall accuracy compared to COS Barrier, with a speed increase of 180 times and minimal instances of extreme errors. In comparison to COS-GPR, CFC's accuracy slightly decreases but exhibits a smaller maximum error. The CFC method has a 180-fold increase in speed compared to the COS Barrier method and maintains linear time complexity as training points per dimension increase.

In conclusion, this study introduces efficient methods based on supervised machine learning for barrier option pricing under Levy processes. The fusion of conventional

methods with advanced approaches leads to significant improvements in both speed and precision. These innovations hold the potential to transform financial derivative valuation, enhancing accuracy and efficiency in the process.

List of Figures

LIST OF TABLES

List of Abbreviations

ALS Alternating Least Squares 60

ANN Artificial Neural Network 1

CG Conjugate Gradient 67

ch.f. characteristic function 1, 87

CPD Canonical Polyadic Decomposition 59, 87

DFT Discrete Fourier Transform 20

DI Down-and-in 9

DNN Deep Neural Network 1

DO Down-and-out 9

FFT Fast Fourier Transform 20

FSA-HTF Fourier Series Approximation via Hidden Tensor Factorization 66

GBM Geometric Brownian Motion 52

GPR Gaussian Process Regression 1, 36, 87

GS Gram-Schmidt 12

LS Least-squares 13

LSTM Long Short-Term Memory 1

NN Neural Network 1, 36, 87

SD Spectral Decomposition 11

SVD Singular Value Decomposition 11, 13, 65

UI Up-and-in 9

UO Up-and-out 9

CONTENTS

1

INTRODUCTION

In the dynamic and intricate landscape of finance, the valuation of derivative instruments, particularly options, assumes a pivotal role in effective risk management and the optimization of portfolios. Within this array of financial tools, barrier options stand out, offering a compelling blend of safeguarding against adverse market shifts and enticing profit potential. Nonetheless, achieving accurate pricing for barrier options remains a formidable undertaking. This challenge is primarily rooted in the gradual convergence of conventional techniques, such as Monte Carlo (MC) simulations, leading to notable computational inefficiencies.

Moreover, the growing adoption of Neural Network (NN), including Deep Neural Network (DNN)/Artificial Neural Network (ANN) and Long Short-Term Memory (LSTM), in the realm of quantitative finance [3], [4] for option pricing, has raised concerns about interpretability. These models often function as enigmatic "black boxes," hindering a clear understanding of the pricing process.

An alternative innovative avenue in the domain of barrier option pricing is the COS Barrier method [1],[2] an approach that has been previously established. This method builds upon the foundational COS method in 2008 [5], which is renowned for its rapid convergence, particularly when dealing with simpler options such as European options.

The two methods developed in this thesis are based on our insight presented and proved in Chapter 3. That is, the COS method for pricing European options can be directly used for pricing barrier options, if one replaces the ch.f. by what we name as "survival ch.f.". This evidence underscores the adaptability of the COS method for barrier option pricing, highlighting its superiority over traditional approaches such as the COS Barrier method [1]. The central aim of both models is to estimate the survival characteristic function (ch.f.). However, within the context of barrier options, the precise modeling of this survival ch.f. becomes more intricate and challenging, primarily due to its integration of the barrier hitting probability inherent in the underlying stochastic process.

The first method we propose, elucidated in Chapter 4, is the COS-GPR method. This approach harnesses Gaussian Process Regression (GPR) to estimate the survival ch.f. for

option valuation using the COS method. GPR, a supervised learning technique widely utilized in quantitative finance [6], employs labeled data to generate predictions. In this context, model parameters of the underlying process serve as inputs, while the outputs pertain to the survival ch.f.. One notable advantage of employing supervised learning methods like GPR is their expedited option pricing, as demonstrated in Chapter 5. Moreover, the inherent transparency of supervised learning aligns with the requisites of the financial industry. Nevertheless, GPR's limitations near data boundaries [6] and the underlying assumptions regarding multivariate outputs [7] necessitate the introduction of a complementary model, a challenge we subsequently address.

In Chapter 6, we introduce the CFC method, an approach that amalgamates the COS method (C) with Fourier-cosine expansion (F) of the survival ch.f., combined with dimensionality reduction through Canonical Polyadic Decomposition (C). Diverging from GPR's localized decomposition, the CFC-method achieves a global decomposition by employing cosine basis functions spanning the entire function space. This globalized approach stays accurate around training boundaries and significantly expedite computation, thereby addressing the limitations of the COS-GPR method. The accuracy of the CFC method, compared to the other two methods, is extensively elaborated upon in Chapter 7, which also involves an analysis of the error behavior when we adjust method parameters to determine optimal values.

As we navigate through this thesis, we delve into the intricate mechanics, strengths, and limitations of these two innovative pricing methods. Our overarching objective is to equip financial practitioners with a comprehensive toolkit, fostering the adoption of efficient and accurate valuation techniques, and building upon the foundational work laid by Fang and Oosterlee [5] and [1].

2

MATHEMATICAL FRAMEWORK

This chapter presents the foundation of all mathematical tools used in this paper. This knowledge is required to understand our research and apply our results in the field of option pricing.

2.1. STOCHASTIC CALCULUS

In this section, we will explore definitions and theorems in the field of stochastic calculus. The notation used throughout is based on references [8] and [9]. These findings are crucial for the formulation and examination of option pricing.

Definition 2.1.1. *(Adapted process). A process $X = \{X_t, t \geq 0\}$ is adapted to a filtration $\{\mathcal{F}, t \geq 0\}$ if for all $t \geq 0$, X_t is $\mathcal{F}_t$-measurable.*

Definition 2.1.2. *(Martingale). Let $M = \{M_t, t \geq 0\}$ be a process defined on the probability space $(\Omega, \mathcal{F}, P)$ equipped with a filtration $\mathcal{F}_t, t \geq 0$. Then M is said to be a martingale if:*

1. *For all $t \geq 0$, M_t is integrable.*
2. *M is an adapted process.*
3. *M satisfies the martingale property, which reads:*

$$\mathbb{E}[M_t \mid \mathcal{F}_s] = M_s, \qquad \forall 0 \leq s \leq t.$$

Definition 2.1.3. *(Semimartingale). A stochastic process $X = \{X_t, t \geq 0\}$ is called a semimartingale if it can be decomposed as follows:*

$$X = X_0 + M + A,$$

where the random variable X_0 is finite and $\mathcal{F}_0$-measurable, the stochastic process M is a local martingale, and the stochastic process A has finite variation.

Definition 2.1.4. *A real-valued process $\{W(t), t \geq 0\}$ is called a Brownian motion if:*

1. *Initial value at 0: $W(0) = 0$.*
2. *Normally distributed increments: For all $0 \leq s < t$, $W(t) - W(s) \sim \mathcal{N}(0, t-s)$.*
3. *Independent increments: For $0 \leq t_0 < t_1 < \ldots < t_n$, the random variables $Y_i = W(t_i) - W(t_{i-1})$, $i = 1, \ldots, n$, are independent.*
4. *Continuous trajectories: The map $t \mapsto W(t)$ is continuous.*

Theorem 2.1.1. *The Brownian motion $W = \{W(t), t \geq 0\}$ is a martingale.*

Definition 2.1.5. *For any square-integrable adapted process $g(t)$ with continuous sample paths, the Itô integral is given by:*

$$I(T) = \int_0^T g(t)dW(t) := \lim_{m\to\infty} I_m(T), \quad \text{in } L^2,$$

Here, $I_m(T) = \int_0^T g_m(t)dW(t)$ for some simple process $g_m(t) = \sum_{j=0}^{n-1} \eta_j \mathbf{1}[t_j, t_{j+1})$, satisfying:

$$\lim_{m\to\infty} \mathbb{E}\left[\int_0^T (g_m(t) - g(t))^2 dt\right] = 0,$$

where η_j is $\mathcal{F}_{t_j}$-measurable for all $j = 0, 1, \ldots, n-1$ and square-integrable.

Theorem 2.1.2. *(Itô's Isometry). For any Brownian motion $W(t)$ and stochastic process $g(t)$, satisfying the usual regularity conditions, the following equality holds:*

$$\mathbb{E}\left[\left(\int_0^T g(t)dW(t)\right)^2\right] = \int_0^T \mathbb{E}[g^2(t)]dt.$$

Theorem 2.1.3. *(Itô's formula). Let $f \in C^2(\mathbb{R})$ and consider a continuous semimartingale X with decomposition $X = M + A$. Then, the stochastic process $(f(X_t))_{t\geq 0}$ is also a semimartingale, and it holds:*

$$f(X_t) = f(X_0) + \int_0^t \frac{\partial f}{\partial x}(X_u)dX_u + \frac{1}{2}\int_0^t \frac{\partial^2 f}{\partial x^2}(X_u)d[X]_u,$$

where $[X]$ denotes the quadratic variation of the process $(X_t)_{t\geq 0}$.
Itô's formula is often expressed in differential form:

$$df(X_t) = \frac{\partial f}{\partial x}(X_t)dX_t + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(X_t)d[X]_t.$$

2.1.1. MODELS DESCRIBING ASSET PATHS

This paper employs two illustrative models, namely the Geometric Brownian Motion (GBM) and the CGMY model, to capture the stochastic dynamics exhibited by asset trajectories. Both models belong to the category of Lévy processes and are characterized by distinct sets of model parameters. Comprehensive insights and theorems pertaining to Lévy processes can be referenced in [10].

The acronym 'CGMY' derives from the initials of its creators: Carr, Geman, Madan, and Yor. The original formulation of this model was introduced in their seminal paper [11] in 2002. This paper provides an in-depth elucidation of the rationale underlying the model's parameterization.

In this study, we extend our analysis beyond the conventional CGMY model to encompass . This extension incorporates the utilization of a positive volatility, adding an extra layer of sophistication to the model's representation.

Definition 2.1.6. *(Lévy process). A Lévy process is a stochastic process $X = \{X_t : t \geq 0\}$ that satisfies the following properties:*

1. *Initial condition: $X_0 = 0$ almost surely.*
2. *Independence of increments: For any $0 \leq t_1 < t_2 < \ldots < t_n < \infty$, the random variables $X_{t_2} - X_{t_1}, X_{t_3} - X_{t_2}, \ldots, X_{t_n} - X_{t_{n-1}}$ are mutually independent.*
3. *Stationary increments: For any $0 \leq s < t$, the increment $X_t - X_s$ has the same distribution as X_{t-s}.*
4. *Continuity in probability: For any $\epsilon > 0$ and $t \geq 0$, it holds that $\lim_{h \to 0} \mathbb{P}(|X_{t+h} - X_t| > \epsilon) = 0$.*

If X is a Lévy process, it is possible to construct a version of X such that the mapping $t \mapsto X_t$ is almost surely right-continuous with left limits.

Definition 2.1.7. *(Characteristic function). Given a stochastic process X and a time t, the characteristic function of the stochastic process at time t is given by:*

$$\varphi_X(u,t) := \mathbb{E}\left[e^{iuX(t)}\right].$$

Moreover, when conditional on the initial value $X(0) = x$ it will also be written with the regular ϕ as:

$$\phi_X(u,t;x) := \mathbb{E}\left[e^{iuX(t)} \mid x\right].$$

Theorem 2.1.4. *(Lévy-Khintchine). The distribution of a Lévy process $X = \{X_t, t \geq 0\}$ is characterized by its characteristic function, given by the Lévy-Khintchine formula:*

$$\varphi_X(u,t) = \mathbb{E}\left[e^{iuX(t)}\right] = \exp\left(t\left(\mu i u - \frac{1}{2}\sigma^2 u^2 + \int_{\mathbb{R}\setminus\{0\}} \left(e^{iux} - 1 - iux\mathbf{1}_{|x|<1}\right)\Pi(dx)\right)\right)$$

where $\mu \in \mathbb{R}$, $\sigma \geq 0$, and Π is a σ-finite measure called the Lévy measure of X, satisfying the property:

$$\int_{\mathbb{R}\setminus\{0\}} \min(1, x^2)\,\Pi(dx) < \infty.$$

This theorem suggests that every Lévy process is uniquely determined by its ch. f. and hence by a Lévy-triplet (μ, σ, Π), where μ represents the drift, σ the diffusion (Brownian motion) and Π the Lévy jump process.

Theorem 2.1.5. *(Lévy relation ch. f.). Given a Lévy process $X = \{X_t, t \geq 0\}$, with $X_0 = x$. The ch. f. of X given initial value follows the relation:*

$$\phi(u, t; x) = \phi(u, t; 0)e^{iux} = \varphi_{levy}(u)e^{iux}, \tag{2.1}$$

where $\phi(u, t; x) := \mathbb{E}[e^{iuX(t)} \mid X_0 = x]$.

Definition 2.1.8. *(GBM Model). A real-valued process $X = \{X_t, t \geq 0\}$ is called a Geometric Brownian Motion (GBM) with drift μ and volatility σ if it satisfies the SDE:*

$$dX(t) = \mu X(t)dt + \sigma X(t)dW(t).$$

The model parameters are in the following domain: $\mu \in \mathbb{R}$ and $\sigma > 0$. Under the risk-neutral measure the drift is defined as $\mu = r - \frac{1}{2}\sigma^2$, where r is the risk-free interest rate. The process follows a log-normal distribution, since the process $\log(X)$ follows a normal distribution.

Theorem 2.1.6. *The ch. f. of the GBM-model for a time t is given by:*

$$\phi_{GBM}(u; t) = \exp\left(iu\mu t - \frac{1}{2}\sigma^2 u^2 t\right).$$

Definition 2.1.9. *(CGMY-model). A real-valued process $X = \{X(t), t \geq 0\}$ follows the CGMY-dynamics, with model parameters (C, G, M, Y, σ_B) and given a risk-free interest rate r, if the following requirements hold:*

1. *The drift is given by*

$$\mu = r - \frac{1}{2}\sigma_B^2 + \bar{\omega},$$

where the drift correction term is given by

$$\bar{\omega} = -C\Gamma(-Y)\left[(M-1)^Y - M^Y + (G+1)^Y - G^Y\right],$$

2. *the diffusion term is given by $\sigma_B > 0$,*

3. *the Lévy density is defined to be*

$$f_{CGMY}(y) = C\left[\frac{e^{-M|y|}}{|y|^{1+Y}}1_{y>0} + \frac{e^{-G|y|}}{|y|^{1+Y}}1_{y<0}\right]$$

The model parameters are in the following domain: $C \geq 0, G \geq 0, M \geq 0, Y \leq 2$ and $B > 0$.

Theorem 2.1.7. *The ch. f. of the CGMYB-model for a time t is given by*

$$\phi_{CGMYB}(u,t) = \exp\left(iu\mu t - \frac{1}{2}\sigma_B^2 u^2 t\right) \cdot \phi_{CGMY}(u,t),$$

where

$$\phi_{CGMY}(u,t) = \exp\left(Ct\Gamma(-Y)\left[(M-iu)^Y - M^Y + (G+iu)^Y - G^Y\right]\right).$$

The drift and the drift correction term are defined as in Definition (2.1.9). Note that when $C = 0$ the ch. f. is the same as for the GBM model, which implies by uniqueness that GBM is a special case of CGMY.

2.2. PROBABILITY THEORY

Probability theory provides a systematic framework for analyzing and quantifying the likelihood of events occurring in various contexts. In option pricing these events are the price paths, from which the option prices can be derived.

Definition 2.2.1. *For a random variable X, the Cumulative Density Function (CDF) is given by:*

$$F_X(x) = \mathbb{P}(X \leq x),$$

which is also often known as just the probability distribution function.

The Probability Density Function (PDF) for a continuous random variable is given by:

$$f_X(x) = \frac{d}{dx} F_X(x).$$

Definition 2.2.2. *For a continuous random variable X, the expectation is given by:*

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x \cdot f_X(x)\, dx \quad (2.3),$$

where $f_X(x)$ is the PDF of X. The variance of X is then given by:

$$\mathbb{V}[X] = \int_{-\infty}^{\infty} (x - \mathbb{E}[X])^2 \cdot f_X(x)\, dx.$$

Definition 2.2.3. *We say that X is a univariate normally distributed with expectation μ and variance σ^2, denoted as $X \sim \mathcal{N}(\mu, \sigma^2)$, when the CDF is given by:*

$$F_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right) du.$$

This implies that the PDF is given by:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Definition 2.2.4. *We say that a d-dimensional vector $\boldsymbol{X}$ follows a multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, denoted as $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, if the PDF is given by:*

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right),$$

2.3. MATHEMATICS BEHIND OPTION PRICING

In this section, we lay the foundation for option pricing by introducing essential mathematical notations, alongside key definitions and underlying theories. These elements will serve as the cornerstone upon which we will develop methods for pricing barrier options. The notational framework and definitions are drawn from [8] and [12].

Options are a versatile financial instrument derived from an underlying asset. This asset path is assumed to be an adapted process S. Options grant holders the *right*, but not the obligation, to buy or sell the asset at a predetermined *strike* price K within a specific timeframe, which is determined by the *maturity* T.

When dealing with options the holder, who buys or shorts, the option has to decide the *side*, which is the type of option. For options one can choose to trade in *call* or *put* options, which are defined below.

Definition 2.3.1. *(European Call). A European call option is a financial contract that gives the holder the right, but not the obligation, to buy a specific underlying asset S from the issuer at a predetermined strike K at the expiration date T.*

Definition 2.3.2. *(European Put). A European put option is a financial contract that gives the holder the right, but not the obligation, to sell a specific underlying asset S from the issuer at a predetermined strike K at the expiration date T.*

At maturity T the holder has to decide whether to exercise the option or not. The holder will only exercise if it is favorable for him (no loss). To write this mathematically for the two different European option types; the payoff of the option can be written as.

Definition 2.3.3. *(Payoff European options). Given an adapted asset process S, maturity T and strike K. The payoff at maturity for a European call and put is given by:*

$$\Lambda^{EU}_{call}(T,S) := \max(S(T)-K,0) := (S(T)-K)^+$$
$$\Lambda^{EU}_{put}(T,S) := \max(K-S(T),0) := (K-S(T))^+.$$

In the financial world also options exist that can be exercised before maturity. The two most popular ones are the *American* style options and *Bermudan* style options, which employ the same payoff function as the European style options. For American options the holder can choose to exercise the options at $0 \le t \le T$. Bermudan options can only be exercised at M pre-specified times $0 \le t_1 \le t_2 \le \cdots \le t_M = T$. This option style is in between the American and European style options. In this paper only options are considered, which can only be exercised at maturity. This includes European options, but also other exotic options.

Exotic options have a more complicated payoff function. These options were introduced in the financial market, since they can have more favorable attributes for the holder than European options. Examples of European options are Asian, Lookback and Basket options. In this paper we only focus on the exotic options called *barrier options*. Barrier options are a type of derivative contract where the payoff and activation of the option depend on whether the underlying asset's price reaches or crosses a predefined barrier level H during the option's lifespan. These barrier levels can be specified as knock-outs (deactivate option) or lower barriers knock-ins (activate option). Investors

may choose to buy barrier options instead of European options for reasons such as lower upfront premiums, customized risk-return profiles, enhanced profit potential, risk management benefits, and opportunities in volatile markets. The definitions on knock-out and knock-in barrier options are given below. In theory there are a lot more different types of barrier options, but in this paper only knock-out and knock-in are considered since these are the most liquid in the real world.

Definition 2.3.4. *(Knock-out option). A knock-out option is a type of option contract that becomes deactivated or "knocked out" if the underlying asset's price reaches or crosses a predefined barrier level during the option's lifespan. Once the barrier level is breached, the option immediately loses its value and ceases to exist.*
One refers to an up-and-out option if the barrier level is above the initial asset price, and a down-and-out option if it is below the barrier level.

Definition 2.3.5. *(Knock-in option). A knock-in option is a type of option contract that becomes activated or "knocked in" if the underlying asset's price reaches or crosses a predefined barrier level during the option's lifespan. Once the barrier level is breached, the option becomes active and starts providing payoff.*
One refers to an up-and-in option if the barrier level is above the initial asset price, and a down-and-in option if it is below the barrier level.

When the barrier option is still activated, it pays off as an European option at maturity T. This means the holder still has the right to exercise if it is profitable. To define the payoff of these two types of barrier options we define:

$$S^{\max} := \max_{[0,T]} S(t) \quad \text{and} \quad S^{\min} := \min_{[0,T]} S(t).$$

Definition 2.3.6. *(Payoff knock-out and knock-in barrier options). Given an adapted asset process S, maturity T, strike K, side (call or put) and barrier level H. The payoff is given by:*

- *Up-and-out (UO) option:* $\Lambda^{UO}_{side}(T,S) := \Lambda^{EU}_{side}(T,S) \cdot \mathbf{1}_{\{S^{\max} \leq H\}}$
- *Down-and-out (DO) option:* $\Lambda^{DO}_{side}(T,S) := \Lambda^{EU}_{side}(T,S) \cdot \mathbf{1}_{\{S^{\min} \geq H\}}$
- *Up-and-in (UI) option:* $\Lambda^{UI}_{side}(T,S) := \Lambda^{EU}_{side}(T,S) \cdot \mathbf{1}_{\{S^{\max} \geq H\}}$
- *Down-and-in (DI) option:* $\Lambda^{DI}_{side}(T,S) := \Lambda^{EU}_{side}(T,S) \cdot \mathbf{1}_{\{S^{\min} \leq H\}}$,

In the field of option pricing or in general the quantitative finance field, fair option pricing for any time $0 \leq t < T$ is of utmost importance due to the principle of no-arbitrage. This principle ensures that there is no possibility to make a risk-free profit in the market that exceeds the risk-free return on a savings account at the bank. The growth of this savings account is determined by the risk-free interest rate r. An arbitrage is present if the following criteria hold:

Definition 2.3.7. *(Arbitrage). An investment strategy is called an arbitrage if the value process V of the strategy satisfies the following two properties:*

1. *Non-zero probablity of winning:*

$$\mathbb{P}(V_T > (1+r)V_0) > 0$$

2. *Zero probability of losing:*

$$\mathbb{P}(V_T \geq (1+r)V_0) = 1,$$

where T is the maturity and r denotes the risk-free interest rate.

In our research, options values are used to benchmark and fit our models to the correct values, in order to find the relation between the input parameters and the option value. One of the most important results in the field of option pricing regarding no arbitrage is the discovery of the Black-Scholes pricing PDE by Black, Scholes and Merton from their well-known groundbraking paper [13] . This pricing PDE describes the relation between the option value process V and the time t, asset price S, volatility σ^2 and risk-free interest rate r. The celebrated formula is as follows:

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rV = 0 \tag{2.2}$$

This expression can be derived using the Feynman-Kac theorem stated in [14], but originally this was derived using the theory of *replication*. This concept essentially mimic the payoff of the option using a portfolio of different assets, and setting the value of these portfolios equal to suffice the principles of no-arbitrage. For European options the solution of V to this PDE is analytically known. The value for an option at time t before maturity and initial price $S(t)$ is given by

$$V_{call}^{EU}(t,S) = S(t)N(d_1) - Ke^{-r(T-t)}N(d_2), \tag{2.3}$$

$$V_{put}^{EU}(t,S) = -S(t)N(-d_1) + Ke^{-r(T-t)}N(-d_2), \tag{2.4}$$

where

$$d_1 = \frac{\log(S(t)/K) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \tag{2.5}$$

$$d_2 = \frac{\log(S(t)/K) + \left(r - \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} = d_1 - \sigma\sqrt{T-t}, \tag{2.6}$$

and the function $N(\cdot)$ denotes the cumulative density function of a standard normal distributed variable. Furthermore, for barrier options, under GBM dynamics the expression

for the value process V is known. Down below the analytical formula for pricing up-and-out barrier calls is stated [12]:

$$V_{call}^{UO}(t,S) = S(t)\left(N(d_1) - N(e_1) - \left(\frac{H}{S(t)}\right)^{1+2r/\sigma^2}(N(f_2) - N(g_2))\right) \tag{2.7}$$

$$- Ke^{-r(T-t)}\left(N(d_2) - N(e_2) - \left(\frac{H}{S(t)}\right)^{-1+2r/\sigma^2}(N(f_1) - N(g_1))\right) \tag{2.8}$$

Here the values d_1, d_2 are defined in (2.5) and(2.6) and the other coordinates are defined as:

$$e_1 = \frac{\log(S(t)/H) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \tag{2.9}$$

$$e_2 = \frac{\log(S(t)/H) + (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \tag{2.10}$$

$$f_1 = \frac{\log(S(t)/H) - (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \tag{2.11}$$

$$f_2 = \frac{\log(S(t)/H) - (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \tag{2.12}$$

$$g_1 = \frac{\log(S(t)K/H^2) - (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \tag{2.13}$$

$$g_2 = \frac{\log(S(t)K/H^2) - (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}. \tag{2.14}$$

The pricing of options using the GBM model is a trivial process due to the availability of analytical expressions for option values. As a result, computations can be performed instantly. However, for general Lévy processes such as CGMY, these analytical expressions are unavailable, necessitating the application of alternative techniques to accurately determine value data for our customized models. The subsequent sections of this chapter, specifically Sections 2.5 and 2.6, will delve into the methodologies employed to address this challenge.

2.4. SINGULAR VALUE DECOMPOSITION (SVD)

Within the realm of linear algebra, matrices that deviate from the realm of normality present intriguing challenges. While normal matrices, such as real-symmetric or Hermitian matrices, readily yield to Spectral Decomposition (SD), enabling them to be transformed into diagonal matrices via unitary or orthogonal transformations, non-normal matrices lack this convenient property. Instead, an illustrious factorization technique known as Singular Value Decomposition (SVD) comes to the forefront, extending the benefits of orthogonal decomposition to the non-normal matrix domain [15].

In contrast to SD's restriction to normal matrices and its focus solely on eigenvalues and eigenvectors, SVD takes a more comprehensive approach. It offers a broader factorization framework, capable of handling rectangular matrices and encompassing both singular values and singular vectors. This versatility significantly broadens its scope, making SVD a more general factorization method than SD.

In the context of this paper, SVD proves invaluable for efficiently approximating training outputs in a machine learning model. Further insights into this decomposition technique can be gleaned from [16] or [17]. A pivotal theorem within SVD can be stated as follows:

Theorem 2.4.1. *(SVD). Let* $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ *with* $m \geq n$. *Then* $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ *where:*

- $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ *is an orthogonal matrix,*
- $\boldsymbol{\Sigma} = diag(\sigma_1, ..., \sigma_n) \in \mathbb{R}^{m \times n}$ *with* $\sigma_1 \geq ... \geq \sigma_n \geq 0$ *and* $\sigma_i \in \mathbb{R}$ *for all* i,
- $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ *is an orthogonal matrix.*

The rank of the matrix has value $r = \min\{l \mid \sigma_l > 0\}$, *which is the singular value with the smallest index that exceeds 0.*

In SVD these matrices are referred to as:

- The column vectors of $\boldsymbol{U} = [\boldsymbol{u}_1 ... \boldsymbol{u}_m]$ are called the *left singular vectors* of $\boldsymbol{A}$,
- The column vectors of $\boldsymbol{V} = [\boldsymbol{v}_1 ... \boldsymbol{v}_n]$ are called the *right singular vectors* of $\boldsymbol{A}$,
- The values $\{\sigma_1, ..., \sigma_n\}$ are called the *non-singular values* of $\boldsymbol{A}$.

CONSTRUCTION OF THE SVD MATRICES

Similarly to the spectral decomposition for normal matrices, the SVD decomposes a matrix into three simpler matrices the matrices for SVD. These matrices, just as in SD, constructed using eigenvalues and eigenvectors. The only difference is that for SD only one set of eigenvectors and eigenvalues is needed, while for SVD two (mostly different) sets of eigenvectors and eigenvalues have to be calculated.
For a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with $m \geq n$, the SVD factorization is calculated using the following steps:

1. Calculate the matrix $\boldsymbol{A}^T\boldsymbol{A}$, which results in an $n \times n$ matrix.
2. Find the eigenvalues and eigenvectors of the matrix obtained in step 1. This can be done using various numerical methods, such as the Gram-Schmidt (GS) or the QR algorithm, which aim to find a set of mutually orthogonal eigenvectors.
3. The eigenvectors obtained in step 2 correspond to the right singular vectors of $\boldsymbol{A}$, stored as the columns of the matrix $\boldsymbol{V}$.

4. Calculate the singular values, which are the square roots of the eigenvalues obtained in step 2. Arrange them in a diagonal matrix $\boldsymbol{\Sigma}$, with non-negative singular values in descending order along the diagonal, i.e. $\boldsymbol{\Sigma}_{ii} = \sigma_i \geq 0$.

5. Calculate the left singular vectors by normalizing the columns of the matrix $\boldsymbol{U}$. Each column of $\boldsymbol{U}$ is obtained by dividing the corresponding eigenvector of $\boldsymbol{A}^T\boldsymbol{A}$ by the square root of the corresponding singular value.

The resulting factorization is the celebrated SVD factorization:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{2.15}$$

The matrix $\boldsymbol{U} \in \mathbb{R}^{m\times m}$ containing the left singular vectors, the matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ is a diagonal matrix containing the singular values, and the matrix $\boldsymbol{V}^T \in \mathbb{R}^{n\times n}$ containing the right singular vectors.

2.4.1. SVD FOR SOLVING LINEAR SYSTEMS

In this paper, SVD is employed to achieve a 'best fit' solution for a system of linear equations. This approach is particularly powerful for addressing linear systems, especially when dealing with overdetermined scenarios, where the number of equations (m) surpasses the count of unknowns (n). This application bears resemblance to multidimensional regression fitting, wherein SVD is employed to determine the Least-squares (LS) solution of the system.

Let's consider a linear system represented by the matrix equation:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.16}$$

Here, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{x}$ is an n-dimensional vector of unknowns, and $\boldsymbol{b} \in \mathbb{R}^m$. The LS solution $\boldsymbol{x}_{LS}$ aims to minimize the Euclidean distance between the estimated solution $\boldsymbol{A}\boldsymbol{x}_{LS}$ and the vector of constants $\boldsymbol{b}$. This leads to an unconstrained optimization problem:

$$\boldsymbol{x}_{LS} = \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2. \tag{2.17}$$

To derive the LS solution using SVD, the following steps are undertaken:

1. Compute the SVD of matrix A:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

2. Calculate the pseudoinverse of $\boldsymbol{\Sigma}$, denoted as $\boldsymbol{\Sigma}^\dagger$, by taking the reciprocal of each non-zero singular value and transposing the resultant matrix.

   The pseudoinverse of $\boldsymbol{\Sigma}$ is given by:

$$\boldsymbol{\Sigma}^\dagger = \begin{bmatrix} \frac{1}{\sigma_1} & & \\ & \ddots & \\ & & \frac{1}{\sigma_n} \end{bmatrix}$$

   where $\sigma_1, \sigma_2, \ldots, \sigma_n$ are the singular values of $\boldsymbol{A}$.

3. Compute the LS solution $\boldsymbol{x}_{\text{LS}}$ using the formula:

$$\boldsymbol{x}_{LS} = \boldsymbol{V}\boldsymbol{\Sigma}^{\dagger}\boldsymbol{U}^{T}\boldsymbol{b}$$

Here, $\boldsymbol{x}_{LS}$ is the vector of least square solutions, representing the solution to the minimization problem described by equation (2.17).

By leveraging SVD and its associated steps, this approach offers an effective means of obtaining robust solutions for linear systems, particularly in scenarios where an overdetermined system needs to be addressed.

2.5. COS METHOD

The COS method, initially presented by Fang and Oosterlee in [5], has found its application in the field of option valuation. Its fundamental principle involves the reconstruction of the unknown probability density function using a Fourier-cosine series expansion. The key insight behind this method lies in the fact that the series coefficients can be obtained from the characteristic function, which is often more readily derived than the density function itself.

In the context of option valuation, the COS method offers a valuable approach to estimate the prices of options. The ch.f. provides essential information about the underlying asset's behavior. Combining this with the option payoff results in accurate and fast calculations of various option contracts.

2.5.1. COSINE SERIES EXPANSION

The COS method is classified as a Fourier-based method, which belongs to a family of techniques relying on the relationship between the density function and its corresponding ch.f.. Specifically, the density function, denoted as $f(x)$, and the ch.f., denoted as $\phi(\omega)$, form a Fourier pair:

$$\phi(\omega) = \int_{-\infty}^{\infty} f(x)e^{i\omega x}dx \tag{2.18}$$

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \phi(\omega)e^{-i\omega x}dx. \tag{2.19}$$

The COS method utilizes the Fourier-cosine series expansion to recover the density function using a approximation of the integral (2.19). It is worth noting that any real function with finite support can be represented by a cosine expansion. Additionally, the Fourier-cosine series expansion provides an optimal approximation for functions with a finite support, as indicated in [18]. The first step to derive the COS method is to truncate the integral representation of the density function of the underlying process in (2.19). An important note is that the function has to decay to zero at both $\pm\infty$ in order for the truncation to be accurate.

For this infinitely support density function f assume that the truncation range is given by the closed interval $[a, b] \in \mathbb{R}$. This defines the cosine representation of the function to be:

$$f(x) = \sum_{k=0}^{\prime\infty} A_k \cos\left(k\pi \frac{x-a}{b-a}\right), \tag{2.20}$$

where the cosine series coefficients are defined by

$$A_k = \frac{2}{b-a}\int_a^b f(x)\cos\left(k\pi\frac{x-a}{b-a}\right)dx. \tag{2.21}$$

The notation $\sum'$ means that the first term of the sum is multiplied by $1/2$. The accuracy loss of the density on this truncated interval $[a,b]$ is very small and hence also the error of the truncated integral of the ch. f. in (2.18). This is important, since there is a link between the Fourier-cosine series coefficients A_k and the ch. f. Looking at the following approximation of the ch. f.

$$\phi_1(\omega) := \int_a^b f(x)e^{i\omega x}dx \approx \int_{-\infty}^{\infty} f(x)e^{i\omega x}dx = \phi(\omega), \tag{2.22}$$

and comparing it to (2.21) arises the expression for the cosine series coefficients:

$$A_k := \frac{2}{b-a}\mathrm{Re}\left\{\phi_1\left(\frac{k\pi}{b-a}\right)\exp\left(-i\frac{k\pi a}{b-a}\right)\right\}. \tag{2.23}$$

Now replacing the truncated ch. f. with the real ch. f. (2.22) gives the approximation $A_k \approx F_k$ of the cosine series coefficients :

$$F_k := \frac{2}{b-a}\mathrm{Re}\left\{\phi\left(\frac{k\pi}{b-a}\right)\exp\left(-i\frac{k\pi a}{b-a}\right)\right\}, \tag{2.24}$$

which gives the following approximation of the density function when combining it with a truncated sum:

$$f(x) \approx \sum_{k=0}^{\prime\infty} F_k \cos\left(k\pi\frac{x-a}{b-a}\right) \approx \sum_{k=0}^{\prime N-1} F_k \cos\left(k\pi\frac{x-a}{b-a}\right). \tag{2.25}$$

Here N describes the number of basis functions used to estimate the function. This summation is the density approximation. This recovery of the density using the ch. f. is the result of the COS method.

To use the COS method for option pricing, the approximated density is used in the discounted expectation of the payoff. That is given some initial log-asset price x, time t, maturity T, strike K and risk-free interest rate r:

$$v(x,t) = e^{-r\Delta t}\int_{-\infty}^{\infty} v(y,T)f(y|x)dy, \tag{2.26}$$

where $v(y,T)$ is the value of the option with log-asset price y at T, i.e. the payoff. The first approximation of this integral, denoted $v_1(x,t)$, is given by the truncation on $[a,b]$:

$$v_1(x,t) \approx e^{-r\Delta t}\int_a^b v(y,T)f(y\mid x)dy \tag{2.27}$$

$$= e^{-r\Delta t}\int_a^b v(y,T)\sum_{k=0}^{\prime\infty} A_k \cos\left(k\pi\frac{x-a}{b-a}\right)dy \tag{2.28}$$

$$= \frac{b-a}{2}e^{-r\Delta t}\sum_{k=0}^{\prime\infty} A_k(x)V_k, \tag{2.29}$$

where the interchanging of integral and summation is used and the payoff series coefficients V_k are defined as:

$$V_k = \frac{2}{b-a}\int_a^b v(y,T)\cos\left(k\pi\frac{y-a}{b-a}\right)dy, \tag{2.30}$$

which are the cosine series coefficients of $v(y,T)$ in y. Truncation this sum using N cosine basis functions gives the next approximation

$$v_2(x,t) = \frac{b-a}{2}e^{-r\Delta t}\sum_{k=0}^{\prime N-1} A_k(x)V_k, \tag{2.31}$$

and the final approximation uses the estimation of the cosine coefficients F_k in (2.24):

$$v(x,t) \approx v_3(x,t) = e^{-r\Delta t}\sum_{k=0}^{\prime N-1}\mathrm{Re}\left\{\phi\left(\frac{k\pi}{b-a};x\right)\exp\left(-i\frac{k\pi a}{b-a}\right)\right\}V_k. \tag{2.32}$$

For European options the payoff series coefficients V_k can be calculated analytically. Throughout the paper these are essential for option pricing and hence the derivation will be stated below.

For European options using the log-asset prices $x = \log(S_0/K)$ and $y = \log(S_T/K)$ the payoff is $v(y,T) = [\alpha K(e^y - 1)]^+$, where α is 1 for a call option and -1 for a put option. They are the following:

$$V_k^{call} = \frac{2}{b-a}K(\chi_k(0,b) - \psi_k(0,b)) \tag{2.33}$$

$$V_k^{put} = \frac{2}{b-a}K(-\chi_k(a,0) + \psi_k(a,0)). \tag{2.34}$$

Here the expressions $\chi_k(c,d) := \int_c^d e^y\cos\left(k\pi\frac{y-a}{b-a}\right)dy$ and $\psi_k(c,d) := \int_c^d \cos\left(k\pi\frac{y-a}{b-a}\right)dy$ are for a given subinterval $[c,d] \subset [a,b]$ derived, using integration by parts, to be:

$$\chi_k(c,d) = \frac{1}{1+\left(\frac{k\pi}{b-a}\right)^2}\left\{e^d\cos\left(k\pi\frac{d-a}{b-a}\right) - e^c\cos\left(k\pi\frac{c-a}{b-a}\right)\right. \tag{2.35}$$

$$\left. + \frac{k\pi}{b-a}e^d\sin\left(k\pi\frac{d-a}{b-a}\right) - \frac{k\pi}{b-a}e^c\sin\left(k\pi\frac{c-a}{b-a}\right)\right\} \tag{2.36}$$

and

$$\psi_k(c,d) = \begin{cases} \frac{b-a}{k\pi}\sin\left(k\pi\frac{d-a}{b-a}\right) - \frac{b-a}{k\pi}\sin\left(k\pi\frac{c-a}{b-a}\right) & ,k \neq 0 \\ d-c & ,k = 0 \end{cases} \tag{2.37}$$

INTEGRATION RANGE $[a,b]$

Accurate option price prediction through the COS method hinges on carefully defining integration bounds. Extensive bounds lead to higher basis function usage (N), increasing complexity. Narrow bounds, however, amplify truncation-induced error. [5] suggests bounds based on:

$$[a,b] = \left[c_1 - L\sqrt{c_2 + \sqrt{c_4}}, c_1 + L\sqrt{c_2 + \sqrt{c_4}}\right], \tag{2.38}$$

with $L = 10$. Coefficients $(c_i)_{i=1,2,4}$, called cumulants, depend on contract and model parameters. These cumulants are detailed in [5] and [8] for discussed models and others. For GBM and CGMY models used here, these cumulants are listed in the Appendix A.2.

2.5.2. COS METHOD UNDER LÉVY PROCESSES

This paper focuses exclusively on Lévy processes, which provide a robust foundation for the research conducted herein. The pivotal reason for this choice lies in the ch.f. ϕ of a general Lévy process $(X(t))_{t\geq 0}$, which follows a vital decomposition as expressed in equation (2.1):

$$\phi(\omega;\boldsymbol{x}) = \phi(\omega;0)\cdot e^{i\omega\boldsymbol{x}} = \varphi_{levy}(\omega)\cdot e^{i\omega\boldsymbol{x}}, \tag{2.39}$$

where $\varphi_{levy}(\omega) := \phi(\omega;0)$. This remarkable property allows the extraction of the initial value $X_0 = \boldsymbol{x}$ from the ch.f., a feature particularly advantageous for option pricing under Lévy processes using formula (2.32). Consequently, equation (2.39) can be reformulated as:

$$v(x,t) \approx v_3(x,t) = e^{-r\Delta t}\sum_{k=0}^{\prime N-1} \mathrm{Re}\left\{\varphi_{levy}\left(\frac{k\pi}{b-a}\right)\exp\left(ik\pi\frac{x-a}{b-a}\right)\right\} V_k. \tag{2.40}$$

It's noteworthy that when evaluating the option value under a Lévy process, the initial log-asset price x can be extracted from the function φ_{levy}, thus reducing the input dimension of this function by one. This reduction proves highly advantageous, as elucidated later in this paper during the process of estimating the values $\varphi_{levy}\left(\frac{k\pi}{b-a}\right)$.

2.6. COS METHOD FOR PRICING DISCRETE BARRIER OPTIONS

A follow up on the paper of [5] about the traditional COS method is another paper by Fang and Oosterlee [1] about pricing Bermudan and discretely monitored barrier options using Fourier-cosine series. Our research only focuses on barrier options, in which from now will be referred to this method as the COS Barrier method. The pricing of barrier options depends on a survival event. This event triggers the option to be in value of out of value.

In this section only the knock-out event is treated, and specifically the up-and-out barrier option. This means that the option is worthless if on the M monitoring dates $\mathcal{T} = \{t_m : 1 \leq m \leq M_{mon}\}$, with $t_{M_{mon}} = T$, the underlying stock process $(S_t)_{t\geq 0}$ is above the barrier level H. The option can only be exercised at maturity T. Equivalently this means that the payoff is:

$$v(x,\min(T,\tau)) = (\alpha(S_T - K))^+ \cdot 1_{\tau>T}, \tag{2.41}$$

where $\tau := \inf\{t_m \geq t_0 : S_{t_m} \geq H\}$ is the first monitoring date that registers a breach of S_t on the barrier level H. Thus the indicator $1_{\tau>T}$ becomes zero if the barrier level is crossed at one of the monitoring dates. Furthermore, x is the log-asset price and $\alpha = 1$ for a call option and -1 for a put option. Note that for the other knock-in and out options only the $1_{\tau>T}$ or τ has to be changed. The following derivations will hence be analogous for the other barrier options and will not be treated seperately.

RECURSIVE RELATION CONTINUATION VALUE

The COS Barrier method uses M_{mon} recursive steps in order to find the initial value of the barrier option. It initialises the option value at $t_{M_{mon}} = T$, since there the value is known (payoff). After that it iterates backwards in time to calculate the option value at one monitoring date prior. This is done until the initial time is reached. Given the set of monitoring dates $\mathcal{T} = \{t_m : 1 \le m \le M\}$ the following recursive relation can be derived for the price of the discretely monitored barrier options at the M monitoring dates.

- For $m = M_{mon}, M_{mon}-1, M_{mon}-2, \ldots, 2$:

$$\begin{cases} c(x,t_{m-1}) & = e^{-r(t_m - t_{m-1})} \int_{-\infty}^{\infty} v(x,t_m) f(y \mid x)\, dy, \\ v(x,t_{m-1}) & = \begin{cases} 0, & \text{if } x \ge h \\ c(x,t_{m-1}), & \text{if } x < h \end{cases} \end{cases} \tag{2.42}$$

  inside the recursive formula:

  - $c(x,t_{m-1})$ is the continuation value at t_{m-1},
  - $v(x,t_{m-1})$ is the option value at t_{m-1} ,
  - $f(y \mid x)$ is the PDF of y given x ,
  - $x = \log(S_{t_{m-1}}/K), y = \log(S_{t_m}/K)$ and $h = \log(H/K)$.

- For $m = 1$:

$$v(x,t_0) = e^{-r(t_1 - t_0)} \int_{-\infty}^{\infty} v(x,t_1) f(y \mid x)\, dy, \tag{2.43}$$

  which represents the barrier option value at initial time t_0.

APPLICATION OF THE COS METHOD

Calculating the integrals of the continuation value in (2.42) and option value in (2.43) will be done using the COS method from Section 2.5. That is, the continuation value $c(x,t_{m-1})$ is estimated using

$$c(x,t_{m-1}) \approx \hat{c}(x,t_{m-1}) = e^{-r\Delta t} \sum_{k=0}^{\prime N-1} F_k(x) V_k(t_m), \tag{2.44}$$

where a truncation range $[a,b]$ and number of basis function N is defined. The coefficients $F_k(x)$ is defined as (2.24). Here $V_k(t_m)$ is the Fourier-cosine series of $v(y,t_m)$ i.e.:

$$V_k(t_m) = \frac{2}{b-a} \int_a^b v(y,t_m) \cos\left(k\pi \frac{y-a}{b-a}\right) dy. \tag{2.45}$$

Then the value of the option $v(x,t_{m-1})$ is estimated iteratively by

$$v(x,t_{m-1}) \approx \hat{v}(x,t_{m-1}) = \begin{cases} 0, & \text{if } x \ge h \\ \hat{c}(x,t_{m-1}), & \text{if } x < h \end{cases}. \tag{2.46}$$

For the calculation of $c(x,t_{m-2})$ another layer of approximation is needed, because of the definition of $V_k(t_{m-1})$ in (2.45). This is calculated using $v(x,t_{m-1}) \approx \hat{v}(x,t_{m-1})$. Hence this continuation value is calculated by:

$$\hat{V}_k(t_m) := \frac{2}{b-a}\int_a^b \hat{v}(y,t_m)\cos\left(k\pi\frac{y-a}{b-a}\right)dy, \tag{2.47}$$

$$c(x,t_{m-1}) \approx e^{-r\Delta t}\sum_{k=0}^{\prime N-1} F_k(x)\hat{V}_k(t_m). \tag{2.48}$$

SOLVING FOR $\hat{V}_k$ USING BACKWARDS INDUCTION

The main insight is that the coefficients $\hat{V}_k(t_m)$ are solvable using backwards induction. That is by (2.46):

$$\hat{V}_k(t_m) = \frac{2}{b-a}\int_a^b \hat{v}(y,t_m)\cos\left(k\pi\frac{y-a}{b-a}\right)dy \tag{2.49}$$

$$= \frac{2}{b-a}\int_a^h \hat{c}(y,t_m)\cos\left(k\pi\frac{y-a}{b-a}\right)dy, \tag{2.50}$$

where $\hat{C}_k(x_1,x_2,t_m) := \frac{2}{b-a}\int_{x_1}^{x_2}\hat{c}(x,t_m)\cos\left(k\pi\frac{y-a}{b-a}\right)dy$. Hence $\hat{V}_k(t_m) = \hat{C}_k(a,h,t_m)$. This means after some calculations that the $\hat{V}_k(t_m)$ are calculated as follows:

- For the terminal condition: $m = M_{mon}$, it is given that $\hat{V}_k(t_{M_{mon}}) = V_k(t_{M_{mon}})$, which is defined as:
  - For $h < 0$:
$$V_k(t_{M_{mon}}) = \begin{cases} 0, & \text{for call option} \\ G_k(a,h), & \text{for put option} \end{cases}, \tag{2.51}$$
  - For $h \geq 0$:
$$V_k(t_{M_{mon}}) = \begin{cases} G_k(0,h), & \text{for call option} \\ G_k(a,0), & \text{for put option} \end{cases} \tag{2.52}$$
    - Here the notation G_k is defined as (2.30). This notation is used to prevent confusion with the other value coefficients $V_k(t_m)$.
- The derivation of the expression $\hat{C}_k$ is a matrix-vector product:
$$\hat{\boldsymbol{C}}(x_1,x_2,t_m) = \frac{e^{-r\Delta t}}{\pi}\text{Im}\{(\mathcal{M}_c + \mathcal{M}_s)\boldsymbol{u}\}. \tag{2.53}$$
  - The elements of the matrices $\mathcal{M}_c$ and $\mathcal{M}_s$ are defined as:
$$(\mathcal{M}_c)_{k,l} := \begin{cases} \frac{(x_2-x_1)\pi i}{b-a}, & \text{if } k=l=0, \\ \frac{\exp\left(i(k+l)\frac{(x_2-a)\pi}{b-a}\right)-\exp\left(i(k+l)\frac{(x_1-a)\pi}{b-a}\right)}{k+l}, & \text{otherwise} \end{cases} \tag{2.54}$$
    and (2.55)
$$(\mathcal{M}_s)_{k,l} := \begin{cases} \frac{(x_2-x_1)\pi i}{b-a}, & \text{if } k=l, \\ \frac{\exp\left(i(l-k)\frac{(x_2-a)\pi}{b-a}\right)-\exp\left(i(l-k)\frac{(x_1-a)\pi}{b-a}\right)}{l-k}, & \text{otherwise} \end{cases} \tag{2.56}$$

– $\boldsymbol{u} := \{u_l\}_{l=0}^{N-1}, \qquad u_l = \varphi_{levy}\left(\frac{l\pi}{b-a}\right), \qquad u_0 = \frac{1}{2}\varphi_{levy}(0)\,V_0(t_{m+1})$

The matrices $\mathcal{M}_c$ and $\mathcal{M}_s$ are respectively part of Hankel and Toeplitz matrices, which is discussed in [19]. Hankel and Toeplitz matrices offer valuable advantages when it comes to the Discrete Fourier Transform (DFT). The matrix-vector products $\mathcal{M}_c\boldsymbol{u}$ and $\mathcal{M}_s\boldsymbol{u}$ involving these special matrices can be transformed into a circular convolution, a mathematical operation that combines two vectors using element-wise multiplication and summation. This circular convolution is equivalent to the inverse DFT of the product of the forward DFTs of the vectors. That is:

$$\boldsymbol{x} * \boldsymbol{y} = \mathcal{D}^{-1}\left[\mathcal{D}(\boldsymbol{x}) \cdot \mathcal{D}(\boldsymbol{y})\right]. \tag{2.57}$$

This relationship to the DFT is key to understanding the efficiency and usefulness of Hankel and Toeplitz matrices. Making clever use of the particular arrangements inherent within these matrices, the circular convolution operation can be calculated with notable expedience through employment of the Fast Fourier Transform (FFT) algorithmic process. The FFT algorithm capitalizes on the symmetry and periodicity of the DFT, significantly reducing the computational complexity. As a result, the computation of Hankel and Toeplitz matrices, such as $\mathcal{M}_c$ and $\mathcal{M}_s$ can be performed more efficiently compared regular matrix-vector multiplication. This FFT procedure reduces the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log_2(N))$. Since this procedure has to be done $M_{mon}-1$ times, the overall complexity of the COS Barrier methods ends up to be $\mathcal{O}((M_{mon}-1)N\log_2(N))$.

RULE OF THUMB FOR THE TRUNCATION RANGE $[a,b]$

Similarly to the traditional COS Method, it is crucial to choose a truncation range $[a,b]$ that approximates the barrier value accurately. A range too large causes the model too be slower, since it will need a higher number of basis function N. A range too small makes the truncation error large.

The proposed truncation range from [1] is not the same as (2.38). Additionally, the initial log-asset price $x = \log(S_0/K)$ is included:

$$[a,b] = \left[c_1 + x - L\sqrt{c_2 + \sqrt{c_4}}, c_1 + x + L\sqrt{c_2 + \sqrt{c_4}}\right], \tag{2.58}$$

where $L = 10$ and $(c_i)_{i=1,2,4}$ are the cumulants of the underlying stochastic process.

2.7. TENSOR CALCULUS

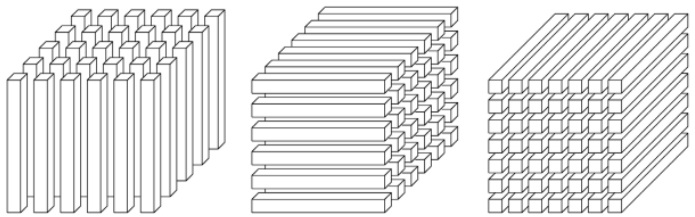
Tensor calculus is an advanced mathematical framework that extends the concepts of matrices and vectors to higher-dimensional objects called tensors. In this paper it is used to describe higher-dimensional problems, which is needed for our second model in Chapter 6. The fundamentals of tensor calculus are explained and illustrated in this section using [20] and [21].

Tensors, on the other hand, introduce additional dimensions and provide a more flexible way to represent and manipulate data. A tensor can be thought of as a multi-dimensional array of numbers, with each element characterized by its position within

the array. While tensors might seem initially daunting due to their higher dimensionality, they are indispensable when modeling and analyzing systems with intricate interactions between multiple variables.

In tensor calculus, one of the fundamental operations is tensor unfolding. This process rearranges the elements of a tensor into a matrix format, unveiling underlying patterns and structures within the tensor. Unfolding tensors enables the application of traditional matrix-based techniques and algorithms. The number of dimensions in a tensor is called the *order* of a tensor. Here every dimension is called a *mode* of the tensor. To indicate a specific element of a dth order tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ an index set $(i_n)_{n=1}^{d}$ is used. Here $1 \leq i_n \leq I_n$ for $n = 1, \ldots, d$.

Before understanding how to do operations with tensors the concept of *fibers* has to be introduced. As matrices have rows and columns, but extending this to higher dimensional tensors this is referred to as fibers. A fiber is an array which is obtained by fixing every index but one. For a three-dimensional tensor the fibers are known as the rows, columns and tubes shown in Figure 2.1 .



(a) Mode-1 (column) fibers: $\mathbf{x}_{:jk}$ (b) Mode-2 (row) fibers: $\mathbf{x}_{i:k}$ (c) Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$

Figure 2.1: Visual illustration of the fibers of a third-order tensor [20].

Performing operations on higher-order tensors can be quite abstract and involve some manipulations to make it more intuitive. A transformation process called *unfolding* is applied to transform higher-order tensors into matrices. As an example we look at unfolding a order-3 tensor with dimension $2 \times 3 \times 4$. This tensor can be rearranged into a 4×6, 3×8 or 2×12 matrix. There are a lot of different ways one can unfold a tensor into a matrix [20], but for our research only the *mode-n unfolding* is relevant. The mode-n unfolding of a tensor $\mathscr{X}$ is denoted as $\mathscr{X}_{(n)}$. It is constructed by setting the mode-n fibers as columns of the matrix. For the example above for $\mathbb{R}^{2\times3\times4}$ the three possible mode-n unfoldings are:

- mode-1 unfolding in $\mathbb{R}^{4\times6}$: using the columns as fibers of the tensor:

$$\mathscr{X}_{(1)} = \begin{bmatrix} x_{111} & x_{121} & x_{131} & x_{112} & x_{122} & x_{132} \\ x_{211} & x_{221} & x_{231} & x_{212} & x_{222} & x_{232} \\ x_{311} & x_{321} & x_{331} & x_{312} & x_{322} & x_{332} \\ x_{411} & x_{421} & x_{431} & x_{412} & x_{422} & x_{432} \end{bmatrix}, \tag{2.59}$$

- mode-2 unfolding in $\mathbb{R}^{3\times 8}$: using the rows as fibers of the tensor:

$$\mathscr{X}_{(2)} = \begin{bmatrix} x_{111} & x_{211} & x_{311} & x_{411} & x_{112} & x_{212} & x_{312} & x_{412} \\ x_{121} & x_{221} & x_{321} & x_{421} & x_{122} & x_{222} & x_{322} & x_{422} \\ x_{131} & x_{231} & x_{331} & x_{431} & x_{132} & x_{232} & x_{332} & x_{432} \end{bmatrix}, \tag{2.60}$$

- mode-3 unfolding in $\mathbb{R}^{2\times 12}$: using the tubes as fibers of the tensor:

$$\mathscr{X}_{(3)} = \begin{bmatrix} x_{111} & x_{211} & x_{311} & x_{411} & x_{121} & x_{221} & x_{321} & x_{421} & x_{131} & x_{231} & x_{331} & x_{431} \\ x_{112} & x_{212} & x_{312} & x_{412} & x_{122} & x_{222} & x_{322} & x_{422} & x_{132} & x_{232} & x_{332} & x_{432} \end{bmatrix}. \tag{2.61}$$

These three unfolding operations are respectively illustrated in Figure 2.1a),b) and c). Once the tensors have been unfolded, the standard matrix operations can be applied. The operations used in this paper will be defined below.

Definition 2.7.1. *(Outer Product). Let $\mathbf{u} \in \mathbb{R}^m, \mathbf{v} \in \mathbb{R}^n$ be two vectors. Their outer product is denoted with $\mathbf{u} \circ \mathbf{v} \in \mathbb{R}^{m\times n}$, and the resulting matrix can be obtained by multiplying each element of $\mathbf{u}$ by each element of $\mathbf{v}$,*

$$\begin{aligned} \mathbf{u} \circ \mathbf{v} &= \begin{bmatrix} u_1 v_1 & u_1 v_2 & \cdots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \cdots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_m v_1 & u_m v_2 & \cdots & u_m v_n \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{u} \cdot v_1 & \mathbf{u} \cdot v_2 & \cdots & \mathbf{u} \cdot v_n \end{bmatrix} \end{aligned}$$

Definition 2.7.2. *(Kronecker Product). The Kronecker product of matrices $A \in \mathbb{R}^{I\times J}$ and $B \in \mathbb{R}^{K\times L}$ is denoted with $A \otimes B \in \mathbb{R}^{IK\times JL}$, and is defined by*

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1J}B \\ a_{21}B & a_{22}B & \cdots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \cdots & a_{IJ}B \end{bmatrix}$$

Definition 2.7.3. *(Khatri-Rao Product). The Khatri-Rao product can be viewed as the columnwise Kronecker product. Given the matrices $A \in \mathbb{R}^{I\times K}$ and $B \in \mathbb{R}^{J\times K}$, the Khatri-Rao product, denoted with $A \odot B \in \mathbb{R}^{IJ\times K}$, is defined by*

$$A \odot B = \begin{bmatrix} a_1 \otimes b_1 & a_2 \otimes b_2 & \cdots & a_K \otimes b_K \end{bmatrix}$$

Definition 2.7.4. *(Hadamard Product). The Hadamard product is the element-wise matrix product. Therefore, given the matrices $A, B \in \mathbb{R}^{I\times J}$, the Hadamard product $A \circledast B \in$*

$\mathbb{R}^{I\times J}$ *produces the matrix*

$$A \circledast B = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \cdots & a_{IJ}b_{IJ} \end{bmatrix}$$

To quantify the similartities between two tensors one can use the matrix norm called the Forbenius norm. This norm can be used to find the distance between two tensors. In this paper, it is used for our second method in Chapter 6.

Definition 2.7.5. *(Frobenius Norm). Given a tensor* $\mathscr{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_d}$*, its Frobenius norm, often abbreviated with F-norm, is defined as the square root of the sum of the squares of all its elements:*

$$\|\mathscr{X}\|_F^2 = \sqrt{\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\cdots\sum_{i_d=1}^{I_d} x_{i_1 i_2 \ldots i_d}^2}$$

3

Our Insight: Pricing Barrier Options Using the COS Method for European Options

This chapter forms the foundation of our research in this paper, as it demonstrates that it is possible to price barrier options using the COS method for European options.. Pricing barrier options is generally computationally intensive. Currently, analytical expressions for barrier option prices are known only for certain models like Geometric Brownian Motion (GBM). For more complex models like the CGMY process, pricing barrier options using the COS Barrier method (as discussed in Section 2.6) can be computationally slow.

In this chapter, we will provide theoretical proof that the COS method for pricing European options can be directly used to price barrier options. Specifically, we will focus on the up-and-out barrier call options. The key idea is to consider the barrier option's value at time t as the expected payoff of the corresponding European call option at maturity, given that the barrier has not been hit by the asset's path. We will also assume that the marginal survival density of the underlying asset price exists, denoted as $\bar{p}(S \mid S_t = s)$. Using these assumptions, we will derive an expression for the barrier option price in terms of a single-dimensional integral.

Furthermore, we will show that the barrier option can be represented using a similar integral, but with a different density function, which we refer to as the survival density.. This will lead to the idea of using the traditional COS method for barrier option pricing by inserting the ch.f. of the survival probability.

The chapter will also present testing evidence to support this theoretical proof. We will compare the barrier option prices obtained using the one-step COS method with those obtained using the COS Barrier method. The results will demonstrate that the two closely each other, even though the method uses the same payoff coefficients V_k as for European options.

The insights gained from this chapter will serve as the foundation for the two models that we will develop in Chapter 4 and 6, by combining machine learning with the one-

step COS method (i.e. the COS method for pricing European options), achieving efficient and accurate pricing for complex Lévy processes like the CGMY model.

3.1. OUR KEY INSIGHT

This chapter's theoretical underpinning emanates from a paper that delves into pricing continuously monitored barrier options using a discrete approximation, as expounded in [22]. Our primary focus centers on the up-and-out barrier call option, delving into its pricing derivation.

The process of pricing these barrier options shares conceptual parallels with the derivation of option prices through the COS method, as elucidated in [5]. Building upon this premise, we seek to establish a similar formulation for barrier options in this section by drawing insights from the paper's findings.

Within the framework of an underlying asset path denoted as $S = S_t, 0 \le t \le T$ with a maturity period of T, our attention gravitates towards instances where the path evades crossing an upper barrier level denoted as H. In this context, we introduce the notion of the barrier's first hitting time, represented as $\tau_H^t := \inf s \ge t \mid S_s \ge H$. The payoff associated with an up-and-out call option upon reaching maturity T is succinctly expressed as $v(S_T, T)$.

In scenarios where the barrier remains unbreached until maturity, the behavior of the up-and-out call option closely mirrors that of a standard European call option. Consequently, the payoff equation simplifies to $v(S_T, T) = (S_T - K)^+$, where K denotes the strike price. As a result, the value of a barrier option at any given time $0 \le t \le T$, contingent upon the underlying asset's valuation S_t, can be succinctly characterized as follows:

$$v_{\text{Bar}}(S_t, t) = e^{-r\Delta t}\mathbb{E}[v_{\text{EU}}(S_T, T)1_{\tau_H^t > T} \mid S_t], \tag{3.1}$$

where $v_{\text{EU}}(S_T, T)$ is the payoff of the European call option at maturity T, and $1_{\tau_H^t > T}$ is the indicator function, which is 1 if $\tau_H^t > T$ and 0 otherwise.
We assume that the marginal survival density of the underlying asset price S exists and is denoted as $\bar{p}(S \mid S_t = s) := \mathbb{P}(T, S \mid S_t = s, \tau_H^t > T)$. The value of the option in Equation (3.1) can thus be written as:

$$v_{\text{Bar}}(S_t, t) = e^{-r\Delta t}\int_0^\infty v_{\text{EU}}(S, T)\mathbb{P}(T, S \mid S_t = s, \tau_H^t > T)dS \quad = e^{-r(T-t)}\int_0^\infty v_{\text{EU}}(S, T)\bar{p}(S)dS, \tag{3.2}$$

which results in a single-dimensional integral. Using the substitutions $x = \log(S_t/K)$ and $y = \log(S_T/K)$, we get:

$$v_{\text{Bar}}(x, t) = e^{-r\Delta t}\int_{-\infty}^\infty v_{\text{EU}}(y, T)\bar{p}(y \mid x)dy. \tag{3.3}$$

Comparing this to the derivation of the COS method for European options in equation (2.26), we notice that we have a similar integral here with the same payoff function as the European option. This implies that we can use the same payoff coefficients V_k as defined in equation (2.30), and only the density coefficients need to be different. Therefore, we get the insight that the value of a barrier option can be calculated using the one-step COS

method for European options, with a ch.f. corresponding to the survival probability of the barrier option.

Furthermore, an important assumption is that Theorem 2.1.5 also holds for the ch.f. $\bar{\phi}$. This means that the following assumption is made for a Lévy process X:

$$\bar{\phi}_H(u,t;x) = \bar{\phi}_H(u,t;0)e^{iux} = \bar{\varphi}_{\text{levy}}(u)e^{iux}, \tag{3.4}$$

where $\bar{\varphi}_{\text{levy}}(u)$ is used to make the notation more convenient. Thus, for pricing barrier options under Lévy processes, the COS representation looks like:

$$v_{\text{Bar}}(x,t) \approx e^{-r\Delta t}\sum_{k=0}^{N-1}{}' \text{Re}\left\{\bar{\varphi}_{\text{levy}}\left(\frac{k\pi}{b-a}\right)\exp\left(ik\pi\frac{x-a}{b-a}\right)\right\}V_k. \tag{3.5}$$

This assumption allows us to extend the COS method for European options to price barrier options under Lévy processes using the same set of basis functions. The next section will present empirical evidence to support this claim and demonstrate the accuracy and efficiency of the traditional COS method for pricing barrier options.

3.2. NUMERICAL EVIDENCE

Here we test and verify the theoretical framework of one-dimensional integration. By deriving the integral for barrier options and contrasting it with the derivation of the COS method for European options, a reasoned inference emerges: the coefficients V_k governing the payoff values remain consistent for barrier options. However, a noticeable divergence arises regarding the survival density, which is integrated into a novel ch.f. denoted as $\bar{\varphi}_{levy}$. The specific nature of this function remains enigmatic.

The numeric investigation relies on the presumption that the value of a barrier option under a Lévy process, with an initial log-asset price x, can be depicted as follows:

$$v_{Bar}(x;t,\boldsymbol{\theta}) \approx e^{-r_{\boldsymbol{\theta}}\Delta t}\sum_{k=0}^{\prime N-1}\text{Re}\left\{\bar{\varphi}_{levy}\left(\frac{k\pi}{b-a}\right)\exp\left(ik\pi\frac{x-a}{b-a}\right)\right\}V_k. \tag{3.6}$$

Within this context, the symbol V_k is defined according to the description provided in (2.30). Notably, for a barrier call, the coefficients align with those of a European call, and the converse holds true for a barrier put. In this equation, $\boldsymbol{\theta}$ denotes a specific blend of model parameters (e.g., μ,σ for GBM), while $\Delta t = T - t$.

Importantly, we introduce two distinct portrayals of the survival ch.f.. This divergence originates from subsequent considerations of boundaries when applying the GPR method to the pricing of barrier options. The initial representation refrains from assuming a predetermined structure for the ch.f., whereas the second representation employs a Fourier expansion to depict the ch.f..

3.2.1. SETUP OF THE LINEAR SYSTEM: REPRESENTATION 1

The objective is to solve $\bar{\varphi}_{levy}\left(\frac{k\pi}{b-a}\right)$ for $k = 0,\ldots,N-1$. It's important to note that $\bar{\varphi}_{levy}$: $\mathbb{R} \to \mathbb{C}$, thus necessitating a dissection of the function into its real and imaginary components. Consequently, we express it as:

$$\bar{\varphi}_{levy}\left(\frac{k\pi}{b-a}\right) = f_R\left(\frac{k\pi}{b-a}\right) + i \cdot f_I\left(\frac{k\pi}{b-a}\right), \tag{3.7}$$

$$R_k := f_R\left(\frac{k\pi}{b-a}\right), \qquad I_k := f_I\left(\frac{k\pi}{b-a}\right). \tag{3.8}$$

While the precise form of $\bar{\varphi}_{levy}$ remains elusive, we possess knowledge of benchmark values $v_{bench}(\boldsymbol{x}; t, \boldsymbol{\theta})$ for a subset of log-asset prices $\boldsymbol{x} \in \mathbb{R}^n$. These benchmarks can be computed in advance using techniques such as COS Barrier or Monte Carlo simulations. The method to validate this fit involves determining the coefficients $\boldsymbol{R}$ and $\boldsymbol{I}$ through a comparison between the benchmark values $v_{bench}(\boldsymbol{x}; t, \boldsymbol{\theta})$ and the COS formula values $v_{BAR}(\boldsymbol{x}; t, \boldsymbol{\theta})$.

To achieve this, a linear system comprising n equations can be derived from the COS formula. By leveraging Euler's identity[1] to expand the complex exponent and utilizing (3.7), the Re-operator can be eliminated:

For a given $\boldsymbol{\theta}$ and t to find $\boldsymbol{R}$ and $\boldsymbol{I}$, this expansion is set equal to $\boldsymbol{v}_{bench}(\boldsymbol{x}; t, \boldsymbol{\theta})$. This results in:

$$\sum_{k=0}^{\prime N-1} \left[R_k \cos\left(k\pi \frac{\boldsymbol{x}-a}{b-a}\right) - I_k \sin\left(k\pi \frac{\boldsymbol{x}-a}{b-a}\right) \right] V_k = e^{r_{\boldsymbol{\theta}} \Delta t} \boldsymbol{v}_{bench}(\boldsymbol{x}; t, \boldsymbol{\theta}), \tag{3.9}$$

where this system of equations can be written in the form

$$\boldsymbol{M} \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} = e^{r_{\boldsymbol{\theta}} \Delta t} \boldsymbol{v}_{bench}(\boldsymbol{x}; t, \boldsymbol{\theta}). \tag{3.10}$$

Since there are two different vectors the matrix $\boldsymbol{M} \in \mathbb{R}^{n \times 2N}$ is split up in two parts like $\boldsymbol{M} = [\boldsymbol{M}_R \;\; \boldsymbol{M}_I]$, where $\boldsymbol{M}_R \in \mathbb{R}^{n \times N}$ and $\boldsymbol{M}_I \in \mathbb{R}^{n \times N}$ correspond respectively to the vectors $\boldsymbol{R}$ and $\boldsymbol{I}$. The elements of these two matrices are defined as follows:

$$(\boldsymbol{M}_R)_{ij} = \begin{cases} \frac{1}{2}\cos\left(j\pi \frac{x_i - a}{b-a}\right) (= 1/2) & \text{if } j = 0 \\ \cos\left(j\pi \frac{x_i - a}{b-a}\right) & \text{if } j \neq 0 \end{cases} \tag{3.11}$$

$$(\boldsymbol{M}_I)_{ij} = \begin{cases} -\frac{1}{2}\sin\left(j\pi \frac{x_i - a}{b-a}\right) (= 0) & \text{if } j = 0 \\ -\sin\left(j\pi \frac{x_i - a}{b-a}\right) & \text{if } j \neq 0 \end{cases}. \tag{3.12}$$

[1] Euler's identity is a fundamental mathematical expression that links the exponential function with trigonometric functions [23]. It states that for any real number η:

$$e^{i\eta} = \cos(\eta) + i \cdot \sin(\eta).$$

This identity elegantly connects the complex exponential function $e^{i\eta}$ with the familiar trigonometric functions $\cos(\eta)$ and $\sin(\eta)$, showcasing the profound interplay between exponential growth and circular motion.

An important note is that the matrix $\boldsymbol{M}$ only has to be calculated once given a set of $\boldsymbol{x}$, since it is not dependent on the model parameters $\boldsymbol{\theta}$ nor the time t. These are all absorbed in the vectors $\boldsymbol{R}$ and $\boldsymbol{I}$. Furthermore it is important to note that the risk-free interest rate $r_{\boldsymbol{\theta}}$ is dependent on the model parameters and is taken to the RHS in (3.10).

SOLVING THE LINEAR SYSTEM USING SVD: REPRESENTATION 1

When tackling the linear system (3.10), a pivotal consideration is the matrix dimension $\boldsymbol{M} \in \mathbb{R}^{n\times 2N}$. This involves n benchmark points $\boldsymbol{x}$ and N basis functions, rendering the linear system to consist of n equations with $2N$ unknowns. While there are three scenarios ($n < 2N$, $n = 2N$, and $n > 2N$), we employ a consistent approach across these cases. Regardless of the case, and given $\boldsymbol{\theta}$ and time t, SVD is employed to deduce the Least Squares (LS) solution for equation (3.10). Specifically, the LS solutions for the real and imaginary ch.f. coefficients are determined through the following optimization problem:

$$\begin{bmatrix} \boldsymbol{R}_{LS} \\ \boldsymbol{I}_{LS} \end{bmatrix} = \underset{\boldsymbol{R},\boldsymbol{I}\in\mathbb{R}^N}{\operatorname{argmin}} \left\| \boldsymbol{M} \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} - e^{r_{\boldsymbol{\theta}}\Delta t} \boldsymbol{v}_{bench}(\boldsymbol{x}; t, \boldsymbol{\theta}) \right\|_2 . \tag{3.13}$$

This optimization task is effectively resolved through SVD, a method elaborated upon in Section 2.4. Subsequent to acquiring the LS solution, these vectors are utilized within the COS formula to compute new values. Consequently, the COS method can be employed to compute barrier options values for a specified $\boldsymbol{\theta}$ and t.

An illustrative example of an up-and-out option within the GBM framework sheds light on how the expansion's option price converges towards benchmark values as the number of benchmark points n increases. In the graphical representation depicted below, the red vertical line signifies the number of unknowns, i.e., $2N$. This particular instance employs $N = 32$, implying a total of 64 real and imaginary ch.f. terms. The horizontal axis denotes the number of benchmark points used to fit the expansion terms. The convergence trend as n increases is visually apparent in Figure 3.1.

3.2.2. SETUP OF THE LINEAR SYSTEM: REPRESENTATION 2

The motivation behind the existence of the second model stems from a limitation in the first model, which fails to consider the interconnectedness between the variables R_k and I_k, both originating from a shared underlying function. This oversight becomes particularly consequential when employing machine learning techniques, as the initial representation struggles with predictive accuracy near the boundaries, especially when accommodating an additional dimension for function input. Moreover, the validation of this revised approach reinforces the applicability of the COS-CPD method.

In the second model, the ch. f. is articulated through its Fourier expansion. This involves delineating the real and imaginary components of the ch. f. as distinct functions, denoted as f_R and f_I respectively, both operating within the interval $[a_\varphi, b_\varphi]$, where $[a, b]$ signifies the truncated integration range for the COS method, and N represents the count of basis functions employed by the COS method. To construct these functions, N_φ basis functions are utilized. This collective framework yields the subsequent formulation for valuing barrier options using the COS method:

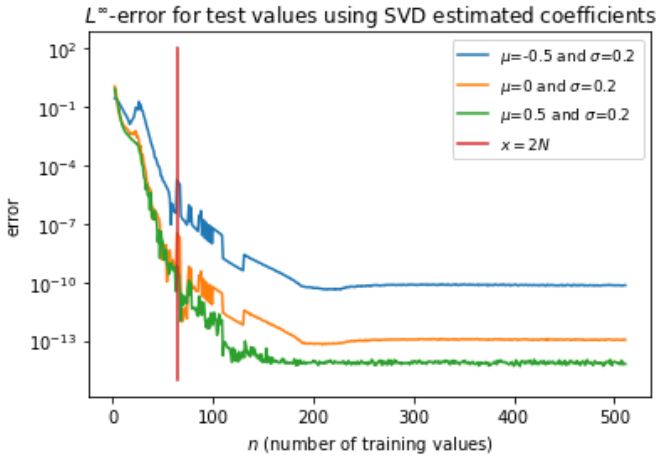


Figure 3.1: Convergence of barrier option prices to benchmark values with increasing n for various examples of (μ,σ) in the GBM Model with $N = 32$.

$$\bar{\varphi}_{levy}(u) = f_R(u) + i \cdot f_I(u) = \sum_{j=0}^{N_\varphi - 1} A_j \cos\left(j\pi \frac{u - a_\varphi}{b_\varphi - a_\varphi}\right) + i \cdot B_j \sin\left(j\pi \frac{u - a_\varphi}{b_\varphi - a_\varphi}\right). \tag{3.14}$$

Just as with the first representation, the same steps are followed to derive a linear system of equations for this representation. The full derivation is stated in the Appendix A.1.

$$\boldsymbol{v}_{BAR}(\boldsymbol{x}; t, \boldsymbol{\theta}) \approx e^{-r_{\boldsymbol{\theta}}\Delta t} {\sum_{k=0}^{\prime N-1}}' \operatorname{Re}\left\{\bar{\varphi}_{levy}\left(\frac{k\pi}{b-a}\right) \exp\left(ik\pi \frac{\boldsymbol{x} - a}{b-a}\right)\right\} V_k \tag{3.15}$$

$$= e^{-r_{\boldsymbol{\theta}}\Delta t} \sum_{j=0}^{N_\varphi - 1} A_j \left[{\sum_{k=0}^{\prime N-1}}' \cos\left(j\pi \frac{u_k - a_\varphi}{b_\varphi - a_\varphi}\right) \cos\left(k\pi \frac{\boldsymbol{x} - a}{b-a}\right) V_k \right] \tag{3.16}$$

$$- e^{-r_{\boldsymbol{\theta}}\Delta t} \sum_{j=0}^{N_\varphi - 1} B_j \left[{\sum_{k=0}^{\prime N-1}}' \sin\left(j\pi \frac{u_k - a_\varphi}{b_\varphi - a_\varphi}\right) \sin\left(k\pi \frac{\boldsymbol{x} - a}{b-a}\right) V_k \right] \tag{3.17}$$

$$:= e^{-r_{\boldsymbol{\theta}}\Delta t} \sum_{j=0}^{N_\varphi - 1} (A_j C_j(\boldsymbol{x}) - B_j S_j(\boldsymbol{x})), \tag{3.18}$$

where $u_k = \frac{k\pi}{b-a}$. The notations $C_j(\boldsymbol{x})$ and $S_j(\boldsymbol{x})$ are used for convenience to describe respectively the sum of the cosines and sines. To find $\boldsymbol{A}$ and $\boldsymbol{B}$ for a given $\boldsymbol{\theta}$ and t, this expansion is set equal to $\boldsymbol{v}_{bench}(\boldsymbol{x}; t, \boldsymbol{\theta})$. This results in:

$$\sum_{j=0}^{N_\varphi - 1} (A_j C_j(\boldsymbol{x}) - B_j S_j(\boldsymbol{x})) = e^{r_{\boldsymbol{\theta}}\Delta t} \boldsymbol{v}_{bench}(\boldsymbol{x}; t, \boldsymbol{\theta}), \tag{3.19}$$

where this system of equations can be written in the form

$$\tilde{\boldsymbol{M}} \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{bmatrix} = e^{r_{\boldsymbol{\theta}} \Delta t} \boldsymbol{v}_{bench}(\boldsymbol{x}; t, \boldsymbol{\theta}). \tag{3.20}$$

Since there are two different vectors the matrix $\tilde{\boldsymbol{M}} \in \mathbb{R}^{n \times 2N_\varphi}$ is split up in two parts like $\tilde{\boldsymbol{M}} = [\tilde{\boldsymbol{M}}_A \;\; \tilde{\boldsymbol{M}}_B]$, where $\tilde{\boldsymbol{M}}_A \in \mathbb{R}^{n \times N_\varphi}$ and $\tilde{\boldsymbol{M}}_B \in \mathbb{R}^{n \times N_\varphi}$ correspond respectively to the vectors $\boldsymbol{A}$ and $\boldsymbol{B}$. The elements of these two matrices are defined as follows:

$$(\tilde{\boldsymbol{M}}_A)_{ij} = C_j(x_i) = {\sum_{k=0}}'^{N-1} \cos\left(j\pi \frac{u_k - a_\varphi}{b_\varphi - a_\varphi}\right) \cos\left(k\pi \frac{x_i - a}{b-a}\right) V_k, \tag{3.21}$$

$$(\tilde{\boldsymbol{M}}_B)_{ij} = -S_j(x_i) = -{\sum_{k=0}}'^{N-1} \sin\left(j\pi \frac{u_k - a_\varphi}{b_\varphi - a_\varphi}\right) \sin\left(k\pi \frac{x_i - a}{b-a}\right) V_k. \tag{3.22}$$

SOLVING THE LINEAR SYSTEM USING SVD: REPRESENTATION 2

Much like the approach employed in solving the first model, the linear system (3.20) is tackled using a consistent optimization procedure. However, a key distinction lies in the dimensionality of matrix $\tilde{\boldsymbol{M}} \in \mathbb{R}^{n \times 2N_\varphi}$. Here, n denotes the count of benchmark points utilized, while N_φ signifies the number of basis functions engaged in the Fourier expansion of the ch.f.. The cases $n < 2N_\varphi$, $n = 2N_\varphi$, and $n > 2N_\varphi$ are again considered. Across all instances, regardless of the relationship between n and $2N_\varphi$, the equation (3.20) is solved using SVD to find the LS solution. This involves determining LS solutions for the real and imaginary components of the ch. f. Fourier-expansion terms, which are governed by the following optimization:

$$\begin{bmatrix} \boldsymbol{A}_{LS} \\ \boldsymbol{B}_{LS} \end{bmatrix} = \underset{\boldsymbol{A},\boldsymbol{B} \in \mathbb{R}^{N_\varphi}}{\operatorname{argmin}} \left\| \tilde{\boldsymbol{M}} \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{bmatrix} - e^{r_{\boldsymbol{\theta}} \Delta t} \boldsymbol{v}_{bench}(\boldsymbol{x}; t, \boldsymbol{\theta}) \right\|_2. \tag{3.23}$$

By incorporating this LS solution and plugging it into the Fourier-expansion of the ch. f., the complete ch.f. is recovered. This contrasts with the first model, where only the values of a specific set of N points are computed. Subsequently, leveraging this estimated ch. f., novel values for barrier options can be ascertained. It's important to note that this process remains contingent upon a fixed $\boldsymbol{\theta}$ and t.

Similarly to the initial representation, the convergence towards benchmark values becomes evident as the quantity of benchmark points augments. In the ensuing visualizations, the vertical line demarcates the count of unknowns, represented as $2N_\varphi$. Within these illustrations, we consider two instances: $N_\varphi = 32$ and $N_\varphi = 64$, signifying a total of 64 and 128 real and imaginary Fourier expansion terms for the ch. f. respectively. Additionally, the number of basis functions within the COS method is designated as $N = 64$. On the horizontal axis, we denote the number of benchmark points utilized for fitting the expansion terms. The progressive convergence as n increases is depicted in Figure 3.2.

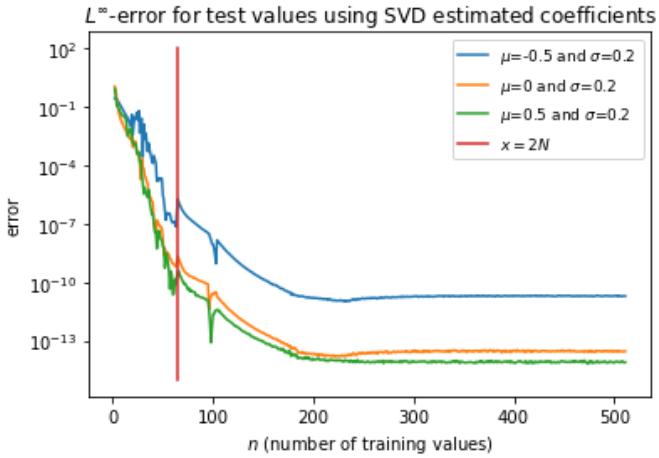


Figure 3.2: Convergence of barrier option prices to benchmark with increasing n for multiple examples of (μ,σ) in the GBM Model with $N_\varphi = 32$ and $N = 64$.

3.2.3. CHOICE OF BENCHMARK VALUES x

Choosing the appropriate grid of benchmark points is crucial to achieving machine precision accuracy in barrier option pricing using the traditional COS method. This importance stems from the behavior of v_{bench} concerning values of S near the barrier level H. Specifically, the value curve is not smooth around H due to the fact that for up-and-out barrier options with initial prices $S \geq H$, the option is worthless, i.e., $v_{\text{bench}}(\log(S/K)) = 0$.

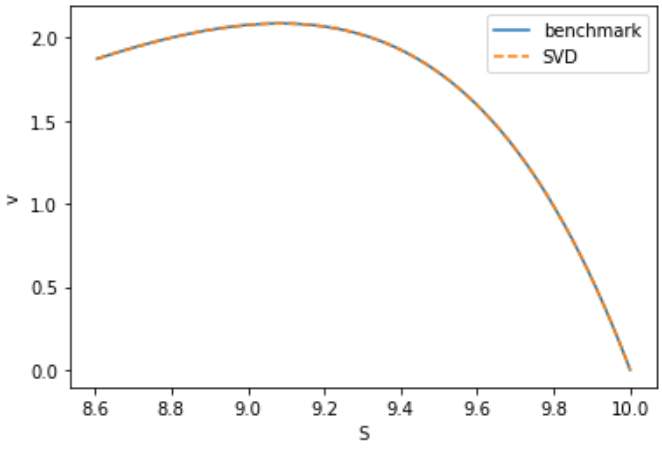
When incorporating asset prices above the barrier level, both representations' LS solutions are also fitted on these values, leading to the presence of a Gibbs phenomenon, as observed in Figure 3.3 around $S = H$. Consequently, this adversely affects the overall accuracy across the entire benchmark axis, resulting in a significant decrease in accuracy. In Figure 3.4, we can see the L^1-error increase from around $\mathcal{O}(10^{-11})$ to $\mathcal{O}(10^{-3})$ when starting to include values after H. The L^∞-error even increases from $\mathcal{O}(10^{-10})$ to $\mathcal{O}(10^{-1})$.

To illustrate this behavior, we consider two benchmark grids [2]: $\boldsymbol{S}_0 = (3 : H : 1000)$ and $\boldsymbol{S}_1 = (3 : H+1 : 1000)$, where H represents the specific barrier value. We define $\boldsymbol{x}_0 = \log(\boldsymbol{S}_0/K)$ and $\boldsymbol{x}_1 = \log(\boldsymbol{S}_1/K)$. As a result, $\boldsymbol{x}_0$ contains benchmark asset values up to H, while $\boldsymbol{x}_1$ includes asset values above H as well.

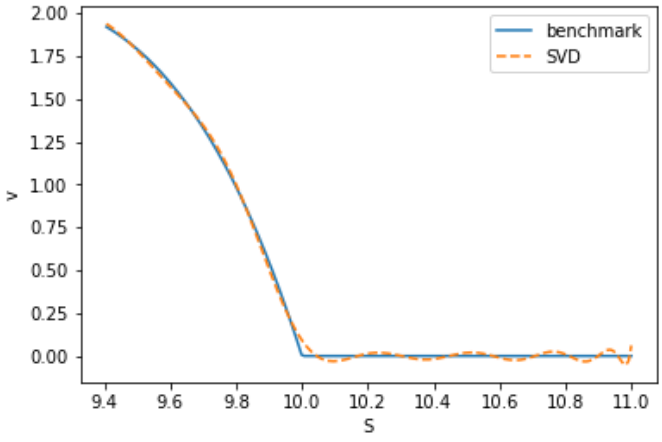
By understanding the impact of the Gibbs phenomenon and carefully selecting the benchmark grids, we can optimize the accuracy of barrier option pricing with the traditional COS method.

In conclusion, the choice of the benchmark grid is critical for achieving machine precision accuracy in barrier option pricing using the traditional COS method. The presence of the Gibbs phenomenon, observed when incorporating asset prices above the

[2] The notation $(a : b : n)$ is consistently utilized in the paper to define a range spanning from boundary a to boundary b, partitioned into n evenly spaced intervals.



(a) Benchmark set $\mathbf{x}_0$.



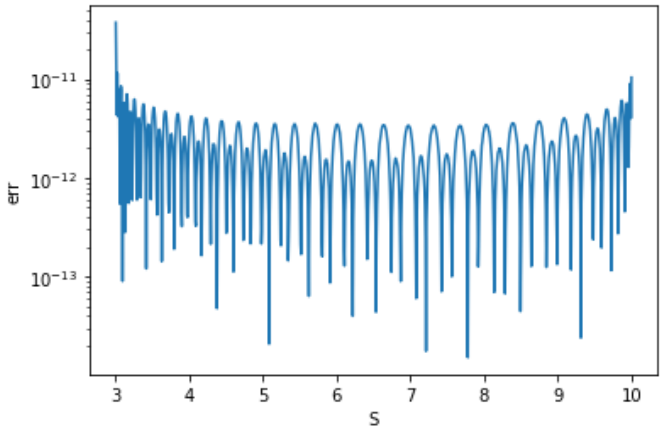
(b) Benchmark set $\mathbf{x}_1$.

Figure 3.3: Comparison of Option Price Estimation: Fitted Expansion Terms using SVD on Training Sets $\mathbf{x}_0$ and $\mathbf{x}_1$.

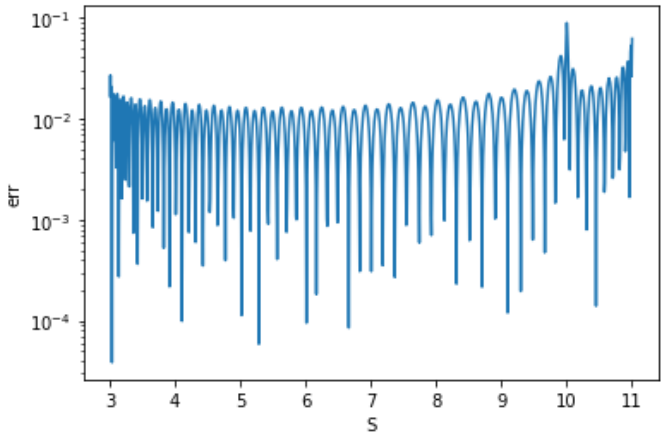
barrier level, negatively impacts the overall accuracy of the pricing model. To optimize accuracy, it is recommended to use the benchmark grid $\boldsymbol{x}_0$, containing asset values up to the barrier level H, thereby avoiding complexities associated with the discontinuity near H. By making this careful selection, reliable and accurate pricing results can be obtained for barrier options using the traditional COS method.

3.2.4. CHOICE OF THE INTEGRATION RANGE $[a, b]$

When utilizing the COS method for pricing barrier options, it is crucial to select a truncation range $[a, b]$ that yields accurate results for the barrier option prices. For European options, the rule of thumb for determining this range is defined by calculating the values



(a) Benchmark set $\mathbf{x}_0$.



(b) Benchmark set $\mathbf{x}_1$.

Figure 3.4: Comparison of Absolute Errors: Option price vs. Fitted option price using SVD-calculated expansion terms on training sets $\mathbf{x}_0$ and $\mathbf{x}_1$.

with (2.38). For the COS Barrier method, which differs from the regular COS method, there is (2.58). However, for pricing barrier options using the traditional COS method, there is no established rule of thumb for determining the truncation range.

In our research, when applying the COS method in our own models for pricing barrier options, we fix the truncation range beforehand to ensure that the SVD of the expansion matches benchmark values up to machine precision. This step allows us to verify that the calculated option values correspond to the benchmark values within very small numerical differences.

4

Our Method 1 for option pricing: The COS-GPR Method

In this chapter, we present our first method called the COS-GPR method. By combining the power of Gaussian Process Regression (GPR) with the COS method, we achieve enhanced accuracy and reduced time complexity compared to existing methods.

Traditionally, option pricing focuses solely on predicting option prices. However, our COS-GPR method takes a different approach by estimating the survival ch.f., i.e. the ch.f. of the underlying process's survival density, which reduces the dimensionality of the problem. This reduction in dimensions streamlines the problem, making the inputs to the training step only the model parameters.

Inside the COS method, the GPR comes into play, approximating a vector of characteristic function (ch.f.) coefficients, and as a result we get a predicted price in the end. We explore two different variations of the COS-GPR method, starting with the intuitive GPR1 model. However, GPR1 lacks the binding relation among the elements within the output vector due to the independence assumption. To address this limitation, we introduce the GPR2 model, incorporating the Fourier-cosine expansion of the survival ch.f. itself to account for the aforementioned binding relation and therefore enhance accuracy.

Throughout the chapter, we delve into the training and theoretical aspects of the COS-GPR method, paving the way for its application in pricing barrier options under the CGMY-model. On this application, the COS-GPR method's performance is evaluated against the 1D COS Barrier method developed in [1] by comparing the two fits to the benchmark value curve. The results demonstrate that our approach outperforms the existing method in terms of accuracy and significantly reduces computational time. However, it is important to note that GPR faces challenges in predicting values near the boundary of the training data set, which is also visualised further in Chapter 5. This is a common phenomenon for GPR prediction.

4.1. BACKGROUND INFORMATION ON GPR

GPR is a powerful non-parametric Bayesian technique that facilitates modeling complex relationships in data without imposing strong assumptions on the underlying function. Utilizing principles of probability and statistics, GPR offers a flexible framework for estimating and predicting unknown values based on benchmark data. In the context of supervised learning, GPR operates as a regression method, focusing on estimating real values, such as the value of a ch.f. or an option.

The choice of GPR over other supervised learning methods is motivated by its extensive research application in quantitative finance [6] and its transparency compared to black-box deep-learning techniques like NN techniques. GPR's intuitive assumptions about the relationship between the underlying structure and the output make it an attractive option for financial modeling.

The training phase, while time-consuming, needs to be performed only once. Once trained, GPR allows for rapid and efficient new predictions, contributing to its computational advantage. However, it's essential to alreaady acknowledge that GPR may encounter limitations near the boundaries of the training set.

4.1.1. CONSTRUCTING THE MODEL

As for all supervised learning methods, GPR is fitted to the data using a training set with n observations $\mathscr{D}_{train} = (\boldsymbol{X}, \boldsymbol{Y}) = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{n}$. Here $\boldsymbol{X} \in \mathbb{R}^{n \times d_{in}}$ symbolises the training input data and $\boldsymbol{Y} \in \mathbb{R}^{n \times d_{out}}$ the training output data. With this training data the goal is to find a mapping $\boldsymbol{f} : \mathbb{R}^{d_{in}} \longrightarrow \mathbb{R}^{d_{out}}$ such that it represents the relation between the inputs and the outputs i.e.:

$$\boldsymbol{y}_i = \boldsymbol{f}(\boldsymbol{x}_i) + \boldsymbol{\epsilon}_i. \tag{4.1}$$

In the GPR method this function f is a Gaussian process and the noise $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}\sigma^2_{GPR})$ is assumed to be independent and identically distributed among observations $i = 1, ..., n$. Here $\mathbf{I}$ is an identity matrix with the correct dimensions.

Definition 4.1.1. *A stochastic process $(X_t)_{t \in T}$ is a Gaussian process iff for every finite set of indices $t_1, ..., t_n$ the vector $(X_{t_1}, ..., X_{t_n})$ assumes a multivariate normal distribution.*

For the model in 4.1, the process $\boldsymbol{f}(\boldsymbol{x})$ is defined by a mean function (here we assume a mean function equal to zero) and a covariance function, also called the *kernel*, $k(\boldsymbol{x}, \tilde{\boldsymbol{x}})$. In the univariate case where f has an one-dimensional output, then for some $(\boldsymbol{X}, f)$:

$$f(\boldsymbol{X}) \sim \mathcal{N}(\boldsymbol{0}, K(\boldsymbol{X}, \boldsymbol{X})). \tag{4.2}$$

This covariance matrix $K(\boldsymbol{X}, \boldsymbol{X})$ is constructed using the kernel $k(\boldsymbol{x}, \tilde{\boldsymbol{x}})$. Namely the coefficients of the matrix $(K(\boldsymbol{X}, \boldsymbol{X}))_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The kernel used throughout the paper is called the squared-exponenital (SE) kernel i.e. the *SE-kernel* . This is one of the most widely used kernels for Gaussian processes. It is dependent on two hyperparameters σ_f^2 and l and the function is defined as:

$$k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \sigma_f^2 \exp\left(\frac{-||\boldsymbol{x} - \tilde{\boldsymbol{x}}||_2^2}{2l^2}\right). \tag{4.3}$$

4.1.2. MULTIVARIATE GPR FOR MODEL OUTPUTS

In this paper, our models require multivariate outputs (vectors of expansion terms). However, a significant assumption made in this context is that there is no cross-correlation between different response variables, implying that all response variables are treated as mutually independent. This simplifying assumption is adopted due to the complexity of GPR with cross-correlation, an area that is still under active research [7].

Thus, we consider the function $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), ..., f_{d_{out}}(\boldsymbol{x}))$, where each individual function f_j is modeled as a separate Gaussian process using the SE-kernel. Naturally, the parameters of the kernels for these functions differ. Specifically, for $j = 1, ..., d_{out}$, a sample $(\boldsymbol{X}, f_j)$ is associated with a Gaussian process $f_j(\boldsymbol{X}) \sim \mathcal{N}(\boldsymbol{0}, K_j(\boldsymbol{X}, \boldsymbol{X}))$, with covariance matrix $K_j(\boldsymbol{X}, \boldsymbol{X})$ defined by the SE-kernel with parameters $\sigma^2_{f,j}$ and l_j.

Since the multivariate outputs are treated independently, the total output of the model, denoted as $\boldsymbol{Y} \in \mathbb{R}^{n \times d_{out}}$, must be split per response variable. Specifically, for each model j, the input $\boldsymbol{X}$ is linked to the corresponding output, which is represented by the j-th column of $\boldsymbol{Y}$, denoted as $\boldsymbol{Y}[:, j]$.

To find the optimal kernel parameters $\sigma^2_{f,j}$ and l_j, we maximize the following objective function, which essentially represents the log-likelihood function:

$$(\hat{\sigma}^2_{f,j}, \hat{l}_j) = \underset{\sigma^2_{f,j}, l_j}{\operatorname{argmax}} \left[-\frac{1}{2} \log(\det(K_j(\boldsymbol{X}, \boldsymbol{X}))) - \frac{1}{2} \boldsymbol{Y}[:, j]^T K_j(\boldsymbol{X}, \boldsymbol{X}) \boldsymbol{Y}[:, j] + c \right]. \tag{4.4}$$

This formulation allows us to effectively model and optimize each individual response variable while acknowledging their independence, facilitating the application of Gaussian Process Regression in a multivariate context.

4.1.3. PREDICTION OF TEST DATA

Let us fix the input dimension j. After training, the model is tested using new inputs $\boldsymbol{X}_{test}$. Using this test set as input, the GPR model returns an estimate for $\boldsymbol{Y}_{test}$, which is related to the mapping 4.1. Then using this mapping , and the distribution of new data $f_{j,test}(\boldsymbol{X}_{test}) \sim \mathcal{N}(\boldsymbol{0}, K(\boldsymbol{X}_{test}, \boldsymbol{X}_{test}))$, the GPR method estimates the $f_{j,test}$ using the following joint distribution:

$$\begin{bmatrix} \boldsymbol{Y}[:, j] \\ \boldsymbol{f}_{j,test}(\boldsymbol{X}_{test}) \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K_j(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2_{GPR} \boldsymbol{I} & K_j(\boldsymbol{X}, \boldsymbol{X}_{test}) \\ K_j(\boldsymbol{X}_{test}, \boldsymbol{X}) & K_j(\boldsymbol{X}_{test}, \boldsymbol{X}_{test}) \end{bmatrix}\right). \tag{4.5}$$

From this distribution, the distribution of $\boldsymbol{f}_{j,test}$ conditioned on the training output for the jth response variable, $\boldsymbol{Y}[:, j]$, is deducted. The expectation (e.g. mean) of this distribution will be the predictor of the test outputs for the jth response variable:

$$\boldsymbol{f}_{j,test} \mid \boldsymbol{Y}[:, j], \boldsymbol{X}, \boldsymbol{X}_{test} \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j), \tag{4.6}$$

where

$$\boldsymbol{\mu}_j = K_j(\boldsymbol{X}_{test}, \boldsymbol{X}) \left[K_j(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2_{GPR} \boldsymbol{I} \right]^{-1} \boldsymbol{Y}[:, j] \tag{4.7}$$

$$\boldsymbol{\Sigma}_j = K_j(\boldsymbol{X}_{test}, \boldsymbol{X}_{test}) - K_j(\boldsymbol{X}_{test}, \boldsymbol{X}) \left[K_j(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2_{GPR} \boldsymbol{I} \right]^{-1} K_j(\boldsymbol{X}, \boldsymbol{X}_{test}). \tag{4.8}$$

This means that the estimate yields

$$\hat{\boldsymbol{f}}_{j,test} = \mathbb{E}[\boldsymbol{f}_{j,test} \mid \boldsymbol{Y}[:,j], \boldsymbol{X}, \boldsymbol{X}_{test}] = \boldsymbol{\mu}_j. \tag{4.9}$$

Then doing this for all $j = 1,...,d_{out}$ gives the total output $\boldsymbol{y}_i = \boldsymbol{Y}[i,:]$ for $i = 1,...,n$. The probabilistic nature of the GPR predictor allows for an estimation that provides not only the mean but also the uncertainty associated with the predictions for each response variable.

4.2. GPR FOR BARRIER OPTION PRICING VIA SURVIVAL CH.F. ESTIMATION

In Chapter 3, we established that barrier options can be efficiently priced using the 1D COS method for pricing European options. Moreover, under Lévy processes, we discovered evidence that the initial price can be extracted from the survival ch.f., reducing the dimensionality by two: one for the initial price and another for the strike, which can be derived as a scalar multiple of V_k (see 2.30).

In this section, we utilize GPR as an approximation method for the survival ch.f.. That is, given the model inputs $\boldsymbol{\theta}$ of the underlying stochastic process, GPR returns the survival ch.f.. Using which, option values can be computed. To achieve this, we first construct a training set comprising input parameters (model parameters) and outputs (ch.f. coefficients). These outputs are calculated using SVD and associated with the corresponding inputs. As discussed in Chapter 3, two representations are generated, leading to the creation of two distinct GPR models.

For Representation 1 from Section 3.2.1, we observed some boundary-related issues. We address and resolve these problems by imposing the binding relation between the training input and output, resulting in improved accuracy and performance for barrier option pricing.

4.2.1. CALIBRATION USING REPRESENTATION 1: GPR1

In this section, we apply GPR to calibrate the first representation introduced in Section 3.2.1. We aim to find the mapping of the survival ch.f. $\boldsymbol{\theta} \mapsto \bar{\varphi}_{levy}(u \mid \boldsymbol{\theta})$ for a fixed value of u. Rather than considering the mapping $(u, \boldsymbol{\theta}) \mapsto \bar{\varphi}_{levy}(u \mid \boldsymbol{\theta})$, which can lead to boundary issues with GPR, we focus on two sets of separate mappings, each set is with $k = 0,...,N-1$, given a truncation range $[a,b]$.

The mappings we seek to find are:

$$\boldsymbol{\theta} \mapsto \mathrm{Re}\left\{\varphi_{levy}\left(\frac{k\pi}{b-a} \mid \boldsymbol{\theta}\right)\right\} := R_k(\boldsymbol{\theta}), \tag{4.10}$$

$$\boldsymbol{\theta} \mapsto \mathrm{Im}\left\{\varphi_{levy}\left(\frac{k\pi}{b-a} \mid \boldsymbol{\theta}\right)\right\} := I_k(\boldsymbol{\theta}). \tag{4.11}$$

Collectively, these mappings can be represented as vectors:

$$\boldsymbol{\theta} \mapsto \mathbf{R}(\boldsymbol{\theta}), \tag{4.12}$$

$$\boldsymbol{\theta} \mapsto \mathbf{I}(\boldsymbol{\theta}). \tag{4.13}$$

To obtain these mappings, we require training data. The training data is initialized as a d-dimensional grid with n total grid points, where d is the dimension of the parameter space Θ. For each training input $\boldsymbol{\theta}_i \in \mathbf{X}$, $i = 1, ..., M$, the corresponding training outputs $\mathbf{R}(\boldsymbol{\theta}_i)$ and $\mathbf{I}(\boldsymbol{\theta}_i)$ are computed using SVD (Section 3.2.1) with a given N, set of log-asset prices $\mathbf{x}$, and truncation range $[a, b]$.

The GPR1 model assumes the form in (4.1), where the mappings $\mathbf{R}(\boldsymbol{\theta}_i)$ and $\mathbf{I}(\boldsymbol{\theta}_i)$ are represented as follows:

$$\mathbf{R}(\boldsymbol{\theta}_i) = \mathbf{f}_R(\boldsymbol{\theta}_i) + \boldsymbol{\epsilon}_i^R, \tag{4.14}$$

$$\mathbf{I}(\boldsymbol{\theta}_i) = \mathbf{f}_I(\boldsymbol{\theta}_i) + \boldsymbol{\epsilon}_i^I. \tag{4.15}$$

Here, the independent noise vectors $\boldsymbol{\epsilon}_i^R, \boldsymbol{\epsilon}_i^I \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}\sigma^2_{GPR})$ include the approximation error from the SVD when numerically calculating the corresponding outputs of the mapping. To facilitate coding convenience, these two mappings are combined into a collective mapping as follows:

$$\begin{bmatrix} \mathbf{R}(\boldsymbol{\theta}_i) \\ \mathbf{I}(\boldsymbol{\theta}_i) \end{bmatrix} = \begin{bmatrix} \mathbf{f}_R(\boldsymbol{\theta}_i) \\ \mathbf{f}_I(\boldsymbol{\theta}_i) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_i^R \\ \boldsymbol{\epsilon}_i^I \end{bmatrix}, \tag{4.16}$$

which is expressed as a sum of $2N$-dimensional vectors:

$$\mathbf{RI}(\boldsymbol{\theta}_i) = \mathbf{f}(\boldsymbol{\theta}_i) + \boldsymbol{\epsilon}_i. \tag{4.17}$$

The process of obtaining this mapping is explained in Section 4.1.2, where binding relation among the different response variables is assumed to be non-existent. Solving the multivariate mapping simplifies to solving $2N$ univariate mappings. Per mapping, the kernel matrix is evaluated, which depends on the kernel parameters. These hyperparameters are obtained by optimizing a log-likelihood function (Section 4.4) based on the output data of the corresponding response variable.

For our research, we used the `GaussianProcessRegressor` function from the Python library `scikit-learn`. This module, even when dealing with multivariate output, assumes independence between the different response variables.

4.2.2. CALIBRATION USING REPRESENTATION 2: GPR2

In this section, we apply GPR to calibrate the second representation introduced in Section 3.2.2. Similar to GPR1, GPR2 also aims to find a mapping $\boldsymbol{\theta} \mapsto \bar{\varphi}_{levy}(u \mid \boldsymbol{\theta})$. However, in GPR2, the functional assumption input u is incorporated into the model. This is achieved by fitting the Fourier-cosine coefficients of the ch.f. using only the model parameters $\boldsymbol{\theta}$ as input. As a result, the GPR2 model directly represents the ch.f. as a function of u, unlike GPR1, which collects the ch.f. values at specific points $k\pi/(b-a)$. From now on, GPR applied to this representation will be referred to as GPR2.

For GPR2, we want to find $2N_\varphi$ different mappings using 3.14. For $j = 0, ..., N_\varphi - 1$, the mappings we aim to obtain are:

$$\boldsymbol{\theta} \mapsto A_j(\boldsymbol{\theta}), \tag{4.18}$$

$$\boldsymbol{\theta} \mapsto B_j(\boldsymbol{\theta}), \tag{4.19}$$

which can be expressed as vectors:

$$\boldsymbol{\theta} \mapsto \mathbf{A}(\boldsymbol{\theta}), \tag{4.20}$$
$$\boldsymbol{\theta} \mapsto \mathbf{B}(\boldsymbol{\theta}). \tag{4.21}$$

The process for obtaining the training data for GPR2 is analogous to the GPR1 model. A d-dimensional grid of model parameters $\mathbf{X}$ is created, and for each training input $\boldsymbol{\theta}_i \in \mathbf{X}$, the corresponding training outputs $\mathbf{A}(\boldsymbol{\theta}_i)$ and $\mathbf{B}(\boldsymbol{\theta}_i)$ are calculated using SVD from Section 3.2.2 for a given N, set of log-asset prices $\mathbf{x}$, and truncation range $[a,b]$, and stored in the training output data $\mathbf{Y}$.

The GPR2 model assumes the form in 4.1, where the mappings $\mathbf{A}(\boldsymbol{\theta}_i)$ and $\mathbf{B}(\boldsymbol{\theta}_i)$ are represented as follows:

$$\mathbf{A}(\boldsymbol{\theta}_i) = \mathbf{f}_A(\boldsymbol{\theta}_i) + \boldsymbol{\epsilon}_i^A, \tag{4.22}$$
$$\mathbf{B}(\boldsymbol{\theta}_i) = \mathbf{f}_B(\boldsymbol{\theta}_i) + \boldsymbol{\epsilon}_i^B. \tag{4.23}$$

The independent noise vectors $\boldsymbol{\epsilon}_i^A, \boldsymbol{\epsilon}_i^B \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}\sigma^2_{GPR})$ capture the approximation error resulting from the SVD process when numerically calculating the corresponding outputs of the mapping. To simplify coding, we combine these two mappings into a collective mapping:

$$\begin{bmatrix} \mathbf{A}(\boldsymbol{\theta}_i) \\ \mathbf{B}(\boldsymbol{\theta}_i) \end{bmatrix} = \begin{bmatrix} \mathbf{f}_A(\boldsymbol{\theta}_i) \\ \mathbf{f}_B(\boldsymbol{\theta}_i) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_i^A \\ \boldsymbol{\epsilon}_i^B \end{bmatrix}, \tag{4.24}$$

written as a sum of $2N_\varphi$-dimensional vectors:

$$\mathbf{AB}(\boldsymbol{\theta}_i) = \mathbf{f}(\boldsymbol{\theta}_i) + \boldsymbol{\epsilon}_i. \tag{4.25}$$

The process for obtaining this mapping is analogous to the GPR1 model using `scikit-learn` and `GaussianProcessRegressor`.

4.3. APPLICATION OF THE COS-GPR METHOD TO EFFICIENT BARRIER OPTION PRICING

Our research primarily aims to develop a fast and accurate pricer for barrier options under Lévy processes. In this section, we introduce the COS-GPR method, our first approach, to price an up-and-out barrier option under the CGMY-process (see definition in 2.1.9). We employ GPR2 from Section 4.2.2 for this pricing, as GPR1 can be analogously used. However, it's worth noting that the analytical expression of the option price under this stochastic process remains unknown.

The current benchmark method used to price this option under CGMY is the COS Barrier method [1], which we discussed in Section 2.6. This method calculates continuation values at a pre-specified number of monitoring dates denoted as M_{mon}. However, a significant drawback of the COS Barrier method is that its computational time increases with a higher number of monitoring dates.

In this section, we conduct a comparison of the computational time between the COS-GPR method and the COS Barrier method for 250 monitoring dates. We will assess the accuracy of the COS-GPR method against the COS Barrier method with 25 monitoring dates for the case of 250 monitoring dates. It's worth noting that the COS Barrier method with a smaller number of monitoring dates provides a good approximation of the COS Barrier method with higher monitoring dates.

For this CGMY method example, we consider fixed values for the diffusion term $\sigma_B = 0.5$ and the interest rate r. Additionally, the contract parameters, including maturity T, barrier level H, and the type of option (call), are also fixed. Consequently, the input parameters are C, G, M, and Y, resulting in an input dimension of $d = 4$. The GPR2 aims to find a mapping $(C, G, M, Y) \mapsto \bar{\varphi}_{levy}(u|C, G, M, Y)$ using the Fourier-cosine series of the unknown survival ch.f.. Following Section 4.2.2, the $2N_\varphi$ mappings we aim to obtain for $j = 0, ..., N_\varphi - 1$ are as follows:

$$(C, G, M, Y) \mapsto A_j(C, G, M, Y) \tag{4.26}$$
$$(C, G, M, Y) \mapsto B_j(C, G, M, Y), \tag{4.27}$$

These mappings are connected to the ch.f. in 3.14, where N_φ represents the number of expansion terms used to describe the real and imaginary parts of the ch.f.. The GPR2 provides an estimation of the ch.f. as follows:

$$\bar{\varphi}_{levy}(u|C, G, M, Y) = \sum_{j=0}^{N_\varphi - 1} A_j(C, G, M, Y) \cos\left(j\pi \frac{u - a_\varphi}{b_\varphi - a_\varphi}\right) + i \cdot B_j(C, G, M, Y) \sin\left(j\pi \frac{u - a_\varphi}{b_\varphi - a_\varphi}\right). \tag{4.28}$$

The next step involves finding these mappings using training data. We begin by setting up a predefined set of training values, where the number of training points per dimension is the same and denoted as m. Consequently, the total number of training points is $M = m^4$. For each training input (C_i, G_i, M_i, Y_i), we calculate the outputs $A_j(C_i, G_i, M_i, Y_i)$ and $B_j(C_i, G_i, M_i, Y_i)$ using SVD (see 3.2.2). Specifically, we solve the problem 3.23 for $i = 1, ..., M$, which in this example can be expressed as:

$$\begin{bmatrix} \boldsymbol{A}(C_i, G_i, M_i, Y_i) \\ \boldsymbol{B}(C_i, G_i, M_i, Y_i) \end{bmatrix} = \underset{\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{N_\varphi}}{\operatorname{argmin}} \left\| \tilde{\boldsymbol{M}} \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{bmatrix} - e^{rT} \boldsymbol{v}_{bench}(\boldsymbol{x}; C_i, G_i, M_i, Y_i) \right\|_2. \tag{4.29}$$

Here, $\boldsymbol{x}$ represents a predefined set of log-asset prices, and $\boldsymbol{v}_{bench}(\boldsymbol{x}; C_i, G_i, M_i, Y_i)$ is calculated using the COS Barrier method with the corresponding number of monitoring dates M_{mon}. The choice of the benchmark grid is motivated by including asset values up to the barrier level, as this optimizes the performance of the SVD and avoids the occurrence of the Gibbs phenomenon (see Section 3.2.3). This output acquisition is part of the pre-processing of the training and only needs to be done once. Once all the outputs have been calculated for both the real and imaginary parts, stored respectively in output matrices $\mathbf{Y}_A$ and $\mathbf{Y}_B$ (both of size $M \times N_\varphi$), we train the GPR2 with multivariate outputs, as explained in Section 4.1.2.

We now proceed to evaluate the model's performance on a single test example denoted as $X_{test} = (C_{test}, G_{test}, M_{test}, Y_{test})$, using the GPR mapping obtained during the training phase. Our primary focus is on the real part coefficient mappings $\mathbf{A}(\boldsymbol{\theta})$, with the understanding that the imaginary part can be treated similarly. For a given index $j = 0, ..., N_\varphi - 1$, we consider the distribution of a new test value, $A_j(X_{test})$, obtained using the found kernel matrices $K_{A,j}(\cdot,\cdot)$:

$$\begin{bmatrix} \boldsymbol{Y}_A[:,j] \\ A_j(\boldsymbol{X}_{test}) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_{A,j}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2_{GPR}\boldsymbol{I} & K_{A,j}(\boldsymbol{X},\boldsymbol{X}_{test}) \\ K_{A,j}(\boldsymbol{X}_{test},\boldsymbol{X}) & K_{A,j}(\boldsymbol{X}_{test},\boldsymbol{X}_{test}) \end{bmatrix}\right). \tag{4.30}$$

From this distribution, we derive the conditional distribution of the test value $A_j(X_{test})$ (conditioned on the training data):

$$A_j(X_{test}) \mid \boldsymbol{Y}_A[:,j], \boldsymbol{X}, \boldsymbol{X}_{test} \sim \mathcal{N}(\boldsymbol{\mu}_{A,j}, \boldsymbol{\Sigma}_{A,j}), \tag{4.31}$$

where the mean $\boldsymbol{\mu}_{A,j}$ and covariance $\boldsymbol{\Sigma}_{A,j}$ are given by:

$$\boldsymbol{\mu}_{A,j} = K_{A,j}(\boldsymbol{X}test, \boldsymbol{X}) \left[K_{A,j}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2_{GPR}\boldsymbol{I}\right]^{-1} \boldsymbol{Y}_A[:,j] \tag{4.32}$$

$$\boldsymbol{\Sigma}_{A,j} = K_{A,j}(\boldsymbol{X}_{test},\boldsymbol{X}_{test}) - K_{A,j}(\boldsymbol{X}_{test},\boldsymbol{X}) \left[K_{A,j}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2_{GPR}\boldsymbol{I}\right]^{-1} K_{A,j}(\boldsymbol{X},\boldsymbol{X}_{test}). \tag{4.33}$$

By assumption, the predictor for this random variable is the mean, the estimate is given by:

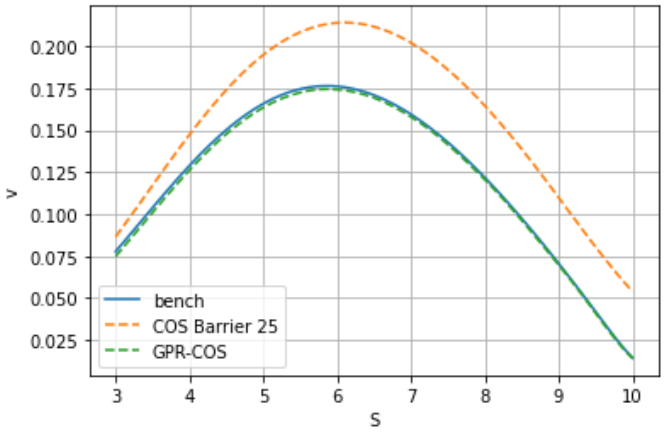
$$\hat{A}_j(X_{test}) = \mathbb{E}[A_j(X_{test}) \mid \boldsymbol{Y}_A[:,j], \boldsymbol{X}, \boldsymbol{X}_{test}] = \boldsymbol{\mu}_{A,j}. \tag{4.34}$$

By repeating this procedure for all j, we obtain the full estimated real part expansion vector $\mathbf{A}(X_{test}) = [\hat{A}_0(X_{test}), \hat{A}_1(X_{test}), ..., \hat{A}_{N\varphi-1}(X_{test})]^T$ for the new test input value $X_{test} = (C_{test}, G_{test}, M_{test}, Y_{test})$. Performing the same analogous procedure for the imaginary part with $B_j(X_{test})$ provides all the necessary Fourier-series expansion terms to calculate the barrier option value using the COS formula derived in A.10. For this example, the barrier option value using COS-GPR is described as follows:

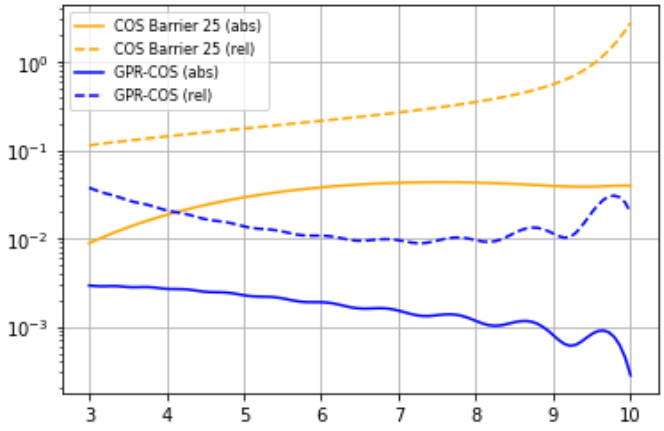
$$\boldsymbol{v}_{BAR}(\boldsymbol{x}; t, X_{test}) = e^{-r\Delta t} \sum_{j=0}^{N_\varphi - 1} (\hat{A}_j(X_{test}) C_j(\boldsymbol{x}) - \hat{B}_j(X_{test}) S_j(\boldsymbol{x})). \tag{4.35}$$

To illustrate, we consider an example where we set the contract parameters as $T = 1$, $H = 10$, $K = 6$ and $M_{mon} =$, and the risk-free interest rate as $r = 0.01$. The diffusion of the CGMY-model is fixed at $\sigma_B = 0.5$. We use the COS formula with $N = 100$ and integration range $[a,b] = [-3,3]$. Within the COS formula, the ch.f. is represented by $N_\varphi = 30$ expansion terms. These expansion terms are estimated using SVD with $n = 100$ benchmark asset points over a model parameter grid with $m = 10$ points per dimension, resulting in a total of $M = 10000$ training points. The resulting estimation for barrier option prices over a asset grid with test value $X_{test} = (0.05, 0.4, 0.2, 0.4)$ is shown in Figure 4.1, together with the corresponding relative and absolute errors.

We observe improved accuracy in the COS-GPR method compared to an existing approach, as demonstrated by a specific test input. However, it's important to note that this



(a) Comparison of barrier option price estimation with 250 monitoring dates using the COS-GPR method and the COS Barrier method with 25 monitoring dates.



(b) Absolute and relative error plots corresponding to the prediction of the two methods.

Figure 4.1: Comparison of the performance of two methods for barrier option pricing under the CGMY-model.

single example doesn't provide the complete accuracy picture. Specifically, GPR-based estimations tend to exhibit reduced accuracy near the boundaries of the training set. This aspect will be further examined in Chapter 5.

In terms of computational efficiency, the COS Barrier option with 25 monitoring dates demonstrates notable efficiency gains, as evidenced by an approximately 7-fold reduction in the average time required for calculating a single option value along the curve. This comparison is summarized in Table 4.1. Further insights into the computational complexity of our initial method will be presented in the subsequent Chapter 5.

Method	CPU Time (seconds)
COS-GPR	1.10×10^{-3}
COS Barrier 25	7.20×10^{-3}

Table 4.1: Comparison between the COS-GPR method, with $m = 10$, and the COS Barrier 25 method for average CPU times for calculating option prices for inputs in the given test set

5

CONVERGENCE TESTS FOR METHOD 1: COS-GPR METHOD

In this chapter, we conduct a thorough numerical error analysis for pricing barrier options under Lévy processes using the COS-GPR method introduced in Chapter 4. Our main focus is to explore the impact of the number of training points on both computational complexity and prediction accuracy. By systematically varying this parameter, we aim to gain insights into the model's performance under different training scenarios. Subsequently, we investigate the effects of varying the truncation range and the number of terms in the COS method to understand their influence on option price prediction. This step allows us to identify the optimal configuration of the COS-GPR method specifically for the CGMY-model example.

Moreover, we address the challenge of accurately predicting option values near the boundary, a known difficulty in the GPR machine learning method. Evaluating the COS-GPR model's accuracy in this region is crucial to assess its practical applicability and robustness.

To provide a comprehensive evaluation, we compare the computational time required by the COS-GPR method against a lower estimated benchmark. This benchmark serves as a reference for assessing the efficiency of our approach in terms of computational speed. By doing so, we obtain valuable insights into the practical feasibility and computational efficiency of the COS-GPR method for pricing barrier options.

The results of this analysis will contribute to our understanding of the trade-offs between computational complexity and prediction accuracy. Additionally, the findings will shed light on the strengths and limitations of the COS-GPR method in the context of barrier option pricing, helping practitioners make informed decisions when using this innovative pricing technique.

5.1. OPTIMAL SAMPLING FREQUENCY

In the context of estimating barrier option prices using the COS-GPR method, the choice of the number of grid points per dimension, denoted by m, plays a crucial role in determining the accuracy and efficiency of the pricing process. The GPR method is a powerful non-parametric regression technique that utilizes a probabilistic framework to estimate the characteristic function (ch.f.), and the parameter m directly influences the granularity of the grid used for the approximation.

One of the primary considerations is the effect of the accuracy of the barrier option pricing when estimating the ch.f.. A higher value of m corresponds to a denser grid, allowing the GPR model to capture more intricate features and nuances in the survival ch.f.. Consequently, increasing m should lead to more accurate ch.f. estimates, especially in regions where the function exhibits rapid changes or non-linear behavior. However, this does not necessarily guarantee a more accurate barrier option price estimation, as we will see in the subsequent analysis.

For the CGMY-model example, we vary the number m between 3 and 14 points per dimension. The primary focus of this analysis is to investigate how the accuracy of the estimated price behaves as we increase the number of grid points and compare it against the benchmark price. Specifically, we are interested in estimating a discretely monitored up-and-out barrier option under the CGMY-model with $M_{mon} = 250$ monitoring dates.

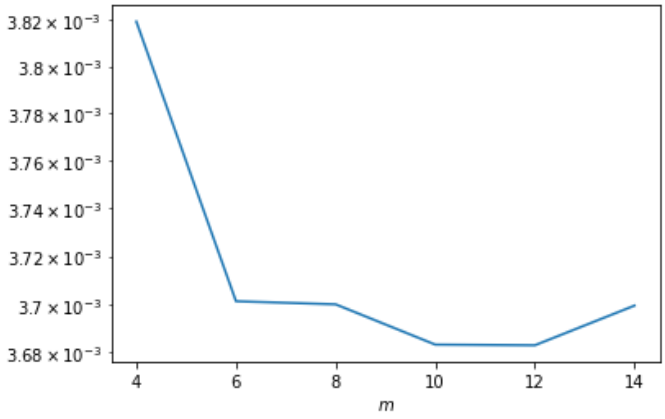


Figure 5.1: Comparison of the L^2-error for the COS-GPR method applied to a test set of barrier options values with a watchtime of $M_{\text{mon}} = 250$, while varying the sampling frequency m.

The analysis includes the L^2-error for different values of m allowing us to comprehensively assess the performance of the COS-GPR method for barrier option pricing under various sampling frequencies. Figure 5.1 presents the comparison of these error metrics for the COS-GPR method applied to a test set of barrier option values $\mathbf{X}_{test} = (0.005 : 0.52 : 6)^4$ with a watchtime of $M_{\text{mon}} = 250$. From the three plots, the sampling frequency is chosen to be set to $m = 10$ based on the observed error behavior.

5.2. OPTIMAL TRUNCATION RANGE OF THE COS METHOD

In this section, we delve into exploring the impact of the truncation range $[a,b]$ on barrier option pricing using the traditional COS method, building upon the insights gained from the optimal sampling frequency analysis in Section 5.1, where $m = 10$ was identified as the most optimal choice. The truncation range $[a,b]$ defines the interval within which the COS method approximates the ch.f., and finding the optimal range is crucial for accurate pricing of barrier options.

As stated in Section 3.2.4, there is no rule-of-thumb for selecting the truncation range for barrier options using the COS method. Therefore, we perform a numerical investigation by considering a wide range of values for c between 0.25 and 75. The parameter c defines the interval as $[-c,c]$, and we keep the sampling frequency m fixed at 10 during this study.

To systematically analyze the behavior, we first examine the errors for different truncation range values. We use the same error metrics, namely L^1-error, L^2-error, and L^∞-error, as employed in the previous analysis. Figure 5.2 presents a comparison of these error metrics for the COS-GPR method with optimal $m = 10$, applied to a test set of barrier option values with a watchtime of $M_{\text{mon}} = 250$, while varying the truncation range $[-c,c]$.

From Figure 5.2a, we observe that the errors vary significantly with different truncation range values. To obtain a more precise analysis, we zoom in on the region where the errors exhibit a notable decrease, specifically between $c = 1.5$ and $c = 3.0$, as shown in Figure 5.2b.

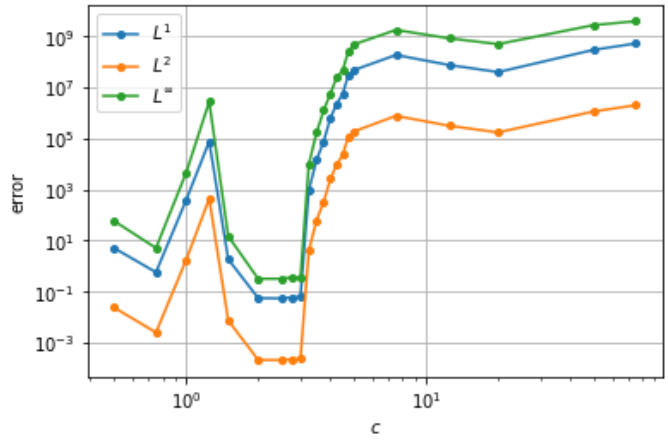
After careful examination of the zoomed-in plot, we identify the optimal choice for the truncation range interval as $c = 2.45$. This value minimizes both the L^1-error and L^2-error, providing significantly improved accuracy compared to other values within the specified range. By selecting $c = 2.45$, we can achieve more precise pricing of barrier options using the COS method in conjunction with the GPR model, enhancing the overall performance and reliability of our approach.

5.3. OPTIMAL NUMBER OF COS TERMS

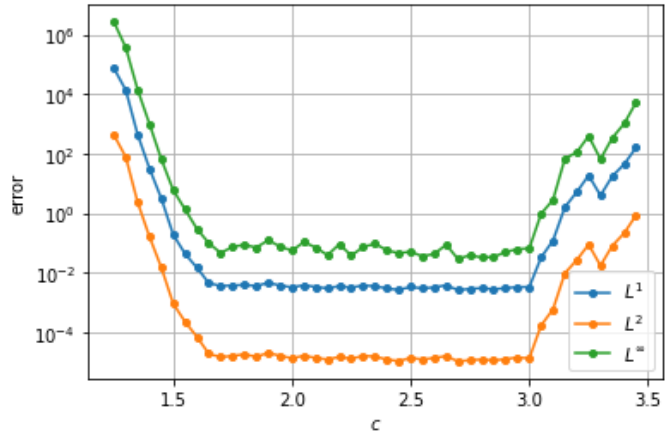
In this section, we explore the impact of the number of expansion terms, denoted by N, on the COS method. Having already established a truncation interval of $[-2.45, 2.45]$ and a fixed sampling frequency of $m = 10$ in previous sections, we now focus on finding the optimal value for N by evaluating the three error metrics. The parameter N is crucial as it influences the complexity of the COS method and how it describes the option price. While the conventional COS method exhibits exponential convergence as N increases for pricing European options [5], our model employs a different expansion to estimate the ch.f.. As a result, the behavior may differ, and a higher N does not necessarily guarantee a better overall approximation.

To conduct the analysis, we vary the values of N within the range of 30 to 200. The same test set utilized in the previous sections will be employed to determine the optimal choice for N. By studying the errors at different N values, we aim to identify the most suitable expansion term count for our model.

Figure 5.3 displays the behaviour of the three error metrics when increasing N. We



(a) Varying c between 0.25 and 75.



(b) Varying c between 1.25 and 3.5.

Figure 5.2: Comparison of three error metrics for the COS-GPR method with optimal $m = 10$ applied to a test set of barrier options values with a watchtime of $M_{\text{mon}} = 250$, while varying the truncation range $[-c, c]$.

find that the optimal choice here is to pick the number of COS terms $N = 96$. For this value all three errors assume their minimum on the selected range of COS terms.

5.4. PERFORMANCE COMPARISON: COS-GPR VS. COS BARRIER METHOD

To observe the improvements from our own method compared to the current benchmark method regarding speed and accuracy, we will show this by estimating the price of a barrier option under the CGMY model with 250 monitoring dates and observe these two properties. The method that we compare our own COS-GPR method with the COS

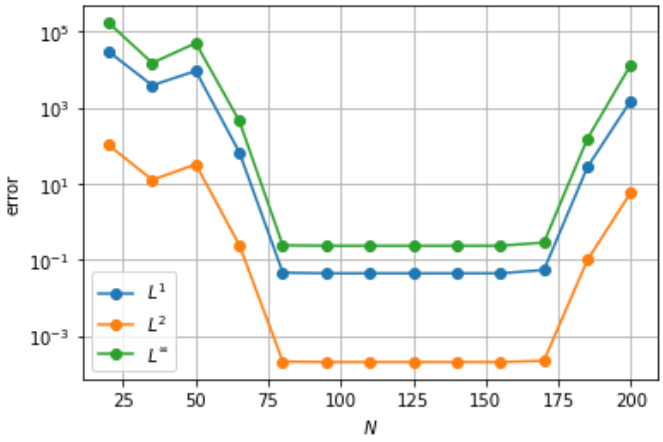


Figure 5.3: Comparison of three error metrics for the COS-GPR method with optimal $m = 10$, truncation range $[-2.45, 2.45]$ applied to a test set of barrier options values with a watchtime of $M_{\text{mon}} = 250$, while varying the number COS terms N.

Barrier method which assumes 25 monitoring dates. Our COS-GPR method uses the optimized settings after initialization of $N_\varphi = 30$, which were derived in previous sections. That is:

- From Section 5.1; the sampling frequency is set to $m = 10$. This implies that the number of training points is $M = m^d = 10000$.
- From Section 5.2; the truncation range, assumed to be symmetric around 0, is $[a, b] = [-c, c]$ with $c = 2.45$.
- From Section 5.3; the number of ch.f. terms in the COS method is set to be $N = 154$.

The performance comparison between the COS-GPR method and the COS Barrier method is done on a test set, which also includes points outside the training set. This is done to get a better overall view of the performance instead of focusing on single input to get a better performance indication. This is a different test set from the one which is used to derive the optimal parameters of the COS-GPR model.

Using a single input we can see the behaviour of the fit of the method to the benchmark. The benchmark here is the option valuated using the COS Barrier method with 250 monitoring dates. We have two test examples

$$\boldsymbol{\theta}_{in} = (C_{in}, G_{in}, M_{in}, Y_{in}) = (0.14, 0.27, 0.15, 0.22),$$
$$\boldsymbol{\theta}_{out} = (C_{out}, G_{out}, M_{out}, Y_{out}) = (0.52, 0.27, 0.15, 0.22),$$

respectively the single test example in and out of the training set. The fits of both samples are plotted in Figure 5.4 for both the COS-GPR method and the COS Barrier method with 25 monitoring dates.

5.4.1. ACCURACY COMPARISON

To perform a more thorough analysis on the prediction capabilities of the COS-GPR and COS Barrier methods, I have decided to extend the analysis on the previous example. The goal is to compare the accuracy of the two methods on a larger test set, which is different from the one used to optimize the model's parameters. This new test set will cover a range of four dimensions with varying values from 0.002 to 10 with a step size of 0.55, defined as $(0.002 : 0.55 : 10)^4$. To compare the two methods, we will iterate over all the different test inputs, generate predictions for each input using both the COS-GPR and COS Barrier methods, and accumulate all the errors to obtain a comprehensive performance evaluation of both models.

Firstly, we will generate a table of the overall errors of all the log-asset prices over all test inputs. These error metrics are the L^1, L^∞, relative L^1, and relative L^∞ errors. For this table these are defined as:

$$L^1\text{-error} = \frac{1}{n \cdot M} \sum_{i=1}^{n} \sum_{j=1}^{M} |v_{bench}(x_i \mid \boldsymbol{\theta}_j) - v_{method}(x_i \mid \boldsymbol{\theta}_j)|, \tag{5.1}$$

$$\text{Relative } L^1\text{-error} = \frac{1}{n \cdot M} \sum_{i=1}^{n} \sum_{j=1}^{M} \left| \frac{v_{bench}(x_i \mid \boldsymbol{\theta}_j) - v_{method}(x_i \mid \boldsymbol{\theta}_j)}{v_{bench}(x_i \mid \boldsymbol{\theta}_j)} \right| \cdot 100\%, \tag{5.2}$$

$$L^\infty\text{-error} = \max_{i,j}(|v_{bench}(x_i \mid \boldsymbol{\theta}_j) - v_{method}(x_i \mid \boldsymbol{\theta}_j)|) \tag{5.3}$$

$$\text{Relative } L^\infty\text{-error} = \frac{\max_{i,j}(|v_{bench}(x_i \mid \boldsymbol{\theta}_j) - v_{method}(x_i \mid \boldsymbol{\theta}_j)|)}{\max_{i,j}(|v_{bench}(x_i \mid \boldsymbol{\theta}_j)|)} \cdot 100\% \tag{5.4}$$

We will look at both the COS-GPR and COS-Barrier method, and their errors have been inserted into Table 5.1, which tells us that the COS-GPR method is for all error metrics more accurate than the COS Barrier 25 method. Additionally, the COS-GPR method with $m = 10$ is 7 times faster than the other method. This table provides a good general indicator of the prediction performance of the two methods. However, these single error metrics do not tell the full story about the fit of the value curve for a single test input. To gain deeper insights, we will perform a more elaborate analysis by looking at these error metrics per test input and collecting them to make a better assessment of the fitting capability of the two methods for each specific input scenario.

Method	Relative L^1 (%)	L^1-error	Relative L^∞ (%)	L^∞-error	CPU (ms)
COS Barrier 25	31.184	$1.681 \cdot 10^{-2}$	25.768	$6.535 \cdot 10^{-2}$	7.204
COS-GPR	1.846	$9.953 \cdot 10^{-4}$	19.889	$5.044 \cdot 10^{-2}$	1.101

Table 5.1: Comparison of our barrier option price estimation method with the COS Barrier method under the CGMY-model using a given set of test values. The errors represent the combined results over all benchmark values for the entire set of test inputs.

The upcoming plots share a conceptual resemblance with an empirical cumulative density function (ECDF), albeit with distinctive characteristics. Our visualization will deviate from the conventional approach of representing cumulative probability below a

given threshold on the vertical axis. Instead, we will illustrate the cumulative count of observations falling under specific error magnitudes.

The selected error metrics for the plot encompass the relative L^1-error across all observations, the relative L^1-error across all inputs, and the L^∞-error across all inputs. It's important to note that these definitions differ from those outlined in equations (5.1) to (5.4) , and we'll delve into their nuances in subsequent explanations.

To provide clarity, let's establish that for a specific error magnitude presented along the horizontal axis, the corresponding point on the vertical axis will denote the count of observations within the test set that exhibit an error magnitude equal to or less than the specified value. Multiple categories of errors will be taken into consideration, yielding distinct outcomes in the graphical representations.

This methodology offers a robust means to enhance our comprehension of the COS-GPR and COS Barrier methodologies' performance across an expansive test dataset. Consequently, it facilitates a more efficient and insightful comparison of their predictive capacities. By adopting this analytical approach, we can discern which approach excels comprehensively and in varied error metric scenarios, thereby aiding us in making a comprehensive assessment of their relative merits.

In this analysis, we employ three distinct error metrics to evaluate the performance disparity between the COS-GPR method and the COS Barrier 25 method. Let's delve into the intricacies of each error metric and its implications:

1. **Relative L^1-error over all observations (Figure 5.5a):**

   This metric centers on the relative absolute error within every observation. For each test input $\boldsymbol{\theta} \in X_{\text{test}}$ and for each log-asset price x, the error computation takes this form:

   $$\text{Relative } L^1\text{-error} = \frac{|v_{\text{bench}}(x|\boldsymbol{\theta}) - v_{\text{method}}(x|\boldsymbol{\theta})|}{v_{\text{bench}}(x|\boldsymbol{\theta})} \times 100\% \tag{5.5}$$

   The resulting plot manifests the cumulative distribution of these relative errors. The visualization distinctly showcases that the COS-GPR method consistently surpasses the COS Barrier 25 method, consistently yielding lower error values. It's noteworthy that the paths of the two methods never intersect, denoting that no error threshold exists wherein COS Barrier 25 outperforms COS-GPR.

2. **Relative L^1-error over the whole curve for a given input (Figure 5.5b):**

   In contrast to the prior metric, this assessment examines the complete value curve for a given input, rather than singular values. The error computation entails:

   $$\text{Relative } L^1\text{-error} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{v_{\text{bench}}(x_i|\boldsymbol{\theta}) - v_{\text{method}}(x_i|\boldsymbol{\theta})}{v_{\text{bench}}(x_i|\boldsymbol{\theta})}\right| \times 100\% \tag{5.6}$$

   The resultant graph visualizes the cumulative distribution of this error across the entire spectrum of log-asset prices for each test input $\boldsymbol{\theta}$. Similarly, the COS-GPR method consistently exhibits superior performance over COS Barrier 25. Notably, while COS-GPR maintains an overall advantage, the maximum relative L^1-error of

the COS Barrier method is marginally smaller, implying potential instances where COS-GPR misestimates the complete value curve. This metric proves instrumental in uncovering structural vulnerabilities within the COS-GPR method during full value curve plotting.

3. **Relative L^∞-error (Figure 5.5c):**

   The relative L^∞-error gauges the utmost absolute percentage deviation between value predictions and benchmark values across an entire log-asset price grid for each input $\boldsymbol{\theta}$. The calculation is defined as:

$$\text{Relative } L^\infty\text{-error} = \frac{\max_i(|v_{\text{bench}}(x_i|\boldsymbol{\theta}) - v_{\text{method}}(x_i|\boldsymbol{\theta})|)}{\max_i(v_{\text{bench}}(x_i|\boldsymbol{\theta}))} \times 100\% \tag{5.7}$$

   The graph universally underscores COS-GPR's superiority over COS Barrier 25, with COS Barrier's errors ranging between 20% – 40% while COS-GPR consistently maintains lower errors. Yet, the COS-GPR method does incur some exceptional errors not mirrored by COS Barrier. This error metric provides valuable insights into the specific scenarios wherein COS-GPR falters in accurate predictions, notwithstanding its overall robust performance.

In summation, the analysis unequivocally affirms COS-GPR's prevailing accuracy over COS Barrier 25. However, it is equally imperative to consider the nuanced contexts wherein each method may exhibit strengths or vulnerabilities, as highlighted through the diverse error metrics employed in this evaluation.

5.4.2. CPU COMPARISON

The computational time of our model is influenced by the device's performance.[1] . In this section, we will derive the analytical asymptotic time complexity for a general d-dimensional model. Subsequently, we will analyze the computational behavior using Geometric Brownian Motion (GBM) with $d = 2$, while varying the sampling frequency m. GBM is chosen as our device can handle only up to $m = 14$ for a 4-dimensional CGMY model. Thus, to assess the empirical time complexity over a broader range, we opt for the GBM approach.

ANALYTICAL TIME COMPLEXITY COS-GPR METHOD

The computational complexity of calculating the ch. f. for the COS method using GPR can be obtained by looking at the prediction procedure. To recall; given a training set $(\mathbf{X}, \mathbf{y})$. In general, for test inputs $\mathbf{X}_{test} \in \mathbb{R}^{m_{test} \times d}$ the predicted value $\mathbf{f}_{test}$ is given by :

$$\mathbf{f}_{test} = K(\boldsymbol{X}_{test}, \boldsymbol{X}) \left[K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2_{GPR} \boldsymbol{I}\right]^{-1} \boldsymbol{y},$$

where K the kernel matrix, is determined by the kernel function $k(\cdot, \cdot)$. The computational of this prediction can be described as $\mathcal{O}((m_{test} \cdot M)^2 \cdot d)$, by observing that:

[1] For this research, we utilized a Intel(R) Core(TM) i7-10700F CPU @ 2.90GHz (8 CPU cores), with the code written in Python 3.8.10.

- The prediction can be written as $\mathbf{f}_{test} = K(\boldsymbol{X}_{test}, \boldsymbol{X})\mathbf{v}$, where $\mathbf{v} \in \mathbb{R}^M$ can be calculated beforehand. Resulting in a regular matrix-vector product with $\mathcal{O}(m_{test}M)$.
- The time it takes to compute $K(\boldsymbol{X}_{test}, \boldsymbol{X})$ using the Squared Exponential (SE) kernel, defined in (4.3), is approximately $\mathcal{O}((m_{test} \cdot M)^2 \cdot d)$ for $m_{test} \cdot M$ matrix elements in d dimensions. This complexity arises from calculating the squared Euclidean distance for each pair of data points and computing the exponential function for each entry in the $m_{test} \times m$ matrix.
- This results in a complexity of $\mathcal{O}((m_{test} \cdot M)^2 \cdot d) + m_{test}M)$, which is the same as $\mathcal{O}((m_{test} \cdot M)^2 \cdot d)$.

From now on we assume that we are looking at a single test input $\mathbf{x}_{test}$ Furthermore, it was assumed that we have m training points per dimensions, i.e. $M = m^d$, which results in a complexity of $\mathcal{O}(m^{2d} \cdot d)$. Using this complexity we can obtain the complexity for calculating barrier option prices using GPR1 and GPR 2 model (from Section 4.2.1 and 4.2.2 respectively).

5

COMPUTATIONAL COMPLEXITY GPR1 MODEL

For a given test value $\boldsymbol{\theta}$ and log-asset price x time complexity of calculating a single barrier option value using GPR1 is analytically derived to be $\mathcal{O}(Nm^{2d}d)$, since:

- Calculating the ch.f. terms $\mathbf{R}(\boldsymbol{\theta})$ and $\mathbf{I}(\boldsymbol{\theta})$ will have a time complexity of $\mathcal{O}(Nm^{2d}d)$,
- The calculating of the sine and cosine part, which is dependent on x, is done seperately from the calculation of the expansion terms using GPR. This will have complexity $\mathcal{O}(N)$ and can be neglected,
- The sum of the COS method can be seen as a inner product with complexity $\mathcal{O}(N)$,
- Combining the three processes the complexity will be $\mathcal{O}(Nm^{2d}d)$.

COMPUTATIONAL COMPLEXITY OF THE GPR2 MODEL

For a given test value $\boldsymbol{\theta}$ and log-asset price x, the time complexity of calculating a single barrier option value using the GPR2 method is given by $\mathcal{O}(N_\varphi m^{2d}d)$, where:

- Calculating the Fourier expansion terms of the ch.f. $\mathbf{A}(\boldsymbol{\theta})$ and $\mathbf{B}(\boldsymbol{\theta})$ has a time complexity of $\mathcal{O}(N_\varphi m^{2d}d)$,
- Calculating the ch.f. coefficients using the Fourier expansion can be seen as an inner product with $2N_\varphi$ elements. This results in a complexity of $\mathcal{O}(N_\varphi)$,
- The sum of the COS method can be seen as an inner product with complexity $\mathcal{O}(N)$,
- Since all processes are calculated separately, the total complexity results in $\mathcal{O}(N_\varphi m^{2d}d + N_\varphi + N)$, which simplifies to $\mathcal{O}(N_\varphi m^{2d}d)$.

It's important to note that for both methods, the number of monitoring dates is not included in the complexity of COS-GPR. However, for pricing options with the COS Barrier method, the computational complexity is equal to $\mathcal{O}((M_{mon}-1)N\log_2(N))$, which was illustrated in Section 2.6. This means that for especially higher monitoring dates, the CPU time of COS-GPR should be significantly lower than that of the COS Barrier method. This will be shown in the next subsection.

By having a lower computational complexity, the COS-GPR method is more efficient, allowing for faster pricing of barrier options compared to the COS Barrier method, especially when considering a larger number of monitoring dates. The advantages of the COS-GPR method in terms of computational speed become more apparent as the problem size increases.

OBSERVED COMPUTATIONAL COMPLEXITY

To compare the analytical computational complexity with empirical results, we conduct tests to observe the CPU time behavior as the sampling frequency increases. Due to memory limitations at $m = 14$ for the 4-dimensional CGMY model, we opt to use the GBM model for our observations. By selecting a lower value of d and utilizing GBM, we can assume larger values of m, enabling a more comprehensive analysis of the CPU time behavior.

The logarithmic linear relationship in the plots suggests exponential complexity, despite the CGMY and GBM models having expected polynomial complexities of $\mathcal{O}(m^8)$ and $\mathcal{O}(m^4)$, respectively. This contradictory observation could be due to the dataset's unique characteristics fact that polynomial functions, particularly with higher degrees, can mimic exponential growth within smaller value ranges. Although polynomial growth slows as values move away from zero[2], this initial resemblance to exponential behavior might explain the linear-like trend in the data.

[2] A known result from analysis shows that for any $k \in \mathbb{R}$, the limit

$$\lim_{x\to\infty} \frac{x^k}{e^x} = 0$$

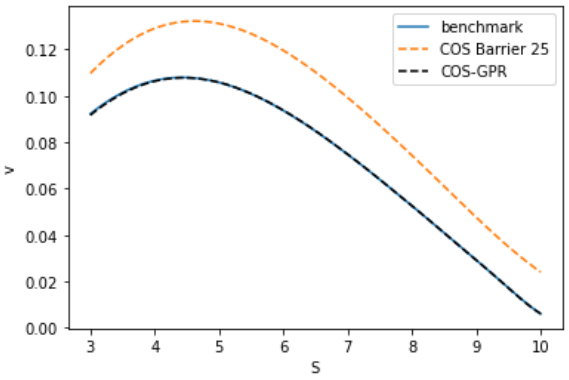
.

m	CPU Time (seconds)
4	1.700×10^{-5}
6	3.993×10^{-5}
8	1.197×10^{-4}
10	4.289×10^{-4}
12	1.576×10^{-3}
14	4.467×10^{-3}

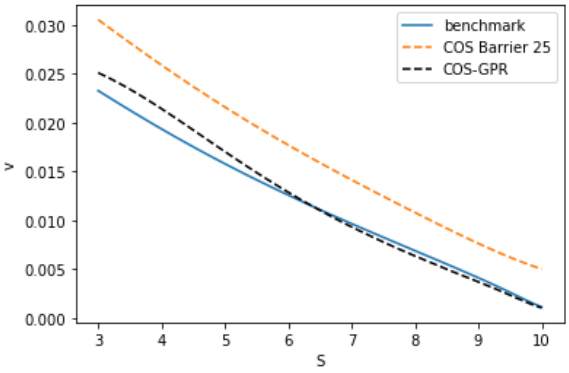
Table 5.2: Values of m and Corresponding CPU Times

m	CPU Time (seconds)
4	$9.40005779 \times 10^{-6}$
10	$1.09817505 \times 10^{-5}$
16	$1.10027313 \times 10^{-5}$
22	$1.09834671 \times 10^{-5}$
28	$1.40339851 \times 10^{-5}$
34	$1.57017946 \times 10^{-5}$
40	$2.03879833 \times 10^{-5}$
46	$2.66747952 \times 10^{-5}$
52	$3.60366344 \times 10^{-5}$
58	$5.01916409 \times 10^{-5}$
64	$6.74517632 \times 10^{-5}$
70	$7.99804688 \times 10^{-5}$
76	$1.08228874 \times 10^{-4}$
82	$1.38113737 \times 10^{-4}$
88	$1.77308679 \times 10^{-4}$
94	$2.32108641 \times 10^{-4}$
100	$2.99650669 \times 10^{-4}$
106	$3.56075740 \times 10^{-4}$
112	$4.28155684 \times 10^{-4}$
118	$5.30181503 \times 10^{-4}$

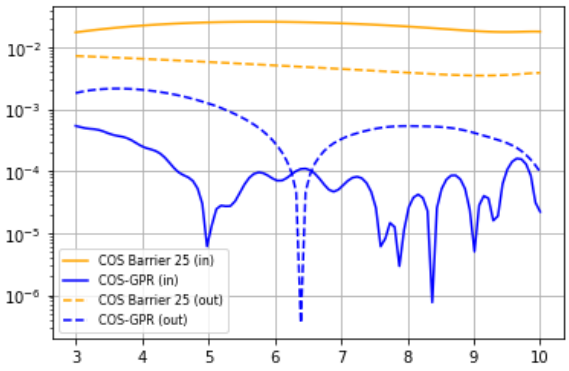
Table 5.3: Values of m and Corresponding CPU Times



(a) Sample from the training set: COS-GPR fitting barrier options prices.

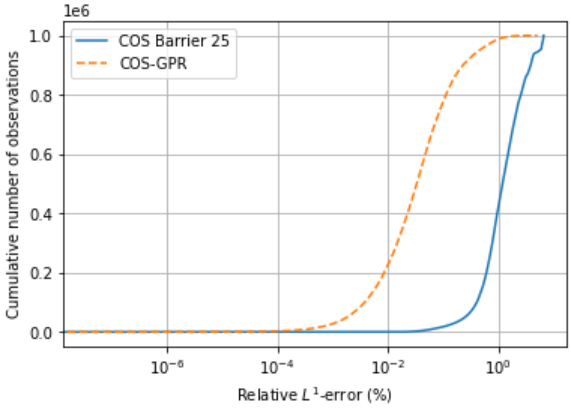


(b) Sample from outside the training set: COS-GPR fitting barrier options prices.

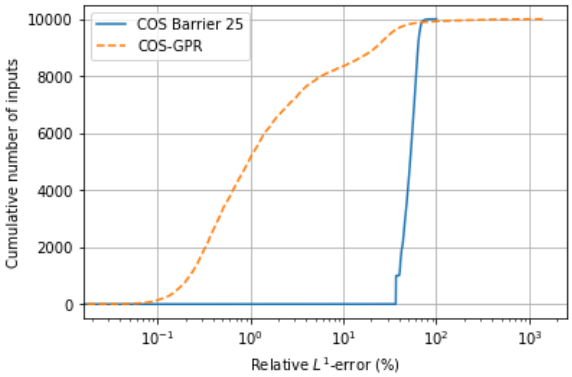


(c) Analyzing Barrier Option Price Errors: Contrasting Examples from Within and Outside the Training Set

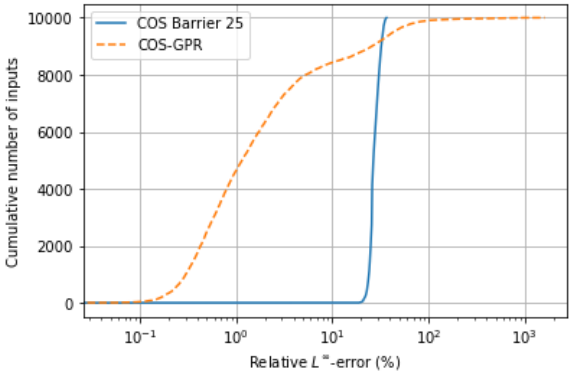
Figure 5.4: Comparison of the COS-GPR method with the COS Barrier method for fitting barrier options prices for two samples: one from the training set (a) and one from outside the training set (b). Additionally, the absolute errors of the two examples are shown in (c).



(a) Relative L^1-error over all observations.

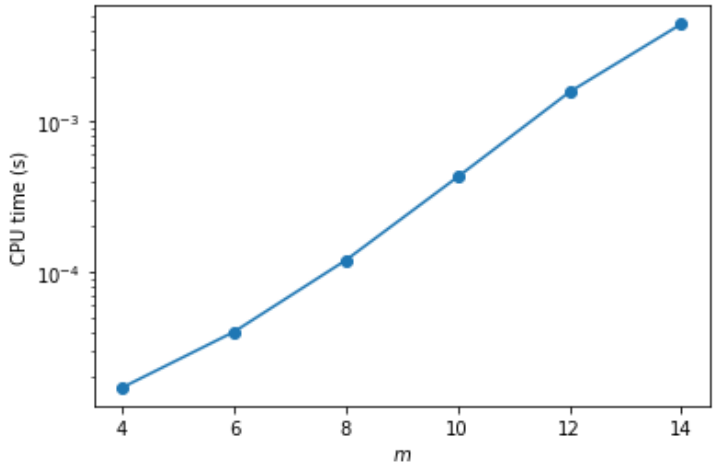


(b) Relative L^1-errors over all inputs.

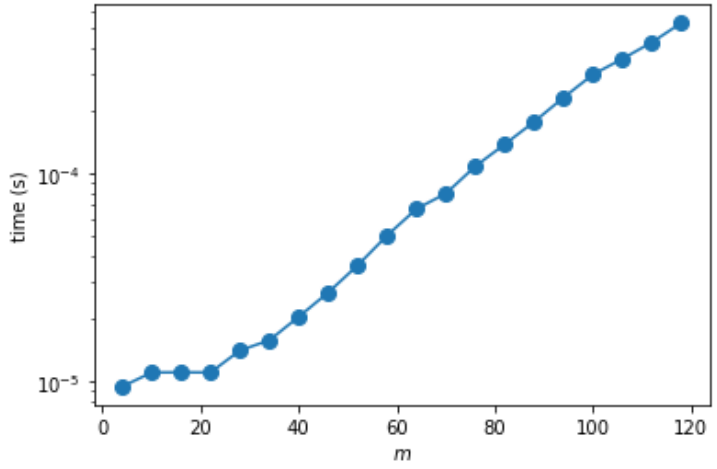


(c) Relative L^∞-errors over all inputs.

Figure 5.5: Three plots illustrating the distribution of observations based on the magnitude of an error metric. The horizontal axis represents the error metric's value, while the vertical axis indicates the cumulative number of observations/input below each corresponding error threshold.



(a) CGMY



(b) GBM

Figure 5.6: CPU observations as m varies for two distinct stochastic underlying processes using the COS-GPR method.

6

OUR METHOD 2 FOR OPTION PRICING: THE CFC METHOD

This chapter introduces our second method for option pricing: the CFC method. The abbreviation comes from the fact that the method uses the COS Method (C), a Fourier-cosine expansion on the ch.f. (F) and the CPD method (C) . To understand the method an introduction is given about Canonical Polyadic Decomposition (CPD), which is heavily reliant on tensor calculus, and how this is used in combination with the COS method to price barrier (and European) options. The motivation for this second model is that compared to the first model it makes use of a global decomposition, namely the one which using Fourier-cosine basis functions. While the COS-GPR method focuses on a localised decomposition, which is a reason that it lacks performance for predictions near the training boundary. Furthermore, CPD applied on the Fourier-cosine expansion decreases the dimension of the model drastically when dealing with higher-dimensional models, e.g. 4-dimensional.

6.1. BACKGROUND INFORMATION: CANONICAL POLYADIC DECOMPOSITION (CPD)

The notation used here is the same as in [20], which was also employed in previous Msc these of FF Quant [24] and [25]. Furthermore, all required definitions of tensor calculus can be found in Section 2.7. The CPD technique involves the factorization of an dth-order tensor into a sum of component rank-one tensors. In this context, rank-one tensors are of central importance. A dth-order tensor $\mathscr{X}$ requires d sets of indices $\{i_n\}$, $n = 1,\ldots,d$, to specify a specific element. Each index i_n ranges from 1 to I_n, representing the mode-n of tensor $\mathscr{X}$. As a convention, an dth-order tensor can be explicitly represented as $\mathscr{X} = \mathbb{R}^{I_1 \times \cdots \times I_d}$, where I_n for $n = 1,\ldots,d$ directly indicates the number of elements in the n-th mode. A tensor $\mathscr{X}$ of order d is considered rank-one if it can be expressed as an outer product of d vectors. i.e.,

$$\mathscr{X} = a_1 \circ a_2 \circ \cdots \circ a_d = \circ_{i=1}^{d} a_i. \tag{6.1}$$

By employing CPD, a dth-order tensor can be factorized as:

$$\mathscr{X} = [\![\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N]\!]_R = \sum_{r=1}^{R} \circ_{i=1}^{d} a_{ir}, \tag{6.2}$$

where $\boldsymbol{A}_n = [a_{n1}, a_{n2}, \ldots, a_{nR}] \in \mathbb{R}^{I_n \times R}$ for $n = 1, 2, \ldots, d$. The rank R is defined as the minimum number of components required to synthesize tensor $\mathscr{X}$, this is explained. This minimal rank is commonly referred to as the canonical rank. It is worth noting that every tensor can be represented by a CPD with $R < \infty$ and, under mild conditions, this representation is unique [26]. However, determining the canonical rank is a challenging problem in the class of NP-hard problems. This implies that there is no algorithm to find its exact value [27]. It can only be checked in polynomial time if the solution is validate. In practice, a lower rank $\tilde{R}$ is often employed to approximate the original tensor:

$$\mathscr{X} \approx \sum_{r=1}^{\tilde{R}} \circ_{i=1}^{d} a_r^i. \tag{6.3}$$

These lower-rank CPD models are widely utilized to mitigate the curse of dimensionality. The expression in Equation (3.1) can also be represented element-wise as follows:

$$\mathscr{X}[i_1, i_2, \ldots, i_d] \approx \sum_{r=1}^{R} \prod_{i=1}^{d} (\mathbf{A}_n[i_n, r]) = \sum_{r=1}^{R} \left(\prod_{n=1}^{d} a_{jr}[i_n] \right), \tag{6.4}$$

where $a_{nr}[i_n]$ denotes the element of the vector a_{nr} at index i_n. This description of the tensor describes that it can be build up by the product of all seperate dimension individually. This relation represents the tensor into its factor matrices, which is called its CPD. It is important to find the corresponding factor matrices given R,N and d to get an accurate CPD.

SOLVING THE FACTOR MATRICES

The CPD of $\mathscr{X}$ is found by find the optimal factor matrices. That is a loss function (e.g. least-squares) is used to find the factor matrices that minimize the distance between the original tensor and the given CPD approximation:

$$\min_{\{\mathbf{A}_i\}_{i=1}^{d}} \left\| \mathscr{X} - [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]_R \right\|_F^2 = \min_{\{\mathbf{A}_i\}_{i=1}^{d}} \left\| \mathscr{X} - \sum_{r=1}^{R} \circ_{i=1}^{d} a_r^i \right\|_F^2. \tag{6.5}$$

Here F denoted the Frobenius norm of the tensor defined in (2.7.5). To solve this minimizaiton problem, we combine the methodology of a previous thesis done at FF Quant [28] and [26]. Here Alternating Least Squares (ALS) is used to iteratively update the factor matrices. This is done by solving all factor matrices individually in a cyclical manner. Hence all matrices are fixed except for the one that is being solved, which results in a conditionally linear problem. The solution is the new update, and using this the next factor matrix is solved. Using tensor unfolding and the Khatri-Rao product, we obtain a

set of minimization problems for the factor matrices, which is equivalent to solving (6.5). The set of minimization problems are for the mode unfoldings $n = 1, ..., d$:

$$\min_{\{\mathbf{A}_n\}_{n=1}^{N}} \left\| \mathscr{X}_{(n)} - \mathbf{A}_n \left(\odot_{i \neq n} \mathbf{A}_i \right)^T \right\|_F^2, \tag{6.6}$$

where $\mathscr{X}_{(n)}$ is the mode-n unfolding of the tensor $\mathscr{X}$. Because of the multilinearity of the factor matrices one can apply the ALS algorithm by fixing every factor matrix except for $\mathbf{A}_n$. These are updated iteratively, until a stopping criterion is triggered, for $n = 1, ..., d$ by:

$$\mathbf{A}_n \leftarrow \underset{\mathbf{A}_n}{\operatorname{argmin}} \left\| \mathscr{X}_{(n)} - \mathbf{A}_n \left(\odot_{i \neq n} \mathbf{A}_i \right)^T \right\| \tag{6.7}$$

6.2. REDUCING DIMENSIONALITY OF FOURIER-COSINE SERIES EXPANSION USING CPD

This section describes how CPD is applied to reduce the dimensionality of the Fourier-cosine expansion for multivariate functions, which has already been analysed in a previous thesis by FF Quant [28]. Such an expansion is needed in our research, since the ch.f. is also a multivariate function.

To understand the structure of how CPD works for multivariate functions, first the Fourier-cosine series expansion of a function $f(x)$ on interval $[a, b]$ is shown here. Furthermore, we describe the number of basis functions with K instead of N to distinct that we are not applying the expansion on the COS method for option pricing:

$$f(x) = {\sum_{k=0}}' {}^{\infty} A_k \cos\left(k\pi \frac{x-a}{b-a} \right) \approx {\sum_{k=0}}' {}^{K-1} A_k \cos\left(k\pi \frac{x-a}{b-a} \right), \tag{6.8}$$

with the one-dimensional Fourier-cosine coefficients A_k being

$$A_k = \frac{2}{b-a} \int_a^b f(x) \cos\left(k\pi \frac{x-a}{b-a} \right) dx. \tag{6.9}$$

For the first dimension the coefficients A_k can easily be stored in a vector $\mathscr{A} = [A_0, A_1, ..., A_{N-1}]$. To describe it more generally; this is a first-order tensor.

Extending the Fourier-cosine transform to a two-dimensional function $f(x_1, x_2)$ defined on $[a_1, b_1] \times [a_2, b_2]$ the approximation of the Fourier-cosine expansion of f becomes

$$f(x_1, x_2) \approx {\sum_{k_1=0}}' {}^{K-1} {\sum_{k_2=0}}' {}^{K-1} \cos\left(k_1\pi \frac{x_1-a_1}{b_1-a_1} \right) \cos\left(k_2\pi \frac{x_2-a_2}{b_2-a_2} \right), \tag{6.10}$$

with two-dimensional Fourier-cosine coefficients given by

$$A_{k_1,k_2} = \frac{2}{b_1-a_1} \frac{2}{b_2-a_2} \int_{a_1}^{b_1} \int_{a_2}^{b_2} f(x_1, x_2) \cos\left(k_1\pi \frac{x_1-a_1}{b_1-a_1} \right) \cos\left(k_2\pi \frac{x_2-a_2}{b_2-a_2} \right) dx_1 dx_2. \tag{6.11}$$

These coefficients $A_{k_1 k_2}$ can be stored in a $K \times K$-dimensional matrix (second order tensor) where the matrix $\mathbf{A}$ is

$$A = \begin{bmatrix} A_{0,0} & A_{0,1} & \cdots & A_{0,N-1} \\ A_{1,0} & A_{1,1} & & \vdots \\ \vdots & & \ddots & \vdots \\ A_{N-1,0} & \cdots & \cdots & A_{N-1,N-1} \end{bmatrix}. \tag{6.12}$$

Now to generalise it over all dimensions; for a $d$dimensional function $f(x_1,...,x_d)$ defined on $\times_{i=1}^{d}[a_i,b_i]$ the Fourier-cosine transform is the following:

$$f(x_1,\ldots,x_d) \approx \sum_{k_1=0}^{\prime K-1}\cdots\sum_{k_d=0}^{\prime K-1} A_{k_1,\ldots,k_d}\prod_{i=1}^{d}\cos\left(k_i\pi\frac{x_i-a_i}{b_i-a_i}\right), \tag{6.13}$$

with d-dimensional Fourier-cosine series coefficients defined as

$$A_{k_1,\ldots,k_d} = \left[\prod_{i=1}^{d}\frac{2}{b_i-a_i}\right]\int_{a_1}^{b_1}\cdots\int_{a_d}^{b_d} f(x_1,\ldots,x_d)\prod_{i=1}^{d}\cos\left(k_i\pi\frac{x_i-a_i}{b_i-a_i}\right)dx_1\cdots dx_d. \tag{6.14}$$

These coefficients are represented by a tensor of the dth order. This means that a coefficient $A \in \mathbb{R}^{K\times K\cdots\times K}$. This implies that with a d-dimensional function, the number of Fourier-cosine coefficients grow exponentially as $\mathcal{O}(K^d)$. With this also the computational time of the d-dimensional integral grows as the dimension increases when computed numerically.

6

APPLYING CPD

This exponential growth in the number of Fourier-cosine terms becomes computationally too heavy when estimating all these coefficients. Here, CPD is applied to describe this tensor in a lower dimension. To do this using the CPD-technique; the d-dimensional tensor of the coefficients is approximated using a sum of rank-1 tensor:

$$\mathcal{A} \approx [\mathbf{A}_1,\mathbf{A}_2,...,\mathbf{A}_d]_R = \sum_{r=1}^{R}\circ_{i=1}^{d} a_{ir}. \tag{6.15}$$

Notice that the set of factor matrices $\{\mathbf{A}_i\}_{i=1}^{d}$, with $\mathbf{A}_i \in \mathbb{R}^{K\times R}$ have a computational complexity of $\mathcal{O}(dKR)$. Hence the computational complexity went from $\mathcal{O}(K^d)$ to $\mathcal{O}(dKR)$, hence as the dimension of the input d increases, the computational time only grows linearly instead of exponentially. These factor matrices are computed iteratively using ALS described in Section 6.1.

When the stopping criterion of the ALS procedure has been triggered and all factor matrices $\{\mathbf{A}_i\}_{i=1}^{d}$ have been derived. The coefficients of the dth-order tensor with Fourier-cosine coefficients can be written as

$$A_{k_1,\ldots,k_d} = \mathcal{A}[k_1,\ldots,k_d] = \sum_{r=1}^{R}\prod_{i=1}^{d} a_r^i[k_i]. \tag{6.16}$$

Substituting this into the Fourier-cosine expansion (6.13) gives

$$f(x_1,\ldots,x_d) \approx \sum_{k_1=0}^{\prime K-1}\cdots\sum_{k_d=0}^{\prime K-1}\sum_{r=1}^{R}\prod_{i=1}^{d} a_r^i[k_i]\cos\left(k_i\pi\frac{x_i-a_i}{b_i-a_i}\right). \tag{6.17}$$

For convenience, we denote $\mathbf{v}_i[k_i] := \cos\left(k_i\pi\frac{x_i-a_i}{b_i-a_i}\right)$. This when filled in into (6.17) simplifies to

$$f(x_1,...,x_d) \approx \sum_{r=1}^{R}\prod_{i=1}^{d} f_{i,r}(x_i) = \sum_{r=1}^{R}\prod_{i=1}^{d}\mathbf{v}_i^T\mathbf{A}_i[:,r]. \tag{6.18}$$

This means that every $f_{i,r}(x_i) = \mathbf{v}_i^T\mathbf{A}_i[:,r]$ describes a one-dimensional Fourier-cosine expansion (for index i). To write it more conveniently using the Hadamard product we get:

$$\hat{f} = \left(\mathbf{v}_1^T\mathbf{A}_1 \circledast \cdots \circledast \mathbf{v}_d^T\mathbf{A}_d\right)\vec{1} = \left(\circledast_{i=1}^{d}\mathbf{v}_i^T\mathbf{A}_i\right)\vec{1}. \tag{6.19}$$

Hence the d-dimensional expansion in (6.13) can be written as a sum of products. Here all sum elements are products of the d univariate expansions, one expansion per input dimension. This eliminates the presence of nested sums from the original expansion, which removes the exponential aspect in the computational time. In the next section this principle is applied on our research, which relates to the estimation of the ch.f. Here the ch.f. has a multidimensional input consisting of the model parameters, which in future examples for the CGMY-model is already 4.

6.3. OUR METHOD 2 FOR BARRIER OPTION PRICING: THE CFC METHOD

In our study, we have introduced a second method for assessing barrier options, referred to as the CFC method. In this section, we will delve into the three sequential steps comprising the CFC method, designed to determine the price of a barrier option.

The initial layer of the method involves employing the 1D COS method for pricing European options, which we extensively expound upon in Chapter 3. This forms the fundamental basis of our findings.

Subsequently, the second layer entails expressing the survival characteristic function as a Fourier-series expansion. impose the binding relation of the Fourier series coefficients. That is, the 1D COS formula indicates that the Fourier series coefficients are actually sampled from the same function. Please note that this insight has led to accurate results in the GPR2 method.

Moving deeper, the third and innermost layer focuses on dimension reduction. Here, we employ Canonical Polyadic Decomposition (CPD) to simplify the expansion's complexity. This intricate maneuver streamlines the problem, making it more manageable.

As we progress through these steps, we generate stored tensors. These tensors serve as the foundation for subsequent computations, enabling us to determine the prices of new barrier options within the context of a given Lévy Process. This method empowers us to perform fresh calculations and expand our ability to price diverse barrier options effectively.

6.3.1. INTRODUCTION TO THE CFC METHOD

PART 1: COS METHOD APPLIED ON BARRIER OPTIONS

To recall, to evaluate barrier options using the traditional COS method the following expression is derived in Chapter 3:

$$v_{Bar}(x;t,\boldsymbol{\theta}) \approx e^{-r_{\boldsymbol{\theta}}\Delta t} \sum_{k=0}^{\prime N-1} \text{Re}\left\{\bar{\varphi}_{levy}\left(\frac{k\pi}{b-a}\right)\exp\left(ik\pi\frac{x-a}{b-a}\right)\right\} V_k.$$

Using benchmark values the survival ch.f. $\bar{\varphi}_{levy}$ can be derived using SVD. In Chapter 3 two different representations of the ch.f. were constructed. This method focuses on representation 2 from Section 3.2.2.

PART 2: FOURIER-COSINE EXPANSION OF THE CH.F.

In this representation the ch.f. was written as a Fourier-cosine expansion, which is defined in (3.14). When entering this expansion inside the COS method for barrier option this resulted in the final form:

$$\boldsymbol{v}_{Bar}(\boldsymbol{x};t,\boldsymbol{\theta}) \approx e^{-r_{\boldsymbol{\theta}}\Delta t} \sum_{j=0}^{N_\varphi-1} (A_j C_j(\boldsymbol{x}) - B_j S_j(\boldsymbol{x})),$$

where the definition of $C_j(\mathbf{x})$ and $S_j(\mathbf{x})$ can found at the derivation above (A.10). Note that in this model the model parameters $\boldsymbol{\theta}$ are fixed. Since the ch.f.,which is the function we want to estimate, is dependent on $\boldsymbol{\theta}$ we now assume the following form:

$$\boldsymbol{v}_{Bar}(\boldsymbol{x};t,\boldsymbol{\theta}) \approx e^{-r_{\boldsymbol{\theta}}\Delta t} \sum_{j=0}^{N_\varphi-1} (A_j(\boldsymbol{\theta}) C_j(\boldsymbol{x}) - B_j(\boldsymbol{\theta}) S_j(\boldsymbol{x})), \tag{6.20}$$

where the Fourier-cosine coefficients of the ch.f. change depending on the value of the model parameters $\boldsymbol{\theta}$. Note that the cosine and the sine part of the sum is not dependent on $\boldsymbol{\theta}$. Hence the functions we want to estimate are $A_j(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$ and $B_j(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$ for $j = 0, \ldots, N_\varphi - 1$, where $d = \dim(\boldsymbol{\theta})$ is the number of model parameters.

Now assuming a multidimensional cosine-expansion for these functions per j. This results in:

$$A_j(\boldsymbol{\theta}) = \sum_{k_1=0}^{\prime K-1} \cdots \sum_{k_d=0}^{\prime K-1} D_{j,k_1,\ldots,k_d} \prod_{i=1}^{d} \cos\left(k_i\pi\frac{\boldsymbol{\theta}_i - a_i}{b_i - a_i}\right) \tag{6.21}$$

$$B_j(\boldsymbol{\theta}) = \sum_{k_1=0}^{\prime K-1} \cdots \sum_{k_d=0}^{\prime K-1} E_{j,k_1,\ldots,k_d} \prod_{i=1}^{d} \cos\left(k_i\pi\frac{\boldsymbol{\theta}_i - a_i}{b_i - a_i}\right), \tag{6.22}$$

which implies that the value of the barrier options is now described as

$$\boldsymbol{v}_{Bar}(\boldsymbol{x};t,\boldsymbol{\theta}) \approx e^{-r_{\boldsymbol{\theta}}\Delta t} \sum_{k_1=0}^{\prime K-1} \cdots \sum_{k_d=0}^{\prime K-1} \sum_{j=0}^{N_\varphi-1} (D_{j,\mathbf{k}} C_j(\boldsymbol{x}) - E_{j,\mathbf{k}} S_j(\boldsymbol{x})) \prod_{i=1}^{d} \cos\left(k_i\pi\frac{\boldsymbol{\theta}_i - a_i}{b_i - a_i}\right). \tag{6.23}$$

This means that per j we want to estimate two $K \times K \times \cdots \times K$ dimensional tensors $D_{j,\mathbf{k}}$ and $E_{j,\mathbf{k}}$, which results in a computational complexity $\mathcal{O}(2K^d N_\varphi)$ unknowns.

PART 3: CPD OF THE FOURIER-COSINE EXPANSION

The CPD part comes into play when we want to estimate these functions. The same procedure is followed as in a previous thesis of FF Quant. Additionally, for our model this procedure has to be done $2N_\varphi$ times, compared to only a single time in the other research paper.

Using CPD applied on Fourier-cosine series, which is described in Section 6.2, the dimension of (6.21) will drop. For this problem CPD reduces the computational complexity from $\mathcal{O}(2K^d N_\varphi)$ to $\mathcal{O}(2dKRN_\varphi)$. If we fix $j = 0,...,N_\varphi - 1$, we observe that for the factor matrices $\{\mathbf{D}_i^j\}_{i=1}^d$ and $\{\mathbf{E}_i^j\}_{i=1}^d$ of the tensors of the Fourier-expansion terms $D_\mathbf{k}$ and $E_\mathbf{k}$, the CPD of the expansion terms will be

$$A_j(\boldsymbol{\theta}) = \left(\circledast_{z=1}^d \boldsymbol{\nu}_z^T \boldsymbol{D}_z^j\right)\vec{1} \tag{6.24}$$

$$B_j(\boldsymbol{\theta}) = \left(\circledast_{z=1}^d \boldsymbol{\nu}_z^T \boldsymbol{E}_z^j\right)\vec{1}. \tag{6.25}$$

These factor matrices have dimension $K \times R$ and the cosine vectors $\boldsymbol{\nu}_z[k_z] := \cos\left(k_z\pi\frac{\boldsymbol{\theta}_z - a_z}{b_z - a_z}\right) \in \mathbb{R}^K$. Here K is the number of cosine basis functions per dimension, and $[a_z, b_z]$ is the domain where the zth variable exists on. The goal is to use training data to estimate these tensors in order to set up the estimated ch.f.. This will then give a closed formula for the barrier option price, which can be used for calibration.

6.3.2. SETUP OF THE TRAINING DATA

In order to find the factor matrices training data is needed. First we fix the index j and only focus on the A part, since the other part will be analogous. Secondly, let us consider the grid of M combinations of d model parameters $\boldsymbol{\theta}$. This will be the training input.

Then for $i = 1,...,M$ the training output will be $A_j(\boldsymbol{\theta}_i)$. These values are derived using SVD in Section 3.2.2. With this a set of log-asset prices $\mathbf{x}$ is needed to calculate a set of benchmark values per $\boldsymbol{\theta}_i$. This means that first the whole vector $\mathbf{A}(\boldsymbol{\theta}_i) = [A_0(\boldsymbol{\theta}_i),...,A_{N_\varphi-1}(\boldsymbol{\theta}_i)]^T \in \mathbb{R}^{N_\varphi}$ is computed using minimization problem (3.23). Then all these vectors are collectively stored in an $M \times N_\varphi$ matrix.

To get the correct output, we split them apart afterwards such that for index j we get the output vector $A_j(\boldsymbol{\theta}) = [A_j(\theta_1),...,A_j(\theta_M)]^T \in \mathbb{R}^M$. We denote the estimated outputs of $A_j(\boldsymbol{\theta})$ and $A_j(\boldsymbol{\theta})$ by $\mathbf{y}_A^j$ and $\mathbf{y}_B^j$.

6.3.3. DERIVING THE TENSOR USING FSA-HTF

Upon configuring the training data, the task of determining the complex Hermitian factor $(ch.f.)$ is reduced to addressing $2N_\varphi$ distinct problems. In this section, we delve into the derivation of the factor matrices denoted as $\mathbf{D}_i^j\, i = 1^d$ and $\mathbf{E}_i^j\, i = 1^d$ using a technique known as hidden tensor factorization. These factor matrices are intricately linked to the training output, particularly for indices $j = 0,...,N_\varphi - 1$.

The factor matrices exhibit a crucial relationship with the training output, expressed as follows for each j:

$$\boldsymbol{y}_A^j = \left(\circledast_{z=1}^d \boldsymbol{V}z^T \boldsymbol{D}_z^j\right)\vec{1} \tag{6.26}$$

$$\boldsymbol{y}_B^j = \left(\circledast z = 1^d \boldsymbol{V}_z^T \boldsymbol{E}_z^j\right)\vec{1}, \tag{6.27}$$

In these equations, the matrix $\boldsymbol{V}_z \in \mathbb{R}^{K\times M}$, with its elements defined as $\boldsymbol{V}z[k,m] = \cos\left(k_z\pi\frac{\theta_z^m - a_z}{b_z - a_z}\right)$. In this context, θ_z^m represents the value associated with the zth coordinate at the sampling point $\boldsymbol{\theta}_m$. Notably, as elucidated in Section 6.1, the traditional CPD technique, which involves the decomposition of the complete tensor, is infeasible for tensors comprised of Fourier coefficients due to the requirement of computing the entire set of coefficients. Instead, we employ the Fourier Series Approximation via Hidden Tensor Factorization (FSA-HTF) approach to deduce the factor matrices $\{\mathbf{D}_i^j\}_{i=1}^d$ and $\{\mathbf{E}_i^j\}_{i=1}^d$ to assemble the complete tensors. This methodology, introduced in [28], was devised to tackle analogous problems, as it obviates the need to compute the entire tensor.

Consequently, equations (6.26) and (6.27) undergo modification, transforming into a regression model. This shift in perspective implies that, for each j, two distinct minimization problems manifest:

$$\min \frac{1}{M}\sum_{m=1}^{M} \mathscr{L}(y_{A,m}^j - A_j(\boldsymbol{\theta}_m)) \qquad \text{for } j = 0,\ldots,N_\varphi - 1, \tag{6.28}$$

$$\min \frac{1}{M}\sum_{m=1}^{M} \mathscr{L}(y_{B,m}^j - B_j(\boldsymbol{\theta}_m)) \qquad \text{for } j = 0,\ldots,N_\varphi - 1, \tag{6.29}$$

where $\mathscr{L}(\cdot)$ denotes a loss function. For our research the squared error is used, since this makes the calculation of the gradient easy. In this manner the Fourier-cosine coefficients are fitted on the ch.f. data. This results in the minimization problems:

$$\min_{\left\{\boldsymbol{D}_z^j\right\}_{z=1}^d} \frac{1}{M}\sum_{m=1}^{M}\left(y_{A,m}^j - \left(\circledast_{z=1}^d\left(\boldsymbol{V}_z[:,m]^T\boldsymbol{D}_z^j\right)\right)\vec{1}\right)^2 \qquad \text{for } j = 0,\ldots,N_\varphi - 1, \tag{6.30}$$

$$\min_{\left\{\boldsymbol{E}_z^j\right\}_{z=1}^d} \frac{1}{M}\sum_{m=1}^{M}\left(y_{B,m}^j - \left(\circledast_{z=1}^d\left(\boldsymbol{V}_z[:,m]^T\boldsymbol{E}_z^j\right)\right)\vec{1}\right)^2 \qquad \text{for } j = 0,\ldots,N_\varphi - 1. \tag{6.31}$$

Recall the definition of the matrix $\boldsymbol{Q}_z = \left(\circledast_{i\neq n}\left(\boldsymbol{A}_i^T\boldsymbol{V}_i\right)\right)$. Here we define $\boldsymbol{Q}_{z,j}^A := \left(\circledast_{i\neq z}\left((\boldsymbol{D}_i^j)^T\boldsymbol{V}_i\right)\right)$ and $\boldsymbol{Q}_{z,j}^B := \left(\circledast_{i\neq z}\left((\boldsymbol{E}_i^j)^T\boldsymbol{V}_i\right)\right)$. Then per j for minimization problem (6.30) all factor matrices $\{\boldsymbol{D}_i^j\}_{i=1}^d$ are fixed except for $\boldsymbol{D}_z^j$. Analogously for (6.31), regarding the matrices $\{\boldsymbol{E}_i^j\}_{i=1}^d$ and $\boldsymbol{E}_z^j$.

$$\min_{\boldsymbol{D}_z^j} \frac{1}{M} \sum_{m=1}^{M} \left(y_{A,m}^j - \boldsymbol{V}_z[:,m]^T \boldsymbol{D}_z^j \boldsymbol{Q}_{z,j}^A[:,m] \right)^2 \qquad \text{for } j = 0,...,N_\varphi - 1, \tag{6.32}$$

$$\min_{\boldsymbol{E}_z^j} \frac{1}{M} \sum_{m=1}^{M} \left(y_{B,m}^j - \boldsymbol{V}_z[:,m]^T \boldsymbol{E}_z^j \boldsymbol{Q}_{z,j}^A[:,m] \right)^2 \qquad \text{for } j = 0,...,N_\varphi - 1. \tag{6.33}$$

Because of the convexity of the problem a (global) minimum can be achieved by setting the gradient of the loss function for the two separate problem equal to zero. That is:

$$\frac{2}{M} \sum_{m=1}^{M} \left(y_{A,m}^j - \boldsymbol{V}_z[:,m]^T \boldsymbol{D}_z^j \boldsymbol{Q}_{z,j}^A[:,m] \right) \cdot \left[-\boldsymbol{V}_z[:,m] (\boldsymbol{Q}_{z,j}^A)^T[:,m] \right] = 0 \qquad \text{for } j = 0,...,N_\varphi - 1, \tag{6.34}$$

$$\frac{2}{M} \sum_{m=1}^{M} \left(y_{B,m}^j - \boldsymbol{V}_z[:,m]^T \boldsymbol{E}_z^j \boldsymbol{Q}_{z,j}^B[:,m] \right) \cdot \left[-\boldsymbol{V}_z[:,m] (\boldsymbol{Q}_{z,j}^A)^T[:,m] \right] = 0 \qquad \text{for } j = 0,...,N_\varphi - 1, \tag{6.35}$$

which leads to the two linear systems per j that have to be solved to obtain $\{\boldsymbol{D}_z^j\}_{z=1}^d$ and $\{\boldsymbol{E}_z^j\}_{z=1}^d$,

$$\frac{1}{M} \sum_{m=1}^{M} \left(\boldsymbol{V}_z[:,m]^T \boldsymbol{D}_z^j \boldsymbol{Q}_{z,j}^A[:,m] \right) \boldsymbol{V}_z[:,m] (\boldsymbol{Q}_{z,j}^A)^T[:,m] = \frac{1}{M} \sum_{m=1}^{M} y_{A,m}^j \boldsymbol{V}_z[:,m] (\boldsymbol{Q}_{z,j}^A)^T[:,m], \tag{6.36}$$

$$\frac{1}{M} \sum_{m=1}^{M} \left(\boldsymbol{V}_z[:,m]^T \boldsymbol{E}_z^j \boldsymbol{Q}_{z,j}^B[:,m] \right) \boldsymbol{V}_z[:,m] (\boldsymbol{Q}_{z,j}^B)^T[:,m] = \frac{1}{M} \sum_{m=1}^{M} y_{B,m}^j \boldsymbol{V}_z[:,m] (\boldsymbol{Q}_{z,j}^B)^T[:,m]. \tag{6.37}$$

To solve equations (6.37) and (6.36) the Conjugate Gradient (CG) method is used, which is explained thoroughly in previous theses at FFQuant [24] and [28].

After all d factor matrices have been found the tensors $D_{j,\mathbf{k}} = [\![\boldsymbol{D}_1^j, \boldsymbol{D}_2^j, ..., \boldsymbol{D}_d^j]\!]$ and $E_{j,\mathbf{k}} = [\![\boldsymbol{E}_1^j, \boldsymbol{E}_2^j, ..., \boldsymbol{E}_d^j]\!]$ can be reconstructed and are used to describe the ch.f. of the underlying Lévy process in (6.24). Which in itself is used in the valuation of the option using equation (6.23). For storage convenience in the code, the collection of tensors is defined as $\mathscr{D} := \{D_{j,\mathbf{k}}\}_{j=1}^{N_\varphi}$ and $\mathscr{E} := \{E_{j,\mathbf{k}}\}_{j=1}^{N_\varphi}$. Here the 4th order tensors $\mathscr{D}, \mathscr{E} \in \mathbb{R}^{N_\varphi \times d \times K \times R}$.

6.4. APPLICATION OF THE CFC METHOD TO EFFICIENT BARRIER OPTION PRICING

In this section, we apply a different method called the CFC method for rapid barrier option pricing under the CGMY-model, as explained in Section 4.3. The CFC method was developed to address two key aspects that set it apart from our first method, the COS-GPR method.

Firstly, the CFC method employs a global decomposition approach, in contrast to the COS-GPR method, which relies on localized decomposition from the supervised method called GPR. This distinction becomes evident in Section 4.3, where it was observed that the accuracy of the barrier option fit diminished significantly for test values of the input parameter C close to the barrier, compared to estimates within the training set.

Secondly, the CFC method boasts a smaller computational complexity. As demonstrated analytically in Chapter 7, the complexity of the CFC method grows linearly as the number of grid points per dimension m increases. On the other hand, the complexity of the COS-GPR method grows polynomially with a degree higher than two. This increase in computational speed will be made evident as well in this section.

We will explore the practical application of the CFC method in the same environment as the COS-GPR method, focusing on pricing barrier options under the dynamics of the CGMY-model. We will provide a detailed explanation of how to apply the CFC method to this example.

Subsequently, we will assess the accuracy and computational efficiency of the CFC method in pricing discretely monitored barrier options with $M_{mon} = 250$. To establish a benchmark for comparison, we will also include the COS Barrier method with $M_{mon} = 25$ as the existing method of estimating this specific barrier option.

To apply the CFC method, we first combine the COS method with the Fourier-cosine series expansion, resulting in a known formula represented by equation (3.6):

$$\boldsymbol{v}_{Bar}(\boldsymbol{x};t,\boldsymbol{\theta}) = e^{-r\Delta t}\sum_{j=0}^{N_\varphi-1}(A_j(\boldsymbol{\theta})C_j(\boldsymbol{x}) - B_j(\boldsymbol{\theta})S_j(\boldsymbol{x})).$$

Similar to the COS-GPR method, we seek to acquire mappings for $j = 0,\ldots,N_\varphi - 1$ in the form of:

$$(C,G,M,Y) \mapsto A_j(C,G,M,Y) \tag{6.38}$$
$$(C,G,M,Y) \mapsto B_j(C,G,M,Y) \tag{6.39}$$

However, the CFC method employs a global decomposition on these mappings, as they are described as a d-dimensional Fourier series expansion. In our case, $d = 4$, as we have four input parameters (C, G, M, and Y). The $2N_\varphi$ different mappings are represented by the expansions:

$$A_j(C,G,M,Y) = \sum_{k_C=0}^{\prime K-1}\sum_{k_G=0}^{\prime K-1}\sum_{k_M=0}^{\prime K-1}\sum_{k_Y=0}^{\prime K-1}\left[D_{j,k_C,k_G,k_M,k_Y}\cos\left(k_C\pi\frac{C-a_C}{b_C-a_C}\right)\cdot\right. \tag{6.40}$$
$$\left.\cos\left(k_G\pi\frac{G-a_G}{b_G-a_G}\right)\cos\left(k_M\pi\frac{M-a_M}{b_M-a_M}\right)\cos\left(k_Y\pi\frac{Y-a_Y}{b_Y-a_Y}\right)\right] \tag{6.41}$$

$$B_j(C,G,M,Y) = \sum_{k_C=0}^{\prime K-1}\sum_{k_G=0}^{\prime K-1}\sum_{k_M=0}^{\prime K-1}\sum_{k_Y=0}^{\prime K-1}\left[E_{j,k_C,k_G,k_M,k_Y}\cos\left(k_C\pi\frac{C-a_C}{b_C-a_C}\right)\cdot\right. \tag{6.42}$$
$$\left.\cos\left(k_G\pi\frac{G-a_G}{b_G-a_G}\right)\cos\left(k_M\pi\frac{M-a_M}{b_M-a_M}\right)\cos\left(k_Y\pi\frac{Y-a_Y}{b_Y-a_Y}\right)\right]. \tag{6.43}$$

To find these mappings, we need to estimate the $K \times K \times K \times K$ dimensional tensors $D_{j,\mathbf{k}}$ and $E_{j,\mathbf{k}}$ for each j. This estimation requires training output data, denoted as $\mathbf{Y}_A$ and $\mathbf{Y}_B$, which are the same as used in the COS-GPR method example (Section 4.3). These matrices have dimensions $M \times N_\varphi$, where $M = m^4$ is the number of training points. The output used for index j is represented by the vector $\mathbf{Y}_A[:, j]$ for the real part and $\mathbf{Y}_B[:, j]$ for the imaginary part. These are also denoted by $\mathbf{y}_A^j$ and $\mathbf{y}_B^j$. More specifically, the data point $y_{A,m}^j$ corresponds to $\mathbf{Y}_A[m, j]$ and $y_{B,m}^j$ corresponds to $\mathbf{Y}_B[m, j]$.

Next, the CPD (Canonical Polyadic Decomposition) part uses this data to find a dimension-reduced Fourier-series expansion by finding factor matrices. For each index j, the two tensors $D_{j,\mathbf{k}}$ and $E_{j,\mathbf{k}}$ are estimated by factor matrices $\{\mathbf{D}_i^j\}_{i=1}^4$ and $\{\mathbf{E}_i^j\}_{i=1}^4$. The optimization process is described as follows:

$$\min_{\mathbf{D}_1^j,\mathbf{D}_2^j,\mathbf{D}_3^j,\mathbf{D}_4^j} \frac{1}{M} \sum_{m=1}^{M} \left(y_{A,m}^j - \left(\circledast_{z=1}^4 \left(\boldsymbol{V}_z[:, m]^T \boldsymbol{D}_z^j \right) \right) \vec{1} \right)^2 \quad \text{for } j = 0, \ldots, N_\varphi - 1, \tag{6.44}$$

$$\min_{\mathbf{E}_1^j,\mathbf{E}_2^j,\mathbf{E}_3^j,\mathbf{E}_4^j} \frac{1}{M} \sum_{m=1}^{M} \left(y_{B,m}^j - \left(\circledast_{z=1}^4 \left(\boldsymbol{V}_z[:, m]^T \boldsymbol{E}_z^j \right) \right) \vec{1} \right)^2 \quad \text{for } j = 0, \ldots, N_\varphi - 1. \tag{6.45}$$

The optimization procedure, detailed in Section 6.3.3, leverages the CPD method in conjunction with the Alternating Least Squares (ALS) algorithm to solve a system of conditionally linear equations. This iterative process cyclically handles all four factor matrices, wherein three matrices are fixed, and the fourth one is solved after random initialization. The procedure continues until a predefined stopping criterion is satisfied.

For the sake of illustration, let's consider an example with specific contract parameters: $T = 1$, $H = 10$, $K = 6$, and a risk-free interest rate of $r = 0.01$. We set the diffusion of the CGMY-model as $\sigma_B = 0.5$. The COS formula is applied with $N = 100$ and an integration range of $[-3, 3]$. Within the COS formula, the characteristic function is represented by $N_\varphi = 30$ expansion terms. These terms are estimated using Singular Value Decomposition (SVD) with $n = 100$ benchmark asset points over a model parameter grid with m points per dimension, resulting in a total of M training points.

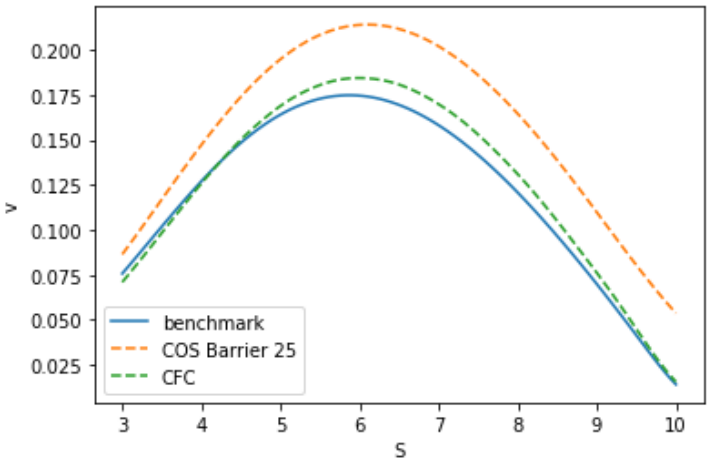
The resulting estimation provides barrier option prices over an asset grid with a test value of $X_{test} = (0.05, 0.4, 0.2, 0.4)$, as depicted in Figure 6.1a.

The associated absolute and relative errors for both methods are presented in Figure 6.1b. Notably, the CFC method exhibits superior accuracy across all points on the curve, outperforming the COS Barrier 25 method in this specific instance.

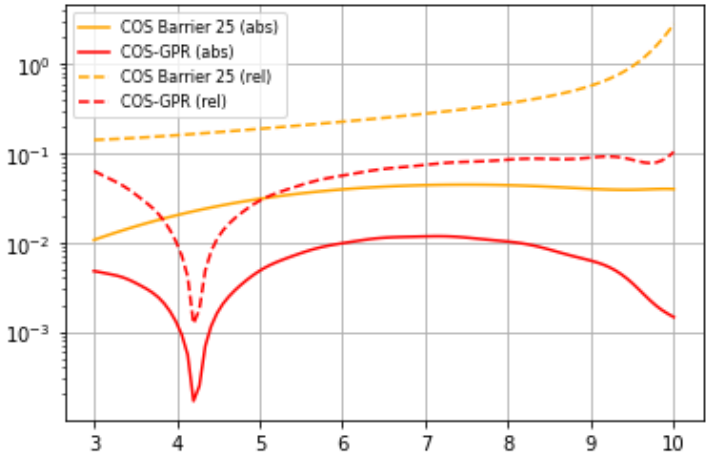
Additionally, Table 6.1 displays a comparison of computational speeds. Evidently, the CFC method significantly outpaces the existing COS Barrier 25 method in terms of efficiency, as illustrated by the following CPU time results:

In Chapter 7, we delve into a comprehensive analysis similar to that of our initial approach, the COS-GPR method. This involves a thorough exploration of the accuracy prediction behavior as model parameters evolve. The CFC method offers greater flexibility, resulting in a more comprehensive and detailed analysis.

Furthermore, our examination extends beyond the COS Barrier method comparison. We also evaluate the performance of the CFC method against our primary approach, the



(a) Comparison of barrier option price estimation with 250 monitoring dates using the CFC method and the COS Barrier method with 25 monitoring dates.



(b) Absolute and relative error plots corresponding to the prediction of the two methods.

Figure 6.1: Comparison of the performance of two methods for barrier option pricing under the CGMY-model.

COS-GPR method. This entails scrutinizing both speed and accuracy across a range of test values, providing a comprehensive understanding of the CFC method's efficacy.

Method	CPU Time (seconds)
CFC	4.01×10^{-5}
COS Barrier 25	7.20×10^{-3}

Table 6.1: Comparing average CPU times for option price calculations on the provided test set: COS-GPR method ($m = 10$) versus COS Barrier 25 method

7

CONVERGENCE TESTS FOR METHOD 2: CFC METHOD

This chapter delves into numerical error analysis for pricing barrier options under Lévy processes, employing our second method: the CFC method, as previously discussed in Chapter 6. In contrast to our first method (the COS-GPR method), the CFC method offers the advantage of using a global decomposition instead of a local one.

The error analysis of the CFC method is conducted in a more comprehensive manner compared to the COS-GPR method. Firstly, we analyze its asymptotic computational complexity. Next, we compare the CPU time required for the CFC Barrier method against the COS Barrier method for a specific set of monitoring dates.

Furthermore, we perform sensitivity analysis on the number of training points in each direction, which directly impacts the number of expansion terms in the CPD. By analyzing these results, we gain insights into the effect of the truncation range on the method's performance. Whereafter we can analyze the behaviour of the number of terms in the COS expansion.

Moreover, we explore the variation in computation time and accuracy by adjusting the rank of the CPD method. This observation helps us understand the behavior of the method as we modify this parameter. Overall, the comprehensive analysis of the CFC method and its comparison to the COS-GPR method will provide valuable insights into its pricing accuracy and computational efficiency.

7.1. MANAGING OVERFITTING: BALANCING SAMPLING POINTS AND EXPANSION TERMS IN FUNCTION APPROXIMATION

The CPD algorithm described in Section 6.2 is a supervised learning technique applied on approximating the Fourier coefficient tensor using a reduced dimension. In the context of supervised learning, the model is built by minimizing the misfit between the training data and the model output, which results in a minimization problem. For a given index j, the objective is to minimize the sum of squares of the total misfits between the

training data values $y^j_{A,m}$ and $y^j_{B,m}$ assigned to the sample $\boldsymbol{\theta}_m$. The values of the Fourier-cosine series expansion evaluated at the same sample $\boldsymbol{\theta}_m$. This minimization problems are expressed as:

$$\min_{D_{j,\mathbf{k}}} \frac{1}{M} \sum_{m=1}^{M} \left(y^j_{A,m} - \left[\sum_{k_1=0}^{K-1} \cdots \sum_{k_N=0}^{K-1} D_{j,\mathbf{k}} \prod_{i=1}^{d} \cos\left(k_i \pi \frac{\theta_i^m - a}{b-a} \right) \right] \right)^2, \tag{7.1}$$

$$\min_{E_{j,\mathbf{k}}} \frac{1}{M} \sum_{m=1}^{M} \left(y^j_{B,m} - \left[\sum_{k_1=0}^{K-1} \cdots \sum_{k_N=0}^{K-1} E_{j,\mathbf{k}} \prod_{i=1}^{d} \cos\left(k_i \pi \frac{\theta_i^m - a}{b-a} \right) \right] \right)^2 \tag{7.2}$$

Here, $\mathbf{k} = (k_1, ..., k_d)$ is a multi-index, and θ_i^m denotes the value of the ith coordinate in the sampling point $\boldsymbol{\theta}_m$.
The minimization problems in (7.1) and (7.2) involve, per j index, K^d Fourier coefficients for which the minimization problem has to be solved. Since the two minimization problems are solved individually from eachother and also per j, the number K^d can be viewed as the degree of complexity the model has in minimizing the error with the training data. A higher value of K^d allows for more complex representations of the data, while a lower value leads to a simpler model. With this case, the Fourier-cosine expansion terms can be fitted to all training points whole
The number of training data points is denoted with M. In our examples following in the paper, we will assume that the number of training points is equal for each dimension. Hence, if we have m training points per dimension, the total training set has $M = m^d$ training points. Within the minimization problem, the number of inputs M is describes the number of values the expansion needs to be matched to. Equivalently, this represents how constrained the model is.

When initializing the training of the model it is crucial to give the right relation between the number of input data $M = m^d$ and the number of expansion terms per dimension K. There are three cases between the relation of K and m. There is $K < m$,$K > m$ and $K = m$. From a prior thesis at FF Quant [28] the optimal choice will be to choose $K = m$, for which the degree of flexibility within the model is equal to the level of constraints imposed during its construction. Striking the right balance between these two factors becomes crucial in achieving an accurate fit to all the training data points while ensuring a reliable approximation of the overall function.

7.2. OPTIMAL SAMPLING FREQUENCY FOR TRAINING DATA IN FUNCTION APPROXIMATION

In the context of estimating barrier option prices using the Fourier-cosine series with the CPD algorithm, the selection of the number of grid points per dimension, denoted by m, is a crucial determinant of both accuracy and computational efficiency. Similar to the COS-GPR method, the CFC method employs a grid-based approach, where a denser grid (higher m) captures finer details in the survival characteristic function.

The main objective of this section is to identify the optimal value of m for the CFC method. To achieve this, we follow a similar approach as we did with the COS-GPR

method in Section 5.1. We generate a range of m values spanning from 4 to 30, leveraging the device's ability to handle a higher sampling rate for the CFC method compared to COS-GPR, as GPR consumes more memory.

The choice of sampling rate m in the CFC method has a direct impact on the number of Fourier cosine terms K, with the relationship $K = m$. Increasing the value of m leads to a more intricate Fourier cosine expansion of the characteristic function. This intricacy facilitates a finer fit, enabling the capture of nuanced features and non-linear patterns. However, it's important to note that a complex fit of the characteristic function doesn't necessarily guarantee more accurate estimations of barrier option prices, a point we will delve into in the subsequent investigation.

It's worth mentioning that we won't be determining an optimal value for m in the context of the CFC method. The reason behind this decision is our intention to maintain consistency in comparing the CFC method with the COS-GPR method. Both methods will employ the same granularity of training grid to ensure a fair and meaningful comparison. Although we will proceed with the rest of the sensitivity analysis to identify other optimal parameters for the CFC method. This endeavor aims to uncover intriguing behavioral insights arising from these parameter variations.

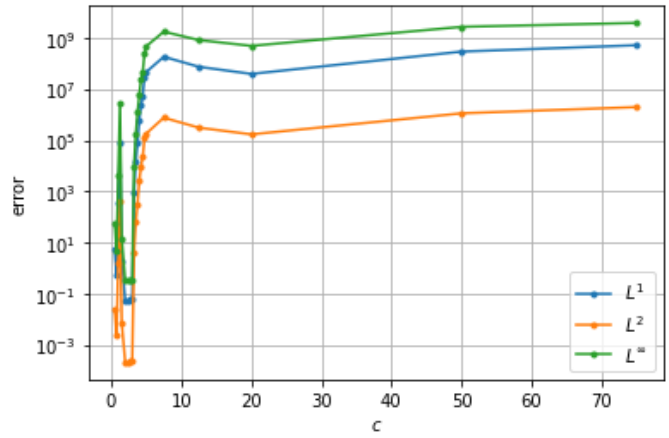
7.3. OPTIMAL TRUNCATION RANGE

The approach for the CFC method mirrors that of Section 5.2 for the COS-GPR method. We will observe analogous error behavior. The truncation range is symmetrically assumed to be $[-c, c]$, with $c > 0$. We will examine error behavior within a value range spanning from 0.25 to 75. As depicted in Figure 7.1a, we focus on the interval 1.25 to 3.5 for a more detailed error analysis, following the zoomed-in view in Figure 7.1b. This analysis reveals an optimal truncation range value of $c = 2.70$ for the CFC method, closely aligned with the COS-GPR method's optimal value of $c = 2.45$. Thus, utilizing the provided test set for the CFC method, we determine the most effective truncation range to be $[-2.70, 2.70]$.

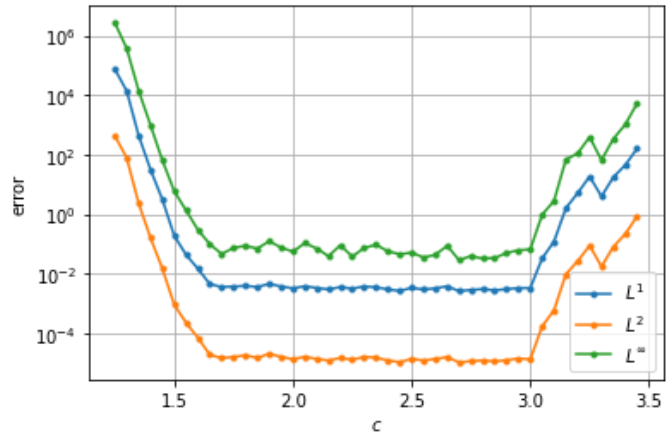
7.4. OPTIMAL NUMBER OF COS EXPANSION TERMS

Having established the optimal truncation range as $[-2.70, 2.70]$, the next step involves determining the appropriate number of expansion terms for the COS method. This parameter, denoted as N in the context of the CFC method, signifies the coordinates within the approximated characteristic function that influence the barrier option's valuation. Notably, in the conventional COS method for pricing European options, a higher value of N generally leads to improved accuracy. However, given our focus on an estimated survival characteristic function and a distinct option type, we anticipate a dissimilar behavior.

Our analysis involves assessing accuracy across a test set, varying N within the range of 60 to 200. Initial examination, as depicted in Figure 7.2a, reveals a broader region of interest lying between 100 and 140. A closer inspection, as shown in Figure 7.2b, pinpoints the optimal value within this precise range to be $N = 98$. This value is selected as the foundation for the CFC method.



(a) Zoomed out.



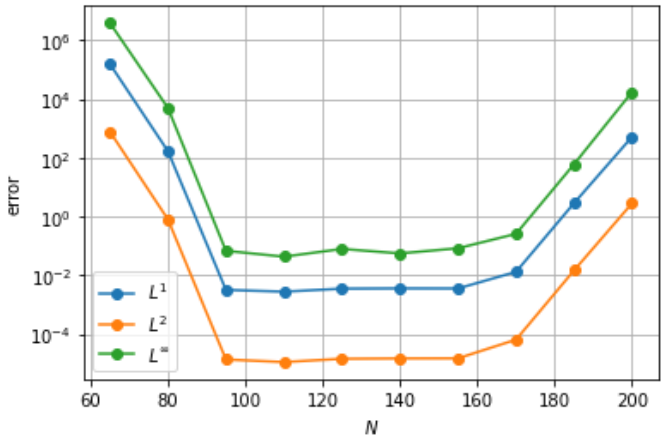
(b) Zoomed in.

Figure 7.1: Comparison of three error metrics for the COS-GPR method with optimal $m = 10$ applied to a test set of barrier options values with a watchtime of $M_{\text{mon}} = 250$, while varying the truncation range $[-c, c]$.

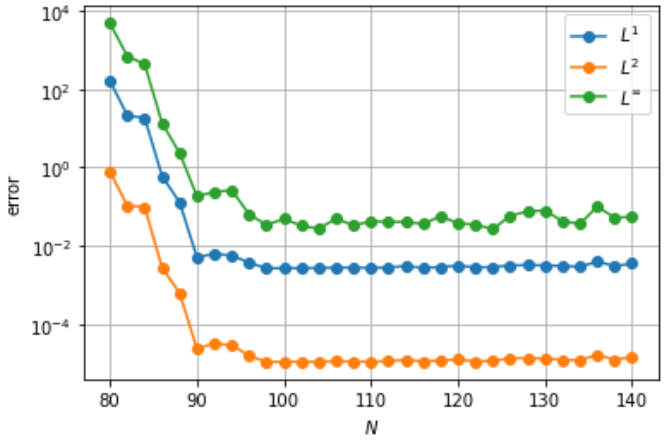
7.5. OPTIMAL RANK

The final segment of the sensitivity analysis for the CFC method delves into the role of the CPD rank. This rank signifies the count of terms employed to characterize the tensor, which corresponds to the value computation of the barrier option in (6.23). While a higher rank facilitates the discovery of more intricate tensor patterns, its increment doesn't necessarily equate to a more accurate option valuation. Consequently, this section conducts an in-depth analysis of accuracy across a range of rank values, spanning from 4 to 60, to shed light on this nuanced relationship.

Illustrated in Figure 7.3, the L^2-error is depicted concerning the rank R for the CGMY-model, utilizing the priorly optimized parameters m, c, and N. Through this visual rep-



(a) Zoomed out.



(b) Zoomed in.

Figure 7.2: Comparison of three error metrics for the CFC method with optimal $m = 10$ and truncation range $[-c, c]$, with $c = 2.7$, applied to a test set of barrier options values with a watchtime of $M_{\text{mon}} = 250$, while varying the number of COS terms N

resentation, we identify the optimal rank value for this specific model to be $R = 15$.

7.6. PERFORMANCE COMPARISON: CFC VS. COS-GPR VS. COS BARRIER METHOD

After optimizing the procedure in the preceding sections, we proceed to evaluate the performance of the CFC method in estimating barrier option values with 250 monitoring dates. Once again, the benchmark is established using the COS Barrier method with 250 monitoring dates. We compare the estimation performance of the CFC method against

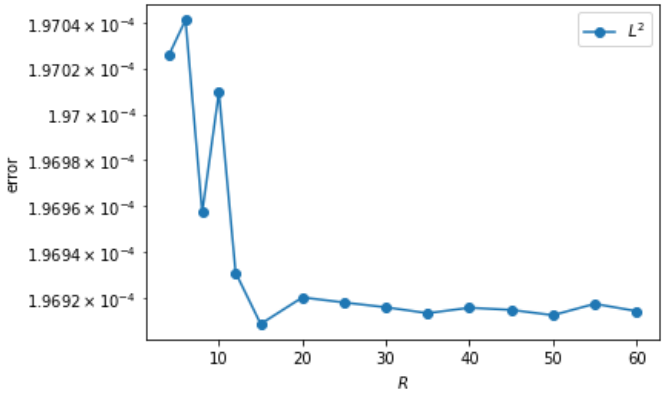


Figure 7.3: Variation of accuracy given by the L^2-error with different ranks

the estimation outcomes of the COS Barrier method with 25 monitoring dates, as well as our initial method, the COS-GPR method. Both the CFC and COS-GPR methods employ their respective optimized settings, which have been derived in this paper. For COS-GPR, these settings can be found at the beginning of Section 5.4. For the CFC method, the optimized settings are as follows:

- Sampling frequency: In accordance with Section 7.1 and 7.2, we set the sampling frequency to $m = 10$, resulting in $M = m^d = 10000$ training points.
- Truncation range: As outlined in Section 7.3, the symmetric truncation range around 0 is set to $[a, b] = [-c, c]$, where $c = 2.70$.
- Number of ch.f. terms: Following Section 7.4, we determine the number of characteristic function terms in the COS method as $N = 98$.
- Rank of the CPD: As per Section 7.5, we fix the rank of the Canonical Polyadic Decomposition (CPD) at $R = 15$.

To assess the performance of these three methods, we first evaluate their effectiveness on two individual test values in order to analyze their fit. One test value lies within the training region, while the other lies outside. These test values are defined as:

$$\boldsymbol{\theta}_{in} = (C_{in}, G_{in}, M_{in}, Y_{in}) = (0.14, 0.27, 0.15, 0.22),$$
$$\boldsymbol{\theta}_{out} = (C_{out}, G_{out}, M_{out}, Y_{out}) = (0.52, 0.27, 0.15, 0.22),$$

where $\boldsymbol{\theta}_{in}$ represents a single test example within the training set, and $\boldsymbol{\theta}_{out}$ represents a test example outside the training set. The fits for both samples are graphed in Figure 7.4, illustrating the performance of the CFC method, the COS-GPR method, and the COS Barrier method with 25 monitoring dates.

7.6.1. ACCURACY COMPARISON

In this section, we seek to determine the most accurate method by assessing their performance over a grid of test points denoted as $(0.002 : 0.55 : 10)^4$. We have three methods to compare: the COS Barrier method, the COS-GPR method, and our proposed CFC method. The plots and tables generated here mirror those in Section 5.4.1. The error metrics employed are also defined in the aforementioned section.

To commence, we calculate the total error and compile it into a single vector. This vector contains the errors computed for each test input against the benchmark curve. Table 7.1 presents four error metrics: Relative L^1, L^1-error, Relative L^∞, and L^∞-error. These metrics are defined in (5.1) upto (5.4).

Method	*Relative L^1 (%)*	*L^1-error*	*Relative L^∞ (%)*	*L^∞-error*	*CPU (ms)*
COS Barrier 25	31.184	$1.681 \cdot 10^{-2}$	25.768	$6.535 \cdot 10^{-2}$	7.204
COS-GPR	1.846	$9.953 \cdot 10^{-4}$	19.889	$5.044 \cdot 10^{-2}$	1.101
CFC	5.304	$2.860 \cdot 10^{-3}$	16.369	$4.151 \cdot 10^{-2}$	0.040

Table 7.1: Comparison of the CFC method, COS-GPR method, and COS Barrier method under the CGMY-model using a set of test values. Errors represent combined results over all benchmark values for the entire set of test inputs.

We focus primarily on the CFC method from this chapter, alongside our initial method (COS-GPR) and the existing COS Barrier method (25 monitoring dates). Their corresponding errors are presented in Table 7.1. Notably, the COS-GPR method outperforms the CFC method in terms of accuracy, specifically when considering the L^1 norm and relative L^1-error. This is probably because our understanding of how to set the truncation ranges in the CFC method is still based on trial and error, and these ranges are fixed in the code. This inflexibility significantly impacts the effectiveness of the CFC method, leading to the observed performance difference between the two methods. Conversely, the CFC method exhibits lower L^∞ and relative L^∞-error values. This indicates that the CFC method's errors are less extreme, although the COS-GPR method consistently outperforms it in terms of average accuracy across all observations. Additionally, the CFC method proves significantly faster than both alternatives, boasting a speed increase of 25 times over our previous method and around 180 times faster than the COS Barrier method with 25 monitoring dates.

For a more comprehensive understanding of error behavior across the value curve, we analyze individual values per test input rather than aggregating across test inputs. This is achieved through plots, depicting the number of observations below a certain error magnitude on the vertical axis against the error magnitude on the horizontal axis. Figure 7.5 showcases these curves for the three discussed methods.

1. **Relative L^1-error over all observations (Figure 7.5a):**

   This metric gauges the relative absolute error across each observation (5.5). The graph reveals that both the CFC method and COS-GPR method outperform the COS Barrier 25 method, with a higher number of errors below a given threshold. Notably, the COS-GPR method surpasses the CFC method in accuracy, as indicated by the earlier growth of its curve. This observation aligns with the data in

Table 7.1.

2. **Relative L^1-error over the whole curve for a given input (Figure 7.5b):**

   In contrast, this assessment examines the entire value curve for a specific input (5.6). While both the COS-GPR and CFC methods maintain superiority over the COS Barrier 25 method, the latter exhibits smaller maximum errors. This suggests that the COS Barrier 25 method may perform better for certain test inputs. Notably, the CFC method yields larger values for this specific error metric.

3. **Relative L^∞-error (Figure 7.5c):**

   This metric quantifies the maximum absolute percentage deviation across the value predictions and benchmark values for each input (5.7). While the COS-GPR method consistently outperforms the other two methods, the COS Barrier 25 method again displays smaller maximum errors. The COS-GPR method records larger values for this error metric.

In conclusion, our analysis reveals that the CFC method exhibits enhanced accuracy compared to the COS Barrier 25 method. However, the COS-GPR method emerges as the ultimate champion, showcasing superior accuracy across all three methods, as evidenced by its steeper growth in the curves depicted in the aforementioned plots. While the CFC method demonstrates a reduction in extreme errors, the COS-GPR method maintains a consistently higher level of accuracy on average. Furthermore, the CFC method excels notably in terms of computational efficiency, boasting a remarkable 25-fold improvement over our previous method and a staggering approximate 180-fold acceleration compared to the COS Barrier method with 25 monitoring dates. It's important to acknowledge that both the COS-GPR method and the CFC method hold untapped potential for improvement in various aspects. Consequently, as further research advances and refines these models, the results could potentially diverge, presenting new insights and outcomes for both methodologies.

7.6.2. CPU COMPARISON

To gain insights into the computational complexity of the CFC method, we will initially deduce the analytical complexity of the model based on the description provided in Chapter 6. Subsequently, we will empirically examine the observed complexity by systematically varying the parameter m, thereby tracking the resulting increase in computational time for the CGMY model scenario.

ANALYTICAL TIME COMPLEXITY CFC METHOD

Looking at the value estimation formula used in the CFC method, it boils down to first obtaining the $2N_\varphi$ mappings $A_j(\boldsymbol{\theta})$ and $B_j(\boldsymbol{\theta})$, which are calculated individually for each j using equations (6.24). Afterward, the next step is to set up the characteristic function (ch.f.) and use it to calculate the value using the COS method representation. The analytical complexity of calculating a barrier option value using the CFC method will come down to $\mathcal{O}(dN_\varphi KR)$.

Firstly, for a given j, the complexity of the expression

$$\left(\circledast_{z=1}^{d} \boldsymbol{v}_z^T \boldsymbol{D}_z^j\right)\vec{1}$$

depends on the dimensions of the involved matrices and vectors.

Let's break down the complexity step by step:

- Hadamard product: The expression inside the parenthesis involves an element-wise product of d matrices $\boldsymbol{\nu}_z^T \boldsymbol{D}_z^j$. If each matrix has dimensions $K \times R$, then the element-wise product will have a complexity of $\mathcal{O}(dKR)$.
- Summation: After performing the element-wise product, the result will be a matrix of dimensions $K \times R$. The next operation is to sum up the elements of this matrix. The complexity of the summation is $\mathcal{O}(KR)$.
- Matrix-vector multiplication: The resulting matrix from the summation will be multiplied by the vector $\vec{1}$ of dimensions $R \times 1$. The complexity of matrix-vector multiplication for a matrix of dimensions $K \times R$ and a vector of dimensions $R \times 1$ is $\mathcal{O}(KR)$.

Overall, the complexity of the entire expression $\left(\circledast_{z=1}^{d} \boldsymbol{\nu}_z^T \boldsymbol{D}_z^j\right)\vec{1}$ is dominated by the element-wise product, which is $\mathcal{O}(dKR)$. The other operations (summation and matrix-vector multiplication) have lower complexity compared to the element-wise product and can be considered $\mathcal{O}(KR)$. Furthermore, the derivation of the complexity is analogous for calculating mapping $B_j(\boldsymbol{\theta})$ using $\left(\circledast_{z=1}^{d} \boldsymbol{\nu}_z^T \boldsymbol{E}_z^j\right)\vec{1}$.

Finally, the barrier option value is expressed using formula (3.6), which is essentially a sum for a single log-asset price x, resulting in a complexity of $\mathcal{O}(2N_\varphi)$, which is dominated by the $\mathcal{O}(dN_\varphi K\ R)$. This means that the computational complexity of the CFC method is $\mathcal{O}(dN_\varphi KR)$ or similarly, since $K = m$, the computational complexity can be written as $\mathcal{O}(dN_\varphi mR)$.

7

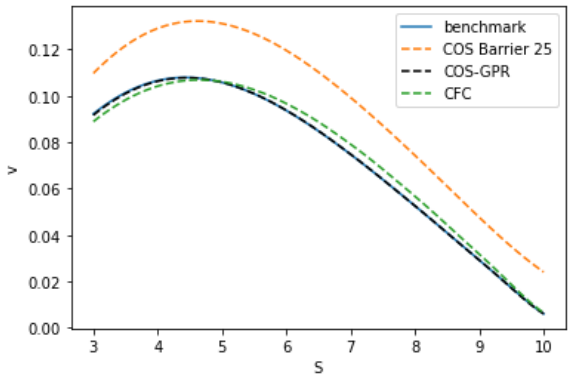
COMPUTATIONAL COMPLEXITY CFC METHOD

Similar to Section 5.4.2, we extend our analysis to examine the CPU time behavior of the CFC method by increasing the sampling frequency and studying the resulting CPU times. Unlike the COS-GPR method, which encounters memory constraints beyond $m = 14$ for the CGMY-model, the CFC method does not face this issue. This distinction provides us with a more comprehensive understanding of the underlying relationship.

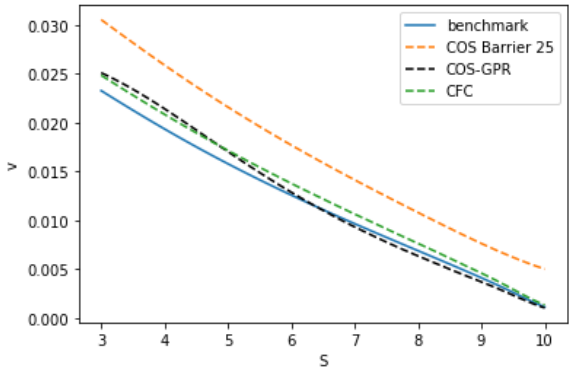
Figure 7.6 presents a clear departure from the GPR-COS approach. Instead of a logarithmic linear association, the CFC method demonstrates a distinct linear trend, indicating a notable difference. This linear behavior aligns analytically with our earlier derivation of computational time for the CGMY-model, presenting a complexity of $\mathcal{O}(4N_\varphi mR)$, thereby validating our computational complexity analysis. The specific numerical values defining the curve in Figure 7.6 are detailed in Table 7.2.

m	CPU (seconds)
6	3.982×10^{-5}
8	3.985×10^{-5}
12	4.001×10^{-5}
18	4.021×10^{-5}
24	4.048×10^{-5}
36	4.069×10^{-5}

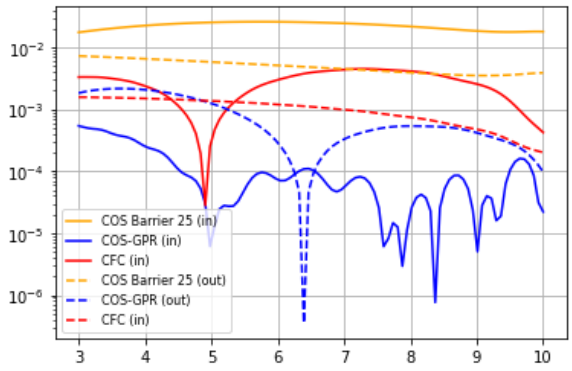
Table 7.2: CPU Times for Different Values of m



(a) Sample from the training set: COS-GPR fitting barrier options prices.

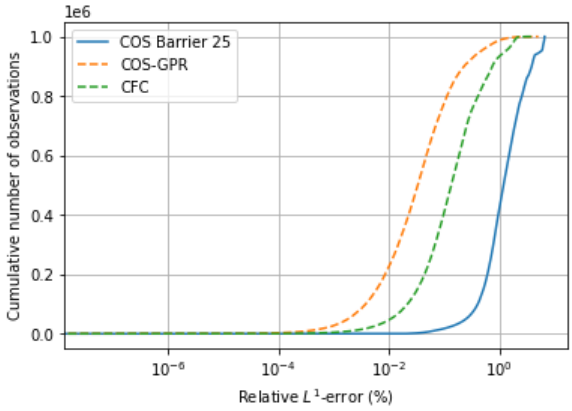


(b) Sample from outside the training set: COS-GPR fitting barrier options prices.

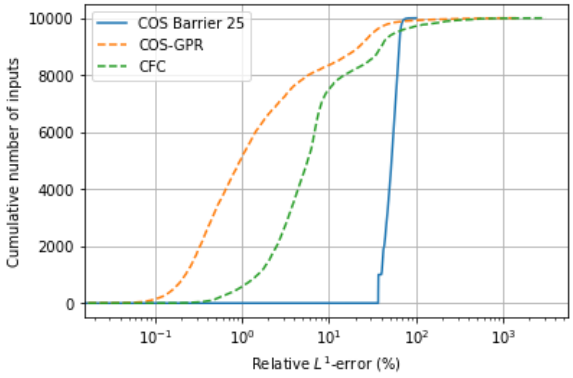


(c) Comparing absolute errors between benchmark option values and the estimated value for the CFC/COS-GPR and COS Barrier 25 method.

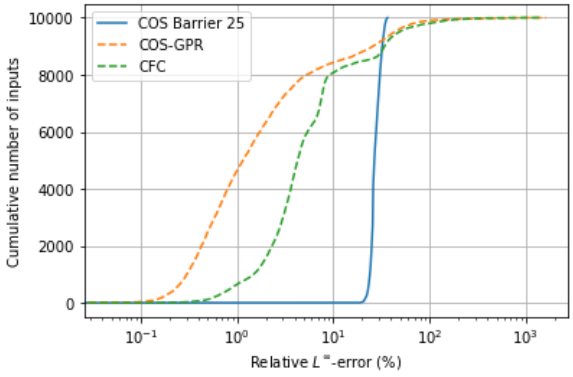
Figure 7.4: Comparison of the CFC method with COS-GPR method and the COS Barrier method for fitting barrier options prices for two samples: one from the training set (a) and one from outside the training set (b). Additionally, the absolute errors of the two examples are shown in (c).



(a) Relative L^1 for all observations.



(b) Relative L^1 for inputs.



(c) Relative L^∞ for inputs.

Figure 7.5: Empirical cumulative distribution functions (ECDFs) for error metrics.

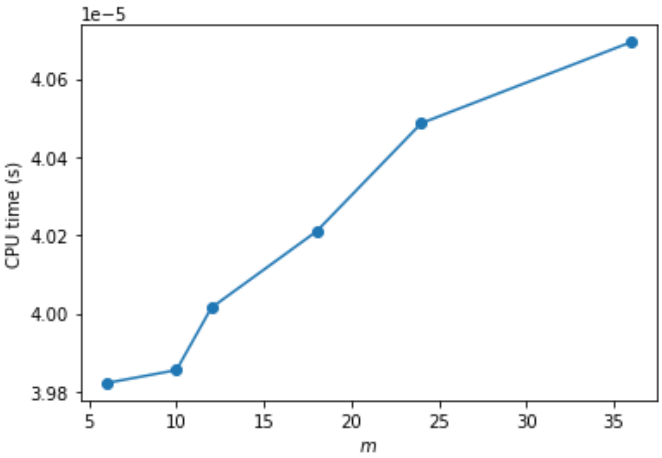


Figure 7.6: Variation in CPU times of the CFC method across different values of parameter m.

8

CONCLUSION

In this research thesis, our primary objective was to develop a highly efficient barrier option pricer. While the COS Barrier method is notably slower than pricing European options using existing methods, we leveraged the natural separation of variables as in the COS Method assumption and the Lévy process to effectively reduce the problem's dimensionality by two. More precisely, the strike and initial log-asset price dimensions are separated from the model parameters in the option pricing formula.

Our theoretical and numerical analyses revealed a promising approach: we demonstrated that we could employ the same payoff coefficients for barrier options as those used for European options within the 1D COS Method framework for pricing European options. However, a key distinction for barrier options is that the ch.f. corresponds to a survival density associated with the barrier level. Unfortunately, this survival density function remains unknown, prompting us to introduce two innovative methods in this thesis to estimate this survival ch.f.

The first method, referred to as COS-GPR, employs GPR to estimate new values of the survival ch.f. based on new test inputs, which correspond to model parameters of the underlying Lévy process. GPR, a supervised learning technique, offers transparency compared to more opaque methods like NN, making it particularly appealing in the financial industry. Our evaluation, involving the estimation of barrier option values with 250 monitoring dates, indicated that the COS-GPR method delivers a sevenfold increase in speed compared to the COS Barrier method with 25 monitoring dates. Moreover, the accuracy of the COS-GPR method consistently surpasses that of the existing method, albeit with some instances of increased error due to GPR's limitations near training boundaries.

To address these challenges, we devised a second model named the CFC method. Unlike the localized approach of COS-GPR, the CFC method employs a global functional decomposition, employing a multidimensional Fourier-series expansion on the ch.f. and utilizing CPD to mitigate exponential complexity growth in the model dimension d. Evaluating the performance of the CFC method against both the COS Barrier method and the COS-GPR method using the CGMY model revealed that the CFC method

generally outperforms the COS Barrier method in terms of accuracy. However, it also entails occasional extreme errors in option values. Comparatively, while the COS-GPR method frequently surpasses the accuracy of the CFC method, it too experiences instances of more pronounced errors.

The hallmark feature of the CFC method is its remarkable speed, boasting a 180-fold acceleration compared to the COS Barrier method and a 25-fold improvement over the COS-GPR method. It is crucial to acknowledge that both of our methods entail numerous parameters that can be optimized, indicating that the full potential of these models is yet to be fully realized. Consequently, the present state represents a snapshot, with ample room for further enhancement and refinement.

8.1. FURTHER RESEARCH

Our research uncovers numerous promising directions for further exploration. One critical aspect involves establishing a robust pricing rule for barrier options using the traditional COS method. Our investigation underscores the significant impact of selecting an appropriate truncation range, as ill-informed choices have resulted in notably inaccurate estimations. Additionally, we aim to support our theoretical findings with rigorous analytical proofs.

To further delve into this evolving investigation, our intention is to conduct a meticulous scrutiny of the COS-GPR method. This approach mandates an exhaustive analysis, particularly in light of the memory-related challenges that have surfaced. Notably, in the context of a 4-dimensional scenario, our progress has been curtailed due to the limitation of only $m = 14$ training points per dimension. Exploring the capabilities of more powerful computing resources, such as the DelftBlue[1], could potentially offer us deeper insights into the method's performance.

Furthermore, our exploration of the GPR method for estimating characteristic function coefficients prompts a deeper investigation into the cross-correlation among outputs. This endeavor holds the potential to utilize the initial representation more effectively, addressing the oversight of neglecting the shared functional origin of coefficients.

Shifting our focus to the CFC method, there remains ample room for optimization within the CPD framework. Realizing its full potential presents an intriguing challenge. The efficiency of this method, a standout feature in our paper, underscores its significance in the quantitative finance domain, where speed is of paramount importance.

Taking a different approach, an alternative path involves evaluating the performance of value estimates when the characteristic function is estimated using neural networks. A particularly intriguing aspect lies in deciphering any noticeable patterns or behaviors that emerge as we manipulate the parameters of these models. The diverse array of supervised learning methods within the machine learning realm introduces complexity and endless possibilities to our research.

While our current focus revolves around Lévy processes, expanding to non-Lévy processes like Heston or other rough volatility models presents an enticing extension. This strategic expansion promises a more comprehensive grasp of intricate financial phenomena, enriching the depth and practical applicability of our findings.

[1] DelftBlue is a high-performance computer situated at the Delft University of Technology.

A

APPENDIX

A.1. DERIVATION OF REPRESENTATION 2

Below is the comprehensive derivation of the barrier option value employing the COS method with representation 2, which involves a Fourier-cosine expansion applied to the characteristic function. The derivation encompasses various techniques, including leveraging Euler's identity to eliminate the real part operator and employing summation interchange.

$$\boldsymbol{v}_{BAR}(\boldsymbol{x};t,\boldsymbol{\theta}) \approx e^{-r_{\boldsymbol{\theta}}\Delta t}\sum_{k=0}^{\prime N-1}\mathrm{Re}\left\{\bar{\varphi}_{levy}\left(\frac{k\pi}{b-a}\right)\exp\left(ik\pi\frac{\boldsymbol{x}-a}{b-a}\right)\right\}V_k \tag{A.1}$$

$$= e^{-r_{\boldsymbol{\theta}}\Delta t}\sum_{k=0}^{\prime N-1}\mathrm{Re}\left\{\left(f_R\left(\frac{k\pi}{b-a}\right)+i\cdot f_I\left(\frac{k\pi}{b-a}\right)\right)\right. \tag{A.2}$$

$$\left.\cdot\left(\cos\left(k\pi\frac{\boldsymbol{x}-a}{b-a}\right)+i\cdot\sin\left(k\pi\frac{\boldsymbol{x}-a}{b-a}\right)\right)\right\}V_k \tag{A.3}$$

$$= e^{-r_{\boldsymbol{\theta}}\Delta t}\sum_{k=0}^{\prime N-1}\mathrm{Re}\left\{\sum_{j=0}^{N_\varphi-1}\left[A_j\cos\left(j\pi\frac{u_k-a_\varphi}{b_\varphi-a_\varphi}\right)+i\cdot B_j\sin\left(j\pi\frac{u_k-a_\varphi}{b_\varphi-a_\varphi}\right)\right]\right. \tag{A.4}$$

$$\left.\cdot\left[\cos\left(k\pi\frac{\boldsymbol{x}-a}{b-a}\right)+i\cdot\sin\left(k\pi\frac{\boldsymbol{x}-a}{b-a}\right)\right]\right\}V_k \tag{A.5}$$

A

$$= e^{-r_{\boldsymbol{\theta}}\Delta t} \sum_{k=0}^{\prime N-1} \sum_{j=0}^{N_\varphi - 1} \left(A_j \cos\left(j\pi \frac{u_k - a_\varphi}{b_\varphi - a_\varphi} \right) \cos\left(k\pi \frac{\boldsymbol{x} - a}{b - a} \right) \right. \tag{A.6}$$

$$\left. - B_j \sin\left(j\pi \frac{u_k - a_\varphi}{b_\varphi - a_\varphi} \right) \sin\left(k\pi \frac{\boldsymbol{x} - a}{b - a} \right) \right) V_k \tag{A.7}$$

$$= e^{-r_{\boldsymbol{\theta}}\Delta t} \sum_{j=0}^{N_\varphi - 1} A_j \left[\sum_{k=0}^{\prime N-1} \cos\left(j\pi \frac{u_k - a_\varphi}{b_\varphi - a_\varphi} \right) \cos\left(k\pi \frac{\boldsymbol{x} - a}{b - a} \right) V_k \right] \tag{A.8}$$

$$- e^{-r_{\boldsymbol{\theta}}\Delta t} \sum_{j=0}^{N_\varphi - 1} B_j \left[\sum_{k=0}^{\prime N-1} \sin\left(j\pi \frac{u_k - a_\varphi}{b_\varphi - a_\varphi} \right) \sin\left(k\pi \frac{\boldsymbol{x} - a}{b - a} \right) V_k \right] \tag{A.9}$$

$$:= e^{-r_{\boldsymbol{\theta}}\Delta t} \sum_{j=0}^{N_\varphi - 1} \left(A_j C_j(\boldsymbol{x}) - B_j S_j(\boldsymbol{x}) \right), \tag{A.10}$$

A.2. CUMULANTS OF CGMY AND GBM

The cumulants and drift-correction term of the CGMY model and the Geometric Brownian Motion are represented by the values c_1, c_2, c_4, and w, respectively. These values, along with those from other established models obtained from [5], are presented in Table A.1:

Table A.1: Cumulants, denoted as c_n, characterize $\ln(S_t/K)$ across various models of the underlying asset. Additionally, we have the drift correction term, represented by w, which fulfills the condition $\exp(-wt) = \varphi(-i,t)$.

Model:	Cumulants:
GBM	$c_1 = \mu T$ $c_2 = \sigma^2 T$ $c_4 = 0$ $w = 0$
CGMY	$c_1 = \mu T + CT\Gamma(1-Y)\left(M^{Y-1} - G^{Y-1}\right)$ $c_2 = \sigma^2 T + CT\Gamma(2-Y)\left(M^{Y-2} + G^{Y-2}\right)$ $c_4 = CT\Gamma(4-Y)\left(M^{Y-4} + G^{Y-4}\right)$ $w = -C\Gamma(-Y)\left[(M-1)^Y - M^Y + (G+1)^Y - G^Y\right]$
VG	$c_1 = (\mu+\theta)T$ $c_2 = \left(\sigma^2 + \nu\theta^2\right)T$ $c_4 = 3\left(\sigma^4\nu + 2\theta^4\nu^3 + 4\sigma^2\theta^2\nu^2\right)T$ $w = \frac{1}{\nu}\ln\left(1 - \theta\nu - \sigma^2\nu/2\right)$
Heston	$c_1 = \mu T + \left(1 - e^{-\lambda T}\right)\frac{\bar{u} - u_0}{2\lambda} - \frac{1}{2}\bar{u}T$ $c_2 = \frac{1}{8\lambda^3}\left(\eta T\lambda e^{-\lambda T}(u_0 - \bar{u})(8\lambda\rho - 4\eta)\right.$ $+ \lambda\rho\eta\left(1 - e^{-\lambda T}\right)(16\bar{u} - 8u_0)$ $+ 2\bar{u}\lambda T\left(-4\lambda\rho\eta + \eta^2 + 4\lambda^2\right)$ $+ \eta^2\left((\bar{u} - 2u_0)e^{-2\lambda T} + \bar{u}\left(6e^{-\lambda T} - 7\right) + 2u_0\right)$ $\left.+ 8\lambda^2(u_0 - \bar{u})\left(1 - e^{-\lambda T}\right)\right)$ $w = 0$

BIBLIOGRAPHY

[1] F. Fang and C. W. Oosterlee, "Pricing early-exercise and discrete barrier options by fourier-cosine series expansions," *Numerische Matematik*, vol. 114, pp. 27–62, 2009. DOI: 10.1007/s00211-009-0252-4.

[2] M. J. Ruijter and C. W. Oosterlee, "Two-dimensional fourier cosine series expansion method for pricing financial options," *SIAM Journal on Scientific Computing*, vol. 34, 5 2012, ISSN: 10957200. DOI: 10.1137/120862053.

[3] Y. Hu, J. Ni, and L. Wen, "A hybrid deep learning approach by integrating lstm-ann networks with garch model for copper price volatility prediction," *Physica A: Statistical Mechanics and its Applications*, vol. 557, p. 124 907, 2020, ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2020.124907. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437120304696.

[4] J. Ruf and W. Wang, "Neural networks for option pricing and hedging: A literature review," *Journal of Computational Finance*, 2019, Available at SSRN: https://ssrn.com/abstract=3486363 or http://dx.doi.org/10.2139/ssrn.3486363.

[5] F. Fang and C. W. Oosterlee, "A novel pricing method for european options based on fourier-cosine series expansions," *SIAM Journal on Scientific Computing*, vol. 31, pp. 826–848, 2 2008, ISSN: 10648275. DOI: 10.1137/080718061.

[6] J. D. Spiegeleer, D. B. Madan, S. Reyners, and W. Schoutens, "Machine learning for quantitative finance: Fast derivative pricing, hedging and fitting," *Quantitative Finance*, vol. 18, pp. 1635–1643, 10 Oct. 2018, ISSN: 14697696. DOI: 10.1080/14697688.2018.1495335.

[7] B. Wang and T. Chen, "Gaussian process regression with multiple response variables," *Chemometrics and Intelligent Laboratory Systems*, vol. 142, pp. 159–165, Mar. 2015, ISSN: 18733239. DOI: 10.1016/j.chemolab.2015.01.016.

[8] C. W. Oosterlee and L. A. Grzelak, "Mathematical modeling computation in finance."

[9] J.-F. L. Gall, *Graduate texts in mathematics brownian motion, martingales, and stochastic calculus*. [Online]. Available: http://www.springer.com/series/136.

[10] D. Applebaum, *Lévy processes-from probability to finance and quantum groups*.

[11] P. Carr, H. Geman, D. B. Madan, and M. Yor, *The fine structure of asset returns: An empirical investigation**.

[12] D. Higham, *An introduction to financial option valuation: mathematics, stochastics and computation*, English. United Kingdom: Cambridge University Press, 2004, ISBN: 0521547571. DOI: 10.2277/0521547571.

A

[13] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 637–654, 1973. [Online]. Available: http://www.jstor.org/stable/1831029.

[14] S. E. Shreve, *Stochastic calculus for finance 2, Continuous-time models*. New York, NY; Heidelberg: Springer, 2004, ISBN: 0387401016 9780387401010. [Online]. Available: http://www.worldcat.org/search?qt=worldcat_org_all&q=0387401016.

[15] F. Dominici, J. J. Faraway, M. Tanner, J. Zidek, and P. J. Smith, "Linear algebra and matrix analysis for statistics chapman hall/crc texts in statistical science series series editors analysis of failure and survival data." [Online]. Available: www.ebook3000.com.

[16] G. Strang, *Introduction to Linear Algebra (Gilbert Strang, 5)*, English, Hardcover. Wellesley-Cambridge Press, Apr. 30, 2023, p. 440, ISBN: 978-1733146678. [Online]. Available: https://lead.to/amazon/com/?op=bt&la=en&cu=usd&key=1733146679.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Third. The Johns Hopkins University Press, 1996.

[18] J. P. Boyd, "Chebyshev and fourier spectral methods second edition." [Online]. Available: http://www-personal.engin.umich.edu/%E2%88%BCjpboyd/2000.

[19] C. Oosterlee and A. Almendral, "On american options under the variance gamma process," *Applied Mathematical Finance*, vol. 14, pp. 131–152, May 2007. DOI: 10.1080/13504860600724885.

[20] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, 2009. DOI: 10.1137/07070111X.

[21] H. Lu, K. N. Plataniotis, and A. Venetsanopoulos, *Multilinear Subspace Learning*. Chapman and Hall/CRC, Dec. 2013, ISBN: 9780429108099. DOI: 10.1201/b16252. [Online]. Available: https://www.taylorfrancis.com/books/9781439857298.

[22] N. Yang, Y. Liu, and Z. Cui, "Pricing continuously monitored barrier options under the sabr model: A closed-form approximation," *Journal of Management Science and Engineering*, vol. 2, pp. 116–131, 2 Jun. 2017, ISSN: 25895532. DOI: 10.3724/SP.J.1383.202006.

[23] J. H. Conway and R. K. Guy, *The book of numbers*, en, 1st ed. New York, NY: Springer, Feb. 1998.

[24] Z. Cheng, "Dimension reduction techniques for multi-dimensional numerical integrations based on fourier-cosine series expansion," M.S. thesis, Delft University of Technology, 2022. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid:f6ffbcd6-df14-425b-84bb-63e4e105205c.

[25] Z. Cheng and M. Brands, "Finding dimension-reduced fourier-cosine series expansion via supervised machine-learning and its application in multi-dimensional integration," 2023.

[26] N. D. Sidiropoulos, L. D. Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," Jul. 2016. DOI: 10.1109/TSP.2017.2690524. [Online]. Available: http://arxiv.org/abs/1607.01668%20http://dx.doi.org/10.1109/TSP.2017.2690524.

[27] J. Håstad, "Tensor rank is np-complete," *Journal of Algorithms*, vol. 11, no. 4, pp. 644–654, 1990, ISSN: 0196-6774. DOI: https://doi.org/10.1016/0196-6774(90)90014-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0196677490900146.

[28] M. Brands, "Solving multivariate expectations using dimension-reduced fourier-cosine series expansion and its application in finance," 2023. [Online]. Available: http://repository.tudelft.nl/.