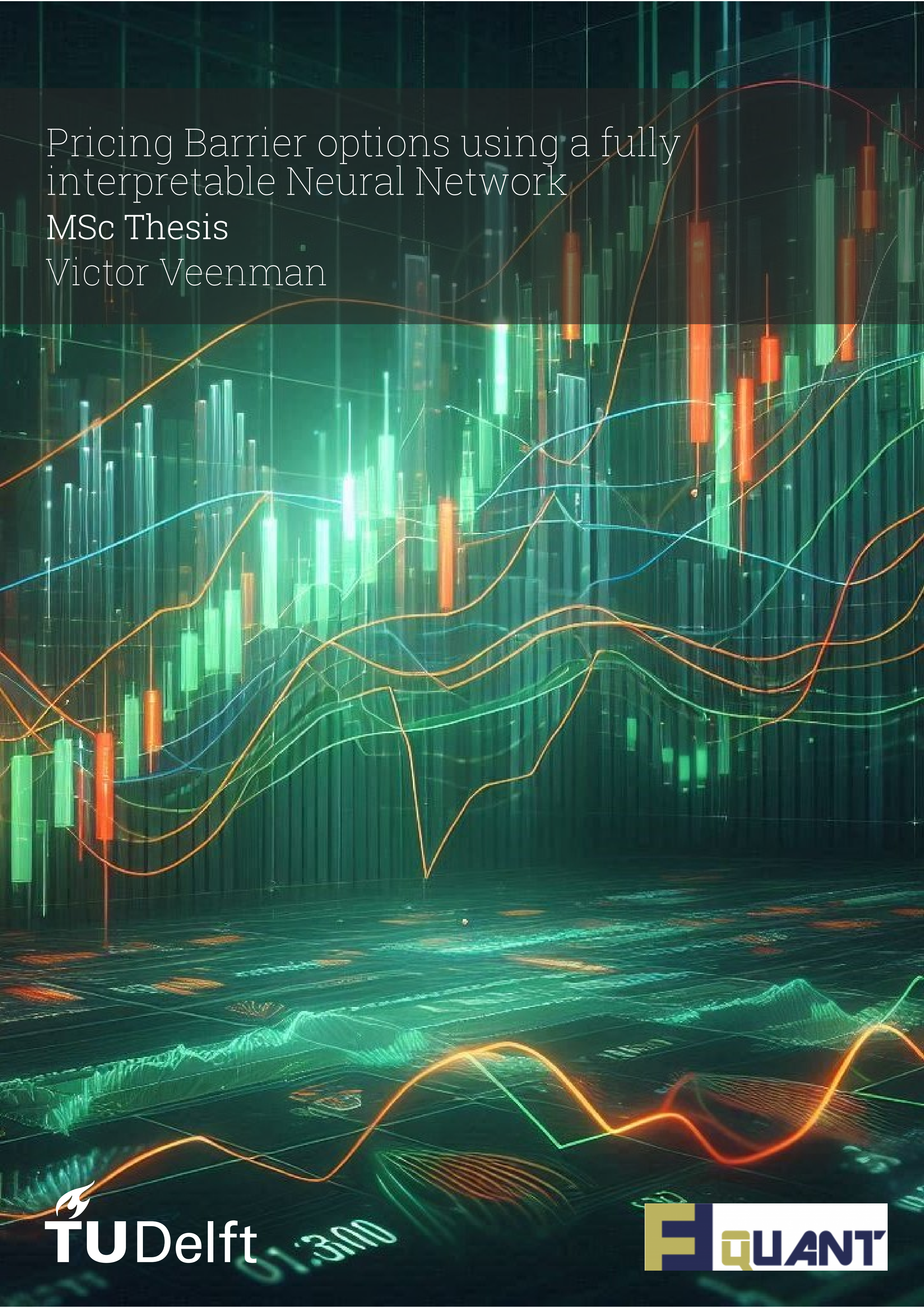


Pricing Barrier options using a fully interpretable Neural Network

MSc Thesis

Victor Veenman

Pricing Barrier options using a fully interpretable Neural Network

by

Victor Veenman

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday October 10, 2024 at 13:00 AM.

Student number:	4833228	
Project duration:	January 15, 2024 – October 15, 2024	
Thesis committee:	Dr. F. Fang,	TU Delft, Supervisor
	Prof. dr. ir. C. Vuik,	TU Delft
	Dr. J. Anderluh,	TU Delft
	Prof. dr. ir. K. In't Hout,	University Antwerp
	Dr. X. Shen,	FF Quant Advisory, Supervisor

Abstract

Although useful for hedging purposes, the valuation of continuously monitored barrier options comes with its challenges. This thesis applies the Dimension-reduced Fourier-cosine series expansion [2] in combination with the pricing PDE to approximate the price function of continuously monitored knock-out barrier options. We significantly extend earlier work presented in [3] in several ways:

1. While [3] focuses on solving option pricing problems under Heston's stochastic volatility model, we tackle a more complex stochastic volatility model, the Stochastic-$\alpha\beta\rho$ (SABR) model, for which no analytical solution exists for the associated distribution;
2. The curse of dimensionality in the original method is addressed using the COS-CPD approach, developed in [2], [4]. "COS" stands for the Fourier-Cosine series expansion method initially introduced in [5] and "CPD" is short for Canonical Polyadic Decomposition [6], [7];
3. We demonstrate that the COS-CPD method corresponds to a Neural Network architecture, which is fully interpretable by construction. We name this network the COS-CPD network;
4. The fully interpretable COS-CPD network is applied to price barrier options under Geometric Brownian Motion (GBM) and SABR model.

We first replicate results of [3] for the case where the underlying asset follows GBM and later directly extend the original method in [3] to the SABR model. That is, the valuation problem of continuously monitored barrier options can be transformed into finding the survival characteristic function.

This survival characteristic function satisfies another PDE, resulting from inserting the Fourier series expansion of option price, acquired through the COS method [5], into the option pricing PDE. As in [3], the choice of sine or cosine expansions is motivated separately for each dimension. For the time dimension especially, we expand on the first order derivative of the unknown function in time instead of expanding on the unknown function directly. Integrating the Fourier expansion of the time derivative then yields the trigonometric expansion. Results in both [3] and our tests demonstrate that the resulting trigonometric expansion in the time dimension has better convergence than directly expanding the unknown function in the time dimension.

We then address the curse of dimensionality in the original method of [3] by applying CPD on the series expansion coefficient tensor, following the idea from [4]. The decomposition is performed via supervised machine learning, where the components, or factor matrices, are solved using Alternating Least Squares (ALS). This algorithm can be formulated as a neural network, which is fully interpretable since it completely based on mathematical derivations.

One issue in this new method is that over fitting can result in very large condition numbers in the linear systems of which the solutions are the factor matrices. By incorporating regularization into the ALS, this problem can be greatly alleviated.

Testing results suggest that the computational time is far superior to Monte Carlo simulations. Note that our method trains the neural network offline, allowing option prices to be instantly computed for a range of maturity times and initial asset prices. Additionally, our COS-CPD model demonstrates superior accuracy compared to other numerical methods and machine learning models.

Keywords: Option pricing, Barrier option, Pricing PDE, Geometric Brownian Motion, SABR Model, COS method, COS-CPD, Fourier series expansion, Canonical Polyadic Decomposition, Neural Network, Machine Learning

Contents

1 Introduction

Derivative pricing is one of the most researched areas in quantitative finance. Accurate pricing of derivatives plays a crucial role in hedging and portfolio management in general. Among these derivatives, options are particularly interesting for their versatility and the various hedging strategies they enable.

Options can roughly be categorized into vanilla options (European and American) and exotic options. Examples of exotic options include Bermudan options, all-or-nothing options, and path dependent options like Asian options and Barrier options. This thesis focuses on Barrier options, which act like European options unless the underlying asset breaches a predetermined barrier before maturity. This breach can be checked on discrete monitoring dates or continuously. We consider continuously monitored Barrier options.

A closed-form pricing formula for continuously monitored barrier options is only available if we assume the underlying asset follows Geometric Brownian Motion (GBM) dynamics. This GBM assumption, however, is not very realistic for modelling stock processes. To better reflect the volatility smiles implied from the market, other models have been developed. Among these models are the stochastic volatility models, where not only the asset price, but also the volatility follows a stochastic process. For this reason, these models are more complex and as a result, closed-form solutions are much more difficult to derive. The stochastic volatility model researched in this thesis is the Stochastic-$\alpha\beta\rho$ (SABR) model.

1.1. Literature review

Over the years, option pricing has seen many different approaches. In their ground breaking research in 1973 [8], Black and Scholes provide an analytic solution for European call and put options where the underlying asset follows a GBM process. All parameters of the Black-Scholes model are easily observed except for the volatility. The *implied volatility* can be computed by finding the volatility σ_{IV} for which the analytically computed option price is equal to the market option price. This implied volatility varies with different strikes and maturities, forming what is known as the *implied volatility surface*.

For barrier options, Hagan et al. [9] noted that we do not know whether we should use the implied volatility at the strike price, the boundaries or somewhere in between. These uncertainties lead to problems with hedging.

Local volatility model

To address these issues, local volatility models were developed [10], [11]. In local volatility models, the volatility is considered piecewise constant. Since options are usually available for specific exercise dates (e.g. 1,2,3,6,12 months from now), the volatility is considered constant for the time periods between those exercise dates. Then the implied volatility for the first month is calibrated on the option prices for the first month, similar to the Black-Scholes method. The implied volatility for the second month is then calibrated on the option prices with a 2-month exercise date, incorporating the previously obtained implied volatility for the first month. This process continues for all time intervals.

With this calibration, the local volatility model can reproduce the market prices for all strikes and maturity times. The model provides consistent delta and vega risks, and thus resolves the hedging issues. However, Hagan et al. [9] commented that "the local volatility model predicts the wrong dynamics of the implied volatility curve, which leads to inaccurate and unstable hedges". They found that the local volatility models predict the market smile and the price of the underlying asset to move in opposite directions.

SABR model

In 2002, Hagan et al. [9] introduced the SABR model. Their goal was to develop a model that accurately fits the market smile and follows the dynamics of the implied volatility curve. The SABR model dynamics consist of the forward price and the stochastic volatility, with the former described by a Constant Elasticity of Variance (CEV) process and the latter by a GBM without drift. Both processes are correlated by correlation coefficient ρ. Hagan et al. have derived analytic formula's for the implied volatility, which can be used to fit the parameters. For long maturities and strikes far out of the money, however, these formulas are not arbitrage free and predict wrong dynamics [12].

The SABR model offers great flexibility in skewness. This allows for providing a good fit to the volatility surface.

Option pricing methods

While Black-Scholes provides exact analytic solution, such results are rarely available for more complex dynamics. Therefore, other methods have been developed to price options with more complicated underlying asset dynamics.

The first method we discuss is closely connected to these exact analytic solutions. Although exact solutions might not be available, closed-form approximations can still be derived. In [13] an approximation of the survival density function under SABR is provided. The barrier option price is then found by evaluating a one-dimensional integral of its payoff function and the survival density. This method has low computational complexity. However, due to the closed-form approximation, there is no convergence to decrease the errors by increasing computational complexity. Therefore, if more accurate results are required, this method may no longer be satisfactory.

The next method is the well-known Monte-Carlo simulation, which is widely used in industry [14]. Paths of the underlying are simulated and the correct option price is the average of the discounted payoff function value for each realisation of the path. According to the law of large numbers, this method converges to the true option price. In practice, Monte Carlo simulation often requires many paths to achieve the desired accuracy. Additionally, discretization of the time dimension also introduces errors, necessitating many time steps for good results, leading to a computationally complex method.

To combat these errors from time discretization Broadie and Kaya [15] introduced an exact simulation scheme to simulate asset prices at time T given some initial values at time t. Their paper focuses on Heston's model, another stochastic volatility model. A similar exact simulation scheme is given for asset prices following SABR dynamics in [16]. These simulation schemes are accurate for non path-dependent options like European options, since they bypass the time discretization completely. As for path-dependent options, no closed-form approximation exists yet, and thus the valuation of those options would still require one to simulate the asset price at multiple time steps to check these path dependencies. A more advanced numerical method, such as the COS method [5], is unfortunately not directly applicable, since the analytic characteristic function of SABR is unknown.

Another popular method of pricing options assuming more complicated stochastic processes is deriving the pricing PDE and solving it. Various numerical methods can be applied to solve these PDEs [17]–[20]. These numerical methods often require extensive calculations to acquire good accuracy. Additionally, for each set of model parameters and initial values, recalculation is required.

A type of methods for option pricing that has recently been very popular, is machine learning. Early methods used an artificial neural network to directly approximate the option price given the values of input parameters [21], [22]. These neural networks are calibrated through fitting to the output values of the unknown functions using corresponding input values. Once the function is trained, option price computation is instantaneous, as long as the computational complexity is linear in the number of nodes. These machine learning models, however, are only applicable after they are trained. Another significant disadvantage of these models is the lack of interpretability. They are thus called "Black box" models. The machine is trained on input/output pairs, but the relationship between these pairs is not understood, making it difficult to validate and explain the model behaviours and capture the induced risks.

Another machine learning method, supervised deep neural networks (SDNNs), developed in [23], does explicitly assume an underlying stochastic process. Therefore, it requires more inputs, but results in a more generally trained model that is applicable to any option where the underlying follows the given stochastic process. This model, however, still lacks interpretability.

In summary, existing methods of option pricing have their advantages and disadvantages.

On the one hand, numerical methods and Monte Carlo simulations, for instance, are highly interpretable and generally intuitive. They allow for a clear understanding of the underlying processes and the assumptions made. However, these methods suffer from large computational costs and need to be recalculated for each option, which can be time-consuming and resource-intensive.

On the other hand, machine learning models offer the advantage of speed. Once trained, these models can instantly provide multiple option prices, making them highly efficient for real-time applications. However, they lack interpretability, often functioning as “black boxes”.

A new barrier pricing method

Recent research work at FF Quant aims to combine the strengths of machine learning methods and the interpretability of analytical/numerical methods [3]. Their model is based on the pricing PDE and combines with the COS method to price continuously monitored barrier options both GBM and Heston’s model. By approximating the function price using the COS method from [5] and substituting this approximation into the pricing PDE, a PDE for the survival characteristic function (ch.f.) is derived. A trigonometric expansion is then used to approximate these survival ch.f.’s. By selecting training points within the expansion domain and substituting the survival ch.f.’s into the new PDE, the problem is translated to a linear system, which can be solved by a direct or iterative solver. The method yielded good accuracy, but suffers greatly from the curse of dimensionality.

Another branch of research carried out at FF Quant focuses on addressing this curse of dimensionality [2], [4]. Their solution, called COS-CPD, combines Canonical Polyadic Decomposition (CPD) with the COS method. The resulting method is sufficiently accurate, while less prone to the curse of dimensionality. CPD decomposes the coefficient tensor of the original approximation into a product of factor matrices. Due to this product of matrices, the problem can no longer be translated into a linear system, so another algorithm, Alternating Least Squares (ALS), is used to train the coefficients. CPD has been studied extensively in literature, of which a comprehensive overview is given by [24]. Regular and fast convergence of the COS-CPD method has been reported in [2] and [4].

1.2. Thesis objective and outline

Following prior research at FF Quant [3], we aim to extend their model and develop an interpretable neural network to price continuously monitored barrier options. These extensions can be divided into two parts. Firstly, unlike [3] which focuses on Heston’s stochastic volatility model, we focus the more complex SABR model, for which no analytical solutions exist for the associated distribution. Secondly, we address the curse of dimensionality by applying the COS-CPD method from [2], [4]. We demonstrate that the COS-CPD method can be presented as a neural network, which we name the COS-CPD network.

The rest of the thesis is organized as follows. In Chapter 2 we go over the preliminary results required for the later chapters. To relate our COS-CPD work to existing neural networks, we discuss several neural networks in Chapter 3. This chapter revisits the COS-CPD method developed in [2], [4] and works out its neural network representation, which is used in later chapters. Chapter 4 then shows numerical results for the convergence rate of trigonometric expansion. Next, in Chapter 5, we replicate results from [3] under the assumption that the underlying asset follows a GBM process. We also extend the original method for better accuracy using change of variables. Chapter 6 closely follows the steps of Chapter 5, but extends SABR dynamics. The stochastic volatility yields a more realistic model, but also increases dimensionality, which leads to computational problems. Therefore, in Chapters 7 and 8 we apply the COS-CPD network for GBM and SABR respectively, which scales better with a higher number of dimensions. Lastly, in Chapter 9 we summarize the results, draw conclusions on the performance and suggest areas for future research.

2
Mathematical Framework

This thesis uses various mathematical tools and techniques. This chapter presents the relevant techniques and some basic derivations to be able to find option prices later in the thesis.

2.1. Stochastic calculus

This section provides definitions and theorems from the field of stochastic calculus. The contents are based on [25] and [26].

Definition 2.1.1 (Filtration)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. A filtration on $(\Omega, \mathcal{F}, \mathbb{P})$ is a family of sub σ-fields $\{\mathcal{F}_t, t \geq 0\}$ of $\mathcal{F}$ indexed by $t \in [0, \infty)$, such that $\mathcal{F}_s \subset \mathcal{F}_t$ for every $s \leq t \leq \infty$.*

Definition 2.1.2 (Adapted process)**.** *A process $X = \{X_t, t \geq 0\}$ is said to be adapted to a filtration $\{\mathcal{F}_t, t \geq 0\}$ if for all $t \geq 0$, X_t is $\mathcal{F}_t$ measurable.*

Definition 2.1.3 (Martingale)**.** *Let $M = \{M_t, t \geq 0\}$ be a process defined on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ equipped with a filtration $\{\mathcal{F}_t, t \geq 0\}$. Then M is said to be a martingale if*

1. *Adepted: M_t is $\mathcal{F}_t$ measurable for all $t \geq 0$,*
2. *Integrable: M_t is integrable for all $t \geq 0$,*
3. *Martingale property: For all $0 \leq s < t$*

$$\mathbb{E}(M_t | \mathcal{F}_s) = M_s, \qquad \forall 0 \leq s \leq t.$$

Definition 2.1.4 (Stopping time)**.** *A stopping time $\tau : \Omega \to [0, \infty)$ is a random variable such that for all $t \geq 0$, the event $\tau \leq t$ belongs to $\mathcal{F}_t$.*

Definition 2.1.5 (Local martingale)**.** *A rightcontinuous with left limits adapted process L is said to be a local martingale if there exist stopping times $\tau^n \uparrow \infty$ such that for each n, the process M^n defined by $M^n := L_{t \wedge \tau^n}$ is a martingale.*

Definition 2.1.6 (Semimartingale)**.** *A stochastic process $X = \{X_t, t \geq 0\}$ is called a semimartingale if it can be decomposed as*

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

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

1. *Starting at 0: $W(0) = 0$,*
2. *Normally distributed increments: For all $0 \leq s < t$, $W(t) - W(s) \simeq N(0, t - s)$.*
3. *Independent increments: For $0 \leq t_0 < t_1 < \cdots < t_n$ 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.*

Definition 2.1.8 (Itô integral). *For any square-integrable adapted process $g(t)$ with continuous sample paths, the Itô integral is given by*

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

where $I_m(T) = \int_0^T g_m(t)dW(t)$ for some elementary process $g_m(t) = \sum_{j=0}^{m-1} \eta_j \mathbb{1}_{[t_j,t_{j+1})}$ 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 $j = 0, \ldots, m-1$ and square-integrable.

Definition 2.1.9 (Itô isometry). *For any stochastic process $g(t)$, satisfying the usual regularity conditions, the following holds*

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

Definition 2.1.10 (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 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 process $(X_t)_{t\geq 0}$. Itô's formula is often expressed in differential form

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

2.2. Underlying process dynamics

The asset price is a continuous stochastic process. Since we do not know its value at future times, we must model the price with a suitable stochastic process for future calculations. This thesis investigates two such models. The first model, geometric brownian motion (GBM), is the most well-known process and allows for simpler expressions at the cost of less realistic assumptions. Many closed-form solutions are known, making it a excellent model to test new pricing approaches. The GBM model is investigated in Chapter 5, reproducing results found in [3]. The second model, the Stochastic-$\alpha\beta\rho$ (SABR) model, is a stochastic volatility model. This has more realistic assumptions, but this comes at the cost of more complicated expressions, no closed-form exact solutions and other challenges. This model is used in Chapter 6.

Geometric brownian motion

A process S_t following a GBM dynamics satisfies the Stochastic Differential Equation (SDE)

$$dS_t = \mu S_t dt + \sigma S_t dW_t^{\mathbb{P}},$$

where μ is the drift, σ the volatility, and $W_t^{\mathbb{P}}$ a Brownian motion under real world measure $\mathbb{P}$. We have to convert this into the risk-neutral measure $\mathbb{Q}$ before we can use it for our calculations. Under the risk-neutral measure, we want the expected asset price value at a future time to grow at the same rate as the risk-free interest rate r. Then

$$\mathbb{E}^{\mathbb{Q}}\left(e^{-r(T-t)}S_T|\mathcal{F}_t\right) = S_t,$$

where T is the future time, t the current time at which we have information $\mathcal{F}_t$. Note that this is true if the underlying stock price follows the GBM process

$$dS_t = rS_t dt + \sigma S_t dW_t^{\mathbb{Q}}, \tag{2.1}$$

where we assume both r and σ to be constant. Through the martingale approach explained in more detail in Section 5.1, we find the Black Scholes pricing PDE (5.1)

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

SABR

The Stochastic-$\alpha\beta\rho$ (SABR) model is a stochastic volatility model. This means that not only the asset price, but also the volatility follows a stochastic process. The model was introduced by Hagan et al. [9] and follows the dynamics

$$\begin{aligned} dS_t &= \sigma_t S_t^{\beta} dW_t^{\mathbb{Q}} \\ d\sigma_t &= \alpha\sigma_t dZ_t^{\mathbb{Q}}, \end{aligned} \tag{2.2}$$

where $W_t^{\mathbb{Q}}$ and $Z_t^{\mathbb{Q}}$ are correlated Brownian motions with correlation ρ, so $dW_t^{\mathbb{Q}} dZ_t^{\mathbb{Q}} = \rho dt$, $-1 < \rho < 1$. $0 \leq \beta \leq 1$ is the skewness parameter and $\alpha \geq 0$ is the volatility of volatility.
We can also rewrite these dynamics in terms of 2 independent Brownian motions.

$$\begin{aligned} dS_t &= \sigma_t S_t^{\beta} d\tilde{W}_t^{\mathbb{Q}} \\ d\sigma_t &= \alpha\sigma_t\left(\rho d\tilde{W}_t^{\mathbb{Q}} + \sqrt{1-\rho^2} d\tilde{Z}_t^{\mathbb{Q}}\right), \end{aligned}$$

2.3. Options

Options are financial contracts that give the holder the right (but not the obligation) to purchase (or sell) an underlying asset for a predetermined price at some future time.

Option pricing is widely researched. Since you can never lose more than the price you paid for the option, it is an attractive investment and at the same time, options can be used for hedging. The most basic form of options are European call and put options.

Definition 2.3.1 (European call/put option)**.** *A European call/put option is a contract signed between two parties at some time t from which the right (but not the obligation) arises to buy/sell a basic asset S with price S_t (underlying) at predetermined time $T > t$ (maturity) for a predetermined price E (strike).*

At maturity time T the holder of the option can choose to either exercise the option or to do nothing. Therefore, the payoff for an option is always non-negative. The payoff can be written as

$$\begin{aligned} V_{call}(T,S) &= (S_t - E)^+ = \max(S_t - K, 0) \\ V_{put}(T,S) &= (E - S_t)^+ = \max(K - S_t, 0), \end{aligned}$$

where S_T is the value of the underlying asset at maturity T and E is the strike price.

Many other options exist, most of which we will not discuss in this thesis. The option type that we discuss in detail is a type of exotic option. These options are called exotic because they consider the path of the underlying asset. The type of options we consider are barrier options.

Definition 2.3.2 (Barrier option)**.** *A Barrier option is a contract signed between two parties at some time t from which the right (but not the obligation) arises to buy/sell a basic asset S with price S_t (underlying) at predetermined time $T > t$ (maturity) for a predetermined price E (strike) if the barrier condition is satisfied. The four main types of barrier conditions are*

- ***Up-and-out**: Asset price starts below the barrier level and does not cross the cross the barrier level B or the contract becomes void. $S_t < B$, $S_\tau < B \forall t < \tau \leq T$,*
- ***Down-and-out**: Asset price starts above the barrier level and does not cross the cross the barrier level B or the contract becomes void. $S_t > B$, $S_\tau > B \forall t < \tau \leq T$,*
- ***Up-and-in**: Asset price starts below the barrier level and has to cross the cross the barrier level B or the contract becomes void. $S_t < B$, $S_\tau > B$ for some $t < \tau \leq T$,*
- ***Down-and-in**: Asset price starts below the barrier level and has to cross the cross the barrier level B or the contract becomes void. $S_t > B$, $S_\tau < B$ for some $t < \tau \leq T$.*

In this thesis we look at the up-and-out barrier option, but the methodology for down-and-out barrier options is identical. Since the sum of knock-out and knock-in barrier options is a European option, these knock-in barrier option prices can be found given a good approximation for the European option price.

If we consider stopping time $\tau = \inf\{\tilde{t} \geq t : S_{\tilde{t}} \geq B\}$ the first time that asset price S_t is greater or equal than barrier B, we find the payoff functions for a up-and-out barrier put and call option

$$V^B_{call}(T,S) = (S_T - E)^+ \mathbb{1}_{\tau > T}$$
$$V^B_{put}(T,S) = (E - S_T)^+ \mathbb{1}_{\tau > T},$$

where E is the strike price. If the barrier is not crossed before the maturity time, this payoff is just equal to that of the European options.

2.3.1. Option valuation

In quantitative finance, the *fair* value of options is heavily researched. A fair price is a price that both the buyer and the seller find acceptable. One of the key principles on which option valuation theory is based, is the no-arbitrage principle.

Definition 2.3.3 (Arbitrage)**.** *An investment strategy is called an arbitrage if the value process V_t of the strategy satisfies the following properties:*

- *Zero initial cost: $V_t \leq 0$,*
- *Positive probability of gain: $\mathbb{P}\Big(V_T > e^{r(T-t)}\Big)V_t > 0$,*
- *Certainty of no loss: $\mathbb{P}\Big(V_T < e^{r(T-t)}V_t\Big) = 0$,*

where t is the initial time, $T > t$ the maturity time and r is the risk-free interest rate.

In other words, the no-arbitrage principle ensures that there can not be an investment strategy with gain without risk.

2.4. Fourier series expansions

Let a function $f(x)$ satisfy the Dirichlet conditions on $[a,b]$, i.e. "*the piecewise function f must be periodic with at most a finite number of discontinuities, and/or a finite number of minima or maxima within one period*" [27, p. 176]. Then there are three common Fourier representations. These are the half-range Fourier cosine-series, the half-range Fourier-sine series, and the full-range Fourier series [28].

$$f(x) = {\sum_{k=0}^{\infty}}{}' A_k \cos\left(k\pi \frac{x-a}{b-a}\right), \tag{2.3a}$$
$$A_k = \frac{2}{b-a}\int_a^b f(x)\cos\left(k\pi\frac{x-a}{b-a}\right)dx,$$
$$f(x) = \sum_{k=1}^{\infty} B_k \sin\left(k\pi\frac{x-a}{b-a}\right), \tag{2.3b}$$
$$B_k = \frac{2}{b-a}\int_a^b f(x)\sin\left(k\pi\frac{x-a}{b-a}\right)dx,$$
$$f(x) = \frac{A_0}{2} + \sum_{k=1}^{\infty} A_k\cos\left(2k\pi\frac{x-a}{b-a}\right) + B_k\sin\left(2k\pi\frac{x-a}{b-a}\right), \tag{2.3c}$$
$$A_k = \frac{1}{b-a}\int_a^b f(x)\cos\left(2k\pi\frac{x-a}{b-a}\right)dx,$$
$$B_k = \frac{1}{b-a}\int_a^b f(x)\sin\left(2k\pi\frac{x-a}{b-a}\right)dx,$$

where the prime in 2.3a indicates that the first term is multiplied by $\frac{1}{2}$.

2.4.1. Convergence of Fourier series

Theorem 2.4.1 (Dirichlet-Jordan test for Fourier series)**.** *Let $f(x)$ be a periodic function of bounded variation. Then the Fourier series expansion $S_n f(x)$ converges as $n \longrightarrow \infty$ at each point of the domain to*

$$\lim_{\varepsilon \to 0} \frac{f(x+\varepsilon)+f(x-\varepsilon)}{2}.$$

In particular, if f is continuous at x, then the Fourier series converges to $f(x)$. Moreover, if f is continuous everywhere, then the convergence is uniform.

2.4.2. Spectral filters

At discontinuities and steep ascents/descents, functions described by Fourier series expansions often show oscillations related to the Gibbs phenomenon. Through spectral filters we aim to reduce the effect of these oscillations and find faster convergence. With filtering we multiply the expansion coefficients by a decreasing function, such that coefficients will decay faster. The following definition is from [29] and [30].

Definition 2.4.1 (Fourier space filter of order p)**.** *A real and even function $\hat{s}(\eta)$ is called a filter of order p if*

1. $\hat{s}(0) = 1, \quad \frac{\partial^m}{\partial \eta^m}\hat{s}(0) = 0, \quad 1 \leq m \leq p-1,$
2. $\hat{s}(\eta) = 0, \quad |\eta| \geq 1\,,$
3. $\hat{s}(\eta) \in C^{p-1}, \quad \eta \in (-\infty, \infty)$

If we consider the Fourier-cosine series, the filtered partial sum would be defined by

$$f_N(x) = \sum_{k=0}^{N-1}{}' \hat{s}\left(\frac{k}{N}\right)\hat{A}_k \cos\left(k\pi \frac{x-a}{b-a}\right). \tag{2.4}$$

Before we show some examples of filters, we first look at an important property that results from the definition.

From conditions (2) and (3) it follows that $\frac{\partial^m}{\partial \eta^m}\hat{s}(1) = 0$ for $0 \leq m \leq p-1$. Essentially, at higher modes ($k \longrightarrow N$), the function will be 0.

Filters have already been widely researched and therefore there are many examples available. We will consider relatively simple examples in this report

- Lanczos filter: $\hat{s}(\eta) = \frac{\sin(\pi\eta)}{\pi\eta}$. This is a filter of order $p = 1$
- Raised cosine filter: $\hat{s}(\eta) = \frac{1}{2}(1+\cos(\pi\eta))$. This is a filter of order $p = 2$
- Exponential filter: $\hat{s}(\eta) = \exp(-\alpha\eta^p)$, where $\alpha = -\log(\epsilon_m)$, with ϵ_m the machine epsilon. This filter has general order p, but p is required to be even.

2.4.3. Option pricing via Fourier-sine (cosine) series

This section presents the sine counterpart of the work of [5]. The sine option pricing formula is derived in Appendix A.1.1. We use the characteristic function, which forms a Fourier pair with the conditional probability density function (pdf) used in the pricing formula. This characteristic function can be used to approximate the conditional pdf through the half-range Fourier-sine series. This results in an approximation of the pricing function (A.1).

$$\begin{aligned} V_2(t,X) &= e^{-r(T-t)}\sum_{p=1}^{N}\hat{\phi}_p(t,X)V_p \\ \hat{\phi}_p(t,X) &= \Im\left[\phi\left(\frac{p\pi}{b-a},t;x\right)\cdot e^{-ip\pi\frac{a}{b-a}}\right] \\ V_p &= \frac{2}{b-a}\int_a^b V(T,y)\sin\left(p\pi\frac{y-a}{b-a}\right)dy, \end{aligned} \tag{2.5}$$

where ϕ is the characteristic function corresponding to $f(y|x)$, $\Im[\cdot]$ denotes the imaginary part, and V_p can be determined analytically if we have a European style payoff. For a European call option we find two analytic solutions. One for a model that considers the log-asset price and one for the asset price as shown in Equations (A.5) and (A.6).

$$V_p^{LogAsset,call} = \frac{2}{b-a}\left(\chi_p^{LogAsset}(\ln(E), b) - K\Psi_p(\ln(E), b)\right) \tag{2.6}$$

$$V_p^{Asset,call} = \frac{2}{b-a}\left(\chi_p^{Asset}(E, b) - K\Psi_p(E, b)\right) \tag{2.7}$$

where Ψ is as in Equation (A.2) and χ is defined in Equations (A.3) and (A.4).

2.5. Closed-form solutions

In this section we present some closed-form solutions that can be used as benchmark values for later approximations.

2.5.1. Up-and-out barrier under GBM

The up-and-out barrier call option with strike price E, barrier B and maturity T has a closed-form solution under GBM [31]. Assume S_t follows GBM with interest rate r and volatility σ, then the option price is given by

$$\begin{aligned} V(t,S) =& S\left(\Phi(d_1) - \Phi(e_1) - \left(\frac{B}{S}\right)^{1+\frac{2r}{\sigma^2}}(\Phi(f_2) - \Phi(g_2))\right) \\ &- Ee^{-r(T-t)}\left(\Phi(d_2) - \Phi(e_2) - \left(\frac{B}{S}\right)^{-1+\frac{2r}{\sigma^2}}(\Phi(f_1) - \Phi(g_1))\right), \end{aligned} \tag{2.8}$$

where $\Phi(\cdot)$ is the cumulative distribution function of a standard normal distribution and

$$\begin{aligned} d_1 &= \frac{\ln\left(\frac{S}{E}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \\ d_2 &= \frac{\ln\left(\frac{S}{E}\right) + \left(r - \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \\ e_1 &= \frac{\ln\left(\frac{S}{B}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \\ e_2 &= \frac{\ln\left(\frac{S}{B}\right) + \left(r - \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \\ f_1 &= \frac{\ln\left(\frac{S}{B}\right) - \left(r - \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \\ f_2 &= \frac{\ln\left(\frac{S}{B}\right) - \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \\ g_1 &= \frac{\ln\left(\frac{SE}{B^2}\right) - \left(r - \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \\ g_2 &= \frac{\ln\left(\frac{SE}{B^2}\right) - \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}}. \end{aligned}$$

2.5.2. Characteristic function of option price under GBM

Section 4.2 of [3] finds a benchmark solution for $\phi(\omega; t, x)$ by transforming the pricing PDE to a heat equation. This is done under the log-asset GBM model, therefore it is important that input x in the function is the log-asset price and a, b are the lower and upper bounds of the log-asset price. The benchmark function is

$$\begin{aligned}\phi(\omega; t, x) &= \sum_{k=1}^{\infty} e^{-\frac{1}{2}\sigma^2(T-t)\left(\left(\frac{k\pi}{b-a}\right)^2+\alpha^2\right)} e^{\alpha x} \sin\left(k\pi\frac{x-a}{b-a}\right)\left(\frac{2}{b-a}\int_{-\infty}^{\infty} e^{(i\omega-\alpha)y}\sin\left(k\pi\frac{y-a}{b-a}\right)dy\right) \\ &\approx \sum_{k=1}^{\infty} e^{-\frac{1}{2}\sigma^2(T-t)\left(\left(\frac{k\pi}{b-a}\right)^2+\alpha^2\right)} e^{\alpha x} \sin\left(k\pi\frac{x-a}{b-a}\right)\left(\frac{2}{b-a}\int_{a}^{b} e^{(i\omega-\alpha)y}\sin\left(k\pi\frac{y-a}{b-a}\right)dy\right), \qquad (2.9)\end{aligned}$$

where the integral is analytically evaluated in Section A.1.2.

$$\int_a^b e^{(i\omega-\alpha)y}\sin\left(k\pi\frac{y-a}{b-a}\right)dy = \frac{1}{(i\omega-\alpha)^2+(\frac{k\pi}{b-a})^2}\frac{k\pi}{b-a}\left(e^{(i\omega-\alpha)a} - e^{(i\omega-\alpha)b}\cos(k\pi)\right).$$

Note that this is a benchmark for the survival ch.f., but not for $\hat{\phi}_p$, so to find the benchmark for $\hat{\phi}_p$ we still have to use the formula

$$\hat{\phi}_p(t, X) = \Im\left[\phi\left(\frac{p\pi}{b-a}; t, x\right)e^{-ip\pi\frac{a}{b-a}}\right]$$

2.6. Collocation methods

Collocation methods are numerical techniques used for solving integral and differential equations. These methods transform the problem into a linear system of equations while introducing some approximations. Collocation methods have a similar approach to those discussed in Chapters 5 and 6.

Selection of basis functions

The collocation method builds an approximation of the function using basis functions. The three most commonly used basis functions are:

- Polynomial basis functions.
- Trigonometric basis functions. Trigonometric basis functions are used for expansions like the Fourier series expansions.
- Radial basis functions (RBF). RBFs are functions whose values depend on the distance from the origin [32]. Functions satisfying $\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|_2)$, where $\|\cdot\|_2$ is the Euclidean norm, are called radial functions.

Approximation of the function

Using the basis functions, an approximation can be made for the function that has to be found. This approximation will be of the form

$$V(t, S) \approx u(t, S) = \sum_{n=1}^{N} a_n u_n(t, S), \qquad (2.10)$$

where $u_n(t, S)$ are the basis functions and a_n are expansion coefficients. The goal now shifts from finding $V(t, S)$ to finding the coefficients a_n for which the approximation becomes sufficiently accurate.

Discretization of the domain and selecting collocation points

We start by discretizing the asset and time domains into a finite set of discrete points. Choosing how we want to discretize the dimensions is crucial to the performance of the method. The collocation points we choose, will determine where the PDE is enforced. We can split the discretization into two main types.

- Uniform grid: We select a number N of collocation points per dimension and select equidistant points over the domain of the dimension. Consider domain $[a, b]$ then the points would be $x_i = a + \frac{i(b-a)}{N-1}, i = 0, \ldots, N-1$.
- Non-uniform grid: If we know some properties of the function we wish to approximate, a non-uniform grid could give a more accurate result. If there are parts of the function where the function has very erratic changes, we require more collocation points around that part to get more accuracy, whereas other parts might require much less points.

A poor choice of collocation points can lead to very wrong results. If we look at the sine function for example, taking the value of $\sin(x)$ at $x = k\pi, k \in \mathbb{N}$, the value is always 0, so one might think we are looking at the function $f(x) = 0$.

Enforcing the PDE and solving the linear system

Next we enforce the PDE. By substituting approximation (2.10) into the PDE. This then results into a linear system where coefficients a_n are the unknowns. The coefficients a_n that satisfy this linear system can be substituted into approximation (2.10) to find the final approximation.

2.7. Tensor calculus

Tensors will play a major role in Canonical Polyadic Decomposition (CPD) and the results of Chapters 7 and 8. This section introduces the general concepts of Tensor calculus.

We start by introducing tensors and basic tensor operations. The definitions, terminology and theorems follow the works of [24] and [33].

Definition 2.7.1 (Tensor)**.** *An Nth-order **tensor** $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a real N-dimensional array, where the index range in the k-th **mode** is* $[1, \ldots, I_k]$.

First and second order tensors are also called vectors and matrices respectively. We now introduce some terminology related to tensors.

- **Entries:** To denote a specific *entry* of the tensor, we can use either one of the following notations. $\mathcal{A}[i_1, i_2, \ldots, i_N] = a_{i_1,i_2,\ldots,i_N} = a_{\mathbf{i}}$, where $\mathbf{i} = (i_1, \ldots, i_N)$
- **Fibers:** *Fibers* are the higher order analogue of matrix rows and columns. Third order tensors have *row* $(\mathcal{A}[:, i_2, i_3])$, *column* $(\mathcal{A}[i_1, :, i_3])$ and *tube* $(\mathcal{A}[i_1, i_2, :])$ fibers.
- **Slices:** For third order tensors, there is a term for 2-dimensional components of the tensor, where 1 dimension has fixed entry. These are called *slices*. Horizontal slices fix rows $(\mathcal{A}[i_1, :, :])$, lateral slices fix columns $(\mathcal{A}[:, i_2, :])$ and frontal slices fix tubes $(\mathcal{A}[:, :, 3])$.

A visual representation of fibers and slices is given in Figure 2.1.

Definition 2.7.2 (Outer product)**.** *Let $\mathbf{u} \in \mathbb{R}^M, \mathbf{v} \in \mathbb{R}^N$ be two vectors. Their **outer product**, denoted by $\mathbf{u} \circ \mathbf{v} \in \mathbb{R}^{M \times N}$ is defined as*

$$\mathbf{u} \circ \mathbf{v} = \begin{pmatrix} u_1 v_1 & u_1 v_2 & \cdots & u_1 v_N \\ u_2 v_1 & \ddots & & u_2 v_N \\ \vdots & & \ddots & \vdots \\ u_M v_1 & u_M v_2 & \cdots & u_M v_N \end{pmatrix}$$

An outer product of N vectors $\mathbf{a}_n$ will be denoted as

$$\mathop{\circ}_{n=1}^{N} \mathbf{a}_n = \mathbf{a}_1 \circ \mathbf{a}_2 \circ \cdots \circ \mathbf{a}_N$$

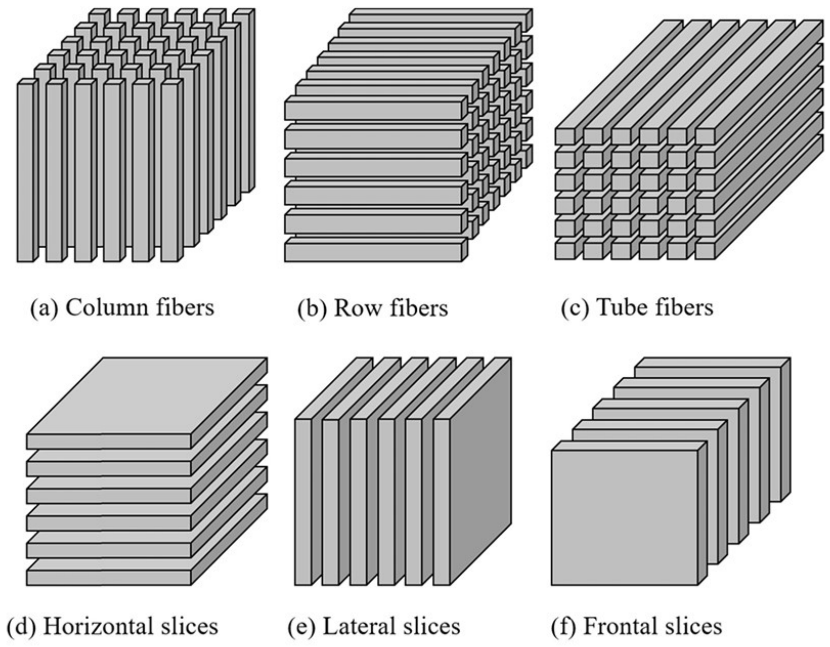


Figure 2.1: Fibers and slices of a third-order tensor [34, p. 25]

For first and second order tensors we know many calculation rules, so rewriting higher order tensors as tensors of order 1 (*vectorization*) or 2 (*unfolding/matricization*) increases the number of tools we can use.

Definition 2.7.3 (Vectorization)**.** ***Vectorization*** *is the reordering of an Nth-order tensor into a vector. The order of vectorization is not unique, but has to be consistent throughout calculations.*

Definition 2.7.4 (mode-n Unfolding)**.** ***Mode-n unfolding*** *or* ***mode-n matricization*** *is the reordering of an Nth-order tensor into a matrix, where the columns are defined by the mth mode. The order of reordering is not unique, but has to be consistant throughout calculations.*
The mode-n unfolding of tensor $\mathcal{A}$ is denoted as $\mathcal{A}_{(n)}$

If we consider an example $\mathcal{A} \in \mathbb{R}^{2\times4\times3}$, we can vectorize this as:

$$\operatorname{vec}(\mathcal{A}) = \begin{pmatrix} a_{1,1,1} \\ a_{1,1,2} \\ a_{1,1,3} \\ a_{1,2,1} \\ \vdots \\ a_{1,4,3} \\ a_{2,1,1} \\ a_{2,1,2} \\ \vdots \\ a_{2,4,3} \end{pmatrix},$$

where we iterate over the third mode first. Then over the second mode and finally over the first mode.

If we consider unfolding of the same matrix, we find that the newly found matrix could be:

$$\mathcal{A}_{(1)} = \begin{pmatrix} a_{1,1,1} & a_{1,1,2} & a_{1,1,3} & a_{1,2,1} & a_{1,2,2} & a_{1,2,3} & a_{1,3,1} & a_{1,3,2} & a_{1,3,3} & a_{1,4,1} & a_{1,4,2} & a_{1,4,3} \\ a_{2,1,1} & a_{2,1,2} & a_{2,1,3} & a_{2,2,1} & a_{2,2,2} & a_{2,2,3} & a_{2,3,1} & a_{2,3,2} & a_{2,3,3} & a_{2,4,1} & a_{2,4,2} & a_{2,4,3} \end{pmatrix}$$

$$\mathcal{A}_{(2)} = \begin{pmatrix} a_{1,1,1} & a_{1,1,2} & a_{1,1,3} & a_{2,1,1} & a_{2,1,2} & a_{2,1,3} \\ a_{1,2,1} & a_{1,2,2} & a_{1,2,3} & a_{2,2,1} & a_{2,2,2} & a_{2,2,3} \\ a_{1,3,1} & a_{1,3,2} & a_{1,3,3} & a_{2,3,1} & a_{2,3,2} & a_{2,3,3} \\ a_{1,4,1} & a_{1,4,2} & a_{1,4,3} & a_{2,4,1} & a_{2,4,2} & a_{2,4,3} \end{pmatrix}$$

$$\mathcal{A}_{(3)} = \begin{pmatrix} a_{1,1,1} & a_{1,2,1} & a_{1,3,1} & a_{1,1,1} & a_{2,1,1} & a_{2,2,1} & a_{2,3,1} & a_{2,4,1} \\ a_{1,1,2} & a_{1,2,2} & a_{1,3,2} & a_{1,1,2} & a_{2,1,2} & a_{2,2,2} & a_{2,3,2} & a_{2,4,2} \\ a_{1,1,3} & a_{1,2,3} & a_{1,3,3} & a_{1,1,3} & a_{2,1,3} & a_{2,2,3} & a_{2,3,3} & a_{2,4,3} \end{pmatrix},$$

where in each we again iterate over the last non-fixed mode first. So for mode-2 unfolding, we first iterate over the third mode and then the first mode. Note that we could have chosen to iterate over the first mode first and the third mode last.

In order to make full use of this vectorized and unfolded form, we also introduce special matrix multiplications.

Definition 2.7.5 (Kronecker Product). *The Kronecker product of matrices $A \in \mathbb{R}^{I \times J}$ and $B \in \mathbb{R}^{M \times N}$ is denoted as $A \otimes B \in \mathbb{R}^{IM \times JN}$ and is defined as*

$$A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \cdots & a_{1,N} \\ a_{2,1}B & \ddots & & a_{2,N} \\ \vdots & & \ddots & \vdots \\ a_{M,1}B & a_{M,2} & \cdots & a_{M,N} \end{pmatrix}.$$

Definition 2.7.6 (Khatri-Rao product). *The **Khatri-Rao product** can be seen as a column-wise Kronecker product. For matrices $A = \begin{pmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_N \end{pmatrix} \in \mathbb{R}^{K \times N}, B = \begin{pmatrix} \mathbf{b}_1 & \cdots & \mathbf{b}_N \end{pmatrix} \in \mathbb{R}^{M \times N}$ we define the Khatri-Rao product, denoted as $A \odot B \in \mathbb{R}^{KM \times N}$, as*

$$A \odot B = \begin{pmatrix} \mathbf{a}_1 \mathbf{b}_1 & \cdots & \mathbf{a}_N \mathbf{b}_N \end{pmatrix}$$

Definition 2.7.7 (Hadamard product). *The **Hadamard product** is the element-wise matrix multiplication. Let $A, B \in \mathbb{R}^{M \times N}$. Denoted by $A \circledast B \in \mathbb{R}^{M \times N}$, the Hadamard product is defined as*

$$A \circledast B = \begin{pmatrix} a_{1,1}b_{1,1} & a_{1,2}b_{1,2} & \cdots & a_{1,N}b_{1,N} \\ a_{2,1}b_{2,1} & \ddots & & a_{2,N}b_{2,N} \\ \vdots & & \ddots & \vdots \\ a_{M,1}b_{M,1} & a_{M,2}b_{M,2} & \cdots & a_{M,N}b_{M,N} \end{pmatrix}$$

The final definition we introduce is about the Frobenius norm of a tensor. The definition is analogous to that of the Frobenius norm for matrices. The Frobenius norm is often used for optimization problems with tensor decomposition, since the objective function is often described as a minimization of the Frobenius norm.

Definition 2.7.8 (Frobenius norm). *For a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ its Frobenius norm, also called F-norm, is defined as the square root of the sum of all entries squared*

$$\|\mathcal{A}\|_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} \left(\mathcal{A}[i_1, i_2, \cdots, i_N]\right)^2}$$

2.8. Canonical Polyadic Decomposition (CPD)

This section explores the basic concept of Canonical Polyadic Decomposition (CPD) and how CPD can be applied to the Fourier-cosine/Fourier-sine series expansions introduced in Equations (2.3a) and (2.3b).

CPD is a tensor decomposition technique. Calculations involving higher order tensors typically result in large computational complexities. Tensor decomposition techniques aim to reduce this computational complexity. Using CPD we can represent high-dimensional tensors as a product of multiple lower-dimensional tensors, making computations more efficient.

Rank-one tensors play a critical role in CPD methods. An Nth-order tensor $\mathcal{A}$ is called a *rank-one* tensor if it can be written as an outer product of N vectors

$$\mathcal{A} = \mathop{\circ}_{n=1}^{N} \mathbf{a}_n.$$

CPD is the factorization of an Nth-order tensor as a sum of component rank-one tensors.

Definition 2.8.1 (Tensor rank)**.** *Let $\mathcal{A}$ be an Nth-order tensor.* ***Rank*** *R is the minimum number of components for which*

$$\mathcal{A} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]_R = \sum_{r=1}^{R} \mathop{\circ}_{n=1}^{N} \mathbf{A}_n[r]$$

holds, where $\mathbf{A}_n = \left[\mathbf{a}_1^n, \ldots, \mathbf{a}_R^n\right] \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$. The minimal rank is commonly called the ***canonical rank****.*

Every tensor admits a CPD of finite rank and is "unique under mild conditions [6, p. 1]". As stated by [7], however, finding this canonical rank is an NP-hard problem. Therefore, there is no easily computable charactarization of the rank. In practice, a lower rank $\tilde{R}$ is often used to approximate the original tensor. Then

$$\mathcal{A} \approx \sum_{r=1}^{\tilde{R}} \mathop{\circ}_{n=1}^{N} \mathbf{A}_n[r]. \tag{2.11}$$

This lower-rank CPD can be used to break the curse of dimensionality.

To find the CPD of a tensor, the matrices $\mathbf{A}_n$ must be determined. This is done by minimizing the least square error (using the Frobenius norm) between the original tensor and the CPD approximation. Then, we aim to solve

$$\min_{\{\mathbf{A}_n\}_{n=1}^{N}} \left\| \mathcal{A} - [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]_R \right\|_F^2 = \min_{\{\mathbf{A}_n\}_{n=1}^{N}} \left\| \mathcal{A} - \sum_{r=1}^{R} \mathop{\circ}_{n=1}^{N} \mathbf{A}_n[r] \right\|_F^2.$$

This optimization problem appears difficult to solve, but its complexity can be reduced significantly through some rewriting. The method we use to solve this is called Alternating Least Squares (ALS). The main idea of ALS is fixing all variables except for one, and then, by variating the one left over, you can cycle over each variable until some stopping criterion is met.

First we provide an equation that relates unfolding to the Khatri-Rao product [6].

$$\mathcal{A}_{(n)} = \mathbf{A}_n \left(\bigodot_{i \neq n} \mathbf{A}_i \right)^T, \qquad n = 1, \ldots, N, \tag{2.12}$$

where

$$\left(\bigodot_{i \neq n} \mathbf{A}_i \right) = \mathbf{A}_N \odot \cdots \odot \mathbf{A}_{n+1} \odot \mathbf{A}_{n-1} \odot \cdots \odot \mathbf{A}_1.$$

The order of unfolding is not unique, but for this equation to hold, we require a specific order. Appendix A.3.1 shows that one should first iterate over $\mathbf{A}_1$, then $\mathbf{A}_2$, and $\mathbf{A}_N$ last.

With this new formulation, we can rewrite our minimization problem to N minimizations for a single tensor mode.

$$\min_{\{\mathbf{A}_n\}_{n=1}^{N}} \left\| \mathcal{A}_{(n)} - \mathbf{A}_n \left(\bigodot_{i \neq n} \mathbf{A}_i \right)^T \right\|_F^2, \qquad n = 1, \ldots, N.$$

This is a multilinear problem, but through ALS we update the factor matrices $\mathbf{A}_n$ one by one while fixing all other matrices $\mathbf{A}_i$. We loop over the matrices $\mathbf{A}_n$ and solve

$$\mathbf{A}_n^{new} = \underset{\{\mathbf{A}_n\}_{n=1}^N}{\arg\min} \left\| \mathcal{A}_{(n)} - \mathbf{A}_n \left(\bigodot_{i \neq n} \mathbf{A}_i \right)^T \right\|_F^2$$

until some stopping criterion is met.

2.8.1. Dimension reduced Fourier-cosine series expansions via CPD

This section is based on the works of a previous thesis performed at FF Quant [2], [4]. They combine CPD with the Fourier-cosine series expansion from [5]. We follow the same steps, but apply it to the Fourier-sine series expansion described in Section 2.4. For a one dimensional function $f(x)$ on domain $[a, b]$, this sine expansion will be given by

$$\begin{aligned} f(x) &= \sum_{k=1}^{\infty} A_k \sin\left(k\pi \frac{x-a}{b-a}\right) \\ A_k &= \frac{2}{b-a} \int_a^b f(y) \sin\left(k\pi \frac{y-a}{b-a}\right) dy. \end{aligned}$$

The approximation would then be

$$f(x) \approx \sum_{k=1}^{K} A_k \sin\left(k\pi \frac{x-a}{b-a}\right),$$

where coefficients A_k are the entries of a first order tensor (vector) of K elements.
For an N-dimensional expansion on the domain $[a_1, b_1] \times \cdots \times [a_N, b_N]$, we have

$$\begin{aligned} f(x_1, x_2, \ldots, x_N) &= \sum_{k_1=1}^{\infty} \cdots \sum_{k_N=1}^{\infty} \mathcal{A}_{\mathbf{k}} \prod_{n=1}^{N} \sin\left(k_n \pi \frac{x_n - a_n}{b_n - a_n}\right) \\ f(x_1, x_2, \ldots, x_N) &\approx \sum_{k_1=1}^{K} \cdots \sum_{k_N=1}^{K} \mathcal{A}_{\mathbf{k}} \prod_{n=1}^{N} \sin\left(k_n \pi \frac{x_n - a_n}{b_n - a_n}\right) \\ A_{\mathbf{k}} &= \int_{a_1}^{b_1} \cdots \int_{a_N}^{b_N} f(y_1, y_2, \ldots, y_N) \prod_{n=1}^{N} \sin\left(k_n \pi \frac{x_n - a_n}{b_n - a_n}\right) dy_N \cdots dy_1 \prod_{n=1}^{N} \frac{2}{b_n - a_n}, \end{aligned} \tag{2.13}$$

where $\mathbf{k} = (k_1, k_2, \ldots, k_N)$ is a multi-index. Now the coefficients $\mathcal{A}_{\mathbf{k}}$ are the entries of an Nth-order tensor $\mathcal{A}$ with K elements in each mode. Note that this tensor has K^N entries, so it scales with $O(N^K)$. If we apply tensor decomposition to this tensor, we find

$$\begin{aligned} \mathcal{A} &\approx \sum_{r=1}^{R} \overset{N}{\underset{n=1}{\circ}} \mathbf{A}_n[r], \\ \mathcal{A}_{\mathbf{k}} &\approx \sum_{r=1}^{R} \prod_{n=1}^{N} \mathbf{A}_n[k_n, r], \end{aligned}$$

where $\mathcal{A}$ is a sum of R tensors created by taking the outer product of N vectors of size K, so $\mathcal{O}(NKR)$. Substituting this approximation for $\mathcal{A}_\mathbf{k}$ into Equation 2.13 yields

$$\begin{aligned}
f(x_1, x_2, \dots, x_N) &\approx \sum_{k_1=1}^{K} \cdots \sum_{k_N=1}^{K} \left[\left(\sum_{r=1}^{R} \prod_{n=1}^{N} \mathbf{A}_n[k_n, r] \right) \cdot \left(\prod_{n=1}^{N} \sin\left(k_n \pi \frac{x_n - a_n}{b_n - a_n} \right) \right) \right] \\
&= \sum_{k_1=1}^{K} \cdots \sum_{k_N=1}^{K} \sum_{r=1}^{R} \prod_{n=1}^{N} \left(\mathbf{A}_n[k_n, r] \sin\left(k_n \pi \frac{x_n - a_n}{b_n - a_n} \right) \right) \\
&= \sum_{r=1}^{R} \sum_{k_1=1}^{K} \cdots \sum_{k_N=1}^{K} \prod_{n=1}^{N} \mathbf{A}_n[k_n, r] \mathbf{s}_n[k_n](x_n), , \qquad (2.14)
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{s}_n(x_n) &= \left(\sin\left(1\pi \tfrac{x_n - a_n}{b_n - a_n} \right) \quad \cdots \quad \sin\left(K\pi \tfrac{x_n - a_n}{b_n - a_n} \right) \right)^T \in \mathbb{R}^K \\
\mathbf{s}_n[k_n](x_n) &= \sin\left(k_n \pi \frac{x_n - a_n}{b_n - a_n} \right) \\
\text{i.e.} \quad \mathbf{s}_2[3](4) &= \sin\left(3\pi \frac{4 - a_2}{b_2 - a_2} \right).
\end{aligned}$$

Note that

$$\begin{aligned}
\sum_{k_N=1}^{K} \prod_{n=1}^{N} \mathbf{A}_n[k_n, r] \mathbf{s}_n[k_n](x_n) &= \sum_{k_N=1}^{K} \left(\mathbf{A}_N[k_N, r] \mathbf{s}_N[k_N](x_N) \right) \prod_{n=1}^{N-1} \mathbf{A}_n[k_n, r] \mathbf{s}_n[k_n](x_n) \\
&= \left(\prod_{n=1}^{N-1} \mathbf{A}_n[k_n, r] \mathbf{s}_n[k_n](x_n) \right) \sum_{k_N=1}^{K} \left(\mathbf{A}_N[k_N, r] \mathbf{s}_N[k_N](x_N) \right),
\end{aligned}$$

because the product up to $N-1$ is no longer dependent on k_N. By rewriting

$$\begin{aligned}
f_{n,r}(x_n) &= \sum_{k_n=1}^{K} \left(\mathbf{A}_n[k_n, r] \mathbf{s}_n[k_n](x_n) \right) \\
&= \mathbf{s}_n^T(x_n) \mathbf{A}_n[:, r],
\end{aligned}$$

we can iterate over each sum of Equation 2.14 and find a new expression for the approximation of f:

$$\begin{aligned}
f(x_1, x_2, \dots, x_N) &= \sum_{r=1}^{R} \prod_{n=1}^{N} f_{n,r}(x_n) \qquad (2.15) \\
&= \left(\circledast_{n=1}^{N} \mathbf{s}_n^T(x_n) \mathbf{A}_n \right) \mathbf{1}, \qquad (2.16)
\end{aligned}$$

where $\mathbf{1} = (1, \dots, 1)^T \in \mathbb{R}^R$. With this we have now rewritten the original approximation with an Nth-order tensor $\mathcal{A}$ of unknown coefficients to a problem with N factor matrices of size $K \times R$ that hold unknown coefficients.

2.9. Other theorems

Theorem 2.9.1 (Fundamental Theorem of Calculus (FTC))**.** *Let $f : [a, b] \to \mathbb{R}$ be continuous. Let $F : [a, b] \to \mathbb{R}$ be defined as*

$$F(x) = \int_a^x f(y) dy.$$

Then $F(x)$ is uniformly continuous on $[a, b]$ and differentiable on (a, b), and

$$F'(x) = f(x),$$

for all $x \in (a, b)$. F is the antiderivative of f.

Corollary 2.9.1.1. *Let $f : [a, b] \to \mathbb{R}$ be continuous and let F be the antiderivative of f on $[a, b]$. Then*

$$\int_a^b f(x)dx = F(b) - F(a).$$

Theorem 2.9.2 (Lebesgue's Dominated Convergence Theorem (DCT))**.** *Let $(f_n)_{n\geq 1}$ be a sequence of measurable functions on a measure space (S, Σ, μ). Suppose that the sequence converges pointwise to a function f, i.e.*

$$\lim_{n\to\infty} f_n(x) = f(x)$$

exists for all $x \in S$. Assume that sequence f_n is dominated by some integrable function g in the sence that

$$|f_n(x)| \leq g(x)$$

for all points $x \in S$ and all n. Then f_n and f are Lebesgue integrable and

$$\lim_{n\to\infty} \int_S f_n d\mu = \int_S \lim_{n\to\infty} f_n d\mu = \int_S f d\mu,$$

and we have

$$\lim_{n\to\infty} \int_S |f_n - f| d\mu = 0$$

Remark 2.9.2.1. *Pointwise convergence of the sequence can be relaxed to hold only μ-almost everywhere. So the set of points where pointwise convergence does not hold should be a measurable set Z with $\mu(Z) = 0$*

3
Artificial Neural Networks

Partial Differential Equations (PDEs) are extensively studied in mathematics due to their relevance in multiple fields such as physics and finance. Solving these PDEs is rarely a trivial task. Traditional numerical methods for solving PDEs include the Finite Difference Method (FDM), Finite Elements Method (FEM), Finite Volume Method (FVM), and spectral methods like Fourier series expansion.

These methods, however, generally suffer heavily from the curse of dimensionality, where the required memory and computational time grows exponentially with respect to the number of dimensions. Spectral methods have additional drawbacks like Gibbs phenomenon at discontinuities or sharp gradients.

In recent decades, machine learning has been investigated to address the weaknesses of these traditional methods. The machine learning methods that we discuss in this thesis belong to the class of *Artificial Neural Networks* (ANNs). ANNs were initially inspired by the biological neural networks structure in brains. In this thesis, "neural networks" refers to artificial neural networks.

An ANN is a group of nodes called *neurons*, connected by *edges*, that represent the synapses (signals) between neurons in a brain. Neurons receive signals from connected neurons. The architecture for these networks depends on the chosen model.

This chapter explores several subclasses of neural networks. We first consider two types of Multilayer Perceptrons (MLPs): Physics-informed neural networks (PINNs) and Fourier neural networks (FNNs). Then, we present Deep Operator Networks (DeepONets) and Kolmogorov-Arnold networks (KANs). Lastly, we introduce a neural network representation of the dimension-reduced COS Method from [5], which we will name the "COS-CPD" network, respecting that it is purely based on mathematical derivation using CPD and COS. The COS-CPD network is used in this thesis.

3.1. Multilayer Perceptrons

The first well-known subclass of ANNs is Multilayer Perceptrons (MLPs) [35]. An MLP assigns a scalar weight to each edge, representing the 'strength' of the signal. Within a neuron, non-linear *activation functions* process input signals and send out other signals to the next neurons. An MLP consists of layers of neurons. The first layer is called the *input layer* and its neurons have no incoming edges, whereas the final layer, the *output layer*, has neurons with exclusively incoming edges and no outgoing edges. In between there are n *hidden layers* of neurons. If $n > 1$, the network is called a *deep neural network*. Figure 3.1 shows the general network architecture for an MLP, where all neurons of layer i are connected to all neurons of layer $i + 1$.

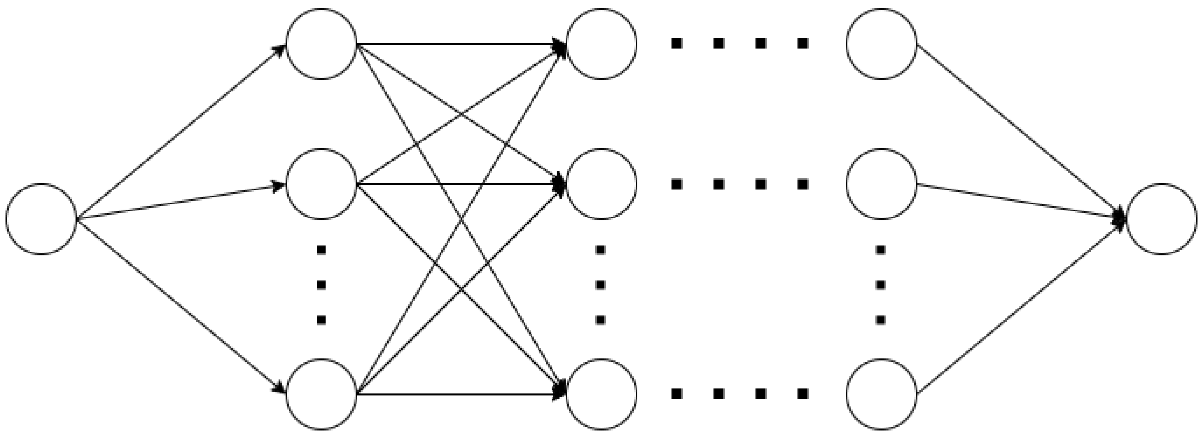


Figure 3.1: Network architecture for general MLPs

MLPs are *feed forward networks*, meaning that there are no loops inside the neural network. Therefore, training must be done using some structure outside the architecture seen in Figure 3.1. To train the model, data pairs of input and output values are required. The output of the neural network is then compared to the known output value through a *loss function*. Commonly, methods such as backpropagation are used to update the weights of the edges to minimize the loss function.

The choice of loss function, activation function and training process differ among types of MLPs. We discuss two of these types.

3.1.1. Physics-Informed Neural Networks

Physics-informed neural networks (PINNs), introduced in [36], are neural networks specifically trained to respect physical laws described by nonlinear PDEs. In [36], the activation function is chosen to be the hyperbolic tangent. The PDE is incorporated into their choice of the loss function. Given a nonlinear PDE of the form

$$f = \frac{\partial u}{\partial t} + \mathcal{N}[u;\lambda] = 0,$$

where $\mathcal{N}$ denotes a nonlinear operator parametrized by λ, the loss function is defined as

$$\begin{aligned} MSE &= MSE_u + MSE_f \\ MSE_u &= \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2 \\ MSE_f &= \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2, \end{aligned}$$

where $\{t_u^i, x_u^i\}$ are the initial and boundary training points on $u(t,x)$ and $\{t_f^i, x_f^i\}$ are collocation points for $f(t,x)$.

The PINN is trained by minimizing this loss function through updating the parameters of $u(t,x)$.

3.1.2. Fourier Neural Networks

Fourier Neural Networks (FNNs) were introduced in 2013 by Liu S. [37]. The FNN model is, as the name suggests, based on Fourier series expansion. For a network with N neurons, the approximation of a function $u \in \mathbb{R}^n \to \mathbb{R}$ would look like

$$\hat{u}(x_1, \ldots, x_n) = \sum_{i=1}^{N} \lambda_{2i}\Phi_{2i} + \lambda_{2i+1}\Phi_{2i+1},$$

where

$$\begin{aligned} \Phi_{2k} &= \sin(w_{2k,1}x_1 + \cdots + w_{2k,n}x_n) \\ \Phi_{2k+1} &= \cos(w_{2k+1,1}x_1 + \cdots + w_{2k+1,n}x_n) \end{aligned}$$

are the sine and cosine terms. The λ_k and w_k are weights. This means that an FNN with N neurons has $4N$ weights to optimize. We should note that, while based on Fourier series expansion, the approximation itself is actually not a multi-dimensional Fourier series expansion.

The machine learning structure can be seen as a network with an input layer, one hidden layer and an output layer. The input layer consists of input variables x_n and the output layer is the function approximation $\hat{u}(\mathbf{x})$. In the hidden layer, the activation function $x \mapsto \sin(\omega_1 x) + \cos(\omega_2 x)$ is applied. Figure 3.2 depicts the scheme for a FNN with a one-dimensional input x.

The weights are updated through backpropagation until a sufficiently accurate approximation $\hat{u}$. This requires a loss function that efficiently captures the accuracy of $\hat{u}(x)$. Commonly, this loss function is a combination of a least squares estimation $||\hat{u} - u||_2^2$ and regularization terms $a_1||\lambda||^2$ and $a_2||w||^2$. Through numerical studies [38] found that L^2-norm regularization seems to yield convergence to the actual solution. While L^1-norm has quicker convergence, it does not always converge to the true solution. From this numerical studies, L^2-norm seems like the better option.

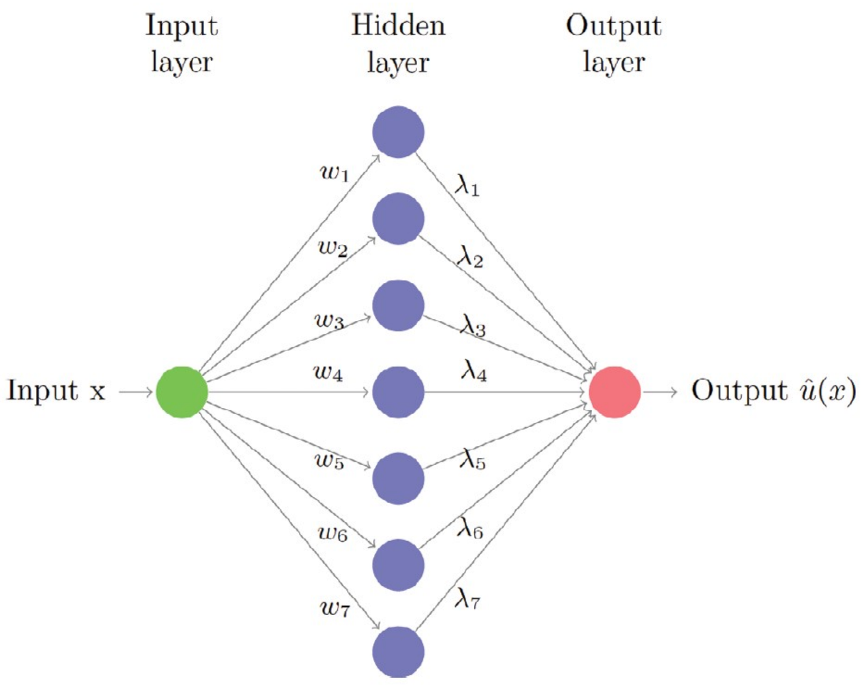


Figure 3.2: FNN scheme with one-dimensional input x. [38, p. 3]

[38] extends the regular FNN structure to compute periodic solutions of differential equations of the form $Pu = f$, where P is the differential operator. The loss function is set to be the residual of the differential equation, so $\mathcal{L} = ||P\hat{u} - f||_2^2$. They also include regularization terms $a_1||\lambda||_2^2$ and $a_2||w||_2^2$ in the loss function. To ensure periodicity of the solution, periodicity requirement terms $a_3||\hat{u}(x+T) - \hat{u}(x)||_2^2$ and $a_4||\hat{u}(x-T) - \hat{u}(x)||_2^2$ are included in the loss function. This gives rise to the combined loss function

$$L(\lambda, w) = ||P\hat{u} - f||_2^2 + a_1||\lambda||_2^2 + a_2||w||_2^2 + a_3||\hat{u}(x+T) - \hat{u}(x)||_2^2 + a_4||\hat{u}(x-T) - \hat{u}(x)||_2^2$$

3.2. DeepONets

Deep Operator Networks (DeepONets), introduced in [39], take a very different approach compared to MLPs. Instead of learning how a function output depends on its input, they learn the general structure of the function based on input functions. Let us consider a PDE

$$Pu = f,$$

where P is the differential operator, u the solution function and f some other function. DeepONets are networks that represent the operator G, which takes function f as the input and outputs solution vector u. Operator G is thus a function of functions, that maps one function space to another.

Training this network is not a trivial task. One needs to split the network into two parts. The first part focuses on the input functions, and the second part considers the input for function u.

For the first part, named the *trunk network*, the continuous input functions must be sampled at discrete points. The most straightforward way to do this is by evaluating these functions in sufficiently, but also finitely many points $\{x_1, \ldots, x_m\}$. These points are called *sensors*. The inputs $\{u(x_1), \ldots, u(x_m)\}$ are translated to outputs $\{t_1, \ldots, t_p\}$.

The second part is the *branch network*, which encodes the location of the output function. If we want to know the value $u(y)$, this input y is fed into the branch network. The outputs of the branch network are called $\{b_1, \ldots, b_p\}$.

Together, these trunk and branch networks form an approximation

$$G(u)(y) \approx \sum_{k=1}^{p} t_k b_k.$$

3.3. Kolmogorov-Arnold Networks

Kolmogorov-Arnold Networks (KANs) are a novel neural network approach introduced very recently in [40]. KANs have a similar structure to MLPs, but differ in the location of activation functions. MLPs have scalar weights on the edges and have fixed activation functions in the neurons that combine the inputs. KANs, however, have variable splines, which act as activation functions, on the edges and the neurons simply sum all the incoming edges. KANs are based on the equation

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right),$$

where $f : [0,1]^n \to \mathbb{R}$ is a smooth function and $\Phi_q : \mathbb{R} \to \mathbb{R}, \phi_{q,p} : [0,1] \to \mathbb{R}$ are the splines. This equation represents a one layer KAN. With this formulation, the general KAN architecture can be derived for multiple layers. Figure 3.3a visualises the architecture of a one-layer KAN. The edges between the input neuron p and hidden layer neuron q represent splines $\phi_{p,q}$. Inside the hidden layer neurons, the inner sum is computed. The edges from hidden layer neuron q to the output neuron describe spline Φ_q. Thus, one layer of KAN can be described by a matrix of 1D functions

$$\mathbf{\Phi} = \{\phi_{q,p}\}, \qquad p \in \{1, 2, \ldots, n_{in}\}, \quad q = \{1, 2, \ldots, n_{out}\}.$$

A KAN with multiple layer can be described by stacking these layers. For clearer notation, we write the i-th neuron in layer l as $x_{l,i}$, there are n_l neurons in layer l, and the outgoing edge from $x_{l,i}$ to $x_{l+1,j}$ is $\phi_{l,j,i}(x_{l,i})$. Then

$$x_{l+1,j} = \sum_{i=1}^{n_l} \phi_{l,j,i}(x_{l,i})$$

and the matrix between layers l and $l+1$ can be described as

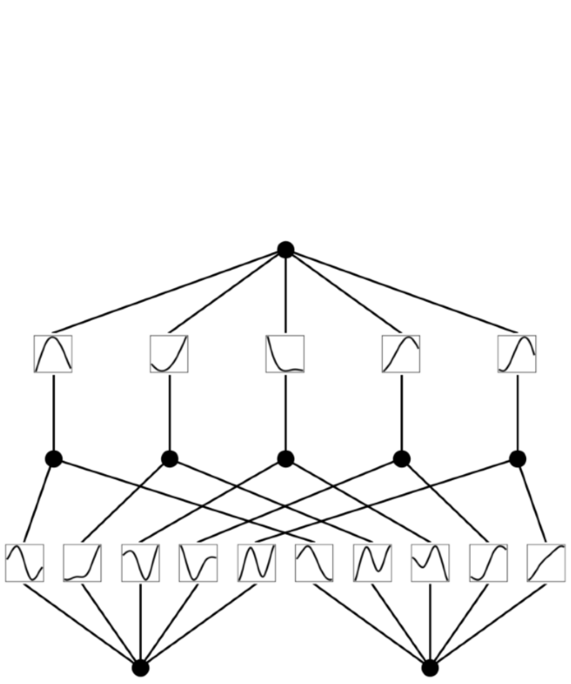
$$\mathbf{x}_{l+1} = \mathbf{\Phi}_l \mathbf{x}_l \tag{3.1}$$

$$\begin{pmatrix} x_{l+1,1} \\ x_{l+1,2} \\ \vdots \\ x_{l+1,n_{l+1}} \end{pmatrix} = \begin{pmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \cdots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,2,n_l}(\cdot) \\ \vdots & \vdots & & \vdots \\ \phi_{l,n_{l+1},1}(\cdot) & \phi_{l,n_{l+1},i}(\cdot) & \cdots & \phi_{l,n_{l+1},n_l}(\cdot) \end{pmatrix} \begin{pmatrix} x_{l,1} \\ x_{l,2} \\ \vdots \\ x_{l,n_l} \end{pmatrix}. \tag{3.2}$$

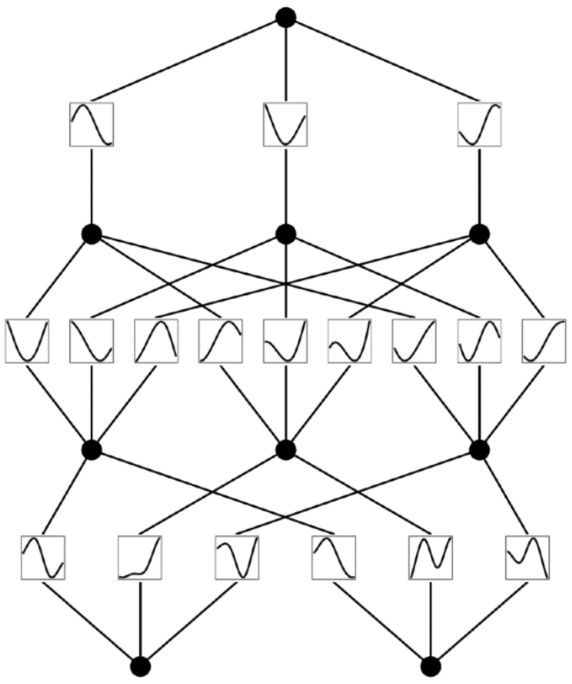
We now consider a KAN with L layers (including input and output layers). Suppose we have input $\mathbf{x}_0 \in \mathbb{R}^{n_0}$, the output will be

$$KAN(\mathbf{x}_0) = (\mathbf{\Phi}_{L-1} \circ \mathbf{\Phi}_{L-2} \circ \cdots \circ \mathbf{\Phi}_0)\mathbf{x}_0, \tag{3.3}$$

which is the composition of all layer matrices. A visualisation of a multilayer KAN is provided in Figure 3.3b.



(a) Network architecture of a Shallow KAN



(b) Network architecture of a Deep KAN

Figure 3.3: Network architectures of a Shallow and Deep KAN. [40, p. 1]

We now know how the KAN architecture works, but how can we actually use variable activation functions on the edges and how can we train them? The activation functions are a sum of a basis function part and a spline function part.

$$\phi(x) = w_b b(x) + w_s spline(x)$$

$$b(x) = \frac{x}{1+e^{-x}}.$$

The $spline(x)$ function is parametrized as a linear combination of B-splines $B_i(x)$ (B-splines are piecewise polynomial functions)

$$spline(x) = \sum_i c_i B_i(x),$$

where coefficients c_i are trainable. These B-splines are defined recursively. A B-spline has degree p. For $p = 0$, we define

$$B_{i,p}(x) = \begin{cases} 1 & t_i \le x < t_{i+1} \\ 0 & else. \end{cases}$$

For higher degree B-splines, we define

$$B_{i,p}(x) = \frac{t-t_i}{t_{i+p}-t_i} B_{i,p}(x) + \frac{t_{i+p+1}-t}{t_{i+p+1}-t_{i+1}} B_{i,p}(x).$$

Our next question is how to train these coefficients c_i. In the code provided by [40] they have done this by implementing a choice for either Adaptive Moment Estimation (ADAM) or Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS).

Lastly, it is important to note that KANs can also be used to solve PDEs. Similar to the PINNs mentioned before, KANs include the boundary conditions into the loss function.

3.4. Our contribution 1: COS-CPD network representation

COS-CPD, introduced in [2] and [4], combines the cos method from [5] and CPD for dimension reduction. In Section 2.8.1, we discussed how a function can be approximated through sine/cosine expansion with CPD. In this section, we demonstrate that the COS-CPD method corresponds to a neural network structure. For a function $f : \mathbb{R}^N \to \mathbb{R}$, we can approximate it as

$$f(x_1, x_2, \dots, x_N) \approx \left(\underset{n=1}{\overset{N}{\circledast}} \mathbf{s}_n^T(x_n)\mathbf{A}_n \right) \mathbf{1},$$

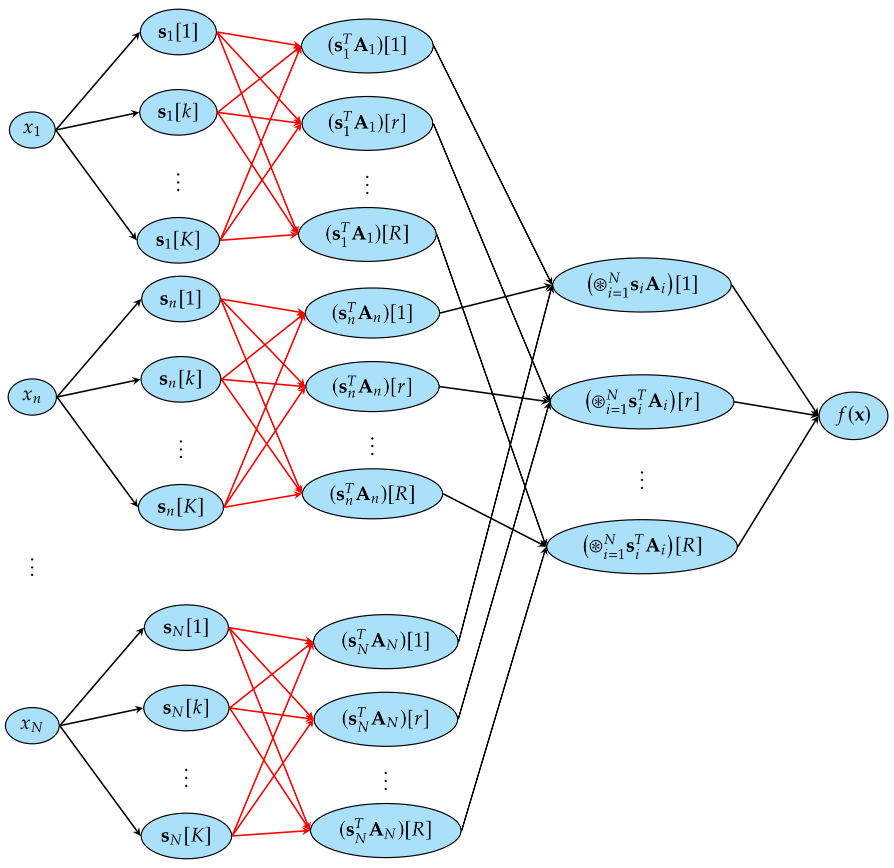
where $\mathbf{s}_n$ are the vectors with sine/cosine terms and $\mathbf{A}_n$ are the CPD factor matrices.

Figure 3.4 depicts the network architecture corresponding to the COS-CPD method. Neurons $x_1, \dots, x_N$ are the inputs for function f. Each input dimension forms its own part of the network architecture. Later these parts will join for the final function approximation. All black edges represent weight 1, so all inputs are passed to the neurons $\mathbf{s}_n[k]$ with weight 1. Inside neurons $\mathbf{s}_n[k]$ there are activation functions $\sin\left(k\pi \frac{x_n - a_n}{b_n - a_n}\right)$.

The next set of neurons are the labelled $(\mathbf{s}_n^T \mathbf{A}_n)[r]$. These are linear combinations of the previous set of neurons with the same dimension n. Since a vector matrix product is the same as taking linear combinations, the weights on the red edges are determined by the entries of factor matrices $\mathbf{A}_n$.

In the next step, the dimensions come together by multiplying the r-th element of each dimension to find neuron $(\circledast_{i=1}^N \mathbf{s}_i^T \mathbf{A}_i)[r]$, so the activation function is simply the product of inputs.

Lastly, all r elements are summed with weight 1 to get the final output $f(\mathbf{x})$. Note that the only edges with trainable weights are the red edges represented by entries of the factor matrices. All other edges have weight one and are included solely to give a clearer representation of what happens within the network architecture.



x_1
x_n
x_N
$\mathbf{s}_1[1]$
$\mathbf{s}_1[k]$
$\mathbf{s}_1[K]$
$\mathbf{s}_n[1]$
$\mathbf{s}_n[k]$
$\mathbf{s}_n[K]$
$\mathbf{s}_N[1]$
$\mathbf{s}_N[k]$
$\mathbf{s}_N[K]$
$(\mathbf{s}_1^T\mathbf{A}_1)[1]$
$(\mathbf{s}_1^T\mathbf{A}_1)[r]$
$(\mathbf{s}_1^T\mathbf{A}_1)[R]$
$(\mathbf{s}_n^T\mathbf{A}_n)[1]$
$(\mathbf{s}_n^T\mathbf{A}_n)[r]$
$(\mathbf{s}_n^T\mathbf{A}_n)[R]$
$(\mathbf{s}_N^T\mathbf{A}_N)[1]$
$(\mathbf{s}_N^T\mathbf{A}_N)[r]$
$(\mathbf{s}_N^T\mathbf{A}_N)[R]$
$(\circledast_{i=1}^N \mathbf{s}_i\mathbf{A}_i)[1]$
$(\circledast_{i=1}^N \mathbf{s}_i^T\mathbf{A}_i)[r]$
$(\circledast_{i=1}^N \mathbf{s}_i^T\mathbf{A}_i)[R]$
$f(\mathbf{x})$

Figure 3.4: Network architecture of COS-CPD

To actually train the network, initial values must be chosen for all factor matrices. Then, the loss function can be minimized through ALS.

In later chapters of this thesis, we demonstrate how to adapt COS-CPD to solve an option pricing PDE. The training procedure for this network is intuitive and straightforward. For instance, consider a partial derivative term with respect to x_n. We can take the derivative of vector $\mathbf{s}_n(x_n)$ and use it as the new activation function in the second set of neurons. All other blocks of $x_m, m \neq n$ remain unchanged. Initial and boundary conditions, similar to those in PINNs and KANs, can be incorporated into the loss function. This thesis considers barrier options, simplifying the problem since the option values on the boundaries are 0. Therefore, by choosing a sine expansion, the network inherently satisfy the boundary conditions.

Similarly, the initial condition is also inherently satisfied, because the approximation is created through expansion on the derivative with respect to time. For initial conditions with value 0, the initial condition can be seen as a left boundary on the function in the time dimension, which is satisfied for the same reasons as the asset boundary conditions.

In other situations, when the boundary and initial conditions are not zero, they must be incorporated into the loss function.

3.4.1. Error analysis

To give a first impression of the performance of the COS-CPD network, we compare it to the KAN method. [40] has worked out an example of solving the Poisson equation. They find an approximation for solution function $\hat{f}(x, y)$ for the problem

$$\begin{aligned} f_{xx} + f_{yy} &= -2\pi \sin(\pi x)\sin(\pi y) \\ f(x,-1) = f(x,1) &= f(-1,y) = f(1,y) = 0. \end{aligned}$$

The analytic solution for this problem is $f(x, y) = \sin(\pi x)\sin(\pi y)$.

Since we attain 0 as values on the boundaries, it makes sense to choose a sine expansion on both dimensions. Therefore, following Section 2.8.1, we find approximation

$$\begin{aligned} \hat{f}(x,y) &= \left(\mathbf{s}_1(x)^T \mathbf{A}_1 \circledast \mathbf{s}_2(y)^T \mathbf{A}_2\right)\mathbf{1} \\ \mathbf{s}_i(x_i) &= \left(\sin\left(1\pi \frac{x_i - a_i}{b_i - a_i}\right), \cdots, \sin\left(K\pi \frac{x_i - a_i}{b_i - a_i}\right)\right)^T \\ \mathbf{z}_i(x_i) &= -\left(\left(\frac{1\pi}{b_i - a_i}\right)^2 \sin\left(1\pi \frac{x_i - a_i}{b_i - a_i}\right), \cdots, \left(\frac{K\pi}{b_i - a_i}\right)^2 \sin\left(K\pi \frac{x_i - a_i}{b_i - a_i}\right)\right)^T, \end{aligned}$$

where $\mathbf{z}_i$ is the second partial derivative of $\mathbf{s}_i$ with respect to x_i. With this, we can find the CPD representation of the PDE and vectorized PDE. In Section 7.4 this process will be further explained. Here we just want to check the accuracy and speed of the COS-CPD method.

$$\begin{aligned} PDE(x,y) &= \mathbf{z}_1(x)^T \mathbf{A}_1 \mathbf{A}_2^T \mathbf{s}_2(y) + \mathbf{s}_1^T(x)\mathbf{A}_1\mathbf{A}_2^T \mathbf{z}_2(y) + 2\pi\sin(\pi x)\sin(\pi y) = 0 \\ VECPDE_1 &= \left(\mathbf{s}_2^T(y)\mathbf{A}_2 \otimes \mathbf{z}_1^T(x) + \mathbf{z}_2^T(y)\mathbf{A}_2 \otimes \mathbf{s}_1^T(x)\right)\mathrm{Vec}(\mathbf{A}_1) + 2\pi\sin(\pi x)\sin(\pi y) = 0 \\ VECPDE_2 &= \left(\mathbf{z}_1^T(x)\mathbf{A}_1 \otimes \mathbf{s}_2^T(y) + \mathbf{s}_1^T(x)\mathbf{A}_1 \otimes \mathbf{z}_2^T(y)\right)\mathrm{Vec}(\mathbf{A}_2) + 2\pi\sin(\pi x)\sin(\pi y) = 0. \end{aligned}$$

Through alternating least squares (ALS), we can determine values for the factor matrices $\mathbf{A}_i$. ALS is a method that separates the dimensions of the network by freezing the variable weights on edges of the dimensions not being iterated over and trains one dimension at a time. ALS will be described in further detail in Section 7.4.The results of COS-CPD and KAN for this problem are shown in Figure 3.5. For COS-CPD, $K = 21$ was chosen to match the number of points used for KAN. The time COS-CPD took for training was 0.050075 seconds, whereas KAN took a total of 49 (36 + 13) seconds for both training parts. As shown in Figure 3.5, both COS-CPD and KAN perform well, with COS-CPD achieving near machine precision. Additionally, COS-CPD has 0 error on the boundaries due to the choice of sine expansion, which is a great feature of COS-CPD.

This could also explain the performance difference between the two methods. Whereas COS-CPD only needs to train the factor matrices to satisfy the internal points, KAN's training must also satisfy the boundary points. If the errors on these boundary points are weighed more, the training will be less accurate on the internal points.

However, it is important to note that this example involves a simple two-dimensional function. Although CPD aims to mitigate the curse of dimensionality, higher dimensions will still significantly increase the computational complexity of the method in the training step. Additionally, the current example is a periodic function in C^{∞}, which exhibits great convergence behaviour for the Fourier based COS-CPD.

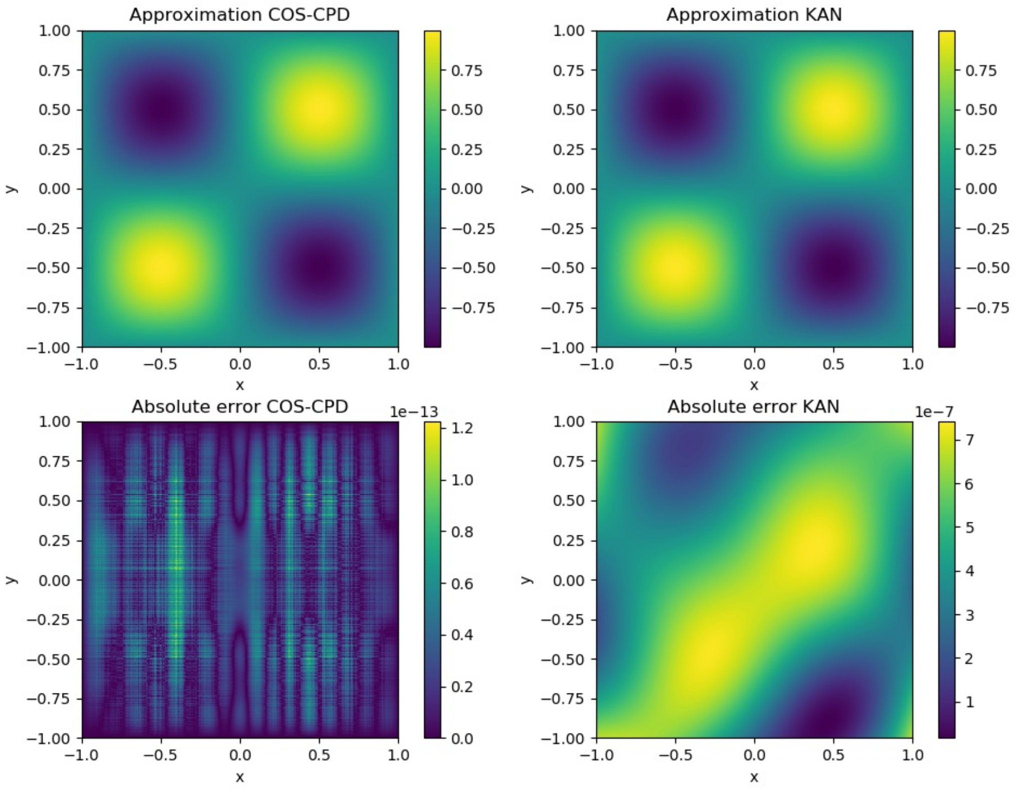


Figure 3.5: COS-CPD and KAN methods for finding a solution to Poisson equation $\nabla^2 f = -2\pi \sin(\pi x) \sin(\pi y)$

4 Convergence of the trigonometric expansion

In this chapter, we examine the convergence behaviour of the trigonometric expansion, resulting from Fourier series expansion of the first-order derivative of the unknown function compared to the convergence of the Fourier series expansion directly on the function.

First, we present numerical testing results that suggest not only a regular convergence of the trigonometric expansion to the original function, but also a faster convergence rate than Fourier series expansion in some cases. Next, we prove convergence of trigonometric expansion.

4.1. Numerical testing results

We consider two functions to demonstrate the numerical convergence of the trigonometric expansion. The first, $f(x) = (x-2)^3 + 4$, is odd around $x = 2$. The second, $g(x) = (x-2)^4 + 4$, is even around $x = 2$. We apply this expansion on the functions on the domain $[0, 4]$. Then on the boundaries we have $f'(0) = f'(4)$ and $g(0) = g(4)$.

We show the convergence and error behaviour of two types of expansions. The first type, which we will refer to as sine-based, examines the behaviour of Fourier-Sine expansion on the function itself, which we call $f_1^s(x)$. We compare this to the trigonometric expansion, $f_2^s(x)$, obtained via integrating out the Fourier-Cosine expansion on the first-order derivative of the function, and the trigonometric expansion, $f_3^s(x)$, resulting from integrating out the Fourier-Sine expansion on the second-order derivative. Then all approximations have sine functions in the expansion sums. We derive these expansions in A.4 and find

$$\begin{aligned}
f_1^s(x) &\approx \sum_{k=1}^{K} A_k \sin\left(k\pi \frac{x-a}{b-a}\right) \\
f_2^s(x) &\approx f(a) + A_0 \frac{x-a}{2} + \sum_{k=1}^{K-1} A_k \sin\left(k\pi \frac{x-a}{b-a}\right) \\
f_3^s(x) &\approx f(a) + f'(a)(x-a) + \sum_{k=1}^{K} A_k \frac{b-a}{k\pi}\left[x - a - \frac{b-a}{k\pi}\sin\left(k\pi \frac{x-a}{b-a}\right)\right].
\end{aligned}$$

The second type of expansions are cosine-based. The expansion on the function is a Fourier-cosine expansion, trigonometric expansion based on expanding the derivative is done with Fourier-sine, and trigonometric expansion based on the second derivative uses Fourier-cosine. These expansions are again derived in A.4 and look like

$$f_1^c(x) \approx \frac{A_0}{2} + \sum_{k=0}^{K-1} A_k \cos\left(k\pi \frac{x-a}{b-a}\right)$$
$$f_2^c(x) \approx f(a) + \sum_{k=1}^{K} A_k \frac{b-a}{k\pi} \left[1 - \cos\left(k\pi \frac{x-a}{b-a}\right)\right]$$
$$f_3^c(x) \approx f(a) + f'(a)(x-a) + \frac{A_0}{4}(x-a)^2 + \sum_{k=1}^{K-1} A_k \left(\frac{b-a}{k\pi}\right)^2 \left[1 - \cos\left(k\pi \frac{x-a}{b-a}\right)\right].$$

Note that in both cases, we need to know the function value at the lower boundary, $f(a)$, to derive the trigonometric expansion based on expansion on the first order derivative, and we need $f(a)$ and $f'(a)$ to build the trigonometric expansion through the second order derivative.

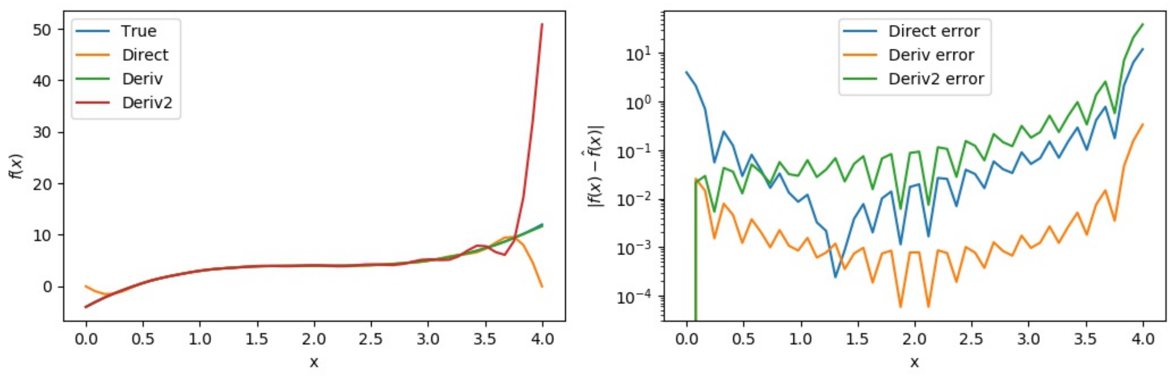
4.1.1. Error behaviour of different types of expansions

Before analysing the behaviour of the constructed expansions, we should note that all expansions are applied on the domain $[0, 4]$, resulting in Gibbs phenomenon at the boundary points $\{0, 4\}$. Note that, as detailed in [3], the Gibbs phenomenon can be greatly alleviated via enlarging the expansion interval slightly. That is, expanding on the domain $[-\varepsilon, 4+\varepsilon]$ to avoid Gibbs' phenomenon on these points. However, to fairly compare the point-wise convergence including on the boundaries, we choose to expand the function exactly on its definition interval.

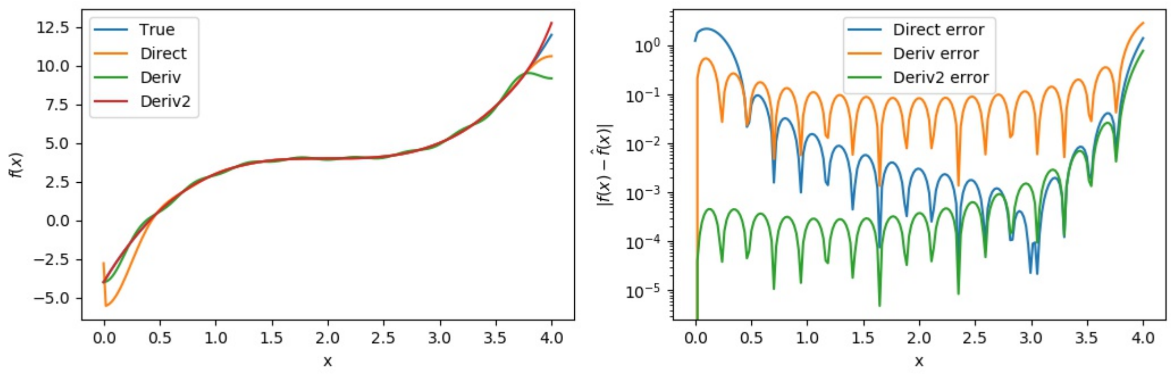
Considering how we constructed the expansions, we would expect direct expansion to suffer the most from Gibbs phenomenon on these points. Due to its construction, trigonometric expansion should perfectly match the original function at the left boundary. Trigonometric expansion based on the second order derivative should additionally match the derivative of the original function at the left boundary.

Figures 4.1 and 4.2 show that, as expected, direct expansion on the function leads to Gibbs phenomenon on both boundaries. Trigonometric expansion matches the function's value at the left boundary, and if it is based on the second derivative, it matches the derivative's value at the left boundary.

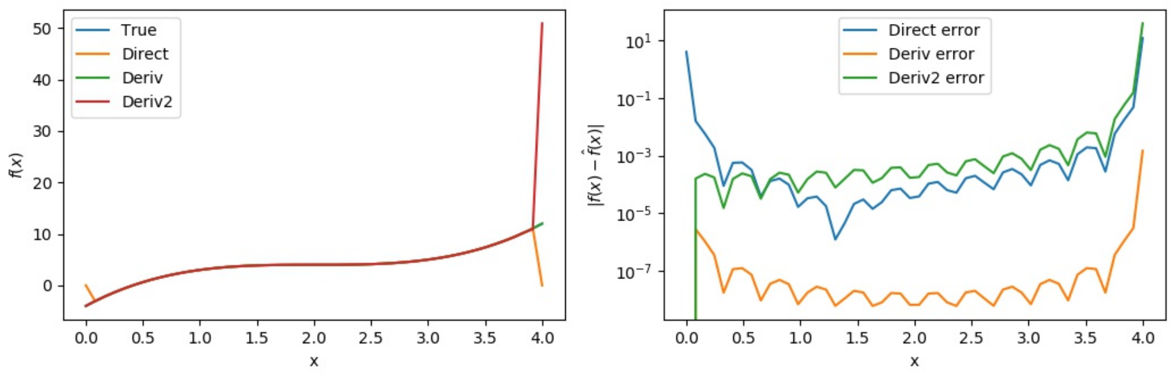
On the right boundary, all expansions struggle to match the actual function value and even more so to match the value of the derivative of the function. The sine-based trigonometric expansion, build with expansion on the derivative, seems to struggle the least with this, especially for the even function from Figure 4.2, this trigonometric expansion matches the function value at the right boundary.



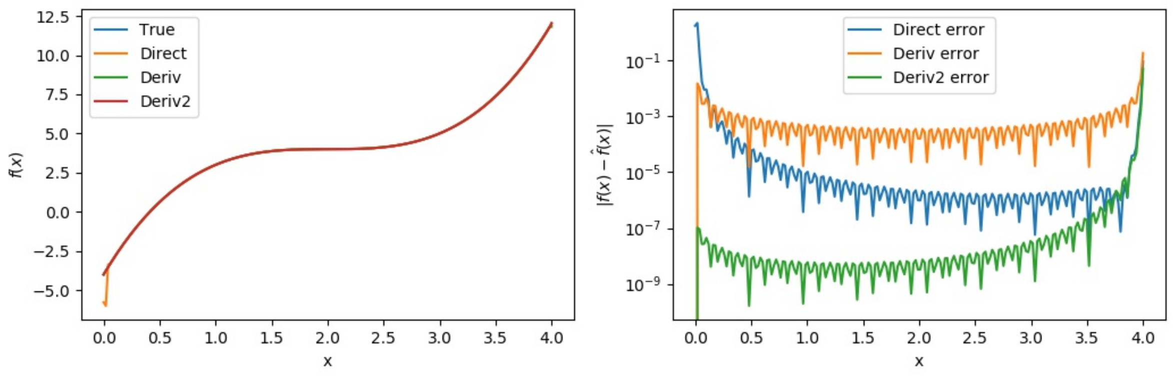
(a) sine-based expansions with $K = 16$ expansion points



(b) cosine-based expansions with $K = 16$ expansion points

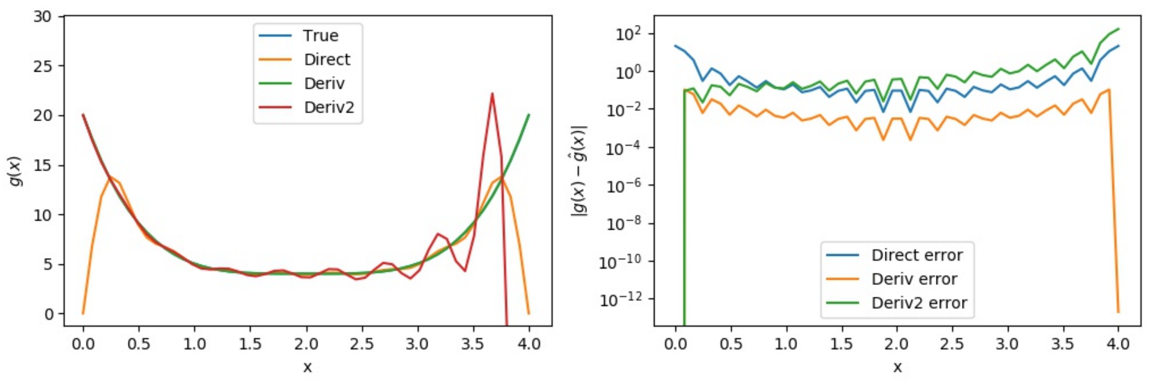


(c) sine-based expansions with $K = 256$ expansion points

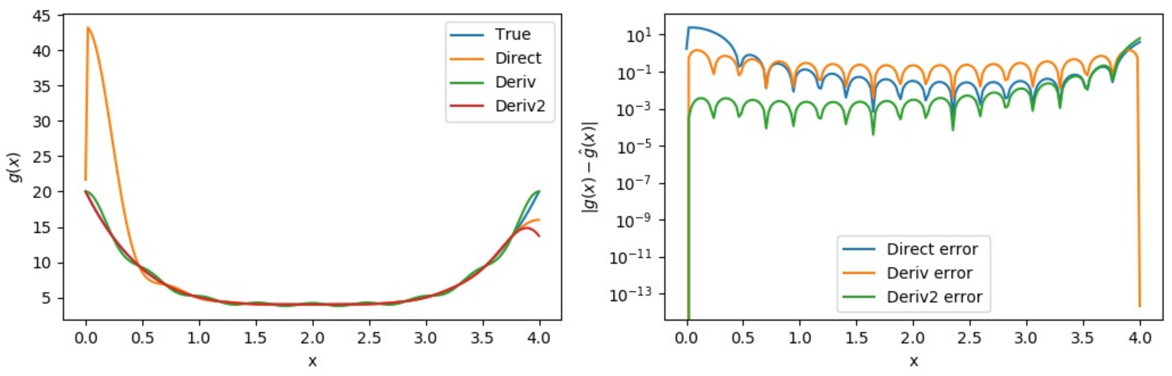


(d) cosine-based expansions with $K = 256$ expansion points

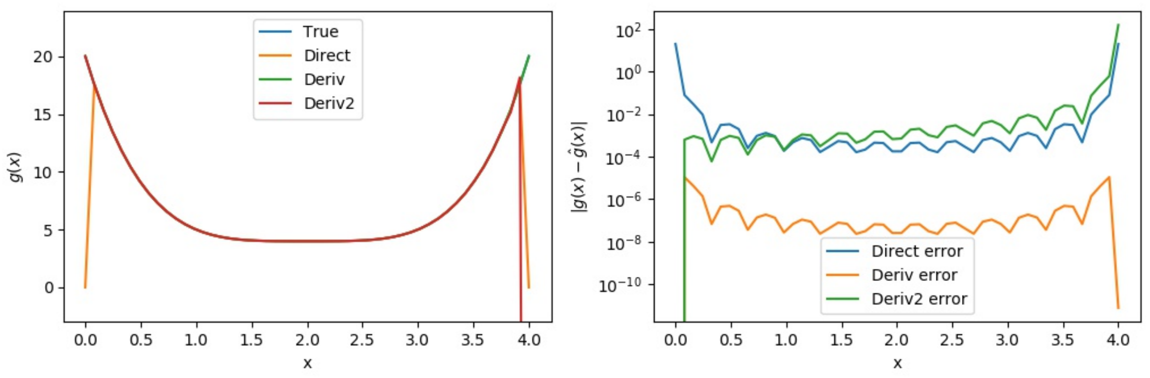
Figure 4.1: Cosine and sine-based expansions of $f(x) = (x - 2)^3 + 4$. Left plots show the function and approximations. Right plots show the errors.



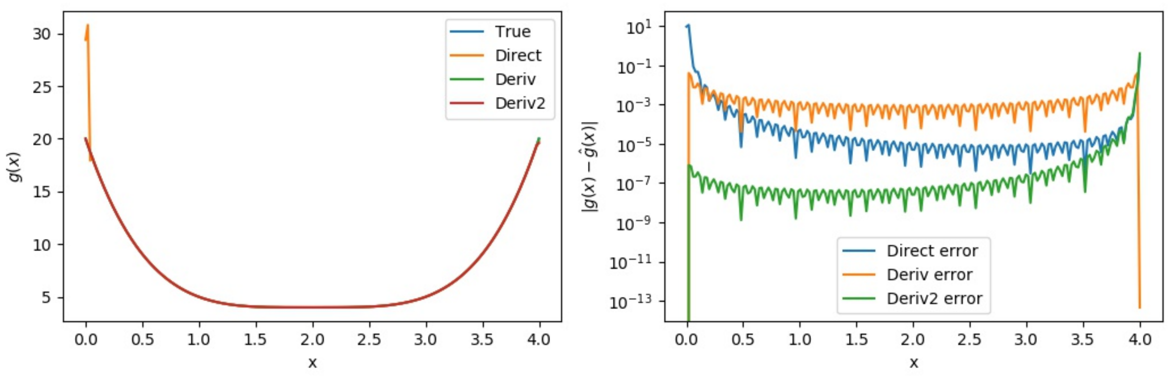
(a) sine-based expansions with $K = 16$ expansion points



(b) cosine-based expansions with $K = 16$ expansion points



(c) sine-based expansions with $K = 256$ expansion points



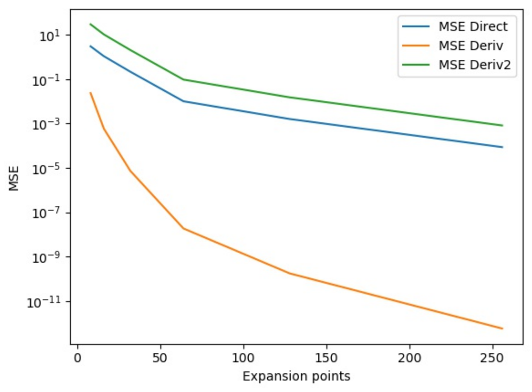
(d) cosine-based expansions with $K = 256$ expansion points

Figure 4.2: Cosine and sine-based expansions of $f(x) = (x-2)^4 + 4$. Left plots show the function and approximations. Right plots show the errors.

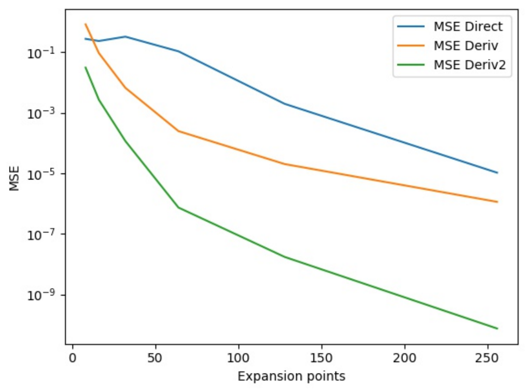
4.1.2. Error convergence of different types of expansions

Next, we study the convergence behaviour of the different expansion types. We examine the mean square error of 200 points within the domain $[0, 4]$, but exclude the points $\{0, 4\}$. At these points, the Gibbs phenomenon occurring at some methods would give an error that dominates all other errors, obscuring the actual error convergence. Figure 4.3 presents the error convergence. For sine-based expansions, trigonometric expansion, built via integrating out the Fourier series expansion of the first order derivative, seems to have a faster convergence rate, regardless of the type of function. For cosine-based expansions, this honour goes to the trigonometric expansion based on the second order derivative. However, both of these expansions do require knowledge of the true function at the left boundary.

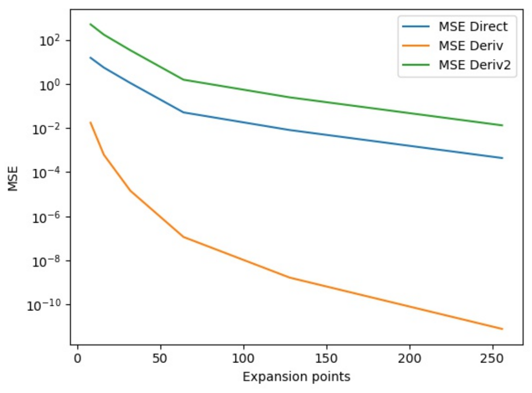
In this thesis, the initial condition is available, which is the value of the function on the left boundary in the time dimension. Therefore, based on the numerical examples from this section, sine-based trigonometric expansion, based on expansion of the derivative, would have the best accuracy.



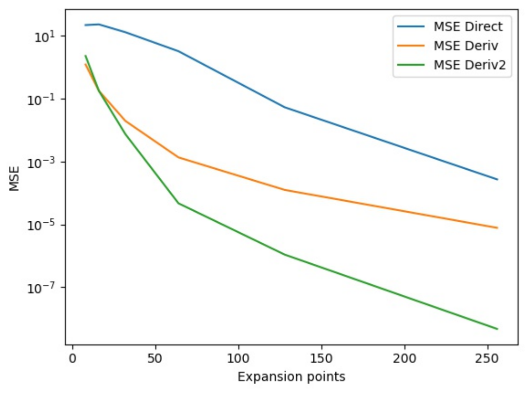
(a) Error convergence of sine-based expansions on odd functions



(b) Error convergence of cosine-based expansions on odd functions



(c) Error convergence of sine-based expansions on even functions



(d) Error convergence of cosine-based expansions on even functions

Figure 4.3: Convergence behaviour of sine-based and cosine-based expansion types

4.2. Pointwise convergence of trigonometric expansion based on derivative

In this section we prove that trigonometric expansion, derived from integrating the half range Fourier-cosine expansion on the derivative of function f (sine-based) has pointwise convergence to function f.

Let $f : [0, L] \to \mathbb{R}$ be the continuous function we wish to approximate. Then doing the half range Fourier-cosine series expansion means that we first create $g : [-L, L] \to \mathbb{R}$ as the even extension of f. Then $g(x) = g(-x)$ and $g'(-x) = -g'(x), x \in (-L, L)\backslash\{0\}$. If we were to periodically extend g to

$h : \mathbb{R} \to \mathbb{R}$, we find that $h(x) = h(-x)$ and

$$h'(-x) \begin{cases} undefined, & x = kL, k \in \mathbb{Z} \\ -h'(x), & otherwise. \end{cases}$$

We should note that h is a $2L$-periodic continuous function and h' is a $2L$-periodic piecewise continuous function with jumps on $x = kL, k \in \mathbb{Z}$. h' is a function of bounded variation and therefore we can apply the Dirichlet-Jordan test for Fourier series (Theorem 2.4.1). Therefore, the Fourier expansion $S_n h'(x)$ converges to $h'(x)$ on all x except for the points $x = kL, k \in \mathbb{Z}$. Since h' is $2L$-periodic, it is sufficient to look at points $x = 0$ and $x = L$. For these points we find

$$\begin{aligned} S_n h'(x) &= \lim_{\varepsilon \to 0} \frac{h'(x+\varepsilon) + h'(x-\varepsilon)}{2} \\ S_n h'(0) &= \lim_{\varepsilon \to 0} \frac{h'(\varepsilon) + h'(-\varepsilon)}{2} \\ &= \lim_{\varepsilon \to 0} \frac{h'(\varepsilon) - h'(\varepsilon)}{2} \\ &= 0 \\ S_n h'(L) &= \lim_{\varepsilon \to 0} \frac{h'(L+\varepsilon) + h'(L-\varepsilon)}{2} \\ &= \lim_{\varepsilon \to 0} \frac{h'(L+\varepsilon) + h'(L-\varepsilon-2L)}{2} \\ &= \lim_{\varepsilon \to 0} \frac{h'(L+\varepsilon) + h'(-(L+\varepsilon))}{2} \\ &= \lim_{\varepsilon \to 0} \frac{h'(L+\varepsilon) - h'(L+\varepsilon)}{2} \\ &= 0. \end{aligned}$$

So using the Dirichlet-Jordan test, we find that the Fourier series expansion of h' converges pointwise to

$$\lim_{n \to \infty} S_n h'(x) = \begin{cases} 0, & x = kL, k \in \mathbb{Z} \\ h'(x), & otherwise. \end{cases}$$

If we consider the domain of interest $[0, L]$, we converge pointwise to $f'(x)$ for all points except for on the boundaries, where it converges to 0.

Gibbs phenomenon around boundaries

At the boundaries we suffer from Gibbs phenomenon. Since we have pointwise convergence for all points near the boundary, we know that the domain where this Gibbs phenomenon causes inaccuracies is decreasingly small for $n \to \infty$. We should, however, also tell something about the magnitude of these errors.

The first thing we should note is that the jump at these points is of size $j = |f'(0) - f'(L)|$. From the piecewise continuity of $f'(x)$ and boundedness of f, we know that this jump size j is also finite.

It is known that the jump size approximated by the Fourier series (and therefore the Gibbs jump size inaccuracy) is of size $j * G_c$, where G_c is the Gibbs constant, equal to $1.8519\ldots$ [41], [42]. Therefore, the approximation including its errors around the boundaries can be bounded by some value $|S_n h'(x)| \leq M$ for some $M \in \mathbb{R}$.

Dominated convergence

We consider domain $[0, L]$, where we find that $\lim_{n\to\infty} S_n h'(x) = f'(x)$ for all $x \in (0, L)$. The set of points that do not converge pointwise to $f'(x)$, $\{0, L\}$, has measure $\mu(\{0, L\}) = 0$, since there are finitely many points. We also know that $f'(x)$ is bounded, so with the Gibbs phenomenon, which is just a percentage of the jump, we have a bounded functions $|S_n h'(x)| \leq M$ for some $M \in \mathbb{R}$.

From this we can apply the dominated convergence Theorem 2.9.2 to find

$$\lim_{n\to\infty}\int_0^x S_n h'(y)d\mu = \int_0^x f'(y)d\mu = f(x) - f(0). \tag{4.1}$$

So the integral of Fourier expansion on the derivative converges to $f(x) - f(0)$.

$$f(x) = f(0) + \lim_{n\to\infty}\int_0^x S_n h'(x)dy. \tag{4.2}$$

5 Pricing barrier options under GBM using trigonometric expansion

In this chapter, we mainly repeat the main derivations presented in [3], which constructs a trigonometric expansion, to approximate the barrier option price. Sections 5.1 to 5.5 replicate results from [3], whereas Section 5.6 introduces the first improvement to the method.

We start from deriving the pricing PDE for barrier options. From this PDE, we derive another PDE that the survival ch.f. satisfies, together with the associated boundary and initial conditions. This PDE is our target to solve, of which the solution is an approximation for the survival ch.f.. Consequently, the one dimensional COS method is used for getting the barrier option price. This approximation is then tested against a closed-form solution to determine the error of the method. Section 5.5 introduces improvements to the approach, which are again tested against the closed-form solution.

Following the idea from Section 5.5, the method is improved even further in Section 5.6.

5.1. The pricing PDE

To find the pricing PDE for an option price $V(t,S)$ with an underlying S that follows GBM, we start from the process dynamics in Equation (2.1)

$$dS_t = rS_t dt + \sigma S_t dW_t.$$

We use the martingale approach to derive the pricing PDE. First, use Itô's lemma on $V(t,S)$ to find

$$\begin{aligned}
dV_t &= \frac{\partial V}{\partial t}dt + \frac{\partial V}{\partial S}dS + \frac{1}{2}\frac{\partial^2 V}{\partial^2 S}dSdS \\
&= \frac{\partial V}{\partial t}dt + \frac{\partial V}{\partial S}(rS_t dt + \sigma S_t dW_t) + \frac{1}{2}\frac{\partial^2 V}{\partial^2 S}\sigma^2 S_t^2 dt \\
&= \left(\frac{\partial V}{\partial t} + rS_t\frac{\partial V}{\partial S} + \frac{\sigma^2 S_t^2}{2}\frac{\partial^2 V}{\partial^2 S}\right)dt + \sigma S_t \frac{\partial V}{\partial S}dW_t
\end{aligned}$$

The option pricing theory requires the discounted option price $e^{-rt}V_t$ to be a martingale, so $d(e^{-rt}V_t)$ should not have a dt term. Use Itô's product rule on $d(e^{-rt}V_t)$ to find

$$\begin{aligned}
d\left(e^{-rt}V_t\right) &= e^{-rt}dV_t + V_t de^{-rt} + de^{-rt}dV_t \\
&= e^{-rt}dV_t + V_t de^{-rt} \\
&= e^{-rt}\left(\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial^2 S} - rV_t\right)dt + e^{-rt}\sigma S\frac{\partial V}{\partial S}dW_t
\end{aligned}$$

Therefore, we find the pricing PDE under GBM:

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial^2 S} - rV = 0. \tag{5.1}$$

We can also transform to a log-asset price using $X_t = \ln(S_t)$ to get

$$\begin{aligned}
\frac{\partial V}{\partial S} &= \frac{\partial V}{\partial X}\frac{\partial X}{\partial S} \\
&= \frac{1}{S}\frac{\partial V}{\partial X} \\
\frac{\partial^2 V}{\partial S^2} &= \frac{\partial}{\partial S}\left(\frac{1}{S}\frac{\partial V}{\partial X}\right) \\
&= \frac{-1}{S^2}\frac{\partial V}{\partial X} + \frac{1}{S^2}\frac{\partial^2 V}{\partial X^2}.
\end{aligned}$$

Therefore, the log-asset price PDE is then

$$\frac{\partial V}{\partial t} + \left(r - \frac{\sigma^2}{2}\right)\frac{\partial V}{\partial X} + \frac{1}{2}\sigma^2\frac{\partial^2 V}{\partial^2 X} - rV_t = 0. \tag{5.2}$$

5.1.1. Valuation of barrier options

Barrier options act like European options unless their respective barrier has been breached at some time before the maturity of the option. We define τ_B to be the stopping time for the first time that the process S_t hits barrier B. For an up-and-out barrier this would be $\tau_B = \inf\{\hat{t} \geq t : S_{\hat{t}} \geq B\}$ or $\tau_B = \inf\{\hat{t} \geq t : X_{\hat{t}} \geq b\}, b = \ln B$ in the log-asset case. We can use this stopping time to write our payoff function as $f(S_t)\mathbb{1}_{\{\tau_B > T\}}$, where $f(\cdot)$ is the regular payoff function of a European option. We use the tower property $\mathbb{E}(XY) = \mathbb{E}(\mathbb{E}(XY|Y))$ to rewrite the option pricing formula

$$\begin{aligned}
V(t,s) &= e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left(f(S_T)\mathbb{1}_{\{\tau_B>T\}}|\mathcal{F}_t\right) \\
&= e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left(f(S_T)\mathbb{1}_{\{\tau_B>T\}}|S_t = s\right) \\
&= e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left(\mathbb{E}^{\mathbb{Q}}\left(f(S_T)\mathbb{1}_{\{\tau_B>T\}}|S_t = s, \tau_B > T\right)\right) \\
&= 0 \cdot p(\tau_B \leq T|S_t = s) + e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left(f(S_T)|S_t = s, \tau_B > T\right)p(\tau_B > T|S_t = s) \\
&= e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left(f(X_T)|X_t = s, \tau_B > T\right)p(\tau_B > T|S_t = s) \\
&= e^{-r(T-t)}\int_{\mathbb{R}} f(y)p(y|S_t = s, \tau_B > T)dy \cdot p(\tau_B > T|S_t = s) \\
&= e^{-r(T-t)}\int_{\mathbb{R}} f(y)p(y, \tau_B > T|S_t = s)dy.
\end{aligned}$$

Here $p(y, \tau_B > T|S_t = s)$ is the joint transition density function such that the realized path does not hit the barrier B before maturity T. In this thesis, this probability density will be called the *survival density function*. The corresponding characteristic function will be called the *survival characteristic function* (survival ch.f.)

We only evaluate the expectation if the barrier is not breached. Therefore, we can formulate a localized version of the Feynman-Kac theorem on an interval $[a,b]$ such that boundary conditions for V are established at the barrier.

Theorem 5.1.1 (Localized Feynman-Kac for GBM)**.** *Consider a process S_t whose dynamics follow GBM and define $\tau_{a\vee b} = \inf\{\hat{t} \geq t : S_{\hat{t}} \leq a \vee S_{\hat{t}} \geq b\}$ to be the first time that S_t exits the interval (a,b). Let $v : [a,b] \longrightarrow \mathbb{R}$ be a continuous payoff function with a compact support (value 0 outside the compact set). Then*

$$V(t,s) = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left[v(S_T)\mathbb{1}_{\{\tau_{a\vee b}>T\}}\middle|S_t = s\right]$$

is the unique, bounded solution of the PDE

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial^2 S} - rV = 0, \qquad a < S < b, \;\; 0 \leq t \leq T,$$

with boundary and terminal conditions

$$\begin{aligned} V(t,a) &= 0, & 0 \le t \le T \\ V(t,b) &= 0, & 0 \le t \le T \\ V(T,S) &= v(S_T), & a < S_T < b. \end{aligned}$$

Proof. We start with uniqueness of the solution. Let $u(t,s)$ be a solution to the PDE. We also consider process $\{M_t, t \ge 0\}$ defined as $M_t = e^{-rt}u(t,s)$. Using Itô's lemma, we find

$$\begin{aligned} dM_t &= u de^{-rt} + e^{-rt} du \\ &= e^{-rt}\left(\frac{\partial u}{\partial t} + rS\frac{\partial u}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 u}{\partial^2 S} - ru\right) dt + e^{-rt}\sigma S \frac{\partial u}{\partial S} dW_t. \end{aligned}$$

Since $u(t,s)$ satisfies the PDE, the dt term is equal to 0 and therefore M_t is a martingale. Since M_t is bounded and therefore integrable, we know

$$\begin{aligned} M_t = \mathbb{E}^{\mathbb{Q}}(M_T|\mathcal{F}_t) &= \mathbb{E}^{\mathbb{Q}}(e^{-rT}u(T,S)|\mathcal{F}_t) \\ &= \mathbb{E}^{\mathbb{Q}}(e^{-rT}v(S_T)\mathbb{1}_{\{\tau_{a\vee b} > T\}}|\mathcal{F}_t) \\ &= e^{-rt}e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}(v(S_T)\mathbb{1}_{\{\tau_{a\vee b} > T\}}|\mathcal{F}_t) \\ &= e^{-rt}V(t,s). \end{aligned}$$

So we uniquely get $u(t,s) = V(t,s)$.
To show that V is indeed a solution of the PDE, we simply check $V(t,a), V(t,b), V(T,s)$ to see that it satisfies the boundary and terminal conditions. $V(t,s)$ is a $\mathbb{Q}$-martingale, since

$$\mathbb{E}^{\mathbb{Q}}\left[e^{-rT}V(T,s)|\mathcal{F}_t\right] = e^{-rt}\mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-t)}v(S_T)\mathbb{1}_{\{\tau_{a\vee b} > T\}}|\mathcal{F}_t\right] = e^{-rt}V(t,s).$$

Using smoothness of V, we can apply Itô's formula (The martingale approach at the start of Section 5.1) to show that V satisfies the PDE. □

From this we can derive the pricing initial boundary value problem (IBVP) for barrier options

$$\begin{cases} \frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial^2 S} - rV = 0, & a < S < b, \ \ 0 \le t \le T \\ V(t,a) = 0, & 0 \le t \le T \\ V(t,b) = 0, & 0 \le t \le T \\ V(T,S) = v(S_T), & a < S_T < b. \end{cases} \tag{5.3}$$

A similar theorem can be formulated for the log-asset case. Then the PDE will be changed to Equation (5.2) and its boundaries follow the same log transformation. So $A = \ln(a), B = \ln(b)$ would be the new boundaries with the same 0 value attained at the boundary. Then the IBVP will be

$$\begin{cases} \frac{\partial V}{\partial t} + \left(r - \frac{\sigma^2}{2}\right)\frac{\partial V}{\partial X} + \frac{1}{2}\sigma^2 \frac{\partial^2 V}{\partial^2 X} - rV = 0, & A < X < B, \ \ 0 \le t \le T \\ V(t,A) = 0, & 0 \le t \le T \\ V(t,B) = 0, & 0 \le t \le T \\ V(T,X) = v(X_T), & A < X_T < B. \end{cases} \tag{5.4}$$

5.2. The PDE for the survival characteristic function

We have found an IBVP for the barrier option price. The next step is to solve this problem using the approximation for $V(t,S)$ found in Equation 2.5, derived in Appendix A.1.1.

$$\begin{aligned} V_2(t,s) &= e^{-r(T-t)}\sum_{p=1}^{K}\hat{\phi}_p(t,s)V_p \\ \hat{\phi}_p(t,s) &= \Im\left[\phi\left(\frac{p\pi}{b-a}, t; s\right)\cdot e^{-ip\pi\frac{a}{b-a}}\right] \\ V_p &= \frac{2}{b-a}\int_a^b V(T,y)\sin\left(p\pi\frac{y-a}{b-a}\right)dy. \end{aligned}$$

We substitute this into the PDE of the IBVP to find

$$\frac{\partial V_2}{\partial t} + rS\frac{\partial V_2}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V_2}{\partial^2 S} - rV = 0$$

$$rV + e^{-r(T-t)}\sum_{p=1}^{K}\frac{\partial\hat{\phi}_p(t,S)}{\partial t}V_p + rSe^{-r(T-t)}\sum_{p=1}^{K}\frac{\partial\hat{\phi}_p(t,S)}{\partial S}V_p + \frac{\sigma^2 S^2}{2}e^{-r(T-t)}\sum_{p=1}^{K}\frac{\partial^2\hat{\phi}_p(t,S)}{\partial S^2}V_p - rV = 0$$

$$e^{-r(T-t)}\sum_{p=1}^{K}\left[\frac{\partial\hat{\phi}_p(t,S)}{\partial t} + rS\frac{\partial\hat{\phi}_p(t,S)}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2\hat{\phi}_p(t,S)}{\partial S^2}\right]V_p = 0.$$

Note that this sum can be seen as the inner product of a vector of payoff coefficients and a vector of the PDEs of $\hat{\phi}_p$. An inner product attain value zero in three cases: the vectors are independent, or either vector is the zero vector. Since this holds for all types of payoff coefficients V_p, this is clearly not a zero vector or independent from the vector of PDEs. Therefore the only remaining possibility is that the vector of PDEs is the zero vector, so

$$\frac{\partial\hat{\phi}_p(t,S)}{\partial t} + rS\frac{\partial\hat{\phi}_p(t,S)}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2\hat{\phi}_p(t,S)}{\partial S^2} = 0$$

for all $p \in \{1,\dots,K\}$. We had boundaries $V(t,a) = V(t,b) = 0$. Therefore, since V_p is not 0, $\hat{\phi}_p(t,a) = \hat{\phi}_p(t,b) = 0$. To find the terminal condition $\hat{\phi}_p(T,S)$, we observe the Fourier-sine series expansion of $V_2(T,S)$

$$V_2(T,S) = \sum_{p=1}^{K} B_p \sin\left(p\pi\frac{S-a}{b-a}\right)$$

$$B_p = \frac{2}{b-a}\int_a^b V_2(T,y)\sin\left(p\pi\frac{y-a}{b-a}\right)dy.$$

Note that $B_p = V_p$ and therefore, $\hat{\phi}_p(T,S) = \sin\left(p\pi\frac{S-a}{b-a}\right)$. We apply change of variables $\tau = T - t$ such that we have an initial condition instead of terminal condition and find IBVP

$$\begin{cases} -\frac{\partial\hat{\phi}_p}{\partial\tau} + rS\frac{\partial\hat{\phi}_p}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2\hat{\phi}_p}{\partial S^2} = 0, \quad a < S < b \\ \hat{\phi}_p(\tau,a) = 0 \\ \hat{\phi}_p(\tau,b) = 0 \\ \hat{\phi}_p(0,S) = \sin\left(p\pi\frac{S-a}{b-a}\right). \end{cases} \tag{5.5}$$

Of course we can use the same approach to derive an IBVP for the log-asset price model

$$\begin{cases} -\frac{\partial\hat{\phi}_p}{\partial\tau} + \left(r - \frac{\sigma^2}{2}\right)\frac{\partial\hat{\phi}_p}{\partial X} + \frac{\sigma^2}{2}\frac{\partial^2\hat{\phi}_p}{\partial X^2} = 0, \quad A < X < B \\ \hat{\phi}_p(\tau,A) = 0 \\ \hat{\phi}_p(\tau,B) = 0 \\ \hat{\phi}_p(0,X) = \sin\left(p\pi\frac{X-a}{b-a}\right). \end{cases} \tag{5.6}$$

5.3. Approximating the survival characteristic function

Using the one-dimensional COS method, the option price is an inner product of Fourier-cosine coefficients and the survival density function, which can be accurately approximated by the survival ch.f., and the coefficients of the payoff function, V_p. We can derive an analytic expression for V_p, so we only need to determine the value of the survival ch.f.'s. The idea, same as in [3], is to resemble the survival ch.f. using Fourier series expansion.

5.3.1. Choices of expansion

The expansions are based on the half-range Fourier series. The expansion coefficients have analytic formulas shown in Equations (2.3b) and (2.3a), but we will not repeat these formulas in this section. This is because we generally do not have the information available to evaluate these analytic formulas. Therefore, we must determine these coefficients through other means.

Fourier-sine expansion

The Fourier-sine expansion for $f : [a, b] \longrightarrow \mathbb{R}$ is given by

$$f(x) \approx \sum_{k=1}^{K} B_k \sin\left(k\pi \frac{x-a}{b-a}\right)$$
$$\frac{df(x)}{dx} \approx \sum_{k=1}^{K} B_k \frac{k\pi}{b-a} \cos\left(k\pi \frac{x-a}{b-a}\right).$$

Expanding a function using the Fourier-sine expansion would result in the function attaining the value zero at the boundaries of its domain $[a, b]$. Its derivative, however, has no such fixed values at the boundaries.

Therefore, the Fourier-sine expansion is a great choice for dimensions with Dirichlet boundary conditions with value zero.

Fourier-cosine expansion

The Fourier-cosine expansion for $f : [a, b] \longrightarrow \mathbb{R}$ is given by

$$f(x) \approx \sum_{k=0}^{K-1}{}' A_k \cos\left(k\pi \frac{x-a}{b-a}\right)$$
$$\frac{df(x)}{dx} \approx -\sum_{k=0}^{K-1}{}' A_k \frac{k\pi}{b-a} \sin\left(k\pi \frac{x-a}{b-a}\right).$$

Unlike its sine counterpart, the Fourier-cosine expansion has non-zero values on the boundaries. Its derivative, however, does attain value zero at its boundaries.

For similar reasoning as its sine counterpart, the Fourier-cosine expansion is best chosen for dimensions that have Neumann boundary conditions with value zero.

Boundary relaxation

Both types of Fourier expansions are best suited for dimensions with some type of boundary conditions on both boundaries. If only one of the boundaries were to have boundary condition 0, this would lead to a less accurate approximation. Boundary relaxation could be a solution for this. Without loss of generality, we will consider the example Dirichlet boundary conditions $f(a) = 0, f(b) = c \neq 0$. The Dirichlet boundary condition on $x = a$ would suggest choosing the Fourier-sine expansion. To prevent our approximation from having $f(b) = 0$, we expand on $[a, b + \epsilon]$. Then $f(b + \epsilon) = 0$, but since this is outside the actual range we consider, $f(b)$ is not forced to be 0. The expansion would then be

$$f(x) \approx \sum_{k=1}^{K} B_k \sin\left(k\pi \frac{x-a}{b-a+\epsilon}\right).$$

Similar arguments hold for $[a - \epsilon, b]$ if we want relaxation on the lower boundary.

Expansion on the first order derivative

A main insight of [3] is that, if the value at the left boundary is known, we can expand on the derivative of the function. Then through integration, we derive an approximation for the function itself. The resulting trigonometric expansion is tested in [3] to have better convergence than directly expanding on the function (as shown in Chapter 4).

$$\begin{aligned}
\frac{df(x)}{dx} &\approx \sum_{k=0}^{K}{}' C_k \cos\left(k\pi\frac{x-a}{b-a}\right) \\
f(x) - f(a) &\approx \int_a^x \frac{C_0}{2} + \sum_{k=1}^{K} C_k \cos\left(k\pi\frac{y-a}{b-a}\right) dy \\
f(x) - f(a) &\approx \frac{C_0}{2}(x-a) + \sum_{k=1}^{K} C_k \frac{b-a}{k\pi} \sin\left(k\pi\frac{y-a}{b-a}\right) \\
f(x) &\approx f(a) + \frac{C_0}{2}(x-a) + \sum_{k=1}^{K} C_k \frac{b-a}{k\pi} \sin\left(k\pi\frac{y-a}{b-a}\right).
\end{aligned}$$

We call this trigonometric expansion on $f(x)$. In the time dimension $a = 0$, so we automatically satisfy the initial condition $f(0)$.

5.3.2. Asset-dimension

First we consider the asset-dimension. In this dimension we have Dirichlet boundary conditions, which would suggest a sine series expansion. Then on the boundaries $x = a, x = b$ we force the function to attain the value 0. We write the asset-dimension as a function $f_1(x)$ and call the sine expansion approximation of this function $S_N f_1(x)$.

$$f_1(x) \approx S_N f_1(x) = \sum_{k_1=1}^{K} A_{k_1} \sin\left(k_1\pi\frac{x-a_1}{b_1-a_1}\right),$$

where a_1, b_1 are the bounds of the expansion interval. If we consider the asset price as underlying, we set $a_1 = a, b_1 = b$ to force the boundary conditions to hold. When considering the log-asset case, we set $a_1 = A = \ln(a), b_1 = B = \ln(b)$.

5.3.3. τ-dimension

As shown in [3], a sine or cosine expansion for $\hat{\phi}_p$ in τ-dimension is not suitable. Therefore, we use a cosine expansion on the derivative with respect to τ. Let us write $f_2(\tau)$ as the function in the τ-dimension. Then $C_N f_2'(\tau)$ is the cosine expansion of the τ derivative. From this, we can use the fundamental theorem of calculus to find a suitable expansion for $f_2(\tau)$.

$$\begin{aligned}
f_2'(\tau) \approx C_N f_2'(\tau) &= \sum_{k_2=0}^{K-1}{}' A_{k_2} \cos\left(k_2\pi\frac{\tau-a_2}{b_2-a_2}\right) \\
T_N f_2(\tau) - f_2(a_2) &= \frac{1}{2}A_0(\tau-a_2) + \sum_{k_2=1}^{K-1} A_{k_2}\frac{b_2-a_2}{k_2\pi}\sin\left(k_2\pi\frac{\tau-a_2}{b_2-a_2}\right) \\
T_N f_2(\tau) &= \frac{1}{2}A_0(\tau-a_2) + f_2(a_2) + \sum_{k_2=1}^{K-1} A_{k_2}\frac{b_2-a_2}{k_2\pi}\sin\left(k_2\pi\frac{\tau-a_2}{b_2-a_2}\right),
\end{aligned}$$

where $a_2 = 0$. With this trigonometric expansion, we automatically satisfy initial condition $T_N f_2(0) = f_2(0)$

5.3.4. Trigonometric expansion of $\hat{\phi}_p$

We now choose to expand $\frac{\partial\hat{\phi}_p}{\partial\tau}$ in the way we just described. This results in

$$\frac{\partial\hat{\phi}_p}{\partial\tau}(\tau,x) \approx \sum_{k_1=1}^{K}\sum_{k_2=0}^{K-1}{}' A_{k_1,k_2}\cos\left(k_2\pi\frac{\tau-a_2}{b_2-a_2}\right)\sin\left(k_1\pi\frac{x-a_1}{b_1-a_1}\right),$$

where x can be asset price S if $a_1 = a, b_1 = b$ or log-asset price X if $a_1 = A, b_1 = B$. Like before, we use the fundamental theorem of calculus to find

$$\begin{aligned}\hat{\phi}_p(\tau, x) \approx T_K\hat{\phi}_p(\tau, x) = &\sum_{k_1=1}^{K}\sum_{k_2=1}^{K-1} A_{k_1,k_2}\frac{b_2-a_2}{k_2\pi}\sin\left(k_2\pi\frac{\tau-a_2}{b_2-a_2}\right)\sin\left(k_1\pi\frac{x-a_1}{b_1-a_1}\right)\\ &+\sum_{k_1=1}^{K} A_{k_1,0}\frac{1}{2}(\tau-a_2)\sin\left(k_1\pi\frac{x-a_1}{b_1-a_1}\right)+\hat{\phi}_p(a_2,x).\end{aligned}$$

The only time at which we have information is the initial time $\tau = 0$. Therefore, we can use this $a_2 = 0$ to further evaluate the trigonometric expansion

$$\begin{aligned}T_K\hat{\phi}_p(\tau, x) = &\sum_{k_1=1}^{K}\sum_{k_2=1}^{K-1} A_{k_1,k_2}\frac{b_2-a_2}{k_2\pi}\sin\left(k_2\pi\frac{\tau-a_2}{b_2-a_2}\right)\sin\left(k_1\pi\frac{x-a_1}{b_1-a_1}\right)\\ &+\sum_{k_1=1}^{K} A_{k_1,0}\frac{\tau}{2}\sin\left(k_1\pi\frac{x-a_1}{b_1-a_1}\right)+\sin\left(p\pi\frac{x-a_1}{b_1-a_1}\right), \quad a_2 = 0.\end{aligned} \tag{5.7}$$

We substitute this into the PDE of IBVP 5.5 or 5.6. By construction, the trigonometric expansion automatically satisfies the boundary conditions and initial condition of both IBVPs. Before we substitute into the PDE, we first define 5 vectors that will aid us with calculating and representing the equation.

We introduce row vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ that represent the x terms for different values of k_1 and the τ terms for different values of k_2 respectively. We also create row vectors $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$ that are the partial derivative of $\mathbf{v}_1$ to x, the partial derivative of $\mathbf{v}_2$ to τ and the second partial derivative of $\mathbf{v}_1$ to x, respectively.

$$\begin{aligned}\mathbf{v}_1 &= \begin{pmatrix}\sin\left(\pi\frac{x-a_1}{b_1-a_1}\right) & \sin\left(2\pi\frac{x-a_1}{b_1-a_1}\right) & \cdots & \sin\left(N\pi\frac{x-a_1}{b_1-a_1}\right)\end{pmatrix}\\ \mathbf{v}_2 &= \begin{pmatrix}\frac{1}{2}\tau & \frac{b_2-a_2}{\pi}\sin\left(\pi\frac{\tau-a_2}{b_2-a_2}\right) & \cdots & \frac{b_2-a_2}{(N-1)\pi}\sin\left((N-1)\pi\frac{\tau-a_2}{b_2-a_2}\right)\end{pmatrix}\\ \mathbf{z}_1 &= \begin{pmatrix}\frac{\pi}{b_1-a_1}\cos\left(\pi\frac{x-a_1}{b_1-a_1}\right) & \frac{2\pi}{b_1-a_1}\cos\left(2\pi\frac{x-a_1}{b_1-a_1}\right) & \cdots & \frac{N\pi}{b_1-a_1}\cos\left(N\pi\frac{x-a_1}{b_1-a_1}\right)\end{pmatrix}\\ \mathbf{z}_2 &= \begin{pmatrix}\frac{1}{2} & \cos\left(\pi\frac{\tau-a_2}{b_2-a_2}\right) & \cdots & \cos\left((N-1)\pi\frac{\tau-a_2}{b_2-a_2}\right)\end{pmatrix}\\ \mathbf{z}_3 &= -\begin{pmatrix}\left(\frac{\pi}{b_1-a_1}\right)^2\sin\left(\pi\frac{x-a_1}{b_1-a_1}\right) & \left(\frac{2\pi}{b_1-a_1}\right)^2\sin\left(2\pi\frac{x-a_1}{b_1-a_1}\right) & \cdots & \left(\frac{N\pi}{b_1-a_1}\right)^2\sin\left(N\pi\frac{x-a_1}{b_1-a_1}\right)\end{pmatrix}.\end{aligned} \tag{5.8}$$

We show matrix A with the expansion coefficients and show the vectorized form of this matrix, since we will need this soon.

$$A = \begin{pmatrix}A_{1,0} & A_{1,1} & \cdots & A_{1,K-1}\\ A_{2,0} & A_{2,1} & \cdots & A_{2,K-1}\\ \vdots & \vdots & \ddots & \vdots\\ A_{K,0} & A_{K,1} & \cdots & A_{K,K-1}\end{pmatrix} \qquad \text{vec}(A) = \begin{pmatrix}A[:,0]\\ A[:,1]\\ \vdots\\ A[:,K-1]\end{pmatrix},$$

where $A[:,k]$ is the k-th column of A. With these vectors and matrix we can give easy formulations for the partial derivatives of $\hat{\phi}_p$

$$\begin{aligned}\frac{\partial\hat{\phi}_p}{\partial\tau} &= \mathbf{v}_1 A\mathbf{z}_2^T\\ \frac{\partial\hat{\phi}_p}{\partial x} &= \mathbf{z}_1 A\mathbf{v}_2^T + \frac{p\pi}{b_1-a_1}\cos\left(p\pi\frac{x-a_1}{b_1-a_1}\right)\\ \frac{\partial^2\hat{\phi}_p}{\partial x^2} &= \mathbf{z}_3 A\mathbf{v}_2^T - \left(\frac{p\pi}{b_1-a_1}\right)^2\sin\left(p\pi\frac{x-a_1}{b_1-a_1}\right),\end{aligned}$$

where x represents the asset or log-asset price.

For higher order tensors we are not able to represent these partial derivatives in the form of a matrix vector multiplications. We can use our vectorized matrix A and the Kronecker product for a different formulation for the partial derivatives. Note that the order of the vectors in the Kronecker product is dependent on the vectorization order chosen for A.

$$\begin{aligned}\frac{\partial \hat{\phi}_p}{\partial \tau} &= (\mathbf{z}_2 \otimes \mathbf{v}_1)\mathrm{vec}(A)\\ \frac{\partial \hat{\phi}_p}{\partial X} &= (\mathbf{v}_2 \otimes \mathbf{z}_1)\mathrm{vec}(A) + \frac{p\pi}{b_1-a_1}\cos\left(p\pi\frac{x-a_1}{b_1-a_1}\right)\\ \frac{\partial^2 \hat{\phi}_p}{\partial X^2} &= (\mathbf{v}_2 \otimes \mathbf{z}_3)\mathrm{vec}(A) - \left(\frac{p\pi}{b_1-a_1}\right)^2\sin\left(p\pi\frac{x-a_1}{b_1-a_1}\right).\end{aligned}$$

By substituting this into the PDE and plugging in training points τ and x, where we evaluate this PDE, we find a linear system of equations in which the coefficients of A are the unknowns. In [3] it has been investigated that M equidistant training points give slightly better results than a non-equidistant grid, where $M = K$, with K the number of expansion terms. We use the previously defined vectors to define several matrices. We let x_m and τ_l be the training points and $m, l = 1, \ldots, M$. We define the matrices as

$$\mathbf{V}_1[m,:] = \left(\sin\left(\pi\frac{x_m-a_1}{b_1-a_1}\right) \quad \sin\left(2\pi\frac{x_m-a_1}{b_1-a_1}\right) \quad \cdots \quad \sin\left(N\pi\frac{x_m-a_1}{b_1-a_1}\right)\right).$$

Matrices $\mathbf{V}_2, \mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_3$ are defined equivalently. Then at training point (x_m, τ_l) we have equation

$$\begin{aligned}&\left((\mathbf{Z}_2[l,:]\otimes\mathbf{V}_1[m,:]) - \mathbf{V}_2[l,:]\otimes\left[rS_m\mathbf{Z}_1[m,:] + \frac{\sigma^2 S_m^2}{2}\mathbf{Z}_3[m,:]\right]\right)\mathrm{vec}(A)\\ &= rS_m\frac{p\pi}{b_1-a_1}\cos\left(p\pi\frac{S_m-a_1}{b_1-a_1}\right) - \frac{\sigma^2 S_m^2}{2}\left(\frac{p\pi}{b_1-a_1}\right)^2\sin\left(p\pi\frac{S_m-a_1}{b_1-a_1}\right)\end{aligned} \tag{5.9}$$

for the asset case or

$$\begin{aligned}&\left((\mathbf{Z}_2[l,:]\otimes\mathbf{V}_1[m,:]) - \mathbf{V}_2[l,:]\otimes\left[\left(r-\frac{\sigma^2}{2}\right)\mathbf{Z}_1[m,:] + \frac{\sigma^2}{2}\mathbf{Z}_3[m,:]\right]\right)\mathrm{vec}(A)\\ &= \left(r-\frac{\sigma^2}{2}\right)\frac{p\pi}{b_1-a_1}\cos\left(p\pi\frac{x_m-a_1}{b_1-a_1}\right) - \frac{\sigma^2}{2}\left(\frac{p\pi}{b_1-a_1}\right)^2\sin\left(p\pi\frac{x_m-a_1}{b_1-a_1}\right)\end{aligned} \tag{5.10}$$

for the log-asset case.

It is worth noting, that this construction is very similar to the collocation method, but with one distinct difference: existing methods in literature apply finite difference in the time dimension, whereas we expand on the time dimension using the trigonometric expansion.

5.4. Results for the GBM log-asset model

In this section, we replicate results from [3] for the GBM model, where we fix $K = M = 64$. We consider an up-and-out barrier option with strike price $E = 90$ and barrier $B = 120$. Therefore, $b_1 = \ln(120)$ and we choose $a_1 = \ln(40)$ as a lower bound such that $\mathbb{P}(x_t < a_1) \approx 0$. We take model parameters $(r, \sigma) = (0.1, 0.1)$ and evaluate the option prices at $\tau \in \{0, 0.1, 0.25, 0.5, 1\}$. We compare the prices obtained with trigonometric expansion to the prices from the benchmark Solution (2.8). The results for the log-asset model are shown in Figure 5.1.

The relative errors, $\frac{|Approximation-Benchmark|}{Benchmark}$, are set to be 0 whenever the benchmark option price is lower than 10^{-1}. Below that threshold, it is more informative to look at the absolute errors. We can see that for options in the money, the relative error barely exceeds 10^{-2}.

There are two cases where we exceed this threshold, namely $\tau = 0$ with any S_0 and $S_0 > 115$ for all τ. To see why this happened we have to go back to Equation 5.7 and substitute $\tau = 0$, which yields

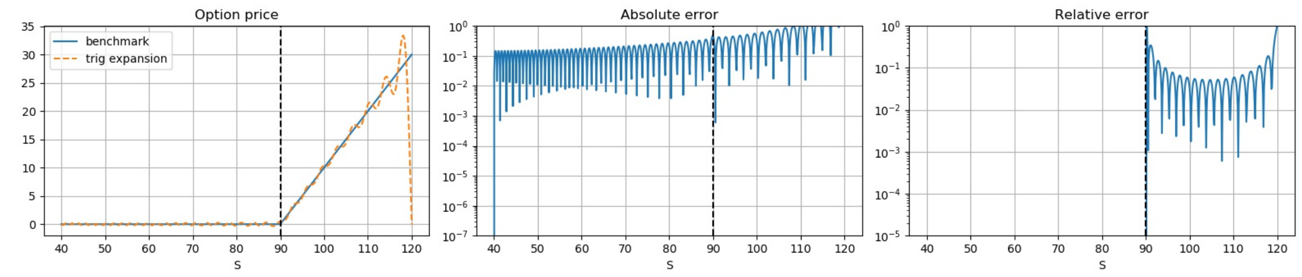
$$T_K\hat{\phi}_p(0,x) = \sin\left(p\pi\frac{x-a_1}{b_1-a_1}\right).$$

Next, we also look at the approximation for the option price in Equation 2.5. If we substitute $\tau = 0$ here ($t = T$), we find

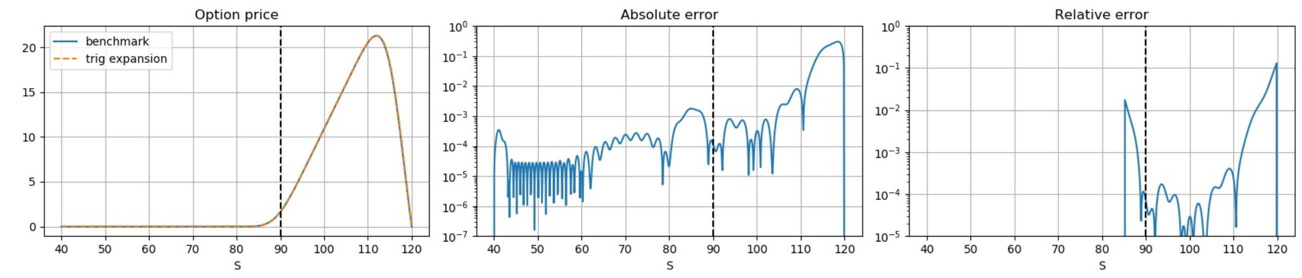
$$
\begin{aligned}
V_2(T, X) &= \sum_{p=1}^{N} \hat{\phi}_p(t, X) V_p \\
\hat{\phi}_p(T, X) &\approx \sin\left(p\pi \frac{x - a_1}{b_1 - a_1}\right) \\
V_p &= \frac{2}{b-a} \int_a^b V(T, y) \sin\left(p\pi \frac{y-a}{b-a}\right) dy,
\end{aligned}
$$

which is simply a Fourier-sine series expansion on $V(T, x)$. From this, we can make two conclusions. Firstly, the approximation of $\hat{\phi}_p$ at $\tau = 0$ is independent of the expansion coefficients and, therefore, also from the method used to train these coefficients. Secondly, Since the payoff function $V(T, x)$ is non-continuous as a periodic function at the barrier, we experience the Gibbs phenomenon here. Furthermore, we have a non-continuous derivative at the strike price.

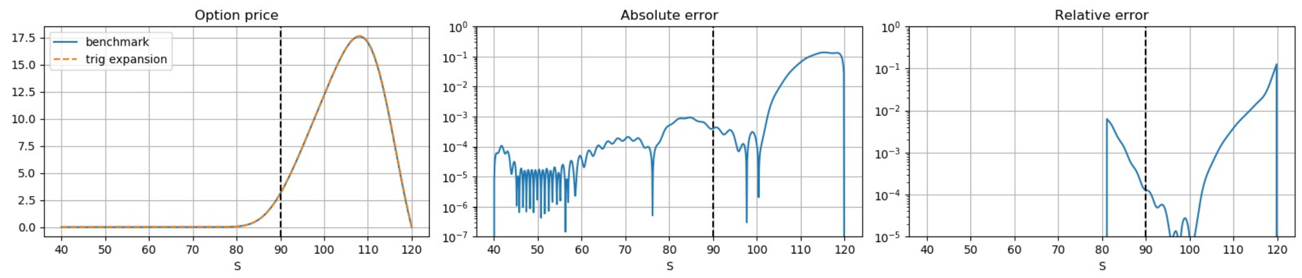
Since our approximation $T_K \hat{\phi}_p(\tau, x)$ is the sum of this initial sine term and a multidimensional sum of the expansion coefficients multiplied with sine or cosine functions, we expect the errors in this initial condition to also effect the approximate option price at other times to maturity. This is observed in the absolute errors in Subfigure 5.1a, which shows that errors significantly increase when the initial asset price is above the strike price, especially near the barrier. This behaviour also appears in other subfigures for a larger time to maturity, although it smoothens for larger maturities.



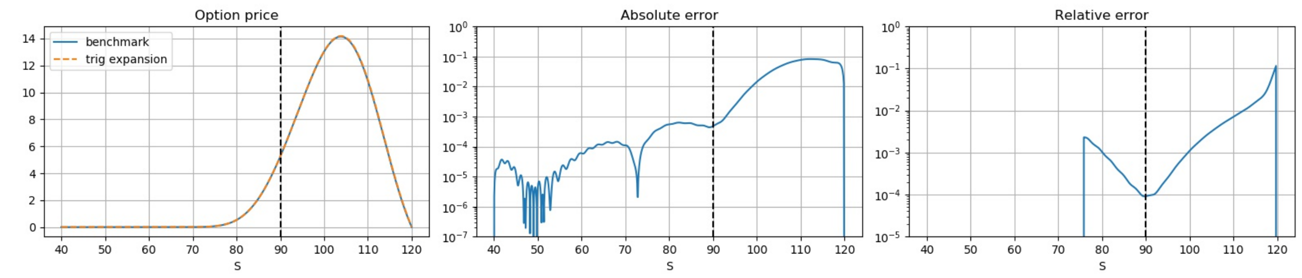
(a) Option price, absolute error and relative error for $\tau = 0$



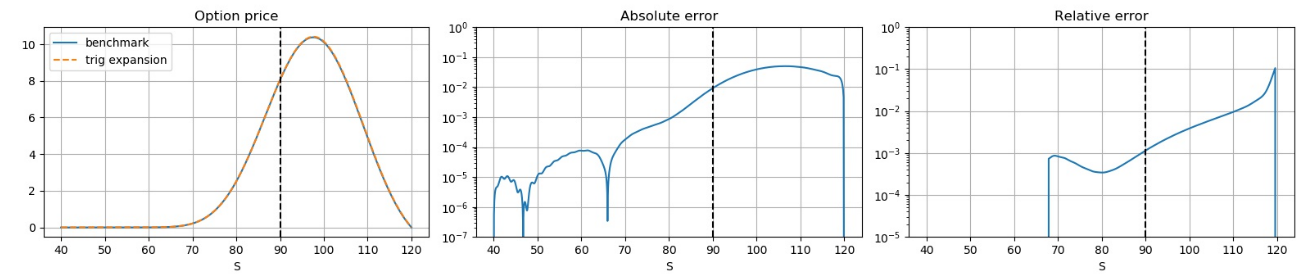
(b) Option price, absolute error and relative error for $\tau = 0.1$



(c) Option price, absolute error and relative error for $\tau = 0.25$



(d) Option price, absolute error and relative error for $\tau = 0.5$



(e) Option price, absolute error and relative error for $\tau = 1$

Figure 5.1: Option price and errors for various τ using trigonometric expansion vs closed-form solution for an up-and-out barrier option with parameters $(r, \sigma, E, B) = (0.1, 0.1, 90, 120)$ under GBM log-asset dynamics.

5.5. Matching the mass of the survival pdf

The coefficients of the trigonometric expansion are found by a supervised machine learning method, in which we insert training points into the PDE and then solve the resulting linear system.

The survival pdf corresponding to a barrier option has concentrated mass towards the barrier. [3] chose an equidistant grid in the log-asset domain, but if we translate it back to the asset-domain, we can see that these points are denser near the lower range of asset points. Figure 5.2 shows how an equidistant grid in the log-asset domain is translated to the corresponding points in the asset domain.

We can clearly see that the translated points are more denser for lower values of the asset price, which is opposite of where the concentrated mass of the survival pdf lies. Therefore, [3] proposed to construct the trigonometric expansion in the asset domain, and consequently use equidistant training points in the asset domain, to better capture the concentrated mass of the survival pdf in the training step. We already found the linear system for the asset domain in Equation 5.9.

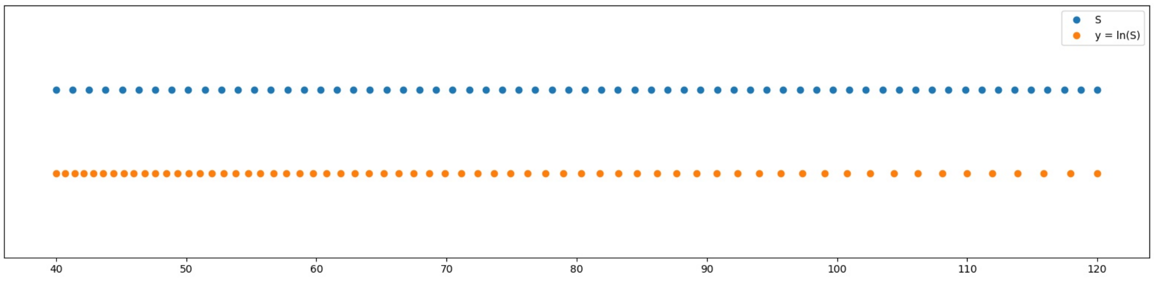


Figure 5.2: Training point density on the asset domain for equidistant points S and points $y = \ln(S)$ that were equidistant on the log-asset domain.

5.5.1. Results for the GBM model

Using the same parameters as for the log-asset model, we now solve the asset model. We again fix $K = M = 64$ and use parameters $(r, \sigma, E, B) = (0.1, 0.1, 90, 120)$ and $\tau \in \{0, 0.1, 0.25, 0.5, 1\}$. The results are shown in Figure 5.3. For each time to maturity, τ, we can see that the accuracy is better after we move from the log-asset to the asset domain, but the error patterns are very similar: for initial asset prices near the barrier, we have larger errors and for $\tau = 0$ we still suffer from the Gibbs phenomenon.

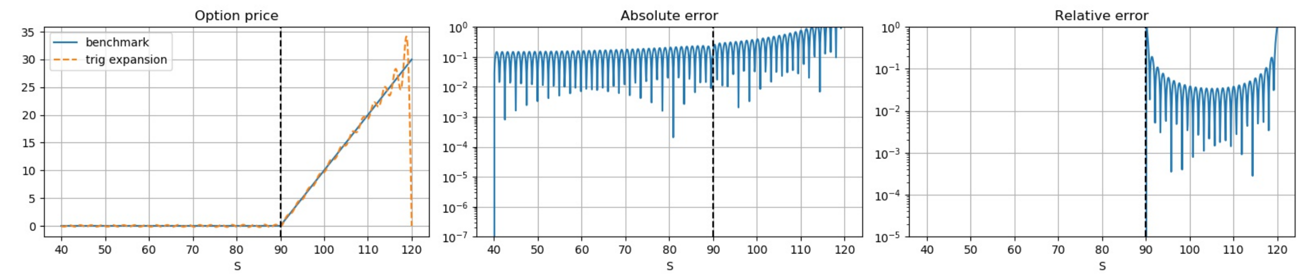
We also compare the results with those from a high-order finite difference scheme [17] and an uninterpretable neural network [23] in Table 5.1. We should note that the methods all assume GBM dynamics, but with different parameters and barrier options with different strike prices and barrier levels. Therefore, the comparison is done through the maximum relative error $\frac{error}{E}$. We also split the metric into two parts. The first, MRE_1, considers relative error near-the-money, which we take as $[75, 105]$ in our case. The second, MRE_2, considers errors far-from-the money, which we take as $[105, 120]$.

The trigonometric expansion has similar order errors to the high order finite difference method from [17]. The trigonometric expansion has the advantage of the calculating time being offline. Whenever we want to calculate the option price after training the model, the calculations are almost instant.

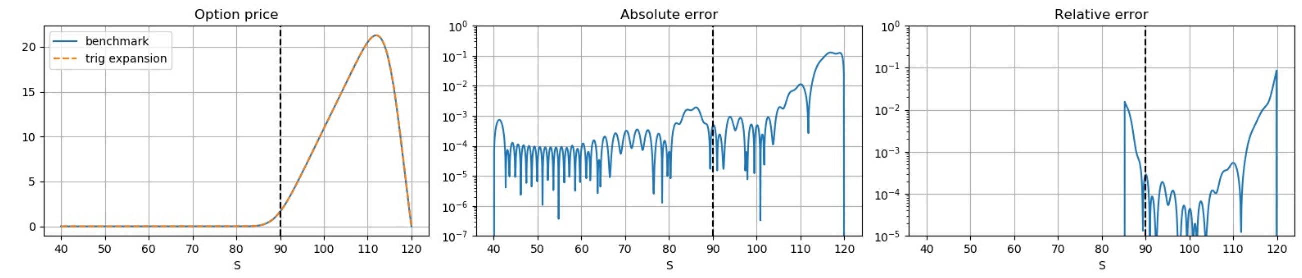
The neural network from [23] has slightly lower relative errors, but it should be noted that they do not have an interpretable neural network. Furthermore, we have no information on the time it takes to train their model.

Method	MRE_1	MRE_2	Time taken (seconds)
Trigonometric expansion (asset model, $K = 64$)	10^{-3}	10^{-2}	194 (offline)
Neural network [23]	10^{-4}	10^{-3}	- (offline)
High order finite difference [17]	10^{-3}	10^{-3}	0.0087

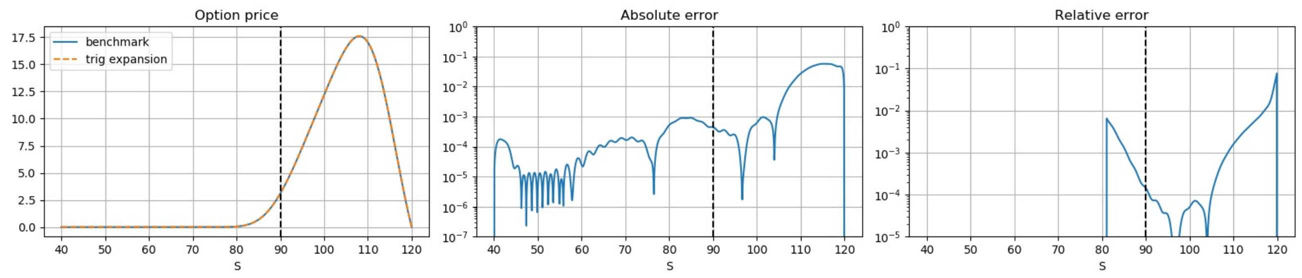
Table 5.1: Performance for different methods of barrier option pricing under GBM. We consider the Maximum relative error $(\frac{error}{E})$ for non-0 time to maturity. MRE_1 and MRE_2 are near-the-money ($S_0 \in [75, 105]$ for trigonometric expansion) and far-out-of-the-money respectively.



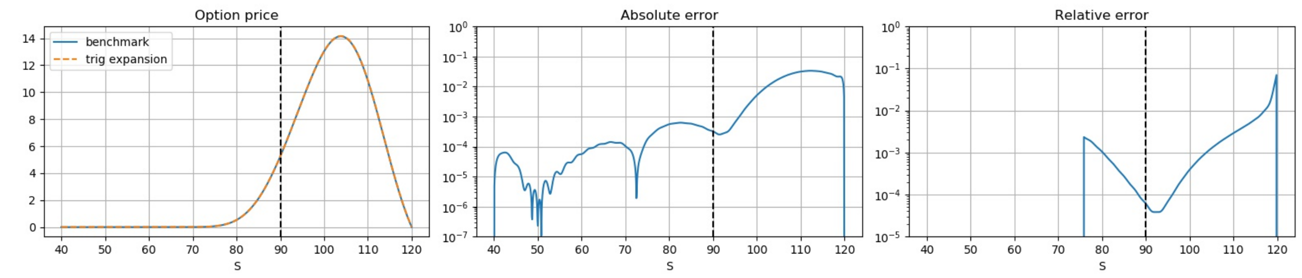
(a) Option price, absolute error and relative error for $\tau = 0$



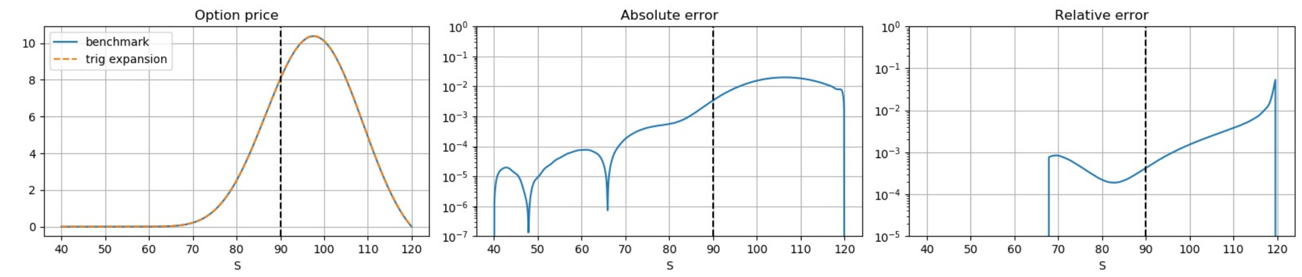
(b) Option price, absolute error and relative error for $\tau = 0.1$



(c) Option price, absolute error and relative error for $\tau = 0.25$



(d) Option price, absolute error and relative error for $\tau = 0.5$



(e) Option price, absolute error and relative error for $\tau = 1$

Figure 5.3: Option price and errors for various τ using trigonometric expansion vs closed-form solution for an up-and-out barrier option with parameters $(r, \sigma, E, B) = (0.1, 0.1, 90, 120)$ under GBM asset dynamics.

5.6. Our contribution 2: Inverted Log-Asset model

In Section 5.5 it is shown that the accuracy of the model is improved when using an equidistant grid of points in the asset domain in the training step, compared to an equidistant grid in the log-asset domain. The reasoning for this, is that the survival pdf has more mass near the barrier compared to relatively low asset prices.

The previous results lead to an insight: if we apply a different change of variables, such that an equidistant grid in that domain leads to a grid slightly denser near the barrier, this might improve the method even further. Figure 5.4 shows translated grid in the asset domain from an equidistant grid in other domains. Two new domains have been included and investigated. The inverted-log-asset domain and the exponential-asset domain, respectively defined as

$$y = \ln(B + \lambda - S)$$
$$y = e^{\lambda S},$$

where $\lambda > 0$ is a parameter related to how dense the points are near the barrier. As seen in Figure 5.4, for the inverted-log-asset domain, a low λ means high density near the barrier and a high λ means a more equidistant grid in the asset domain. For the exponential-asset domain, low λ is more spread out and high λ results in a higher density near the barrier. Later we look into the choice of λ, but first we must decide on which transformation to use.

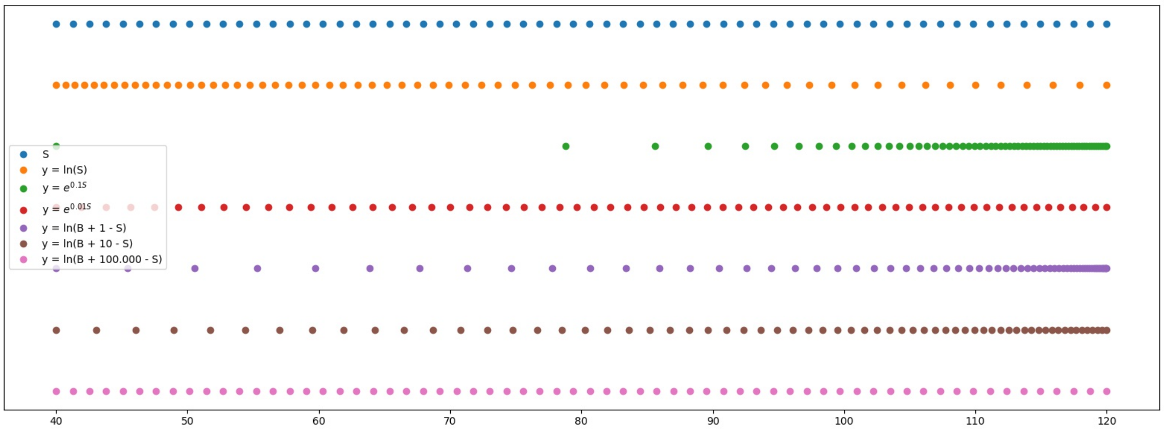


Figure 5.4: Distribution of $M = 64$ training points on the asset domain for equidistant grids on other domains.

5.6.1. Choosing a change of variables

$y = e^{\lambda S}$ is an easier function to use for change of variables for the PDE and doesn't require the extra parameter B, but for our analytically solved integral $\chi_p(c, d)$, we would get

$$\chi_p(c, d) = \int_c^d \frac{1}{\lambda} \ln(y) \sin\left(p\pi \frac{y - a}{b - a}\right) dy,$$

which is very hard to solve analytically.

For $y = \ln(B + \lambda - S)$, the change of variables for the PDE is slightly more complex, but for $\chi_p(c, d)$ we have

$$\begin{aligned}\chi_p(c, d) &= \int_c^d (B + \lambda - e^y) \sin\left(p\pi \frac{y - a}{b - a}\right) dy \\ &= (B + \lambda) \int_c^d \sin\left(p\pi \frac{y - a}{b - a}\right) dy - \int_c^d e^y \sin\left(p\pi \frac{y - a}{b - a}\right) dy,\end{aligned}$$

which is analytically solvable. Therefore, we choose the form $y = \ln(B + \lambda - S)$.

5.6.2. Choice of λ

Next, we determine which value of λ to use. We consider a lower bound A and upper bound B for the asset domain. The training points are taken from an equidistant grid between $a = \ln(B + \lambda - B)$ and $b = \ln(B + \lambda - A)$. This means that the points y_i, where $i = 0, \dots, M - 1$, are defined as

$$\begin{aligned} y_i &= a + \frac{i(b-a)}{M-1} \\ &= \ln(\lambda) + \frac{i}{M-1}(\ln(B+\lambda-A) - \ln(\lambda)). \end{aligned}$$

These would be represented on the asset domain as

$$\begin{aligned} S_i &= B + \lambda - e^{y_i} \\ &= B + \lambda - e^{\ln(\lambda)} e^{\frac{i}{M-1}(\ln(B+\lambda-A)-\ln(\lambda))} \\ &= B + \lambda - \lambda e^{\frac{i}{M-1}(\ln(B+\lambda-A)-\ln(\lambda))} \end{aligned}$$

The distance between points on the asset domain would be

$$\begin{aligned} S_{i+1} - S_i &= B + \lambda - \lambda e^{\frac{i+1}{M-1}(\ln(B+\lambda-A)-\ln(\lambda))} - B - \lambda + \lambda e^{\frac{i}{M-1}(\ln(B+\lambda-A)-\ln(\lambda))} \\ &= \lambda \left(e^{\frac{i}{M-1}(\ln(B+\lambda-A)-\ln(\lambda))} - e^{\frac{i+1}{M-1}(\ln(B+\lambda-A)-\ln(\lambda))} \right) \\ &= \lambda e^{\frac{i}{M-1}(\ln(B+\lambda-A)-\ln(\lambda))} \left(1 - e^{\frac{1}{M-1}(\ln(B+\lambda-A)-\ln(\lambda))} \right) \\ &= \lambda e^{\frac{i}{M-1}(\ln(B+\lambda-A)-\ln(\lambda))} \left(1 - \left(\frac{e^{\ln(B+\lambda-A)}}{e^{\ln(\lambda)}} \right)^{\frac{1}{M-1}} \right) \\ &= \lambda e^{\frac{i}{M-1}(\ln(B+\lambda-A)-\ln(\lambda))} \left(1 - \left(\frac{B+\lambda-A}{\lambda} \right)^{\frac{1}{M-1}} \right) \end{aligned}$$

Note that for $\lambda \longrightarrow \infty$, we find that $\ln(B + \lambda - A) \longrightarrow \ln(\lambda)$ and therefore,

$$e^{\frac{i}{M-1}(\ln(B+\lambda-A)-\ln(\lambda))} \approx 1.$$

Then for large λ, $S_{i+1} - S_i$ is no longer dependent on i and therefore, there must be an equidistant grid in the asset domain. Since the left and right boundary are fixed, we find that this grid must coincide with the training points from an equidistant grid in the asset domain. For smaller values of λ, however, there is still a dependency on i.

The 'best' choice of λ would depend on many factors like the time to maturity and the interest rate, since these influence the survival pdf of the barrier option. For larger maturities, we want a more evenly spaced grid and for very small τ, it might be better to use smaller λ. We construct an equation that can solve for optimal λ for a given proportion p of points that should be to the right of an asset price P. For example, say we have bounds $(a, b) = (40, 120)$, $M = 32$ points and we want $p = 0.75$ of the points to be on the left of $P = 65$. Then we want $S_i = P$, where $i = 0.25 * 32 = 8$. Note that this transformation does not allow us to have a grid that is denser on the left side, and therefore we find a bound for p

$$p \geq 1 - \frac{P-a}{b-a}.$$

Now we can introduce the equation that would solve for λ

$$\begin{aligned} \ln(b + \lambda - P) &= \ln(b + \lambda - b) + p\Big[\ln(b + \lambda - a) - \ln(b + \lambda - b)\Big] \\ \ln(b + \lambda - P) &= \ln(\lambda) + p\Big[\ln(b + \lambda - a) - \ln(\lambda)\Big] \\ b + \lambda - P &= \lambda \left(\frac{b+\lambda-a}{\lambda} \right)^p . \end{aligned} \tag{5.11}$$

This does not have a closed form solution, but can be solved numerically. Furthermore, the choice of p and P have high influence to the model and these have to be chosen based on personal insight.

5.6.3. Applying change of variables to the PDE

We let $y = \log(B + \lambda - S)$, which means $S = B + \lambda - e^y$ then

$$\begin{aligned}
\frac{\partial y}{\partial S} &= \frac{-1}{B + \lambda - S} \\
\frac{\partial^2 y}{\partial S^2} &= \frac{-1}{(B + \lambda - S)^2} \\
\frac{\partial V}{\partial S} &= \frac{\partial V}{\partial y}\frac{\partial y}{\partial S} = \frac{-1}{B + \lambda - S}\frac{\partial V}{\partial y} \\
&= -e^{-y}\frac{\partial V}{\partial y} \\
\frac{\partial^2 V}{\partial S^2} &= \frac{\partial V}{\partial S}\left(\frac{-1}{B + \lambda - S}\frac{\partial V}{\partial y}\right) \\
&= \frac{-1}{(B + \lambda - S)^2}\frac{\partial V}{\partial y} + \frac{1}{(B + \lambda - S)^2}\frac{\partial^2 V}{\partial y^2} \\
&= -e^{-2y}\frac{\partial V}{\partial y} + e^{-2y}\frac{\partial^2 V}{\partial y^2}.
\end{aligned}$$

Plugging this into our PDE yields:

$$\begin{aligned}
&\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} - rV = 0 \\
&\frac{\partial V}{\partial t} - \left(rSe^{-y} + \frac{S^2\sigma^2 e^{-2y}}{2}\right)\frac{\partial V}{\partial y} + \frac{S^2\sigma^2 e^{-2y}}{2}\frac{\partial^2 V}{\partial y^2} - rV = 0 \\
&\frac{\partial V}{\partial t} - \frac{(B + \lambda - e^y)}{e^y}\left(r + (B + \lambda - e^y)\frac{\sigma^2}{2}e^{-y}\right)\frac{\partial V}{\partial y} + \frac{(B + \lambda - e^y)^2\sigma^2 e^{-2y}}{2}\frac{\partial^2 V}{\partial y^2} - rV = 0.
\end{aligned}$$

We again use the approximation

$$V(t, y) \approx V_2(t, y) = e^{-r(T-t)}\sum_{p=1}^{K}\hat{\phi}_p(t, y)V_p,$$

and identical arguments as before to yield

$$\begin{cases}
-\frac{\partial\hat{\phi}_p}{\partial\tau} - \frac{(B+\lambda-e^y)}{e^y}\left(r + (B + \lambda - e^y)\frac{\sigma^2}{2}e^{-y}\right)\frac{\partial\hat{\phi}_p}{\partial y} + \frac{(B+\lambda-e^y)^2\sigma^2 e^{-2y}}{2}\frac{\partial^2\hat{\phi}_p}{\partial y^2} = 0, a < y < b \\
\hat{\phi}_p(\tau, a) = 0 \\
\hat{\phi}_p(\tau, b) = 0 \\
\hat{\phi}_p(0, y) = \sin\left(p\pi\frac{y-a}{b-a}\right),
\end{cases} \tag{5.12}$$

where $a = \log(B + \lambda - b_{asset}), b = \log(B + \lambda - a_{asset})$, so the lower and upper boundaries switch due to the $-S$ in the transformation. Of course, for barrier options, a_{asset} and b_{asset} are the lower and upper barrier of the option.

As noted before, we can analytically solve V_p. Equations (A.2) and (A.3) derived in Appendix A are the closed-form solutions.

$$
\begin{aligned}
\Psi_p(c,d) &= \int_c^d \sin\left(p\pi\frac{y-a}{b-a}\right)dy \\
&= \begin{cases} \frac{b-a}{p\pi}\left[\cos\left(p\pi\frac{c-a}{b-a}\right)-\cos\left(p\pi\frac{d-a}{b-a}\right)\right], & p\neq 0 \\ 0 & p=0 \end{cases} \\
\chi_p^{LogAsset}(c,d) &= \int_c^d e^y \sin\left(p\pi\frac{y-a}{b-a}\right)dy \\
&= \frac{1}{1+\left(\frac{p\pi}{b-a}\right)^2}\left[e^d\sin\left(p\pi\frac{d-a}{b-a}\right)-e^c\sin\left(p\pi\frac{c-a}{b-a}\right)\right. \\
&\quad \left. -\frac{p\pi}{b-a}e^d\cos\left(p\pi\frac{d-a}{b-a}\right)+\frac{p\pi}{b-a}e^c\cos\left(p\pi\frac{c-a}{b-a}\right)\right]
\end{aligned}
$$

Note that $p = 0$ is not a valid case, since the sine expansion makes it such that we start at $p = 1$. Again, we have E as the strike price. For a European call option in our log-asset setting we have payoff $V(T,y) = (B+\lambda-e^y-E)^+$. So for $y \geq \ln(B+\lambda-E)$, we have payoff 0. Therefore, we can rewrite V_p as

$$
\begin{aligned}
V_p &= \frac{2}{b-a}\int_a^b (B+\lambda-e^y-E)^+\sin\left(p\pi\frac{y-a}{b-a}\right)dy \\
&= \frac{2}{b-a}\int_a^{\ln(B+\lambda-E)} (B+\lambda-e^y-\xi)\sin\left(p\pi\frac{y-a}{b-a}\right)dy \\
&= \frac{2}{b-a}\left(-\chi_p^{LogAsset}(a,\ln(B+\lambda-E))+(B+\lambda-E)\Psi_p(a,\ln(B+\lambda-E))\right).
\end{aligned}
$$

5.6.4. Trigonometric expansion

Now we check whether any update is needed to the trigonometric expansion after applying the change of variables. For $y = a, y = b$ we still have boundaries of the PDE that are equal to 0, so a sine expansion is still appropriate in the asset (inverted-log-asset) dimension. For the time (τ) dimension, nothing has changed, so the trigonometric expansion obtained from integrating out the Fourier-cosine expansion on the derivative with respect to time is also still the desired expansion.

Since the expansions themself have not changed, the matrices involved in the linear system from the training step remain unchanged. The only thing that has changed, is the PDE. The resulting linear system reads

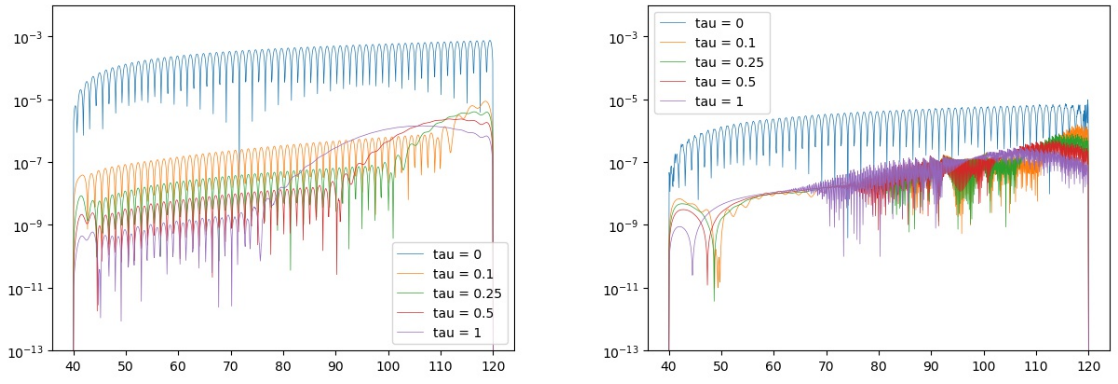
$$
\begin{aligned}
&\left(-(\mathbf{Z}_2[l,:]\otimes\mathbf{V}_1[m,:])+\mathbf{V}_2[l,:]\otimes\left[-\frac{(B+\lambda-e^{y_m})}{e^{y_m}}\left(r+(B+\lambda-e^y)\frac{\sigma^2}{2}e^{-y_m}\right)\mathbf{Z}_1[m,:]\right.\right. \\
&\left.\left.+\frac{(B+\lambda-e^{y_m})^2\sigma^2e^{-2y_m}}{2}\mathbf{Z}_3[m,:]\right]\right)\text{vec}(A) \\
&= \frac{(B+\lambda-e^{y_m})}{e^{y_m}}\left(r+(B+\lambda-e^y)\frac{\sigma^2}{2}e^{-y_m}\right)\frac{p\pi}{b_1-a_1}\cos\left(p\pi\frac{y_m-a_1}{b_1-a_1}\right) \\
&+\frac{(B+\lambda-e^{y_m})^2\sigma^2e^{-2y_m}}{2}\left(\frac{p\pi}{b_1-a_1}\right)^2\sin\left(p\pi\frac{y_m-a_1}{b_1-a_1}\right).
\end{aligned}
$$

5.6.5. Results

The performance of the new model is dependent on λ. Therefore, we investigated 3 values of λ. The first λ value is very large to check whether the results correspond to those of the regular asset-domain model, as indicated from analytic analysis in Section 5.6.2. The next choice of λ is for a relatively small value, such that we have a far denser allocation of training points towards the barrier. Lastly we also choose a value of λ based on Equation (5.11).

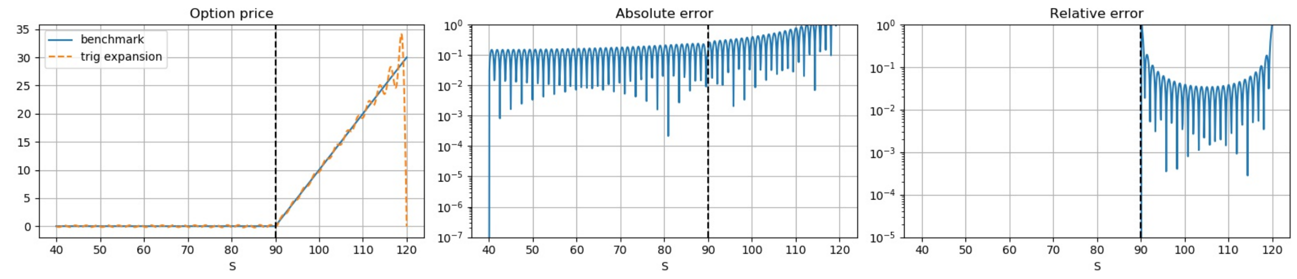
Inverted-log-asset domain with large λ

The first value of λ for which we analyse the results is 'a large value'. In the experiments, we use $\lambda = 1 \cdot 10^{10}$. The option prices and absolute and relative erros can be seen in Figure 5.6. If we compare these plots to the plots of the asset dynamics in Figure 5.3, we can see that they are (almost) identical. This further confirms our analytic findings from Section 5.6.2: for large λ, the inverted-log-asset model and the asset model perform the same, because the training points are identical. Figure 5.5 shows the difference between the asset model and inverted-log-asset model for different values of τ and 2 values of λ. We can see that the difference decreases for larger λ. Clearly, both models perform almost identically.

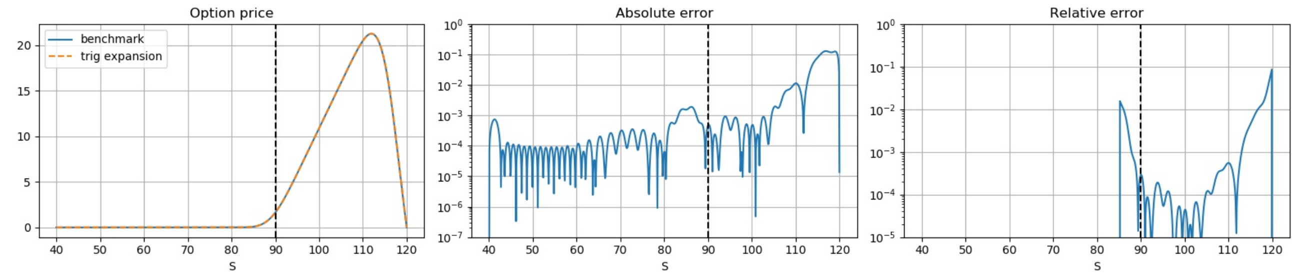


(a) Difference for λ = 1.000.000 **(b)** Difference for λ = 100.000.000

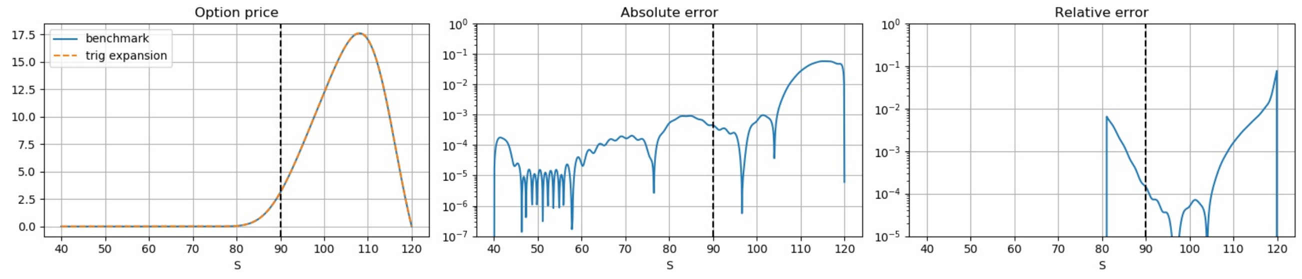
Figure 5.5: Difference between the option price found with the asset model vs the inverted-log-asset model with large λ



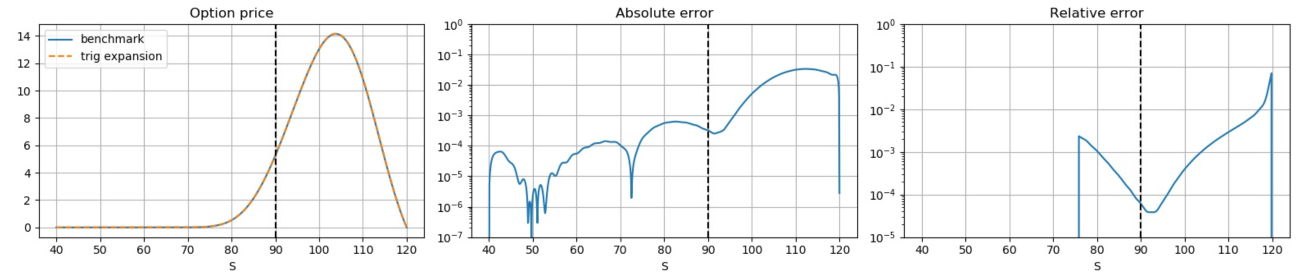
(a) Option price, absolute error and relative error for $\tau = 0$



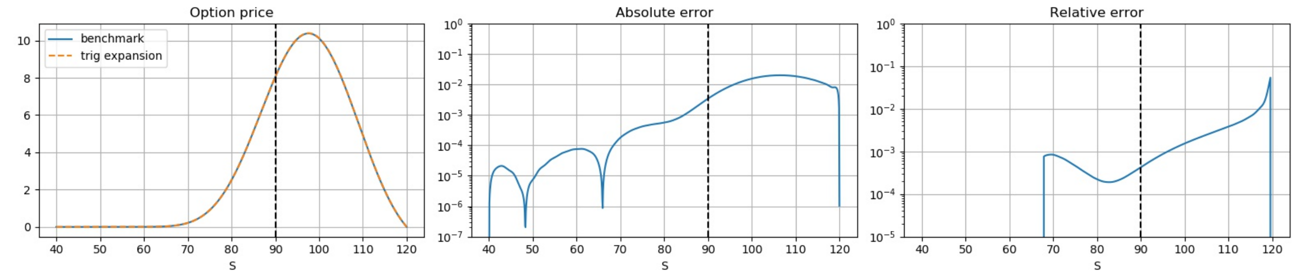
(b) Option price, absolute error and relative error for $\tau = 0.1$



(c) Option price, absolute error and relative error for $\tau = 0.25$



(d) Option price, absolute error and relative error for $\tau = 0.5$



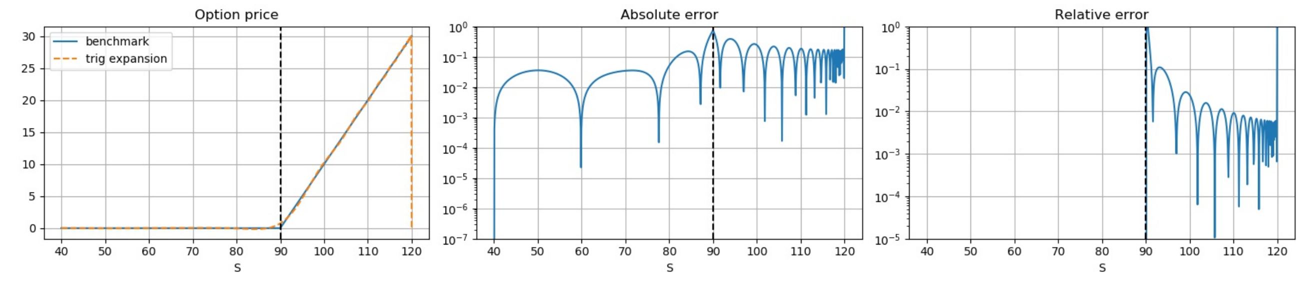
(e) Option price, absolute error and relative error for $\tau = 1$

Figure 5.6: Option price and errors for various τ using trigonometric expansion vs closed-form solution for an up-and-out barrier option with parameters $(r, \sigma, E, B) = (0.1, 0.1, 90, 120)$ under GBM inverted-log-asset dynamics with $\lambda = 10^{10}$.

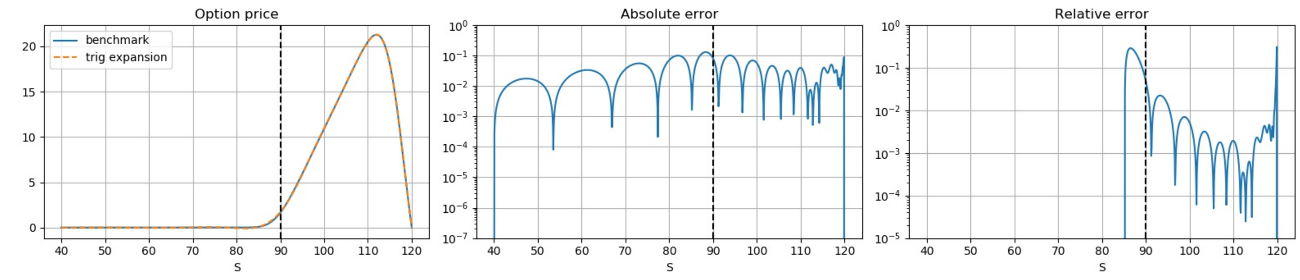
Inverted-log-asset domain with small λ

Next, we look at the behaviour of the model when λ is relatively small. We choose $\lambda = 10^{-5}$. The results for this model can be seen in Figure 5.7. Compared to a large λ (or equivalently, an equidistant grid in the asset domain), we see that errors near the barrier have decreased, but errors near the money and near the lower bound have increased significantly.

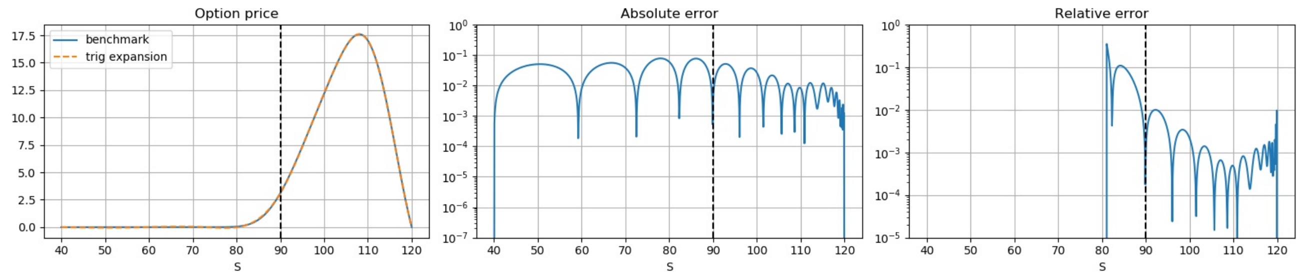
This is expected from how the transformation was defined. Due to the small λ and consequently very few training points for lower asset prices, the density mass on the low asset prices part is poorly captured. Equation (5.11) suggests that portion p of points lie to the right of the halfway point $P = \frac{a+b}{2} = 80$ is $p \approx 0.96$. So with $M = 64$ training points, this results in approximately 2 to 3 training points in the lower half of the asset price domain.



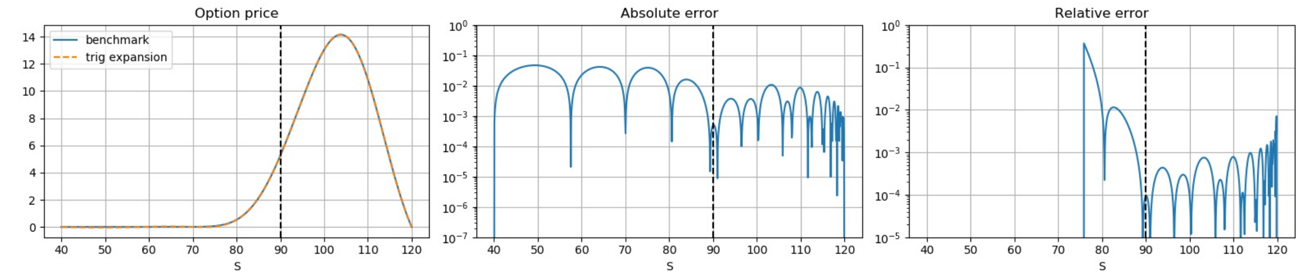
(a) Option price, absolute error and relative error for $\tau = 0$



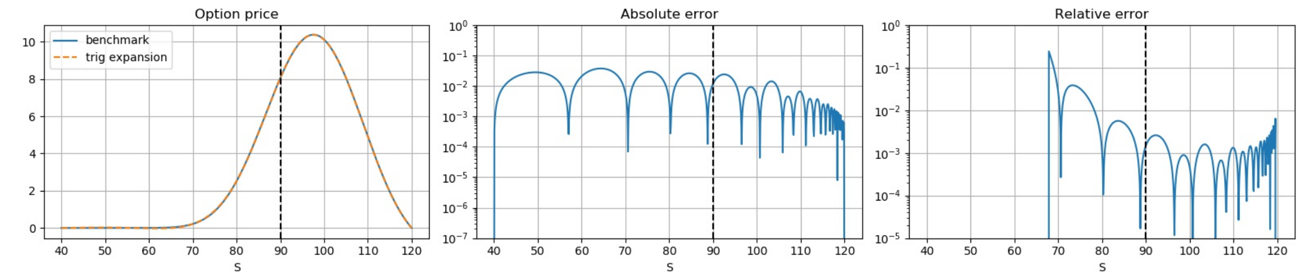
(b) Option price, absolute error and relative error for $\tau = 0.1$



(c) Option price, absolute error and relative error for $\tau = 0.25$



(d) Option price, absolute error and relative error for $\tau = 0.5$



(e) Option price, absolute error and relative error for $\tau = 1$

Figure 5.7: Option price and errors for various τ using trigonometric expansion vs closed-form solution for an up-and-out barrier option with parameters $(r, \sigma, E, B) = (0.1, 0.1, 90, 120)$ under GBM inverted-log-asset dynamics with $\lambda = 10^{-5}$.

Inverted-log-asset domain with optimal λ

Lastly we look at a value of λ that should give us better accuracy in theory. We assume the same model parameters as before, so $(r, \sigma, a, b, E) = (0.1, 0.1, 40, 120, 90)$. We choose $P = \frac{b+a}{2} = 80$ in the center between a and b. Then $p \downarrow 0.5$ would lead to $\lambda \to \infty$. We first decide the value of p and then, using Equation (5.11), we can find λ that satisfies the equation through a numerical root finder like `scipy.optimize.root()`.

For a general feel of the performance, several values of λ have been tested. Of the values $\lambda \in \{1 \cdot 10^{-5}, 0.001, 0.1, 1, 10, 1 \cdot 10^{10}\}$, $\lambda = 1$ had the best looking performance and $\lambda = 10$ also performed quite well. $\lambda = 1$ corresponds to $p \approx 0.845$. Three other cases that have been tested were $p = 0.85$ ($\lambda = 0.85356746$), which had very similar performance to $\lambda = 1$, $p = 0.8$ ($\lambda = 3.12380311$), which had slightly worse performance, and $p = 0.75$ ($\lambda = 7.65951536$), which had performance somewhat similar to $\lambda = 10$ but out of the 3 it appeared to give the worst fit. Therefore, the results in this section are based on a model with $\lambda = 1$.

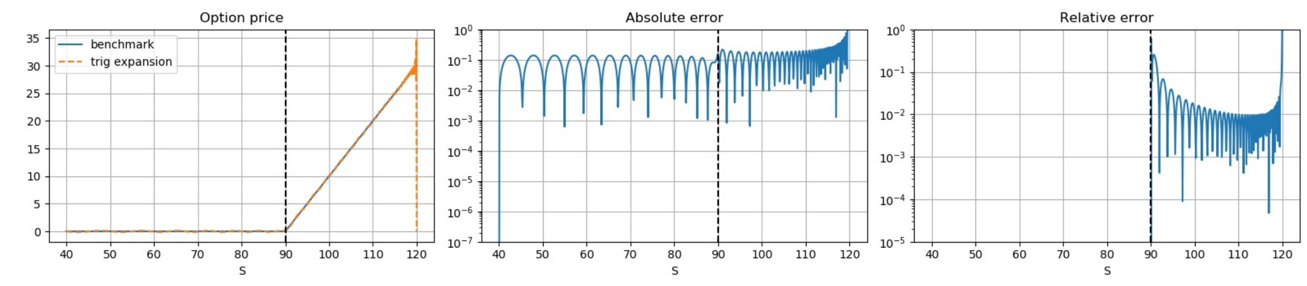
The results for $\lambda = 1$ are presented in Figure 5.8. We compare these results to the results from the Figure 5.3 (or 5.6, since they look identical).

A general trend we see, which we already noticed for very small λ, is that the errors near the barrier decrease and errors near the money and for lower asset prices increase. For smaller maturities we see modest improvements. For $\tau = 1$, however, the error has been decreased by tenfold in some places and barely increased for low asset prices. We have already seen that longer maturities seem to lead to a smoother trigonometric function, and thus seem to be in favour of this change of variables.

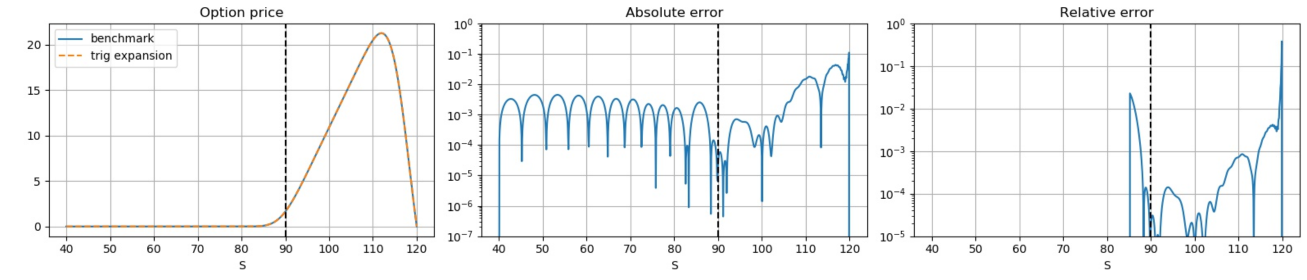
Table 5.2 summarizes the same comparison to other methods from literature as Table 5.1, but now also with the inverted log-asset model. Except for $\tau = 0$, where there is only a slight improvement, we significantly improve the general performance of the method. The inverted log-asset model also outperforms both methods from literature.

Method	MRE_1	MRE_2	Time taken (seconds)
Trigonometric expansion (asset model, $K = 64$)	10^{-3}	10^{-2}	194 (offline)
Trig. exp. (inv. log-asset model $K = 64$)	10^{-5}	10^{-3}	190 (offline)
Neural network [23]	10^{-4}	10^{-3}	- (offline)
High order finite difference [17]	10^{-3}	10^{-3}	0.0087

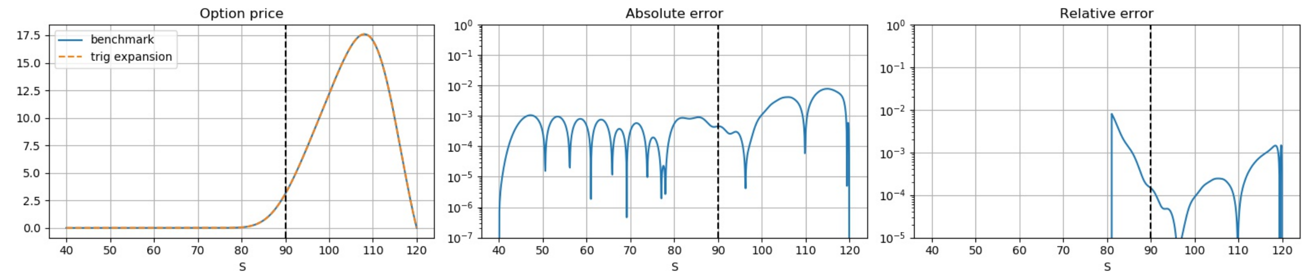
Table 5.2: Performance for different methods of barrier option pricing under GBM. We consider the Maximum relative error $\left(\frac{error}{\mathrm{E}}\right)$ for non-0 time to maturity. MRE_1 and MRE_2 are near-the-money ($S_0 \in [75, 105]$ for trigonometric expansion) and far-from-the-money respectively.



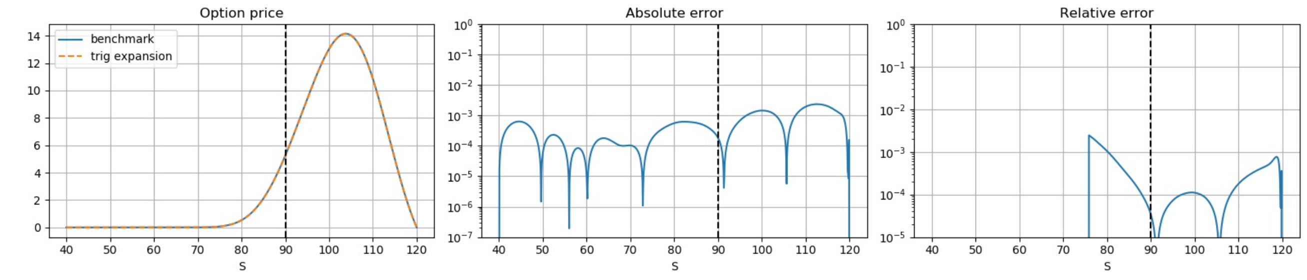
(a) Option price, absolute error and relative error for $\tau = 0$



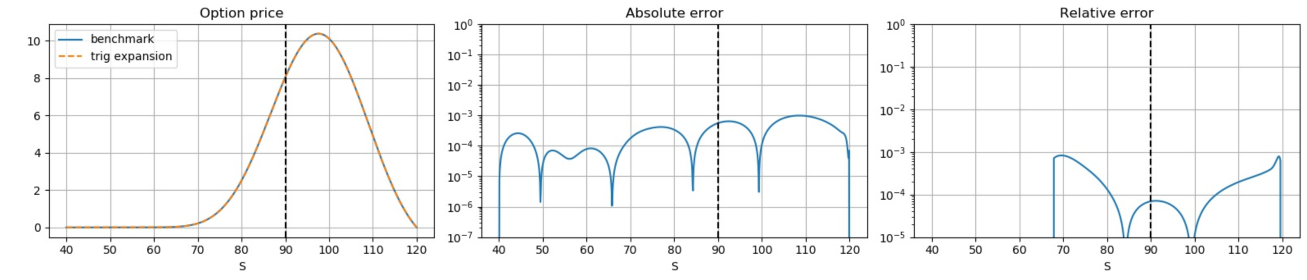
(b) Option price, absolute error and relative error for $\tau = 0.1$



(c) Option price, absolute error and relative error for $\tau = 0.25$



(d) Option price, absolute error and relative error for $\tau = 0.5$



(e) Option price, absolute error and relative error for $\tau = 1$

Figure 5.8: Option price and errors for various τ using trigonometric expansion vs closed-form solution for an up-and-out barrier option with parameters $(r, \sigma, E, B) = (0.1, 0.1, 90, 120)$ under GBM inverted-log-asset dynamics with $\lambda = 1$.

5.7. Conclusions

In Sections 5.1 and 5.2 we derived the pricing PDE and an IBVP for the survival ch.f., which can be used to approximate the barrier option price using the COS method from [5].

Section 5.3 constructs a trigonometric expansion, to approximate the survival ch.f., with unknown expansion coefficients. The approximation is inserted into the PDE of the survival ch.f., resulting in a linear system that can be solved to find the unknown coefficients.

In Section 5.4 we compare the results for the log-asset model to an analytic solution. While promising, the model had significantly increasing errors when the initial asset price approaches the upper barrier. Additionally, for $\tau = 0$, the method's construction leads to worse convergence behaviour, due to the Gibbs phenomenon near the barrier, even affecting options with longer maturity times.

To improve performance near the upper boundary, where the probability density mass is denser, more training points near the upper barrier are allocated. Conversely, fewer training points are required at lower initial asset prices due to less probability density mass. Section 5.5 addresses this by moving to the asset domain, which yields more favourable results.

Finally, Section 5.6 further improves the accuracy of the method through a variable transformation, resulting in the inverted-log-asset model. This model outperforms the previous models, as well as the finite difference method and neural network method from literature.

One potential problem with the trigonometric expansion model is that the resulting linear system from the PDE, has a matrix with K^{2N} elements, where K is the number of expansion points and N is the dimension. Therefore, this system becomes very large as the number of dimensions increases.

6 Our contribution 3: Pricing barrier options under SABR using trigonometric expansion

In Chapter 5 we used trigonometric expansion to approximate the barrier option price under GBM. In [3] this method was also applied to Heston's model. In this chapter, we explore another stochastic volatility model, the SABR model.

We follow the same steps as in Chapter 5, starting with the pricing PDE in Section 6.1. Next, we find the corresponding survival ch.f. PDE in Section 6.2, and in Section 6.3 we use trigonometric expansion to approximate that function. Section 6.4 presents and discusses the results, followed by variable transformation, similar to Section 5.6, to improve the results in Section 6.5.

6.1. The pricing PDE

The first step for pricing the barrier options under SABR, following the previous chapter, is to find the pricing PDE for an option price $V(t, S, \sigma)$ with underlying S and volatility σ. We start from the SABR dynamics in Equation (2.2)

$$\begin{aligned} dS_t &= \sigma_t S_t^{\beta} dW_t^{\mathbb{Q}} \\ d\sigma_t &= \alpha\sigma_t dZ_t^{\mathbb{Q}} \\ dW_t^{\mathbb{Q}} dZ_t^{\mathbb{Q}} &= \rho dt, \end{aligned}$$

where α is the volatility of volatility and β determines the slope of the implied skew. β is often fixed before calibrating the model to a market. In this thesis, we consider the commonly used $\beta = 0.5$ when showing and analysing results. Lastly, ρ is the correlation between the Brownian Motion processes.

The parameters can also be described by their effect on the implied volatility curve. Generally we fix β. Then σ mainly determines the curve's height, α controls the curvature of the implied volatility curve, and ρ controls the curve's skew (similar effect on the curve as beta).

We use the martingale approach to find the PDE (derivation in Appendix A.2.1) as follows:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma_t^2 S_t^{2\beta}\frac{\partial^2 V}{\partial S^2} + \frac{1}{2}\alpha^2\sigma_t^2\frac{\partial^2 V}{\partial \sigma^2} + \alpha\sigma_t^2 S_t^{\beta}\rho\frac{\partial^2 V}{\partial S\partial\sigma} - rV = 0. \tag{6.1}$$

6.1.1. Pricing PDE for barrier options

To find the IBVP for pricing barrier options, we follow the exact same steps as for the GBM case from Section 5.1.1, but with slightly different equations.

Theorem 6.1.1 (Localized Feynman-Kac for SABR). *Consider a process S_t whose dynamics follow SABR and define $\tau_{a\vee b} = \inf\{\hat{t} \geq t : S_{\hat{t}} \leq a \vee S_{\hat{t}} \geq b\}$ to be the first time that S_t exits the interval (a, b). Let $v : [a, b] \longrightarrow \mathbb{R}$*

be a continuous payoff function with a compact support (value 0 outside the compact set). Then

$$V(t,s,\sigma_0) = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left[v(S_T)\mathbb{1}_{\{\tau_{a\vee b}>T\}}\middle|S_t = s, \sigma_t = \sigma_0\right]$$

is the unique, bounded solution of the PDE

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma_t^2 S_t^{2\beta}\frac{\partial^2 V}{\partial S^2} + \frac{1}{2}\alpha^2\sigma_t^2\frac{\partial^2 V}{\partial \sigma^2} + \alpha\sigma_t^2 S_t^\beta \rho\frac{\partial^2 V}{\partial S \partial \sigma} - rV = 0, \quad 0 \le t \le T,\ a < S < b,\ \sigma > 0$$

with boundary and terminal conditions

$$\begin{aligned} &V(t,a,\sigma) = 0, && 0 \le t \le T, \sigma > 0 \\ &V(t,b,\sigma) = 0, && 0 \le t \le T, \sigma > 0 \\ &\frac{\partial V}{\partial t}\Big|_{\sigma=0} - rV(t,S,0) = 0, && 0 \le t \le T, a < S < b \\ &V(T,S,\sigma) = v(S_T), && a < S_T < b, \sigma > 0. \end{aligned}$$

The proof is analogous to that in Theorem 5.1.1. From the localized Feynman-Kac theorem, we find the IBVP for the SABR model, i.e.,

$$\begin{cases} \frac{\partial V}{\partial t} + \frac{1}{2}\sigma_t^2 S_t^{2\beta}\frac{\partial^2 V}{\partial S^2} + \frac{1}{2}\alpha^2\sigma_t^2\frac{\partial^2 V}{\partial \sigma^2} + \alpha\sigma_t^2 S_t^\beta \rho\frac{\partial^2 V}{\partial S \partial \sigma} - rV = 0, & 0 \le t \le T,\ a < S < b,\ \sigma > 0 \\ V(t,a,\sigma) = 0, & 0 \le t \le T, \sigma > 0 \\ V(t,b,\sigma) = 0, & 0 \le t \le T, \sigma > 0 \\ \frac{\partial V}{\partial t}\Big|_{\sigma=0} - rV(t,S,0) = 0, & 0 \le t \le T, a < S < b \\ V(T,S,\sigma) = v(S_T), & a < S_T < b, \sigma > 0. \end{cases} \tag{6.2}$$

6.2. The PDE for the survival characteristic function

Following the method laid out in Chapter 5, we substitute the SIN pricing formula for V into the problem. We use the approximation from Equation (2.5).

$$V_2(t,S,\sigma) = e^{-r(T-t)}\sum_{p=1}^{K}\hat{\phi}_p(t,S,\sigma)V_p \tag{6.3}$$

$$\hat{\phi}_p(t,S) = \Im\left[\phi\left(\frac{p\pi}{b-a}, t; x, \sigma\right)\cdot e^{-ip\pi\frac{a}{b-a}}\right]$$

$$V_p = \frac{2}{b-a}\int_a^b V(T,y)\sin\left(p\pi\frac{y-a}{b-a}\right)dy,$$

After substituting this into the PDE and based on the same arguments as in Section 5.2, we find the PDE for the survival ch.f.:

$$\frac{\partial\hat{\phi}_p}{\partial t} + \frac{1}{2}\sigma_t^2 S_t^{2\beta}\frac{\partial^2\hat{\phi}_p}{\partial S^2} + \frac{1}{2}\alpha^2\sigma_t^2\frac{\partial^2\hat{\phi}_p}{\partial\sigma^2} + \alpha\sigma_t^2 S_t^\beta\rho\frac{\partial^2\hat{\phi}_p}{\partial S\partial\sigma} = 0. \tag{6.4}$$

Again, like in the GBM case, the boundary conditions easily follow, since V_p is not zero, so $\hat{\phi}_p(t,a,\sigma) = \hat{\phi}_p(t,b,\sigma) = 0$. Based on the same reasoning as before, we use sine expansion on $V(T,S,\sigma)$ to derive the terminal condition for the survival ch.f.:

$$\hat{\phi}_p(T,S,\sigma) = \sin\left(p\pi\frac{S-a}{b-a}\right).$$

With these bounds and the initial condition, we can formulate the IBVP for our survival ch.f., i.e.,

$$\begin{cases} -\frac{\partial\hat{\phi}_p}{\partial\tau} + \frac{1}{2}\sigma_\tau^2 S_\tau^{2\beta}\frac{\partial^2\hat{\phi}_p}{\partial S^2} + \frac{1}{2}\alpha^2\sigma_\tau^2\frac{\partial^2\hat{\phi}_p}{\partial\sigma^2} + \alpha\sigma_\tau^2 S_\tau^\beta\rho\frac{\partial^2\hat{\phi}_p}{\partial S\partial\sigma} = 0, & 0 \le \tau \le T,\ a < S < b,\ \sigma > 0 \\ \hat{\phi}_p(\tau,a,\sigma) = 0, & 0 \le \tau \le T, \sigma > 0 \\ \hat{\phi}_p(\tau,b,\sigma) = 0, & 0 \le \tau \le T, \sigma > 0 \\ \frac{\partial\hat{\phi}_p}{\partial\tau}(\tau,S,0) = 0, & 0 \le \tau \le T, a < S < b \\ \hat{\phi}_p(0,S,\sigma) = \sin\left(p\pi\frac{S-a}{b-a}\right), & a < S_T < b, \sigma > 0. \end{cases} \tag{6.5}$$

6.3. Approximating the survival characteristic function

The final step is to construct the trigonometric expansion for the survival ch.f.. In Section 5.3 we already decided on the expansions for τ and S, so we only need to determine this for the volatility σ.

6.3.1. Expansion for volatility-dimension

Since the volatility is not bounded from above, we should choose an upper bound b_3 such that the probability $p(\sigma > b_3) \approx 0$. Note that a large value for b_3 leads to a large domain, which would require more training points.

For the volatility dimension, we have one boundary condition: $\frac{\partial \hat{\phi}_p}{\partial \tau}(\tau, S, 0) = 0$. Note that taking the derivative with respect to the time dimension would consider the volatility dimension as a constant, which holds for all values of the volatility. Therefore, we can treat it as a Dirichlet boundary condition. We choose Fourier-sine expansion with boundary relaxation for the upper boundary.

Our approximation for the volatility dimension reads

$$f_3(\sigma) \approx \sum_{k_3=1}^{K} A_{k_3} \sin\left(k_3\pi \frac{\sigma - a_3}{b_3 - a_3 + \epsilon}\right).$$

6.3.2. Trigonometric expansion of the survival characteristic function

We can now expand the time derivative of $\hat{\phi}_p$ as a 3D Fourier series to find

$$\frac{\partial \hat{\phi}_p}{\partial \tau}(\tau, S, \sigma) \approx \sum_{k_1=1}^{K} {\sum_{k_2=0}^{K-1}}' \sum_{k_3=1}^{K} \mathcal{A}_{k_1,k_2,k_3} \cos\left(k_2\pi \frac{\tau - a_2}{b_2 - a_2}\right) \sin\left(k_1\pi \frac{S - a_1}{b_1 - a_1}\right) \sin\left(k_3\pi \frac{\sigma - a_3}{b_3 - a_3 + \epsilon}\right).$$

Then using the fundamental theorem of calculus to integrate this with respect to time, we find

$$\begin{aligned}\hat{\phi}_p(\tau, S, \sigma) \approx &\sum_{k_1=1}^{K} \sum_{k_2=1}^{K-1} \sum_{k_3=1}^{K} \mathcal{A}_{k_1,k_2,k_3} \frac{b_2 - a_2}{k_2\pi} \sin\left(k_2\pi \frac{\tau - a_2}{b_2 - a_2}\right) \sin\left(k_1\pi \frac{S - a_1}{b_1 - a_1}\right) \sin\left(k_3\pi \frac{\sigma - a_3}{b_3 - a_3 + \epsilon}\right) \\ &+ \sum_{k_1=1}^{K} \sum_{k_3=1}^{K} \mathcal{A}_{k_1,0,k_3} \frac{\tau}{2} \sin\left(k_1\pi \frac{S - a_1}{b_1 - a_1}\right) \sin\left(k_3\pi \frac{\sigma - a_3}{b_3 - a_3 + \epsilon}\right) \\ &+ \sin\left(p\pi \frac{S - a_1}{b_1 - a_1}\right).\end{aligned} \tag{6.6}$$

Note that all boundary conditions and the initial condition are satisfied by this formulation. Therefore, we do not have to consider the boundary and initial conditions when training the model.

Next, we consider $M \times M \times M$ training points. τ_l, S_m, σ_n are the l-th time training point, m-th asset training point and n-th volatility training point respectively.

From an implementation point of view, we define matrices for which the rows are defined as

$$\begin{aligned}
\mathbf{V}_1[m,:] &= \left(\sin\left(\pi\frac{S_m - a_1}{b_1 - a_1}\right) \quad \sin\left(2\pi\frac{S_m - a_1}{b_1 - a_1}\right) \quad \cdots \quad \sin\left(N\pi\frac{S_m - a_1}{b_1 - a_1}\right)\right) \\
\mathbf{V}_2[l,:] &= \left(\tfrac{1}{2}\tau \quad \tfrac{b_2 - a_2}{\pi}\sin\left(\pi\frac{\tau_l - a_2}{b_2 - a_2}\right) \quad \cdots \quad \tfrac{b_2 - a_2}{(N-1)\pi}\sin\left((N-1)\pi\frac{\tau_l - a_2}{b_2 - a_2}\right)\right) \\
\mathbf{V}_3[n,:] &= \left(\sin\left(\pi\frac{\sigma_n - a_3}{b_3 - a_3 + \epsilon}\right) \quad \sin\left(2\pi\frac{\sigma_n - a_3}{b_3 - a_3 + \epsilon}\right) \quad \cdots \quad \sin\left(N\pi\frac{\sigma_n - a_3}{b_3 - a_3 + \epsilon}\right)\right) \\
\mathbf{Z}_1[m,:] &= \left(\tfrac{\pi}{b_1 - a_1}\cos\left(\pi\frac{S_m - a_1}{b_1 - a_1}\right) \quad \tfrac{2\pi}{b_1 - a_1}\cos\left(2\pi\frac{S_m - a_1}{b_1 - a_1}\right) \quad \cdots \quad \tfrac{N\pi}{b_1 - a_1}\cos\left(N\pi\frac{S_m - a_1}{b_1 - a_1}\right)\right) \\
\mathbf{Z}_2[l,:] &= \left(\tfrac{1}{2} \quad \cos\left(\pi\frac{\tau_l - a_2}{b_2 - a_2}\right) \quad \cdots \quad \cos\left((N-1)\pi\frac{\tau_l - a_2}{b_1 - a_2}\right)\right) \\
\mathbf{Z}_3[n,:] &= \left(\tfrac{\pi}{b_3 - a_3 + \epsilon}\cos\left(\pi\frac{\sigma_n - a_3}{b_3 - a_3 + \epsilon}\right) \quad \tfrac{2\pi}{b_3 - a_3 + \epsilon}\cos\left(2\pi\frac{\sigma_n - a_3}{b_3 - a_3 + \epsilon}\right) \quad \cdots \quad \tfrac{N\pi}{b_3 - a_3 + \epsilon}\cos\left(N\pi\frac{\sigma_n - a_3}{b_3 - a_3 + \epsilon}\right)\right) \\
\mathbf{Z}_4[m,:] &= -\left(\left(\tfrac{\pi}{b_1 - a_1}\right)^2\sin\left(\pi\frac{S_m - a_1}{b_1 - a_1}\right) \quad \left(\tfrac{2\pi}{b_1 - a_1}\right)^2\sin\left(2\pi\frac{S_m - a_1}{b_1 - a_1}\right) \quad \cdots \quad \left(\tfrac{N\pi}{b_1 - a_1}\right)^2\sin\left(N\pi\frac{S_m - a_1}{b_1 - a_1}\right)\right) \\
\mathbf{Z}_5[n,:] &= -\left(\left(\tfrac{\pi}{b_3 - a_3 + \epsilon}\right)^2\sin\left(\pi\frac{\sigma_n - a_3}{b_3 - a_3 + \epsilon}\right) \quad \left(\tfrac{2\pi}{b_3 - a_3 + \epsilon}\right)^2\sin\left(2\pi\frac{\sigma_n - a_3}{b_3 - a_3 + \epsilon}\right) \quad \cdots \quad \left(\tfrac{N\pi}{b_3 - a_3 + \epsilon}\right)^2\sin\left(N\pi\frac{\sigma_n - a_3}{b_3 - a_3 + \epsilon}\right)\right).
\end{aligned}$$

These matrices correspond to the sine and cosine expansion terms for $\hat{\phi}_p$ and its partial derivatives.
Similar to the GBM case we can use these matrix representations combined with a vectorized $\mathcal{A}$. We plug this into the PDE for $\hat{\phi}_p$ and use

$$\sum_{k_1=1}^{K}\sum_{k_2=0}^{K-1}\sum_{k_3=1}^{K}\mathcal{A}[k_1,k_2,k_3]\left(\mathbf{Z}_2[l,k_2]\cdot\mathbf{V}_3[n,k_3]\cdot\mathbf{V}_1[m,k_1]\right)=\left(\mathbf{Z}_2[l,:]\otimes\mathbf{V}_3[n,:]\otimes\mathbf{V}_1[m,:]\right)\mathrm{vec}(\mathcal{A}).$$

Then the LHS of the equation becomes

$$\Bigg[-\Big(\mathbf{Z}_2[l,:]\otimes\mathbf{V}_3[n,:]\otimes\mathbf{V}_1[m,:]\Big)$$
$$+\mathbf{V}_2[l,:]\otimes\left(\frac{\sigma_n^2}{2}\mathbf{V}_3[n,:]\otimes S_m^{2\beta}\mathbf{Z}_4[m,:]+\alpha\rho\sigma_n^2\mathbf{Z}_3[n,:]\otimes S_m^{\beta}\mathbf{Z}_1[m,:]+\frac{\alpha^2\sigma_n^2}{2}\mathbf{Z}_5[n,:]\otimes\mathbf{V}_1[m,:]\right)\Bigg]\mathrm{vec}(\mathcal{A}),$$

where the vectorized $\mathrm{vec}(\mathcal{A})$ is created by first variating the k_1, then the k_3 and lastly k_2, so it looks like

$$\begin{pmatrix} A[:,k_2=0,k_3=1]\\ \vdots\\ A[:,k_2=0,k_3=N]\\ A[:,k_2=1,k_3=1]\\ \vdots\\ A[:,k_2=N-1,k_3=N]\end{pmatrix},$$

where $A[:,k_2,k_3]$ is the column vector with all values for k_1 and fixed k_2,k_3.
The RHS becomes

$$\frac{\sigma_n^2 S_m^{2\beta}}{2}\left(\frac{p\pi}{b_1-a_1}\right)^2\sin\left(p\pi\frac{S_m-a_1}{b_1-a_1}\right).$$

6.4. Results

The linear system derived in the previous section consists of a $\mathbb{R}^{K^3\times K^3}$ matrix and a $\mathbb{R}^{K^3}$ vector. The solution of the linear system is the $\mathbb{R}^{K^3}$ vector of expansion coefficients of the survival ch.f.. These large matrices quickly lead to memory issues. Due to these memory constraints, the largest possible K that could be used was $K=26$. The `scipy.linalg.solve()` function was used to solve the linear system. For $K=26$, computing the coefficients took a little over 1 hour.

Benchmark values

For GBM, we had a closed-form solution to use as a benchmark for the trigonometric expansion. However, for SABR, no such closed-form solution exists for barrier options (though [9] does provide a closed-form approximation for European options). Therefore, we use Monte Carlo simulation with 1000 timesteps and $1,000,000$ simulation paths as a benchmark. The convergence order for Monte Carlo simulation is known to be $O(\sqrt{n})$, with n the number of simulations. We use the Euler Forward scheme for the Monte Carlo simulation, which has a convergence rate of $O(\Delta t)$. Note that this time discretization also causes continuously monitored barrier options to be discretely monitored. There is no guarantee that this benchmark is sufficiently accurate. Therefore, we also use five points with $100,000,000$ simulations and $10,000$ timesteps. If the error at these points is better than with fewer timesteps and paths, the benchmark is not sufficiently accurate. The five sample points on which the errors are measured are $(S_0,\tau)=\{(90,0.5),(90,1),(105,0.1),(105,0.5),(115,0.9)\}$, providing a good mix of at-the-money, near-barrier and near the "highest price" cases. The errors related to these points are shown in Table 6.1.

Performance of trigonometric expansion under SABR

For these results, we use model parameters $(r,\alpha,\beta,\rho,\sigma_0)=(0.1,0.2,0.5,-0.85,0.8)$, and for the option and expansion parameters, we use $(a_1, b_1, E, a_3, b_3, K) = (40, 120, 90, 0, 2, 26)$. The results are shown in Figure 6.1. In most cases, the relative error is under 1%. However, near the barrier, the relative error increases, similar to the behaviour observed in Chapter 5. Initial asset prices slightly below the strike price also incur a large relative error, but the absolute error does not increase, indicating that the higher relative error comes from the lower benchmark price.

Next, we examine Table 6.1 to determine whether these errors with more accurate benchmark are lower than those shown in the figure. For all sample points, the errors are nearly identical, suggesting that the Monte Carlo benchmark with 1,000 time steps and 1,000,000 paths is sufficiently accurate as a benchmark for this method.

Point	Absolute Error	Relative Error
$(90, 0.5)$	$4.44 \cdot 10^{-3}$	$4.94 \cdot 10^{-5}$
$(90, 1)$	$1.15 \cdot 10^{-2}$	$1.28 \cdot 10^{-4}$
$(105, 0.1)$	$2.97 \cdot 10^{-2}$	$2.8 \cdot 10^{-4}$
$(105, 0.5)$	$4.19 \cdot 10^{-2}$	$3.99 \cdot 10^{-4}$
$(115, 0.9)$	$1.32 \cdot 10^{-1}$	$1.11 \cdot 10^{-3}$

Table 6.1: Absolute and relative errors of trigonometric expansion with $K = 26$

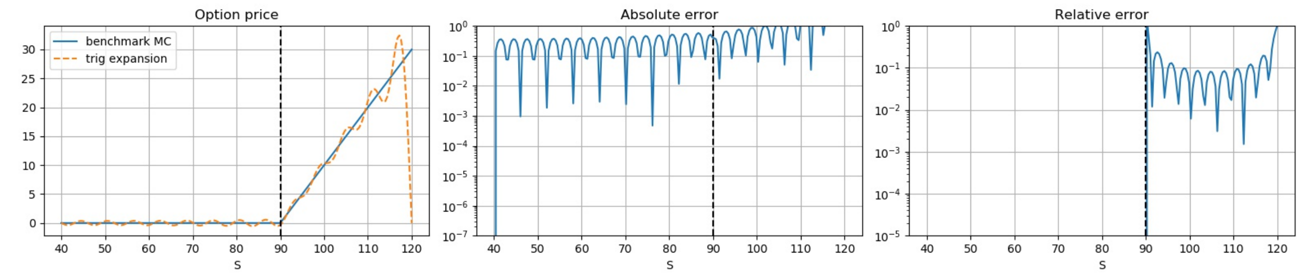
Comparing performance to closed-form approximations

We also compare our model to results from Yang et al. [13], where up-and-out put options are priced for small maturity times $T = \frac{1}{12}$, $T = \frac{1}{52}$ and $T = \frac{1}{252}$. They used parameters and initial values $(S_0, \sigma_0, \beta, E, barrier) = (100, 0.1, 0.9, 100, 103)$, and used various values of α and ρ. For our pricing with the trigonometric expansion, we use $K = 26$ and we choose our model bounds to be $[90, 103]$ for the asset, $[0, 1/11]$ for time to maturity and $[0, 1]$ for volatility. The option price in [13] is given as the expectation of the payoff without discounting, so risk-free interest rate r is set to 0. For the results of this paper and our model, a Monte Carlo simulation with 1,000,000 samples and 25,200 timesteps per year is used as benchmark. In Table 6.2 these results are summarized. MC denotes the Monte Carlo benchmark, Yang denotes absolute relative error of the prices from [13] and Trig is the absolute relative error of option prices using our model. The absolute relative error is calculated using $\frac{|appr.-MC|}{MC}$.

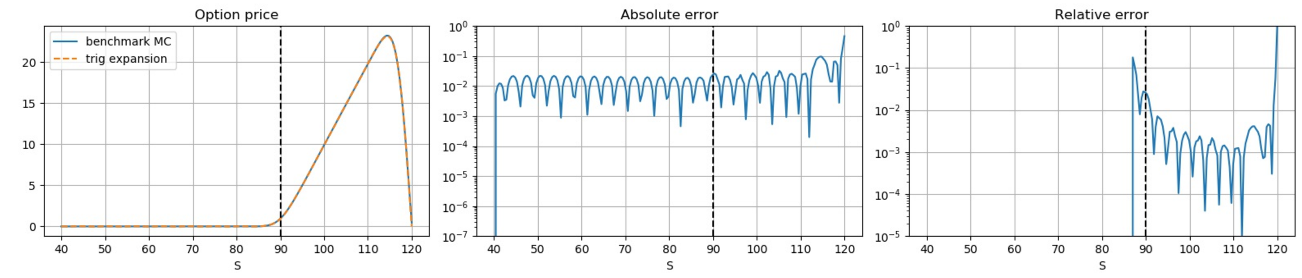
Note that Yang et al. [13] and our model have similar performance for $\tau = \frac{1}{12}$, but for the smaller times to maturity, our models errors are larger. This supports our earlier findings that for small maturities, our method suffers from the Gibbs phenomenon at $\tau = 0$. We should also consider that $\tau = \frac{1}{252}$ is less than $\frac{1}{20}$ of the expansion domain in the time dimension. If we were to take a smaller expansion domain for the time dimension, we can achieve higher accuracy, but this does require us to train multiple models.

	$\tau = \frac{1}{252}$			$\tau = \frac{1}{52}$			$\tau = \frac{1}{12}$		
(ρ, α)	MC	Yang	Trig.	MC	Yang	Trig.	MC	Yang	Trig.
(0,0.1)	0.1570	0.79%	6.15%	0.3454	0.58%	1.00%	0.7230	0.57%	0.54%
(0,0.3)	0.1568	0.91%	6.01%	0.3460	0.40%	0.80%	0.7217	0.38%	0.65%
(0,0.5)	0.1569	0.82%	6.07%	0.3463	0.31%	0.71%	0.7257	0.93%	0.09%
(-0.1,0.1)	0.1570	0.78%	6.15%	0.3457	0.48%	0.89%	0.7233	0.60%	0.39%
(-0.3,0.1)	0.1573	0.61%	6.33%	0.3470	0.09%	0.46%	0.7213	0.32%	0.64%
(-0.5,0.1)	0.1572	0.65%	6.10%	0.3441	0.95%	1.31%	0.7205	0.21%	0.75%

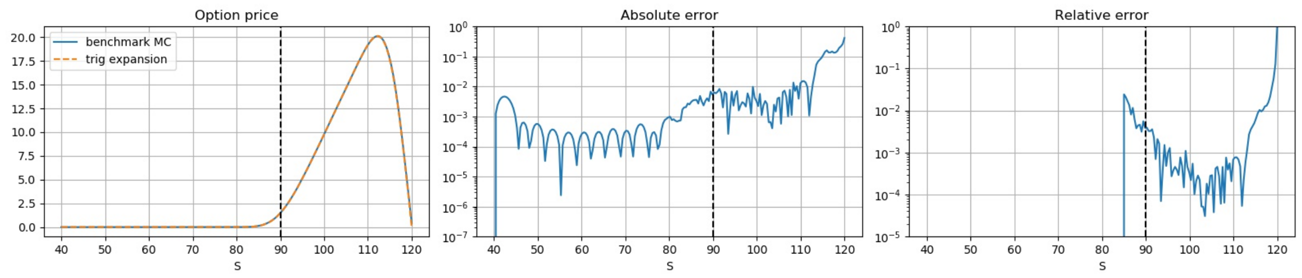
Table 6.2: Comparisons of up-and-out option prices obtained with Monte Carlo, a Closed-Form approximation [13] and the trigonometric expansion.



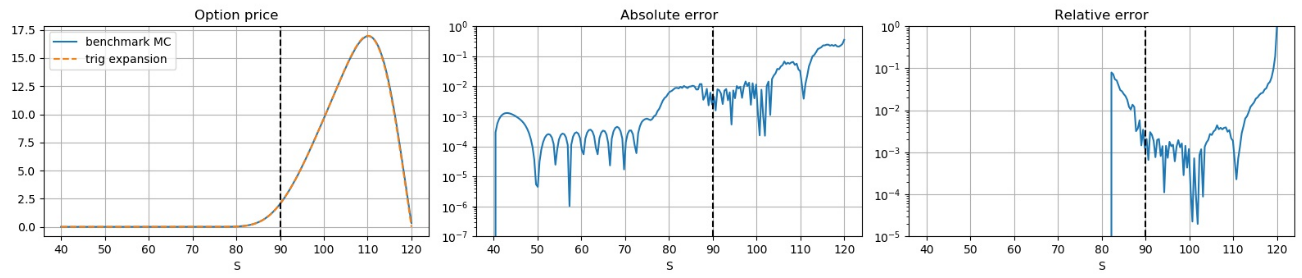
(a) Option price, absolute error and relative error for $\tau = 0$



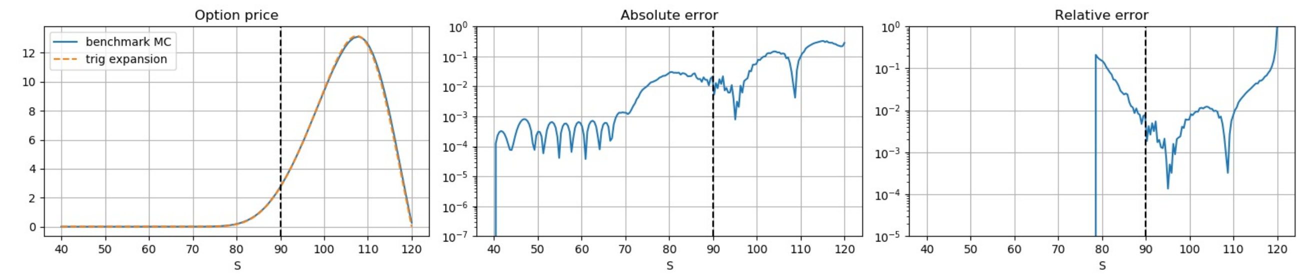
(b) Option price, absolute error and relative error for $\tau = 0.1$



(c) Option price, absolute error and relative error for $\tau = 0.25$



(d) Option price, absolute error and relative error for $\tau = 0.5$



(e) Option price, absolute error and relative error for $\tau = 1$

Figure 6.1: Option price and errors for various τ using trigonometric expansion vs Monte Carlo benchmark for an up-and-out barrier option with parameters $(r, \alpha, \beta, \rho, \sigma_0) = (0.1, 0.2, 0.5, -0.85, 0.8)$ and $(a_1, b_1, E, a_3, b_3, K) = (40, 120, 90, 0, 2, 26)$ under SABR regular asset and regular volatility dynamics.

6.5. Model improvements through change of variables

For GBM we have seen that change of variables can improve the accuracy of our method. First in Section 5.5 we found that moving to the asset price domain improves the accuracy of our method. In Section 5.6 we go one step further in trying to cover more of the concentrated density mass near the barrier via change of variables.

6.5.1. Log-volatility

The SABR model introduces the stochastic volatility dimension. From the definition of the dynamics, $\sigma \geq 0$, but we do not have an upper bound for the volatility. In practice, the volatility is usually under 100%. It can, however, stay relatively small. Then equidistant training points between 0 and b_3 might not sufficiently capture the main mass of the density function in volatility. A solution is transforming the volatility domain to a log-volatility domain. $x = \ln(\sigma + \lambda_{vol})$, where $\lambda_{vol} > 0$ influences how much the training points skewed to the lower boundary of the volatility. Large λ_{vol} behaves the same as the regular volatility domain and small λ_{vol} corresponds to a very skewed distribution of training points.

Transformed PDE and linear system

We use transformation of variables $x = \ln(\sigma + \lambda_{vol})$, so $\frac{dx}{d\sigma} = \frac{1}{\sigma+\lambda_{vol}}$. The partial derivatives of V are changed to:

$$\begin{aligned}
\frac{\partial V}{\partial \sigma} &= \frac{\partial V}{\partial x}\frac{\partial x}{\partial \sigma} \\
&= \frac{1}{\sigma + \lambda_{vol}}\frac{\partial V}{\partial x} \\
\frac{\partial^2 V}{\partial \sigma^2} &= \frac{\partial}{\partial \sigma}\left(\frac{1}{\sigma + \lambda_{vol}}\frac{\partial V}{\partial x}\right) \\
&= \frac{1}{(\sigma + \lambda_{vol})^2}\frac{\partial^2 V}{\partial x^2} - \frac{1}{(\sigma + \lambda_{vol})^2}\frac{\partial V}{\partial x} \\
\frac{\partial V}{\partial S \partial \sigma} &= \frac{1}{\sigma + \lambda_{vol}}\frac{\partial V}{\partial S \partial x}.
\end{aligned}$$

This yields the PDE of the log-volatility SABR model

$$\begin{aligned}
&\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^{2\beta}\frac{\partial^2 V}{\partial S^2} + \frac{1}{2}\alpha^2\sigma^2\frac{\partial^2 V}{\partial \sigma^2} + \alpha\sigma^2 S^{\beta}\rho\frac{\partial^2 V}{\partial S \partial \sigma} - rV = 0 \\
&\frac{\partial V}{\partial t} + \frac{(e^x - \lambda_{vol})^2}{2}S^{2\beta}\frac{\partial^2 V}{\partial S^2} + \frac{\alpha^2}{2}\frac{(e^x - \lambda_{vol})^2}{e^{2x}}\left(\frac{\partial^2 V}{\partial x^2} - \frac{\partial V}{\partial x}\right) + \alpha\frac{(e^x - \lambda_{vol})^2}{e^x}S^{\beta}\rho\frac{\partial V}{\partial S \partial x} - rV = 0 \qquad (6.7)
\end{aligned}$$

Our boundary conditions change to

$$\begin{cases}
V(t,a,x) = 0 \\
V(t,b,x) = 0 \\
\frac{\partial V}{\partial t}\Big|_{x=-\infty} - rV(t,S,-\infty) = 0 \\
V(T,S,x) = v(S_T),
\end{cases}$$

Our approximation $V_2(t,S,x)$ is identical to the regular volatility case. The analytic evaluation of coefficients V_k has not changed either. Therefore, we can plug this approximation into our PDE to obtain the PDE for the survival ch.f., following the same reasoning as before:

$$\begin{cases}
-\frac{\partial \hat{\phi}_p}{\partial \tau} + \frac{(e^x-\lambda_{vol})^2}{2}S^{2\beta}\frac{\partial^2 \hat{\phi}_p}{\partial S^2} + \frac{\alpha^2}{2}\frac{(e^x-\lambda_{vol})^2}{e^{2x}}\left(\frac{\partial^2 \hat{\phi}_p}{\partial x^2} - \frac{\partial \hat{\phi}_p}{\partial x}\right) + \alpha\frac{(e^x-\lambda_{vol})^2}{e^x}S^{\beta}\rho\frac{\partial \hat{\phi}_p}{\partial S \partial x} = 0 \\
\hat{\phi}_p(\tau,a,x) = 0 \\
\hat{\phi}_p(\tau,b,x) = 0 \\
\frac{\partial \hat{\phi}_p}{\partial \tau}\Big|_{x=-\infty} = 0 \\
\hat{\phi}_p(0,S,x) = \sin\left(p\pi\frac{S-a}{b-a}\right),
\end{cases}$$

Trigonometric expansion

For the time-to-maturity and asset-price dimensions we keep the same expansions as in Chapter 5. For x the bounds change compared to what we had before. Before, we clearly had a bound $a_3 = 0$ where we wanted the function to attain the value 0. Therefore, the sine-expansion was a logical choice. If we keep sine expansion on $[a_3, b_3 + \epsilon]$, we find LHS

$$\Bigg[-\Big(\mathbf{Z}_2[l,:] \otimes \mathbf{V}_3[n,:] \otimes \mathbf{V}_1[m,:]\Big) + \mathbf{V}_2[l,:] \otimes \Bigg(\frac{(e^{x_n} - \lambda_{vol})^2}{2}\mathbf{V}_3[n,:] \otimes S_m^{2\beta}\mathbf{Z}_4[m,:]$$

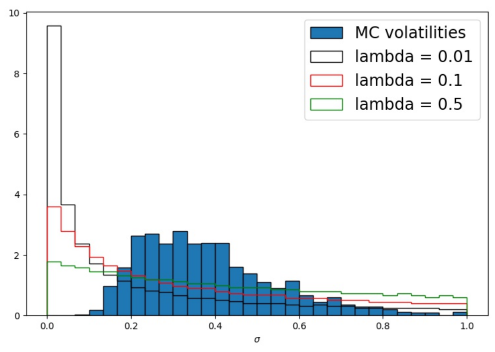
$$+ \alpha\rho\frac{(e^{x_n} - \lambda_{vol})^2}{e^{x_n}}\mathbf{Z}_3[n,:] \otimes S_m^{\beta}\mathbf{Z}_1[m,:] + \frac{\alpha^2}{2}\frac{(e^{x_n} - \lambda_{vol})^2}{e^{2x_m}}\Big(\mathbf{Z}_5[n,:] - \mathbf{Z}_3[n,:]\Big) \otimes \mathbf{V}_1[m,:]\Bigg)\Bigg]\text{vec}(\mathcal{A}),$$

and RHS

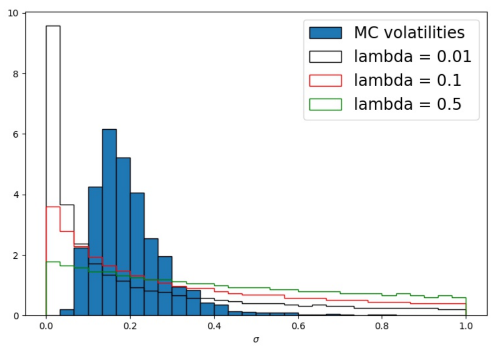
$$\frac{(e^{x_n} - \lambda_{vol})^2}{2}S^{2\beta}\left(\frac{p\pi}{b_1 - a_1}\right)^2 \sin\left(p\pi\frac{S_m - a_1}{b_1 - a_1}\right).$$

Choice of λ_{vol}

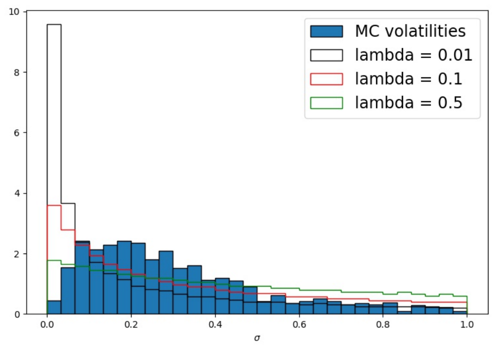
Before we analyse the results of the transformed model, we should choose an appropriate value for λ_{vol}. We use Monte Carlo simulation to find an empirical probability density for the volatility. Then by comparing some values of λ_{vol}, we can choose the optimal value. As mentioned before, taking log-volatility would likely be most useful for smaller values of σ_0. Other parameters that will greatly influence the distribution of the volatility are the volatility of volatility α and the time to maturity τ.
Figure 6.2 shows the distribution of σ after running a Monte Carlo simulation with 1000 time steps and 50,000 paths. We can clearly see that a smaller τ results in less spread of the values and a larger cluster of values close to the initial volatility σ_0. A smaller initial volatility value obviously shifts the distribution towards smaller values, but also seems to yield slightly less spread of the points. Lastly, an increase in the volatility of volatility α seems to increase the spread somewhat and slightly shifts the points towards smaller values of σ.
For low values of σ_0, a lower value of λ_{vol} would be a great fit. For these examples, however, $\lambda_{vol} = 0.01$ seems to be the worst fit of the 3. In plots 6.2a and 6.2d, $\lambda_{vol} = 0.5$ seems to be the best fit by a small margin, whereas $\lambda_{vol} = 0.1$ seems to be better for 6.2b and 6.2c.



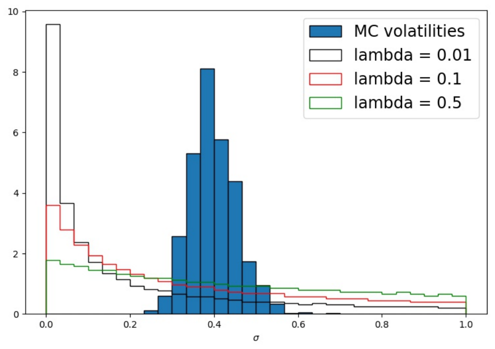
(a) $\sigma_0 = 0.4$, $\alpha = 0.3$ and $\tau = 1$



(b) $\sigma_0 = 0.2$, $\alpha = 0.3$ and $\tau = 1$



(c) $\sigma_0 = 0.4$, $\alpha = 0.6$ and $\tau = 1$

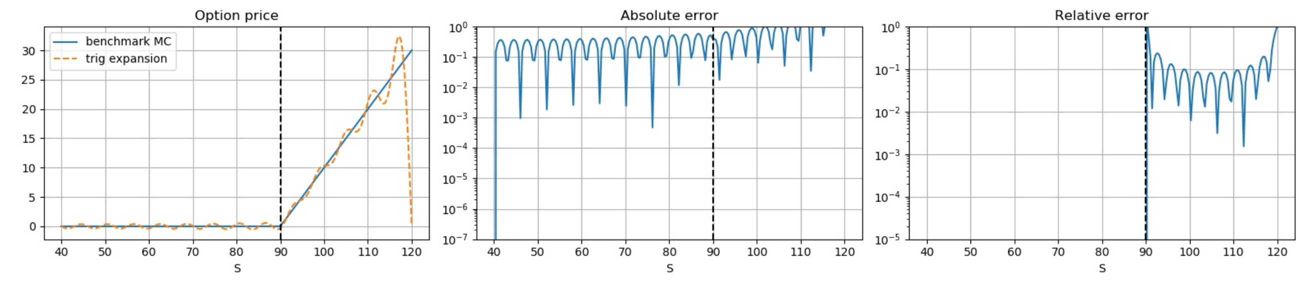


(d) $\sigma_0 = 0.4$, $\alpha = 0.3$ and $\tau = 0.1$

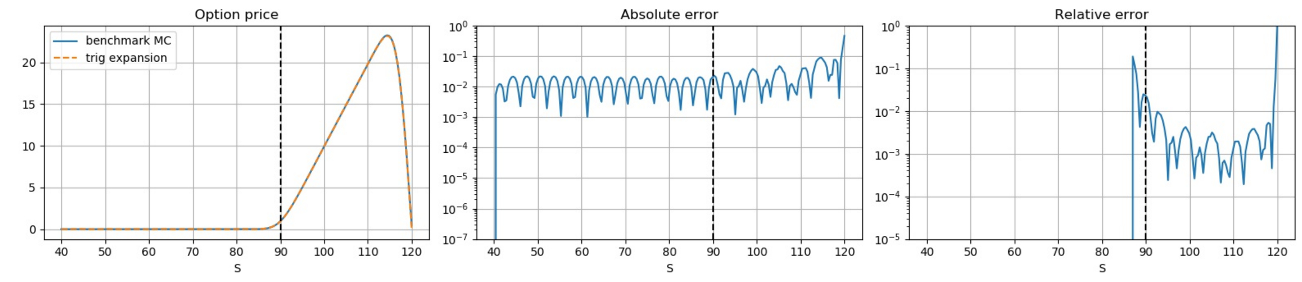
Figure 6.2: Behaviour of the empirical probability density of volatility under SABR when τ, σ_0 or α changes.

Results

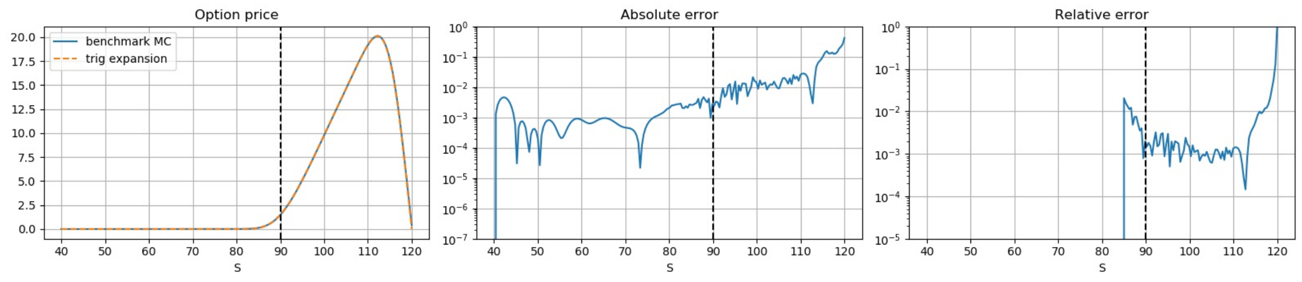
Figure 6.3 shows the results for $\lambda_{vol} = 0.5$. For shorter maturities, we seem to have slightly better performance, especially for the asset prices in the money. For longer maturities, the error levels seems to be similar to the original one before the log-transformation in volatility. In general, this looks like a slight improvement, but we should note that we gain a lot of freedom using this transformation. With $\lambda_{vol} \longrightarrow \infty$, this acts like the regular volatility SABR model, so we can always choose λ_{vol} such that the performance does not decrease.



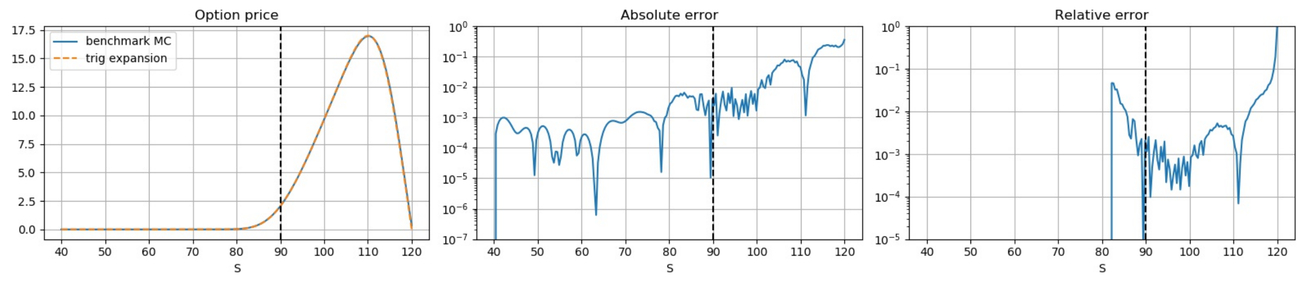
(a) Option price, absolute error and relative error for $\tau = 0$



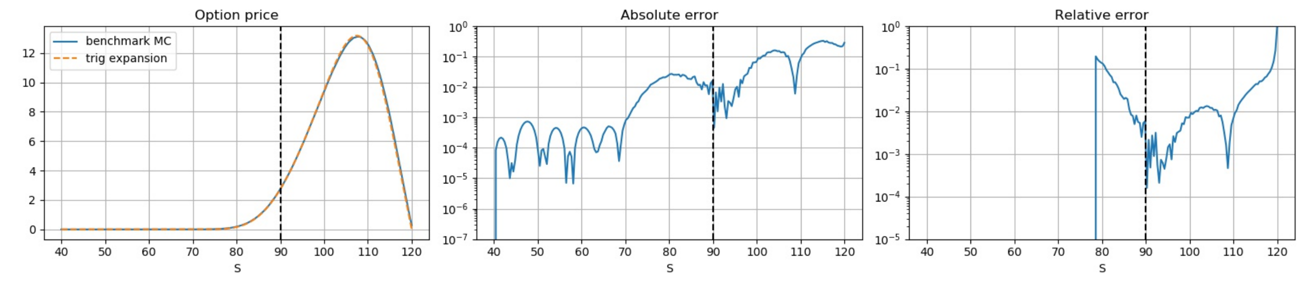
(b) Option price, absolute error and relative error for $\tau = 0.1$



(c) Option price, absolute error and relative error for $\tau = 0.25$



(d) Option price, absolute error and relative error for $\tau = 0.5$



(e) Option price, absolute error and relative error for $\tau = 1$

Figure 6.3: Option price and errors for various τ using trigonometric expansion vs Monte Carlo benchmark for an up-and-out barrier option with parameters $(r, \alpha, \beta, \rho, \sigma_0) = (0.1, 0.2, 0.5, -0.85, 0.8)$ and $(a_1, b_1, E, a_3, b_3, K) = (40, 120, 90, 0, 2, 26)$ under SABR regular asset and log-volatility dynamics with $\lambda_{vol} = 0.5$.

6.5.2. Inverted-log-asset

Besides the log-volatility, we also apply the inverted-log-asset transformation, $y = \ln(B + \lambda_{asset} - S)$. Here λ_{asset} can again be chosen using Equation (5.11). Since $\lambda_{vol} = 1$ (so $p \approx 0.845$) worked well for the GBM case, it thus makes sense to use it in the SABR model as well. It is, however, noteworthy that we now have significantly less expansion terms (26 vs 64), so there are only 4 training points to the left of the midway point instead of the 10 training points we had with GBM. Therefore, we also test for $p = 0.75$, which yields $\lambda_{asset} \approx 7.5$.

Transformed PDE and linear system

The transformation is identical to that for the GBM case in Section 5.6.3, which, after being applied to PDE (6.7), result in the following PDE

$$\begin{aligned}\frac{\partial V}{\partial t} &+ \frac{(e^x - \lambda_{vol})^2}{2}(B + \lambda_{asset} - e^y)^{2\beta} e^{-2y}\left(\frac{\partial^2 V}{\partial y^2} - \frac{\partial V}{\partial y}\right) - \alpha\frac{(e^x - \lambda_{vol})^2}{e^x}(B + \lambda_{asset} - e^y)^{\beta}\rho e^{-y}\frac{\partial V}{\partial y \partial x} \\ &+ \frac{\alpha^2}{2}\frac{(e^x - \lambda_{vol})^2}{e^{2x}}\left(\frac{\partial^2 V}{\partial x^2} - \frac{\partial V}{\partial x}\right) - rV = 0,\end{aligned} \tag{6.8}$$

where B is the upper barrier ($= b_{asset}$). We then find the IBVP for the survival ch.f. to be

$$\begin{cases} -\frac{\partial \hat{\phi}_p}{\partial \tau} + \frac{(e^x - \lambda_{vol})^2}{2}(B + \lambda_{asset} - e^y)^{2\beta} e^{-2y}\left(\frac{\partial^2 \hat{\phi}_p}{\partial y^2} - \frac{\partial \hat{\phi}_p}{\partial y}\right) \\ \quad -\alpha\frac{(e^x - \lambda_{vol})^2}{e^x}(B + \lambda_{asset} - e^y)^{\beta}\rho e^{-y}\frac{\partial \hat{\phi}_p}{\partial y \partial x} + \frac{\alpha^2}{2}\frac{(e^x - \lambda_{vol})^2}{e^{2x}}\left(\frac{\partial^2 \hat{\phi}_p}{\partial x^2} - \frac{\partial \hat{\phi}_p}{\partial x}\right) = 0, \\ \hat{\phi}_p(\tau, a, x) = 0, \\ \hat{\phi}_p(\tau, b, x) = 0, \\ \frac{\partial \hat{\phi}_p}{\partial \tau}\Big|_{x=-\infty} = 0, \\ \hat{\phi}_p(0, y, x) = \sin\left(p\pi\frac{y-a}{b-a}\right), \end{cases} \tag{6.9}$$

where $a = \ln(B + \lambda - b_{asset}), b = \ln(B + \lambda - a_{asset})$. We have seen in the GBM case, that the expansions can be kept the same. Then the linear system will have LHS matrix defined as

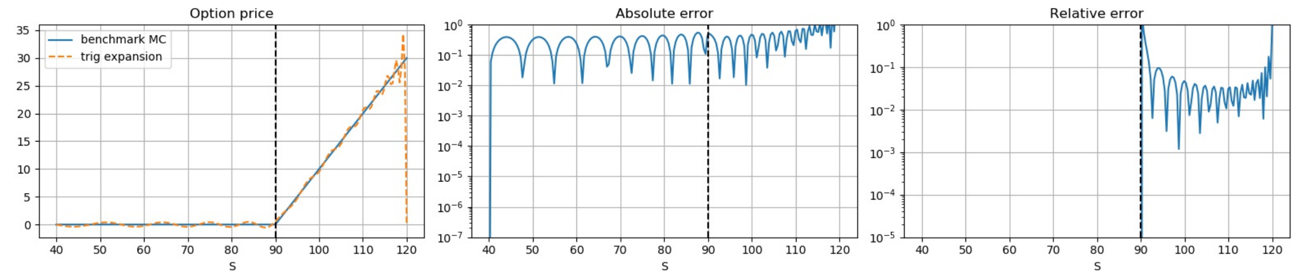
$$\begin{aligned}\Bigg[&-\Big(\mathbf{Z}_2[l,:] \otimes \mathbf{V}_3[n,:] \otimes \mathbf{V}_1[m,:]\Big) \\ &+ \mathbf{V}_2[l,:] \otimes \Bigg(\frac{(e^{x_n} - \lambda_{vol})^2}{2}\mathbf{V}_3[n,:] \otimes (B + \lambda_{asset} - e^{y_m})^{2\beta} e^{-2y_m}\Big(\mathbf{Z}_4[m,:] - \mathbf{Z}_1[m,:]\Big) \\ &- \alpha\rho\frac{(e^{x_n} - \lambda_{vol})^2}{e^{x_n}}\mathbf{Z}_3[n,:] \otimes (B + \lambda_{asset} - e^{y_m})^{\beta} e^{-y_m}\mathbf{Z}_1[m,:] \\ &+ \frac{\alpha^2}{2}\frac{(e^{x_n} - \lambda_{vol})^2}{e^{2x_n}}\Big(\mathbf{Z}_5[n,:] - \mathbf{Z}_3[n,:]\Big) \otimes \mathbf{V}_1[m,:]\Bigg)\Bigg]\text{vec}(\mathcal{A}),\end{aligned}$$

and RHS vector as

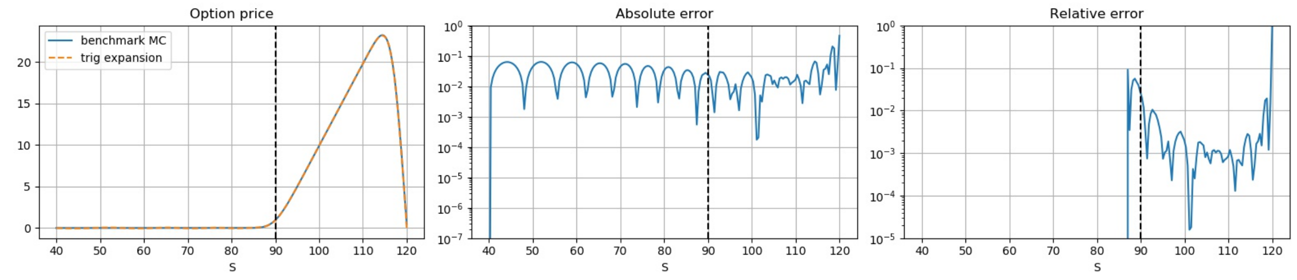
$$\frac{(e^{x_n} - \lambda_{vol})^2}{2}(B + \lambda_{asset} - e^{y_m})^{2\beta} e^{-2y_m}\left[\left(\frac{p\pi}{b_1 - a_1}\right)^2 \sin\left(p\pi\frac{y_m - a_1}{b_1 - a_1}\right) + \left(\frac{p\pi}{b_1 - a_1}\right)\cos\left(p\pi\frac{y_m - a_1}{b_1 - a_1}\right)\right].$$

results

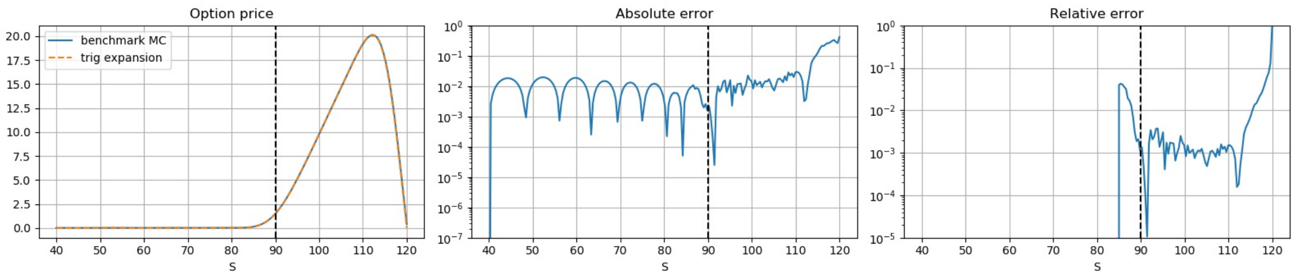
Numerical tests were conducted for both values of $\lambda_{asset} \in \{1, 7.5\}$ and $\lambda_{asset} = 7.5$ seems to perform slightly better. If a larger number of expansion terms becomes possible, $\lambda_{asset} = 1$ should be investigated again. The results with $\lambda_{asset} = 7.5$ are provided in Figure 6.4. There seems to be a slight improvement, but it is negligible compared to the improvement we saw for GBM in Section 5.6. Since this change of variables is based on having training points at denser parts of the probability density, it is possible that this improvement is more noticeable for a larger number of expansion points.



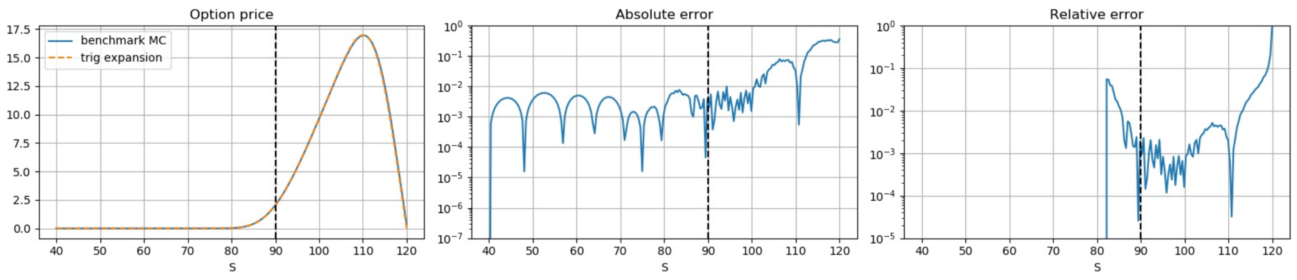
(a) Option price, absolute error and relative error for $\tau = 0$



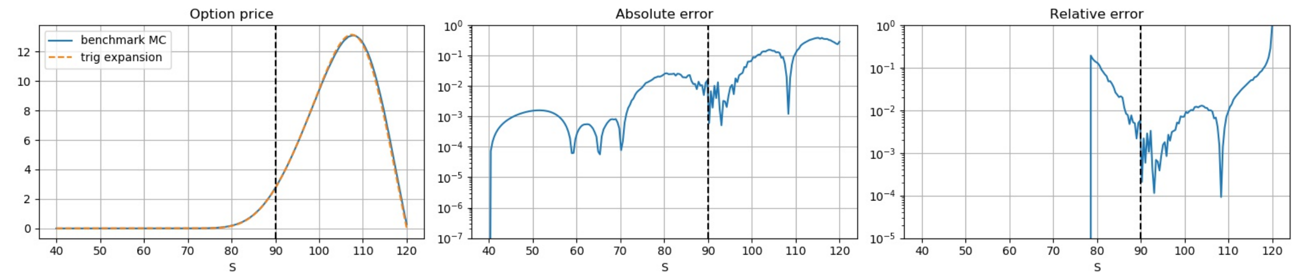
(b) Option price, absolute error and relative error for $\tau = 0.1$



(c) Option price, absolute error and relative error for $\tau = 0.25$



(d) Option price, absolute error and relative error for $\tau = 0.5$



(e) Option price, absolute error and relative error for $\tau = 1$

Figure 6.4: Option price and errors for various τ using trigonometric expansion vs Monte Carlo benchmark for an up-and-out barrier option with parameters $(r, \alpha, \beta, \rho, \sigma_0) = (0.1, 0.2, 0.5, -0.85, 0.8)$ and $(a_1, b_1, E, a_3, b_3, K) = (40, 120, 90, 0, 2, 26)$ under SABR inverted-log-asset and log-volatility dynamics with $\lambda_{asset} = 7.5, \lambda_{vol} = 0.5$.

6.6. Conclusion

By applying the same method as in Chapter 5, we approximated $\hat{\phi}_p$ with trigonometric expansion. Substituting this expansion into the PDE yielded a large linear system. Since SABR considers expansion in three dimensions, compared to the two dimensional case of GBM, the memory required to solve the linear system increased dramatically. The largest possible number of expansion points was $K = 26$. This curse of dimensionality is a major problem for this method.

The results from this method, however, do seem promising if this curse of dimensionality is tackled. We have seen that the error level and behaviour are comparable to those for GBM. For options with a very small time to maturity, the errors are relatively large. This is likely due to the Gibbs phenomenon close to $\tau = 0$. Compared to the closed-form approximation in [13], our errors are very similar for larger maturities. It is important to note, however, that our training took over an hour for each set of model parameters, whereas the closed form approximation only had to compute a one dimensional integral and therefore had a computation time of under 1 second. Theory tells us that we should converge to smaller errors for a larger number of expansion points.

For SABR, the variable transformations did not improve the accuracy much. A possible explanation for this is that we do not have enough expansion points. For a larger number of expansion points, this variable transformation can still improve the results, but this requires more investigation.

In general, the results look promising if we are able to use dimension reduction techniques to increase the number of expansion points per dimension.

7 Our contribution 4: Pricing barrier options under GBM using the COS-CPD network

The linear system that we need to solve in the training step consists of an $K^N \times K^N$ matrix and a K^N vector. For $N = 3$ in Chapter 6 we already saw that this gave memory issues. In this and the next chapters, we apply the COS-CPD network that we detailed in Section 3.4 to solve option pricing PDEs. The aim is to greatly alleviate the curse of dimensionality inherited in the trigonometric expansion method developed in Chapters 5 and 6.

In this chapter, we focus on the 2-dimensional expansion under GBM, as a testing case, due to the existence of analytical solutions under GBM. If this method proves to be efficient, the dimensionality can be increased to solve the expansion under SABR and further improvements to the method can be explored, when memory is less of an issue.

We start in Section 7.1 with the PDE for the survival ch.f. as a preparation step. In Section 7.2 we apply CPD to find a new formulation for the survival ch.f. approximation. This formulation is then inserted into the PDE, leading to multiple representations of the PDE in Section 7.3. In Section 7.4 we elaborate on the ALS as a means to train the model. Section 7.5 performs an error analysis, before we discuss the results in Section 7.6.

Lastly, we will also try a different approach, directly approximating the option price instead of the survival ch.f., resulting in a much simpler model if successful. This approach is discussed and tested in Section 7.7.

7.1. Characteristic function PDE under GBM

If we look at Equations (5.5), (5.6), and (5.12) we see that the PDE under GBM can be written as

$$-\frac{\partial \hat{\phi}_p}{\partial \tau} + c_1(S,\tau,r,\sigma^2)\frac{\partial \hat{\phi}_p}{\partial S} + c_2(S,\tau,r,\sigma^2)\frac{\partial^2 \hat{\phi}_p}{\partial S^2} = 0, \tag{7.1}$$

where c_1 and c_2 depend on the type of transformation used. These are defined as

$$c_1(S,\tau,r,\sigma^2) = \begin{cases} rS, & \text{Regular-asset} \\ r - \frac{\sigma^2}{2}, & \text{Log-asset} \\ -\frac{(B+\lambda-e^S)}{e^S}\left(r + (B+\lambda-e^S)\frac{\sigma^2}{2}e^{-S}\right), & \text{Inverted-log-asset} \end{cases}$$

$$c_2(S,\tau,r,\sigma^2) = \begin{cases} \frac{\sigma^2 S^2}{2}, & \text{Regular-asset} \\ \frac{1}{2}\sigma^2, & \text{Log-asset} \\ \frac{(B+\lambda-e^S)^2\sigma^2 e^{-2S}}{2}, & \text{Inverted-log-asset.} \end{cases}$$

Note that S represents the log-asset or inverted-log-asset price in the respective models. The inverted-log-asset exhibited bad convergence in combination with CPD and will therefore not be used in this chapter. As seen in Chapter 5, working on the asset domain directly has better properties than the in log-asset domain, so this chapter considers S as the asset price. It is worth investigating why CPD and the inverted-log-asset transformation do not seem compatible.

7.2. Approximating the ch.f. with CPD

We start from Equation (5.7), which is the trigonometric expansion for $\hat{\phi}_p(S,\tau)$, which is in fact the function that defines the Fourier-cosine series expansion coefficients of the survival density function, but is referred to as survival ch.f. for its close relation with the survival ch.f.:

$$\begin{aligned} T_K\hat{\phi}_p(\tau,S) &= \sum_{k_1=1}^{K}\sum_{k_2=1}^{K-1}\mathcal{A}_{k_1,k_2}\frac{b_2-a_2}{k_2\pi}\sin\left(k_2\pi\frac{\tau-a_2}{b_2-a_2}\right)\sin\left(k_1\pi\frac{S-a_1}{b_1-a_1}\right) \\ &\quad+\sum_{k_1=1}^{K}\mathcal{A}_{k_1,0}\frac{\tau}{2}\sin\left(k_1\pi\frac{S-a_1}{b_1-a_1}\right)+\sin\left(p\pi\frac{S-a_1}{b_1-a_1}\right),\quad a_2=0. \end{aligned}$$

We can rewrite this as

$$T_K\hat{\phi}_p(\tau,S)=\sum_{k_1=1}^{K}\sum_{k_2=0}^{K-1}\Big(\mathcal{A}_{k_1,k_2}\mathbf{s_1}(x_1)[k_1]\cdot\mathbf{s_2}(x_2)[k_2]\Big)+\sin\left(p\pi\frac{S-a_1}{b_1-a_1}\right),\quad a_2=0, \tag{7.2}$$

where $\mathbf{x}=(x_1,x_2)=(S,\tau)$ and

$$\mathbf{s_1}(x_1)=\begin{pmatrix}\sin\left(1\pi\frac{x_1-a_1}{b_1-a_1}\right)\\ \vdots\\ \sin\left(k_1\pi\frac{x_1-a_1}{b_1-a_1}\right)\\ \vdots\\ \sin\left(K\pi\frac{x_1-a_1}{b_1-a_1}\right)\end{pmatrix},\quad \mathbf{s_2}(x_2)=\begin{pmatrix}\frac{x_2}{2}\\ \frac{b_2-a_2}{1\pi}\sin\left(1\pi\frac{x_2-a_2}{b_2-a_2}\right)\\ \vdots\\ \frac{b_2-a_2}{k_2\pi}\sin\left(k_2\pi\frac{x_2-a_2}{b_2-a_2}\right)\\ \vdots\\ \frac{b_2-a_2}{(K-1)\pi}\sin\left((K-1)\pi\frac{x_2-a_2}{b_2-a_2}\right)\end{pmatrix}.$$

From here we introduce the CPD approximation as described in Equation (2.11).

$$\begin{aligned}\mathcal{A}&\approx\sum_{r=1}^{R}\circ_{n=1}^{2}\mathbf{A}_n[:,r],\\ \mathcal{A}_{\mathbf{k}}&\approx\sum_{r=1}^{R}\prod_{n=1}^{2}\mathbf{A}_n[k_n,r].\end{aligned}$$

Substituting this into our previous expression for $\hat{\phi}_p$ results in

$$\begin{aligned}\hat{\phi}_p(\mathbf{x})&\approx\sum_{k_1=1}^{K}\sum_{k_2=0}^{K-1}\left(\mathcal{A}_{k_1,k_2}\prod_{n=1}^{2}\mathbf{s}_n(x_n)[k_n]\right)+\sin\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\\ &\approx\sum_{k_1=1}^{K}\sum_{k_2=0}^{K-1}\left(\sum_{r=1}^{R}\prod_{n=1}^{2}\mathbf{A}_n[k_n,r]\mathbf{s}_n(x_n)[k_n]\right)+\sin\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\\ &=\sum_{r=1}^{R}\sum_{k_1=1}^{K}\sum_{k_2=0}^{K-1}\prod_{n=1}^{2}\Big(\mathbf{A}_n[k_n,r]\mathbf{s}_n(x_n)[k_n]\Big)+\sin\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right).\end{aligned}$$

As we have seen in the final parts of Section 2.8.1, this can be rewritten as

$$\begin{aligned}\hat{\phi}_p(\mathbf{x}) &\approx \left(\overset{2}{\underset{n=1}{\circledast}} \mathbf{s}_n^T(x_n)\mathbf{A}_n\right)\mathbf{1} + \sin\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right) \\ &= \left(\mathbf{s}_1^T(x_1)\mathbf{A}_1 \circledast \mathbf{s}_2^T(x_2)\mathbf{A}_2\right)\mathbf{1} + \sin\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right),\end{aligned}$$

where $\mathbf{1} = (1,\dots,1)^T \in \mathbb{R}^R$. Evaluating the first part of this approximation can also be seen as a feedforward neural network. This is the COS-CPD neural network described in Section 3.4. Updating of the edge weights (factor matrices $\mathbf{A}_n$) is done using the PDE, which is described in the following sections.

7.3. Reformulating the ch.f. PDE

We want to substitute this approximation for $\hat{\phi}_p(\mathbf{x})$ into the PDE, so we first have to introduce the partial derivatives for the vectors $\mathbf{s}_1(x_1)$ and $\mathbf{s}_2(x_2)$.

$$\mathbf{z}_1 = \frac{\partial}{\partial S}\mathbf{s}_1 = \begin{pmatrix} \frac{1\pi}{b_1-a_1}\cos\left(1\pi\frac{x_1-a_1}{b_1-a_1}\right) \\ \vdots \\ \frac{k_1\pi}{b_1-a_1}\cos\left(k_1\pi\frac{x_1-a_1}{b_1-a_1}\right) \\ \vdots \\ \frac{K\pi}{b_1-a_1}\cos\left(K\pi\frac{x_1-a_1}{b_1-a_1}\right)\end{pmatrix},\ \mathbf{z}_2 = \frac{\partial}{\partial \tau}\mathbf{s}_2 = \begin{pmatrix} \frac{1}{2} \\ \cos\left(1\pi\frac{x_2-a_2}{b_2-a_2}\right) \\ \vdots \\ \cos\left(k_2\pi\frac{x_2-a_2}{b_2-a_2}\right) \\ \vdots \\ \cos\left((K-1)\pi\frac{x_2-a_2}{b_2-a_2}\right)\end{pmatrix},$$

$$\mathbf{z}_3 = \frac{\partial^2}{\partial S^2}\mathbf{s}_1 = -\begin{pmatrix} \left(\frac{1\pi}{b_1-a_1}\right)^2\sin\left(1\pi\frac{x_1-a_1}{b_1-a_1}\right) \\ \vdots \\ \left(\frac{k_1\pi}{b_1-a_1}\right)^2\sin\left(k_1\pi\frac{x_1-a_1}{b_1-a_1}\right) \\ \vdots \\ \left(\frac{K\pi}{b_1-a_1}\right)^2\sin\left(K\pi\frac{x_1-a_1}{b_1-a_1}\right)\end{pmatrix}.$$

Now we have all the ingredients to substitute the characteristic function into PDE (7.1). They yield the equation:

$$\begin{aligned}PDE(\mathbf{x}) &= -\frac{\partial\hat{\phi}_p}{\partial\tau} + c_1(S,\tau,r,\sigma^2)\frac{\partial\hat{\phi}_p}{\partial S} + c_2(S,\tau,r,\sigma^2)\frac{\partial^2\hat{\phi}_p}{\partial S^2} = 0 \\ PDE(\mathbf{x}) &= -\left(\mathbf{s}_1^T(x_1)\mathbf{A}_1 \circledast \mathbf{z}_2^T(x_2)\mathbf{A}_2\right)\mathbf{1} + c_1(\mathbf{x})\left[\left(\mathbf{z}_1^T(x_1)\mathbf{A}_1 \circledast \mathbf{s}_2^T(x_2)\mathbf{A}_2\right)\mathbf{1} + \frac{p\pi}{b_1-a_1}\cos\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\right] \\ &\quad + c_2(\mathbf{x})\left[\left(\mathbf{z}_3^T(x_1)\mathbf{A}_1 \circledast \mathbf{s}_2^T(x_2)\mathbf{A}_2\right)\mathbf{1} - \left(\frac{p\pi}{b_1-a_1}\right)^2\sin\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\right] = 0.\end{aligned}$$

Note how this PDE formulation has a Hadamard product between terms with $\mathbf{A}_1$ and $\mathbf{A}_2$. Therefore, deriving a linear system for finding their coefficients is no longer possible. In Section 7.4 we will discuss a method to circumvent this, but to use that method, we need to reformulate the PDE such that one of the factor matrices is fully factored out. We rewrite the PDE as

$$\begin{aligned}
PDE_1(\mathbf{x}^m) &= -\mathbf{s}_1^T(x_1^m)\mathbf{A}_1\mathbf{W}_1^m + c_1(\mathbf{x}^m)\left[\mathbf{z}_1^T(x_1^m)\mathbf{A}_1\mathbf{Q}_1^m + \frac{p\pi}{b_1-a_1}\cos\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\right] \\
&\quad + c_2(\mathbf{x}^m)\left[\mathbf{z}_3^T(x_1^m)\mathbf{A}_1\mathbf{Q}_1^m - \left(\frac{p\pi}{b_1-a_1}\right)^2\sin\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\right] = 0 \\
\mathbf{W}_1^m &= \left[\mathbf{A}_2^T\mathbf{z}_2(x_2^m)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{Q}_1^m &= \left[\mathbf{A}_2^T\mathbf{s}_2(x_2^m)\right] \in \mathbb{R}^{R\times 1} \\
PDE_2(\mathbf{x}^m) &= -\mathbf{z}_2^T(x_2^m)\mathbf{A}_2\mathbf{W}_2^m + c_1(\mathbf{x}^m)\left[\mathbf{s}_2^T(x_2^m)\mathbf{A}_2\mathbf{Q}_{2a}^m + \frac{p\pi}{b_1-a_1}\cos\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\right] \\
&\quad + c_2(\mathbf{x}^m)\left[\mathbf{s}_2^T(x_2^m)\mathbf{A}_2\mathbf{Q}_{2b}^m - \left(\frac{p\pi}{b_1-a_1}\right)^2\sin\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\right] = 0 \\
\mathbf{W}_2^m &= \left[\mathbf{A}_1^T\mathbf{s}_1(x_1^m)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{Q}_{2a}^m &= \left[\mathbf{A}_1^T\mathbf{z}_1(x_1^m)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{Q}_{2b}^m &= \left[\mathbf{A}_1^T\mathbf{z}_3(x_1^m)\right] \in \mathbb{R}^{R\times 1}.
\end{aligned}$$

With this formulation, we can vectorize the factor matrices and find the vectorized forms:

$$\begin{aligned}
PDE_1(\mathbf{x}^m) =& \left(-\left[(\mathbf{W}_1^m)^T \otimes \mathbf{s}_1^T(x_1^m)\right] + c_1(\mathbf{x}^m)\left[(\mathbf{Q}_1^m)^T \otimes \mathbf{z}_1^T(x_1^m)\right] + c_2(\mathbf{x}^m)\left[(\mathbf{Q}_1^m)^T \otimes \mathbf{z}_3^T(x_1^m)\right]\right)\text{vec}(\mathbf{A}_1) \\
&+ c_1(\mathbf{x}^m)\left[\frac{p\pi}{b_1-a_1}\cos\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\right] - c_2(\mathbf{x}^m)\left[\left(\frac{p\pi}{b_1-a_1}\right)^2\sin\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\right] = 0 \\
PDE_2(\mathbf{x}^m) =& \left(-\left[(\mathbf{W}_2^m)^T \otimes \mathbf{z}_2^T(x_2^m)\right] + c_1(\mathbf{x}^m)\left[(\mathbf{Q}_{2a}^m)^T \otimes \mathbf{s}_2^T(x_2^m)\right] + c_2(\mathbf{x}^m)\left[(\mathbf{Q}_{2b}^m)^T \otimes \mathbf{s}_2^T(x_2^m)\right]\right)\text{vec}(\mathbf{A}_2) \\
&+ c_1(\mathbf{x}^m)\left[\frac{p\pi}{b_1-a_1}\cos\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\right] - c_2(\mathbf{x}^m)\left[\left(\frac{p\pi}{b_1-a_1}\right)^2\sin\left(p\pi\frac{x_1-a_1}{b_1-a_1}\right)\right] = 0.
\end{aligned}$$

Note that these are just linear systems in the form $LHS_n\text{vec}(\mathbf{A}_n) - RHS_n = 0$, so linear solvers can be used to find coefficients of $\mathbf{A}_n$, given that we have the coefficients of all $\mathbf{A}_m, m \neq n$.

7.4. Finding factor matrices $\mathbf{A}_n$ through ALS

As mentioned before, deriving a linear system that solves for both $\mathbf{A}_1$ and $\mathbf{A}_2$ is not possible. For that reason we find the coefficients iteratively through the Alternating Least Squares (ALS) algorithm.

Faber, Bro, and Hopke [43] compared ALS with six different methods. Among these, ALS provided the highest quality solutions, while the alternating slicewise diagonalization (ASD) method [44] was a viable alternative for scenarios requiring low computation time. In another study, Tomasi and Bro [45] compared ALS and ASD with four other methods and three variants utilizing Tucker-based compression, followed by computing a CPD of the reduced array [46]. This comparison included the damped Gauss-Newton (dGN) method and a variant called PMF3 [47], both of which optimize all factor matrices simultaneously. Contrary to Faber, Bro, and Hopke's findings, ASD was found to be inferior to other alternating-type methods. Derivative-based methods (dGN and PMF3) generally outperformed ALS in terms of convergence properties but had a larger computational complexity.

In this chapter, we use the same ALS method as used and improved in [2]. Algorithm 1 shows the pseudocode for ALS. We require a several inputs: dimension N, number of expansion points K, and CPD rank R. We also require the model parameters r, σ^2 and lower and upper bounds of expansion a_1, b_1, a_2, b_2.

In Algorithm 1, we compute the residual of the PDE at predetermined points, which should differ from the training points. Due to the fewer expansion coefficients compared to before, we might not fully converge to zero residual at these points. Therefore, we want to set a maximum number of ALS iterations to avoid getting stuck in an infinite loop.

As shown in [4], the number of training points in each dimension should be equal to the number of expansion terms K.

One last remark about the algorithm is why we solve $M\text{vec}(\mathbf{A}_n) = f$. Note that $\text{vec}(\mathbf{A}_n) \in \mathbb{R}^{KR}$ and $LHS_n \in \mathbb{R}^{K^N \times KR}$, because it has as many rows as there are training points. Therefore, this is not a square linear system and it is not easily solved. To bypass this, we solve this by changing to normal equations $A^TAx = A^Tb$.

Another explanation for choosing these normal equations comes from the essence of the problem. We try to find $\text{vec}(\mathbf{A}_n)$ such that the Euclidean distance $||LHS_n\text{vec}(\mathbf{A}_n) - RHS_n||_2^2$ is minimized. We can rewrite this distance as

$$\begin{aligned}
||LHS_n\text{vec}(\mathbf{A}_n) - RHS_n||_2^2 &= \text{vec}(\mathbf{A}_n)^T LHS_n^T LHS_n \text{vec}(\mathbf{A}_n) - 2RHS_n^T LHS_n \text{vec}(\mathbf{A}_n) + RHS_n^T RHS_n \\
&= \text{vec}(\mathbf{A}_n)^T M \text{vec}(\mathbf{A}_n) - 2f^T \text{vec}(\mathbf{A}_n) + RHS_n^T RHS_n \\
M &= LHS^T LHS \\
f &= LHS^T RHS
\end{aligned}$$

In the minimization problem we can neglect the constant term $RHS_n^T RHS_n$. Furthermore, we can divide the whole expression by 2, since the minimum would be at the same $\mathbf{A}_n$. We find the gradient

$$M\text{vec}(\mathbf{A}_n) - f.$$

Finding the minimum is equivalent to setting this gradient equal to zero. This results in the previously mentioned normal equations.

Automatically M is symmetric, which can speed up calculations. The solving step is done using `scipy.linalg.solve()`.

Algorithm 1 Calibrating $\mathbf{A}_n$ through ALS

```
Input: $N, K, R, r, \sigma^2, a_1, b_1, a_2, b_2, p, Max\_ALS\_iters$
Initialize $\mathbf{A}_1, \mathbf{A}_2$
Create training points $\mathbf{x}$
for $i = 1, ..., Max\_ALS\_iters$ do
    for $n = 1, 2$ do
        $M = LHS_n^T LHS_n$
        $f = LHS_n^T RHS_n$
        Solve $M\text{vec}(\mathbf{A}_n) = f$
        Update $\mathbf{A}_n$
    end for
    $residuals = PDE(x)^2$
    if $residuals < \epsilon$ then
        stop algorithm and return $\mathbf{A}_n$
    end if
end for
```

7.4.1. Choosing the number of ALS iterations

The maximum number of ALS iterations greatly influences the time taken, as the residuals are rarely small enough to satisfy the stopping criterion. Therefore, we should first analyse what a good choice would be. Figure 7.1 shows how the mean square error of the residuals behaves for multiple ALS iterations. One can see that after 3 iterations all errors seem to be fully converged. Additionally, we see that increasing the rank has a bigger effect when we have a larger number of expansion terms. Initially, the error dominance comes from the number of expansion terms, but once this is sufficiently large, the dominant error source becomes the CPD rank. Therefore, an increasingly large K also requires an increasingly large R to improve the model accuracy. For the remaining calculations, we choose `Max_ALS_iters` = 3.

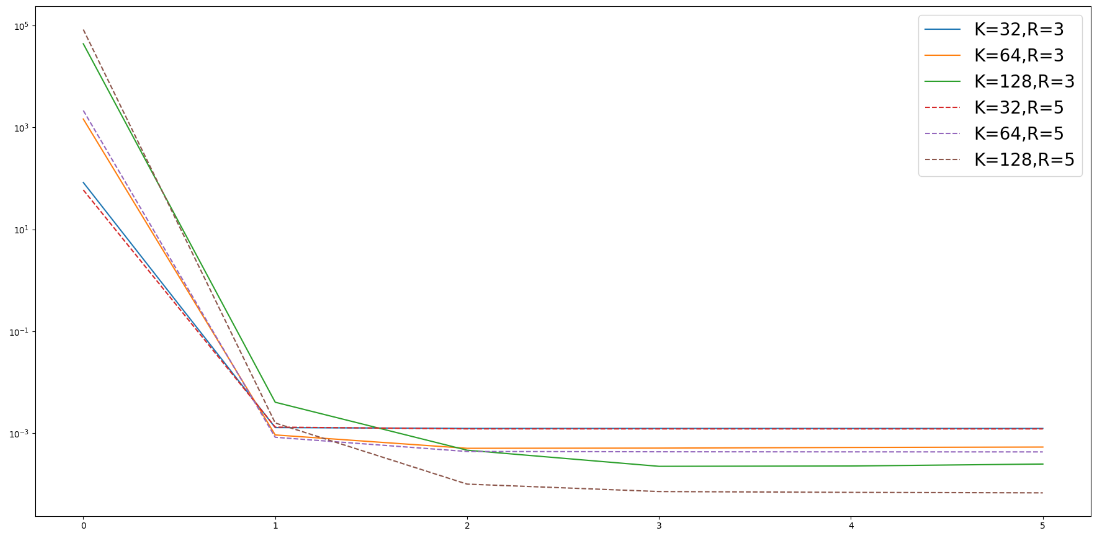


Figure 7.1: Mean square error for 200 points with different expansion and CPD parameters

7.4.2. Computational complexity

We introduced CPD to reduce the computational complexity we had in Chapter 5. Table 7.1 summarizes the memory required for the two most memory intensive steps in the algorithms. In the direct approach from Chapter 5, the LHS matrix from the linear system is already square and is used for solving the linear system. The memory required for this matrix is $O(K^{2N})$.

With the CPD included in the algorithm, the most expensive step is generating the LHS matrix with memory $O(K^{N+1}R)$ and performing $LHS^T LHS$. Solving the system is done for a matrix with memory $O(KR \cdot KR)$.

Note that LHS is never used in Algorithm 1 except for finding M. Therefore, at the cost of much higher computation time, it would be possible to skip computing the full matrix LHS and instantly find M instead.

Step	Direct approach	with CPD
Memory LHS	$K^N \times K^N$	$K^N \times KR$
Linear system size	$K^N \times K^N$	$KR \times KR$
Storing training points	-	$K^N \times N$

Table 7.1: Memory requirements for direct approach and the algorithm with CPD.

7.5. Error analysis of the characteristic function approximation

Before we look at the results using CPD, we first analyse the error behaviour of the approximation of $\hat{\phi}_p$ compared to the direct approach of Chapter 5. We also compare $\hat{\phi}(t,S)$ found using the two methods with the benchmark as described in Equation (2.9). We set $(N,K,R) = (2,64,7)$ for our expansions. For each $\{p \in 1,\dots,64\}$ we evaluate $\hat{\phi}$ on an equidistant grid with 200 points in each dimension, i.e. 40.000 points in total. For each of these p, we take the average over all the points to calculate the mean square error with respect to benchmark, the average absolute difference between the 2 methods, and the average relative difference between the 2 methods. The results are presented in Figure 7.2.

In this figure, we can see that both methods have very similar accuracy at the same place. The absolute errors between the two methods are generally around 10^{-4}. This suggests that the option price approximations should give very similar accuracy levels with and without CPD. Note that there are some large relative errors. The reason is that $|\hat{\phi}|$ is very small there. Those large relative errors are more prominent for larger p, where $|\hat{\phi}|$ is smaller. This, however, raises no issues, since the expansion term of a large p will barely contribute to the final pricing function due to the small value of $\hat{\phi}$ thereof.

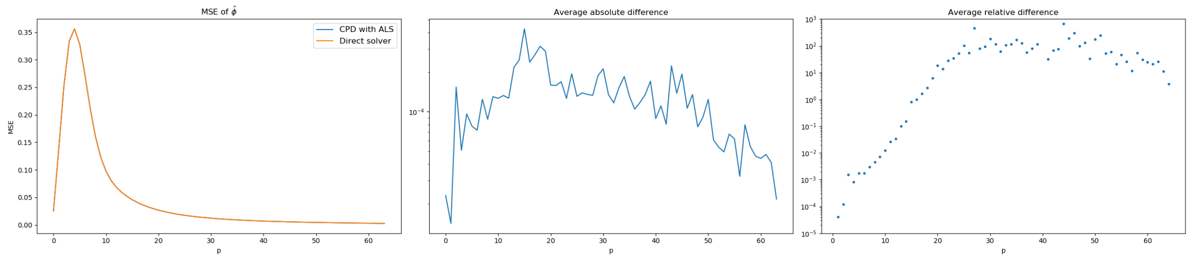


Figure 7.2: Error behaviour of $\hat{\phi}$ for the direct method from Chapter 5 and the new method of this chapter that includes CPD.

7.6. Results

We use the same parameters as in Appendix 5, i.e. $N = 2$ and $K = 64$. The model parameters set $(r, \sigma^2) = (0.1, 0.01)$. For the option parameters, we have $(E, B) = (90, 120)$. The only new parameter, the rank, is chosen to be $R = 8$. This choice was made after numerical investigation. A larger R leads to more accurate results, but beyond $R = 8$, the results do not improve significantly relative to the increase in computing time. The errors are calculated by comparing to the benchmark from Equation (2.8).

The results are presented in Figure 7.3. Compared to the results from Figure 5.3, the error behaviours are similar in both cases, but in Figure 7.3, the errors for low asset prices are slightly larger. It is expected that the direct method from Chapter 5 has lower errors, as CPD introduced an extra approximation. Since the CPD method scales better to higher dimensions and a larger K, we can conclude that the CPD method will likely be the method with more potential in higher dimensions.

7.6.1. Condition numbers

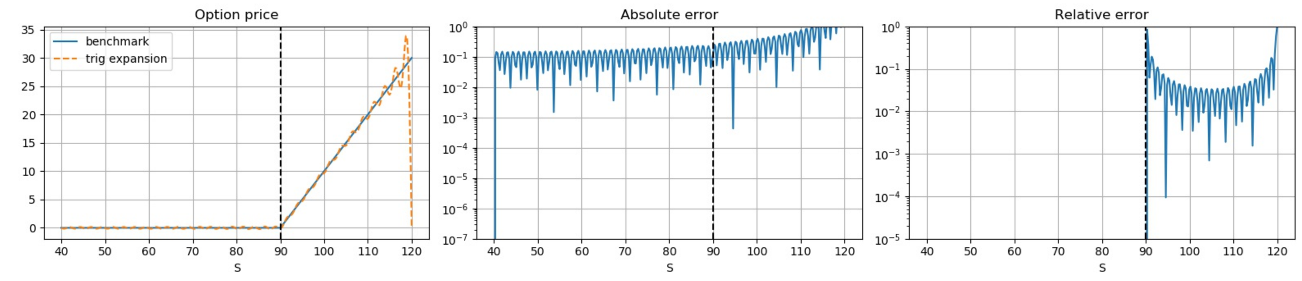
It is important to note the large condition numbers of matrix $M = LHS^T LHS$. A large condition number raises difficulty to the linear system solver, resulting less accurate results. In this algorithm, the condition number can be quite large at some points, often being around 10^5 in the first ALS iteration and increasing to between 10^9 and 10^{14}, depending on p by the third ALS iteration.

Preconditioning can be used to decrease the condition number. Most preconditioning matrices are designed for sparse matrices, but our matrix M is dense. After applying the simple Jacobian preconditioner (diagonal of M), the preconditioned matrix has condition number 10^3 for the first ALS iteration and between 10^8 and 10^{12} for the third ALS iteration, depending on the value of p. No further investigation has been done on preconditioning due to the complexity of preconditioning with a dense matrix. A small summary of the condition numbers is made in Table 7.2

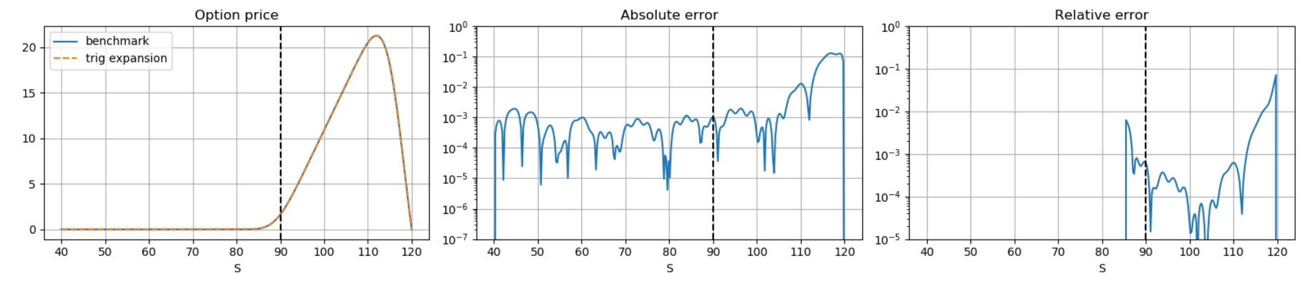
We tried to analyse whether the condition number is the main error source, but it is difficult to isolate this as an error source from the other ones. We should keep the condition number in mind when increasing the number of dimensions.

ALS iteration	Without preconditioning	Jacobian preconditioning
1	10^5	10^3
3 (small p)	10^9	10^8
3 (large p)	10^{14}	10^{12}

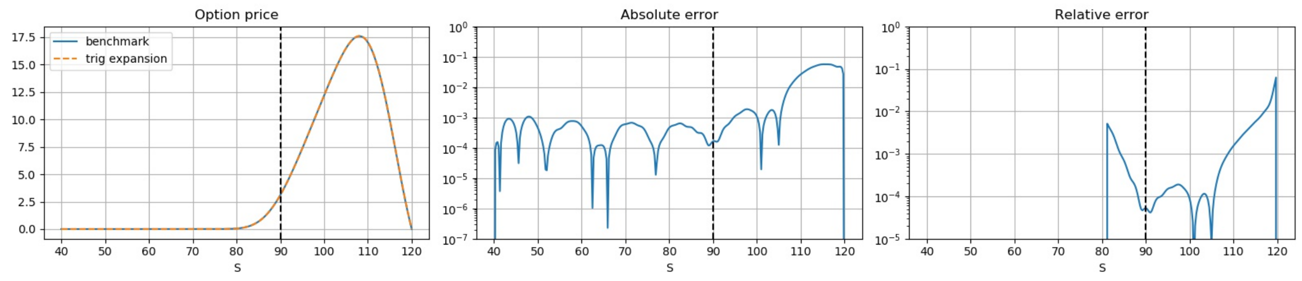
Table 7.2: Condition numbers of system matrix M



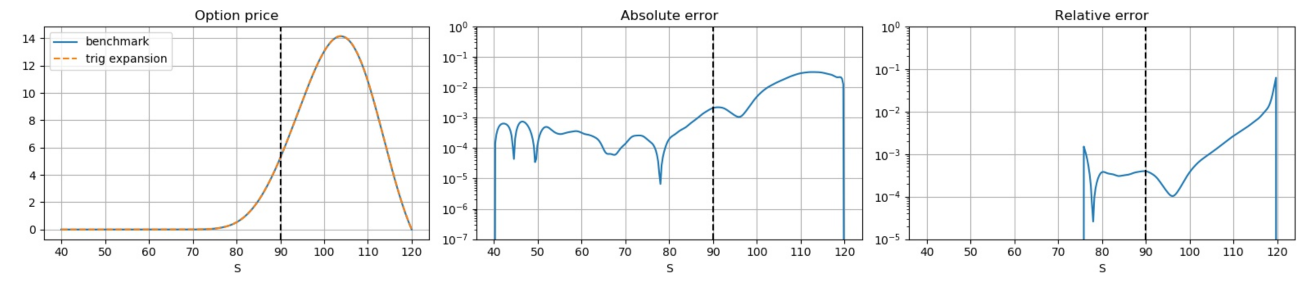
(a) Option price, absolute error and relative error for $\tau = 0$



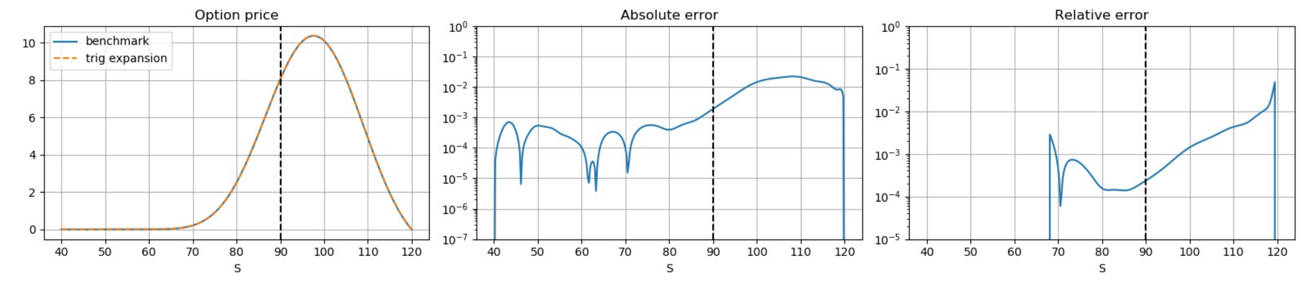
(b) Option price, absolute error and relative error for $\tau = 0.1$



(c) Option price, absolute error and relative error for $\tau = 0.25$



(d) Option price, absolute error and relative error for $\tau = 0.5$



(e) Option price, absolute error and relative error for $\tau = 1$

Figure 7.3: Option price and errors for various τ using trigonometric expansion with CPD vs closed-form solution for an up-and-out barrier option with parameters $(r, \sigma^2) = (0.1, 0.01)$ and $(a_1, b_1, a_2, b_2, E) = (40, 120, 0, 1, 90)$ under regular-asset GBM dynamics.

7.7. Directly approximating option price

So far, we have been following the steps laid out in [3], i.e. we first approximate the option price through the COS method, then inserting the SIN formula into the PDE to yield a new PDE for the survival ch.f. $\hat{\phi}_p$. Then through trigonometric expansion (and optionally CPD), we find approximations for these $\hat{\phi}_p$, which can be substituted back into the SIN formula of the option price.

A natural question is whether we can avoid going through these extra steps and directly solve the pricing PDE via trigonometric expansion (and with CPD). This question is answered below.

7.7.1. Pricing PDE

We first apply $\tau = T - t$ to Equation (5.1) to find

$$-\frac{\partial V}{\partial \tau} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial^2 S} - rV = 0, \tag{7.3}$$

with the boundary conditions $V(\tau, a_1) = V(\tau, b_1) = 0$, where a_1 and b_1 are the lower and upper barrier of the option. The initial condition $V(0, S)$ is defined by the option payoff function. So for a barrier call option this would be $V(0, S) = (S - E)^+$.

7.7.2. Trigonometric expansion for the option price

We once again employ trigonometric expansion obtained via integrating out the Fourier expansion of the first order derivative in time, since this incorporates the initial condition. Again, we can use sine expansion on the asset dimension to automatically include the boundary conditions, resulting in

$$\begin{aligned} V(\tau, S) \approx & \sum_{k_1=1}^{K}\sum_{k_2=1}^{K-1} \mathcal{A}_{k_1,k_2} \frac{b_2 - a_2}{k_2 \pi} \sin\left(k_2\pi \frac{\tau - a_2}{b_2 - a_2}\right) \sin\left(k_1 \pi \frac{S - a_1}{b_1 - a_1}\right) \\ & + \sum_{k_1=1}^{K} \mathcal{A}_{k_1,0} \frac{\tau}{2} \sin\left(k_1\pi \frac{S - a_1}{b_1 - a_1}\right) + V(0, S), \quad a_2 = 0. \end{aligned}$$

For $S = a_1$ or $S = b_1$, we get $V(\tau, a_1) = V(0, a_1)$ and $V(\tau, b_1) = V(0, b_1)$. For $\tau = 0$, we are just left with the initial condition $V(0, S)$. So it satisfies the boundary conditions and initial condition. After applying CPD, we get

$$V(\tau, S) \approx \left(\mathbf{s}_1^T(x_1)\mathbf{A}_1 \circledast \mathbf{s}_2^T(x_2)\mathbf{A}_2\right)\mathbf{1} + V(0, S),$$

where $\mathbf{s}_i$ are still defined the same as for the survival ch.f. expansion.

7.7.3. Reformulating the PDE

When we substitute the expansion of the unknown function into the PDE it satisfies, we have to generate the derivatives of the expansion approximation. Note that $\frac{\partial V(0,S)}{\partial \tau} = 0$, but the derivative in the asset price dimension is not fully defined. If we consider European call payoff $V(0, S) = (S - E)^+$, we get

$$\frac{\partial V(0, S)}{\partial S} = \begin{cases} 0, & S < E \\ 1, & S > E \\ undefined, & S = E \end{cases}$$

$$\frac{\partial^2 V(0, S)}{\partial S^2} = \begin{cases} 0, & S \neq E \\ undefined, & S = E. \end{cases}$$

This non-continuous behaviour is known to cause issues with Fourier based methods, due to the Gibbs phenomenon at those discontinuities.

Further steps are identical to Section 7.3. The factor matrices are also solved the same way, i.e. using Algorithm 1.

7.7.4. Results

Unfortunately, this method did not converge to the actual solution. Therefore, no comparisons are made to either the benchmark or the method developed in Chapters 5 and 6.

We could not pinpoint where this lack of convergence comes from, but the discontinuities in the derivatives of the initial condition could be the reason. Further research can be done along this direction, but this thesis does not deem this method a viable option.

7.8. conclusion

In this chapter we used CPD to decrease the dimensionality. We have seen that the computational complexity is greatly reduced, which allows us to deal with a higher number of dimensions.

The accuracy level with CPD is very similar to that of the model without CPD. We do, however, see somewhat larger errors for far out-of-the-money options. In return for this, we get a method that scales better with a higher number of dimensions. Whether this trade-off is worth it, can not be seen from the 2-dimensional GBM case.

One big downside of this method is that it is not compatible with the change of variables explored in Section 5.6. That change of variables gave a significant improvement of performance, which is now lost. It is not clear why CPD does not work well in combination with the inverted log-asset transformation. This would require more investigation.

So far, in this thesis, we have applied trigonometric expansion to $\hat{\phi}_p$ to get the results. Therefore, we have to find the expansion coefficients for all $p \in \{1, 2, \dots, K\}$. At last, we also tried applying trigonometric expansion method directly to the option price PDE. This did not give converging results, likely due to the discontinuities in the derivatives of the initial condition.

8

Our contribution 5: Pricing barrier options under SABR using the COS-CPD network

This chapter applies the COS-CPD network, as applied in Chapter 7 to the pricing problem discussed in Chapter 6.

8.1. Characteristic function PDE under SABR

We start from the PDE in IBVP (6.5). As in Chapter 7, we introduce coefficients c_1, c_2 and c_3 for more clarity throughout the derivations. Then our target PDE can be written as

$$
\begin{aligned}
-\frac{\partial \hat{\phi}_p}{\partial \tau} + c_1 \frac{\partial^2 \hat{\phi}_p}{\partial S^2} + c_2 \frac{\partial^2 \hat{\phi}_p}{\partial \sigma^2} + c_3 \frac{\partial^2 \hat{\phi}_p}{\partial S \partial \sigma} = 0 \\
c_1 = \frac{1}{2}\sigma^2 S^{2\beta} \\
c_2 = \frac{1}{2}\alpha^2 \sigma^2 \\
c_3 = \alpha \rho \sigma^2 S^{\beta}.
\end{aligned}
$$

8.2. Approximating the ch.f. with CPD

We start from Equation (6.6) and rewrite it in the same form as Equation (7.2):

$$
\hat{\phi}_p(\tau, S) \approx \sum_{k_1=1}^{K} \sum_{k_2=0}^{K-1} \sum_{k_3=1}^{K} \left(\mathcal{A}_{k_1,k_2,k_3} \mathbf{s}_1(x_1)[k_1] \cdot \mathbf{s}_2(x_2)[k_2] \cdot \mathbf{s}_3(x_3)[k_3] \right) + \sin\left(p\pi \frac{S - a_1}{b_1 - a_1} \right), \tag{8.1}
$$

where $\mathbf{x} = (x_1, x_2, x_3) = (S, \tau, \sigma)$ and

$$
\mathbf{s}_1(x_1) = \begin{pmatrix} \sin\left(1\pi \frac{x_1-a_1}{b_1-a_1}\right) \\ \vdots \\ \sin\left(k_1\pi \frac{x_1-a_1}{b_1-a_1}\right) \\ \vdots \\ \sin\left(K\pi \frac{x_1-a_1}{b_1-a_1}\right) \end{pmatrix}, \quad
\mathbf{s}_2(x_2) = \begin{pmatrix} \frac{x_2}{2} \\ \frac{b_2-a_2}{1\pi} \sin\left(1\pi \frac{x_2-a_2}{b_2-a_2}\right) \\ \vdots \\ \frac{b_2-a_2}{k_2\pi} \sin\left(k_2\pi \frac{x_2-a_2}{b_2-a_2}\right) \\ \vdots \\ \frac{b_2-a_2}{(K-1)\pi} \sin\left((K-1)\pi \frac{x_2-a_2}{b_2-a_2}\right) \end{pmatrix}, \quad
\mathbf{s}_3(x_3) = \begin{pmatrix} \sin\left(1\pi \frac{x_3-a_3}{b_3-a_3+\epsilon}\right) \\ \vdots \\ \sin\left(k_3\pi \frac{x_3-a_3}{b_3-a_3+\epsilon}\right) \\ \vdots \\ \sin\left(K\pi \frac{x_3-a_3}{b_3-a_3+\epsilon}\right) \end{pmatrix}.
$$

Next, we can introduce the CPD approximation from Equation (2.11) again. This results in

$$\mathcal{A} \approx \sum_{r=1}^{R} \circ_{n=1}^{3} \mathbf{A}_n[:, r],$$
$$\mathcal{A}_{\mathbf{k}} \approx \sum_{r=1}^{R} \prod_{n=1}^{3} \mathbf{A}_n[k_n, r].$$

After substituting these into the expression for $\hat{\phi}_p$ and rearranging the terms as before, we get the following:

$$\hat{\phi}_p(\mathbf{x}) \approx \left(\mathbf{s}_1^T(x_1)\mathbf{A}_1 \circledast \mathbf{s}_2^T(x_2)\mathbf{A}_2 \circledast \mathbf{s}_3^T(x_3)\mathbf{A}_3\right)\mathbf{1} + \sin\left(p\pi\frac{x_1 - a_1}{a_1 - b_1}\right).$$

where $\mathbf{1} = (1, \dots, 1)^T \in \mathbb{R}^R$.

8.3. Rewriting the ch.f. PDE

Our next step is substituting the new formulation of $\hat{\phi}_p$ into the PDE, but before we can do that, we have to find the partial derivatives of $\mathbf{s}_i$. We write these as

$$\mathbf{z}_1 = \frac{\partial}{\partial S}\mathbf{s}_1 = \begin{pmatrix} \frac{1\pi}{b_1-a_1}\cos\left(1\pi\frac{x_1-a_1}{b_1-a_1}\right) \\ \vdots \\ \frac{k_1\pi}{b_1-a_1}\cos\left(k_1\pi\frac{x_1-a_1}{b_1-a_1}\right) \\ \vdots \\ \frac{K\pi}{b_1-a_1}\cos\left(K\pi\frac{x_1-a_1}{b_1-a_1}\right) \end{pmatrix}, \mathbf{z}_2 = \frac{\partial}{\partial \tau}\mathbf{s}_2 = \begin{pmatrix} \frac{1}{2} \\ \cos\left(1\pi\frac{x_2-a_2}{b_2-a_2}\right) \\ \vdots \\ \cos\left(k_2\pi\frac{x_2-a_2}{b_2-a_2}\right) \\ \vdots \\ \cos\left((K-1)\pi\frac{x_2-a_2}{b_2-a_2}\right) \end{pmatrix},$$

$$\mathbf{z}_3 = \frac{\partial}{\partial \sigma}\mathbf{s}_3 = \begin{pmatrix} \frac{1\pi}{b_3-a_3+\epsilon}\cos\left(1\pi\frac{x_3-a_3}{b_3-a_3+\epsilon}\right) \\ \vdots \\ \frac{k_3\pi}{b_3-a_3+\epsilon}\cos\left(k_3\pi\frac{x_3-a_3}{b_3-a_3+\epsilon}\right) \\ \vdots \\ \frac{K\pi}{b_3-a_3+\epsilon}\cos\left(K\pi\frac{x_3-a_3}{b_3-a_3+\epsilon}\right) \end{pmatrix}, \mathbf{z}_4 = \frac{\partial^2}{\partial S^2}\mathbf{s}_1 = -\begin{pmatrix} \left(\frac{1\pi}{b_1-a_1}\right)^2\sin\left(1\pi\frac{x_1-a_1}{b_1-a_1}\right) \\ \vdots \\ \left(\frac{k_1\pi}{b_1-a_1}\right)^2\sin\left(k_1\pi\frac{x_1-a_1}{b_1-a_1}\right) \\ \vdots \\ \left(\frac{K\pi}{b_1-a_1}\right)^2\sin\left(K\pi\frac{x_1-a_1}{b_1-a_1}\right) \end{pmatrix},$$

$$\mathbf{z}_5 = \frac{\partial^2}{\partial \sigma^2}\mathbf{s}_3 = -\begin{pmatrix} \left(\frac{1\pi}{b_3-a_3+\epsilon}\right)^2\sin\left(1\pi\frac{x_3-a_3}{b_3-a_3+\epsilon}\right) \\ \vdots \\ \left(\frac{k_3\pi}{b_3-a_3+\epsilon}\right)^2\sin\left(k_3\pi\frac{x_3-a_3}{b_3-a_3+\epsilon}\right) \\ \vdots \\ \left(\frac{K\pi}{b_3-a_3+\epsilon}\right)^2\sin\left(K\pi\frac{x_3-a_3}{b_3-a_3+\epsilon}\right) \end{pmatrix}.$$

With this, we can substitute $\hat{\phi}_p$ into the PDE to find

$$\begin{aligned} PDE(\mathbf{x}) = &-\left(\mathbf{s}_1^T(x_1)\mathbf{A}_1 \circledast \mathbf{z}_2^T(x_2)\mathbf{A}_2 \circledast \mathbf{s}_3^T(x_3)\mathbf{A}_3\right)\mathbf{1} \\ &+ c_1(\mathbf{x})\left[\left(\mathbf{z}_4^T(x_1)\mathbf{A}_1 \circledast \mathbf{s}_2^T(x_2)\mathbf{A}_2 \circledast \mathbf{s}_3^T(x_3)\mathbf{A}_3\right)\mathbf{1} - \left(\frac{p\pi}{b_1 - a_1}\right)^2 \sin\left(p\pi\frac{x_1 - a_1}{b_1 - a_1}\right)\right] \\ &+ c_2(\mathbf{x})\left(\mathbf{s}_1^T(x_1)\mathbf{A}_1 \circledast \mathbf{s}_2^T(x_2)\mathbf{A}_2 \circledast \mathbf{z}_5^T(x_3)\mathbf{A}_3\right)\mathbf{1} + c_3(\mathbf{x})\left(\mathbf{z}_1^T(x_1)\mathbf{A}_1 \circledast \mathbf{s}_2^T(x_2)\mathbf{A}_2 \circledast \mathbf{z}_3^T(x_3)\mathbf{A}_3\right)\mathbf{1} = 0. \end{aligned}$$

The vectorized versions of the PDE can be found in Appendix A.5.

8.4. Results

The methodology for finding the factor matrices through ALS is unaltered. Unlike in the GBM case, we have no analytic benchmark for $\hat{\phi}_p$ so alternative benchmark values or methods are needed.

Figure 8.1 shows the results for the CPD approach with model parameters $(N, K, R) = (3, 32, 20)$ and $(a_1, b_1, a_2, b_2, a_3, b_3) = (40, 120, 0, 1, 0, 1)$. The dynamic and option parameters are $(r, \alpha, \beta, \rho, \sigma_0, E) = (0.1, 0.2, 0.5, -0.85, 90)$. We can compare these results to Figure 6.1.

By including CPD, the computing time and memory were reduced significantly. Training the model was almost 10 times faster and K can now be chosen around $K = 64$. This did come at the cost of accuracy. Compared to the approach without CPD, the relative error is almost 10 times larger. To investigate the source of the error, we have increased the rank to $R = 50$, which, however, leads to no significant decrease in the errors. Increasing the number of expansion terms, $K = 48$, also barely decreased the errors, despite training of this model taking more than double the time that training of the non-CPD model with $K = 26$ did.

Like in the case of GBM, the condition numbers are very large. This could have an impact on the accuracy of the solver used to solve the linear system in the training step. Inaccuracies can also be caused by the efficiency issue in the ALS procedure, as summarized in [24]: it is not guaranteed that ALS finds the global optimum solution. A third possible reason for inaccurate results is over-fitting, i.e. the degrees of freedom provided are larger than the number of constraints. This can cause ALS detect and return a local optimum.

8.4.1. Condition number

Table 8.1 shows the condition numbers with and without preconditioning. In Table 8.2 we show the condition numbers for $p = 1$ in every iteration of the ALS algorithm for a better understanding of when the condition number is large. We can see that the large condition numbers mainly play a role with the isolated $\mathbf{A}_3$, i.e., the third factor matrix. A possible explanation for this is that optimizing the first and second factor matrix already causes the PDE to almost be satisfied, leading to ALS finding a local optimum when updating the third factor matrix. This problem is similar to the problem of overfitting.

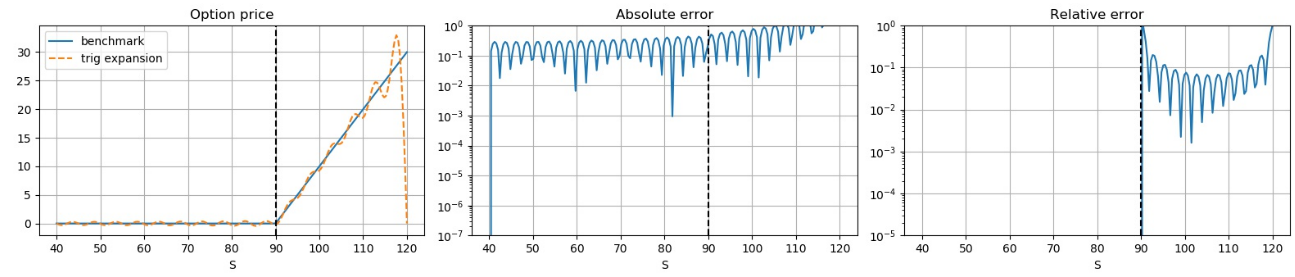
These condition numbers are much larger compared to the condition numbers in the GBM case, therefore, it is very likely that these large condition numbers are causes of the low accuracy.

ALS iteration	Without preconditioning	Jacobian preconditioning
1	10^5	10^3
3 (small p)	10^{20}	10^{18}
3 (large p)	10^{21}	10^{19}

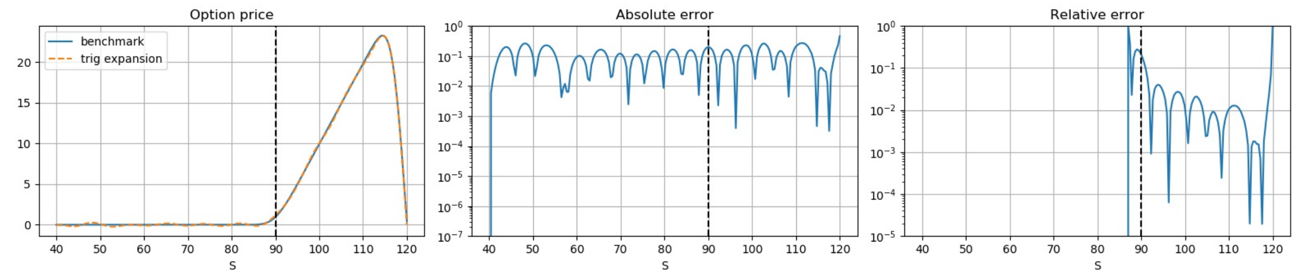
Table 8.1: Condition numbers of system matrix M

ALS iteration	Isolated $\mathbf{A}_1$	Isolated $\mathbf{A}_2$	Isolated $\mathbf{A}_3$
1	10^2	10^5	10^{20}
2	10^5	10^8	10^{20}
3	10^5	10^8	10^{20}

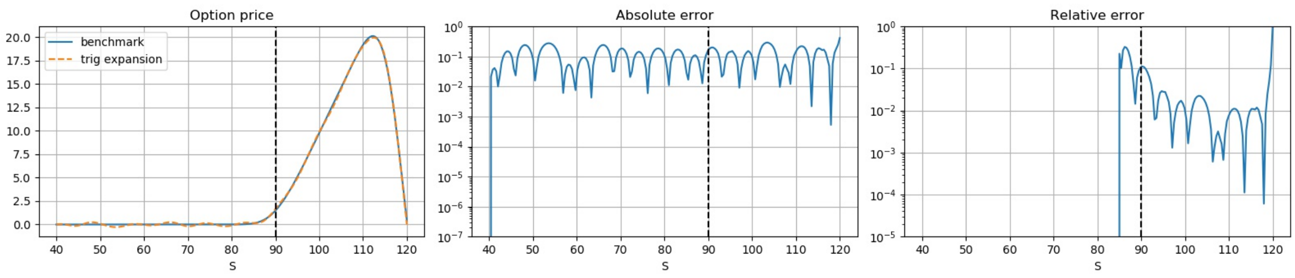
Table 8.2: Condition numbers of system matrix M for $p = 1$



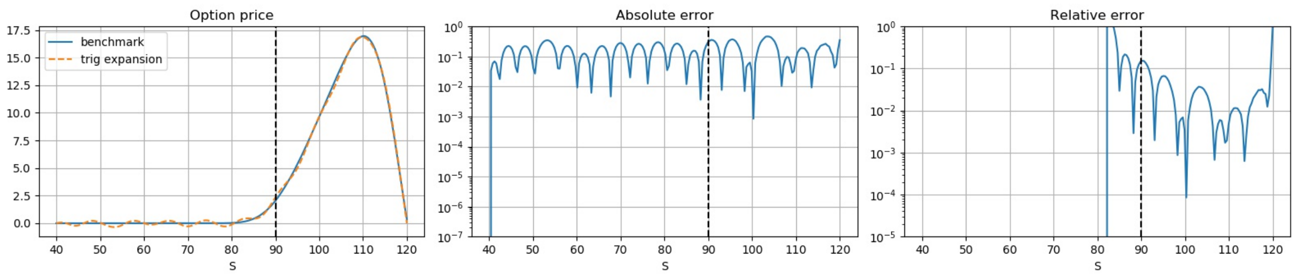
(a) Option price, absolute error and relative error for $\tau = 0$



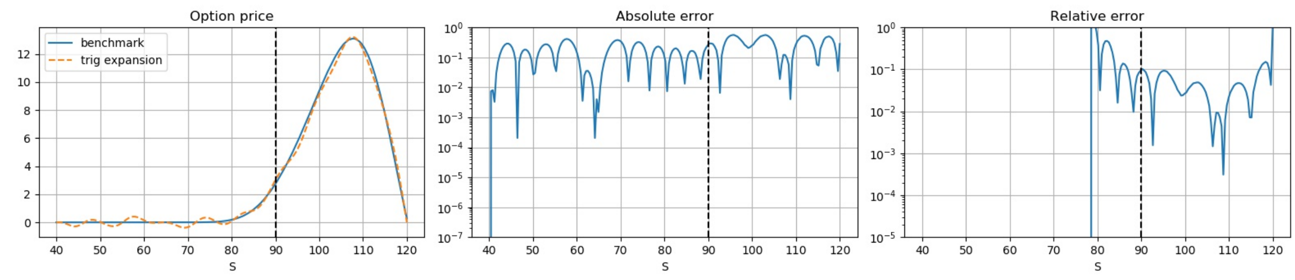
(b) Option price, absolute error and relative error for $\tau = 0.1$



(c) Option price, absolute error and relative error for $\tau = 0.25$



(d) Option price, absolute error and relative error for $\tau = 0.5$



(e) Option price, absolute error and relative error for $\tau = 1$

Figure 8.1: Option price and errors for various τ using trigonometric expansion with CPD vs Monte Carlo benchmark for an up-and-out barrier option with parameters $(r, \alpha, \beta, \rho, \sigma_0) = (0.1, 0.2, 0.5, -0.85)$ and $(a_1, b_1, a_2, b_2, a_3, b_3, E) = (40, 120, 0, 1, 0, 1, 90)$ under regular-asset SABR dynamics.

8.5. Regularization

As discussed in Section 8.4.1, the large condition numbers might be a result of overfitting. Overfitting is a well-researched problem and therefore multiple solutions are available. One such solution is by increasing the number of training points. In our case, this might not be a great solution, since the number of training points determines our main computational complexity.

Another solution is adding regularization. By adding a term to the minimization problem that regulates how much a coefficient contributes to the overall solution, the coefficients that barely contribute will be set to 0 or a very small value (depending on the type of regularization). We discuss 2 types of regularization, Lasso (L_1) and Ridge (L_2). For now, consider the problem

$$\arg\min_x \ \|Ax - b\|_2^2$$

Lasso regularization is regularization using the L_1 norm. The minimization problem is adjusted to

$$\arg\min_x \ \|Ax - b\|_2^2 + \lambda \sum_i |x[i]|,$$

where λ is the regularization weight. The main advantage of Lasso regularization is that some coefficients are driven to be exactly 0. This results into a more sparse solution. It is, however, much harder to implement, since there is no vectorization possible to incorporate the absolute values.

Ridge regularization acts very similarly to Lasso regularization, but instead uses the L_2 norm. The minimization problem becomes

$$\arg\min_x \ \|Ax - b\|_2^2 + \lambda \sum_i \|x[i]\|_2^2,$$

where λ is the regularization weight. We can rewrite these norms into vector-matrix form.

$$\|Ax - b\|_2^2 + \lambda \sum_i \|x[i]\|_2^2 = x^T A^T A x - 2b^T Ax + b^T b + \lambda x^T x,$$

where $b^T b$ can be left out, since we are doing a minimization problem and this is just a constant. Due to the convex nature of a least squares problem, minimizing the above equation is equivalent to setting the gradient to zero.

$$\begin{aligned} 2A^T Ax - 2b^T A + 2\lambda x &= 0 \\ (A^T A + \lambda I)x &= b^T A, \end{aligned}$$

where I is the identity matrix.

Note that this is very similar to the normal equations in Algorithm 1. The only difference is that matrix M now has an additional part, i.e. $M + \lambda I$.

8.6. Results with regularization

We apply the ridge regularization and analyse the results and condition numbers again. After some numerical experimentation, we find that $\lambda = 0.001$ seems like the best choice of the regularization weight.

Table 8.3 shows a table in the same setting as Table 8.2, but now with regularization. Note how the largest recorded condition number drops from 10^{20} to 10^5. This new largest condition number is for $\mathbf{A}_1$ and no longer for $\mathbf{A}_3$. This could still be considered a large condition number, but it will likely cause much less problems than before.

The results are shown in Figure 8.2. Comparing this to the results from Figure 8.1, we see that in general the the errors are reduced. At some points this difference is about 10^{-1}, but compared to the results from Figure 6.1, we still do not have the level of accuracy as we had before.

ALS iteration	Isolated $\mathbf{A}_1$	Isolated $\mathbf{A}_2$	Isolated $\mathbf{A}_3$
1	10^2	10^3	10^5
2	10^5	10^3	10^5
3	10^5	10^3	10^5

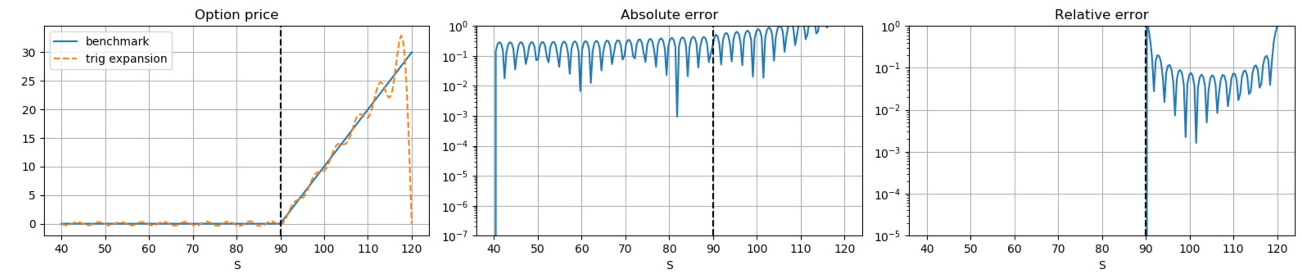
Table 8.3: Condition numbers of regularized system matrix $M + \lambda I$ for $p = 1$

We also recreate Table 6.2, but now with COS-CPD on SABR. The new results with $K = 32, R = 20, \lambda = 0.001$ are shown in Table 8.4. Note how in general, all errors are larger than without CPD, similar to what we saw in Figure 8.2. Something that is different, is that the condition numbers were very large again (mostly $\mathcal{O}(10^{19})$), even with regularization.

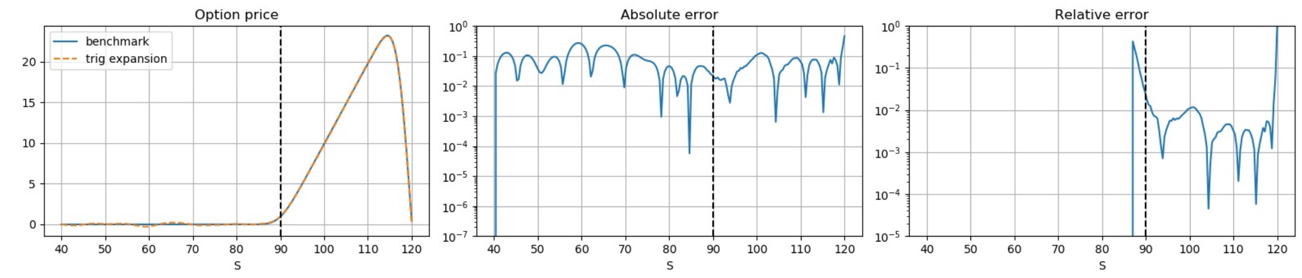
Another important measure of performance is the computational time. Our COS-CPD network required around 15 minutes of training for each model. The closed-form approximation from [13] only requires a one-dimensional numerical integration, taking several milliseconds per calculated option price.

	$\tau = \frac{1}{252}$			$\tau = \frac{1}{52}$			$\tau = \frac{1}{12}$		
(ρ, α)	MC	Yang	COS-CPD	MC	Yang	COS-CPD	MC	Yang	COS-CPD
(0,0.1)	0.1570	0.79%	1.93%	0.3454	0.58%	0.51%	0.7230	0.57%	2.27%
(0,0.3)	0.1568	0.91%	2.56%	0.3460	0.40%	4.25%	0.7217	0.38%	1.03%
(0,0.5)	0.1569	0.82%	2.14%	0.3463	0.31%	0.86%	0.7257	0.93%	0.35%
(-0.1,0.1)	0.1570	0.78%	6.90%	0.3457	0.48%	1.30%	0.7233	0.60%	0.73%
(-0.3,0.1)	0.1573	0.61%	5.01%	0.3470	0.09%	1.83%	0.7213	0.32%	2.00%
(-0.5,0.1)	0.1572	0.65%	2.56%	0.3441	0.95%	2.03%	0.7205	0.21%	1.01%

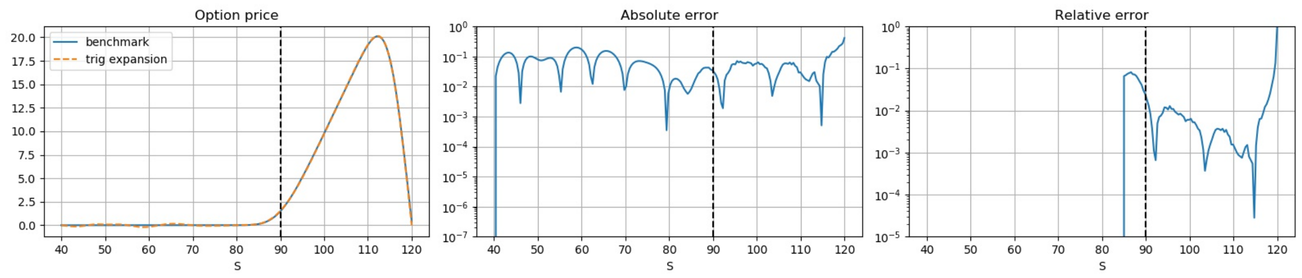
Table 8.4: Comparisons of up-and-out option prices obtained with Monte Carlo, a Closed-Form approximation [13] and the trigonometric expansion.



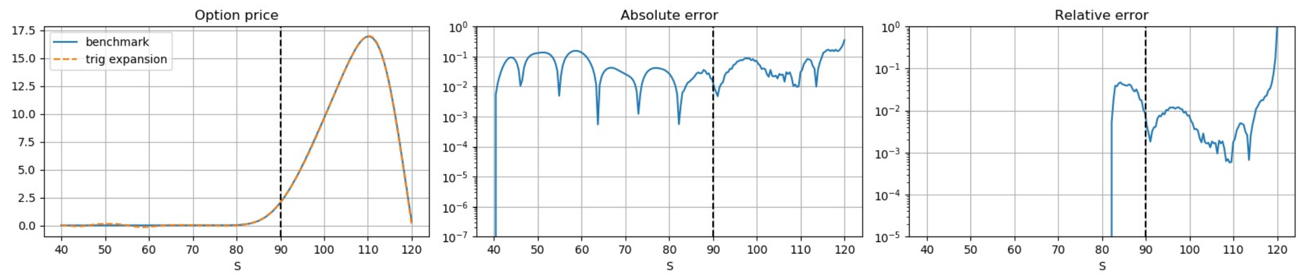
(a) Option price, absolute error and relative error for $\tau = 0$



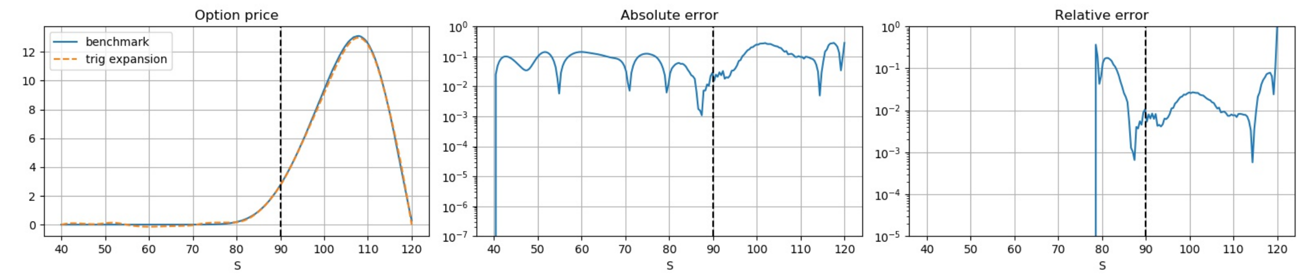
(b) Option price, absolute error and relative error for $\tau = 0.1$



(c) Option price, absolute error and relative error for $\tau = 0.25$



(d) Option price, absolute error and relative error for $\tau = 0.5$



(e) Option price, absolute error and relative error for $\tau = 1$

Figure 8.2: Option price and errors for various τ using trigonometric expansion with CPD vs Monte Carlo benchmark for an up-and-out barrier option with parameters $(r, \alpha, \beta, \rho, \sigma_0) = (0.1, 0.2, 0.5, -0.85)$ and $(a_1, b_1, a_2, b_2, a_3, b_3, E) = (40, 120, 0, 1, 0, 1, 90)$ under regular-asset SABR dynamics with regularization.

8.7. Conclusion

Compared to Chapter 7, applying CPD to the method raised several issues. The first major issue is the condition numbers. Without any regularization, these condition numbers reached order 10^{20}, significantly influencing solutions of the linear systems. Therefore, we applied Ridge regularization to the model to prevent potential overfitting that may have caused the problem. With regularization, the condition numbers decreased to around 10^{5} in some of the testing cases. The results of the model also improved slightly. However, these results are still significantly worse than the results we had in Chapter 6.

This is not fully unexpected, since we also saw a slightly worse results when we included CPD in the model under GBM. Increasing the rank R or number of expansion points K, however, does not give a significant improvement in the accuracy. Since we do observe convergence for R and K in the GBM case, it leads us to believe that CPD errors and series truncation errors are not the dominating errors in the case of SABR.

When we applied the regularization to the testing case from Table 8.4, we again encountered very large condition numbers. Therefore, a direct solver might not be suitable for this model. We have also tried the model using the Conjugate Gradient (CG) method as a solver for the linear systems, but it produced almost identical results.

9 Conclusions and discussion

9.1. Conclusions

In this thesis, we outlined the steps leading to the development of an interpretable neural network for pricing continuously monitored barrier options. Although we primarily discussed up-and-out call barrier options, this method is applicable to any knock-out barrier option.

We began with a simple model assuming the underlying followed GBM dynamics. Using the COS method, we translated the problem into finding the survival ch.f.. We approximated this survival ch.f. with trigonometric expansion, which is based on integrating out the Fourier expansion on the first order derivative with respect to time. By substituting this approximation into the survival ch.f. PDE, we derived a linear system, which can be solved to determine the expansion coefficients. Results are compared with those from other methods in the literature. While the results look promising, it is important to note that the method has less accuracy for very short maturities, due to the fact that one unified truncation range is used across different maturities.

Next, we updated the model to assume SABR dynamics for the underlying. The inclusion of the stochastic volatility, introduced an extra dimension in the trigonometric expansion, making the method suffer from the curse of dimensionality. Consequently, we were limited to $K = 26$ expansion points in each dimension by memory consumption. The results were significantly worse than those for GBM, which was expected as we had fewer expansion points. Compared to literature, our errors were very comparable, except for options with very short maturities. The change of variables that significantly improved results for GBM had only a slight effect here. This change of variables worked well because we matched the density of training points to the area where the function was more variable. With fewer available training points, this density matching was less effective.

Furthermore, we applied CPD to mitigate issues of expansions in higher dimensions. The accuracy of the model with CPD was slightly worse than that without CPD. Additionally, with CPD, we were unable to apply the transformation of variables that previously yielded more accurate results. Furthermore, we encountered very large condition numbers when solving the linear systems.

When we extended the SABR model with CPD, we faced several issues. As with GBM, the model was less accurate than the model without CPD. While we resolved the computational complexity problem, a new error source was included. Increasing the number of expansion points K or rank R did not yield significant improvements, indicating that this new error source dominated the overall error. Regularization was included to improve the performance. This increased accuracy, but not to the level experienced in the model without CPD. Since the only difference between the models in Chapters 6 and 8 is the inclusion of CPD, this error must have originated from CPD. The fact that adding regularization helped to some extent in restoring the accuracy level indicates that the key issue could be in the ALS method. It seems that the optimization procedure based on ALS does not find the global optimum.

To summarize: there is good potential in the method, as we have removed the curse of dimensionality, providing many opportunities. However, more research work is needed to achieve better accuracy.

9.2. Discussion

This thesis still has a few open ends that could benefit from further research. This discussion highlights some of these promising areas.

Error source in CPD

The additional error source introduced by CPD in Chapter 8 is not entirely understood. Increasing the rank from $R = 20$ to $R = 50$ does not seem to improve convergence. Since COS-CPD has been successfully applied to other problems, as seen in [4] and our example in Section 3.4.1, the method itself should have better accuracy. As pointed out earlier, the fact that adding regularization helped to improve the accuracy indicates the optimization procedure based on ALS does not seem to be finding the global optimum.

Other dimension reduction techniques, such as tensor train decomposition [48], could be explored to see if they would improve the performance.

Inverted Log-Asset model

In Section 5.6, we discussed the Inverted Log-Asset transformation for GBM, which yielded much better results than the regular asset model, with an adequate choice of λ. However, when applying the same to the SABR dynamics in Section 6.5, the results did not show a similar performance improvement. This could be due to the relatively low number of expansion points, but this is not certain.

With CPD, this number of expansion points can be increased to properly investigate whether this is indeed the reason. Unfortunately, as mentioned in Chapter 7, the Inverted Log-Asset transformation does not converge to the true solution. Understanding why that is the case requires more investigation. Since the results in the non-CPD model were very promising, this research direction could lead to a significant performance increase.

Model parameters as variables

Currently, our models expand on two or three variables (GBM and SABR, respectively). If we train the model and save the coefficients, all options with the same model parameters can be priced almost instantly. However, if any model parameters differ, the saved coefficients cannot be used and a new set of coefficients must be trained.

With CPD, we have greatly reduced the impact of the curse of dimensionality. Therefore, it might be possible to include extra dimensions corresponding to these model parameters. By expanding on those dimensions, these parameters become inputs for the options price, allowing one training to suffice for all possible options. For this to be applicable, the CPD methods would need to achieve higher accuracy first.

Bibliography

[1] M. C. Designer, *Cover image*, Generated with prompt "create an image that depicts financial markets. Don't show any numbers or letters. Have graphs with green, red, orange line segments and a darker background".

[2] Z. Cheng, *Dimension reduction techniques for multi-dimensional numerical integrations based on fourier-cosine series expansion*, 2022. [Online]. Available: http://resolver.tudelft.nl/uuid:f6ffbcd6-df14-425b-84bb-63e4e105205c.

[3] S. Pereira, *Solving the partial differential equation for pricing barrier options via trigonometric expansions*, 2023. [Online]. Available: http://resolver.tudelft.nl/uuid:d887565e-6df7-4ff2-8a7a-5ba0cee342ec.

[4] M. Brands, *Solving multivariate expectations using dimension-reduced fourier-cosine series expansion and its application in finance*, 2023. [Online]. Available: http://resolver.tudelft.nl/uuid:8d4ba32b-e72e-4092-be41-69c193bc0b11.

[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, no. 2, pp. 826–848, 2009. DOI: 10.1137/080718061.

[6] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, *et al.*, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on signal processing*, vol. 65, no. 13, pp. 3551–3582, 2017. DOI: 10.1109/TSP.2017.2690524.

[7] J. Håstad, "Tensor rank is np-complete," in *Automata, Languages and Programming: 16th International Colloquium Stresa, Italy, July 11–15, 1989 Proceedings 16*, Springer, 1989, pp. 451–460. DOI: 10.1016/0196-6774(90)90014-6.

[8] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of political economy*, vol. 81, no. 3, pp. 637–654, 1973.

[9] P. Hagan, D. Kumar, A. Lesniewski, and D. Woodward, "Managing smile risk," *Wilmott Magazine*, vol. 1, pp. 84–108, Jan. 2002.

[10] B. Dupire *et al.*, "Pricing with a smile," *Risk*, vol. 7, no. 1, pp. 18–20, 1994.

[11] E. Derman and I. Kani, "Riding on a smile," *Risk*, vol. 7, no. 2, pp. 32–39, 1994.

[12] J. Obłój, "Fine-tune your smile: Correction to hagan et al," *arXiv preprint arXiv:0708.0998*, 2007.

[13] 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, no. 2, pp. 116–131, 2017. DOI: 10.3724/SP.J.1383.202006.

[14] P. P. Boyle, "Options: A monte carlo approach," *Journal of financial economics*, vol. 4, no. 3, pp. 323–338, 1977. DOI: 10.1016/0304-405X(77)90005-8.

[15] M. Broadie and Ö. Kaya, "Exact simulation of stochastic volatility and other affine jump diffusion processes," *Operations research*, vol. 54, no. 2, pp. 217–231, 2006. DOI: 10.1287/opre.1050.0247.

[16] N. Cai, Y. Song, and N. Chen, "Exact simulation of the sabr model," *Operations Research*, vol. 65, no. 4, pp. 931–951, 2017. DOI: 10.1287/opre.2017.1617.

[17] M. J. Dilloo and D. Y. Tangman, "A high-order finite difference method for option valuation," *Computers & Mathematics with Applications*, vol. 74, no. 4, pp. 652–670, 2017. DOI: 10.1016/j.camwa.2017.05.006.

[18] N. Thakoor, D. Y. Tangman, and M. Bhuruth, "A spectral approach to pricing of arbitrage-free sabr discrete barrier options," *Computational Economics*, vol. 54, no. 3, pp. 1085–1111, 2019. DOI: 10.1007/s10614-018-9868-8.

[19] D. Tangman, A. Gopaul, and M. Bhuruth, “Numerical pricing of options using high-order compact finite difference schemes,” *Journal of Computational and Applied Mathematics*, vol. 218, no. 2, pp. 270–280, 2008. DOI: 10.1016/j.cam.2007.01.035.

[20] Y. Li, C. N. Sam, Y. Hon, and K. Ng, “An integration preconditioning method for solving option pricing problems,” *International Journal of Computer Mathematics*, vol. 98, no. 2, pp. 367–388, 2021. DOI: 10.1080/00207160.2020.1746960.

[21] J. M. Hutchinson, A. W. Lo, and T. Poggio, “A nonparametric approach to pricing and hedging derivative securities via learning networks,” *The journal of Finance*, vol. 49, no. 3, pp. 851–889, 1994. DOI: 10.1111/j.1540-6261.1994.tb00081.x.

[22] M. Malliaris and L. Salchenberger, “A neural network model for estimating option prices,” *Applied Intelligence*, vol. 3, pp. 193–206, 1993. DOI: 10.1007/BF00871937.

[23] T. Karatas, A. Oskoui, and A. Hirsa, “Supervised deep neural networks (dnns) for pricing/calibration of vanilla/exotic options under various different processes,” in *Peter Carr Gedenkschrift: Research Advances in Mathematical Finance*, World Scientific, 2024, pp. 445–474. DOI: 10.1142/9789811280306_0013.

[24] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009. DOI: 10.1137/07070111X.

[25] C. W. Oosterlee and L. A. Grzelak, *Mathematical modeling and computation in finance: with exercises and Python and MATLAB computer codes*. World Scientific, 2019.

[26] R. L. Karandikar and B. V. Rao, *Introduction to stochastic calculus*. Springer, 2018.

[27] X.-S. Yang, *Engineering Mathematics with Examples and Applications*. Acedemic press, 2017. DOI: https://doi.org/10.1016/B978-0-12-809730-4.00021-5.

[28] A. Hurn, K. Lindsay, and A. McClelland, “On the efficacy of fourier series approximations for pricing european options,” *Applied Mathematics*, pp. 2786–2807, 2014. DOI: 10.4236/am.2014.517267.

[29] M. Ruijter, M. Versteegh, and C. W. Oosterlee, “On the application of spectral filters in a fourier option pricing technique,” *Journal of Computational Finance*, vol. 19, no. 1, pp. 75–106, 2015.

[30] D. Gottlieb and C.-W. Shu, “On the gibbs phenomenon and its resolution,” *SIAM review*, vol. 39, no. 4, pp. 644–668, 1997.

[31] D. Higham, *An Introduction to Financial Option Valuation: Mathematics, Stochastics and Computation*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511800948.

[32] W. Chen, Z.-J. Fu, and C.-S. Chen, *Recent advances in radial basis function collocation methods*. Springer, 2014.

[33] H. Lu, K. N. Plataniotis, and A. Venetsanopoulos, *Multilinear subspace learning: dimensionality reduction of multidimensional data*. CRC press, 2013. DOI: 10.1201/b16252.

[34] R. You, Y. Yao, and J. Shi, “Tensor-based ultrasonic data analysis for defect detection in fiber reinforced polymer (frp) composites,” *Chemometrics and Intelligent Laboratory Systems*, vol. 163, pp. 24–30, 2017. DOI: https://doi.org/10.1016/j.chemolab.2017.02.007.

[35] A. K. Jain, J. Mao, and K. M. Mohiuddin, “Artificial neural networks: A tutorial,” *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[36] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational physics*, vol. 378, pp. 686–707, 2019. DOI: 10.1016/j.jcp.2018.10.045.

[37] S. Liu, “Fourier neural network for machine learning,” in *2013 International Conference on Machine Learning and Cybernetics*, IEEE, vol. 1, 2013, pp. 285–290. DOI: 10.1109/ICMLC.2013.6890482.

[38] M. Ngom and O. Marin, “Fourier neural networks as function approximators and differential equation solvers,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 14, no. 6, pp. 647–661, 2021. DOI: 10.1002/sam.11531.

[39] L. Lu, P. Jin, and G. E. Karniadakis, “Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators,” *arXiv preprint arXiv:1910.03193*, 2019. DOI: 10.1038/s42256-021-00302-5.

[40] Z. Liu, Y. Wang, S. Vaidya, *et al.*, “Kan: Kolmogorov-arnold networks,” *arXiv preprint arXiv:2404.19756*, 2024. DOI: 10.48550/arXiv.2404.19756.

[41] H. Wilbraham, “On a certain periodic function,” *The Cambridge and Dublin Mathematical Journal*, vol. 3, pp. 198–201, 1848. [Online]. Available: https://books.google.com/books?id=JrQ4AAAAMAAJ&pg=PA198.

[42] *Gibbs constant*. [Online]. Available: https://oeis.org/A036792.

[43] N. K. M. Faber, R. Bro, and P. K. Hopke, “Recent developments in candecomp/parafac algorithms: A critical review,” *Chemometrics and Intelligent Laboratory Systems*, vol. 65, no. 1, pp. 119–137, 2003. DOI: 10.1016/S0169-7439(02)00089-8.

[44] J.-h. Jiang, H.-l. Wu, Y. Li, and R.-q. Yu, “Three-way data resolution by alternating slice-wise diagonalization (asd) method,” *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 14, no. 1, pp. 15–36, 2000. DOI: 10.1002/(SICI)1099-128X(200001/02)14:1%3C15::AID-CEM571%3E3.0.CO;2-Z.

[45] G. Tomasi and R. Bro, “A comparison of algorithms for fitting the parafac model,” *Computational Statistics & Data Analysis*, vol. 50, no. 7, pp. 1700–1734, 2006. DOI: 10.1016/j.csda.2004.11.013.

[46] R. Bro and C. A. Andersson, “Improving the speed of multiway algorithms: Part ii: Compression,” *Chemometrics and intelligent laboratory systems*, vol. 42, no. 1-2, pp. 105–113, 1998. DOI: 10.1016/S0169-7439(98)00011-2.

[47] P. Paatero, “A weighted non-negative least squares algorithm for three-way ‘parafac’factor analysis,” *Chemometrics and Intelligent Laboratory Systems*, vol. 38, no. 2, pp. 223–242, 1997. DOI: 10.1016/S0169-7439(97)00031-2.

[48] I. V. Oseledets, “Tensor-train decomposition,” *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011. DOI: 10.1137/090752286.

A Derivations

A.1. Option pricing via Fourier-sine (cosine) series

A.1.1. Option pricing with the sine-method

The steps in this section are analogous to [5], but with the sine expansion instead of cosine expansion. If pdf f and characteristic function ϕ form a Fourier pair, they satisfy the relation

$$\begin{aligned} f(x) &= \frac{1}{2\pi}\int_{\mathbb{R}} \phi(\omega)e^{-i\omega x}d\omega, \\ \phi(\omega) &= \int_{\mathbb{R}} f(x)e^{ix\omega}dx. \end{aligned}$$

Since f will tend to 0 for x very small or large, we can approximate this by setting appropriate bounds a, b where $f(x)$ outside this range is approximately 0.

$$\phi(\omega) = \int_{\mathbb{R}} f(y)e^{i\omega y}dy \approx \int_a^b f(y)e^{i\omega y}dy = \phi_1(\omega).$$

Next, we rewrite

$$\begin{aligned} \sin\left(k\pi\frac{y-a}{b-a}\right) &= \Im\left[e^{ik\pi\frac{y-a}{b-a}}\right] \\ &= \Im\left[e^{iy\frac{k\pi}{b-a}}e^{-ik\pi\frac{a}{b-a}}\right]. \end{aligned}$$

Therefore we can rewrite B_k from the half-range Fourier sine series in Equation 2.3b as

$$\begin{aligned} B_k &= \frac{2}{b-a}\Im\left[\int_a^b f(y)e^{iy\frac{k\pi}{b-a}}dy \cdot e^{ik\pi\frac{-a}{b-a}}\right] \\ &= \frac{2}{b-a}\Im\left[\phi_1\left(\frac{k\pi}{b-a}\right)\cdot e^{ik\pi\frac{-a}{b-a}}\right]. \end{aligned}$$

Next we introduce $F_k \approx B_k$, defined as

$$F_k = \frac{2}{b-a}\Im\left[\phi\left(\frac{k\pi}{b-a}\right)\cdot e^{-ik\pi\frac{a}{b-a}}\right].$$

Since we approximated B_k we can also approximate $f(y)$ using this approximation F_k

$$\begin{aligned} f_1(y) &= \sum_{k=1}^{\infty} F_k \sin\left(k\pi\frac{y-a}{b-a}\right) \\ f_2(y) &= \sum_{k=1}^{N} F_k \sin\left(k\pi\frac{y-a}{b-a}\right). \end{aligned}$$

We recall the pricing formula for a European option as

$$\begin{aligned}V(t,x) &= e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left(V(T,y)|\mathcal{F}_t\right)\\ &= e^{-r(T-t)}\int_{\mathbb{R}} V(T,y)f(y|x)dy.\end{aligned}$$

We apply our approximation of the density function to this conditional density to find and approximation for the option price.

$$\begin{aligned}V(t,X) &= e^{-r(T-t)}\int_{\mathbb{R}} V(T,y)f(y|x)dy\\ V_1(t,X) &= e^{-r(T-t)}\int_a^b V(T,y)f(y|x)dy\\ V_2(t,X) &= e^{-r(T-t)}\int_a^b V(T,y)\sum_{k=1}^{N} F_k\sin\left(k\pi\frac{y-a}{b-a}\right)dy\\ &= e^{-r(T-t)}\int_a^b V(T,y)\sum_{k=1}^{N}\frac{2}{b-a}\Im\left[\phi\left(\frac{k\pi}{b-a},t;x\right)\cdot e^{-ik\pi\frac{a}{b-a}}\right]\sin\left(k\pi\frac{y-a}{b-a}\right)dy\\ &= e^{-r(T-t)}\sum_{k=1}^{N}\Im\left[\phi\left(\frac{k\pi}{b-a},t;x\right)\cdot e^{-ik\pi\frac{a}{b-a}}\right]\frac{2}{b-a}\int_a^b V(T,y)\sin\left(k\pi\frac{y-a}{b-a}\right)dy.\end{aligned}$$

By introducing 2 new variables, we can simplify this expression.

$$\begin{aligned}\hat{\phi}_p(t,X) &= \Im\left[\phi\left(\frac{p\pi}{b-a},t;x\right)\cdot e^{-ip\pi\frac{a}{b-a}}\right]\\ V_p &= \frac{2}{b-a}\int_a^b V(T,y)\sin\left(p\pi\frac{y-a}{b-a}\right)dy\\ V_2(t,X) &= e^{-r(T-t)}\sum_{p=1}^{N}\hat{\phi}_p(t,X)V_p. \end{aligned}\tag{A.1}$$

Here ϕ is the survival ch.f. corresponding to $p(y|x)$. V_p is the coefficient corresponding to the payoff of the option. If we consider a European payoff, we can find an analytic solution for these coefficients V_p. We let X be the asset price and E the strike price. Thenn we have payoff $(X-E)^+$ for a call option and $(E-X)^+$ for a put option. If X is the log-asset price, we have payoffs $(e^X-E)^+$ and $(E-e^X)^+$ for a call and put option respectively. We introduce an integral $\Psi_(c,d)$ defined as

$$\begin{aligned}\Psi_p(c,d) &= \int_c^d \sin\left(p\pi\frac{y-a}{b-a}\right)dy\\ &= \begin{cases}\frac{b-a}{p\pi}\left[\cos\left(p\pi\frac{c-a}{b-a}\right)-\cos\left(p\pi\frac{d-a}{b-a}\right)\right], & p\neq 0\\ 0 & p=0,\end{cases}\end{aligned}\tag{A.2}$$

where $[c,d]\subset[a,b]$. Note that in this case (sine expansion) the case $p=0$ will not occur. We introduce one more integral. This integral will be different for the asset-price case and the log-asset price case. We treat the log-asset price case first.

$$
\begin{aligned}
\chi_p^{LogAsset}(c,d) &= \int_c^d e^y \sin\left(p\pi\frac{y-a}{b-a}\right) dy \\
\chi_p^{LogAsset}(c,d) &= \left[e^y \sin\left(p\pi\frac{y-a}{b-a}\right)\right]_{y=c}^d - \frac{p\pi}{b-a}\int_c^d e^y \cos\left(p\pi\frac{y-a}{b-a}\right) dy \\
&= \left[e^y \sin\left(p\pi\frac{y-a}{b-a}\right)\right]_{y=c}^d - \frac{p\pi}{b-a}\left[e^y \cos\left(p\pi\frac{y-a}{b-a}\right)\right]_{y=c}^d \\
&\quad - \left(\frac{p\pi}{b-a}\right)^2 \chi_p^{LogAsset}(c,d) \\
\left[1+\left(\frac{p\pi}{b-a}\right)^2\right]\chi_p^{LogAsset}(c,d) &= \left[e^y \sin\left(p\pi\frac{y-a}{b-a}\right)\right]_{y=c}^d - \frac{p\pi}{b-a}\left[e^y \cos\left(p\pi\frac{y-a}{b-a}\right)\right]_{y=c}^d \\
\chi_p^{LogAsset}(c,d) &= \frac{1}{1+\left(\frac{p\pi}{b-a}\right)^2}\left[e^d \sin\left(p\pi\frac{d-a}{b-a}\right) - e^c \sin\left(p\pi\frac{c-a}{b-a}\right)\right. \\
&\quad \left. - \frac{p\pi}{b-a}e^d \cos\left(p\pi\frac{d-a}{b-a}\right) + \frac{p\pi}{b-a}e^c \cos\left(p\pi\frac{c-a}{b-a}\right)\right].
\end{aligned}
\tag{A.3}
$$

For the asset price case, the calculations are slightly easier, which results in

$$
\begin{aligned}
\chi_p^{asset}(c,d) &= \int_c^d y \sin\left(p\pi\frac{y-a}{b-a}\right) dy \\
&= \left[-y\frac{b-a}{p\pi}\cos\left(p\pi\frac{y-a}{b-a}\right)\right]_{y=c}^d - \int_c^d -\frac{b-a}{p\pi}\cos\left(p\pi\frac{y-a}{b-a}\right) \\
&= \frac{b-a}{p\pi}\left(c\cos\left(p\pi\frac{c-a}{b-a}\right) - d\cos\left(p\pi\frac{d-a}{b-a}\right)\right) + \frac{b-a}{p\pi}\int_c^d \cos\left(p\pi\frac{y-a}{b-a}\right) \\
&= \frac{b-a}{p\pi}\left(c\cos\left(p\pi\frac{c-a}{b-a}\right) - d\cos\left(p\pi\frac{d-a}{b-a}\right)\right) + \left(\frac{b-a}{p\pi}\right)^2\left[\sin\left(p\pi\frac{y-a}{b-a}\right)\right]_{y=c}^d \\
\chi_p^{asset}(c,d) &= \frac{b-a}{p\pi}\left(c\cos\left(p\pi\frac{c-a}{b-a}\right) - d\cos\left(p\pi\frac{d-a}{b-a}\right)\right) \\
&\quad + \left(\frac{b-a}{p\pi}\right)^2\left(\sin\left(p\pi\frac{d-a}{b-a}\right) - \sin\left(p\pi\frac{c-a}{b-a}\right)\right).
\end{aligned}
\tag{A.4}
$$

Now we can rewrite our coefficients V_p in terms of Ψ and χ from Equations (A.2), (A.4) and (A.3). Note that for a call option with asset price S, $S < E$ results in 0 payoff, so the lower bound of the integral can be set to E. similarly, for a put option, the upper bound can be set to E. Therefore we find

$$
\begin{aligned}
V_p^{LogAsset,call} &= \frac{2}{b-a}\int_{\ln(E)}^b (e^y - E)\sin\left(p\pi\frac{y-a}{b-a}\right) dy \\
V_p^{LogAsset,call} &= \frac{2}{b-a}\left(\chi_p^{LogAsset}(\ln(E),b) - E\Psi_p(\ln(E),b)\right) \\
V_p^{LogAsset,put} &= \frac{2}{b-a}\left(E\Psi_p(a,\ln(E)) - \chi_p^{LogAsset}(a,\ln(E))\right)
\end{aligned}
\tag{A.5}
$$

$$
\begin{aligned}
V_p^{Asset,call} &= \frac{2}{b-a}\int_E^b (y-E)\sin\left(p\pi\frac{y-a}{b-a}\right) dy \\
V_p^{Asset,call} &= \frac{2}{b-a}\left(\chi_p^{Asset}(E,b) - E\Psi_p(E,b)\right) \\
V_p^{Asset,put} &= \frac{2}{b-a}\left(E\Psi_p(a,E) - \chi_p^{LogAsset}(a,E)\right).
\end{aligned}
\tag{A.6}
$$

A.1.2. Survival characteristic function integral under GBM log-asset

The integral of Equation (2.9) can be computed analytically through integration by parts.

$$\begin{aligned}
I_1 =& \int_a^b e^{(i\omega-\alpha)y} \sin\left(k\pi \frac{y-a}{b-a}\right) dy \\
=& \frac{1}{i\omega-\alpha}\left[e^{(i\omega-\alpha)y} \sin\left(k\pi\frac{y-a}{b-a}\right)\right]_{y=a}^{b} - \frac{1}{i\omega-\alpha}\frac{k\pi}{b-a}\int_a^b e^{(i\omega-\alpha)y}\cos\left(k\pi\frac{y-a}{b-a}\right) dy \\
=& \frac{1}{i\omega-\alpha}(0-0) - \frac{1}{i\omega-\alpha}\frac{k\pi}{b-a} I_2 \\
=& -\frac{1}{i\omega-\alpha}\frac{k\pi}{b-a} I_2
\end{aligned}$$

We use the same steps on I_2

$$\begin{aligned}
I_2 =& \int_a^b e^{(i\omega-\alpha)y} \cos\left(k\pi \frac{y-a}{b-a}\right) dy \\
=& \frac{1}{i\omega-\alpha}\left[e^{(i\omega-\alpha)y} \cos\left(k\pi\frac{y-a}{b-a}\right)\right]_{y=a}^{b} + \frac{1}{i\omega-\alpha}\frac{k\pi}{b-a}\int_a^b e^{(i\omega-\alpha)y}\sin\left(k\pi\frac{y-a}{b-a}\right) dy \\
=& \frac{1}{i\omega-\alpha}\left(e^{(i\omega-\alpha)b}\cos(k\pi) - e^{(i\omega-\alpha)a}\right) + \frac{1}{i\omega-\alpha}\frac{k\pi}{b-a} I_1
\end{aligned}$$

Substituting this expression for I_2 into the expression of I_1 yields

$$\begin{aligned}
I_1 =& -\frac{1}{i\omega-\alpha}\frac{k\pi}{b-a}\left(\frac{1}{i\omega-\alpha}\left(e^{(i\omega-\alpha)b}\cos(k\pi) - e^{(i\omega-\alpha)a}\right) + \frac{1}{i\omega-\alpha}\frac{k\pi}{b-a} I_1\right) \\
\left(1 + \left(\frac{1}{i\omega-\alpha}\right)^2\left(\frac{k\pi}{b-a}\right)^2\right) I_1 =& \left(\frac{1}{i\omega-\alpha}\right)^2 \frac{k\pi}{b-a}\left(e^{(i\omega-\alpha)a} - e^{(i\omega-\alpha)b}\cos(k\pi)\right) \\
\left(\frac{(i\omega-\alpha)^2(b-a)^2 + (k\pi)^2}{(i\omega-\alpha)^2(b-a)^2}\right) I_1 =& \left(\frac{1}{i\omega-\alpha}\right)^2 \frac{k\pi}{b-a}\left(e^{(i\omega-\alpha)a} - e^{(i\omega-\alpha)b}\cos(k\pi)\right) \\
\left(\frac{(i\omega-\alpha)^2(b-a)^2 + (k\pi)^2}{(b-a)^2}\right) I_1 =& \frac{k\pi}{b-a}\left(e^{(i\omega-\alpha)a} - e^{(i\omega-\alpha)b}\cos(k\pi)\right) \\
I_1 =& \frac{(b-a)^2}{(i\omega-\alpha)^2(b-a)^2 + (k\pi)^2}\frac{k\pi}{b-a}\left(e^{(i\omega-\alpha)a} - e^{(i\omega-\alpha)b}\cos(k\pi)\right) \\
=& \frac{1}{(i\omega-\alpha)^2 + (\frac{k\pi}{b-a})^2}\frac{k\pi}{b-a}\left(e^{(i\omega-\alpha)a} - e^{(i\omega-\alpha)b}\cos(k\pi)\right)
\end{aligned}$$

A.2. Derivations related to pricing under the SABR model

A.2.1. Martingale approach for SABR pricing PDE

We assume the SABR dynamics from (2.2). Note that our discounted price $e^{-rt}V_t$ should be a martingale under the risk-neutral measure $\mathbb{Q}$. We apply 2-dimensional Itô's lemma on $dV(t,S,\sigma)$, which yields

$$\begin{aligned}
dV_t =& \frac{\partial V}{\partial t}dt + \frac{\partial V}{\partial S}dS + \frac{\partial V}{\partial \sigma}d\sigma + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}dSdS + \frac{1}{2}\frac{\partial^2 V}{\partial \sigma^2}d\sigma d\sigma + \frac{\partial^2 V}{\partial S\partial\sigma}dSd\sigma \\
=& \frac{\partial V}{\partial t}dt + \frac{\partial V}{\partial S}\sigma_t S_t^\beta dW_t + \frac{\partial V}{\partial \sigma}\alpha\sigma_t dZ_t + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma_t^2 S_t^{2\beta}dt + \frac{1}{2}\frac{\partial^2 V}{\partial \sigma^2}\alpha^2\sigma_t^2 dt + \frac{\partial^2 V}{\partial S\partial\sigma}\alpha\sigma_t^2 S_t^\beta \rho dt \\
=& \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma_t^2 S_t^{2\beta}\frac{\partial^2 V}{\partial S^2} + \frac{1}{2}\alpha^2\sigma_t^2\frac{\partial^2 V}{\partial \sigma^2} + \alpha\sigma_t^2 S_t^\beta \rho\frac{\partial^2 V}{\partial S\partial\sigma}\right)dt + \frac{\partial V}{\partial S}\sigma_t S_t^\beta dW_t + \frac{\partial V}{\partial\sigma}\alpha\sigma_t dZ_t.
\end{aligned}$$

With this, we can use Itô's product rule on $de^{-rt}V_t$ to find

$$\begin{aligned} de^{-rt}V_t &= e^{-rt}dV_t + V_t de^{-rt} + de^{-rt}dV_t \\ &= e^{-rt}dV_t - re^{-rt}V_t dt \\ &= e^{-rt}\left[\left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma_t^2 S_t^{2\beta}\frac{\partial^2 V}{\partial S^2} + \frac{1}{2}\alpha^2\sigma_t^2\frac{\partial^2 V}{\partial \sigma^2} + \alpha\sigma_t^2 S_t^\beta \rho\frac{\partial^2 V}{\partial S\partial\sigma} - rV_t\right)dt + \frac{\partial V}{\partial S}\sigma_t S_t^\beta dW_t + \frac{\partial V}{\partial \sigma}\alpha\sigma_t dZ_t\right]. \end{aligned}$$

Since our discounted price is a martingale, this dt-term should be equal to 0, which yields the PDE

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma_t^2 S_t^{2\beta}\frac{\partial^2 V}{\partial S^2} + \frac{1}{2}\alpha^2\sigma_t^2\frac{\partial^2 V}{\partial \sigma^2} + \alpha\sigma_t^2 S_t^\beta \rho\frac{\partial^2 V}{\partial S\partial\sigma} - rV = 0.$$

A.3. CPD

A.3.1. Unfolding and Khatri-Rao product

Although not technically a derivation, it is important to know the order of unfolding that is required for Equation (2.12) to hold. Without loss of generality we consider the three dimensional case unfolding around the second mode, so

$$\mathcal{A} = \sum_{r=1}^{R} \mathbf{a}_r^1 \circ \mathbf{a}_r^2 \circ \mathbf{a}_r^3$$

and

$$\mathcal{A}_{(2)} = \mathbf{A}_2\left(\mathbf{A}_3 \odot \mathbf{A}_1\right)^T,$$

where $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3 \in \mathbb{R}^{N\times R}$ We evaluate the Khatri-Rao product to find

$$\mathbf{A}_3 \odot \mathbf{A}_1 = \begin{pmatrix} \mathbf{a}_1^3 \otimes \mathbf{a}_1^1 & \mathbf{a}_2^3 \otimes \mathbf{a}_2^1 & \cdots & \mathbf{a}_R^3 \otimes \mathbf{a}_R^1 \end{pmatrix}.$$

Now evaluating the matrix product yields

$$\mathbf{A}_2\left(\mathbf{A}_3 \odot \mathbf{A}_1\right)^T = \sum_{r=1}^{R}\begin{pmatrix} c_r[1]\cdot(\mathbf{a}_r^3 \otimes \mathbf{a}_r^1)[1] & c_r[1]\cdot(\mathbf{a}_r^3 \otimes \mathbf{a}_r^1)[2] & \cdots & c_r[1]\cdot(\mathbf{a}_r^3 \otimes \mathbf{a}_r^1)[N^2] \\ c_r[2]\cdot(\mathbf{a}_r^3 \otimes \mathbf{a}_r^1)[1] & c_r[2]\cdot(\mathbf{a}_r^3 \otimes \mathbf{a}_r^1)[2] & \cdots & c_r[2]\cdot(\mathbf{a}_r^3 \otimes \mathbf{a}_r^1)[N^2] \\ \vdots & \vdots & \ddots & \vdots \\ c_r[N]\cdot(\mathbf{a}_r^3 \otimes \mathbf{a}_r^1)[1] & c_r[N]\cdot(\mathbf{a}_r^3 \otimes \mathbf{a}_r^1)[2] & \cdots & c_r[N]\cdot(\mathbf{a}_r^3 \otimes \mathbf{a}_r^1)[N^2] \end{pmatrix}.$$

The kronecker product makes it such that we first iterate over the elements of $\mathbf{a}_r^1$ and then over $\mathbf{a}_r^3$, so the unfolding of $\mathcal{A}_{(2)}$ first iterates over $\mathbf{A}_1$ and then over $\mathbf{A}_3$. For an M-dimensional tensor, this would imply iterating in order $\mathbf{A}_1, \dots, \mathbf{A}_{m-1}, \mathbf{A}_{m+1}, \dots, \mathbf{A}_M$ for mode-m unfolding.

A.4. Deriving trigonometric expansions

We start with Fourier-Sine series expansion on the regular function. Then the expansion on the derivative should be cosine based and expansion on the second derivative should be sine based again.

The expansion on the derivative is followed by simply applying the fundamental theorem of calculus.

$$\begin{aligned} f'(y) &\approx \frac{A_0}{2} + \sum_{k=1}^{K-1} A_k \cos\left(k\pi\frac{y-a}{b-a}\right) \\ \int_a^x f'(y)dy &\approx \int_a^x \frac{A_0}{2} + \sum_{k=1}^{K-1} A_k \cos\left(k\pi\frac{y-a}{b-a}\right)dy \\ f(x) - f(a) &\approx \frac{x-a}{2}A_0 + \sum_{k=1}^{K-1} A_k \int_a^x \cos\left(k\pi\frac{y-a}{b-a}\right)dy \\ f(x) &\approx f(a) + \frac{x-a}{2}A_0 + \sum_{k=1}^{K-1} A_k\frac{b-a}{k\pi}\sin\left(k\pi\frac{y-a}{b-a}\right) \end{aligned}$$

Now we consider sine based expansion for the second derivative

$$
\begin{aligned}
f''(z) &\approx \sum_{k=1}^{K} A_k \sin\left(k\pi\frac{z-a}{b-a}\right)\\
\int_a^x\int_a^y f''(z)dzdy &\approx \int_a^x\int_a^y \sum_{k=1}^{K} A_k \sin\left(k\pi\frac{z-a}{b-a}\right)dzdy\\
\int_a^x f'(y)-f'(a)dy &\approx \int_a^x \sum_{k=1}^{K} -A_k\frac{b-a}{k\pi}\left[\cos\left(k\pi\frac{y-a}{b-a}\right)-1\right]dy\\
f(x)-f(a)-f'(a)(x-a) &\approx \sum_{k=1}^{K} A_k\frac{b-a}{k\pi}\int_a^x\left[1-\cos\left(k\pi\frac{y-a}{b-a}\right)\right]dy\\
f(x)-f(a)-f'(a)(x-a) &\approx \sum_{k=1}^{K} A_k\frac{b-a}{k\pi}\left[x-a-\frac{b-a}{k\pi}\sin\left(k\pi\frac{x-a}{b-a}\right)\right]\\
f(x) \approx f(a)+f'(a)(x-a)+\sum_{k=1}^{K} A_k\frac{b-a}{k\pi}&\left[x-a-\frac{b-a}{k\pi}\sin\left(k\pi\frac{x-a}{b-a}\right)\right].
\end{aligned}
$$

We do the same for the Cosine based expansions. For expansion on the derivative we find

$$
\begin{aligned}
f'(y) &\approx \sum_{k=1}^{K} A_k \sin\left(k\pi\frac{y-a}{b-a}\right)\\
\int_a^x f'(y)dy &\approx \int_a^x \sum_{k=1}^{K} A_k \sin\left(k\pi\frac{y-a}{b-a}\right)dy\\
f(x)-f(a) &\approx \sum_{k=1}^{K} A_k \int_a^x \sin\left(k\pi\frac{y-a}{b-a}\right)dy\\
f(x) \approx f(a)+\sum_{k=1}^{K} A_k&\frac{b-a}{k\pi}\left[1-\cos\left(k\pi\frac{x-a}{b-a}\right)\right].
\end{aligned}
$$

Similarly, for expansion on the second derivative we get

$$
\begin{aligned}
f''(z) &\approx \frac{A_0}{2}+\sum_{k=1}^{K-1} A_k \cos\left(k\pi\frac{z-a}{b-a}\right)\\
\int_a^x\int_a^y f''(z)dzdy &\approx \int_a^x\int_a^y \frac{A_0}{2}+\sum_{k=1}^{K-1} A_k \cos\left(k\pi\frac{z-a}{b-a}\right)dzdy\\
\int_a^x f'(y)-f'(a)dy &\approx \int_a^x \frac{(y-a)}{2}A_0+\sum_{k=1}^{K-1} A_k\frac{b-a}{k\pi}\sin\left(k\pi\frac{y-a}{b-a}\right)dy\\
f(x)-f(a)-f'(a)(x-a) &\approx \frac{A_0}{4}\left(x^2-2ax-a^2\right)+\sum_{k=1}^{K-1} A_k\int_a^x\frac{b-a}{k\pi}\sin\left(k\pi\frac{y-a}{b-a}\right)dy\\
f(x) \approx f(a)+f'(a)(x-a)+\frac{A_0}{4}(x-a)^2&+\sum_{k=1}^{K-1} A_k\left(\frac{b-a}{k\pi}\right)^2\left[1-\cos\left(k\pi\frac{x-a}{b-a}\right)\right].
\end{aligned}
$$

A.5. SABR CPD PDE expressions

The PDE under SABR has a Hadamard product between factor matrices. Therefore a vectorized version of the PDE is not possible. We factor out each of factor matrices to find equivalent equations to the original PDE that can be vectorized. The original PDE from Section 8.3 is:

$$\begin{aligned}
PDE(\mathbf{x}) = &-\left(\mathbf{s}_1^T(x_1)\mathbf{A}_1 \circledast \mathbf{z}_2^T(x_2)\mathbf{A}_2 \circledast \mathbf{s}_3^T(x_3)\mathbf{A}_3\right)\mathbf{1} \\
&+ c_1(\mathbf{x})\left[\left(\mathbf{z}_4^T(x_1)\mathbf{A}_1 \circledast \mathbf{s}_2^T(x_2)\mathbf{A}_2 \circledast \mathbf{s}_3^T(x_3)\mathbf{A}_3\right)\mathbf{1} - \left(\frac{p\pi}{b_1 - a_1}\right)^2 \sin\left(p\pi\frac{x_1 - a_1}{b_1 - a_1}\right)\right] \\
&+ c_2(\mathbf{x})\left(\mathbf{s}_1^T(x_1)\mathbf{A}_1 \circledast \mathbf{s}_2^T(x_2)\mathbf{A}_2 \circledast \mathbf{z}_5^T(x_3)\mathbf{A}_3\right)\mathbf{1} + c_3(\mathbf{x})\left(\mathbf{z}_1^T(x_1)\mathbf{A}_1 \circledast \mathbf{s}_2^T(x_2)\mathbf{A}_2 \circledast \mathbf{z}_3^T(x_3)\mathbf{A}_3\right)\mathbf{1} = 0.
\end{aligned}$$

The PDE where factor matrices are factored out looks like:

$$\begin{aligned}
PDE_1(\mathbf{x}) = &-\mathbf{s}_1^T(x_1)\mathbf{A}_1\mathbf{W}_{1a} + c_1(\mathbf{x})\left[\mathbf{z}_4^T(x_1)\mathbf{A}_1\mathbf{W}_{1b} - \left(\frac{p\pi}{b_1 - a_1}\right)^2 \sin\left(p\pi\frac{x_1 - a_1}{b_1 - a_1}\right)\right] \\
&+ c_2(\mathbf{x})\mathbf{s}_1^T(x_1)\mathbf{A}_1\mathbf{W}_{1c} + c_3(\mathbf{x})\mathbf{z}_1^T(x_1)\mathbf{A}_1\mathbf{W}_{1d} = 0. \\
\mathbf{W}_{1a} =& \left[\mathbf{A}_2^T\mathbf{z}_2(x_2) \circledast \mathbf{A}_3^T\mathbf{s}_3(x_3)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{W}_{1b} =& \left[\mathbf{A}_2^T\mathbf{s}_2(x_2) \circledast \mathbf{A}_3^T\mathbf{s}_3(x_3)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{W}_{1c} =& \left[\mathbf{A}_2^T\mathbf{s}_2(x_2) \circledast \mathbf{A}_3^T\mathbf{z}_5(x_3)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{W}_{1d} =& \left[\mathbf{A}_2^T\mathbf{s}_2(x_2) \circledast \mathbf{A}_3^T\mathbf{z}_3(x_3)\right] \in \mathbb{R}^{R\times 1} \\
PDE_2(\mathbf{x}) = &-\mathbf{z}_2^T(x_2)\mathbf{A}_2\mathbf{W}_{2a} + c_1(\mathbf{x})\left[\mathbf{s}_2^T(x_2)\mathbf{A}_2\mathbf{W}_{2b} - \left(\frac{p\pi}{b_1 - a_1}\right)^2 \sin\left(p\pi\frac{x_1 - a_1}{b_1 - a_1}\right)\right] \\
&+ c_2(\mathbf{x})\mathbf{s}_2^T(x_2)\mathbf{A}_2\mathbf{W}_{2c} + c_3(\mathbf{x})\mathbf{s}_2^T(x_2)\mathbf{A}_2\mathbf{W}_{2d} = 0. \\
\mathbf{W}_{2a} =& \left[\mathbf{A}_1^T\mathbf{s}_1(x_1) \circledast \mathbf{A}_3^T\mathbf{s}_3(x_3)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{W}_{2b} =& \left[\mathbf{A}_1^T\mathbf{z}_4(x_1) \circledast \mathbf{A}_3^T\mathbf{s}_3(x_3)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{W}_{2c} =& \left[\mathbf{A}_1^T\mathbf{s}_1(x_1) \circledast \mathbf{A}_3^T\mathbf{z}_5(x_3)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{W}_{2d} =& \left[\mathbf{A}_1^T\mathbf{z}_1(x_1) \circledast \mathbf{A}_3^T\mathbf{z}_3(x_3)\right] \in \mathbb{R}^{R\times 1} \\
PDE_3(\mathbf{x}) = &-\mathbf{s}_3^T(x_3)\mathbf{A}_3\mathbf{W}_{3a} + c_1(\mathbf{x})\left[\mathbf{s}_3^T(x_3)\mathbf{A}_3\mathbf{W}_{3b} - \left(\frac{p\pi}{b_1 - a_1}\right)^2 \sin\left(p\pi\frac{x_1 - a_1}{b_1 - a_1}\right)\right] \\
&+ c_2(\mathbf{x})\mathbf{z}_5^T(x_3)\mathbf{A}_3\mathbf{W}_{3c} + c_3(\mathbf{x})\mathbf{z}_3^T(x_3)\mathbf{A}_3\mathbf{W}_{3d} = 0. \\
\mathbf{W}_{1a} =& \left[\mathbf{A}_1^T\mathbf{s}_1(x_1) \circledast \mathbf{A}_2^T\mathbf{z}_2(x_2)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{W}_{1b} =& \left[\mathbf{A}_1^T\mathbf{z}_4(x_1) \circledast \mathbf{A}_2^T\mathbf{s}_2(x_2)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{W}_{1c} =& \left[\mathbf{A}_1^T\mathbf{s}_1(x_1) \circledast \mathbf{A}_2^T\mathbf{s}_2(x_2)\right] \in \mathbb{R}^{R\times 1} \\
\mathbf{W}_{1d} =& \left[\mathbf{A}_1^T\mathbf{z}_1(x_1) \circledast \mathbf{A}_2^T\mathbf{s}_2(x_2)\right] \in \mathbb{R}^{R\times 1}
\end{aligned}$$

From this form, we can easily find the vectorized form.

$$PDE_1(\mathbf{x}) = \Bigg(- \Big[\mathbf{W}_{1a}^T \otimes \mathbf{s}_1^T(x_1)\Big] + c_1(\mathbf{x})\Big[\mathbf{W}_{1b}^T \otimes \mathbf{z}_4^T(x_1)\Big] + c_2(\mathbf{x})\Big[\mathbf{W}_{1c}^T \otimes \mathbf{s}_1^T(x_1)\Big]$$
$$+ c_3(\mathbf{x})\Big[\mathbf{W}_{1d}^T \otimes \mathbf{z}_1^T(x_1)\Big]\Bigg)\text{vec}(\mathbf{A}_1) = c_1(\mathbf{x}) \left(\frac{p\pi}{b_1 - a_1}\right)^2 \sin\left(p\pi \frac{x_1 - a_1}{b_1 - a_1}\right)$$
$$PDE_2(\mathbf{x}) = \Bigg(- \Big[\mathbf{W}_{2a}^T \otimes \mathbf{z}_2^T(x_2)\Big] + c_1(\mathbf{x})\Big[\mathbf{W}_{2b}^T \otimes \mathbf{s}_2^T(x_2)\Big] + c_2(\mathbf{x})\Big[\mathbf{W}_{2c}^T \otimes \mathbf{s}_2^T(x_2)\Big]$$
$$+ c_3(\mathbf{x})\Big[\mathbf{W}_{2d}^T \otimes \mathbf{s}_2^T(x_2)\Big]\Bigg)\text{vec}(\mathbf{A}_2) = c_1(\mathbf{x}) \left(\frac{p\pi}{b_1 - a_1}\right)^2 \sin\left(p\pi \frac{x_1 - a_1}{b_1 - a_1}\right)$$
$$PDE_3(\mathbf{x}) = \Bigg(- \Big[\mathbf{W}_{3a}^T \otimes \mathbf{s}_3^T(x_3)\Big] + c_1(\mathbf{x})\Big[\mathbf{W}_{3b}^T \otimes \mathbf{s}_3^T(x_3)\Big] + c_2(\mathbf{x})\Big[\mathbf{W}_{3c}^T \otimes \mathbf{z}_5^T(x_3)\Big]$$
$$+ c_3(\mathbf{x})\Big[\mathbf{W}_{3d}^T \otimes \mathbf{z}_3^T(x_3)\Big]\Bigg)\text{vec}(\mathbf{A}_3) = c_1(\mathbf{x}) \left(\frac{p\pi}{b_1 - a_1}\right)^2 \sin\left(p\pi \frac{x_1 - a_1}{b_1 - a_1}\right)$$

A.6. Decay of Fourier series expansion coefficients

We will look into the order of convergence of Fourier expansion coefficients.

Lemma A.6.1. *Let $f \in L^2[a,b]$ and its derivatives up to $f^{(k-1)}$ be continuous as periodic functions with $f^{(k-1)}(a) = f^{(k-1)}(b) = 0$. Let $f^{(k)}$ be piecewise continuous with a finite number of discontinuities in its derivative. Then the Fourier sine series expansion coefficients are bounded as shown below*

$$f(x) = \sum_{n=1}^{\infty} A_n \sin\left(n\pi \frac{x-a}{b-a}\right)$$
$$|A_n| \leq \frac{C}{n^{k+1}}$$

for some constant C.

Proof. We start by proving for $k = 1$.

Suppose f is continuous as periodic function with $f(a) = 0$, let f' be piecewise continuous and let f'' have finitely many discontinuities. Then

$$\begin{aligned} A_n &= \frac{2}{b-a} \int_a^b f(x) \sin\left(n\pi \frac{x-a}{b-a}\right) dx \\ &= \frac{2}{b-a}\left(-\frac{b-a}{n\pi} \left[f(x) \cdot \cos\left(n\pi \frac{x-a}{b-a}\right)\right]_a^b - \int_a^b -\frac{b-a}{n\pi} f'(x) \cdot \cos\left(n\pi\frac{x-a}{b-a}\right) dx \right) \\ &= \frac{2}{n\pi}\left(-\left[f(b)\cdot\cos(n\pi) - f(a)\cdot\cos(0)\right] + \int_a^b f'(x)\cdot \cos\left(n\pi \frac{x-a}{b-a}\right) dx\right) \end{aligned}$$

Since f is continuous as periodic function, $f(a) = f(b) = 0$. Therefore,

$$-\left[f(b)\cdot\cos(n\pi) - f(a)\right] = 0$$

We can further evaluate the integral with a second step of integration by parts.

$$\begin{aligned}
I &= \int_a^b f'(x)\cos\left(n\pi\frac{x-a}{b-a}\right)dx \\
&= \frac{b-a}{n\pi}\left[f'(x)\sin\left(n\pi\frac{x-a}{b-a}\right)\right]_a^b - \frac{b-a}{n\pi}\int_a^b f''(x)\sin\left(n\pi\frac{x-a}{b-a}\right)dx \\
&= \frac{b-a}{n\pi}\left[f'(b)\sin(n\pi) - f'(a)\sin(0)\right] - \frac{b-a}{n\pi}\int_a^b f''(x)\sin\left(n\pi\frac{x-a}{b-a}\right)dx \\
&= -\frac{b-a}{n\pi}\int_a^b f''(x)\sin\left(n\pi\frac{x-a}{b-a}\right)dx
\end{aligned}$$

Due to piecewise continuity of $f'(x)$, we know that $f''(x)$ can be bounded. Combined with the bound $|\sin(y)| \leq 1, \forall y \in \mathbb{R}$, we can say something about the bounds of expansion coefficients A_n.

$$\begin{aligned}
|A_n| &= \left|\frac{2(b-a)}{(n\pi)^2}\int_a^b f''(x)\sin\left(n\pi\frac{x-a}{b-a}\right)dx\right| \\
&\leq \frac{2(b-a)}{(n\pi)^2}\int_a^b |M_1|dx \\
&= \frac{2(b-a)^2M_1}{(n\pi)^2} \\
&= \frac{2(b-a)^2M_1}{\pi^2}\frac{1}{n^2} = \frac{M_2}{n^2}
\end{aligned}$$

where $M_2 = \frac{2(b-a)^2M_1}{\pi^2}$, which is some new constant in $\mathbb{R}$.

For larger k, so with more continuous derivatives, we repeatedly use integration by parts to find $\frac{C}{n}$ term for each further k. Therefore our bound is $|A_n| \leq \frac{C}{n^{k+1}}, C \in \mathbb{R}$ □

Corollary A.6.0.1. *Let f be piecewise continuous on $[a,b]$ with $|f(a)|, |f(b)| \leq M$ for some $M \in \mathbb{R}$ and let $f'(x)$ piecewise continuous. Then the Fourier sine series expansion coefficients are bounded as shown below*

$$\begin{aligned}
f(x) &= \sum_{n=1}^{\infty} A_n \sin\left(n\pi\frac{x-a}{b-a}\right) \\
|A_n| &\leq \frac{C}{n}
\end{aligned}$$

for some constant C.

Proof. Suppose f is piecewise continuous on $[a,b]$ with $|f(a)|, |f(b)| \leq M_1$ for some $M_1 \in \mathbb{R}$ and let $f'(x)$ piecewise continuous. From piecewise continuity of f and $f'(x)$, we can bound $|f'(x)| \leq M_2$ for some $M_2 \in \mathbb{R}$. Then

$$\begin{aligned}
A_n &= \frac{2}{b-a}\int_a^b f(x)\sin\left(n\pi\frac{x-a}{b-a}\right)dx \\
&= \frac{2}{b-a}\left(-\frac{b-a}{n\pi}\left[f(x)\cdot\cos\left(n\pi\frac{x-a}{b-a}\right)\right]_a^b - \int_a^b -\frac{b-a}{n\pi}f'(x)\cdot\cos\left(n\pi\frac{x-a}{b-a}\right)dx\right) \\
&= \frac{2}{n\pi}\left(-\left[f(b)\cdot\cos(n\pi) - f(a)\cdot\cos(0)\right] + \int_a^b f'(x)\cdot\cos\left(n\pi\frac{x-a}{b-a}\right)dx\right) \\
|A_n| &= \frac{2}{n\pi}\left|-\left[f(b)\cdot\cos(n\pi) - f(a)\right] + \int_a^b f'(x)\cdot\cos\left(n\pi\frac{x-a}{b-a}\right)dx\right| \\
&\leq \frac{2}{n\pi}\left|f(b)\cdot(-1)^n - f(a)\right| + \frac{2}{n\pi}\left|\int_a^b f'(x)\cdot\cos\left(n\pi\frac{x-a}{b-a}\right)dx\right| \\
&\leq \frac{2}{n\pi}\left(\left|f(b)\right|\cdot\left|(-1)^n\right| + \left|f(a)\right|\right) + \frac{2}{n\pi}\int_a^b\left|f'(x)\cdot\cos\left(n\pi\frac{x-a}{b-a}\right)\right|dx \\
&\leq \frac{4M_1}{n\pi} + \frac{2}{n\pi}\int_a^b\left|f'(x)\right|dx \\
&\leq \frac{4M_1}{n\pi} + \frac{2M_2}{n\pi}(b-a) \\
&= \frac{C}{n} \\
C &= \frac{4M_1 + 2M_2(b-a)}{\pi}.
\end{aligned}$$

So we find that the Fourier series coefficients are bounded by $|A_n| \leq \frac{C}{n}, C \in \mathbb{R}$ □