# Dimension-reduced Fourier Cosine Series Expansion based on Tensor Train Decomposition and its Application in Finance

MSc Thesis

Auke Schaap

Delft University of Technology

TU Delft

# Dimension-reduced Fourier Cosine Series Expansion based on Tensor Train Decomposition and its Application in Finance

by

Auke Schaap

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday September 12, 2025 at 15:00.


| | |
|---|---|
| Student number: | 4457919 |
| Thesis committee: | Dr. F. Fang (supervisor) |
| | Prof.Dr.ir. C. Vuik |
| | Prof.Dr. H.M. Schuttelaars |
| Project Duration: | July, 2024 - September, 2025 |
| Institution: | Delft University of Technology, |
| | Faculty of Electrical Engineering, |
| | Mathematics & Computer Science (EEMCS) |
| Master Programme: | Applied Mathematics |
| Specialisation: | Financial Engineering |


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

TU Delft

# Abstract

The valuation of multi-asset financial derivatives requires the computation of expectations with respect to high-dimensional probability distributions, a task that is severely hampered by the curse of dimensionality. Fourier-based methods such as the COS method have proven successful in one or two dimensions, but do not scale well to higher dimensions, as they involve a tensor of Fourier coefficients whose size grows exponentially with the dimension. This thesis develops a new method, COS-TT, which combines the COS method with the tensor train (TT) decomposition to efficiently compute multivariate expectations using only the characteristic function. The approach approximates the Fourier-cosine coefficient tensor in the TT format using cross approximation, based on the DMRG greedy algorithm. The method is shown to be equivalent to the functional tensor train decomposition, and theoretical error bounds are derived that quantify contributions from truncating the Fourier-cosine series, approximating the coefficients with the COS method, truncating the TT decomposition to finite rank, and approximating the coefficient tensor with the DMRG-cross algorithm. The method is applied to pricing European basket options under geometric Brownian motion. As part of this, analytic expressions for the integrals required to price basket options using the COS method are derived, which might be of independent interest. Numerical experiments demonstrate that the COS-TT method converges, and is accurate for pricing European basket options for up to 26 assets, a substantial improvement over previous tensor methods based on the COS method, as well as an improvement over other methods based on cross approximation, and comparable to methods based on tensor completion, without using Chebyshev interpolation. These results show that COS-TT provides a promising new approach for computing high-dimensional expectations from characteristic functions, with potential applications in computational finance and beyond.

# Preface

It's done. It was a long journey, but I made it. This thesis marks the end of my time in Delft. When I started in Delft, I would have never predicted that I would finish with a degree in Applied Mathematics. I have learned so much during this time, and met so many wonderful people. I could not have done it alone.

I would like to thank dr. Fang for her supervision during this project. Her guidance and ideas have been invaluable, and I have learned more than ever. She challenges me to be better, and I am grateful for that. I would also like to thank the other people involved with this research topic. Gijs, who was always there to answer my questions. It means a lot that you took the time to help me. Marnix and Zhimin, for writing such clear theses, which formed the basis for my understanding of the topics.

A special thank you goes out to Samuel, who introduced me to the paper on DMRG greedy. This has been a game changer for my project, and I can not repay this debt. I wish you all the best in your future career, which I am sure will be bright.

I would like to thank my friends for their support during all these years. Jesper, for enduring all my rants about math and programming, and still wanting to hang out. I can not deny that you have been an inspiration to me. You make me want to be the best version of myself. Maaike, for accepting me for who I am, and being there through thick and thin. We have been through so much together, and I am grateful for every moment. Florien, for always being there to listen, and for the countless memories we have made together. Teunis, for being my rock during hard times, and pushing me to cycle harder. Lex and Nabilah, for being my family away from home, and making me feel safe even when I am far away from home.

Veerle, thank you for being my partner in crime. Thank you for being patient with me, for believing in me, and for supporting me. When I'm with you, the world no longer seems so daunting. I am excited for our future together, and I can not wait to spend more time with you. I love you.

Finally, and most of all, I would like to thank my family. Without my parents' support and encouragement, I would not be where I am today. Your belief in me means the world to me, and I am grateful for everything you have done for me. Thank you for always being there for me, and for loving me unconditionally. Jesse, thank you for being my little brother, and for always making me, and everyone else, smile. I am proud of the person you are becoming, through all the challenges you faced.

*Auke Schaap*
*Delft, September 2025*

## Statement on Use of AI

*Artificial intelligence tools such as ChatGPT (GPT-5) and Github Copilot were used during the making of this thesis. This includes, but is not limited to, drafting, writing, refining and restructuring text passages, code and mathematical derivations. Artificial intelligence was not used to analyze, validate or interpret any scientific results, numerical experiments or mathematical derivations. The author has critically reviewed, corrected and approved all content generated with the assistance of AI, ensuring its accuracy, originality and compliance with the TU Delft Code of Conduct and the Netherlands Code of Conduct for Research Integrity. The author of this thesis takes full responsibility for the content of this thesis.*

# Contents

# Ch. 1

# Introduction

The valuation of multi-asset financial derivatives requires the computation of expectations with respect to high-dimensional probability distributions. While in one or two dimensions these problems have been studied extensively[21, 2, 43, 12], these methods do not scale well to higher dimensions, as they suffer from the curse of dimensionality. Developing numerical methods to overcome this challenge is of both theoretical and practical interest in computational finance.

In essence, the problem boils down to solving an integral of the form

$$\mathbb{E}[g(\mathbf{X})] = \int_{\mathbb{R}^d} g(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})\, d\mathbf{x},$$

where $f$ is the joint density function of the random vector $\mathbf{X} \in \mathbb{R}^d$, and $g$ is a function of the random vector. Traditional numerical quadrature methods would require a number of function evaluations that grows exponentially with the dimension $d$, making them infeasible.

Furthermore, the density function $f$ is often not known explicitly, but its characteristic function $\varphi_{\mathbf{X}}(\boldsymbol{\omega}) = \mathbb{E}[\exp(i\boldsymbol{\omega} \cdot \mathbf{X})]$ is available in closed form. This problem has successfully been tackled in low dimensions with Fourier-based methods, such as the COS method [2], which uses the characteristic function to reconstruct the density function with a Fourier-cosine series expansion. However, this does not alleviate the curse of dimensionality, as in multiple dimensions a tensor of Fourier coefficients arises.

Recently, tensor methods have been proposed as a promising approach to tackle multi-asset pricing problems in finance [14, 9, 34, 52, 24, 32]. These methods exploit the low-rank structure of high-dimensional functions to reduce the computational complexity of the problem. They formulate the problem in terms of tensors, which are multi-dimensional arrays, and use tensor decompositions to approximate these tensors in a compressed format, which allows for efficient computation of the integral.

Several tensor decompositions exist, each with its own advantages and disadvantages. Furthermore, the algorithms to compute these decompositions differ in terms of complexity, stability, and robustness.

The tensor train decomposition [40] possesses several desirable properties, among them an algorithm to compute it without requiring access to the full tensor [39]. It has been shown to be successful in integration problems [19], and has successfully been applied to option pricing problems [32].

While the COS method has previously been combined with other tensor decomposition techniques[14, 9, 34, 52], no attempt has been made to combine it with the tensor train decomposition. This thesis aims to fill this gap by developing a new method, called COS-TT, which combines the COS method with the tensor train decomposition to efficiently compute multivariate expectations from the characteristic function.

## 1.1 Contributions

The main contribution of this thesis is the development and analysis of a new method for computing multivariate expectations from their characteristic functions. This method, called COS-TT, combines the COS method for Fourier-cosine series expansions with low-rank tensor decomposition of the coefficient tensor in the tensor-train format. The resulting approximation is shown to be equivalent to the functional tensor train decomposition [6], and theoretical error bounds are derived on the approximation error. The method is applied to pricing European basket options under geometric Brownian motion, and is shown to be highly accurate for up to 26 assets. This is a substantial improvement over previous tensor methods [14, 9] based on the COS method, as well as an improvement over a method based on cross approximation [32], and comparable to a method based on tensor completion [24], without using Chebyshev interpolation.

Further contributions include the derivation of analytic expressions for the integrals required to price basket options using the COS method, which might be of independent interest.

Additionally, a new idea to avoid the exponential scaling of the multidimensional coefficient formula is explored. The method relies on the decomposition of the characteristic function by a functional tensor train decomposition. However, as the cores of this decomposition inherit the regularity of the characteristic function, the number of quadrature nodes required to accurately compute the decomposition is currently too high to be practical.

## 1.2 Related Work

There are several popular alternatives to compute multivariate expectations, that have been applied in computational finance. One successful method worth mentioning is to perform quadrature on sparse grids [3, 16, 15]. This revolves around the Smolyak sum [48], which combines low-dimensional quadrature rules. The result in an approximation that achieves much of the accuracy of full tensor-product quadrature, while using significantly fewer quadrature nodes. This method has also been successfully applied to option pricing problems [44].

There are also other tensor decompositions can be used in place of the tensor train decomposition. One option is the canonical polyadic decomposition (CPD) [30, 29, 13], which represents a tensor as a sum of rank-1 tensors:

$$\mathcal{A} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \otimes \mathbf{a}_r^{(2)} \otimes \ldots \otimes \mathbf{a}_r^{(d)}.$$

Even though it is one of the most widely used decompositions, it has several drawbacks. Most notably, for $d \geq 3$, the sets of tensors with CP rank $\leq R$ are generally non-closed, meaning that a best rank-$R$ approximation may not exist [18, 27, 36]. Such non-existence is not a rare occurrence either [18]. Moreover, even when a valid decomposition exists, existing algorithms may fail to compute it [18, 35].

Nevertheless, the CPD has been successfully applied to option pricing problems [14, 9, 34, 52]. These methods combine the COS method with the CPD to efficiently compute multivariate expectations from their characteristic functions.

Some of the drawbacks of the CPD can be mitigated by using the Tucker decomposition [50, 17, 51], which is stable, and always exists for any tensor. The Tucker decomposition introduces an additional core tensor, which captures the interactions between different modes. However, the decomposition scales exponentially with the order of the tensor, and is, therefore, not suitable for high-dimensional problems. The Hierarchical Tucker decomposition [28, 26, 41] extends Tucker by recursively applying Tucker to subgroups of modes, replacing the

single large core tensor with a hierarchy of small core tensors connected via a dimension tree. Writing in matrix form gives the *tensor train decomposition* which is applied here.

## 1.3 Outline

The rest of this thesis is structured as follows. Chapter 2 introduces the necessary mathematical background on Fourier analysis and the COS method, in both the single dimensional and multi-dimensional setting. This includes an analysis of the approximation errors in section 2.3. Chapter 3 follows up with an introduction to tensor calculus, and introduces the tensor train decomposition and the algorithms to compute it. The discrete concept is extend to a continuous setting in section 3.4, where the functional tensor train decomposition is introduced and its properties are discussed. The new COS-TT method is presented in chapter 4, which describes how the previously introduced concepts can be combined to compute multivariate expectations from their characteristic functions. This includes a novel idea to avoid the exponential scaling of the multidimensional coefficient formula in section 4.3. Error bounds on the approximation error of the COS-TT method are derived in chapter 5. The method is then applied to pricing European basket options under geometric Brownian motion in chapter 6, which also presents several numerical experiments to demonstrate the performance and accuracy of the method. Finally, chapter 7 concludes the thesis with a summary of the results and suggestions for future work.

# Ch. 2

# Fourier Series and the COS Method

This chapter provides theoretical analysis on one specific type of spectral expansion: the Fourier cosine series. This expansion gives an optimal approximation for non-periodic functions on a bounded interval [8]. Section 2.1 introduces Fourier series in more detail. Section 2.2 discusses how the Fourier coefficients can be approximated using the Fourier transform. Convergence properties are discussed in section 2.3, leading to a bound on the approximation error. As this work revolves around multidimensional functions, section 2.4 extends the notion of Fourier cosine series to multiple dimensions, and derives the coefficient approximation in terms of the multidimensional Fourier transform. Finally, section 2.5 applies the COS method to option pricing, deriving the details for pricing without the log-transform.

For an in-depth treatment of spectral methods, see [8], which, together with the older [7] forms the reference for most of this chapter. Reasoning similar to section 2.3 can be found in [23]. Additionally, a formal treatment of Fourier series can be found in [22].

## 2.1 Fourier Series and Cosine/Sine Expansions

**Standard Fourier Series on** $[-\pi, \pi]$ Let $f$ be a $2\pi$–periodic, piecewise smooth function. Its Fourier series on $[-\pi, \pi]$ is

$$f(x) \sim \frac{A_0}{2} + \sum_{k=1}^{\infty} \left( A_k \cos(kx) + B_k \sin(kx) \right),$$

with coefficients

$$A_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx)\, dx, \qquad B_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx)\, dx.$$

Here $\sim$ is written instead of $=$ because, in general, the Fourier series converges only *pointwise* to $f$ at points where $f$ is continuous, whereby the series converges to $f(x)$. However, at points of discontinuity, the series converges to the midpoint value:

$$\lim_{N \to \infty} S_N(x) = \tfrac{1}{2}(f(x^-) + f(x^+)).$$

This is the Dirichlet convergence theorem. Therefore, the equality sign does not hold at discontinuities.

**Fourier Cosine/Sine Series** Suppose $f$ is defined on $[0, \pi]$. Extend $f$ evenly to $[-\pi, \pi]$:

$$f_e(x) = \begin{cases} f(x), & 0 \le x \le \pi, \\ f(-x), & -\pi \le x < 0. \end{cases}$$

Since $f_e$ is even, all sine terms vanish in its Fourier series. This leads to the **Fourier-cosine series**:

$$f(x) \sim \frac{A_0}{2} + \sum_{n=1}^{\infty} A_n \cos(nx), \quad x \in [0, \pi],$$

where

$$A_n = \frac{2}{\pi}\int_0^\pi f(x)\cos(nx)\,dx, \qquad n \geq 0.$$

If instead the *odd extension* is defined

$$f_o(x) = \begin{cases} f(x), & 0 \leq x \leq \pi, \\ -f(-x), & -\pi \leq x < 0, \end{cases}$$

then the **Fourier-sine series** is obtained:

$$f(x) \sim \sum_{n=1}^{\infty} B_n \sin(nx), \quad x \in [0,\pi],$$

where

$$B_n = \frac{2}{\pi}\int_0^\pi f(x)\sin(nx)\,dx, \qquad n \geq 1.$$

**Hilbert Space Expansion Formulation** The space $L^2([0,\pi])$ with inner product

$$\langle f, g\rangle_{[0,\pi]} = \frac{2}{\pi}\int_0^\pi f(x)\,g(x)\,dx$$

is a Hilbert space. The functions

$$\{E_n(x) = \cos(nx) : n = 0, 1, 2, \dots\}$$

form an orthogonal basis for even extensions. Therefore, for $f \in L^2([0,\pi])$, the Fourier cosine expansion can be written as

$$f = \sum_{n=0}^{\infty} \langle f, E_n\rangle_{[0,\pi]}\, E_n,$$

where convergence is understood in the $L^2$ sense.

In the Hilbert space setting, functions are identified up to sets of measure zero, and convergence is in the $L^2$ norm. Thus, we may legitimately write $=$ instead of $\sim$. This is in contrast to the pointwise formulation, where the correct symbol is $\sim$ due to possible discontinuities.

**General interval $[a,b] \in \mathbb{R}$.** If $f$ is defined on an arbitrary finite interval $[a,b] \in \mathbb{R}$ of length $L = b - a$ and set

$$\omega_n = \frac{n\pi}{L}, \qquad n = 0, 1, 2, \dots.$$

then the Fourier-cosine series of $f$ on $[a,b]$ takes the form

$$f(x) \sim \frac{A_0}{2} + \sum_{n=1}^{\infty} A_n \cos(\omega_n(x-a)),$$

with coefficients

$$A_n = \frac{2}{L}\int_a^b f(x)\cos(\omega_n(x-a))\,dx, \qquad n \geq 0.$$

For a compact notation, this is written as

$$f(x) \sim \sum_{n=0}^{\infty}{}' A_n(\omega_n(x-a)), \tag{2.1}$$

where $\sum'$ denotes that the first term, the coefficient value is halved.

**$L^2$ Treatment** Throughout the rest of this chapter, the Fourier-cosine series is always understood in the Hilbert space sense, and therefore the equality sign $=$ is used instead of $\sim$. The functional tensor train decomposition is constructed in $L^2$, through the Hilbert-Schmidt decomposition, and therefore all results are naturally expressed in this norm. To remain consistent, the Fourier-cosine series is therefore also treated in the $L^2$ sense. However, it should be noted that the Gibbs phenomenon may still occur.

Furthermore, if the cosine series is truncated after $K$ terms, an approximation is obtained, which will be written as the mapping

$$f(x) \approx P_K f(x) = \sum_{k=0}^{K}{}' A_k \cos\left(\frac{k\pi}{b-a}(x-a)\right). \tag{2.2}$$

## 2.2 The COS Method

An important insight in [2] is that the Fourier coefficients can be approximated in terms of the Fourier transform of $f$. This allows one to compute the coefficients without access to the original function $f$, by knowing its Fourier transform. In probability theory, this is known as the characteristic function, and it is often easier to obtain in closed form than the probability density function itself. The *Fourier transform* and its inverse are given by[1]

$$\varphi(\omega) = \int_{-\infty}^{\infty} f(x)e^{i\omega x}dx, \qquad f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \varphi(\omega)e^{-i\omega x}d\omega.$$

Additionally, the Fourier transform operator is written as $\mathfrak{F}$, so that $\varphi = \mathfrak{F}f$ and $f = \mathfrak{F}^{-1}\varphi$.

Suppose that a function $f$ is approximated by a Fourier cosine series on the interval $[a, b]$, as in equation (2.2), and suppose that this series is truncated to $K$ terms. Now suppose that $[a, b] \in \mathbb{R}$ is chosen such that

$$\hat{\varphi}(\omega) = \int_a^b f(x)e^{i\omega x}dx \approx \int_{-\infty}^{\infty} f(x)e^{i\omega x}dx = \varphi(\omega).$$

This can often be done accurately thanks to the properties of the Fourier transform. As it can be shown that the coefficients of $f$ can be expressed as

$$A_k = \frac{2}{b-a}\,\mathrm{Re}\left\{\hat{\varphi}\left(\frac{k\pi}{b-a}\right)\exp\left(-i\frac{k\pi}{b-a}a\right)\right\}, \tag{2.3}$$

they can then be accurately approximated by

$$A_k \approx F_k = \frac{2}{b-a}\,\mathrm{Re}\left\{\varphi\left(\frac{k\pi}{b-a}\right)\exp\left(-i\frac{k\pi}{b-a}a\right)\right\}, \tag{2.4}$$

that is, by the Fourier transform on the infinite domain. If $F_k$ is then inserted into the truncated Fourier cosine series,

$$f(x) \approx \sum_{k=0}^{K}{}' F_k \cos\left(\frac{k\pi}{b-a}(x-a)\right), \tag{2.5}$$

[1]There are several conventions for defining the Fourier transform and its inverse. In mathematics, specifically probability theory, it's common to define the forward transform with a positive sign, and the inverse with a negative sign, as done here. The opposite convention is equally valid, and often used in engineering. Additionally, there are several conventions regarding the frequency $\omega$. Probability theory typically uses the non-unitary transform, in terms of angular frequency. As a result of these conventions, the fourier transform and the characteristic function can be linked via

$$\varphi(\omega) = \int_{-\infty}^{\infty} f(x)e^{i\omega x}dx = \mathbb{E}\left[e^{i\omega X}\right].$$

an approximation of $f$ is obtained, using only the Fourier transform of $f$. This result is known as the *COS method*. For example, for probability distributions, this means that the density function can be approximated by a Fourier cosine series, using only the characteristic function.

## 2.3 Error and Convergence Analysis

The original COS paper [2] presents error bounds for the COS method using the supremum norm, or $L^\infty$ norm. Because $L^2$ convergence is used throughout this work to reuse error bounds for the functional tensor train decomposition in section 3.4, the error analysis is redone here in the $L^2$ norm.

For a square-integrable function $f$ with support on $[a, b]$ this approximation introduces the following errors. First, the Fourier cosine series is truncated after $K$ terms, and is therefore not an exact representation of $f$. Furthermore, the Fourier coefficients $A_k$ are approximated by $F_k$, which is based on the approximation of the Fourier transform $\hat{\varphi}$ by $\varphi$, the Fourier transform of $f$.

Using the triangle inequality, the approximation error can be bounded as

$$\|f - \hat{P}_K f\|_{L^2} \leq \underbrace{\|f - P_K f\|_{L^2}}_{\text{series truncation}} + \underbrace{\|P_K f - \hat{P}_K f\|_{L^2}}_{\text{coefficient approximation}}, \tag{2.6}$$

where the series truncation error by $\varepsilon_2 = \|f - P_K f\|_{L^2}$, and the coefficient approximation error by $\varepsilon_3 = \|P_K f - \hat{P}_K f\|_{L^2}$. Both errors can be expressed in norms of $f$, and therefore, the $L^1$ norm of the tail of $f$ is introduced,

$$\|f\|_{L^1(\mathbb{R}\setminus[a,b])} = \int_{\mathbb{R}\setminus[a,b]} |f(t)|dt.$$

**Proposition 1** (Total Approximation Error)**.** The total approximation error can be bounded by

$$\|f - \hat{P}_K f\|_{L^2} \leq CK^{-m}\|f^{(m)}\|_{L^2} + Q\,\|f\|_{L^1(\mathbb{R}\setminus[a,b])},$$

where $C$ and $Q$ are constants, and $m$ is the number of square-integrable weak derivatives of $f$ on $[a, b]$, and $f^{(m)}$ is the $m$-th weak derivative of $f$.

### 2.3.1 Series Truncation Error

The error introduced by truncating the Fourier cosine series after $K$ terms is well understood. Specifically, the series truncation error can be expressed in terms of Fourier coefficients $A_k$ that are neglected in the truncation.

**Proposition 2.** The series truncation error $\varepsilon_2$ is given by

$$\varepsilon_2 = \sqrt{\frac{b-a}{2}} \left( \sum_{k=K+1}^{\infty} |A_k|^2 \right)^{1/2}.$$

This follows from Parseval's identity, and a proof can be found in [22], p. 124, or in section A.2.

Because of this equality, it is worthwhile to investigate the rate of convergence of the Fourier cosine series. This requires introducing some definitions about rates of convergence, taken from [8], or the older [7]. A similar line of reasoning can be found in [23].

**Definition 1** (Algebraic Index of Convergence)**.** *If the coefficients $A_k$ of the series satisfy*

$$A_k = \mathcal{O}(k^{-n}), \quad k >> 1$$

*then $n$ is the* algebraic index of convergence.

**Definition 2** (Exponential Index of Convergence)**.** *If the algebraic index of convergence $n$ is unbounded, the series is said to converge* exponentially. *If $s$ and $q > 0$ are constants, and the coefficients $A_k$ satisfy*

$$A_k = \mathcal{O}(s \exp(-qk^r)), \quad k >> 1$$

*then the* exponential index of convergence *is given by $r$. The convergence is called* subgeometric *when $r < 1$. When $r = 1$, the convergence is either* geometric, *when*

$$A_k = \mathcal{O}(k^{-n} \exp(-qk)), \quad k >> 1,$$

*or* supergeometric, *when*

$$A_k = \mathcal{O}\left(k^{-n} \exp\left(-\frac{k}{j} \ln(k)\right)\right), \quad k >> 1,$$

*for some constant $j > 0$.*

**Proposition 3** (Convergence of Fourier cosine series, [7], p. 70)**.** The convergence of the Fourier cosine series on a finite interval for a function $f$ which is analytic everywhere on the interval, including the endpoints, is geometric, that is,

$$A_k = \mathcal{O}(sk^{-n} \exp(-qk)), \quad k >> 1,$$

for some constants $s$, $n$ and $q$. The constant $q$, called the *asymptotic rate of convergence*, is determined by the location in the complex plane of the nearest singularity of $f$ to the interval of expansion. The exponent $n$ is determined by the type and strength of the singularity.

If the function $f$ has discontinuities[a] either in $f$ itself or in one of its derivatives, then the coefficients will decrease algebraically, as

$$A_k = \mathcal{O}(sk^{-n}), \quad k >> 1,$$

for some constants $s$ and $n$.

[a] This leads to the well-known Gibbs phenomenon, where the series overshoots at the points of discontinuity.

In summary, if $f$ is analytic, the Fourier cosine series converges at a geometric rate, and if $f$ has finite smoothness or has discontinuities, the error decays only algebraically. This result can be formalized by introducing the Sobolev space $H^m([a, b])$, which is the space of functions with square-integrable weak derivatives up to order $m$. That is,

$$H^m([a, b]) = \{f \in L^2([a, b]) : D^\alpha f \in L^2([a, b])\},$$

where $D^\alpha f$ is the $\alpha$-th weak derivative of $f$, and $0 \leq \alpha \leq m$. The following result can then be stated.

**Proposition 4** (Series Truncation Error, [11], p. 270)**.** Let $f \in H^m([a,b])$ for some $m > 0$. Then, the series truncation error can be bounded in the $L^2$ norm as

$$\varepsilon_2 = \|f - P_K f\|_{L^2} \leq CK^{-m}\|f^{(m)}\|_{L^2},$$

where $C$ is a constant independent of $K$ and $f$, and $f^{(m)}$ is the $m$-th weak derivative of $f$.

This follows from Parseval's identity, and a proof can be found in [11, p. 270].

### 2.3.2 Coefficient Approximation Error

The error introduced by approximating the Fourier coefficients $A_k$ by $F_k$ has also been studied well [2, 23, 9]. For example, in [2, Lem. 4.1], an error similar to $|\varepsilon_3|$ is bounded in terms of the tail of $f$. Therefore, define by

$$\varepsilon_4 := \int_{\mathbb{R}\setminus[a,b]} f(t)dt = \|f\|_{L^1(\mathbb{R}\setminus[a,b])} \tag{2.7}$$

the $L^1$ norm of the tail of $f$. Then the coefficient approximation error can be bounded as follows.

**Proposition 5** (Coefficient Approximation Error)**.** The coefficient approximation error can be bounded as

$$\|\varepsilon_3\|_{L^2} \leq \sqrt{\frac{2(K+3)}{b-a}}\,\varepsilon_4,$$

where $\varepsilon_4$ is defined in equation (2.7).

A proof can be found in section A.2. This result shows that the coefficient approximation error is bounded by the $L^1$ norm of the tail of $f$, scaled by a factor that grows with $\sqrt{K}$.

## 2.4 Extension to Multiple Dimensions

The extension of the Fourier cosine series to multiple dimensions is straightforward. To simplify the derivations, some new notation is introduced. First, the multidimensional expansion requires summation over indices $k_1, \ldots, k_d$, up to $K_1, \ldots, K_d$. Summation and integration are then written in a reduced form, i.e.

$$\sum_{\mathbf{k}=0}^{\mathbf{K}} = \sum_{k_1=0}^{K_1} \ldots \sum_{k_d=0}^{K_d} \qquad \text{and} \qquad \int_{\mathbf{a}}^{\mathbf{b}} d\mathbf{x} = \int_{a_1}^{b_1} \ldots \int_{a_d}^{b_d} dx_1 \ldots dx_d,$$

where the indices are gathered in $\mathbf{k} = (k_1, \ldots, k_d)$, the summation limits in $\mathbf{K} = (K_1, \ldots, K_d)$, the integration variables in $\mathbf{x} = (x_1, \ldots, x_d)$, and the integration limits in $\mathbf{a} = (a_1, \ldots, a_d)$ and $\mathbf{b} = (b_1, \ldots, b_d)$.

The construction of the multidimensional series starts by extending the cosine basis into a product of the one-dimensional cosine basis functions. If the scaling factors are included in the basis functions, the resulting basis is orthonormal in $L^2([\mathbf{a}, \mathbf{b}])$, simplifying the expressions even further. The scaling factors are given by

$$\beta_{k_j}^{(j)} = \begin{cases} \sqrt{\frac{1}{b_j - a_j}} & k_j = 0, \\ \sqrt{\frac{2}{b_j - a_j}} & k_j \geq 1. \end{cases} \tag{2.8}$$

Then, the product basis $\Phi_{\mathbf{k}}$ is defined in terms of the one-dimensional basis functions $\phi_{k_j}^{(j)}$ as

$$\Phi_{\mathbf{k}} = \prod_{j=1}^{d} \phi_{k_j}^{(j)}(x_j), \qquad \phi_{k_j}^{(j)}(x_j) = \beta_{k_j}^{(j)} \cos\left(\frac{k_j \pi}{b_j - a_j}(x_j - a_j)\right), \tag{2.9}$$

where $\mathbf{k} \in \mathbb{N}^d$. Because the basis is orthonormal in $L^2([\mathbf{a}, \mathbf{b}])$, the coefficients are defined as

$$\mathcal{C}_{\mathbf{k}} = \langle f, \Phi_{\mathbf{k}} \rangle = \int_{\mathbf{a}}^{\mathbf{b}} f(\mathbf{x}) \Phi_{\mathbf{k}} d\mathbf{x}. \tag{2.10}$$

This also means that there is no need to explicitly scale the first coefficient by $\frac{1}{2}$. This inner product notation will be used throughout this thesis, but it must be noted that in the case of the Fourier coefficients the domain of integration is $[\mathbf{a}, \mathbf{b}]$. The resulting multidimensional Fourier cosine series expansion is then given by

$$f(\mathbf{x}) = \sum_{\mathbf{k}=0}^{\mathbf{K}} \mathcal{C}_{\mathbf{k}} \Phi_{\mathbf{k}}(\mathbf{x}). \tag{2.11}$$

In fact, this is a mapping from $L^2([\mathbf{a}, \mathbf{b}])$ to the space spanned by the set of product basis functions $\{\Phi_{\mathbf{k}}\}_{\mathbf{k}=0}^{\mathbf{K}}$. This map will be indicated by $P_{\mathbf{K}}$, so that

$$P_{\mathbf{K}} f = \sum_{\mathbf{k}=0}^{\mathbf{K}} \mathcal{C}_{\mathbf{k}} \Phi_{\mathbf{k}}. \tag{2.12}$$

**Remark.** It is important to note that due to normalizing the cosine basis the coefficients are different than the coefficients that would be obtained by simply extending the unnormalized basis of equation (2.2) to multiple dimensions. The difference is in the scaling factors $\beta_{k_j}^{(j)}$, which are included in the basis functions here, but not in equation (2.2). This means that the coefficients here are different up to a scaling factor. Formally, the coefficients are given by

$$\mathcal{C}_{\mathbf{k}} = \frac{\langle f, \Phi_{\mathbf{k}} \rangle}{\|\Phi_{\mathbf{k}}\|^2} = \langle f, \Phi_{\mathbf{k}} \rangle,$$

when the basis is normalized. For product basis $\tilde{\Phi}_{\mathbf{k}}$ without scaling factors, i.e.

$$\tilde{\Phi}_{\mathbf{k}}(\mathbf{x}) = \prod_{j=1}^{d} \cos\left(\frac{k_j \pi}{b_j - a_j}(x_j - a_j)\right),$$

the norm is

$$\|\tilde{\Phi}_{\mathbf{k}}\|^2 = \prod_{j=1}^{d} \begin{cases} b_j - a_j, & k_j = 0 \\ \frac{b_j - a_j}{2}, & k_j \geq 1, \end{cases} \tag{2.13}$$

and therefore

$$\tilde{\mathcal{C}}_{\mathbf{k}} = \frac{\langle f, \tilde{\Phi}_{\mathbf{k}} \rangle}{\|\tilde{\Phi}_{\mathbf{k}}\|^2} := \left(\prod_{j=1}^{d} \frac{2}{b_j - a_j}\right) \int_{\mathbf{a}}^{\mathbf{b}} f(\mathbf{x}) \tilde{\Phi}_{\mathbf{k}} d\mathbf{x}, \tag{2.14}$$

as long as the summations where $k_j = 0$ are weighted by $\frac{1}{2}$. Since $\Phi_{\mathbf{k}} = \left(\prod_{j=1}^{d} \beta_{k_j}^{(j)}\right) \tilde{\Phi}_{\mathbf{k}}$

we have the relation

$$\tilde{\mathcal{C}}_{\mathbf{k}} = \frac{\mathcal{C}_{\mathbf{k}}}{\left(\prod_{j=1}^{d} \beta_{k_j}^{(j)}\right) \|\tilde{\Phi}_{\mathbf{k}}\|^2} = \mathcal{C}_{\mathbf{k}} \prod_{j=1}^{d} \begin{cases} \frac{1}{\sqrt{b_j - a_j}}, & k_j = 0, \\ \sqrt{\frac{2}{b_j - a_j}}, & k_j \geq 1. \end{cases}$$

The result in equation (2.5) from the COS method can also be extended to multiple dimensions [31, 43]. The idea is similar, and starts with the multidimensional Fourier transform, defined as

$$f(\mathbf{x}) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \varphi(\boldsymbol{\omega}) e^{-i\boldsymbol{\omega}\cdot\mathbf{x}} d\boldsymbol{\omega}, \qquad \varphi(\boldsymbol{\omega}) = \int_{\mathbb{R}^d} f(\mathbf{x}) e^{i\boldsymbol{\omega}\cdot\mathbf{x}} d\mathbf{x},$$

The Fourier transform operator is again written as $\mathfrak{F}$, so that $\varphi = \mathfrak{F} f$ and $f = \mathfrak{F}^{-1}\varphi$.

Again, supposing that the integral is approximated very well by truncating the domain to $[\mathbf{a}, \mathbf{b}]$, i.e.

$$\hat{\varphi}(\boldsymbol{\omega}) = \int_{\mathbf{a}}^{\mathbf{b}} f(\mathbf{x}) e^{i\boldsymbol{\omega}\cdot\mathbf{x}} d\mathbf{x} \approx \int_{\mathbb{R}^d} f(\mathbf{x}) e^{i\boldsymbol{\omega}\cdot\mathbf{x}} d\mathbf{x} = \varphi(\boldsymbol{\omega}),$$

an identity similar to equation (2.3) can be derived. Using Euler's formula, $e^{i\boldsymbol{\omega}\cdot\mathbf{x}} = \cos(\boldsymbol{\omega}\cdot\mathbf{x}) + i\sin(\boldsymbol{\omega}\cdot\mathbf{x})$,

$$\begin{aligned} \hat{\varphi}(\boldsymbol{\omega}) &= \int_{\mathbf{a}}^{\mathbf{b}} f(\mathbf{x}) e^{i\boldsymbol{\omega}\cdot\mathbf{x}} d\mathbf{x} \\ e^{-i\boldsymbol{\omega}\cdot a}\hat{\varphi}(\boldsymbol{\omega}) &= e^{-i\boldsymbol{\omega}\cdot\mathbf{a}} \int_{\mathbf{a}}^{\mathbf{b}} f(\mathbf{x}) e^{i\boldsymbol{\omega}\cdot\mathbf{x}} d\mathbf{x} \\ e^{-i\boldsymbol{\omega}\cdot a}\hat{\varphi}(\boldsymbol{\omega}) &= \int_{\mathbf{a}}^{\mathbf{b}} f(\mathbf{x}) e^{i\boldsymbol{\omega}\cdot(\mathbf{x}-\mathbf{a})} d\mathbf{x} \\ \mathrm{Re}\left\{e^{-i\boldsymbol{\omega}\cdot a}\hat{\varphi}(\boldsymbol{\omega})\right\} &= \int_{\mathbf{a}}^{\mathbf{b}} f(\mathbf{x}) \cos(\boldsymbol{\omega}\cdot(\mathbf{x}-\mathbf{a})) d\mathbf{x}. \end{aligned} \tag{2.15}$$

To apply it, the following product-to-sum identity is required.

**Lemma 1.** Given $d$ real numbers $\mathbf{x} = (x_1, \ldots, x_d)^\top$,

$$\prod_{j=1}^{d} \cos(x_j) = \frac{1}{2^{d-1}} \sum_{\mathbf{s}\in\mathcal{S}} \cos(\mathbf{s}\cdot\mathbf{x}) = \frac{1}{2^{d-1}} \sum_{\mathbf{s}\in\mathcal{S}} \cos\left(\sum_{j=1}^{d} s_j x_j\right),$$

where the summation is over the set of all sign vectors in $d$ dimensions, with the first component always being 1. That is,

$$\mathcal{S}_d = \{(1, s_2, s_3, \ldots, s_d) : s_i \in \{-1, 1\} \text{ for } i = 2, \ldots, d\},$$

which means that $|\mathcal{S}_d| = 2^{d-1}$.

*Proof.* The proof follows by induction from the fact that the cosine is an even function and from the trigonometric identities. □

Using this identity, the product basis can be rewritten as

$$\prod_{j=1}^{d} \beta_{k_j}^{(j)} \cos\left(\frac{k_j\pi}{b_j - a_j}(x_j - a_j)\right) = \frac{\prod_{j=1}^{d} \beta_{k_j}^{(j)}}{2^{d-1}} \sum_{\mathbf{s}\in\mathcal{S}} \cos\left(\sum_{j=1}^{d} \frac{s_j k_j \pi}{b_j - a_j}(x_j - a_j)\right). \tag{2.16}$$

Substituting this into the coefficients gives

$$\mathcal{C}_{\mathbf{k}} = \int_{\mathbf{a}}^{\mathbf{b}} f(\mathbf{x}) \frac{\prod_{j=1}^{d} \beta_{k_j}^{(j)}}{2^{d-1}} \sum_{\mathbf{s}\in\mathcal{S}} \cos\left( \sum_{j=1}^{d} \frac{s_j k_j \pi}{b_j - a_j} (x_j - a_j) \right) d\mathbf{x} \tag{2.17}$$

$$= \frac{\prod_{j=1}^{d} \beta_{k_j}^{(j)}}{2^{d-1}} \sum_{\mathbf{s}\in\mathcal{S}} \int_{\mathbf{a}}^{\mathbf{b}} f(\mathbf{x}) \cos\left( \sum_{j=1}^{d} \frac{s_j k_j \pi}{b_j - a_j} (x_j - a_j) \right) d\mathbf{x}, \tag{2.18}$$

Combining equation (2.15) and equation (2.18) yields

$$\mathcal{C}_{\mathbf{k}} = \frac{\prod_{j=1}^{d} \beta_{k_j}^{(j)}}{2^{d-1}} \sum_{\mathbf{s}\in\mathcal{S}} \mathrm{Re}\left\{ \exp\left( -i \sum_{j=1}^{d} \frac{s_j k_j \pi a_j}{b_j - a_j} \right) \hat{\varphi}\left( \frac{s_1 k_1 \pi}{b_1 - a_1}, \ldots, \frac{s_d k_d \pi}{b_d - a_d} \right) \right\}. \tag{2.19}$$

Approximating $\hat{\varphi}$ by $\varphi$ then gives the approximation for the coefficients,

$$\mathcal{C}_{\mathbf{k}} \approx \mathcal{F}_{\mathbf{k}} = \frac{\prod_{j=1}^{d} \beta_{k_j}^{(j)}}{2^{d-1}} \sum_{\mathbf{s}\in\mathcal{S}} \mathrm{Re}\left\{ \exp\left( -i \sum_{j=1}^{d} \frac{s_j k_j \pi a_j}{b_j - a_j} \right) \varphi\left( \frac{s_1 k_1 \pi}{b_1 - a_1}, \ldots, \frac{s_d k_d \pi}{b_d - a_d} \right) \right\}. \tag{2.20}$$

This can then be inserted into the multidimensional Fourier cosine series expansion in equation (2.11), and truncated after $\mathbf{K}$ terms, to give the multidimensional COS method:

$$f(\mathbf{x}) \approx P_{\mathbf{K}} f(\mathbf{x}) \approx Q_{\mathbf{K}} f(\mathbf{x}) := \sum_{\mathbf{k}=0}^{\mathbf{K}} \mathcal{F}_{\mathbf{k}} \Phi_{\mathbf{k}}(\mathbf{x}). \tag{2.21}$$

This expression retains the key property of the COS method, which is the ability to express the coefficients in terms of the characteristic function, and can, therefore, be used to approximate multidimensional functions from their characteristic functions. However, this comes at the cost of having to evaluate $2^{d-1}$ terms for each coefficient, which is computationally expensive in high dimensions.

## 2.5 The COS Method for Option Pricing

The COS method arose as a technique for pricing options. As such, some background is provided in this thesis as to how the method can be applied to price options. It closely follows the original derivations in [2]. The focus in this section is on vanilla European options.

An often employed technique in option pricing, is to price in the Fourier domain [12]. For most classes of asset processes the probability density is not known. Instead, its Fourier transform is used, which is typically easier to obtain. For example, for *affine processes* the characteristic function is available by solving a system of Ricatti-type ordinary differential equations [20]. This includes many widely used models in mathematical finance, such as the Lévy processes and Heston model. The application of the COS method for option pricing follows a similar path.

### 2.5.1 Fundamentals

The fundamental problem of option pricing is to determine the arbitrage-free value of a financial derivative based on the underlying assets. This amounts to computing the expectation of the discounted payoff of the option under the risk-neutral measure. For an option expiring at time $T$ with payoff function $\Lambda$ in terms of the asset, assuming a constant interest rate, the price of the option at time $t = 0$ is given by

$$v(x_0, t_0) = e^{-r(T-t_0)} \mathbb{E}^{\mathbb{Q}} \left[ \Lambda(x, T) \mid x_0 \right] = e^{-r(T-t_0)} \int_{\mathbb{R}} \Lambda(x, T) f(x \mid x_0)\, dx$$

where $r$ is the risk-free interest rate, $\mathbb{E}^{\mathbb{Q}}$ is the expectation under the risk-neutral measure and $f(x \mid x_0)$ is the probability density function of the asset price at time $T$, conditional on the asset price at time $t_0$.

Assuming that the density is negligible outside the interval $[a, b]$, the integration range can be safely truncated, so that

$$v(x_0, t_0) \approx e^{-r(T-t_0)} \int_a^b \Lambda(x, T) f(x \mid x_0)\, dx.$$

The density can then be replaced by its Fourier cosine series expansion on $[a, b]$,

$$f(x \mid x_0) = \sum_{k=0}^{\infty}{}' A_k \cos\left(\frac{k\pi}{b-a}(x-a)\right),$$

where the coefficients $A_k$ are defined as

$$A_k(x) = \frac{2}{b-a} \int_a^b f(x \mid x_0) \cos\left(\frac{k\pi}{b-a}(x-a)\right) dx.$$

After substituting this into the pricing function, and interchanging the summation and integration, the pricing function becomes

$$v_{[a,b]}(x_0, t_0) = e^{-r(T-t_0)} \sum_{k=0}^{\infty}{}' A_k(x_0) V_k,$$

where $V_k$ is defined as

$$V_k = \int_a^b \Lambda(x, T) \cos\left(\frac{k\pi}{b-a}(x-a)\right) dx,$$

i.e. the cosine coefficients of the payoff function. Similar to the previous derivations, this series can be truncated after $K$ terms, and the coefficients $A_k$ can be approximated using $F_k$ from equation (2.4). This results in the COS formula for option pricing,

$$v(x_0, t_0) \approx e^{-r(T-t_0)} \sum_{k=0}^{K}{}' \operatorname{Re}\left\{\varphi\left(\frac{k\pi}{b-a} \mid x_0\right) e^{-i\frac{k\pi}{b-a}a}\right\} V_k. \tag{2.22}$$

# Ch. 3

# Tensor Calculus

This chapter provides a brief introduction to tensor calculus. It is assumed that the reader is familiar with basic linear algebra, including matrices and vectors. Section 3.1 will introduce the basic concepts of tensors, and define the notation used throughout this thesis. Afterward, section 3.2 discusses discrete tensor decompositions, which are a powerful tool for approximating and manipulating tensors. These decompositions are the foundation for avoiding the curse of dimensionality.

An in-depth treatment of tensor calculus can be found in [27], which provides both a functional analysis of tensor spaces and a numerical treatment. Additionally, [6] treats functional tensor train decompositions in great detail. This chapter draws heavily on both of these sources.

## 3.1 What is a tensor?

Tensors are a generalization of matrices and vectors that can be used to represent high-dimensional data. A *tensor* will be defined as a $d$-dimensional array of real numbers, and denoted by a calligraphic letter, e.g. $\mathcal{X}$. The natural number $d$ is also called the *order* of the tensor. The numbers in the array are called *elements* or *entries* of the tensor. To identify an element of a tensor, $d$ indices need to be specified, one for each dimension. Analogous to the $i$-th dimension, the names "the $i$-th *mode*" or "the $i$-th *axis*" are often used. The element at position $\mathbf{i} = (i_1, i_2, \dots, i_d)$ of a tensor $\mathcal{X}$ will be denoted by

$$\mathcal{X}_{\mathbf{i}} \quad \text{or} \quad \mathcal{X}[i_1, i_2, \dots, i_d].$$

Each index $i_k$ ranges from 1 to $n_k$, $n_k$ being the size of the $k$-th mode. Let $I_k$ be the set of all indices for mode $k$, i.e. $I_k = \{1, 2, \dots, n_k\}$. Then a $d$-th order tensor $\mathcal{X}$ can be viewed as an element of $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$. Often, $n_k$ is written directly in place of $I_k$, for brevity, e.g. $\mathcal{X} \in \mathbb{R}^{5 \times 2 \times 3}$. Figure 3.1 shows an example of a third-order tensor. The total number of elements of the tensor is given by

$$\prod_{k=1}^{d} n_k = \mathcal{O}(n^d),$$

where $n = \max_k(n_k)$. As the number of elements grows exponentially with the order of the tensor, it is apparent that storage requirements can quickly become incredibly large.

To store a multidimensional array in a computer, it needs to be *flattened* or *unfolded* to a one-dimensional array. This is done by defining a mapping from the $d$-dimensional index to a one-dimensional index, also called a *lexicographic ordering*. Matrices are a simple example, which are often stored in *row-major* or *column-major* order. Tensors can also be unfolded to matrices, which allows tensor operations to be expressed in terms of matrices. This enables the use of highly optimized matrix routines in modern libraries.

### 3.1.1 Operations

Many tensor operations can be defined as generalizations of operations on matrices and vectors. For example, addition and scalar multiplication are easily defined element-wise. More

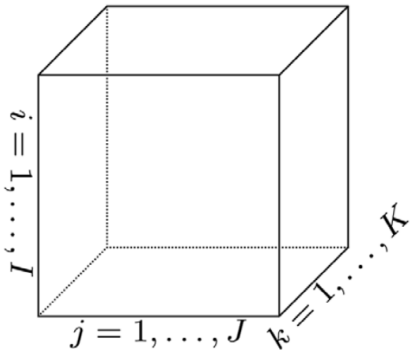


**Figure 3.1:** A visual example of a third order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ [35].

complex operations require a bit more care, as they involve multiple dimensions.

Starting from $d$ vectors $\mathbf{a}^{(j)} \in \mathbb{R}^{I_j}$, with $j = 1, 2, \ldots, d$, the *outer product* of these vectors, written as

$$\mathbf{a}^{(1)} \otimes \mathbf{a}^{(2)} \otimes \ldots \otimes \mathbf{a}^{(d)} = \bigotimes_{i=1}^{d} \mathbf{a}^{(i)}$$

is defined as the product resulting in the $d$-th order tensor with entries

$$\mathcal{X}[i_1, i_2, \ldots, i_d] = a_{i_1}^{(1)} a_{i_2}^{(2)} \ldots a_{i_d}^{(d)}.$$

A tensor constructed in this way is called an *elementary* or *rank one* tensor. Figure 3.2 shows a visual representation of the outer product of three vectors. The resulting tensor is a cube, with each dimension corresponding to one of the vectors.

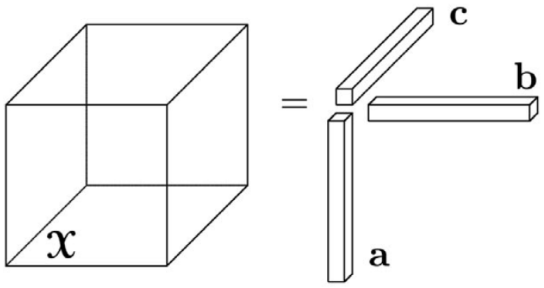


**Figure 3.2:** Tensor $X$ represented as an outer product of three vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ [35].

The *contraction* of two tensors is a generalization of the dot product of vectors and the matrix product. It is defined as the sum over one or more shared indices between two tensors, thereby reducing their order. For instance, consider a third-order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$ and a vector $v \in \mathbb{R}^K$. Contracting over the third mode, a shared index $k$, produces a second order tensor, a matrix $\mathcal{M} \in \mathbb{R}^{I \times J}$, with entries

$$\mathcal{M}[i, j] = \sum_{k=1}^{K} \mathcal{T}[i, j, k]\, v_k.$$

More generally, contracting a tensor $\mathcal{A}$ of order $d$ with a tensor $\mathcal{B}$ of order $e$ over $k$ shared indices results in a new tensor $\mathcal{C}$ of order $d + e - 2k$. A *full contraction* of two tensors is the contraction over all shared indices, resulting in a scalar, analogous to the Euclidean inner product

of vectors. For two tensors $\mathcal{A}, \mathcal{B}$ of order $d$, this is written as

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1} \cdots \sum_{i_d} \mathcal{A}[i_1, i_2, \ldots, i_d] \mathcal{B}[i_1, i_2, \ldots, i_d].$$

An example of this is a weighted sum, which would be an inner product of a rank-one weight tensor with another tensor. As the number of dimensions of the tensors increases, the computational cost of evaluating this inner product grows exponentially, making it a challenging operation to compute in practice.

## 3.2 Discrete Tensor Decomposition

Even though tensors provide a framework to represent calculations with high-dimensional data, they still suffer from the curse of dimensionality, as the number of entries grows exponentially with the tensor's order. This motivates the use of *tensor decompositions*, which exploits structure hidden in the data to yield more compact representations[1]. Such decompositions can substantially reduce storage requirements and computational complexity, thereby mitigating the curse of dimensionality.

One option is the *canonical polyadic decomposition*[2] (CPD), which expresses a given tensor $\mathcal{A}$ as a sum of rank one tensors [30, 13, 29]:

$$\mathcal{A} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \otimes \mathbf{a}_r^{(2)} \otimes \ldots \otimes \mathbf{a}_r^{(d)}.$$

Even though it is one of the most widely used decompositions, it has several drawbacks. Most notably, for $d \geq 3$, the sets of tensors with CP rank $\leq R$ are generally non-closed, meaning that a best rank-$R$ approximation may not exist [18, 27, 36]. Such non-existence is not a rare occurrence either [18]. Moreover, even when a valid decomposition exists, existing algorithms may fail to compute it [18, 35].

### 3.2.1 Tensor Train Decomposition

A better alternative is the *tensor train decomposition* (TT-decomposition) [40], which is numerically stable and guaranteed to exist for any tensor. In a TT-decomposition, the elements of a $d$-th order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ are represented as products of matrices,

$$\mathcal{A}[i_1, i_2, \ldots, i_d] = G_1(i_1)\, G_2(i_2) \cdots G_d(i_d),$$

where each $G_j(i_j)$ is a matrix of size $r_{j-1} \times r_j$, with $r_0 = 1$ and $r_d = 1$. In fact, each $G_j$ can be seen as a three-dimensional tensor of size $r_{j-1} \times n_j \times r_j$, and $G_j(i_j)$ is the $i_j$-th slice of this tensor. Although the more consistent notation would be $\mathcal{G}[:, i_j, :]$ for the $i_j$-th slice of the $j$-th core tensor, the convention in the TT literature is to write $G_j(i_j)$. The integers $r_j$ are called the *rank* of the core tensor $G_j$. By convention, $r_0 = r_d = 1$, which is why the product evaluates to a scalar. An alternative way to write this is elementwise,

$$\mathcal{A}[i_1, i_2, \ldots, i_d] = \sum_{\alpha_1, \ldots, \alpha_{d-1}=1}^{\mathbf{r}} G_1(1, i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \cdots G_d(\alpha_{d-1}, i_d, 1),$$

[1] Note that the terms "decomposition" and "representation" can be seen as two sides of the same coin. In this work, they will be used interchangeably. For a more in-depth discussion, see [27, p. 236].

[2] In literature this decomposition has several names. Hackbusch [27] refers to this as the $r$-term representation, as well as the (canonical) polyadic decomposition. The latter is found in literature as CPD, as well as CANDECOMP. In other areas, specifically chemistry, the name PARAFAC is used, short for *parallel factors* [29].

where $\mathbf{r} = (r_1, \dots, r_{d-1})$. Each core is connected by a shared index $\alpha_k$, over which the tensors are summed. This is where the name *tensor train* comes from.

The TT-format is known under different names in various scientific communities. In quantum physics, it is referred to as a *matrix product state* (MPS) [38, 47], and the algorithms associated with its construction and optimization are often studied under the umbrella of the *density matrix renormalization group* (DMRG) [53, 46, 47]. More generally, these can be seen as a special case of a *tensor network*, a broader framework where a high-order tensor is represented as a network of low-order tensors connected via contracted indices [38, 10]. Throughout this work, however, the term *tensor train* (TT) will be used for consistency with the numerical linear algebra literature.

### 3.2.2 Properties

The TT-decomposition has several nice properties. Most importantly, every tensor admits an exact TT representation [40, Thm. 2.1]. Moreover, in contrast to CP case, the sets of tensors with bounded TT ranks are closed, so best low-rank TT approximations exist and can be computed quasi-optimally [40, Cor. 2.4].

Additionally, storage requirements of a tensor in TT-format scale linearly with the dimension $d$. Consider a $d$-th order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, with $n = \max_k(n_k)$. The full tensor requires $\mathcal{O}(n^d)$ storage. In the TT-format, however, it is only necessary to store the $d$ core tensors $G_k$, each of which has a size of $r_{k-1} \times n_k \times r_k$, where $r_k$ is the rank of the $k$-th core tensor. Letting $r = \max_k(r_k)$, the combined storage requirement is $\mathcal{O}(dnr^2)$.

The TT-format also allows for efficient tensor operations. Specifically, the contraction of a tensor in TT-format with a rank-one tensor can be computed efficiently.

**Theorem 1** (Contraction). Let $\mathcal{X}$ be a tensor of order $d$ with modes $n_1, \dots, n_d$ in TT-format and let $\mathcal{Y}$ be a rank-one tensor, i.e.

$$\mathcal{X}[i_1, \dots, i_d] = \prod_{k=1}^{d} G_k(i_k), \qquad \mathcal{Y}[i_1, \dots, i_d] = \prod_{k=1}^{d} y_{i_k}^{(k)}.$$

Then the inner product $\langle \mathcal{X}, \mathcal{Y} \rangle$ simplifies to

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \prod_{j=1}^{d} \left( \sum_{i_j=1}^{n_j} G_j(i_j) y_{i_j}^{(j)} \right). \tag{3.1}$$

*Proof.* The commutative property of scalar-matrix multiplication allows the grouping into one product.

$$\begin{aligned}
\langle \mathcal{X}, \mathcal{Y} \rangle &= \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \mathcal{X}[i_1, \dots, i_d] \mathcal{Y}[i_1, \dots, i_d] \\
&= \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \prod_{j=1}^{d} G_j(i_j) \prod_{k=1}^{d} y_{i_k}^{(k)} \\
&= \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \prod_{j=1}^{d} G_j(i_j) y_{i_j}^{(j)}
\end{aligned}$$

As each product term only depends on a single index, the sums can be separated.

$$\begin{aligned}\langle \mathcal{X}, \mathcal{Y} \rangle &= \sum_{i_1=1}^{n_1} G_1(i_1) y_{i_1}^{(1)} \cdot \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} \prod_{k=2}^{d} G_k(i_k) y_{i_k}^{(k)} \\ &= \left( \sum_{i_1=1}^{n_1} G_1(i_1) y_{i_1}^{(1)} \right) \left( \sum_{i_2=1}^{n_2} G_2(i_2) y_{i_2}^{(2)} \right) \cdots \left( \sum_{i_d=1}^{n_d} G_d(i_d) y_{i_d}^{(d)} \right) \\ &= \prod_{j=1}^{d} \left( \sum_{i_j=1}^{n_j} G_j(i_j) y_{i_j}^{(j)} \right)\end{aligned}$$

□

This operation is linear in the number of dimensions $d$, instead of exponential, which is a significant advantage in high-dimensional settings, assuming that the tensor rank $r$ is not too large. Specifically, let $r = \max_j r_j$ and $n = \max_j n_j$, the maximum tensor rank and mode size, respectively. The summand is a scalar-matrix product, which has complexity $\mathcal{O}(r^2)$, hence the sum has complexity $\mathcal{O}(nr^2)$. What remains are $d$ matrix-matrix products, which have complexity $\mathcal{O}(r^3)$, and, therefore, total complexity is $\mathcal{O}(dr^3 + nr^2)$.

## 3.3 Finding Decompositions

A central question in applying the tensor train decomposition, is how to compute it in practice. Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ be a $d$-th order tensor. The goal is to find a rank-$r$ TT-decomposition $\mathcal{A}_{\text{TT}}$, such that the approximation error $\mathcal{A} - \mathcal{A}_{\text{TT}}$ is minimal.

This error can be measured in different ways. Common choices are the Frobenius norm, defined as the square root of the sum of the squares of its elements,

$$\|\mathcal{A} - \mathcal{A}_{\text{TT}}\|_F = \sqrt{\sum_{i_1}^{n_1} \cdots \sum_{i_d}^{n_d} \left(\mathcal{A}[i_1, \ldots, i_d] - \mathcal{A}_{\text{TT}}[i_1, \ldots, i_d]\right)^2},$$

or the Chebyshev norm, defined as the maximum absolute value of its elements,

$$\|\mathcal{A} - \mathcal{A}_{\text{TT}}\|_C = \max_{i_1, \ldots, i_d} \left|\mathcal{A}[i_1, \ldots, i_d] - \mathcal{A}_{\text{TT}}[i_1, \ldots, i_d]\right|.$$

Both will be used to derive a meaningful error bound.

Several families of algorithms exist, differing in whether they require access to the full tensor or how ranks are selected. A brief evolution of the algorithms from the perspective of numerical analysis literature is provided[3].

The original construction of the TT-decomposition relies on successive singular value decompositions of unfoldings of the tensor [40], and was named *TT-SVD*. This procedure is deterministic and quasi-optimal, i.e. truncating the SVD at a tolerance $\epsilon$ yields a TT decomposition with a provable error bound of $\sqrt{d-1}\epsilon$ [40, Cor. 2.4]. However, access to the full tensor is required, which is often infeasible in high-dimensional settings.

A way to address this issue is to use tensor cross approximation [39]. These methods aim to construct a low-rank approximation by selecting a small subset of the tensor's elements. This technique stems from linear algebra, where it is known as *cross* or *pseudo-skeleton* approximation [5, 25]. For a matrix, this constitutes selecting a small number of rows and columns to

[3]Because the TT decomposition was rediscovered in the numerical analysis community, and the true evolution encompasses developments in the quantum physics communities as well, these algorithms were not necessarily proposed chronologically.

approximate the full matrix. This idea is extended to tensors by selecting *fibers* of the tensor, the tensor analogue of rows and columns, and arranging them into *slices*, which are submatrices of the tensor.

The *TT-Cross* algorithm [39] constructs a tensor-train decomposition by considering unfoldings of the tensor along successive modes, effectively treating it as a matrix, and applying cross approximation to each unfolding. At each step, *interpolation sets* are chosen, consisting of selected rows and columns (which are fibers of the tensor). A common strategy is to follow the *maximum-volume* principle, which seeks a submatrix of size $r \times r$ with maximal determinant in absolute value. Since finding an exact maximum-volume submatrix is NP-hard [4], practical algorithms employ heuristics to approximate this selection.

A limitation of this approach is that the ranks need to be specified in advance. If these are underestimated, the approximation error can be significant, while overestimating the ranks may lead to unnecessary computational costs. The DMRG approach relaxes this by considering two neighbouring modes simultaneously, a *superblock*, allowing for adaptive rank selection [53, 46]. This rank is determined by computing a full-rank decomposition of the superblock. After splitting the superblock into two TT cores, the algorithm proceeds to the next pair, sweeping back and forth through the train. Instead of using a full-rank decomposition a "greedy" interpolation approach can be used to approximate each superblock, thereby reducing cost even further [45]. This method, called DMRG-greedy, is what is used to compute the TT decomposition throughout this thesis.

An alternative approach is to use Riemannian *tensor completion* [49, 24], which aims to find a low-rank tensor that fits a given set of observed entries. This is useful in the context where only a subset of the tensor's elements are known or can be computed, i.e. the tensor is only partially observed. In this case, however, each element can be computed on demand, so a cross-approximation method is more suitable.

### 3.3.1 Error and Convergence

For the following analysis, let $\mathcal{C}$ be a $d$-th order tensor, and let $\mathbf{r} = (r_1, \dots, r_{d-1})$ be predefined TT ranks. Denote by $\mathcal{C}^\star$ the best rank-$\mathbf{r}$ approximation of $\mathcal{C}$ in the TT format.

In the ideal scenario, a rigorous bound on the approximation error of the algorithms to compute the TT decomposition is provided. The TT-SVD algorithm has such an error bound, which is derived from the properties of the SVD itself. Similar to the matrix case, truncation is performed at each unfolding, and the resulting approximation error can be bounded in terms of the discarded singular values. As a consequence, TT-SVD provides a quasi-optimal approximation, which is summarized as the following proposition.

**Proposition 6** (Quasi-optimality of TT-SVD, [40], Cor. 2.4)**.** Then the tensor $\hat{\mathcal{C}}$ computed by the TT-SVD algorithm satisfies

$$\|\mathcal{C} - \hat{\mathcal{C}}\|_F \leq \sqrt{d-1}\|\mathcal{C} - \mathcal{C}^\star\|_F.$$

For cross-approximation methods the situation is different. Several results exist that provide bounds on the approximation error, but they require careful interpretation. Savostyanov proves that if the interpolation sets are chosen according to the maximum-volume principle, the resulting approximation is quasi-optimal [45].

**Proposition 7** (Quasi-optimality of Maximum-Volume Principle [45], Thm. 1)**.** If, for an approximation $\hat{\mathcal{C}}$ computed with cross approximation, the interpolation sets are chosen according to the maximum-volume principle, and if $\|\mathcal{C} - \mathcal{C}^\star\|_F$ and $\|\mathcal{C} - \mathcal{C}^\star\|_C$ are sufficiently small, then $\hat{\mathcal{C}}$ satisfies

$$\|\mathcal{C} - \hat{\mathcal{C}}\|_C \leq (2r + \kappa r + 1)^{\lceil \log_2(d) \rceil}(r+1)\|\mathcal{C} - \mathcal{C}^\star\|_F,$$
$$\|\mathcal{C} - \hat{\mathcal{C}}\|_C \leq (2r + \kappa r + 1)^{\lceil \log_2(d) \rceil}(r+1)^2\|\mathcal{C} - \mathcal{C}^\star\|_C,$$

where $\kappa$ describes the conditioning of the selected submatrices.

In practice this means that selecting interpolation indices from well-conditioned submatrices ensures that the maximum error is close to the best possible rank-$r$ approximation. The bound does not grow exponentially with the dimension $d$, implying that this property does not blow up in high-dimensional settings. Nevertheless, the bound is still quite pessimistic, and for low values of $d$ and $r$ it can be very large. One of the experiments of Savostyanov shows that the actual error is overestimated by a factor of $> 2^{19}$ [45, Sec. 7.1]. Furthermore, the error bound is in terms of the Chebyshev norm, which means that only the maximum absolute error is controlled. In other words, the tensor does not have big error 'spikes'. Generalizing this to every element of the tensor gives a bound that grows with $\sqrt{n^d}$, which is very loose[4], and does therefore not say much about the stability of the method.

The global stability of cross approximation was instead proven in [42, Thm. 2], which establishes that there exists a choice of subtensors that yields an error bound scaling only logarithmically with the tensor order $d$. Furthermore, this existence result is extended by showing that any "reasonable" choice of interpolation sets gives stability.

**Proposition 8** (Global Stability of Cross Approximation, [42], Thm. 2)**.** For arbitrary choices of interpolation sets that preserve rank, the error of the approximation $\hat{\mathcal{C}}$ can be controlled by

$$\|\mathcal{C} - \hat{\mathcal{C}}\|_F \lesssim a^4(r + cr\epsilon + c^2\epsilon^2)^{\lceil \log_2(d) \rceil} \cdot (\epsilon + c\epsilon^2 + c^2\epsilon^3),$$

where $\epsilon$ is the largest low-rank approximation error in all the unfolding matrices, $a$ describes how well the submatrices capture the singular vectors, and $c$ relates to the conditioning of the selected submatrices. The notation $\lesssim$ means that the inequality holds up to a constant factor.

Reasonable can then be interpreted as having $a$ and $c$ not too large. The maximum-volume principle provides such a choice[42].

However, computing the maximum-volume submatrix is NP-hard [4]. In practice, heuristic algorithms are used to find a good approximation of the maximum-volume submatrix [45]. Therefore, these theoretical results cannot be directly applied to practical algorithms. Nevertheless, they provide insight into the stability of cross approximation methods.

[4]This is because, for $n = \max_k(n_k)$, it holds that

$$\|\mathcal{C} - \hat{\mathcal{C}}\|_F \leq \sqrt{n^d}\|\mathcal{C} - \hat{\mathcal{C}}\|_C.$$

This can be interpreted as the worst-case scenario, where every element of the tensor has the maximum error.

## 3.4 From Discrete to Continuous: Functional Tensor Trains

In order to approximate multidimensional functions directly in their continuous form, the tensor train format can be extended to the functional setting. The resulting functional tensor train (FTT) provides a decomposition of functions rather than discrete tensors, and forms the basis for applying spectral approximation theory to high-dimensional problems. This extension offers theoretical guarantees, such as convergence rates, regularity of the cores, and a relationship between the smoothness of the original function and the rank of the decomposition. A rigorous treatment requires functional analysis, which is beyond the scope of this thesis. Therefore, only a brief introduction is given, so that its theoretical results can be used. The reasoning follows [6].

With the Singular Value Decomposition (SVD), any real $m \times n$ matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top,$$

or, equivalently,

$$\mathbf{A}(i,j) = \sum_{k=1}^{r} \mathbf{u}_k(i)\sigma_k\mathbf{v}_k(j),$$

where $\sigma_k$ are the singular values, and $\mathbf{u}_k(i) \in \mathbb{R}^m$ and $\mathbf{v}_k(j) \in \mathbb{R}^n$ are orthonormal sets of column vectors. A similar idea holds for functions of two variables. If $f(x,y)$ is square integrable, then it can be written as

$$f(x,y) = \sum_{k=1}^{\infty} \sqrt{\lambda(k)}\,\gamma(x;k)\,\varphi(k;y),$$

where each $\gamma(x;k)$ and $\varphi(k;y)$ are orthonormal functions in the $x-$ and $y-$variables, respectively, with $k$ serving as the index. The $\lambda(k)$ play the role of singular values. This is called the *Schmidt decomposition* [6, Def. 7], and can be seen as the functional analogue of the SVD.

This decomposition can be extended to higher dimensions by applying it recursively to different partitions of the variables. A function $g(x_1, x_2, \dots, x_d)$ can be considered as a function of $x_1$ and $(x_2, \dots, x_d)$, and the decomposition would yield

$$g(x_1, \dots, x_d) = \sum_{\alpha_1=1}^{\infty} \sqrt{\lambda(\alpha_1)}\,\gamma_1(x_1;\alpha_1)\,\varphi_1(\alpha_1;x_2,\dots,x_d)$$

This can be continued by applying the same decomposition to $\sqrt{\lambda(\alpha_1)}\varphi(\alpha_1;x_2,\dots,x_d)$. The result is a product of "cores" $\gamma_k$, each depending only on a single variable $x_k$ and two neighbouring index variables $\alpha_{k-1}$ and $\alpha_k$:

$$g(x_1, \dots, x_d) = \sum_{\alpha_1,\dots,\alpha_{d-1}=1}^{\infty} \gamma_1(1;x_1;\alpha_1)\cdots\gamma_d(\alpha_{d-1};x_d;1). \tag{3.2}$$

This is the *functional tensor train decomposition*, the continuous analogue of the discrete tensor train decomposition. Like the discrete version, such a decomposition always exists [6, Def. 7]. Now, each 'core' is a function in one variable, so the $d-$dimensional problem has been reduced to a sequence of 1D building blocks. Truncating the sums to a fixed rank yields an approximation of $g$ in the FTT format. For a rank $\mathbf{r} = (r_1, \dots, r_d)$, this will be written as

$$g(\mathbf{x}) \approx g_{\mathbf{r}\text{-TT}}(\mathbf{x}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \gamma_j(\alpha_{j-1}, x_j, \alpha_j), \tag{3.3}$$

where $\alpha_0 = \alpha_d = 1$, and it is understood that summation starts at $\boldsymbol{\alpha} = (1, \dots, 1)$.

### 3.4.1 Inner product

Because of these 1D building blocks, the FTT inherits the contraction principle, i.e. the efficient inner product.

**Theorem 2.** Let $I_1 \times \cdots \times I_d = \mathbf{I} \subset \mathbb{R}^d$ be a closed and bounded domain, with finite product measure $\mu$, and let $g \in L^2(\mathbf{I})$ be represented in the functional tensor train format as

$$g(x_1, \dots, x_d) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \gamma_j(\alpha_{j-1}, x_j, \alpha_j),$$

and let $w \in L^2(\mathbf{I})$ be a separable function, i.e. $w(x_1, \dots, x_d) = \prod_{j=1}^{d} w_j(x_j)$. Then the inner product $\langle g, w \rangle_{L^2_\mu}$ factors into 1D integrals,

$$\langle g, w \rangle = \prod_{j=1}^{d} \int_{I_j} \gamma_j(\cdot, x_j, \cdot) w_j(x_j) \, d\mu_j(x_j).$$

The proof of this is analogous to the discrete case. Again, the complexity is linear in $d$, instead of exponential. The $d$-dimensional integral has been reduced to a sequence of $d$ one-dimensional integrals. This is what makes the tensor train format powerful for high-dimensional integration.

### 3.4.2 Error and Convergence

Having defined the functional tensor train decomposition, the next question is how well a rank-$\mathbf{r}$ FTT approximates the original function. In this discussion, the domain is taken to be $I_1 \times I_2 \times \cdots \times I_d = \mathbf{I} \subset \mathbb{R}^d$, is assumed closed and bounded, and is equipped with a finite product measure $\mu$. Then, for a function $f \in L^2(\mathbf{I})$, the aim is to derive a bound on the error

$$\| f - f_{\mathbf{r}\text{-TT}} \|_{L^2_\mu}.$$

The FTT plays the same role for functions as the SVD does for matrices. Hence, truncating the decomposition at rank $\mathbf{r}$ yields the best possible approximation of $f$ in the $L^2_\mu$ norm, among all tensor trains with the same rank [6, Prop. 9]. The error is controlled by the singular values $\lambda_i(\alpha_i)$ that were discarded when truncating to rank $\mathbf{r}$.

**Proposition 9** (Prop. 9 [6]). The approximation error of the rank-$\mathbf{r}$ FTT approximation is bounded by the discarded singular values,

$$\| f - f_{\mathbf{r}\text{-TT}} \|^2_{L^2_\mu} \leq \sum_{i=1}^{d-1} \sum_{\alpha_i = r_i + 1}^{\infty} \lambda_i(\alpha_i).$$

Thus, faster decay of the singular values leads to better approximations for a given rank. A key insight is that the decay is related to the smoothness of the function $f$. As a result, smoother functions can be approximated more accurately with lower ranks. This is summarized in the following theorem.

**Theorem 3** (Thm. 13, eq. 56 [6]). Let $f \in L^2(\mathbf{I})$. If $f$ is $k$-times differentiable with bounded derivatives, then the error in of the rank-$\mathbf{r} = (r, \dots, r)$ FTT approximation decays at least algebraically with the rank,

$$\|f - f_{\mathbf{r}\text{-TT}}\|_{L^2_\mu} \leq C \cdot r^{-\frac{k-1}{2}},$$

for some constant $C$ depending on $f$ and the dimension.

Finally, the last important observation is that the cores $\gamma_k$ inherit the regularity of $f$ [6, Thm. 15]. If the original function is continuous, then the cores are also continuous, and if the original function is differentiable, then the cores are differentiable as well. This implies that the spectral approximation theory developed in Chapter 2 can be applied to the cores individually. This theoretical result forms the backbone of the error analysis of the COS-TT method, and will be developed further in Chapter 5.

# Ch. 4

# Our Novel Method: COS-TT

As stated before, the aim is to solve the expectation

$$\mathbb{E}[g(\mathbf{X})] = \int_{\mathbb{R}^d} g(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x}. \tag{4.1}$$

Often in financial mathematics, the function $g$ admits a factorization into lower-dimensional components. Formally, consider a partition $\mathcal{P} = \{I_1, \dots, I_m\}$ of the index set $\{1, \dots, d\}$, such that the subsets are pairwise disjoint and their union covers all indices. Then $g$ can be written as

$$g(\mathbf{x}) = \prod_{j=1}^{m} g_j(x_{I_j}), \tag{4.2}$$

where $g_j : \mathbb{R}^{|I_j|} \to \mathbb{R}$ are functions defined on the lower-dimensional spaces, and $x_{I_j}$ is the subvector of $\mathbf{x}$ corresponding to the indices in $I_j$. The simplest example is the univariate case, where $g(\mathbf{x}) = g_1(x_1) \cdots g_d(x_d)$, and this is assumed throughout this section. However, the methods derived here also hold for the more general case, and can be derived accordingly.

Such a factorization does not imply that the expectation $\mathbb{E}[g(\mathbf{X})]$ can be decomposed into lower-dimensional expectations, since that requires the corresponding random variables to be independent. In other words, the joint-distribution of the random variables must factorize accordingly. This is what the functional tensor train decomposition offers.

With the introduction of the functional tensor train decomposition, it was assumed that $f$ was defined on a closed and bounded domain. Therefore, we consider that $f$ has support on the bounded domain $[\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^d$, where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ with $a_j < b_j$ for all $j$, and the expectation is approximated by

$$\mathbb{E}[g(\mathbf{X})] \approx \int_{[\mathbf{a},\mathbf{b}]} g(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x} = \langle g, f \rangle. \tag{4.3}$$

Thus, the plan is to decompose the probability density function $f(\mathbf{x})$ with a functional tensor train decomposition, and use theorem 2 together with the separability of $g$ to solve the expectation. Three distinct methods are proposed:

1. If the probability density of the process is known, the FTT decomposition can be constructed by evaluating $f$ on a quadrature grid.

2. If the probability distribution is not known, each core of the FTT can be approximated spectrally. It will be shown that a functional tensor train decomposition of $f$ can be constructed differently than the Schmidt decomposition as in [6]. The target function, which is the density function in financial applications, is expanded by a Fourier-cosine series, and the coefficient tensor is replaced by its tensor train decomposition. This is a novel development and will be referred to as **COS-TT**.

3. Alternatively, to avoid computing the Fourier coefficients, theoretically, the joint characteristic function of the process can be decomposed with a functional tensor train decomposition, which is the second novel method developed in this thesis and is referred to as **COS-TT-CHF**.

Each method is discussed in detail in the following sections.

> **Remark.** The derivations in this section are done using the functional tensor train decomposition. A reasoning based on the discrete tensor train decomposition yields the same results, and is often more intuitive. However, to provide rigorous bounds on the errors introduced when applying these methods, the functional tensor train decomposition is required. A derivation using the discrete tensor train decomposition is provided in appendix B to the readers to facilitate their understanding.

## 4.1 Existing Method 1: Use known PDF with FTT

First, consider the case where the probability density function $f(\mathbf{x})$ is known. An FTT of $f$ can then be constructed by evaluating $f$ on a quadrature grid, following the methods in [6]. That is,

$$f(\mathbf{x}) \approx f^{\star}_{\mathbf{r}\text{-TT}}(\mathbf{x}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \gamma_j(\alpha_{j-1}, x_j, \alpha_j), \tag{4.4}$$

where $f^{\star}_{\mathbf{r}\text{-TT}}(\mathbf{x})$ signifies the best rank-$\mathbf{r}$ approximation of $f$ in the functional tensor train format. This FTT is obtained by applying the DMRG-greedy algorithm, resulting in the approximation $\hat{f}_{\mathbf{r}\text{-TT}}(\mathbf{x})$ of the form

$$f^{\star}_{\mathbf{r}\text{-TT}}(\mathbf{x}) \approx \hat{f}_{\mathbf{r}\text{-TT}}(\mathbf{x}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \hat{\gamma}_j(\alpha_{j-1}, x_j, \alpha_j). \tag{4.5}$$

As hinted at, theorem 2 can then be applied to reduce the dimensionality of the problem.

$$\begin{aligned} \langle g, \hat{f}_{\mathbf{r}\text{-TT}} \rangle &= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \langle g_j, \hat{\gamma}_j(\alpha_{j-1}, \cdot, \alpha_j) \rangle \\ &= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \int_{a_j}^{b_j} g_j(x) \hat{\gamma}_j(\alpha_{j-1}, x, \alpha_j)\, dx. \end{aligned}$$

Thus, the problem is reduced to the computation of one-dimensional integrals, which can be approximated using numerical quadrature. The final approximation can be expressed as

$$\boxed{\langle g, \hat{f}_{\mathbf{r}\text{-TT}} \rangle \approx \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \sum_{i_j=1}^{N_j} w_{i_j}\, g_j(x_{i_j}) \hat{\gamma}_j(\alpha_{j-1}, x^{(i_j)}, \alpha_j).} \tag{4.6}$$

## 4.2 Novel Method 1: COS-TT

The construction of the FTT cores requires that the density $f$ is known. If the density of the process is not known, the cores of the FTT cannot be constructed directly. If, instead, the characteristic function of the process is known, the COS method can be applied to construct the cores by a Fourier cosine approximation. This method is named *COS-TT*.

The setup is as in section 2.4, where the coefficient tensor $\mathcal{F}$ is truncated at $\mathbf{K}$. The aim is to show that the method yields an FTT representation of $Q_{\mathbf{K}} f$. If that is the case, the coefficients can be approximated by $\mathcal{F}$.

First, assume that $\mathcal{F}$ admits an *exact* rank-$\mathbf{r}$ TT decomposition

$$\mathcal{F}_{\mathbf{k}} = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} B_j(\alpha_{j-1}, k_j, \alpha_j). \tag{4.7}$$

This decomposition is used to construct a spectral approximation of the cores of $f_{\mathbf{r}\text{-TT}}$. From this decomposition, define, for each $j$, the functional core as

$$\Gamma_j(\alpha_{j-1}, x_j, \alpha_j) = \sum_{k_j=0}^{K_j} B_j(\alpha_{j-1}, k_j, \alpha_j)\phi_{k_j}^{(j)}(x_j). \tag{4.8}$$

In other words, the functional core is spectrally synthesized from the coefficient core and the univariate cosine basis. Note that each core is of rank $\mathbf{r}$. The resulting COS-TT approximation of $f$ is then defined as

$$\boxed{f_{\mathbf{r}\text{-CTT}}(\mathbf{x}) := \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \Gamma_j(\alpha_{j-1}, x_j, \alpha_j).} \tag{4.9}$$

This FTT is clearly rank-$\mathbf{r}$. It is simple to show that this construction actually results in $Q_{\mathbf{K}}f$, thereby justifying the definition.

$$\begin{aligned}
f_{\mathbf{r}\text{-CTT}}(\mathbf{x}) &= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \Gamma_j(\alpha_{j-1}, x_j, \alpha_j) \\
&= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \left( \sum_{k_j=0}^{K_j} B_j(\alpha_{j-1}, k_j, \alpha_j)\phi_{k_j}^{(j)}(x_j) \right) \\
&= \sum_{\mathbf{k}}^{\mathbf{K}} \left( \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} B_j(\alpha_{j-1}, k_j, \alpha_j) \right) \prod_{j=1}^{d} \phi_{k_j}^{(j)}(x_j) \\
&= \sum_{\mathbf{k}}^{\mathbf{K}} \mathcal{F}_{\mathbf{k}} \Phi_{\mathbf{k}}(x).
\end{aligned}$$

In chapter 5, the spectral tensor train decomposition from [6] is derived and it is shown that both routes yield the same result.

### 4.2.1 Computing the expectation

Having defined the COS-TT approximation of $f$, the next step is to show how it can be used to compute the expectation in equation (4.3). This starts by approximating $f$ by the multidimensional COS method approximation truncated at $\mathbf{K}$, i.e.

$$f(\mathbf{x}) \approx Q_{\mathbf{K}} f(\mathbf{x}). \tag{4.10}$$

Subsequently, this is approximated by the best rank-$\mathbf{r}$ COS-TT approximation, yielding

$$Q_{\mathbf{K}} f(\mathbf{x}) \approx f^{\star}_{\mathbf{r}\text{-CTT}}(\mathbf{x}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \Gamma_j(\alpha_{j-1}, x_j, \alpha_j), \tag{4.11}$$

obtained from the best[1] rank-$\mathbf{r}$ TT decomposition of the coefficient tensor $\mathcal{F}_{\mathbf{k}}$. However, in practice that decomposition is obtained by applying the DMRG-greedy algorithm, which yields the approximation $\hat{\mathcal{F}}_{\mathbf{k}}$ of the form

$$\hat{\mathcal{F}}_{\mathbf{k}} = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \hat{B}_j(\alpha_{j-1}, k_j, \alpha_j). \tag{4.12}$$

[1] It is important to note that now it is not assumed that this decomposition is exact.

This is what is used to build the cores of the COS-TT approximation, resulting in

$$f^{\star}_{\mathbf{r}\text{-CTT}}(\mathbf{x}) \approx \hat{f}_{\mathbf{r}\text{-CTT}}(\mathbf{x}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \hat{\Gamma}_j(\alpha_{j-1}, x_j, \alpha_j). \tag{4.13}$$

Having built the COS-TT approximation of $f$, the expectation can again be computed using theorem 2

$$\begin{aligned}
\langle g, \hat{f}_{\mathbf{r}\text{-CTT}} \rangle &= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \langle g_j, \hat{\Gamma}_j(\alpha_{j-1}, \cdot, \alpha_j) \rangle \\
&= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \langle g_j, \sum_{k_j=0}^{K_j} \hat{B}_j(\alpha_{j-1}, k_j, \alpha_j) \phi^{(j)}_{k_j} \rangle \\
&= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \sum_{k_j=0}^{K_j} \hat{B}_j(\alpha_{j-1}, k_j, \alpha_j) \langle g_j, \phi^{(j)}_{k_j} \rangle \\
&= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \sum_{k_j=0}^{K_j} \hat{B}_j(\alpha_{j-1}, k_j, \alpha_j) \int_{a_j}^{b_j} g_j(x) \phi^{(j)}_{k_j}(x)\, dx
\end{aligned} \tag{4.14}$$

Again, only one-dimensional integrals need to be computed, which can be done by numerical quadrature. Because these integrals only depend on $g_j$ and $\phi^{(j)}_{k_j}$, and not on the cores $\hat{B}_j$, it could also be that they can be solved analytically. Specifically, in the case worked out in chapter 6, that is possible, thereby reducing the problem to the computation of $\hat{B}_j$. In the case that they can not be solved, they can at least be precomputed, as they do not depend on the process. The final approximation is given by

$$\boxed{\langle g, \hat{f}_{\mathbf{r}\text{-CTT}} \rangle = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \sum_{k_j=0}^{K_j} \hat{B}_j(\alpha_{j-1}, k_j, \alpha_j) \int_{a_j}^{b_j} g_j(x) \phi^{(j)}_{k_j}(x)\, dx.} \tag{4.15}$$

The method derived here relies on the computation of the coefficients $\mathcal{F}_{\mathbf{k}}$ via equation (2.20). This computation still scales exponentially with $d$, which has a big impact on the computation time of the coefficient cores.

## 4.3 Novel Method 2: COS-TT-CHF

An idea to avoid this exponential scaling, is to directly decompose the joint characteristic function $\varphi(\boldsymbol{\omega})$ of $f$. Let $f \in L^2(\mathbb{R})$, and let $C_{\mathbf{k}}$ be the coefficients of the multidimensional Fourier cosine series expansion of $f$ on $[\mathbf{a}, \mathbf{b}]$, as defined in equation (2.10). To start, replace the joint density by its fourier transform.

$$\begin{aligned}
\mathcal{C}_{\mathbf{k}} &= \int_{\mathbf{a}}^{\mathbf{b}} f(\mathbf{x}) \Phi_{\mathbf{k}}(\mathbf{x})\, d\mathbf{x} \\
&= \frac{1}{(2\pi)^d} \int_{\mathbf{a}}^{\mathbf{b}} \int_{\mathbb{R}^d} \varphi(\boldsymbol{\omega}) e^{-i\boldsymbol{\omega}\cdot\mathbf{x}} \Phi_{\mathbf{k}}(\mathbf{x}) d\boldsymbol{\omega}\, d\mathbf{x}
\end{aligned}$$

Because the basis functions are bounded, i.e. $|\Phi_{\mathbf{k}}(\mathbf{x})| \leq \prod_{j=1}^{d} \beta^{(j)}_{k_j}$, and $|e^{-i\boldsymbol{\omega}\cdot\mathbf{x}}| = 1$, the integrand is absolutely integrable as long as $\varphi \in L^1(\mathbb{R}^d)$. Then Fubini/Tonelli's theorem can be

applied to change the order of integration, yielding

$$\mathcal{C}_{\mathbf{k}} = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \varphi(\boldsymbol{\omega}) \left( \int_{\mathbf{a}}^{\mathbf{b}} e^{-i\boldsymbol{\omega}\cdot\mathbf{x}} \Phi_{\mathbf{k}}(\mathbf{x}) d\mathbf{x} \right) d\boldsymbol{\omega}$$

It turns out this integral is known analytically. A proof can be found in section A.1.

**Lemma 2.** The solution to the inner integral is given by

$$\int_{\mathbf{a}}^{\mathbf{b}} \Phi_{\mathbf{k}}(\mathbf{x}) e^{-i\boldsymbol{\omega}\cdot\mathbf{x}} d\mathbf{x} = \mathbf{I}_{\mathbf{k}}(\boldsymbol{\omega}) = \prod_{j=1}^{d} I_j(\omega_j, k_j), \tag{4.16}$$

which, in the case $\omega_j = 0$, is given by

$$I_j(0, k_j) = \begin{cases} 0 & k_j \neq 0, \\ b_j - a_j & k_j = 0, \end{cases} \tag{4.17}$$

and in the case $\omega_j \neq 0$ is given by

$$I_j(\omega_j, k_j) = \begin{cases} e^{-i\omega_j a_j} \left( \frac{1-e^{-i[(b_j-a_j)\omega_j - k_j\pi]}}{i[(b_j-a_j)\omega_j - k_j\pi]} + \frac{1-e^{-i[(b_j-a_j)\omega_j + k_j\pi]}}{i[(b_j-a_j)\omega_j + k_j\pi]} \right) & k_j \neq 0, \\ \frac{e^{-i\omega_j a_j} - e^{-i\omega_j b_j}}{i\omega_j} & k_j = 0. \end{cases} \tag{4.18}$$

As a result, the coefficients can be written in terms of the joint characteristic function, as

$$\mathcal{C}_{\mathbf{k}} = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \varphi(\boldsymbol{\omega}) \mathbf{I}_{\mathbf{k}}(\boldsymbol{\omega})\, d\boldsymbol{\omega} = \frac{\langle \varphi, \mathbf{I}_{\mathbf{k}} \rangle}{(2\pi)^d}. \tag{4.19}$$

The idea is to approximate the joint characteristic function using a rank-$\mathbf{r}$ FTT, to efficiently compute the inner product. For that to work, $\varphi$ must be defined on a closed and bounded domain. So $\varphi$ with support $\boldsymbol{\omega} \in D_1 \times \ldots \times D_d = \mathbf{D} \subset \mathbb{R}^d$ is considered. Then, the rank-$\mathbf{r}$ FTT approximation of $\varphi$ is defined as

$$\varphi_{\mathbf{r}\text{-TT}}(\boldsymbol{\omega}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \gamma_j(\alpha_{j-1}, \omega_j, \alpha_j).$$

Then $\mathcal{C}_{\mathbf{k}}$ can be approximated as

$$\mathcal{C}_{\mathbf{k}} \approx \frac{\langle \varphi_{\mathbf{r}\text{-TT}}, \mathbf{I}_{\mathbf{k}} \rangle}{(2\pi)^d} = \frac{1}{(2\pi)^d} \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \langle \gamma_j(\alpha_{j-1}, \cdot, \alpha_j), I_j(\cdot, k_j) \rangle \tag{4.20}$$

$$= \frac{1}{(2\pi)^d} \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \int_{D_j} \gamma_j(\alpha_{j-1}, \omega, \alpha_j) I_j(\omega, k_j)\, d\omega. \tag{4.21}$$

This result is more elegant than it seems. Essentially, the coefficients have been written as a TT decomposition, as the inner product above is a function of $k_j$ only. That is, the coefficients are decomposed as

$$\mathcal{C}_{\mathbf{k}} = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} B_j(\alpha_{j-1}, k_j, \alpha_j),$$

where the cores $B_j$ are given by

$$B_j(\alpha_{j-1}, k_j, \alpha_j) = \frac{1}{2\pi}\langle \gamma_j(\alpha_{j-1}, \cdot, \alpha_j), I_j(\cdot, k_j)\rangle.$$

This means that the same derivations as for the COS-TT method can be applied, without ever computing the coefficients explicitly. Thus, with COS-TT-CHF, the FTT of $f$ is given by

$$\boxed{f_{\mathbf{r}\text{-CTT-CHF}}(\mathbf{x}) := \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \Gamma_j(\alpha_{j-1}, x_j, \alpha_j),} \tag{4.22}$$

where the cores are defined as

$$\begin{aligned}
\Gamma_j(\alpha_{j-1}, x_j, \alpha_j) &= \sum_{k_j=0}^{K_j} B_j(\alpha_{j-1}, k_j, \alpha_j)\phi_{k_j}^{(j)}(x_j) \\
&= \sum_{k_j=0}^{K_j} \frac{\langle \gamma_j(\alpha_{j-1}, \cdot, \alpha_j), I_j(\cdot, k_j)\rangle}{2\pi}\phi_{k_j}^{(j)}(x_j) \\
&= \frac{1}{2\pi}\sum_{k_j=0}^{K_j} \phi_{k_j}^{(j)}(x_j) \int_{D_j} \gamma_j(\alpha_{j-1}, \omega, \alpha_j) I_j(\omega, k_j)\, d\omega.
\end{aligned}$$

### 4.3.1 Computing the expectation

With this method, the method to compute the expectation is similar. First, $f$ is approximated by a truncated cosine expansion on $[\mathbf{a}, \mathbf{b}]$, yielding $P_{\mathbf{K}}f$. Then, this cosine expansion is approximated by the best rank-$\mathbf{r}$ COS-TT-CHF approximation, yielding

$$P_{\mathbf{K}}f(\mathbf{x}) \approx f^{\star}_{\mathbf{r}\text{-CTT-CHF}}(\mathbf{x}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \Gamma_j(\alpha_{j-1}, x_j, \alpha_j).$$

The cores of this approximation are defined in terms of the best rank-$\mathbf{r}$ FTT of $\varphi(\boldsymbol{\omega})$, that is,

$$\varphi^{\star}_{\mathbf{r}\text{-TT}}(\boldsymbol{\omega}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \gamma_j(\alpha_{j-1}, \omega_j, \alpha_j).$$

These are constructed using the DMRG-greedy algorithm, yielding the approximation $\hat{\varphi}_{\mathbf{r}\text{-TT}}$ of the form

$$\varphi^{\star}_{\mathbf{r}\text{-TT}}(\boldsymbol{\omega}) \approx \hat{\varphi}_{\mathbf{r}\text{-TT}}(\boldsymbol{\omega}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \hat{\gamma}_j(\alpha_{j-1}, \omega_j, \alpha_j).$$

This means that the cores $\Gamma_j$ are approximated by

$$\Gamma_j(\alpha_{j-1}, x_j, \alpha_j) \approx \hat{\Gamma}_j(\alpha_{j-1}, x_j, \alpha_j) = \frac{1}{2\pi}\sum_{k_j=0}^{K_j} \phi_{k_j}^{(j)}(x_j) \int_{D_j} \hat{\gamma}_j(\alpha_{j-1}, \omega, \alpha_j) I_j(\omega, k_j)\, d\omega.$$

Now that the COS-TT-CHF approximation has been established, the expectation can be computed using theorem 2:

$$\langle g, \hat{f}_{\mathbf{r}\text{-CTT-CHF}}\rangle = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \langle g_j, \hat{\Gamma}_j(\alpha_{j-1}, \cdot, \alpha_j)\rangle$$

$$
\begin{aligned}
&= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \langle g_j, \sum_{k_j=0}^{K_j} \hat{B}_j(\alpha_{j-1}, k_j, \alpha_j) \phi_{k_j}^{(j)} \rangle \\
&= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \sum_{k_j=0}^{K_j} \hat{B}_j(\alpha_{j-1}, k_j, \alpha_j) \langle g_j, \phi_{k_j}^{(j)} \rangle \\
&= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \sum_{k_j=0}^{K_j} \frac{1}{2\pi} \langle \gamma_j(\alpha_{j-1}, \cdot, \alpha_j), I_j(\cdot, k_j) \rangle \cdot \langle g_j, \phi_{k_j}^{(j)} \rangle \, dx \\
&= \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \sum_{k_j=0}^{K_j} \frac{1}{2\pi} \int_{D_j} \gamma_j(\alpha_{j-1}, \omega, \alpha_j) I_j(\omega, k_j) \, d\omega \int_{a_j}^{b_j} g_j(x) \phi_{k_j}^{(j)}(x) \, dx
\end{aligned}
$$

What is left is a product of one-dimensional integrals. Like before, the latter might be solvable analytically. The former needs to be approximated using numerical quadrature. This results in the final approximation for the inner product

$$
\boxed{\langle g, \hat{f}_{\mathbf{r}\text{-CTT-CHF}} \rangle \approx \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \sum_{k_j=0}^{K_j} \frac{1}{2\pi} \sum_{i_j=1}^{N_j} w_{i_j} \hat{\gamma}_j \left(\alpha_{j-1}, \omega^{(i_j)}, \alpha_j\right) I_j \left(\omega^{(i_j)}, k_j\right) \int_{a_j}^{b_j} g_j(x) \phi_{k_j}^{(j)}(x) \, dx.}
\tag{4.23}
$$

# Ch. 5

# Error Analysis

As highlighted during the derivations in chapter 4, several approximations are made. Each of these approximations introduces an error, which accumulates in the final approximation. To rigorously assess the accuracy of the methods, it is crucial to define the error sources and their impact on the overall approximation. This chapter contains an analysis of these errors.

The goal is to provide a theoretical error bound on the function approximation error $f - \hat{f}_{\text{r-CTT}}$. As the COS-TT method is the main contribution of this thesis, it will be the subject of this analysis, but the results of this analysis can be partially reused for the other methods.

The chapter is structured as follows. Section 5.1 provides the setup necessary to formally discuss the error components. Section 5.2 contains the main analysis, where each error component is discussed in detail, and bounds are derived. Finally, these bounds can be applied to the other methods, which is discussed in section 5.3.

## 5.1 Set up

Before delving into the specifics, it is important to set the stage for the analysis by clarifying the assumptions and framework within which the methods operate.

Several norms are possible to measure the error of the approximations. To derive a central result, the $L^2$ norm is used throughout this chapter. This choice is motivated by the construction of the functional tensor train decomposition in [6], which is constructed in $L^2$, and therefore all results are naturally expressed in this norm. While this provides a solid description of the overall error, it does not directly translate to pointwise error bounds. Specifically, as Fourier series approximations are known to exhibit the Gibbs phenomenon, it is still relevant to study pointwise errors. For this analysis, the reader is referred to the relevant section 2.3.

The starting assumption is that $f$ is defined on a closed and bounded domain. However, as $f$ can be a density defined on $\mathbb{R}^d$, $f$ is treated as having support on the bounded domain $[\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^d$, where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ with $a_j < b_j$ for all $j$.

Secondly, one of the key results of the error analysis in section 3.4.2, is that the cores of the FTT inherit the regularity of $f$. For this to hold, some restrictions are needed on $f$. Ideally, these are presented without introducing too much functional analysis. However, at least the following needs to be defined. Define by $\mathcal{H}^k$ the Sobolev spaces

$$\mathcal{H}^k = \left\{ f \in L^2([\mathbf{a}, \mathbf{b}]) \,:\, \sum_{|\boldsymbol{\alpha}| \leq k} \|D^{\boldsymbol{\alpha}} f\|_{L^2}^2 < \infty \right\},$$

where $D^{\boldsymbol{\alpha}} f$ is the $\boldsymbol{\alpha}$-th weak derivative of $f$ and $k \geq 0$. The subscript $|\boldsymbol{\alpha}| \leq k$ means that the sum is over all vectors $\alpha$ with length less than or equal to $k$. The $k$ then indicates the order of weak derivatives that are required to exist. Thus, this is a way to formalize the regularity properties of the functions in the space. Note that $\mathcal{H}^0 = L^2$, which validates their use for

analysis. These spaces are equipped with a norm $\|\cdot\|_{\mathcal{H}^k}$ and a seminorm $|\cdot|_{\mathcal{H}^k}$ defined as

$$\|f\|_{\mathcal{H}^k} = \left( \sum_{|\boldsymbol{\alpha}| \leq k} \|D^{\boldsymbol{\alpha}} f\|_{L^2}^2 \right)^{1/2} \qquad |f|_{\mathcal{H}^k} = \left( \sum_{|\boldsymbol{\alpha}| = k} \|D^{\boldsymbol{\alpha}} f\|_{L^2}^2 \right)^{1/2}. \tag{5.1}$$

Both are useful for bounding errors.

## 5.2 Analysis of COS-TT

A short overview of the approximations made during the COS-TT derivation is:

$$\begin{aligned}
f(\mathbf{x}) &\approx P_{\mathbf{K}} f(\mathbf{x}) = \sum_{\mathbf{k}=0}^{\mathbf{K}} \mathcal{C}_{\mathbf{k}} \Phi_{\mathbf{k}}(\mathbf{x}) && \text{(Series truncation)} \\
&\approx Q_{\mathbf{K}} f(\mathbf{x}) = \sum_{\mathbf{k}=0}^{\mathbf{K}} \mathcal{F}_{\mathbf{k}} \Phi_{\mathbf{k}}(\mathbf{x}) && \text{(Coefficient approximation)} \\
&\approx f^{\star}_{\mathbf{r}\text{-CTT}}(\mathbf{x}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \Gamma_j(\alpha_{j-1}; x_j; \alpha_j) && \text{(rank-}\mathbf{r}\text{ truncation)} \\
&\approx \hat{f}_{\mathbf{r}\text{-CTT}}(\mathbf{x}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \hat{\Gamma}_j(\alpha_{j-1}; x_j; \alpha_j) && \text{(Algorithmic approximation)}
\end{aligned}$$

This in turn leads to the following separation of the error:

$$\begin{aligned}
\|f - \hat{f}_{\mathbf{r}\text{-CTT}}\|_{L^2} \leq\ & \underbrace{\|f - P_{\mathbf{K}} f\|_{L^2}}_{(\mathbf{I})\text{ Series truncation}} + \underbrace{\|P_{\mathbf{K}} f - Q_{\mathbf{K}} f\|_{L^2}}_{(\mathbf{II})\text{ Coefficient approximation}} \\
& + \underbrace{\|Q_{\mathbf{K}} f - f^{\star}_{\mathbf{r}\text{-CTT}}\|_{L^2}}_{(\mathbf{III})\text{ rank-}\mathbf{r}\text{ truncation}} + \underbrace{\|f^{\star}_{\mathbf{r}\text{-CTT}} - \hat{f}_{\mathbf{r}\text{-CTT}}\|_{L^2}}_{(\mathbf{IV})\text{ Algorithmic approximation}}
\end{aligned} \tag{5.2}$$

Each of these error terms can be bounded separately, by the error bounds derived in the respective subsections.

**Theorem 4** (COS-TT Error Bound)**.** The total approximation error of the COS-TT method can be bounded by

$$\|f - \hat{f}_{\mathbf{r}\text{-CTT}}\|_{L^2} \leq C_1 K^{-n} |f|_{\mathcal{H}^n} + C_2 \|f\|_{L^1(\mathbb{R}^d \setminus [\mathbf{a},\mathbf{b}])} + \|f\|_{\mathcal{H}^n} \sqrt{d-1} \frac{(r+1)^{-(k-1)/2}}{\sqrt{k-1}} + \varepsilon_{\text{TT}}$$

where $C_1$ is a constant depending on $n$ and $d$, respectively, and $\varepsilon_{\text{TT}}$ is the error incurred by the DMRG-greedy approximation.

### 5.2.1 Series truncation

The series truncation error is the error incurred by truncating the Fourier cosine series expansion of $f$ to a finite number of terms $\mathbf{K}$. In the one-dimensional case, the derivations in section 2.3 show that the resulting error can be expressed in terms of the neglected coefficients. These coefficients are then shown to decay exponentially for sufficiently smooth functions, and a formal bound is derived (proposition 4).

The multidimensional extension is natural, and the conclusion is similar. Instead of repeating the full derivations here, the result from [6] is used directly, which leads to the following bound on the error.

**Proposition 10** (Series Truncation Bound [6], Prop. 3). Let $f \in \mathcal{H}^n([\mathbf{a}, \mathbf{b}])$ such that it has at least $n$ continuous weak derivatives. Then, the error incurred by truncating the Fourier cosine series expansion of $f$ to $\mathbf{K} = (K, \dots, K)$ terms in each dimension is bounded by

$$\|f - P_{\mathbf{K}} f\|_{L^2} \leq C_1 K^{-n} |f|_{\mathcal{H}^n}$$

where $C_1$ is a constant depending on $n$.

For simplicity, this is restricted to the case where the series truncation is applied uniformly across all dimensions, as this is the most common scenario encountered in practice.

### 5.2.2 Coefficient approximation

The second error source to consider is the error due to approximating the Fourier coefficients $\mathcal{C}_{\mathbf{k}}$ by $\mathcal{F}_{\mathbf{k}}$. This error has been studied in the one-dimensional case in section 2.3, and the results can be extended to multiple dimensions. Because the cosine basis has been defined differently, the result will be slightly different. Again, define

$$\varepsilon_4 = \|f\|_{L^1(\mathbb{R}^d \setminus [\mathbf{a}, \mathbf{b}])} \tag{5.3}$$

**Proposition 11.** The error incurred by approximating the Fourier coefficients $\mathcal{C}_{\mathbf{k}}$ by $\mathcal{F}_{\mathbf{k}}$ is bounded by

$$\|P_{\mathbf{K}} f - Q_{\mathbf{K}} f\|_{L^2} \leq \varepsilon_4 \left( \sum_{\mathbf{k}}^{\mathbf{K}} \prod_{j=1}^{d} (\beta_{k_j}^{(j)})^2 \right)^{1/2} = C_2 \|f\|_{L^1(\mathbb{R}^d \setminus [\mathbf{a}, \mathbf{b}])}$$

*Proof.* Similar to the one-dimensional case, we can express the error as

$$\begin{aligned} \varepsilon_3 &= P_{\mathbf{K}} f(\mathbf{x}) - Q_{\mathbf{K}} f(\mathbf{x}) \\ &= \sum_{\mathbf{k}}^{\mathbf{K}} (\mathcal{C}_{\mathbf{k}} - \mathcal{F}_{\mathbf{k}}) \Phi_{\mathbf{k}}(\mathbf{x}). \end{aligned}$$

Using orthonormality of the basis functions,

$$\begin{aligned} \|\varepsilon_3\|_{L^2}^2 &= \int_{[\mathbf{a}, \mathbf{b}]} \left( \sum_{\mathbf{k}}^{\mathbf{K}} (\mathcal{C}_{\mathbf{k}} - \mathcal{F}_{\mathbf{k}}) \Phi_{\mathbf{k}}(\mathbf{x}) \right)^2 d\mathbf{x} \\ &= \sum_{\mathbf{k}}^{\mathbf{K}} |\mathcal{C}_{\mathbf{k}} - \mathcal{F}_{\mathbf{k}}|^2 \end{aligned}$$

Then, using the definition of equation (2.17), the difference between the coefficients can be expressed as

$$\mathcal{C}_{\mathbf{k}} - \mathcal{F}_{\mathbf{k}} = \int_{\mathbb{R} \setminus [\mathbf{a}, \mathbf{b}]} f(\mathbf{x}) \frac{\prod_{j=1}^{d} \beta_{k_j}^{(j)}}{2^{d-1}} \sum_{\mathbf{s} \in \mathcal{S}} \cos \left( \sum_{j=1}^{d} \frac{s_j k_j \pi}{b_j - a_j} (x_j - a_j) \right) d\mathbf{x}$$

This looks cumbersome to work with, but the cosine basis can be restored using equation (2.16), which gives

$$\mathcal{C}_{\mathbf{k}} - \mathcal{F}_{\mathbf{k}} = \int_{\mathbb{R} \setminus [\mathbf{a}, \mathbf{b}]} f(\mathbf{x}) \Phi_{\mathbf{k}}(\mathbf{x}) d\mathbf{x}$$

In the 1D case, it was used that $|\cos(\cdot)| \leq 1$. In the multidimensional case, it holds that

$$|\Phi_{\mathbf{k}}(\mathbf{x})| \leq \prod_{j=1}^{d} \beta_{k_j}^{(j)}. \tag{5.4}$$

Because $f$ is assumed to be non-negative, this leads to

$$\begin{aligned}
|\mathcal{C}_{\mathbf{k}} - \mathcal{F}_{\mathbf{k}}| &\leq \int_{\mathbb{R}\setminus[\mathbf{a},\mathbf{b}]} |f(\mathbf{x})||\Phi_{\mathbf{k}}(\mathbf{x})| d\mathbf{x} \\
&\leq \int_{\mathbb{R}\setminus[\mathbf{a},\mathbf{b}]} |f(\mathbf{x})||\prod_{j=1}^{d} \beta_{k_j}^{(j)}| d\mathbf{x} \\
&= \left(\prod_{j=1}^{d} \beta_{k_j}^{(j)}\right) \int_{\mathbb{R}^d\setminus[\mathbf{a},\mathbf{b}]} f(\mathbf{x}) d\mathbf{x} \\
&= \left(\prod_{j=1}^{d} \beta_{k_j}^{(j)}\right) \varepsilon_4
\end{aligned}$$

Therefore,

$$\|\varepsilon_3\|_{L^2}^2 \leq \sum_{\mathbf{k}}^{\mathbf{K}} \left(\prod_{j=1}^{d} \beta_{k_j}^{(j)}\right)^2 \varepsilon_4^2 = \varepsilon_4^2 \sum_{\mathbf{k}}^{\mathbf{K}} \prod_{j=1}^{d} (\beta_{k_j}^{(j)})^2$$

and the result follows by taking the square root.

□

### 5.2.3 Rank truncation

The second error source to consider is the error due to the rank-$\mathbf{r}$ truncation. In the ideal scenario, the theory developed in [6], which is partially addressed in the chapter 3, is applied. To do that, it needs to be shown that the spectral tensor train decomposition constructed in [6, Sec. 4.5.1] yields the same results as when the coefficient tensor is decomposed directly, thereby justifying the use of their analysis. This does not depend on the approximation of $\mathcal{C} \approx \mathcal{F}$. However, the derivations in [6, Sec. 4.5.1] start at the rank-$\mathbf{r}$ FTT approximation of $f$. Instead, this starts at the untruncated FTT of $f$,

$$f_{\text{TT}}(\mathbf{x}) = \sum_{\boldsymbol{\alpha}}^{\infty} \prod_{j=1}^{d} \gamma_j(\alpha_{j-1}; x_j; \alpha_j).$$

Applying the operator $P_{\mathbf{K}}$ to this FTT yields

$$P_{\mathbf{K}} f_{\text{TT}} = \sum_{\mathbf{k}=0}^{\mathbf{K}} \mathcal{C}_{\mathbf{k}} \Phi_{\mathbf{k}}(\mathbf{x}),$$

where the coefficients are given by

$$\mathcal{C}_{\mathbf{k}} = \langle f_{\text{TT}}(\mathbf{x}), \Phi_{\mathbf{k}}(\mathbf{x}) \rangle = \int_{[\mathbf{a},\mathbf{b}]} \left( \sum_{\boldsymbol{\alpha}}^{\infty} \prod_{j=1}^{d} \gamma_j(\alpha_{j-1}; x_j; \alpha_j) \right) \Phi_{\mathbf{k}}(\mathbf{x}) d\mathbf{x}.$$

**Theorem 5** (Spectral Tensor Train). Let $f_{\text{TT}}(\mathbf{x})$ be the untruncated tensor train representation of $f$, as defined in equation (3.2). Then,

$$\langle f_{\text{TT}}(\mathbf{x}), \Phi_{\mathbf{k}}(\mathbf{x})\rangle = \sum_{\boldsymbol{\alpha}}^{\infty} \prod_{j=1}^{d} \langle \gamma_j(\alpha_{j-1}; x_j; \alpha_j), \phi_{k_j}^{(j)}(x_j)\rangle, \tag{5.5}$$

or in integral form,

$$\int_{[\mathbf{a},\mathbf{b}]} \left( \sum_{\boldsymbol{\alpha}}^{\infty} \prod_{j=1}^{d} \gamma_j(\alpha_{j-1}; x_j; \alpha_j) \right) \Phi_{\mathbf{k}}(\mathbf{x}) d\mathbf{x} = \sum_{\boldsymbol{\alpha}}^{\infty} \prod_{j=1}^{d} \int_{a_j}^{b_j} \gamma_j(\alpha_{j-1}; x_j; \alpha_j) \phi_{k_j}^{(j)}(x_j) dx_j. \tag{5.6}$$

*Proof.* The reasoning for this is based on partial sums, and the convergence properties of the FTT established by [6]. To be precise, some notation is defined before proceeding.

Let the partial sums $S_{\mathbf{r}} \in L^2([\mathbf{a}, \mathbf{b}])$ be defined as

$$S_{\mathbf{r}}(\mathbf{x}) := \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} f_{\boldsymbol{\alpha}}(\mathbf{x}), \quad \text{where } f_{\boldsymbol{\alpha}}(\mathbf{x}) := \prod_{j=1}^{d} \gamma_j(\alpha_{j-1}; x_j; \alpha_j),$$

that is, the rank-$\mathbf{r}$ truncated FTT of $f$. Remember that the multi-index $\mathbf{r} = (r_1, \dots, r_{d-1})$ defines the truncation in each dimension, and the notation $\sum_{\boldsymbol{\alpha}}^{\mathbf{r}}$ is shorthand for $d$ summations from $\alpha_i = 1$ to $\alpha_i = r_i$, $i = 1, \dots, d-1$. Additionally, $\alpha_0 = \alpha_d = 1$ by convention. The partial sums are defined in order to take a limit. See [6, eq. 54] for a similar argument. These limits will be taken by setting $\mathbf{r} = (r, \dots, r)$, and letting $r \to \infty$ (i.e. uniformly over all indices). Furthermore, it has been established in [6, Thm. 13] that $S_{\mathbf{r}} \to f$ in $L^2$ as $r \to \infty$.

Lastly, the basis functions $\Phi_{\mathbf{k}}(\mathbf{x}) \in L^2([\mathbf{a}, \mathbf{b}])$ are defined as before (see equation (2.9)).

With this setup, the following argument can be made,

$$|\langle f, \Phi_{\mathbf{k}}\rangle - \langle S_{\mathbf{r}}, \Phi_{\mathbf{k}}\rangle| = |\langle f - S_{\mathbf{r}}, \Phi_{\mathbf{k}}\rangle| \leq \|f - S_{\mathbf{r}}\|_{L^2} \|\Phi_{\mathbf{k}}\|_{L^2}$$

by the Cauchy Schwarz inequality in $L^2$. This implies that

$$\langle S_{\mathbf{r}}, \Phi_{\mathbf{k}}\rangle \to \langle f, \Phi_{\mathbf{k}}\rangle \quad \text{as } r \to \infty,$$

or in other words, $\lim_{r\to\infty} \langle S_{\mathbf{r}}, \Phi_{\mathbf{k}}\rangle = \langle f, \Phi_{\mathbf{k}}\rangle$. The inner product in the limit now consists of finite sums, and can therefore be expanded as

$$\langle S_{\mathbf{r}}, \Phi\rangle = \langle \sum_{\boldsymbol{\alpha} \in I_{\mathbf{r}}} f_{\boldsymbol{\alpha}}, \Phi\rangle = \sum_{\boldsymbol{\alpha} \in I_{\mathbf{r}}} \langle f_{\boldsymbol{\alpha}}, \Phi\rangle,$$

meaning that

$$\langle f, \Phi_{\mathbf{k}}\rangle = \lim_{r\to\infty} \langle S_{\mathbf{r}}, \Phi_{\mathbf{k}}\rangle = \sum_{\boldsymbol{\alpha}}^{\infty} \langle f_{\boldsymbol{\alpha}}, \Phi_{\mathbf{k}}\rangle$$

The result follows by expanding the inner products into their univariate functions,

$$\langle f_{\boldsymbol{\alpha}}, \Phi_{\mathbf{k}}\rangle = \prod_{j=1}^{d} \langle \gamma_j(\alpha_{j-1}; \cdot; \alpha_j), \phi_{k_j}^{(j)}\rangle. \tag{5.7}$$

□

Theorem 5 shows that the coefficients admit a tensor train decomposition. Specifically,

$$\mathcal{C}_{\mathbf{k}} = \sum_{\boldsymbol{\alpha}}^{\infty} \prod_{j=1}^{d} \beta_j(\alpha_{j-1}, k_j, \alpha_j) \qquad \beta_j(\alpha_{j-1}, k_j, \alpha_j) = \langle \gamma_j(\alpha_{j-1}; \cdot; \alpha_j), \phi_{k_j}^{(j)} \rangle \tag{5.8}$$

where each coefficient core $\beta_j$ is the projection of $\gamma_j$ onto the univariate basis functions.

This proves the duality between the FTT of $f$ and the TT decomposition of the coefficient tensor $\mathcal{C}$. That is, from the FTT of $f$, the coefficients admit a TT decomposition, which is given by the projection of the functional cores onto the basis functions, equation (5.8). Similarly, from the TT decomposition of the coefficients, the FTT of $f$ can be constructed by spectral synthesis, with each core defined as in equation (4.8). This duality is the key to applying the results from [6] to the COS-TT method. Truncating at rank $\mathbf{r}$ then yields the rank-$\mathbf{r}$ COS-TT decomposition of $f$,

$$\sum_{\mathbf{k}}^{\mathbf{K}} \left( \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \beta_j(\alpha_{j-1}, k_j, \alpha_j) \right) \Phi_{\mathbf{k}}(\mathbf{x}) = \sum_{\boldsymbol{\alpha}}^{\mathbf{r}} \prod_{j=1}^{d} \left( \sum_{k_j=0}^{K_j} \beta_j(\alpha_{j-1}, k_j, \alpha_j) \phi_{k_j}^{(j)}(x_j) \right) = f_{\mathbf{r}\text{-CTT}}(\mathbf{x}).$$

**Proposition 12** (Rank-r Truncation Bound, [6], Thm. 13). Let $f \in \mathcal{H}^n([\mathbf{a}, \mathbf{b}])$ such that it has at least $n$ continuous derivatives (in the weak sense) on $[\mathbf{a}, \mathbf{b}]$. Then, the error incurred by truncating the COS-TT decomposition of $f$ to rank $\mathbf{r} = (r, \dots, r)$ is bounded by

$$\|f - f^{\star}_{\mathbf{r}\text{-CTT}}\|_{L^2}^2 \leq \|f\|_{\mathcal{H}^n}^2 \, (d-1) \frac{(r+1)^{-n-1}}{n-1}.$$

### 5.2.4 DMRG-Greedy Approximation

To use any theoretical bounds derived on the algorithmic error, the function approximation error needs to be related to the coefficient tensor approximation error. Luckily, there is an equivalence relation, as summarized in the following theorem.

**Theorem 6.** The error between the best rank-$\mathbf{r}$ COS-TT approximation, and the approximation obtained by the DMRG-Greedy algorithm is exactly the error in the coefficient tensor approximation. That is,

$$\|f^{\star}_{\mathbf{r}\text{-CTT}} - \hat{f}_{\mathbf{r}\text{-CTT}}\|_{L^2} = \|\mathcal{C}^{\star} - \hat{\mathcal{C}}\|_F,$$

where $\mathcal{C}^{\star}$ is the best rank-$\mathbf{r}$ approximation of the coefficient tensor $\mathcal{C}$, and $\hat{\mathcal{C}}$ is the coefficient tensor obtained from the DMRG-Greedy algorithm.

*Proof.* Remember that $f^{\star}_{\mathbf{r}\text{-CTT}}$ and $\hat{f}_{\mathbf{r}\text{-CTT}}$ can be expressed in terms of their coefficient tensors:

$$f^{\star}_{\mathbf{r}\text{-CTT}}(\mathbf{x}) = \sum_{\mathbf{k}}^{\mathbf{K}} \mathcal{C}^{\star}_{\mathbf{k}} \Phi_{\mathbf{k}}(\mathbf{x}), \qquad \hat{f}_{\mathbf{r}\text{-CTT}}(\mathbf{x}) = \sum_{\mathbf{k}}^{\mathbf{K}} \hat{\mathcal{C}}_{\mathbf{k}} \Phi_{\mathbf{k}}(\mathbf{x}).$$

Then

$$f^{\star}_{\mathbf{r}\text{-CTT}} - \hat{f}_{\mathbf{r}\text{-CTT}} = \sum_{\mathbf{k}}^{\mathbf{K}} (\mathcal{C}^{\star}_{\mathbf{k}} - \hat{\mathcal{C}}_{\mathbf{k}}) \Phi_{\mathbf{k}}.$$

Since the sum is finite, squaring, integrating and exchanging summation and integration gives

$$\begin{aligned}\|f^\star_{\text{r-CTT}} - \hat{f}_{\text{r-CTT}}\|^2_{L^2} &= \int_{[\mathbf{a},\mathbf{b}]} \left( \sum_{\mathbf{k}}^{\mathbf{K}} (\mathcal{C}^\star_{\mathbf{k}} - \hat{\mathcal{C}}_{\mathbf{k}}) \Phi_{\mathbf{k}}(\mathbf{x}) \right)^2 d\mathbf{x} \\ &= \sum_{\mathbf{k}}^{\mathbf{K}} \sum_{\mathbf{j}}^{\mathbf{J}} (\mathcal{C}^\star_{\mathbf{k}} - \hat{\mathcal{C}}_{\mathbf{k}})^2 \int_{[\mathbf{a},\mathbf{b}]} \Phi_{\mathbf{k}}(\mathbf{x}) \Phi_{\mathbf{j}}(\mathbf{x}) d\mathbf{x}\end{aligned}$$

By orthonormality of the basis functions, the integral is $1$ if $\mathbf{k} = \mathbf{j}$ and $0$ otherwise. Therefore,

$$\|f^\star_{\text{r-CTT}} - \hat{f}_{\text{r-CTT}}\|^2_{L^2} = \sum_{\mathbf{k}}^{\mathbf{K}} (\mathcal{C}^\star_{\mathbf{k}} - \hat{\mathcal{C}}_{\mathbf{k}})^2 = \|\mathcal{C}^\star - \hat{\mathcal{C}}\|^2_F.$$

□

Thus, the error incurred in the approximation of the best rank-$\mathbf{r}$ COS-TT approximation by using the DMRG-Greedy algorithm can be exactly quantified by the error in the coefficient tensor approximation. A nice consequence of this theorem is that

$$\|P_{\mathbf{K}} f - f^\star_{\text{r-CTT}}\|_{L^2} = \|\mathcal{C} - \mathcal{C}^\star\|_F, \tag{5.9}$$

which means that the rank truncation error for COS-TT can also be directly interpreted as the rank truncation error of the coefficient tensor.

Therefore, any error bounds derived for the DMRG-Greedy algorithm can be directly applied to the function approximation error. However, as discussed in section 3.3, the DMRG-Greedy algorithm only applies the maximum-volume principle heuristically, and therefore, proposition 8 does not directly apply.

If, instead of the DMRG-Greedy algorithm, the TT-SVD algorithm is used to compute the rank-$\mathbf{r}$ approximation of the coefficient tensor, the bound from proposition 6 can be applied. Using equation (5.9), this can be related to the function approximation error, and this yields

$$\begin{aligned}\|\mathcal{C} - \hat{\mathcal{C}}\|_F &\leq \sqrt{d-1} \|\mathcal{C} - \mathcal{C}^\star\|_F \\ &= \sqrt{d-1} \|P_{\mathbf{K}} f - f^\star_{\text{r-CTT}}\|_{L^2} \\ &\leq \|f\|_{\mathcal{H}^n} (d-1) \frac{(r+1)^{-(k-1)/2}}{\sqrt{k-1}}.\end{aligned} \tag{5.10}$$

As the DMRG-Greedy algorithm is used in practice, the error incurred by this algorithm is denoted by $\varepsilon_{\text{TT}}$ in the total error bound.

## 5.3 Other Methods

The other methods developed in chapter 4 make similar approximations, and therefore the error analysis can be adapted accordingly. The following sections briefly discuss how the error analysis applies to these methods.

### 5.3.1 FTT on the density

When the an FTT is applied directly on the density, a rank-$\mathbf{r}$ truncation is done, as well as an algorithmic approximation to compute the functional tensor cores. Additionally, to evaluate expectations, the inner product needs to be approximated numerically.

The total approximation error can be separated as

$$\|f - \hat{f}_{\mathbf{r}\text{-FTT}}\|_{L^2} \leq \underbrace{\|f - f^{\star}_{\mathbf{r}\text{-FTT}}\|_{L^2}}_{(\mathbf{I})\text{ rank-}\mathbf{r}\text{ truncation}} + \underbrace{\|f^{\star}_{\mathbf{r}\text{-FTT}} - \hat{f}_{\mathbf{r}\text{-FTT}}\|_{L^2}}_{(\mathbf{II})\text{ Algorithmic approximation}}$$

These errors have been bounded before in the context of the COS-TT method, and therefore the same bounds can be applied here. This results in the following error bound.

**Corollary 1.** The total approximation error of applying the FTT method on the density can be bounded by

$$\|f - \hat{f}_{\mathbf{r}\text{-FTT}}\|_{L^2} \leq \|f\|_{\mathcal{H}^n} \sqrt{d-1} \frac{(r+1)^{-(k-1)/2}}{\sqrt{k-1}} + \varepsilon_{\text{TT}},$$

where $\varepsilon_{\text{TT}}$ is the error incurred by the DMRG-greedy approximation.

# Ch. 6

# Application: pricing $n$-dimensional Basket options

In section 2.5 the COS method has been introduced as a method to price options on a single asset. It can also be applied to options on multiple assets [31, 43]. In the case of a basket option, the payoff function is defined in terms of the weighted sum of multiple assets. Consequently, calculating the expectation involves the joint distribution of the underlying assets, resulting in a multidimensional integration problem. This is why the methods developed in chapter 4 are well-suited to tackle this problem.

Formally, the asset market is expressed as a vector of $n$ stocks, $\mathbf{S}(t) = (S_1(t), S_2(t), \dots, S_n(t))$, each following a stochastic process under the risk-neutral measure $\mathbb{Q}$. A European-style basket option is considered, where the value of the basket is given by the weighted sum of the individual asset prices. It is often simpler to model the log prices of the assets, $\mathbf{X}(t)$, where each $X_j(t) = \log(S_j(t))$, under which the option has the payoffs

$$\Lambda^{(p)}(\mathbf{X}(T); K) = \left( K - \sum_{j=1}^{n} \theta_j e^{X_j(T)} \right)^+, \quad \Lambda^{(c)}(\mathbf{X}(T); K) = \left( \sum_{j=1}^{n} \theta_j e^{X_j(T)} - K \right)^+. \tag{6.1}$$

where $\theta_j$ are the weights of the assets, $K$ is the strike price, the superscripted + indicates the maximum of the argument and zero, and the superscripts $(p)$ and $(c)$ denote put and call options respectively. The price of the option at time $t = 0$ is then given by

$$V(0; K) = e^{-rT} \mathbb{E}^{\mathbb{Q}} \left[ \Lambda(\mathbf{X}(T); K) \right].$$

This price will be determined using the COS method described in section 2.5.

## 6.1 Model

To apply the COS method, the price of the basket at time $T$ is treated as a random variable $H$,

$$H_{\mathbf{X}(T)} = \sum_{j=1}^{n} \theta_j e^{X_j(T)}. \tag{6.2}$$

Note that $H$ is not in log-price, but in regular asset price, so the coefficients $V_k$ need to be derived differently (see chapter C). This random variable has a probability density function $f_H(x)$, which depends on the joint distribution of the log prices $\mathbf{X}(T)$. This distribution is not known, but can be approximated. Assuming that the density is negligible outside the interval $[a, b]$, the COS method can be applied to approximate the function, yielding

$$f_H(x) \approx \sum_{k=0}^{N}{}' F_k \cos\left( \frac{k\pi}{b-a}(x-a) \right),$$

which has coefficients

$$F_k = \frac{2}{b-a} \operatorname{Re}\left\{\varphi_H\left(\frac{k\pi}{b-a}\right) \cdot \exp\left(-i\frac{k\pi a}{b-a}\right)\right\},$$

defined in terms of the characteristic function $\varphi_H(\omega)$. This characteristic function is not known either, but a basic change of variables shows

$$\varphi_H(\omega) = \mathbb{E}\left[e^{i\omega H}\right] = \mathbb{E}\left[\exp\left(i\omega \sum_{j=1}^{n} \theta_j e^{X_j(T)}\right)\right] = \int_{\mathbb{R}^n} e^{i\omega H} f_{\mathbf{X}(T)}(\mathbf{x})\, d\mathbf{x}.$$

This change of variables is sometimes known as the *law of the unconscious statistician*.

> **Remark.** Here, the characteristic function is dependent on the maturity time $T$. If $\mathbf{X}(T)$ is expressed in terms of standard normal random variables, independent of $T$ can be obtained. This way, a single approximation can be used for all possible maturity times. That is,
>
> $$X_i(T) \sim \mu_i^{\mathbf{X}}(T) + \sigma_i^{\mathbf{X}} Z_i,$$
>
> with $Z_i = \frac{1}{\sqrt{T}} W_i(T) \sim \mathcal{N}(0,1)$. For simplicity, this is currently omitted.

After this transformation, the problem is expressed in terms of the joint distribution $f_{\mathbf{X}(T)}$. This means that the methods developed in chapter 4 can be applied. If the distribution is known, it can be approximated by an FTT and the integration can be performed efficiently. If the joint characteristic function $\varphi_{\mathbf{X}(T)}$ is known instead, the COS-TT method can be applied. For either to work, note that $e^{i\omega H}$ is separable, i.e.

$$e^{i\omega H} = \exp\left(i\omega \sum_{j=1}^{n} \theta_j e^{X_j(T)}\right) = \prod_{j=1}^{n} \exp\left(i\omega\theta_j e^{X_j(T)}\right).$$

### 6.1.1 Joint probability density and characteristic function

The pricing method developed here relies on the availability of the joint probability density function, or the joint characteristic function of the random vector $\mathbf{X}(T)$. For *affine processes*, the joint characteristic function is available in closed form up to solving a system of Riccati-type ordinary differential equations [20]. This includes many widely used models in mathematical finance, such as the Lévy processes and Heston model. The class of models for which the density function is available, is far more restrictive.

In this work the underlying assets are assumed to follow geometric Brownian motion, for which both the joint probability density function and the joint characteristic function can be derived, so the methods can be compared. The dynamics of the log stock prices under the risk-neutral measure are given by

$$dX_i(t) = \left(r - \frac{\sigma_i^2}{2}\right) dt + \sigma_i dW_i(t).$$

where $r$ is the risk-free interest rate, $\sigma_i$ is the volatility of the stock, and $W_i(t)$ is a Brownian motion. These Brownian motions $W_i(t)$ can be correlated, but it is assumed that their volatility structure is diagonal. Let $\boldsymbol{\rho}$ denote the correlation matrix of the Brownian motions, such that

$$\mathbb{E}[dW_i(t)\, dW_j(t)] = \rho_{ij}\, dt,$$

with $\rho_{ij}$ is the correlation coefficient between $W_i(t)$ and $W_j(t)$. Equivalently, this can be written as $\mathbf{W}(t) = \mathbf{L}\mathbf{B}(t)$, where $\mathbf{B}(t)$ is a vector of independent Brownian motions, and $\mathbf{L}$ is a Cholesky factor such that $\mathbf{L}\mathbf{L}^\top = \boldsymbol{\rho}$.

**Theorem 7.** $\mathbf{X}(t) = (X_1(t), X_2(t), \dots, X_n(t))$ is normally distributed at time $t$, with mean vector

$$\mathbb{E}[\mathbf{X}(t)] = \mathbf{X}_0 + \left(r - \frac{\boldsymbol{\sigma}^2}{2}\right) t,$$

and covariance matrix

$$\mathrm{Cov}(\mathbf{X}(t)) = \mathrm{diag}\,(\boldsymbol{\sigma})\,\boldsymbol{\rho}\,\mathrm{diag}\,(\boldsymbol{\sigma})t.$$

*Proof.* From the dynamics of $dX_i(t)$, it follows that the mean is given directly by the drift term, so

$$\mathbb{E}[\mathbf{X}(t)] = \mathbf{X}_0 + \left(r - \frac{\boldsymbol{\sigma}^2}{2}\right) t.$$

For the covariance note that standard $n$-dimensional Brownian motion $\mathbf{B}(t)$ is normally distributed with mean zero and covariance $\mathbf{I}t$. In other words, $\mathbf{B}(t)$ has independent components $B_i(t)$, each of which is normally distributed with mean zero and variance $t$. Since the correlation matrix $\boldsymbol{\rho}$ is symmetric and positive definite, there exists a Cholesky decomposition $\mathbf{L}\mathbf{L}^\top = \boldsymbol{\rho}$. Correlated Brownian motions $\mathbf{W}(t)$ can be defined as $\mathbf{W}(t) = \mathbf{L}\mathbf{B}(t)$. Then, as matrix multiplication is linear,

$$\mathbf{W}(t) = \mathbf{L}\mathbf{B}(t) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{L}(\mathbf{I}t)\mathbf{L}^\top\right) = \mathcal{N}(\mathbf{0}, \boldsymbol{\rho}t).$$

Consequently, $\mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{W}(t)$ has mean zero and covariance matrix $\mathrm{diag}(\boldsymbol{\sigma})\,\boldsymbol{\rho}\,\mathrm{diag}(\boldsymbol{\sigma})t$. □

Because $\mathbf{X}(t)$ is multivariate normal with mean $\boldsymbol{\mu}(t)$ and covariance matrix $\boldsymbol{\Sigma}(t)$, its characteristic function is given by

$$\varphi_{\mathbf{X}(t)}(\boldsymbol{\omega}) = \exp\left(i\boldsymbol{\omega}^\top\boldsymbol{\mu}(t) - \frac{1}{2}\boldsymbol{\omega}^\top\boldsymbol{\Sigma}(t)\boldsymbol{\omega}\right).$$

### 6.1.2 Domain truncation

Another assumption that must be satisfied is the truncation to a closed bounded domain. This has to be done twice. First, the density $f_H$ is approximated with a Fourier cosine expansion, which relies on the assumption that $f_H \in L^2([a,b])$. Secondly, the characteristic function $\varphi_H(\omega)$ is reconstructed by truncating the joint density $f_{\mathbf{X}}(\mathbf{x})$ to a bounded domain $[\mathbf{a},\mathbf{b}]$. The choice of these truncation ranges is crucial for the accuracy of the method. A common approach is to use the cumulants of the distribution to determine the truncation range [2].

The cumulant generating function is defined as the logarithm of the characteristic function,

$$K(t) = \log(\varphi(t)) = \sum_{n=1}^{\infty} \kappa_n \frac{(it)^n}{n!} \tag{6.3}$$

The $n$-th cumulant $\kappa_n$ is then found by taking the $n$-th derivative of the cumulant generating function and evaluating at $\omega = 0$. For the multivariate normal distribution $f_{\mathbf{X}}$, the cumulants are known in closed form, and given by

$$\kappa_1 = \mathbb{E}[\mathbf{X}], \qquad \kappa_2 = \mathrm{Cov}(\mathbf{X}), \qquad \kappa_n = 0 \ \text{ for } n > 2. \tag{6.4}$$

The truncation range is then defined as $[\mathbf{a},\mathbf{b}]$ with

$$[a_j, b_j] = \left[\kappa_1^{(j)} - L_1\sqrt{\kappa_2^{(j)}},\ \kappa_1^{(j)} + L_1\sqrt{\kappa_2^{(j)}}\right], \tag{6.5}$$

where $L_1$ is a parameter that controls the size of the truncation range.

For the density of $H$, however, the cumulants are not known in closed form. As an approximation of $\varphi_H(\omega)$ is obtained, the derivatives of the cumulant generating function can be approximated by finite differences.

$$\begin{aligned}
K^{(n)}(0) &= \frac{d^n}{dt^n}\log(\varphi_H(t))\Big|_{t=0} \\
&= \frac{d^n}{dt^n}\sum_{m=1}^{\infty}\kappa_m\frac{(it)^m}{m!}\Big|_{t=0} \\
&= \sum_{m=1}^{\infty} i^m\kappa_m\frac{d^n}{dt^n}\frac{t^m}{m!}\Big|_{t=0} \\
&= 0+\cdots+0+i^n\kappa_n\frac{d^n}{dt^n}\frac{t^n}{n!}\Big|_{t=0}+\sum_{m=n+1}^{\infty} i^m\kappa_m\frac{d^n}{dt^n}\frac{t^m}{m!}\Big|_{t=0} \\
&= i^n\kappa_n\frac{n!}{n!}\Big|_{t=0}+0 \\
&= i^n\kappa_n
\end{aligned}$$

After working out the imaginary factors, the cumulants are given by

$$\kappa_1 = -iK^{(1)}(0) \qquad \kappa_2 = -K^{(2)}(0) \qquad \kappa_4 = K^{(4)}(0)$$

The truncation range is then defined similarly,

$$[a,b] = \left[\kappa_1 - L_2\sqrt{\kappa_2+\sqrt{\kappa_4}},\ \kappa_1 + L_2\sqrt{\kappa_2+\sqrt{\kappa_4}}\right] \tag{6.6}$$

where $L_2$ controls the size of the truncation range.

## 6.2 Application

Now that the setup is complete, the solution methods can be applied. The approximation of the characteristic function $\varphi_H(\omega)$ will be derived for each method.

### 6.2.1 Method 1: FTT on $f_{\mathbf{X}}$

Assuming that the probability density function is known, it can be approximated by a rank-$\mathbf{r}$ FTT, which allows for efficient approximation of the characteristic function. The characteristic function can then be computed as equation (4.6), i.e.

$$\varphi_H(\omega) \approx \langle e^{i\omega H}, \hat{f}_{\mathbf{r}\text{-TT}}(\mathbf{x})\rangle \approx \sum_{\boldsymbol{\alpha}}^{\mathbf{r}}\prod_{j=1}^{d}\sum_{i_j=1}^{N_j} w_{i_j}\exp(i\omega\theta_j e^{x^{(i_j)}})\,\gamma_j(\alpha_{j-1}, x^{(i_j)}, \alpha_j), \tag{6.7}$$

with a numerical method of choice to evaluate the integrals.

### 6.2.2 Method 2: COS-TT

In the case that the joint density is not known, the COS-TT method can be applied. The probability density is approximated with a multidimensional Fourier cosine series expansion, and DMRG-greedy is applied to the coefficient tensor. The resulting approximation of the characteristic function, as in equation (4.15), is then given by

$$\varphi_H(\omega) \approx \sum_{\boldsymbol{\alpha}}^{\mathbf{r}}\prod_{j=1}^{d}\sum_{k_j=0}^{K_j}\hat{B}_j(\alpha_{j-1}, k_j, \alpha_j)\int_{a_j}^{b_j}\exp(i\omega\theta_j e^x)\phi_{k_j}^{(j)}(x)\,dx \tag{6.8}$$

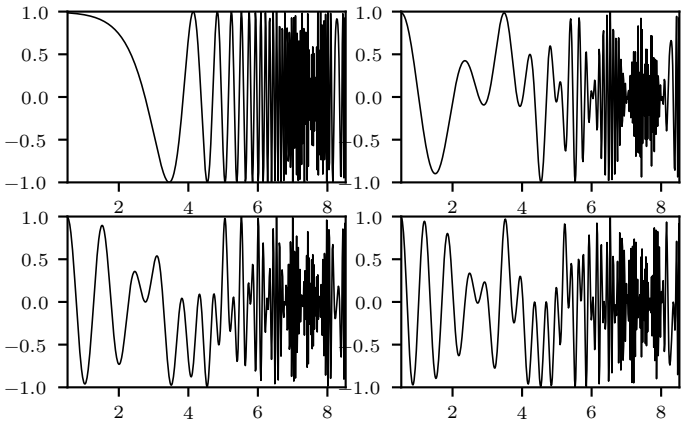

**Figure 6.1:** Four plots with 500 samples of the integrand $I_j(0.3, k_j)$ for $k_j = 0, 8, 16, 24$. The integrand is highly oscillatory, making numerical quadrature challenging. A discretization error is already observed for this setup.

This approximation requires computing the integrals

$$I_j(k_j, \omega) = \int_{a_j}^{b_j} \exp(i\omega\theta_j e^x) \cos\left(\frac{k_j\pi}{b_j - a_j}(x - a_j)\right) dx,$$

which depend on $\omega$ but are independent of the cores $\hat{B}_j$. Unfortunately, the integrand is highly oscillatory, as can be seen in figure 6.1. However, we found that they admit an analytic solution in terms of the lower incomplete gamma function,

$$\gamma(s, x) = \int_0^x t^{s-1} e^{-t}\, dt,$$

and this is presented as the following theorem, which is a side contribution to the existing literature of this thesis.

**Theorem 8.** The integrals $I_j(k_j, \omega)$ can be solved in terms of the lower incomplete gamma function $\gamma(s, x)$ as follows. For $\omega = 0$,

$$I_j(k_j, 0) = \begin{cases} 0, & \text{if } k_j \neq 0, \\ b_j - a_j, & \text{if } k_j = 0. \end{cases}$$

When $\omega \neq 0$,

$$I_j(k_j, \omega) = \begin{cases} \frac{1}{2} I_j^{(+)}(k_j, \omega) + \frac{1}{2} I_j^{(-)}(k_j, \omega), & \text{if } k_j \neq 0, \\ \mathrm{Ei}\left(i\omega\theta_j e^{b_j}\right) - \mathrm{Ei}\left(i\omega\theta_j e^{a_j}\right), & \text{if } k_j = 0, \end{cases}$$

where $\mathrm{Ei}(x)$ is the exponential integral function. The functions $I_j^{(+)}$ and $I_j^{(-)}$ are defined as:

$$I_j^{\pm}(k_j, \omega) = e^{\mp i\eta_j a_j} (-i\omega\theta_j)^{\pm i\eta_j} \left(\gamma\left(\pm i\eta_j, -i\omega\theta_j e^{b_j}\right) - \gamma\left(\pm i\eta_j, -i\omega\theta_j e^{a_j}\right)\right).$$

where $\eta_j = k_j\pi/(b_j - a_j)$. The complex power $z^s$ is evaluated as $\exp(s \log(z))$, for which the principal branch of the complex logarithm is chosen.

*Proof.* For simplicity the subscripts are omitted, and the integral under consideration is

$$I = \int_a^b \exp\left(i\omega\theta e^x\right) \cos\left(\eta(x - a)\right)\, dx. \tag{6.9}$$

Before proceeding, it is important to treat the case where $\omega = 0$ separately. In that case,

$$I = \int_a^b \cos\left(\eta(x-a)\right)\, dx = \begin{cases} \frac{1}{\eta}\sin\left(\eta(b-a)\right) = \frac{1}{\eta}\sin\left(k\pi\right) = 0, & \text{if } \eta \neq 0, \\ \int_a^b dx = b-a, & \text{if } \eta = 0. \end{cases}$$

Now, for the case when $\omega \neq 0$, the cosine terms can be separated as follows:

$$\cos\left(\eta(x-a)\right) = \frac{1}{2}\left(\exp\left(i\eta(x-a)\right) + \exp\left(-i\eta(x-a)\right)\right). \tag{6.10}$$

This enables rewriting the integral into two parts, $I = \frac{1}{2}\left(I_+ + I_-\right)$, where

$$I = \frac{1}{2}\left(\int_a^b \exp\left(i\omega\theta e^x\right)\exp\left(i\eta(x-a)\right)\, dx + \int_a^b \exp\left(i\omega\theta e^x\right)\exp\left(-i\eta(x-a)\right)\, dx\right). \tag{6.11}$$

Now substitute $u = e^x$, which gives $dx = du/u$, and integral bounds $e^{a_j}$, $e^{b_j}$. Then

$$I_\pm = e^{\mp i\eta a}\int_{e^a}^{e^b} u^{\pm i\eta - 1} e^{i\omega\theta u}\, du. \tag{6.12}$$

This representation allows for the use of the lower incomplete gamma function to evaluate the integrals more easily. The substitution is

$$\int u^{s-1}e^{i\omega\theta u}\, du = (-i\omega\theta)^{-s}\gamma(s, -i\omega\theta u). \tag{6.13}$$

where the power is interpreted via the principal branch of the complex logarithm. Then, for $\eta \neq 0$:

$$I_\pm = e^{\mp i\eta a}(-i\omega\theta)^{\pm i\eta}\left(\gamma\left(\pm i\eta, -i\omega\theta e^b\right) - \gamma\left(\pm i\eta, -i\omega\theta e^a\right)\right), \tag{6.14}$$

and for $\eta = 0$:

$$I = \int_a^b \exp\left(i\omega\theta e^x\right)\, dx = \mathrm{Ei}\left(i\omega\theta e^x\right)\Big|_a^b, \tag{6.15}$$

where $\mathrm{Ei}(x)$ is the exponential integral function

$$\mathrm{Ei}(x) = -\int_{-x}^{\infty} \frac{e^{-t}}{t}\, dt. \tag{6.16}$$

□

While this is not a solution in terms of elementary functions, it does provide a more feasible way to compute the integrals numerically.

## 6.3 Numerical results

A number of numerical experiments were performed, with the aim of showing the convergence and performance of the COS-TT method, in the context of pricing basket options. For that reason, the experiments are run in a variety of dimensions, and for different truncation parameters $L_1$, $L_2$ and number of Fourier terms $K_j$ and $N$. The Fourier series truncation $\mathbf{K}$ is chosen uniformly in all dimensions, $K_j = K$, and the rank $\mathbf{r}$ is done similarly, $r_j = r$.

Because the options are priced under the assumption that the underlying assets follow geometric Brownian motion, the joint density $f_{\mathbf{X}}$ is known in closed form, as given in theorem 7. This allows for a direct analysis of the error in the reconstruction of the joint density

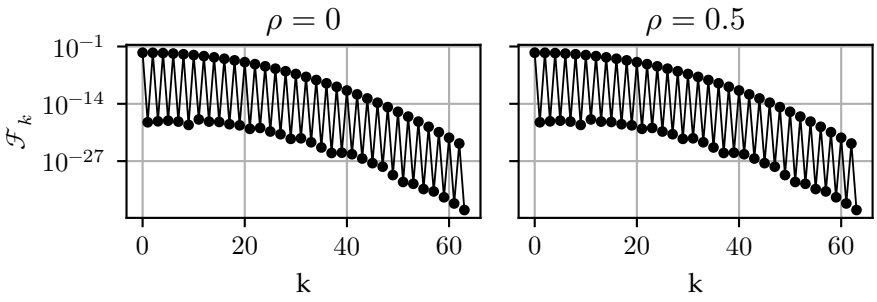


**Figure 6.2:** Decay of the cosine coefficients $\mathcal{F}[k, 0, 0, 0]$ for a 4-dimensional basket option computed using the COS method, with $K = 64$ terms and truncation range parameter $L_1 = 10$. The coefficients are plotted on a logarithmic scale, showing an supergeometric decay.

$f_{\mathbf{X}}$, by comparing the approximation with the exact density on a set of samples. Furthermore, as an approximation of the coefficients is available due to the COS method, the error in the approximation of the coefficient tensor $\mathcal{F}$ can be determined for low dimensions.

For the final calculation of the option price, the COS-TT method is directly compared with a Monte Carlo reference simulation.

Other tensor methods for option pricing include the COS-CPD method [14, 9] a cross approximation based method in [32] and a tensor completion method in [24]. The results of COS-TT are compared in terms of performance and computational time.

The common parameters of the option are as follows. The initial asset price is $S_0 = 100$, and the stocks in the basket are equally weighted, $\theta_j = 1/d$. The volatilities are $\sigma_j = 0.4$ and the stocks are equally correlated by $\rho_{ij} = \rho$ for $i \neq j$. Furthermore, the risk-free interest rate is $r = 0$, and the maturity time is $T = 1$ and the strike price is fixed at 100.

The code to run these experiments is publicly available on GitHub[1]. All experiments were run on a single node of DelftBlue [1] with 64 cores and 250 GB RAM, unless stated otherwise. The TT decompositions were computed using the DMRG-greedy algorithm [19], which is implemented in Fortran, and accelerated by OpenMP, and using Intel MKL as a BLAS backend.

The computation of the lower incomplete gamma and exponential integral functions can be done accurately using numerical libraries such Python's `mpmath`[2] or C++'s `Boost`[3]. A faster method for the lower incomplete gamma function is to use a Lanczos approximation [37], which was implemented specifically for this work. The performance of the exponential integral function from `mpmath.ei` was deemed satisfactory.

### 6.3.1 Convergence

The convergence of the COS-TT method starts with the convergence of the cosine coefficients. These are known to converge rapidly for smooth functions, with a supergeometric rate of convergence. The same is observed here, as shown in figure 6.2, where the decay of the cosine coefficients approximation $\mathcal{F}[k, 0, 0, 0]$ is plotted for a 4-dimensional basket option, i.e. a fiber of the coefficient tensor $\mathcal{F}$.

The error of approximating the coefficient tensor $\mathcal{F}$ with a low-rank tensor decomposition can be determined exactly in low dimensions, by reconstructing the full tensor from the TT-decomposition. For 56 Fourier terms in each dimension, this tensor has approximately $10^7$ entries, which is still feasible to store in memory. This error is due to both the low-rank ap-

[1] `https://github.com/aukeschaap/msc-thesis`
[2] https://mpmath.org
[3] https://www.boost.org

| Max Rank | Rank | Method | Time (s) | Error |
|---|---|---|---|---|
| | 24 | CPD | 19.4 | $4.4 \times 10^{-4}$ |
| 24 | 24 | TT-SVD | 2.37 | $6.4 \times 10^{-12}$ |
| | 24 | DMRG - Greedy | 0.12 | $2.6 \times 10^{-11}$ |
| | 40 | CPD | 27.8 | $2.0 \times 10^{-4}$ |
| 40 | 40 | TT-SVD | 4.96 | $1.3 \times 10^{-15}$ |
| | 33 | DMRG - Greedy | 0.25 | $4.2 \times 10^{-15}$ |
| | 56 | CPD | 36.8 | $1.6 \times 10^{-4}$ |
| 56 | 56 | TT-SVD | 7.71 | $1.2 \times 10^{-15}$ |
| | 33 | DMRG - Greedy | 0.22 | $5.0 \times 10^{-14}$ |

**Table 6.1:** Frobenius-norm error of the rank-$r$ decomposition of $\mathcal{F}$ for a basket of 4 stocks, with corr $= 0.5$, $L_1 = 8$, $K = 56$. The computations were performed on an 11th Gen Intel Core i5 with 8 cores @ 2.4 GHz.

proximation as well as the accuracy of the algorithm used to obtain the approximation.

Table 6.1 compares the Frobenius-norm error of the CPD, TT-SVD and DMRG-greedy methods for a basket of 4 stocks, for several ranks. Because the DMRG-greedy method is adaptive, there is a difference between the maximum allowed rank and the actual rank of the approximation. In the case of rank 40 and 56, the actual rank is lower than the maximum allowed rank, indicating that the tensor can be approximated accurately enough with a lower rank. This behaviour is also observed in the other experiments. Both TT-SVD and DMRG-greedy achieve machine precision accuracy, while CPD is limited to an error of around $10^{-4}$. Note that the rank of the CPD approximation can not be increased further, as it has been shown in [14] that overfitting occurs for $r > K$. Furthermore, DMRG-greedy is significantly faster than TT-SVD, without sacrificing accuracy.

Because the joint density $f_{\mathbf{X}}$ is known in closed form, the error in its approximation can be determined directly. This can be used to study the convergence of the COS-TT method for the approximation of $f_{\mathbf{X}}$. This is tested on a sample of $10^4$ uniformly distributed points for different truncation range parameters $L_1$ and number of Fourier terms $K$. The sampling interval is taken as the smallest truncation range, $L_1 = 4$, to compare the errors in the same points. This means that for approximations with smaller truncation ranges, the values are closer to the boundary than for larger truncation ranges, which is why for those values the error can not achieve machine precision. The results are shown in figure 6.3, where the mean absolute error is plotted for different dimensions.

As expected, the error decreases as $K$ increases, showing the supergeometric convergence of the COS method in this scenario. For larger truncation intervals, the initial error is higher, and more terms in the Fourier series are required to reach a certain accuracy. This is expected; doubling the truncation interval doubles the period of the cosine functions, i.e.

$$\cos\left(\frac{k\pi}{b-a}(x-a)\right) = \cos\left(\frac{2k\pi}{2(b-a)}(x-a)\right),$$

which means that double the number of terms is required to achieve the same accuracy. Thus, a balance must be found between a truncation range that is large enough to capture the density accurately, and small enough to allow for an accurate approximation with a limited number of Fourier terms.

Lastly, it must be noted that relatively more of the uniform samples are outliers as the dimension grows. This results in lower average values of the density, seen in table 6.2, and therefore a lower absolute error for the same approximation. Figure 6.4 instead shows the

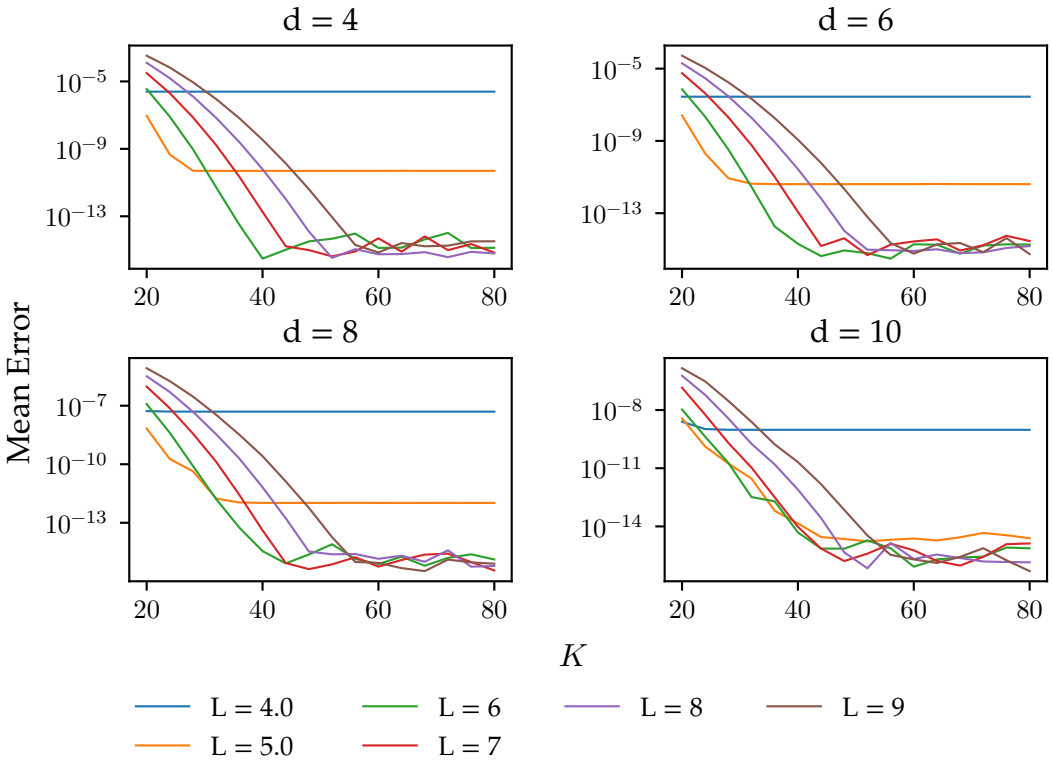


**Figure 6.3:** Error in the joint probability density function of a basket option in different dimensions, computed using the COS-TT method with DMRG-greedy, for different truncation range parameters $L_1$ and number of Fourier terms $K$. The error is computed as the mean *absolute* error at $10^4$ uniformly distributed samples within the smallest truncation range, $L_1 = 4$.

| $d$ | Mean value |
|---|---|
| 4 | $7.4 \times 10^{-3}$ |
| 6 | $1.7 \times 10^{-4}$ |
| 8 | $6.3 \times 10^{-6}$ |
| 10 | $5.2 \times 10^{-8}$ |

**Table 6.2:** Mean values of the gaussian density $f_{\mathbf{X}}$ of the test sample for different dimensions $d$.

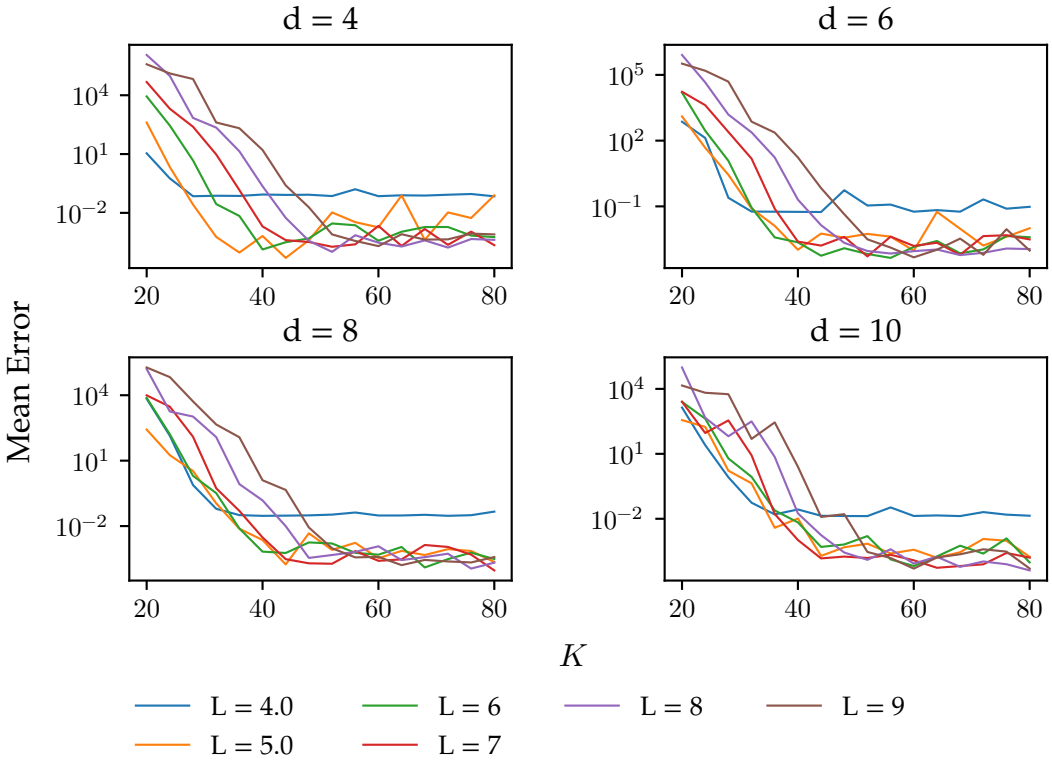


**Figure 6.4:** Error in the joint probability density function of a basket option in different dimensions, computed using the COS-TT method with DMRG-greedy, for different truncation range parameters $L_1$ and number of Fourier terms $K$. The error is computed as the mean *relative* error at $10^4$ uniformly distributed samples within the smallest truncation range, $L_1 = 4$.

relative error, which better accounts for this effect. However, as values are close to machine precision, the relative error can blow up. To prevent this, samples where the density is below $10^{-14}$ are assumed to have no error.

Convergence is observed here as well, although the relative error is higher than the absolute error. The same effect of the truncation range on the number of Fourier terms required to reach a certain accuracy is also observed.

Using these convergence results, parameters for further experiments are determined. To capture enough of the tail of the density, $L_1 = 8$ is chosen, which is expected to require $K \approx 56$ Fourier terms to reach machine precision. It must be noted that these parameters are dependent on the model under consideration, i.e. the distribution of $\mathbf{X}$, and should be determined for each model separately.

After reconstructing the characteristic function $\varphi_H(\omega)$, the cumulants of $f_H$ are approximated using finite differences, which are used to determine a truncation range for $H$. A step size of $h = 10^{-3}$ is used for the finite differences. With the characteristic function, $f_H$ is then reconstructed using the COS method. Figure 6.5 shows $f_H$ for a 15-dimensional basket option, with $L_2 = 10$ and several values of $N$, which follows the expected shape.

It is worthwile to investigate the tail behaviour of the reconstructed density $f_H$. Using a logarithmic scale, oscillations in the tail are observed for low values of $N$, i.e. the bottom plot of figure 6.6. These are attributed to the number of terms $N$ in the COS method approximation of $f_H$. That is, to construct an accurate approximation of $f_H$, on an interval $L_2$ of choice, a sufficient number of terms $N$ must of course be used. However, if $N$ is too large, the approximation

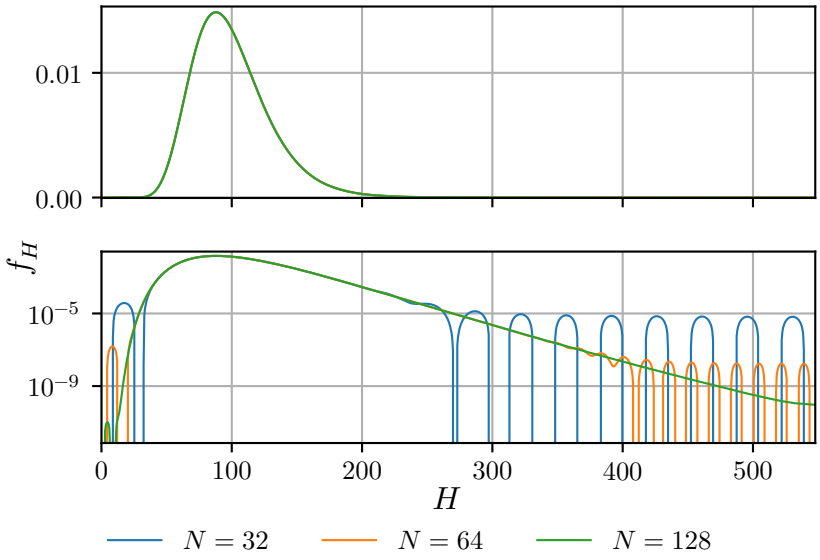


**Figure 6.5:** Reconstructed density $f_H$ for a 15-dimensional basket option, with $L_1 = 8$ and $K = 56$, for various values of $N$. The width of the truncation range for $H$ is set to $L_2 = 10$. The top plot shows the full density on a linear scale, while the bottom plot shows the plot density on a logarithmic scale, to highlight oscillations.

of coefficients $F_k$ is not accurate enough.

Similarly, a balance must be struck here, between the accuracy of $\varphi_H(\omega)$, which is controlled by $L_1$ and $K$, and the truncation range $L_2$ and number of terms $N$ used to reconstruct $f_H$. Higher values of $L_2$ with an appropriate $N$ will lead to a better approximation the option price.

The effect of $L_2$ on the tail is shown in figure 6.6. For low values of $L_2$, a smaller portion of the tail is captured, but a lower number of terms $N$ is required to reach a good accuracy. For higher values of $L_2$, more of the tail is captured, but oscillations appear in the tail for low values of $N$. Increasing $N$ removes these oscillations, but at a higher computational cost.

This behaviour is dependent on the accuracy of the approximation of $\varphi_H(\omega)$, which is controlled by $L_1$ and $K$. Figure 6.7 shows these effects. For low values of $L_1$, the approximation of $\varphi_H(\omega)$ is not accurate enough to reconstruct $f_H$ well, and oscillations appear in the tail, which are independent of $N$. For higher values of $L_1$, the approximation of $\varphi_H(\omega)$ is more accurate, and the oscillations in the tail can be controlled by $N$. This information can be used to choose appropriate parameters for the calculation of the option price, but it must be noted that these parameters are dependent on the model under consideration.

### 6.3.2 Performance

To assess the performance of the COS-TT method, several experiments were run to determine how the execution time scales with the parameters $L_1$, $K$, and the dimension $d$.

Table 6.3 shows the execution time and error of the COS-TT method for a for a varying number of dimensions $d$, with $L_1 = 8$ and $K = 56$. It can be seen that the training time, i.e. the time to compute the TT decomposition, grows exponentially with the dimension. The cause of this is the calculation of the coefficients via the COS method, i.e. equation (2.20), which requires computing the exponential sum

$$\mathcal{F}_{\mathbf{k}} = \frac{\prod_{j=1}^{d} \beta_{k_j}^{(j)}}{2^{d-1}} \sum_{\mathbf{s}\in\mathcal{S}} \mathrm{Re}\left\{ \exp\left( -i \sum_{j=1}^{d} \frac{s_j k_j \pi a_j}{b_j - a_j} \right) \varphi\left( \frac{s_1 k_1 \pi}{b_1 - a_1}, \dots, \frac{s_d k_d \pi}{b_d - a_d} \right) \right\},$$

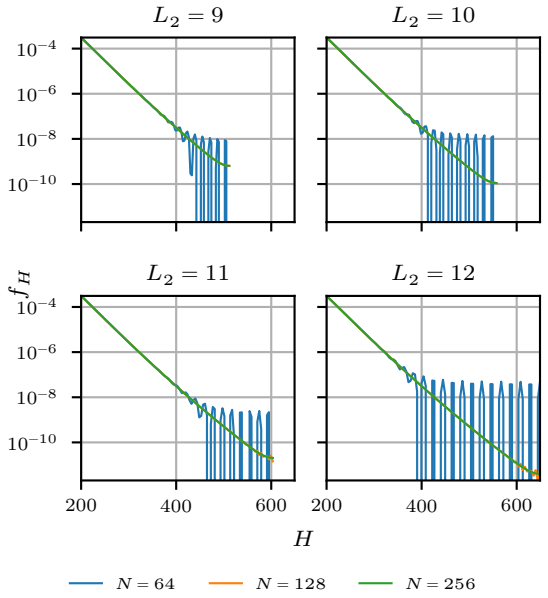


**Figure 6.6:** Tail behaviour of $f_H$ for different values of the truncation parameter $L_2$, for a 10-dimensional basket option, with $L_1 = 8$ and $K = 56$. The number of terms $N$ is varied to show its effect on the oscillations in the tail.

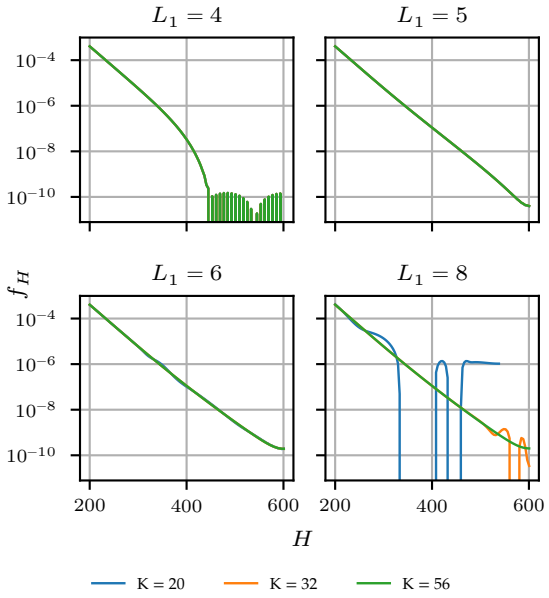


**Figure 6.7:** Tail behaviour of $f_H$ for different values of the truncation parameter $L_1$, for a 4-dimensional basket option, with $N = 144$ and $L_2 = 10$. The number of terms $K$ is varied to show its effect on the oscillations in the tail.

where $\mathcal{S}$ is the powerset of all sign combinations of length $d$, with the first element fixed to 1. That is,

$$\mathcal{S}_d = \{(1, s_2, s_3, \dots, s_d) : s_i \in \{-1, 1\} \text{ for } i = 2, \dots, d\},$$

and $|\mathcal{S}_d| = 2^{d-1}$.

While this can not be avoided, choosing a lower maximum rank requires computing less coefficients, which can help to reduce the execution time. Table 6.4 and figure 6.9 show the execution time for $d = 16$, with varying $L_1$ and $K$. The rank has been set to $K$, and therefore, the execution time grows as $K$ increased. Choosing a lower value of $L_1$ can help to reduce the number of coefficients required, and will therefore also reduce the execution time. However, this comes at the cost of accuracy, as shown in figure 6.3 and figure 6.4.

To push the method to its limits, the values of $L_1$ and $K$ are lowered. The result is shown in table 6.5, where the execution time and error in the option price is shown for dimensions 25 and 26, with $L_1 = 5$ and $K = 28$. The resulting approximation of the option price is still within $10^{-4}$ of the reference Monte Carlo simulation.

### 6.3.3 Comparisons

Beside the comparison with a Monte Carlo simulation, the COS-TT method can also be compared with other tensor methods for multi-asset option pricing, such as the COS-CPD method [14, 9], a cross approximation based method in [32] and a tensor completion method in [24]. All these methods have been tested on multi-asset options, but the parameters are different. Therefore, a comparison is given in terms of performance and computational time.

**COS-CPD** The COS-CPD method is very similar to the COS-TT method; the main difference lies in the replacement of the (functional) TT decomposition with a CPD decomposition. This requires different algorithms to compute the decomposition. The method is tested in [9] by pricing a basket option with 6 underlying assets. The training times were not reported.

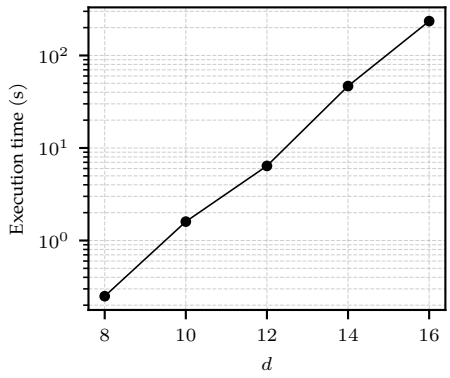


**Figure 6.8:** The training time of the underlying DMRG-greedy algorithm for different dimensions $d$, with $L_1 = 8$ and $K = 56$. The training time grows exponentially with the dimension.

| $d$ | $r$ | $n_{\text{evals}}$ | Time (s) |
|---|---|---|---|
| 8 | 39.5 | 634,390 | 0.25 |
| 10 | 41.7 | 924,261 | 1.6 |
| 12 | 46.0 | 1,364,666 | 6.4 |
| 14 | 52.5 | 2,125,586 | 46.7 |
| 16 | 54.3 | 2,580,027 | 235 |

**Table 6.3:** Training times and number of evaluations of the underlying DMRG-greedy algorithm for different dimensions $d$, with $L_1 = 8$ and $K = 56$. The reported rank $r$ is the average rank of the decomposition.

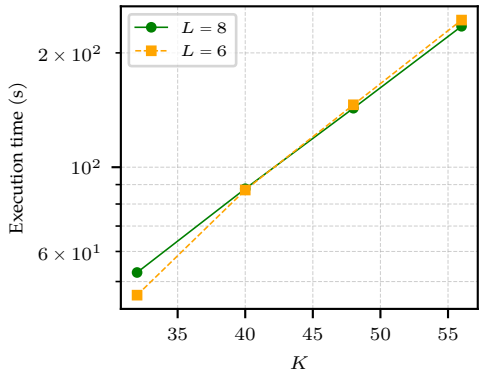


**Figure 6.9:** The training time of the underlying DMRG-greedy algorithm for $d = 16$, with varying $L_1$ and $K$.

| $L_1$ | $d$ | $K$ | $r$ | $n_{\text{evals}}$ | Time (s) |
|---|---|---|---|---|---|
| | 16 | 32 | 32.0 | 505,200 | 53 |
| | 16 | 40 | 39.9 | 968,374 | 88 |
| | 16 | 48 | 47.3 | 1,647,906 | 143 |
| 8 | 16 | 56 | 54.3 | 2,580,027 | 235 |
| | 16 | 32 | 31.9 | 505,035 | 46 |
| | 16 | 40 | 39.4 | 965,750 | 87 |
| | 16 | 48 | 46.7 | 1,640,066 | 146 |
| 6 | 16 | 56 | 53.4 | 2,561,046 | 244 |

**Table 6.4:** Training time and number of function evaluations of the underlying DMRG-greedy algorithm for $d = 16$, with varying $L_1$ and $K$. The reported rank $r$ is the average rank of the decomposition.

| $d$ | $N$ | Call Error | Put Error | Time (h) |
|---|---|---|---|---|
| 25 | 16 | $1.43 \times 10^{-2}$ | $1.56 \times 10^{-2}$ | 11.2 |
| | 24 | $2.71 \times 10^{-4}$ | $1.85 \times 10^{-3}$ | |
| | 32 | $9.12 \times 10^{-5}$ | $2.98 \times 10^{-4}$ | |
| | 40 | $1.36 \times 10^{-4}$ | $2.05 \times 10^{-4}$ | |
| 26 | 16 | $1.43 \times 10^{-2}$ | $1.53 \times 10^{-2}$ | 22.5 |
| | 24 | $2.70 \times 10^{-4}$ | $1.84 \times 10^{-3}$ | |
| | 32 | $8.47 \times 10^{-5}$ | $2.94 \times 10^{-4}$ | |
| | 40 | $1.29 \times 10^{-4}$ | $2.00 \times 10^{-4}$ | |

**Table 6.5:** Error in call and put option price for 25 and 26 dimensions, with $L_1 = 5$ and $K = 28$, for several values of $N$, compared to a reference Monte Carlo simulation of $10^9$ samples. The reported time is the training time of the DMRG-greedy algorithm.

The training was done with 12 training points per dimension, resulting in a total training set size of $12^6$ points. The training set size grows exponentially with the number of dimensions, which quickly becomes infeasible in higher dimensions. The COS-TT method does not suffer from this problem. While the rank grows with the dimension, and the number of function evaluations required grows as well, the growth is not exponential. This allows for accurate pricing up to 26 dimensions, as shown in table 6.5.

**Cross approximation** In [32], the authors price basket options under geometric Brownian motion, using a quantum-inspired algorithm. The option price is rewritten as an integral of the characteristic function and the Fourier transform of the payoff function. This integral can then be efficiently computed in tensor train format. The decompositions are computed using TT-Cross [39]. With this method the authors are able to price basket options up to 15 dimensions, after which the method becomes unstable. The accuracy can be controlled by a tolerance parameter, which is set to $5 \times 10^{-5}$ in the experiments. Furthermore, it is also reported that the training time grows only weakly with the dimension, and is, for example, $\approx 327$ seconds for 15 dimensions.

While the training time of the COS-TT method for 15 dimensions is slower, i.e. $\approx 100$ seconds, the training has been done in parallel, and not on a standard laptop. Furthermore, the training time of the COS-TT method grows exponentially with the dimension.

Nevertheless, the COS-TT method is able to price options of up to 26 dimensions, as shown in table 6.5, which is significantly higher than the maximum of 15 dimensions reported in [32]. Additionally, the accuracy of the COS-TT method can be controlled by the parameters $L_1$, $K$, $L_2$ and $N$, which lead to a provable higher accuracy if desired.

**Tensor completion** Lastly, the method in [24] uses Riemannian tensor completion to price basket options parametrically. The option price is treated as a multivariate function over a parameter space, and approximated with a Chebyshev polynomial tensor. Instead of pricing on the full grid, only a small subset is priced using a reference pricer. Using tensor completion, a low-rank tensor train decomposition of the full tensor is constructed. It is shown that this method can be used to price basket options up to at least 25 dimensions, with an accuracy of at least $10^{-4}$. The reported training time is around 22 minutes for 25 dimensions, but does not include the time required to price the samples using the reference pricer, which is not reported.

The COS-TT method is able to achieve a similar accuracy, as shown in table 6.5. However, the training time is significantly higher due to the exponential scaling in the calculation of the coefficients. The accuracy in the COS-TT method can be controlled by the method itself, while the accuracy of the method in [24] is dependent on the accuracy of the reference pricer.

Furthermore, the method in [24] is parametric, meaning that the option can be priced for different values of the parameters, such as the strike price $K$ or maturity time $T$. The COS-TT method is independent of the strike, and the training can also be made independent of the maturity time.

### 6.3.4 COS-TT-CHF

Unfortunately, constructing the decomposition of the joint characteristic function discretization for COS-TT-CHF was not successful. The reason for this lies within the oscillatory structure of the characteristic function $\varphi_{\mathbf{X}}(\boldsymbol{\omega})$. While the function is smooth, it requires a high amount of quadrature points to be accurately approximated, as shown in figure 6.10.

Currently, these nodes are sampled incorrectly, leading to an inaccurate approximation of the characteristic function. For three dimensions, TT-SVD was able to achieve a reproduction error of $10^{-2}$. To test a higher number of dimensions, the DMRG-greedy algorithm was rewrit-

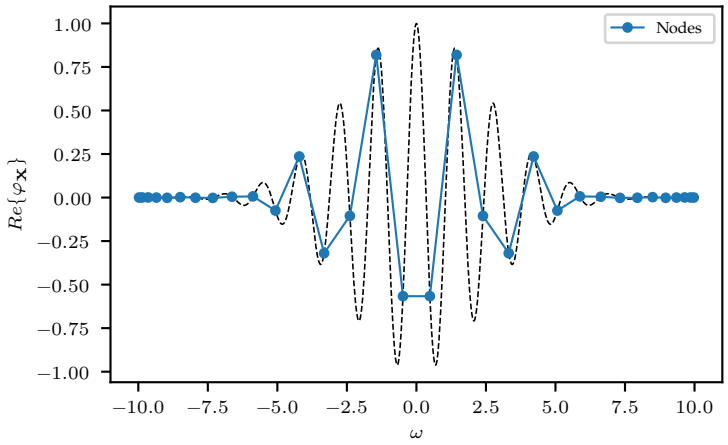


**Figure 6.10:** Oscillatory structure of the characteristic function $\text{Re}\{\varphi_{\mathbf{X}}(\boldsymbol{\omega})\}$ for a 3-dimensional basket option, with corr $= 0.5$. The function is plotted on the square $[-10, 10]^2$. There are 32 Gauss-Legendre nodes.

ten to support complex numbers. However, due to undersampling of the area of interest, the algorithm was not able to converge to a satisfactory solution.

In any case, it is likely that the rank required to capture the oscillations would be high, leading to long training times and high evaluation times.

# Ch. 7

# Conclusions and Future Work

This thesis proposed and analysed a new method, COS–TT, for computing multivariate expectations from their characteristic functions. It combines the COS method for Fourier-cosine series expansions with low-rank tensor decomposition of the coefficient tensor in the tensor-train format. This formulation was shown to be equivalent to the functional tensor train decomposition [6], and theoretical error bounds were derived on the approximation error. The method is accurate for pricing European basket options under geometric Brownian motion up to 26 assets, but is applicable to the calculation of any high-dimensional expectation where the characteristic function is known. The method was confirmed with numerical experiments, and was shown to outperform previous tensor methods based on the COS method [9] and cross approximation [32], and to achieve a similar result to a method based on tensor completion [24], without using Chebyshev interpolation.

## 7.1 Summary of Results

The method works as follows. First, the multivariate density function is approximated with a truncated Fourier-cosine series expansion, where the coefficients are computed directly from the characteristic function using the COS method. The coefficient tensor is then approximated in the tensor-train format using the DMRG-cross algorithm. Because this format has the property that the multidimensional inner product reduces to a sequence of univariate inner products, the expectation can be computed efficiently.

A central error bound was derived, describing the convergence of the method in the $L^2$ norm, and additional pointwise results were provided wherever possible. The error consists of four parts. First, an error due to truncating the Fourier-cosine series expansion, which decreases exponentially in the number of terms if the density is sufficiently smooth, and algebraically close to the points of singularity. The second error is due to the approximation of the coefficients by the COS method, which is shown to depend on the tail of the density. Further, an error is incurred by truncating the tensor-train decomposition to finite rank. Using the functional tensor train decomposition, this error was expressed in terms of the density. Finally, an algorithmic error is made by approximating the coefficient tensor with the DMRG-cross algorithm, which can not be bounded in general, but is observed to be small in practice.

Part of this theoretical analysis was the derivation of the duality between the functional tensor train decomposition and the tensor-train decomposition of the coefficient tensor. This duality has proven the equivalence between the two decompositions, and allowed the application of the COS-TT method to the spectral tensor train decomposition, and vice-versa. This duality was used to derive a bound on the error incurred by truncating the COS-TT decomposition to finite rank.

The method was applied to pricing European basket options under geometric Brownian motion. The basket option was priced with the COS method, which required expressing the characteristic function of the value of the basket as a function of the joint density. This density was reconstructed with the COS–TT method, and the characteristic function was reconstructed by solving the multidimensional expectation using the tensor-train inner product. This re-

quired computing the integrals

$$I_j(k_j, \omega) = \int_{a_j}^{b_j} \exp(i\omega\theta_j e^x) \cos\left(\frac{k_j\pi}{b_j - a_j}(x - a_j)\right) dx,$$

for which solutions were derived in terms of the lower incomplete gamma function.

To demonstrate the convergence and performance of the method, several numerical experiments were run. The approximation of the joint density was shown to converge exponentially up to machine precision, in both low- and high-dimensional settings. The method was then used to price European basket options for up to 26 assets, and was shown to converge to the reference solution in all cases. The COS-TT method improves the previous result of 6 assets significantly [9], outperforms a comparable cross approximation method [32] in both accuracy and speed, and achieves a similar result to a method based on tensor completion [24], without using Chebyshev interpolation.

## 7.2 Future Work

While the COS-TT method is a promising new approach for the valuation of multi-asset options, and for computing high-dimensional expectations from characteristic functions in general, there are several avenues for future research.

First and foremost, the method is only tested in the context of pricing European basket options under geometric Brownian motion. There are many other options, models and scenarios to which the method can be applied.

Additionally, the method can be improved in several ways. The main bottleneck of the method is the calculation of the Fourier-cosine coefficients, which requires $O(n^{d-1})$ evaluations of the characteristic function. Interpolation methods such as Chebyshev interpolation have been used successfully in the context of tensor methods [24], and could be applied here as well. This would open the door to higher-dimensional problems.

Furthermore, the quantized tensor-train format [33] can be used to reduce the complexity of the expectation calculation, if necessary. After quantics folding, the inner product can be computed in $O(d \log_q(n) r^2)$ operations instead of $O(dnr^2)$, which would result in larger speed-ups as $n$ increases.

There are also several theoretical questions that remain open. The error bound derived in this thesis is a first step in the analysis of the method, but the analysis is not yet complete. Specifically, the error due to the DMRG-greedy algorithm remains unbounded, and while the analysis in the $L^2$ norm is a good start, pointwise results should be added whereever possible. Finally, the duality between the functional tensor train and the tensor-train decomposition of the coefficient tensor should be explored further, as it might have applications beyond the COS-TT method.

Lastly, the COS-TT-CHF method can be investigated further. Currently, a TT can not be produced for more than three dimensions. However, with better sampling strategies, and the new complex-valued implementation of the DMRG-greedy algorithm, this might be possible.

# Bibliography


[1] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 2)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2. 2025.

[2] "A novel pricing method for European options based on Fourier-cosine series expansions". In: *SIAM Journal on Scientific Computing* 31.2 (2009), pp. 826–848.

[3] Volker Barthelmann, Erich Novak, and Klaus Ritter. "High dimensional polynomial interpolation on sparse grids". In: *Advances in Computational Mathematics* 12.4 (2000), pp. 273–288.

[4] John J Bartholdi III. "A good submatrix is hard to find". In: *Operations Research Letters* 1.5 (1982), pp. 190–193.

[5] Mario Bebendorf. "Approximation of boundary element matrices". In: *Numerische Mathematik* 86.4 (2000), pp. 565–589.

[6] Daniele Bigoni, Allan P Engsig-Karup, and Youssef M Marzouk. "Spectral tensor-train decomposition". In: *SIAM Journal on Scientific Computing* 38.4 (2016), A2405–A2439.

[7] J.P. Boyd. *Chebyshev & Fourier Spectral Methods*. Lecture Notes in Engineering. Springer Berlin Heidelberg, 1989. ISBN: 9783540514879.

[8] John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[9] Marnix Brands. *Solving multivariate expectations using dimension-reduced Fourier–Cosine series expansion and its application in finance*. 2023. URL: https://resolver.tudelft.nl/uuid:8d4ba32b-e72e-4092-be41-69c193bc0b11.

[10] Jacob C Bridgeman and Christopher T Chubb. "Hand-waving and interpretive dance: an introductory course on tensor networks". In: *Journal of physics A: Mathematical and theoretical* 50.22 (2017), p. 223001.

[11] Claudio Canuto et al. *Spectral methods: fundamentals in single domains*. Springer, 2006.

[12] Peter Carr and Dilip Madan. "Option valuation using the fast Fourier transform". In: *Journal of computational finance* 2.4 (1999), pp. 61–73.

[13] J Douglas Carroll and Jih-Jie Chang. "Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition". In: *Psychometrika* 35.3 (1970), pp. 283–319.

[14] Zhiming Cheng. "Dimension reduction techniques for multi-dimensional numerical integrations based on Fourier-cosine series expansion". MA thesis. Delft University of Technology (TU Delft), 2022. URL: https://resolver.tudelft.nl/uuid:f6ffbcd6-df14-425b-84bb-63e4e105205c.

[15] Patrick R. Conrad and Youssef M. Marzouk. "Adaptive Smolyak Pseudospectral Approximations". In: *SIAM Journal on Scientific Computing* 35.6 (2013), A2643–A2670. DOI: 10.1137/120890715. URL: https://doi.org/10.1137/120890715.

[16] Paul G. Constantine, Michael S. Eldred, and Eric T. Phipps. "Sparse pseudospectral approximation method". In: *Computer Methods in Applied Mechanics and Engineering* 229-232 (2012), pp. 1–12. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2012.03.019. URL: https://www.sciencedirect.com/science/article/pii/S0045782512000953.

[17] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. "A multilinear singular value decomposition". In: *SIAM journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1253–1278.

[18] Vin De Silva and Lek-Heng Lim. "Tensor rank and the ill-posedness of the best low-rank approximation problem". In: *SIAM Journal on Matrix Analysis and Applications* 30.3 (2008), pp. 1084–1127.

[19] Sergey Dolgov and Dmitry Savostyanov. "Parallel cross interpolation for high-precision calculation of high-dimensional integrals". In: *Computer Physics Communications* 246 (2020), p. 106869.

[20] Darrell Duffie, Damir Filipovic, and Walter Schachermayer. *Affine processes and application in finance*. Tech. rep. National Bureau of Economic Research, 2002.

[21] Ernst Eberlein, Kathrin Glau, and Antonis Papapantoleon. "Analysis of Fourier transform valuation formulas and applications". In: *Applied Mathematical Finance* 17.3 (2010), pp. 211–240.

[22] Robert E Edwards. "Fourier Series in L2". In: *Fourier Series: A Modern Introduction Volume 1*. Springer, 1979, pp. 130–147.

[23] FANG Fang. "The COS method: An efficient Fourier method for pricing financial derivatives". In: *Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft Institute of Applied Mathematics* (2010).

[24] Kathrin Glau, Daniel Kressner, and Francesco Statti. "Low-rank tensor approximation for Chebyshev interpolation in parametric option pricing". In: *SIAM Journal on Financial Mathematics* 11.3 (2020), pp. 897–927.

[25] Sergei A Goreinov, Eugene E Tyrtyshnikov, and Nickolai L Zamarashkin. "A theory of pseudoskeleton approximations". In: *Linear algebra and its applications* 261.1-3 (1997), pp. 1–21.

[26] Lars Grasedyck. "Hierarchical singular value decomposition of tensors". In: *SIAM journal on matrix analysis and applications* 31.4 (2010), pp. 2029–2054.

[27] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer Series in Computational Mathematics. Springer International Publishing, 2019. ISBN: 9783030355548.

[28] Wolfgang Hackbusch and Stefan Kühn. "A new scheme for the tensor representation". In: *Journal of Fourier analysis and applications* 15.5 (2009), pp. 706–722.

[29] Richard A Harshman et al. "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis". In: *UCLA working papers in phonetics* 16.1 (1970), p. 84.

[30] Frank L Hitchcock. "The expression of a tensor or a polyadic as a sum of products". In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189.

[31] Gero Junike and Hauke Stier. *From characteristic functions to multivariate distribution functions and European option prices by the damped COS method*. 2024. arXiv: 2307.12843 [q-fin.CP]. URL: https://arxiv.org/abs/2307.12843.

[32] Michael Kastoryano and Nicola Pancotti. "A highly efficient tensor network algorithm for multi-asset Fourier options pricing". In: *arXiv preprint arXiv:2203.02804* (2022).

[33] Boris N Khoromskij. "O (d log N)-quantics approximation of N-d tensors in high-dimensional numerical modeling". In: *Constructive Approximation* 34.2 (2011), pp. 257–280.

[34] Levi Klomp. "Pricing Barrier Options under Lévy Processes using Dimension-reduced Cosine Expansion". Master's thesis. Delft University of Technology, 2023. URL: https://resolver.tudelft.nl/uuid:a08e7ee1-fef0-4f9e-9e72-abe90f714a13.

[35] Tamara G Kolda and Brett W Bader. "Tensor decompositions and applications". In: *SIAM review* 51.3 (2009), pp. 455–500.

[36] JB Kruskal, RA Harshman, and ME Lundy. "How 3-MFA data can cause degenerate PARAFAC solutions, among other relationships". In: *Multiway data analysis*. 1989, pp. 115–122.

[37] Cornelius Lanczos. "A precision approximation of the gamma function". In: *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis* 1.1 (1964), pp. 86–96.

[38] Román Orús. "A practical introduction to tensor networks: Matrix product states and projected entangled pair states". In: *Annals of physics* 349 (2014), pp. 117–158.

[39] Ivan Oseledets and Eugene Tyrtyshnikov. "TT-cross approximation for multidimensional arrays". In: *Linear Algebra and its Applications* 432.1 (2010), pp. 70–88.

[40] Ivan V Oseledets. "Tensor-train decomposition". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317.

[41] Ivan V Oseledets and Eugene E Tyrtyshnikov. "Breaking the curse of dimensionality, or how to use SVD in many dimensions". In: *SIAM Journal on Scientific Computing* 31.5 (2009), pp. 3744–3759.

[42] Zhen Qin et al. "Error analysis of tensor-train cross approximation". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 14236–14249.

[43] Marjon J Ruijter and Cornelis W Oosterlee. "Two-dimensional Fourier cosine series expansion method for pricing financial options". In: *SIAM Journal on Scientific Computing* 34.5 (2012), B642–B671.

[44] Michael Samet et al. "Optimal damping with hierarchical adaptive quadrature for efficient fourier pricing of multi-asset options in L\'evy models". In: *arXiv preprint arXiv:2203.08196* (2022).

[45] Dmitry V Savostyanov. "Quasioptimality of maximum-volume cross interpolation of tensors". In: *Linear Algebra and its Applications* 458 (2014), pp. 217–244.

[46] Ulrich Schollwöck. "The density-matrix renormalization group". In: *Reviews of modern physics* 77.1 (2005), pp. 259–315.

[47] Ulrich Schollwöck. "The density-matrix renormalization group in the age of matrix product states". In: *Annals of physics* 326.1 (2011), pp. 96–192.

[48] Sergey Smolyak. "Quadrature and interpolation formulas for tensor products of certain classes of functions". In: *Soviet Math. Dokl.* Vol. 4. 1963, pp. 240–243.

[49] Michael Steinlechner. "Riemannian optimization for high-dimensional tensor completion". In: *SIAM Journal on Scientific Computing* 38.5 (2016), S461–S484.

[50] Ledyard R Tucker. "Some mathematical notes on three-mode factor analysis". In: *Psychometrika* 31.3 (1966), pp. 279–311.

[51] M Alex O Vasilescu and Demetri Terzopoulos. "Multilinear analysis of image ensembles: Tensorfaces". In: *European conference on computer vision*. Springer. 2002, pp. 447–460.

[52] V. J. Veenman. "Pricing barrier options using a fully interpretable neural network". Master's thesis. Delft University of Technology, 2024. URL: https://resolver.tudelft.nl/uuid:7103911a-8015-4e42-a671-bbf7bbd1221b.

[53] Steven R. White. "Density matrix formulation for quantum renormalization groups". In: *Phys. Rev. Lett.* 69 (19 Nov. 1992), pp. 2863–2866. DOI: 10.1103/PhysRevLett.69.2863. URL: https://link.aps.org/doi/10.1103/PhysRevLett.69.2863.

# Ch. A

# Proofs

This appendix contains the proofs of theorems and propositions stated in the main text.

## A.1 Inner integrals

**Lemma 3.** The solution to the inner integral is given by

$$\int_{\mathbf{a}}^{\mathbf{b}} e^{-i\boldsymbol{\omega}\cdot\mathbf{x}} \prod_{j=1}^{n} \cos\left(k_j\pi\frac{x_j-a_j}{b_j-a_j}\right) d\mathbf{x} = \prod_{j=1}^{n} \frac{(b_j-a_j)}{2} I_j(\omega_j,k_j), \tag{A.1}$$

which, in the case $\omega_j = 0$, is given by

$$I_j(0,k_j) = \begin{cases} 0 & k_j \neq 0, \\ b_j - a_j & k_j = 0, \end{cases} \tag{A.2}$$

and in the case $\omega_j \neq 0$ is given by

$$I_j(\omega_j,k_j) = \begin{cases} e^{-i\omega_j a_j}\left(\frac{1-e^{-i\left[(b_j-a_j)\omega_j-k_j\pi\right]}}{i\left[(b_j-a_j)\omega_j-k_j\pi\right]} + \frac{1-e^{-i\left[(b_j-a_j)\omega_j+k_j\pi\right]}}{i\left[(b_j-a_j)\omega_j+k_j\pi\right]}\right) & k_j \neq 0, \\ \frac{e^{-i\omega_j a_j}-e^{-i\omega_j b_j}}{i\omega_j} & k_j = 0. \end{cases} \tag{A.3}$$

*Proof.* Start begin by turning it into a product of one-dimensional integrals.

$$\begin{aligned}\int_{\mathbf{a}}^{\mathbf{b}} e^{-i\boldsymbol{\omega}\cdot\mathbf{x}} \prod_{j=1}^{n} \cos\left(k_j\pi\frac{x_j-a_j}{b_j-a_j}\right) d\mathbf{x} &= \int_{\mathbf{a}}^{\mathbf{b}} \prod_{j=1}^{n} e^{-i\omega_j x_j} \cos\left(k_j\pi\frac{x_j-a_j}{b_j-a_j}\right) d\mathbf{x} \\ &= \prod_{j=1}^{n} \int_{a_j}^{b_j} e^{-i\omega_j x_j} \cos\left(k_j\pi\frac{x_j-a_j}{b_j-a_j}\right) dx_j\end{aligned}$$

These one-dimensional integrals can be solved explicitly.

First, consider the case $\omega_j = 0$. If $k_j = 0$, then the integral evaluates to $b_j - a_j$. If $k_j \neq 0$, then the integral evaluates

$$\int_{a_j}^{b_j} \cos\left(k_j\pi\frac{x_j-a_j}{b_j-a_j}\right) dx_j = \frac{(b_j-a_j)}{k_j\pi} \sin\left(k_j\pi\frac{x_j-a_j}{b_j-a_j}\right)\Bigg|_{a_j}^{b_j} = 0.$$

Now consider the case where $\omega_j \neq 0$. Again, if $k_j = 0$, then this gives $\int_{a_j}^{b_j} e^{-i\omega_j x_j} dx_j$. This is a standard integral of the form $\int e^{-i\alpha u} du = \frac{1-e^{-i\alpha}}{i\alpha}$, which evaluates to

$$\int_{a_j}^{b_j} e^{-i\omega_j x_j} dx_j = \frac{e^{-i\omega_j a_j} - e^{-i\omega_j b_j}}{i\omega_j}.$$

In the case that both $\omega_j \neq 0$ and $k_j \neq 0$, substitute $u_j = \frac{x_j - a_j}{b_j - a_j}$, which gives $x_j = u_j(b_j - a_j) + a_j$ and $dx_j = (b_j - a_j)du_j$. The limits of integration change accordingly: when $x_j = a_j$, $u_j = 0$; when $x_j = b_j$, $u_j = 1$. Thus, this gives

$$\int_{a_j}^{b_j} e^{-i\omega_j x_j} \cos\left(k_j \pi \frac{x_j - a_j}{b_j - a_j}\right) dx_j = \int_0^1 e^{-i\omega_j (u_j(b_j - a_j) + a_j)} \cos\left(k_j \pi u_j\right) (b_j - a_j) du_j$$
$$= (b_j - a_j) e^{-i\omega_j a_j} \int_0^1 e^{-i\omega_j (b_j - a_j) u_j} \cos\left(k_j \pi u_j\right) du_j.$$

Noting that $\cos(k_j \pi u_j) = \frac{1}{2}\left(e^{ik_j \pi u_j} + e^{-ik_j \pi u_j}\right)$, we can continue, and write

$$= \frac{(b_j - a_j)}{2} e^{-i\omega_j a_j} \int_0^1 e^{-i\omega_j (b_j - a_j) u_j} \left(e^{ik_j \pi u_j} + e^{-ik_j \pi u_j}\right) du_j$$
$$= \frac{(b_j - a_j)}{2} e^{-i\omega_j a_j} \left( \int_0^1 e^{-i\left[(b_j - a_j)\omega_j - k_j \pi\right] u_j} du_j + \int_0^1 e^{-i\left[(b_j - a_j)\omega_j + k_j \pi\right] u_j} du_j \right).$$

The integrals are again standard integrals of the form $\int e^{-i\alpha u} du = \frac{1 - e^{-i\alpha}}{i\alpha}$, which gives us

$$= \frac{(b_j - a_j)}{2} e^{-i\omega_j a_j} \left( \frac{1 - e^{-i\left[(b_j - a_j)\omega_j - k_j \pi\right]}}{i\left[(b_j - a_j)\omega_j - k_j \pi\right]} + \frac{1 - e^{-i\left[(b_j - a_j)\omega_j + k_j \pi\right]}}{i\left[(b_j - a_j)\omega_j + k_j \pi\right]} \right),$$

which concludes the proof. □

## A.2 Error Analysis of COS Method

**Proposition 13.** Let $f \in L^2([0, L])$, and define the cosine basis functions

$$\phi_k(x) = \cos\left(\frac{k\pi}{L} x\right), \qquad A_k = \frac{2}{L} \int_0^L f(x) \phi_k(x) dx, \quad k \geq 0 \tag{A.4}$$

and the truncated cosine series

$$\hat{f}(x) = \frac{A_0}{2} + \sum_{k=1}^{K} A_k \phi_k(x) \tag{A.5}$$

Then

$$\|f - \hat{f}\|_{L^2}^2 = \frac{L}{2} \sum_{k=K+1}^{\infty} |A_k|^2.$$

This follows from Parseval's formula, and the necessary details can be found in [22], p. 124, for example in equation 8.2.4 or the proof of 8.2.1. Alternatively, a proof can be found below.

*Proof.*

$$f - \hat{f} = \sum_{k=K+1}^{\infty} A_k \phi_k$$

Then the norm can be computed as

$$\|f - \hat{f}\|_{L^2}^2 = \| \sum_{k=K+1}^{\infty} A_k \phi_k \|_{L^2}^2 = \sum_{k=K+1}^{\infty} \sum_{j=K+1}^{\infty} A_k A_j \langle \phi_k, \phi_j \rangle_{L^2}$$

Using the orthogonality of cosines,

$$\langle \phi_k, \phi_j \rangle_{L^2} = \int_a^b \phi_k(x)\phi_j(x)dx = \begin{cases} 0 & k \neq j, \\ \frac{b-a}{2} & k = j \neq 0, \\ b-a & k = j = 0, \end{cases} \tag{A.6}$$

The sum reduces to

$$\sum_{k=K+1}^{\infty} \sum_{j=K+1}^{\infty} A_k A_j \langle \phi_k, \phi_j \rangle_{L^2} = \sum_{k=K+1}^{\infty} |A_k|^2 \langle \phi_k, \phi_k \rangle_{L^2} = \frac{L}{2} \sum_{k=K+1}^{\infty} |A_k|^2$$

□

**Proposition 14** (Coefficient Approximation Error)**.** The coefficient approximation error can be bounded as

$$\|\varepsilon_3\|_{L^2} \leq \sqrt{\frac{2(K+3)}{b-a}}\, \varepsilon_4,$$

where $\varepsilon_4$ is defined in equation (2.7).

*Proof.* First, the difference between the two approximations can be expressed as

$$\begin{aligned}
\varepsilon_3 &= {\sum_{k=0}^{K}}' (A_k - F_k) \cos\left(\frac{k\pi}{b-a}(x-a)\right) \\
&= {\sum_{k=0}^{K}}' \frac{2}{b-a} \left( \mathrm{Re}\left\{ e^{-i\frac{k\pi}{b-a}a} \left( \widehat{\varphi}\left(\frac{k\pi}{b-a}\right) - \varphi\left(\frac{k\pi}{b-a}\right)\right)\right\}\right) \cos\left(\frac{k\pi}{b-a}(x-a)\right) \\
&= {\sum_{k=0}^{K}}' \frac{2}{b-a} \mathrm{Re}\left\{ e^{-i\frac{k\pi}{b-a}a} \int_{\mathbb{R}\setminus[a,b]} f(t) e^{i\frac{k\pi}{b-a}t} dt \right\} \cos\left(\frac{k\pi}{b-a}(x-a)\right) \\
&= {\sum_{k=0}^{K}}' \frac{2}{b-a} \mathrm{Re}\left\{ \int_{\mathbb{R}\setminus[a,b]} f(t) e^{i\frac{k\pi}{b-a}(t-a)} dt \right\} \cos\left(\frac{k\pi}{b-a}(x-a)\right)
\end{aligned}$$

Noting that $f$ is real-valued, and using Euler's formula, this can be simplified to

$$\varepsilon_3 = {\sum_{k=0}^{K}}' \frac{2}{b-a} \left( \int_{\mathbb{R}\setminus[a,b]} f(t) \cos\left(\frac{k\pi}{b-a}(t-a)\right) dt \right) \cos\left(\frac{k\pi}{b-a}(x-a)\right).$$

Now, define

$$E_4(k) := \frac{2}{b-a} \int_{\mathbb{R}\setminus[a,b]} f(t) \cos\left(\frac{k\pi}{b-a}(t-a)\right) dt.$$

Then it follows that

$$\|\varepsilon_3\|_{L^2}^2 = \int_a^b \left| {\sum_{k=0}^{K}}' E_4(k) \cos\left(\frac{k\pi}{b-a}(x-a)\right) \right|^2 dx.$$

Using orthogonality of the cosine basis, this can be simplified to

$$\|\varepsilon_3\|_{L^2}^2 = (b-a)|E_4(0)|^2 + \frac{b-a}{2} \sum_{k=0}^{K} |E_4(k)|^2$$

Focussing on $E_4(k)$, it can be bounded by

$$\begin{aligned}|E_4(k)| &\leq \frac{2}{b-a}\int_{\mathbb{R}\setminus[a,b]} |f(t)|\left|\cos\left(\frac{k\pi}{b-a}(t-a)\right)\right| dt \\ &\leq \frac{2}{b-a}\int_{\mathbb{R}\setminus[a,b]} |f(t)|dt \\ &\leq \frac{2}{b-a}\int_{\mathbb{R}\setminus[a,b]} f(t)dt = \frac{2}{b-a}\varepsilon_4,\end{aligned}$$

Thus, the resulting $L^2$ error can be bounded as

$$\begin{aligned}\|\varepsilon_3\|_{L^2}^2 &\leq (b-a)\left(\frac{2}{b-a}\varepsilon_4\right)^2 + \frac{b-a}{2}(K+1)\left(\frac{2}{b-a}\varepsilon_4\right)^2 \\ &= \frac{4}{b-a}\varepsilon_4^2 + \frac{2(K+1)}{b-a}\varepsilon_4^2 \\ &= \frac{2(K+3)}{b-a}\varepsilon_4^2\end{aligned}$$

The result then follows by taking the square root. □

# Ch. B

# Discrete Approach

A consequence of theorem 1, is that multidimensional quadrature can be performed linearly in $d$, as shown in the following corollary.

**Corollary 2** (Multidimensional quadrature with TT). Suppose we have a function $f(\mathbf{x})$, and we want to compute the integral

$$I = \int_{[a,b]^d} f(\mathbf{x})\, d\mathbf{x}$$

using numerical quadrature. Let $\{x^{(i_j)}\}_{i=1}^{n_j}$ be quadrature nodes and $\{w_{i_j}\}_{i=1}^{n_j}$ the corresponding quadrature weights for each dimension $j$. Then, the integral is approximated by

$$I \approx \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} w_{i_1} \cdots w_{i_d}\, f(x^{(i_1)}, \dots, x^{(i_d)}).$$

If $f$ is decomposed in tensor train format, i.e. $f\left(x^{(i_1)}, \dots, x^{(i_d)}\right) = \prod_{j=1}^{d} F_j(i_j)$, then the integral can be approximated by

$$I \approx \prod_{j=1}^{d} \left( \sum_{i_j=1}^{n_j} w_j F_j(i_j) \right).$$

## B.1 Existing Method 1: Use known PDF with TT

First, consider the case where the probability density function $f(\mathbf{x})$ is known. We apply numerical quadrature to compute the expectation. Let $\{x^{(i_j)}\}_{i=1}^{N_j}$ be quadrature nodes and $\{w_{i_j}\}_{i=1}^{N_j}$ the corresponding quadrature weights for each dimension $j$. Then, with full numerical quadrature, the integral can be approximated as

$$\int_{\mathbb{R}^d} g(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x} \approx \sum_{i_1=1}^{N_1} \cdots \sum_{i_d=1}^{N_d} w_{i_1} g_1(x^{(i_1)}) \cdots w_{i_d} g_d(x^{(i_d)})\, f(x^{(i_1)}, \dots, x^{(i_d)})$$

Next, treat the discretized function $f$ as a tensor $\mathcal{G}$, with elements $\mathcal{G}[i_1, i_2, \dots, i_d] = f(x^{(i_1)}, \dots, x^{(i_d)})$. If $\mathcal{G}$ is decomposed by a tensor train decomposition, $\mathcal{G}[i_1, i_2, \dots, i_d] \approx \prod_{j=1}^{d} G_j(i_j)$, the contraction principle can be used to avoid explicit summation. That is,

$$\sum_{i_1=1}^{N_1} \cdots \sum_{i_d=1}^{N_d} w_{i_1} g_1(x^{(i_1)}) \cdots w_{i_d} g_d(x^{(i_d)})\, f_{\mathbf{X}}(x^{(i_1)}, \dots, x^{(i_d)}) \approx \prod_{j=1}^{d} \left( \sum_{i_j=1}^{N_j} w_{i_j} g_j(x^{(i_j)}) G_j(i_j) \right). \quad \text{(B.1)}$$

## B.2 Novel Method 1: COS-TT

If the probability distribution of the process is not known, it can be approximated by the multidimensional COS method aproximation in equation (2.21). The coefficient tensor $\mathcal{F}$ admits a rank-$\mathbf{r}$ tensor train decomposition, and can be decomposed into tensor train format,

$$\mathcal{F}[k_1, \dots, k_d] \approx \prod_{j=1}^{d} F_j(k_j)$$

The resulting approximation of $f$ is then

$$f(\mathbf{x}) \approx \prod_{j=1}^{d} \left( \sum_{k_j=0}^{K_j} F_j(k_j) \phi_{k_j}^{(j)}(x_j) \right) \tag{B.2}$$

To compute the expectation, the contraction principle of theorem 1 can be applied again. This leads to the following result.

**Theorem 9** (COS-TT)**.** The expectation in equation (4.3) can be approximated by

$$\int_{\mathbf{a}}^{\mathbf{b}} g(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x} \approx \prod_{j=1}^{d} \left( \sum_{k_j=0}^{K_j} F_j(k_j) \int_{a_j}^{b_j} g_j(x) \phi_{k_j}^{(j)}(x_j)\, dx \right). \tag{B.3}$$

*Proof.* The assumption when approximating $f(\mathbf{x})$ by a cosine expansion is that $[\mathbf{a}, \mathbf{b}]$ is chosen such that the integral is sufficiently approximated by the truncated one, i.e.

$$\int_{\mathbb{R}^d} g(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x} \approx \int_{\mathbf{a}}^{\mathbf{b}} g(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x}.$$

Continuing, we substitute the cosine expansion for $f(\mathbf{x})$, and take the integral inside the sum.

$$\begin{aligned}
\int_{\mathbf{a}}^{\mathbf{b}} g(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x} &\approx \int_{\mathbf{a}}^{\mathbf{b}} g(\mathbf{x}) \left( \sum_{\mathbf{k}}^{\mathbf{K}} \mathcal{F}_{\mathbf{k}} \Phi_{\mathbf{k}} \right) d\mathbf{x} \\
&= \sum_{\mathbf{k}}^{\mathbf{K}} \mathcal{F}_{\mathbf{k}} \int_{\mathbf{a}}^{\mathbf{b}} g(\mathbf{x}) \Phi_{\mathbf{k}}\, d\mathbf{x} \\
&= \sum_{\mathbf{k}}^{\mathbf{K}} \mathcal{F}_{\mathbf{k}} \prod_{j=1}^{d} \int_{a_j}^{b_j} g_j(x) \phi_{k_j}^{(j)}(x_j)\, dx
\end{aligned}$$

Then, we approximate the coefficient tensor by a tensor train decomposition, i.e. $\mathcal{F}_{\mathbf{k}} \approx \prod_{j=1}^{d} F_j(k_j)$. Together with the contraction principle, we find the final result.

$$\begin{aligned}
\sum_{\mathbf{k}}^{\mathbf{K}} \mathcal{F}_{\mathbf{k}} \prod_{j=1}^{d} \int_{a_j}^{b_j} g_j(x) \phi_{k_j}^{(j)}(x_j)\, dx &\approx \sum_{\mathbf{k}}^{\mathbf{K}} \prod_{j=1}^{d} F_j(k_j) \int_{a_j}^{b_j} g_j(x) \phi_{k_j}^{(j)}(x_j)\, dx \\
&= \prod_{j=1}^{d} \left( \sum_{k_j=0}^{K_j} F_j(k_j) \int_{a_j}^{b_j} g_j(x) \phi_{k_j}^{(j)}(x_j)\, dx \right)
\end{aligned}$$

□

As shown in theorem 8, for certain payoff functions, the integral

$$\int_{a_j}^{b_j} g_j(x)\phi_{k_j}^{(j)}(x_j)\,dx$$

has a closed-form solution. In the case that no explicit solution is available, numerical integration methods can be employed. The resulting approximation is then

$$\int_{\mathbb{R}^d} g(\mathbf{x})f(\mathbf{x})\,d\mathbf{x} \approx \prod_{j=1}^{d}\left(\sum_{k_j=0}^{K_j} F_j(k_j)\sum_{i_j=1}^{N_j} w_{i_j} g_j(x^{(i_j)})\phi_{k_j}^{(j)}(x^{(i_j)})\right). \tag{B.4}$$

## B.3 Method 3: COS-TT-CHF

An idea to avoid the exponential scaling of the COS-TT method is to directly decompose the joint characteristic function $\varphi(\boldsymbol{\omega})$ of the process. As derived in section 4.3, the coefficients can be written in terms of the joint characteristic function by equation (4.19), as

$$\mathcal{C}_{\mathbf{k}} = \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} \varphi(\boldsymbol{\omega})\mathbf{I}_{\mathbf{k}}(\boldsymbol{\omega})\,d\boldsymbol{\omega} = \frac{\langle\varphi, \mathbf{I}_{\mathbf{k}}\rangle}{(2\pi)^d}. \tag{B.5}$$

Instead of decomposing the joint characteristic function with a functional tensor train decomposition, we discretize it on quadrature nodes $\{\omega^{(i_j)}\}_{i_j=1}^{N_j}$. Then, the discretized joint characteristic function can be approximated

$$\varphi(\omega^{(i_1)}, \dots, \omega^{(i_d)}) \approx \prod_{j=1}^{d} G_j(i_j).$$

We then substitute this into the integral, and approximate the integral by numerical integration.

$$\begin{aligned}
\langle\varphi, \mathbf{I}_{\mathbf{k}}\rangle &\approx \sum_{i_1=1}^{N_1}\cdots\sum_{i_d=1}^{N_d} w_{i_1}\cdots w_{i_d}\,\varphi(\omega^{(i_1)}, \dots, \omega^{(i_d)})\mathbf{I}_{\mathbf{k}}(\omega^{(i_1)}, \dots, \omega^{(i_d)}) \\
&\approx \sum_{i_1=1}^{N_1}\cdots\sum_{i_d=1}^{N_d}\prod_{j=1}^{d} w_{i_j} G_j(i_j) I_j(\omega^{(i_j)}, k_j) \\
&= \prod_{j=1}^{d}\left(\sum_{i_j=1}^{N_j} w_j\, G_j(i_j)\, I_j(\omega^{(i_j)}, k_j)\right)
\end{aligned}$$

Thus we conclude that the coefficients can be approximated by

$$\boxed{\mathcal{C}_{\mathbf{k}} \approx \prod_{j=1}^{d}\left(\frac{1}{2\pi}\sum_{i_j}^{N_j} w_j\, I_j(\omega^{(i_j)}, k_j)\, G_j(i_j)\right).} \tag{B.6}$$

This result is more elegant than it seems. We have essentially written the coefficients as matrices $H_j(k_j)$, where each $H_j(k_j)$ is a function of the index $k_j$. We note that each $H_j(k_j)$ is of size $r_{j-1} \times r_j$, just like the matrices in the TT format. Thus, we can treat the coefficients as

a transformation of the tensor train of $\varphi(\boldsymbol{\omega})$ on quadrature points $\omega^{(i_j)}$, to a tensor train of the coefficients $\mathcal{C}_{\mathbf{k}}$ on the cosine terms $k_j$. Therefore, we write

$$\mathcal{C}_{\mathbf{k}} \approx \prod_{j=1}^{d} H_j(k_j)$$

From this point on, the derivation is exactly the same as for theorem 9. Thus, we find

$$\int_{\mathbf{a}}^{\mathbf{b}} g(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x} \approx \prod_{j=1}^{d} \left( \sum_{k_j=0}^{K_j} H_j(k_j) \int_{a_j}^{b_j} g_j(x) \phi_{k_j}^{(j)}(x)\, dx \right).$$

or, substituting for $H_j(k_j)$,

$$\int_{\mathbf{a}}^{\mathbf{b}} g(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x} \approx \prod_{j=1}^{d} \left( \sum_{k_j=0}^{K_j} \left( \frac{1}{2\pi} \sum_{i_j}^{N_j} w_j\, I_j(\omega^{(i_j)}, k_j)\, G_j(i_j) \right) \int_{a_j}^{b_j} g_j(x) \phi_{k_j}^{(j)}(x)\, dx \right).$$

# Ch. C

# Pricing without log-transform

The COS method, as introduced in section 2.5, relies on the characteristic function of the log-asset price. However, in some cases, the characteristic function of the regular asset price is known instead. This requires different coefficients $V_k$ in the COS pricing formula in equation (2.22). In this section the coefficients are derived for pricing under the asset price, and under the ratio of the asset price and the strike price.

Consider an asset with price $S_t$ at time $t$, and initial price $S_0$ at time $t_0$. The payoff for European call and put options is given by

$$\Lambda(x,T) = (\alpha(x-K))^+ \qquad \text{with} \qquad \alpha = \begin{cases} 1 & \text{for a call,} \\ -1 & \text{for a put,} \end{cases}$$

where $K$ is the strike price and the $+$ denotes the positive part, i.e. $x^+ = \max(x,0)$. Furthermore, let $\psi_k$, $\xi_k$ and $\chi_k$ be the following integrals:

$$\begin{aligned} \psi_k(c,d) &= \int_c^d \cos\left(k\pi\frac{y-a}{b-a}\right) dy, \\ \xi_k(c,d) &= \int_c^d y\cos\left(k\pi\frac{y-a}{b-a}\right) dy, \\ \chi_k(c,d) &= \int_c^d e^y\cos\left(k\pi\frac{y-a}{b-a}\right) dy. \end{aligned}$$

The solutions to these integrals, when $k \neq 0$, are given by

$$\begin{aligned} \psi_k(c,d) &= \frac{b-a}{k\pi}\left[\sin\left(k\pi\frac{y-a}{b-a}\right)\right]_c^d, \\ \xi_k(c,d) &= \frac{b-a}{k\pi}\left[y\sin\left(k\pi\frac{y-a}{b-a}\right) + \frac{b-a}{k\pi}\cos\left(k\pi\frac{y-a}{b-a}\right)\right]_c^d, \\ \chi_k(c,d) &= \frac{1}{1+\left(\frac{k\pi}{b-a}\right)^2}\left[\cos\left(k\pi\frac{y-a}{b-a}\right)e^y + \frac{k\pi}{b-a}\sin\left(k\pi\frac{y-a}{b-a}\right)e^y\right]_c^d \end{aligned}$$

and, when $k = 0$, are given by

$$\begin{aligned} \psi_0(c,d) &= d-c, \\ \xi_0(c,d) &= \frac{1}{2}(d^2-c^2), \\ \chi_0(c,d) &= e^d - e^c. \end{aligned}$$

### Asset price

Assume that the characteristic function of the asset price, given by $\varphi_{S_T}(\omega \mid y) = \mathbb{E}\left[e^{i\omega S_T} \mid S_0 = y\right]$ is known. The coefficients $V_k$ are then given by the integral

$$V_k = \frac{2}{b-a}\int_a^b (\alpha(x-K))^+\cos\left(k\pi\frac{x-a}{b-a}\right) dx.$$

**Call** For a call option, the integral has non-zero contribution only when $x - K \geq 0$, so for $x \geq K$.

$$V_k = \frac{2}{b-a} \int_K^b (x - K) \cos\left(k\pi \frac{x-a}{b-a}\right) dx$$

As a result,

$$\begin{aligned} V_k^{\text{call}} &= \frac{2}{b-a} \left( \int_K^b x \cos\left(k\pi \frac{x-a}{b-a}\right) dx - K \int_K^b \cos\left(k\pi \frac{x-a}{b-a}\right) dx \right) \\ &= \frac{2}{b-a} \left(\xi_k(K, b) - K\psi_k(K, b)\right). \end{aligned}$$

**Put** For a put option, the integral has non-zero contribution only when $K - x \geq 0$, so for $x \leq K$.

$$V_k = \frac{2}{b-a} \int_a^K (K - x) \cos\left(k\pi \frac{x-a}{b-a}\right) dx$$

As a result,

$$\begin{aligned} V_k^{\text{put}} &= \frac{2}{b-a} \left( K \int_a^K \cos\left(k\pi \frac{x-a}{b-a}\right) dx - \int_a^K x \cos\left(k\pi \frac{x-a}{b-a}\right) dx \right) \\ &= \frac{2}{b-a} \left(-\xi_k(a, K) + K\psi_k(a, K)\right). \end{aligned}$$

### Price ratio

To price under the price ratio $Y_T := S_T/K$, the variables need to be transformed accordingly. Then

$$\Lambda(x, T) = \left(\alpha\left(\frac{x}{K} - K\right)\right)^+ = (\alpha K(x-1))^+.$$

Assuming that the characteristic function of the asset price price is known, $\varphi_{S_T}(\omega \mid y)$, the characteristic function of the ratio can be derived.

$$\begin{aligned} \varphi_{Y_T}(\omega \mid y) &= \mathbb{E}\left[e^{i\omega Y} \mid Y_0 = y\right] \\ &= \mathbb{E}\left[e^{i\omega \frac{S_T}{K}} \mid \frac{S_0}{K} = y\right] \\ &= \mathbb{E}\left[e^{i\frac{\omega}{K} S_T} \mid S_0 = yK\right] \\ &= \varphi_{S_T}\left(\frac{\omega}{K} \mid yK\right) \end{aligned}$$

Thus, the COS pricing formula in equation (2.22) can be applied with $\varphi\left(\frac{\omega}{K} \mid yK\right)$, and coefficients $V_k$ given by the integral

$$V_k = \frac{2}{b-a} \int_a^b (\alpha K(x-1))^+ \cos\left(k\pi \frac{x-a}{b-a}\right) dx$$

**Call** For a call option, the integral has non-zero contribution only when $x - 1 \geq 0$, so for $x \geq 1$.

$$V_k = \frac{2}{b-a} \int_1^b K(x-1) \cos\left(k\pi \frac{x-a}{b-a}\right) dx$$

As a result,

$$V_k^{\text{call}} = \frac{2}{b-a} K \left( \int_1^b x \cos\left(k\pi \frac{x-a}{b-a}\right) dx - \int_1^b \cos\left(k\pi \frac{x-a}{b-a}\right) dx \right)$$
$$= \frac{2}{b-a} K \left( \xi_k(1,b) - \psi_k(1,b) \right).$$

**Put** For a put option, the integral has non-zero contribution only when $1 - x \geq 0$, so for $x \leq 1$.

$$V_k = \frac{2}{b-a} \int_a^1 K(1-x) \cos\left(k\pi \frac{x-a}{b-a}\right) dx$$

As a result,

$$V_k^{\text{put}} = \frac{2}{b-a} K \left( \int_a^1 \cos\left(k\pi \frac{x-a}{b-a}\right) dx - \int_a^1 x \cos\left(k\pi \frac{x-a}{b-a}\right) dx \right)$$
$$= \frac{2}{b-a} K \left( -\xi_k(a,1) + \psi_k(a,1) \right).$$