Function approximation for option pricing and risk management

Methods, theory and applications

by

Christian Pötz

Submitted in partial fulfillment of the requirements of the degree of
Doctor of Philosophy

School of Mathematical Sciences
Queen Mary, University of London
United Kingdom

July 2020

Statement of originality

I, Christian Pötz, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.
I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.
I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.
I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.
The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature: *Christian Pötz*
Date: July 2, 2020

Details of collaboration and publications:

- *The Chebyshev method for the implied volatility.* Joint work with Kathrin Glau, Paul Herold and Dilip B. Madan. Published in the *Journal of Computations Finance*, 23(3).

- *A new approach for American option pricing: The Dynamic Chebyshev method.* Joint work with Kathrin Glau and Mirco Mahlstedt. Published in the *SIAM Journal of Scientific Computing*, 41(1):B153–B180.

- *Speed-up credit exposure calculation of pricing and risk management.* Joint work with Kathrin Glau and Ricardo Pachon. *Quantitative Finance (2020)*: 1-19.

- *Fast calculation of credit exposures for barrier and Bermudan options using Chebyshev interpolation.* Joint work with Kathrin Glau and Ricardo Pachon. Earlier version of the paper mentioned above.

- *Spread option in a 2D Black–Scholes model.* Collaboration with Kathrin Glau, Behnam Hashemi and Mirco Mahlstedt. Example for *chebfun3* in the *chebfun* toolbox, available at *http://www.chebfun.org/examples/applics/BlackScholes2D.html.*

Abstract

This thesis investigates the application of function approximation techniques for computationally demanding problems in finance. We focus on the use of Chebyshev interpolation and its multivariate extensions. The main contribution of this thesis is the development of a new pricing method for path-dependent options. In each step of the dynamic programming time-stepping we approximate the value function with Chebyshev polynomials. A key advantage of this approach is that it allows us to shift all model-dependent computations into a pre-computation step. For each time step the method delivers a closed form approximation of the price function along with the options' delta and gamma. We provide a theoretical error analysis and find conditions that imply explicit error bounds. Numerical experiments confirm the fast convergence of prices and sensitivities. We use the new method to calculate credit exposures of European and path-dependent options for pricing and risk management. The simple structure of the Chebyshev interpolation allows for a highly efficient evaluation of the exposures. We validate the accuracy of the computed exposure profiles numerically for different equity products and a Bermudan swaption. Benchmarking against the least-squares Monte Carlo approach shows that our method delivers a higher accuracy in a faster runtime. We extend the method to efficiently price early-exercise options depending on several risk-factors. As an example, we consider the pricing of callable bonds in a hybrid two-factor model. We develop an efficient and stable calibration routine for the model based on our new pricing method. Moreover, we consider the pricing of early-exercise basket options in a multivariate Black-Scholes model. We propose a numerical smoothing in the dynamic programming time-stepping using the smoothing property of a Gaussian kernel. An extensive numerical convergence analysis confirms the efficiency.

Acknowledgments

First and foremost, I would like to thank my supervisor Kathrin Glau for her vigorous support during the last four years. Her own passion and commitment for research as well as her scientific curiosity and creativity have driven this thesis forward. Throughout my studies, she has always given me the opportunity to develop my own ideas, valued my contributions and our fruitful discussions have been a great help for me. Without her guidance and support this thesis would not have been possible. Furthermore, I would like to thank my other co-authors Paul Herold, Dilip B. Madan, Mirco Mahlstedt and Ricardo Pachon for their good collaboration as well as their extensive feedback and interesting inputs.

I thank all the members of the chair of M13 at the Technical University of Munich, especially my former PhD colleagues, for making the first one and half years of my PhD such a positive experience. My deep gratitude goes to the management board of the KPMG Center of Excellence in Risk Management for financing my PhD position in Munich. The possibility to do a three months industry placement at KPMG to gain insight into consulting is gratefully acknowledged. In this context, I would like to thank Franz Lorenz for his great supervision throughout this time.

I am very grateful for the financial support from the School of Mathematical Sciences for the second part of my studies at Queen Mary University of London. I thank my former and current fellow PhD students who gave me a warm welcome and made my time in London such an enjoyable and amazing experience. Many thanks to Wolfram Just for becoming my second supervisor and to Alexander Gnedin for his help and support as the head of our research group. My deep gratitude goes to Mikhail Soloveitchik for giving me the opportunity to apply my university research to a real-world problem during an industry placement at HSBC. Moreover, a big thank you to Linus Wunderlich and Domagoj Demeterfi for proof reading chapters of this thesis.

Furthermore, I thank my parents for their constant support and encouragement throughout my entire studies. Most of all, I would like to thank Annkatrin for her love and support during these intense years.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

At the heart of modern mathematical finance stands the pricing of financial derivatives that depend on the price of one or more underlyings. The most important class of derivatives that are investigated in mathematical finance are option contracts. In the simplest case, the price of an option can be calculated as its discounted expected payoff at maturity under the so called pricing measure $\mathbb{Q}$ also known as the risk-neutral measure. The resulting pricing approach is often called risk-neutral valuation and was introduced by [33], see also [14] for more details. The core of computational finance is the numerical computation of such option prices if there is no analytic solution. See for instance [115] and [101] for an introduction to computational finance. The complexity of this task is determined by the type of option that is priced and the stochastic model used for the underlying risk factors. The price of a European vanilla call or put option depends only on the distribution of the underlying at maturity whereas path-dependent options such as early-exercise and barrier option depend on the distribution of the underlying over the option's lifetime. An American option is an option that gives the option holder the right to exercise his right at any time point until the maturity of the option. The majority of stock options traded in the market are of American type whereas equity index options are often European options, see for example [72]. The early exercise feature makes the valuation of the option more challenging. More exotic financial products require usually also more complex stochastic models that capture relevant additional risks such as stochastic volatility, jumps or stochastic interest rates. Further computational complexity occurs for products on more than one underlying such as basket options or call options on the maximum of different assets. We refer to [136] for a more practical introduction to option pricing and to [14] as well as [41] for a comprehensive overview on the mathematical theory behind option pricing. The first and main step in computational finance is to find efficient numerical methods that compute the price of a given option

in a suitable stochastic model. Here, efficiency refers to the accuracy of the resulting option price in comparison to the runtime spent on its computation. In a second step, one is often also interested in the sensitivities of the option price, i.e. how does the value of the option change for a small change in the value of the underlying. There are two main applications for numerical option pricing methods, on the one side the trading and hedging of options and on the other side the risk management of a portfolio of derivative contracts. See [96] for an interesting comparison of these two worlds of quantitative finance. For many years, trading was the most important application of new option pricing methods. However, in recent years, there has been a shift from trading to risk management in quantitative finance. Driven by new regulatory requirements, the risk management of large trading books of derivatives has become more and more computationally demanding. In the following, we will briefly discuss how option pricing differs between trading and risk management.

Option pricing in trading and risk management

For trading, the price of an (exotic) product has to be computed in order to sell it to a counterparty. First, the relevant risk factors are identified for the specific product and a suitable stochastic model is chosen. The critical part is then the calibration of the model to market data in order to make sure that the price of the exotic product is free of arbitrage. Calibration means that the model parameters are optimized such that the model prices match the prices of derivatives traded in the market. For equity models, the prices of vanilla call and put options for different maturities and strikes can be observed and are used for the calibration. For credit derivatives, the spreads of credit default swaps (CDS) are a common market instrument used for calibration. This calibration can be computationally demanding, especially if the model depends on large number of model parameters. The same set of simple financial instruments (e.g. vanilla options, CDS instruments) is priced many times for different model parameters until the optimal parameters are found. Once this calibration is performed, the price of the exotic product can be computed. See for example [113] for the calibration of different equity models and [21] for the calibration of interest rate and credit models.

The complexity of the model calibration is thus closely related to the complexity of the model itself. For equities, the famous model of [15] has only one parameter, the volatility, that cannot directly be observed in the market. The volatility is therefore computed implicitly from option prices observed in the market, this is called the *Black-Scholes implied volatility*. In contrast to the model assumptions, different options on the same underlying have different implied volatilities, an effect known as smile. This means that the Black-Scholes model can only be calibrated to one option but not to a whole

surface, i.e all options on the same underlying that differ only in strike and maturity. Over the years, different models have been developed that tackle this problem. This includes local volatility models such as the constant elasticity of variance model presented in [114] or Dupire's local volatility model presented in [39]. Other examples are the stochastic volatility model of [69] and models with jumps such as [94], Lévy models such as the CGMY model of [29] or a combination of jumps and stochastic volatility such as the model of [8].

This model price calculated under the risk-neutral measure is however not the price for which an exotic product can be sold. Additional valuation adjustments have to be added (or subtracted) from the model price to obtain the final market price. For a OTC (over-the-counter) trade between two counterpartys, credit risk plays an important role and the model price is reduced by a bivariate counterparty credit valuation adjustment (CVA), see [64] for an overview on counterparty credit risk. For example, an option from a bank with a good credit score is more valuable than the same option from a bank with a bad credit score. The CVA accounts for this difference in prices. It is usually computed by simulating the value of the derivative over the options lifetime and multiplying the calculated exposure by the probability that the counterparty defaults. Calculating this exposure is computationally demanding since it requires the valuation of the option at different time points in the future. Similar calculation occur in the computation of other valuation adjustment such as margin valuation adjustment (MVA) or funding valuation adjustment (FVA). We refer to [63] for an overview on valuation adjustments.

In risk management, one wants quantify the market risk and counterparty credit risk on the level of a portfolio or trading book at a future time point or over a certain time horizon. In this context, the focus shifts from calibration to market instruments to a scenario simulation based approach. Frequently, a portfolio of derivatives is evaluated for a large number of risk scenarios, simulated under the real-world (or physical) measure $\mathbb{P}$. This means that the same type of option has to be repeatedly priced for different input values. Different risk metrics can then be extracted from the empirical distribution of the portfolio values. Here, one is typically not only interested in the mean of the distribution but also in tail measures. The size of the portfolio as well as the number of risk factors make risk management computationally very demanding. We refer to [1] for an overview on option pricing from a (market) risk management perspective. Recent regulatory requirements such as the fundamental review of the trading book (FRTB) put additional pressure on banks and demand a higher number of calculations. As stated in [2]: "Efficiencies are always welcome, but especially now in view of the significantly higher computational capacity and storage needs of FRTB (such as a tenfold increase in the number of P&L vector calculations over an entire portfolio, and the demands of

desk-level reporting)". In order to cope with this demand, banks need either to increase the computational power or employ more efficient numerical pricers.

Both perspectives, trading and risk management, justify the need for efficient numerical pricing methods. Starting with one-dimensional European options different numerical pricing methods have been developed. They can be roughly divided into (binomial) tree methods, PDE methods, Fourier type methods and simulation based methods. See for example [115], [136] and [101] for more details on these methods. Many of the developed methods have been extended or modified in order to price path-dependent options or options on more than one asset. Each of the different classes has its merits and demerits and their is clearly no method that always outperform the others. For most of the method holds that they focus on calculating one price of a specific product in a specific class of models rather than on repeated calls of the same numerical pricing routine.

For calculating one price of univariate option, the differences in runtimes between different methods are usually small. However, if the same option is priced many times for varying starting values the efficiency of the methods differs a lot. For example, if the expected exposure of a trade is calculated the same product is usually priced for 50,000 or more simulation paths at up to 50 time steps yielding 2.5 million calls of the same pricing routine. Here, it is desirable that the pricing method is able to deliver accurate option prices on an interval of input factors without re-running all computations. Function approximation methods can help to tackle this problem by providing an approximation of the option price as a function of its input factors.

In this thesis, we will introduce a new pricing approach for path-dependent options based on a function approximation method, the Chebyshev polynomial interpolation. We will show that the new method is well-suited for the calculation of an option's exposure since it delivers an approximation of the option price and its sensitivities over the option's lifetime. We will then extend the presented pricing method to multivariate options.

Multivariate early-exercise options

A large class of option pricing problems are essentially multidimensional pricing problems. This is the case for options on multiple underylings such as basket, spread or rainbow options as well as options that depend on more than one risk factor. Examples are equity basket options on several stocks, stochastic volatility and stochastic interest rates in stock price models, credit derivatives with interest rate and default risk or foreign exchange (FX) options with a stochastic FX rate and stochastic interest rates in both currencies. While some basket option pricing problems can be truly high-dimensional, the majority of the problems is in up to five dimensions. Typically the different risk fac-

tors are modelled using either a (geometric) Brownian motion or an Ornstein-Uhlenbeck process and thus, they are conditionally normally or log-normally distributed. For many products it is important that one accounts for correlation between the different risk factors and if the option has an additional early-exercise feature, the pricing of such options becomes challenging. Especially, if one is interested in the expected exposure of multivariate options and the same product has to be priced for a high number of simulated scenarios. As an example, [119] show the importance of stochastic volatility and stochastic interest rates for the pricing and risk management of FX derivatives and they obtain a model with four risk factors.

Most standard pricing methods such as PDE methods or Fourier methods can be extended into multivariate dimensions. A straightforward extension leads however to an exponential grows of the number of nodal values, often referred to as the *curse of dimensionality*. For example, doubling the number of grid points from 50 to 100 is often still easy to handle, in three dimensions the step from 50^3 to 100^3 is an increase of factor eight and leads to one million points which makes a method often infeasible. Obtaining accurate results can therefore require long runtimes, especially if the option is early-exercisable. Thus, it is crucial to ensure a fast convergence and a low number of nodal points per dimension. In contrast, simulation based methods do not suffer from the curse of dimensionality. They are usually simple and can provide fast results if accuracy is not critical. However, they come with a slow convergence and achieving a higher accuracy requires an infeasible number of simulations. Additionally, they introduce a simulation noise and this makes the calculation of sensitivities unstable. These drawbacks are only justifiable if the dimension is very high and no other method is available.

In this thesis, we want to close this gap and provide a new pricing method for multivariate early-exercise options. Exploring the beneficial properties of Chebyshev interpolation and combining it with a numerical smoothing approach leads to fast error decay and an efficient pricing method. We will focus on two main pricing problems in different asset classes. First, we consider the pricing of credit derivatives with an early-exercise feature in a two-factor interest rate/credit model driven by two correlated Ornstein-Uhlenbeck processes. Here, the main computational challenge is the calibration of the two-factor model to credit spreads obtained form CDS instruments. Due to the correlation, the pricing of such a CDS is a bivariate pricing problem itself and is significantly more challenging than a calibration in a one-dimensional short rate model. The second problem that we consider is the pricing of basket options on up to five assets in a multivariate Black-Scholes model assuming a positive correlation between the assets. Here, we combine our pricing method with a smoothing concept in order to reduce the computational complexity.

Function approximation methods in finance

Function approximation methods are a large class of approximation methods that approximate a function $f : \mathbb{R}^d \to \mathbb{R}$ by a weighted sum of basis functions

$$f(\boldsymbol{x}) \approx \sum\nolimits_j w_j \varphi_j(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$$

for basis functions $\varphi_j : \mathcal{X} \to \mathbb{R}$. We write $\boldsymbol{x}$ to indicate that d might be bigger than 1 and we write x if $d = 1$. The function f might be explicitly or implicitly given and the approximation domain $\mathcal{X}$ can be compact or unbounded. This framework is very general and includes a simple polynomial interpolation of an univariate function as well as a high-dimensional deep neural network. Closely related to the approximation of a function is the numerical integration of a function f. We can approximate the integral of f as

$$\int_{\mathbb{R}^d} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \approx \sum\nolimits_j w_j \int_{\mathcal{X}} \varphi_j(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}.$$

If the integral of the basis function are known or can be efficiently calculated, this can be an efficient quadrature approach. Similarly, we can use function approximation methods to compute integrals with respect to a probability density function.

In this thesis, we exploit Chebyshev polynomial interpolation in order to obtain efficient methods for the pricing of path-dependent options. The use of polynomials and the Chebyshev series is not new and has already been investigated in different papers. In the following, we present a short literature overview.

[88] propose to use polynomial basis function in a least-square regression in their pricing algorithm for American option. The fitted polynomials are then used to approximate the optimal early-exercise policy. [124] compares the performance of different families of polynomials that can be used in the algorithm. [43] proposed a new pricing method for European options based on a Fourier cosine expansion and extended the approach to path-dependent options in [44]. More precisely, the density of the underlying is approximated by its (truncated) Fourier cosine expansion and the series coefficients can be approximated using the characteristic function. The smoothness of the density function yields a fast convergence of the series coefficietns. It remains to compute integrals of the payoff funciton times the cosine basis function of the series. For standard option payoffs, these integrals are available in closed form. As pointed out in [101], the cosine series is equivalent to a Chebyshev series under the correct variable transformation. See also [46] for a discussion on the connection between a Chebyshev series and a Fourier series. Overall, it is the pricing method that is the closest to our new approach.

Related to the COS method of [43] is the pricing method of [103] that is based on Chebyshev quadrature and an efficient splitting of the domain. This paper uses algorithm of the *chebfun* Matlab toolbox, available online at *www.chebfun.org*. The same toolbox has also been exploited for option pricing in [31], who propose to use the Legendre series for the pricing and hedging of options. Moreover, [28] and [98] develop an efficient numerical PDE solver using Chebyshev polynomials for the pricing PDE of European and early-exercise options.

A general framework for the approximation of option prices using Chebyshev polynomials has been introduced in [49]. They investigate the smoothness of option prices as a function of their parameters and they show that for a large class of models and payoffs, this function is infinitely often continuously differentiable and has an extension into the complex domain. This theoretical investigation motivates them to interpolate option prices in their parameters to speed-up recurrent pricing tasks. The analyticity of the option price ensures an exponential convergence for Chebyshev polynomial interpolation. The proposed approach has been extended to high-dimensional problems in [57]. We present this static Chebyshev approach in more details in Section 2.3 and discuss how it is related to our new pricing method for path-dependent options.

Outline of the thesis

The thesis is divided into a preliminary chapter about Chebyshev polynomial interpolation and its application in finance, a chapter about a new dynamic pricing algorithm using Chebyshev interpolation and two chapters about the application of this algorithms for credit exposure calculation and to multivariate option pricing. We visualized the structure in Figure 1.1. In the following, we describe the chapters in more detail and highlight the main contributions.

Chapter 2 is the preliminary chapter of Chebyshev interpolation and we explain why we have specifically chosen Chebyshev interpolation as function approximation technique. We provide theoretical convergence results, discuss the smoothness of option prices and present promising numerical results. More precisely, in Section 2.1 we introduce the univariate Chebyshev polynomial interpolation and present the relevant convergence results. In Section 2.2, we discuss several possibilities for a multivariate Chebyshev interpolation, including convergence results and possible ways to tackle the curse of dimensionality. Section 2.3 deals with the application of Chebyshev interpolation in finance. We give a brief introduction to the mathematical theory behind option pricing and summarize the most import types of asset price models. We analyse the smoothness of option prices as a function of different parameters and show that this function is often analytic, i.e. it is infinitely often continuously differentiable and can be locally written

as a power series. Then we introduce the Chebyshev interpolation for parametric option pricing idea proposed by [49] in order to speed-up recurrent pricing tasks. This method results in a two steps approach consisting of an offline-phase or pre-computation step where prices are computed at the set of nodal points and an online phase where the Chebyshev interpolant is evaluated instead of the original pricer. We refer to this method also as the static Chebyshev method.

We investigate a particular application of the Chebyshev interpolation in finance in Section 2.4. There, we introduce a new numerical approximation method for the Black-Scholes implied volatility. The presented method has been published in a joint paper *"The Chebyshev method for the implied volatility"* ([56]) with co-authors Kathrin Glau, Paul Herold and Dilip B. Madan. The implied volatility is the inverse of the call price function and itself also an analytic. It is one of the most important quantities in finance and needs to be computed frequently for a large set of different input parameters. Since there is no closed form solution, it is an ideal application for a (bivariate) Chebyshev interpolation. We select a suitable interpolation domain based on market data and interpolate on this domain using a domain splitting and appropriate transformations. The resulting method is tested against two benchmark methods. We show that our method is able to cover all options observed in the market and improves the efficiency of state-of-the art benchmark methods.

Chapter 3 is the core of this thesis and contains the introduction of our new dynamic pricing method for path-dependent options using Chebyshev interpolation. This chapter is based on the paper *"A new approach for American option pricing: The dynamic Chebyshev method"* ([59]) published together with co-authors Kathrin Glau and Mirco Mahlstedt. The main idea of the dynamic Chebyshev method has also been presented in the PhD thesis *"Complexity Reduction for Option Pricing"* of Mirco Mahlstedt. The proposed method is a novel pricing approach in a dynamic programming framework that includes the pricing of early-exercise options as well as discretely monitored barrier options. We provide a theoretical error analysis of the new method and discuss several aspects regarding the implementation. An empirical convergence analysis is conducted as well as a first performance comparison to a benchmark approach. We present the approach in a fairly general set-up and then tailor it to different applications in Chapter 4 and Chapter 5.

More precisely, in Chapter 4, we investigate the numerical calculation of credit exposures for CVA and for counterparty credit risk management. The chapter is based on the paper *"Speed-up credit exposure calculations for pricing and risk management"* ([61]) with Kathrin Glau and Ricardo Pachon and on a previous version of the paper named *"Fast Calculation of Credit Exposures for Barrier and Bermudan options us-*

ing Chebyshev interpolation" ([60]). The computational intensive problem fits well into our dynamic framework and we can apply our new dynamic pricing method. We show that our new dynamic Chebyshev method delivers a closed form approximation over the option's lifetime. The simple polynomial structure of the approximation allows for an efficient evaluation of credit exposure. We provide different numerical examples for equity options (European, early-exercise, barrier) and a Bermudan interest rate swaption. Our investigation shows that the new approach combines the accuracy of a full re-evaluation with the speed of a simple least-squares Monte Carlo approximation. We conclude the chapter with a discussion of the economic consequences of using an accurate numerical pricing routine instead of a simple approximation.

In Chapter 5 we consider the pricing of multivariate early-exercise options where the underlying is conditionally normally distributed. We start with a general multivariate dynamic pricing algorithm and provide a first numerical example. Then, we investigate in detail the pricing of a callable bond (bond with embedded early-exercise call option) in a two-factor model with stochastic interest rates and stochastic default intensity. We explain how our pricing method can be used for the calibration of the two-factor model to credit spreads. An extensive numerical investigation shows the efficiency and the stability of the resulting calibration and pricing approach. In the last part of the chapter, we consider a basket option in a multivariate Black-Scholes model. We present the smoothing concept for European basket options of [11]. Empirically, we show that the dynamic Chebyshev method can be turned into an efficient quadrature method for basket options using this smoothing. Then we extend this approach to early-exercise options and we propose a new type of numerical smoothing. We conclude the chapter with a numerical convergence analysis of the resulting pricing method. We observe a fast error decay and an efficiency gain compared to a least-squares Monte Carlo approach.

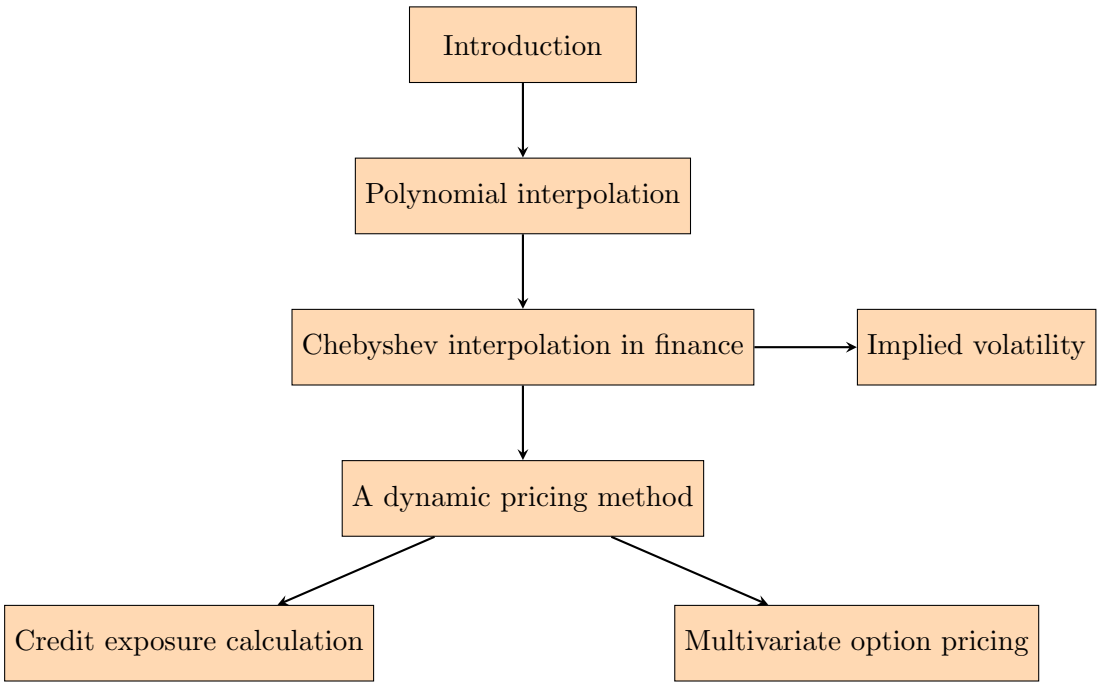

Figure 1.1: Structure of the thesis.

Chapter 2

Chebyshev interpolation in finance

In this chapter, we discuss Chebyshev interpolation as a form of function approximation and its application in finance. We start with the univariate Chebyshev interpolation and discuss how it can be extended to higher dimensions. Then we give a brief overview on parametric problems in option pricing and investigate the smoothness of pricing functions. We discuss the application of Chebyshev interpolation in this context and provide a larger numerical example, the computation of the Black-Scholes implied volatility using a bivariate Chebyshev interpolation.

2.1 Chebyshev polynomial interpolation

This section deals with the univariate Chebyshev polynomial interpolation. We introduce the important definitions and concepts regarding polynomial interpolation in Chebyshev points that are relevant for the remaining chapters of this thesis. The first aim of this section is to be the reference point whenever one of the basic definitions in polynomial interpolation is needed. The second, more general aim of this section (and essentially this chapter) is to provide a justification why we have specifically chosen Chebyshev interpolation as a function approximation technique.

2.1.1 Polynomial interpolation

The main idea behind polynomial interpolation is the approximation of a function using polynomials that is exact at a set of nodal points. Let $x_0, \dots, x_n$ be a set of $n+1$ distinct points in the interval $[-1, 1]$ and let $f_j := f(x_j)$, $j = 0, \dots, n$ be the values of a function

$f : [-1, 1] \to \mathbb{R}$. Then there exists an unique polynomial p_n of degree n that interpolates these function values, i.e. $p_n(x_j) = f_j$ for all $j = 0, \ldots, n$.

The interpolant p_n of f can be written as a linear combination of basis functions that span the vector space of all polynomials up to degree n. This means p_n can be written as

$$p_n(x) = \sum_{j=0}^{n} w_j \phi_j(x) \qquad \text{for} \qquad x \in [-1, 1] \tag{2.1}$$

for weights w_j and polynomial basis functions $\phi_j : [-1, 1] \to \mathbb{R}$. Different choices of basis functions are possible and we will later see that some choices are better suited than others. A simple and very convenient way to express the polynomial interpolant of a function f is via the Lagrange interpolation formula. For any set of distinct points $x_0, \ldots, x_n$ we can write

$$p_n(x) = \sum_{j=0}^{n} f_j l_j(x) \qquad \text{with} \qquad l_j(x) = \frac{\prod_{k \neq j}(x - x_k)}{\prod_{k \neq j}(x_j - x_k)}. \tag{2.2}$$

The basis function l_j is the j-th Lagrange polynomial with $l_j(x_k) = 1$ if $j = k$ and 0 otherwise, see Chapter 5 in [129]. This formula is a special case of (2.1) and it is of particular use because it is the same for any set of nodal points and the weights are exactly the function values $f_j = f(x_j)$. We will later see that this is the right form to investigate the stability of polynomial interpolations.

The crucial question is if the polynomial interpolant p_n is a good approximation for the function f and if the approximation error $\|f - p_n\|_\infty$ in the maximum norm converges for $n \to \infty$. Note that the convergence of the interpolation depends only on the set of (nodal) points $x_0, \ldots, x_n$ and is independent of the choice of basis functions ϕ_j. For computational purposes however, the right choice of basis functions is critical since it influences the speed and the stability of the evaluation of p_n.

First, we would like to know which functions can be approximated using polynomials. The following well-known theorem of Karl Weierstraß shows that any continuous function f on $[-1, 1]$ can be approximated by polynomials with arbitrary accuracy.

Theorem 1 (Weierstrass approximation theorem)**.** *Let $f : [-1, 1] \to \mathbb{R}$ be a continuous function and let $\varepsilon > 0$ be arbitrary. Then there exists a polynomial p such that*

$$\|f - p\|_\infty := \max_{x \in [-1,1]} |f(x) - p(x)| < \varepsilon. \tag{2.3}$$

Proof. See Theorem 6.1 in [129]. □

This theorems proves that any continuous function can be approximated by polynomials but it does not answer the question how to find such a polynomial. Ideally, we would like to find a sequence or a scheme of nodal points $x_0, \dots, x_n$ such that the interpolant in these points p_n of a function f converges towards f for all continuous functions. Unfortunately, as pointed out in [128], *"no polynomial interpolation scheme, no matter how the points are distributed, will converge for all such functions"*.

Even for smooth functions, polynomial interpolation does not necessarily converge. A straightforward approach for choosing interpolation points are equidistant points $x_k = -1 + 2k/n,\ k = 0, \dots, n$. The famous example of [110] shows that polynomial interpolation of a function f can be unstable even though f is an analytic function. We consider the Runge function

$$f(x) = \frac{1}{1 + 25x^2} \quad \text{for} \quad x \in [-1, 1], \tag{2.4}$$

which is bounded by 1 and strictly positive. Figure 2.1 shows the function and its polynomial interpolation in 11 (left plot) and 23 (right plot) equidistant points. We observe that close to the end points -1 and 1 the interpolation is not stable and it does not converge. In the left plot we observe that the maximal value of the interpolant is 2 and in the right plot it is already 120.

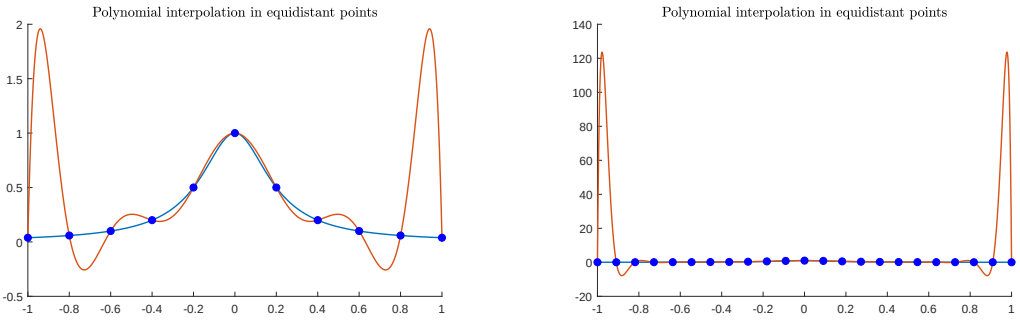

Figure 2.1: Polynomial interpolation of the Runge function on $[-1, 1]$ using 11 (left plot) and 23 (right plot) equidistant points.

From these results, one could naively conclude that polynomial interpolation is not a good idea in general. See the discussions around this topic in [128] and [129]. This is however not the case, and there are sets of nodal points which yield convergence for a large class of continuous functions. One set that has particularly promising properties are the Chebyshev points which guarantee convergence for all Lipschitz-continuous functions. We will introduce these points in the next section. Before we do so, we briefly

discuss two fundamental concepts in polynomial approximation theory: Best polynomial approximation and the Lebesgue constant.

Best approximation is for a given continuous function the polynomial of a certain degree that has the minimal approximation error in the maximum norm. In the literature, it is therefore also called *minimax* approximation. We define the vector space of all continuous functions $f : [-1, 1] \to \mathbb{R}$ as $C([-1, 1])$ and the vector space of all polynomials of degree $n \in \mathbb{N}$ as

$$\mathcal{P}_n := \big\{p \in C([-1, 1]) : \ p \text{ is polynomial of degree n}\big\}. \tag{2.5}$$

This allows us to define the best approximation in the vector space $\mathcal{P}_n$ equipped with the maximum norm.

Definition 1 (Best approximation)**.** *Let $f \in \mathcal{C}([-1, 1])$. The best polynomial approximation of degree n of f is defined as the polynomial $p^\star \in \mathcal{P}_n$ such that*

$$\|f - p^\star\|_\infty \leq \|f - p\|_\infty \tag{2.6}$$

for all $p \in \mathcal{P}_n$.

As shown in Theorem 10.1 of [129], this best approximation $p^\star$ is unique and the approximation error oscillates. In the 1930s, the Russian mathematician Evgeny Yakovlevich Remez proposed an algorithm for computing such a best approximation, the Remez algorithm. More recently, [105] introduced an optimized version of the algorithm. Theoretically, this is a very interesting concept, however, in many examples polynomial interpolation in Chebyshev points is almost as good as the best approximation and considerably more practical. For instance, for an analytic function the Chebyshev interpolation and the best approximation have the same error bound up to a factor of 2. See [129] for more details on this comparison.

In the previous Runge example we have seen that the maximal value of the polynomial interpolant p_n can be much higher than the maximum of f and might increase in n. A suitable polynomial interpolation should ensure that the norm of $\|p_n\|_\infty$ does not explode in relation to $\|f\|_\infty$. For a given set of nodal points $x_0, \ldots, x_n$, the polynomial interpolation in these points is a linear operator $I_n : C([-1, 1]) \to \mathcal{P}_n$ with $I_n(f) = p_n$. We are interested in the operator norm of I_n which is in the context of polynomial interpolation the so-called Lebesgue constant Λ. Consider the Lagrange form (2.2), then we obtain

$$\|I_n(f)\|_\infty = \|p_n\|_\infty = \max_{x\in[-1,1]} \Big| \sum_{j=0}^{n} f(x_j) l_j(x) \Big| \leq \|f\|_\infty \max_{x\in[-1,1]} \sum_{j=0}^{n} |l_j(x)|$$

and the Lebesgue constant is defined as

$$\Lambda_n := \max_{x\in[-1,1]} \sum_{j=0}^{n} |l_j(x)|. \tag{2.7}$$

This means that $\|p_n\|_\infty \leq \Lambda_n \|f\|_\infty$ and the Lebesgue constant measures the stability of the interpolation. Moreover, the Lebesgue constant can be used to characterize interpolation points that are almost optimal. The following theorem uses the Lebesgue constant to measure which polynomial interpolations are close to the best approximation. These interpolations are called near-best approximations.

Theorem 2 (Near-best approximation)**.** *Let $f \in C([-1,1])$ and let Λ_n be the Lebesgue constant of a polynomial interpolation operator I_n. For $p_n = I_n(f)$ holds*

$$\|f - p_n\|_\infty \leq (1 + \Lambda_n)\|f - p^\star\|_\infty, \tag{2.8}$$

where $p^\star \in \mathcal{P}_n$ is the best polynomial approximation of f.

Proof. See Theorem 15.1 in [129]. □

The smaller the Lebesgue constant, the closer is a polynomial interpolation to the best approximation for a given degree n. The Lebesgue constant can be small but it will grow at least logarithmically in n for any set of interpolation points. More precisely, [22] shows that

$$\Lambda_n > \frac{2}{\pi}\log(n+1) + 0.52125 \tag{2.9}$$

for the Lebesgue constant of any polynomial interpolation. For polynomial interpolation in equidistant points we obtain from [129]:

$$\Lambda_n > \frac{2^{n-2}}{n^2} \quad \text{and} \quad \Lambda_n \sim \frac{2^{n+1}}{en\log(n)}, \quad \text{for } n \to \infty. \tag{2.10}$$

We observe that the Lebesgue constant for equidistant points grows exponentially in n. This is the theoretical explanation of the Runge example and shows why equidistant points are not a suitable set of interpolation points. In contrast, in the next section we will see that the Lebesgue constant of Chebyshev points is close to the lower bound (2.9).

2.1.2 Chebyshev points and Chebyshev polynomials

In this section, we introduce the Chebyshev points as well as the Chebyshev polynomials and we discuss their promising properties.

We follow Chapter 1.5 of [46] and consider the following problem. Find the polynomial $p_n \in \mathcal{P}_n$ with leading coefficient 1 such that $\|p_n\|_\infty$ is minimized. This is essentially a *minimax* problem as studied for the best approximation (2.6) with $f = 0$. The solution to this problem will oscillate between the maximum M and the minimum $-M$ at $n+1$ points. As stated in [46], the trigonometric functions $\cos(\theta)$, $\sin(\theta)$ fulfil such criteria and $cos(\theta) \mapsto \cos(n\theta)$ is a polynomial of degree n. This motivates the following definition of the Chebyshev polynomials.

Definition 2 (Chebyshev polynomials)**.** *The function* $T_n : [-1, 1] \to \mathbb{R}$ *with*

$$T_n(x) = \cos(n\theta), \quad for \quad \cos(\theta) = x \tag{2.11}$$

is the n-th Chebyshev polynomial.

The letter T comes from the different transliterations of the name of the Russian mathematician Pafnuty Chebyshev such as Tchebychev (in French) and Tschebyschow (in German). The Chebyshev polynomials can also be defined via recursion as

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad \text{for} \quad n \geq 1, \tag{2.12}$$

with $T_0(x) = 1$ and $T_1(x) = x$. The equivalence of the two definitions follows directly from the cosine identity

$$\cos((n+1)\theta) + cos((n-1)\theta) = 2\cos(\theta)\cos(n\theta)$$

and $\cos(\theta) = x$. From the definition (2.12) we immediatly see that $T_n(x)$ is a polynomial of degree n in x with leading coefficient 2^{n-1} for $n \geq 1$. From [46] follows that $p_n(x) = 2^{-(n-1)}T_n(x)$ is the *minimax*-polynomial for degree n with leading coefficient 1 that minimizes $\|p_n\|_\infty$. The extrema of T_n and thus p_n are the $(n+1)$ *Chebyshev points* (of the second kind) and are given by

$$x_k = \cos(k\pi/n), \quad k = 0, \ldots, n. \tag{2.13}$$

Closely related are the roots of T_n, the Chebyshev points of the first kind, given by

$$x_k = \cos\Big(\frac{2k+1}{2n}\pi\Big), \quad k = 0, \ldots, n-1.$$

In this thesis, we will only use the points defined in (2.13) and simply refer to them as *Chebyshev points*. Figure 2.2 shows the Chebyshev polynomials $T_1, \ldots, T_n$ for $n = 6$ with the corresponding Chebyshev points as the extrema. Figure 2.3 shows the $(n+1)$ points for different values of n. The Chebyshev points are not equidistantly distributed but clustered at the end points -1 and 1. Due to this clustering the polynomial interpolation

in the Chebyshev points becomes stable. Moreover, we observe that the points are not nested, however, the $(n+1)$ points $x_0, \ldots, x_n$ are included in the $(2n+1)$ points $x_0, \ldots, x_{2n}$. In the remaining part of the section we discuss different properties of the Chebyshev points and the Chebyshev polynomials.

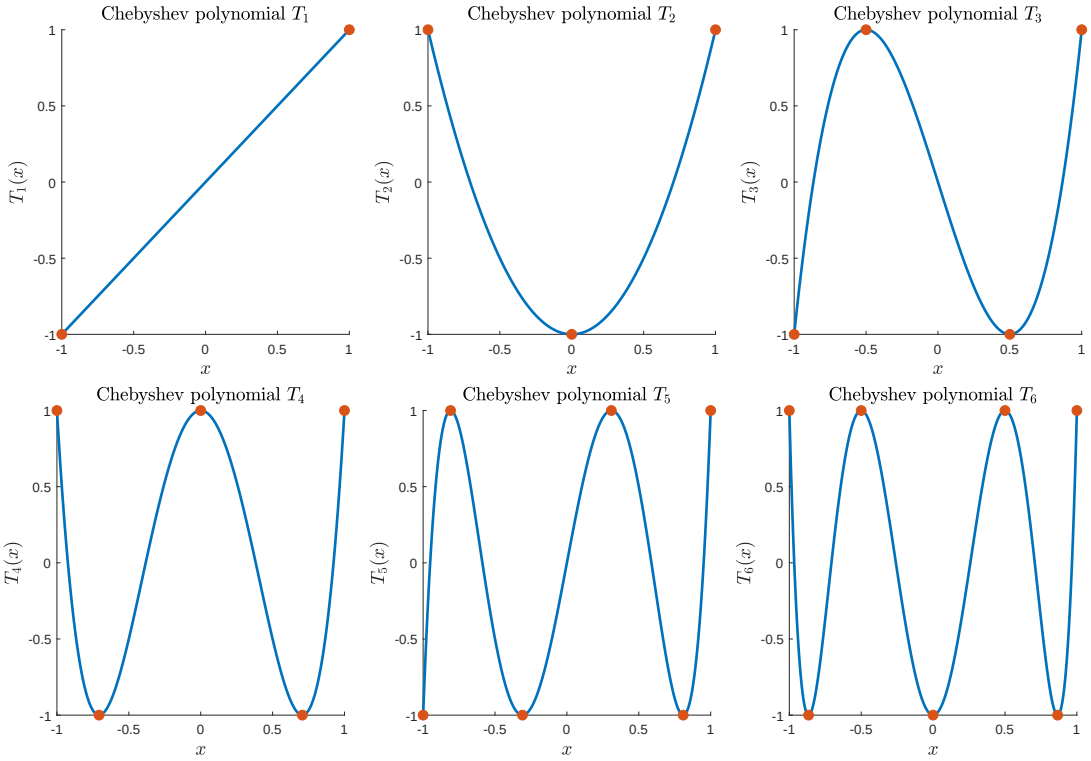

Figure 2.2: Chebyshev polynomials $T_1, \ldots, T_6$ and the corresponding Chebyshev points.

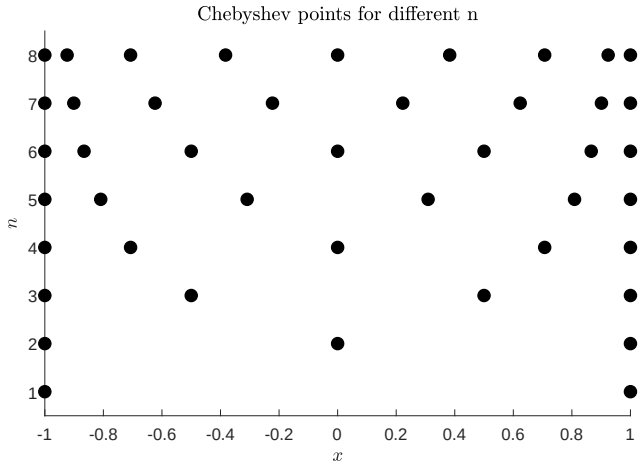

Figure 2.3: Chebyshev points $x_k = \cos(k\pi/n)$, $k = 0, \ldots, n$ for different levels of $n = 1, \ldots, 8$.

Orthogonality of Chebyshev polynomials

The Chebyshev polynomials $T_0, \ldots, T_n$ form an orthogonal basis of the vector space $\mathcal{P}_n$ with respect to the weight function

$$w(x) = \frac{1}{\sqrt{1-x^2}} \quad \text{for} \quad x \in [-1, 1]. \tag{2.14}$$

Consider the inner product associated with this weight function

$$(f, g)_w := \int_{-1}^{1} w(x) f(x) g(x) \mathrm{d}x \quad \text{for} \quad f, g \in C([-1, 1]).$$

For the Chebyshev polynomials holds

$$(T_j, T_k)_w = 0 \quad \text{if} \quad j \neq k,$$

see [129]. An appropriate scaling then turns the Chebyshev polynomials into an orthonormal basis. The Chebyshev polynomials are a special case of the Jacobi polynomials $P_n^{(\alpha,\beta)}$ that are orthogonal with respect to the general weight function $w(x) = (1-x)^\alpha (1+x)^\beta$. Orthogonal polynomials and their properties have been extensively studied in the literature, see e.g. [132] for an earlier reference and [52] for a more recent reference. Note that in the literature sometimes all orthogonal polynomials are called Chebyshev polynomials.

Chebyshev series and Chebyshev interpolation

The question arises under which condition we can write a function $f \in C([-1, 1])$ as an (infinite) sum of Chebyshev polynomials. From [129] we obtain the following result.

Theorem 3 (Chebyshev series)**.** *Let $f \in C([-1, 1])$ be Lipschitz continuous. Then f has a unique representation as an absolutely and uniformly convergent Chebyshev series*

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x) \quad \textit{with} \quad a_k = \frac{2}{\pi} \int_{-1}^{1} \frac{f(x) T_k(x)}{\sqrt{1-x^2}} dx \tag{2.15}$$

and the coefficient a_0 is multiplied by $1/2$.

Proof. See Theorem 3.1 of [129]. □

We recall that a function $f \in C([-1, 1])$ is Lipschitz continuous if there exists a constant $L > 0$ such that

$$|f(x) - f(y)| \leq L|x - y| \quad \forall\, x, y \in [-1, 1].$$

The theorem is based on the orthogonality of the Chebyshev polynomials and can be motivated in the following way. We multiply f by the polynomial T_k and integrate over $[-1,1]$ using the weight function as defined in (2.14). Then we obtain

$$\int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}\mathrm{d}x = \sum_{j=0}^{\infty} a_j \int_{-1}^{1} \frac{T_j(x)T_k(x)}{\sqrt{1-x^2}}\mathrm{d}x = \sum_{j=0}^{\infty} a_j (T_j, T_k)_w = a_k (T_k, T_k)_w = a_k \frac{\pi}{2}.$$

Since the Chebyshev series is absolutely convergent, $|a_k|$ converges towards 0. The truncation of the series is then a promising candidate for a polynomial approximation (of degree n) of f, i.e.

$$f(x) \approx f_n(x) = \sum_{k=0}^{n} a_k T_k(x) \quad x \in [-1,1].$$

In fact, this expansion is the continuous least-squares approximation of the function f in a weighted L^2 norm. The following least-squares measure is minimized by f_n

$$S = \int_{-1}^{1} w(x)(f(x) - p_n(x))^2 \mathrm{d}x \quad \text{for} \quad p_n \in \mathcal{P}_n$$

see equation (42) and (43) in [46]. This property is again based on the orthogonality of Chebyshev polynomials. In a similar way we can also write the polynomial interpolation of a (Lipschitz) continuous function f as a weighted sum of Chebyshev polynomials.

Proposition 1 (Chebyshev interpolation)**.** *Let $f \in C([-1,1])$ be Lipschitz continuous. The polynomial interpolation of degree n of f in the $(n+1)$ Chebyshev points $x_k = \cos(k\pi/n)$, $k = 0, \ldots, n$ can be written as*

$$p_n(x) = \sum_{j=0}^{n}{}'' c_j T_j(x) \quad \textit{with} \quad c_j = \frac{2}{n} \sum_{k=0}^{n}{}'' f(x_k) T_j(x_k) \tag{2.16}$$

where the two primes indicate that the first and the last summand is multiplied by $1/2$.

Proof. See the derivation of equation (51) in [46]. □

For notational convenience we sometimes shift the two primes of the interpolant to the coefficients and obtain the slightly modified formula

$$p_n(x) = \sum_{j=0}^{n} c_j T_j(x) \quad \text{with} \quad c_j = \frac{2^{\mathbb{1}_{0<j<n}}}{n} \sum_{k=0}^{n}{}'' f(x_k) T_j(x_k). \tag{2.17}$$

Using only the first interpolation coefficients in (2.16) leads to a good fit in a discrete

least-squares metric. More precisely, the expansion

$$p_n(x) = \sum_{j=0}^{n}{}'' c_j T_j(x) \quad \text{with} \quad c_j = \frac{2}{n} \sum_{k=0}^{N}{}'' f(x_k) T_j(x_k) \tag{2.18}$$

for $n \leq N$ minimizes the discrete L^2 measure

$$S = \sum_{k=0}^{N}{}'' (f(x_k) - p(x_k))^2 \quad \text{for} \quad p \in \mathcal{P}_n.$$

See [46] for more details. We have seen that the Chebyshev series is connected to a continuous least-squares fit and the Chebyshev interpolation to a discrete least-squares fit. Thereore, the coefficients of the Chebyshev interpolant can be seen as a discretized version of the coefficients of the Chebyshev series. Moreover, each c_j can be written as a series of coefficients a_k, see [129].

Lebesgue constant and stability

For the Chebyshev points the Lebesgue constant as defined in (2.7) is bounded by

$$\Lambda_n \leq \frac{2}{\pi} \log(n+1) + 1 \quad \text{and} \quad \Lambda_n \sim \frac{2}{\pi} \log(n), \ n \to \infty, \tag{2.19}$$

see Theorem 15.2 in [129]. We directly see that the Lebesgue constant grows significantly slower than the Lebesgue constant of the polynomial interpolation in equidistant points (2.10). In fact, comparison to the lower bound (2.9) reveals that the Chebyshev points are almost optimal interpolation nodes. The only difference compared to the lower bound is the added constant that is independent of the number of points n.

We consider again the Runge example and interpolate the Runge function (2.4) in the Chebyshev nodes. Figure 2.4 shows the resulting interpolant for 11 (left plot) and 23 (right plot) points. Here, the interpolation does converge and there are no instabilities as for the interpolation in equidistant points, compare Figure 2.1.

Evaluation of Chebyshev interpolations

From a computational point of view it is essential that the interpolation (2.16) can be evaluated efficiently. Here, efficiency means that the evaluation of the interpolant can be implemented in a way that is fast and at the same time numerically stable. We illustrate this problem with a small example of a polynomial of degree 4

$$p_4(x) = a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0.$$

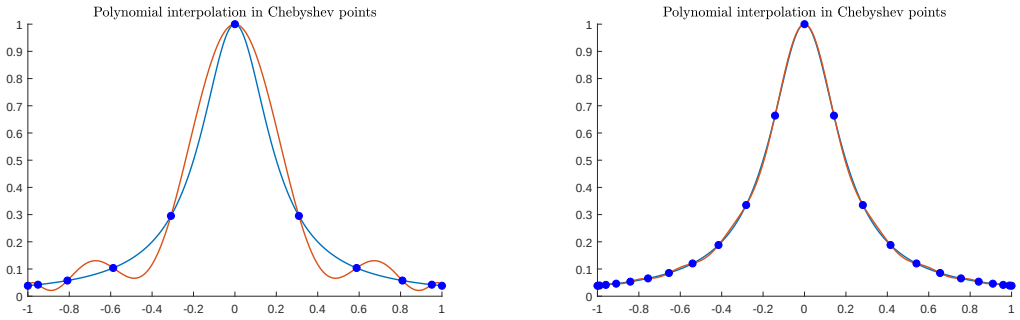

Figure 2.4: Polynomial interpolation of the Runge function on $[-1, 1]$ using 11 (left plot) and 23 (right plot) Chebyshev points.

A straightforward evaluation would require $2n-1$ multiplications, i.e. $n-1$ to compute $x^2, \ldots, x^n$ and n multiplications with the coefficients. For $n = 4$, we have 7 multiplications. If we write the same polynomial slightly differently

$$p_4(x) = a_0 + x\big(a_1 + x\big(a_2 + x(a_3 + xa_4)\big)\big)$$

we only need 4 multiplications. In its general form for a polynomial of degree n this method is called Horner's scheme. For the Chebyshev interpolation we do not use the monomials $x, x^2, \ldots, x^n$ and a direct application of Horner's scheme is not possible. A generalization of this idea is provided in [32] for polynomials ϕ_n that satisfy the recurrence relation

$$\phi_{n+1}(x) + \alpha_n \phi_n(x) + \beta_n \phi_{n-1}(x) = 0.$$

This is the case for the Chebyshev polynomials, see equation (2.12). From [46] we obtain the recursion

$$\begin{aligned} &b_k(x) = 2xb_{k+1}(x) - b_{k+2}(x) + c_j \quad k \leq n \\ &\text{with starting values} \quad b_{n+1}(x) = b_{n+2}(x) = 0 \end{aligned}$$

and thus

$$\sum_{j=0}^{n} c_j T_j(x) = \frac{1}{2}\big(c_0 + b_0(x) - b_2(x)\big). \tag{2.20}$$

This algorithm is often called Clenshaw's algorithm. As for Horner's scheme, the evaluation requires only n multiplications and Section 3.13 in [46] shows that this method *"is perfectly stable, the local rounding errors are not amplified, and the upper bound to the error cannot exceed the arithmetic sum of the individual local rounding errors"*.

Chebyshev interpolation on a general interval

So far we have only considered the standard Chebyshev interpolation on the interval $[-1,1]$. In most applications we will encounter problems on a general interval $[a,b] \subset \mathbb{R}$. In this case, we can use linear transformations and transform the interval $[a,b]$ to $[-1,1]$. In order to do so, we define the linear transformation τ as

$$\tau_{[a,b]} : [-1,1] \to [a,b] \quad \text{with} \quad \tau_{[a,b]}(z) = a + 0.5(b-a)(z+1) \tag{2.21}$$

and its inverse

$$\tau^{-1}_{[a,b]} : [a,b] \to [-1,1] \quad \text{with} \quad \tau_{[a,b]}(x) = -1 + 2\frac{(x-a)}{(b-a)}. \tag{2.22}$$

Assume $f \in C([a,b])$, then we can define a function $\tilde{f} \in C([-1,1])$ by $\tilde{f}(z) = f(\tau_{[a,b]}(z))$. For the coefficients of the Chebyshev interpolation $\tilde{p}_n$ of $\tilde{f}$ holds

$$c_j = \frac{2}{n}{\sum_{k=0}^{n}}'' \tilde{f}(z_k)T_j(z_k) = \frac{2}{n}{\sum_{k=0}^{n}}'' f(\tau_{[a,b]}(z_k))T_j(z_k) = \frac{2}{n}{\sum_{k=0}^{n}}'' f(x_k)T_j(z_k)$$

for Chebyshev points $z_k = \cos(k\pi/n)$, $k = 0,\dots,n$ and transformed nodal points $x_k = \tau_{[a,b]}(z_k)$. The function f at $x \in [a,b]$ is then interpolated by

$$f(x) = \tilde{f}(\tau^{-1}_{[a,b]}(x)) \approx \tilde{p}_n(\tau^{-1}_{[a,b]}(x)) = \sum_{j=0}^{n} c_j T_j(\tau^{-1}_{[a,b]}(x)) = \sum_{j=0}^{n} c_j p_j(x) =: I_n(f)(x)$$

for the transformed Chebyshev polynomials $p_j(x) = T_j(\tau^{-1}_{[a,b]}(x))\mathbb{1}_{[a,b]}(x)$.

The *chebfun* software project

A very useful software package for function approximation using Chebyshev is the *chebfun* toolbox for Matlab that is based on [9]. This toolbox can be found at *www.chebfun.org* and its functionality is described in detail in [37]. The package provides a good starting point for exploring Chebyshev interpolation and covers a wide range of different applications including approximation, quadrature, root-finding and solving differential equations. We refer to their web page for a list of different examples. Moreover, the package offers extensions to two dimensions (*chebfun2*) and three dimensions (*chebfun3*). For the majority of the experiments of this thesis we used our own implementation of the univariate Chebyshev interpolation rather than the *chebfun* package(s). This allowed us to tailor our implementation to the applications relevant for us and obtain the best performance. We use however the bivariate extension of the *chebfun* package and we will briefly introduce it at a later point.

2.1.3 Convergence results

In this section, we provide convergence results for the polynomial interpolation in Chebyshev points. Generally speaking, the convergence of polynomial interpolations depends on the smoothness of the approximated function. For this purpose smoothness is measured in the number of derivatives of the function that exits. A special emphasis is put on functions that are infinitely often continuously differentiable and can be analytically continued into the complex plane. A generic approach to obtain convergence results for any set of interpolation points is to prove results for the best approximation and then use (2.8) and the Lebesgue constant. For the Chebyshev points it is however possible to prove the convergence results directly.

First, we establish a result that links the approximation error to the coefficients of the Chebyshev series.

Proposition 2. *Let $f \in C([-1,1])$ be a Lipschitz continuous function, with Chebyshev series $f(x) = \sum_k a_k T_k(x)$. For the truncation of the series f_n and the polynomial interpolation in $(n+1)$ Chebyshev points p_n holds*

$$\|f - f_n\|_\infty \leq \sum_{k=n+1}^{\infty} |a_k| \quad \textit{and} \quad \|f - p_n\|_\infty \leq 2 \sum_{k=n+1}^{\infty} |a_k|. \tag{2.23}$$

Proof. The first results follows immediately from

$$f(x) - f_n(x) = \sum_{k=0}^{\infty} a_k T_k(x) - \sum_{k=0}^{n} a_k T_k(x) = \sum_{k=n+1}^{\infty} a_k T_k(x)$$

and $|T_k(x)| \leq 1$. The second result is based on the aliasing formula for the Chebyshev coefficients

$$c_k = a_k + (a_{k+2n} + a_{k+4n} + \ldots) + (a_{-k+2n} + a_{-k+4n} + \ldots) \quad \text{for} \quad 1 \leq k \leq n-1,$$

see Theorem 4.1 and Theorem 4.2 of [129] for more details. □

The proposition shows that the approximation error of the truncated series and the interpolation error are exactly the same up to a factor 2. From now on we will only focus on the latter one. In order to obtain convergence results we need to study the decay of the coefficients $|a_k|$ depending on the smoothness of the function f. First we introduce the concept of a Bernstein ellipse, see Chapter 8 of [129].

Definition 3 (Bernstein ellipse)**.** *For $\varrho > 1$ a Bernstein ellipse $\mathcal{B}([-1,1],\varrho)$ is defined as the open region in the complex domain which is bounded by an ellipse with foci ± 1*

and semiminor axis b_ϱ and semimajor axis a_ϱ, given by

$$a_\varrho = \frac{\varrho + \varrho^{-1}}{2} \qquad \text{and} \qquad b_\varrho = \frac{\varrho - \varrho^{-1}}{2},$$

with $a_\varrho + b_\varrho = \varrho$.

The following theorem shows that the coefficients of the Chebyshev series of a Lipschitz continuous function decays exponentially fast if the function has an analytic continuation to the complex plane. This decay is determined by the size ϱ of the Bernstein ellipse on which the function has an analytic extension. This result for the coefficients together with the equation (2.23) shows that the error of the Chebyshev interpolation converges exponentially fast. From [129] we obtain the following result.

Theorem 4. *Let $f \in C([-1,1])$ be an analytic function that can be analytically extended to a Bernstein ellipse $\mathcal{B}([-1,1],\varrho)$ for $\varrho > 1$ and assume $\sup_{x\in\mathcal{B}([-1,1],\varrho)} |f(x)| \leq V$ for a constant $V > 0$. Then the coefficients of the Chebyshev series of f satisfy $|a_0| \leq V$ and*

$$|a_k| \leq 2V\varrho^{-k} \quad \text{for} \quad k \geq 1.$$

The approximation error of the Chebyshev polynomial interpolation p_n satisfies

$$\|f - p_n\|_\infty \leq 4V\frac{\varrho^{-n}}{\varrho - 1}. \tag{2.24}$$

Proof. See Theorem 8.1 (for the first part) and Theorem 8.2 (for the second part) in [129]. □

The assumption of analyticity can be too strong in some applications of interest. If we relax this assumption and consider the larger class of differentiable functions we still obtain algebraic convergence. The convergence rate for a function f will then depend on the number of derivatives $f^p := \frac{d^p f}{dx^p}$ that are available. For a p-times continuously differentiable function the interpolation error decays with n^{-p}. From [129] we obtain the following result.

Theorem 5. *Let $f \in C([-1,1])$ and $p \geq 0$. Assume f and its derivatives up to f^{p-1} are absolutely continuous and f^p is of bounded variation V. Then we obtain for the coefficients of the Chebyshev series of f that*

$$|a_k| \leq \frac{2V}{\pi(k-p)^{p+1}}.$$

Furthermore, for the interpolation error of p_n holds

$$\|f - p_n\|_\infty \leq \frac{4V}{\pi p(n-p)^p}. \tag{2.25}$$

Proof. The proof follows from Theorem 7.1 and Theorem 7.2 of [129]. □

Note that a function is of bounded variation if the L^1 norm of its derivative is finite. A function is absolutely continuous if it can be written as the integral of its derivative, i.e. $f : [-1, 1] \to \mathbb{R}$ is absolutely continuous if

$$f(x) = f(-1) + \int_{-1}^{x} f'(y)\mathrm{d}y.$$

Derivative of the Chebyshev interpolant

Since the Chebyshev interpolant is a polynomial it is continuously differentiable. Based on the recurrence relation (2.12) we obtain the following result.

Proposition 3. *The derivative of the n-th Chebyshev polynomial T_n is a polynomial of degree $n-1$ and can be written as a sum of Chebyshev polynomials given by*

$$T_n'(x) = 2n \sum_{j=0}^{n-1}{}' T_j(x) \mathbb{1}_{(n+j) \bmod 2=1}.$$

Let $p_n(x) = \sum_{j=0}^{n} w_j T_j(x)$ be a weighted sum of Chebyshev polynomials. The derivative of p_n is then given by

$$p_n'(x) = \sum_{j=0}^{n-1}{}' \tilde{w}_j T_j(x) \quad \textit{with} \quad \tilde{w}_j = 2 \sum_{k=j+1}^{n} k w_k \mathbb{1}_{(k+j) \bmod 2=1},$$

where $\sum'$ indicates that the first term is multiplied with $1/2$.

Proof. First we show that for the derivative of T_n, $n \geq 1$ holds

$$\frac{\mathrm{d}T_n}{\mathrm{d}x}(x) = 2n \sum_{k=0}^{n-1}{}' T_k(x) \mathbb{1}_{(n-1+k) \bmod 2=0}.$$

For $n = 1$ we use that $T_1(x) = x$ and $T_1'(x) = 1 = T_0(x)$, we obtain

$$\frac{\mathrm{d}T_1}{\mathrm{d}x}(x) = 2 \sum_{k=0}^{0}{}' T_k(x) \mathbb{1}_{(0+k) \bmod 2=0} = 2\frac{1}{2} T_0(x) \mathbb{1}_{0 \bmod 2=0} = T_0(x).$$

Assume the formula holds for $j = 0, \dots, n$. From [93] we obtain the identity

$$2T_n(x) = \frac{1}{n+1}T'_{n+1}(x) - \frac{1}{n-1}T'_{n-1}(x) \quad n > 1.$$

This yields

$$\begin{aligned}
T'_{n+1}(x) &= 2(n+1)T_n(x) + \frac{(n+1)}{(n-1)}T'_{n-1}(x) \\
&= 2(n+1)T_n(x) + \frac{(n+1)}{(n-1)}2(n-1)\sum_{j=0}^{n-2}{}'T_j(x)\mathbb{1}_{(n-2+j) \bmod 2=0} \\
&= 2(n+1)\Big(T_n(x)\mathbb{1}_{(n+n) \bmod 2=0} + \sum_{j=0}^{n-2}{}'T_j(x)\mathbb{1}_{(n+j) \bmod 2=0}\Big) \\
&= 2(n+1)\sum_{j=0}^{n}{}'T_j(x)\mathbb{1}_{(n+j) \bmod 2=0}.
\end{aligned}$$

We used that $(n+j) \bmod 2 = (n+j-2) \bmod 2$ and $\mathbb{1}_{(n+n-1) \bmod 2=0} = 0$. This proves the first part of the preposition. Moreover, the result yields for the derivative of a weighted sum of Chebyshev polynomials p_n

$$\begin{aligned}
p'_n(x) = \sum_{j=0}^{n} w_j T'_j(x) &= \sum_{j=0}^{n} w_j 2j \sum_{k=0}^{j-1}{}'T_k(x)\mathbb{1}_{(j-1+k) \bmod 2=0} \\
&= \sum_{j=0}^{n}\sum_{k=0}^{j-1}{}'w_j 2jT_k(x)\mathbb{1}_{(j-1+k) \bmod 2=0} \\
&= \sum_{j=1}^{n}\sum_{k=0}^{j-1}{}'w_j 2jT_k(x)\mathbb{1}_{(j-1+k) \bmod 2=0} \\
&= \sum_{k=0}^{n-1}{}'\sum_{j=k+1}^{n} w_j 2jT_k(x)\mathbb{1}_{(j-1+k) \bmod 2=0} \\
&= \sum_{k=0}^{n-1}{}'T_k(x)\Big(\underbrace{\sum_{j=k+1}^{n} w_j 2j\mathbb{1}_{(j-1+k) \bmod 2=0}}_{=:\tilde{w}_k}\Big)
\end{aligned}$$

where $\sum'$ indicates that the first term is multiplied with $1/2$ if $k = 0$. This was our claim. □

More generally, if we have a Chebyshev interpolation $I_n(f)(x) = \sum_{j=0}^{n} c_j p_j(x)$ on an

interval $\mathcal{X}$ with $p_j(x) = T_j(\tau_{\mathcal{X}}^{-1}(x))$, its derivative is given by

$$\frac{\mathrm{d}}{\mathrm{d}x} I_n(f)(x) = \sum_{j=0}^{N} c_j \frac{\mathrm{d}T_j}{\mathrm{d}\tau_{\mathcal{X}}^{-1}(x)} \frac{\mathrm{d}\tau_{\mathcal{X}}^{-1}}{\mathrm{d}x}(x) = \frac{\mathrm{d}\tau_{\mathcal{X}}^{-1}}{\mathrm{d}x}(x) \sum_{j=0}^{n-1} \tilde{c}_j p_j(x)$$

where we used the chain rule and Proposition 3. The coefficients $\tilde{c}_j$ are given by

$$\tilde{c}_j = 2 \sum_{k=j+1}^{n}{}' k c_k \mathbb{1}_{(k+j) \bmod 2 = 1}.$$

The question arises if the derivative of the Chebyshev interpolant of a function f is also a good approximation for the derivative of f. For the convergence analysis we define a Sobolev space using a weighted L^2 norm, see [125]. For $f \in C([-1,1])$ we define the norm

$$||f||_{L_T^2}^2 = \frac{1}{\pi} \int_{-1}^{1} w(x) f(x) \mathrm{d}x \tag{2.26}$$

for the Chebyshev weight function $w(x)$ as defined in (2.14). For $s \in \mathbb{N}$, we define the Sobolev space

$$W_T^s = \left\{ f : [-1,1] \to \mathbb{R} : \ ||f||_{W_T^s}^2 := \sum_{k=0}^{s} \left|\left| \frac{d^k f}{dx^k} \right|\right|_{L_T^2}^2 < \infty \right\}. \tag{2.27}$$

For functions in W_T^s we obtain a similar convergence results for the Chebyshev interpolation as in Theorem 5. Moreover, the derivatives of the Chebyshev interpolant approximate the derivatives of the function. From Corollary 4.3 in [125] we obtain the following result.

Theorem 6. *Let $f \in C([-1,1])$ be in W_T^s for some $s \in \mathbb{N}$ with $s \geq 1$ and let p_n be the Chebyshev interpolant of f. For any σ with $0 \leq 2\sigma \leq s$, we obtain*

$$||f - p_n||_{W_T^\sigma} \leq C_s ||f||_{W_T^s} n^{-s+2\sigma}. \tag{2.28}$$

This result means that we "loose" two orders of convergence for every derivative we want to approximate simultaneously. Note that for $\sigma = 0$ we have a classical convergence result of the Chebyshev interpolation in a (weighted) L_2-norm. A similar result for the approximation in the Sobolev norm can be obtained in the case of an analytic function. Essentially, the term n^{-s} is in this case replaced by a term that decreases exponentially fast. From [125] we obtain the following result.

Theorem 7. *Let $f \in C([-1,1])$ be an analytic function that can be analytically extended*

to a Bernstein ellipse $\mathcal{B}([-1,1],\varrho)$ *for* $\varrho > 1$ *and assume* $\sup_{x\in\mathcal{B}([-1,1],\varrho)} |f(x)| \leq V$ *for a constant* $V > 0$*. Then we obtain for the polynomial approximation* p_n *in the Sobolev norm for* $\sigma \geq 0$*,*

$$\|f - p_n\|_{W_T^\sigma} \leq C_\sigma \frac{V}{\sinh(\varrho)} n^{2\sigma} \varrho^{-n}$$

for some constant $C_\sigma > 0$*.*

In fact, the derivative of the Chebyshev interpolation does not only converge to the derivative of the function in the L^2 norm but also in the stronger L^∞ norm. From [129] we obtain a convergence result similar to Theorem 4 for the derivatives.

Theorem 8. *Let* $f \in C([-1,1])$ *be an analytic function that can be analytically extended to the closure of a Bernstein ellipse* $\mathcal{B}([-1,1],\varrho)$ *for* $\varrho > 1$*. For each integer* s*, the* s*th derivatives of the Chebyshev projections* f_n *and interpolations* p_n *of* f *satisfy*

$$\|f^s - f_n^s\|_\infty = \mathcal{O}(\varrho^{-n}) \quad \textit{and} \quad \|f^s - p_n^s\|_\infty = \mathcal{O}(\varrho^{-n})$$

for $n \to \infty$*.*

Proof. See Theorem 21.1 in [129]. □

2.2 Multivariate extensions

The fundamental idea of approximating a function using polynomials such that the resulting approximation is exact at a set of interpolation points can be extend to higher dimensions. In a multivariate set-up however, it is less obvious how we can find a unique interpolation of degree n given a set of distinct nodal points. The first question that arises is how we define the degree of a multivariate polynomial $x_1^{k_1} x_2^{k_2} \dots x_d^{k_d}$ or written in a simpler form $\boldsymbol{x}^{\boldsymbol{k}}$ with $\boldsymbol{k} = (k_1, k_2, \dots, k_d)$. Standard choices are the maximum norm $\|\boldsymbol{k}\|_\infty = \max_{1\leq i\leq d} k_i$ and the sum $k_1 + k_2 + \ldots + k_d$, i.e. the 1-norm of $\boldsymbol{k}$. The first one leads to the so-called tensor product (TP) and the second one to the total degree (TD) space of polynomials. The two choices lead to very different types of approximations and both have their merits and demerits.

Any univariate set of interpolation nodes and basis functions can be used to generate a tensor product interpolation in d dimensions. Let $f \in C([-1,1]^d)$ be the vector space of continuous functions $f : [-1,1]^d \to \mathbb{R}$. Then we can write the interpolation of f as

$$I_n(f)(\boldsymbol{x}) = \sum_{\|\boldsymbol{j}\|_\infty \leq n} c_{\boldsymbol{j}} \varphi_{\boldsymbol{j}}(\boldsymbol{x}) = \sum_{j_1=0}^{n} \dots \sum_{j_d=0}^{n} c_{j_1,\dots,j_d} \prod_{i=1}^{d} \varphi_{j_i}(x_i) \tag{2.29}$$

for one-dimensional basis functions $\varphi_0, \varphi_1, \ldots, \varphi_n$. This isotropic interpolation can be generalized to an anisotropic one, i.e. the tensor product of univariate interpolations of different degrees. For $\boldsymbol{n} \in \mathbb{N}^d$ we write

$$I_{\boldsymbol{n}}(f)(\boldsymbol{x}) = \sum_{0 \leq \boldsymbol{j} \leq \boldsymbol{n}} c_{\boldsymbol{j}} \varphi_{\boldsymbol{j}}(\boldsymbol{x}) = \sum_{j_1=0}^{n_1} \cdots \sum_{j_d=0}^{n_d} c_{j_1,\ldots,j_d} \prod_{i=1}^{d} \varphi_{j_i}(x_i)$$

where $0 \leq \boldsymbol{j} \leq \boldsymbol{n}$ is a componentwise comparison. In both cases we need $(n_1+1)\cdots(n_d+1)$ points and the coefficients $c_{\boldsymbol{j}}$ build a d-variate tensor in $\mathbb{R}^{n_1 \times \ldots \times n_d}$. This approach is straightforward and yields a unique interpolation polynomial for every set of distinct univariate nodal points. The drawback is that the number of nodal points and polynomials grows exponentially in the dimension d. Even for a moderately low number of points n the total number of points can be infeasibly high. This is often called the *curse of dimensionality* and makes the method impractical for some applications. Sometimes, it is however possible to compress the large tensor into a low-rank tensor.

In contrast, the interpolation of bounded total degree of a function $f \in C([-1,1]^d)$ is given by

$$I_n^{TD}(f)(\boldsymbol{x}) = \sum_{\|\boldsymbol{j}\|_1 \leq n} c_{\boldsymbol{j}} \varphi_{\boldsymbol{j}}(\boldsymbol{x}) \tag{2.30}$$

for $\boldsymbol{x} \in [-1,1]^d$. The number of points grows significantly slower in d and this can make the interpolation more attractive for higher dimensions. The choice of a set of nodal points that guarantee a unique interpolation polynomial in the total degree is challenging. In two dimensions, the so-called (bivariate) Padua points introduced in [27] are a suitable choice and extensions of these points to three dimensions and the general d-dimensional case have been made, ses [18] and the references therein. All these sets of nodal points have in common that they are subsets of tensor product Chebyshev grids chosen in a meaningful way. The underlying idea is that the approximation using a smaller but carefully selected grid can achieve almost the same accuracy as a full tensor grid. A comparison of different methods for multivariate function approximation and quadrature is provided in [130].

In this section we will discuss the tensor product interpolation and provide convergence results. We briefly introduce low-rank tensor approximation and provide an overview on sparse grids.

2.2.1 Tensor based Chebyshev interpolation

The tensor based Chebyshev interpolation (or tensor product) is the straightforward extension of the univariate polynomial interpolation to higher dimensions. For the interpolation of a d-variate function, a grid of $(n+1)^d$ points is used and the resulting interpolant is a polynomial of degree n in each dimension. For $\boldsymbol{n} \in \mathbb{N}^d$, the tensor grid of Chebyshev points are given by

$$\boldsymbol{x_k} = (x_{k_1}, \ldots, x_{k_d}) \quad \text{with} \quad x_{k_i} = \cos(k_i \pi / n_i) \quad \text{for} \quad 0 \leq k_i \leq n_i,\ i = 1, \ldots, d.$$

The corresponding multivariate Chebyshev polynomials are defined as products of the one-dimensional Chebyshev polynomials, i.e.

$$T_{\boldsymbol{j}}(\boldsymbol{x}) := \prod_{i=1}^{d} T_{j_i}(x_i) \quad \text{for} \quad \boldsymbol{x} \in [-1,1]^d.$$

Now we are in a position to define the multivariate Chebyshev interpolation.

Definition 4 (Multivariate Chebyshev interpolation)**.** *The multivariate Chebyshev polynomial interpolation of a function $f \in \mathcal{C}([-1,1]^d)$ for degree $\boldsymbol{n} \in \mathbb{N}^d$ is given by*

$$I_{\boldsymbol{n}}(f)(\boldsymbol{x}) := \sum_{0 \leq \boldsymbol{j} \leq \boldsymbol{n}} c_{\boldsymbol{j}} T_{\boldsymbol{j}}(\boldsymbol{x}) = \sum_{j_1=0}^{n_1} \ldots \sum_{j_d=0}^{n_d} c_{j_1,\ldots,j_d} \prod_{i=1}^{d} T_{j_i}(x_i) \tag{2.31}$$

with coefficients

$$\begin{aligned} c_{\boldsymbol{j}} &= \Big(\prod_{i=1}^{d} \frac{2^{\mathbb{1}_{0<j_i<n_i}}}{n_i} \Big) \sum_{0 \leq \boldsymbol{k} \leq \boldsymbol{n}}{}'' f(\boldsymbol{x_k}) T_{\boldsymbol{j}}(\boldsymbol{x_k}) \\ &= \Big(\prod_{i=1}^{d} \frac{2^{\mathbb{1}_{0<j_i<n_i}}}{n_i} \Big) \sum_{k_1=0}^{n_1}{}'' \ldots \sum_{k_d=0}^{n_d}{}'' f(x_{k_1}, \ldots, x_{k_d}) \prod_{i=1}^{d} T_{j_i}(x_{k_i}). \end{aligned}$$

Here $I_{\boldsymbol{n}}(f)$ can be seen as an operator from the space of continuous functions on $[-1,1]^d$ to the space of all polynomials of degree $\boldsymbol{n}$. The Chebyshev interpolation on $[-1,1]^d$ can then be extended to any hyperrectangular

$$\mathcal{X} = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_d, \overline{x}_d] \subset \mathbb{R}^d$$

using the linear transformations (2.21) and (2.22) in each dimension.

Convergence results

For the error analysis of the tensor based interpolation we need to generalize the concept of a Bernstein ellipse. The following definition is based on [58].

Definition 5 (Generalized Bernstein ellipse)**.** *Let $\boldsymbol{\varrho} \in (1,\infty)^d$ and $\mathcal{X} \subset \mathbb{R}^d$ a hyperrectangle given by $\mathcal{X} = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_d, \overline{x}_d]$. The generalized Bernstein ellipse is defined by*

$$\mathcal{B}(\mathcal{X}, \varrho) := \mathcal{B}([\underline{x}_1, \overline{x}_1], \varrho_1) \times \ldots \times \mathcal{B}([\underline{x}_d, \overline{x}_d], \varrho_d)$$

where $\mathcal{B}([\underline{x}_i, \overline{x}_i], \varrho_i)$ are the transformations of the Bernstein ellipses $\mathcal{B}([-1,1], \varrho_i)$ for $i = 1, \ldots, d$.

Using this definition we are in a position to extend the convergence results of (2.24) for univariate analytic functions to the multivariate case. From [58] we obtain the following error bound.

Theorem 9. *Let $f \in \mathcal{C}(\mathcal{X})$ for a hyperrectangle $\mathcal{X} \subset \mathbb{R}^d$. Assume f has an analytic extension to some generalized Bernstein ellipse $\mathcal{B}(\mathcal{X}, \boldsymbol{\varrho})$ for some $\boldsymbol{\varrho} \in (1,\infty)^d$ and $\sup_{x \in \mathcal{B}(\mathcal{X},\varrho)} |f(x)| \leq V$ for some $V > 0$. Then we obtain for the approximation error of the multivariate Chebyshev interpolation $\widehat{f} := I_{\boldsymbol{n}}(f)$ for $\boldsymbol{n} \in \mathbb{N}^d$*

$$\|f - \widehat{f}\|_\infty \leq \min\{\alpha(\varrho, \boldsymbol{n}, d, V), \beta(\varrho, \boldsymbol{n}, d, V)\} =: \varepsilon_{int}(\varrho, \boldsymbol{n}, d, V) \tag{2.32}$$

with error bounds

$$\alpha(\varrho, \boldsymbol{n}, d) = \min_{\sigma \in S_d} \sum_{i=1}^{d} \left(4V \frac{\varrho_{\sigma(i)}^{-n_i}}{\varrho_i - 1} \; + \; \sum_{k=2}^{d} 4V \frac{\varrho_{\sigma(k)}^{-n_k}}{\varrho_{\sigma(i)} - 1} \cdot 2^{k-1} \frac{(k-1) + 2^{k-1} - 1}{\prod_{j=1}^{k-1}(1 - \varrho_{\sigma(j)}^{-1})} \right) \tag{2.33}$$

$$\beta(\varrho, \boldsymbol{n}, d) = 2^{(d/2)+1} \cdot V \cdot \left(\sum_{i=1}^{d} \varrho_i^{-2n_i} \prod_{j=1}^{d} \frac{1}{1 - \varrho_j^{-2}} \right)^{-1/2} \tag{2.34}$$

where S_d is the symmetric group of d elements.

As pointed out in [58], it depends on the specific specification of $\boldsymbol{n}$ and $\boldsymbol{\varrho}$ which of the two error bounds is actually sharper. From the theorem follows the more general results

$$\|f - \widehat{f}\|_\infty \leq C \underline{\varrho}^{-\underline{n}} \quad \text{with} \quad \underline{\varrho} = \min_{1 \leq i \leq d} \varrho_i, \; \underline{n} = \min_{1 \leq i \leq d} n_i \tag{2.35}$$

for a constant $C > 0$. Assuming that all n_i's are the same, the error decays in the d-th root of the total number of points $(n+1)^d$. This is called the curse of dimensionality. In order to maintain the same convergence as in the univariate case, the total number

of points increases exponentially in the dimension. Similar results can be obtained for polynomial interpolation of differentiable functions.

Interpolation with distortion

In many practical applications we would like to approximate functions that we do not know analytically. Instead, we need to calculate the nodal values $f(\boldsymbol{x_k})$ of the multivariate Chebyshev interpolation numerically. Typical examples are situations where f is the solution of a PDE or a conditional expectation of a stochastic process. The function values at the nodal points are then computed via numerical quadrature, finite differences or Monte Carlo simulation. All these numerical routines introduce an additional error and result in distorted values $\widehat{f}(\boldsymbol{x_k}) \approx f(\boldsymbol{x_k})$. This distortion error affects then the overall quality of the interpolation.

Depending on the numerical technique used to compute $\widehat{f}(\boldsymbol{x_k})$, it is often possible to bound the maximal distortion error. On the other hand, Monte Carlo simulation means that the distortion error becomes a random variable. In both cases, the distortion value will influence the overall error and the maximal achievable accuracy. In the context of option pricing this idea of interpolation with distortion error has been investigated in [49]. The following proposition provides a convergence result if the distortion is deterministic and can be bounded by a constant.

Proposition 4. *Let $f \in \mathcal{C}(\mathcal{X})$ for a hyperrectangle $\mathcal{X} \subset \mathbb{R}^d$ be a real-valued function with an analytic extension to some generalized Bernstein ellipse $\mathcal{B}(\mathcal{X}, \varrho)$ for $\varrho \in (1, \infty)^d$ with $\sup_{\boldsymbol{x} \in \mathcal{B}(\mathcal{X}, \varrho)} |f(\boldsymbol{x})| \leq B$. Assume distorted values $f^\varepsilon(\boldsymbol{x_k}) = f(\boldsymbol{x_k}) + \varepsilon(\boldsymbol{x_k})$ with $|\varepsilon(\boldsymbol{x_k})| \leq \bar{\varepsilon}$ at all nodes $\boldsymbol{x_k}$. Then*

$$\max_{\boldsymbol{x} \in \mathcal{X}} \left| f(\boldsymbol{x}) - I_{\boldsymbol{n}}(f^\varepsilon)(\boldsymbol{x}) \right| \leq \varepsilon_{int}(\varrho, \boldsymbol{n}, d, B) + \bar{\varepsilon} \Lambda_{\boldsymbol{n}}.$$

with $\varepsilon_{int}(\varrho, \boldsymbol{n}, d, B) := \min\{\alpha(\varrho, \boldsymbol{n}, d), \beta(\varrho, \boldsymbol{n}, d)\}$ for α, β as defined in (2.33) *and* (2.34) *and Lebesgue constant $\Lambda_{\boldsymbol{n}} \leq \prod_{i=1}^d \left(\frac{2}{\pi} \log(n_i + 1) + 1\right)$.*

Proof. Using the linearity of the interpolation operator we obtain for the Chebyshev interpolation of f^ε with $f^\varepsilon(\boldsymbol{x_k}) = f(\boldsymbol{x_k}) + \varepsilon(\boldsymbol{x_k})$ that

$$I_{\boldsymbol{n}}(f^\varepsilon)(\boldsymbol{x}) = I_{\boldsymbol{n}}(f)(\boldsymbol{x}) + I_{\boldsymbol{n}}(\varepsilon)(\boldsymbol{x}).$$

The tensor-based multivariate Chebyshev interpolation $I_{\boldsymbol{n}}(\varepsilon)$ can be written in Lagrange

form

$$I_{\boldsymbol{n}}(\varepsilon)(\boldsymbol{x}) = \sum_{\mathbf{0}\leq \boldsymbol{j}\leq \boldsymbol{n}} \varepsilon(\boldsymbol{x}_{\boldsymbol{j}})\lambda_{\boldsymbol{j}}(\boldsymbol{x}) \quad \text{with} \quad \lambda_{\boldsymbol{j}}(\boldsymbol{x}) = \prod_{i=1}^{d} \ell_{j_i}(\tau^{-1}_{[\underline{x}_i,\overline{x}_i]}(x_i))$$

where $\ell_{j_i}(z) = \prod_{k\neq j_i} \frac{z-z_k}{z_{j_i}-z_k}$ is the j_i-th Lagrange polynomial. This yields

$$\max_{\boldsymbol{x}\in\mathcal{X}} |I_{\boldsymbol{n}}(\varepsilon)(\boldsymbol{x})| = \max_{\boldsymbol{x}\in\mathcal{X}} \Big| \sum_{\mathbf{0}\leq \boldsymbol{j}\leq \boldsymbol{n}} \varepsilon(\boldsymbol{j})\lambda_{\boldsymbol{j}}(\boldsymbol{x})\Big| \leq \overline{\varepsilon} \max_{\boldsymbol{x}\in\mathcal{X}} \sum_{\mathbf{0}\leq \boldsymbol{j}\leq \boldsymbol{n}} |\lambda_{\boldsymbol{j}}(\boldsymbol{x})| =: \overline{\varepsilon}\Lambda_{\boldsymbol{n}}.$$

The term $\Lambda_{\boldsymbol{n}}$ is the Lebesgue constant of the (multivariate) Chebyshev nodes which is given by

$$\Lambda_{\boldsymbol{n}} = \max_{\boldsymbol{x}\in\mathcal{X}} \sum_{\mathbf{0}\leq \boldsymbol{j}\leq \boldsymbol{n}} \big|\lambda_{\boldsymbol{j}}(\boldsymbol{x})\big| = \max_{\boldsymbol{x}\in\mathcal{X}} \sum_{\mathbf{0}\leq \boldsymbol{j}\leq \boldsymbol{n}} \prod_{i=1}^{d} \big|\ell_{j_i}(x_i)\big| = \prod_{i=1}^{d} \max_{x_i\in[\underline{x}_i,\overline{x}_i]} \sum_{j_i=0}^{n_i} \Big|\ell_{j_i}\big(\tau^{-1}_{[\underline{x}_i,\overline{x}_i]}(x_i)\big)\Big|.$$

Since $\max_{x_i\in[\underline{x}_i,\overline{x}_i]} \sum_{j_i=0}^{n_i} |\ell_{j_i}(\tau^{-1}_{[\underline{x}_i,\overline{x}_i]}(x_i))| = \max_{z\in[-1,1]} \sum_{j_i=0}^{n_i} |\ell_{j_i}(z)| = \Lambda_{n_i}$, which is the Lebesgue constant of the univariate Chebyshev interpolation, we have $\Lambda_{\boldsymbol{n}} = \prod_{i=1}^{d} \Lambda_{n_i}$. From (2.19) we obtain for the univariate Chebyshev interpolation $\Lambda_n \leq \frac{2}{\pi}\log(n+1)+1$ and hence

$$\Lambda_{\boldsymbol{n}} \leq \prod_{i=1}^{d} \Big(\frac{2}{\pi}\log(n_i+1)+1\Big). \tag{2.36}$$

For the distorted Chebyshev interpolation holds

$$\big|f(\boldsymbol{x}) - I_{\boldsymbol{n}}(f^{\varepsilon})(\boldsymbol{x})\big| \leq |f(\boldsymbol{x}) - I_{\boldsymbol{n}}(f)(\boldsymbol{x})| + |I_{\boldsymbol{n}}(\varepsilon)(\boldsymbol{x})|.$$

Therefore, the proposition follows directly from (2.36) and Theorem 9. □

A more general version of this result can be found in [49, Theorem 2.5] and includes also the case of a stochastic distortion error. Here, we observe again how critical the Lebesgue constant is for a stable and accurate interpolation. It ensures that a small distortion can not lead to a large interpolation error even for a high number of nodal points.

2.2.2 Complexity reduction for polynomial interpolation

The presented tensor product Chebyshev interpolation suffers from the curse of dimensionality. From the error bound (2.35) follows that the error decay is of order $\mathcal{O}(\varrho^{-\sqrt[d]{n}})$ in the total number of points n. This tensor product interpolation will only be efficient if the function is smooth enough and ϱ is large and thus n remains small. If the function

is not smooth enough or the dimension becomes too high, a full tensor approach is no longer feasible. There are three main challenges for the full tensor Chebyshev interpolation: The computation of the nodal values $f(\boldsymbol{x_k})$, the storage of the coefficients and the evaluation of the interpolant. Depending on the application, just one or all of them can be a limitation for the applicability of the Chebyshev interpolation in multivariate dimensions.

In this section we present concepts how the different challenges of multivariate interpolation can be tackled. We start with a bivariate Chebyshev interpolation and discuss the application of a singular value decomposition (SVD) and the implementation of the *chebfun2* package. Then we briefly discuss low-rank tensor compression in high dimensions and a tensor completion algorithm.

Bivariate Chebyshev, SVD and *chebfun2*

In two dimensions, the Chebyshev interpolation can be written in form of matrix multiplications. Let $f : [-1,1]^2 \to \mathbb{R}$ be a continuous function. The bivariate Chebyshev interpolation of f can be written as

$$I_n(f)(x,y) = \begin{pmatrix} T_0(x), & \dots, & T_n(x) \end{pmatrix} \begin{pmatrix} c_{0,0} & \dots & c_{0,n} \\ \vdots & & \vdots \\ c_{n,0} & \dots & c_{n,n} \end{pmatrix} \begin{pmatrix} T_0(y) \\ \vdots \\ T_n(y) \end{pmatrix} =: \mathcal{T}_1(x)\mathcal{C}\mathcal{T}_2(y).$$

The matrix of coefficients $\mathcal{C} \in \mathbb{R}^{n\times n}$ grows quadratically in the number of points n. The matrix admits the singular value decomposition $\mathcal{C} = UDV^T$ for a diagonal matrix of singular values $D = \operatorname{diag}(d_1, \dots, d_k)$ and two orthogonal matrices U, V. Using only the k biggest singular values results in a rank k approximation of the function f, i.e.

$$I_n(f)(x,y) \approx \sum_{j=1}^{k} d_j \Big(\sum_{i_1=0}^{n} U_{i_1,k} T_{i_1}(x) \Big) \Big(\sum_{i_2=0}^{n} V_{k,i_2}^T T_{i_2}(y) \Big).$$

This means that the full tensor is approximated by k products of one-dimensional Chebyshev interpolations that require $2nk$ entries in total plus k singular values $d_1, \dots, d_k$. If the singular values decay fast, this can reduce the storage significantly. Moreover, the evaluation of the interpolant becomes more efficient. The chosen rank k using the singular value decomposition is optimal in a (discrete) L^2-sense. The drawback of this approach is that its effort is $\mathcal{O}(n^3)$ and the full tensor of coefficients has to be computed first. This means that the number of function evaluations is still $(n+1)^2$, see [127] for more details. This approach is therefore only of interest if we can use an offline-online decomposition and shift the singular value decomposition into the offline step.

[127] propose to use a Gaussian elimination with complete pivoting instead that results in a near optimal rank k and requires only $\mathcal{O}(k^2n + k^3)$ operations. The idea of their algorithm is the following: Find (x_0, y_0) that maximizes $|f|$ and construct a rank 1 approximation of f by

$$f_1(x, y) := \frac{1}{f(x_0, y_0)} f(x_0, y) f(x, y_0) \approx d_1 c_1(y) r_1(x)$$

where c_1 is a Chebyshev interpolation of the slice $y \mapsto f(x_0, y)$ and r_1 is the Chebyshev interpolation of $x \mapsto f(x, y_0)$. The weight d_1 is called the pivot value. Repeat this procedure for the residuum $f - f_1$ and obtain a new rank 2 approximation of f. The algorithm stops if $f - f_k$ is small enough, e.g. close to machine precision. The resulting approximation is then given by

$$f(x, y) \approx \sum_{j=1}^{k} d_j c_j(y) r_j(x)$$

where $r_j(x)$ and $c_j(y)$ are univariate Chebyshev interpolations of degree n. We refer to [127] for more details on the implementation. The resulting algorithm in the Matlab package *chebfun2* avoids the computation of a full tensor first and needs often only a small number of function evaluations.

Low rank tensor compression

Next we want to extend the rank k approximation of the full tensor product interpolation to the multivariate cases. Instead of a singular value decomposition of the coefficient matrix we need to find a method to decompose and compress a tensor of coefficients $\mathcal{C} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$. Different low rank tensor compression techniques have been developed for this tasked, we refer to [62] for a literature overview. We will focus on one of them and briefly introduce the tensor train (TT) format of [102]. The idea is to write every entry of a larger tensor in d dimension as a product of d matrices of smaller size. A tensor $\mathcal{C} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ in the tensor train format can be written as

$$\mathcal{C}(i_1, i_2, \ldots, i_d) = G_1(i_1) G_2(i_2) \ldots G_d(i_d) \quad \text{with} \quad G_\mu(i_\mu) \in \mathbb{R}^{r_{\mu-1} \times r_\mu}$$

with conditions $r_0 = r_d = 1$. The matrices $G_\mu(i_\mu)$ form essentially three dimensional arrays of size $r_{\mu-1} \times n_\mu \times r_\mu$ for every $\mu = 1, \ldots, d$. The effort for compressing a full tensor into the TT-format is of order $\mathcal{O}(dnr^3)$ for the maximal rank $r = \max_{0 \le \mu \le d} r_\mu$ and hence, scales linearly in the dimension and in the number of points n. It reduces storage requirements significantly if the rank r is small in comparison to n and in high dimensions d. Standard linear algebra operations such as vector times matrix operations can be

efficiently done using this format. We again refer to [102] for a detailed investigation of the TT-format and a discussion of the implementation of the tensor compression in Matlab.

Similarly to the singular value decomposition, the full tensor has to be computed first before it can be compressed into a low-rank structure. This limits the applicability of the compression format in high dimensions significantly. Fortunately, there exists different completion algorithms that construct an approximation of a high-dimensional tensor in a low-rank format by using only relatively few entries. For the tensor train format, [123] presents a tensor completion algorithm based on Riemannian optimization. Given a set of indices of the tensor and a fixed maximal rank the algorithm finds the optimal low-rank tensor. Usually, the (optimal) rank of a full tensor is unknown and has to be found adaptively, see [123] for more details. The rank adaptive completion algorithm comes with the drawback of a large computational overhead. Therefore, especially in medium high dimension, it is not always faster than working with the full tensor. This computational bottleneck vanishes if the completion can be done in an offline step.

[57] combine a multivariate Chebyshev interpolation with low-rank tensor compression and a tensor completion algorithm. They show numerically that the tensor of Chebyshev coefficients admits a low-rank structure for analytic functions. A similar approach has also been investigated in the master thesis [137] who uses the hierarchical Tucker decomposition of [84] and the black box approximation technique of [5] instead.

Alternatives for multivariate interpolation

In the introduction of this section, we briefly mentioned the idea of polynomial interpolation using the total degree. Exploring the same idea, sparse grids are a popular choice for multivariate interpolation. Sparse grids refers to methods that combine tensor product interpolations of smaller order to obtain an interpolation polynomial with bounded total degree. By combining only the tensor products that contribute most to the overall error, the approaches aim to overcome the curse of dimensionality. Sparse grids were introduced in [139] and the idea of combining tensor product interpolation goes back to [120]. There are two main types of sparse grids, either they are based on local basis functions such as piecewise linear (or piecewise quadratic) functions or on global polynomials. The combination of Smolyak's sparse grid algorithm with Chebyshev polynomial interpolation was proposed in [7]. On comprehensive introduction to sparse grids using local basis functions is given in [23]. Sparse grids can be further improved by using adaptive sparse grids algorithm. This is especially interesting for anisotropic problems.

A common alternative to interpolation approaches are multivariate function approx-

imations using least-squares fitting. A large number of nodal points is generated and then the coefficients of a multivariate polynomial of form (2.29) or (2.30) is fitted by minimizing the least-squares error. The resulting approximation polynomial of degree n is then optimal in a discrete (and possibly weighted) L^2-norm. The nodal points can either be sampled randomly which adds simulation noise or in a deterministic way according to an appropriate sampling algorithm. A design of determinsitic nodal points is for example provided in [141] and [18]. In their approaches, the number of points needs to scale quadratically in the number of polynomials to ensure optimal convergence. For our purposes this makes these types of approaches often infeasible.

We refer again to [130] for a comparison of different methods for multivariate function approximation and quadrature. The author suggests that neither the tensor product nor the total degree is optimal. The first one oversamples and uses too many polynomials whereas the latter one undersamples and uses too few polynomials. Based on these findings [131] suggests to use an Euclidean degree polynomial, i.e. to define the approximation using the Euclidean norm

$$I_n^{EC}(f)(\boldsymbol{x}) = \sum_{\|\boldsymbol{j}\|_2 \leq n} c_{\boldsymbol{j}} \varphi_{\boldsymbol{j}}(\boldsymbol{x}).$$

For analytic functions, this approximation can achieve a better accuracy than the total degree while using at the same time less polynomials than the full tensor product. The results are so far however theoretical and finding an appropriate set of interpolation points is an open research question.

2.3 Application of Chebyshev in finance

In this section, we investigate how Chebyshev interpolation can be used in finance in order to speed-up option pricer. The idea behind this section was first introduced in [49] and has also been investigated in [47] as well as in [91] and in [107]. We will investigate option prices as functions of their model and payoff parameters. Then we show how we can use Chebyshev interpolation in order to explore the smoothness of this function. Before we can introduce the general concept of parametric option pricing we start with a short review of the fundamentals of option pricing.

2.3.1 Fundamentals of option pricing

In this section we want to introduce the main concepts of option pricing in mathematical finance. In order to keep this introduction brief we focus on equity options and ignore for the moment other asset classes such as interest rate, fixed income, foreign exchange

and commodities. While there are asset class specific differences, many of the introduced concepts for equity options are also relevant for other products. A more detailed overview and a discussion of the stochastic calculus behind mathematical finance can for example be found in [14] or [41].

The two most common option types are a call and a put option. A call option gives the holder of the option the right to buy a stock S at a future time point T for a fixed price K. The option holder will only use this exercise right if the price of the stock a time point T indicated as S_T is higher than the strike. In this case, the option is worth $S_T - K$ to him and otherwise the value is zero for the option holder. The resulting payoff can be written as $(S-K)^+ := \max\{S-K, 0\}$. The put option is similar, but here the option holder has the right to sell the stock for the fixed price K. It is common practice that the stock itself is not physically purchased and the option holder only receives the payoff. The basic question of mathematical finance is: *What is the fair value of a call option?*

The standard approach in the valuation of options is to define a market and model the price of the stock as a stochastic process $(S_t)_{t\geq 0}$ given an initial stock price S_0. Once such a model is defined one can calculate the price of the option. The most important model is the Black-Scholes or Black-Scholes-Merton model introduced by [15] and [95]. They assume that there is a bank account with a fixed interest $r \geq 0$ at which all market participants can borrow or invest money. This process is modelled by the equation

$$\mathrm{d}B_t = rB_t\mathrm{d}t \quad \text{with} \quad B_0 = 1$$

and has the explicit solution $B_t = \exp(rt)$. It can be used to estimate today's value of a future cash-flow. The stock price process is modelled by the stochastic differential equation (SDE)

$$\frac{\mathrm{d}S_t}{S_t} = \mu\mathrm{d}t + \sigma\mathrm{d}W_t \quad \text{with} \quad S_0 = s_0 \tag{2.37}$$

for a drift $\mu \geq 0$, a volatility $\sigma > 0$ and W_t is a Brownian motion under a probability measure $\mathbb{P}$. This means W_t is a stochastic process starting at 0 that has normally distributed increments $W_t - W_s \sim \mathcal{N}(0, t-s)$ for $t \geq s$. The SDE (2.37) has the explicit solution

$$S_t = S_0 \exp\big((\mu - 0.5\sigma^2)t + \sigma W_t\big). \tag{2.38}$$

The price of a call option in such a stock price model can be calculated as the discounted expected payoff $e^{-rT}\mathbb{E}[(S_T - K)^+]$. Here, the expectation is however not calculated

under $\mathbb{P}$ but under an equivalent measure $\mathbb{Q}$ for which the discounted stock price process is a martingale. The measure $\mathbb{Q}$ is called the pricing or equivalent martingale measure and we call this pricing formula risk-neutral valuation. See [14] for more details and a discussion of the underlying idea of no arbitrage. In the Black-Scholes model, the price of a call option is given by

$$C_{BS}(S, K, \sigma, r, T) = S\Phi(d_1) - e^{-rT}K\Phi(d_2)$$
$$d_1 = \frac{\log(S/K) + (r + 0.5\sigma^2)T}{\sigma\sqrt{T}}, \quad \text{and} \quad d_2 = d_1 - \sigma\sqrt{T},$$

where Φ is the cumulative distribution function of a standard normally distributed variable. Equivalently, the call price is characterized as a solution of the so-called Black-Scholes PDE. We refer again to [14] for more details.

The Black-Scholes model is widely seen as the starting point of modern option pricing theory and thus the core of modern mathematical finance. Numerous papers have investigated if the model actually holds in reality and what the models shortcomings are. This analysis can be done from two different perspectives. The first approach is to investigate the time series of daily log returns $\log(S_{t+1}/S_t)$ over a longer time period. The Black-Scholes model predicts that $\log(S_{t+1}/S_t)$ is normally distributed, but unfortunately, the empirical distribution does not confirm these claims. For example, returns are not symmetric and large losses (negative returns) occur significantly more often then theoretically predicted. The true distribution of the log returns has fatter tails than a normal distribution. Moreover, one typically observes periods of lower and periods of higher volatility (so-called volatility cluster) which is not consisted with a constant parameter σ.

The second approach is to investigate option prices observed in the market. Assume we observe the price of a call option $\widehat{C}$ and the option's strike K, maturity T, initial stock price S_0 and the risk-free interest rate r. Then there is one unique σ such that $C_{BS}(S_0, K, T, r, \sigma) = \widehat{C}$. This is called the market *implied volatility* and plays a crucial role in finance. Its computation will be discussed in detail in Section 2.4. Taking two options on the same underlying that differ only in their maturity we expect to obtain the same implied volatility. In practice, this is not the case and their seems to be a time-dependence in the volatility. Unfortunately, even if we compare options with the same maturity and different strikes we observe different levels of implied volatility. This effect is often called the implied volatility smile, see [39]. Sometimes it is also called skew to indicate that the different volatilities across strikes are not symmetric.

Since the early seventies different types of stock price models have been developed

that tackle this problem. We will give a short overview and refer to some of the most relevant papers. A more comprehensive overview and more detailed explanations can be found in many standard textbooks on option pricing, see for example [101], [41] and [51].

- **Generalized Black-Scholes**: The simplest modification is to make the parameters and especially the volatility in the Black-Scholes model time dependent. This allows for a consistent evaluation of several options with the same strike but different maturities.

- **Local volatility models**: The next logical extension is to make the volatility function dependent on the price of the underlying stock. This leads to the class of so-called local volatility models where the stock price is modelled by

$$\frac{\mathrm{d}S_t}{S_t} = r_t \mathrm{d}t + \sigma(t, S_t)\mathrm{d}W_t$$

for a volatility function σ. One example is the constant elasticity of variance (CEV) model that sets $\sigma(t, S_t) = \sigma S_t^{(\beta-2)/2}$ a parameter β and the stock price is modelled by

$$\frac{\mathrm{d}S_t}{S_t} = r\mathrm{d}t + \sigma S_t^{(\beta-2)/2}\mathrm{d}W_t. \tag{2.39}$$

For $\beta = 2$ we obtain again the standard Black-Scholes model whereas from market data one typically obtains a $\beta < 2$. See [114] for more details.

The most important example of a local volatility model is the Dupire model introduced by [39]. This model can be perfectly calibrated to the option prices observed in the market at a fixed time point. However, the model is not able to capture the time dynamics of the volatility surface as stated in [67].

- **Stochastic volatility models**: A different approach towards the modelling of stock price and volatilities was introduced by [69]. In this paper, the author suggests to make the volatility stochastic and introduces a second Brownian motion. The model is described by the two SDEs

$$\begin{aligned} \frac{\mathrm{d}S_t}{S_t} &= r\mathrm{d}t + \sqrt{v_t}\mathrm{d}W_t^1 \\ \mathrm{d}v_t &= \kappa(\theta - v_t)\mathrm{d}t + \gamma\sqrt{v_t}\mathrm{d}W_t^2 \end{aligned} \tag{2.40}$$

where $\mathrm{d}W_t^1\mathrm{d}W_t^2 = \varrho\mathrm{d}t$. The resulting process for the variance v_t is mean-reverting with speed of mean-reversion κ, long-term mean θ and volatility of volatility γ. The introduction of stochastic volatility results in a more realistic behaviour of

the stock price process and reproduces better the empirically observed volatility clusters. In contrast to Dupire's model, the Heston model cannot be perfectly calibrated to the option price surface. A better fit can be achieved by the SABR model introduced in [67]. For more details on stochastic volatility models and the calibration to the volatility surface we refer to [51].

- **Models with jumps**: As mentioned above, large negative returns occur empirically significantly more often then theoretically predicted if we assume that the underlying risk factor is essentially a diffusion process. Even with stochastic volatility the tails of the resulting distribution are not fat enough. This can be changed if jumps are added to the model. [94] suggests to use a Poisson process in order to model jumps. The resulting SDE for the stock price process is given by

$$\frac{\mathrm{d}S_t}{S_t} = (r - \lambda\mathbb{E}[e^J - 1])\mathrm{d}t + \sigma\mathrm{d}W_t + (e^J - 1)\mathrm{d}X_t^P$$

  under the pricing measure $\mathbb{Q}$, see [101]. The process X_t^P is a Poisson process with intensity $\lambda > 0$ and jump size J normally distributed with mean α and volatility β. A large class of jump models are given by Lévy models that include jump-diffusion models and pure jump models that do not rely on a Brownian motion. See [41] for a detailed introduction to option pricing with jump processes.

- **More complex models**: The above mentioned extension and variations can be further combined to obtain more promising properties. The most relevant examples are the model of [8] that combines Merton's jump diffusion model with a stochastic volatility and the class of local stochastic volatility models. Other extension are for instance obtained by a combination of stochastic volatility with stochastic interest rates.

Note that the increasing complexity of the model also makes the computation of option prices more complicated. Typically extensions loose the advantage of a closed form solution that the Black-Scholes model admits. Instead, one has to use a numerical pricing routine to either compute the expectation of the payoff or solve the corresponding PDE. The more complex the model becomes, the more difficult it is to calculate accurate option prices in a short runtime. An overview on numerical methods for option pricing can for example be found in [54], [115] or [101].

2.3.2 Analyticity of parametric option prices

In this section, we look at parametric option prices and investigate their properties. With parametric option prices we refer to the function that maps certain parameters

(e.g. payoff and model parameters) to the corresponding option price, i.e.

$$Price : \mathcal{P} \ni \boldsymbol{p} = (\boldsymbol{p}_1, \boldsymbol{p}_2) \mapsto Price^{\boldsymbol{p}} = \mathbb{E}[g^{\boldsymbol{p}_1}(X^{\boldsymbol{p}_2})] \quad \text{for} \quad \mathcal{P} \subset \mathbb{R}^D$$

for a European option depending on a d-dimensional risk factor $X^{\boldsymbol{p}_2}$. This notation was introduced in [49]. A simple example would be the price of a call option in the Black-Scholes model given by

$$\mathcal{P} \ni (S_0, K, r, T, \sigma) \mapsto \mathbb{E}\big[e^{-rT}\big(S_0 e^{(r-0.5\sigma^2)T+\sigma W_T} - K\big)^+\big] \quad \text{with} \quad \mathcal{P} \subset [0, \infty)^5.$$

In this example we know that the call payoff has a kink and is not differentiable with respect to S_0. However, the call option price is differentiable and the first two derivatives with respect to S_0 (i.e. Delta and Gamma) exists. The Gaussian kernel in the model smoothes the kink. This is the motivation to investigate the smoothness of option prices as a function of the parameters more systematically. A suitable way of doing so is by investigating a Fourier type formulation of the option price, see [49].

The idea of using the fast Fourier transform (FFT) for option pricing has been introduced in [30]. Since then, a variety of numerical approaches that rely on the Fourier transform have been developed. They all have in common that the characteristic function of the underlying is required in closed form which is the case for many popular asset price models. The Fourier pricing formula which we will use in this section is based on [40]. The idea behind their Fourier pricing formula is the application of Parseval's theorem for the Fourier transform, see [138], which yields the following expression

$$\mathbb{E}[g(X)] = \int_{\mathbb{R}} g(x) f(x) \mathrm{d}x = \frac{1}{2\pi} \int_{\mathbb{R}} \widehat{g}(-z) \varphi(z) \mathrm{d}z$$

where $\widehat{g}$ is the Fourier transform of g given by

$$\widehat{g}(z) = \int_{\mathbb{R}} g(x) e^{izx} \mathrm{d}x$$

and φ is the characteristic function of the random variable X, i.e. the Fourier transform of the density f of X, given by

$$\varphi(z) = \mathbb{E}[e^{izX}] = \int_{\mathbb{R}} f(x) e^{izx} \mathrm{d}x.$$

For this formula to hold we require that the payoff and the characteristic function fulfil some integrability conditions. A standard call option payoff as a function of the log-asset price is however not integrable and Parseval's theorem cannot be applied directly. A straightforward solution is to multiply g with an exponential damping factor $\exp(\alpha x)$

for an appropriate $\alpha \in \mathbb{R}$.

For the multivariate case we follow [50] and [49]. Let $\mathcal{P} = \mathcal{P}^1 \times \mathcal{P}^2 \subset \mathbb{R}^D$ be the hyperrectangular parameter domain with $\mathcal{P}^1 \subset \mathbb{R}^{D_1}$, $\mathcal{P}^2 \subset \mathbb{R}^{D_2}$ and $\alpha \in \mathbb{R}^d$ the damping weight. Let $\mathbb{R}^d \ni z \mapsto \varphi^{\boldsymbol{p}_2}(z)$ be the characteristic function of the risk factors $X^{\boldsymbol{p}_2}$. Note that we consider two different dimensions, the dimension of the underlying asset price process d and the dimension of the parameter space D. For the Fourier pricing formula we assume the following integrability condition

$$\boldsymbol{x} \mapsto e^{\langle \alpha, \boldsymbol{x} \rangle} g^{\boldsymbol{p}_1}(\boldsymbol{x}) \in L^1(\mathbb{R}^d) \quad \text{for all} \quad \boldsymbol{p}_1 \in \mathcal{P}^1. \tag{Int}$$

Moreover, we impose the exponential moment condition

$$\mathbb{E}\big[e^{-\langle \alpha, \boldsymbol{X}^{\boldsymbol{p}_2} \rangle}\big] < \infty \quad \text{for all} \quad \boldsymbol{p}_2 \in \mathcal{P}^2. \tag{Exp}$$

Now we are in the position to introduce the Fourier pricing formula.

Proposition 5 (Fourier pricing formula)**.** *Let $\mathcal{P} = \mathcal{P}^1 \times \mathcal{P}^2 \subset \mathbb{R}^D$ be a hyperrectangular and $\alpha \in \mathbb{R}^d$ such that* (Int) *and* (Exp) *hold. Assume $\boldsymbol{z} \mapsto \widehat{g^{\boldsymbol{p}_1}}(-\boldsymbol{z} - i\alpha)\varphi^{\boldsymbol{p}_2}(\boldsymbol{z} + i\alpha)$ is in $L^1(\mathbb{R}^d)$ for all $\boldsymbol{p} = (\boldsymbol{p}_1, \boldsymbol{p}_2) \in \mathcal{P}$. Then we obtain*

$$Price^{\boldsymbol{p}} = \mathbb{E}\big[g^{\boldsymbol{p}_1}(\boldsymbol{X}^{\boldsymbol{p}_2})\big] = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \widehat{g^{\boldsymbol{p}_1}}(-\boldsymbol{z} - i\alpha)\varphi^{\boldsymbol{p}_2}(\boldsymbol{z} + i\alpha) d\boldsymbol{z}. \tag{2.41}$$

Proof. The proposition follows from Theorem 3.2 in [40]. □

For numerical purposes one can slightly modify the formula and integrate only over the real part of the integrand. From [50] we obtain

$$\int_{\mathbb{R}^d} \widehat{g^{\boldsymbol{p}_1}}(-\boldsymbol{z} - i\alpha)\varphi^{\boldsymbol{p}_2}(\boldsymbol{z} + i\alpha) \mathrm{d}\boldsymbol{z} = 2 \int_{\mathbb{R}_+ \times \mathbb{R}^{d-1}} \Re\Big(\widehat{g^{\boldsymbol{p}_1}}(-\boldsymbol{z} - i\alpha)\varphi^{\boldsymbol{p}_2}(\boldsymbol{z} + i\alpha)\Big) \mathrm{d}\boldsymbol{z}. \tag{2.42}$$

This integrand can now be truncated and then computed using a numerical quadrature technique. In this section, we use (2.41) to analyse the analyticity of parametric option prices. Following [49] we introduce two additional conditions on the Fourier transform of the payoff and the characteristic function. First, for the payoff we impose

$$\begin{aligned} &\forall\, \boldsymbol{z} \in \mathbb{R}^d \quad \boldsymbol{p}_1 \mapsto \widehat{g^{\boldsymbol{p}_1}}(-\boldsymbol{z} - i\alpha) \text{ is analytic in } \mathcal{B}(\mathcal{P}^1, \varrho_1), \text{for } \varrho_1 \in (1, \infty)^{D_1} \\ &\text{and } \exists\, c_1, c_2 > 0 \quad \text{s.t.} \quad \sup_{\boldsymbol{p}_1 \in \mathcal{B}(\mathcal{P}^1, \varrho_1)} |\widehat{g^{\boldsymbol{p}_1}}(-\boldsymbol{z} - i\alpha)| \leq c_1 e^{c_2 |\boldsymbol{z}|} \text{ for all } \boldsymbol{z} \in \mathbb{R}^d \end{aligned} \tag{Ana1}$$

and for the characteristic function we impose

$$\begin{aligned}&\forall\ \boldsymbol{z} \in \mathbb{R}^d \quad \boldsymbol{p}_2 \mapsto \varphi^{\boldsymbol{p}_2}(\boldsymbol{z} + i\alpha) \text{ is analytic in } \mathcal{B}(\mathcal{P}^2, \varrho_2), \text{for } \varrho_2 \in (1,\infty)^{D_2} \text{ and}\\ &\exists\ c_1, c_2 > 0,\ \beta \in (1,2] \quad \text{s.t.} \quad \sup_{\boldsymbol{p}_2 \in \mathcal{B}(\mathcal{P}^2,\varrho_2)} |\varphi^{\boldsymbol{p}_2}(\boldsymbol{z} + i\alpha)| \leq c_1 e^{-c_2|\boldsymbol{z}|^\beta} \text{ for all } \boldsymbol{z} \in \mathbb{R}^d.\end{aligned} \tag{Ana2}$$

This allows us to introduce the following theorem on the analyticity of option prices.

Theorem 10 (Analyticity of parametric option prices)**.** *Let $\mathcal{P} = \mathcal{P}^1 \times \mathcal{P}^2 \subset \mathbb{R}^D$ be a hyperrectangle, $\alpha \in \mathbb{R}^D$ and $\varrho = (\varrho_1, \varrho_2) \in (1,\infty)^D$ such that the conditions* (Int) *and* (Exp) *as well as* (Ana1) *and* (Ana2) *hold. Then the function $\mathcal{P} \ni \boldsymbol{p} \mapsto Price^{\boldsymbol{p}}$ has an analytic extension to the Bernstein ellipse $\mathcal{B}(\mathcal{P}, \varrho)$.*

Proof. See the proof of Theorem 3.2 in [49]. □

The structure of conditions allows us to investigate payoff functions and models separately. The conditions (Int) and (Ana1) are satisfied for most standard payoff functions such as of calls, puts and digitals as a function of the log-asset price or strike. For example, if we choose $\alpha < -1$, the Fourier transform of the call option payoff $g(x) = (e^x - K)^+$ is given by

$$\widehat{g}(z) = \frac{K^{iz+1+\alpha}}{(iz+\alpha)(iz+1+\alpha)} \quad \text{for} \quad z \in \mathbb{R}.$$

The corresponding conditions on the characteristic function (Exp) and (Ana2) hold in a large class of asset price models and for most parameters. We refer to [48] (an earlier and longer version of [49]) and the references therein as well as [47] and [107] for different examples of payoffs and models. As an example, the price of a call or put option in the Black-Scholes model is analytic for the parameters $(S_0, K, r, T, \sigma) \in (0,\infty)^5$.

2.3.3 Chebyshev interpolation in option pricing

So far, we have seen that option prices are in many cases analytic functions of their parameters. This means we can interpolate them using only a few Chebyshev nodal points. Following [49] we obtain the following Chebyshev interpolation of the option price

$$Price^{\boldsymbol{p}} \approx \sum\nolimits_{\boldsymbol{j}} c_{\boldsymbol{j}} T_{\boldsymbol{j}}(\boldsymbol{p})$$

using the multivariate Chebyshev interpolation as introduced in Section 2.2.1. From the convergence results for the (tensor based) Chebyshev interpolation we conclude that the

approximation error should theoretically decay (sub)exponentially fast. This means we can use Chebyshev interpolation to approximate the price function using an existing numerical pricing routine and then evaluate the interpolation instead of calling the original pricing routine. The resulting method looks as follows:

- *Problem:* Existing pricer $Price^{\boldsymbol{p}}$ for parameters $\boldsymbol{p} \in \mathbb{R}^D$ should be approximated using Chebyshev interpolation.
- *Interpolation domain:* Fix a hyperrectangular interpolation domain

$$\mathcal{P} := [\underline{p}_1, \overline{p}_1] \times \ldots \times [\underline{p}_D, \overline{p}_D] \subset \mathbb{R}^D$$

  and $\boldsymbol{N} = (N_1, \ldots, N_D) \in \mathbb{N}^D$. Compute nodal points $\boldsymbol{p_k} = \tau_{\mathcal{P}}(\boldsymbol{z_k})$ for the multivariate Chebyshev points $\boldsymbol{z_k} = (z_{k_1}, \ldots, z_{k_D})$ with $z_{k_i} = \cos(k_i \pi / N_i)$, $i = 0, \ldots, N_i$.
- *Set-up interpolation:* Calculate prices at the Chebyshev nodes $Price^{\boldsymbol{p_k}}$ for all $\boldsymbol{p_k}$. Compute and store Chebyshev coefficients

$$\begin{aligned} c_{\boldsymbol{j}} &= \prod_{i=1}^{D} \Big(\frac{2^{\mathbb{1}_{0<j_i<N_i}}}{N_i} \Big) \sum\nolimits_{\boldsymbol{k}} Price^{\boldsymbol{p_k}} T_{\boldsymbol{j}}(\boldsymbol{z_k}) \\ &= \prod_{i=1}^{D} \Big(\frac{2^{\mathbb{1}_{0<j_i<N_i}}}{N_i} \Big) \sum_{k_1=0}^{N_1} \ldots \sum_{k_D=0}^{N_D} Price^{\boldsymbol{p_k}} \prod_{i=1}^{D} T_{j_i}(z_{k_i}). \end{aligned}$$

- *Pricing*: For $\boldsymbol{p} \in \mathcal{P}$, the option price can be computed as

$$Price^{\boldsymbol{p}} \approx \sum_{\boldsymbol{j}} c_{\boldsymbol{j}} p_{\boldsymbol{j}}(\boldsymbol{p}) = \sum_{j_1=0}^{N_1} \ldots \sum_{j_D=0}^{N_D} c_{(j_1,\ldots,j_d)} \prod_{i=1}^{D} T_{j_i}(\tau^{-1}_{[\underline{p}_i, \overline{p}_i]}(p_i)).$$

The second and the third step can be seen as the offline phase and the fourth step, the actual pricing, as the online phase of the approach. Only the coefficients $c_{\boldsymbol{j}}$ and the lower and upper bounds of the domain have to be stored after the offline phase. The presented framework is very general and can be applied to options that are not in the scope of the theoretical analysis in the previous section.

When analysing the accuracy of the Chebyshev interpolation one has to take into account that a numerical pricing routine is used in order to compute $Price^{\boldsymbol{p_k}}$. The overall error will therefore depend on the error between the true values $Price^{\boldsymbol{p_k}}$ and the computed values $\widehat{Price}^{\boldsymbol{p_k}}$, the so-called distortion error. We cannot expect the Chebyshev method to be more accurate than the reference pricer. The distortion error can be either deterministic and bounded in absolute terms or the error can be stochastic

and we know its distribution. The first one occurs when a method such as a PDE solver or a numerical quadrature is used and the second one occurs if the values at the nodal points are calculated via Monte Carlo simulation.

Let $\varepsilon_{\boldsymbol{k}} = Price^{\boldsymbol{p_k}} - \widehat{Price^{\boldsymbol{p_k}}}$ be the error at the grid points. As in [49] we assume that either $|\varepsilon_{\boldsymbol{k}}| \leq \bar{\varepsilon}$ or that $\varepsilon_{\boldsymbol{k}}$ is normally distributed with $\mathcal{N}(0, \sigma_{\boldsymbol{k},M})$. Here M is the number of simulations of the corresponding Monte Carlo simulation. We define the error bound

$$\varepsilon^\star(\boldsymbol{N}) := \begin{cases} \bar{\varepsilon} \\ \sqrt{2\log\left(2\prod_{i=1}^{D}(N_i+1)\right)}\max_{\boldsymbol{k}}\sigma_{\boldsymbol{k},M}. \end{cases}$$

From [49] we obtain the following convergence results for expected error in the maximum norm for analytic option prices with distortion at the nodal points. The following theorem can be seen as an extension of Proposition 4.

Theorem 11. *Let $\mathcal{P} \subset \mathbb{R}^D$ a hyperrectengular. Assume $\mathcal{P} \ni \boldsymbol{p} \mapsto Price^{\boldsymbol{p}}$ has an analytic extension to some generalized Bernstein ellipse $\mathcal{B}(\mathcal{P}, \varrho)$, $\varrho \in (1,\infty)^D$ and the price function is bounded on this ellipse. Assume that $|\varepsilon_{\boldsymbol{k}}| \leq \bar{\varepsilon}$ or $\varepsilon_{\boldsymbol{k}} \sim \mathcal{N}(0, \sigma_{\boldsymbol{k},M})$. Then we have*

$$\mathbb{E}\Big[\max_{\boldsymbol{p}\in\mathcal{P}}\Big|Price^{p} - \sum_{\boldsymbol{j}} c_{\boldsymbol{j}} p_{\boldsymbol{j}}(\boldsymbol{p})\Big|\Big] \leq C\varrho^{-\underline{N}} + \Lambda_{\boldsymbol{N}}\varepsilon^\star(\boldsymbol{N}), \tag{2.43}$$

for $C > 0$, $\underline{N} = \min_i N_i$ and $\Lambda_{\boldsymbol{N}}$ is the Lebesgue constant of the multivariate Chebyshev interpolation.

Proof. See Theorem 2.5 of [49]. □

The first part of the error bound is the classical Chebyshev interpolation error and the second term is the distortion error. In the deterministic case we observe directly that the overall error level cannot be better than the maximal error at the nodal points multiplied by the Lebesgue constant. From the analyticity it follows that all derivatives exist and the error results of the Chebyshev interpolation yield that the derivative of the interpolant converges against the derivative of the option price. From a practical perspective, this means that the Chebyshev method delivers the option's sensitivities, i.e. the Greeks such as Delta, Gamma and Vega as well when we interpolate in the initial stock price or the volatility.

Advantages of Chebyshev for parametric option pricing

The use of Chebyshev interpolation for parametric option pricing can yield significant performance improvements since the evaluation of a Chebyshev interpolant is often significantly faster than calling a pricer. Note that with "pricer" we refer to a numerical pricing routine such as a PDE solver, a numerical quadrature or a Monte Carlo simulation. The presented type of approximation becomes interesting when the same pricer has to be called for a large number of different input parameters.

Moreover, the approach admits a useful offline/online decomposition: The Chebyshev interpolation can be prepared in an offline step and only the coefficients are stored. In the online step, a real-time evaluation of option prices is possible instead of the slow evaluation of the original pricer. The slower the original pricer is, the higher is the possible gain in efficiency in the online step. See the numerical experiments in [49] and [91] for a detailed investigation of the potential of Chebyshev interpolation for parametric option pricing.

The merits of the presented approach are that it can be integrated into an existing pricing library, it is a very simple approach to speed-up calculations, polynomials are well understood numerical objects and an extensive error analysis as well as convergence results are available.

Interesting applications of Chebyshev interpolation are for example the calibration of options to market data, credit exposure calculations and more general risk management of trading books. Moreover, there are further applications of Chebyshev interpolation in option pricing that go beyond a straightforward interpolation in parameters. These approaches are still able to exploit many of the promising properties of Chebyshev interpolation and deliver good results. A first application of Chebyshev interpolation is the approximation of the implied volatility, the inverse of the call option price function, in Section 2.4. A second application and core of the thesis is then the use of Chebyshev interpolation in a dynamic programming framework and its possible applications and extensions. We call the approach of [49] the static Chebyshev method and the new approach the dynamic Chebyshev method.

Example: Calibration to spread options

In [55], we present an example of a possible application of Chebyshev in finance, the calibration of spread options. This example uses the *chebfun3* package and has been uploded to *www.chebfun.org*. In this toy example, we look at the calibration of the correlation parameter in a bivariate Black-Scholes model to a set of spread options on two underlyings. For the calibration, the distance between the market prices and model prices

for different strikes and maturities has to be minimized over the correlation parameter ϱ. The numerical optimization requires repeated calls of the pricer for each of the option and different values of ϱ. The nature of the task makes it an interesting application of Chebyshev interpolation, since speed is crucial but at the same time the parameter fitting should also be accurate.

Our idea is to interpolate the price of the spread option as a function of the strike K, the maturity T and the correlation ϱ using Chebyshev interpolation and *chebfun3*. As mentioned earlier, in *chebfun3* the curse of dimensionality of a full tensor based interpolation is avoided by using low-rank approximations. We show that the evaluation of the resulting interpolation is significantly faster than the original pricer. In our example, the pricing using the Chebyshev interpolant was more than 250 times faster with a maximal pricing error of 0.03%. This gain in speed is then explored in the calibration routine.

Challenges and limitations of Chebyshev in finance

The main challenge for the presented static Chebyshev method is the dimensionality of the problem, i.e. the number of parameters in which we want to interpolate. Note that this definition of dimensionality might differ from the dimensionality of the underlying stochastic process. In the previous example, we considered a bivariate model but our interpolation was in three different parameters. In this section, we have therefore used the notation d for the number of assets and D for the dimensionality of the interpolation. A straightforward application of tensor based interpolation leads to an exponential increase of the number of nodal points in the dimension. As mentioned earlier, this curse of dimensionality can be tackled by techniques such as sparse grids and low-rank approximations, see [57] for an application of the latter. However, even with these dimension reduction techniques, high dimensional interpolations are challenging. Similarly, the size of the interpolation domain has a significant impact on the number of nodal points. Especially in multivariate dimensions, a careful choice of the interpolation domain is crucial. In some applications it is better to split a larger domain into several subdomains.

The second major limitation of the static Chebyshev method is that it still requires repeated calls of a pricer in order to compute the values at the Chebyshev nodes. It is therefore only beneficial when the number of option prices that have to be calculated are an order of magnitude higher than the number of nodal points or if one can really explore the offline/online decomposition. Another limitation of the Chebyshev interpolation is that it is not shape preserving, i.e. the interpolant does not inherit the properties of the (pricing) function. For example, if we interpolate a monotonically increasing function, the interpolant might not be monotonically increasing. One can modify the Chebyshev

interpolation to be shape-preserving, see [26], but this comes at additional computational costs. However, the fast convergence in the maximum norm guarantees that the static Chebyshev method is almost shape preserving. This is sufficient for most practical applications.

2.4 Implied volatility

In this section, we consider one particular application of Chebyshev interpolation in finance, the efficient computation of the so-called Black-Scholes implied volatility. This problem is highly relevant for practitioners but is also interesting from an academic perspective.

This section is based on joint work with Kathrin Glau, Paul Herold and Dilip B. Madan and the presented results have been published in our paper "The Chebyshev method for the implied volatility". The paper builds in parts on the master thesis of [68]. The results that are discussed in this section are only the ones to which the author has contributed.

2.4.1 Motivation

The Black-Scholes *implied volatility* is one of the most important quantities in finance. The one parameter in the Black-Scholes model that cannot be observed using market data is the volatility of the underlying asset process. The Black-Scholes call price function is strictly monotone increasing in volatility and [92] show that under no-arbitrage assumptions there exists always a unique (positive) volatility such that the model price equals the observed market price. This unique volatility is called the Black-Scholes implied volatility.

The implied volatility can be seen as a universal language in the daily business of trading, hedging, model calibration and more generally in risk management. Usually, option prices are quoted in terms of implied volatilities instead of absolute prices. For high frequency trading in particular, very accurate real-time evaluations of the implied volatility are required for large data sets. As stated in [10] and [111] in practice, often millions of option prices have to be inverted in real-time for instance by large data providers. For more details on the use of the implied volatility we refer to [56] and the references therein.

Unfortunately, the solution of this inverse problem is not available in an explicit form and thus a numerical approximation method is required. Due to importance of the implied volatility in the financial industry an efficient numerical solution is crucial.

From a practitioner's perspective a suitable numerical method needs to fulfil the following requirements

- *large domain of input variables*, i.e. options with very low or high volatilities as well as options with moneyness varying from far out of the money to deep in the money,
- *high efficiency* for a given requirement in terms of accuracy and speed.

Moreover, it is desirable if the method additionally delivers

- *real-time evaluations* of the implied volatility even for very large data sets,
- *closed-form* solutions with accessible derivatives,
- *easy implementation and maintenance.*

Due to the high relevance of the implied volatility and the different computational challenges arising in the computation, the problem is an ideal candidate to show the potential of Chebyshev polynomial interpolation in finance.

In the last 40 years, different methods have been introduced to tackle this problem in the academic literature. The proposed methods can be divided into two main classes, iterative root-finders and non-iterative approximation methods. In the Black-Scholes model, the implied volatility depends on the observable parameters S_0, K, T, r and the option premium C. It can be calculated as the root of the function

$$\sigma \mapsto C^{BS}(S_0, K, T, r, \sigma) - C^{Mkt}$$

where $C^{BS}(S_0, K, T, r, \sigma)$ is the model price of a call option in the Black-Scholes model and C^{Mkt} is the observed market price for a pair of strike K and maturity T. The implied volatility can thus be calculated using a (classical) numerical root finder. [92] provide a possible starting value which ensures that the Newton-Raphson root-finding algorithm will always converge and return the correct implied volatility. This straightforward approach is simple and easy to implement. However, for some parameter constellations the number of iterative steps increases significantly and the method becomes slow. Other root-finders such as a Brent-Dekker algorithm or a bisection are also possible but typically less efficient.

The second class of non-iterative methods aim to approximate the implied volatility as a functions of the parameters S_0, K, T, r and the premium C. The computational effort to approximate a function depending on five variables is challenging. Fortunately,

we can reduce the dimensionality as stated in [75] amongst others by normalizing the call price

$$c = \frac{C(S_0, K, T, r, \sigma)}{\sqrt{S_0 e^{-rT} K}}.$$

The normalized call price c is a bivariate function given by

$$\begin{aligned} c(x, v) &= e^{\frac{x}{2}} \Phi\left(\frac{x}{v} + \frac{v}{2}\right) - e^{-\frac{x}{2}} \Phi\left(\frac{x}{v} - \frac{v}{2}\right) \text{ with} \qquad (2.44) \\ x &= \log(S_0 e^{rT}/K) = rT + \log(S_0/K) \\ v &= \sigma\sqrt{T} \end{aligned}$$

where x measures the *moneyness* (the option is out of the money if $x < 0$, at the money if $x \approx 0$ and in the money if $x > 0$) and v corresponds to the *time-scaled volatility*. Furthermore, call prices of in the money options can be expressed by those of out of the money options, namely

$$c(-x, v) = c(x, v) + e^{-\frac{x}{2}} - e^{\frac{x}{2}}. \qquad (2.45)$$

Hence the parameter domain can be reduced to $x \leq 0$ and consequently the call price is normalized to values in $[0, 1]$. To calculate the implied volatility σ for a call price C it is thus sufficient to solve Equation (2.44) for v using the normalized call price c. Overall, the approximation of the implied volatility $v(x, c)$ becomes a bivariate (interpolation) problem in the moneyness x and the normalized call price c.

Examples of such non-iterative approaches are the rational approximation methods of [86] and [111]. These methods can be very fast but unfortunately the domain for which they set up the approximation is very restrictive and excludes option prices that occur in practice.

More recently, [75] proposed a more sophisticated method that combines the two types of approaches. [75] explores the limit behaviour of the call price function and uses rational approximation for the initial guess and then two iterative steps of Householder's method to achieve a very high accuracy. For a more detailed literature review we refer again to [56] and the references therein.

We propose a new method for the implied volatility using bivariate Chebyshev interpolation. The efficiency of such an approach depends critically on the choice of the interpolation domain. As pointed out in [68] and [56], a straightforward implementation of the Chebyshev interpolation might yield poor results if the chosen domain is too

large. On the other hand, a small domain makes the method impractical because not all parameter constellations occurring in the market are covered. We can overcome this challenge by splitting a sufficiently larg domain into subdomains and apply suitable scaling functions. This approach is motivated by the choice of the approximation domains in [75].

In the following, we will first show that the domain proposed by [86] is too restrictive and then fix a suitable interpolation domain for our approximation method. Then we will tailor the bivariate Chebyshev interpolation to the problem and explain the algorithmic structure of the presented approach. We conclude the section with a numerical comparison of the new approach and two benchmark methods, a Newton-Raphson root finder and the method of [75].

2.4.2 Interpolation domain

In this section, we define a suitable interpolation domain for our method based on market data. Then we introduce a splitting of the domain resulting in four subdomains which allow for a better interpolation. For more details we refer to [68].

Choice of the interpolation domain

To find an appropriate interpolation domain, we investigate option data of the DAX, the EURO STOXX 50, the Standard & Poor's 500 (S&P 500) and the VIX index from *Thomson Reuters Eikon*. For all options with non-zero trading volume we compute the forward moneyness x and the time-scaled volatility $\sigma\sqrt{T}$. As a comparison, we check if the resulting parameters are covered by the domain of Li. Figure 2.5 illustrates the option parameters for all four indices. For all four indices we observe that a relevant part of the options is not covered by the domain of Li. We observe moneyness between -1.5 and 2 as well as time-scaled volatilities up to 1. In different markets or under different market conditions one can expect to observe even more extreme option parameters. For example volatilities become considerably higher during a financial crisis. This motivates us to set up a Chebyshev interpolation of the implied volatility on a significantly larger domain which covers all relevant option data.

Domain splitting and scaling

To derive an approximation of the implied volatility on a sufficiently large domain, we further inspect the normalized call price. The implied volatility is not analytic at $c(x) = 0$ and $c(x) = e^{\frac{x}{2}}$. Therefore the maximal possible interval needs to be restricted to $0 < v_{min}(x) < v_{max}(x) < \infty$ with corresponding call prices $0 < c_{min} < c_{max} < e^{\frac{x}{2}}$. This assumption is not restrictive if the chosen v_{min} is small enough. Extending the

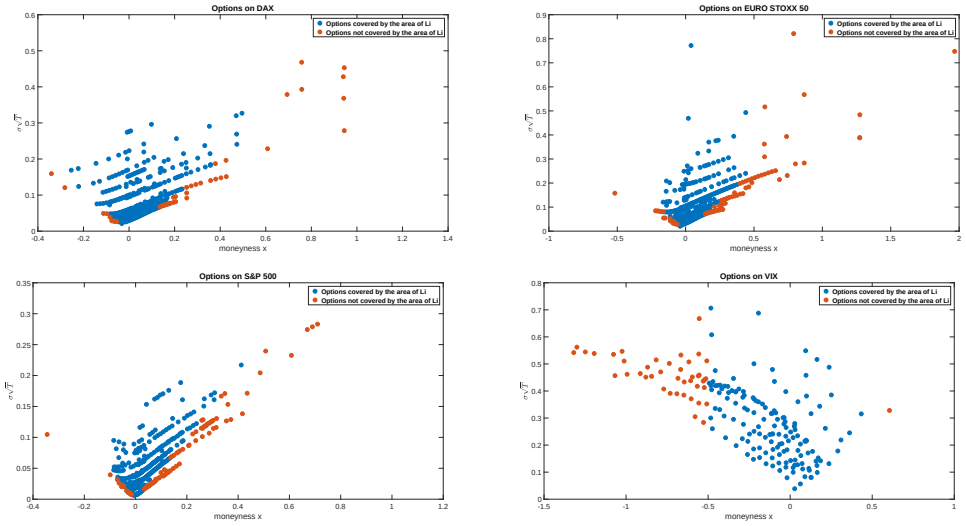

Figure 2.5: Moneyness x and time-scaled volatility $\sigma\sqrt{T}$ of options on four different indices. We only considered options with positive trading volume.

domain towards the maximal interval decreases the rate of convergence. To reduce this impact, we exploit the limit behaviour of the normalized call price. Figure 2.6 shows for a fixed moneyness x the normalized call price as a function of the time-scaled volatility. We observe that the call price is flat for very low as well as very high volatilities and almost linear around the point of inflection. This motivates us to split the domain into three parts depending on the call price

$$D_1 := [c_{min}(x), c_1(x)], \quad D_2 := [c_1(x), c_2(x)], \quad D_3 := [c_2(x), c_{max}(x)] \tag{2.46}$$

with corresponding volatilities $0 < v_{min}(x) < v_1(x) < v_2(x) < v_{max}(x)$, i.e. $c_{min}(x) = c(v_{min}(x), x)$ and so on. The idea of splitting the domain is based on the method of [75].

For each domain we will tailor a bivariate Chebyshev interpolation. Where call prices are flat its inverse becomes very steep. Hence, a direct polynomial interpolation is not well-suited. Fortunately, by exploiting the asymptotic behaviour of the call price function, we resolve the problem. On each interval, we define a scaling function $\phi_{i,x} : D_i \to [-1, 1]$ for $i \in \{1, 2, 3\}$ which transforms the call price to $[-1, 1]$ for each $x \in [x_{min}, x_{max}]$. For the resulting functions $\tilde{v} : [-1, 1]^2 \to \mathbb{R}$, $(\tilde{c}, \tilde{x}) \mapsto v(c, x)$ with $x = \varphi^{-1}(\tilde{x})$ and $c = \phi_{i,x}^{-1}(\tilde{c})$ for $i \in \{1, 2, 3\}$ where φ is the linear scaling as defined in (2.22). For a given call price c and moneyness $x \leq 0$ the implied volatility can then be approximated by

$$v(c, x) \approx I_i^{N_1^i, N_2^i}(\phi_{i,x}(c), \varphi(x)) \text{ where } i \text{ satisfies } c \in D_i$$

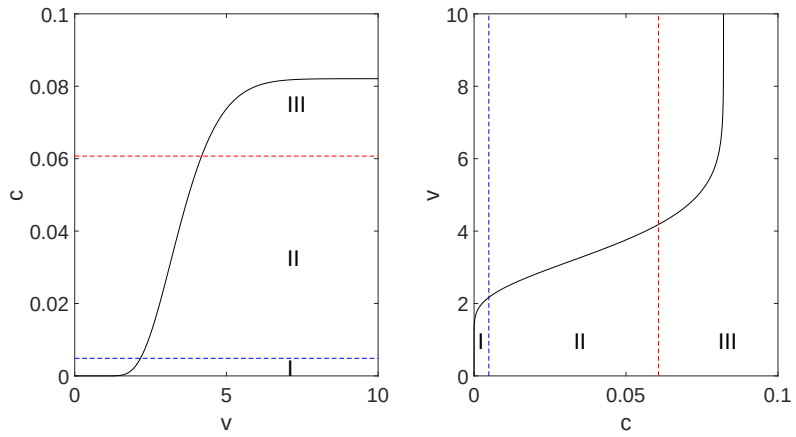

Figure 2.6: Splitting of the normalized call price (c) depending on the time-scaled volatility (v) and its inversion for $x = -5$ into three parts.

where $I_i^{N_1^i,N_2^i}$ is the bivariate Chebyshev interpolation of degree (N_1^i, N_2^i) on domain $i = 1, 2, 3$. The derivation of the scaling functions $\phi_{i,x}$ and the domains D_i can be found in the Appendix A.

Error analysis

The following theorem is the theoretical foundation of the high efficiency of the proposed approximation method. Thanks to the analyticity of the Black-Scholes call price and the scaling functions, we gather that the implied volatility function is analytic. We know that for such a function the convergence of the bivariate Chebyshev interpolation is sub-exponential in the number of nodal points.

Theorem 12. *Let $\phi_i^{-1}(\tilde{c}, \tilde{x})$ be analytically continuable to some open region around $[-1, 1]^2$ and let $0 < \phi_i^{-1}([-1, 1], x) < e^{\frac{x}{2}}$ for each $x \in [-1, 1]$. Then there exist constants $\rho_1, \rho_2 > 1, V > 0$ such that $\tilde{v}(\tilde{c}, \tilde{x}) := v(\phi_i^{-1}(\tilde{c}, \tilde{x}), \phi_x^{-1}(\tilde{x}))$ is analytic and for its bivariate Chebshev interpolation $I_i^{N_1^i,N_2^i}(\tilde{c}, \tilde{x}) := \sum_{j=0}^{N_1^i} \sum_{k=0}^{N_2^i} a_{jk} T_j(\tilde{c}) T_k(\tilde{x})$ holds*

$$\max_{(\tilde{c},\tilde{x})\in[-1,1]^2} |\tilde{v}(\tilde{c}, \tilde{x}) - I_i^{N_1^i,N_2^i}(\tilde{c}, \tilde{x})| \leq 4V \left(\frac{\rho_1^{-2N_1^i} + \rho_2^{-2N_2^i}}{(1 - \rho_1^{-2})(1 - \rho_2^{-2})} \right)^{\frac{1}{2}}.$$

Proof. The proof of the theorem can be found in [68]. □

From the analyticity of the implied volatility function follows that it is infinitely often continuously differentiable in each area and in this case the Chebyshev interpolation also approximates all derivatives, see Theorem 8. This can be exploited when the implied volatility is used in a gradient-based optimization routine.

We can enhance the efficiency even further by exploiting the low-rank structure of the bivariate functions. To do so, in our implementation we use the *chebfun2* algorithm based on [127], see Section 2.2.2 for a brief description of the *chebfun2* algorithm.

2.4.3 Algorithmic structure

In this section, we discuss the algorithmic structure of our method and provide details on the implementation. The algorithms are listed as pseudocode in the Appendix A.

As a starting point for the approximation of the implied volatility function, we split the interpolation domain into four different areas, see Figure 2.7. For each area, we approximate the implied volatility by a separate bivariate Chebyshev interpolation using the corresponding scaling function $\phi_{i,x}$ in c. For the sake of a lucid presentation, we list the different areas and transformations below. A list of all functions required for the implementation are listed as pseudocode in Algorithm 8 in the appendix.

Area I: For $x \in [-5, -0.0348]$ and $c \in [c_{min}(x), c_1(x)]$ we have

$$\phi_{1,x}(c) := 2\frac{\tilde{\phi}_1(c) - \tilde{\phi}_1(c_{min}(x))}{1 - \tilde{\phi}_1(c_{min}(x))} - 1.$$

Area I': For $x \in [-0.0348, 0]$ and $c \in [c_{min}(x), c_1(x)]$ we again use transformation $\phi_{1,x}(c)$.

Area II: For $x \in [-5, 0]$ and $c \in [c_1(x), c_2(x)]$ we have

$$\phi_{2,x}(c) := 2\frac{c - c_1(x)}{c_2(x) - c_1(x)} - 1.$$

Area III: For $x \in [-5, 0]$ and $c \in [c_1(x), c_{max}(x)]$ we have

$$\phi_{3,x}(c) := \frac{2\tilde{\phi}_3(c)}{\tilde{\phi}_3(c_{max}(x))} - 1.$$

The call prices $c_{min}(x), c_1(x), c_2(x)$ and $c_{max}(x)$ correspond to the volatilities

$$v_{min}(x) = 0.001 - 0.03x, \quad v_1(x) = 0.25 - 0.4x, \quad v_2(x) = 2 - 0.4x, \quad v_{max}(x) = 6.$$

Moreover, we replace the boundary call prices $c_1(x)$, $c_2(x)$ and $c_{max}(x)$ by univariate interpolations to reduce the runtime further. The evaluation of $c_{min}(x)$, however, is done directly, since for low volatilities the call price is hard to approximate. For this step we

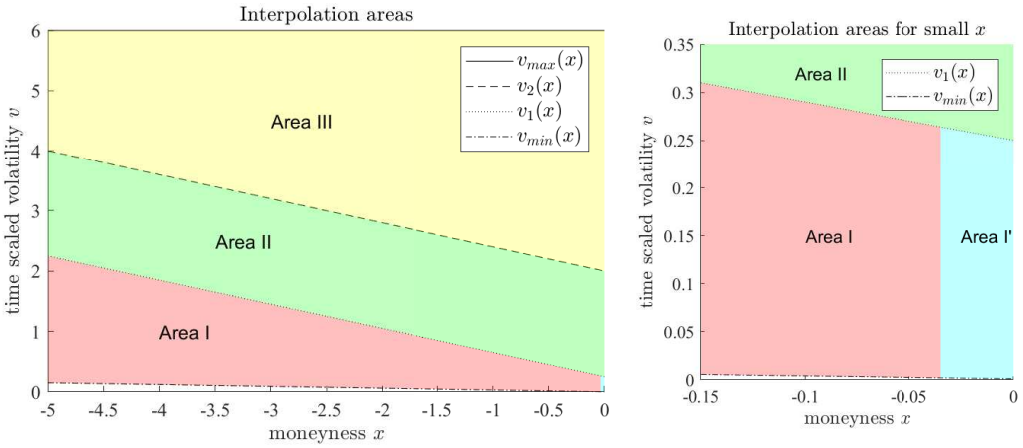

Figure 2.7: The four different interpolation areas of the Chebyshev method.

can use the implementation of the call price function provided in [75], which is of very high precision.

Online/offline decomposition

Our method can be split into an offline phase that has only to be done once and an online phase.

- *offline-phase* (preparation):
  In each area, we compute the implied volatilities on an $N \times N$ grid of Chebyshev points. Then we apply the *chebfun2* algorithm with pre-specified accuracy and obtain a low-rank approximation. For the calculation of the implied volatility at the grid points one can either use Jäckel's method (see Appendix A, Algorithm 9) or an iterative root finder (see Algorithm 10). For example the bisection method is well suited to compute very accurate implied volatilities at the nodal points. We were able to reach an accuracy in the region of 10^{-13} with this approach.

- *online-phase* (real-time evaluation):
  In the online phase implied volatilities are computed from real-time data, containing a vector of call prices $C \in \mathbb{R}^n$ and the corresponding strikes $K \in \mathbb{R}^n$, spot prices $S_0 \in \mathbb{R}^n$, maturities $T \in \mathbb{R}^n$ and interest rates $r \in \mathbb{R}^n$. Algorithm 11 in Appendix A provides a pseudocode version for the online step.

  – *Normalization*: We calculate the normalized call price c and the forward moneyness x from the data. Option prices with $x > 0$ need to be transformed to prices with moneyness $-x$ by Formula (2.45).

  – *Splitting*: For each pair (x, c), we need to find the corresponding area. As the

computation of $c_{min}(x)$ requires the most computational effort, we proceed as follows. First, we compute $c_{max}(x)$ and check if $c \leq c_{max}(x)$. Next, we check if $c < c_2(x)$ and eventually $c < c_1(x)$. Only in the latter case, do we compute $c_{min}(x)$ and check whether $c \geq c_{min}(x)$.

- *Transformation*: We compute the transformed call prices $\phi_{i,x}(c)$ and moneyness $\phi_x(x)$ with the respective transformations.

- *Evaluation*: We evaluate the bivariate Chebyshev interpolations provided in the *offline-phase* at the transformed call prices and moneyness to obtain the time-scaled implied volatility.

The runtime of the *online*-phase is primarily determined by the *splitting* and the *evalutation*-phase. The evaluation of the bivariate interpolations can be done in different ways and can be performed in very few computational steps depending on the required accuracy.

In the offline phase, the $chebfun2$ algorithm returns in each area a $chebfun2$-object which is a Chebyshev interpolant in low rank form $I^{N_1,N_2} = \sum_{j=1}^{k} d_j c_j(y) r_j(x)$ where $r_j(x)$ and $c_j(y)$ are univariate Chebyshev interpolations of size N_1 and N_2. One can now either store the four $chebfun2$-objects directly and use them in the online evaluation (as presented in Algorithm 11) or one can extract the coefficients and evaluate the low rank approximation manually. The first choice is simpler and for many applications sufficient, the latter can be slightly faster when it is implemented in an efficient way. For the second choice one can use Clenshaw's algorithm to evaluate the polynomials c_j and r_j.

Depending on the application, one can use different accuracies in the offline phase. Table 2.4.3 displays the ranks k and the grid sizes N_1, N_2 of the low rank interpolation operator for the three specified accuracies 10^{-6} (low accuracy), 10^{-9} (medium accuracy) and 10^{-12} (high accuracy). As expected, the ranks and grid sizes are higher for a higher accuracy. Moreover, we observe that we need more interpolation nodes in Area I and Area I' to obtain the same level of accuracy as in Area II and Area III.

Area	low accuracy	medium accuracy	high accuracy
Area I	$k = 10$, $N_1 = 25$, $N_2 = 36$	$k = 16$, $N_1 = 46$, $N_2 = 79$	$k = 22$, $N_1 = 67$, $N_2 = 122$
Area I'	$k = 9$, $N_1 = 27$, $N_2 = 18$	$k = 16$, $N_1 = 51$, $N_2 = 39$	$k = 23$, $N_1 = 77$, $N_2 = 57$
Area II	$k = 6$, $N_1 = 21$, $N_2 = 20$	$k = 11$, $N_1 = 36$, $N_2 = 33$	$k = 14$, $N_1 = 51$, $N_2 = 47$
Area III	$k = 5$, $N_1 = 11$, $N_2 = 9$	$k = 7$, $N_1 = 17$, $N_2 = 14$	$k = 9$, $N_1 = 23$, $N_2 = 19$

Table 2-A: Rank k and grid sizes N_1, N_2 of the low rank Chebyshev interpolation in the different areas for three different levels of pre-specified accuracy.

2.4.4 Numerical Results

In this section we present a numerical investigation of the Chebyshev method for the implied volatility. We compare our approximation method to

- the method of [75],
- the approximation formula given in [86],
- the approximation formula given in [86] with the proposed polishing of two Newton-Raphson iterations,
- the Newton-Raphson algorithm with the starting point given in [92]. The algorithm terminates if $|v_n - v_{n-1}| < 10^{-6}$.

In order to do so, we first choose a domain $\mathcal{D}_1$ on which all methods can be applied and compare the resulting errors and runtimes. On the complete domain $\mathcal{D}_2$, we compare the proposed method to the [75] method and the Newton-Raphson algorithm as those are the only ones that can also be applied on this set. Finally, we include actual market data. All codes are written in *Matlab* R2014a and the experiments are run on a computer with Intel Xeon CPU with 3.10 GHz with 20 MB SmartCache. We refer to [56] for a more detailed presentation of the results and some additional plots.

Comparison on Domain $\mathcal{D}_1$

The domain on which all methods work is the domain of [86] bounded below by $v_{min}(x)$, i.e.

$$\mathcal{D}_1 := \left\{ -0.5 \leq x \leq 0.5, 0 \leq v \leq 1, \max\left(\frac{|x|}{2}, v_{min}(-|x|)\right) \leq v \right\}$$

See Figure 2.8 for a comparison of the domain of [86] and the domain of the Chebyshev method. On $\mathcal{D}_1$ we compute normalized call prices on a 1000×1000-grid of equidistantly distributed points, see [68]. We compare the runtimes and errors in the time-scaled volatilities $\Delta v := |v - v^{imp}|$ and the repricing errors $\Delta c := |c(x, v) - c(x, v^{imp})|$ of the methods.

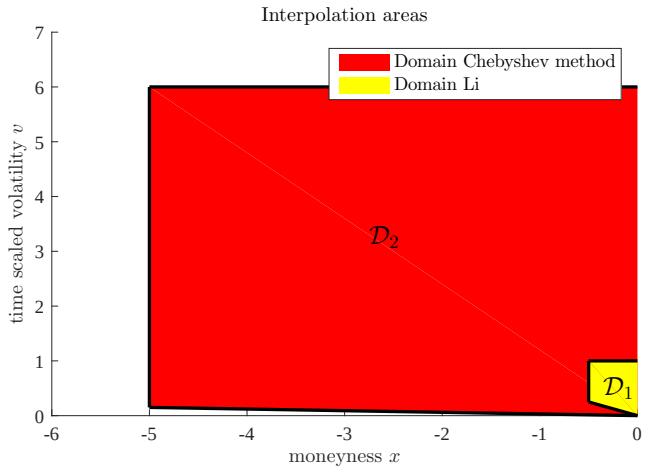

Figure 2.8: Domain $\mathcal{D}_2$ of the Chebyshev interpolation (red), domain of Li (yellow) and domain $\mathcal{D}_1$ as the intersection of both.

Table 2-B shows the maximal and the mean error in terms of the time-scaled volatilities and the normalized call prices as well as the runtime as a proportion of the runtime of the Newton-Raphson method, which takes $1.45s$. For the Chebyshev method, the runtime measures the time of the online phase. The [75] method comes with a solution close to machine precision for all input parameters and thus qualifies as our reference method in the offline-phase of the Chebyshev approximation. Also the Newton-Raphson algorithm reaches very high precision. The approximation of [86], however, is not able to reach the same range of precision. As Table 2-B shows, the mean error of σ is a factor 10^{10} higher than Jäckel's approximation. The proposed modification of [86] with two additional Newton-Raphson steps reduces the error. However, for low volatilities the effect is rather small and the maximal error is still in the region of 10^{-5}, see Table 2-B. For the Chebyshev method, the error is determined by the pre-specified accuracy and the maximal error is close to the mean error. When comparing the runtimes, approximation formula given in [86] is the fastest. It comes, however, with the lowest precision of a maximal error in σ of $3.26 \cdot 10^{-3}$. For a higher precision in the range of 10^{-5}, the Chebyshev method with low accuracy turns out to be faster than the approximation formula given in [86] with two steps Newton-Raphson. Comparing the mean, the same holds for the Chebyshev method with medium accuracy. For very high precisions the Chebyshev method with high accuracy is faster than the Newton-Raphson approach. Compared to Jäckel's method, the Chebyshev approach is two times faster but with a maximal error of 10^{-11} instead of 10^{-14}.

Method	max $\lvert\Delta\sigma\rvert$	mean $\lvert\Delta\sigma\rvert$	max $\lvert\Delta c\rvert$	mean $\lvert\Delta c\rvert$	runtime
Jäckel	$2.80 \cdot 10^{-14}$	$4.57 \cdot 10^{-16}$	$1.67 \cdot 10^{-15}$	$9.99 \cdot 10^{-17}$	1.39
Li	$3.26 \cdot 10^{-3}$	$3.42 \cdot 10^{-4}$	$2.15 \cdot 10^{-4}$	$9.43 \cdot 10^{-5}$	0.12
Li with 2 steps of Newton-Raphson	$2.02 \cdot 10^{-5}$	$6.12 \cdot 10^{-9}$	$1.10 \cdot 10^{-6}$	$3.89 \cdot 10^{-10}$	0.63
Newton-Raphson	$2.05 \cdot 10^{-10}$	$6.32 \cdot 10^{-14}$	$2.91 \cdot 10^{-11}$	$1.00 \cdot 10^{-14}$	1
Chebyshev method (low accuracy)	$1.52 \cdot 10^{-5}$	$1.40 \cdot 10^{-6}$	$4.91 \cdot 10^{-6}$	$3.94 \cdot 10^{-7}$	0.40
Chebyshev method (medium accuracy)	$3.20 \cdot 10^{-8}$	$2.17 \cdot 10^{-9}$	$3.52 \cdot 10^{-9}$	$5.92 \cdot 10^{-10}$	0.55
Chebyshev method (high accuracy)	$4.88 \cdot 10^{-11}$	$4.78 \cdot 10^{-12}$	$1.51 \cdot 10^{-11}$	$1.41 \cdot 10^{-12}$	0.67

Table 2-B: Interpolation error and runtimes on domain $\mathcal{D}_1$.

Method	max $\lvert\Delta\sigma\rvert$	mean $\lvert\Delta\sigma\rvert$	max $\lvert\Delta c\rvert$	mean $\lvert\Delta c\rvert$	runtime
Jäckel	$5.30 \cdot 10^{-13}$	$5.35 \cdot 10^{-15}$	$2.55 \cdot 10^{-15}$	$7.10 \cdot 10^{-17}$	0.52
Newton-Raphson	$8.34 \cdot 10^{-8}$	$6.64 \cdot 10^{-12}$	$1.94 \cdot 10^{-11}$	$1.28 \cdot 10^{-15}$	1
Chebyshev method (low accuracy)	$2.55 \cdot 10^{-5}$	$1.85 \cdot 10^{-6}$	$4.63 \cdot 10^{-6}$	$1.42 \cdot 10^{-7}$	0.14
Chebyshev method (medium accuracy)	$4.42 \cdot 10^{-8}$	$2.38 \cdot 10^{-9}$	$4.02 \cdot 10^{-9}$	$1.36 \cdot 10^{-10}$	0.16
Chebyshev method (high accuracy)	$1.66 \cdot 10^{-10}$	$1.32 \cdot 10^{-11}$	$1.52 \cdot 10^{-11}$	$4.83 \cdot 10^{-13}$	0.20

Table 2-C: Interpolation error and runtimes on domain $\mathcal{D}_2$.

Comparison on Domain $\mathcal{D}_2$

Next, we compare the Chebyshev method on the large domain $\mathcal{D}_2$ to the Newton-Raphson approach and the algorithm of Jäckel. The errors and runtimes are again computed on a 1000×1000 grid. Table 2-C shows the maximal and the mean error as well as the runtimes scaled as in the previous experiment. Here, the Newton-Raphson method takes $4.29s$. To reach a medium accuracy in the maximal error in the range of 10^{-8}, the Chebyshev method is more than six times faster than the Newton-Raphson approach. Moreover, the Chebyshev method is able to reach higher accuracies of 10^{-10} and still needs only 20% of the runtime of Newton-Raphson. Jäckel's method reaches very high precisions and is faster than Newton-Raphson. Compared the Jäckel method, the Chebyshev method allows us to pre-specify accuracies and reduce the runtimes significantly. For example, if accuracies in the region of 10^{-8} are sufficient, the Chebyshev method is more than three times faster than Jäckel's approach.

Comparison for market data

In Section 2.4.2 we investigated market data of options and concluded that a significant part of the options is not covered by the domain of [86]. This was the motivation to consider a much larger interpolation domain for the Chebyshev method. An empirical investigation confirms that all the options shown in Figure 2.5 lie within our domain.

Method	max $\lvert\Delta\sigma\rvert$	mean $\lvert\Delta\sigma\rvert$	max $\lvert\Delta c\rvert$	mean $\lvert\Delta c\rvert$	runtime
Jäckel	$8.05 \cdot 10^{-16}$	$1.40 \cdot 10^{-16}$	$2.11 \cdot 10^{-15}$	$2.43 \cdot 10^{-16}$	0.89
Newton-Raphson	$1.78 \cdot 10^{-10}$	$2.91 \cdot 10^{-12}$	$7.72 \cdot 10^{-12}$	$2.22 \cdot 10^{-13}$	1
Chebyshev method (low)	$1.57 \cdot 10^{-5}$	$2.95 \cdot 10^{-6}$	$4.44 \cdot 10^{-6}$	$4.78 \cdot 10^{-7}$	0.37
Chebyshev method (medium)	$4.19 \cdot 10^{-8}$	$3.87 \cdot 10^{-9}$	$3.45 \cdot 10^{-9}$	$3.98 \cdot 10^{-10}$	0.48
Chebyshev method (high)	$1.73 \cdot 10^{-11}$	$2.21 \cdot 10^{-12}$	$2.70 \cdot 10^{-12}$	$2.91 \cdot 10^{-13}$	0.58

Table 2-D: Interpolation error and runtimes for S&P 500 market data.

Next, we compare the Chebyshev method on this market data to the Newton-Raphson approach and the algorithm of Jäckel. The errors and runtimes are computed for options on the S&P 500 index traded on 7/17/2017 (Source Thomson Reuters Eikon). We use the same options as for Figure 2.5. To obtain more reliable results for the runtime comparison we compute the implied volatilities of the options 5,000 times.

Table 2-D shows the maximal and the mean error as well as the runtimes scaled as in the previous two experiments. Here, the Newton-Raphson method takes $5.72s$. The Chebyshev method is the fastest of the three methods and reaches the target accuracies. The method is about twice as fast as the Newton-Raphson approach for similar accuracies. Again, Jäckel's method reaches very high precisions but it is significantly slower than the Chebyshev method.

Besides the observed gain in efficiency, the Chebyshev method enjoys conceptual advantages. It delivers a closed-form approximation in a simple and easy to use polynomial structure.

2.4.5 Conclusion

The example of the implied volatility shows the potential of Chebyshev interpolation for problems in financial engineering. The approximation of the implied volatility is a very specific example but many of observed the conceptual advantages can also be exploited in other applications. This includes

- *Closed form approximation formula*: The Chebyshev interpolation has a simple *polynomial structure* and can be evaluated efficiently. This structure can be further explored to express derivatives in a closed form.

- *Error analysis*: A theoretical error analysis is available for the Chebyshev interpolation and shows (sub)exponential convergence for the implied volatility. Here, we used that the price of a European call is an analytic function of the model parameters in the Black-Scholes. A fast convergence guarantees that we can approximate

the function with a low number of nodal points.

- *Easy Implementation*: Once the interpolation operator is set up in an offline phase, the polynomial structure of the approximation formula leads to a simple code. This facilitates the transfer of the code to other systems and programming languages as part of the maintenance. The method only requires the coefficients of the Chebyshev interpolation to be stored.

- *Splitting*: For a larger interpolation domain an appropriate splitting of the domain can significantly reduce the number of nodal points and thus increase the efficiency of the approximation.

The presented Chebyshev method for the implied volatility enjoys a high flexibility and the approach can be transferred to similar problems. For example, [56] exploit the method to approximate the implied volatility in the Laplace option pricing model of [90].

Chapter 3

The dynamic Chebyshev method

In Chapter 2 we have seen that Chebyshev interpolation is an efficient function approximation method that can be explored for static problems in option pricing. It performs very well for the approximation of European option prices and can also be used for the approximation of the implied volatility function. In this chapter we investigate path-dependent options where the resulting pricing problem is dynamic. The most import examples are early-exercise options such as American options and Bermudan options. For both option types holds that their value can be calculated as an optimal stopping problem. In contrast to a European option in the Black-Scholes model, there is no explicit solution to this problem. In lack of explicit solutions, different numerical methods haven been developed to tackle this problem.

The early-exercise feature poses an additional numerical challenge when pricing these types of option. For example, a straightforward Monte Carlo simulation is not suitable to price an American put option in the Black-Scholes model. Over the past forty years different approaches have been developed that tackle this problem.

One of the first algorithms to compute American put option prices in the Black-Scholes model has been proposed by [19]. In this approach, the related partial differential inequality is solved by a finite difference scheme. A rich literature further developing the PDE approach has accrued since, including methods for jump models ([85], [70]), extensions to two dimensions ([66]) and combinations with complexity reduction techniques ([65]). Besides PDE based methods a variety of other approaches has been introduced, many of which trace back to the solution of the optimal stopping problem by the dynamic programming principle, see e.g. [106]. For Fourier based solution schemes we refer to [89], [44]. Simulation based approaches are of fundamental importance, the most prominent representative of this group is the least-squares Monte Carlo (LSM) approach of

[88], we refer to [54] and [83] for an overview of different Monte-Carlo methods.

Typically, Fourier and PDE methods are highly accurate, compared to simulation, however, they are less flexible towards changes in the model and particularly in the dimensionality. In order to reconcile the advantages of the PDE and Fourier approach with the flexibility of Monte Carlo simulation, we propose a new approach based on Chebyshev interpolation.

In this chapter we will first formulate the pricing problem as a general dynamic programming problem. Then we introduce our new method and provide a first modification that incorporates a domain splitting in the state space. We investigate both approaches numerically and provide convergence results if the pricing problem is analytic or piecewise analytic. We discuss the implementation of the method and conduct an empirical convergence analysis. We conclude the chapter with a benchmark test and an outline of different extensions of the method.

This chapter is based on a research collaboration with Kathrin Glau and Mirco Mahlstedt. In parts, the results of this chapter are published in our joint paper "A new approach for American option pricing: The Dynamic Chebyshev method", see [59]. Moreover, a previous version of the presented method is also included in the PhD thesis "Complexity Reduction for Option Pricing" of [91]. The results that are discussed in this chapter are only the ones to which the author has contributed.

3.1 A new pricing algorithm for path-dependent options

In this section, we introduce a new pricing method for path-dependent options. The most important example of this type are American options, i.e. options that can be exercised at any time point until maturity. Like most approaches, we discretize the continuous time problem of pricing an American option and then solve it. Hence, we actually compute the price of a Bermudan option. It is well known that the Bermudan price converges towards the American option price. Therefore, a Bermudan option with a high number of exercise rights or an extrapolation technique can be used to obtain the American option price, see [53]. The pricing of Bermudan options is similar to the pricing of discretely monitored barrier options as for example stated in [44]. Our proposed new approach will be general enough to cover this pricing problem as well.

We consider a general dynamic programming time-stepping in discrete time for a Markov process X_t and value function V_t. We propose to approximate the value function in every time step using Chebyshev interpolation, i.e. $V_t \approx \sum_j c_j^t p_j$. The choice of Chebyshev polynomials is motivated by the promising properties of Chebyshev interpo-

lation such as

- The vector of coefficients $(c_j^{t+1})_{j=0,\dots,N}$ is explicitly given as a linear combination of the values $V_t(x_k)$ at the Chebyshev grid points x_k. The pricing is then done on a discrete grid of Chebyshev points $x = x_k$.
- Exponential convergence of the interpolation for analytic functions and polynomial convergence of differential functions depending on the order.
- The interpolation can be implemented in a numerically stable way.

See Chapter 2 for more details on Chebyshev interpolation. The computation of the continuation value at a single time step coincides with the pricing of a European option. In Section 2.3 we have seen that the interpolation of European option prices using Chebyshev interpolation shows to be highly promising and exponential convergence is established for a large set of models and option types. Moreover, the approximation of the value function with Chebyshev polynomials has already proven to be beneficial for optimal control problems in economics, see [78] and [26].

The key advantage of our approach for American option pricing is that it collects all model-dependent computations in generalized conditional moments $\Gamma_{j,k}$. If there is no closed-form solution, their calculation can be shifted into an offline phase prior to the time-stepping. Depending on the underlying model a suitable numerical technique such as Monte Carlo, PDE and Fourier transform methods can be chosen, which reveals the high flexibility of the approach. Once the generalized conditional moments $\Gamma_{j,k}$ are computed, the backward induction is solved on a discrete Chebyshev grid. This avoids any computations of conditional expectations during the time-stepping. For each time step the method delivers a closed form approximation of the price function $x \mapsto \sum c_j^t T_j(x)$ along with the option's Delta and Gamma. Since the family of generalized conditional moments $\Gamma_{j,k}$ are independent of the value function, they can be used to generate multiple outputs including the option prices for different strikes, maturities and different payoff profiles. We will later explore this structure in the context of credit exposure calculation for pricing and risk-management.

The offline-online decomposition separates model and payoff yielding a modular design. We exploit this structure for a thorough error analysis and find conditions that imply explicit error bounds. They reflect the modularity by decomposing into a part stemming from the Chebyshev interpolation, from the time-stepping and from the offline computation. Under smoothness conditions the asymptotic convergence behaviour is deduced.

3.1.1 American options, optimal stopping and dynamic programming

We follow the comprehensive summary of [99] and start with an American put option in the Black-Scholes stock price model. Let $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{Q})$ be a filtered probability space where $\mathbb{Q}$ is the risk-neutral pricing measure. Assume a bank account paying a continuously compounded risk-free interest rate $r \geq 0$ that is modeled by

$$\mathrm{d}B_t = rB_t\mathrm{d}t, \quad \text{with} \quad B_0 = 1.$$

Moreover, assume a risky asset, i.e. a stock, S_t which is modeled by the following SDE

$$\mathrm{d}S_t = S_t\big(r\mathrm{d}t + \sigma\mathrm{d}W_t\big), \quad S_0 > 0$$

for a volatility $\sigma > 0$ and W_t is a Brownian motion under $\mathbb{Q}$. The value of the American put option is given by the following theorem.

Theorem 13. *The arbitrage-free value of an American put option with strike K and maturity T in the Black-Scholes model is given by*

$$V(s,t) = \sup_{\tau\in\mathcal{T}_{t,T}} \mathbb{E}\big[e^{-r(\tau-t)}(K - S_\tau)^+ \big| S_t = s\big]$$

for $s \in (0,\infty)$ and $t \in [0,T]$. The set $\mathcal{T}_{t,T}$ refers to all stopping times with respect to the filtration $(\mathcal{F}_t)_{t\geq 0}$ that have values in $[t,T]$.

Proof. See Theorem 3.1 and Remark 3.3 of [99] and the further references therein. □

The theorem formalizes the intuitive idea that the holder of an American (put) option will maximize his expect return by finding the optimal exercise point until the maturity T. Closely related to the American option is the pricing problem of a Bermudan option. A Bermudan option is an option that can only be exercised at a set of (finitely many) time points before the maturity T. It can be seen as the discrete-time analogy of the continuous time problem of pricing an American option. In the following, we will focus on the pricing of Bermudan options. For numerical methods it is more convenient to work in discrete time instead of continuous time.

We start with a more general optimal stopping problem following [106]. Let the stochastic process $X = (X_t)_{t\leq T}$ be a Markov process with state space $\mathbb{R}^d$ defined on the filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{P})$. Let $g : [0,T] \times \mathbb{R}^d \to \mathbb{R}$ be a continuous function with $\mathbb{E}\left[\sup_{0\leq t\leq T} |g(t, X_t)|\right] < \infty$. Then

$$V(t,x) := \sup_{t\leq\tau\leq T} \mathbb{E}\left[g(\tau, X_\tau)|X_t = x\right] \qquad \text{for all} \quad (t,x) \in [0,T]\times\mathbb{R}^d$$

over all stopping times τ, see (2.2.2′) in [106]. In discrete time, the optimal stopping problems can be solved with dynamic programming. Namely, with time stepping $t = t_0 < \ldots < t_{n_T} = T$ the solution of the optimal stopping problem can be calculated via backward induction

$$\begin{aligned} V_T(\boldsymbol{x}) &= g(T, \boldsymbol{x}) \\ V_{t_u}(\boldsymbol{x}) &= \max\left(g(t_u, \boldsymbol{x}), \mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = \boldsymbol{x}]\right). \end{aligned}$$

Note that n refers to the number of time steps between t and T. For notational convenience, we indicate the value function at each time step with subscript t_u to directly refer to the time step t_u. For a detailed overview of optimal control problems in discrete time we refer to [106].

If we ignore the early-exercise possibility and set $g(t_u, x) = 0$ for $t_u < T$ the problem boils down to the pricing of a European option. Moreover, discretely monitored barrier options can be priced via backward induction, see for example [44]. We consider an up-and-out call option with strike K and barrier B in the Black-Scholes model and assume that the barrier is discretely monitored at dates $t = t_0 < \ldots < t_{n_T} = T$. If the stock price is above the barrier at any of the monitoring days the option is knocked out and has zero value. The idea of barrier options is to make plain vanilla European call or put options cheaper by introducing the knock-out barrier. The value of such an option can be computed via backward induction

$$\begin{aligned} V_T(S) &= g(T, S) = (S - K)^+ \mathbb{1}_{(0,B]}(S) \\ V_{t_u}(S) &= e^{-r(t_{u+1} - t_u)} \mathbb{E}[V_{t_{u+1}}(S_{t_{u+1}})|S_{t_u} = S] \mathbb{1}_{(0,B]}(S). \end{aligned}$$

An efficient numerical pricing algorithm for early-exercise option is thus also a good candidate for the pricing of barrier options. This motivates us to define a slightly more general dynamic programming problem that includes American, Bermudan, European and barrier options.

Definition 6. *Let $\boldsymbol{X} = (\boldsymbol{X}_t)_{t \leq T}$ a Markov process with state space $\mathbb{R}^d$. For a continuous function $g : [0, T] \times \mathbb{R}^d \longrightarrow \mathbb{R}$ with $\mathbb{E}\left[\sup_{0 \leq t \leq T} |g(t, \boldsymbol{X}_t)|\right] < \infty$ we define the general Dynamic Programming Problem (DPP) with value function V_t as*

$$V_T(\boldsymbol{x}) = g(T, \boldsymbol{x}) \tag{3.1}$$

$$V_{t_u}(\boldsymbol{x}) = f\left(g(t_u, \boldsymbol{x}), \mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = \boldsymbol{x}]\right), \tag{3.2}$$

where $0 = t_0 < \ldots < t_n = T$ and $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is Lipschitz continuous with constant L_f.

Similar to the previous section, we write variables in bold to indicate that their are multivariate vectors and not scalars. We will introduce our new numerical method for this general dynamic programming problem.

3.1.2 A dynamic pricing algorithm using Chebyshev interpolation

In this section, we introduce a new pricing method based on Chebyshev polynomial interpolation. Consider the time step in the backward induction as defined in (3.2),

$$V_t(x) = f\left(g(t,x), \mathbb{E}[V_{t+1}(X_{t+1})|X_t = x]\right),$$

with time steps $t_u < t_{u+1} < \ldots < T$ and payoff function g. The computational challenge is to compute $\mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x]$ for *for all time steps* t_u *and all states* x, where $V_{t_{u+1}}$ depends on *all previous time steps.*

The general idea of our approach is to approximate the value function in each time step by Chebyshev polynomial interpolation. For the moment we assume that X_t is a one-dimensional process. We express the value function $V_{t_{u+1}}$ as a finite sum of Chebyshev polynomials T_j times coefficients c_j^{u+1}. In this case, the conditional expectations become

$$\mathbb{E}[V_{t+1}(X_{t+1})|X_t = x] \approx \sum c_j^{t+1} \mathbb{E}[T_j(X_{t+1})|X_t = x] = \sum c_j^{t+1} \Gamma_{j,k},$$

with generalized moments $\Gamma_{j,k} := \mathbb{E}[T_j(X_{t+1})|X_t = x]$. Due to the linearity of the expectations we only have to compute conditional expectations of Chebyshev polynomials instead of conditional expectations of the value function $V_{t_{u+1}}$. In order to compute the coefficients in each time step we require the function values at the Chebyshev points. This allows us to solve the backward induction on a discretized grid.

In the general d-dimensional set-up, we start the pricing algorithm by fixing a (hyper)rectangular domain

$$\mathcal{X} = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_d, \overline{x}_d] \subset \mathbb{R}^d.$$

We start at $T = t_{n_T}$ and apply a Chebyshev interpolation to the function $g(T,x)$, i.e. for $x \in \mathcal{X}$,

$$V_T(\boldsymbol{x}) = g(T, \boldsymbol{x}) \approx \sum\nolimits_{\boldsymbol{j}} c_{\boldsymbol{j}}(T) p_{\boldsymbol{j}}(x) =: \widehat{V}_T(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathcal{X}$$

where $p_{\boldsymbol{j}}$ are the multivariate (transformed) Chebyshev polynomials for multi-index $\boldsymbol{j} = (j_1, \ldots, j_d)$ with $0 \leq j_i \leq N_i$ for Chebyshev degree N_i and $i = 1, \ldots, d$. At the first time step t_{n_T-1}, the derivation of the conditional expectation $\mathbb{E}[g(t_{n_T}, X_{t_{n_T}})|X_{t_{n_T-1}} = \boldsymbol{x}]$ is

replaced by $\mathbb{E}[\sum_j c_{\boldsymbol{j}}(t_{n_T})p_{\boldsymbol{j}}(X_{t_{n_T}})|X_{t_{n_T-1}} = \boldsymbol{x}]$ yielding

$$\begin{aligned} V_{t_{n_T-1}}(\boldsymbol{x}) &= f\left(g(t_{n_T-1},\boldsymbol{x}), \mathbb{E}[V_{t_{n_T}}(\boldsymbol{X}_{t_{n_T}})|\boldsymbol{X}_{t_{n_T-1}} = \boldsymbol{x}]\right) \\ &\approx f\Big(g(t_{n_T-1},\boldsymbol{x}), \mathbb{E}\Big[\sum\nolimits_{\boldsymbol{x}} c_{\boldsymbol{j}}(t_{n_T})p_{\boldsymbol{j}}(\boldsymbol{X}_{t_n})\Big|\boldsymbol{X}_{t_{n-1}} = \boldsymbol{x}\Big]\Big) \\ &= f\Big(g(t_{n-1},\boldsymbol{x}), \sum\nolimits_{\boldsymbol{j}} c_{\boldsymbol{j}}(t_{n_T})\mathbb{E}\Big[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{n_T}})\Big|\boldsymbol{X}_{t_{n_T-1}} = \boldsymbol{x}\Big]\Big). \end{aligned}$$

At time step t_{n-1}, the value function $V_{t_{n-1}}$ needs only to be evaluated at the specific Chebyshev nodes. Hence, denoting with $\boldsymbol{x_k} = (x_{k_1}, \ldots, x_{k_d})$ the Chebyshev nodes, it suffices to evaluate

$$V_{t_{n-1}}(\boldsymbol{x_k}) \approx f\Big(g(t_{n-1},\boldsymbol{x_k}), \sum\nolimits_{\boldsymbol{j}} c_{\boldsymbol{j}}(t_n)\mathbb{E}\Big[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_n})\Big|\boldsymbol{X}_{t_{n-1}} = \boldsymbol{x_k}\Big]\Big) =: \widehat{V}_{t_{n-1}}(\boldsymbol{x_k}). \quad (3.3)$$

A linear transformation of the nodal values $\widehat{V}_{t_{n-1}}(\boldsymbol{x_k})$ yields the Chebyshev coefficients according to Definition 4 which determines the Chebyshev interpolation $\widehat{V}_{t_{n-1}}$. We apply this procedure iteratively as described in detail in Algorithm 1.

The stochastic part is gathered in the expectations of the Chebyshev polynomials conditioned on the Chebyshev nodes, i.e. $\Gamma_{\boldsymbol{j},\boldsymbol{k}}(t_u) = \mathbb{E}[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_k}]$. Moreover, if an equidistant time stepping is applied the computation can be further simplified. If for the underlying stochastic process

$$\Gamma_{\boldsymbol{j},\boldsymbol{k}}(t_u) = \mathbb{E}[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_k}] = \mathbb{E}[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_1})|\boldsymbol{X}_{t_0} = \boldsymbol{x_k}] =: \Gamma_{\boldsymbol{j},\boldsymbol{k}} \quad (3.4)$$

for $u = 0, \ldots, n-1$, then the conditional expectations need to be computed only for one time step, see Algorithm 2. One can pre-compute these conditional expectations and thus the method allows for an offline/online decomposition.

If the value function $\boldsymbol{x} \mapsto V_{t_u}(\boldsymbol{x})$ is analytic, the Chebyshev interpolation converges exponentially fast. We will see that a European and a discretely monitored basket option fall into this category. For analytic value functions we can hope for a fast convergence of the method due to the promising properties of the static Chebyshev interpolation. Unfortunately, the American put does not fall into the category of option prices with analytic value functions. We can still apply the new pricing algorithm and use more nodal points or we have to modify the pricing algorithm. We will discuss both ideas in the next section.

Algorithm 1 Dynamic Chebyshev algorithm

Require: $\overline{N} \in \mathbb{N}^D$, $\mathcal{X} = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_d, \overline{x}_d]$, $0 = t_0, \ldots, t_n = T$

1: Define Chebyshev points $\boldsymbol{z_k} = (z_{k_1}, \ldots, z_{k_D})$ and nodal points $\boldsymbol{x_k} = \tau_{\mathcal{X}}(\boldsymbol{z_k})$, $0 \leq k_i \leq N_i$ for $i = 1, \ldots, d$
2: **Pre-computation step:**
3: For all $\boldsymbol{j}, \boldsymbol{k}$ and all t_u, $u = 0, \ldots, n-1$
4: Compute $\Gamma_{\boldsymbol{j},\boldsymbol{k}}(t_u) = \mathbb{E}[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}}) | \boldsymbol{X}_{t_u} = \boldsymbol{x_k}]$
5: **Time** T
6: $\widehat{V}_T(\boldsymbol{x_k}) = g(T, \boldsymbol{x_k})$, derive
7: $c_{\boldsymbol{j}}(T) = D_{\overline{N}}(\boldsymbol{j}) {\sum_{\boldsymbol{k}}}'' \widehat{V}_T(\boldsymbol{x_k}) T_{\boldsymbol{j}}(\boldsymbol{z_k})$ with $D_{\overline{N}}(\boldsymbol{j}) = \prod_{i=1}^{d} \left(\frac{2^{\mathbb{1}_{0<j_i<N_i}}}{N_i}\right)$
8: Obtain Chebyshev interpolation $\widehat{V}_T(\boldsymbol{x}) = \sum_{\boldsymbol{j}} c_{\boldsymbol{j}}(T) p_{\boldsymbol{j}}(\boldsymbol{x})$ of $V_T(\boldsymbol{x})$
9: **Iterative time stepping** from $t_{u+1} \to t_u$, $u = n-1, \ldots, 1$
10: Given Chebyshev interpolation of $\widehat{V}_{t_{u+1}}(\boldsymbol{x}) = \sum_{\boldsymbol{j}} c_{\boldsymbol{j}}(t_{u+1}) p_{\boldsymbol{j}}(\boldsymbol{x})$
11: Derivation of $\widehat{V}_{t_u}(\boldsymbol{x_k})$ at the nodal points
12: $\widehat{V}_{t_u}(\boldsymbol{x_k}) = f(g(t_u, \boldsymbol{x_k}), \sum_{\boldsymbol{j}} c_{\boldsymbol{j}}(t_{u+1}) \Gamma_{\boldsymbol{j},\boldsymbol{k}}(t_u))$
13: Derive $c_{\boldsymbol{j}}(t_u) = D_{\overline{N}}(j) {\sum_{\boldsymbol{k}}}'' \widehat{V}_{t_u}(\boldsymbol{x_k}) T_{\boldsymbol{j}}(\boldsymbol{z_k})$
14: Obtain Chebyshev interpolation $\widehat{V}_{t_u}(\boldsymbol{x}) = \sum_{\boldsymbol{j}} c_{\boldsymbol{j}}(t_u) p_{\boldsymbol{j}}(\boldsymbol{x})$ of $V_{t_t}(\boldsymbol{x})$
15: **Deriving the solution at** $t = 0$
16: $\widehat{V}_0(\boldsymbol{x}) = \sum_{\boldsymbol{j}} c_{\boldsymbol{j}}(0) p_{\boldsymbol{j}}(\boldsymbol{x})$

Algorithm 2 Simplified Dynamic Chebyshev algorithm

Require: Time steps $0 = t_1, \ldots, t_n = T$ with $\Delta t := t_u - t_{u-1}$

1: Replace in Algorithm 1 Lines 2-4 with:
2: **Pre-computation step:**
3: Compute $\Gamma_{\boldsymbol{j},\boldsymbol{k}} = \mathbb{E}[p_{\boldsymbol{j}}(\boldsymbol{X}_{\Delta t}) | \boldsymbol{X}_0 = \boldsymbol{x_k}]$ for all polynomials $p_{\boldsymbol{j}}$ and all nodal points $\boldsymbol{x_k}$

3.1.3 Pricing of the American put

We recall the dynamic programming problem of an American put for the log stock price X_t with $S_t = e^{X_t}$ in the Black-Scholes model

$$\begin{aligned} V_T(x) &= (K - e^x)^+ \\ V_{t_u}(x) &= \max\left\{(K - e^x)^+, e^{-r(t_{u+1}-t_u)} \mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}}) | X_{t_u} = x]\right\}. \end{aligned}$$

Obviously, the payoff function $x \mapsto (K - e^x)^+$ has a kink at $x = \log(K)$ and is thus not differentiable on $\mathbb{R}$. However, this is not a significant limitation since we know that the conditional expectation

$$\mathbb{R} \ni x \mapsto e^{-r(t_n - t_{n-1})} \mathbb{E}\big[(K - e^{X_T})^+ | X_{t_{n-1}} = x\big]$$

is an analytic function and equals to the Black-Scholes price of an European option. In the pricing algorithm one can therefore simply start at t_{n-1} and calculate the conditional expectation at the nodal points explicitly. We refer to this idea as "smoothing the payoff" and discuss the underlying principle in more detail in Section 3.4.1.

The computational challenge of pricing an American put via backward induction is taking the maximum in every time step. We know that the maximum function is not continuous and the mapping $x \mapsto V_{t_u}(x)$ does not have an analytic extension into the complex domain. We will discuss the properties of the American put option price in more detail and provide two possible solutions.

In continuous time, the price of an American put is given by the optimal stopping problem

$$V_t(x) = \sup_{\tau \in \mathcal{T}_{t,T}} \mathbb{E}\big[e^{-r(\tau-t)}(K - e^{X_\tau})^+ \big| X_t = x\big].$$

For every $t \in [0,T]$ we can define two sets

$$\mathcal{C}_t = \big\{x \in \mathbb{R} \,\big|\, V_t(x) > (K - e^x)^+\big\} \quad \text{and} \quad \mathcal{E}_t = \big\{x \in \mathbb{R} \,\big|\, V_t(x) = (K - e^x)^+\big\}.$$

From the properties of the American put option follows that there is an unique $b_t^\star \in \mathbb{R}$ such that $\mathcal{E}_t = (-\infty, b_t^\star]$ and $\mathcal{C}_t = (b_t^\star, \infty)$. We call $\mathcal{C}_t$ the continuation region, $\mathcal{E}_t$ the exercise region and $b_t^\star$ the optimal exercise boundary. One can show that $t \mapsto b_t^\star$ is a nondecreasing and continuous function on $[0,T)$ with $\lim_{t\to T} b_t^\star = \log(K)$, see [99] and Chapter 25 in [106].

Exploiting the optimal exercise boundary, we obtain from [99] the following representation of the American put option price

$$V_t(x) \;=\; \mathbb{E}\Big[e^{-r(T-t)}(K - e^{X_T})^+ \Big| X_t = x\Big] \;+\; \mathbb{E}\Big[\int_t^T e^{-r(s-t_u)} rK \mathbb{1}_{\{X_s < b_s^\star\}} \mathrm{d}s \Big| X_t = x\Big]$$

The first term is the value of a European option with the same parameters and the second term is the additional early-exercise premium. It follows immediately that the price of a European and an American put option coincide if interest rates are zero.

In the Black-Scholes model one can show that for the derivatives of the American put value function holds

$$\lim_{x \searrow b_t} \frac{dV_t(x)}{dx} = -1 \quad \text{for almost every } t \in [0,T] \tag{3.5}$$

see [99]. This property is often called the "smooth fit" at the optimal exercise boundary and it implies that the function $x \mapsto V_t(x)$ is at least one time continuously differentiable. As pointed out in [99], the financial implication of the smoothness is that it enables a continuous adjustment of the hedging portfolio for an American put. There are no jumps in the option's delta around the optimal exercise boundary.

From a numerical point of view, the smoothness ensures that the interpolation of the value function will still convergence algebraically. This justifies the use of the general dynamic Chebyshev as presented in Algorithm 1 to price an American put. The number of nodal points will however be higher than for an European option. Alternatively, we can explore the fact that in the exercise region the value function $x \mapsto (K - e^x)$ is analytic and the same holds true for $x \mapsto \mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x]$ in the continuation region. Hence, the function $x \mapsto V_{t_u}(x)$ is piecewise analytic and we propose to split the interpolation domain in the dynamic Chebyshev algorithm.

At each time step t_u in the backward induction we will conduct the following steps

- compute the splitting point/ exercise boundary $b^\star_{t_u}$,
- interpolate the value function on each interval $[\underline{x}, b^\star_{t_u}]$ and $[b^\star_{t_u}, \overline{x}]$.

At the final time step $t_n = T$, the splitting point is $b^\star_T = k$ and we only have to approximate the payoff on $\mathcal{X}_{1,T} = [\underline{x}, b^\star_T]$, i.e. for $x \in \mathcal{X}$ we obtain

$$g(x) \approx I_N(g|_{\mathcal{X}_{1,T}})(x) + I_N(g|_{\mathcal{X}_{2,T}})(x) = I_N(g|_{\mathcal{X}_{1,T}})(x)$$

where $I_N(g|_{\mathcal{X}_{1,T}})$ is the Chebyshev interpolation of g on the interval $\mathcal{X}_{1,T}$

$$I_N(g|_{\mathcal{X}_{1,T}})(x) = \sum_{j=0}^{N} c_j^1(T) p_j^{1,T}(x) \quad \text{with} \quad p_j^{1,T}(x) := T_j(\tau_{\mathcal{X}_{1,T}}(x)) \mathbb{1}_{\mathcal{X}_{1,T}}(x).$$

At time step t_{n-1} we can use this approximation to compute the (continuation) value of the option, i.e.

$$\begin{aligned} V_{t_{n-1}}(x) &= \max\left\{g(x), \mathbb{E}[V_{t_n}(X_{t_n})|X_{t_{n-1}} = x]\right\} \\ &= \max\left\{g(x), \mathbb{E}[g(X_{t_n})|X_{t_{n-1}} = x]\right\} \\ &\approx \max\left\{g(x), \mathbb{E}[I_N(g|_{X_{1,T}})(X_{t_n})|X_{t_{n-1}} = x]\right\}. \end{aligned}$$

In order to approximate this function with Chebyshev interpolation we have to find the

exercise boundary $b^\star_{t_{n-1}} \in [\underline{x}, b^\star_{t_n}]$ given by

$$g(b^\star_{t_{n-1}}) - \mathbb{E}[I_N(g|_{X_{1,T}})(\mathcal{X}_{t_n})|X_{t_{n-1}} = b^\star_{t_{n-1}}] = 0.$$

The boundary can be calculated using a numerical root finder such as the Newton-Raphson algorithm. Then we split the domain $\mathcal{X}$ into the subdomains $\mathcal{X}_{1,t_{n-1}} = [\underline{x}, b^\star_{t_{n-1}}]$ and $\mathcal{X}_{2,t_{n-1}} = (b^\star_{t_{n-1}}, \overline{x}]$. On each subdomain, the value function is analytic and we will approximate the value function with Chebyshev polynomial interpolation. We define nodal points $x_{1,k} = \tau_{\mathcal{X}_{1,t_{n-1}}}(z_k)$ and $x_{2,k} = \tau_{\mathcal{X}_{2,t_{n-1}}}(z_k)$ for $k = 0, \ldots, N$ and compute the nodal values

$$\widehat{V}_{t_{n-1}}(x_{1,k}) := g(x_{1,k}) \quad \text{and} \quad \widehat{V}_{t_{n-1}}(x_{2,k}) := \mathbb{E}[I_N(g|_{X_{1,T}})(X_{t_n})|X_{t_{n-1}} = x_{2,k}].$$

Using this nodal values we can calculate the corresponding Chebyshev coefficients $c^1_j(t_{n-1})$, $c^2_j(t_{n-1})$ and obtain an interpolation of the value function

$$\begin{aligned} V_{t_{n-1}}(x) &\approx I_N(V_{t_{n-1}}|_{X_{1,t_{n-1}}})(x) + I_N(V_{t_{n-1}}|_{X_{2,t_{n-1}}})(x) \\ &= \sum_{j=0}^{N} c^1_j(t_{n-1}) p_j^{1,t_{n-1}}(x) \; + \; \sum_{j=0}^{N} c^2_j(t_{n-1}) p_j^{2,t_{n-1}}(x). \end{aligned}$$

This leads to the following time-stepping to solve the problem via backward induction. Assume we have an approximation $V_{t_{u+1}}(x) \approx I_N(V_{t_{u+1}}|_{X_{1,t_{u+1}}})(x) + I_N(V_{t_{u+1}}|_{X_{2,t_{u+1}}})(x)$, at t_u we will conduct the following steps:

1. Find the splitting point $b^\star_{t_u}$ such that

$$g(b^\star_{t_u}) - \mathbb{E}[I_N(V_{t_{u+1}}|_{\mathcal{X}_{1,t_{u+1}}})(X_{t_{u+1}}) + I_N(V_{t_{u+1}}|_{\mathcal{X}_{2,t_{u+1}}})(X_{t_{u+1}})|X_{t_u} = b^\star_{t_u}] = 0.$$

2. Split the domain $\mathcal{X}$ into the two sets $\mathcal{X}_{1,t_u} = [\underline{x}, b^\star_{t_u}]$ and $\mathcal{X}_{2,t_u} = (b^\star_{t_u}, \overline{x}]$.

3. Interpolate the value function on each of the sets using Chebyshev interpolation. Calculate the nodal values

$$\begin{aligned} &V_{t_u}|_{\mathcal{X}_{1,t_u}}(x_{1,k}) = g(x_{1,k}) \\ &V_{t_u}|_{\mathcal{X}_{2,t_u}}(x_{2,k}) = \mathbb{E}[I_N(V_{t_{u+1}}|_{\mathcal{X}_{1,t_{u+1}}})(X_{t_{u+1}}) + I_N(V_{t_{u+1}}|_{\mathcal{X}_{2,t_{u+1}}})(X_{t_{u+1}})|X_{t_u} = x_{2,k}] \end{aligned}$$

for $x_{1,k} = \tau_{\mathcal{X}_{1,t_{n-1}}}(z_k)$ and $x_{2,k} = \tau_{\mathcal{X}_{2,t_{n-1}}}(z_k)$, $k = 0, \ldots, N$. Then calculate the coefficients $c^1_j(t_u)$, $c^2_j(t_u)$.

4. Obtain the Chebyshev approximation of V_{t_u} given by

$$V_{t_u}(x) \approx I_N(V_{t_u}|\mathcal{X}_{1,t_u})(x) + I_N(V_{t_u}|\mathcal{X}_{2,t_u})(x).$$

The detailed algorithm is described in Algorithm 3.

Compared to the classical dynamic Chebyshev method we are now able to tackle the American put option pricing problem in a proper way and we will be able to obtain an exponential order of convergence. However, the new algorithm has one major drawback since it does not allow for an offline-online decomposition. In order to see this we have to look at the conditional expectations which are calculated in each time step, i.e. for $i = 1, 2$

$$\begin{aligned}
&\mathbb{E}[I_N(V_{t_{u+1}}|\mathcal{X}_{i,t_{u+1}})(X_{t_{u+1}})|X_{t_u} = x] \\
&\qquad = \mathbb{E}[\sum_{j=0}^{N} c_j^i(t_{u+1}) p_j^{i,t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x] \\
&\qquad = \sum_{j=0}^{N} c_j^i(t_{u+1}) \mathbb{E}[p_j^{i,t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x] \\
&\qquad = \sum_{j=0}^{N} c_j^i(t_{u+1}) \mathbb{E}[T_j(\tau_{\mathcal{X}_{i,t_{u+1}}}(x)) \mathbb{1}_{\mathcal{X}_{i,t_{u+1}}}(X_{t_{u+1}})|X_{t_u} = x],
\end{aligned}$$

where the conditional expectations $\mathbb{E}[T_j(\tau_{\mathcal{X}_{i,t_{u+1}}}(x)) \mathbb{1}_{\mathcal{X}_{i,t_{u+1}}}(X_{t_{u+1}})|X_{t_u} = x]$ depend on $\mathcal{X}_{i,t_{u+1}}$ and thus on splitting point $b^\star_{t_{u+1}}$. These splitting points are calculated during the backward induction and are not known before. Therefore, a pre-computation is not possible. Two different variations of the dynamic Chebyshev algorithm that tackle this problem are discussed in [91]. In Section 3.7 we will briefly mention another related modification of the dynamic Chebyshev method that enables again an offline-online decomposition.

3.1.4 A dynamic Chebyshev algorithm with splitting

In this section we generalize the algorithm presented in the previous section to general multivariate dynamic programming problem defined in (3.1) and (3.2). The main idea is to split the interpolation domain $\mathcal{X} = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_d, \overline{x}_d]$ into sub-domains, such that, on each sub-domain, the payoff/value function is analytic. One each sub-domain we can then employ a separate Chebyshev interpolation.

For the American put on one asset the sub-domains of $\mathcal{X} = [\underline{x}, \overline{x}]$ where the two intervals $[\underline{x}, b^\star_{t_u}]$ and $[b^\star_{t_u}, \overline{x}]$ where $b^\star_{t_u}$ is the optimal exercise boundary and $b^\star_T = \log(K)$.

Algorithm 3 Dynamic Chebyshev with splitting for an American put option

Require: $N_1, N_2 \in \mathbb{N}$, $\mathcal{X} = [\underline{x}, \overline{x}]$, $0 = t_0, \ldots, t_n = T$, strike K

Determine Chebyshev points z_{1,k_1}, $k_1 = 0, \ldots, N_1$, z_{2,k_2}, $k_2 = 0, \ldots, N_2$

procedure INITIAL TIME T

  Set $b^\star_{t_n} = \log(K)$, $\mathcal{X}_{1,t_n} = [\underline{x}, b^\star_{t_n}]$, $\mathcal{X}_{2,t_n} = (b^\star_{t_n}, \overline{x}]$, $x_{1,k} = \tau_{\mathcal{X}_{1,t_n}}(z_{1,k})$

  $\widehat{V}_T(x_{1,k}) = K - e^{x_{1,k}}$, $k = 0 : N_1$, derive

  $c^1_{j_1}(T) = \left(\frac{2^{\mathbb{1}\{0<j_1<N_1\}}}{N_1}\right) {\sum_{k=0}^{N_1}}'' \widehat{V}_T(x_{1,k}) T_{j_1}(z_{1,k})$

  Obtain Chebyshev interpolation $\widehat{V}_T(x) = \sum_{j_1=0}^{N_1} c^1_{j_1}(T) p^{1,T}_{j_1}(x)$

procedure ITERATIVE TIME STEPPING FROM $t_{u+1} \to t_u$, $u = n-1, \ldots, 1$

  Given $\widehat{V}_{t_{u+1}}(x) = \sum_{j_1=0}^{N_1} c^1_{j_1}(t_{u+1}) p^{1,t_{u+1}}_{j_1}(x) + \sum_{j_2=0}^{N_2} c^2_{j_2}(t_{u+1}) p^{2,t_{u+1}}_{j_2}(x)$

  Determine splitting point $b^\star_{t_u}$ as root of

  $x_0 \mapsto g(x_0) - \mathbb{E}[\widehat{V}_{t_{u+1}}(X_{t_{u+1}}) | X_{t_u} = x_0]$

  Set $\mathcal{X}_{1,t_u} = [\underline{x}, b^\star_{t_u}]$, $\mathcal{X}_{2,t_u} = (b^\star_{t_u}, \overline{x}]$, $x_{1,k} = \tau_{\mathcal{X}_{1,t_u}}(z_{1,k})$, $x_{2,k} = \tau_{\overline{\mathcal{X}}_{2,t_u}}(z_{2,k})$

  Apply Chebyshev interpolation on both intervals

  $\widehat{V}_{t_u}(x_{1,k}) = K - e^{x_{1,k}}$, $k = 0 : N_1$, derive

  $c^1_{j_1}(t_u) = \left(\frac{2^{\mathbb{1}\{0<j_1<N_1\}}}{N_1}\right) {\sum_{k=0}^{N_1}}'' \widehat{V}_{t_u}(x_{1,k}) T_{j_1}(z_{1,k})$

  $\widehat{V}_{t_u}(x_{1,k}) = \mathbb{E}[\widehat{V}_{t_{u+1}}(X_{t_{u+1}}) | X_{t_u} = x_{2,k}]$, $k = 0 : N_2$, derive

  $c^2_{j_2}(t_u) = \left(\frac{2^{\mathbb{1}\{0<j_2<N_2\}}}{N_2}\right) {\sum_{k=0}^{N_2}}'' \widehat{V}_{t_u}(x_{2,k}) T_{j_2}(z_{2,k})$

  Obtain Chebyshev interpolation

  $\widehat{V}_{t_u}(x) = \sum_{j_1=0}^{N_1} c^1_{j_1}(t_u) p^{1,t_u}_{j_1}(x) + \sum_{j_2=0}^{N_2} c^2_{j_2}(t_u) p^{2,t_u}_{j_2}(x)$

procedure DERIVING THE SOLUTION AT T=0

  $\widehat{V}_0(x) = \sum_{j_1=0}^{N_1} c^1_{j_1}(t_0) p^{1,t_0}_{j_1}(x) + \sum_{j_2=0}^{N_2} c^2_{j_2}(t_0) p^{2,t_0}_{j_2}(x)$

For multivariate options such as basket options the sub-domains or splitting regions are not necessarily (hyper)rectangulars. For example, the payoff of a basket put option on two assets $(K - 0.5(S_1 + S_2))^+$ has its kink at $S_1 + S_2 = 2K$. However, the exercise region $S_1 + S_2 \leq 2K$ is not a rectangle in $\mathbb{R}^2$. Therefore, we have to introduce a more general notation for the partition of $\mathcal{X}$ into sub-domains and the corresponding multivariate Chebyshev interpolation.

Definition 7 ($\mathcal{Q}$-partition)**.** *Let $\mathcal{X} \subset \mathbb{R}^d$ be a compact set. We define a $\mathcal{Q}$-partition of $\mathcal{X}$ as a partition $\mathcal{X}^{\mathcal{Q}} = \{\mathcal{X}_1, \ldots, \mathcal{X}_{\mathcal{Q}}\}$, $\mathcal{Q} \in \mathbb{N}$ of disjoint sets $\mathcal{X}_1, \ldots, \mathcal{X}_{\mathcal{Q}}$ such that*

$$\mathcal{X} = \bigcup_{l=1}^{\mathcal{Q}} \mathcal{X}_l \text{ and } \forall \mathcal{X}_l \in \mathcal{X}^{\mathcal{Q}} \;\; \exists\, \varphi_{\mathcal{X}_l} : [-1,1]^d \to \overline{\mathcal{X}}_l, \text{ bijective and analytic.} \tag{3.6}$$

If all $\varphi_{\mathcal{X}_l}$ are linear functions we call $\mathcal{X}^{\mathcal{Q}}$ a linear $\mathcal{Q}$-partition of $\mathcal{X}$.

For such a partition we can define a multivariate version of (one-dimensional) piecewise polynomial interpolation in Chebyshev points.

Definition 8 (Piecewise multivariate Chebyshev interpolation)**.** *Let $f : \mathcal{X} \longrightarrow \mathbb{R}$ and*

$\mathcal{X}^{\mathcal{Q}}$ a $\mathcal{Q}$-partition of $\mathcal{X} \subset \mathbb{R}^d$. Assume f is Lipschitz continuous on each closure $\overline{\mathcal{X}}_l$ for $\mathcal{X}_l \in \mathcal{X}^{\mathcal{Q}}$. For $\boldsymbol{N}_l \in \mathbb{N}^d$, $l = 1, \dots, \mathcal{Q}$ we can define the piecewise multivariate Chebyshev interpolation of the function f as

$$I^{\mathcal{Q}}_{\overline{N}^*}(f)(\boldsymbol{x}) = \sum_{l=1}^{\mathcal{Q}} I_{\boldsymbol{N}_l}(f\big|_{\mathcal{X}_l})(\boldsymbol{x}) \quad \textit{with} \quad \overline{N}^* = \{\boldsymbol{N}_1, \dots, \boldsymbol{N}_{\mathcal{Q}}\}. \tag{3.7}$$

where $I_{\boldsymbol{N}_l}(f\big|_{\mathcal{X}_l})$ is the multivariate Chebyshev interpolation of f on $\mathcal{X}_l$ given by

$$\begin{aligned} I_{\boldsymbol{N}_l}(f\big|_{\mathcal{X}_l})(\boldsymbol{x}) &= \sum_{0 \le \boldsymbol{j} \le \boldsymbol{N}_l} c^l_{\boldsymbol{j}} p^l_{\boldsymbol{j}}(\boldsymbol{x}) \\ c^l_j &= \Big(\prod_{i=1}^{d} \frac{2^{\mathbb{1}_{0<j_i<N_{l,i}}}}{N_{l,i}} \Big) \sum_{0 \le \boldsymbol{k} \le \boldsymbol{N}_l} f(x^{l,\boldsymbol{k}}) T_{\boldsymbol{j}}(z^k) \end{aligned}$$

with transformed Chebyshev polynomials $p^l_{\boldsymbol{j}}(x) = T_{\boldsymbol{j}}(\varphi^{-1}_{\mathcal{X}_l}(\boldsymbol{x}))\mathbb{1}_{\mathcal{X}_l}(\boldsymbol{x})$.

We assume that the value function is analytic on each element of the $\mathcal{Q}$-partition. This generalizes the idea of a piecewise analytic value function that we have seen for the American put option.

Assumptions 1 (Piecewise analytic DPP)**.** *Let a DPP be given as in 3.1 and 3.2. We assume that the DPP is piecewise analytic, i.e. we assume that for each $u = 0, \dots, n$ there exists a $\mathcal{Q}$-partition $\mathcal{X}^{\mathcal{Q}_u}$ of $\mathcal{X}$ such that for all $\mathcal{X}_{l,u} \in \mathcal{X}^{\mathcal{Q}_u}$ the function $\mathcal{X}_{l,u} \ni \boldsymbol{x} \mapsto V_{t_u}(\boldsymbol{x})$ has an analytic extension to a generalized Bernstein ellipse $\mathcal{B}(\mathcal{X}_{l,u}, \varrho_{l,u})$ with $\varrho_{l,u} \in (1, \infty)^d$.*

Under this assumptions we can introduce an extension of Algorithm 1 with splitting in every time step. Assume that at time point t_{u+1} the function $V_{t_{u+1}}$ is approximated by

$$\widehat{V}_{t_{u+1}}(\boldsymbol{x}) = \sum_{l=1}^{Q_{u+1}} I_{\overline{N}_{l,u+1}}(V_{t_{u+1}}\big|_{\mathcal{X}_{l,u+1}})(\boldsymbol{x}) = \sum_{l=1}^{Q_{u+1}} \sum_{0 \le \boldsymbol{j} \le \boldsymbol{N}_{l,u+1}} c^l_{\boldsymbol{j}}(t_{u+1}) p^{l,u+1}_{\boldsymbol{j}}(\boldsymbol{x})$$

with $p^{l,u+1}_{\boldsymbol{j}}(\boldsymbol{x}) := T_{\boldsymbol{j}}(\varphi^{-1}_{\mathcal{X}_{l,u+1}}(\boldsymbol{x}))\mathbb{1}_{\mathcal{X}_{l,u+1}}(\boldsymbol{x})$. At time point t_u this yields for the value function

$$\begin{aligned} V_{t_u}(\boldsymbol{x}) &= f\left(g(t_u, \boldsymbol{x}), \mathbb{E}[V_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}]\right) \\ &\approx f\left(g(t_u, \boldsymbol{x}), \mathbb{E}[\widehat{V}_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}]\right) \\ &= f\Big(g(t_u, \boldsymbol{x}), \mathbb{E}\Big[\sum_{l=1}^{Q_{u+1}} \sum_{0 \le \boldsymbol{j} \le \boldsymbol{N}_{l,u+1}} c^l_{\boldsymbol{j}}(t_{u+1}) p^{l,u+1}_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})\Big|\boldsymbol{X}_{t_u} = \boldsymbol{x}\Big]\Big) \end{aligned}$$

$$= f\Big(g(t_u, \boldsymbol{x}), \sum_{l=1}^{Q_{u+1}} \sum_{0 \le \boldsymbol{j} \le \boldsymbol{N}_{l,u+1}} c_{\boldsymbol{j}}^{l}(t_{u+1}) \mathbb{E}\big[p_{\boldsymbol{j}}^{l,u+1}(\boldsymbol{X}_{t_{u+1}}) | \boldsymbol{X}_{t_u} = \boldsymbol{x}\big]\Big).$$

In order to approximate the value function with a piecewise multivariate Chebyshev interpolation we have to compute the nodal values $\widehat{V}_{t_u}(\boldsymbol{x}_{\boldsymbol{k}}^{l})$ with nodal points $\boldsymbol{x}_{\boldsymbol{k}}^{l} = \varphi_{\mathcal{X}_{l,u}}(\boldsymbol{z}_{\boldsymbol{k}})$ for all $\boldsymbol{k}$ and $\mathcal{X}_{l,u} \in \mathcal{X}^{\mathcal{Q}_u}$, i.e.

$$\widehat{V}_{t_u}(\boldsymbol{x}_{\boldsymbol{k}}^{l}) = f\Big(g(t_u, \boldsymbol{x}_{\boldsymbol{k}}^{l}), \sum_{l=1}^{Q_{u+1}} \sum_{0 \le \boldsymbol{j} \le \boldsymbol{N}_{l,u+1}} c_{\boldsymbol{j}}^{l}(t_{u+1}) \mathbb{E}\big[p_{\boldsymbol{j}}^{l,u+1}(\boldsymbol{X}_{t_{u+1}}) | \boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}^{l}\big]\Big).$$

The values $\widehat{V}_{t_u}(\boldsymbol{x}_{\boldsymbol{k}}^{l})$ can be used to set up $\mathcal{Q}_u$ multivariate Chebyshev interpolations on the sets $\mathcal{X}_{l,u} \in \mathcal{X}^{\mathcal{Q}_u}$ yielding a piecewise multivariate Chebyshev interpolation $\widehat{V}_{t_u}$ of V_{t_u} given by

$$\widehat{V}_{t_u}(\boldsymbol{x}) = \sum_{l=1}^{Q_u} I_{\overline{N}_{l,u}}(V_{t_u}|_{\mathcal{X}_{l,u}})(\boldsymbol{x}) = \sum_{l=1}^{Q_u} \sum_{0 \le \boldsymbol{j} \le \boldsymbol{N}_{l,u}} c_{\boldsymbol{j}}^{l}(t_u) p_{\boldsymbol{j}}^{l,u}(\boldsymbol{x}).$$

The method is described in detail in Algorithm 4.

In the previous section we briefly mentioned that the dynamic Chebyshev algorithm for an American put does not admit an offline-online decomposition. The same holds for the general dynamic Chebyshev algorithm with splitting as presented in this section. In this most general version the algorithm is mostly interesting from a theoretical view point. For any practical application it needs to be tailored to the specific product and payoff. We expect that the method will only be feasible if the number of domains $\mathcal{Q}_u$ is relatively small. Moreover, it must be simple to find the partition $\mathcal{X}^{\mathcal{Q}_u}$ in every time step. For the American put in one dimension this is the case but there is no straightforward extension to higher dimension of this optimal exercise boundary. Additionally, the computation of the generalized moments depends on the transformations $\varphi_{\mathcal{X}_l}$ that might be more complicated than the linear transformation $\tau_{\mathcal{X}}$. A more efficient alternative to this splitting algorithm for the pricing of multivariate early-exercise option is introduced in Section 5.4. This modification of the dynamic Chebyshev algorithm is based on a new pricing approach for European basket options presented in [11].

3.2 Theoretical error analysis

In this section we analyse the convergence behaviour of the dynamic Chebyshev method and its extended version using splitting in two different scenarios. First, we assume that the value function is analytic and second, we consider a value function that is piecewise

Algorithm 4 Dynamic Chebyshev algorithm with splitting

Require: Domain $\mathcal{X} = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_d, \overline{x}_d]$ and time stepping $0 = t_0, \ldots, t_n = T$

procedure INITIAL TIME T
  Find $\mathcal{Q}$-partitions $\mathcal{X}^{\mathcal{Q}_n}$ and fix $\overline{N}_{l,u} \in \mathbb{N}^d$
  for $l = 1$ **to** Q_n **do**
    Define nodal points $\boldsymbol{x}_{\boldsymbol{k}}^l = \varphi_{\mathcal{X}_l}(\boldsymbol{z}_{\boldsymbol{k}}^l)$
    $\widehat{V}_T(\boldsymbol{x}_{\boldsymbol{k}}^l) = g(t_n, \boldsymbol{x}_{\boldsymbol{k}}^l),\ k \in \mathcal{I}(\overline{N}_{l,n})$, derive
    $c_{\boldsymbol{j}}^1(T) = \big(\prod_{i=1}^d \frac{2^{\mathbb{1}_{\{0<j_i<N_1\}}}}{N_{1,n,i}}\big) \sum_{\boldsymbol{k}\in\mathcal{I}(\overline{N}_{l,n})}{}'' \widehat{V}_T(\boldsymbol{x}_{\boldsymbol{k}}^l) T_j(\boldsymbol{z}_{\boldsymbol{k}}^l)$
  Obtain interpolation $\widehat{V}_{t_n}(\boldsymbol{x}) = \sum_{l=1}^{Q_n} \sum_{\boldsymbol{j}\in\mathcal{I}(\overline{N}_{l,n})} c_{\boldsymbol{j}}^l(t_n) p_{\boldsymbol{j}}^{l,n}(\boldsymbol{x})$

procedure ITERATIVE TIME STEPPING FROM $t_{u+1} \to t_u,\ u = n-1, \ldots, 1$
  Given $\widehat{V}_{t_{u+1}}(\boldsymbol{x}) = \sum_{l=1}^{Q_{u+1}} \sum_{\boldsymbol{j}\in\mathcal{I}(\overline{N}_{l,u+1})} c_{\boldsymbol{j}}^l(t_{u+1}) p_{\boldsymbol{j}}^{l,u+1}(\boldsymbol{x})$
  Find $\mathcal{Q}$-partitions $\mathcal{X}^{\mathcal{Q}_n}$ and fix $\overline{N}_{l,u} \in \mathbb{N}^d$
  for $l = 1$ **to** Q_n **do**
    Define new nodal points $\boldsymbol{x}_{\boldsymbol{k}}^l = \varphi_{\mathcal{X}_l}(\boldsymbol{z}_{\boldsymbol{k}}^l)$
    Calculate new nodal values
    $\widehat{V}_{t_u}(\boldsymbol{x}_{\boldsymbol{k}}^l) = f\Big(g(t_u, \boldsymbol{x}_{\boldsymbol{k}}^l), \sum_{l=1}^{Q_{u+1}} \sum_{\boldsymbol{j}\in\mathcal{I}(\overline{N}_{l,u+1})} c_{\boldsymbol{j}}^l(t_{u+1}) \mathbb{E}\big[p_{\boldsymbol{j}}^{l,u+1}(\boldsymbol{X}_{t_{u+1}}) | \boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}^l\big]\Big)$
    Derive coefficients
    $c_{\boldsymbol{j}}^1(t_u) = \big(\prod_{i=1}^d \frac{2^{\mathbb{1}_{\{0<j_i<N_1\}}}}{N_{1,u,i}}\big) \sum_{\boldsymbol{k}\in\mathcal{I}(\overline{N}_{l,u})}{}'' \widehat{V}_{t_u}(\boldsymbol{x}_{\boldsymbol{k}}^l) T_j(\boldsymbol{z}_{\boldsymbol{k}}^l)$
  Obtain multivariate Chebyshev interpolation
  $\widehat{V}_{t_u}(\boldsymbol{x}) = \sum_{l=1}^{Q_u} \sum_{\boldsymbol{j}\in\mathcal{I}(\overline{N}_{l,u})} c_{\boldsymbol{j}}^l(t_u) p_{\boldsymbol{j}}^{l,u}(\boldsymbol{x})$

procedure DERIVING THE SOLUTION AT T=0
  $\widehat{V}_0(\boldsymbol{x}) = \sum_{l=1}^{Q_0} \sum_{\boldsymbol{j}\in\mathcal{I}(\overline{N}_{l,n})} c_{\boldsymbol{j}}^l(0) p_{\boldsymbol{j}}^{l,0}(\boldsymbol{x})$

analytic. The ideas behind the two main proofs can be extended to value function that are continuously differentiable and have bounded mixed derivatives.

3.2.1 Error analysis for analytic value functions

In this section we analyse the error of Algorithm 1, i.e.

$$\varepsilon_{t_u} := \max_{\boldsymbol{x}\in\mathcal{X}} |V_{t_u}(\boldsymbol{x}) - \widehat{V}_{t_u}(\boldsymbol{x})|. \tag{3.8}$$

Two different error sources occur at t_u, the classical interpolation error of the Chebyshev interpolation and a distortion error at the nodal points. With distortion, we refer to the computational noise that comes from the fact that we do not observe the correct function values $V_{t_u}(\boldsymbol{x}_{\boldsymbol{k}})$ at the nodal points but distorted values $\widehat{V}_{t_u}(\boldsymbol{x}_{\boldsymbol{k}})$. We call the error induced from this noise distortion error. Hence, we need the convergence result for the Chebyshev interpolation which incorporates a distortion error at the nodal points, see Proposition 4. We use this result to investigate the error of the dynamic Chebyshev method. First,

we introduce the following assumption.

Assumptions 2. *We assume $\mathcal{X} \ni \boldsymbol{x} \mapsto V_{t_u}(\boldsymbol{x})$ is a real valued function that has an analytic extension to a generalized Bernstein ellipse $\mathcal{B}(\mathcal{X}, \varrho_{t_u})$ with $\varrho_{t_u} \in (1,\infty)^d$ and $\sup_{\boldsymbol{x}\in\mathcal{B}(\mathcal{X},\varrho_{t_u})} |V_{t_u}(\boldsymbol{x})| \leq B_{t_u}$ for $u = 1,\dots,n_T$.*

Proposition 6 provides conditions on the process $\boldsymbol{X}$ and the functions f and g that guarantee Assumptions 2. Under these assumptions, we can apply Proposition 4 to obtain an error bound for the dynamic Chebyshev method at each time step. This error bound has a recursive structure, since the values of V_{t_u} depend on the conditional expectation of $V_{t_{u+1}}$. The interpolation error of the final time step is of form (2.32). At any other time step t_u an additional distortion error by approximating the function values at the nodal points by

$$V_{t_u}(\boldsymbol{x_k}) \approx f\Big(g(t_u, \boldsymbol{x_k}), \sum\nolimits_{\boldsymbol{j}} c_{\boldsymbol{j}}(t_{u+1})\mathbb{E}[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_k}]\Big) = \widehat{V}_{t_u}(\boldsymbol{x_k})$$

comes into play. Proposition 4 yields

$$\varepsilon_{t_u} := \max_{\boldsymbol{x}\in\mathcal{X}} |V_{t_u}(\boldsymbol{x}) - \widehat{V}_{t_u}(\boldsymbol{x})| \leq \varepsilon_{int}(\varrho_{t_u}, \boldsymbol{N}, d, B_{t_u}) + \Lambda_{\overline{N}} F_{t_u},$$

where $F_{t_u} := \max_{\boldsymbol{j}} |V_{t_u}(\boldsymbol{x_j}) - \widehat{V}_{t_u}(\boldsymbol{x_j})|$. The term F_{t_u} depends on the function f and the interpolation error at the previous time step t_{u+1}.

Moreover, two additional error sources can influence the error bound. If there is no closed-form solution for the generalized moments $\mathbb{E}[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_k}]$ a numerical technique, e.g. a numerical quadrature or a Monte Carlo method, introduces an additional error. The former is typically deterministic and bounded whereas the latter is stochastic. In order to incorporate this error in the following error analysis we introduce some additional notation. The conditional expectation can be seen as a linear operator which operates on the vector space of all continuous functions $\mathcal{C}(\mathbb{R}^d)$ with finite L^∞-norm

$$\Gamma_{t_u}^{\boldsymbol{k}} : \mathcal{C}(\mathbb{R}^d) \to \mathbb{R} \quad \text{with} \quad \Gamma_{t_u}^{\boldsymbol{k}}(f) := \mathbb{E}[f(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_k}].$$

We define the subspace of all d-variate polynomials $\mathcal{P}_{\boldsymbol{N}}(\mathcal{X}) := \operatorname{span}\{p_{\boldsymbol{j}},\ 0 \leq \boldsymbol{j} \leq \boldsymbol{N}\}$ equipped with the L^∞-norm. We assume the operator $\Gamma_{t_u}^{\boldsymbol{k}}$ is approximated by a linear operator $\widehat{\Gamma}_{t_u}^{\boldsymbol{k}} : \mathcal{P}_{\boldsymbol{N}}(\mathcal{X}) \to \mathbb{R}$ on $\mathcal{P}_{\boldsymbol{N}}(\mathcal{X})$ which fulfils one of the two following conditions. For all $u = 0,\dots,n_T$ the approximation is either deterministic and the error is bounded by a constant $\overline{\delta}$,

$$\|\Gamma_{t_u}^{\boldsymbol{k}} - \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}\|_{op} := \sup_{\substack{p\in\mathcal{P}_{\boldsymbol{N}} \\ \|p\|=1}} \Big|\Gamma_{t_u}^{\boldsymbol{k}}(p) - \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}(p)\Big| \leq \overline{\delta} \quad \text{for} \quad 0 \leq \boldsymbol{k} \leq \boldsymbol{N} \tag{GM}$$

or the approximation is stochastic and uses M samples of the underlying process and the polynomials p may have stochastic coefficients. In this case we assume the error bound

$$\|\Gamma_{t_u}^{\boldsymbol{k}} - \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}\|_{op} := \sup_{\substack{p\in\mathcal{P}_{\boldsymbol{N}} \\ \|p\|_\infty^\star=1}} \mathbb{E}\left[\left|\Gamma_{t_u}^{\boldsymbol{k}}(p) - \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}(p)\right|\right] \leq \delta^\star(M) \quad \text{for} \quad 0 \leq \boldsymbol{k} \leq \boldsymbol{N} \tag{GM*}$$

with norm $\|p\|_\infty^\star := \max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}[|p(\boldsymbol{x})|]$. In order to incorporate stochasticity of $\widehat{V}_{t_u}(\boldsymbol{x})$, we replace (3.8) by

$$\varepsilon_{t_{u+1}} := \max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}\left[\left|V_{t_u}(\boldsymbol{x}) - \widehat{V}_{t_u}(\boldsymbol{x})\right|\right]. \tag{3.9}$$

Note that in the deterministic case (3.8) and (3.9) coincide. Additionally, a truncation error is introduced by restricting to a compact interpolation domain $\mathcal{X}$. We assume that the conditional expectation of the value function outside this domain is bounded by a constant

$$\mathbb{E}[V_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})\mathbb{1}_{\mathbb{R}^d\setminus\mathcal{X}}|\boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}] \leq \varepsilon_{tr}. \tag{TR}$$

In order to obtain a meaningful convergence result we implicitly assume that this truncation error decays fast enough with increasing size of the domain $\mathcal{X}$. The following theorem provides an error bound for the dynamic Chebyshev method.

Theorem 14. *Let the DPP be given as in Definition 6. Assume the regularity Assumptions 2 hold and the boundedness of the truncation error* (TR)*. Then we have*

$$\varepsilon_{t_u} \leq \sum_{j=u}^{n_T} C^{j-u}\varepsilon_{int}^j \;+\; \Lambda_{\boldsymbol{N}} L_f \sum_{j=u+1}^{n_T} C^{j-(u+1)}(\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j) \tag{3.10}$$

with $\varepsilon_{gm} = \overline{\delta}$ if assumption (GM) *holds and $\varepsilon_{gm} = \delta^\star(M)$ if assumption* (GM*) *holds and $C = \Lambda_{\boldsymbol{N}} L_f(1+\varepsilon_{gm})$, $\overline{V}_j = \max_{\boldsymbol{x}\in\mathcal{X}} |V_{t_j}(\boldsymbol{x})|$ and $\varepsilon_{int}^j = \varepsilon_{int}(\varrho_{t_j}, N, d, B_{t_j})$.*

Proof. Consider a DPP as defined in Definition 6, i.e. we have a Lipschitz continuous function

$$|f(x_1, y_1) - f(x_2, y_2)| \leq L_f(|x_1 - x_2| + |y_1 - y_2|).$$

Assume that the regularity Assumption 2 and the assumption on the truncation error (TR) hold. Then we have to distinguish between the deterministic case (GM) and the stochastic case (GM*). In the first case, the expectation in the error bound can simply be ignored. First, we apply Proposition 4. At time point T there is no random part and

no distortion error. Thus,

$$\max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}\left[|V_T(\boldsymbol{x}) - \widehat{V}_T(\boldsymbol{x})|\right] = \max_{\boldsymbol{x}\in\mathcal{X}} |V_T(\boldsymbol{x}) - \widehat{V}_T(\boldsymbol{x})| \leq \varepsilon_{int}(\varrho_{t_n}, N, d, M_{t_n}).$$

For the ease of notation we will from know on write $\varepsilon_{int}^j = \varepsilon_{int}(\varrho_{t_j}, N, d, M_{t_j})$. We obtain for the error at t_u

$$\max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}\left[|V_{t_u}(\boldsymbol{x}) - \widehat{V}_{t_u}(\boldsymbol{x})|\right] \leq \varepsilon_{int}^u + \Lambda_{\boldsymbol{N}} F(f, t_u) \tag{3.11}$$

with maximal distortion error $F(f, t_u) = \max_{0\leq \boldsymbol{k}\leq \boldsymbol{N}} \mathbb{E}\left[|V_{t_u}(\boldsymbol{x}_{\boldsymbol{k}}) - \widehat{V}_{t_u}(\boldsymbol{x}_{\boldsymbol{k}})|\right]$.

Note that whether (GM) or (GM*) hold, an approximation error of the conditional expectation of $\widehat{V}_{t_{u+1}}$ is made, i.e. $\mathbb{E}[\widehat{V}_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}] = \Gamma_{t_u}^{\boldsymbol{k}}(\widehat{V}_{t_{u+1}}) \approx \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}(\widehat{V}_{t_{u+1}})$. The Lipschitz continuity of f yields

$$\begin{aligned}
\left|V_{t_u}(\boldsymbol{x}_{\boldsymbol{k}}) - \widehat{V}_{t_u}(\boldsymbol{x}_{\boldsymbol{k}})\right| &= \left|f\left(g(t_u, \boldsymbol{x}_{\boldsymbol{k}}), \Gamma_{t_u}^{\boldsymbol{k}}(V_{t_{u+1}})\right) - f\Big(g(t_u, \boldsymbol{x}_{\boldsymbol{k}}), \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}(\widehat{V}_{t_{u+1}})\Big)\right| \\
&\leq L_f\Big(\left|g(t_u, \boldsymbol{x}_{\boldsymbol{k}}) - g(t_u, \boldsymbol{x}_{\boldsymbol{k}})\right| + \left|\Gamma_{t_u}^{\boldsymbol{k}}(V_{t_{u+1}}) - \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}(\widehat{V}_{t_{u+1}})\right|\Big) \\
&= L_f\Big(\left|\Gamma_{t_u}^{\boldsymbol{k}}(V_{t_{u+1}}) - \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}(\widehat{V}_{t_{u+1}})\right|\Big) \\
&\leq L_f\Big(\left|\Gamma_{t_u}^{\boldsymbol{k}}(V_{t_{u+1}}1_{\mathcal{X}}) - \Gamma_{t_u}^{\boldsymbol{k}}(\widehat{V}_{t_{u+1}})\right| + \left|\Gamma_{t_u}^{\boldsymbol{k}}(V_{t_{u+1}}1_{\mathbb{R}^d\setminus\mathcal{X}})\right| \\
&\qquad + \left|\Gamma_{t_u}^{\boldsymbol{k}}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}(\widehat{V}_{t_{u+1}})\right|\Big).
\end{aligned}$$

Next, we consider the expectation for each of the three error terms. For the first term we obtain

$$\begin{aligned}
\mathbb{E}\left[\left|\Gamma_{t_u}^{\boldsymbol{k}}(V_{t_{u+1}}\mathbb{1}_{\mathcal{X}}) - \Gamma_{t_u}^{\boldsymbol{k}}(\widehat{V}_{t_{u+1}})\right|\right] &= \mathbb{E}\left[\left|\mathbb{E}[V_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})\mathbb{1}_{\mathcal{X}} - \widehat{V}_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}]\right|\right] \\
&\leq \max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}\left[|V_{t_{u+1}}(\boldsymbol{x}) - \widehat{V}_{t_{u+1}}(\boldsymbol{x})|\right] = \varepsilon_{t_{u+1}}
\end{aligned}$$

and for the second term we have

$$\mathbb{E}\left[\left|\Gamma_{t_u}^{\boldsymbol{k}}(V_{t_{u+1}}1_{\mathbb{R}^d\setminus\mathcal{X}})\right|\right] \leq \mathbb{E}\left[\varepsilon_{tr}\right] = \varepsilon_{tr}.$$

For the last term we have to distinguish two cases. If we assume (GM) holds, the operator norm yields

$$\begin{aligned}
\left|\Gamma_{t_u}^{\boldsymbol{k}}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}(\widehat{V}_{t_{u+1}})\right| &= \left|\left(\Gamma_{t_u}^{\boldsymbol{k}} - \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}\right)\left(\widehat{V}_{t_{u+1}}\right)\right| \\
&\leq \left|\left|\Gamma_{t_u}^{\boldsymbol{k}} - \widehat{\Gamma}_{t_u}^{\boldsymbol{k}}\right|\right|_{op} \left|\left|\widehat{V}_{t_{u+1}}\right|\right|_{\infty} \\
&\leq \overline{\delta}\left|\left|\widehat{V}_{t_{u+1}}\right|\right|_{\infty}.
\end{aligned}$$

Next, we consider the second case and assume that (GM*) holds. Then we have

$$\mathbb{E}\left[\left|\Gamma^{\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}^{\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}})\right|\right] \leq \left|\left|\Gamma^{\boldsymbol{k}}_{t_u} - \widehat{\Gamma}^{\boldsymbol{k}}_{t_u}\right|\right|_{op} \left|\left|\widehat{V}_{t_{u+1}}\right|\right|^{\star}_{\infty} \leq \delta^{\star}(M) \left|\left|\widehat{V}_{t_{u+1}}\right|\right|^{\star}_{\infty}.$$

Hence in either case the following bound holds

$$\mathbb{E}\left[\left|\Gamma^{\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}^{\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}})\right|\right] \leq \varepsilon_{gm} \max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}\left[|\widehat{V}_{t_{u+1}}(\boldsymbol{x})|\right]$$

with $\varepsilon_{gm} = \overline{\delta}$ if assumption (GM) holds and $\varepsilon_{gm} = \delta^{\star}(M)$ if assumption (GM*) holds. We need an upper bound for the maximum of the Chebyshev approximation

$$\max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}\left[|\widehat{V}_{t_{u+1}}(\boldsymbol{x})|\right] \leq \max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}\left[|\widehat{V}_{t_{u+1}}(\boldsymbol{x}) - V_{t_{u+1}}(\boldsymbol{x})|\right] + \max_{\boldsymbol{x}\in\mathcal{X}} |V_{t_{u+1}}(\boldsymbol{x})| \leq \varepsilon_{t_{u+1}} + \overline{V}_{u+1}$$

with $\overline{V}_{u+1} := \max_{\boldsymbol{x}\in\mathcal{X}} |V_{t_{u+1}}(\boldsymbol{x})|$. Hence, the error bound (3.11) becomes

$$\varepsilon_{t_u} \leq \varepsilon^u_{int} + \Lambda_{\overline{N}} L_f\big((1+\varepsilon_{gm})\varepsilon_{t_{u+1}} + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\big).$$

By induction, we now show (3.10). For $u = n_T$ we have $\varepsilon_{t_{n_T}} \leq \varepsilon^{n_T}_{int}$ and therefore (3.10) holds for $u = n_T$. We assume that for $n_T, \ldots, u+1$ equation (3.10) holds. For the error ε_{t_u} we obtain

$$\begin{aligned}
\varepsilon_{t_u} &\leq \varepsilon^u_{int} + \Lambda_{\boldsymbol{N}} L_f\big((1+\varepsilon_{gm})\varepsilon_{t_{u+1}} + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\big) \\
&\leq \varepsilon^u_{int} + \Lambda_{\boldsymbol{N}} L_f\Big((1+\varepsilon_{gm})\Big(\sum_{j=u+1}^{n_T} C^{j-(u+1)}\varepsilon^j_{int} \;+\; \Lambda_{\boldsymbol{N}} L_f \sum_{j=u+2}^{n_T} C^{j-(u+2)}(\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j)\Big) \\
&\quad + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\Big) \\
&= \varepsilon^u_{int} + C\sum_{j=u+1}^{n_T} C^{j-(u+1)}\varepsilon^j_{int} \;+ \Lambda_{\boldsymbol{N}} L_f\Big(C\sum_{j=u+2}^{n_T} C^{j-(u+2)}(\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j) \\
&\quad + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\Big) \\
&= \varepsilon^u_{int} + \sum_{j=u+1}^{n_T} C^{j-u}\varepsilon^j_{int} \;+ \Lambda_{\boldsymbol{N}} L_f\Big(\sum_{j=u+2}^{n_T} C^{j-(u+1)}(\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j) + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\Big) \\
&= \sum_{j=u}^{n_T} C^{j-u}\varepsilon^j_{int} \;+ \Lambda_{\boldsymbol{N}} L_f \sum_{j=u+1}^{n_T} C^{j-(u+1)}(\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j)
\end{aligned}$$

which was our claim. □

The following corollary provides a simplified version of the error bound (3.10) presenting its decomposition into three different errors (interpolation error ε_{int}, truncation error ε_{tr} and error from the numerical computation of the generalized moments ε_{gm}).

Corollary 1. *Let the setting be as in Theorem 14. Then the error is bounded by*

$$\varepsilon_{t_u} \leq \big(\varepsilon_{int}(\underline{\varrho}, N, d, \overline{B}) + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}\big)\,\widetilde{C}^{n_T+1-u} \tag{3.12}$$

with $\widetilde{C} = \max\{2, C\}$, $\underline{\varrho} = \min_{1\leq u\leq n_T} \varrho_{t_u}$, $\overline{B} = \max_{1\leq u\leq n_T} B_{t_u}$, $\overline{V} = \max_{u\leq j\leq n_T} \overline{V}_j$.

Moreover, if $\varepsilon_{tr} = 0$, $L_f \leq 1$ *and* $N = N_i$, $i = 1, \ldots, d$ *the error bound can be simplified further. Under* (GM*) $\delta^\star(M) \leq c/\sqrt{M}$, $c > 0$ *yields*

$$\varepsilon_{t_u} \leq \tilde{c}_1 \varrho^{-N} \log(N)^{d\,n_T} \;+\; \tilde{c}_2 \log(N)^{d\,n_T} M^{-0.5}.$$

for some constants $\tilde{c}_1, \tilde{c}_2 > 0$. *Under* (GM) *the term* $M^{-0.5}$ *is replaced by* $\overline{\delta}$.

Proof. Assuming $C > 2$ and using the geometric series, the first term in the error bound (3.10) can be rewritten as

$$\sum_{j=u}^{n_T} C^{j-u}\varepsilon_{int}^{j} \leq \overline{\varepsilon}_{int} \sum_{j=u}^{n_T} C^{j-u} = \overline{\varepsilon}_{int} \sum_{k=0}^{n_T-u} C^{k} = \overline{\varepsilon}_{int} \left(\frac{1 - C^{n_T+1-u}}{1-C}\right) \leq \overline{\varepsilon}_{int}\, C^{n_T+1-u},$$

where $\overline{\varepsilon}_{int} = \max_j \varepsilon_{int}^{j} = \max_j \varepsilon_{int}(\varrho_{t_j}, N, d, B_{t_j}) \leq \varepsilon_{int}(\underline{\varrho}, N, d, \overline{B})$ for $\underline{\varrho} = \min_{1\leq u\leq n} \varrho_u$ and $\overline{B} = \max_{1\leq u\leq n} B_{t_u}$. For $C \leq 2$ the sum is bounded by $\overline{\varepsilon}_{int}\, 2^{n_T+1-u}$. Similarly, we obtain for the second term in the error bound (3.10) with $\beta = (\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j)$

$$\Lambda_{\overline{N}} L_f \sum_{j=u+1}^{n_T} C^{j-(u+1)}\beta_j \leq \Lambda_{\boldsymbol{N}} L_f\, \overline{\beta} \sum_{k=0}^{n_T-(u+1)} C^{k} \leq \Lambda_{\boldsymbol{N}} L_f\, \overline{\beta} C^{n_T-u} \leq \overline{\beta} C^{n_T+1-u}$$

where $\overline{\beta} = \max_j \beta_j$. Moreover, we used $\Lambda_{\boldsymbol{N}} L_f \leq \Lambda_{\boldsymbol{N}} L_f(1+\varepsilon_{gm}) = C$ in the last step. Thus, we obtain the following error bound (3.10)

$$\varepsilon_{t_u} \leq (\overline{\varepsilon}_{int} \;+\; \overline{\beta})\,\tilde{C}^{n_T+1-u} = \big(\varepsilon_{int}(\underline{\varrho}, N, d, \overline{B}) + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}\big)\,\tilde{C}^{n_T+1-u},$$

where $\tilde{C} = \max\{2, C\}$ and $\overline{V} = \max_j \overline{V}_j$, which shows (3.12).

Furthermore, for the error bound of the multivariate Chebyshev interpolation holds $\varepsilon_{int}(\underline{\varrho}, N, D, \overline{B}) \leq c_1 \varrho^{-N}$ for a constant $c_1 > 0$ if $N = N_i$, $i = 1, \ldots, d$. For the Lebesgue constant of the Chebyshev interpolation exists a constant $c_2 > 0$ such that

$$\Lambda_{\boldsymbol{N}} \leq \prod_{i=1}^{d} \big(\frac{2}{\pi}\log(N+1) + 1\big) \leq \prod_{i=1}^{d} \big(\frac{4}{\pi} + 1\big)\log(N) \leq c_2 \log(N)^d.$$

Under (GM*), $\delta^\star(M) \leq c/\sqrt{M}$, $c > 0$ yields with $\varepsilon_{tr} = 0$, $L_f \leq 1$

$$\begin{aligned}\varepsilon_{t_u} &\leq \left(\varepsilon_{int}(\underline{\varrho}, N, d, \overline{B}) + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}\right)\left(\Lambda_{\boldsymbol{N}} L_f(1+\varepsilon_{gm})\right)^{n_T+1-u} \\ &\leq \left(c_1\varrho^{-N} + c\overline{V}M^{-0.5}\right)\left(c_2\log(N)^d(1+cM^{-0.5})\right)^{n_T} \\ &\leq \tilde{c}_1\varrho^{-N}\log(N)^{d\,n_T} \;+\; \tilde{c}_2\log(N)^{d\,n_T}M^{-0.5}\end{aligned}$$

and this converges towards zero for $N \to \infty$ if $\sqrt{M} > \log(N)^{d\,n_T}$. If (GM) holds we have $\varepsilon_{gm} = \overline{\delta}$ and the term $M^{-0.5}$ is replaced by $\overline{\delta}$. □

The following proposition provides conditions under which the value function has an analytic extension to some generalized Bernstein ellipse and Assumptions 2 hold.

Proposition 6. *Consider a DPP as defined in* (3.1) *and* (3.2) *with equidistant time-stepping and* $g_t(\boldsymbol{x}) := g(t, \boldsymbol{x})$. *Let* $\boldsymbol{X} = (\boldsymbol{X}_t)_{0\leq t\leq T}$ *be a Markov process with stationary increments. Assume* $e^{\langle\eta,\cdot\rangle}g_{t_u}(\cdot) \in L^1(\mathbb{R}^d)$ *for some* $\boldsymbol{\eta} \in \mathbb{R}^d$ *and* g_{t_u} *has an analytic extension to the generalized Bernstein ellipse* $\mathcal{B}(\mathcal{X}, \varrho_g)$ *for* $u = 0, \ldots, n$. *Furthermore, assume* $f : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ *has an analytic extension to* $\mathbb{C}^2$. *If*

(i) the characteristic function $\varphi^{\boldsymbol{x}}$ *of* $\boldsymbol{X}_{\Delta t}$ *with* $\boldsymbol{X}_0 = \boldsymbol{x}$ *is in* $L^1(\mathbb{R}^d)$ *for every* $\boldsymbol{x} \in \mathcal{X}$,

(ii) for every $\boldsymbol{z} \in \mathbb{R}^d$ *the mapping* $\boldsymbol{x} \mapsto \varphi^{\boldsymbol{x}}(\boldsymbol{z} - i\boldsymbol{\eta})$ *has an analytic extension to* $B(\mathcal{X}, \varrho_\varphi)$ *and there are constants* $\alpha \in (1, 2]$ *and* $c_1, c_2 > 0$ *such that* $\sup_{\boldsymbol{x}\in B(\mathcal{X},\varrho_\varphi)} |\varphi^{\boldsymbol{x}}(\boldsymbol{z})| \leq c_1 e^{-c_2|\boldsymbol{z}|^\alpha}$ *for all* $\boldsymbol{z} \in \mathbb{R}^d$,

then the value function $\boldsymbol{x} \mapsto V_{t_u}(\boldsymbol{x})$ *of the DPP has an analytic extension to* $B(\mathcal{X}, \varrho)$ *with* $\varrho = \varrho_g$.

Proof. At T the value function $\boldsymbol{x} \mapsto V_T(\boldsymbol{x})$ is analytic since $V_T(\boldsymbol{x}) = g_T(\boldsymbol{x})$ and g_T has an analytic extension by assumption. Moreover, $e^{\langle\boldsymbol{\eta},\cdot\rangle}g_T(\cdot) \in L^1(\mathbb{R}^d)$ for some $\boldsymbol{\eta} \in \mathbb{R}^d$. We assume $e^{\langle\boldsymbol{\eta},\cdot\rangle}V_{t_{u+1}}(\cdot) \in L^1(\mathbb{R}^d)$ and $V_{t_{u+1}}$ has an analytic extension to $B(\mathcal{X}, \varrho)$. Then the function

$$\boldsymbol{x} \mapsto V_{t_u}(\boldsymbol{x}) = f\left(g_{t_u}(\boldsymbol{x}), \mathbb{E}[V_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}]\right)$$

is analytic if $\boldsymbol{x} \mapsto \mathbb{E}[V_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}] = \mathbb{E}[V_{t_{u+1}}(\boldsymbol{X}^{\boldsymbol{x}}_{\Delta t})]$ has an analytic extension. From [49, Conditions 3.1] we obtain conditions (A1)-(A4) under which a function of the form $(\boldsymbol{p}^1, \boldsymbol{p}^2) \mapsto \mathbb{E}[f^{\boldsymbol{p}^1}(\boldsymbol{X}^{\boldsymbol{p}^2})]$ is analytic. In our case we only have the parameter $\boldsymbol{p}^2 = \boldsymbol{x}$ and so $\boldsymbol{X}^{p^2} = \boldsymbol{X}^{\boldsymbol{x}}_{\Delta t}$. Condition (A1) is satisfied since $e^{\langle\boldsymbol{\eta},\cdot\rangle}V_{t_{u+1}}(\cdot) \in L^1(\mathbb{R}^d)$ and for (A2)

we have to verify that $|\widehat{V}_{t_{u+1}}(-\boldsymbol{z}-i\boldsymbol{\eta})| \leq c_1 e^{c_2|\boldsymbol{z}|}$ for constants $c_1, c_2 > 0$.

$$\begin{aligned}|\widehat{V}_{t_{u+1}}(-\boldsymbol{z}-i\boldsymbol{\eta})| &= \Big| \int_{\mathbb{R}^d} e^{i\langle \boldsymbol{y}, -\boldsymbol{z}-i\boldsymbol{\eta}\rangle} V_{t_{u+1}} \mathrm{d}\boldsymbol{y} \Big| \\ &\leq \int_{\mathbb{R}^d} |e^{-i\langle \boldsymbol{y},\boldsymbol{z}\rangle}| \left| e^{\langle \boldsymbol{y},\eta\rangle} V_{t_{u+1}}(\boldsymbol{y}) \right| \mathrm{d}\boldsymbol{y} \\ &\leq \|e^{\langle \boldsymbol{\eta},\cdot\rangle} V_{t_{u+1}}(\cdot)\|_{L^1}\end{aligned}$$

and thus (A2) holds. The remaining conditions (A3)-(A4) are equivalent to our conditions (i)-(ii) and [49, Theorem 3.2] yields the analyticity of $\boldsymbol{x} \mapsto \mathbb{E}[V_{t_{u+1}}(\boldsymbol{X}^{\boldsymbol{x}}_{\Delta t})]$ on the Bernstein ellipse $\mathcal{B}(\mathcal{X}, \varrho_\varphi)$. Hence, $\boldsymbol{x} \mapsto V_{t_u}(\boldsymbol{x})$ is a composition of analytic functions and therefore analytic on the intersection of the domains of analyticity $\mathcal{B}(\mathcal{X}, \varrho_\varphi) \cap \mathcal{B}(\mathcal{X}, \varrho_g) = \mathcal{B}(\mathcal{X}, \varrho)$ with $\varrho = \min\{\varrho_g, \varrho_\varphi\}$. It remains to prove that $e^{\langle \boldsymbol{\eta},\cdot\rangle} V_{t_u}(\cdot) \in L^1(\mathbb{R}^d)$. Here the Lipschitz continuity of f yields

$$\|e^{\langle \boldsymbol{\eta},\cdot\rangle} V_{t_u}(\cdot)\|_{L^1} \leq L_f \left(\|e^{\langle \boldsymbol{\eta},\cdot\rangle} g_{t_u}(\cdot)\|_{L^1} + \|e^{\langle \boldsymbol{\eta},\cdot\rangle} V_{t_{u+1}}(\cdot)\|_{L^1} \right) < \infty.$$

□

Often, the discrete time problem (3.1) and (3.2) is an approximation of a continuous time problem, for example if a Bermudan option is used as a proxy for an American option. In this case, we are interested in the error behaviour for $n \to \infty$.

Remark 1. *Assume the setup of Corollary 1. Moreover, assume that $\varepsilon_{tr} = \varepsilon_{gm} = 0$. If we let N and n go to infinity, we have to ensure that the error bound tends to zero. We use that $\varepsilon_{int}(\underline{\varrho}, N, d, \overline{B}) \leq C_1 \varrho^{-\underline{N}}$ for a constant $C_1 > 0$ and $\underline{N} = \min_i N_i$. The following condition on the relation between n_T and N ensures convergence*

$$n_T < \frac{\log(\varrho)}{C_1 d} \cdot \frac{\underline{N}}{\log(\Lambda_{\boldsymbol{N}}) + \log(L_f)} + 1.$$

3.2.2 Error analysis for piecewise analytic value functions

In this section we investigate the error decay of the dynamic Chebyshev algorithm with splitting if the value function is piecewise analytic. The procedure is similar to the non-splitting algorithm. We require a convergence result for the multivariate Chebyshev interpolation with distortion. Then we are in position to prove an analogous convergence result. We start with the convergence result for the static Chebyshev interpolation on a $\mathcal{Q}$-partition.

Proposition 7. *Let $\mathcal{X} \ni \boldsymbol{x} \mapsto f(\boldsymbol{x})$ for $\mathcal{X} \subset \mathbb{R}^d$ be a real valued function. Assume we have a $\mathcal{Q}$-partition of $\mathcal{X}$ such that $f\big|_{\mathcal{X}_l}$ has an analytic extension to some generalized*

Bernstein ellipse $\mathcal{B}(\mathcal{X}_l, \varrho_l)$ *for* $\varrho_l \in (1,\infty)^d$ *and* $\sup_{\boldsymbol{x}\in\mathcal{B}(\mathcal{X}_l,\varrho_l)} |g(\boldsymbol{x})| \leq B_l$ *for* $l = 1,\dots,\mathcal{Q}$. *Assume distorted values* $f^{\varepsilon}(\boldsymbol{x}_{\boldsymbol{k}}^l) = f(\boldsymbol{x}_{\boldsymbol{k}}^l) + \varepsilon(\boldsymbol{x}_{\boldsymbol{k}}^l)$ *with* $|\varepsilon(\boldsymbol{x}_{\boldsymbol{k}}^l)| \leq \bar{\varepsilon}_l$ *for all* $0 \leq \boldsymbol{k} \leq \boldsymbol{N}$, $l = 1,\dots,\mathcal{Q}$. *Then*

$$\max_{\boldsymbol{x}\in\mathcal{X}} \left| f(\boldsymbol{x}) - I_{\overline{N}^*}(f^{\varepsilon})(\boldsymbol{x}) \right| \leq \max_{1\leq l\leq\mathcal{Q}} \varepsilon_{int}(\varrho_l, \boldsymbol{N}_l, d, B_l) + \bar{\varepsilon}_l \Lambda_{\boldsymbol{N}_l}. \tag{3.13}$$

where $\varepsilon_{int}(\varrho_l, \boldsymbol{N}_l, d, B_l)$ *is the error bound of the tensor based Chebyshev interpolation as shown in Theorem 9.*

Proof. Let $\boldsymbol{x} \in \mathcal{X}$. If $\boldsymbol{x} \in \mathcal{X}_l$, then the function $f_l := f\big|_{\mathcal{X}_l}$ satisfies the assumptions from Proposition 4 with B_l and the generalized Bernstein ellipse $\mathcal{B}(\mathcal{X}_l, \varrho_l)$. This yields

$$\max_{\boldsymbol{x}\in\mathcal{X}_l} \left| f_l(\boldsymbol{x}) - I_{\boldsymbol{N}_l}(f_l^{\varepsilon})(\boldsymbol{x}) \right| \leq \varepsilon_{int}(\varrho_l, \boldsymbol{N}_l, d, B_l) + \bar{\varepsilon}_l \Lambda_{\boldsymbol{N}_l}.$$

From $\max_{\boldsymbol{x}\in\mathcal{X}} \left| f(\boldsymbol{x}) - I_{\overline{N}^*}(f^{\varepsilon})(\boldsymbol{x}) \right| \leq \max_{1\leq l\leq\mathcal{Q}} \max_{\boldsymbol{x}\in\mathcal{X}_l} \left| f_l(\boldsymbol{x}) - I_{\overline{N}_l}(f_l^{\varepsilon})(\boldsymbol{x}) \right|$ follows the assertion directly. □

In the following, we use the result from Proposition 7 to investigate the error behaviour of the dynamic Chebyshev algorithm with splitting of $\mathcal{X}$ in several sub-domains, i.e. applying on each sub-domain a Chebyshev interpolation. Note that, in this case, we allow the splitting of the domain $\mathcal{X}$ at each time step t_u into sub-domains as in (3.6) and that, between different time steps, the number of sub-domains may change. Additionally, we allow the use of different numbers of nodal points in the Chebyshev interpolations $\boldsymbol{N}_{l,u} = (N_{1,l,u},\dots,N_{d,l,u})$ at each time step t_u and on each sub-interval. We define the additional notation $V_{l,t_u} := V_{t_u}\big|_{\mathcal{X}_{l,t_u}}$.

Similar to the error analysis for the dynamic Chebyshev algorithm we have to incorporate additional error sources such as the truncation error and the generalized moment error. We assume, that the truncation error is bounded by a constant ε_{TR}, i.e.

$$\mathbb{E}[V_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})\mathbb{1}_{\mathbb{R}^d\setminus\mathcal{X}}|\boldsymbol{X}_{t_u} = \boldsymbol{x}] \leq \varepsilon_{TR} \quad \text{for} \quad \boldsymbol{x} \in \mathcal{X}. \tag{3.14}$$

Moreover, a second error comes into play when the conditional expectations cannot be computed perfectly,

$$\left| \Gamma_{t_u,t_{u+1}}(p_{\boldsymbol{j}})(\boldsymbol{x}_{\boldsymbol{k}}) - \Gamma^{\delta}_{t_u,t_{u+1}}(p_{\boldsymbol{j}})(\boldsymbol{x}_{\boldsymbol{k}}) \right| \leq \overline{\delta}. \tag{3.15}$$

In order to incorporate the generalized moments error in the following error analysis we introduce some additional notation. The notation is a modification of the notation we introduced for the dynamic Chebyshev algorithm without splitting. We define linear

operators which operate on the vector space of all continuous functions $\mathcal{C}(\mathbb{R}^d)$ with finite L^∞-norm by

$$\Gamma_{t_u}^{l,\boldsymbol{k}} : \mathcal{C}(\mathbb{R}^d) \to \mathbb{R} \quad \text{with} \quad \Gamma_{t_u}^{\boldsymbol{k}}(f) := \mathbb{E}[f(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}^l].$$

for $\boldsymbol{x}_{\boldsymbol{k}}^l = \tau_{\mathcal{X}_{l,u}}(\boldsymbol{z}_{\boldsymbol{k}})$. We define the subspace of all continuous functions which are piecewise d variate polynomials $\mathcal{P}_{\overline{N}^\star}(\mathcal{X}) := \text{span}\{p_{\boldsymbol{j}}^{l,u},\ 0 \le \boldsymbol{j} \le \boldsymbol{N}_{l,u},\ l = 1,\dots,\mathcal{Q}_u,\ u = 0,\dots,n_T\}$ equipped with the L^∞-norm.

We assume the operator $\Gamma_{t_u}^{l,\boldsymbol{k}}$ is approximated by a linear operator $\widehat{\Gamma}_{t_u}^{l,\boldsymbol{k}} : \mathcal{P}_{\overline{N}^\star}(\mathcal{X}_{l,u}) \to \mathbb{R}$ on $\mathcal{P}_{\overline{N}^\star}(\mathcal{X}_{l,u})$ which fullfills one of the two following conditions. For all $u = 0,\dots,n_T$ the approximation is either deterministic and the error is bounded by a constant $\overline{\delta}$,

$$\|\Gamma_{t_u}^{l,\boldsymbol{k}} - \widehat{\Gamma}_{t_u}^{l,\boldsymbol{k}}\|_{op} := \sup_{\substack{p \in \mathcal{P}_{\overline{N}}^\star \\ \|p\|=1}} \left|\Gamma_{t_u}^{l,k}(p) - \widehat{\Gamma}_{t_u}^{l,k}(p)\right| \le \overline{\delta} \quad \text{for} \quad 0 \le \boldsymbol{k} \le \boldsymbol{N}_{l,u} \tag{GMsp}$$

with $l = 1,\dots,\mathcal{Q}_u$, or the approximation is stochastic and uses M samples of the underlying process and the polynomials p may have stochastic coefficients. In this case we assume the error bound

$$\|\Gamma_{t_u}^{l,\boldsymbol{k}} - \widehat{\Gamma}_{t_u}^{l,\boldsymbol{k}}\|_{op} := \sup_{\substack{p \in \mathcal{P}_{\overline{N}}^\star \\ \|p\|_\infty^\star=1}} \mathbb{E}\left[\left|\Gamma_{t_u}^{l,\boldsymbol{k}}(p) - \widehat{\Gamma}_{t_u}^{l,\boldsymbol{k}}(p)\right|\right] \le \delta^\star(M) \quad \text{for} \quad 0 \le \boldsymbol{k} \le \boldsymbol{N}_{l,u} \tag{GMsp*}$$

for $l = 1,\dots,\mathcal{Q}_u$ with norm $\|p\|_\infty^\star = \max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}[|p(\boldsymbol{x})|]$. The following error results incorporates both error types.

Theorem 15. *Let a Dynamic Programming Principle be given as in* (3.1) *and* (3.2) *and assume Assumptions 1 hold. Further, let $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be Lipschitz continuous with constant L_f and assume* (3.14) *holds. Then we obtain*

$$\varepsilon_{t_u} \le \sum_{j=u}^{n_T} C^{j-u} \tilde{\varepsilon}_{int}^j \ + \Lambda_{\boldsymbol{N}^\star} L_f \sum_{j=u+1}^{n_T} C^{j-(u+1)}(\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j) \tag{3.16}$$

with

$$\tilde{\varepsilon}_{int}^u = \max_{l=1,\dots,\mathcal{Q}_u} \varepsilon_{int}(\varrho_{l,t_u}, \boldsymbol{N}_{l,u}, d, B_{l,t_u})$$

and $C = \Lambda_{\boldsymbol{N}^\star} L_f(1 + \varepsilon_{gm})$, $\varepsilon_{gm} = \overline{\delta}$ if (GMsp) *holds and $\varepsilon_{gm} = \delta^\star$ if* (GMsp*) *holds.*

Proof. The proof is based on the proof of Theorem 14. Consider a piecewise analytic

DPP as in Assumptions 1, i.e. we have a Lipschitz continuous function

$$|f(x_1, y_1) - f(x_2, y_2)| \leq L_f(|x_1 - x_2| + |y_1 - y_2|).$$

Moreover assume for the truncation error holds (TR) and either (GMsp) or (GMsp*) hold. In the first case, the expectation in the error bound can simply be ignored. At the inital time $t_{n_T} = T$, we apply Proposition 7 with $\bar{\varepsilon}_l = 0$ resulting in,

$$\max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}[|V_T(\boldsymbol{x}) - \widehat{V}_{t_n}(\boldsymbol{x})|] = \max_{\boldsymbol{x}\in\mathcal{X}} |V_T(\boldsymbol{x}) - \widehat{V}_{t_n}(\boldsymbol{x})| \leq \max_{l=1,\dots,\mathcal{Q}_n} \varepsilon_{int}(\varrho_{l,t_n}, \boldsymbol{N}_{l,n}, d, B_{l,t_n}).$$

At time step t_u, we can again apply Proposition 7 and obtain

$$\max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}[|V_{t_u}(\boldsymbol{x}) - \widehat{V}_{t_u}(\boldsymbol{x})|] \leq \max_{l=1,\dots,\mathcal{Q}_u} \left(\varepsilon_{int}(\varrho_{l,t_u}, \boldsymbol{N}_{l,u}, d, B_{l,t_u}) + \Lambda_{\boldsymbol{N}_{l,u}} F(f, t_u, l)\right)$$

with

$$F(f, t_u, l) = \max_{0\leq \boldsymbol{k}\boldsymbol{N}_{l,u}} \mathbb{E}[|V_{t_u}(\boldsymbol{x}^l_{\boldsymbol{k}}) - \widehat{V}_{t_u}(\boldsymbol{x}^l_{\boldsymbol{k}})|] \qquad \text{for} \qquad \boldsymbol{x}^l_{\boldsymbol{k}} = \varphi_{\mathcal{X}_{l,u}}(\boldsymbol{z}_{\boldsymbol{k}}).$$

For the ease of notation we will write $\varepsilon^{l,u}_{int}$ instead of $\varepsilon_{int}(\varrho_{l,t_u}, \boldsymbol{N}_{l,u}, d, B_{l,t_u})$. The following proof is analogue to the proof of Theorem 14. Note that in both cases (GMsp) or (GMsp*) an approximation error of the conditional expectation is made, i.e. $\mathbb{E}[\widehat{V}_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = \boldsymbol{x}^l_{\boldsymbol{k}}] = \Gamma^{l,\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}}) \approx \widehat{\Gamma}^{l,\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}})$. The Lipschitz continuity of f yields

$$\begin{aligned}
\left|V_{t_u}(\boldsymbol{x}^l_{\boldsymbol{k}}) - \widehat{V}_{t_u}(\boldsymbol{x}^l_{\boldsymbol{k}})\right| &\leq L_f\Big(\left|\Gamma^{l,\boldsymbol{k}}_{t_u}(V_{t_{u+1}}) - \widehat{\Gamma}^{l,\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}})\right|\Big) \\
&\leq L_f\Big(\left|\Gamma^{l,\boldsymbol{k}}_{t_u}(V_{t_{u+1}}1_{\mathcal{X}}) - \Gamma^{l,\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}})\right| + \left|\Gamma^{l,\boldsymbol{k}}_{t_u}(V_{t_{u+1}}1_{\mathbb{R}^d\setminus\mathcal{X}})\right| \\
&\qquad + \left|\Gamma^{l,\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}^{l,\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}})\right|\Big).
\end{aligned}$$

Next, we consider the expectation for each of the three error terms. For the first term we obtain

$$\mathbb{E}\left[\left|\Gamma^{l,\boldsymbol{k}}_{t_u}(V_{t_{u+1}}1_{\mathcal{X}}) - \Gamma^{l,\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}})\right|\right] \leq \max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}\left[|V_{t_{u+1}}(\boldsymbol{x}) - \widehat{V}_{t_{u+1}}(\boldsymbol{x})|\right] = \varepsilon_{t_{u+1}}$$

and for the second term we have

$$\mathbb{E}\left[\left|\Gamma^{l,\boldsymbol{k}}_{t_u}(V_{t_{u+1}}1_{\mathbb{R}^d\setminus\mathcal{X}})\right|\right] \leq \mathbb{E}\left[\varepsilon_{tr}\right] = \varepsilon_{tr}.$$

For the last term we have to distinguish two cases. If we assume (GMsp) holds, the operator norm yields

$$\left|\Gamma^{l,\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}^{l,\boldsymbol{k}}_{t_u}(\widehat{V}_{t_{u+1}})\right| \leq \left|\left|\Gamma^{l,\boldsymbol{k}}_{t_u} - \widehat{\Gamma}^{l,\boldsymbol{k}}_{t_u}\right|\right|_{op} \left|\left|\widehat{V}_{t_{u+1}}\right|\right|_{\infty} \leq \overline{\delta} \left|\left|\widehat{V}_{t_{u+1}}\right|\right|_{\infty}.$$

Next, we consider the second case and assume that (GMsp*) holds. Then we have

$$\mathbb{E}\left[\left|\Gamma_{t_u}^{l,\boldsymbol{k}}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}_{t_u}^{l,\boldsymbol{k}}(\widehat{V}_{t_{u+1}})\right|\right] \leq \left|\left|\Gamma_{t_u}^{l,\boldsymbol{k}} - \widehat{\Gamma}_{t_u}^{l,\boldsymbol{k}}\right|\right|_{op} \left|\left|\widehat{V}_{t_{u+1}}\right|\right|_{\infty}^{\star} \leq \delta^{\star}(M) \left|\left|\widehat{V}_{t_{u+1}}\right|\right|_{\infty}^{\star}.$$

Hence in either case the following bound holds

$$\mathbb{E}\left[\left|\Gamma_{t_u}^{l,\boldsymbol{k}}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}_{t_u}^{l,\boldsymbol{k}}(\widehat{V}_{t_{u+1}})\right|\right] \leq \varepsilon_{gm} \max_{x\in\mathcal{X}} \mathbb{E}\left[|\widehat{V}_{t_{u+1}}(x)|\right]$$

with $\varepsilon_{gm} = \overline{\delta}$ if assumption (GMsp) holds and $\varepsilon_{gm} = \delta^{\star}(M)$ if assumption (GMsp*) holds. We can use the upper bound for $\widehat{V}_{t_{u+1}}$ from the proof of Theorem 14

$$\max_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}\left[|\widehat{V}_{t_{u+1}}(\boldsymbol{x})|\right] \leq \varepsilon_{t_{u+1}} + \overline{V}_{u+1}$$

with $\overline{V}_{u+1} := \max_{\boldsymbol{x}\in\mathcal{X}} |V_{t_{u+1}}(\boldsymbol{x})|$. Hence, the error bound becomes

$$\begin{aligned}\varepsilon_{t_u} &\leq \max_{l=1,\dots,\mathcal{Q}_u} \varepsilon_{int}^{l,u} + \Lambda_{\boldsymbol{N}_{l,u}} L_f\big((1+\varepsilon_{gm})\varepsilon_{t_{u+1}} + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\big) \\ &\leq \tilde{\varepsilon}_{int}^{u} + \Lambda_{\boldsymbol{N}^{\star}} L_f\big((1+\varepsilon_{gm})\varepsilon_{t_{u+1}} + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\big)\end{aligned}$$

with $\tilde{\varepsilon}_{int}^{u} = \max_{l=1,\dots,\mathcal{Q}_u} \varepsilon_{int}^{l,u}$ and

$$\boldsymbol{N}^{\star} = (N_1^{\star},\dots,N_D^{\star}) \in \mathbb{N}^d \quad \text{with} \quad N_i^{\star} = \max_{u=0,\dots,n} \max_{l=1,\dots,\mathcal{Q}_u} N_{i,l,u} \quad \text{for} \quad i = 1,\dots,d.$$

Following the induction in the proof of Theorem 14 with $\tilde{\varepsilon}_{int}^{u}$ instead of ε_{int}^{u} yields our results

$$\varepsilon_{t_u} \leq \sum_{j=u}^{n_T} C^{j-u}\tilde{\varepsilon}_{int}^{j} \; + \Lambda_{\boldsymbol{N}^{\star}} L_f \sum_{j=u+1}^{n_T} C^{j-(u+1)}(\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j)$$

with $C = \Lambda_{\boldsymbol{N}^{\star}} L_f(1+\varepsilon_{gm})$. □

Similar to Proposition 6, we can find conditions for the interpolation on several intervals such that analyticity is given as in Assumption 1.

3.3 Computation of generalized moments

One of the key aspects of the dynamic Chebyshev method is the shift of all model dependent calculations into a pre-computation step. Naturally, the question arises how the resulting generalized moments (3.4) can be derived, i.e. how we can compute

$$\Gamma_{\boldsymbol{j},\boldsymbol{k}}(t_u) = \mathbb{E}[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}] = \mathbb{E}[T_{\boldsymbol{j}}(\tau_{\mathcal{X}}^{-1}(\boldsymbol{X}_{t_{u+1}}))\mathbb{1}_{\mathcal{X}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}]$$

in an efficient way. Defining $\boldsymbol{Y} = \tau_{\mathcal{X}}^{-1}(\boldsymbol{X}_{t_{u+1}})$ conditioned on $\boldsymbol{X}_{t_u} = \boldsymbol{x_k}$ we obtain

$$\mathbb{E}[T_{\boldsymbol{j}}(\boldsymbol{Y})\mathbb{1}_{[-1,1]^d}(\boldsymbol{Y})].$$

In this section, we discuss different approaches how these generalized moments can be computed. Depending on the distribution of $\boldsymbol{Y}$, we can divide the approaches into two main groups. For some models, we can find analytic formulas for the generalized moments. Otherwise we have to employ a numerical integration routine. We can summarize the approaches as follows:

Case 1: Analytic formula
For an univariate normally distributed variable Y, the truncated moments are explicitly available. More generally, [79] provide a recursive formula for the multivariate truncated moments $\mathbb{E}[\boldsymbol{Y}^{\boldsymbol{m}}\mathbb{1}_{[-1,1]^d}(\boldsymbol{Y})]$ if $\boldsymbol{Y}$ is multivariate Gaussian. In this section, we will provide similar formulas for the expectation of the (truncated) Chebyshev polynomials. First for an univariate Gaussian variable Y and then for a multivariate variable $\boldsymbol{Y}$.

Case 2: Numerical integration
In general, there will be no analytic formulas for the (generalized) moments of the (multivariate) random variable $\boldsymbol{Y}$. Hence numerical integration techniques come into play. Depending on the availability, either quadrature techniques using the density or characteristic function, finite difference solvers for the associated PDE or as the most general case Monte Carlo simulation can be used as numerical integration techniques. The choice of the numerical integration technique depends on the underlying stochastic model and if the density or characteristic function is available in closed form.

Case 3: Semi-analytic approach
In some models, we can combine the analytic formula and numerical integration. Assume their is no analytic formula for the double truncated moments $\mathbb{E}[\boldsymbol{Y}^{\boldsymbol{m}}\mathbb{1}_{[-1,1]^d}(\boldsymbol{Y})]$ but for the non truncated moments $\mathbb{E}[\boldsymbol{Y}^{\boldsymbol{m}}]$. If the probability that $\boldsymbol{Y} \in [-1,1]^d$ is close to 1 we can replace $\mathbb{E}[\boldsymbol{Y}^{\boldsymbol{m}}\mathbb{1}_{[-1,1]^d}(\boldsymbol{Y})]$ by $\mathbb{E}[\boldsymbol{Y}^{\boldsymbol{m}}]$ and use the identity $T_j(x) = \sum_{i=0}^{j} a_i x^i$ to obtain the generalized moments $\mathbb{E}[T_{\boldsymbol{j}}(\boldsymbol{Y})\mathbb{1}_{[-1,1]^d}(\boldsymbol{Y})]$. For this ansatz we require a distribution with a fast decay of the tail distribution in order to control the error and the stability of this approach. In terms of the stochastic process $\boldsymbol{X}_t$ this means that for all $\boldsymbol{x_k}$ where $\mathbb{E}[\boldsymbol{X}_{t_{u+1}}^{\boldsymbol{m}}\mathbb{1}_{[-1,1]^d}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_k}]$ is close to $\mathbb{E}[\boldsymbol{X}_{t_{u+1}}^{\boldsymbol{m}}|\boldsymbol{X}_{t_u} = \boldsymbol{x_k}]$ we use an analytic formula. For all other $\boldsymbol{x_k}$ we compute $\mathbb{E}[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_k}]$ numerically. Polynomial processes are a class of stochastic processes for which analytic formulas for the moments $\mathbb{E}[\boldsymbol{Y}^{\boldsymbol{m}}]$ are often available.

We start with an analytic formula for the generalized moments in one dimension. An

extension of this formula to the multivariate moments can be found in Appendix B. In the remaining part of this section we discuss several numerical integration methods that can be employed in the pre-computation step of the dynamic Chebyshev method.

3.3.1 Analytic expressions for the generalized moments

Throughout this subsection we assume that the stochastic process $(X_t)_{t\geq 0}$ is a one-dimensional process that has normally distributed increments. This setting includes import models such as the Black-Scholes model for equities or the Hull-White and the Black-Karasinski model for short rates. In this setting, the generalized moments can be written as

$$\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u}=x_k] = \mathbb{E}[T_j(\tau_{\mathcal{X}}^{-1}(X_{t_{u+1}}))1_{\mathcal{X}}(X_{t_{u+1}})|X_{t_u}=x_k] = \mathbb{E}[T_j(Y)\mathbb{1}_{[-1,1]}(Y)]$$

for $Y=\tau_{\mathcal{X}}^{-1}(X_{t_{u+1}})$ conditioned on $X_{t_u}=x_k$. As $\tau_{\mathcal{X}}$ is linear the random variable Y is also normally distributed. The following proposition formalizes this approach.

Proposition 8. *Assume that $(X_t)_{t\geq 0}$ is a one-dimensional stochastic process with normally distributed increments, i.e.*

$$X_{t_{u+1}}|X_{t_u}=x_0 \sim \mathcal{N}\big(\mu(x_0,t_u,\Delta t),\sigma^2(x_0,t_u,\Delta t)\big)$$

for $\Delta t = t_{u+1}-t_u$. In this case we obtain for the generalized moments

$$\begin{aligned}&\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u}=x] = \mathbb{E}[T_j(Y)\mathbb{1}_{[-1,1]}(Y)]\\ &Y\sim\mathcal{N}\Big(1-2\frac{\overline{x}-\mu(x_0,t_u,\Delta t)}{\overline{x}-\underline{x}},\big(\frac{2}{\overline{x}-\underline{x}}\big)^2\sigma^2(x_0,t_u,\Delta t)\Big).\end{aligned}$$

Proof. From the properties of a normally distributed variable follows

$$\mathbb{E}\big[p_j(X_{t_{u+1}})\big|X_{t_u}=x\big] = \mathbb{E}\big[p_j\big(\mu(x_0,t_u,\Delta t)+\sigma(x_0,t_u,\Delta t)Z\big)\big] \quad Z\sim\mathcal{N}(0,1).$$

The definition of the transformed Chebyshev polynomials p_j and the inverse of the linear transformation $\tau_{[\underline{x},\overline{x}]}$ yield

$$\begin{aligned}&\mathbb{E}\big[p_j\big(\mu(x_0,t_u,\Delta t)+\sigma(x_0,t_u,\Delta t)Z\big)\big]\\ &= \mathbb{E}[T_j\big(\tau^{-1}_{[\underline{x},\overline{x}]}(\mu(x_0,t_u,\Delta t)+\sigma(x_0,t_u,\Delta t)Z\big)\mathbb{1}_{[\underline{x},\overline{x}]}\big(\mu(x_0,t_u,\Delta t)+\sigma(x_0,t_u,\Delta t)Z\big)]\\ &= \mathbb{E}\Big[T_j\Big(1-2\frac{\overline{x}-\big(\mu(x_0,t_u,\Delta t)+\sigma(x_0,t_u,\Delta t)Z\big)}{\overline{x}-\underline{x}}\Big)\mathbb{1}_{[\underline{x},\overline{x}]}\big(\mu(x_0,t_u,\Delta t)+\sigma(x_0,t_u,\Delta t)Z\big)\Big]\\ &= \mathbb{E}[T_j(Y)\mathbb{1}_{[-1,1]}(Y)]\end{aligned}$$

with Y defined as

$$Y = 1 - 2\frac{\overline{x} - x}{\overline{x} - \mu(x_0, t_u, \Delta t)} + \frac{2}{\overline{x} - \underline{x}}\sigma(x_0, t_u, \Delta t)Z$$

and we used that for a linear transformation holds

$$\begin{aligned}
&\mathbb{1}_{[\underline{x},\overline{x}]}\big(\mu(x_0, t_u, \Delta t) + \sigma(x_0, t_u, \Delta t)Z\big) \\
&= \mathbb{1}_{[\tau^{-1}_{[\underline{x},\overline{x}]}(\underline{x}),\tau^{-1}_{[\underline{x},\overline{x}]}(\overline{x})]}\big(\tau^{-1}_{[\underline{x},\overline{x}]}\big(\mu(x_0, t_u, \Delta t) + \sigma(x_0, t_u, \Delta t)Z\big)\big) \\
&= \mathbb{1}_{[-1,1]}(Y).
\end{aligned}$$

The properties of the normal distribution yield our claim. □

In the proposition we considered the most general case where both drift and volatility can depend on the starting value x_0 and the time step Δt. In many models that are of interest we can simplify this expression. Assume that X_t is the log-stock price in a Black-Scholes model given by the SDE

$$\mathrm{d}X_t = (r - 0.5\sigma^2)\mathrm{d}t + \sigma \mathrm{d}W_t \quad X_0 = x_0$$

for interest rate r and volatility σ. In this case $X_{t_{u+1}}|X_{t_u} = x_k \sim \mathcal{N}(x_k + \Delta t\, b, \Delta t\sigma^2)$ with $b = (r - \frac{1}{2}\sigma^2)$ and for the generalized moments holds

$$\begin{aligned}
&\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x] = \mathbb{E}[T_j(Y)\mathbb{1}_{[-1,1]}(Y)] \\
&Y \sim \mathcal{N}\Big(1 - 2\frac{\overline{x} - x}{\overline{x} - \underline{x}} + \frac{2}{\overline{x} - \underline{x}}\Delta t\, b, \big(\frac{2}{\overline{x} - \underline{x}}\big)^2\Delta t\sigma^2\Big).
\end{aligned}$$

Similar results can be obtained for the Hull-White model or the Black-Karasinski model.

If the stochastic model does not omit normally distributed increments, the presented formulas do not hold. However, in many models it is possible to approximate $X_{t_{u+1}}|X_{t_u} = x_0$ with a normally distributed variable if Δt is small. We assume that the process $(X_t)_{t\geq 0}$ is modelled by the following SDE

$$\mathrm{d}X_t = \alpha(t, X_t)\mathrm{d}t + \beta(t, X_t)\mathrm{d}W_t, \quad X_0 = x_0.$$

Then we can discretize the SDE using the Euler—Maruyama method and one time step of length $\Delta t = t_{u+1} - t_u$ is given by

$$X_{t_{u+1}} \approx X_{t_u} + \alpha(t_u, X_{t_u})\Delta t + \beta(t_u, X_{t_u})\sqrt{\Delta t}Z \qquad Z \sim \mathcal{N}(0, 1).$$

Hence, $X_{t_{u+1}}|X_{t_u} = x_0$ is approximately normally distributed with

$$X_{t_{u+1}}|X_{t_u} = x_0 \sim \mathcal{N}(x + \alpha(t_u, x_0)\Delta t, \beta(t_u, x_0)^2\Delta t).$$

We can apply Proposition 8 and obtain for the conditional expectations of the Chebyshev polynomials

$$\begin{aligned}&\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x] \approx \mathbb{E}[T_j(Y)\mathbb{1}_{[-1,1]}(Y)]\\ &Y \sim \mathcal{N}\Big(1 - 2\frac{\overline{x} - x}{\overline{x} - \underline{x}} + \frac{2\alpha(t_u, x)}{\overline{x} - \underline{x}}\Delta t,\ \big(\frac{2}{\overline{x} - \underline{x}}\big)^2\Delta t\beta(t_u, x)^2\Big).\end{aligned}$$

In situations where we would typically use Monte Carlo simulation in order to compute the generalized moments we can now use the same analytic formula as if the increments where normally distributed. We use the same time discretization as in the Monte Carlo approach but the analytic formula allows us to omit any simulation noise.

So far we also assumed that we are interested in the expectation of p_j on the whole domain $[\underline{x}, \overline{x}]$. The following results provides a similar result if we are only interested in a subdomain $[c, d] \subset [\underline{x}, \overline{x}]$.

Proposition 9. *Assume we are in the setting of Proposition 8. Let $\underline{x} \leq a \leq c \leq \overline{x}$, then we have*

$$\mathbb{E}[p_j(X_{t_{u+1}})\mathbb{1}_{[a,c]}(X_{t_{u+1}})|X_{t_u} = x] = \mathbb{E}[T_j(Y)\mathbb{1}_{[l,u]}(Y)]$$

with $l = \tau^{-1}_{[\underline{x},\overline{x}]}(a)$, $u = \tau^{-1}_{[\underline{x},\overline{x}]}(c)$ and

$$Y \sim \mathcal{N}\Big(1 - 2\frac{\overline{x} - x}{\overline{x} - \underline{x}} + \frac{2}{\overline{x} - \underline{x}}\Delta t\, b,\ \big(\frac{2}{\overline{x} - \underline{x}}\big)^2\Delta t\sigma^2\Big).$$

Proof. The proof is exactly the same as for Proposition 8. Only for the indicator function we obtain

$$\mathbb{1}_{[\underline{a},\overline{c}]}(x + X_{\Delta t})] = \mathbb{1}_{[\tau^{-1}_{[\underline{x},\overline{x}]}(a), \tau^{-1}_{[\underline{x},\overline{x}]}(c)]}(\tau^{-1}_{[\underline{x},\overline{x}]}(x + X_{\Delta t})) = \mathbb{1}_{[\tilde{a},\tilde{c}]}(Y).$$

with $l = \tau^{-1}_{[\underline{x},\overline{x}]}(a)$ and $u = \tau^{-1}_{[\underline{x},\overline{x}]}(c)$. □

We have seen that the conditional expectation of the transformed Chebyshev polynomials defined on $[\underline{x}, \overline{x}]$ can be reduced to the expectation of Chebyshev polynomials on $[-1, 1]$. The following proposition provides a recursive formula that allows us to compute these expectations analytically. It is based on the recursive definition of the Chebyshev polynomials (2.12) and the formula for the derivative presented in Proposition 3.

Proposition 10. *Let $Y \sim \mathcal{N}(\mu, \sigma^2)$ be a normally distributed random variable with density f and distribution function F, let $-1 \leq l \leq u \leq 1$. The truncated generalized moments $\mu_j = \mathbb{E}[T_j(Y)\mathbb{1}_{[l,u]}(Y)]$ and the expectations of the derivatives of the Chebyshev polynomials $\mu'_j = \mathbb{E}[T'_j(Y)\mathbb{1}_{[l,u]}]$ are recursively defined by*

$$\mu_{n+1} = 2\mu\mu_n - 2\sigma^2\big(T_n(u)f(u) - T_n(l)f(l) - 2n{\sum_{j=0}^{n-1}}'\mu_j\mathbb{1}_{(n+j) \bmod 2=1}\big) - \mu_{n-1}, \quad n \geq 1$$

and starting values $\mu_0 = F(u) - F(l)$, $\mu_1 = \mu\mu_0 - \sigma^2(f(u) - f(l))$.

Proof. Let $Y \sim \mathcal{N}(\mu, \sigma^2)$ be a normally distributed variable. We obtain for the expectation of T_0

$$\mu_0 = \mathbb{E}[\mathbb{1}_{[l,u]}(Y)] = \int_l^u f(y)\mathrm{d}y = F(u) - F(l).$$

For the remaining part of our proof we make use of the following property of the density f of a normally distributed variable

$$f'(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}(-2\frac{(x-\mu)}{2\sigma^2}) = f(x)(-2\frac{(x-\mu)}{2\sigma^2}) = (-\frac{1}{\sigma^2})xf(x) + \frac{\mu}{\sigma^2}f(x)$$

It follows that $xf(x) = \mu f(x) - \sigma^2 f'(x)$. Using this property we obtain

$$\begin{aligned}\mu_1 = \mathbb{E}[Y\mathbb{1}_{[l,u]}(Y)] &= \int_l^u yf(y)\mathrm{d}y \\ &= \mu\int_l^u f(y)\mathrm{d}y - \sigma^2\int_l^u f'(y)\mathrm{d}y \\ &= \mu\mu_0 - \sigma^2(f(u) - f(l)).\end{aligned}$$

Assume the moments μ_j are known for $j = 0, \ldots, n$. The Chebyshev polynomials T_n satisfy the following recursive formula $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$. For the generalized moments we obtain

$$\mu_{n+1} = \mathbb{E}[T_{n+1}(Y)\mathbb{1}_{[l,u]}(Y)] = 2\mathbb{E}[YT_n(Y)\mathbb{1}_{[l,u]}(Y)] - \mathbb{E}[T_{n-1}(Y)\mathbb{1}_{[l,u]}(Y)].$$

The second term is simply μ_{n-1} and for the first term we obtain

$$\begin{aligned}\mathbb{E}[YT_n(Y)\mathbb{1}_{[l,u]}(Y)] &= \int_l^u yT_n(y)f(y)\mathrm{d}y \\ &= \mu\int_l^u T_n(y)f(y)\mathrm{d}y - \sigma^2\int_l^u T_n(y)f'(y)\mathrm{d}y\end{aligned}$$

$$= \mu\mu_n - \sigma^2\Big(T_n(y)f(y)\Big|_l^u - \int_l^u T_n'(y)f(y)\mathrm{d}y\Big)$$
$$= \mu\mu_n - \sigma^2\big(T_n(u)f(u) - T_n(l)f(l) - \mu_n'\big).$$

Altogether we obtain

$$\begin{aligned}\mu_{n+1} &= 2\mathbb{E}[YT_n\mathbb{1}_{[l,u]}(Y)] - \mathbb{E}[T_{n-1}\mathbb{1}_{[l,u]}(Y)] \\ &= 2\big(\mu\mu_n - \sigma^2\big(T_n(u)f(u) - T_n(l)f(l) - \mu_n'\big)\big) - \mu_{n-1}.\end{aligned}$$

From Proposition 3 follows for the expectation of the derivative of the polynomials

$$\mu_{n+1}' = 2(n+1)\sum_{j=0}^{n}{}'\mu_j\mathbb{1}_{(n+j) \bmod 2=0}, \quad n \geq 0$$

where $\sum'$ indicates that the first term is multiplied with $1/2$. Combining the two results yields

$$\mu_{n+1} = 2\mu\mu_n - 2\sigma^2\big(T_n(u)f(u) - T_n(l)f(l) - 2n\sum_{j=0}^{n-1}{}'\mu_j\mathbb{1}_{(n+j) \bmod 2=1}\big) - \mu_{n-1}$$

for $-1 \leq l \leq u \leq 1$, which was our claim. □

In the relevant special case where $[c,d] = [\underline{x}, \overline{x}]$ and thus $[l,u] = [-1,1]$ we can simplify the result.

Remark 2. *Assume the conditions Proposition 10 hold for $l = -1$ and $u = 1$. Then we obtain*

$$\mu_{n+1} = 2\mu\mu_n - 2\sigma^2\big(f(1) - T_n(-1)f(-1) - 2n\sum_{j=0}^{n-1}{}'\mu_j\mathbb{1}_{(n+j) \bmod 2=1}\big) - \mu_{n-1} \tag{3.17}$$

and starting values $\mu_0 = F(1) - F(-1)$, $\mu_1 = \mu\mu_0 - \sigma^2(f(1) - f(-1))$.

3.3.2 Numerical integration for the generalized moments

In this section, we discuss several numerical approaches that can be used in order to calculate the generalized moments if the underlying is not normally distributed. We present the ideas for an univariate process, the extension to higher dimension is possible, see [91].

Probability density function

For the derivation of $\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_k]$, let the density function of the random

variable $X_{t_{u+1}}|X_{t_u} = x_k$ be given as $f^{u,k}(x)$. Then, the conditional expectation can be derived by solving an integral,

$$\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_k] = \int_{\underline{x}}^{\overline{x}} T_j(\tau^{-1}_{[\underline{x},\overline{x}]}(y))\, f^{u,k}(y)\mathrm{d}y$$

using $p_j(y) = T_j(\tau^{-1}_{\mathcal{X}}(y))1_{\mathcal{X}}(y)$. This approach is rather intuitive and easy to implement. Classical quadrature techniques such as Clenshaw-Curtis or Gauss-Legendre are in most situation sufficient to obtain accurate results.

Depending on the distribution $f^{u,k}$ the interval $[\underline{x}, \overline{x}]$ might be too large and the integrand is almost zero on a large part of the domain. For example assume $X_{t_{u+1}}|X_{t_u} = x_k$ is normally distributed with mean $x_k + \mu\Delta t$ and volatility $\sigma^2\Delta t$. For $k = 0$ we have $x_k = \underline{x}$ and for small values Δt only the interval $[\underline{x}, \underline{x} + \varepsilon]$ for some $\varepsilon > 0$ is important. We propose the following algorithm to tackle this problem:

Fix a $p \in (0, 1)$ depending on the aimed accuracy of the pricing algorithm. Then we compute for each $k = 0, \dots, N$ the p and the $1 - p$ quantile q_p, q_{1-p} and set the interpolation domain

$$[x^k_{min}, x^k_{max}] \quad \text{with} \quad x^k_{min} = \max\{\underline{x}, q_p\} \quad \text{and} \quad x^k_{max} = \min\{\overline{x}, q_{1-p}\}. \tag{3.18}$$

This domain can be used to integrate all p_j with $j = 0, \dots, N$.

Fourier Transformation

In many models we do not know the density in closed form but instead the characteristic function is available. Important examples are models with jumps such as the Merton jump-diffusion model, the large class of Lévy models and models based on affine processes. In these models we can apply the ideas of Fourier pricing in order to compute the generalized moments. The idea behind Fourier pricing is to transform an integral over the distribution into an integral over the characteristic function. The use of Fourier methods for the efficient pricing of options goes back to the seminal work of [30].

Assume the process $(X_t)_{t\geq 0}$ has stationary increments and the characteristic function φ of $X_{\Delta t}$ is explicitly available. The following proposition provides a version of the Fourier pricing formula (2.41) for the generalized moments.

Proposition 11. *Assume a stochastic process $(X_t)_{t\geq 0}$ with characteristic function ϕ_{x_0} of the random variable $X_{t_{u+1}}|X_{t_u}$. The generalized moments can then be computed as*

$$\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_0] = \frac{1}{\pi}\int_0^\infty \Re\left(\widehat{T}_j(-z)e^{iz(1-2\frac{\overline{x}}{\overline{x}-\underline{x}})}\,\varphi_{x_0}\Big(\frac{2z}{\overline{x}-\underline{x}}\Big)\right) dz. \tag{3.19}$$

Proof. We obtain for the generalized moments

$$\begin{aligned}\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u}=x_k] &= E[T_j(\tau^{-1}_{[\underline{x},\overline{x}]}(X_{t_{u+1}}))1_{[\underline{x},\overline{x}]}(X_{t_{u+1}})|x_0] \\ &= \int_{-\infty}^{\infty} T_j(\tau^{-1}_{[\underline{x},\overline{x}]}(x))1_{\{\underline{x}\le x\le\overline{x}\}} f(x|x_0)\mathrm{d}x \\ &= \frac{\overline{x}-\underline{x}}{2}\int_{-\infty}^{\infty} T_j(y)1_{\{-1\le y\le 1\}} \underbrace{f(\tau_{[\underline{x},\overline{x}]}(y)|x_0)}_{=f_{x_0}(\tau(y))}\mathrm{d}y.\end{aligned}$$

In the last step we used the substitution $y=\tau^{-1}_{\underline{x},\overline{x}}(x)$. The Chebyshev polynomials T_j on $\mathbb{R}$ should be seen as $T_j(x)\mathbb{1}_{[-1,1]}(x)$. Parseval's identity yields

$$\int_{-\infty}^{\infty} T_j(y)f_{x_0}(\tau(y))\mathrm{d}y = \frac{1}{2\pi}\int_{-\infty}^{\infty}\overline{\widehat{T_j}(z)}\widehat{f_{x_0}(\tau(\cdot))}(z)\mathrm{d}z.$$

The Fourier transform of the density is given by

$$\begin{aligned}\widehat{f_{x_0}(\tau(\cdot))}(z) &= \int_{-\infty}^{\infty} e^{izx} f_{x_0}(\tau(x))\mathrm{d}x \\ &\overset{y=\tau_{[\underline{x},\overline{x}]}(x)}{=} \frac{2}{\overline{x}-\underline{x}}\int_{-\infty}^{\infty} e^{iz\tau^{-1}_{[\underline{x},\overline{x}]}(y)} f_{x_0}(y)\mathrm{d}y \\ &= \frac{2}{\overline{x}-\underline{x}}\int_{-\infty}^{\infty} e^{iz(1-2\frac{\overline{x}}{\overline{x}-\underline{x}}+\frac{2}{\overline{x}-\underline{x}}y)} f_{x_0}(y)\mathrm{d}y \\ &= \frac{2}{\overline{x}-\underline{x}}e^{iz(1-2\frac{\overline{x})}{\overline{x}-\underline{x}}}\int_{-\infty}^{\infty} e^{iz\frac{2}{\overline{x}-\underline{x}}y} f_{x_0}(y)\mathrm{d}y \\ &= \frac{2}{\overline{x}-\underline{x}}e^{iz(1-2\frac{\overline{x}}{\overline{x}-\underline{x}})}\ \varphi_{x_0}\left(z\frac{2}{\overline{x}-\underline{x}}\right),\end{aligned}$$

where φ_{x_0} is the characteristic function of the respective stochastic process with starting value x_0. The Fourier transform of T_j is the integral

$$\widehat{T_j}(z) = \int_{-\infty}^{\infty} e^{izx}T_j(x)1_{[-1,1]}(x)\mathrm{d}x = \int_{-1}^{1} e^{izx}T_j(x)\mathrm{d}x.$$

Thus we obtain

$$\begin{aligned}E[T_j(\tau^{-1}_{[\underline{x},\overline{x}]}(X_{t_{k+1}}))1_{\{\underline{x}\le X_{t_{k+1}}\le\overline{x}\}}|x_0] &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\overline{\widehat{p_j}(z)}e^{iz(1-2\frac{\overline{x}}{\overline{x}-\underline{x}})}\ \varphi_{x_0}\left(\frac{2z}{\overline{x}-\underline{x}}\right)\mathrm{d}z \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\widehat{p_j}(-z)e^{iz(1-2\frac{\overline{x}}{\overline{x}-\underline{x}})}\ \varphi_{x_0}\left(\frac{2z}{\overline{x}-\underline{x}}\right)\mathrm{d}z\end{aligned}$$

In the last step we used that $\overline{\widehat{p_j}(z)}=\widehat{p_j}(-z)$, since p_j is a real valued function. Further-

more, similar to (2.42) we can write

$$\int_{-\infty}^{\infty} \widehat{p_j}(-z) e^{iz(1-2\frac{\overline{x}}{\overline{x}-\underline{x}})}\, \varphi_{x_0}\left(\frac{2z}{\overline{x}-\underline{x}}\right) \mathrm{d}z = 2\int_0^{\infty} \Re\left(\widehat{p_j}(-z) e^{iz(1-2\frac{\overline{x}}{\overline{x}-\underline{x}})}\, \varphi_{x_0}\Big(\frac{2z}{\overline{x}-\underline{x}}\Big)\right) \mathrm{d}z.$$

This proves our claim. □

The Fourier transform of the Chebyshev polynomials $\widehat{T_j}$ are presented in [35] and the authors also provide a Matlab implementation. For all Lévy processes with starting value x_0 we have for the characteristic function $\varphi_{x_0}(z) = e^{ix_0 z}\varphi(z)$.

Monte-Carlo simulation

Lastly, especially in cases for which neither a probability density function, nor a characteristic function of the underlying process is given, Monte-Carlo simulation is a suitable choice. For every nodal point x_k one simulates N_{MC} paths $X^i_{t_{u+1}}$ of $X_{t_{u+1}}$ with starting value $X_{t_u} = x_k$. These simulations can then be used to approximate

$$\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_k] \approx \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} p_j(X^i_{t_{u+1}})$$

for every $0 \leq j \leq N$. The polynomials p_j can be evaluated using Clenshaw's algorithm. For an overview of Monte-Carlo simulation from SDEs and variance reduction techniques we refer to [54] and [83].

3.4 The dynamic Chebyshev method in practice

In the section we move from the theoretical investigation of the method to the implementation in practice. This section should be the connection between the first three sections of the chapter and the empirical investigations in the two sections afterwards. We discuss how the method should be implemented, analyse the computational complexity of the pre-computation step and the backward induction and examine the expected convergence behaviour.

3.4.1 Implementation of the method

We focus on the implementation of the method for an American (or Bermudan) put option and an up-and-out barrier call option. The presented ideas can be easily extended to other problems that can be written as (3.1) and (3.2). We recall the DPP of the American put option

$$V_T^{Am}(x) = g(x) = (K - e^x)^+$$

$$V_{t_u}^{Am}(x) = \max\{K - e^x, e^{-r\Delta t}\mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x]\}$$

and the DPP of the up-and-out barrier call option

$$V_T^{Bar}(x) = g(x) = (K - e^x)^+ \mathbb{1}_{(-\infty,b]}(x)$$
$$V_{t_u}^{Bar}(x) = e^{-r\Delta t}\mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x]\mathbb{1}_{(-\infty,b]}(x)$$

for a barrier b. For a large class of models, we know that the value function $x \mapsto V_{t_u}^{Am}(x)$ is continuously differentiable and piecewise analytic on any interval $[\underline{x}, \overline{x}]$ for $\underline{x} < k := \log(K) < \overline{x}$. However, it is not analytic on the whole domain $[\underline{x}, \overline{x}]$. In contrast, the value function $x \mapsto V_{t_u}^{Bar}$ is analytic on any interval $[\underline{x}, \overline{x}]$ for $\underline{x} < k < \overline{x} \leq b$ in many stock price models.

Interpolation domain and truncation error

A suitable choice for the interpolation domain $\mathcal{X} = [\underline{x}, \overline{x}]$ is crucial for the implementation of the method since the restriction to a compact domain introduces a truncation error. The theoretical error analysis shows that the overall error cannot be below this truncation error. One the other hand, the larger the interpolation domain is, the more interpolation nodes are required for the same accuracy.

For the up-and-out barrier call option the natural upper bound is exactly the barrier b because $V_{t_u}(x)$ is zero for $x > b$. Moreover, we can exploit that $V_{t_u}(x) \to 0$ for $x \to -\infty$ to find a suitable lower bound $\underline{x}$. Depending on the required accuracy we fix an error threshold $\varepsilon > 0$ and find a lower bound $\underline{x}$ such that $\mathbb{Q}(X_T > log(K)|X_0 = \underline{x}) < \varepsilon$, i.e. we ensure that the probability of ending in the money is small enough.

For the American put the choice of an appropriate interpolation domain is more complex. We can exploit that $V_{t_u}^{Am}(x)$ goes to zero for $x \to \infty$ and follow the same procedure as for the barrier option. Unfortunately, for x small enough $V_{t_u}^{Am}(x) = K - e^x$ and converges towards the strike K. Hence, the truncation at the lower end would yield to a significantly larger truncation error. In order to tackle this we can exploit the asymptotic behaviour of the payoff and use that the American option is always exercised if x is below some $\underline{x}$ that is small enough. The value function can be approximated by

$$V_{t_{u+1}}(x) \approx (K - e^x)\mathbb{1}_{\{x<\underline{x}\}} + \widehat{V}_{t_{u+1}}(x)\mathbb{1}_{\{x\in\mathcal{X}\}}$$

and thus

$$\mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x_k] \approx \mathbb{E}[(K - e^{X_{t_{u+1}}})\mathbb{1}_{\{X_{t_{u+1}}<\underline{x}\}}|X_{t_u} = x_k] + \sum_{j=0}^{N} c_j(t_{u+1})\Gamma_{j,k}(t_u)$$

for $\underline{x}$ small and $\overline{x}$ large enough. One can precompute $\mathbb{E}[(K-e^{X_{t_{u+1}}})\mathbb{1}_{\{X_{t_{u+1}}<\underline{x}\}}|X_{t_u}=x_k]$. We emphasize that similar modifications to reduce the truncation error can be found for other payoff profiles, e.g. for digitals, butterfly options or any other combination of different put options.

From a theoretical point of view, the suggested procedure guarantees that the truncation error is negligible. For the American option, computing the additional conditional expectations is often an unnecessary computational effort. Instead, one can simple choose a large enough interpolation domain such that for all values of interest x_0 the probability $\mathbb{P}(\min_{t\in[0,T]} X_t \leq \underline{x}|X_0=x_0)$ is small enough. In this case, the truncation error is again negligible compared to the overall error level. The choice can be made explicitly if the underlying is conditionally normally distributed.

Remark 3. *Assume a stochastic process $(X_t)_{t\geq 0}$ where $X_T|X_0=x_0$ is normally distributed with mean $\mu(x_0,T)$ and volatility $\sigma(x_0,T)$. Assume we are interested in starting values $x_0\in[x_{lo},x_{up}]$, then we set the interpolation domain to*

$$[\underline{x},\overline{x}]=[\mu(x_{lo},T)-k\sigma(x_0,T),\mu(x_{lo},T)+k\sigma(x_0,T)]$$

for some integer $k>0$. A standard choice is $k=4$ or $k=5$.

Setting $k=4$ means that the probability of $X_T\in[\underline{x},\overline{x}]$ given an $X_0=x_0\in[x_{lo},x_{up}]$ is above 99.994% and the domain is large enough for many applications. In order to ensure a higher accuracy we need to increase k to 5 or 6. Similar results can be found for other distributions using a lower and an upper quantile. Note that we are often only interested in one price and set $x_0=x_{lo}=x_{up}$.

In many applications, we know that the value function converges towards zero on one side and is increasing on the other end of the domain. In this case, it might be beneficial to choose different values of k, i.e.

$$[\underline{x},\overline{x}]=[\mu(x_{lo},T)-k_{low}\sigma(x_0,T),\mu(x_{lo},T)+k_{up}\sigma(x_0,T)].$$

For example, for a put option we typically want to use a bigger $k_{low}>k_{up}$ and the other way for a call option.

Initial time step: Smoothing the payoff

For both option types holds that the payoff function $g(x)$ has a kink at $k=\log(K)$ and an accurate interpolation would require a high polynomial degree N. The presented dynamic Chebyshev approach with splitting is a possibility to tackle this problem. For the American put it splits the interpolation domain at the early-exercise boundary which

is exactly the strike at maturity.

However, there is a simpler approach that we call "smoothing the payoff" and that ensures a better convergence.

Remark 4. *Instead of interpolating the non-smooth payoff function g in the dynamic Chebyshev algorithm (Algorithm 1), we can directly compute the conditional expectations*

$$\mathbb{E}[V_T(X_T)|X_{t_{n_T-1}} = x_k] = \mathbb{E}[g(X_T)|X_{t_{n_T-1}} = x_k]$$

for $k = 0, \dots, N$ at time point t_{n_T-1}. Essentially, it means that we start the pricing algorithm at t_{n_T-1} instead of at time point $t_{n_T} = T$.

For this task, we can use the same techniques as for the generalized moments, such as quadrature or Monte Carlo simulation. In most models, we have that $x \mapsto \mathbb{E}[g(X_T)|X_{t_{n-1}} = x]$ is analytic and hence, this "smoothing" will improve the overall convergence and accuracy of the method. In the next section, we will investigate the smoothing effect on the error decay numerically.

For the American put option we know that the conditional expectation of the payoff function is the value of an European put option. In the Black-Scholes model, we have an analytic formula for the value of a put option that can be used in the first time step. The following proposition provides a similar result if we add a barrier to the payoff of a European call option.

Proposition 12. *Assume a Black-Scholes framework with stock price process $(S_t)_{t\geq 0}$, interest rate r, volatility $\sigma > 0$ and pricing measure $\mathbb{Q}$. The price of a call option with strike K, maturity T and barrier B at time t,*

$$B(S_0, t) := e^{-r(T-t)}\mathbb{E}[(S_T - K)^+ \mathbb{1}_{S_T \leq B}|S_t = S_0]$$

is given by

$$B(S_0, t) = S_0\big(\Phi(b_1) - \Phi(d_1)\big) - e^{-r(T-t)}K\big(\Phi(b_2) - \Phi(d_2)\big)$$

with

$$b_1 = \frac{\log(B/S_0) - (\mu + \frac{\sigma^2}{2})(T-t)}{\sqrt{T-t}\sigma}, \qquad d_1 = \frac{\log(K/S_0) - (\mu + \frac{\sigma^2}{2})(T-t)}{\sqrt{T-t}\sigma}$$

and $b_2 = b_1 + \sqrt{T-t}\sigma$, $d_2 = d_1 + \sqrt{T-t}\sigma$.

Proof. The proof of this proposition works similar to the derivation of the Black-Scholes formula via integration, see for example [41]. We assume $t = 0$. The stock price can be

written as $S_T = S_0 e^{X_T}$ with $X_T = (r - \frac{\sigma^2}{2})T + \sigma W_T$ for a Brownian motion W_t. Hence, X_T is normally distributed with mean $\mu_T = (r - \frac{\sigma^2}{2})T$ and variance $T\sigma^2$. The price of a barrier call option with strike K, barrier B and maturity T is given by

$$e^{rT} B(S_0, 0) = \int_{\mathbb{R}} (S_0 e^x - K)^+ \mathbb{1}_{\{S_0 e^x \leq B\}} f(x) \mathrm{d}x = \int_{\log(K/S_0)}^{\log(B/S_0)} \big(S_0 e^x f(x) - K f(x)\big) \mathrm{d}x$$

where f is the density of X_T. For the first part of the integral holds

$$\int_{\log(K/S_0)}^{\log(B/S_0)} S_0 e^x f(x) \mathrm{d}x = \frac{S_0}{\sqrt{2\pi T}\sigma} \int_{\log(K/S_0)}^{\log(B/S_0)} e^x e^{-\frac{(x-\mu_T)^2}{2T\sigma^2}} \mathrm{d}x.$$

We can rewrite the exponent using $\mu_T = (r - \frac{\sigma^2}{2})T$ and we obtain

$$\begin{aligned}
x - \frac{(x-\mu_T)^2}{2T\sigma^2} &= -\frac{x^2 - 2x\mu_T + \mu_T^2 - 2T\sigma^2 x}{2T\sigma^2} \\
&= -\frac{x^2 - 2x((r - \frac{\sigma^2}{2})T + T\sigma^2) + ((r - \frac{\sigma^2}{2})T)^2}{2T\sigma^2} \\
&= -\frac{x^2 - 2x(r + \frac{\sigma^2}{2})T + ((r + \frac{\sigma^2}{2})T)^2 - 2r\sigma^2 T^2}{2T\sigma^2} \\
&= -\frac{(x - (r + \frac{\sigma^2}{2})T)^2}{2T\sigma^2} + rT.
\end{aligned}$$

This yields

$$\begin{aligned}
&\frac{1}{\sqrt{2\pi T}\sigma} \int_{\log(K/S_0)}^{\log(B/S_0)} e^x e^{-\frac{(x-\mu_T)^2}{2T\sigma^2}} \mathrm{d}x \\
&= \frac{e^{rT}}{\sqrt{2\pi T}\sigma} \int_{\log(K/S_0)}^{\log(B/S_0)} e^{-\frac{(x-(r+\frac{\sigma^2}{2})T)^2}{2T\sigma^2}} \mathrm{d}x \\
&= e^{rT} \Big(\mathbb{Q}\Big((r + \frac{\sigma^2}{2})T + \sqrt{T}\sigma Z \leq \log\big(\frac{B}{S_0}\big) \Big) - \mathbb{Q}\Big((r + \frac{\sigma^2}{2})T + \sqrt{T}\sigma Z \leq \log\big(\frac{K}{S_0}\big) \Big) \Big) \\
&= e^{rT} \Big(\mathbb{Q}\Big(Z \leq \frac{\log(B/S_0) - (r + \frac{\sigma^2}{2})T}{\sqrt{T}\sigma} \Big) - \mathbb{Q}\Big(Z \leq \frac{\log(K/S_0) - (r + \frac{\sigma^2}{2})T}{\sqrt{T}\sigma} \Big) \Big) \\
&= e^{rT} \big(\Phi(b_1) - \Phi(d_1) \big)
\end{aligned}$$

with

$$b_1 := \frac{\log(B/S_0) - (\mu + \frac{\sigma^2}{2})(T-t)}{\sqrt{T-t}\sigma} \quad d_1 := \frac{\log(K/S_0) - (\mu + \frac{\sigma^2}{2})(T-t)}{\sqrt{T-t}\sigma}.$$

For the second part of the integral holds

$$\int_{\log(K/S_0)}^{\log(B/S_0)} f(x) \mathrm{d}x = \mathbb{Q}(X_T \leq \log(B/S_0)) - \mathbb{Q}(X_T \leq \log(K/S_0))$$

$$
\begin{aligned}
&= \mathbb{Q}(\mu_T + \sqrt{T}\sigma Z \leq \log(B/S_0)) - \mathbb{Q}(\mu_T + \sqrt{T}\sigma Z \leq \log(K/S_0)) \\
&= \mathbb{Q}(Z \leq \frac{\log(B/S_0) - \mu_T}{\sqrt{T}\sigma}) - \mathbb{Q}(Z \leq \frac{\log(K/S_0) - \mu_T}{\sqrt{T}\sigma}) \\
&= \Phi(b_2) - \Phi(d_2)
\end{aligned}
$$

for $Z \sim \mathcal{N}(0,1)$ and

$$
\begin{aligned}
b_2 &:= \frac{\log(B/S_0) - \mu_T}{\sqrt{T}\sigma} = \frac{\log(B/S_0) - (r - \frac{\sigma^2}{2})T}{\sqrt{T}\sigma} = b_1 + \sqrt{T}\sigma \\
d_2 &:= \frac{\log(K/S_0) - \mu_T}{\sqrt{T}\sigma} = \frac{\log(K/S_0) - (r - \frac{\sigma^2}{2})T}{\sqrt{T}\sigma} + \sqrt{T}\sigma.
\end{aligned}
$$

Overall we have

$$
\begin{aligned}
B(S_0,0) &= e^{-rT}\mathbb{E}[(S_T - K)^+ \mathbb{1}_{S_T \leq B}] \\
&= e^{-rT} S_0 \int_{\log(K/S_0)}^{\log(B/S_0)} e^x f(x)\mathrm{d}x - e^{-rT} K \int_{\log(K/S_0)}^{\log(B/S_0)} f(x)\mathrm{d}x \\
&= S_0\big(\Phi(b_1) - \Phi(d_1)\big) - e^{-rT} K\big(\Phi(b_2) - \Phi(d_2)\big).
\end{aligned}
$$

Replacing T with $T - t$ proves our claim. □

Convergence of the option's sensitivities

We know that the Chebyshev interpolation is also able to approximate the derivatives of the interpolated function, see Theorem 6 and Theorem 7. This can be exploited in the dynamic Chebyshev algorithm to compute the option's sensitivities with respect to the (log-) stock price. The option's Delta and Gamma can be computed by taking the first or second derivative of

$$
S \mapsto \widehat{V}_0(\log(S)) = \sum_{j=0}^{N} c_j(t_0) p_j(\log(S)).
$$

Using Proposition 3 we obtain for the first and the second derivative of $x \mapsto \widehat{V}_0(x)$

$$
\begin{aligned}
\widehat{V}_0'(x) &= \sum_{j=0}^{N-1} \tilde{c}_j(t_0) p_j(x) \quad \text{with} \quad \tilde{c}_j(t_0) = 2 \sum_{k=j+1}^{N} k c_k(t_0) \mathbb{1}_{(k+j) \bmod 2 = 1} \\
\widehat{V}_0''(x) &= \sum_{j=0}^{N-2} \tilde{\tilde{c}}_j(t_0) p_j(x) \quad \text{with} \quad \tilde{\tilde{c}}_j(t_0) = 2 \sum_{k=j+1}^{N-1} k \tilde{c}_j(t_0) \mathbb{1}_{(k+j) \bmod 2 = 1}.
\end{aligned}
$$

This yields for Delta and Gamma

$$\frac{\partial Price(S)}{\partial S} \approx \widehat{V}_0'(\log(S))\frac{1}{S} \quad \text{and} \quad \frac{\partial^2 Price(S)}{\partial S^2} \approx \frac{1}{S^2}\Big(\widehat{V}_0''(\log(S)) - \widehat{V}_0'(\log(S))\Big).$$

Thus Delta and Gamma are expressed as the sum of derivatives of Chebyshev polynomials. In particular, their derivation comes without any additional computational costs in the offline phase or in the time stepping.

3.4.2 Computational complexity

Next, we investigate the complexity and thus the computational cost of the dynamic Chebyshev algorithm. In order to do so, the offline/online structure of the method has to be taken into account. We assume an equidistant time stepping and that the stationarity assumption (3.4) holds.

In the offline or pre-computation step, we thus need to compute the $(N+1)^2$ generalized moments $\Gamma_{j,k} = \mathbb{E}[p_j(X_{\Delta t})|X_0 = x_k]$. If there is no analytic formula, a numerical integration requires the evaluation of the integrand at M different points. More precisely, when using numerical quadrature techniques to compute the moments, the evaluation of the integrand at M_{quad} quadrature points is required and similarly for the Monte Carlo approach on M_{MC} samples. In total, the complexity of the offline phase scales with $N^2 M_{quad}$ or $N^2 M_{MC}$. The complexity can be reduced when the straightforward approach for the moment calculation is replaced with a more sophisticated approach. Moreover, parallelization can help to reduce the runtime significantly. It is important to acknowledge that the three quantities N, M_{quad} and M_{MC} are on a different scale. The number of Monte Carlo simulations is typically much higher than the number of quadrature points or Chebyshev nodes. For example 50,000 might be a good choice for M_{MC} whereas $M_{quad} = 500$ is typically high enough. Once a required accuracy is fixed, both, the Chebyshev degree N and the number of integration points M have to be chosen accordingly. Later on, we will investigate the optimal relation between N and M_{MC} in more detail.

After the offline phase all model-dependent quantities are readily available. Moreover, the effort of the offline phase is independent of the number of time steps n_T or the number of different payoffs that have to be priced. For example, the pricing of an option surface, i.e. options on the same underlying with different strikes and maturities, can be done using the same generalized moments.

In the online phase, we need to compute the vector of nodal values $\widehat{V}_{t_u}(x_k)$ for all $k = 0, \ldots, N$ and then the coefficient vector $c_j^{t_u}$ for $j = 0, \ldots, N$ in every time step.

Both require the multiplication of a vector with $N+1$ entries with an $(N+1)\times(N+1)$ matrix. Hence, the total effort scales with $n_T N^2$ where n_T is the number of time steps. The complexity of the online phase is therefore independent of the numerical method applied in the offline calculations. Since N is typically relatively small the method becomes very efficient.

3.4.3 Expected convergence behaviour

Before we perform a numerical convergence analysis, we recall the theoretical error analysis and point out what type of error decay we can expect in the experiments.

First, we consider an analytic value function in the dynamic Chebyshev method and assume no truncation error. In this case we know from Corollary 1 that the following error bound holds

$$\varepsilon_{t_u} \leq c_1 \varrho^{-N} \log(N)^{dn_T} + c_2 \log(N)^{dn_T} \overline{\delta}$$

or with $\delta^\star$ instead of $\overline{\delta}$ if (GM*) holds. This yields for the log-error

$$\begin{aligned} \log(\varepsilon_{t_u}) &\leq \log\left(c_1 \varrho^{-N} \log(N)^{dn_T} + c_2 \log(N)^{dn_T} \overline{\delta}\right) \\ &= \log\left(c_1 \varrho^{-N} \log(N)^{dn_T}\right) + \log\left(1 + \frac{c_2}{c_1} \varrho^N \overline{\delta}\right) \\ &= \log(c_1) - \log(\varrho) N + dn_T \log(\log(N)) + \log\left(1 + \frac{c_2}{c_1} \varrho^N \overline{\delta}\right). \end{aligned} \tag{3.20}$$

If we assume that $\varrho^N \overline{\delta} \leq 1$ the log-error as a function of N is bounded by a function of the form $N \mapsto c - m_1 N + m_2 \log(\log(N))$ for constants $c, m_1, m_2 > 0$ and since the linear term dominates the log log-term, we expect to observe an exponential error decay in N. For $\varrho^N \overline{\delta} > 1$ we obtain

$$\begin{aligned} \log\left(1 + \frac{c_2}{c_1} \varrho^N \overline{\delta}\right) &= \log\left(\frac{c_2}{c_1} \varrho^N \overline{\delta}\right) + \log\left(\frac{c_1}{c_2} \varrho^{-N} \overline{\delta}^{-1} + 1\right) \\ &\leq \log\left(\frac{c_2}{c_1}\right) + \log(\varrho) N + \log(\overline{\delta}) + \log\left(\frac{c_1}{c_2} + 1\right). \end{aligned}$$

When we plug this term into (3.20), the terms $-\log(\varrho)N$ and $\log(\varrho)N$ cancel each other out. Combining the two cases yields two observations for the convergence behaviour. First, for a fixed accuracy $\overline{\delta}$ or $\delta^\star$ in the offline phase the method will converge in the online phase and the maximal reachable accuracy is limited by $\overline{\delta}$ resp. $\delta^\star$. Second, in terms of the total computational effort one should choose N subject to the accuracy of the generalized moments. For example in the Monte Carlo case the optimal N is a function of the number of simulations or of the number of quadrature points in the Fourier case. In the Monte Carlo case the error $\delta^\star$ decays typically with $cM^{-0.5}$ and

from $\varrho^N \delta^\star \leq 1$ follows $N \leq \tilde{c} \log(M)$ for some constant $\tilde{c} > 0$. If we fix N like this the complexity of the offline phase $N^2 M$ becomes $\log^2(M)M$ and the complexity of the online phase N^2 becomes $\log^2(M)$. In the Fourier case the error $\overline{\delta}$ depends on the regularity of the integrand which is model dependent. Typically the error will decrease much faster than the Monte Carlo error. Similar results can be obtained for a piecewise analytic value function and the dynamic Chebyshev method with splitting.

An example for a path-dependent option where we can expect exponential convergence is an up-and-out call option.

Remark 5. *When conditions (i) and (ii) of Proposition 6 hold and the smoothing of the payoff as stated in Remark 4 is applied then the value function of an up-and-out barrier option*

$$V_{t_u} : [\underline{x}, b] \ni x \mapsto \mathbb{E}[V_{t_u}(X_{t_{u+1}})|X_{t_u} = x]\mathbb{1}_{(-\infty,b]}(x)$$

is a function with an analytic extension to some Bernstein ellipse $\mathcal{B}([\underline{x}, b], \varrho)$, $\varrho \in (1, \infty)$ for all $u = 0, \ldots, n-1$. In this case the convergence result for the dynamic Chebyshev method of Corollary 3.4 holds.

From [49] we know that conditions (i) and (ii) of Proposition 6 are for example fulfilled in the Black-Scholes model.

If the value function is only continuously differentiable and not analytic, we can no longer apply the results from Section 3.2.1. Nevertheless, we can apply the dynamic Chebyshev method and we will provide a (rough) estimate of the expected convergence behaviour. We know that the convergence of the Chebyshev interpolation is of polynomial order for continuously differentiable functions, see Theorem 6. If we simply replace the term ϱ^{-N} by a term N^{-p} for a $p \in \mathbb{N}$, we can perform the same calculations as in the analytic case. We obtain for the log-error

$$\log(\varepsilon_{t_u}) \leq \log(c_1) - p \log(N) + d n_T \log(\log(N)) + \log\left(1 + \frac{c_1}{c_2} N^{-p} \overline{\delta}\right).$$

Assuming $N^p \overline{\delta} \leq 1$ suggests that the log-error as a function of N is bounded by a function $N \mapsto c - m_1 \log(N) + m_2 \log(\log(N))$ for constants $c, m_1, m_2 > 0$. In this case the log-error is approximately linear in $\log(N)$. If we use Monte Carlo simulation in the offline step with decay $cM^{-0.5}$ the condition $N^p \delta^\star \leq 1$ implies for $p = 1$ that $N \leq \tilde{c}\sqrt{M}$ and more general $N \leq \tilde{c} M^{0.5/p}$. Similarly to the analytic case, if we choose $N = \tilde{c} M^{0.5/p}$ the complexity of the offline phase $N^2 M$ becomes $M^{1+1/p}$ and the complexity of the online phase $n_T N^2$ becomes $n_T M^{1/p}$.

Examples where the value function is only continuously differentiable are early-

exercise options. Due to the maxima function in the evaluation of the value function in every time step, the function is not analytic around the optimal exercise point. However, from "smooth-fit" property (3.5) that the value function of an American put option in the Black-Scholes model is still continuously differentiable at the exercise point. [12] shows that this smoothness property also holds in a jump-diffusion model.

3.5 Empirical error analysis

In this section we investigate the convergence behaviour of the dynamic Chebyshev method. We price a barrier call option and an American put option along with the options' Delta and Gamma in the Black-Scholes model and the Merton jump diffusion model. Both pricing problems are described in more detail in Section 3.4. As a benchmark approach we choose the COS method of [44] and we use the *Matlab* implementations of method which were provided for the benchmarking project of [135]. The provided implementations are slightly modified to fit for our examples.

For the experiments, we use the following parameter sets in the Black-Scholes model

$$K = 100, \quad r = 0.03, \quad \sigma = 0.25, \quad T = 1,$$

and for the Merton jump diffusion model

$$K = 100, \quad r = 0.03, \quad \alpha = -0.5, \quad \beta = 0.4, \quad \sigma = 0.25, \quad \lambda = 0.4$$

and we use 32 time steps. The jump parameters α, β and λ are taken from [135]. Furthermore, we test different (numerical) approaches for calculating the generalized moments: Integrating over the density, the Fourier approach using the characteristic function and the closed-form solution if the underlying process has Gaussian increments.

3.5.1 Convergence for analytic value functions

We price a discretely monitored barrier option with call payoff $g_T(x) = (e^x - K)^+ \mathbb{1}_{(\infty,b]}(x)$ and barrier $b = \log(125)$ in the Black-Scholes model. For the following experiments, we used the density approach to calculate the generalized moments implemented with the *Matlab* quadrature routine *quadgk* with an absolute as well as relative error tolerance of 10^{-13} and we set $\underline{x} = \log(10)$. Prices and sensitivities are calculated on a grid of starting values equally distributed between 80 and 120 and compared to the benchmark method. From Section 3.4.3 follows that the value function is analytic on $\mathcal{X} = [\underline{x}, b]$ and we can expect an exponential error decay.

The left plot in Figure 3.1 shows the log-error (in absolute terms) for an increasing

number of nodes $N = 10, \ldots, 100$. The log-error for the prices as well as for Delta and Gamma decays linearly in N as we expected and reaches an accuracy below 10^{-12}. With only 50 nodal points the method is already able to achieve an accuracy below 10^{-6}.

The right plot in Figure 3.1 shows the same experiment without the smoothing in the initial time step. The method still converges but the decay of the log-error is no longer linear.

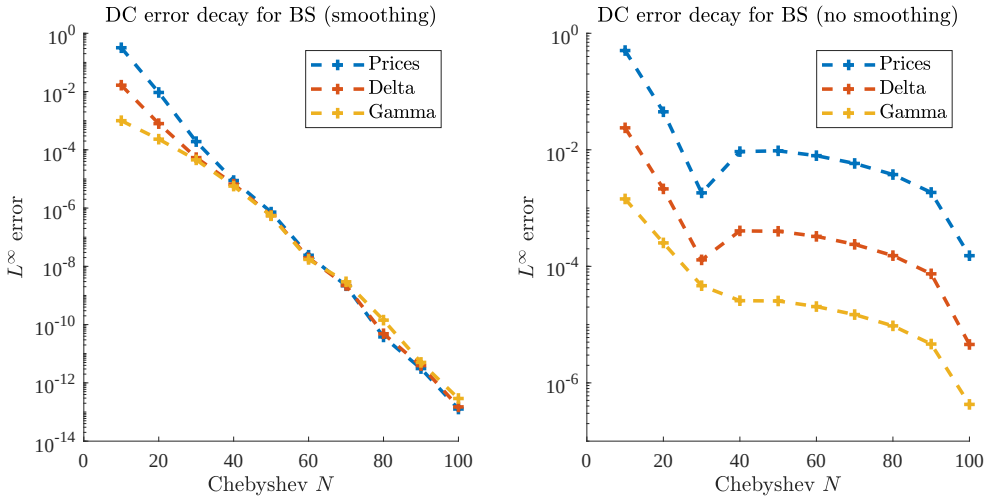

Figure 3.1: Error decay of the dynamic Chebyshev method for an up-an-out barrier call option in the BS model using smoothing in the first time step(left) and without smoothing (right). The conditional expectation of the Chebyshev polynomials are calculated with the density function.

3.5.2 Convergence for differentiable value functions

Next, we price a Bermudan put option in the Black-Scholes and in the Merton model. Here, the value function is only continuously differentiable but not analytic. The expected decay of the log-error is therefore slower than a linear decay and behaves approximately like $-p\log(N)$, see Section 3.4.3. Therefore, we should need more nodal points as in the analytic case to obtain the same accuracy.

For both models the generalized moments are computed by the Fourier approach as stated in Proposition 11. We truncate the integral at $|\xi| \leq 250$ and use Clenshaw-Curtis with 500 nodes for the numerical integration. For the Fourier transform of the Chebyshev polynomials the implementation of [35] is used. We run the dynamic Chebyshev method for an increasing number of Chebyshev nodes $N = 32, 64, \ldots, 512$. Doubling of the polynomial degree ensures that the Chebyshev points are nested. We fix an interpolation domain $[\log(S_{min}), \log(S_{max})]$ with $S_{max} = 350$ and $S_{min} = 15$ in the Black-Scholes model and $S_{min} = 5$ in the Merton jump-diffusion model. Then, option prices and their

sensitivities delta and gamma are calculated on a grid of different values of S_0 equally distributed between 70% and 130% of the strike K. The resulting prices and Greeks are compared to our benchmark method and the maximum error over the grid is calculated. Moreover, we compare the realized error decay to a theoretical error decay of N^{-2}, i.e. an error decay of order two in space. We recall that no time discretization error is made for Bermudan options if the generalized moments can be evaluated accurately.

Figure 3.2 shows the error decay (in absolute terms) for the Black-Scholes model (left hand side) and the Merton model (right hand side). We observe that the method converges and an error below 10^{-3} is reached for $N = 256$ Chebyshev nodes. The speed of the convergence is similar for both stock price models and slower than in the barrier option example. The plots demonstrate an approximately polynomial error decay in N. In both models, the error decay is better than order two without applying any splitting at the exercise boundary.

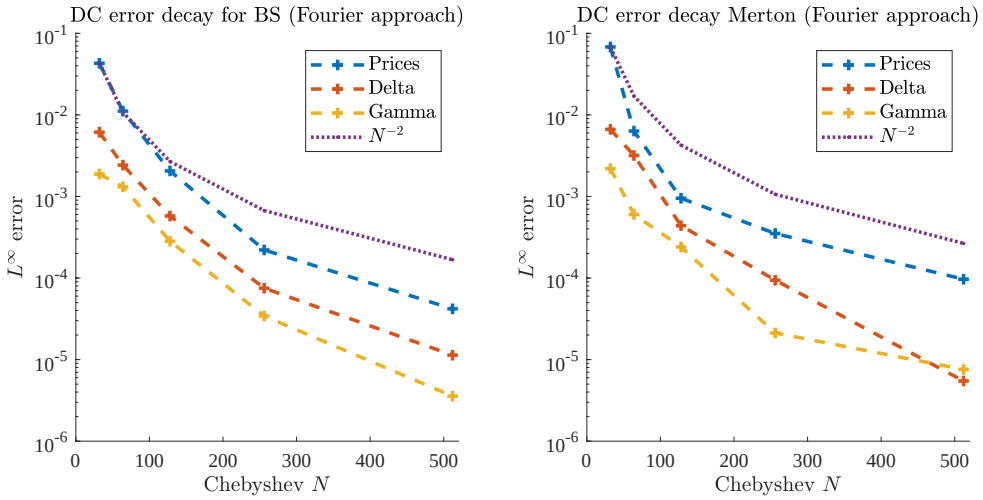

Figure 3.2: Error decay of the dynamic Chebyshev for a Bermudan option in the BS model (left) and the Merton model (right). The conditional expectation of the Chebyshev polynomials are calculated with the Fourier transformation.

3.5.3 Convergence for a bivariate barrier option

In this section, we provide evidence that the method also works for multivariate options by looking at the convergence of the dynamic Chebyshev method for a barrier option with two barriers. We consider two assets S_t^1, S_t^2 and an option with payoff

$$g(x_1, x_2) = (e^{x_1} - K)^+ 1_{(-\infty, b_1]}(x_1) \mathbb{1}_{(-\infty, b_2]}(x_2) \qquad \text{with} \quad x_1 = \log(S_1),\ x_2 = \log(S_2)$$

strike K and barrier b. This type of option is also referred as outside/rainbow barrier option as a second (outside) underlying is included to generate an additional discount

compared to a standard (barrier) option. An economic example could be that a company would like to hedge against increasing prices of a commodity only if the economy stays at or below its current level. In the scenario of an economic boom higher prices cover the increase in costs and no hedge is required. Different examples of options with multiple barriers and their economic interpretation can for example be found in [42].

Here, we assume that both assets follow a geometric Brownian motion and hence we are in a bivariate Black-Scholes type model. We fix the following model parameters

$$K = 100, \quad r = 0.03, \quad \sigma_1 = 0.25, \quad \sigma_2 = 0.2, \quad \rho = 0.4, \quad T = 1,$$

and choose as barrier $b_1 = \log(125)$ and $b_2 = \log(120)$. For the calculation of the generalized moments we use the density approach implemented using Clenshaw-Curtis quadrature with 500 points in each dimension. We fix a bivariate interpolation domain $\mathcal{X} = [\underline{x}_1, b_1] \times [\underline{x}_2, b_2]$ with $\underline{x}_1 = \log(20)$ and $\underline{x}_2 = \log(20)$. We run the dynamic Chebyshev method for an increasing number of points $N = N_1 = N_2$ ranging from 10 to 40 and calculate prices on a two-dimensional grid of starting values equally distributed in $[90, 110] \times [90, 110]$. For the comparison of prices, we run the method with $N = 50$.

Figure 3.3 shows the resulting error decay. We still observe that the log-error decays almost linear in the total number of N^2 grid points as theoretically predicted, see Section 3.4.3. In a general D-dimensional framework we can expect to need N^D points for the same error behaviour in N. This is often called the course of dimensionality. This numerical experiment should only been seen as a toy example that shows that the theoretical convergence results for $d > 1$ hold numerically. In Chapter 5, we discuss how the dynamic Chebyshev method can be efficiently used to price multivariate options in detail.

3.5.4 Convergence for piecewise analytic functions

Next, we implemented the dynamic Chebyshev algorithm with splitting as provided in Algorithm 3. We repeat the experiments for the Bermudan put from Section 3.5.2. Using the splitting algorithm we expect exponential convergence and a lower number of nodal points should be sufficient to reach the same level of accuracy. We run the dynamic Chebyshev method for an increasing number of Chebyshev nodes $N = 5, 10, \ldots, 75$. We fix an interpolation domain $[\log(S_{min}), \log(S_{max})]$ with $S_{min} = 10$ and $S_{max} = 500$ in the Black-Scholes model and $S_{min} = 2$ and $S_{max} = 850$ in the Merton jump-diffusion model. Then, option prices are calculated on a grid of different values of S_0 equally distributed between 70% and 130% of the strike K. Moreover, the option's sensitivities delta and gamma are computed on the same grid. The conditional expectations are

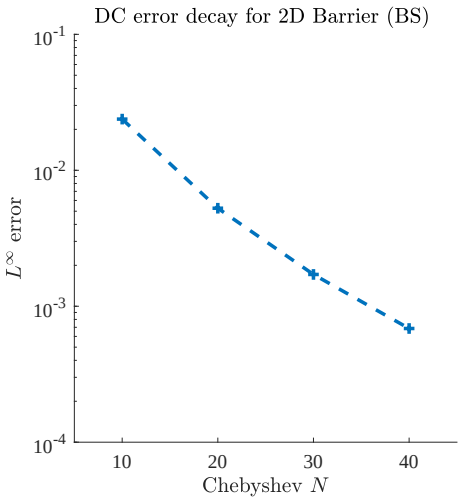

Figure 3.3: Error decay of the dynamic Chebyshev approach for a bivariate barrier option in a two-dimensional Black-Scholes model. The conditional expectation of the Chebyshev polynomials are calculated using the density function.

again calculated using the Fourier approach and for the Black-Scholes model we use also the closed form solution.

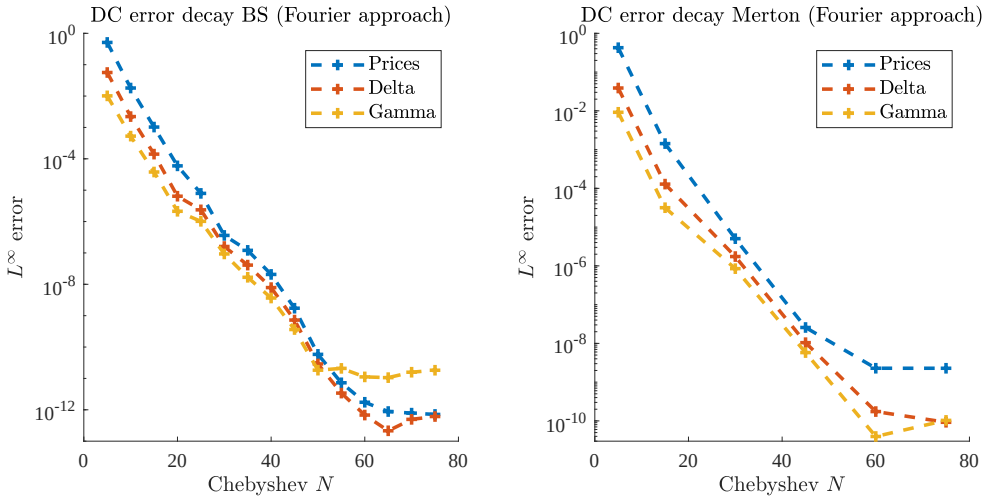

Figure 3.4: Error decay of the dynamic Chebyshev method with splitting for a Bermudan option in the BS model (left) and the Merton model (right). The conditional expectation of the Chebyshev polynomials are calculated with the Fourier transformation.

Figure 3.4 shows the error decay for the Black-Scholes model (left plot) and for the Merton jump-diffusion model (right plot). As expected, we observe that the log-error decays linearly in the Chebyshev degree N. With only 26 nodal points we reach an absolute error of 10^{-4} and with 50 points we obtain an error in the region of 10^{-8}.
Figure 3.5 shows the error decay for the Black-Scholes model combined with the analytic

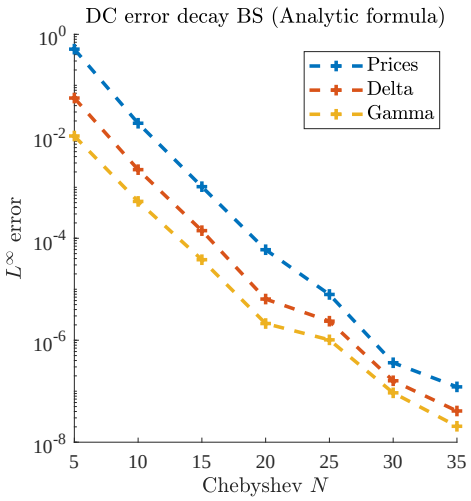

Figure 3.5: Error decay of the dynamic Chebyshev with splitting for a Bermudan option in the Black-Scholes model. The conditional expectation of the Chebyshev polynomials are calculated with a recursive formula for the truncated moments.

solution for the generalized moments. The behaviour is the same as for the Fourier approach for $N = 5, 10, \ldots, 35$. The advantage is that the analytic solution is significantly faster than the integration routine in the Fourier approach. However, the drawback of this recursive formula is that the approach becomes unstable for higher values of N. Fortunately, we can easily fix this by increasing the number of time steps n_T.

3.6 Benchmarking of the method

So far, we have empirically investigated the error decay of the method for option prices and their derivatives. In this section, we compare the dynamic Chebyshev method with the least-squares Monte Carlo approach of [88] in terms of accuracy and runtime. Here, we consider American options or more precisely, Bermudan options with a high number of exercise rights per year.

We look at a whole option price surface with varying maturities and strikes. We choose 9 different maturities between one month and two years given by

$$T \in \{1/12, 2/12, 3/12, 6/12, 9/12, 1, 15/12, 18/12, 2\}$$

and strikes equally distributed between 80% and 120% of the current stock price $S_0 = 100$ in steps of 5%. We fix $n = 504$ time steps (i.e. exercise rights) per year, which is equivalent to two ticks per trading day (assuming 252 trading days per year). We use a relatively high number of exercise rights to ensure that the solution in discrete time is a good approximation of the continuous time problem of pricing an American put.

We compare the dynamic Chebyshev method to the least-squares Monte Carlo approach. For the dynamic Chebyshev method we also use Monte Carlo simulation to calculate the generalized moments in the pre-computation step. We use a Chebyshev degree of $N = 32, 64, 128$ and $N = 256$ and run both methods for an increasing number of Monte-Carlo paths.

The convergence of the DC method depends on both, the number of nodes N and the number of Monte Carlo paths M. For an optimal convergence behaviour one needs to find a reasonable relationship between these factors. The analysis of the expected convergence behaviour in Section 3.4.3 shows that the number of Chebyshev nodes N should be $c\sqrt{M}$ for a constant $c > 0$. This implies that if we double the number of nodal points, the number of Monte Carlo samples should increase by a factor of four. Therefore, we fix the number of samples M as

$$M \in \{2{,}000,\ 8{,}000,\ 32{,}000,\ 128{,}000\}\,.$$

3.6.1 The Black-Scholes model

As a first benchmark, we use the Black-Scholes model with the same parameters as in Section 3.5 and an initial stock price of $S_0 = 100$. In order to price options with different strikes in one run of the dynamic Chebyshev method, we interpolate in the log-moneyness $x = \log(S/K)$ instead of the log-stock price. We fix an interpolation domain $\mathcal{X} = [\underline{x}, \overline{x}]$ with $\underline{x} = \log(0.2)$ and $\overline{x} = \log(3)$.

Figure 3.6 shows the price surface and the error surface for $N = 256$ and $M = 128000$. The error was estimated by using the COS method as benchmark. We reach a maximal error below $5 \cdot 10^{-3}$ on the whole option surface.

In Figure 3.7 the log-error is shown as a function of the log-runtime for both methods. The left figure compares the total runtimes and the right figure compares the offline runtime. For the dynamic Chebyshev method the total runtime includes the offline-phase and the online phase. The offline-phase consists of the simulation of one time step of the underlying process $X_{\Delta t}$ for $N + 1$ starting values $X_0 = x_k$ and of the computation of the conditional expectations $E[p_j(X_{\Delta t})|X_0 = x_k]$ for $j, k = 0, \ldots, N$. The online phase is the actual pricing of the American option for all strikes and maturities. Similar, the total runtime of the least-squares Monte Carlo method includes the simulation of the Monte Carlo paths (offline-phase) and the pricing of the option via backward induction (online-phase). The corresponding runtimes and errors are displayed in Table 3-A.

We observe that the dynamic Chebyshev method reaches the same accuracy in a significantly lower runtime. For example, a maximum error of 0.1 is reached in a total runtime of $0.16s$ with the dynamic Chebyshev method whereas the LSM approach needs $63.35s$. This means the dynamic Chebyshev method is nearly 400 times faster for the same accuracy. For the actual pricing in the online phase, the gain in efficiency is even higher. We observe that the dynamic Chebyshev method outperforms the least-squares Monte Carlo method in terms of the total runtime and the pure online runtime. Moreover, we observe that the performance gain from splitting the computation into an offline and an online phase is much higher for the dynamic Chebyshev method. For instance, in the example above the online runtime of the dynamic Chebyshev method is $0.04s$ whereas the LSM takes $62.33s$, a factor of 1,500 times more.

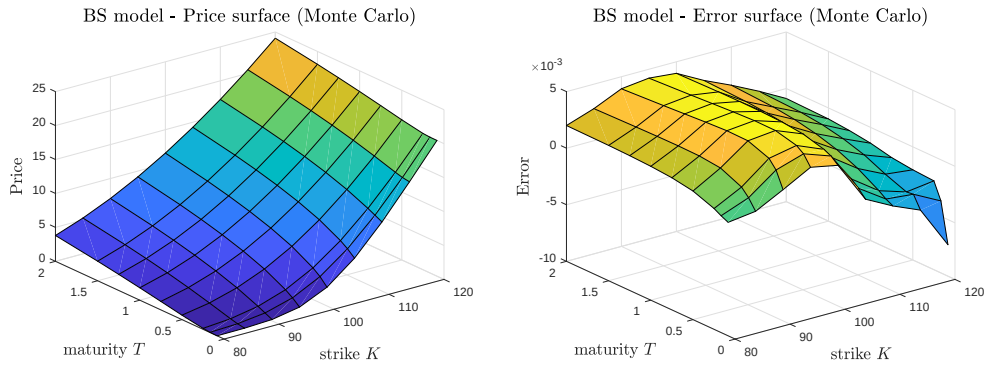

Figure 3.6: Price surface and corresponding error of the dynamic Chebyshev method in the Black-Scholes model. The conditional expectations are calculated with Monte Carlo simulation.

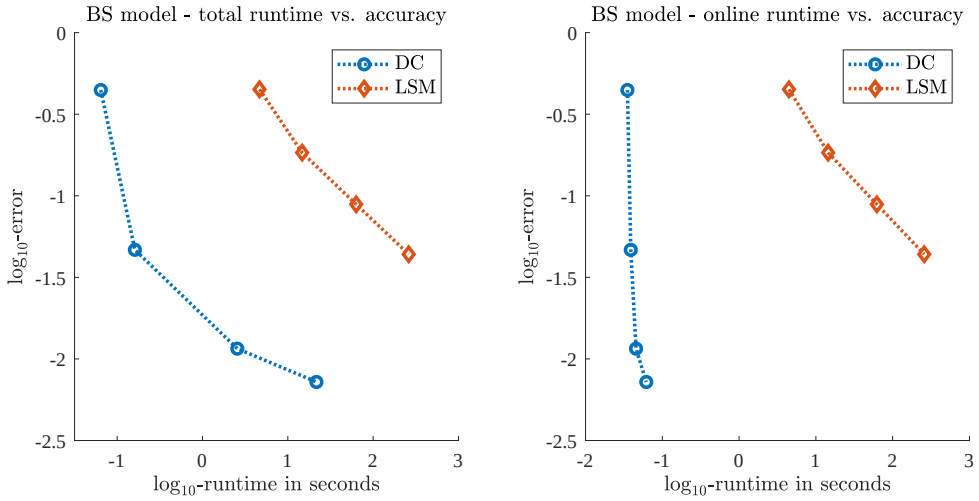

Figure 3.7: Log-Log plot of the total/online runtime vs. accuracy. Comparison of the dynamic Chebyshev method with the least-squares Monte Carlo algorithm.

	dynamic Chebyshev			least-squares Monte Carlo		
paths	total	online	error	total	online	error
M=2000	0.06s	0.04s	0.4447	4.66s	4.46s	0.4491
M=8000	0.16s	0.04s	0.0466	14.74s	14.45s	0.1838
M=32000	2.56s	0.05s	0.0116	63.35s	62.33s	0.0886
M=128000	21.65s	0.06s	0.0072	261.26s	257.70s	0.0438

Table 3-A: Comparison of the dynamic Chebyshev method with the least-squares Monte Carlo algorithm in the Black-Scholes model. The table shows total and online runtime vs. accuracy for different number of MC paths.

One striking advantage of the dynamic Chebyshev method is that once the conditional expectations are calculated, they can be used to price the whole option surface. The pure pricing, i.e. the online phase, is highly efficient. Furthermore, one only needs to simulate one time step Δt of the underlying stochastic process instead of the complete path. We investigate this efficiency gain by varying the number of options and the number of time steps (exercise rights). From Section 3.4.2, we know that the computational complexity of the offline phase is independent of the number of time steps and the number of payoffs/options we want to price. Once the generalized moments are calculated the pricing of an option requires only one run of the online time stepping. Figure 3.7 shows that the online runtime even for pricing a complete option surface is less than 1% of the total runtime. Therefore we can expect that varying the number of options and the number of exercise rights has nearly no effect on the total runtime of the dynamic Chebyshev method.

We use both methods to price an increasing number of options on the same underlying and compare the runtimes. We fixed $M = 32{,}000$ simulation paths for both methods and used a Chebyshev degree of $N = 128$. Similarly, we price options with an increasing number of exercise points per year and compare the runtimes. Figure 3.8 compares the total runtime of the dynamic Chebyshev method with the total runtime of the least-squares Monte Carlo method for an increasing number of options and for an increasing number of time steps. As expected, we can empirically confirm that the efficiency gain by the dynamic Chebyshev method increases with the number of options and the number of exercise rights. In both cases, the runtime of the DC method stays nearly constant whereas the runtime of the least-squares Monte Carlo method increases approximately linearly.

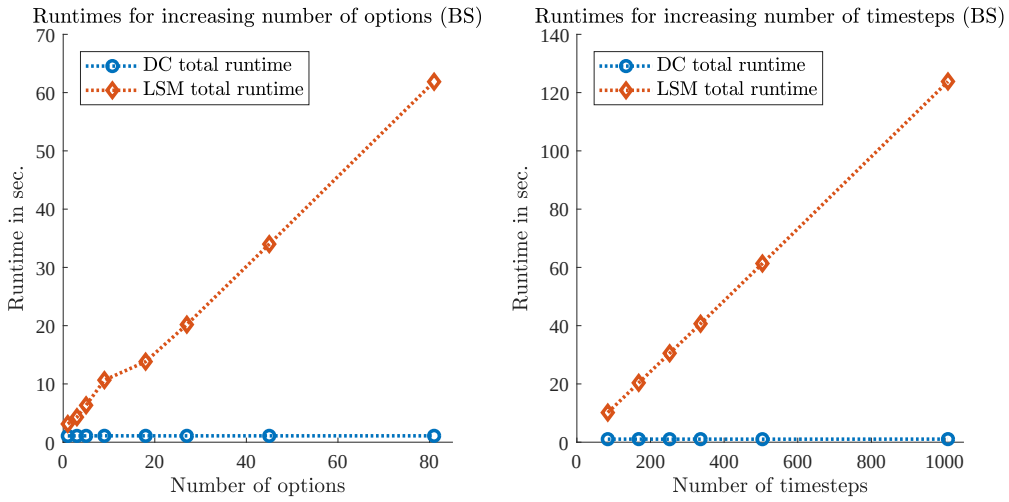

Figure 3.8: Total runtime of the dynamic Chebyshev and the least-squares Monte Carlo method for an increasing number of options (left) and an increasing number of timesteps (right).

3.6.2 The CEV model

Next, we perform the same experiments using the constant elasticity of variance (CEV) model for the underlying stock price process. This model belongs to the class of local volatility models and it is an example of a model where neither the density nor the characteristic form exist in closed form. For the CEV model as defined in (2.39) we fix the following parameters.

$$\sigma = 0.3, \quad r = 0.03, \quad \beta = 1.5.$$

Again, we compare the dynamic Chebyshev and the least-squares Monte Carlo method by computing the prices of an option price surface. We use the same parameter specifications for K, T and n_T. We fix an interpolation domain $\mathcal{X} = [\underline{x}, \overline{x}]$ with $\underline{x} = \log(50)$ and $\overline{x} = \log(180)$.

Figure 3.9 shows the price surface and the error surface for $N = 256$ and $M = 128{,}000$. The error is calculated using a binomial tree implementation for the CEV model based on [100].

In Figure 3.10 the log-error is shown as a function of the log-runtime for both methods. The left figure compares the total runtimes and the right figure compares the offline runtimes. The corresponding runtimes and error levels are also displayed in Table 3-B. Again, we observe that the dynamic Chebyshev method is faster for the same accuracy and it profits more from an offline-online decomposition. For example, the total runtime

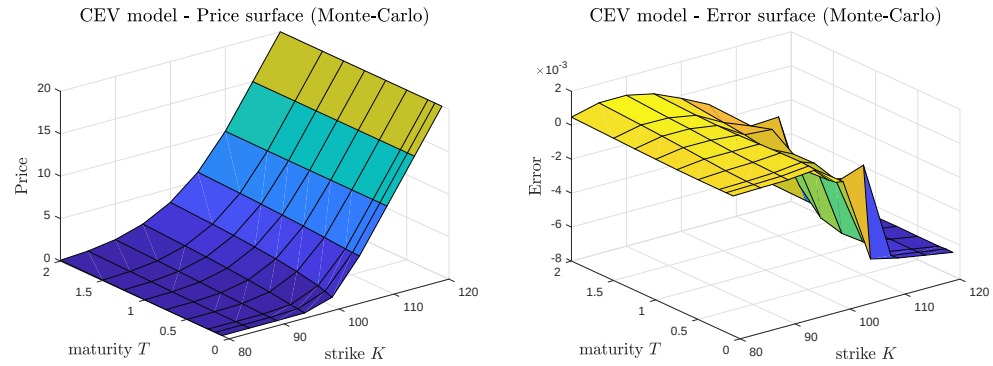

Figure 3.9: Price surface and corresponding error of the dynamic Chebyshev method in the CEV model. The conditional expectations are calculated with Monte Carlo simulation.

of the dynamic Chebyshev method to reach an accuracy of approximately 0.01 is $2.6s$ whereas the least-squares Monte Carlo method takes $444s$. For the online runtimes this out-performance is $1s$ to $408s$.

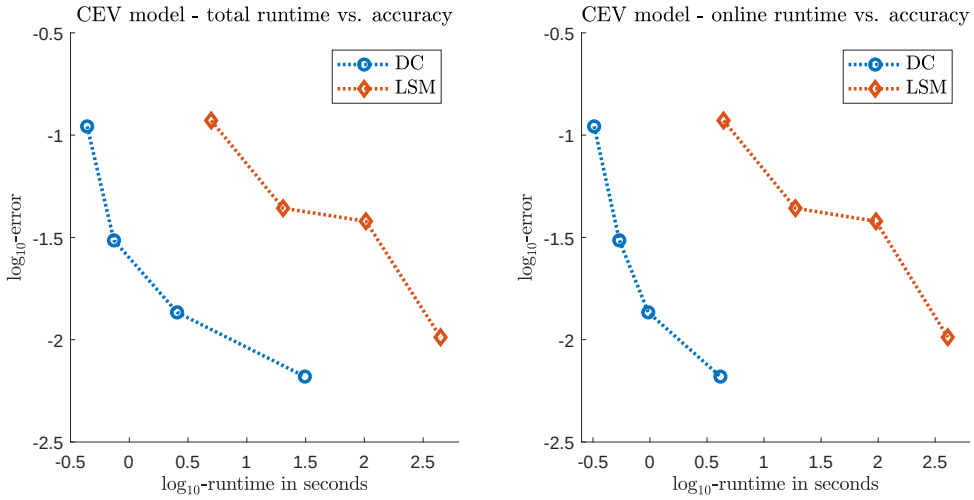

Figure 3.10: Log-Log plot of the total/online runtime vs. accuracy. Comparison of the dynamic Chebyshev method with the least-squares Monte Carlo algorithm.

Investigating this efficiency gain further, we look at the performance for different numbers of options and time steps (exercise rights). Similarly to the last section, Figure 3.11 compares the total runtime of the DC method with the total runtime of the LSM method for an increasing number of options and time steps. In both cases, the runtime of the DC method stays nearly constant whereas the runtime of the LSM method increases approximately linearly. This observation is consistent with the theoretical considerations in Section 3.4.2 and the findings for the Black-Scholes model in Section 3.6.1.

	dynamic Chebyshev			least-squares Monte Carlo		
paths	total	online	error	total	online	error
M=2000	0.44s	0.32s	0.1103	4.97s	4.41s	0.1179
M=8000	0.74s	0.54s	0.0306	20.34s	18.81s	0.0439
M=32000	2.55s	0.97s	0.0136	102.98s	95.65s	0.0379
M=128000	31.18s	4.14s	0.0066	444.87s	407.56s	0.0103

Table 3-B: Comparison of the dynamic Chebyshev method with the least-squares Monte Carlo algorithm in the CEV model. The table shows total and online runtime vs. accuracy for different number of MC paths.

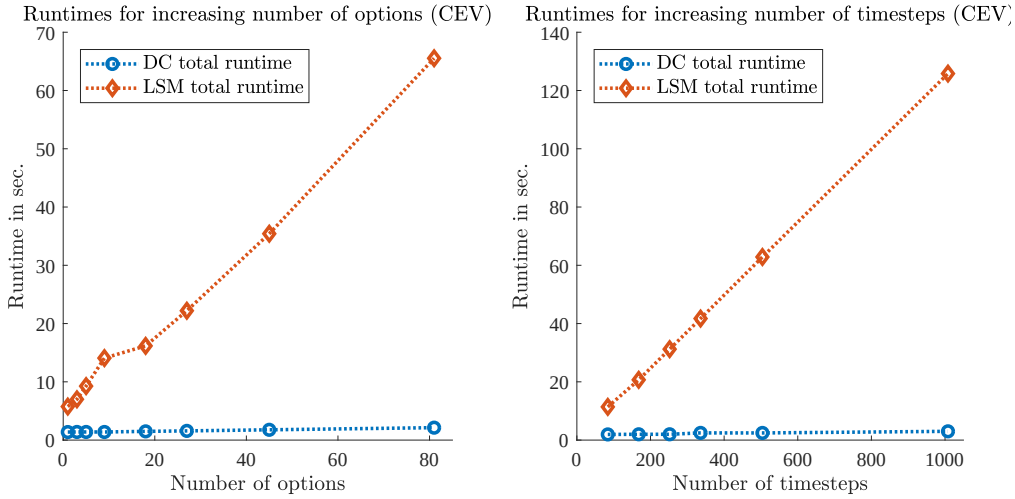

Figure 3.11: Total runtime of the DC and the LSM method for an increasing number of options (left) and an increasing number of timesteps (right).

3.6.3 Pre-computation step: Analytic formula vs. Simulation

For the benchmarking against the least-squares Monte Carlo method we have also used Monte Carlo simulation to compute the generalized moments. In general, this will be the slowest and least efficient approach to compute these moments due to the slow convergence of the Monte Carlo simulation. From the complexity analysis we know that the effort increases quadratic in the number of Monte-Carlo simulations. Recall that we chose $N \approx \sqrt{M_{MC}}$ and thus the offline complexity $\mathcal{O}(N^2 M_{MC})$ becomes $\mathcal{O}(N^4)$. This limits the efficiency of the overall method.

Nevertheless, we observed that the new method outperformed the least-squares Monte Carlo approach. We expect this performance gain to be even bigger if a different technique is used in the pre-computation step. We will illustrate this by comparing Monte

Carlo simulation to the analytic formula for the generalized moments.

In the CEV model, the log-returns are not normally distributed and neither the density nor the characteristic function of the underlying are available in closed-form. We show that it is nevertheless possible to use the analytic formula as well as numerical quadrature. The stock price in the CEV model is modelled by SDE (2.39) and we can approximate $S_{\Delta t}$ for small $\Delta t > 0$ by

$$S_{\Delta t} \approx S_0 + rS_0\Delta t + \sigma S_0^{\beta/2}\sqrt{\Delta t}Z \quad Z \sim \mathcal{N}(0,1)$$

simulating from this discretization yields the Euler—Maruyama scheme which we used in the previous experiments. Similarly we can find an expression for the log-stock price $X_t = \log(S_t)$. Ito's formula yields

$$\begin{aligned}
\mathrm{d}X_t &= \frac{1}{S_t}\big(rS_t\mathrm{d}t + \sigma S_t^{\beta/2}\mathrm{d}W_t\big) - \frac{1}{2}\frac{1}{S_t^2}\sigma^2 S_t^{\beta}\mathrm{d}t \\
&= r\mathrm{d}t - \frac{1}{2}\sigma^2 S_t^{\beta-2}\mathrm{d}t + \sigma S_t^{\beta/2-1}\mathrm{d}W_t \\
&= \big(r - \frac{1}{2}\sigma^2(e^{X_t})^{\beta-2}\big)\mathrm{d}t + \sigma(e^{X_t})^{\beta/2-1}\mathrm{d}W_t
\end{aligned}$$

and thus we obtain for $X_{\Delta t}$ the approximation

$$X_{\Delta t} \approx X_0 + (r - \frac{1}{2}\sigma^2(e^{X_0})^{\beta-2}\big)\Delta t + \sigma(e^{X_0})^{\beta/2-1}\sqrt{\Delta t}Z \quad Z \sim \mathcal{N}(0,1).$$

We define the right hand side as $\widehat{X}_{\Delta t}$ and we can directly see that $\widehat{X}_{\Delta t} \sim \mathcal{N}(X_0 + (r - \frac{1}{2}\sigma^2(e^{X_0})^{\beta-2}\big)\Delta t, \sigma^2(e^{X_0})^{\beta-2}\Delta t)$. We could now either use the density of the normal distribution or its characteristic function and numerical quadrature or we can directly explore the analytic formula for the generalized moments.

We repeat the benchmarking experiments form Section 3.6.2 and include the dynamic Chebyshev method with the analytic formula in the pre-computation step. Figure 3.12 shows the comparison of the dynamic Chebyshev methods and the least-square Monte Carlo method (left plot) and the runtimes of the simulation and the analytic formula approach as a function of the number of nodal points N (right plot). We observe that using the analytic formula in the pre-computation step reduces the overall runtime significantly due to the lower computational complexity, see the discussion in Section 3.4.2.

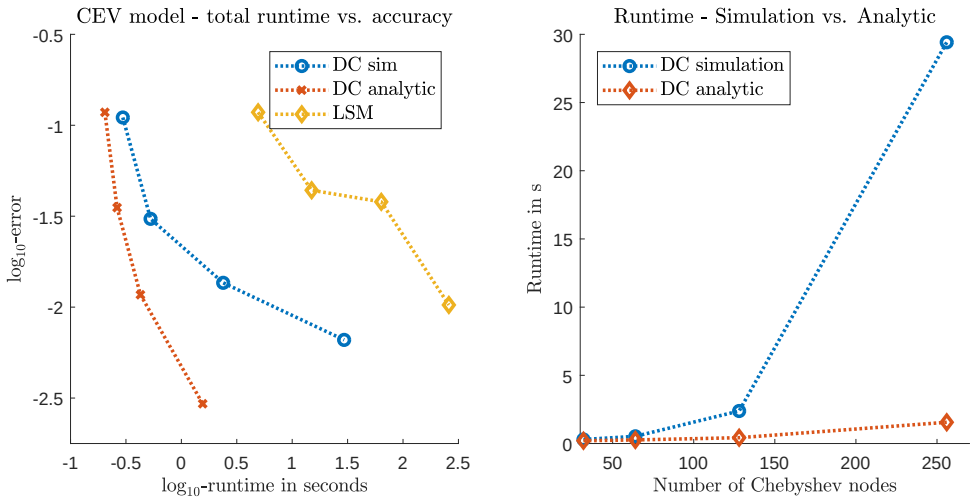

Figure 3.12: Comparison of the DC method for the CEV model using simulation vs. analytic formula in the pre-computation step. Plot of the total runtime vs. accuracy (left hand side) and of the runtime as a function of the number of nodal points.

3.7 Extension of the method and outlook

In this chapter, we introduced a new pricing method for path-dependent options with a focus on barrier options and early-exercise options. Core of the approach is the approximation of the value function in every time-step of a dynamic programming problem with a weighted sum of basis functions $\sum_j c_j^t p_j$. This means the new approach belongs to the group of pricing methods that explore function approximation techniques in finance. We focussed on Chebyshev polynomial interpolation because of its promising properties discussed in Chapter 2. Other function approximation methods could be used as well, especially in multivariate dimensions. Important is the fast convergence of the method and the efficient computation of the expectations of the basis functions.

We provided a comprehensive theoretical error analysis of the method for a dynamic programming problem in d dimensions and we also investigated an extension of the method with splitting. We have seen that the error converges exponentially fast for discretely monitored barrier options and algebraically for American (and Bermudan) options. For the latter one an exponential error decay can be achieved if the additional splitting at the optimal exercise boundary is used. We discussed the implementation of the method in detail and provided a recursive formula for the conditional expectation of the Chebyshev basis functions if the underlying is normally distributed.

We validated our theoretical results numerically and provided an empirical error analysis using different models and option types. Besides the Black-Scholes model we considered a local volatility model (CEV model) and a model with jumps (Merton's

jump-diffusion model). Moreover, we compared the performance of the method to the regression based Monte Carlo method of [88]. This comparison is a first indicator of the high numerical potential of the new pricing method.

So far, the numerical testing was mainly a proof of concept. In the next two chapters we consider the two main applications in the method as motivated in the introduction in Chapter 1. First, the calculation of credit exposures and second, the pricing of early-exercise options that depend on multiple risk factors. Besides these two applications, one can think of several other problems where the dynamic Chebyshev method could be successfully applied.

So far, we investigated the method only for equity options but the method is not limited to this asset class. We will see in the next chapters that it can also be used for the pricing of Bermudan interest rate swaptions and callable bonds (i.e. bonds with an embedded option). An application to other fixed income products or to foreign exchange options is possible. Another type of products that provides an interesting application of the presented method are so-called swing options in energy markets. These options are early-exercise options with more than one exercise right and they can also be priced via (multiple) backward-induction(s). For example, [17] as well as [81] extend the least-squares Monte Carlo method to swing options and [140] apply the COS method to solve this pricing problem.

Efficiency of the method and limitations

In one dimension, there are two main reasons for the efficiency of the new method. The first reason is that all stochastic parts are shifted into the generalized moments $\Gamma_{j,k}$ that can be computed in a pre-computation step. These moments are available in closed form if the underlying process has Gaussian increments. In this case, no approximation error or simulation error is made with respect to the underlying distribution. This allowed us to consider a fine time-discretization with small Δt that is often difficult to handle for function approximation methods. Moreover, the same conditional expectations can be used to price several options on the same underlying.

The second reason for efficiency is the fast convergence and the low number of grid points. For a discretely monitored barrier option the new method was able to achieve an accuracy below 10^{-10} with less than 100 grid points. One reason for the fast error decay was the application of the smoothing the payoff idea mentioned in Remark 4. In general, this smoothing is beneficial as long as one can efficiently calculate the prices of European options in the underlying stochastic model.

For early-exercise options we observed a different convergence behaviour even with the

smoothing. The value function itself is not analytic and a higher number of grid points is required to achieve satisfying error levels. We were able to recover the exponential convergence by applying a domain splitting in every time-step. This splitting comes however with the computational drawback of computing the exercise point in every time step. Since the domain varies over time, a pre-computation of the conditional expectations is no longer possible. In the following, we will briefly introduce two possible modifications or extensions of the method that address this limitations. The first one is a simplification of the splitting approach and can be seen as an intermediate step between the standard dynamic Chebyshev method and the one with splitting. The second idea is a parametric version of the dynamic Chebyshev method with splitting that enables again a form of offline-online decomposition. More details on these modifications of the dynamic Chebyshev method can be found in [91].

Fixed splitting at the strike K

The problematic part about the dynamic Chebyshev method with splitting is the search of a new exercise boundary in every step. For an American put option, we propose a fix splitting at the strike K at every time step in Algorithm 3, i.e. we split $[\underline{x}, \overline{x}]$ into $[\underline{x}, k]$ and $[k, \overline{x}]$ for $k = \log(K)$. This simplifies the algorithm significantly and allows again for a decomposition into a pre-computation step and the backward time-stepping.

Assume the value function at time point t_{u+1} is approximated by two Chebyshev interpolants, i.e. $V_{t_{u+1}} = \widehat{V}^1_{t_{u+1}} \mathbb{1}_{[\underline{x},k]} + \widehat{V}^2_{t_{u+1}} \mathbb{1}_{(k,\overline{x}]}$. In order to approximate V_{t_u} we require the nodal values for two sets of nodal points x^1_k and x^2_k given by

$$V_{t_u}(x^1_k) = \max\Big\{ g(t_u, x^1_k), \sum_{j=0}^{N_1} c^1_j(t_{u+1}) \mathbb{E}\big[p_j(X_{t_{u+1}}) \mathbb{1}_{[\underline{x},k]} | X_{t_u} = x^1_k\big] + \sum_{j=0}^{N_2} c^2_j(t_{u+1}) \mathbb{E}\big[p_j(X_{t_{u+1}}) \mathbb{1}_{(k,\overline{x}]} | X_{t_u} = x^1_k\big] \Big\}$$

and the equivalent expression for the values $V_{t_u}(x^2_k)$. Hence, in the pre-computation step we need to calculate four different sets of conditional expectations for polynomials $p_j \mathbb{1}_{[\underline{x},k]}$ and $p_j \mathbb{1}_{(k,\overline{x}]}$ and starting values x^1_k and x^2_k. In comparison to a Chebyshev interpolation on the whole domain $[\underline{x}, \overline{x}]$ with N points we can choose a lower N_1, N_2. If we set $N_1 = N_2 = N/2$, the number of conditional expectations which we have to compute in the pre-computation step is exactly the same.

This modification of the algorithm combines different advantages in one method. First, it makes the smoothing the payoff idea obsolete since the payoff is smooth on each subinterval $[\underline{x}, k]$ and $[k, \overline{x}]$. Second, we know that the option price changes its

behaviour around the strike and the two interpolations require often less grid points than one interpolation on a large interval. We will provide a small toy example to illustrate the potential of this modification.

We consider again a Bermudan put option in the Black-Scholes model and compare the error decay of the dynamic Chebyshev method with and without splitting at the strike. We fix an initial stock price $S_0 = 100$, a strike $K = 100$, a volatility $\sigma = 0.25$ and an interest rate of $r = 0.03$. We consider an option with maturity $T = 1$ and 32 exercise rights. As a benchmark, we use again the COS method of [43]. Figure 3.13 shows the resulting error decay for both approaches. We observe that the method with splitting yields a lower error for the same number of nodal points. This modification of

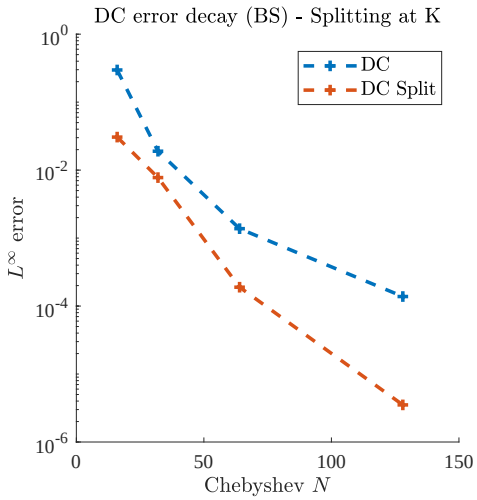

Figure 3.13: Error decay of the dynamic Chebyshev method and the dynamic Chebyshev method with splitting at the srike for a Bermudan option in the BS model. The conditional expectation of the Chebyshev polynomials are calculated with the Fourier transformation.

the dynamic Chebyshev method seems to be promising and we will make use of it in the next chapter.

Parametric dynamic Chebyshev method

So far, we used the dynamic Chebyshev method to price options for a fixed set of modal parameters. We extend this method to a parametric pricing algorithm using the so-called Magic point empirical interpolation. The resulting method is able to handle the domain splitting more conveniently by considering the splitting point as an additional parameter.

Consider the problem of interpolating a parametric set of functions $\mathcal{U} := \{h_p : \Omega \to \mathbb{R} \,|\, p \in \mathcal{P}\}$ where $\Omega \subset \mathbb{C}^d$ is a compact domain and $\mathcal{P}$ a compact parameter set. [6]

proposed a greedy procedure wherein the objective is to minimize the L^∞-error for the approximation of functions in the set $\mathcal{U}$. This greedy algorithm results in

- points $z_1^*, \dots, z_M^* \in \Omega$, the so-called Magic points
- basis functions $\theta_1^M, \dots, \theta_M^M$.

The resulting Magic point interpolation operator is given by

$$h_p(z) \approx \sum_{m=1}^{M} h_p(z_m^*)\theta_m^M(z) \quad \forall z \in \Omega, \quad \forall p \in \mathcal{P}.$$

This interpolation method can be turned into a quadrature rule by integrating the basis functions. [50] proposed to use this method to efficiently calculate option prices via the Fourier pricing formula (2.41). This motivates us to use Magic point interpolation to compute the generalized moments in an efficient way via the Fourier formula (3.19). We define the parametric integrands

$$h_p(z) = \Re\Big(\widehat{p_j}(-z)e^{iz(1-2\frac{\overline{x}}{\overline{x}-\underline{x}})}\ \varphi_{x_0}\big(\frac{2z}{\overline{x}-\underline{x}}\big|\tilde{p}\big)\Big)$$

with $p = [\underline{x}, x_0, \tilde{p}]$, where $\tilde{p}$ are parameters of the underlying stochastic process. Using the Magic Point algorithm we find magic points $z_1^\star, \dots, z_M^\star$ and functions $\theta_1^M, \dots, \theta_M^M$ such that the functions h_p can be approximated by

$$h_p(z) \approx \sum_{m=1}^{M} h_p(z_m^\star)\theta_m(z)$$

and thus

$$\begin{aligned}\mathbb{E}[p_j(X_{\Delta t})|X_0 = x_0] &\approx \frac{1}{\pi}\sum_{m=1}^{M} h_p(z_m^\star)w_m^M \\ &= \frac{1}{\pi}\sum_{m=1}^{M} \Re\Big(\widehat{p_j}(-z_m^\star)e^{iz_m^\star(1-2\frac{\overline{x}}{\overline{x}-\underline{x}})}\ \varphi_{x_0}\big(\frac{2z_m^\star}{\overline{x}-\underline{x}}\big|\tilde{p}\big)\Big)w_m^M\end{aligned}$$

with weights $w_m^M = \int_{-\infty}^{\infty}\theta_m(z)\mathrm{d}z$.

The advantage is that the greedy algorithm for the Magic point has only to be done once for each model. Afterwards, the pricing becomes very efficient and a splitting of the domain can be performed dynamically. A detailed investigation of this algorithm goes beyond the scope of this thesis and is left for future research. First numerical experiments showed already a fast error decay of this algorithm. Moreover, [91] obtained promising results for a similar parametric extension of the dynamic Chebyshev method.

Chapter 4

Efficient computation of credit exposure

In this chapter, we discuss how credit exposures in pricing and risk management can be efficiently calculated using Chebyshev interpolation. Credit exposures are used to estimate, for example, counterparty credit risk (and consequently the regulatory capital of financial firms), initial margins of collateralized trades, Credit Valuation Adjustments (CVA), Debit Valuation Adjustments (DVA) and, more recently, Funding Valuation Adjustments (FVA). The exposure of a trade at time t is defined as

$$E_t(X_t) = \max\{V_t(X_t), 0\},$$

where X_t is the risk factor that drives the price V_t at time t of a portfolio of derivatives. In essence, the credit exposure calculation projects forward in time the distributions of relevant underlying assets, which follow appropriate stochastic models, and obtains the associated distributions of the values of the derivatives in scope, up to their longest maturity. The specifics of this calculation vary with each application. For example, for CVA and DVA the calculation is performed at netting set level while for FVA is done at portfolio level. Netting means that positions with positive and negative exposure can offset each other. A netting set refers than to all derivatives traded with the same counterparty that can be used for netting. For CVA, negative exposures are floored to zero before taking a discounted average under the risk-neutral pricing measure $\mathbb{Q}$. In contrast, in order to quantify credit risk, one needs to assess the distribution of the exposure $E_t(X_t)$ under the real-world measure $\mathbb{P}$. For instance, the upper quantiles at the level of 95, 97.5 or 99% are standard quantities in risk management.

The mentioned distributions are usually obtained through Monte Carlo simulation: On some chosen time points, the derivatives are re-evaluated on various scenarios, randomly drawn from the distribution of the underlying asset, and from the resulting distribution the required metric is extracted. This ansatz for the exposure calculation is often called a full re-evaluation approach. See [64] and [63] for an overview of credit exposure and its calculation. The crux of the calculation are the repeated calls of the pricers which can be computationally expensive.

We propose to use the dynamic Chebyshev method to efficiently compute credit exposures of path-dependent options under both, the risk-neutral and the real-world measure. We introduce the basic definitions of credit exposure for pricing and risk management and provide a static Chebyshev solution. Then we discuss how the dynamic Chebyshev pricing method can be used for the exposure calculation of path-dependent options. We numerically investigate the new approach for exposure calculation and the economic advantages of a full re-evaluation over common ad-hoc simplifications.

Our numerical investigation confirms that the proposed method is able to produce accurate exposure profiles under the risk-neutral and the real-world measure. In our numerical experiments we validate the method for three different equity products (European, barrier and Bermudan option) and a Bermudan interest rate swaption. As models we consider the Black-Scholes and the Merton jump-diffusion stock price model and the Hull-White short rate model. We benchmark our method against a least-squares Monte Carlo approach. The numerical comparison reveals that the proposed method is able to deliver a higher accuracy in a faster runtime.

This chapter is joint work with Kathrin Glau and Ricardo Pachon. The main results in this section have been published in the working paper "Speed-up credit exposure calculations for pricing and risk management" and in an earlier version "Fast Calculation of Credit Exposures for Barrier and Bermudan options using Chebyshev interpolation".

4.1 Credit exposure for pricing and risk management

Credit exposures are important quantities in the pricing of derivatives as well as in risk management of derivative portfolios. For these two different applications there exists also two (slightly) different definitions of expected exposure. We consider a derivative (or a portfolio of derivatives) with price process V_t. For risk and capital calculation purposes, the expected exposure (EE) is defined as

$$EE_0^{risk}(t) = \mathbb{E}^{\mathbb{P}}\big[\max(V_t, 0)|\mathcal{F}_0\big], \tag{4.1}$$

where $\mathbb{P}$ refers to the real-world measure, $\mathcal{F}_0$ is the filtration at $t = 0$, and V_t is the value of the derivative at time t. The potential future exposure (PFE) of the derivative is defined as

$$PFE_0^{risk}(t) = \inf\{y : \mathbb{P}(\max(V_t, 0) \leq y) \geq \alpha\}. \tag{4.2}$$

for a level $\alpha \in (0, 1)$. The price process V_t depends typically on several risk factors such as the value of the underlying, the volatility of the underlying and interest rates. The influence of each risk factor on the price can vary significantly. Therefore it is a reasonable simplification to consider only the most relevant factor(s) for the exposure calculation. For an option this is typically the value of the underlying.

We consider the class of path-dependent options that can be written in the form of (3.1) and (3.2), i.e. that are characterised by a set of exercise dates $t_0, \ldots, t_n = T$, and the value function $V_{t_u}(x)$ is of the form

$$\begin{aligned} V_T(x) &= g(x), \\ V_{t_u}(x) &= f\Big(g(t_u, x), \mathbb{E}^{\mathbb{Q}}[D(t_u, t_{u+1})V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x]\Big), \end{aligned} \tag{4.3}$$

where $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a Lipschitz continuous function, and $g : [0, T] \times \mathbb{R} \to \mathbb{R}$, with $g(T, x) = g(x)$. Here X_t is the underlying risk factor, $D(t_u, t_{u+1}) = B_{t_u}/B_{t_{u+1}}$ is the discount factor between t_u and t_{u+1} and where $B(t)$ is the bank account

$$B(t) = B(0) \exp\Big(\int_0^t r(s)\mathrm{d}s\Big) \quad \text{with} \quad B(0) = 1, \tag{4.4}$$

where r is the money markets continuously compounded interest rate, and $t < T$. We focus on three different option types that can be expresses this way,

- *Bermudan options:* In this case the value function is given as

$$V_{t_u}(x) = \max\Big\{g(x), \mathbb{E}^{\mathbb{Q}}\left[D(t_u, t_{u+1})V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x\right]\Big\}.$$

- *European options:* European options correspond to Bermudan options with no early exercise. In this case the value function becomes

$$V_{t_u}(x) = \mathbb{E}^{\mathbb{Q}}\left[D(t_u, t_{u+1})V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x\right].$$

- *Barrier options:* A discretely monitored up-and-out barrier option with barrier B

can be written in the same form with value function

$$V_{t_u}(x) = \mathbb{E}^{\mathbb{Q}}\left[D(t_u, t_{u+1})V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x\right]\mathbb{1}_{x\leq B}.$$

Similarly, we can use the framework for down-and-out barrier options.

The expected exposure also appears when pricing the basis between the counterparty risk-free value of a trade and its valuation when accounting for counterparty risk. This difference arises from the risk that a trade is in favour of one counterparty but the other one defaults before the trade matures. This Credit Valuation Adjustment (CVA) is equivalent to the price of a contingent credit default swap (CDS), whose value follows from the fundamental arbitrage theorem:

$$\frac{\text{CVA}_0}{B(0)} = \mathbb{E}^{\mathbb{Q}}\left[\int_{s=0}^{s=T}\frac{\max(V_s,0)\cdot d\mathbf{1}_{(\tau\leq s)}}{B(s)}\right] = \int_{s=0}^{s=T}\mathbb{E}^{\mathbb{Q}}\left[\frac{\max(V_s,0)\cdot d\mathbf{1}_{(\tau\leq s)}}{B(s)}\right],$$

where $\mathbb{Q}$ is the associated risk-neutral measure and $\mathbf{1}_{(\tau\leq s)}$ is the default indicator for the counterparty which equals 1 if s is less than the default time τ and 0 otherwise. The integral over time can be discretized over time buckets, and in the special case that the value of the derivative and the default event are independent, the expectation can be expressed as the product of two terms, one accounting exclusively for the default probability and the other one for the positive exposure of the trade. This exposure is calculated as

$$EE_0^{price}(t) = \mathbb{E}^{\mathbb{Q}}\Big[\frac{\max(V_t,0)}{B(t)}\Big|\mathcal{F}_0\Big] = D(0,t)\mathbb{E}^{\mathbb{Q}}\big[\max(V_t,0)|\mathcal{F}_0\big], \tag{4.5}$$

assuming that $B(0) = 1$. Moreover, we define the $\mathbb{Q}$-counterpart of PFE^{risk} as

$$PFE_0^{price}(t) = \inf\{y : \mathbb{Q}(D(0,t)\max(V_t,0)\leq y)\geq\alpha\}. \tag{4.6}$$

The differences between the risk and the pricing exposures, i.e., expressions (4.1) and (4.5), is that the former uses the real-world measure for diffusing the risk factors, while the latter uses the risk-neutral measure (the pricing of V_t in both cases, of course, uses $\mathbb{Q}$). Additionally, for pricing exposures we also incorporate a discount factor at time point t. As we will see in Section 4.3, the structure of our methodology does not change much when calculating either one of them.

For many derivative portfolios the expected exposure cannot be calculated analytically and simulation approaches come into play. Risk factors X_t^i, $i = 1,\ldots,M$ are simulated either under the pricing measure $\mathbb{Q}$ or under the real-world measure $\mathbb{P}$. The

expected exposure is then computed as the empirical mean

$$EE_0^{risk}(t) = \mathbb{E}^{\mathbb{P}}[\max\{V_t(X_t), 0\}] \approx \frac{1}{M}\sum_{i=1}^{M} \max\{V_t(X_t^i), 0\}$$

and the potential future exposure as the empirical quantile. Hence, the value of the trade/derivative has to be calculated for a large number of simulated risk factors. If there is no analytic solution for the valuation available a full re-evaluation becomes computationally demanding. When there is no closed form solution for the price of the derivative, e.g. for path-dependent options, a straightforward approach would lead to nested Monte Carlo simulations. Moreover, often a high number of scenario simulations is required to obtain stable results, particularly for tail measures. In credit risk management an additional challenge arises from the change of measure, i.e. scenarios need to be generated under the real-world measure, nonetheless pricing is done under the risk-neutral measure. Hence, additionally to simulating the paths of the underlying under $\mathbb{Q}$, the scenario paths need to be simulated under $\mathbb{P}$. A naive simplification would be to assemble the risk quantities also under the pricing measure $\mathbb{Q}$. As reported in [122], *"since the banks are already heavily invested in CVA calculations, it is becoming popular to take this shortcut"*. The analysis of [122] clearly shows the perils of this approach and emphasizes the importance of calculating credit risk quantities under the real-world measure.

In the literature regression-based methods are studied in order to avoid nested Monte Carlo simulation, see for instance [112], who calculates the exposure and CVA for derivatives without analytic solution (e.g. Bermudan options) based on a modification of the least-squares Monte Carlo approach of [88]. For the exposure calculation under the real-world measure in a Black-Scholes type model a change of measure using the Radon–Nikodym density is employed. Furthermore, [80] and [45] apply the stochastic grid bundling method (SGBM) of [76] to credit exposure calculation and compare it to a least-squares Monte Carlo algorithm. Their comparison reveals severe deficiencies of the L-S approach. Namely, the L-S price introduces numerical noise that leads to inaccurate exposure, especially in its tail distribution. While the bundling technique in the SGBM is able to reduce the Monte Carlo noise and produce more accurate results, it comes at a higher computational cost. A different method is investigated in [117], who calculate the exposure for Bermudan options on one asset, based on the COS method for early-exercise options of [44]. The method produces accurate results under $\mathbb{Q}$ and $\mathbb{P}$ without any change of measure. However, due to its higher runtimes it is mostly suitable for benchmarking.

4.2 A static Chebyshev approach for exposure calculate

The repeated calls of a numerical option pricing routine for different realizations of an underlying risk factor makes the computation of credit exposure an interesting application for Chebyshev interpolation. In Section 2.3.3 we presented the idea of a static Chebyshev method for parametric option prices. If the same pricer is repeatedly called for varying parameters one can simple approximate the pricer using Chebyshev interpolation. Since option prices are often analytic functions of their parameters a few Chebyshev nodes are sufficient to obtain accurate approximations. We suggest to use this idea to speed-up the credit exposure calculation via full re-evaluation. If the existing pricer for a derivative is slow, the static Chebyshev approach can reduce runtimes significantly.

In the exposure calculation, we propose to approximate the function $\boldsymbol{x} \mapsto V_t(\boldsymbol{x})$ with Chebyshev polynomials, i.e.

$$V_t(\boldsymbol{x}) \approx \widehat{V}_t(\boldsymbol{x}) = \sum_{\|\boldsymbol{j}\|_\infty \leq n} c_{\boldsymbol{j}} p_{\boldsymbol{j}}(\boldsymbol{x}) \quad \text{for} \quad \boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$$

and to replace the value function with its Chebyshev approximation

$$EE_t(\boldsymbol{x}) = \mathbb{E}^{\mathbb{Q}}[\max\{V_t(\boldsymbol{X}_t), 0\}] \approx \frac{1}{M} \sum_{i=1}^{M} \max\{\widehat{V}_t(\boldsymbol{X}_t^i), 0\}.$$

This is a good example of a direct and practical useful application of Chebyshev interpolation in finance. We know that the function $\boldsymbol{x} \mapsto V_t(\boldsymbol{x})$ is analytic for many option payoffs and asset models. Hence, only a relatively few nodal points are required to get an accurate estimation of the price function. Moreover, the number of simulated scenarios M is typically large (e.g. 10,000, 50,000 or 250,000) and the two aspects combined yield a large efficiency gain compared to a full re-evaluation. In the following, we describe the resulting algorithm for the credit exposure calculation of European options.

Algorithm: Credit exposure of European options

Let $(\boldsymbol{X}_t)_{t \leq T}$ be an $\mathbb{R}^d$-valued stochastic process of risk factors. Calculate the expected exposure for a European option with value $V_t(\boldsymbol{x}) = \mathbb{E}^{\mathbb{Q}}[D(t,T)g(\boldsymbol{X}_T)|\boldsymbol{X}_t = \boldsymbol{x}]$.

Simulation step:

1. Simulate paths $\boldsymbol{X}_{t_0}^i, \ldots, \boldsymbol{X}_{t_n}^i$, $i = 1, \ldots, M$ of the risk factors $\boldsymbol{X}_t$ either under the pricing measure $\mathbb{Q}$ or under the real world measure $\mathbb{P}$.

Exposure calculation at time step t_u:

1. Define the interpolation domain

$$\mathcal{X}^{t_u} = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_d, \overline{x}_d]$$

   with $\underline{x}_j = \min_{1 \le i \le M} X^i_{t_u,j}$ and $\overline{x}_j = \max_{1 \le i \le M} X^i_{t_u,j}$ for $j = 1, \ldots, d$.

2. Define nodal points $\boldsymbol{x_k} = \tau_{\mathcal{X}^{t_u}}(\boldsymbol{z_k})$, $\|\boldsymbol{k}\|_\infty \le n$ for a given Chebyshev degree n.

3. Calculate the option price at the nodal points $V_t(\boldsymbol{x_k})$ using an existing numerical pricing routine.

4. Compute Chebyshev coefficients $c_{\boldsymbol{j}}$ and set up the Chebyshev interpolation

$$V_{t_u}(\boldsymbol{x}) \approx \widehat{V}_{t_u}(\boldsymbol{x}) = \sum_{\|\boldsymbol{j}\|_\infty \le n} c_{\boldsymbol{j}} p_{\boldsymbol{j}}(\boldsymbol{x}).$$

5. Price the option on each sample paths $i = 1, \ldots, M$ and compute the exposure

$$E^i_{t_u} = \max\{\widehat{V}_{t_u}(\boldsymbol{X}^i_{t_u}), 0\} = \max\Big\{ \sum_{\|\boldsymbol{j}\|_\infty \le n} c_{\boldsymbol{j}} p_{\boldsymbol{j}}(\boldsymbol{X}^i_{t_u}), 0 \Big\}.$$

6. Calculate empirically the required risk metric, e.g. the expected exposure

$$EE^{risk}_0 = \mathbb{E}^{\mathbb{P}}\left[\max\{V_{t_u}, 0\}\right] \approx \frac{1}{M} \sum_{i=1}^{M} E^i_{t_u}.$$

Note that the value function $V_t(\boldsymbol{x})$ might also be the value of a portfolio of European options instead of the value of a single option. This means the netting effect can be incorporated by looking at the portfolio of all derivatives traded with one counterparty.

Advantages and drawbacks of the static approach

Once the Chebyshev approximation of the value function is set-up its evaluation can be performed very efficiently and will typically be much faster than the original pricer. This allows us to easily increase the number of simulation paths without increasing the runtime accordingly. The new approach is particularly beneficial when the standard pricer for the option is relatively slow. For example a (multivariate) integral, a PDE method or a Monte Carlo solver. Instead of solving the problem on a typically large grid we only require the evaluation of a low number of polynomials. Depending on the required accuracy the number of nodal points n is chosen. Especially when a relatively low accuracy is required this can increase efficiency even further.

The number of Chebyshev points needed to achieve a required accuracy depends highly on the smoothness of the value function and the size of the interpolation domain. For short maturities the option price might be less smooth. In this case convergence is often improved by splitting the interpolation domain in several subdomains, see [104]. If the simulated risk factor is a diffusion process few outliers can lead to a very large interpolation domain. In this case it might be beneficial to interpolate on a smaller domain and treat the outliers separately. Note that we can only expect to increase efficiency if $(n+1)^d < M$, i.e. the number of nodal points is smaller than the number of simulation paths. Otherwise we would calculate more prices at the nodal points than we actually need.

The drawback of this simple approach is however, that the method still relies on the availability of an existing pricer for each product and the efficiency gain is mainly driven by the relationship between evaluating the Chebyshev interpolation and calling the pricer. This means the approach requires still nested Monte Carlo simulation for many products but for fewer scenarios than in the full re-evaluation. In the next section, we show how we can use the dynamic Chebyshev method for exposure calculation that does no rely on an existing pricer.

4.3 A dynamic approach for exposure calculation

In this section we presented a unified approach for the calculation of credit exposure for different types of path-dependent options based on the dynamic Chebyshev method. We consider the dynamic programming problem as presented in equation (4.3)

$$\begin{aligned} V_T(x) &= g(x), \\ V_{t_u}(x) &= f\Big(g(t_u,x), \mathbb{E}^{\mathbb{Q}}[D(t_u,t_{u+1})V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u}=x]\Big), \end{aligned}$$

for a Lipschitz continuous function $f : \mathbb{R}\times\mathbb{R}\to\mathbb{R}$ and a function $g : [0,T]\times\mathbb{R}\to\mathbb{R}$ with $g(T,x) = g(x)$.

Using the dynamic Chebyshev method, the backward induction is solved on a finite domain $\mathcal{X} = [\underline{x}, \overline{x}]$. Assume we have at t_{u+1} an approximation $\widehat{V}_{t_{u+1}}$ with $V_{t_{u+1}}(x) \approx \widehat{V}_{t_{u+1}}(x) = \sum_j c_j(t_{u+1})p_j(x)$. In this case the backward induction becomes

$$\begin{aligned} V_{t_u}(x_k) &= f\left(g(t_u,x_k), \mathbb{E}^{\mathbb{Q}}\left[D(t_u,t_{u+1})V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u}=x_k\right]\right) \\ &\approx f\Big(g(t_u,x_k), \sum_{j=0}^{N} c_j(t_{u+1})\mathbb{E}^{\mathbb{Q}}\left[D(t_u,t_{u+1})p_j(X_{t_{u+1}})|X_{t_u}=x_k\right]\Big), \end{aligned}$$

where we exploited the linearity of the conditional expectation. From (2.17) we obtain the following formula for the coefficients of the Chebyshev interpolation of V_t

$$c_j(t_u) = \frac{2^{1_{0<j<N}}}{N} \sum_{k=0}^{N}{}'' V_{t_u}(x_k) T_j(z_k).$$

We see that the coefficients c_j carry the information of the payoff, and the conditional expectations $\mathbb{E}^{\mathbb{Q}}\left[D(t_u, t_{u+1}) p_j(X_{t_{u+1}}) | X_{t_u} = x_k\right]$ carry the information of the stochastic process.

Now, we are in a position to efficiently evaluate the different exposure measures introduced in Section 4.1. Assume we have simulated M paths of the underlying risk factor. Then we price the option along the paths using the closed form approximation

$$V_{t_u}(X^i_{t_u}) \approx \sum_{j=0}^{N} c_j(t_u) p_j(X^i_{t_u}) \quad \text{for} \quad i = 1, \ldots, M \quad \text{and} \quad u = 0, \ldots, n.$$

These values can now be used to calculate the expected exposure or the potential future exposure for a given level α. In the case of a Bermudan option one has to take into account that by exercising the option at t_u the exposure becomes zero. Similarly, if the barrier option is knocked out the exposure at all future time steps is zero. These two effects yield a decreasing exposure for both types of options.

Discounting:
If the interest rate is our risk factor, i.e. $r(t) = r(t, X_t)$, we simplify the expectation of the discounted basis function in the following way. Assume that the time stepping $\Delta t = t_{u+1} - t_u$ is small, then we can write

$$\begin{aligned}
&\mathbb{E}^{\mathbb{Q}}\left[D(t_u, t_{u+1}) p_j(X_{t_{u+1}}) | X_{t_u} = x_k\right] \\
&\quad = \mathbb{E}^{\mathbb{Q}}\left[\exp\Big(-\int_{t_u}^{t_{u+1}} r(s, X_s)\mathrm{d}s\Big) p_j(X_{t_{u+1}}) | X_{t_u} = x_k\right] \\
&\quad \approx \mathbb{E}^{\mathbb{Q}}\left[\exp\big(-\Delta t r(t_u, X_{t_u})\big) p_j(X_{t_{u+1}}) | X_{t_u} = x_k\right] \\
&\quad = \exp\big(-\Delta t r(t_u, x_k)\big) \mathbb{E}^{\mathbb{Q}}\left[p_j(X_{t_{u+1}}) | X_{t_u} = x_k\right],
\end{aligned}$$

where we assume that the discount factor is constant on a small interval. Otherwise, if the discount factor is deterministic we can simply write

$$\mathbb{E}^{\mathbb{Q}}\left[D(t_u, t_{u+1}) p_j(X_{t_{u+1}}) | X_{t_u} = x_k\right] = D(t_u, t_{u+1}) \mathbb{E}^{\mathbb{Q}}\left[p_j(X_{t_{u+1}}) | X_{t_u} = x_k\right].$$

In both cases, we only need to pre-compute the expectations $\mathbb{E}^{\mathbb{Q}}\left[p_j(X_{t_{u+1}}) | X_{t_u} = x_k\right]$.

The resulting pricing algorithm is for all three option types (Bermudan, barrier, European) essentially the same. However, the efficiency of the method is directly related to the smoothness of the value function. As a result the number of nodal points required for a given accuracy varies. Moreover, the size of the interpolation domain influences the number of nodal points that are required for a target accuracy.

The resulting algorithm for exposure calculation under the pricing measure and under the real-world measure with the dynamic Chebyshev method is presented in the following two sections.

Exposure calculation for pricing

Here, we consider the computation of the exposure under the pricing measure $\mathbb{Q}$. The main application is the computation of the expected exposure EE_0^{price} as an ingredient of the CVA calculation.

Algorithm: Exposure of Bermudan options under $\mathbb{Q}$
This algorithm provides a framework to calculate the expected exposure and the potential future exposure for a Bermudan option. A European option can be seen as a special case and falls also in the scope of this algorithm.

1. Simulation of risk factors:
Simulate M paths of the underlying risk factor $X_{t_0}^i, \ldots, X_{t_n}^i$, $i = 1, \ldots, M$ under the pricing measure $\mathbb{Q}$.

2. Preparation of the pricing algorithm:
Find a suitable interpolation domain $\mathcal{X} = [\underline{x}, \overline{x}]$ and calculate the nodal points $x_k = \tau_{\mathcal{X}}(\cos(k\pi/N))$, $k = 0, \ldots, N$ for this domain. Pre-compute the conditional expectations of the basis function under the pricing measure $\mathbb{Q}$

$$\Gamma_{k,j} = \mathbb{E}^{\mathbb{Q}}\left[p_j(X_{\Delta t})|X_0 = x_k\right].$$

3. Initialization of the pricing algorithm:
Start pricing at maturity T and compute nodal values $\widehat{V}_T(x_k) = g(T, x_k)$ for all $k = 0, \ldots, N$ for the payoff function $g(T, x_k)$. Calculate Chebyshev coefficients $c_j(T)$ using the nodal values $\widehat{V}_T(x_k)$. For all paths compute the exposure $E_T^i = \max\{g(T, X_T^i), 0\}$.

4. Exposure calculation via backward induction:
Iterative time stepping $t_{u+1} \to t_u$: Assume we have a Chebyshev approximation $V_{t_{u+1}}(x) \approx \widehat{V}_{t_{u+1}}(x) = \sum_j c_j(t_{u+1}) p_j(x)$. Then

- compute nodal values

$$\widehat{V}_{t_u}(x_k) = \begin{cases} \max\{g(x_k), D_u(x_k)\sum_{j=0}^{N} c_j(t_{u+1})\Gamma_{k,j}\}, & \text{if } t_u \text{ is exercise day,} \\ D_u(x_k)\sum_{j=0}^{N} c_j(t_{u+1})\Gamma_{k,j}, & \text{otherwise} \end{cases}$$

  with discount factor $D_u(x_k) = D(t_u, t_{u+1}, x_k)$

- calculate new coefficients $c_j(t_u)$ using nodal values $\widehat{V}_{t_u}(x_k)$,
- price the option for all simulation paths $V^i_{t_u} = \widehat{V}_{t_u}(X^i_{t_u}) = \sum_{j\in J} c_j(t_u)p_j(X^i_{t_u})$,
- calculate exposure $E^i_{t_u} = D(0,t_u)\max\{V^i_{t_u}, 0\}$,
- if the option is exercised (i.e $V^i_{t_u} = g(X^i_{t_u})$), update the exposure at all future time steps on this path $E^i_{t_j}$, $j = u+1, \ldots, n$.

5. Calculation of expected exposure:
Obtain an approximation of the expected future exposures

$$EE_0^{price}(t_u) = \mathbb{E}^{\mathbb{Q}}\left[D(t_u)\max\{V_{t_u}, 0\}\right] \approx \frac{1}{M}\sum_{i=1}^{M} D(t_u)E^i_{t_u},$$

and an approximation of the potential future exposures

$$PFE_0^{price}(t_u) = \inf\left\{y : \mathbb{Q}\big(E_t(x) \le y\big) \ge \alpha\right\} \approx \inf\left\{y : \frac{\#\{E^i_{t_u} \le y\}}{M} \ge \alpha\right\}.$$

for all $u = 0, \ldots, n$.

Modification of the algorithm for barrier options
The presented algorithm for Bermudan options can be modified to calculate the exposure of barrier options. In this case the interpolation domain is chosen depending on the barrier. There is no early exercise, however, we need to take care of the knock-out feature. For an up-and-out call option with barrier B and $b = \log(B)$, the following modifications are added to the algorithm. First, the interpolation domain is set as $\mathcal{X} = [\underline{x}, b]$. Second, the iterative time stepping from $t_{u+1} \to t_u$ is modified in the following way. Assume we have a Chebyshev approximation $V_{t_{u+1}}(x) \approx \widehat{V}_{t_{u+1}}(x) = \sum_j c_j(t_{u+1})p_j(x)$,

- compute nodal values $\widehat{V}_{t_u}(x_k) = D_u(x_k)\sum_{j=0}^{N} c_j\Gamma_{k,j}$ and new coefficients $c_j(t_u)$,
- price the option for all simulation paths $V^i_{t_u} = \widehat{V}_{t_u}(X^i_{t_u}) = \sum_{j\in J} c_j(t_u)p_j(X^i_{t_u})$ if $X^i_{t_u} \le b$ and $V^i_{t_u} = 0$ otherwise,
- calculate the exposure $E^i_{t_u} = D(0,t_u)\max\{V^i_{t_u}, 0\}$,

- if the option is knocked-out, i.e if $X^i_{t_u} > b$ update the exposure at all future time steps on this path $E^i_{t_j}$, $j = u+1, \ldots, n$.

Exposure calculation for risk management

In this section we present an algorithm for the exposure calculation under the real-world measure $\mathbb{P}$.

Algorithm: Exposure of Bermudan options under $\mathbb{P}$
This algorithm provides a framework to calculate the expected exposure and the potential future exposure for a Bermudan option.

1. Simulation of risk factors:
Simulate M paths of the underlying risk factor $X^i_{t_0}, \ldots, X^i_{t_n}$, $i = 1, \ldots, M$ under the real-world measure $\mathbb{P}$.

2. Preparation of the pricing algorithm:
Find a suitable interpolation domain $\mathcal{X} = [\underline{x}, \overline{x}]$ and calculate the nodal points $x_k = \tau_{\mathcal{X}}(\cos(k\pi/N))$, $k = 0, \ldots, N$ for this domain. Pre-compute the conditional expectations of the basis function under the pricing measure $\mathbb{Q}$

$$\Gamma_{k,j} = \mathbb{E}^{\mathbb{Q}}\left[p_j(X_{\Delta t})|X_0 = x_k\right].$$

3. Initialization of the pricing algorithm:
Start pricing at maturity T and compute nodal values $\widehat{V}_T(x_k) = g(T, x_k)$ for all $k = 0, \ldots, N$ for the payoff function $g(T, x_k)$. Calculate Chebyshev coefficients $c_j(T)$ using the nodal values $\widehat{V}_T(x_k)$. For all paths compute the exposure $E^i_T = \max\{g(T, X^i_T), 0\}$.

4. Exposure calculation via backward induction:
Iterative time stepping $t_{u+1} \to t_u$: Assume we have a Chebyshev approximation $V_{t_{u+1}}(x) \approx \widehat{V}_{t_{u+1}}(x) = \sum_j c_j(t_{u+1})p_j(x)$. Then

- compute nodal values

$$\widehat{V}_{t_u}(x_k) = \begin{cases} \max\{g(x_k), D_u(x_k)\sum_{j=0}^N c_j(t_{u+1})\Gamma_{k,j}\}, & \text{if } t_u \text{ is exercise day,} \\ D_u(x_k)\sum_{j=0}^N c_j(t_{u+1})\Gamma_{k,j}, & \text{otherwise} \end{cases}$$

  with discount factor $D_u(x_k) = D(t_u, t_{u+1}, x_k)$

- calculate new coefficients $c_j(t_u)$ using nodal values $\widehat{V}_{t_u}(x_k)$,

- price the option for all simulation paths $V^i_{t_u} = \widehat{V}_{t_u}(X^i_{t_u}) = \sum_{j\in J} c_j(t_u)p_j(X^i_{t_u})$,

- calculate exposure $E^i_{t_u} = \max\{V^i_{t_u}, 0\}$,
- if the option is exercised (i.e $V^i_{t_u} = g(X^i_{t_u})$), update the exposure at all future time steps on this path $E^i_{t_j}$, $j = u+1, \dots, n$.

5. Calculation of expected exposure:
Obtain an approximation of the expected future exposures

$$EE^{risk}_0(t_u) = \mathbb{E}^{\mathbb{P}}\left[\max\{V_{t_u}, 0\}\right] \approx \frac{1}{M}\sum_{i=1}^{M} E^i_{t_u},$$

and an approximation of the potential future exposures

$$PFE^{risk}_0(t_u) = \inf\left\{y : \mathbb{P}\big(E_t(x) \le y\big) \ge \alpha\right\} \approx \inf\left\{y : \frac{\#\{E^i_{t_u} \le y\}}{M} \ge \alpha\right\}.$$

for all $u = 0, \dots, n$.

Similarly to the exposure calculation for pricing we can modify the algorithm for barrier options.

A comparison with the algorithms in the previous section shows that the exposure calculation under $\mathbb{Q}$ and $\mathbb{P}$ has the same structure. The only difference is that the paths of the risk factor(s) are simulated under a different measure and the exposure is not discounted. Moreover, if we are interested in the PFE we need a higher number of simulation paths since the PFE is a tail measure.

Conceptional benefits of the method

The presented algorithms provide efficient solutions for the exposure calculation. Moreover, the structure of the new approach comes with conceptual benefits, which can be exploited in practice. See Chapter 3 for more details.

- *Closed form expression for the conditional expectations*: If the underlying process $X_{t_{u+1}}|X_{t_u} = x$ is normally distributed the conditional expectations of the Chebyshev polynomials $\mathbb{E}^{\mathbb{Q}}\left[p_j(X_{t_{u+1}})|X_{t_u} = x_k\right]$ can be calculated analytically. See Proposition 10. Examples are the Black-Scholes model (with log-stock price X_t), the Vasicek model or the one factor Hull-White model (both with interest rate X_t). More generaly, assume for instance the underlying process is modelled via an SDE of the form

$$\mathrm{d}X_t = \alpha(t, X_t)\mathrm{d}t + \beta(t, X_t)\mathrm{d}W_t$$

for a standard Brownian motion W_t. Euler—Maruyama discretization yields for $X_{t_{u+1}}|X_{t_u} = x$ the approximation

$$X_{t_{u+1}} \approx x + \alpha(t_u, x)(t_{u+1} - t_u) + \beta(t_u, x)\sqrt{t_{u+1} - t_u}Z =: \widehat{X}^x_{t_{u+1}} \qquad Z \sim \mathcal{N}(0,1)$$

and the right hand side is thus normally distributed.

- *Delta and Gamma as by-product of the method*: For Delta and Gamma the polynomial structure of the Chebyshev approximation allows for a direct computation without re-running the time-stepping. Instead we only need to differentiate a polynomial, see Section 3.4.1. For Delta we obtain

$$\frac{\partial V_t}{\partial x}(x) \approx \sum_{j=0}^{N} c_j^t \frac{\partial p_j}{\partial x}(x),$$

  which is again a polynomial with degree $N-1$ and for Gamma we obtain

$$\frac{\partial^2 V_t}{\partial x^2}(x) \approx \sum_{j=0}^{N} c_j^t \frac{\partial^2 p_j}{\partial x^2}(x),$$

  a polynomial of degree $N-2$. These formulas can be used to calculate the derivative of V_t with respect to x_t. For the derivative w.r.t. x_0 we obtain via chain rule $\partial V_t/\partial x_0 = \partial V_t/\partial x_t \cdot \partial x_t/\partial x_0$.

- *Several options on one underlying:* The structure of the dynamic Chebyshev algorithm for exposure calculation exhibits additional benefits for derivative portfolios. For instance, consider non-directional strategies and structured products that offer different levels of capital protection or enhanced exposure. They are typically constructed from a combination of European options, with different strikes and maturities, together with Bermudan options and barrier options. Such structures are essentially a portfolio of derivatives on the same underlying asset, and in this case, the pricing and exposure calculation can be simplified by choosing the same interpolation domain. First, we only need to compute the conditional moments once and then we can use them for all options. Second, we require less computations in the exposure calculation. Assume we have two options and we are at time step t_u of the dynamic Chebyshev algorithm. We have two Chebyshev approximations $\widehat{V}^1_{t_u} = \sum c_j^1(t_u)p_j$ and $\widehat{V}^2_{t_u} = \sum c_j^2(t_u)p_j$. For the exposure calculation we need to compute $\widehat{V}^{1/2}_{t_u}(X^i_{t_u}) = \sum c_j^{1/2}(t_u)p_j(X^i_{t_u})$ for all risk factors $i = 1, \ldots, M$. Hence the evaluation of the Chebyshev polynomials p_j at the risk factors $X^i_{t_u}$ is the same and has only to be done once. In summary, with low additional effort, we can calculate the exposure of several options on one underlying.

We refer to Section 3.4.1 for more details on the efficient implementation of the method.

4.4 Credit exposure of equity options

In this section, we investigate the dynamic Chebyshev method numerically by calculating the credit exposure profiles of European and path-dependent equity options. We analyse the accuracy of the exposure profiles produced by the dynamic Chebyshev method by comparing them to a full re-evaluation. Then we investigate the method's performance and compare it to the popular least-squares Monte Carlo approach. Moreover, we check the influence of the proposed splitting of the domain at the strike, as suggested in Section 3.7, on the method's performance.

4.4.1 Description of the experiments

For the numerical experiments we consider four different products: A European put option and an up-and-out barrier call option in the Black-Scholes model, a Bermudan put option in the Merton jump diffusion model and a Bermudan receiver swaption in the Hull-White short rate model. In the Black-Scholes and the Hull-White model the risk factor is normally distributed and we can use the analytic formula for the conditional expectations of the Chebyshev polynomials.

We compute the expected exposure EE_t^{price} under the pricing measure $\mathbb{Q}$ and the expected exposure EE_t^{risk} under the real-world measure $\mathbb{P}$ as well as the potential future exposures PFE_t^{price} and PFE_t^{risk} under both measures. For the calculation of the exposure measures we use 50,000 and 150,000 simulation paths of the underlying risk factors and a time discretization of 50 time steps per year. The relatively high number of simulation paths is needed to obtain a stable estimate of the PFE over the lifetime of the derivative. Since the PFE is a tail measure it is more sensitive to the number of simulations than the expected exposure.

We run the dynamic Chebyshev method for a different number of nodal points N and the dynamic Chebyshev with splitting approach with $N_1 = N_2 = N/2$ nodal points. For the least-squares Monte Carlo method we use the monomials up to degree 5 plus the payoff of the product as basis functions in the regression. The pricing is done using 150,000 paths of the underlying risk factors and then we use a second set of paths for the calculation of the exposure. Using two different sets of paths for pricing and exposure calculation reduces the bias of the least-squares Monte Carlo method. See [80] for a description on how to use the least-squares Monte Carlo approach to calculate credit exposures under the pricing and the real-world measure. In our implementation of the least-squares Monte Carlo approach for exposure calculation we use 7 basis functions for

the European and Bermudan equity options, 8 for the barrier option and 5 basis function for the Bermudan swaptions. For the pricing we use a separate set of 150,000 simulation paths of the underlying risk factor.

For the experiments in this section and the next section, we briefly recall the following three asset price models and explain how we compute the generalized moments in each of them.

The Black-Scholes model
In the classical model of [15] the stock price process is modelled by the SDE

$$\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t.$$

with drift μ and volatility $\sigma > 0$ under the real-world measure $\mathbb{P}$. Under the pricing measure $\mathbb{Q}$ the drift equals r. Exploiting the fact that the log-returns $X_t = \log(S_t/S_0)$ are normally distributed we can use the analytic formula for the generalized moments $\Gamma_{k,j}$. As model parameter we fix volatility $\sigma = 0.25$, real-world drift $\mu = 0.1$, interest rate $r = 0.03$ and initial stock price $S_0 = 100$.

The Merton jump diffusion model
The jump diffusion model introduced by [94] adds jumps to the classical Black-Scholes model. The log-returns follow a jump diffusion with volatility σ and added jumps arriving at rate $\lambda > 0$ with normal distributed jump sizes according to $\mathcal{N}(\alpha, \beta^2)$. The stock price under $\mathbb{P}$ is modelled by the SDE

$$\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t + \mathrm{d}J_t$$

for a compound Poisson process J_t with rate λ. The characteristic function of the log-returns $X_t = \log(S_t/S_0)$ under the pricing measure $\mathbb{Q}$ is given by

$$\varphi(z) = exp\left(t\left(ibz - \frac{\sigma^2}{2}z^2 + \lambda\left(e^{iz\alpha - \frac{\beta^2}{2}z^2} - 1\right)\right)\right)$$

with risk-neutral drift

$$b = r - \frac{\sigma^2}{2} - \lambda\left(e^{\alpha + \frac{\beta^2}{2}} - 1\right).$$

In our experiments we calculate the conditional expectations $\Gamma_{k,j}$ using numerical integration and the Fourier transforms of the Chebyshev polynomials along with the characteristic function of X_t. We fix the parameters

$$\sigma = 0.25, \quad \alpha = -0.5, \quad \beta = 0.4, \quad \lambda = 0.4 \quad r = 0.03 \quad \text{and} \quad \mu = 0.1$$

and initial stock price $S_0 = 100$.

The Hull-White model

The Hull-White model as described in Chaper 3.3 of [21] is a short rate model where the rate process $(r_t)_{t\geq 0}$ is a mean reverting Ornstein–Uhlenbeck process described by the SDE

$$\mathrm{d}r_t = (\theta(t) - ar_t)\mathrm{d}t + \sigma \mathrm{d}W_t$$

where the long term mean $\theta(t)$ can be fitted to the term structure of the market and the speed of mean reversion a and the volatility σ are constant. One can write $r_t = \alpha(t) + x_t$ for a deterministic function $\alpha(t)$ given by

$$\alpha(t) = f^M(0,t) + \frac{\sigma^2}{2a^2}(1 - e^{-at})^2$$

where $f^M(0,t)$ is the market forward rate for maturity T obtained from market discount factors $P^M(0,T)$ via

$$f^M(0,T) = -\frac{\partial \ln(P^M(0,T))}{\partial T}.$$

The process $(x_t)_{t\geq 0}$ is modelled by the SDE

$$\mathrm{d}x_t = -ax_t\mathrm{d}t + \sigma \mathrm{d}W_t \quad x_0 = 0$$

and $x_t | x_s = x_0$ is normally distributed with

$$\mathbb{E}[x_t | x_s = x_0] = x_0 e^{-a(t-s)}, \quad \text{and} \quad \mathrm{Var}[x_t | x_s = x_0] = \frac{\sigma^2}{2a}(1 - e^{-2a(t-s)}).$$

Hence, we can use the analytic formula for the generalized moments.

As parameters we fix $a_q = 0.02$ and $\sigma_q = 0.02$ under $\mathbb{Q}$ and $a_p = 0.015$ and $\sigma_p = 0.01$ under $\mathbb{P}$ and we assume a flat forward rate $f^M(0,t) = 0.01$. All parameters are taken from [45].

4.4.2 European option in the Black-Scholes model

In this section, we calculate the expected exposure and the potential future exposure of a European put option in the Black-Scholes model. In this case, we have an analytic formula for the option price V_t at any time point t and we can investigate the accuracy of the dynamic Chebyshev method for exposure calculation. We consider an at-the-money option with strike $K = 100$ and maturity $T = 1$. We fix an interpolation domain in the

log stock price at $[\underline{x}, \overline{x}] = [\log(29), \log(373)]$ based on five times the standard deviation, see Remark 3.

Figure 4.1 shows the resulting exposure profiles and Table 4-A shows the values of the exposures at maturity. The expected exposure under the pricing measure is constant since

$$\begin{aligned} EE_t^{price} &= D(0,t)\mathbb{E}^{\mathbb{Q}}[\max\{V_t(X_t), 0\}] \\ &= D(0,t)\mathbb{E}^{\mathbb{Q}}[D(t,T)\mathbb{E}^{\mathbb{Q}}[g(X_T)|X_t]] \\ &= D(0,T)\mathbb{E}^{\mathbb{Q}}[g(X_T)] = V_0 \end{aligned}$$

for all European options. Under the real-world measure the positive drift of the underlying yields a decreasing exposure for a put option. The PFE increases under both measures since it is mainly driven by the diffusion term in the model.

We observe that the exposure profiles of the dynamic Chebyshev method and the true exposure profile are indistinguishable. In Table 4-B we see the corresponding relative errors for different Chebyshev N's. Here, the notation DC_N refers to the dynamic Chebyshev method of degree N and DC_{N_1,N_2} refers to the dynamic Chebyshev method with domain splitting of degree N_1 and N_2. The error is calculated as the maximum over the simulation period and displayed in relative terms with respect to the initial option price. For $N = 128$ the error is below 10^{-4} for both quantities and under both measures. Already for $N = 64$ nodal points or, if splitting is applied, $N_1 = N_2 = 16$ the error is below 1% in each cases. Figure 4.2 shows the error in the exposure profiles over the option's lifetime of the dynamic Chebyshev method with $N = 128$ and compares it with the least-squares Monte Carlo approach. Whereas the dynamic Chebyshev method is able to produce stable results, the least-squares Monte Carlo method is only able to produces accurate prices at $t = 0$ but adds additional simulation noise over the option's lifetime. For the PFE, the least-squares Monte Carlo method has an relative error of nearly 2%. Note that at time point $t = T$ the option price is equivalent to the payoff and there is no pricing error. All three methods produce the same exposure at maturity.

Table 4-C shows the corresponding runtimes for $M = 50{,}000$ and $M = 150{,}000$ simulation paths of the underlying risk factor. The runtimes of the dynamic Chebyshev method increases approximately linearly in M and is in the same region as the runtime of the analytic pricer. Moreover, the measure under which the risk factors are simulated has no influence on the runtime of the method. The fact that the new numerical method is competitive in comparison to a analytic formula indicates a high efficiency of the approach. In comparison to the least-squares Monte Carlo method, the dynamic Chebyshev method for exposure calculation is as fast or faster and, as already seen, able

to produce more accurate results. For example, the dynamic Chebyshev method with $N_1 = N_2 = 32$ is in all cases more accurate than the least-squares Monte Carlo method but also always faster.

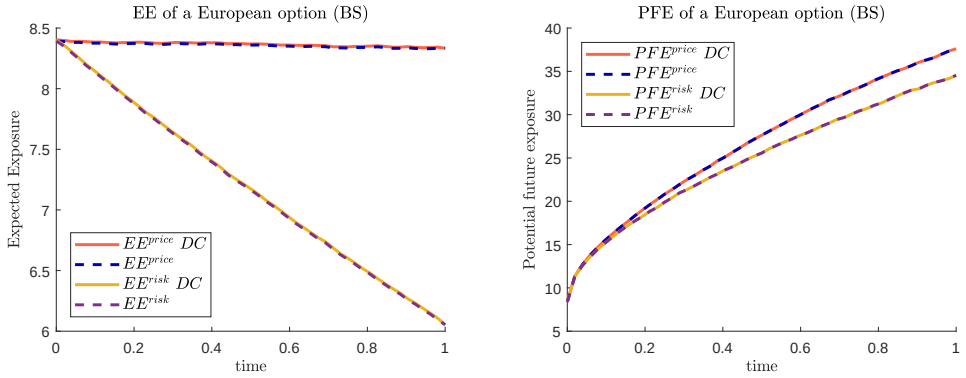

Figure 4.1: Expected exposure (left figure) and potential future exposure (right figure) of a European put option in the Black-Scholes model, calculated under the pricing measure $\mathbb{Q}$ and the real world measure $\mathbb{P}$ using $M = 150{,}000$ simulations.

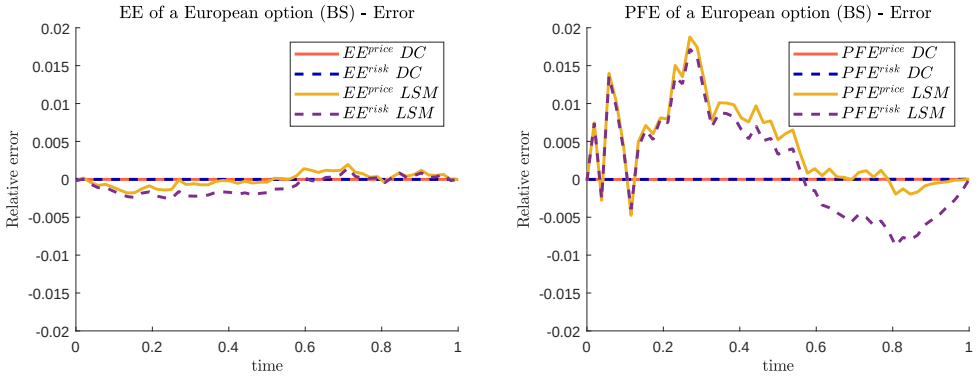

Figure 4.2: Relative error of the expected exposure (left figure) and potential future exposure (right figure) of a European put option in the Black-Scholes model of the dynamic Chebyshev method and the least-squares method. Calculated under the pricing measure $\mathbb{Q}$ and the real world measure $\mathbb{P}$ using $M = 150{,}000$ simulations.

Price	EE_T^{price}	PFE_T^{price}	EE_T^{risk}	PFE_T^{risk}
8.3930	8.3338	37.6163	6.0530	34.5426

Table 4-A: Reference values for option price, EE and PFE of a European put option in the Black-Scholes model using $M = 150{,}000$ simulations.

Overall, the experiment confirms that the new approach is able to produce accurate credit exposure profiles both under the pricing measure $\mathbb{Q}$ and the real-world measure $\mathbb{P}$. Moreover, we have seen that the computations under the real-world measure are as fast as the computation under the pricing measure. For the European put option

	Price	EE^{price}	PFE^{price}	EE^{risk}	PFE^{risk}
DC_{32}	0.0020	0.0095	0.0020	0.0138	0.0020
DC_{64}	0.0011	0.0012	0.0011	0.0017	0.0011
DC_{128}	0.0000	0.0000	0.0000	0.0000	0.0000
$DC_{16,16}$ split	0.0003	0.0003	0.0020	0.0003	0.0020
$DC_{32,32}$ split	0.0000	0.0000	0.0011	0.0000	0.0011
$DC_{64,64}$ split	0.0000	0.0000	0.0000	0.0000	0.0000
LSM	0.0002	0.0019	0.0188	0.0025	0.0172

Table 4-B: Maximal relative error of option price, EE and PFE of a European put option in the Black-Scholes model for $M = 150{,}000$ simulations. Comparison of the dynamic Chebyshev approach for different N with an analytic formula.

	Sim.	DC_{32}	DC_{64}	DC_{128}	$DC_{16,16}$	$DC_{32,32}$	$DC_{64,64}$	LSM	BS
$\mathbb{Q}$	$50k$	0.26s	0.30s	0.38s	0.27s	0.29s	0.36s	0.71s	0.31s
	$150k$	0.90s	1.06s	1.31s	0.89s	0.94s	1.10s	1.22s	1.11s
$\mathbb{P}$	$50k$	0.27s	0.32s	0.40s	0.29s	0.31s	0.37s	0.69s	0.33s
	$150k$	0.92s	1.08s	1.35s	0.90s	0.94s	1.12s	1.18s	1.12s

Table 4-C: Runtimes of the exposure calculation using the dynamic Chebyshev method for different N. Comparison with the analytic Black-Scholes formula.

the runtimes were comparable to using the analytic Black-Scholes formula. Building on these very promising results we will investigate the performance for derivatives which are path-dependent and therefore in general more difficult to price.

4.4.3 Barrier option in the Black-Scholes model

In this section, we calculate the expected exposure and the potential future exposure of a discretely monitored up-and-out barrier call option in the Black-Scholes model. Due to the additional discretely monitored barrier the option becomes path-dependent and there is no longer an analytic solution. In order to compute reference prices we use the COS method provided in the benchmarking project of [135]. The codes are slightly modified in order to fit into our experiments. We consider an option with strike $K = 100$, barrier $B = 130$ and maturity $T = 1$. We fix an interpolation domain in the log stock price at $[\underline{x}, \overline{x}] = [\log(29), \log(B)]$ based on six times the standard deviation for the lower bound, see Remark 3. We assume that the barrier option is discretely monitored and the monitoring dates coincide with dates for the exposure calculation.

Figure 4.3 shows the resulting exposure profiles and Table 4-D shows the values of the exposures at maturity. The expected exposure under the pricing measure is constant over time which can be justified by the same arguments as for the European option. Under the real-world measure, the expected exposure increases slightly and for the PFE we observe also an increase. In comparison to the European option we see a slower increase in the beginning and a faster increase close to maturity. Here, we observe the effect of the barrier which means that an increase in the stock price also leads to a higher risk of triggering the barrier and a zero exposure afterwards. This effect is stronger for a longer time to maturity. As for the European option, the exposure profiles of the dynamic Chebyshev method and the exposure profile of the full re-evaluation are indistinguishable. Figure 4.4 shows the error of the exposure profiles computed with the dynamic Chebyshev method for $N = 64$ and with the least squares Monte Carlo approach. For the least squares Monte Carlo approach we added an additional basis function compared to the European version to better fit the barrier. In Table 4-E we see the corresponding relative errors for different Chebyshev N's and the error of the least-squares Monte Carlo method with 150,000 pricing paths. The error is calculated as the maximum over the simulation period and is displayed in relative terms. For $N = 64$ the error is below 10^{-4} for both quantities. Figure 4.4 shows the relative error of the dynamic Chebyshev method and the least-squares Monte Carlo method over the option's lifetime. We can again observe that a strong fluctuation in the error of the least-squares Monte Carlo method and a stable and very low error for the dynamic Chebyshev method.

Table 4-F shows the corresponding runtimes for $M = 50{,}000$ and $M = 150{,}000$ simulation paths of the underlying risk factor. We observe that the dynamic Chebyshev method is more than 100 times faster than doing a full-revaluation approach using an already competitive pricer. Compared to the least-squares Monte Carlo method the dynamic Chebyshev method produces more accurate estimates while also being faster. The barrier yields a smaller interpolation domain and therefore a lower number of interpolation nodes for the dynamic Chebyshev method. On the other side, the least-squares Monte Carlo method does not profit from the barrier but we had to add an additional basis function to achieve a satisfying accuracy.

Price	EE_T^{price}	PFE_T^{price}	EE_T^{risk}	PFE_T^{risk}
2.6453	2.6678	21.3718	3.0641	22.9297

Table 4-D: Reference values for option price, EE and PFE of a barrier call option in the Black-Scholes model using $M = 150{,}000$ simulations.

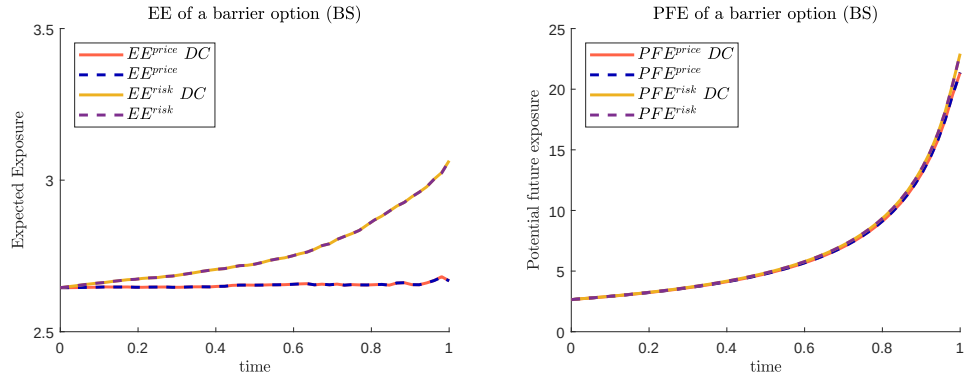

Figure 4.3: Expected exposure (left figure) and potential future exposure (right figure) of a barrier call option in the Black-Scholes model, calculated under the pricing measure $\mathbb{Q}$ and the real world measure $\mathbb{P}$ using $M = 150{,}000$ simulations.

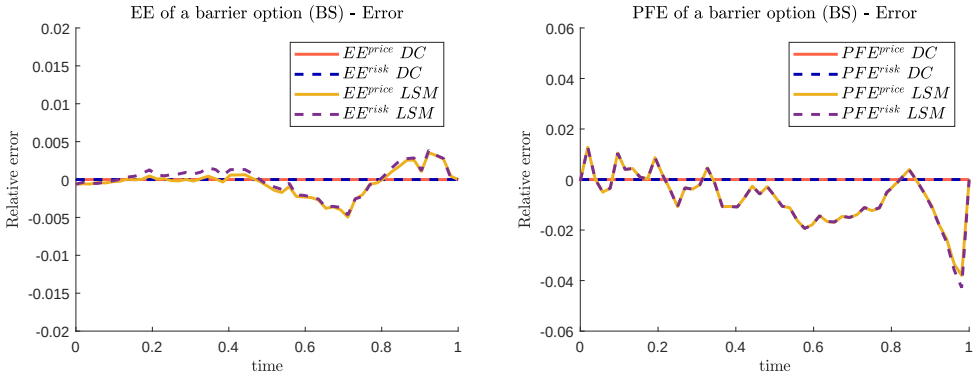

Figure 4.4: Relative error of the expected exposure (left figure) and potential future exposure (right figure) of a barrier up-and-out call option in the Black-Scholes model of the dynamic Chebyshev method and the least-squares method. Calculated under the pricing measure $\mathbb{Q}$ and the real world measure $\mathbb{P}$ using $M = 150{,}000$ simulations.

	Price	EE^{price}	PFE^{price}	EE^{risk}	PFE^{risk}
DC_{16}	0.0064	0.0074	0.0110	0.0069	0.0109
DC_{32}	0.0000	0.0006	0.0007	0.0004	0.0004
DC_{64}	0.0000	0.0000	0.0000	0.0000	0.0000
LSM	0.0006	0.0050	0.0380	0.0047	0.0428

Table 4-E: Maximal relative error of option price, EE and PFE of a barrier call option in the Black-Scholes model for $M = 150{,}000$ simulations. Comparison of the dynamic Chebyshev approach for different N and the LSM approach with a full re-evaluation.

	Sim.	DC_{16}	DC_{32}	DC_{64}	LSM	Full re-eval
$\mathbb{Q}$	$50k$	0.31s	0.31s	0.38s	1.77s	52.0s
	$150k$	0.98s	1.05s	1.23s	2.76s	144.8s
$\mathbb{P}$	$50k$	0.27s	0.29s	0.33s	1.62s	47.3s
	$150k$	0.92s	0.97s	1.13s	2.58s	140.3s

Table 4-F: Runtimes of the exposure calculation of a barrier call option using the dynamic Chebyshev method for different N. Comparison with a full re-evaluation using the COS method.

4.4.4 Bermudan option in the Merton jump-diffusion model

Here, we consider a Bermudan put option in the Merton jump-diffusion model. The early-exercise feature makes the option path-dependent and the jump component of the stock price model poses an additional computational challenge. Similar to the barrier option we can again use the COS method provided in the benchmarking project of [135] for the calculation of reference prices. The codes are slightly modified in order to fit into our experiments. We consider an option with strike $K = 100$, maturity $T = 1$ and we assume that the dates used for the exposure calculation are also the exercise dates of the option. We fix an interpolation domain in the log stock price at $[\underline{x}, \overline{x}] = [\log(0.2), \log(400)]$.

Figure 4.5 shows the resulting exposure profiles and Table 4-G shows the values of the exposures at maturity. We observe a decreasing expected exposure under both measures due to the early exercise feature of the option. For the PFE we observe an increasing exposure in the beginning resulting from the diffusion term and a decreasing exposure afterwards. As for the European option, the exposure profiles of the dynamic Chebyshev method and the exposure profile of the full re-evaluation are indistinguishable. In Table 4-H we see the corresponding relative errors for different Chebyshev N's. Due to the early-exercise feature, a higher number of nodal points is required for a similar accuracy of the option prices. Moreover, an exact estimation of the exercise barrier is critical for a correct estimation of the exposure. A miscalculation of the exercise barrier at time point t_u does not only influence the exposure EE_{t_u} but also the exposure at all future time points. For example, an exercise barrier that is too low means that the option is exercised for too many paths and the exposure at future time points is underestimated. Figure 4.6 shows the relative error of the dynamic Chebyshev method and the least-squares Monte Carlo method over the option's lifetime. We can see that the least-squares Monte Carlo method struggles to provide accurate estimations in the tail and thus an accurate value for the PFE. In contrast, the dynamic Chebyshev method produces stable and accurate results for both quantities.

Table 4-I shows the corresponding runtimes for $M = 50{,}000$ and $M = 150{,}000$ simulation paths of the underlying risk factor. We observe that the dynamic Chebyshev method is more than 100 times faster than doing a full-revaluation. The comparison to the least-squares Monte Carlo method shows again that the dynamic Chebyshev method is able to deliver both, more accurate results and faster runtimes.

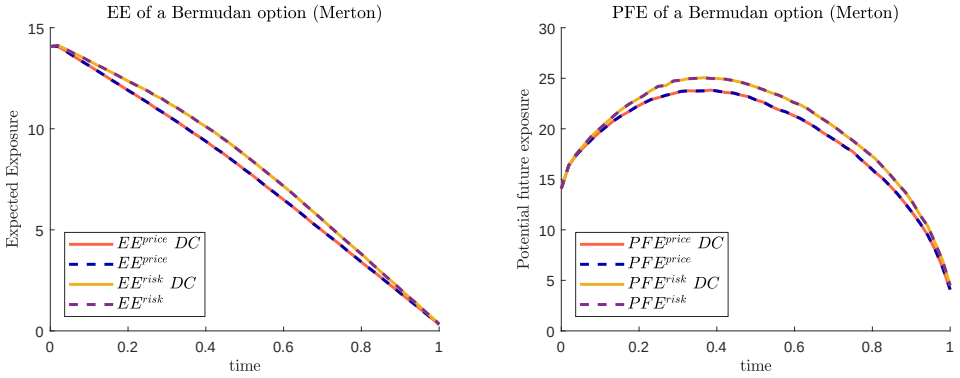

Figure 4.5: Expected exposure (left figure) and potential future exposure (right figure) of a Bermudan put option in the Merton jump-diffusion model, calculated under the pricing measure $\mathbb{Q}$ and the real world measure $\mathbb{P}$ using $M = 150{,}000$ simulations.

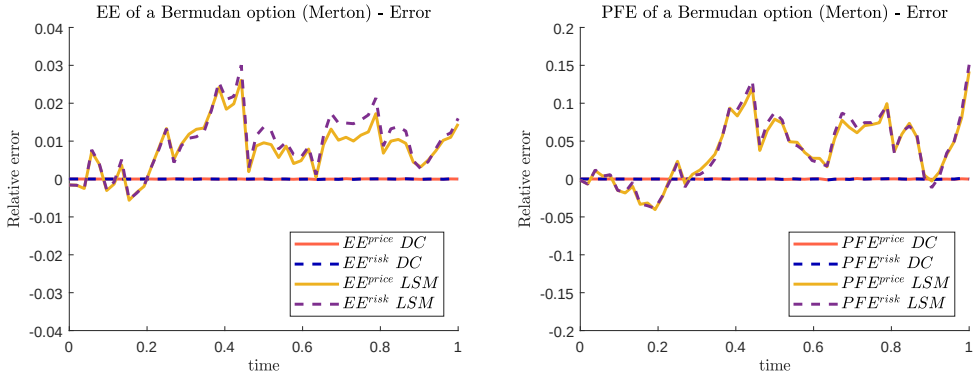

Figure 4.6: Relative error w.r.t. to the initial stock price of the expected exposure (left figure) and potential future exposure (right figure) of a Bermudan put option in the Merton jump-diffusion model of the dynamic Chebyshev method and the least-squares method. Calculated under the pricing measure $\mathbb{Q}$ and the real world measure $\mathbb{P}$ using $M = 150{,}000$ simulations.

Price	EE_T^{price}	PFE_T^{price}	EE_T^{risk}	PFE_T^{risk}
14.0739	0.3160	4.0892	0.3423	4.5004

Table 4-G: Reference values for option price, EE and PFE of a Bermudan put option in the Merton jump-diffusion model using $M = 150{,}000$ simulations.

	Price	EE^{price}	PFE^{price}	EE^{risk}	PFE^{risk}
DC_{128}	0.0001	0.0028	0.0366	0.0041	0.0508
DC_{256}	0.0000	0.0005	0.0045	0.0007	0.0044
DC_{512}	0.0000	0.0001	0.0009	0.0001	0.0013
$DC_{64,64}$	0.0000	0.0018	0.0102	0.0027	0.0130
$DC_{128,128}$	0.0000	0.0003	0.0013	0.0005	0.0019
$DC_{256,256}$	0.0000	0.0001	0.0009	0.0001	0.0006
LSM	0.0016	0.0261	0.1420	0.0300	0.1508

Table 4-H: Maximal relative error w.r.t. to the initial stock price of option price, EE and PFE of a Bermudan put option in the Merton jump-diffusion model for $M = 150{,}000$ simulations. Comparison of the dynamic Chebyshev approach for different N and the LSM approach with a full re-evaluation.

	Sim.	DC_{128}	DC_{256}	DC_{512}	$DC_{64,64}$	$DC_{128,128}$	$DC_{256,256}$	LSM	full
$\mathbb{Q}$	$50k$	0.77s	0.95s	1.65s	0.71s	0.86s	1.28s	1.99s	106s
	$150k$	2.54s	3.11s	4.33s	2.41s	2.83s	3.38s	3.79s	317s
$\mathbb{P}$	$50k$	0.74s	0.93s	1.66s	0.73s	0.86s	1.31s	1.98s	108s
	$150k$	2.38s	2.91s	4.37s	2.26s	2.53s	3.21s	3.63s	320s

Table 4-I: Runtimes of the exposure calculation of a Bermudan put option using the dynamic Chebyshev method for different N. Comparison with a full re-evaluation using the COS method.

4.5 Credit exposure of Bermudan swaptions

In this section, we describe how the dynamic Chebyshev method can be used to price Bermudan swaptions. A swaption is an interest rate derivative that allows the option holder to enter into a swap contract, i.e. a swaption is an option on an interest rate swap. Similar to equity products, a European swaption allows the holder to exercise his right at maturity and a Bermudan swaption allows the holder to exercise the option at a set of pre-defined exercise dates. In the first part of the section we will give a brief overview on swaps and swaptions. For more details we refer to [21]. In the second part of the section we discuss how the pricing of interest rate derivatives falls into the scope of the dynamic Chebyshev method.

4.5.1 Interest rate derivatives: Swaps and swaptions

We start with an introduction to swaps and swaptions, two of the most important interest rate derivatives. This section is based on [21, Chapter 1]. Assume we have a bank account $B(t)$ that pays the risk-free interest rate with

$$\mathrm{d}B(t) = r(t)B(t)\mathrm{d}t \quad \text{and} \quad B(0) = 1$$

for a short rate process $(r_t)_{t\geq 0}$. The differential equation has the solution

$$B(t) = \exp\left(\int_0^t r_s \mathrm{d}s\right).$$

Among the most common models for the stochastic short rate are the models of [134], [71] and [16]. Using the bank account we can define (stochastic) discount factors

$$D(t,T) = \frac{B(t)}{B(T)} = \exp\left(-\int_t^T r_s \mathrm{d}s\right)$$

which connect cash-flows at time T to cash-flows at time t. For stochastic interest rates, the discount factor becomes a random variable. The expectation of $D(t,T)$ is the value $P(t,T)$ at time t of a zero coupon that pays 1 at maturity T.

Next, we introduce an interest rate swap. A *swap* is an interest rate derivative that exchanges payments of a fixed leg and a floating leg at a set of payment days $T_1, \ldots, T_{n+1}$. The payments of the fixed leg at T_i are defined as $N\tau_i K$ where N is the notional amount of the contract, K is a fixed interest rate and $\tau_i = T_i - T_{i-1}$ is the time interval between payment days with starting point T_0. The payments of the floating leg at T_i are given by $N\tau_i L(T_{i-1}, T_i)$ where $L(T_{i-1}, T_i)$ is the floating rate fixed at time point T_{i-1}.

A payer swap means that the fixed leg is paid to get the floating rate and vice versa the receiver swap means that the fixed leg is received. The discounted payoff of a swap at any time $t \leq T_0$ is given by

$$\sum_{i=1}^{n} D(t,T_i) N\tau_i \delta(L(T_{i-1},T_i) - K)$$

with $\delta = 1$ for a payer swap and $\delta = -1$ for a receiver swap. From [21] we obtain for the value of a receiver swap (i.e. the expected discounted payoff) at time t

$$Swap^{rec}(t) = N\sum_{i=1}^{n} \tau_i P(t,T_i)(K - F(t,T_{i-1},T_i))$$

where $P(t,T_i)$ is the value of a zero coupon bond and $F(t,T_{i-1},T_i)$ is the forward rate given by

$$F(t,T_{i-1},T_i)=\frac{1}{T_i-T_{i-1}}\Big(\frac{P(t,T_{i-1})}{P(t,T_i)}-1\Big)$$

which is the expectation of $L(T_{i-1},T_i)$. We can rewrite the value of the receiver swap as

$$Swap^{rec}(t)=\underbrace{-NP(t,T_0)+NP(t,T_{n+1})}_{\text{floating leg}}+\underbrace{N\sum_{i=1}^{n}\tau_i KP(t,T_i)}_{\text{fixed leg}}.$$

The interest K for which the value of the swap is zero is called the forward swap rate and is given by

$$S_{T_0,T_{n+1}}(t)=\frac{P(t,T_0)-P(t,T_{n+1})}{\sum_{i=1}\tau_i P(t,T_i)}. \tag{4.7}$$

The payoff of a receiver swaption is then the value of a receiver swap if it is positive. Otherwise, the option holder would not use his exercise right. From [21] we obtain for a receiver option with maturity T_0 the discounted payoff

$$ND(t,T_0)\Big(\sum_{i=1}^{n}P(T_0,T_i)\tau_i(K-F(T_0,T_{i-1},T_i))\Big)^+.$$

Using the forward swap rate we can rewrite the payoff as

$$ND(0,T_0)(K-S_{T_0,T_{n+1}}(T_0))^+\sum_{i=1}^{n}P(T_0,T_i)\tau_i.$$

The fixed rate K of the swap becomes the strike of the swaption. We define an option to be at-the-money if $K=S_{T_0,T_{n+1}}(0)$. A Bermudan swaption means that the holder can enter into the swap at multiple time points. For simplicity we will always assume that the exercise dates of a swaption with payment days $T_1,\dots,T_n$ of the underlying swap are $T_0,\dots,T_n$ if not stated otherwise.

4.5.2 Pricing of Bermudan swaptions

Similar to a Bermudan equity option we can price a Bermudan swaption via backward induction. See for example [45]. We assume that the short rate process $(r_t)_{t\geq 0}$ is driven by a risk factor $(x_t)_{t\geq 0}$ such that $r_t=r(t,x_t)$. For example in the Hull-White model we can write $r(t)=\alpha(t)+x(t)$ for a deterministic function α that fits the term structure of the market. The payoff function g of a Bermudan receiver swaption with exercise dates

$T_0, \dots, T_n$ can be written as

$$g(T_i, x) = N(K - S_{T_i,T_{n+1}}(T_i, x))^+ \sum_{k=i+1}^{n} P(T_i, T_k)\tau_k$$

where

$$S_{T_0,T_{n+1}}(t, x) = \frac{P(t, T_0, x) - P(t, T_{n+1}, x)}{\sum_{i=1} \tau_i P(t, T_i, x)}$$

and

$$P(t, T, x) = \mathbb{E}\big[D(t, T)\big|x_t = x\big] = \mathbb{E}\Big[\exp\Big(\int_t^T r(s, x_s)\mathrm{d}s\Big)\Big|x_t = x\Big].$$

Assume a time grid t_u, $u = 0, \dots, n_T$ with $t_0 = 0$, $t_{n_T} = T_n$ that contains all exercise dates $T_0, \dots, T_n$. Then we can write the pricing problem of a Bermudan swaption in the form of a dynamic programming problem

$$\begin{aligned}
V_{T_n}(x) &= g(T_n, x) = N(K - S_{T_n,T_{n+1}}(T_n, x))^+ P(T_n, T_{n+1})\tau_k \\
V_{t_u}(x) &= \begin{cases} \max\big\{g(t_u, x), CV_{t_u}(x)\big\}, & \text{if } t_u \in \{T_0, \dots, T_n\} \\ CV_{t_u}(x), & \text{otherwise} \end{cases} \\
\text{with} \quad CV_{t_u}(x) &= \mathbb{E}\big[D(t_u, t_{u+1})V_{t_{u+1}}(x_{t_{u+1}})\big|x_{t_u} = x\big],
\end{aligned}$$

see [45]. Here, the notation CV_{t_u} refers to the continuation value of the swaption at time point t_u. This backward induction fits into the scope of the dynamic Chebyshev method. Assume we have a Chebyshev approximation $\sum_j c_j^{u+1} p_j$ of the value function $V_{t_{u+1}}$ and we want to interpolate V_{t_u}. Let x_k, $k = 0, \dots, N$ be the Chebyshev nodal points. Then we obtain for the continuation value

$$\begin{aligned}
CV_{t_u}(x_k) &\approx \mathbb{E}\big[D(t_u, t_{u+1}) \sum_j c_j^{u+1} p_j(x_{t_{u+1}})\big|x_{t_u} = x\big] \\
&= \sum_j c_j^{u+1} \mathbb{E}\big[D(t_u, t_{u+1}) p_j(x_{t_{u+1}})\big|x_{t_u} = x\big]
\end{aligned}$$

and $V_{t_u}(x_k) = \max\{g(t_u, x_k), CV_{t_u}(x_k)\}$ if t_u is an exercise date.

For this pricing problem, two additional challenges arise in comparison to a Bermudan equity option. First, the discount factor $D(t_u, t_{u+1})$ is stochastic and second, the discount factor depends on the risk factor $x_{t_{u+1}}$. We can either calculate discounted conditional expectations or we approximate the discount factor. Assume that the time stepping

$\Delta t = t_{u+1} - t_u$ is small, then we can write

$$\begin{aligned}
&\mathbb{E}^{\mathbb{Q}}\left[D(t_u, t_{u+1})p_j(x_{t_{u+1}})|x_{t_u} = x_k\right] \\
&\quad = \mathbb{E}^{\mathbb{Q}}\left[\exp\Big(-\int_{t_u}^{t_{u+1}} r(s, x_s)\mathrm{d}s\Big)p_j(x_{t_{u+1}})|x_{t_u} = x_k\right] \\
&\quad \approx \mathbb{E}^{\mathbb{Q}}\left[\exp\big(-\Delta t\, r(t_u, x_{t_u})\big)p_j(x_{t_{u+1}})|x_{t_u} = x_k\right] \\
&\quad = \exp\big(-\Delta t\, r(t_u, x_k)\big)\mathbb{E}^{\mathbb{Q}}\left[p_j(x_{t_{u+1}})|x_{t_u} = x_k\right],
\end{aligned}$$

where we assumed that the discount factor is constant on a small interval. Hence, the pre-computation of the conditional expectations is the same as for an equity option.

The second challenge is that the payoff function $g(t_u, x_k)$ is more complex and requires the computation of zero coupon bond prices $P(t_u, T_j, x_k)$. If these prices are available in closed form the implementation of the dynamic Chebyshev method becomes straightforward. This is for example the case in the Hull-White model. Otherwise, we need to apply the dynamic Chebyshev method also to the discount factors $P(0, T_j, x_k)$ for $j = 0, \ldots, n+1$. In general, the discount factor $P(0, T_j)$ can be seen as a European option in the short rate model with constant payoff 1 and maturity T_j. We can price them with the dynamic Chebyshev method using the same time grid t_u and the same conditional expectations as for the swaption.

4.5.3 Bermudan swaption in the Hull-White model

Here, we consider a Bermudan receiver swaption in the Hull-White model. Similar to the equity case, the early-exercise feature makes the option path-dependent and poses an additional computational challenge. Additionally, the payoff function is more complex and requires the pricing of a reciever swap. The Hull-White model is briefly described in Section 4.4.1. We consider a swaption with strike $K = 0.01094$ and maturity $T = 5$ which can be exercised yearly starting at $T_1 = 1$ and the swap terminates at $T + 1$ and payments are also exchanged on a yearly basis. A detailed description of the pricing problem can be found in [45]. From this paper we also obtain a reference price of $V_0 = 5.463$. We assume that the swaption is cash-settled and hence there is no credit exposure after the option is exercised. We fix an interpolation domain in x_t based on five times (for $N = 32, 64, 128$) or six times (for $N = 256$, used as reference price) the standard deviation of the underlying, see Remark 3.

Figure 4.7 shows the resulting exposure profiles and Table 4-J shows the price and the values of the exposures at maturity. The price of the dynamic Chebyshev method is the same as the reference price V_0. Similarly to the equity Bermudan option, the expected exposure decreases over time and the PFE increases first and then decreases. In contrast

to the equity option we observe big jumps in the exposure profiles of the swaption. In our example, the swaption is only exerciseable once per year and the exposure jumps downwards at these days. In contrast, the equity option was weekly exercisable and the exercise dates coincided with the exposure calculation dates. Thus, we did not observe any jumps in the exposure profiles. In Table 4-K we see the corresponding relative errors w.r.t. to the swaption price for different Chebyshev N's. Here we used a dynamic Chebyshev method with higher accuracy to compute reference prices. As for an equity Bermudan option, an exact estimation of the exercise barrier is critical for a correct estimation of the exposure and leads to a higher N. However, since the volatility is lower, the interpolation domain is smaller and we need less nodes than for the Bermudan equity option. Figure 4.8 shows the relative error of the dynamic Chebyshev method and the least-squares Monte Carlo method over the option's lifetime.

Table 4-L shows the corresponding runtimes for $M = 50{,}000$ and $M = 150{,}000$ simulation paths of the underlying risk factor. Overall the runtime are slightly slower than for the equity products since the maturity is with five years much longer. The comparison of the dynamic Chebyshev method with the least-squares Monte Carlo method reveals again a significantly higher efficiency. This is especially the case when it comes to the computation of the tail measure PFE.

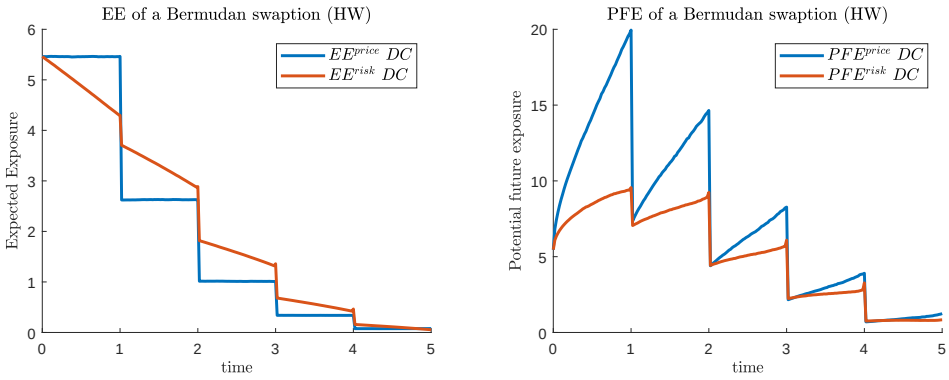

Figure 4.7: Expected exposure (left figure) and potential future exposure (right figure) of a Bermudan swaption in the Hull-White model, calculated under the pricing measure $\mathbb{Q}$ and the real world measure $\mathbb{P}$ using $M = 150{,}000$ simulations.

Price	EE_T^{price}	PFE_T^{price}	EE_T^{risk}	PFE_T^{risk}
5.4628	0.0771	1.2489	0.0540	0.8348

Table 4-J: Reference values for option price, EE and PFE of a Bermudan receiver swaption in the Hull-White model using $M = 150{,}000$ simulations.

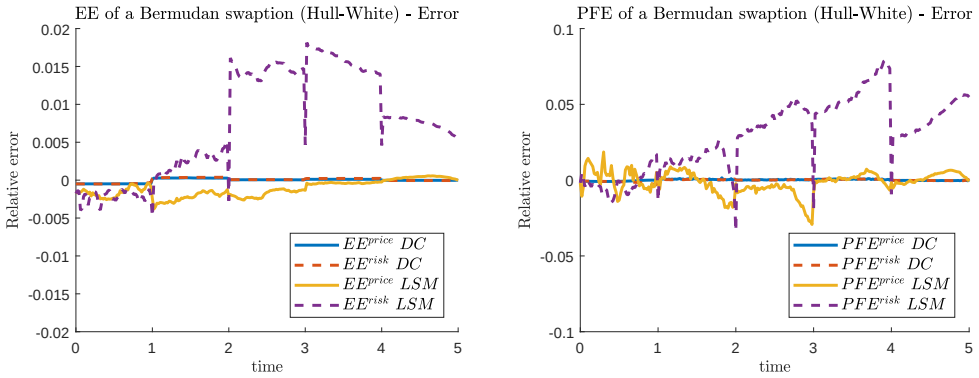

Figure 4.8: Relative error w.r.t. to the initial swaption price of the expected exposure (left figure) and potential future exposure (right figure) of a Bermudan receiver swaption in the Hull-White model of the dynamic Chebyshev method and the least-squares method. Calculated under the pricing measure $\mathbb{Q}$ and the real world measure $\mathbb{P}$ using $M = 150{,}000$ simulations.

	Price	EE^{price}	PFE^{price}	EE^{risk}	PFE^{risk}
DC_{32}	0.0148	0.0165	0.0444	0.0245	0.0351
DC_{64}	0.0026	0.0026	0.0069	0.0031	0.0166
DC_{128}	0.0005	0.0005	0.0016	0.0005	0.0032
$DC_{16,16}$	0.0010	0.0022	0.0323	0.0034	0.0563
$DC_{32,32}$	0.0005	0.0005	0.0015	0.0006	0.0026
$DC_{64,64}$	0.0002	0.0002	0.0016	0.0002	0.0014
LSM	0.0069	0.0277	0.1463	0.0331	0.1533

Table 4-K: Maximal relative error w.r.t. to the initial swaption price, EE and PFE of a Bermudan receiver swapption in the Hull-White model for $M = 150{,}000$ simulations. Comparison of the dynamic Chebyshev approach for different N and the LSM approach with a full re-evaluation.

4.5.4 Summary of the experiments

In this section, we analysed the dynamic Chebyshev method for credit exposure calculation numerically. In Chapter 3, we have validated the method for the pricing of options in different asset models. The experiments of this section show that the method is moreover well suited for credit exposure calculation of path-dependent options such as Bermudan and barrier equity options and Bermudan swaptions. Our examples show that the method can be applied to different models which require different numerical techniques for the moment calculation.

	Sim.	DC_{32}	DC_{64}	DC_{128}	$DC_{16,16}$	$DC_{32,32}$	$DC_{64,64}$	LSM	DC ref.
$\mathbb{Q}$	$50k$	1.11s	1.26s	1.65s	1.43s	1.53s	1.96s	8.00s	2.49s
	$150k$	4.09s	5.12s	6.66s	4.41s	4.74s	5.76s	13.56s	9.54s
$\mathbb{P}$	$50k$	1.20s	1.40s	1.92s	1.41s	1.44s	1.89s	8.22s	3.11s
	$150k$	3.96s	4.96s	6.34s	4.12s	4.44s	5.48s	13.41s	9.37s

Table 4-L: Runtimes of the exposure calculation of a Bermudan swaption using the dynamic Chebyshev method for different N and the least-squares Monte Carlo method. As a reference method we use the dynamic Chebyshev method with $N = 256$.

The experiments show that the dynamic Chebyshev method is able to produce stable and accurate results even for tail measures like the potential future exposure. It can handle the measure change from the pricing measure to the real-world measure without an additional computational effort and is therefore suited for the credit exposure calculation in pricing and risk management.

The comparison with the popular least-squares Monte Carlo method approach revealed the efficiency of the method in terms of accuracy vs. runtime and we have seen that this is especially the case for the computation of the PFE. The least-squares Monte Carlo method was not able to produce accurate prices in the tail for early-exercise options. Moreover, pricing methods based on Monte Carlo simulation and regression add additional simulation noise to the exposure calculation which is omitted in the new approach. Other methods that can improve the accuracy in comparison to the LSM are for example the stochastic grid bundling method. However, this method is more than a factor of two times slower than the LSM for a Bermudan swaption in the Hull-White model as shown in [45] and will therefore also be slower than the dynamic Chebyshev method.

Moreover, the experiments have shown that introducing an additional splitting in the dynamic Chebyshev method reduces the number of nodal points and can improve the efficiency of the exposure calculation further. This is mainly interesting for large interpolation domains and early-exercise options.

4.6 Empirical investigation of exposure profiles

So far, we have seen that the dynamic Chebyshev method is able to calculate accurate exposure profiles for equity and interest rate derivatives. In this section, we want to investigate the exposure profiles of path-dependent equity options in more detail. We empirically analyse the effects of the model choice on the exposure profiles as well as

the effect of a barrier and an early-exercise feature on the profiles in comparison to a European option. Due to the computational complexity of calculating credit exposures for exotic trades, choosing a simpler model and replacing a path-dependent option by a European option are two ad-hoc simplifications that are popular among practitioners to speed-up the calculations.

In our analysis, we investigate the effect of the model choice on the exposure profile by comparing the more complex Merton jump-diffusion model and the constant elasticity of variance (CEV) model with the simpler Black-Scholes model. Moreover, we compare the exposure profiles of path-dependent barrier and Bermudan options with the one of European vanilla options.

For a meaningful comparison we need to make sure that the models are calibrated to the same instruments. From [25] we obtain the following parameter for the Merton jump-diffusion model

$$\sigma = 0.1936, \quad \alpha = -0.2, \quad \beta = 0.2194, \quad \lambda = 0.2935$$

and a risk-free interest rate of $r = 0.0016$ and in the CEV model we obtain $\sigma = 0.25$ and $\beta = 1.96$. For a maturity of $T = 1$ the corresponding implied volatility in the Black-Scholes model is given by $\sigma_{BS} = 0.2335$. As drift, we fix $\mu_{MRT} = 0.1$ in the Merton model and choose drifts $\mu_{CEV} = 0.0464$ and $\mu_{BS} = 0.0477$ in the other two models in order to match this drift.

4.6.1 Credit exposure of Barrier options

We start with the investigation of the expected exposure and the potential future exposure of a barrier option in the three different equity models. We consider a call option with up-and-out barrier $B = 150$ and strike $K = 100$ in the three stock price models with $S_0 = 100$. We fix an interpolation domain $\mathcal{X} = [\underline{x}, \overline{x}]$ with $\underline{x} = \log(10)$ and $\overline{x} = \log(B)$. Moreover, we fix $N = 64$ Chebyshev points. We calculate both, the expected exposure and the potential future exposure under the real-world measure $\mathbb{P}$ using 150,000 simulation paths.

The left plot in Figure 4.9 shows the expected exposure (EE) in the three models over the option's lifetime and for a fixed number of simulation paths. The corresponding option prices and exposure values are displayed in Table 4-M. While the prices of the European call option differ only slightly due to a small calibration error, we observe larger differences for the prices of the barrier options. These pricing differences reflected the different (tail) distributions of the stock price in the three models.

For a barrier option two different effects have an influence on the expected exposure. On the one hand, the option is an at-the-money call option and due to the positive drift of the underlying (under the real-world measure) we expect that over the option's lifetime more and more paths will be in the money which leads to an increase of the expected exposure. On the other hand, when the underlying increases there is a higher risk that the underlying reaches the barrier and the option is knocked out. Thus the expected exposure could decay. Depending on the model properties and the relation of S_0 and B one of the effects might dominate the other. In our experiment, the expected exposure increases in all three models and the profiles are approximately parallel. They only differ in value due to the pricing differences at $t = 0$.

The right plot in Figure 4.9 shows the corresponding potential future exposure (PFE) at the $97,5\%$ level in the three models. The potential future exposure increases over time due to the positive drift and the diffusion term in all three models. The PFE converges towards the maximal possible exercise value $B - K$ which is 50. In the Merton model, the PFE increases faster than in the other two models due to the additional jumps which make larger stock price movements more likely.

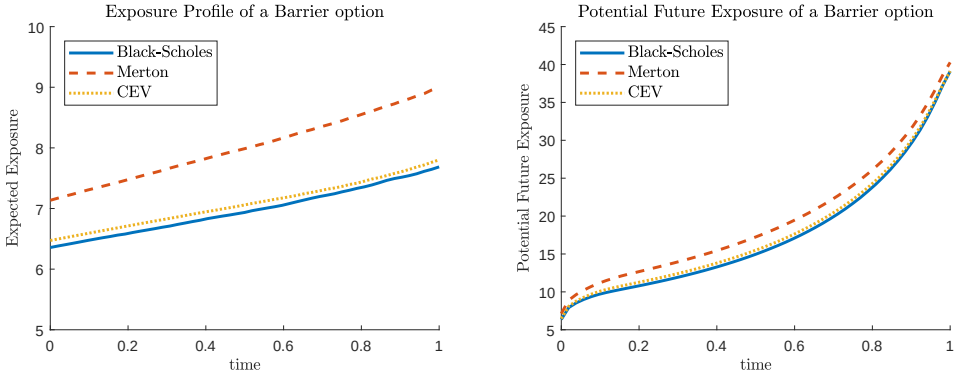

Figure 4.9: Expected exposure (EE) and potential future exposure (PFE) for a barrier option in the Black-Scholes model, the Merton model and the CEV model with maturity $T = 1$ and $N_{sim} = 150{,}000$ under the real world measure.

Figure 4.10 shows the comparison of the exposure profiles of barrier options with the exposure profiles of European options in the three models. The expected exposure and the potential future exposure in all three models is smaller for the barrier option due to the knock-out feature. For both option types the expected exposure behaves relatively similar and moves more or less parallel. For the potential future exposure we observe different behaviours over time. For the European case the diffusion term results in an increasing exposure over time. This effect is less strong in the barrier case due to the risk of a knock-out. Closer to maturity the risk of reaching the barrier becomes smaller and the PFE increases faster than its European counterpart. We conclude that replacing

barrier by European options yields an overestimation of the credit exposure. This is an indication of a very conservative practice. In the potential future exposure case the difference is substantial. Here, the dynamic Chebyshev method yields more precise result which could lead to lower capital requirements.

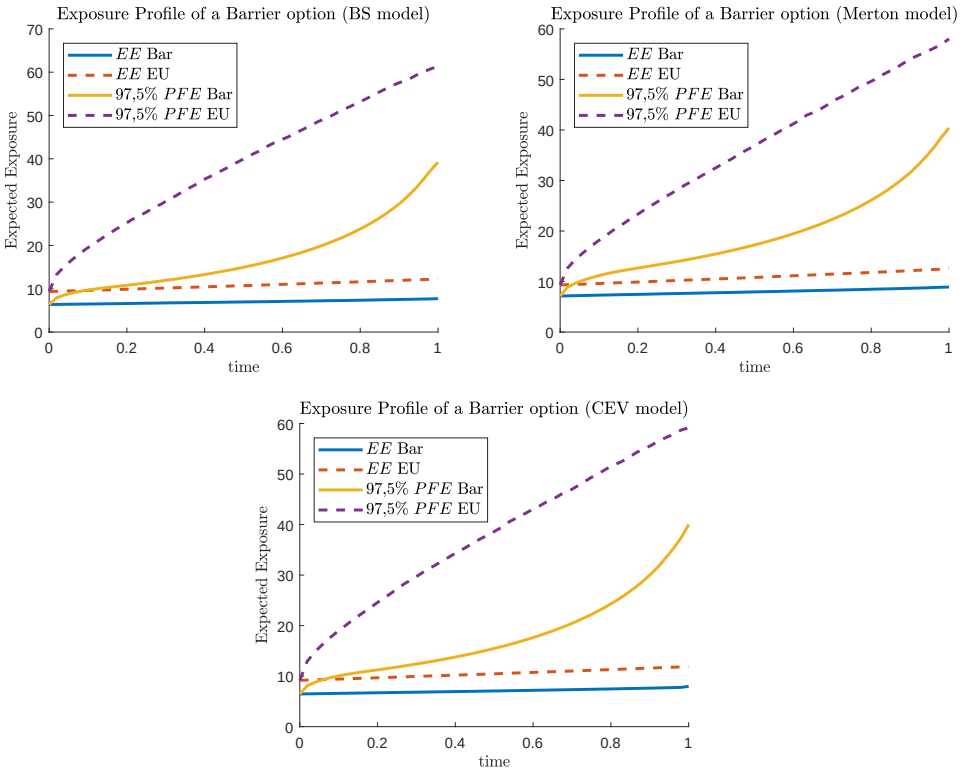

Figure 4.10: Expected exposure (EE) and potential future exposure (PFE) profiles for a barrier option and a European call option with maturity $T = 1$ in the Black-Scholes model, the Merton model and the CEV model with $N_{sim} = 150{,}000$ under the real world measure.

		Price	EE at T	PFE at T
BS	European call	9.37	12.22	61.37
	Barrier call	6.36	7.71	39.21
Merton	European call	9.32	12.61	58.03
	Barrier call	7.13	8.91	40.38
CEV	European call	9.12	11.87	59.13
	Barrier call	6.45	7.97	39.98

Table 4-M: Option price, expected exposure and potential future exposure of a European call and a Barrier up-and-out call option in the Black-Scholes (BS) model, the Merton model and the CEV model.

4.6.2 Credit exposure of Bermudan options

In this section, we repeat the empirical analysis of the credit exposure of barrier options in the previous section for Bermudan options. We consider a Bermudan put option with strike $K = 100$ and 52 exercise rights per year and a European option with the same strike. We fix the interpolation domain $\mathcal{X} = [\underline{x}, \overline{x}]$ with $\underline{x} = \log(0.2)$ and $\overline{x} = \log(450)$. Moreover, we fix $N = 256$ Chebyshev points. We calculate both, the expected exposure and the potential future exposure under the real-world measure $\mathbb{P}$ using 150,000 simulation paths.

For the European put we expect a decreasing exposure due to the positive drift and the negative delta of the option. In contrast, the PFE should increase because of the diffusion term and the increasing volatility. For the Bermudan put, we expect that the early-exercise feature yields an even faster decline of the expected exposure. We know that if the option is exercised the exposure becomes zero. As we approach maturity, the optimal exercise boundary of the Bermudan put converges towards the option's strike. Therefore, the likeliness that the option is exercised early if the underlying is below the strike increases over the option's lifetime. Similarly, the likeliness that the option ends in-the-money if the underling is above the strike decreases over time. Both effects yield a decreasing exposure over the lifetime of the option. For the potential future exposure we expect to see a different effect. The diffusion term should lead to an increasing PFE whereas the early-exercise feature still leads to a decreasing PFE over time.

Figure 4.11 shows the expected exposure (left plot) and the potential future exposure (right plot) for the European put. The corresponding option values and exposures for both, the European and the Bermudan put are displayed in Table 4-N. While the expected exposure is almost identical over the option's lifetime, the potential future exposure differs between the models. We observe a steeper increase of the PFE in the Merton model compared to the other two models. This indicates that the model choice is already critical for vanilla products.

Figure 4.12 shows the expected exposure (left plot) and the potential future exposure (right plot) for the Bermudan put. We observe that the expected exposure decreases over the lifetime of the option in all three different models. Close to maturity the option is for most paths either already exercised or it is out-of-the-money. The potential future exposure first increases and then decreases in all three models. The profiles of the Black-Scholes and CEV model behave similarly in both cases. The profiles of the Merton model exhibit however a slightly different shape.

Figure 4.13 shows the comparison of exposure profiles of Bermudan options with the one of European options in the three market models. Due to the additional early-exercise

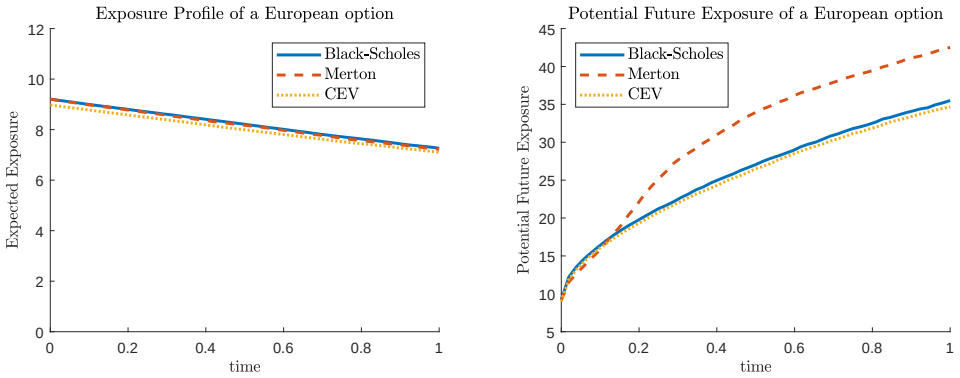

Figure 4.11: Expected exposure (EE) and potential future exposure (PFE) for a European put option in the Black-Scholes model, the Merton model and the CEV model with maturity $T = 1$ and $N_{sim} = 150{,}000$ under the real world measure.

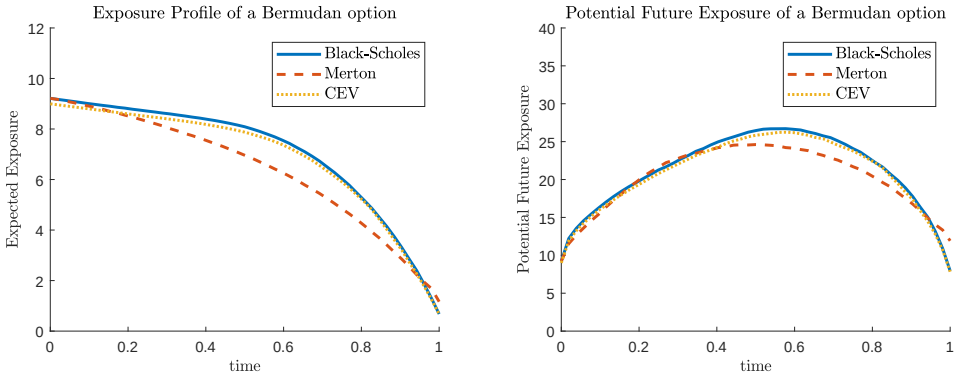

Figure 4.12: Expected exposure (EE) and potential future exposure (PFE) for a Bermudan put option in the Black-Scholes model, the Merton model and the CEV model with maturity $T = 1$ and $N_{sim} = 150{,}000$ under the real world measure.

feature, Bermudan put option prices are higher than the ones of European puts. With this in mind, the difference between the prices can be seen as the added premium of the early exercise possibility. The early-exercise premium depends on the the risk-free interest rate and the dividend yield. In our experiments, the very low rate and zero dividends make the premium almost vanish, see Table 4-N. Over the option's lifetime however, we observe that the early-exercise feature has a significant influence and the expected exposure of a Bermudan option decreases faster than its European counterpart. More precisely, the European and the Bermudan profiles are almost identical in the beginning and diverge towards maturity. For the potential future exposure we observe the same effects. In absolute terms the effect is stronger as for the expected exposure, in relative terms it is slightly lower.

We conclude that replacing a Bermudan by a European option in the exposure calculation has two different effects. First, for most of the option's lifetime it highly over-

estimates both the option's expected exposure and potential future exposure and seems therefore to be too conservative. Secondly, close to $t = 0$ the exposure of the European option underestimates the Bermudan option's exposure slightly. On a netting set level one can therefore not predict with certainty the effect of this simplification. Scenarios are possible where the replacement of a Bermudan by a European option in the exposure calculation leads to a lower CVA. Hence, one cannot conclude that this simplification is always a conservative practice.

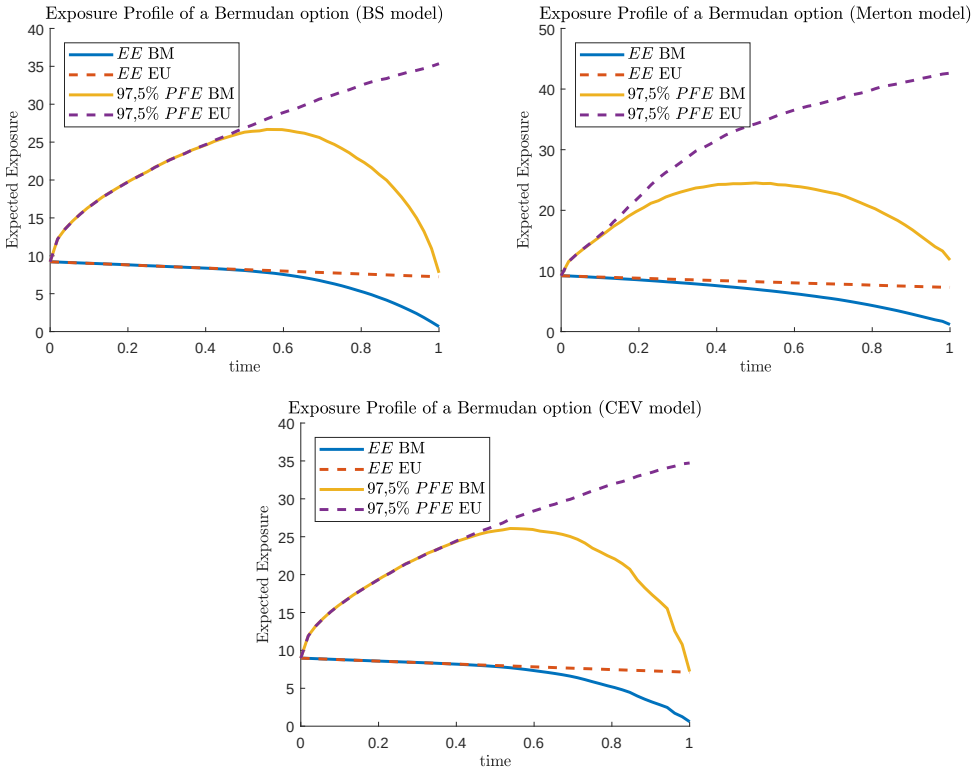

Figure 4.13: Expected exposure (EE) and potential future exposure (PFE) profiles for a Bermudan option and a European put option with maturity $T = 1$ in the Black-Scholes model, the Merton model and the CEV model with $N_{sim} = 150{,}000$ under the real world measure.

In order to get a better understanding of the difference between European and Bermudan options we vary the number of exercise rights per year. We consider five different Bermudan options in the Black-Scholes model and we use a higher rate $r = 0.03$ to better show the differences between European and Bermudan options. Figure 4.14 shows the resulting exposure profile and the corresponding potential future exposure. Both are calculated on a daily basis, i.e. on 252 (trading) days. We observe that the exposure of a Bermudan option drops on the exercise days and between the exercise days behaves similar to a European option. The drops are smaller on the short end and become larger close to maturity. The potential future exposure shows a very similar behaviour. More

		Price	EE at T	PFE at T
BS	European put	9.21	7.23	35.35
	Bermudan put	9.21	0.66	7.75
Merton	European put	9.21	7.29	42.63
	Bermudan put	9.21	1.17	11.80
CEV	European put	8.98	7.11	34.73
	Bermudan put	8.99	0.59	7.22

Table 4-N: Option price, expected exposure and potential future exposure of a European put and a Bermudan put option in the Black-Scholes (BS) model, the Merton model and the CEV model.

exercise rights yield a smoother exposure profile for Bermudan options. Furthermore, we observe that the difference in the exposure profiles between a European option and a Bermudan option with 4 exercise dates is already substantial. On the other hand, the profiles for a Bermudan option with 36 and one with 84 exercise rights are relatively similar. The effect of adding additional exercise rights on the exposure seems to decrease with a higher number of exercise rights.

We conclude that replacing Bermudan options by European options leads to significantly different exposure profiles. However, when it comes to efficiency, one could replace a Bermudan option with a high exercise frequency (or an American option which can always be exercised) by a Bermudan option with only a few exercise rights.

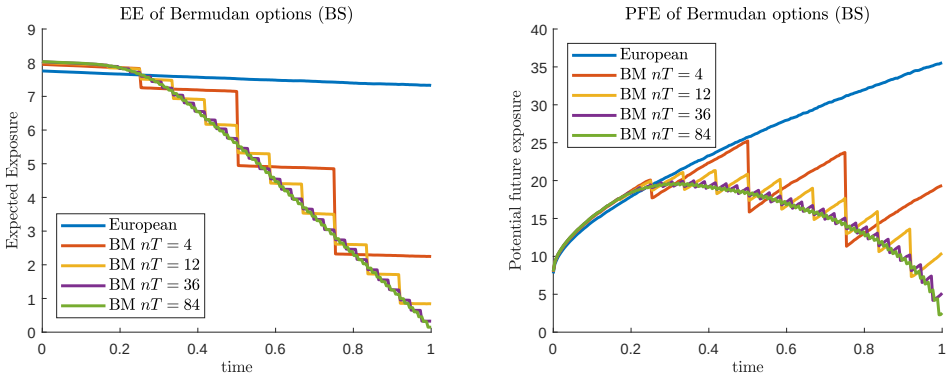

Figure 4.14: Expected exposure (EE) and potential future exposure (PFE) for a Bermudan put option with different exercise frequencies and a European option in the Black-Scholes model with $N_{sim} = 150{,}000$ under the real world measure.

4.7 Conclusion

In this chapter, we investigated the credit exposure calculation for pricing and risk management using Chebyshev interpolation. We proposed a static Chebyshev method as an ad-hoc improvement for a full re-evaluation approach with a slow but accurate numerical pricer. The main contribution of this chapter is a new unified framework for the exposure calculation of path-dependent options based on the dynamic Chebyshev method. The numerical experiments showed that the method is well-suited for the exposure calculation and the structure of the approach yields high efficiency. The dynamic Chebyshev method for exposure calculation admits several qualitative advantages.

- The method offers a high flexibility, the price and the credit exposure of many different products can be calculated with this algorithm and it can be used in different stock price models.
- The structure of the method allows us to explore additional knowledge of the model by choosing different techniques to compute the conditional moments in the pre-computation. This increases the efficiency of the method compared to standard approaches such as least-squares Monte Carlo method.
- The calculated credit exposure can be aggregated on different levels and enables the efficient computation of CVA and other risk metrics on a portfolio level.
- The polynomial structure of the approximation of the value function enables an efficient computation of the option's sensitivities Delta and Gamma in every time step.

An extensive numerical performance analysis confirmed the validity and the efficiency of the method. The approach is able to combine the accuracy of a full re-evaluation approach with the speed of a simpler least-squares Monte Carlo approach. So far, we have focused our analysis on options that are driven by only one risk factor. In the next section, we discuss an efficient version of the dynamic Chebyshev method for path-dependent options with more than one risk factor. These methods are than also interesting candidates for the exposure calculation. We think that a multivariate dynamic Chebyshev method could be interesting for calculating exposure of path-dependent options that depend on up to three risk factors.

The empirical investigation of the exposure profile provides insight into the behaviour of the profiles for different option types and asset models with practical implications. In order to speed-up the exposure calculation common simplifications in practice include the choice of a simple model for the underlying risk factor and the replacement of com-

plex options by simpler options. Our experiments reveal that the first simplification affects the results for barrier and Bermudan options. One reason for different exposure profiles is that even though the models are calibrated to the same instruments, prices for more exotic products can still differ significantly. This effect is not a surprise and is a commonly known issue in calibration and pricing. For barrier options, we have seen that these pricing differences are also reflected in the exposure profiles. However, for example, for European options we have seen that even though prices are very close, the potential future exposure can differ between models with jumps and without. This effect is less obvious and it implies, that even though both models, Black-Scholes and Merton jump-diffusion are a similarly suitable choice for pricing vanilla options, they lead to significantly different credit exposure profiles and thus credit risks. Therefore, it is essential to have a method for the exposure calculation which is *flexible* in the model choice. In particular, one needs to be able to handle different model features such as jumps and local volatility. As in practice a variety of models will be used to quantify the credit exposure a method which allows to easily switch between different models is desirable. As the experiments show the dynamic Chebyshev method for exposure calculation exhibits this flexibility in the models.

Moreover, the experiments show that the replacement of Bermudan options by European options yields significantly different exposure profiles and two different problems occur. First, the exposure of the European option overestimates the exposure of the Bermudan for most of the option's lifetime and second we cannot conclude that this simplification is conservative. Therefore, we recommend to compute the exposure for Bermudan options directly. The presented dynamic Chebyshev method is able to do so in an efficient way.

Chapter 5

Multivariate early-exercise options

In this chapter we consider the pricing of early-exercise options that depend on more than one risk factor. Many relevant option pricing problems are multivariate problems and the pricing of these options is computationally challenging. This is even more the case when one considers options with an early-exercise feature and thus adds a path-dependency to the problem. A variety of different methods have been developed to tackle this problem each of them with different merits and demerits. In relatively low dimensions, it is common to extend classical one-dimensional approaches such as PDE methods or Fourier based methods. These methods produce stable and accurate results but they typically suffer from the curse of dimensionality. On the other hand, simulation based methods such as the least-squares Monte Carlo method can be easily extended to medium and high dimensions since they do not directly suffer from this curse of dimensionality. However, we have seen in the previous chapters that such methods introduce simulation noise and struggle to deliver very accurate results. Moreover, simulation based methods often fail to produce stable sensitivities with respect to the relevant input factors.

We investigate the extension of the dynamic Chebyshev method to multivariate option pricing problems with a focus on asset models where the underlying risk factors are conditionally multivariate normally distributed. We discuss one particular bivariate example in detail to show the potential of our approach. Then we focus on European basket options in a multivariate Black-Scholes model and investigate the efficiency of different quadrature techniques. Finally, we extend this analysis to basket options which admit an early-exercise feature.

5.1 A multivariate dynamic Chebyshev algorithm

We consider the dynamic programming problem of a basket option with payoff function g on d-underlyings driven by an $\mathbb{R}^d$-valued stochastic process $(\boldsymbol{X}_t)_{t\geq 0}$. For the moment, we ignore interest rates and assume $r = 0$. The value of a Bermudan basket option with exercise points t_u, $u = 0, \ldots, n_T$ is computed via the following backward induction

$$\begin{aligned} V_T(\boldsymbol{x}) &= g(\boldsymbol{x}) \\ V_t(\boldsymbol{x}) &= \max\left\{g(\boldsymbol{x}), \mathbb{E}\big[V_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}\big]\right\} \end{aligned}$$

for $\boldsymbol{x} \in \mathbb{R}^d$. Solving this problem by means of the dynamic Chebyshev algorithm using a straightforward tensor based Chebyshev interpolation can be challenging. In the pre-computation step this requires the computation of the conditional expectations of $(N+1)^d$ multivariate polynomials $p_{\boldsymbol{j}}$, $\boldsymbol{j} = (j_1, \ldots, j_d)$ given $(N+1)^d$ starting values $\boldsymbol{x}_k$ with $\boldsymbol{k} = (k_1, \ldots, k_d)$ given by

$$\Gamma^u_{\boldsymbol{j},\boldsymbol{k}} = \mathbb{E}[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}]$$

at each time point t_u. Hence, the number of conditional expectations grows exponentially in d with N^{2d}. Even in relatively low dimensions $d > 1$ this is not feasible. In the following sections, we will show that one can overcome this curse of dimensionality in the pre-computation step in two relevant scenarios. First, the computations are simplified if we assume that $(\boldsymbol{X}_t)_{t\geq 0}$ is a vector of uncorrelated processes. Second, we can find a more efficient solution if we assume that the stochastic process is conditionally multivariate normally distributed.

5.1.1 Independent risk-factors

The first special case that we investigate is the case of (conditionally) independent risk factors. We assume that $\boldsymbol{X}_{t_{u+1}}$ given $\boldsymbol{X}_{t_u} = \boldsymbol{x}$ is a vector of d independent processes and we obtain for the conditional expectations of the multivariate Chebyshev polynomials

$$\mathbb{E}[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}] = \mathbb{E}\Big[\prod_{i=1}^{d} p_{j_i}(X^i_{t_{u+1}})\Big|\boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}\Big] = \prod_{i=1}^{d}\mathbb{E}[p_{j_i}(X^i_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}_{\boldsymbol{k}}].$$

This means instead of N^{2d} the effort is only proportional to dN^{d+1} since in each dimensions we compute the expectations of p_{j_i}, $0 \leq j_i \leq N$ for each Chebyshev grid point $\boldsymbol{x}_{\boldsymbol{k}}$. Further, if we assume that $X^i_{t_{u+1}}$ depends only on the starting value $X^i_{t_u} = x_{k_i}$, we can

simplify the expression above

$$\prod_{i=1}^{d} \mathbb{E}[p_{j_i}(X^i_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_k}] = \prod_{i=1}^{d} \mathbb{E}[p_{j_i}(X^i_{t_{u+1}})|X^i_{t_u} = x_{k_i}].$$

In each dimension $i = 1, \dots, d$ we need to compute

$$\Gamma^{u,i}_{j_i,k_i} = \mathbb{E}[p_{j_i}(X^i_{t_{u+1}})|X^i_{t_u} = x_{k_i}] \quad \text{for} \quad j_i = 0, \dots, N_i \quad \text{and} \quad k_i = 0, \dots, N_i.$$

Again, if we choose an equidistant time-stepping and the process has stationary increments the computation of the conditional expectations become independent of the time stepping.

Hence, the calculations in the pre-computation step are equivalent to d times the pre-computation step of the one-dimensional dynamic Chebyshev method. The effort scales with dN^2 and we no longer observe the curse of dimensionality in the pre-computation step.

In the backward induction we still need to compute the values $V_{t_u}(\boldsymbol{x_k})$ for all grid points $\boldsymbol{x_k}$ and thus obtain a full tensor $\widehat{V}_{t_u} \in \mathbb{R}^{N_1 \times \dots \times N_d}$. Fortunately, each step in the backward induction can be written as simple matrix times tensor multiplications. We will illustrate this for $d = 2$ and then extend the concept to a general $d > 2$.

Let C^{u+1} be the matrix of interpolation coefficients of the bivariate Chebyshev interpolation at t_{u+1} given by

$$C^{u+1}_{j_1,j_2} = \frac{2^{\mathbb{1}_{0<j_1<N_1}} 2^{\mathbb{1}_{0<j_2<N_1}}}{N_1 N_2} \sum_{k_1=0}^{N_1}{}'' \sum_{k_2=0}^{N_2}{}'' \widehat{V}^{u+1}_{k_1,k_2} T_{j_1}(z_{k_1}) T_{j_2}(z_{k_2})$$

where $\widehat{V}^{u+1}$ is the matrix of nodal values at t_{u+1}. We recall that the continuation value at the nodal points $\boldsymbol{x_k}$ at time step t_u is given by

$$\begin{aligned}
\widehat{V}^u_{k_1,k_2} &= \sum_{j_1=0}^{N_1} \sum_{j_2=0}^{N_2} C^{u+1}_{j_1,j_2} \mathbb{E}[p_{j_1}(X^1_{t_{u+1}}) p_{j_2}(X^2_{t_{u+1}})|(X^1_{t_u}, X^2_{t_u}) = (x_{k_1}, x_{k_2})] \\
&= \sum_{j_1=0}^{N_1} \sum_{j_2=0}^{N_2} C^{u+1}_{j_1,j_2} \mathbb{E}[p_{j_1}(X^1_{t_{u+1}})|X^1_{t_u} = x_{k_1}] \mathbb{E}[p_{j_2}(X^2_{t_{u+1}})|X^2_{t_u} = x_{k_2}] \\
&= \sum_{j_1=0}^{N_1} \sum_{j_2=0}^{N_2} C^{u+1}_{j_1,j_2} \Gamma^1_{j_1,k_1} \Gamma^2_{j_2,k_2}.
\end{aligned}$$

Due to the independence of the conditional expectation we can write this equation in

matrix form as

$$\widehat{V}^u = (\Gamma^1)^T C^{u+1} \Gamma^2.$$

If the option is exercised at time point t_u we update the matrix of nodal values accordingly. In the same way we can write the coefficient matrix as a product of $\widehat{V}^u$ and two auxiliary matrices consisting of entries $T_{j_i}(z_{k_i})$ for each dimension. This matrix notation makes the method very easy to implement and highly efficient.

In more than two dimensions $\widehat{V}^{u+1}$ becomes at tensor and the simple matrix times matrix needs to be replaced by a form of matrix times tensor product. From [3] we obtain the definition of the n-mode tensor product $\times_n$ of a tensor with a matrix.

Definition 9. *Let $\mathcal{A} \in \mathbb{R}^{N_1 \times \ldots \times N_d}$ be a tensor and $U \in \mathbb{R}^{N_n \times M_n}$ a matrix. We define the n-mode product $\mathcal{A} \times_n U$ of a $\mathcal{A}$ and U as a tensor in $\mathbb{R}^{N_1 \times \ldots \times M_n \times \ldots \times N_d}$ with entries*

$$(\mathcal{A} \times_n U)(i_1, \ldots, i_{n-1}, j_n, i_{n+1}, \ldots, i_d) = \sum_{i_n=1}^{N_n} \mathcal{A}(i_1, \ldots, i_d) U(j_n, i_n).$$

For example, if $d = 2$ we can write the singular value decomposition of a $m \times n$ matrix A in terms of two n-mode multiplications as

$$A = U \Sigma V^T = \Sigma \times_1 U \times_2 V$$

for a $m \times k$ matrix U, a $m \times k$ matrix V and a $k \times k$ matrix Σ.

We can use this notation in the multivariate dynamic Chebyshev method in d dimension. In the backward induction, every time step $t_{u+1} \to t_u$ can be written as

$$\widehat{V}^u = C^{u+1} \times_1 \Gamma^1 \times_2 \ldots \times_d \Gamma^d$$

where C^{u+1} is the tensor of coefficients at t_{u+1} and $\widehat{V}^u$ is the tensor of nodal values at t_u. In *Matlab*, the n-mode matrix times tensor multiplication can be implemented using the tensor toolbox of [4].

Our initial assumption of zero correlation between the risk factors is a significant limitation of this approach. Usually risk factors such as different stock prices or stochastic rates and stochastic credit spreads are strongly correlated. In some cases, it might be justified to ignore this correlation since its effect on the pricing is comparably small. In these situations the presented multivariate dynamic Chebyshev method can be directly applied.

However, in many situations the effects of ignoring the correlation between the risk factors is significant. Fortunately, if the process is conditionally normally distributed, transformations are available such that the transformed process becomes conditionally independent. In the next section, we present such a transformation and the resulting algorithm for a multivariate Black-Scholes model.

5.1.2 Extension to the multivariate Black-Scholes model

In this section, we consider a multivariate extension of the classical Black-Scholes model and price basket options with an early-exercise feature in this model. The log-stock prices are normally distributed and we can write them as a linear transformation of a set of independent risk factors. In this case, the multivariate dynamic Chebyshev method can be simplified in the same way as seen in the previous section.

We assume a market with d assets modelled by the following SDE under the risk-neutral pricing measure

$$\mathrm{d}S_t^i = S_t^i\big(r\mathrm{d}t + \sigma_i \mathrm{d}W_t^i\big), \quad i = 1, \dots, d$$

with starting value S_0 and for a risk-free interest rate $r \geq 0$, volatilities $\sigma_1, \dots, \sigma_d > 0$ and standard Brownian motions $W_t^1, \dots, W_t^d$. Assume that the Brownian motions W_t^i are correlated with

$$\mathrm{d}W_t^i \mathrm{d}W_t^j = \varrho_{ij}\mathrm{d}t \quad \text{for} \quad \varrho_{ij} \in [-1, 1]$$

with $\varrho_{ii} = 1$, $i = 1, \dots, d$. The log-returns $X_t^i = \log(S_t^i)$ fulfils the SDE

$$\mathrm{d}X_t^i = \big(r - \frac{\sigma_i^2}{2}\big)\mathrm{d}t + \sigma_i \mathrm{d}W_t^i.$$

For the d-variate stochastic process $\boldsymbol{X}_t = (X_t^1, \dots, X_t^d)$ it holds for $t > s$

$$\boldsymbol{X}_t | \boldsymbol{X}_s = \boldsymbol{x} \ \sim \ \mathcal{N}_d\big(\boldsymbol{x} + \boldsymbol{\mu}_{t-s}, \Sigma_{t-s}\big)$$

$$\boldsymbol{\mu}_{t-s} = \begin{pmatrix} r - \frac{\sigma_1^2}{2} \\ \vdots \\ \vdots \\ r - \frac{\sigma_d^2}{2} \end{pmatrix} (t-s) \quad \text{and} \quad \Sigma_{t-s} = \begin{pmatrix} \sigma_1^2 & \varrho_{12}\sigma_1\sigma_2 & \dots & \dots & \varrho_{1d}\sigma_1\sigma_d \\ \varrho_{12}\sigma_2\sigma_1 & \ddots & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \varrho_{1d}\sigma_d\sigma_1 & \dots & \dots & \dots & \sigma_d^2 \end{pmatrix} (t-s).$$

For notational convenience we write $\boldsymbol{\mu} = \boldsymbol{\mu}_1$ and $\Sigma = \Sigma_1$. We assume throughout this section that the covariance matrix has full rank and thus is positive definite. From Σ positive definite and symmetric follows that Σ has a singular value decomposition $\Sigma = O\Lambda O^T$ for an orthogonal matrix O with $O^{-1} = O^T$ and a diagonal matrix Λ. Exploring this result, we can rewrite the covariance matrix as

$$\Sigma = O\Lambda O^T = (O\Lambda^{1/2})(O\Lambda^{1/2})^T,$$

where $\Lambda^{1/2}$ is a diagonal matrix with diagonal elements $\Lambda_{i,i}^{1/2} = \sqrt{\Lambda_{i,i}}$ for $i = 1, \ldots, d$. The process $\boldsymbol{X}_t$ has stationary increments and $\boldsymbol{X}_{t_{u+1}} | \boldsymbol{X}_{t_u} = \boldsymbol{x}$ is equal in distribution to the variable $\boldsymbol{x} + \boldsymbol{X}_{\Delta t}$ with $\Delta t = t_{u+1} - t_u$. For the multivariate normal distribution under linear transformations holds

$$\boldsymbol{x} + \boldsymbol{X}_{\Delta t} = O\Lambda^{1/2}\sqrt{\Delta t}\boldsymbol{Z} + \boldsymbol{\mu}_{\Delta t} + \boldsymbol{x} = O\tilde{\boldsymbol{Z}} + \boldsymbol{\mu}_{\Delta t} + \boldsymbol{x} \quad \text{with} \quad \tilde{\boldsymbol{Z}} \sim \mathcal{N}_d(0, \Lambda_{\Delta t}) \tag{5.1}$$

see for example [126]. Here, $\tilde{\boldsymbol{Z}}$ is a vector of d independent normally distributed random variables with variances $\Delta t \Lambda_{i,i}$. The idea of our new method is to interpolate in the variable $\tilde{\boldsymbol{Z}}$ instead of $\boldsymbol{X}$. In this case we can explore that

$$\mathbb{E}\big[p_{\boldsymbol{j}}(\tilde{\boldsymbol{Z}})\big] = \mathbb{E}\big[\prod_{i=1}^{d} p_{j_1}(\tilde{Z}_i)\big] = \prod_{i=1}^{d} \mathbb{E}\big[p_{j_1}(\tilde{Z}_i)\big].$$

The remaining challenge is to incorporate this transformation into the framework of the multivariate dynamic Chebyshev algorithm. We start again with the dynamic programming problem

$$\begin{aligned} V_T(\boldsymbol{x}) &= g(\boldsymbol{x}) \\ V_{t_u}(\boldsymbol{x}) &= \max\Big\{g(\boldsymbol{x}), \mathbb{E}\Big[V_{t_{u+1}}(\boldsymbol{X}_{t+1}) | \boldsymbol{X}_{t_u} = \boldsymbol{x}\Big]\Big\}. \end{aligned}$$

Our goal is to modify the dynamic Chebyshev algorithm in such a way that we can solve the curse of dimensionality in the pre-computation step. Using (5.1) we can write the continuation value as

$$\mathbb{E}\Big[V_{t_u}(\boldsymbol{X}_{t_u}) | \boldsymbol{X}_{t_{u-1}} = \boldsymbol{x}\Big] = \mathbb{E}\Big[V_{t_u}(\boldsymbol{x} + \boldsymbol{X}_{\Delta t})\Big] = \mathbb{E}\Big[V_{t_u}(O\tilde{\boldsymbol{Z}} + \boldsymbol{\mu}_{\Delta t} + \boldsymbol{x})\Big].$$

Defining $\boldsymbol{z} = O^{-1}\boldsymbol{x}$ yields

$$\mathbb{E}\Big[V_{t_u}(\boldsymbol{X}_{t_u}) | \boldsymbol{X}_{t_{u-1}} = O\boldsymbol{z}\Big] = \mathbb{E}\Big[V_{t_u}(O\tilde{\boldsymbol{Z}} + \boldsymbol{\mu}_{\Delta t} + O\boldsymbol{z})\Big] = \mathbb{E}\Big[\tilde{V}_{t_u}(\boldsymbol{z} + \tilde{\boldsymbol{Z}})\Big]$$

for the modified value function $[\underline{z}, \overline{z}]^d \ni \tilde{\boldsymbol{z}} \mapsto \tilde{V}_{t_u}(\tilde{\boldsymbol{z}}) := V_{t_u}(O\tilde{\boldsymbol{z}} + \boldsymbol{\mu}_{\Delta t})$. Assume we

have a d-dimensional Chebyshev approximation of $\tilde{V}_{t_u}$ given by $\sum_{\boldsymbol{j}} c_{\boldsymbol{j}}^u p_{\boldsymbol{j}}$. Then we can approximate the expectation of $\tilde{V}_{t_u}(\tilde{\boldsymbol{z}})$ by

$$\mathbb{E}\Big[\tilde{V}_{t_u}(\boldsymbol{z}+\tilde{\boldsymbol{Z}})\Big] \approx \mathbb{E}\Big[\sum\nolimits_{\boldsymbol{j}} c_{\boldsymbol{j}}^u p_{\boldsymbol{j}}(\boldsymbol{z}+\tilde{\boldsymbol{Z}})\Big] = \sum\nolimits_{\boldsymbol{j}} c_{\boldsymbol{j}}^u \mathbb{E}\Big[p_{\boldsymbol{j}}(\boldsymbol{z}+\tilde{\boldsymbol{Z}})\Big].$$

Here, the independence of the variables $\tilde{Z}_i$ yields

$$\mathbb{E}\Big[p_{\boldsymbol{j}}(\boldsymbol{z}+\tilde{\boldsymbol{Z}})\Big] = \mathbb{E}\Big[\prod_{i=1}^{d} p_{j_i}(z_i+\tilde{Z}_i)\Big] = \prod_{i=1}^{d} \mathbb{E}\big[p_{j_i}(z_i+\tilde{Z}_i)\big].$$

Following this approach, we can approximate the continuation value at time point t_{u-1} using only products of conditional expectations of one-dimensional Chebyshev interpolations. For this, we need the Chebyshev interpolation of $[\underline{z},\overline{z}]^d \ni \tilde{\boldsymbol{z}} \mapsto \tilde{V}_{t_u}(\tilde{\boldsymbol{z}})$ and hence, we require the function values $\tilde{V}_{t_u}(\boldsymbol{z_k}) = V_{t_u}(O\boldsymbol{z_k}+\boldsymbol{\mu}_{\Delta t})$ at the Chebyshev nodal points $\boldsymbol{z_k}$. For $O\boldsymbol{z_k}+\boldsymbol{\mu}_{\Delta t} = O(\boldsymbol{z_k}+O^{-1}\boldsymbol{\mu}_{\Delta t})$ we get

$$\begin{aligned} V_{t_u}(O\boldsymbol{z_k}+\boldsymbol{\mu}_{\Delta t}) &= \max\Big\{g(O\boldsymbol{z_k}+\boldsymbol{\mu}_{\Delta t}), \mathbb{E}\Big[V_{t_{u+1}}(\boldsymbol{X}_t)|\boldsymbol{X}_{t_u} = O\boldsymbol{z_k}+\boldsymbol{\mu}_{\Delta t}\Big]\Big\} \\ &= \max\Big\{g(O\boldsymbol{z_k}+\boldsymbol{\mu}_{\Delta t}), \mathbb{E}\Big[V_{t_{u+1}}(O(\boldsymbol{z_k}+O^{-1}\boldsymbol{\mu}_{\Delta t}+\tilde{\boldsymbol{Z}})+\boldsymbol{\mu}_{\Delta t})\Big]\Big\}. \end{aligned}$$

Assume $\sum_{\boldsymbol{j}} c_{\boldsymbol{j}}^{u+1} p_{\boldsymbol{j}}$ is an approximation of $[\underline{z},\overline{z}]^d \ni \tilde{\boldsymbol{z}} \mapsto V_{t_{u+1}}(O\tilde{\boldsymbol{z}}+\boldsymbol{\mu}_{\Delta t})$. Then we obtain

$$\begin{aligned} V_{t_u}(O\boldsymbol{z_k}+\boldsymbol{\mu}_{\Delta t}) &= \max\Big\{g(O\boldsymbol{z_k}+\boldsymbol{\mu}_{\Delta t}), \mathbb{E}\Big[V_{t_{u+1}}(O(\boldsymbol{z_k}+O^{-1}\boldsymbol{\mu}_{\Delta t}+\tilde{\boldsymbol{Z}})+\boldsymbol{\mu}_{\Delta t})\Big]\Big\} \\ &\approx \max\Big\{g(O\boldsymbol{z_k}+\boldsymbol{\mu}_{\Delta t}), \sum\nolimits_{\boldsymbol{j}} c_{\boldsymbol{j}}^{u+1} \prod_{i=1}^{d} \mathbb{E}\big[p_{j_i}(z_{k_i}+(O^{-1}\boldsymbol{\mu}_{\Delta t})_i+\tilde{Z}_i)\big]\Big\} \end{aligned}$$

where $z_{k_i}+(O^{-1}\boldsymbol{\mu}_{\Delta t})_i+\tilde{Z}_i \sim \mathcal{N}(z_{k_i}+(O^{-1}\boldsymbol{\mu}_{\Delta t})_i, \Delta t\Lambda_{i,i})$. The expectations

$$\mathbb{E}\big[p_{j_i}(z_{k_i}+(O^{-1}\boldsymbol{\mu}_{\Delta t})_i+\tilde{Z}_i)\big] \quad 0 \le k_i \le N_i,\ 0 \le j_i \le N_i \quad \text{for} \quad i = 1,\dots,d$$

can be pre-computed. This is essentially d times the offline step of the one-dimensional dynamic Chebyshev algorithm. The calculated nodal values $V_{t_u}(O\boldsymbol{z_k}+\boldsymbol{\mu}_{\Delta t})$ are then used to compute the coefficients of the multivariate Chebyshev interpolation $c_{\boldsymbol{j}}^u$, similar to the normal dynamic Chebyshev algorithm. In order to recover $V_{t_u}(\boldsymbol{x})$ we have

$$V_{t_u}(\boldsymbol{x}) = V_{t_u}(\boldsymbol{x}-\boldsymbol{\mu}_{\Delta t}+\boldsymbol{\mu}_{\Delta t}) = V_{t_u}(O(O^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_{\Delta t}))+\boldsymbol{\mu}_{\Delta t}) = \tilde{V}_{t_u}(O^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_{\Delta t})).$$

The backward induction in the new approach is essentially the same as the backward induction in the (normal) dynamic Chebyshev methods, only the pre-computation of the generalized moments has changed.

Fixing an interpolation domain

An interpolation domain can be fixed using a confidence interval for a normal distribution, see Remark 3. First, we recall that

$$\boldsymbol{x} + \boldsymbol{X}_{\Delta t} = O\tilde{\boldsymbol{Z}} + \boldsymbol{\mu}_{\Delta t} + \boldsymbol{x} = O(O^{-1}\boldsymbol{x} + \tilde{\boldsymbol{Z}}) + \boldsymbol{\mu}_{\Delta t} \quad \text{with} \quad \tilde{\boldsymbol{Z}} \sim \mathcal{N}_d(0, \Lambda_{\Delta t}).$$

for $\Delta t = t_{u+1} - t_u$. For the multivariate dynamic Chebyshev method we require an interpolation domain which is large enough for the random variable $(O^{-1}\boldsymbol{x} + \tilde{\boldsymbol{Z}})$. This means the probability that it lies in the domain should be close to 1.

Let $\boldsymbol{S}_0$ be the vector of starting values for which we want to price the option and $\boldsymbol{x}_0 = \log(\boldsymbol{S}_0)$. We transform the vector using O^{-1} and obtain $\boldsymbol{z}_0 = O^{-1}\boldsymbol{x}_0$. Next, we define a rectangular around this point using the standard deviation of $\tilde{\boldsymbol{Z}}$ and adjust for the maturity T. Let $\tilde{\sigma}_i$ be the standard deviation of $\tilde{Z}_i$, scaling it from the time step Δt to the interval $[0, T]$ gives us $\sigma_i = \tilde{\sigma}_i\sqrt{T/\Delta t}$. This allows us to define the interpolation domain $\mathcal{Z} = [\underline{z}_1, \overline{z}_1] \times \ldots \times [\underline{z}_d, \overline{z}_d]$ with $\underline{z}_i = z_{0,i} - k\sigma_i$ and $\overline{z}_i = z_{0,i} + k\sigma_i$ for k large enough. A reasonable choice here is $k = 4$ or $k = 5$ which implies that the process stays in the interpolation domain with a probability of 99.99367% or 99.99994%.

The multivariate dynamic Chebyshev algorithm

Here, we summarize the presented algorithm and formulate it in terms of tensors and matrices. This formulation allows for a simple implementation of the code in *Matlab* using the tensor toolbox of [4] if $d > 2$. Similarly to the standard dynamic Chebyshev method we can split the algorithm in a pre-computation step and the pricing via backward-induction.

Pre-computation step:

1. Pre-requisites: Maturity T, positive definite covariance matrix Σ, interest rate r, starting value $\boldsymbol{X}_0$ and uniform time stepping $0 = t_0 < \ldots < t_{n_T} = T$ with $t_{u+1} - t_u = \Delta t$.

2. Eigenvalue decomposition of the covariance matrix: Perform an eigenvalue decomposition of Σ and obtain $\Sigma = O\Lambda O^T$ for a diagonal matrix $\Lambda = \text{diag}(\Lambda_1, \ldots, \Lambda_d)$ and an orthogonal matrix O. Define corresponding drift $\boldsymbol{\mu} = (r - 0.5\Sigma_{11}, \ldots, r - 0.5\Sigma_{dd})^T$ and $\boldsymbol{\mu}^{\Delta t} = \boldsymbol{\mu}\Delta t$.

3. Fix a domain $\mathcal{Z} = [\underline{z}_1, \overline{z_1}] \times \ldots \times [\underline{z}_d, \overline{z_d}]$ via

$$[\underline{z}_i, \overline{z}_i] = [(O^{-1}(\boldsymbol{\mu}T + \boldsymbol{X}_0))_i - k(T\Lambda_i)^{1/2}, (O^{-1}(\boldsymbol{\mu}T + \boldsymbol{X}_0))_i + k(T\Lambda_i)^{1/2}]$$

for $i = 1, \dots, d$ and $k \in \{3, 4, 5\}$. For Chebyshev degree $\boldsymbol{N} = (N_1, \dots, N_d) \in \mathbb{N}^d$, define d-variate nodal points $\boldsymbol{z_k}$. Create vectors of nodal points

$$\boldsymbol{Z}^i = \frac{\underline{z}_i + \overline{z}_i}{2} + \frac{\overline{z}_i - \underline{z}_i}{2} \begin{pmatrix} \cos(0\pi/N_i) \\ \vdots \\ \cos(N_i\pi/N_i) \end{pmatrix} \in \mathbb{R}^{N_i+1} \quad \text{for} \quad i = 1, \dots, d.$$

4. Pre-computation of conditional expectations: For $i = 1, \dots, d$, compute

$$\Gamma^i_{j,k} := \mathbb{E}\big[p_j(z_k + (O^{-1}\boldsymbol{\mu}_{\Delta t})_i + \tilde{Z}_i)\big], \quad 0 \le k \le N_i,\ 0 \le j \le N_i$$

where $\tilde{Z}_i \sim \mathcal{N}(0, \Lambda_{i,i})$. In matrix form, we compute parameters

$$(\tilde{\mu}^i_0, \dots, \tilde{\mu}^i_N)^T = -1 + \frac{2}{\overline{z}_i - \underline{z}_i}(\boldsymbol{Z}^i + (O^T\boldsymbol{\mu}_{\Delta t})_i - \underline{z}_i) \in \mathbb{R}^{N_i+1},$$
$$\tilde{\sigma_i}^2 = \Big(\frac{2}{\overline{z}_i - \underline{z}_i}\Big)^2 \Lambda_i \Delta t$$

and obtain the matrices

$$\Gamma^i = \begin{pmatrix} \Gamma(\tilde{\mu}^i_0, \tilde{\sigma_i}^2, 0) & \dots & \dots & \Gamma(\tilde{\mu}^i_N, \tilde{\sigma_i}^2, 0) \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ \Gamma(\tilde{\mu}^i_0, \tilde{\sigma_i}^2, N) & \dots & \dots & \Gamma(\tilde{\mu}^i_N, \tilde{\sigma_i}^2, N) \end{pmatrix} \quad \text{for} \quad i = 1, \dots, d$$

where $\Gamma(\tilde{\mu}^i_k, \tilde{\sigma_i}^2, j) = \mathbb{E}[T_j(Y)\mathbb{1}_{[-1,1]}(Y)]$ for $Y \sim \mathcal{N}(\tilde{\mu}^i_k, \tilde{\sigma_i}^2)$.

5. Pre-computation of auxiliary matrices for the computation of the coefficients

$$T^i = \frac{2}{N_i} \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \dots & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & \cos(\frac{\pi}{N_i}) & \dots & \cos(\frac{\pi(N_i-1)}{N_i}) & \frac{1}{2}\cos(\pi) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{4} & \frac{1}{2}\cos(\pi) & \dots & \frac{1}{2}\cos(\pi(N_i-1)) & \frac{1}{4}\cos(\pi N_i) \end{pmatrix} \quad i = 1, \dots, d.$$

Pricing via backward-induction:

1. Fix a payoff function g and compute tensor $\mathcal{G}$ of grid values $G_{\boldsymbol{k}} = g(O\boldsymbol{z}_{\boldsymbol{k}} + \boldsymbol{\mu}_{\Delta t})$

2. Initial time step: Use the results from step 5. to calculate nodal values $\tilde{V}_T(\boldsymbol{z}_{\boldsymbol{k}}) = V_T(O\boldsymbol{z}_{\boldsymbol{k}} + \boldsymbol{\mu}_{\Delta t}) = G_{\boldsymbol{k}}$ and coefficients $c^T_{\boldsymbol{j}}$. The tensor of coefficients is computed using the tensor of nodal values $\mathcal{V}^T$ via

$$\mathcal{C}^T = \mathcal{V}^T \times_1 T^1 \times_2 \ldots \times_d T^d.$$

3. Time stepping $t_{u+1} \to t_u$: Assume we have coefficients $c^{u+1}_{\boldsymbol{j}}$, then we obtain new coefficients

$$c^u_{\boldsymbol{j}} = \Big(\prod_{i=1}^d \frac{2^{\mathbb{1}_{0<j_i<N_i}}}{N_i}\Big) {\sum_{\boldsymbol{k}}}'' \tilde{V}_{t_u}(\boldsymbol{z}_{\boldsymbol{k}}) \prod_{i=1}^d \cos\Big(j_i \frac{k_i \pi}{N_i}\Big)$$
$$\text{for} \quad \tilde{V}_{t_u}(\boldsymbol{z}_{\boldsymbol{k}}) = \max\Big\{G_{\boldsymbol{k}}, \sum\nolimits_{\boldsymbol{j}} c^{u+1}_{\boldsymbol{j}} \prod_{i=1}^d \Gamma^i_{j_i,k_i}\Big\}.$$

   We can write this in tensor form for the tensor of coefficients $\mathcal{C}^{u+1}$ as

$$\mathcal{C}^u = \mathcal{V}^u \times_1 T^1 \times_2 \ldots \times_d T^d \quad \text{for} \quad \mathcal{V}^u = \max\{\mathcal{G}, \mathcal{C}^{u+1} \times_1 \Gamma^1 \times_2 \ldots \times_d \Gamma^d\}.$$

   Here, the maximum function on two tensors refers to the elementwise maximum and returns again a tensor.

4. Approximation of the option price at $t = 0$

$$V_0(\boldsymbol{x}) = \tilde{V}_0(O^{-1}(\boldsymbol{x} - \boldsymbol{\mu}^{\Delta t})) \approx \sum\nolimits_{\boldsymbol{j}} c^0_{\boldsymbol{j}}\, p_{\boldsymbol{j}}(O^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_{\Delta t})).$$

   This evaluation can be performed via tensor times matrix multiplication for $\boldsymbol{z} = O^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_{\Delta t})$

$$V_0(\boldsymbol{x}) = \mathcal{C}^0 \times_1 P^1 \times_2 \ldots \times_d P^d \quad \text{for} \quad P^i = \big(p_0(\boldsymbol{z}_i), \ldots, p_N(\boldsymbol{z}_i)\big)^T \quad i = 1, \ldots, d.$$

First time step: "Smoothing the payoff"
We could start the backward induction by interpolating the payoff $\tilde{\boldsymbol{z}} \mapsto \tilde{V}_T(\tilde{\boldsymbol{z}}) = g(O\tilde{\boldsymbol{z}} + \boldsymbol{\mu}_{\Delta t})$ and then calculating coefficients $c^T_{\boldsymbol{j}}$. However, since g is typically not differentiable this approach would require a relatively high number of nodal points and is therefore not efficient. Instead, it is better to compute the continuation value at $T - 1$ directly, using numerical quadrature techniques and the density of the normal distribution. This

is the same "smoothing the payoff" idea that we introduced in Section 3.4.1.

For a higher accuracy, we can replace the initial time step 6. in the algorithm by a new time step 6. and start the time stepping for $t_{n_T-1} \to t_{n_T-2}$.

6. First time step: Compute continuation values

$$CV_{t_{n_T-1}}(\boldsymbol{z_k}) = \mathbb{E}\Big[g(O(\boldsymbol{z_k} + O^{-1}\boldsymbol{\mu}_{\Delta t} + \tilde{\boldsymbol{Z}}) + \boldsymbol{\mu})\Big]$$

for $\tilde{\boldsymbol{Z}} \sim \mathcal{N}_d(0, \Lambda)$. Use these values to compute nodal values $\tilde{V}_{t_{n_T-1}}(\boldsymbol{z_k})$ and Chebyshev coefficients $c_{\boldsymbol{j}}^{n_T-1}$.

In order to compute the conditional expectation more efficiently one can apply the smoothing idea of [11]. They consider the pricing of European basket options, i.e. d-variate quadrature problem with a non-smooth integrand. [11] explore the smoothing properties of the density of a normal distribution and obtain a $d-1$ dimensional quadrature problem with a smooth integrand. We will discuss this in more detail in Section 5.3.

5.1.3 A first numerical experiment

We investigate the dynamic Chebyshev algorithm in a model where the underlying risk factors are bivariate normally distributed. We price a basket put option on two assets with a Bermudan-style exercise feature using the dynamic Chebyshev method and compare the results with the least-squares Monte Carlo algorithm.

The pricing problem is defined as

$$\begin{aligned} V_T(x_1, x_2) &= g(x_1, x_2) = \max\left\{K - 0.5e^{x_1} - 0.5e^{x_2}, 0\right\} \\ V_t(x_1, x_2) &= \max\left\{g(x_1, x_2),\, \mathbb{E}\big[V_{t_{u+1}}(X^1_{t_{u+1}}, X^2_{t_{u+1}})\big|X^1_{t_u} = x_1, X^2_{t_u} = x_2\big]\right\} \end{aligned}$$

where $(X_t^1, X_t^2)_{t\geq 0}$ is a stochastic process with normally distributed increments, i.e.

$$\begin{pmatrix} X_t^1 \\ X_t^2 \end{pmatrix} \sim \mathcal{N}_2\Big(\boldsymbol{X}_0 + \boldsymbol{\mu}t, t\boldsymbol{\Sigma}\Big) \quad \text{with} \quad \boldsymbol{\mu} = \begin{pmatrix} r - 0.5\sigma_1^2 \\ r - 0.5\sigma_2^2 \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \varrho\sigma_1\sigma_2 \\ \varrho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}.$$

We fix as parameters

$$K = 100, \quad T = 1, \quad r = 0.03, \quad \sigma_1 = 0.25, \quad \sigma_2 = 0.2, \quad \text{and} \quad \varrho = 0.4.$$

Moreover, we assume an equidistant time stepping with 252 time steps per year, i.e.

daily exercise. For the experiments, we fix an interpolation domain in the transformed $\tilde{\boldsymbol{Z}}$ variable using five times the standard deviation. The generalized moments are calculated using the analytic formula presented in Remark 2.

We perform two experiments in this model. First, we run the dynamic Chebyshev algorithm for an increasing number of nodal points with $N = 16, 32, 64, 128$ and we calculate prices on a grid of 11×11 starting values, $S_0^1 = \exp(X_0^1), S_0^2 = \exp(X_0^2)$ equally distributed between 90 and 110. From these prices, we estimate the empirical order of convergence (EOC). As a second experiment, we compare the dynamic Chebyshev method with the least-squares Monte Carlo method for an increasing number of nodal points/simulation paths in terms of accuracy and runtime. We run the dynamic Chebyshev method for $N = 16, 32, 64$ and the Monte Carlo simulation for $M =$ 5,000, 20,000, 80,000. The price is computed for a starting value of $X_0^1 = X_0^2 = \log(100)$.

For the least-squares Monte Carlo approach we use antithetic variates as variance reduction technique, 6 basis functions for the regression and we take the mean error over 50 runs for more stable results. In both experiments we apply the "smoothing the payoff" idea in the dynamic Chebyshev algorithm using the density function and Clenshaw-Curtis quadrature to compute the continuation value at $T - 1$, with 500 quadrature points in each dimension. The reference price is computed using the least-squares Monte Carlo algorithm with 15 basis function and $M = 400{,}000$ samples and we took the average price over five hundred runs. As basis functions for the least-squares Monte-Carlo algorithm we use $1, S_1, S_2, S_1^2, S_2^2, S_1S_2$ for the version with 6 functions and additionally $S_1^3, S_2^3, S_1^4, S_2^4, S_1^2S_2, S_1S_2^2, S_1^3S_2, S_1S_2^3, S_1^2S_2^2$ for the one with 15 functions.

Figure 5.1 shows the results for both experiments. In the left plot, we observe the empirical order of convergence as a function of the Chebyshev N. We observe that the algorithm converges and the method reaches an error smaller than 10^{-3} for 64 nodal points per dimension. The right picture shows the comparison between the dynamic Chebyshev method and the least-squares Monte Carlo approach. We observe that the Chebyshev method achieves a much higher accuracy in the same runtime. For example, an accuracy below 10^{-2} is achieved in only $0.03s$ whereas the Monte Carlo approach requires $1.2s$ for a lower accuracy of $2.8{\cdot}10^{-2}$. This indicates that the dynamic Chebyshev method is able to outperform the least-squares Monte Carlo approach in this bivariate example.

5.1.4 Omit calculation of coefficients

When computing option prices with the dynamic Chebyshev method we are often only interested in the price at $t = 0$ and thus we do not require the coefficients c_j^u in every time

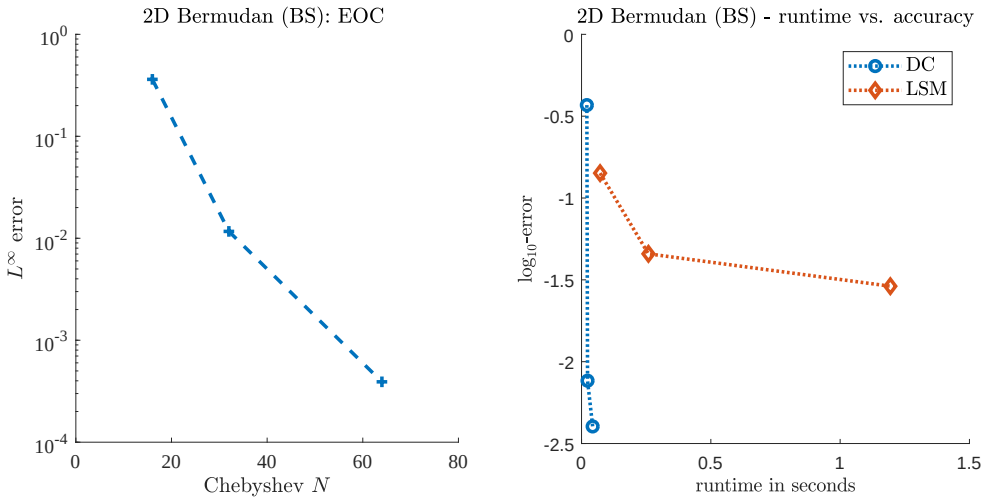

Figure 5.1: Error decay of the dynamic Chebyshev method for a basket put option on two assets with Bermudan exercise feature. Left plot: Empirical order of convergence in the Black-Scholes model. Right plot: Comparison with least-squares Monte Carlo in terms of runtime and accuracy.

step. In this case, we can simplify the dynamic Chebyshev method and compute only the nodal values in the time stepping and not the coefficients. This should approximately half the run time of the time stepping. For the overall runtime, the effect depends on the relation between the runtime of the pre-computation step and the runtime of the time stepping.

We will present this idea for the univariate dynamic Chebyshev method. At time step t_u we need to compute the continuation values at the grid points

$$\begin{aligned}
\mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x_n] &\approx \sum_{j=0}^{N} c_j^{u+1} \mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_k] \\
&= \sum_{j=0}^{N} \frac{2^{\mathbb{1}_{0<j<N}}}{N} \sum_{k=0}^{N}{}'' V_{t_{u+1}}(x_k) T_j(z_k) \mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_k] \\
&= \sum_{k=0}^{N}{}'' V_{t_{u+1}}(x_k) \Big(\sum_{j=0}^{N} \frac{2^{\mathbb{1}_{0<j<N}}}{N} T_j(z_k) \mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_n] \Big) \\
&= \sum_{k=0}^{N} V_{t_{u+1}}(x_k) w_k^n
\end{aligned}$$

for weights

$$w_k^n = \frac{2^{\mathbb{1}_{0<k<N}}}{N} \sum_{j=0}^{N}{}'' T_j(z_k) \mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_n]. \tag{5.2}$$

The weights w_k^n can be calculated in the pre-computation step and in the time stepping one only has to calculate

$$V_{t_u}(x_k) \approx \max\Big\{g(x_k), \sum_{k=0}^{N} V_{t_{u+1}}(x_k) w_k^n\Big\}.$$

Every time step of the dynamic Chebyshev method can thus be seen as a numerical quadrature rule of the value function V_{t_u} with respect to the density of the underlying risk factor. The weights w_k^n form a linear operator in $\mathbb{R}^{(N+1)\times(N+1)}$ that maps the vector of nodal values at t_{u+1} onto the vector of nodal values at time point t_u. This simplification of the dynamic Chebyshev algorithm can be extended to higher dimensions. We will provide a bivariate version explicitly and then state a general d-variate version.

Bivariate dynamic Chebyshev method:
Consider a bivariate dynamic programming problem with value function $(x_1, x_2) \mapsto V_{t_u}(x_1, x_2)$ and a stochastic process (X_t^1, X_t^2). Let $\sum_{j_1}\sum_{j_2} c_{j_1,j_2}^{u+1} p_{j_1} p_{j_2}$ be the Chebyshev interpolation of $V_{t_{u+1}}$. Then the continuation value at t_u can be written as

$$\begin{aligned}
&\mathbb{E}\Big[V_{t_{u+1}}(X_{t_{u+1}}^1, X_{t_{u+1}}^2)|X_{t_u}^1 = x_{n_1}\, X_{t_u}^2 = x_{n_2}\Big] \\
&\quad\approx \sum_{j_1=0}^{N_1}\sum_{j_2=0}^{N_2} c_{j_1,j_2}^{u+1}\mathbb{E}\Big[p_{j_1,j_2}(X_{t_{u+1}}^1, X_{t_{u+1}}^2)|X_{t_u}^1 = x_{n_1}\, X_{t_u}^2 = x_{n_2}\Big] \\
&\quad= \sum_{j_1=0}^{N_1}\sum_{j_2=0}^{N_2}\Big(\frac{2^{\mathbb{1}_{0<j_1<N_1}}}{N_1}\frac{2^{\mathbb{1}_{0<j_2<N_2}}}{N_2}\sum_{k_1=0}^{N_1}{}''\sum_{k_2=0}^{N_2}{}'' V_{t_{u+1}}(x_{k_1}, x_{k_2}) T_{j_1,j_2}(z_{k_1}, z_{k_2})\Big) \\
&\qquad\qquad\cdot \mathbb{E}\Big[p_{j_1,j_2}(X_{t_{u+1}}^1, X_{t_{u+1}}^2)|X_{t_u}^1 = x_{n_1}\, X_{t_u}^2 = x_{n_2}\Big] \\
&\quad= \sum_{k_1=0}^{N_1}\sum_{k_2=0}^{N_2} V_{t_{u+1}}(x_{k_1}, x_{k_2}) w_{k_1,k_2}^{n_1,n_2}
\end{aligned}$$

with weights

$$w_{k_1,k_2}^{n_1,n_2} = \frac{2^{\mathbb{1}_{0<k_1<N_1}+\mathbb{1}_{0<k_2<N_2}}}{N_1 N_2}\sum_{j_1=0}^{N_1}{}''\sum_{j_2=0}^{N_2}{}'' T_{j_1,j_2}(z_{k_1}, z_{k_2})\mathbb{E}\Big[p_{j_1,j_2}(X_{t+1}^1, X_{t+1}^2)|X_t^1 = x_{n_1}\, X_t^2 = x_{n_2}\Big]$$

If $X_{t_{u+1}}^1$, $X_{t_{u+1}}^2$ are conditionally independent, we can use that $p_{j_1,j_2} = p_{j_1} p_{j_2}$ and we obtain

$$\begin{aligned}
&\mathbb{E}\Big[p_{j_1,j_2}(X_{t_{u+1}}^1, X_{t_{u+1}}^2)|X_{t_u}^1 = x_{n_1}\, X_{t_u}^2 = x_{n_2}\Big] \\
&\quad= \mathbb{E}\Big[p_{j_1}(X_{t_{u+1}}^1)|X_{t_u}^1 = x_{n_1}\Big]\mathbb{E}\Big[p_{j_2}(X_{t_{u+1}}^2)|X_{t_u}^2 = x_{n_2}\Big].
\end{aligned}$$

Together with $T_{j_1,j_2} = T_{j_1} T_{j_2}$ this yields $w_{k_1,k_2}^{n_1,n_2} = w_{k_1}^{n_1} w_{k_2}^{n_2}$ where $w_{k_1}^{n_1}$ and $w_{k_2}^{n_2}$ are one-

dimensional weights as defined in (5.2). Overall, the continuation value and thus be written as

$$\mathbb{E}\Big[V_{t_{u+1}}(X^1_{t_{u+1}}, X^2_{t_{u+1}})|X^1_{t_u} = x_{n_1}\, X^2_{t_u} = x_{n_2}\Big] \approx \sum_{k_1=0}^{N_1}\sum_{k_2=0}^{N_2} V_{t_{u+1}}(x_{k_1}, x_{k_2}) w^{n_1}_{k_1} w^{n_2}_{k_2}. \quad (5.3)$$

The bivariate version of this algorithm can be written in matrix form and allows an efficient implementation in *Matlab*.

Multivariate dynamic Chebyshev method:
Consider a d-variate dynamic programming problem with value function $V_t(\boldsymbol{x})$ and an $\mathbb{R}^d$-valued stochastic process $(\boldsymbol{X}_t)_{t\geq 0}$. Let $\sum_{\boldsymbol{j}} c^{u+1}_{\boldsymbol{j}} p_{\boldsymbol{j}}$ be the Chebyshev interpolation of $V_{t_{u+1}}$. We obtain

$$\begin{aligned}
&\mathbb{E}\Big[V_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_n}\Big] \approx \sum\nolimits_{\boldsymbol{j}} c^{u+1}_{\boldsymbol{j}} \mathbb{E}\big[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_n}\big] \\
&= \sum\nolimits_{\boldsymbol{j}} \Big(\Big(\prod_{i=1}^{d} \frac{2^{\mathbb{1}_{0<j_i<N_i}}}{N_i}\Big) \sum\nolimits_{\boldsymbol{k}}'' V_{t_{u+1}}(\boldsymbol{x_k}) T_{\boldsymbol{j}}(\boldsymbol{z_k})\Big) \mathbb{E}\big[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_n}\big] \\
&= \sum\nolimits_{\boldsymbol{k}} V_{t_{u+1}}(\boldsymbol{x_k}) \boldsymbol{w^n_k}
\end{aligned}$$

with weights

$$\boldsymbol{w^n_k} = \Big(\prod_{i=1}^{d} \frac{2^{\mathbb{1}_{0<k_i<N_i}}}{N_i}\Big) \sum\nolimits_{\boldsymbol{j}}'' T_{\boldsymbol{j}}(\boldsymbol{z_k}) \mathbb{E}\big[p_{\boldsymbol{j}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x_n}\big].$$

Similarly to the bivariate case we can write the weights as a product of d one-dimensional weights $\boldsymbol{w^n_k} = w^{n_1}_{k_1} \cdots w^{n_d}_{k_d}$ if $\boldsymbol{X}_{t_{u+1}}$ conditioned on $\boldsymbol{X}_{t_u}$ is a vector of independent random variables.

Numerical example: Bivariate dynamic Chebyshev

We compare the bivariate dynamic Chebyshev algorithm without computation of coefficients to the normal dynamic Chebyshev algorithm. For this we use the pricing of a bivariate basket option in a two-dimensional Black-Scholes model. We use the same specifications as for the numerical example in Section 5.1.3. We run both methods for an increasing number of Chebyshev nodes N with $N = 16, 32, 64, 128$ and compare the runtimes of both approaches.

Figure 5.2 shows the runtime of the modified algorithm divided by the runtime of the normal dynamic Chebyshev algorithm. We observe that the modification is a significant improvement to the algorithm and this effect becomes more important for a higher number of nodal points. This confirms the theoretical prediction that the runtime of the

backward induction is approximately halved. Note, that the price differences between the two approaches is below 10^{-11}, i.e. meaning the two methods produce the same result apart from rounding errors.

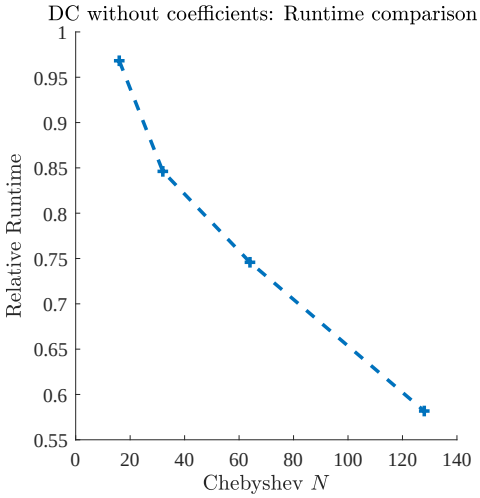

Figure 5.2: Comparison of the the bivariate dynamic Chebyshev algorithm without coefficients with the normal bivariate dynamic Chebyshev method for an increasing number of nodal points.

5.2 Pricing of callable bonds

In this section, we investigate one application of the (bivariate) dynamic Chebyshev method in detail, the pricing of callable bonds. A callable bond is a fixed income derivative that combines a vanilla bond with an embedded option. The bond issuer has the right to call the bond before maturity at a pre-specified price. The value of such a callable bond is driven by interest rates and default risk. The resulting pricing problem can be formulated as a bivariate dynamic programming problem in a two factor rate/credit model.

The goal of this section is to tailor the dynamic Chebyshev method to this pricing problem and provide empirical evidence of its benefits. We start with a short motivation and discuss the practical relevance of callable bonds. Then, we introduce a two-factor rate/credit model and we formulate the pricing problem as a dynamic programming problem. We present a dynamic Chebyshev algorithm for this problem and explain how this algorithm can be used for the model calibration. We conclude the section with an empirical analysis of the convergence behaviour of the pricing algorithm and the stability of the sensitivities.

This section is based on work that the author has conducted during an internship at

HSBC Bank plc in the credit quantitative analytics team of Mikhail Soloveitchik. The presented material has been developed by the author and does not contain any sensitive information or data. It has been approved for use in academic research and education. All numerical experiments have been re-run by the author.

5.2.1 Callable defaultable bonds

Callable corporate bonds belong to the most important fixed income instruments in financial markets. In the United States alone, over 70% of the 1.4 trillion USD of corporate bond issuance in 2019 were callable bonds. See the data from the Securities Industry and Financial Markets Association presented in Figure 5.3. This is a tremendous increase compared to the late nineties, when the yearly US corporate bond issuance had a volume below 600 billion USD and was dominated by non-callable bonds.

Callable debt offers companies additional financial flexibility by allowing them to rebuy their own bonds if interest rates are low enough. The bond issuer will call the bonds when it is cheaper for him to refinance via the issue of a new bond due to lower interest rates or a lower credit spread. For companies, it can be seen as a hedge against falling interest rates. Due to the additional optionality for the bond issuer, the callable bond must be cheaper from an investor's point of view. Therefore, callable bonds can offer higher returns than non-callable bonds.

Essentially, buying a callable bond is equivalent to buying a non-callable bond with the same coupon payments and selling an option on this bond back to the bond issuer. Usually, the issuer has the right to terminate the contract at one of several predefined dates (exercise dates) prior to maturity and pay the strike price instead of all outstanding cash-flows. This means that the embedded option is typically of Bermudan type. There exits two different classes of models for callable bonds, structural models and reduced form models. Structural models look at company specific quantities such as a firm's asset and liability structure and take this as a starting point for pricing. The main challenge here is the availability of information. The reduced form models use market information to calibrate a stochastic short rate model and price the callable bond in this model. We will focus on these models.

Since the late seventies, different numerical approaches for pricing callable bonds have been developed. For an early work on the pricing of callable bonds in a PDE framework see [20]. They propose to use finite-differences to solve the pricing PDE in a one-dimensional Gaussian short rate model. The default risk of the bond is ignored. Based on [20], different modifications have been proposed that tackled stability issues, improved performance and extended the approach to different short rate models. For

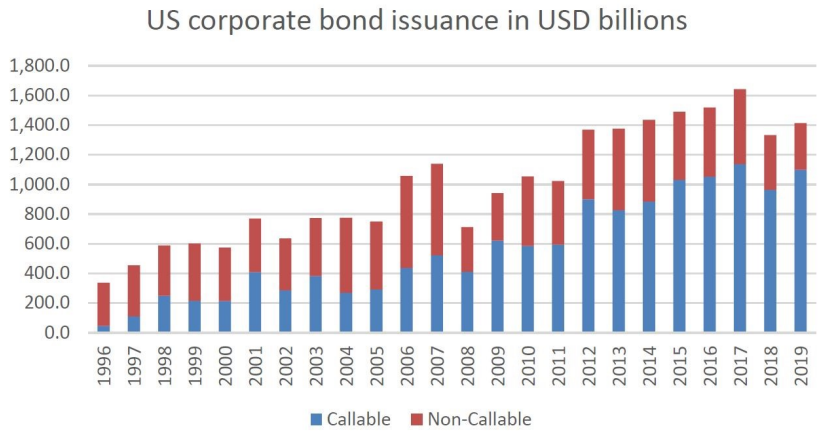

Figure 5.3: Volume of US corporate bond issues from 1996 to 2019 measured in billion US dollar. Source: Securities Industry and Financial Markets Association, research report "US corporate bond issuance" from March 9, 2020.

example [34] present an accurate numerical PDE method for the pricing of callable bonds.

As an alternative to the PDE approaches, [13] formulate the pricing problem as a dynamic programming problem and solve the backward induction using piecewise linear interpolation. More recently, [87] proposed an approach based on the eigenfunction expansion of the pricing operators that is able to handle short rate models with jumps and pure jump models. All these approaches suffer from one major limitation, the restriction to one-dimensional short rate models. Considering only one risk factor means that the correlation between risk-free interest rates and credit spreads cannot be captured. Moreover, it limits the ability to calibrate the model to market data.

[38] propose to use a two-factor model for pricing defaultable bonds where the default probability is modelled via a stochastic hazard rate (or credit intensity). Using two separate processes for the risk-free interest rate and the credit intensity allows for more accurate calibration to market instruments and correlation can be incorporated into the model. Such a two-factor model is also the right framework for the pricing of callable defaultable bonds. Unfortunately, pricing an early-exercise product in such a two-factor model is challenging and straightforward numerical PDE approaches are often not feasible.

Relatively few literature exists that considers the pricing of callable bonds in such a two factor model. Recently, [118] proposed a dynamic programming approach based

on Clenshaw-Curtis quadrature in a two-factor Vasicek model. The reported runtimes of around 60s seem however to be too slow for practical purposes. As an alternative, [77] propose to use separate processes for the default intensity and the call intensity and come up with a closed-form approximation for the callable bond price under some simplifications. The additional call intensity process is calibrated to a callable bond and can then be used to price a second callable bond from the same issuer. Thus, this method works only if at least one callable bond price is already available for the model calibration.

Therefore, we suggest to apply the bivariate dynamic Chebyshev method as presented in Section 5.1.2 for the pricing of callable bonds. We will use a two-factor model where the interest rate is normally distributed and the credit intensity is log-normally distributed as presented in [133]. This model can be calibrated to the risk-free yield curve and the credit spreads observed in this market. From a practical point of view, a suitable numerical pricing method should be capable of calibrating the underlying two-factor model efficiently to instruments quoted in the market, compute the callable bond price accurately and produce stable sensitivities with respect to the market instruments. The sensitivities are relevant for hedging purposes as well as for the risk management of the trading book. We will show that the bivariate dynamic Chebyshev method is able to fulfil these requirements.

5.2.2 Problem formulation

Here, we describe the pricing problem of a callable bond in the presence of credit risk in a two-factor rate/credit model. First we introduce the instrument and the corresponding two-factor model. Then we are in a position to formulate the actual pricing problem.

A callable bond

We focus on defaultable corporate bonds that pay a fixed coupon rate. For such a (callable) bond with maturity T we introduce the following notation

- coupon payment dates $0 = T_0 < T_1 < \ldots < T_{n_C} = T$,
- coupon payments $C_i = cN(T_i - T_{i-1})$, $i = 1, \ldots, n_C$, where c is the coupon rate and N is the notional of the bond,
- exercise dates $t_1 < \ldots < t_m = T$ and strike prices $K(\tau)$

In our examples, we will ignore the possibility of a notice period before calling the bond.

The two-factor model

For the pricing of a callable bond we introduce a two-factor hybrid model for the interest rate and the hazard rate of the credit intensity. We consider a two-factor model where the short rate $r(t)$ is modelled by a Hull-White model (see [71]) and the hazard rate $\lambda(t)$ follows the model of [16]. In the Hull-White model, the short rate is normally distributed and negative interest rates are possible. This has long be seen as a weakness of the model but turned out to be an advantage in the current interest rate environment. The process in the Black-Karasinski model is log-normally distributed and it is ensured that the hazard rate is always positive.

We divide the short rate and the hazard rate into a deterministic and a stochastic part modelled by a generic Ornstein–Uhlenbeck process. More precisely, we write

$$\begin{aligned} r(t) &= r(t, x_r(t)) = \phi_r(t) + x_r(t) \\ \lambda(t) &= \lambda(t, x_\lambda(t)) = e^{\phi_\lambda(t)+x_\lambda(t)} \end{aligned}$$

for deterministic functions ϕ_r, ϕ_λ and stochastic processes x_r, x_λ modelled by

$$\begin{aligned} \mathrm{d}x_r(t) &= -a_r x_r(t)\mathrm{d}t + \sigma_r \mathrm{d}W_t^r, \qquad x_r(0) = 0 \\ \mathrm{d}x_\lambda(t) &= -a_\lambda x_\lambda(t)\mathrm{d}t + \sigma_\lambda \mathrm{d}W_t^\lambda, \quad x_\lambda(0) = 0 \\ \mathrm{d}W_t^r \mathrm{d}W_t^\lambda &= \varrho \mathrm{d}t \end{aligned} \tag{5.4}$$

The functions ϕ_r, ϕ_λ are later used to fit the term structure implied by the market. For more details on the Hull-White and the Black-Karasinski model see [21] and we refer to [133] for a slightly different notation of the same two-factor model.

The stochastic process x_r (and therefore also x_λ) is normally distributed with

$$\begin{aligned} &\mathbb{E}[x_r(t)|x_r(s) = x_r] = x_r e^{-a_r(t-s)}, \\ &\sigma_r^2(t-s) := Var[x_r(t)|x_r(s) = x_r] = \frac{\sigma_r^2}{2a_r}\big(1 - e^{-2a_r(t-s)}\big), \end{aligned}$$

see [21]. For the two dimensional process $(x_r(t), x_\lambda(t))$ holds for $t > s$

$$\begin{pmatrix} x_r(t) \\ x_\lambda(t) \end{pmatrix} \Big| \begin{pmatrix} x_r(s) = x_r \\ x_\lambda(s) = x_\lambda \end{pmatrix} \sim \mathcal{N}_2\Big(\mu^{t-s} \begin{pmatrix} x_r \\ x_\lambda \end{pmatrix}, \Sigma^{t-s}\Big)$$

$$\mu^{t-s} = \begin{pmatrix} e^{-a_r(t-s)} & 0 \\ 0 & e^{-a_\lambda(t-s)} \end{pmatrix}, \quad \Sigma^{t-s} = \begin{pmatrix} \sigma_r^2(t-s) & \varrho\sigma_r(t-s)\sigma_\lambda(t-s) \\ \varrho\sigma_r(t-s)\sigma_\lambda(t-s) & \sigma_\lambda^2(t-s) \end{pmatrix}.$$

In order to simplify the notation we define the bivariate process $\boldsymbol{x}(t) := (x_r(t), x_\lambda(t))$ and $\boldsymbol{x} = (x_r, x_\lambda)$. As usually, if we write a variable in bold we refer to a multivariate (in this section bivariate) vector.

Calibration of the two-factor model

The merit of the presented version of the Hull-White and Black-Karasinski model is that the functions ϕ_r and ϕ_λ can be calibrated to match the term structure implied by the market. The interest rate term structure ϕ_r is calibrated to discount factors obtained from the risk-free yield curve. This is independent of the credit intensity model and results in a one-dimensional calibration problem.

The hazard rate term structure ϕ_λ is calibrated to CDS (Credit Default Swap) par rates implied by the markets. Note that par rate or swap rate of CDS instrument is the fixed interest rate for which the price of the CDS is zero, see [21] for more details. Either they are directly observed or implied by the credit curve generated from market quotes. A CDS is an insurance for credit risk and protects against the default of a bond issuer. The protection buyer pays periodically a premium of rate times notional and is in turn protected in case of default. The price of such an instrument is driven by interest rate risk as well as credit risk. The calibration of the hazard rate is therefore a bivariate problem. The price of a CDS instrument with maturity T, rate $R_{0,T}$ and notional 1 is given by

$$CDS(R_{0,T}) = \sum_{j=1}^{n} \mathbb{E}\big[D(T_j)\mathbb{1}_{\{\tau > T_j\}}\big](T_j - T_{j-1})R_{0,T} \;-\; LGD\mathbb{E}\big[\mathbb{1}_{\{0<\tau\leq T\}}D(\tau)\big]$$

where τ is the default time, $T_1, \ldots, T_n$ are the payment dates of the CDS with $T_0 = 0$ and $D(T_j)$ is the stochastic discount factor. The default time τ is usually defined as the first jump time of a Poisson process with (stochastic) intensity, the so-called hazard rate $\lambda(t)$ and it holds

$$\mathbb{Q}(\tau > t) = \mathbb{E}\Big[\exp\Big(-\int_0^t \lambda(s)\mathrm{d}s\Big)\Big],$$

see [21] for more details. The loss given default (LGD) in the CDS formula equals 1 minus the recovery rate of the bond. If a bond has a recovery rate of 40%, the CDS offers protection for the remaining 60% of the notional value. The first part of the CDS formula is called the premium leg of the CDS and the second part is called the protection leg. We call $R_{0,T}$ the par rate or the swap rate of the CDS is $CDS(R_{0,T}) = 0$. For more details on hazard rates, default probabilities and CDS swaps we refer again to [21].

In our numerical examples, we assume that the speed of mean reversion a_r, a_λ as well as the volatilities σ_r, σ_λ and the correlation ϱ are given. This means that only the term structures ϕ_r and ϕ_λ have to be calibrated. In practice, σ_r can be relatively easily obtained from interest rate options such as caps and floors. For the credit volatility σ_λ there might not be suitable market data and it has to be computed from historical data. Similarly, the correlation parameter ϱ is often computed using historical data. The speed of mean reversion has a relatively small influence and can just be fixed.

Formulation of the pricing problem

We start with the valuation of a non-callable defaultable bond with maturity T. The price a defaultable bond is computed as the discounted value of all cash-flows where the discounting reflects the interest rate as well as the credit risk. For the moment, we assume that there is no recovery, i.e. the bond issuer loses the complete notional N in the event of a default. The value V_t^{Bond} of such a bond is then given by

$$V_t^{Bond} = \sum_{t \le T_i \le T} D^{risky}(t, T_i) C_i + D^{risky}(t, T) N$$

with risky discount factor

$$D^{risky}(s,t) = \mathbb{E}\big[Z(s,t) \big| \mathcal{F}_s \big] \quad \text{for} \quad Z(s,t) := \exp\Big(- \int_s^t \big(r(u) + \lambda(u) \big) \mathrm{d}u \Big),$$

see [38]. The pricing formula implies that we discount all cash-flows with a rate $r + \lambda$ that reflects the risk-free rate as well as the individual credit risk. Using this concept, we can write a CDS instrument as

$$CDS(R_{0,T}) = \sum_{j=1}^{n} \mathbb{E}[Z(0,T_j)](T_j - T_{j-1}) R_{0,T} \; - \; LGD \mathbb{E}\Big[\int_0^T \lambda(s) Z(0,s) \mathrm{d}s \Big]. \tag{5.5}$$

Additionally, we define the riskless discount factor

$$D(s,t) = \mathbb{E}\big[\exp\big(- \int_s^t r(u) \mathrm{d}u \big) \big| \mathcal{F}_s \big].$$

For a simpler notation we will write $Z(t)$ instead of $Z(0,t)$ and replace $D(0,t)$ by $D(t)$.

The price of a callable defaultable bond is given by a (discrete-time) optimal stopping problem. The bond issuer minimizes the (discounted expected) amount he needs to pay to the investor, i.e.

$$V(0) = \min_{\tau \in \mathcal{T}_{Ex}} \mathbb{E}\big[\sum_{T_i < \tau} Z(T_i) C_i + Z(\tau) \min\{ K(\tau), V^{Bond}(\tau) \} \big]$$

where $\mathcal{T}_{Ex}$ is the set of all stopping times which take values in the discrete set of exercise dates $\{t_1, \ldots, t_m\}$. This optimal stopping problem of a callable bond can be reformulated as a dynamic programming problem which can be solved via backward induction, see [38]. We obtain

$$\begin{aligned} V_T(\boldsymbol{x}) &= V_T^{Bond}(\boldsymbol{x}) \\ V_{t_u}(\boldsymbol{x}) &= \min\big\{K(t_u), \mathbb{E}_{t_u,\boldsymbol{x}}\big[Z(t_u,t_{u+1})V_{t_{u+1}}(\boldsymbol{x}(t_{u+1})) + \sum_{t_u \le T_i < t_{u+1}} Z(t_u,T_i)C_i\big]\big\} \\ &= \min\big\{K(t_u), \mathbb{E}_{t_u,\boldsymbol{x}}\big[Z(t_u,t_{u+1})V_{t_{u+1}}(\boldsymbol{x}(t_{u+1}))\big] + \sum_{t_u \le T_i < t_{u+1}} \mathbb{E}_{t_u,\boldsymbol{x}}\big[Z(t_u,T_i)C_i\big]\big\} \end{aligned}$$

where $\mathbb{E}_{t_u,\boldsymbol{x}}[\,\cdot\,]$ refers to the conditional expectation $\mathbb{E}[\,\cdot\,|\boldsymbol{x}(t_u) = \boldsymbol{x}]$. We refer to the time-stepping $0 = t_0 < \ldots < t_u < \ldots < t_n = T$ as the pricing grid. Each exercise date needs to be included in the grid, but in general, the pricing grid can be finer. If t_u is not an exercise date we set $K(t_u) = \infty$.

5.2.3 A new pricing algorithm for callable bonds

In this section, we introduce the dynamic Chebyshev method for callable bonds. For the moment, we will assume that the bond pays no coupons and there is no recovery. In this case, the pricing equation is as follows

$$\begin{aligned} V_{t_u}(\boldsymbol{x}) &= \min\big\{K(t_u), \mathbb{E}\big[Z(t_u,t_{u+1})V_{t_{u+1}}(\boldsymbol{x}(t_{u+1}))\big|\boldsymbol{x}(t_u) = \boldsymbol{x}\big]\big\} \\ \boldsymbol{x}(t_{u+1})|\boldsymbol{x}(t_u) = \boldsymbol{x} &\sim \mathcal{N}_2(\mu^{\Delta t}\boldsymbol{x}, \Sigma^{\Delta t}) \end{aligned}$$

where $\Delta = t_{u+1} - t_u$. This pricing problem fits into the general scope of the dynamic Chebyshev algorithm and we can apply similar ideas as presented in Section 5.1 for a general d-dimensional framework. In contrast to equity options in a multivariate Black-Scholes model, the discount factor $Z(t_u, t_{u+1})$ is stochastic. Moreover, the starting value $\boldsymbol{x}$ is not just added to the drift but multiplied by the matrix $\mu^{\Delta t}$. We will see that this is an additional challenge in the case that interest rate and hazard rate are correlated. Adding coupon payments and a non-zero recovery rate poses a further challenge. On the other hand, the terminal payoff of the (callable) bond is constant and there is no kink in the payoff function that has to be smoothed in the first time step. This will improve the overall convergence of the method.

Idea of the algorithm

We apply the bivariate dynamic Chebyshev algorithm and approximate the value function $V_{t_u}(\boldsymbol{x})$ of a callable bond with Chebyshev polynomials in every time step t_u. Assume we have an approximation of $V_{t_{u+1}} \approx \sum_{\boldsymbol{j}} c_{\boldsymbol{j}}^{u+1} p_{\boldsymbol{j}}$, for multi-index $\boldsymbol{j} = (j_r, j_\lambda)$, coefficient

$c_{\boldsymbol{j}}^{u+1}$ and bivariate polynomials $p_{\boldsymbol{j}}$. At time t_u, we need to compute the function values on a grid of nodal points $\boldsymbol{x_k} = (x_{k_r}, x_{k_\lambda})$ in order to approximate V_{t_u}, i.e.

$$\widehat{V}_{t_u}(\boldsymbol{k}) := \min\big\{K(t_u), \sum\nolimits_{\boldsymbol{j}} c_{\boldsymbol{j}}^{u+1} \mathbb{E}_{t_u,\boldsymbol{x_k}}\big[Z(t_u,t_{u+1}) p_{\boldsymbol{j}}(\boldsymbol{x}(t_{u+1}))\big]\big\},$$

$\boldsymbol{k} = (k_r, k_\lambda)$. Here, the random variable $Z(t_u, t_{u+1})$ depends on the values of the bivariate stochastic process $\boldsymbol{x}(t)$ for $t \in [t_u, t_{u+1}]$. This path-dependency makes a direct computation of the conditional expectation computationally demanding. If the time discretization is fine enough, the variable $Z(t_u, t_{u+1})$ can be approximated as constant on the time interval $[t_u, t_{u+1}]$. More precisely, we can approximate the conditional expectation by

$$\begin{aligned}
\mathbb{E}_{t_u,\boldsymbol{x_k}}\big[Z(t_u,t_{u+1}) p_{\boldsymbol{j}}(\boldsymbol{x}(t_{u+1}))\big] &= \mathbb{E}_{t_u,\boldsymbol{x_k}}\big[\exp\big(-\int_{t_u}^{t_{u+1}} r(s)+\lambda(s)\mathrm{d}s\big) p_{\boldsymbol{j}}(\boldsymbol{x}(t_{u+1}))\big] \\
&\approx \mathbb{E}_{t_u,\boldsymbol{x_k}}\big[\exp\big(-\Delta t(r(t_u)+\lambda(t_u))\big) p_{\boldsymbol{j}}(\boldsymbol{x}(t_{u+1}))\big] \\
&\approx \exp\big(-\Delta t(r(t_u,x_{k_r})+\lambda(t_u,x_{k_\lambda}))\big)\mathbb{E}_{t_u,\boldsymbol{x_k}}[p_{\boldsymbol{j}}(\boldsymbol{x}(t_{u+1}))]
\end{aligned}$$

where $\Delta t = t_{u+1} - t_u$ is the length of the time interval. We define the matrix of risky discount factors D_u^{risky} with entries

$$D_{u,k_r,k_\lambda}^{risky} = \exp\big(-\Delta t(r(t_u,x_{k_r})+\lambda(t_u,x_{k_\lambda}))\big).$$

Exploring this simplification, all stochastic information of the model is again hidden in the generalized moments

$$\Gamma_{\boldsymbol{j},\boldsymbol{k}}^u := \mathbb{E}_{t_u,\boldsymbol{x_k}}\big[p_{\boldsymbol{j}}(\boldsymbol{x}(t_{u+1}))\big] = \mathbb{E}_{t_u,x_{k_r},x_{k_\lambda}}\big[p_{\boldsymbol{j}}(x_r(t_{u+1}),x_\lambda(t_{u+1}))\big].$$

Since $\boldsymbol{x}(t_{u+1})$ is conditionally normally distributed, these generalized moments are available in closed form in the two-factor Hull-White/Black-Karasinski model.

Algorithm for uncorrelated processes

For the moment, we assume that the two processes x_r and x_λ are independent, i.e. $\varrho = 0$. In this case, the dynamic Chebyshev algorithm can be simplified, see Section 5.1.1. For the conditional expectations $\Gamma_{\boldsymbol{j},\boldsymbol{k}}^u$ holds

$$\Gamma_{\boldsymbol{j},\boldsymbol{k}}^u = \mathbb{E}_{t_u,x_{k_r},x_{k_\lambda}}\big[p_{\boldsymbol{j}}(x_r(t_{u+1}),x_\lambda(t_{u+1}))\big] = \mathbb{E}_{t_u,x_{k_r}}[p_{j_r}(x_r(t_{u+1}))]\mathbb{E}_{t_u,x_{k_\lambda}}[p_{j_\lambda}(x_\lambda(t_{u+1}))]$$

due to the independence of the processes and the fact that $x_r(t_{u+1})$ depends only on the starting value x_{k_r} and vice versa. Instead of computing the expectations of N^2 polynomials $p_{\boldsymbol{j}}$ for N^2 different starting values $\boldsymbol{x_k}$, we only compute the expectations of

N polynomials for N starting values in each dimension for each time step.

Moreover, if the coefficients $a_r, a_\lambda, \sigma_r, \sigma_\lambda$ are constant and we use an equidistant time stepping for the pricing, the pre-computation becomes independent of the number of time steps. We have

$$\Gamma^r_{j_r,k_r} := \mathbb{E}_{x_{k_r}}[p_j(x_r(\Delta t))] = \mathbb{E}_{t_u,x_{k_r}}[p_{j_r}(x_r(t_{u+1}))]$$

for all time steps t_u and similarly, we can define $\Gamma^\lambda_{j_\lambda,k_\lambda}$. For the matrix of risky discount factors holds

$$D^{risky}_{u,k_r,k_\lambda} = \exp(-\Delta t\, r(t_u, x_{k_r}))\exp(-\Delta t\, \lambda(t_u, x_{k_\lambda})) = D^{rate}_{u,k_r} D^{credit}_{u,k_\lambda}.$$

This means that the matrix D^{risky}_u has rank one (i.e. it can be written as the product of a column vector D^{rate}_u and a row vector D^{credit}_u) and we call the discount factor separable. Here, D^{rate}_u is exactly the riskless discount factor at t_u. Note that the discount factors are still time dependent because of the time-dependent term structures ϕ_r and ϕ_λ.

Adding coupon payments

Adding coupon payments into the presented pricing algorithm is straightforward. If there are coupon payment dates in the interval $[t_u, t_{u+1})$ we add their discounted value at time point t_u. If the time interval is small enough, we can again assume that the interest rate and the hazard rate are constant. This means we compute the value of the coupons $VC_{\boldsymbol{k}}$ at note $\boldsymbol{k}$ as

$$VC_{\boldsymbol{k}} = \sum_{t_u \le T_i < t_{u+1}} \mathbb{E}_{t_u,\boldsymbol{x_k}}\big[Z(t_u,T_i)\big]C_i \approx \sum_{t_u \le T_i < t_{u+1}} \exp\big(-(T_i - t_u)(r(t_u,x_{k_r}) + \lambda(t_u,x_{k_\lambda}))\big)C_i$$

at every time step t_u and add it to the continuation value. If the coupon dates are a subset of the pricing time grid this can be further simplified and we can ignore the discounting $Z(t_u, T_i)$ form T_i to t_u.

Risky callable bonds with recovery

So far, we only considered bonds without any recovery at default. In this section, we will add recovery to the pricing problem. We assume that in the event of a default, the bond owner will receive a recovery payment as percentage R of the notional N. We call R the recovery rate of the bond. At time t_u, we add the expected discounted value of

the default payment for the interval $[t_u, t_{u+1}]$. For the nodal point $\boldsymbol{x_k}$ we obtain

$$\begin{aligned} DefaultPayment_{u,\boldsymbol{k}} &= \mathbb{E}_{t_u,\boldsymbol{x_k}}\Big[\int_{t_u}^{t_{u+1}} \lambda(s)Z(t_u,s)RN\mathrm{d}s\Big] \\ &= RN\int_{t_u}^{t_{u+1}} \mathbb{E}_{t_u,\boldsymbol{x_k}}[\lambda(s)Z(t_u,s)]\mathrm{d}s \\ &\approx RN\int_{t_u}^{t_{u+1}} \mathbb{E}_{t_u,\boldsymbol{x_k}}[\lambda(t_u)Z(t_u,t_u+0.5\Delta t)]\mathrm{d}s \\ &= RN\Delta t\lambda(t_u,x_{k_\lambda})\exp\big(-0.5\Delta t\big(r(t_u,x_{k_r})+\lambda(t_u,x_{k_\lambda})\big)\big). \end{aligned}$$

Here, we assumed again that $\lambda(s)$ is piecewise constant on a small interval and we approximated $Z(t_u,s)$ on $[t_u,t_{u+1}]$ by $Z(t_u,t_u+0.5\Delta t)$.

In summary, the value of a defaultable bond at time point t_u is the sum of its discounted continuation value plus the discounted values of all coupons between t_u and the next time step t_{u+1} and the expectation of the discounted value at default over this time period. The discounted factor considers the interest rate risk as well as the hazard rate of the credit risk.

Choice of interpolation domain

Critical for the convergence behaviour of the algorithm is the choice of the interpolation domain. A small domain will typically lead to a significant truncation error and an underestimation of the bond price. In contrast, the larger the domain the more nodal points are required for the same accuracy. The optimal size depends on the volatility of the underlying risk factors and the aspired accuracy. We set

$$[\underline{x}_i,\overline{x}_i] = [\mu(x_0^i,0,T)-k\sigma(T),\mu(x_0^i,0,T)+k\sigma(T)] \quad x_T^i|x_0^i \sim \mathcal{N}(\mu(x_0^i,0,T),\sigma^2(T))$$

for $i = r,\lambda$. As mentioned in Section 5.1.2, setting $k = 4$ corresponds to a confidence interval of 99.994% and is sufficiently large for most applications. In order to ensure a significantly higher accuracy we need to increase k to 5 or 6. Since $x_r(0) = x_\lambda(0) = 0$, the drift $\mu(x_0^i,0,T)$ is zero and the interpolation domain is symmetric around the origin. For an even Chebyshev degree N_r, $x_r = 0$ will be the midpoint of the nodal points x_{k_r}, $k_r = 0,\ldots,N_r$.

Bivariate pricing algorithm

Here, we present the algorithmic structure in matrix form for a callable bond assuming that the two processes $x_r(t)$ and $x_\lambda(t)$ are uncorrelated and there are no coupon payments and zero recovery.

1. **Initialization of the algorithm:** Fix interpolation degree (N_r, N_λ), number of time steps n_T, and number of standard deviations k. Set domain

$$[\underline{x}_r, \overline{x}_r] = [-k\sigma_r(T), k\sigma_r(T)] \quad [\underline{x}_\lambda, \overline{x}_\lambda] = [-k\sigma_\lambda(T), k\sigma_\lambda(T)]$$

and define interpolation points

$$\begin{aligned}\boldsymbol{X}^r &= \frac{\underline{x}_r + \overline{x}_r}{2} + \frac{\overline{x}_r - \underline{x}_r}{2}\big(\cos(0\pi/N_r), \dots, \cos(N_r\pi/N_r)\big)^T,\\ \boldsymbol{X}^\lambda &= \frac{\underline{x}_\lambda + \overline{x}_\lambda}{2} + \frac{\overline{x}_\lambda - \underline{x}_\lambda}{2}\big(\cos(0\pi/N_\lambda), \dots, \cos(N_\lambda\pi/N_\lambda)\big)\end{aligned}$$

where ^{T} means that the vector is transposed. Define the time grid for the pricing with fixed step size $\Delta t = T/n_T$.

2. **Compute conditional expectations:** Fix drift and volatility

$$\begin{aligned}&(\tilde{\mu}_0^i, \dots, \mu_N^i)^T = \tau^{-1}_{[\underline{x}_i, \overline{x}_i]}\big(e^{-a_i\Delta t}\boldsymbol{X}^i\big),\\ &\tilde{\sigma}_i^2 = \Big(\frac{2}{\overline{x}_i - \underline{x}_i}\Big)^2\sigma_i(\Delta t)^2 = \Big(\frac{2}{\overline{x}_i - \underline{x}_i}\Big)^2\sigma_i^2\Big(\frac{1}{2a_i}\big(1 - e^{-2a_i\Delta t}\big)\Big)\end{aligned}$$

and compute the matrix of conditional expectations

$$\Gamma^i = \Big(\mathbb{E}[T_j(Y_k)\mathbb{1}_{[-1,1]}(Y_k)]\Big)_{j,k} \quad Y_k \sim \mathcal{N}(\tilde{\mu}_k^i, \tilde{\sigma}_i^2),$$

for $i = r, \lambda$ using formula (3.17).

3. **Compute auxiliary matrices:**

$$\mathcal{T}_i = \frac{2}{N_i}\begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \dots & \frac{1}{2} & \frac{1}{4}\\ \frac{1}{2} & \cos(\frac{\pi}{N_i}) & \dots & \cos(\frac{\pi(N_i-1)}{N_i}) & \frac{1}{2}\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \frac{1}{4} & \frac{1}{2}\cos(\pi) & \dots & \frac{1}{2}\cos(\pi(N_i-1)) & \frac{1}{4}\cos(\pi N_i)\end{pmatrix} \quad i = r, \lambda.$$

Omit calculation of coefficients: If we are not interested in the coefficients, compute

$$\widehat{\Gamma}^r = (\Gamma^r)^T\mathcal{T}_r, \qquad \widehat{\Gamma}^\lambda = \mathcal{T}_r\Gamma^\lambda.$$

4. **First time step:** At maturity, set $CV_{t_{n_T-1}} = \widehat{V}_T$, the notional of the bond.

5. **Time stepping:** $t_{u+1} \to t_u$

   – **Compute discount factors:**

   $$D_u^{risky} = D_u^{rate} D_u^{credit}$$
   $$\text{with} \quad D^{rate} = \exp(-\Delta t r(t_u, \boldsymbol{X}^r)), \quad D^{credit} = \exp(-\Delta t \lambda(t_u, \boldsymbol{X}^\lambda))$$
   $$\text{where } r(t_u, x_r) = \phi_r(t_u) + x_r \text{ and } \lambda(t_u, x_\lambda) = \exp(\phi_\lambda(t_u) + x_\lambda).$$

   – **Update nodal values:**

   $$\widehat{V}_{t_u} = \begin{cases} \min\{K(t_u), D_u^{risky} * CV_{t_u}\}, & \text{if bond is callable at } t_u \\ D_u^{risky} * CV_{t_u} & \text{otherwise} \end{cases}$$

   where $*$ refers to the point-wise multiplication.

   – **Compute new continuation values:**

   $$CV_{t_{u-1}} = \begin{cases} (\Gamma^r)^T C_{t_u} \Gamma^\lambda, & \text{for coefficient matrix } \; C_{t_u} = \mathcal{T}_r \widehat{V}_{t_u} \mathcal{T}_\lambda \\ \widehat{\Gamma}^r \widehat{V}_{t_u} \widehat{\Gamma}^\lambda, & \text{if computation of coefficients is omitted.} \end{cases}$$

6. **Final time step:** Compute D_0^{risky} and final nodal values $\widehat{V}_0 = D_0^{risky} * CV_0$.

7. **Compute price:** Get option value at starting point $(x_r(0), x_\lambda(0)) = (0, 0)$.

Algorithm for correlated processes

In the general case of correlated risk factors the algorithm becomes more complicated. We can no longer write the conditional expectation of a bivariate Chebyshev polynomial as the product of the expectations of two one-dimensional polynomials. We want to explore the ideas of Section 5.1.2 and find an appropriate linear transformation $S_{\Delta t}$ such that $\boldsymbol{x}(t_{u+1})|\boldsymbol{x}(t_u) = \boldsymbol{x}_0$ is equal in distribution to $S_{\Delta t}\boldsymbol{z}(t_{u+1})|\boldsymbol{z}(t_u) = \boldsymbol{z}_0$ with $\boldsymbol{z}_0 = S_{\Delta t}^{-1}\boldsymbol{x}_0$ and $\boldsymbol{z}(t_{u+1})$ is a vector of two independent processes.

Consider the normally distributed bivariate process $\boldsymbol{x}(t) := (x_r(t), x_\lambda(t))$ starting in $\boldsymbol{x}(0) = (x_r, x_\lambda)$. We can write

$$\boldsymbol{x}(t) = \begin{pmatrix} e^{-a_r(t)} & 0 \\ 0 & e^{-a_\lambda(t)} \end{pmatrix} \begin{pmatrix} x_r \\ x_\lambda \end{pmatrix} + \begin{pmatrix} \sigma_r(t) & 0 \\ \varrho\sigma_\lambda(t) & \sigma_\lambda(t)\sqrt{1-\varrho^2} \end{pmatrix} \boldsymbol{Z} \quad \text{for} \quad \boldsymbol{Z} \sim \mathcal{N}_2(\boldsymbol{0}, \boldsymbol{1}_2)$$

where $\mathbf{0}$ is the zero vector and $\mathbf{1}_2$ is the 2×2 identity matrix. Instead of interpolating in $\boldsymbol{x}(t)$ we want to interpolate in an independent variable. In order to do so, let us look at $\boldsymbol{x}(t_{u+1})|\boldsymbol{x}(t_u) = \boldsymbol{x}_0$ with $\boldsymbol{x}_0 = (x_r, x_\lambda)$ in more detail

$$\begin{aligned}
\boldsymbol{x}(t_{u+1}) &= \begin{pmatrix} e^{-a_r\Delta t} & 0 \\ 0 & e^{-a_\lambda\Delta t} \end{pmatrix} \begin{pmatrix} x_r \\ x_\lambda \end{pmatrix} + \begin{pmatrix} \sigma_r(\Delta t) & 0 \\ \varrho\sigma_\lambda(\Delta t) & \sigma_\lambda(\Delta t)\sqrt{1-\varrho^2} \end{pmatrix} \boldsymbol{Z} \\
&= \begin{pmatrix} e^{-a_r\Delta t} & 0 \\ 0 & e^{-a_\lambda\Delta t} \end{pmatrix} \begin{pmatrix} x_r \\ x_\lambda \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ \varrho\frac{\sigma_\lambda(\Delta t)}{\sigma_r(\Delta t)} & \sqrt{1-\varrho^2} \end{pmatrix} \begin{pmatrix} \sigma_r(\Delta t) & 0 \\ 0 & \sigma_\lambda(\Delta t) \end{pmatrix} \boldsymbol{Z} \\
&= S_{\Delta t}\left(S_{\Delta t}^{-1}\mu_{\Delta t} \begin{pmatrix} x_r \\ x_\lambda \end{pmatrix} + \sigma(\Delta t)\boldsymbol{Z} \right) \\
&= S_{\Delta t}\tilde{\boldsymbol{Z}} \quad \text{with} \quad \tilde{\boldsymbol{Z}} \sim \mathcal{N}_2\big(S_{\Delta t}^{-1}\mu_{\Delta t}\boldsymbol{x}_0, \sigma(\Delta t)\big)
\end{aligned}$$

for the matrices

$$S_{\Delta t} = \begin{pmatrix} 1 & 0 \\ \varrho\frac{\sigma_\lambda(\Delta t)}{\sigma_r(\Delta t)} & \sqrt{1-\varrho^2} \end{pmatrix}, \quad \mu_{\Delta t} = \begin{pmatrix} e^{-a_r\Delta t} & 0 \\ 0 & e^{-a_\lambda\Delta t} \end{pmatrix}, \quad \sigma_{\Delta t} = \begin{pmatrix} \sigma_r(\Delta t) & 0 \\ 0 & \sigma_\lambda(\Delta t) \end{pmatrix}.$$

The random variable $\tilde{\boldsymbol{Z}}$ corresponds to the conditional expectation $\boldsymbol{z}(t_{u+1})|\boldsymbol{z}(t_u) = \boldsymbol{z}_0$ with $z_0 = S_{\Delta t}^{-1}\boldsymbol{x}_0$ for a new independent process $\boldsymbol{z}(t) = (z_r(t), z_\lambda(t))$ given by (5.4) with $\varrho = 0$. Instead of working with the process $\boldsymbol{x}(t)$ we will work with the independent process $\boldsymbol{z}(t)$.

We define the transformed value function $\tilde{V}_{t_u}(\boldsymbol{z}) := V_{t_u}(S_{\Delta t}\boldsymbol{z})$. Assume we have a Chebyshev interpolation $\sum_{\boldsymbol{j}} c_{\boldsymbol{j}}^{u+1} p_{\boldsymbol{j}}$ of $\boldsymbol{z} \mapsto \tilde{V}_{t_{u+1}}(\boldsymbol{z})$ and want to obtain an interpolation of $\tilde{V}_{t_u}$. In order to do so, we require the nodal values $\tilde{V}_{t_u}(\boldsymbol{z}_{\boldsymbol{k}}) = V_{t_u}(S_{\Delta t}\boldsymbol{z}_{\boldsymbol{k}})$ and thus the conditional expectations

$$\begin{aligned}
\mathbb{E}\big[V_{t_{u+1}}(\boldsymbol{x}(t_{u+1}))|\boldsymbol{x}(t_u) = S_{\Delta t}\boldsymbol{z}_{\boldsymbol{k}}\big] &= \mathbb{E}\big[V_{t_{u+1}}(S_{\Delta t}\tilde{\boldsymbol{Z}})|\boldsymbol{x}(t_u) = S_{\Delta t}\boldsymbol{z}_{\boldsymbol{k}}\big] \\
&= \mathbb{E}\big[\tilde{V}_{t_{u+1}}(\tilde{\boldsymbol{Z}})|\boldsymbol{x}(t_u) = S_{\Delta t}\boldsymbol{z}_{\boldsymbol{k}}\big] \\
&= \mathbb{E}\big[\tilde{V}_{t_{u+1}}\big(S_{\Delta t}^{-1}\mu_{\Delta t}S_{\Delta t}\boldsymbol{z}_{\boldsymbol{k}} + \sigma(\Delta t)\boldsymbol{Z}\big)\big] \\
&= \sum\nolimits_{\boldsymbol{j}} c_{\boldsymbol{j}}^{u+1}\mathbb{E}\big[p_{\boldsymbol{j}}\big(S_{\Delta t}^{-1}\mu_{\Delta t}S_{\Delta t}\boldsymbol{z}_{\boldsymbol{k}} + \sigma(\Delta t)\boldsymbol{Z}\big)\big]
\end{aligned}$$

for $\boldsymbol{Z} \sim \mathcal{N}_2(\mathbf{0}, \mathbf{1}_2)$. Assume both processes x_r and x_λ have the same speed of mean

reversion, i.e. $a_r = a_\lambda$. In this case, the drift simplifies to

$$S_{\Delta t}^{-1}\mu_\Delta S_{\Delta t}\boldsymbol{z}_k = S_{\Delta t}^{-1}(e^{-a_r\Delta t}\mathbf{1}_2)S_{\Delta t}\boldsymbol{z}_k = e^{-a_r\Delta t}\mathbf{1}_2\boldsymbol{z}_k.$$

In this case, the conditional expectations of the bivariate Chebyshev polynomials can written as

$$\begin{aligned}&\mathbb{E}\big[p_{\boldsymbol{j}}\big(S_{\Delta t}^{-1}\mu_{\Delta t}S_{\Delta t}\boldsymbol{z_k} + \sigma(\Delta t)\boldsymbol{Z}\big)\big]\\ &\quad= \mathbb{E}\big[p_{j_r}\big(e^{-a_r\Delta t}z_{k_r} + \sigma_r(\Delta t)Z_r\big)\big]\mathbb{E}\big[p_{j_\lambda}\big(e^{-a_\lambda\Delta t}z_{k_\lambda} + \sigma_\lambda(\Delta t)Z_\lambda\big)\big]\end{aligned}$$

where Z_r, Z_λ are two independent standard normally distributed random variables. This means, the pre-computation step for the two correlated processes has the same complexity as in the case of zero correlation. The discount factor at the new nodal points $\boldsymbol{z_k}$ is given by $D_u^{risky} = D_u^{risky}((S_{\Delta t}\boldsymbol{z_k})_1, (S_{\Delta t}\boldsymbol{z_k})_2)$.

If the drift $S_{\Delta t}^{-1}\mu_\Delta S_{\Delta t}\boldsymbol{z}_k$ does not simplify to a diagonal matrix, the conditional expectations of p_{j_λ} will depend on both, z_{k_r} and z_{k_λ}. This is the case if either $a_r \neq a_\lambda$ or if the volatility becomes time-dependent.

Separability of discount factors

The problem with the presented transformation is that the new risky discount factor is not separable, i.e. the discount factor

$$\begin{aligned}D_u^{risky} &= \exp\Big(-\Delta t\big(\phi_r(t_u) + z_{k_r} + \exp(\phi_\lambda(t_u) + \varrho\frac{\sigma_\lambda}{\lambda_r}z_{k_r} + \sqrt{1-\varrho^2}z_{k_\lambda})\big)\Big)\\ &= \exp\Big(-\Delta t\big(\phi_r(t_u) + z_{k_r} + \exp(\varrho\frac{\sigma_\lambda}{\lambda_r}z_{k_r})\exp(\phi_\lambda(t_u) + \sqrt{1-\varrho^2}z_{k_\lambda})\big)\Big)\end{aligned}$$

cannot be written as $D_{u,k_r}^{rate}D_{u,k_\lambda}^{credit}$. Therefore, we have to modify our approach slightly.

Instead of using the process $\boldsymbol{x}(t) := (x_r(t), x_\lambda(t))$ we could apply the same linear transformation to the process $\boldsymbol{x}(t) := (x_\lambda(t), x_r(t))$ and obtain an independent process $\boldsymbol{z}(t) := (z_\lambda(t), z_r(t))$. In this case, the matrices become

$$S_{\Delta t} = \begin{pmatrix}1 & 0\\ \varrho\frac{\sigma_r(\Delta t)}{\sigma_\lambda(\Delta t)} & \sqrt{1-\varrho^2}\end{pmatrix},\quad \mu_{\Delta t} = \begin{pmatrix}e^{-a_\lambda\Delta t} & 0\\ 0 & e^{-a_r\Delta t}\end{pmatrix},\quad \sigma_{\Delta t} = \begin{pmatrix}\sigma_\lambda(\Delta t) & 0\\ 0 & \sigma_r(\Delta t)\end{pmatrix}.$$

We assume again that $a_\lambda = a_r$, thus $\frac{\sigma_r(\Delta t)}{\sigma_\lambda(\Delta t)} = \frac{\sigma_r}{\sigma_\lambda}$ and $S_{\Delta t}$ becomes independent of the length of the time-step Δt, i.e. $S_{\Delta t} = S$. The discount factor D_u^{risky} is separable with

$$D_u^{risky} = \exp\Big(-\Delta t\Big(\phi_r(t_u) + (S\boldsymbol{z_k})_2 + \exp(\phi_\lambda(t_u) + (S\boldsymbol{z_k})_1)\Big)\Big)$$

$$
\begin{aligned}
&= \exp\Big(- \Delta t\Big(\phi_r(t_u) + \varrho\frac{\sigma_r}{\sigma_\lambda} z_{k_\lambda} + \sqrt{1-\varrho^2} z_{k_r} + \exp(\phi_\lambda(t_u) + z_{k_\lambda})\Big)\Big) \\
&= \exp\Big(- \Delta t\Big(\varrho\frac{\sigma_r}{\sigma_\lambda} z_{k_\lambda} + \exp(\phi_\lambda(t_u) + z_{k_\lambda})\Big)\Big) \exp\Big(- \Delta t\Big(\phi_r(t_u) + \sqrt{1-\varrho^2} z_{k_r}\Big)\Big) \\
&= \exp\Big(- \Delta t\Big(\varrho\frac{\sigma_r}{\sigma_\lambda} z_{k_\lambda} + \lambda(t_u, z_{k_\lambda})\Big)\Big) \exp\Big(- \Delta t\, r(t_u, \sqrt{1-\varrho^2} z_{k_r})\Big).
\end{aligned}
$$

This means we can separate the terms which depend on z_{k_λ} from the terms depending on z_{k_r} and write $D_u^{risky} = D_{u,k_\lambda}^{credit} D_{u,k_r}^{rate}$.

5.2.4 Calibration of the rate/credit model

In this section, we describe the calibration of the two factor rate/credit model with the dynamic Chebyshev method.

For the calibration, we require CDS par rates plus riskless discount factors for the same maturities. We assume that the corresponding CDS are maturing in regular intervals, for example quarterly, and the premium is paid in the same frequency. Let $0 = T_0 < T_1 < \ldots < T_{n_{cds}}$ be the maturities with $T_i = iT_1$ and R_{0,T_i} the corresponding par rates. These quarterly par rates can be obtained via bootstrapping from the actual CDS par rates observed in the market. Let $D(T_i)$ be the riskless discount factors for $T_1, \ldots, T_{n_{cds}}$. For notational convenience we define $T_0 = 0$. We calibrate term structures as piecewise constant on $[T_{i-1}, T_i)$ and denote the value of ϕ_r on $[T_{i-1}, T_i)$ as $\phi_r(T_i)$. First we calibrate ϕ_r to the riskless discount factors and then ϕ_λ to the CDS par rates.

We start with a straightforward approach and point out its weaknesses. Then we introduce a second, more efficient approach that significantly reduces the number of matrix multiplications required for the calibration.

A straightforward calibration approach

The straightforward approach for calibrating the two-factor model is to call a numerical pricer for each discount factor and CDS and then to minimize the distance between the market price and the model price. For the interest rate term structure ϕ_r one starts with the first discount factor $D(T_1)$ and calibrates ϕ_r on $[T_0, T_1)$. In this case, the calibration is done by minimizing the distance between the discount factor and the price of a zero-coupon bond with maturity T_1 computed by the univariate dynamic Chebyshev method. For the minimization one can use a standard numerical optimization routine with an appropriate initial guess $\phi_r(T_1)$ and a target error tolerance. Having calibrated ϕ_r on the first interval, the next discount factor $D(T_2)$ can be calibrated and so on. For the

maturity T_i we obtain

$$D(T_i) = \mathbb{E}\Big[\exp\Big(-\int_0^{T_i} r(s, x_r(s))\mathrm{d}s\Big)\Big] = \mathbb{E}\Big[\prod_{j=1}^{i}\exp\Big(-\int_{T_{j-1}}^{T_j}\big(\phi_r(T_j) + x_r(s)\big)\mathrm{d}s\Big)\Big]$$

and we minimize the distance between the observed discount factor and the discount factor computed in our stochastic model over $\phi_r(T_i)$.

For the credit term structure, we use the bivariate dynamic Chebyshev method to price both, the protection leg and the premium leg of the CDS. Once the model is calibrated to the first par rate we continue with R_{0,T_2} and so on until we have calibrated the function ϕ_λ to all rates.

Similarly to the pricing of callable bonds with the dynamic Chebyshev method, we use an equidistant time grid with n_q steps on every interval $[T_i, T_{i+1}]$. For the first CDS instrument with pricing formula (5.5), the protection leg is approximated by

$$\begin{aligned}\mathbb{E}\Big[\int_0^{T_1}\lambda(s)Z(s)\mathrm{d}s\Big] &= \sum_{u=0}^{n_q-1}\mathbb{E}\Big[\int_{t_u}^{t_{u+1}}\lambda(s)Z(s)\mathrm{d}s\Big]\\ &\approx \sum_{u=0}^{n_q-1}\mathbb{E}\Big[\Delta t\lambda\big(\frac{t_u+t_{u+1}}{2}\big)Z\big(\frac{t_u+t_{u+1}}{2}\big)\Big]\\ &\approx \sum_{u=0}^{n_q-1}\Delta t\mathbb{E}\Big[Z(t_u)\lambda(t_u)\exp\big(-\frac{\Delta t}{2}(r(t_u, x_r(t_u)) + \lambda(t_u, x_\lambda(t_u)))\big)\Big].\end{aligned}$$

Here, we explored that ϕ_λ, ϕ_r are constant on $[0, T_1)$ and we approximated $x_r(t), x_\lambda(t)$ as constant on $[t_u, t_{u+1}]$. The expectations can be seen as present values of zero-coupon bonds with maturity t_u and terminal payoff $\lambda(t_u)\exp\big(-0.5\Delta t(r(t_u, x_r(t_u)) + \lambda(t_u, x_\lambda(t_u)))\big)$.

Similarly, the premium leg is $R_{0,T_1}T_1$ times the present value of a zero-coupon bond with maturity T_1, i.e. $\mathbb{E}[Z(T_1)]$. After calibrating the first CDS we can write the protection leg of the second CDS as

$$\mathbb{E}\Big[\int_0^{T_2}\lambda(s)Z(s)\mathrm{d}s\Big] = \mathbb{E}\Big[\int_0^{T_1}\lambda(s)Z(s)\mathrm{d}s\Big] + \sum_{u=n_q}^{2n_q-1}\mathbb{E}\Big[\int_{t_u}^{t_{u+1}}\lambda(s)Z(s)\mathrm{d}s\Big].$$

This implies that the protection leg of the second CDS is the protection leg of the first CDS plus n_q new terms. Thus, the calibration of all n_{cds} CDS instruments requires the

computation of $n_{cds}n_q$ terms

$$\mathbb{E}\Big[Z(t_u)\lambda(t_u)\exp\big(-\frac{\Delta t}{2}(r(t_u,x_r(t_u))+\lambda(t_u,x_\lambda(t_u)))\big)\Big]$$

and n_{cds} terms

$$\mathbb{E}\big[Z(T_i)\big]=\mathbb{E}\Big[\exp\Big(-\int_0^{T_i}\big(r(s,x_r(s))+\lambda(s,x_\lambda(s))\big)\mathrm{d}s\Big)\Big].$$

The drawback of this straightforward calibration approach is its computational complexity. For every discount factor we need to call a new dynamic Chebyshev pricer and no previously computed information is re-used. The number of matrix multiplications that have to be done grows proportionally to $\frac{n_{cds}(n_{cds}+1)}{2}$ times the number of time steps n_q. This inefficiency comes from the fact that the calibration is a forward scheme from T_1 to $T_{n_{cds}}$ but the pricing is then done via backward induction. In the following, we introduce a modified calibration routine which reduces the complexity to n_{cds}.

Efficient bond pricing

For both, the calibration of the rate as well as of the credit term structure, an efficient way to price zero coupon bonds (with and without credit risk) for different maturities is crucial. We start with the bivariate case of a zero coupon bond with credit risk (i.e. a risky discount factor) and derive the riskless case from this. Assume that the interest rate and the credit intensity are uncorrelated. For a zero coupon bond with credit risk every time step in the bivariate dynamic Chebyshev method looks as follows

$$\widehat{V}_{t_u}=D_u^{risky}*\widehat{\Gamma}_r\widehat{V}_{t_{u+1}}\widehat{\Gamma}_\lambda$$

and at maturity T, $\widehat{V}_T$ is a $(N_r+1)\times(N_\lambda+1)$ matrix of ones denoted by $\mathbf{1}_{N_r,N_\lambda}$. We recall that the discount factor D_u^{risky} is separable and the matrix D_u^{risky} can be written as the product of a column vector $D_u^{rate}=\exp(-\Delta t r(t_u,\boldsymbol{x}_r))$ and a row vector $D_u^{credit}=\exp(-\Delta t\lambda(t_u,\boldsymbol{x}_\lambda))$. We define new matrices

$$L_r=D_u^{rate}*\widehat{\Gamma}_r\quad\text{and}\quad L_\lambda=D_u^{credit}*\widehat{\Gamma}_\lambda$$

where the pointwise multiplication of a column vector with a matrix means that each matrix column is pointwise multiplied with the vector. For the time stepping of the dynamic Chebyshev method we obtain

$$\widehat{V}_{t_u}=L_r\widehat{V}_{t_{u+1}}L_\lambda.$$

The matrices L_r, L_λ only change if $\phi_r(t)$ or $\phi_\lambda(t)$ change. Hence, they are the same on any interval $[T_i, T_{i+1})$ and we write L_r^i and L_λ^i to indicate the interval. Let n_q be the number time-steps in the dynamic Chebyshev method in $[T_i, T_{i+1})$. The risky discount factor $D^{risky}(T_1) = \mathbb{E}[Z(T_1)]$ is then the price of the zero coupon bond at $t = 0$, i.e. the entry of $\widehat{V}_0$ where $x_\lambda(0) = x_r(0) = 0$, for an even Chebyshev degree N this is the s-th entry with $s = N/2 + 1$ in each dimension. We obtain this entry by multiplying $\widehat{V}_0$ with a (row) vector $\boldsymbol{e}_s$ from the left and $\boldsymbol{e}_s^T$ from the right, where $\boldsymbol{e}_s$ is one at entry s and zero otherwise, i.e.

$$D^{risky}(T_1) \approx \boldsymbol{e}_s (L_r^1)^{n_q} \mathbf{1}_{N_r,N_\lambda} (L_\lambda^1)^{n_q} \boldsymbol{e}_s^T.$$

The computed vectors $\boldsymbol{e}_s(L_r^1)^{n_q}$ and $(L_\lambda^1)^{n_q}\boldsymbol{e}_s^T$ can then be stored and used again for the next discount factor

$$D^{risky}(T_2) \approx \boldsymbol{e}_s (L_r^1)^{n_q} (L_r^2)^{n_q} \mathbf{1}_{N_r,N_\lambda} (L_\lambda^2)^{n_q} (L_\lambda^1)^{n_q} \boldsymbol{e}_s^T.$$

Similarly, we can calculate the riskless discount factors $D(T_i)$ using only the matrices L_r^i. The riskless discount factor is a zero coupon bond without credit risk and can be approximated by

$$D(T_i) \approx \boldsymbol{e}_s (L_r^1)^{n_q} \dots (L_r^i)^{n_q} \mathbf{1}^T$$

where $\mathbf{1}$ is a row vector of ones. Moreover, the vectors $\boldsymbol{e}_s(L_r^1)^{n_q} \dots (L_r^i)^{n_q}$ for $i = 1, \dots, n_{cds}$ can be computed once for the calibration of the interest rate term structure and then be re-used for the calibration of the credit intensity.

The modified calibration approach has two significant advantages. First, the number of matrix multiplications scales only linearly in n_{df} since the information computed for $D^{risky}(T_i)$ are re-used for $D^{risky}(T_{i+1})$. Second, the matrix power $L_r^{n_q}$ requires only $2\log_2(n_q)$ matrix multiplications instead of n_q if the idea of exponentiation by squaring is explored. For n_q even we can write $L_r^{n_q} = L_r^{n_q/2} L_r^{n_q/2}$ and for n_q odd we can write $L_r^{n_q} = L_r L_r^{(n_q-1)/2} L_r^{(n_q-1)/2}$. This procedure can then be repeated for $L_r^{n_q/2}$ or $L_r^{(n_q-1)/2}$ and so on.

If interest rate and hazard are correlated we use the transformations of the process $\boldsymbol{x}(t) = (x_\lambda(t), x_r(t))$ described in Section 5.2.3 and explore the separability of the discount factors

$$\begin{aligned} D_u^{risky} &= \exp\Big(-\Delta t \Big(\varrho \frac{\sigma_r}{\sigma_\lambda} z_{k_\lambda} + \exp(\phi_\lambda(t_u) + z_{k_\lambda}) \Big) \Big) \exp\Big(-\Delta t \Big(\phi_r(t_u) + \sqrt{1-\varrho^2} z_{k_r} \Big) \Big) \\ &= \exp\Big(-\Delta t \Big(\varrho \frac{\sigma_r}{\sigma_\lambda} z_{k_\lambda} + \lambda(t_u, z_{k_\lambda}) \Big) \Big) \exp\Big(-\Delta t\, r(t_u, \sqrt{1-\varrho^2} z_{k_r}) \Big). \end{aligned}$$

Note that the calibration of the rate term structure remains the same since it depends only on $x_r(t)$. This means, we can calibrate ϕ_r using the one-dimensional dynamic Chebyshev method, independently of the correlation between rate and credit.

Overall, the new scheme requires far less matrix multiplications and leads to a significant runtime reduction. Further speed-up can be achieved by using a different optimization routine or root-finder in the calibration of each discount factor. Exploring the monotonicity of the discount factors in $\phi_r(T_i)$ and $\phi_\lambda(T_i)$ can be beneficial. In our experiments, a root-finder based on a bisection or a secant method required significantly fewer steps than a standard minimization routine.

Efficient pricing of CDS instruments

Using the efficient pricing of defaultable zero coupon bonds, we immediately obtain a more efficient pricer for CDS instruments of different maturities. We recall that the premium leg is just a sum of risky bonds and the premium leg can also be seen as sum of bonds with a different payoff profile.

In the bivariate dynamic Chebyshev algorithm, the time step $t_{u+1} \to t_u$ in the absence of an early-exercise possibility or coupon payments is given by

$$\widehat{V}_{t_u} = D_u^{risky} * (\widehat{\Gamma}_\lambda \widehat{V}_{t_{u+1}} \widehat{\Gamma}_r) = L_\lambda^i \widehat{V}_{t_{u+1}} L_r^i$$

on the interval $[T_{i-1}, T_i]$. The matrices L_λ and L_r are defined as

$$L_\lambda^i = \exp\big(-\Delta t \exp(\phi_\lambda(T_i) + \boldsymbol{x}_\lambda)\big) * \widehat{\Gamma}_\lambda \text{ and } L_r^i = \widehat{\Gamma}_r * \exp\big(-\Delta t(\phi_r(T_i) + \boldsymbol{x}_r)\big)$$

if interest rate and credit intensity are uncorrelated. In the general case with correlation we obtain

$$\begin{aligned} L_\lambda^i &= \exp\big(-\Delta t(\varrho \frac{\sigma_r}{\sigma_\lambda} \boldsymbol{z}_\lambda + \exp(\phi_\lambda(T_i) + \boldsymbol{z}_\lambda)\big) * \widehat{\Gamma}_\lambda \\ L_r^i &= \widehat{\Gamma}_r * \exp\big(-\Delta t(\phi_r(T_i) + \sqrt{1-\varrho^2}\boldsymbol{z}_r)\big). \end{aligned}$$

Here $\boldsymbol{x}_\lambda, \boldsymbol{z}_\lambda$ are column vectors and $\boldsymbol{x}_r, \boldsymbol{z}_r$ are row vectors of the Chebyshev nodal points. Note that due to the linear transformation S, the matrix L_r^i in the two-factor model does not coincide with the matrix L_r^i used for the riskless discount factors.

The present value of the protection leg on $[t_{u-1}, t_u]$ in $[0, T_1]$ is then given by

$$\widehat{V}_0 = (L_\lambda^1)^{u-1} \tilde{\lambda}(T_1, \boldsymbol{x}_\lambda) D(T_1, \boldsymbol{x}_r) (L_r^1)^{u-1}$$

where $D(T_1, \boldsymbol{x}_r) = \exp(-0.5\Delta t(\phi_r(T_1) + \boldsymbol{x}_r))$ and

$$\tilde{\lambda}(T_1, \boldsymbol{x}_\lambda) = \lambda(T_1, \boldsymbol{x}_\lambda) * \exp(-0.5\Delta t \lambda(T_1, \boldsymbol{x}_\lambda)) \quad \text{with} \quad \lambda(T_1, \boldsymbol{x}_\lambda) = \exp(\phi_\lambda(T_1) + \boldsymbol{x}_\lambda).$$

The actual present value of the protection leg is then the entry of $\widehat{V}_0$ where $x_\lambda(0) = x_r(0) = 0$, given by

$$\widehat{V}_0(0) = \boldsymbol{e}_s(L_\lambda^1)^{u-1}\tilde{\lambda}(T_1, \boldsymbol{x}_\lambda)D(T_1, \boldsymbol{x}_r)(L_r^1)^{u-1}\boldsymbol{e}_s^T,$$

similarly to the value of zero coupon bond with maturity T_1 given by

$$\widehat{V}_0(0) = \boldsymbol{e}_s(L_\lambda^1)^{n_q}\mathbf{1}^T\mathbf{1}(L_r^1)^{n_q}\boldsymbol{e}_s^T$$

where $\mathbf{1}$ is a row vector of ones. Re-using the information computed for the first CDS instrument, the present value of the protection leg on $[t_{u-1}, t_u]$ in the interval $[T_1, T_2]$ is given by

$$\widehat{V}_0(0) = \boldsymbol{e}_s(L_\lambda^1)^{n_q}(L_\lambda^2)^{u-1}\tilde{\lambda}(T_2, \boldsymbol{x}_\lambda)D(T_2, \boldsymbol{x}_r)(L_r^2)^{u-1}(L_r^1)^{n_q}\boldsymbol{e}_s^T$$

and the value of a zero coupon bond with maturity T_2 is given by

$$\widehat{V}_0(0) = \boldsymbol{e}_s(L_\lambda^1)^{n_q}(L_\lambda^2)^{n_q}\mathbf{1}^T\mathbf{1}(L_r^2)^{n_q}(L_r^1)^{n_q}\boldsymbol{e}_s^T.$$

The vectors $\boldsymbol{e}_s(L_\lambda^1)^{n_q} \dots (L_\lambda^i)^{n_q}$ and $(L_r^i)^{n_q} \dots (L_r^1)^{n_q}\boldsymbol{e}_s^T$ can then be stored and used again for the next CDS instrument.

Similar to the case of riskless discount factors it is more efficient to use a root finder to calibrate $\phi_\lambda(T_j)$. Since we do not know the derivative of the CDS with respect to $\phi_\lambda(T_j)$ in closed from we cannot use Newton's method. However, the closely related secant method performed very well in our experiments. As an initial guess we used the hazard rate in the static (non-stochastic model) and a small shift of this value as a second initial guess. Moreover, it is more efficient to compute the root of $\exp(\phi_\lambda(T_j)) \mapsto CDS(R_{0,T_j})$ instead of the root of $\phi_\lambda(T_j) \mapsto CDS(R_{0,T_j})$. As an alternative, one could also use a bracketing method such as a version of the Brent-Dekker root-finding algorithm.

For an efficient implementation, the vectors $(L_r^j)^{u-1} \dots (L_r^1)^{n_q}\boldsymbol{e}_s^T$ for $u = 1, \dots, n_q + 1$ should be computed prior to the root-finder and can then be used in each iterative step of the root-finding algorithm. Their computation requires n_q matrix times vector multiplications for each CDS instrument.

Summary calibration routine

Next, we summarize the presented calibration routine for the two-factor model using the dynamic Chebyshev method.

Input:
CDS par rates R_{0,T_i}, maturities $T_1, \dots, T_{n_{cds}}$ with $T_{i+1} - T_i = T_1$ and riskless discount factors for the same days $D(T_i)$, recovery rate $R \in [0,1]$ (or $LGD = 1 - R$).

Calibration of ϕ_r:

- Calibrate ϕ_r as a piecewise constant function on $[T_{i-1}, T_i)$, $i = 1, \dots, n_{cds}$
- Calibration problem: For $i = 1, \dots, n_{cds}$ find root of

  $$\phi_r(T_i) \mapsto D(T_i) - \mathbf{1}(L_r^i)^{n_q} \dots (L_r^1)^{n_q} \boldsymbol{e}_s^T, \quad \text{with} \quad L_r^i = \exp(-\Delta t(\phi_r(T_i) + \boldsymbol{x}_r)) * \widehat{\Gamma}_r$$

  for n_T pricing steps between T_{i-1} and T_i, row vector $\boldsymbol{x}_r$ of nodal points, row vector $\mathbf{1}$ of ones and row vector $\boldsymbol{e}_s$ is one at $s = N/2 + 1$ for N even
- Root-finding via bisection type method or secant method
- Good initial guess $\widehat{\phi}_r(T_i) = -\log(D(T_i)/D(T_{i-1}))/(T_i - T_{i-1})$
- Store vector $(L_r^i)^{n_q} \dots (L_r^1)^{n_q} \boldsymbol{e}_s^T$ and re-use it for the next discount factor

Calibration of ϕ_λ:

- Calibrate ϕ_λ as a piecewise constant function on $[T_{i-1}, T_i)$, $i = 1, \dots, n_{cds}$
- Compute protection leg and premium leg for $CDS(R_{0,T_i})$

$$\begin{aligned} Protection(T_i) = Protection(T_{i-1}) \\ + LGD \sum_{u=0}^{n_T-1} \boldsymbol{e}_s (L_\lambda^1)^{n_q} \dots (L_\lambda^i)^{u-1} \tilde{\lambda}(T_i, \boldsymbol{z}_\lambda) \\ \cdot D(T_i, \boldsymbol{z}_r)(L_r^i)^{u-1} \dots (L_r^1)^{n_q} \boldsymbol{e}_s^T \\ Premium(T_i) = \sum_{j=1}^{i} \boldsymbol{e}_s (L_\lambda^1)^{n_q} \dots (L_\lambda^j)^{n_q} \mathbf{1}^T \mathbf{1} (L_r^j)^{n_q} \dots (L_r^1)^{n_q} \boldsymbol{e}_s^T \end{aligned}$$

with column vector $\boldsymbol{z}_\lambda$, row vector $\boldsymbol{z}_r$ and

$$\begin{aligned} L_\lambda^i &= \exp\big(-\Delta t \big(\varrho \frac{\sigma_r}{\sigma_\lambda} \boldsymbol{z}_\lambda + e^{\phi_\lambda(T_i)} e^{\boldsymbol{z}_\lambda} \big) \big) * \widehat{\Gamma}_\lambda \\ L_r^i &= \exp\big(-\Delta t \big(\phi_r(T_i) + \sqrt{1 - \varrho^2} \boldsymbol{z}_r \big) \big) * \widehat{\Gamma}_r \end{aligned}$$

$$\tilde{\lambda}(T_i) = e^{\phi_\lambda(T_i)} e^{\boldsymbol{z}_\lambda} \exp\big(-0.5\Delta t \big(\varrho \frac{\sigma_r}{\sigma_\lambda} \boldsymbol{z}_\lambda + e^{\phi_\lambda(T_i)} e^{\boldsymbol{z}_\lambda} \big) \big)$$
$$\tilde{D}(T_i) = \exp\big(-0.5\Delta t \big(\phi_r(T_i) + \sqrt{1-\varrho^2}\boldsymbol{z}_r \big) \big)$$

- Compute root of $\exp(\phi_\lambda(T_i)) \mapsto R_{0,T_i} T_1 Premium(T_i) - Protection(T_i)$
- Root-finding using a secant method or Brent-Dekker
- Store vectors $\boldsymbol{e}_s (L_\lambda^1)^{n_q} \dots (L_\lambda^i)^{n_q}$ and $(L_r^j)^{n_q} \dots (L_r^1)^{n_q} \boldsymbol{e}_s^T$.

5.2.5 Empirical investigation of the algorithm

In this section, we investigate the dynamic Chebyshev method for callable bonds numerically. We consider non-callable and callable defaultable bonds with and without coupon payments in the two-factor rate/credit model presented in Section 5.2.2. For our experiments we consider a bond with maturity $T = 5$ years, notional 100 and annual coupons of 5%. As model parameters we fix

$$\alpha_r = 0.01, \qquad \sigma_r = 0.03, \qquad \alpha_\lambda = 0.01, \qquad \sigma_\lambda = 0.6, \qquad \rho \in \{0, 0.5\}.$$

For the callable bond we assume that the option can be exercised after $1, 2, 3$ and 4 years at a strike price given in Table 5-A. Moreover, we consider the bond without recovery and with a recovery of 40%. Overall, we consider eight different scenarios.

We want to investigate the accuracy and the runtime of the presented method as well as its convergence behaviour. For this, we run the dynamic Chebyshev method for increasing $N = 16, 32, 64$ in each dimension, using 20 and 40 time steps per year and we choose the interpolation domain based on four times the standard deviation of the underlying process. We compute reference prices using $N = 128$ points, 80 time steps per year and 5 times the standard deviation as domain. All experiments have been performed in *Python* version 3.7.4 using the *numpy* package version 1.18.1.

Time	Strike (no coupons)	Strike (with coupons)
1	65	85
2	70	90
3	75	95
4	80	100

Table 5-A: Exercise dates and strike prices of the callable bonds.

In order to test the calibration procedure we need CDS par rates and discount factors

that represent realistic market data. In our numerical examples, we calibrate the two-factor model to par rates of CDS instruments maturing quarterly and riskless discount factors for the same maturities. The rates and the discount factors required for the calibration of the model are displayed in Table 5-B. The par rates have been computed via bootstrapping from five CDS instruments with maturity 1,2,3,4 and 5 years and corresponding par rates 0.73%, 1.41%, 2.00%, 2.57% and 3.09%. Moreover, it is assumed that the premium leg of the five instruments is paid annually. The rootfinding step in the calibration is performed with a target error tolerance of 10^{-6}.

Maturity	$D(T)$	CDS rate (no rec)	CDS rate (rec)
0	1	-	-
0.25	0.988481	0.007206	0.007194
0.5	0.977198	0.007090	0.007078
0.75	0.965403	0.007130	0.007117
1	0.953647	0.007149	0.007136
1.25	0.94203	0.009833	0.009810
1.5	0.930971	0.011509	0.011477
1.75	0.919701	0.012799	0.012758
2	0.908545	0.013764	0.013717
2.25	0.89747	0.015695	0.015636
2.5	0.886735	0.017170	0.017100
2.75	0.875895	0.018433	0.018351
3	0.865170	0.019482	0.019388
3.25	0.854427	0.021209	0.021102
3.5	0.844152	0.022590	0.022471
3.75	0.833661	0.023868	0.023731
4	0.823289	0.024981	0.024827
4.25	0.812910	0.026485	0.026319
4.5	0.802990	0.027731	0.027552
4.75	0.792863	0.028918	0.028720
5	0.782862	0.029980	0.029761

Table 5-B: Discount factors (without default risk) and CDS par rates (without recovery and with recovery of $R = 40\%$) used for model calibration.

Accuracy and runtimes of the method

We start with the investigation of the accuracy of the bond prices using the proposed calibration routine. An overview of the different scenarios along with reference prices of the non-callable and the callable bond is displayed in Table 5-C. As a first sanity check, we see that the callable bond is cheaper than the non-callable bond. Moreover, we observe that adding yearly coupon payments of 5% increases the overall value by around 20 for the non-callable bond and slightly more for the callable bond. We note that in case of coupon payments, we also had to change the strike prices of the bond. Assuming a non-zero recovery makes the bond more valuable because the bond owner receives a payment in case of recovery. The correlation parameter has a smaller but not negligible influence and makes the bond more expensive.

Scenario	Correlation	Coupon	Recovery	Non-callable	Callable
1	Zero	No	No	66.8558	56.9431
2	Positive	No	No	67.2185	56.3092
3	Zero	Yes	No	87.1385	81.2028
4	Positive	Yes	No	87.5320	80.3419
5	Zero	No	Yes	68.0632	58.4311
6	Positive	No	Yes	68.6168	57.9735
7	Zero	Yes	Yes	87.4958	81.7794
8	Positive	Yes	Yes	88.0960	81.1082

Table 5-C: Reference prices of the non-callable and the callable bond computed with a dynamic Chebyshev method with $N = 128$ in each dimension and 80 time steps per year.

Table 5-D displays the relative pricing error of the dynamic Chebyshev method with $N = 32$, $N = 64$ and 20 and 40 time steps per year. We observe that the dynamic Chebyshev method with $N = 64$ and 40 time steps is for all scenarios able to achieve a relative accuracy below 10^{-4} and thus an error of less than 0.01%. Moreover, a Chebyshev N of 32 and 20 time steps is already enough to obtain an error of less than 0.1% in all scenarios. Overall, we observe that both, the number of nodal points and the number of time steps per year influence the accuracy. In the next section, we will investigate the convergence behaviour in more detail.

Table 5-E displays the corresponding runtimes of the dynamic Chebyshev for calibration plus pricing as well as the individual runtimes. Overall, the runtimes of calibration plus pricing are always less than 50 ms, measured on a standard laptop using *Python*. If one is satisfied with a slightly lower accuracy, the whole procedure takes only half of

		1	2	3	4	5	6	7	8
$DC_{32,20}$	non-callable	0.55	1.09	0.44	0.90	0.54	1.27	0.44	1.06
	callable	0.21	4.20	1.84	0.70	1.25	0.72	1.71	0.31
$DC_{64,20}$	non-callable	0.57	1.12	0.47	0.92	0.57	1.30	0.47	1.10
	callable	0.98	0.20	0.01	0.81	1.01	1.26	0.35	0.69
$DC_{32,40}$	non-callable	0.62	0.92	0.50	0.75	0.59	0.96	0.49	0.80
	callable	0.05	4.64	2.02	0.35	1.02	1.24	1.90	0.74
$DC_{64,40}$	non-callable	0.63	0.93	0.51	0.76	0.60	0.97	0.50	0.81
	callable	0.71	0.26	0.21	0.44	0.74	0.75	0.12	0.23

Table 5-D: Pricing error for a non-callable and a callable bond of the dynamic Chebyshev method in different scenarios. Error in 10^{-4}.

the runtime. Doubling the Chebyshev N and thus increasing the total number of nodal points by a factor of 4 increases the total runtime by a factor 4. Mainly, because the runtime of the pricing increases. Doubling the number of time steps has a smaller influence and leads to an increase of around 25%. Depending on the Chebyshev N and the number of time steps, at least 2/3 of the runtime is the calibration of the model, even more if N is smaller. From the table we observe that the different scenarios have only a small influence on the runtime of the method. For example for $DC_{64,40}$, the runtime in scenario 8 with correlation and a non-zero recovery rate is only 3 ms longer than the runtime in scenario 1 with zero correlation and no recovery. This is an indicator for an overall high efficiency of the new method.

Empirical convergence analysis

Next, we investigate the convergence behaviour of the dynamic Chebyshev method for callable bonds. In order to do so, we compute the empirical order of convergence (EOC) and compare it to a theoretical error decay of N^{-2}. A quadratic error decay is for example achieved by a standard finite difference PDE solver using a Crank–Nicolson time discretization. As discussed in Section 5.2.1, finite difference methods are popular for callable bonds (and essentially option pricing in general) and thus present a good comparison for our pricing method. We consider the same eight different scenarios presented in Table 5-C. In each scenario, we compute the empirical order of convergence for the dynamic Chebyshev method with 20 and with 40 time steps per year.

Figure 5.4 shows the resulting convergence plots for a callable bond with and without coupon payments in a model with zero correlation and no recovery. Figure 5.5 shows

		1	2	3	4	5	6	7	8
	total	23	26	25	25	26	25	25	25
$DC_{32,20}$	calibration	19	21	19	19	20	19	20	19
	pricing	5	5	6	6	6	6	5	6
	total	38	38	38	38	38	39	39	41
$DC_{64,20}$	calibration	27	27	27	28	27	27	27	28
	pricing	10	11	11	10	12	12	12	13
	total	27	29	29	28	30	29	31	30
$DC_{32,40}$	calibration	21	23	23	22	23	22	23	23
	pricing	6	6	6	6	7	7	7	7
	total	46	46	47	47	48	50	50	49
$DC_{64,40}$	calibration	33	32	32	33	32	34	34	33
	pricing	14	14	15	15	16	16	17	16

Table 5-E: Total runtime, calibration time and pricing time of the dynamic Chebyshev method in eight different scenarios. All runtimes are shown in milliseconds.

the same convergence plots in a model with correlation and Figure 5.6 and Figure 5.7 show the same plots in a model with positive recovery. For all eight scenarios, we observe that the empirical order of convergence of the dynamic Chebyshev method is better than quadratic. The different scenarios have only a minor influence on the convergence rate. We conclude from the experiments that the new method has an advantage over classical finite difference solver in term of convergence behaviour.

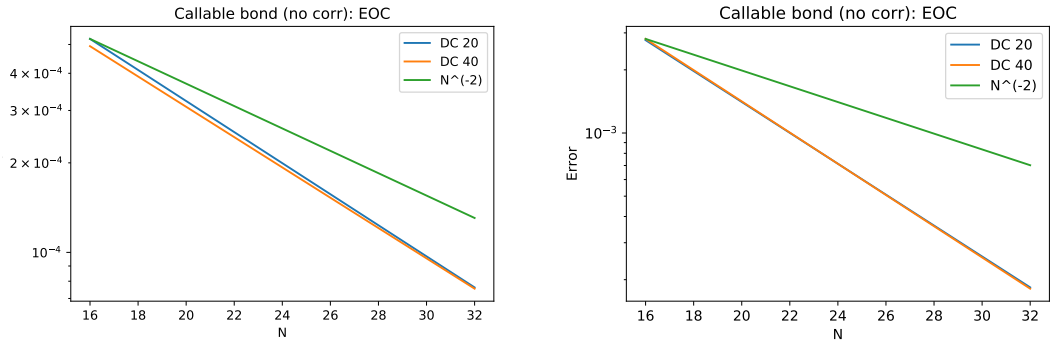

Figure 5.4: Empirical order of convergence of the dynamic Chebyshev method compared to a theoretical convergence of N^{-2} for a callable bond without coupons (left) and with coupons (right). We assume a model with zero correlation and no recovery.

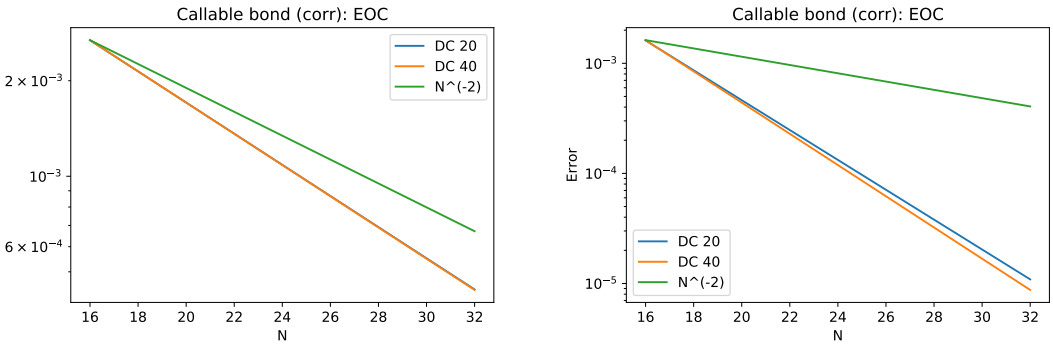

Figure 5.5: Empirical order of convergence of the dynamic Chebyshev method compared to a theoretical convergence of N^{-2} for a callable bond without coupons (left) and with coupons (right). We assume a model with positive correlation and no recovery.

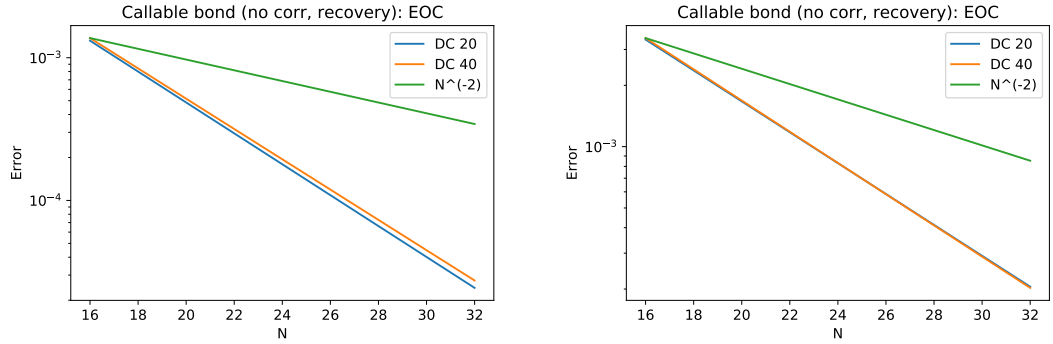

Figure 5.6: Empirical order of convergence of the dynamic Chebyshev method compared to a theoretical convergence of N^{-2} for a callable bond without coupons (left) and with coupons (right). We assume a model with zero correlation and a non-zero recovery rate.

Error behaviour for varying strikes

So far, we investigated the proposed approach for one specific callable bond. Here, we investigate how the error of the dynamic Chebyshev method behaves if we vary the strikes of the callable bond. For the experiment, we use scenario 8 in Table 5-C, i.e. we consider a model with positive correlation and assume a bond with recovery and coupon payments. We vary the strike K_1 at $T = 1$ between 50 and 130 and assume that the strike at $T = 2$ is $K_1 + 5$ and so on. We price the callable bond using the dynamic Chebyshev method with $N = 32$ and 20 time steps per year and the one with $N = 64$ and 40 time steps per year. In order to compute reference prices we use the dynamic Chebyshev method with $N = 128$ and 80 time steps per year.

Figure 5.8 shows the bond prices for varying strikes (left plot) and the error of the dynamic Chebyshev method (right plot). We observe that the price of a callable bond is increasing in the strike and converges towards the price of a non-callable bond. If

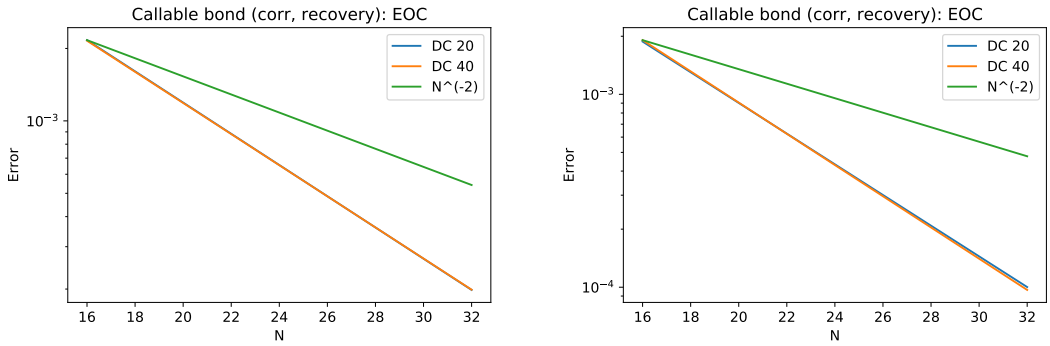

Figure 5.7: Empirical order of convergence of the dynamic Chebyshev method compared to a theoretical convergence of N^{-2} for a callable bond without coupons (left) and with coupons (right). We assume a model with positive correlation and a non-zero recovery rate.

the strike is high enough, the bond issuer will almost never call the bond. The error plot shows that the relative error is of the dynamic Chebyshev method with $N = 64$ is for all strikes below 10^{-4}. For small strikes it is even below 10^{-5}. The error of the dynamic Chebyshev method with $N = 32$ behaves similarly but is slightly higher. This numerical experiment shows that the method is able to price out-of-the-money as well as at-the-money and in-the-money callable bonds.

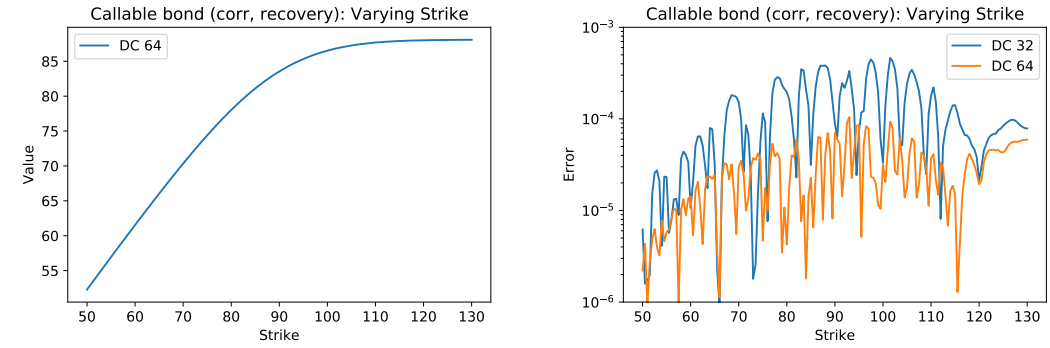

Figure 5.8: Pricing error of the dynamic Chebyshev method for a callable bond with varying strikes.

Stability of sensitivities

Next, we investigate the numerical stability of the calibration and the pricing using the dynamic Chebyshev method with respect to the input factors. We will bump (i.e. shift) the interest rate and the credit curve for different tenors, re-run the calibration and pricing algorithm and then compute the pricing delta. For the interest rate, we bump the riskless discount factors and for the credit intensity we bump the credit spreads presented in Table 5-B. For both, rate and credit we apply a parallel shift of the curve (i.e. shift

all input factors simultaneously) as well as individual shifts of the input factors. The individual shifts are conducted on a yearly basis, this means that we bump the discount factors or CDS rates with maturity $0.25, 0.5, 0.75, 1$ together for year 1 and the same for the years 2 to 5. For the interest rate, we apply shifts between -0.1 and 0.1 and for the credit spreads we apply shifts between -0.0025 and 0.0025. A positive shift of the riskless discount factor corresponds to lower interest rates and should lead to higher bond prices. In contrast, a higher credit spread means a higher default risk and should lead to a lower bond price.

We consider a callable bond in scenario 8 presented in Table 5-C, i.e. a callable bond with positive correlation, non-zero recovery and coupon payments, for this experiment. The use the bivariate dynamic Chebyshev method with $N = 64$ and 40 time steps per year.

Figure 5.9 shows the parallel shift (left plot) and the individual shifts (right plot) of the riskless discount factors. The parallel shift shows that a positive shift of the discount factors yields higher bond prices as expected. From the individual shifts we observe that the shifts on the lower end of the interest rate curve has a bigger influence on the overall price. Figure 5.10 shows the parallel shift (left plot) and the individual shifts (right plot) of the credit spreads. As expected, we observe that higher credit spreads cause lower bond prices. For all scenarios, we could observe that the computed pricing deltas look reasonable and there are no instabilities in the plotted delta profiles.

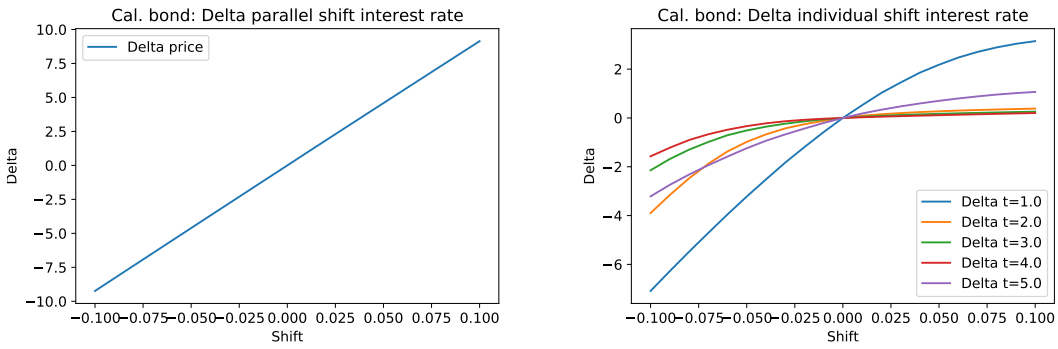

Figure 5.9: Delta of the dynamic Chebyshev method for different parallel shifts (left plot) and individual bumps (right plot) of the interest rate curve for a callable bond with coupons and a recovery of $R = 40\%$.

Conclusion experiments

The numerical experiments confirm that the proposed bivariate dynamic Chebyshev method is a valid method for the calibration and pricing of callable defaultable bonds in a hybrid interest rate/credit model. The experiments show that the method produces

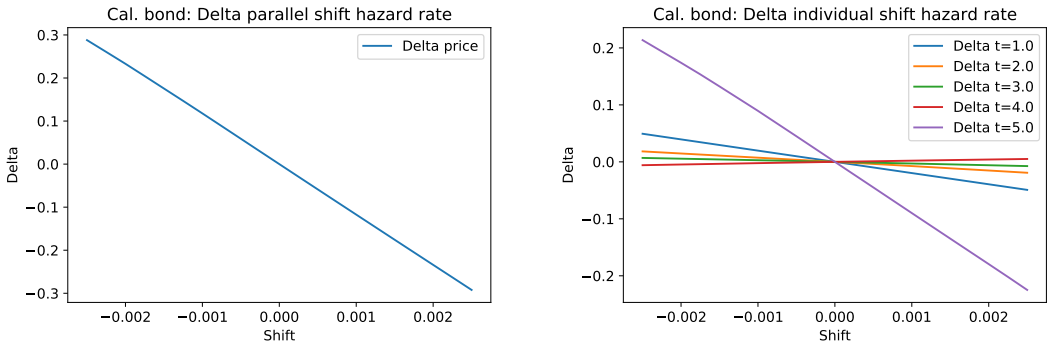

Figure 5.10: Delta of the Dynamic Chebyshev method for different parallel shifts (left plot) and individual bumps (right plot) of the hazard rate curve for a callable bond with coupons and a recovery of $R = 40\%$.

accurate results using only a low number of nodal points and the empirical order of convergence is better than a quadratic error decay. We conclude that the dynamic Chebyshev method is therefore advantageous in comparison to common PDE solvers using finite differences. The reported runtimes of the method are fast and the complete calibration plus pricing takes less than 50ms seconds on a mid-range laptop using *Python*.

We investigated the proposed method further and showed that the new approach is able to produce accurate results for in-the-money as well as for at-the-money and out-of-the money embedded options. Moreover, we verified the stability of the calibration and pricing routine with respect to parallel and individual shifts of the interest rate and credit curve used for calibration.

We believe that this numerical investigation is a good starting point for further testing and future research. We considered one specific bond in one market environment but it would be interesting to investigate the method's behaviour for different maturities as well as for different interest rate and credit curves. Moreover, we considered only callable bonds but the presented method is also a promising candidate for the pricing of other credit derivatives. Extending the method to puttable bonds should for example be possible.

5.3 Pricing basket options using quadrature

In this section, we consider the pricing of basket options in a multivariate Black-Scholes model. In Section 5.1.2, we have shown that the log-stock price process in the multivariate Black-Scholes model can be transformed into an independent process using a singular value decomposition of the covariance matrix. In this case, the multivariate dynamic Chebyshev method is simplified and the complexity of the pre-computation

step grows only linearly in the dimension d. The tensor of nodal points grows however still exponentially in d. In this section, we discuss a smoothing idea that ensures that the total number of nodal points stays as small as possible and enables a more efficient pricing of multivariate options. We start with the pricing of European basket options and then extend the suggested approach to basket options with an early-exercise feature.

Computing European option prices is essentially a numerical integration problem of calculating the expected value of a (payoff) function. Let $\boldsymbol{S}_T = (S_T^1, \dots, S_T^d)$ be the value of the underlying stocks at maturity T. Then the value of a basket call option C_B and a basket put option P_B are given by

$$C_B = e^{-rT}\mathbb{E}\Big[\Big(\sum_{i=1}^{d} w_i S_T^i - K\Big)^+\Big] \quad \text{and} \quad P_B = e^{-rT}\mathbb{E}\Big[\Big(K - \sum_{i=1}^{d} w_i S_T^i\Big)^+\Big]$$

for a deterministic interest rate $r \geq 0$, strike K and weights $w_1, \dots, w_d$. Furthermore, in most stochastic models we can alternatively compute this expectation by solving a P(I)DE.

Two problems arise when we compute this expectation numerically. First, the option payoff is not differentiable at the strike K and therefore, a high number of function evaluations is required for an accurate evaluation of the expectation. The second problem is the dimensionality d of the basket option. Classical univariate pricing techniques suffer from the curse of dimensionality when they are extended to higher dimensions. The number of function evaluations grows exponentially in the number of assets. The combination of the two problems means that even in two or three dimensions the number of grid points (and thus function evaluations) required for a reasonably high accuracy is often several thousand points or more.

Combining tensor based methods with dimension reduction techniques such as sparse grids can reduce the number of function evaluations significantly. Several paper propose to use (adaptive) sparse grids either as a quadrature technique or in order to solve the associated PDE. Apaptive sparse grid quadrature for finance is for example investigated in [11] and adaptive sparse grid for PDEs is for example investigated in [24]. While adaptive sparse grid methods are often able to provide good approximation using only a small number of function evaluations they typically cause a significant computational overhead.

The most common approach for option pricing in multivariate dimensions is the use of Monte Carlo simulation and quasi Monte Carlo simulation since they do not suffer from the curse of dimensionality. However, the rate of convergence of these methods is slow and a high number of (simulation) points is required for accurate results. For

Monte Carlo simulation in particular a numerical noise is introduced that makes the calculation of sensitivities less stable. Bumping one of the input factors and re-run the simulation does not always yield a price movement into the right direction. For example, even though the price function is increasing in the volatility, a Monte Carlo simulation for a higher σ might still return a lower price.

In order to find an efficient and stable pricing algorithm for basket options, we employ the so-called 'smoothing the payoff' idea proposed by [11] and compare three different quadrature techniques for the resulting integration problem. We show that for medium-sized dimension a direct application of numerical quadrature techniques omits the computational overhead of adaptive sparse grids reported by [11]. We then show how the new quadrature approach can be turned into an interpolation method for basket options, similar to the static Chebyshev method introduced in Section 2.3.3. We build on this interpolation and extend the smoothing the payoff idea to the dynamic programming problem of pricing a basket option with early exercise feature. The efficiency of the resulting dynamic Chebyshev pricing method is confirmed by a numerical convergence analysis and a comparison of the method's performance to the least-squares Monte Carlo approach.

We start by discussing the dynamic Chebyshev method as a quadrature approach and compare it to other quadrature techniques such as Clenshaw-Curtis, Gauss-Legendre and Gauss-Hermite.

5.3.1 Dynamic Chebyshev for quadrature

In this section, we show that the dynamic Chebyshev method can be seen as a quadrature technique for conditional expectations. For a stochastic process $(X_t)_{t\geq 0}$ and a function $g : \mathbb{R} \to \mathbb{R}$ we want to compute the conditional expectation

$$\mathbb{E}\big[g(X_T)\big|X_0 = x_0\big] = \int_{-\infty}^{\infty} g(x) f_{X_T}(x|x_0)\mathrm{d}x \tag{5.6}$$

where $f_{X_T}(\,\cdot\,|x_0)$ is the density of $X_T|X_0 = x_0$. A classical numerical quadrature technique such as Clenshaw-Curtis or Gauss-Legendre provides quadrature points $x_1, \ldots, x_M$ and corresponding weights $w_1, \ldots, w_M$ such that

$$\int_{-\infty}^{\infty} g(x) f_{X_T}(x|x_0)\mathrm{d}x \approx \int_{\underline{x}}^{\overline{x}} g(x) f_{X_T}(x|x_0)\mathrm{d}x \approx \sum_{j=1}^{M} w_j g(x_j) f_{X_T}(x_j|x_0)$$

for a suitable integration domain $[\underline{x}, \overline{x}]$ and under some appropriate integrability conditions on g and f_{X_T}. Hence, the integral is approximated by a weighted sum of function

values at a set of nodal points, for example Chebyshev points in the case of the Clenshaw-Curtis quadrature.

In this sense, each time step in the dynamic Chebyshev method can be seen as a form of numerical quadrature. We discussed the idea that the calculation of the coefficients can be omitted and the algorithm operates only on the nodal values, see Section 5.1.4. Assume that we have the nodal values of the value function at t_{u+1} given by $\widehat{V}_{t_{u+1}}(x_0), \dots, \widehat{V}_{t_{u+1}}(x_N)$. At time point t_u, the continuation value at the nodal point x_k is given by

$$\widehat{V}_{t_u}(x_k) = \mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x_k] \approx \sum_{j=0}^{N} w_{jk} \widehat{V}_{t_{u+1}}(x_j)$$

for weights w_{jk} as defined in (5.2). This can be generalized to a quadrature formula for conditional expectations of the form (5.6). Assume we have a Chebyshev interpolation $\sum_j c_j p_j$ of a Lipschitz continuous function $g : \mathbb{R} \to \mathbb{R}$ on the interpolation domain $[\underline{x}, \overline{x}]$. Then we can approximate (5.6) as

$$\mathbb{E}\big[g(X_T)\big|X_0 = x_0\big] \approx \mathbb{E}\big[\sum_{j=0}^{N} c_j p_j(X_T)\big|X_0 = x_0\big] = \sum_{j=0}^{N} c_j \mathbb{E}\big[p_j(X_T)\big|X_0 = x_0\big] = \sum_{j=0}^{N} c_j \Gamma_j^0.$$

Here, Γ_j^0 refers to the conditional expectation of p_j given a starting value x_0, i.e.

$$\Gamma_j^0 := \mathbb{E}\big[p_j(X_T)\big|X_0 = x_0\big].$$

All stochastic information are hidden in the generalized moments Γ_j^0, the zero indicates that the generalized moments depend on the starting value x_0. From the definition of the Chebyshev coefficients follows the quadrature type formula

$$\begin{aligned}\sum_{j=0}^{N} c_j \Gamma_j^0 &= \sum_{j=0}^{N}{}'' \frac{2}{N} \sum_{k=0}^{N}{}'' g(x_k) T_j(z_k) \Gamma_j^0 \\ &= \sum_{k=0}^{N} g(x_k) \Big(\frac{2^{\mathbb{1}_{0<k<N}}}{N} \sum_{j=0}^{N}{}'' T_j(z_k) \Gamma_j^0 \Big) = \sum_{k=0}^{N} g(x_k) w_k^0.\end{aligned}$$

The weights w_k^0 are a linear transformation of the generalized moments using a matrix of entries $(2/N)T_j(z_k)$, where the first and last column and row are multiplied by $1/2$. The resulting quadrature algorithm is displayed in Algorithm 5 for the slightly more general problem of computing $\mathbb{E}[g(X)]$ for a random variable X. If the underlying stochastic process is normally distributed we have an explicit formula for the generalized moments Γ_j^0 and thus the weights w_k^0, see formula (3.17).

Algorithm 5 Univariate dynamic Chebyshev quadrature

Require: Function $g : \mathbb{R} \to \mathbb{R}$, random variable X and Chebyshev degree N

1: Fix interpolation domain $[\underline{x}, \overline{x}]$

2: Define Chebyshev points $z_k = \cos(k\pi/N)$ and nodal points $x_k = \tau_{[\underline{x},\overline{x}]}(z_k)$, $0 \le k \le N$

3: **Quadrature weights:**

4: $\quad \Gamma_j = \mathbb{E}[p_j(X)] = \mathbb{E}[T_j(\tau^{-1}_{[\underline{x},\overline{x}]}(X))\mathbb{1}_{[\underline{x},\overline{x}]}(X)]$

5: $\quad$ Compute $w_k = \frac{2^{\mathbb{1}_{0<k<N}}}{N} \sum_{j=0}^{N}{}'' T_j(z_k)\Gamma_j$, $k = 0, \dots, N$

6: **Compute integral**

7: $\quad \mathbb{E}[g(X)] \approx \sum_{k=0}^{N} g(x_k)w_k$

Comparison with Clenshaw-Curtis quadrature

The dynamic Chebyshev quadrature with $(N+1)$ points looks similar to the Clenshaw-Curtis quadrature with $M = N + 1$. In both cases, we require the function values at the same Chebyshev points and we multiply them with some pre-computed quadrature weights. Naturally, the question arises how the two approaches differ or if they are essentially the same.

We consider again the integration problem (5.6). The idea of Clenshaw-Curtis is to interpolate the integrand $x \mapsto g(x)f(x|x_0)$ in the Chebyshev points and obtain

$$\int_{-\infty}^{\infty} g(x)f(x|x_0)\mathrm{d}x \approx \int_{\underline{x}}^{\overline{x}} \sum_{k=1}^{M} g(x_k)f(x_k|x_0)l_k(x)\mathrm{d}x = \sum_{k=1}^{M} g(x_k)f(x_k|x_0) \int_{\underline{x}}^{\overline{x}} l_k(x)\mathrm{d}x$$

where l_k are the Lagrange polynomials corresponding to the Chebyshev points $x_1, \dots, x_M$. The Clenshaw-Curtis weights are then essentially the integrated polynomials with respect to the Lebesgue measure. These weights are independent of the specific integration problem and just need to be computed once. We refer to [129] for more details on Clenshaw-Curtis integration. The error decay of this quadrature is determined by the smoothness of the integrand

$$[\underline{x}, \overline{x}] \ni x \mapsto g(x)f(x|x_0).$$

The dynamic Chebyshev quadrature on the other hand starts with the interpolation of the function $x \mapsto g(x)$ and the quadrature weights are then computed as integrated polynomials with respect to the density $f(\,\cdot\,|x_0)$. If f is the density of a normally distributed random variable we obtain an analytic formula for the quadrature weights. In general, they have to be calculated numerically and are problem specific. However, they can be re-used if the expectations of several functions g are computed with respect to the same distribution. The Clenshaw-Curtis quadrature is specifically designed for an integration

w.r.t. the Lebesgue measure, i.e. the weights are the integrated Chebyshev polynomials w.r.t. to the Lebesgue measure. In contrast, the dynamic Chebyshev quadrature can be applied to a wider class of (probability) measures and the weights are calculated by integrating the Chebyshev polynomials w.r.t. a (probability) measure μ.

In summary, for the Clenshaw-Curtis quadrature, the underlying random variable X_T has no influence on the choice of the weights and the quadrature uses only information of the density at the nodal points. In contrast, the weights of the dynamic Chebyshev quadrature depend on the distribution of X_T on the domain $[\underline{x}, \overline{x}]$. Depending on the application this additional information might be beneficial. In case of a normally distributed variable, the Gauss-Hermite quadrature is an interesting comparison for the dynamic Chebyshev approach. For the Gauss-Hermite quadrature both, points and weights are chosen with respect to the underlying normal distribution. Moreover, the quadrature is defined on $(-\infty, \infty)$ and no truncation error is made. The Gauss-Hermite quadrature should converge faster since points and weights are chosen in an optimized way. However, this procedure cannot be extended to other models/random variables.

An advantage of the dynamic Chebyshev approach is that it can also be used if the density of X_T is unknown. In this case, other techniques have to be used in order to compute the generalized moments, see the discussion in Section 3.3.2. The Clenshaw-Curtis quadrature cannot be used if the density of the random variable is not available in closed form.

Numerical investigation

In this section, we provide empirical evidence that the presented dynamic Chebyshev quadrature is a suitable approach for integration problems in option pricing. Moreover, we investigate the effect of a smooth integrand on the convergence behaviour. As a toy example, we consider (5.6) for the Black-Scholes call and put option price, i.e.

$$g(x) = C_{BS}(e^x, K, \sigma, r, \tau) \quad \text{and} \quad g(x) = P_{BS}(e^x, K, \sigma, r, \tau)$$

for a strike K, volatility σ interest rate r and time to maturity τ. We assume that the process $(X_t)_{t\geq 0}$ is a Brownian motion with drift where $X_T = x_0 + (r - 0.5\sigma^2)T + \sigma\sqrt{T}Z$ for $Z \in \mathcal{N}(0,1)$. We calculate the conditional expectations of g for both option types using the dynamic Chebyshev quadrature method and compare the method's accuracy to the Clenshaw-Curtis quadrature and the Gauss-Legendre quadrature for the same number of quadrature point. We fix the following parameters

$$x_0 = \log(100), \quad K = 100, \quad r = 0.03, \quad \sigma = 0.25, \quad T = 1, \quad \tau = 0.5.$$

The domain of integration is chosen as $\mathcal{X} = [\underline{x}, \overline{x}] = [(r-0.5\sigma^2)T - 6\sigma\sqrt{T}, (r-0.5\sigma^2)T + 6\sigma\sqrt{T}]$ and ensures that the probability of $X_T \notin \mathcal{X}$ is below $2 \cdot 10^{-9}$. We calculate reference values using *Matlab*'s integration routine *quadgk* with an absolute error tolerance of 10^{-14} and a larger integration domain $[(r - 0.5\sigma^2)T - 9\sigma\sqrt{T}, (r - 0.5\sigma^2)T + 9\sigma\sqrt{T}]$.

Figure 5.11 shows the resulting error decay for all three quadrature techniques. For both integrands, we observe that the dynamic Chebyshev approach is as good as the other two quadrature routines and the differences are negligible. All methods are able to converge and reach a maximal accuracy below 10^{-8} and thus in the region of the truncation error. This confirms that the dynamic Chebyshev quadrature is a suitable numerical approach for option pricing.

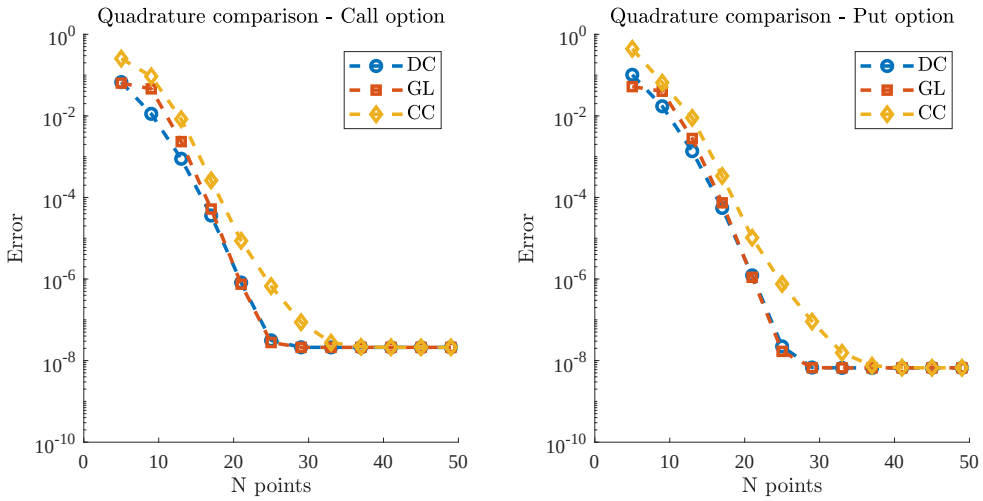

Figure 5.11: Comparison of the dynamic Chebyshev quadrature against Clenshaw-Curtis and Gauss-Legendre quadrature. The integrand is the Black-Scholes call price (left) and put price (right) for a maturity of $T = 1$.

Next, we investigate how the convergence behaviour of the dynamic Chebyshev quadrature changes depending on the smoothness of the integrand. For this, we will vary the time to maturity τ from 0 to 1. For $\tau = 0$, the option price is the payoff function and we know that both, the call and the put option payoff, have a kink at the strike K. However, for every $\tau > 0$ the option price becomes an analytic function of the log-stock price. The domain of analyticity depends on size of τ. The larger the time to maturity τ becomes, the smoother will be the option price. We know that Chebyshev interpolation and quadrature performs better for smooth functions. The experiments here should give us a good indication how strong this smoothness effect actually is. Figure 5.12 shows the error decay for both integrands and $\tau = 0, 1/12, 1/4, 1/2, 1$. First we note that there is no difference between the call and the put option and we can focus on the call option here. We observe that the quadrature error is in the region of 10^{-8} for

31 nodal points if $\tau = 1$. However, for $\tau = 0$ we are not able to reach an accuracy of 10^{-2} with up to 50 nodal points. We observe that even a small τ of 1/12 leads already to a significantly faster error decay.

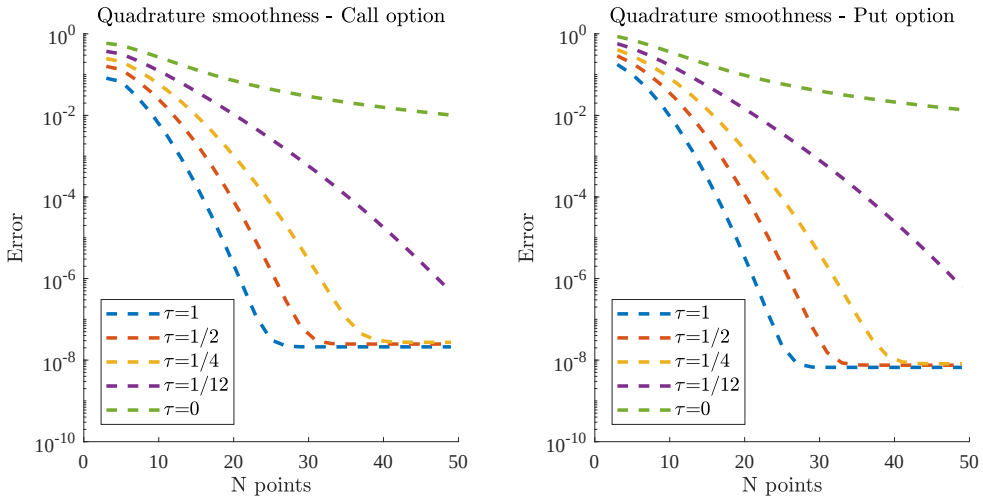

Figure 5.12: Investigation of the effect of a smooth integrand on the convergence behaviour of the dynamic Chebyshev quadrature. The integrands are Black-Scholes call prices (left) and put prices (right) for different maturities.

Quadrature for multivariate functions

The presented quadrature approaches in one dimension have multivariate extensions. Consider a d-variate stochastic process $(\boldsymbol{X}_t)_{t\geq 0}$ and a continuous function $g : \mathbb{R}^d \to \mathbb{R}$. The multivariate version of (5.6) is given by

$$\mathbb{E}\big[g(\boldsymbol{X}_T)\big|\boldsymbol{X}_0 = \boldsymbol{x}_0\big] = \int_{\mathbb{R}^d} g(\boldsymbol{x}) f_{\boldsymbol{X}_T}(\boldsymbol{x}|\boldsymbol{x}_0)\mathrm{d}\boldsymbol{x} \tag{5.7}$$

where $f_{\boldsymbol{X}_T}(\,\cdot\,|\boldsymbol{x}_0)$ is the density of $\boldsymbol{X}_T|\boldsymbol{X}_0 = \boldsymbol{x}_0$. Numerically, we can compute the integral by applying for example a Clenshaw-Curtis or Gauss-Legendre quadrature in every dimension $i = 1,\dots,d$. The multivariate quadrature points $\boldsymbol{x}_{\boldsymbol{k}} = (x_{k_1},\dots,x_{k_d})$ are a tensor grid of one-dimensional points x_{k_i}, $k_i = 1,\dots,M_i$ and the corresponding weights $w_{\boldsymbol{k}} = w_{k_1}\dots w_{k_d}$ are the product of weights w_{k_i}, $k_i = 1,\dots,M$ for $i = 1,\dots,d$. Overall, we obtain

$$\begin{aligned}\int_{\mathbb{R}^d} g(\boldsymbol{x}) f_{\boldsymbol{X}_T}(\boldsymbol{x}|\boldsymbol{x}_0)\mathrm{d}\boldsymbol{x} &\approx \sum\nolimits_{\boldsymbol{k}} w_{\boldsymbol{k}} g(\boldsymbol{x}_{\boldsymbol{k}}) f_{\boldsymbol{X}_T}(\boldsymbol{x}_{\boldsymbol{k}}|\boldsymbol{x}_0)\\ &= \sum_{k_1=1}^{M_1}\dots\sum_{k_d=1}^{M_d} w_{k_1}\dots w_{k_d} g(x_{k_1},\dots,x_{k_d}) f_{\boldsymbol{X}_T}(x_{k_1},\dots,x_{k_d}|\boldsymbol{x}_0).\end{aligned}$$

If $\boldsymbol{X}_T$ is a vector of independent random variables we can further simplify this expression and write the joint density function as a product of the marginal density functions

$$f_{\boldsymbol{X}_T}(x_{k_1}, \dots, x_{k_d}|\boldsymbol{x}_0) = \prod_{i=1}^{d} f_{X_T^i}(x_{k_i}|x_0^i).$$

In the same way, the multivariate version of the dynamic Chebyshev approach can be used as a multivariate quadrature. For an integration problem of the form (5.7) it can be written as

$$\int_{\mathbb{R}^d} g(\boldsymbol{x}) f_{\boldsymbol{X}_T}(\boldsymbol{x}|\boldsymbol{x}_0)\mathrm{d}\boldsymbol{x} \approx \sum\nolimits_{\boldsymbol{k}} w_{\boldsymbol{k}}^0 g(\boldsymbol{x}_{\boldsymbol{k}}).$$

In both cases, we have a tensor of nodal values $\mathcal{G} = (g(\boldsymbol{x}_{\boldsymbol{k}}))_{\boldsymbol{k}} \in \mathbb{R}^{M\times\dots\times M}$ and a tensor of quadrature weights. The tensor of weights has rank one and can be written as the product of d vectors of one-dimensional weights $W^i = [w_1, \dots, w_{M_i}]^T$. The quadrature can then be written as tensor times vector multiplications, see Definition 9. The dynamic Chebyshev quadrature in form of tensor times matrix multiplication is given by

$$\mathbb{E}\big[g(\boldsymbol{X}_T)\big|\boldsymbol{X}_0 = \boldsymbol{x}_0\big] = \mathcal{G} \times_1 W^1 \times_2 \dots \times_d W^d. \tag{5.8}$$

The problematic part in (5.8) is the full tensor $\mathcal{G}$ that can be infeasible to work with. Depending on the number of points per dimension this might already be the case for relatively low dimensions d. The first challenge is the computation of the tensor itself, the second one is the storage of all entries and the third challenge are the n-mode multiplications. It is important to have a smooth integrand g in order to ensure that the number of nodal points per dimension is relatively small and can still be computed efficiently. If d is large enough, additional dimension reduction techniques need to be applied. In the next section, we explore the use of low-rank tensor compression and a tensor completion algorithm as proposed in [57] to tackle this problem.

5.3.2 Chebyshev quadrature for basket options

We want to use the presented multivariate quadrature method for the efficient pricing of basket options. The basket call option payoff is however not differentiable and we have to introduce a smoothing first. We consider a multivariate Black-Scholes with d assets $\boldsymbol{S}_t = (S_t^1, \dots, S_t^d)$ modelled by the following SDE under the risk-neutral pricing measure

$$\mathrm{d}S_t^i = S_t^i\big(r\mathrm{d}t + \sigma_i \mathrm{d}W_t^i\big), \quad \text{for} \quad i = 1, \dots, d$$

with starting value S_0, interest rate $r \geq 0$ and volatilities $\sigma^1, \ldots, \sigma^d > 0$. Assume that the Brownian motions W_t^i are correlated with

$$\mathrm{d}W_t^i \mathrm{d}W_t^j = \varrho_{ij} \mathrm{d}t \quad \text{for} \quad \varrho_{ij} \in [-1, 1]$$

with $\varrho_{ii} = 1$, $i = 1, \ldots, d$. The SDE for the stock price has the solution

$$S_t^i = S_0^i e^{(r - 0.5\sigma^2)t + \sigma W_t^i} \quad \text{for} \quad i = 1, \ldots, d,$$

see Formula (2.38). Following the notation of [11], we can rewrite the basket call option as

$$\begin{aligned} C_B = e^{-rT}\mathbb{E}\Big[\Big(\sum_{i=1}^d w_i S_T^i - K\Big)^+\Big] &= e^{-rT}\mathbb{E}\Big[\Big(\sum_{i=1}^d w_i S_0^i e^{(r-0.5\sigma^2)T + \sigma W_T^i} - K\Big)^+\Big] \\ &= \mathbb{E}\Big[\Big(\sum_{i=1}^d \beta_i e^{X_i} - e^{-rT}K\Big)^+\Big] \end{aligned}$$

where $\boldsymbol{X} = (X_1, \ldots, X_d) \sim \mathcal{N}_d(0, \Sigma)$ for modified weights β_i and time-scaled covariance matrix Σ given by

$$\beta_i = w_i S_0^i e^{-0.5\sigma_i^2 T} \quad \text{and} \quad \Sigma_{ij} = \sigma_i \sigma_j \varrho_{ij} T.$$

Note that the random variable X_i is not the usual log-stock price $X_t^i = \log(S_t^i)$ but only the diffusion part of X_T^i. Similarly, we obtain for the basket put option

$$P_B = e^{-rT}\mathbb{E}\Big[\Big(K - \sum_{i=1}^d w_i S_T^i\Big)^+\Big] = \mathbb{E}\Big[\Big(e^{-rT}K - \sum_{i=1}^d \beta_i e^{X_i}\Big)^+\Big].$$

In both cases, we can write the option price as a d-dimensional integral using the density $\varphi_{\boldsymbol{X}}$ of the multivariate normal distribution and we obtain for the call option

$$\mathbb{E}\Big[\Big(\sum_{i=1}^d \beta_i e^{X_i} - e^{-rT}K\Big)^+\Big] = \int_{\mathbb{R}^d} \Big(\sum_{i=1}^d \beta_i e^{\boldsymbol{x}} - e^{-rT}K\Big)^+ \varphi_{\boldsymbol{X}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}.$$

Since the payoff function is not differentiable, we will explore the smoothing properties of the Gaussian density function.

Smoothing the payoff of basket options

We start with a special case and assume that all assets are independent and the covariance matrix Σ is a diagonal matrix with entries $\Sigma_{i,i} = \sigma_i^2 T$. In this case, we can write

$$C_B = \mathbb{E}\Big[\big(\sum_{i=1}^{d} \beta_i e^{X_i} - e^{-rT}K\big)^+\Big] = \mathbb{E}\Big[\mathbb{E}\Big[\big(\beta_1 e^{X_1} - \big(e^{-rT}K - \sum_{i=2}^{d} \beta_i e^{X_i}\big)\big)^+ \Big| \overline{\boldsymbol{X}}\Big]\Big]$$

for $\overline{\boldsymbol{X}} = (X_2, \ldots, X_d)$ independent of X_1. The inner conditional expectation can be seen as a call option in the one-dimensional Black-Scholes model with a modified strike. Let $C_{BS}(S, K, \sigma)$ be the Black-Scholes call price for maturity $T = 1$ and zero interest rate. In this case, we can write

$$\begin{aligned} C_B &= \mathbb{E}\Big[\mathbb{E}\Big[\big(\beta_1 e^{X_1} - \big(e^{-rT}K - \sum_{i=2}^{d} \beta_i e^{X_i}\big)\big)^+ \Big| \overline{\boldsymbol{X}}\Big]\Big] \\ &= \mathbb{E}\Big[\mathbb{E}\Big[\big(w_1 S_0^1 e^{-0.5\sigma_1^2 T} e^{X_1} - \big(e^{-rT}K - \sum_{i=2}^{d} \beta_i e^{X_i}\big)\big)^+ \Big| \overline{\boldsymbol{X}}\Big]\Big] \\ &= \mathbb{E}\Big[C_{BS}\Big(w_1 S_0^1, e^{-rT}K - \sum_{i=2}^{d} \beta_i e^{X_i}, \sigma_1\sqrt{T}\Big)\Big]. \end{aligned}$$

Here, the modified strike can be negative and we interpret the Black-Scholes call price as $C_{BS}(S, K, \sigma) = S - K$ for $K < 0$. Calculating the price of a basket call option is now transformed into a $(d-1)$-dimensional integration problem in $X_2, \ldots, X_d$ with a smooth integrand. These two improvements should lead to a significantly lower number of nodal points when using a numerical quadrature technique.

This modification is a special case of the general smoothing the payoff procedure presented in [11]. Their idea is to find a suitable linear transformation V such that $\boldsymbol{X} = V\boldsymbol{Y}$ for a vector of independent random variables $\boldsymbol{Y}$. The transformation V is chosen in way that allows us to write $\sum_{i=1}^{d} \beta_i e^{X_i} = e^{Y_1} h(\overline{\boldsymbol{Y}})$ for some function h and $\overline{\boldsymbol{Y}} := (Y_2, \ldots, Y_d)$ independent of Y_1. We obtain for the basket call

$$\begin{aligned} \mathbb{E}\Big[\big(\sum_{i=1}^{d} \beta_i e^{X_i} - e^{-rT}K\big)^+\Big] &= \mathbb{E}\Big[\big(e^{Y_1} h(\overline{\boldsymbol{Y}}) - e^{-rT}K\big)^+\Big] \\ &= \mathbb{E}\Big[\mathbb{E}\big[\big(e^{Y_1} h(\overline{\boldsymbol{Y}}) e^{-rT} - K\big)^+ \big| h(\overline{\boldsymbol{Y}})\big]\Big]. \end{aligned}$$

The inner conditional expectation is a one-dimensional integration problem of a European call option for a normally distributed variable Y_1. This can be calculated analytically and yields a smooth function of the $d-1$ dimensional random variable $\overline{\boldsymbol{Y}}$ in the outer expectation. The crucial step is the choice of the linear transformation V.

We know that if Σ is a positive definite covariance matrix, there exists an eigenvalue decomposition of Σ such that $\Sigma = V\Lambda V^T$ for a diagonal matrix $\Lambda = \operatorname{diag}(\lambda_1^2, \dots, \lambda_d^d)$ and an invertible matrix $V \in \mathbb{R}^{d\times d}$. From Lemma 3.1 in [11] follows that for each $\boldsymbol{v} \in \mathbb{R}^d$ we find a decomposition of this form such that $\boldsymbol{v}$ is the first column of V, for example $\boldsymbol{v} = \mathbf{1} = [1, \dots, 1]^T$. This allows us to define a new vector of independent variables $\boldsymbol{Y} := V^{-1}\boldsymbol{X}$ such that $\boldsymbol{Y} \sim \mathcal{N}_d(0, \Lambda)$ and we can write $\sum_{i=1}^d \beta_i e^{X_i} = e^{Y_1} h(\overline{\boldsymbol{Y}})$.

Proposition 13. *Assume a multivariate Black-Scholes model with positive definite covariance matrix Σ. Then there exits a diagonal matrix $\Lambda = diag(\lambda_1^2, \dots, \lambda_d^d)$ and an invertible matrix $V \in \mathbb{R}^{d\times d}$ with $V_{i,1} = 1$, $i = 1, \dots, d$ such that $\Sigma = V\Lambda V^T$. Using this decomposition, the price of a European basket call option is given by*

$$C_B = \mathbb{E}\Big[C_{BS}\big(h(\boldsymbol{Z})e^{\lambda_1^2/2}, e^{-rT}K, \lambda_1\big)\Big] \quad \textit{for} \quad \boldsymbol{Z} \sim \mathcal{N}_{d-1}\big(0, \overline{D}\big) \tag{5.9}$$

where $\overline{D} = diag(\lambda_2^2, \dots, \lambda_d^2)$ and function h given by

$$h(\overline{\boldsymbol{y}}) = h(y_2, \dots, y_d) = \sum_{i=1}^d \beta_i \exp\Big(\sum_{j=2}^d V_{i,j} y_j\Big).$$

Proof. See Lemma 3.2 and Proposition 3.3 in [11]. □

The basket call option can thus be computed as a expectation of the $d-1$ dimensional vector of independent random variables $\boldsymbol{Z}$. The function

$$\mathbb{R}^{d-1} \ni \boldsymbol{z} \mapsto C_{BS}\big(h(\boldsymbol{z})e^{\lambda_1^2/2}, K, \lambda_1\big)$$

is a composition of smooth functions and thus itself smooth. In fact, it is not just continuously differentiable but also admits an analytic continuation into the complex domain, see the results in Section 2.3.2. Similar results can be obtained if the basket call option is replaced by a basket put option. In this case, the smoothed integrand will be the put option price in the Black-Scholes model and equation (5.9) becomes

$$P_B = \mathbb{E}\Big[P_{BS}\big(h(\boldsymbol{Z})e^{\lambda_1^2/2}, e^{-rT}K, \lambda_1\big)\Big] \quad \text{for} \quad \boldsymbol{Z} \sim \mathcal{N}_{d-1}\big(0, \overline{D}\big)$$

where P_{BS} is the price of a European put option with $r = 0$ and $T = 1$ in the Black-Scholes model. The question remains how we can obtain the linear transformation V given the vector $\boldsymbol{v}$. From [11] we obtain the following construction.

Remark 6. *Let $\Sigma \in \mathbb{R}^{d\times d}$ be positive definite and symmetric and let $\boldsymbol{v} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$. We define $\boldsymbol{w} := \Sigma^{-1}\boldsymbol{v}$. Then the matrix*

$$\tilde{\Sigma} := \Sigma - \frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{w}} \in \mathbb{R}^{d\times d}$$

is positive semi-definite and symmetric with rank $d-1$. *Let* $\lambda_2^2 \geq \ldots \geq \lambda_d^2$ *be the* $d-1$ *eigenvalues of* $\tilde{\Sigma}$ *and* $\boldsymbol{v}_2, \ldots, \boldsymbol{v}_d$ *the corresponding eigenvectors. Defining* $V = [\boldsymbol{v}, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_d]$ *and* $D = diag(\lambda_1^2, \ldots, \lambda_d^2)$ *for* $\lambda_1^2 = (\boldsymbol{v}^T \boldsymbol{w})^{-1}$ *yields* $\Sigma = VDV^T$ *and* V *is invertible.*

For a high-dimensional d, the resulting quadrature of the smoothed basket option can be written as tensor times vector multiplication, as in (5.8). The difference is that the tensor of nodal points depends only on $d-1$ dimensions and only the vector of weights $W^2, \ldots, W^d$ are required. Overall, we expect that the combination of a dimension reduction plus the smooth integrand results in a significant lower number of quadrature points.

5.3.3 Numerical investigation of the method

In this section, we investigate the effect of the smoothing and the dimension reduction numerically. We analyse the convergence behaviour of three different quadrature techniques, Clenshaw-Curtis quadrature, dynamic Chebyshev quadrature and Gauss-Hermite quadrature. Moreover, we compare the quadrature approaches to a standard Monte Carlo simulation. [11] focused their numerical investigation on the comparison of adaptive sparse grids with Monte Carlo simulation and quasi Monte Carlo simulation. They showed that adaptive sparse grids achieve a fast convergence in terms of number of function evaluations but are typically relatively slow in comparison to simulation methods. In our experiments, we show that for low and medium high dimensions, a tensor product quadrature combines a fast convergence with low runtimes.

We consider basket call options on 2, 3 and 5 different stocks and investigate the convergence using the non-smooth payoff and the Black-Scholes price as integrand. Furthermore, we calculate the error decay as a function of the runtime and compare the quadrature approaches to the Monte Carlo simulation. In the bivariate case, we additionally compare the quadrature techniques to a benchmark method using the $2d$ COS method introduced in [109].

For the experiments, we consider a multivariate Black-Scholes model with d assets and we fix the following parameters

$$S_0^i = 100, \quad \sigma_i = 0.25, \quad r = 0.03 \quad \text{for} \quad i = 1, \ldots, d$$

and a basket call option with maturity $T = 1$, a strike of $K = 100$ and equal weights $w_i = (1/d)$, $i = 1, \ldots, d$. The integration domain of the Clenshaw-Curtis and dynamic Chebyshev is defined in terms of the standard deviation of the underlyings. The Gauss-Hermite quadrature is defined on the whole real line and no truncation error is made. For

the Monte Carlo simulation we use antithetic variates as variance reduction technique and compute the error as the median error over 50 runs to filter out the simulation noise. If not stated otherwise, we measure the error in relative terms.

Bivariate basket option

We start with a bivariate basket call option given

$$C_B = e^{-rT}\mathbb{E}\big[\big(0.5S_0^1 e^{X_T^1} + 0.5S_0^2 e^{X_T^2} - K\big)^+\big] \quad \text{with} \quad \boldsymbol{X}_T = (X_1, X_2)^T \sim \mathcal{N}_2(\boldsymbol{\mu}, \Sigma_T).$$

The drift is the usual Black-Scholes drift $\mu_i = (r - 0.5\sigma_i^2)T$ and the covariance matrix can be written as

$$\Sigma_T = \begin{pmatrix} \sigma_1^2 & \varrho\sigma_1\sigma_2 \\ \varrho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} T \quad \text{for} \quad \varrho \in \{0, 0.4\}.$$

The smoothing the payoff approach leads directly to a quadrature problem in an independent variable $\boldsymbol{Z}$. For the full tensor quadrature without smoothing, we introduce an appropriate linear transformation to obtain an integration problem with independent variables. In the case of positive correlation $\varrho = 0.4$ we can rewrite $\boldsymbol{X}_T$ as

$$\boldsymbol{X}_T = \boldsymbol{\mu} + \begin{pmatrix} 1 & 0 \\ \varrho\frac{\sigma_2}{\sigma_1} & \sqrt{1-\varrho^2} \end{pmatrix} \boldsymbol{Z} \quad \text{for} \quad \boldsymbol{Z} \sim \mathcal{N}_2(\boldsymbol{0}, D) \quad \text{with} \quad D = \operatorname{diag}(\sigma_1^2 T, \sigma_2^2 T).$$

We can now interpolate in $\boldsymbol{Z}$ instead of $\boldsymbol{X}_T$. We use $M = 3, 5, 9, 17, 33$ quadrature points per dimension and choose an integration domain based on $k = 4$ times the standard deviation for the non-smoothed integrand and $k = 4, 5, 6, 7, 8$ for the smoothed problem. For the Monte Carlo simulation we use

$$M = 1{,}000,\ 4{,}000,\ 16{,}000,\ 64{,}000,\ 256{,}000 \tag{5.10}$$

simulation paths. As a comparison we calculate a reference price using *Matlab*'s integration routine *quadgk* an absolute error tolerance of 10^{-14} and 10 times the standard deviation as domain. We obtain a reference value of 8.566232... in the case of zero correlation and 9.783636... in the case of positive correlation. We verify if the reference price computed with the quadrature approach is in the confidence interval of a Monte Carlo pricer with the highest number of sample paths.

Figure 5.13 shows a comparison of the error decay of the dynamic Chebyshev, the Gauss-Hermite and the Clenshaw-Curtis quadrature using the smoothing and without

smoothing. The left plot displays the error decay for zero correlation and the right plot displays the same error decay for positive correlation. We observe a significant faster error decay when the smoothing is applied then without smoothing in both cases. For example, an accuracy below 10^{-2} requires 100 times more nodal points if no smoothing the payoff is explored. The significant improvement comes from the smoothness of the Black-Scholes price as integrand as well as from the dimension reduction from a bivariate integration problem to an univariate integration problem. The differences between the dynamic Chebyshev method and Clenshaw-Curtis integration are relatively small for the zero correlation case. In the general case of a positive correlation the dynamic Chebyshev method is able to achieve a significantly higher accuracy for the same number of nodal points. For 17 quadrature points the dynamic Chebyshev quadrature is around two orders of magnitude more accurate than Clenshaw-Curtis. The Gauss-Hermite quadrature outperforms the other two integration routines in both scenarios.

The higher λ_1 in Equation (5.9) the smoother is the resulting integrand. The dynamic Chebyshev method seems to profit more from this smoothness and is less affected from a small λ_2 then the Clenshaw-Curtis quadrature. We observed similar effects for the zero correlation case if σ_1 was bigger than σ_2.

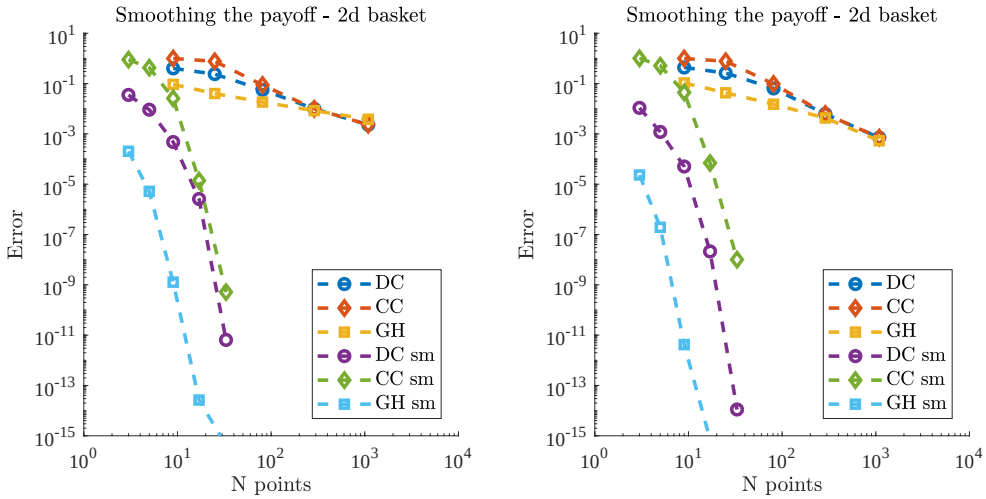

Figure 5.13: Error decay of the Gauss-Hermite (GH), dynamic Chebyshev (DC) and Clenshaw-Curtis (CC) quadrature for a basket call option with and without smoothing the payoff. Correlation is assumed to be zero (left plot) and positive (right plot).

Figure 5.14 shows the comparison of the dynamic Chebyshev quadrature with a Monte-Carlo integration. In combination with the smoothing, the quadrature approach is able to reach a relative accuracy of 10^{-10} in less than 10^{-4} seconds, In contrast, the Monte Carlo integration is not able to reach an accuracy of 10^{-4} in 10^{-2} seconds. The smoothing has reduced the relative error of the Monte Carlo integration in absolute

terms but the order of convergence remains the same. In contrast, the differences in the runtimes between dynamic Chebyshev and Gauss-Hermite quadrature are relatively small, see Figure 5.15.

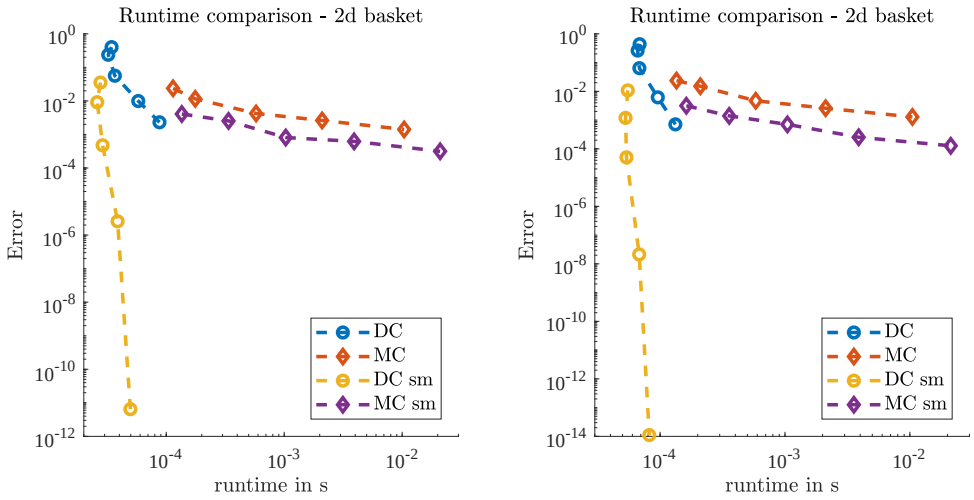

Figure 5.14: Performance comparison of the dynamic Chebyshev (DC) quadrature and Monte Carlo simulation for a basket call option with and without smoothing the payoff. Correlation is assumed to be zero (left plot) and positive (right plot).

Comparison with a benchmark method

Next, we compare the error decay of the three different quadrature approaches for the smoothed integrand with a bivariate benchmark pricer. As a benchmark we use the bivariate COS method introduced in [109]. For an (arithmetic) basket call option we obtain the following parameters

$$\boldsymbol{S}_0 = (110, 90), \quad \sigma_1 = 0.3, \quad \sigma_2 = 0.2, \quad \varrho = 0.25, \quad r = 0.04, \quad T = 1, \quad K = 100 \tag{5.11}$$

from parameter set I in [109] and we assume equal weights. They provide a reference price of $Price = 10.173230$ which we use to validate or own reference price. Moreover, they provide the absolute errors of the bivariate COS method for increasing number of terms of the series expansions, see Table 5-F.

We run the three quadrature approaches for $M = 3, 5, 9, 17, 33, 65$ quadrature points and choose an integration domain based on $k = 4, 5, 6, 7, 8, 9$ times the standard deviation. Using the reference method as for the previous experiment, we obtain a reference price of 10.173230... that matches the reference prices provided by [109].

Figure 5.15 shows the error decay of all three quadrature approaches as a function

$N_1(=N_2)$	10	20	40	80
total pts	100	400	1600	6400
error	$2.98 \cdot 10^{0}$	$1.91 \cdot 10^{-1}$	$4.82 \cdot 10^{-5}$	$1.29 \cdot 10^{-6}$

Table 5-F: Absolute error of the bivariate COS method for a European basket call option, see Table 4 in [109] for parameter $\mathcal{Q} = 1{,}000$.

of the total points and a function of the runtime. Additionally, Table 5-G displays the number of quadrature points and the error of the three approaches. All three methods are able to achieve an accuracy close to machine precision with only 65 quadrature weights and a runtime of around 10^{-4} seconds. Again, the Gauss-Hermite quadrature converges the fastest and is able to achieve an accuracy below 10^{-10} with only 17 quadrature points and just 9 quadrature points are already enough to achieve an accuracy below 10^{-6}. In comparison, Tabel 5-F shows that the bivariate COS method needs 6400 points in total to achieve an accuracy of 10^{-6}. In terms of runtimes, the Clenshaw-Curtis quadrature is the fastest and the dynamic Chebyshev and the Gauss-Hermite quadrature are slightly slower. Here, the performance differences are the computation of the quadrature nodes and weights for the different methods. For instance, the weights of the Clenshaw-Curtis quadrature depend only on N whereas the weights of the dynamic Chebyshev quadrature depend on μ and σ. The differences between the three methods are however small. In contrast, [109] report a runtime of $2 \cdot 10^{-1}$ seconds (also in *Matlab*) for this particular example with $N_1 = N_2 = 80$. We expect that the quadrature methods would still be a factor of two to three orders of magnitude faster if the comparison were done on the same machine.

quad. pts	3	5	9	17	33	65
error CC	$7.50 \cdot 10^{0}$	$3.20 \cdot 10^{0}$	$8.84 \cdot 10^{-2}$	$2.29 \cdot 10^{-3}$	$4.51 \cdot 10^{-7}$	$2.24 \cdot 10^{-13}$
error DC	$4.88 \cdot 10^{-3}$	$1.70 \cdot 10^{-2}$	$1.14 \cdot 10^{-2}$	$1.35 \cdot 10^{-4}$	$3.16 \cdot 10^{-10}$	$2.49 \cdot 10^{-14}$
error GH	$3.12 \cdot 10^{-3}$	$2.95 \cdot 10^{-4}$	$5.81 \cdot 10^{-7}$	$3.24 \cdot 10^{-11}$	$5.33 \cdot 10^{-15}$	$7.11 \cdot 10^{-14}$

Table 5-G: Absolute error of the Clenshaw-Curtis (CC), dynamic Chebyshev (DC) and Gauss-Hermite (GH) quadrature for a European basket call option on two stocks.

In this example, the three quadrature approaches are advantegous in comparison to the bivariate COS method of [109]. All three quadrature approaches achieved the same accuracy as the COS method using only 33 points in total instead of 80 points per dimension. We emphasize that our new method is specifically designed for the pricing of basket options in the multivariate Black-Scholes method whereas the bivariate COS method can be applied in a variety of two-factor models and for different payoff profiles.

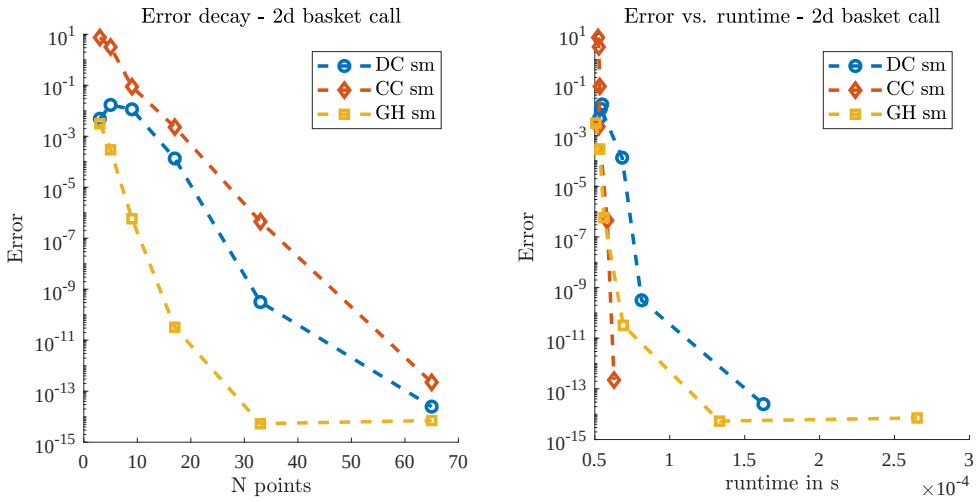

Figure 5.15: Comparison of the Gauss-Hermite (GH), dynamic Chebyshev (DC) and Clenshaw-Curtis (CC) quadrature for a basket call option with smoothing the payoff and assuming positive correlation. Error decay in absolute terms is shown as a function of the number of quadrature points (left) and of the runtime (right).

Note that we ignored for simplicity the influence of the parameter $\mathcal{Q}$ in [109] that refers to the number of points used for calculating the coefficients of the cosine expansion and focused only on the number of coefficients/nodal points. As stated in [109], the numerical computation of the coefficients is the "most time-consuming part" and one reason for the slower runtimes. Overall, this numerical experiments is a further indicator of the potential of the new quadrature approaches for basket options.

3d basket option

Next, we consider a basket call option on three assets in a multivariate Black-Scholes model with covariance matrix

$$\Sigma = \begin{pmatrix} 0.0625 & 0.0281 & 0.0313 \\ 0.0281 & 0.0625 & 0.0406 \\ 0.0313 & 0.0406 & 0.0625 \end{pmatrix} T.$$

For a basket option on three assets the resulting smoothed problem is then a bivariate integration problem. We use again $M = 3, 5, 9, 17, 33$ quadrature points per dimension and choose an integration domain based on $k = 4$ times the standard deviation for the non-smoothed integrand and $k = 4, 5, 6, 7, 8$ for the smoothed problem. For the Monte Carlo simulation we use increasing numbers of simulation paths as stated in (5.10). For the error analysis we calculate a reference price using the Clenshaw-Curtis quadrature for

the smoothed integrand with $M = 257$ points per dimension and 10 times the standard deviation as domain. We obtain a reference value of 9.712263. We verify if the reference price computed with the quadrature approach is in the confidence interval of a Monte Carlo pricer with the highest number of sample paths.

The left plot in Figure 5.16 shows the error decay of the three quadrature methods. This numerical experiment confirms again the high potential of the smoothing for basket options. The differences between the three quadrature approaches becomes more relevant in comparison to the bivariate example. The Gauss-Hermite quadrature requires the fewest points for the same level of accuracy and is able to reach an accuracy of 10^{-8} with a total of just 25 quadrature points. The dynamic Chebyshev method shows a slower error decay than the Gauss-Hermite but outperforms the Clenshaw-Curtis quadrature. It achieves an error of $4 \cdot 10^{-10}$ with only 289 total points or 17 points per dimension whereas the Clenshaw-Curtis quadrature has still an error of $2 \cdot 10^{-4}$ for the same number of points.

The right plot in Figure 5.16 shows a comparison of the dynamic Chebyshev quadrature with a Monte Carlo integration in terms of runtime vs. accuracy. Combined with the smoothing, the dynamic Chebyshev quadrature is significantly more efficient then the Monte Carlo integration. It is faster and at same time also more accurate. For the Monte Carlo approach itself, the smoothing has a smaller influence. It can reduce the overall level of the error but the smoothing has no influence on the speed of convergence. This is in line with the findings of [11]. Moreover, it is more costly to evaluate the smoothed integrand than the original payoff function.

5d basket option

As a next example, we consider a basket call option on five assets, Hence, the resulting smoothed problem is still a four-dimensional quadrature problem and a computational challenge for standard quadrature techniques. Therefore, we want to investigate if the full-tensor quadrature approaches are still efficient. Again, we consider a multivariate Black-Scholes model with covariance matrix given by

$$\Sigma = \begin{pmatrix} 0.0625 & 0.0250 & 0.0187 & 0.0281 & 0.0313 \\ 0.0250 & 0.0625 & 0.0219 & 0.0500 & 0.0406 \\ 0.0187 & 0.0219 & 0.0625 & 0.0125 & 0.0375 \\ 0.0281 & 0.0500 & 0.0125 & 0.0625 & 0.0313 \\ 0.0313 & 0.0406 & 0.0375 & 0.0313 & 0.0625 \end{pmatrix} T.$$

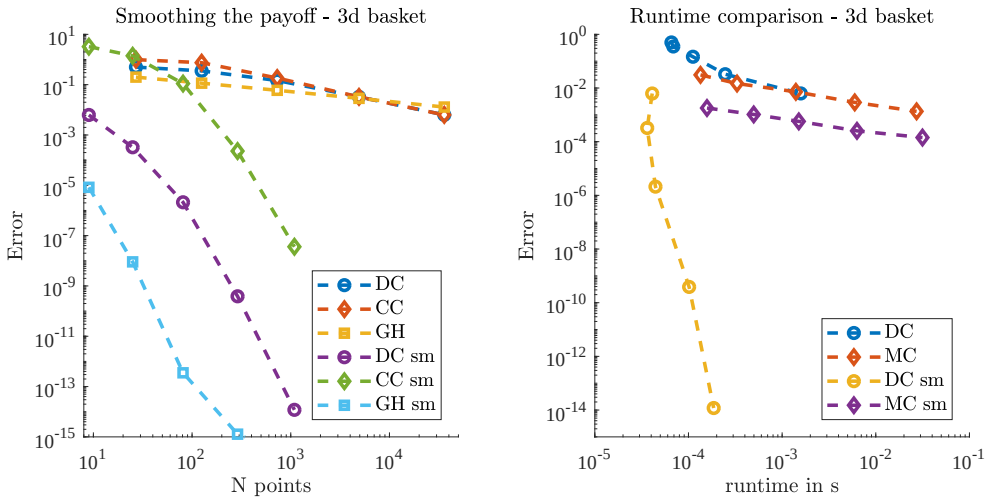

Figure 5.16: Error decay of the dynamic Chebyshev (DC), Clenshaw-Curtis (CC) and Gauss-Hermite (GH) quadrature for a basket call option on three assets (left plot) and performance comparison of the dynamic Chebyshev quadrature with a Monte Carlo simulation (right plot).

We use $M = 3, 5, 9, 17$ quadrature points per dimension and choose an integration domain based on $k = 4$ times the standard deviation for the non-smoothed integrand and $k = 4, 5, 6, 7$ for the smoothed problem. For the Monte Carlo simulation we use increasing numbers of simulation paths as stated in (5.10). For the error analysis we calculate a reference price using the Clenshaw-Curtis quadrature for the smoothed integrand with $M = 129$ points per dimension and 10 times the standard deviation as domain. We obtain a reference value of 9.051446. We verify if the reference price computed with the quadrature approach is in the confidence interval of a Monte Carlo pricer with the highest number of sample paths.

The left plot in Figure 5.17 shows the error decay of the three quadrature methods with and without smoothing. The left plot shows the resulting error decay for uncorrelated assets and the right plot shows the corresponding error decay for correlated assets. In this example, we observe significantly bigger differences between the three quadrature approaches for the smoothed integrand. Again, the Gauss-Hermite quadrature has the lowest error and is able to achieve a high accuracy with only 9 points per dimension. The Clenshaw-Curtis quadrature starts at a relatively high error level and needs therefore more points to achieve the same level of accuracy. The actual speed of convergence is then essentially the same as for the dynamic Chebyshev quadrature.

The pricing of basket options on five assets is usually done using Monte Carlo or quasi Monte Carlo methods, especially if speed is essential. Classical quadrature techniques are often more accurate but also significantly slower. The right plot in Figure 5.17 shows

a comparison of the dynamic Chebyshev quadrature and the Gauss-Hermite quadrature with a Monte Carlo integration. Combined with the smoothing, both quadrature techniques are more efficient then the Monte Carlo integration. They are able to achieve a significantly higher accuracy in the same runtime. For example the Gauss-Hermite quadrature is able to achieve an accuracy below 10^{-10} in under 10^{-3} seconds using only 6561 quadrature points in total. In contrast, the Monte Carlo simulation is only able to reach an accuracy of 10^{-2} in the same time. While the higher accuracy of the quadrature approaches was expected, it is surprising how fast both approaches are.

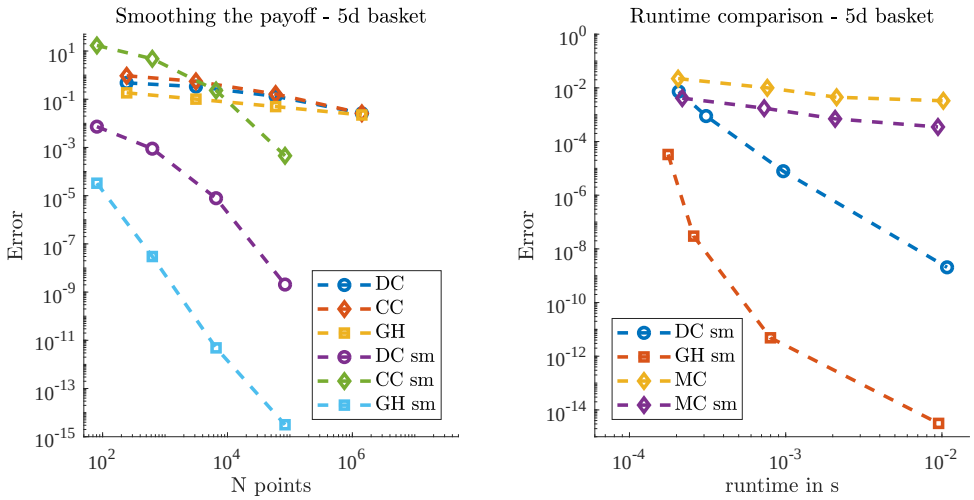

Figure 5.17: Error decay of the dynamic Chebyshev (DC), Clenshaw-Curtis (CC) and Gauss-Hermite (GH) quadrature for a basket call option on five assets (left plot) and performance comparison of the dynamic Chebyshev quadrature and the Gauss-Hermite quadrature with a Monte Carlo simulation (right plot).

5.3.4 Merits and limitations of the quadrature approaches

The presented smoothing the payoff for a basket option combined with an efficient quadrature technique provided promising numerical results for dimensions 2 to 5. When using Gauss-Hermite quadrature often only 5 nodal points per dimension were required in order to achieve a satisfactory accuracy. For the dynamic Chebyshev quadrature between 9 and 17 points per dimension achieved a similar accuracy. Even for a basket option on five assets, the quadrature approaches were as fast as a Monte Carlo simulation while achieving a several orders of magnitude higher accuracy. This high efficiency due to the smoothness of the integrand allows us to extend the method from low-dimensions to moderately high dimensions. This leads naturally to the following question: *Up to which dimension can we still employ the full tensor quadrature approach?*

Considering a basket option on eight assets leads to a 7 dimensional integration problem. Applying 5 points per dimension results in $8 \cdot 10^4$ points overall, still significant

less then a standard Monte Carlo simulation with a comparable accuracy. However, at some point, the full tensor of n^{d-1} points is now longer feasible to compute, even if n is as low as 5 or only 3 in some cases. In this situation an additional dimension reduction technique has to be employed. [11] showed that adaptive sparse grids are able to deliver accurate basket option prices in up to 25 dimensions. However, they also state that the construction of the adaptive sparse grid comes with a large numerical overhead.

Here, we want to investigate an alternative candidate that could reduce the computational complexity and the storage requirements. [57] suggest the use of low-rank tensors in the tensor train format (TT-format) and a tensor completion algorithm for the interpolation of European option prices. See Section 2.2.2 for a short introduction to low-rank tensor compression. We want to investigate if this can also be used to improve the Gauss-Hermite or dynamic Chebyshev quadrature. Depending on the structure of the tensor, the TT-format compression reduces the storage requirements tremendously. For example, in $d = 10$ [11] were able to approximate a tensor of size 6^{10} with less than 1,000 points and achieved a pricing error of just $5.6 \cdot 10^{-4}$, see Table 6 and Table 7 in [57]. We want to know if the tensor of function values of the smoothed problem admits a similar low-rank structure.

Numerical investigation of the low-rank structure

We consider a basket call option for $d = 8$ and $d = 10$ underlyings in a multivariate Black-Scholes model where the stocks are correlated. The correlation matrix is computed using the algorithm proposed in [36] that relies only on the specification of $d - 1$ parameters. We run the Gauss-Hermite quadrature and the dynamic Chebyshev quadrature for an increasing number of nodal points and compute the function values of the integrand as a full tensor. Then we compress this tensor using the tensor train Matlab toolbox of [102] for an error tolerance ε. The basket option price is then computed using the full tensor and tensor times matrix multiplication and using the low-rank tensor. We compare the size of both tensors as well as the rank-structure and the resulting pricing error.

For $d = 8$ we use $M = 3, 5, 7, 9$ points per dimension and an integration domain based on $k = 4, 4, 5, 5$ times the standard deviation. The reference price, computed with the Gauss-Hermite quadrature using 17 points per dimension, is given by 6.956891. Table 5-H shows the low-rank structure for the dynamic Chebyshev quadrature and $d = 8$. We see that for $n = 9$ points, the TT-format requires 1% of the original tensor points and the resulting pricing error is only slightly worse. Table 5-I shows the same low-rank structure for the Gauss-Hermite quadrature and $d = 8$. We observe, that Gauss-Hermite quadrature is again more accurate but is also less suitable for compression. In order to obtain a similar pricing error as the full tensor it still needs around 10% of the points.

n	rank	total	compression	ε	log-err LR	log-err FT
3	(1,3,9,14,12,6,3,1)	1251	57.2%	10^{-3}	-1.71	-1.63
5	(1,5,18,27,22,10,4,1)	7195	9.21%	10^{-4}	-3.14	-3.07
7	(1,7,21,30,24,11,4,1)	12712	1.54%	10^{-4}	-3.54	-3.31
9	(1,9,32,59,45,17,5,1)	51255	1.07%	10^{-5}	- 4.27	-4.62

Table 5-H: Low-rank structure of the 7-dimensional integrand at Chebyshev points of a basket option on 8 assets after smoothing the payoff. Comparison of the pricing error in a log scale using low-rank compression (LR) and without low-rank compression (FT).

n	rank	total	compression	ε	log-err LR	log-err FT
3	(1,3,6,8,7,6,3,1)	564	25.8%	10^{-3}	-3.32	-3.78
5	(1,5,20,34,28,13,4,1)	10785	13.8%	10^{-5}	-6.14	-6.90
7	(1,7,35,74,60,21,6,1)	60718	7.37%	10^{-6}	-7.89	-7.38
9	(1,9,67,210,157,40,8,1)	488340	10.21%	10^{-8}	- 10.37	-9.23

Table 5-I: Low-rank structure of the 7-dimensional integrand at Gauss-Hermite points of a basket option on 8 assets after smoothing the payoff. Comparison of the pricing error in a log scale using low-rank compression (LR) and without low-rank compression (FT).

For $d = 10$ we consider only the Gauss-Hermite quadrature and we use $M = 2, 3, 4, 5, 6$ points per dimension. The reference price, also computed with the Gauss-Hermite quadrature using 9 points per dimension, is given by 6.824155. Table 5-J shows the corresponding low-rank structure for a basket option on 10 assets using the Gauss-Hermite quadrature. Figure 5.18 displays the error decay of the low-rank and the full-tensor quadrature. In this experiment, we observe a better compression and for example only 6,060 points are enough to achieve a pricing error below 10^{-4}. However, in all experiments we observe that the maximal rank is relatively high compared to the number of points n. This is an indicator that the underlying tensor does not admit a good low-rank structure. For example for $n = 5$ the maximal rank of the 9 dimensional Gauss-Hermite low-rank tensor is already 53.

Overall, the experiments confirm that the low-rank compression is able to reduce the storage requirements of the full tensor. However, the resulting maximal rank is still relatively large for the approximation of the smoothed integrand. We do not observe a compression factor in a similar order of magnitude as [57] observed for the low-rank approximation of option prices. In practical applications we do not want to compute the full tensor first and then compress it into a low-rank approximation. Instead, we

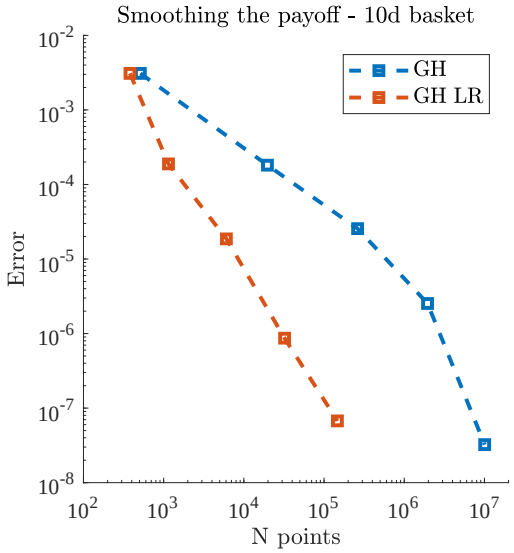

Figure 5.18: Error decay of Gauss-Hermite quadrature for a basket call option with $d = 10$, comparison of the low-rank approximation with the full-tensor.

n	rank	total	compression	ε	err LR	err FT
2	(1,2,4,6,8,6,5,4,2,1)	380	74.2%	10^{-3}	-2.51	-2.51
3	(1,3,7,9,10,9,7,5,3,1)	1149	5.84%	10^{-3}	-3.72	-3.72
4	(1,4,13,21,23,17,12,7,3,1)	6060	2.31%	10^{-4}	-4.73	-4.59
5	(1,5,20,41,53,39,24,12,4,1)	32205	1.65%	10^{-5}	- 6.06	-5.59
6	(1,6,31,78,113,79,42,17,5,1)	146838	1.46%	10^{-6}	-7.17	-7.49

Table 5-J: Low-rank structure of the 9-dimensional integrand at Gauss-Hermite points of a basket option on 10 assets after smoothing the payoff. Comparison of the pricing error in a log scale using low-rank compression (LR) and without low-rank compression (FT).

want to use only a few function values and directly obtain a low-rank tensor. This can be done using the completion algorithm proposed in [123]. Since the optimal rank structure is unknown beforehand, it has to be found via a rank adaptive version of the completion algorithm, see [57]. In our example, the higher rank of the tensors make the rank adaptive tensor completion algorithm of [57] unsuitable for this specific problem. Even though the TT-format offers some compression compared to the full-tensor it is difficult to explore this theoretical improvement in an efficient way.

5.4 Smoothing in a dynamic framework

The motivation for the dynamic Chebyshev quadrature came from the idea that pricing a European option is essentially the first time step of the dynamic Chebyshev method for early-exercise options. Thus, it seems interesting to go back from the European case to a more general dynamic programming problem that includes early-exercise options as well as the possibility to have time-dependent coefficients in the Black-Scholes model. In this section we want to combine the smoothing the payoff concept and the multivariate dynamic Chebyshev algorithm introduced in Section 5.1. Before we do so, we have to investigate if the smoothing the payoff also works for a small time step.

5.4.1 Basket options with short maturities

In order to numerically investigate if the smoothing also works for short maturities, we re-run the experiments for $d = 2$ and $d = 3$ and fix $T = 0.02$ instead of $T = 1$. This corresponds to a maturity of one week or a Bermudan option that is weekly exercisable. We assume that in both scenarios, the underlying assets are correlated. It is known that the density of the log-returns becomes steeper for small time-scaled volatilities and there might be a smaller smoothing effect. On the other hand, the integration domain becomes smaller and this makes a quadrature more feasible. Moreover, we do the same experiments for a basket put option in order to investigate if the smoothing effect holds for the put option as well.

We use the same specifications as in the previous section for the bivariate and trivariate case. For $d = 2$, we obtain a reference price of 1.209971 for the call and 1.149989 for the basket put. For $d = 3$, we obtain a reference price of 1.200553 for the call and 1.140571 for the basket put. Figure 5.19 shows the resulting error decay for a bivariate basket call option (left plot) and a basket call option on three underlyings (right plot). We observe that the smoothing the payoff technique works even better for short maturities. The Gauss-Hermite quadrature requires only 3 points (if $d = 2$) or 9 points (if $d = 3$) in order to obtain a relative error of 10^{-7}. Similarly, the dynamic Chebyshev quadrature is able to achieve very satisfactory error levels for a low number of quadrature points. In contrast, the Clenshaw-Curtis quadrature requires significantly more points to achieve very accurate results.

Figure 5.20 shows the same error decay for a bivariate basket put option (left plot) and a basket put option on three underlyings (right plot). We observe an almost identical convergence behaviour for the basket put option as for the basket call. This confirms numerically that the smoothing is applicable for both payoff types. Overall, we observe again that the Gauss-Hermite quadrature converges the fastest if a single European

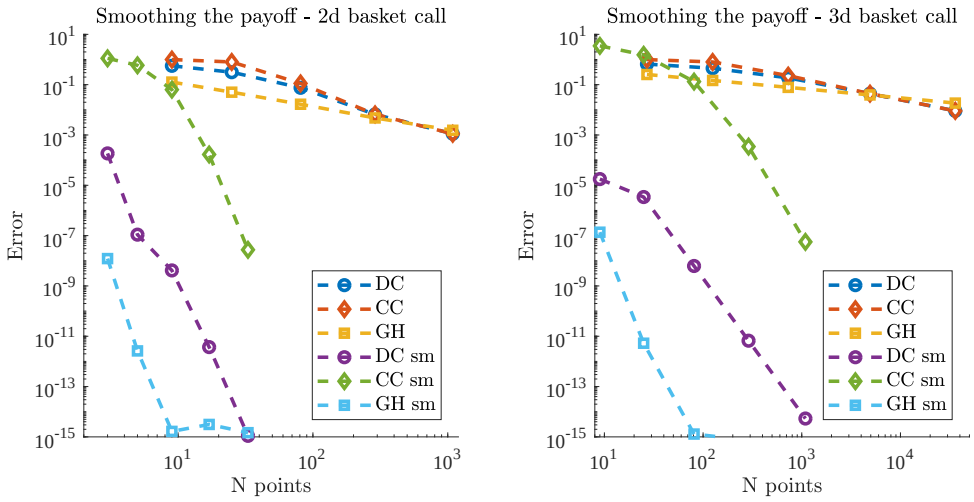

Figure 5.19: Error decay of the dynamic Chebyshev (DC), Clenshaw-Curtis (CC) and Gauss-Hermite (GH) quadrature for a basket call option with and without smoothing the payoff. Maturity is $T = 0.02$ and we consider a basket on two assets (left plot) and three assets (right plot).

basket option is priced. In the next section, we will see that the dynamic Chebyshev quadrature can be modified to price Bermudan basket options as well. A similar pricing approach for Bermudan basket options using the Gauss-Hermite quadrature is however not possible.

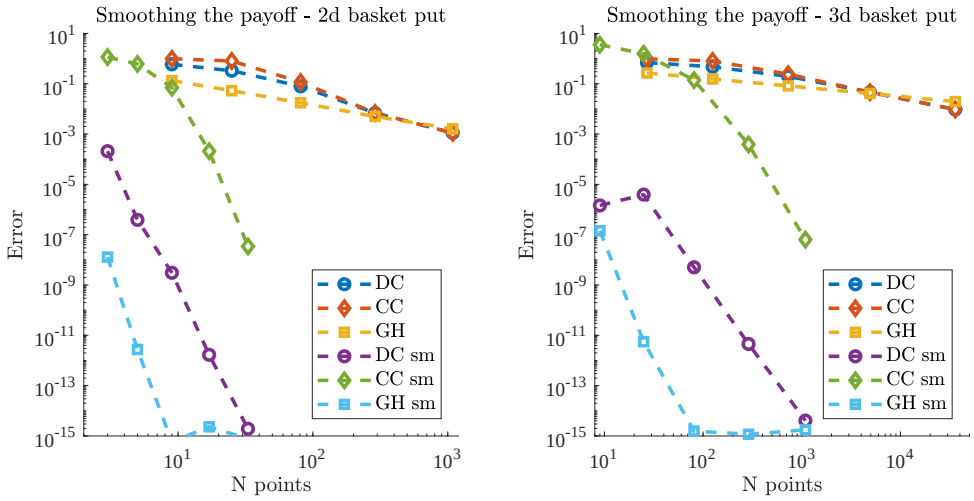

Figure 5.20: Error decay of the dynamic Chebyshev (DC), Clenshaw-Curtis (CC) and Gauss-Hermite (GH) quadrature for a basket put option with and without smoothing the payoff. Maturity is $T = 0.02$ and we consider a basket on two assets (left plot) and three assets (right plot).

Interpolation of basket options

The goal of the section is to extend the smoothing the payoff idea from European option pricing to the pricing of early-exercise options. In the dynamic Chebyshev pricing algorithm, we need an interpolation of the option price in each time step. To fit into this set-up, the smoothing the payoff concept has to be slightly modified in order to obtain an interpolation of the European option price after one time step. From Proposition 13 for the price of a basket call option with equal weights follows

$$\begin{aligned}&\mathbb{E}[e^{-r\Delta t}g(\boldsymbol{X}_T)|\boldsymbol{X}_{t_{n_T-1}}=\boldsymbol{x}_0]=\mathbb{E}\big[C_{BS}(h(\boldsymbol{Z})e^{\lambda_1^2/2},e^{-r\Delta t}K,\lambda_1)\big]\\&\text{with}\quad h(\boldsymbol{z})=\sum_{i=1}^d\beta_i\exp\Big(\sum_{j=2}^d V_{i,j}z_j\Big)=(1/d)\sum_{i=1}^d e^{x_0^i}e^{-0.5\sigma_i^2\Delta t}\exp\Big(\sum_{j=2}^d V_{i,j}z_j\Big)\end{aligned}$$

for $\Delta t = t_{n_T} - t_{n_T-1}$. If we want to interpolate the price of a basket option we require nodal values

$$\mathbb{E}[e^{-r\Delta t}g(\boldsymbol{X}_T)|\boldsymbol{X}_{t_{n_T-1}}=\boldsymbol{x_k}]$$

at the multivariate Chebyshev points $\boldsymbol{x_k}$. These values are then used to compute the coefficients of the Chebyshev interpolation. A naive approach would be to compute the conditional value for each $\boldsymbol{x_k}$ separately. This would however be very slow and there is a more efficient approach for the dynamic Chebyshev quadrature. Consider the tensor notation of the quadrature problem as in (5.8). The vector of quadrature weights can then simply be replaced by a matrix of weights as defined in (5.2).

A similar extension of the Gauss-Hermite quadrature is not feasible since changing the mean or volatility will yield a different set of quadrature points. We illustrate this difference between the Gauss-Hermite quadrature and the dynamic Chebyshev quadrature at a simplified example. Assume we want to interpolate $V_0 : \boldsymbol{x} \mapsto \mathbb{E}[V_1(\boldsymbol{x}+\boldsymbol{Z})]$ for $\boldsymbol{Z} \sim \mathcal{N}(0, \mathbb{1}_d)$. This means that we need to compute the values of V_0 at a grid of nodal points $\boldsymbol{x_k} = (x_{k_1}, \ldots, x_{k_d})$. With the dynamic Chebyshev approach we obtain

$$V_0(\boldsymbol{x_k}) = \mathbb{E}[V_1(\boldsymbol{x_k}+\boldsymbol{Z})] \approx \sum\nolimits_{\boldsymbol{j}} w_{j_1,k_1}\ldots w_{j_d,k_d}V_1(\boldsymbol{x_j}),$$

see Section 5.1.4. This approach requires $(N+1)^d$ values $V_1(\boldsymbol{x_j})$ and d times $(N+1)^2$ weights w_{j_i,k_i}. In contrast, using Gauss-Hermite quadrature with quadrature nodes $\boldsymbol{z_j}$ we obtain

$$V_0(\boldsymbol{x_k}) = \mathbb{E}[V_1(\boldsymbol{x_k}+\boldsymbol{Z})] \approx \sum\nolimits_{\boldsymbol{j}} w_{j_1}\ldots w_{j_d}V_1(\boldsymbol{z_j}+\boldsymbol{x_k}),$$

where the starting value $\boldsymbol{x_k}$ is shifting the nodal points $\boldsymbol{z_j}$ instead of changing the weights. Thus, we require a significantly higher number of $(N+1)^{2d}$ values $V_1(\boldsymbol{z_j}+\boldsymbol{x_k})$ and d times $(N+1)$ weights w_{j_i}. Even though the Gauss-Hermite points achieve the fastest convergence for European basket options they are not the most suitable choice for a dynamic pricing algorithm.

In Section 5.1.2 we reduced the multivariate dynamic Chebyshev algorithm to the case of an independent process if the model is multivariate normally distributed. We used that if $\boldsymbol{X} \sim \mathcal{N}_d(\boldsymbol{\mu}, \Sigma)$ then $\boldsymbol{X}$ is equal in distribution to $\boldsymbol{\mu}+O\boldsymbol{Z}$ where $\boldsymbol{Z} \sim \mathcal{N}_d(\boldsymbol{0}, \Lambda)$ for a diagonal matrix Λ, an orthogonal matrix O and $\Sigma = O\Lambda O^T$. This general decomposition can easily be replaced by the covariance matrix decomposition $\Sigma = VDV^T$ introduced in Remark 6. By doing so, we can apply the smoothing the payoff concept in the first time step of the pricing algorithm. This leads to a smoother problem and reduces the required number of nodal points significantly. In this case, we interpolate in the independent variable $\boldsymbol{Z}$ and we replace $\boldsymbol{x_k}$ by starting values $\boldsymbol{\mu}+V\boldsymbol{z_k}$. See Section 5.1.2 for more details.

We note that the interpolation of a European basket option price can be interesting in its own. If the same basket option has to be priced for different starting values the interpolation can be evaluated instead. An example is the calculation of credit exposure as presented in Chapter 4. In Section 2.3.3, we have introduced the idea of a static Chebyshev method that interpolates an option price in its parameters. If the original pricer is slow, the Chebyshev approximation can lead to a significant performance gain. [57] extend this approach to higher dimensions and consider among other problems the pricing of basket options in a multivariate Black-Scholes model. The resulting algorithm enables a offline-online decomposition and speed-ups the pricing in the online step. However, the approach of [57] comes with the drawback of a computationally heavy offline phase. Moreover, it requires a benchmark pricer at the nodal points and is therefore limited by the maximal accuracy of the chosen benchmark approach, for example a Monte Carlo pricer. This benchmark approach could be replaced by the dynamic Chebyshev quadrature for basket options in the multivariate Black-Scholes model.

5.4.2 Numerical smoothing

So far, we have seen that the smoothing the payoff idea can be used in the first time step of the dynamic Chebyshev algorithm. By doing so, we omit the interpolation of the payoff function that has a kink at the strike. However, in the backward induction we still have the problem that the value function of an early-exercise option is only once continuously differentiable due to the maximum function. Here, we can also apply a sort of smoothing to improve the convergence behaviour. Consider the value function at time

point t_u given by

$$\boldsymbol{x} \mapsto V_{t_u}(\boldsymbol{x}) = \{g(\boldsymbol{x}), \mathbb{E}[V_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}]\}.$$

This function is continuously differentiable but in general not smoother. A direct interpolation requires therefore a relatively high number of nodal points in each dimension in order to obtain accurate results. However, we know that $\boldsymbol{x} \mapsto \mathbb{E}[V_{t_{u+1}}(\boldsymbol{X}_{t_{u+1}})|\boldsymbol{X}_{t_u} = \boldsymbol{x}]$ is smooth and $g(\boldsymbol{x})$ can be smoothed using the Gaussian density function. Let Σ be the covariance matrix of $\boldsymbol{X}_{\Delta t} = \boldsymbol{X}_{t_{u+1}} - \boldsymbol{X}_{t_u}$ and assume we have a decomposition $\Sigma = VDV^T$ as described in Remark 6. Then we can write

$$\begin{aligned}\mathbb{E}[V_{t_u}(\boldsymbol{X}_{t_u})|\boldsymbol{X}_{t_{u-1}} = \boldsymbol{x}] = \mathbb{E}[V_{t_u}(\boldsymbol{x} + \boldsymbol{X}_{\Delta t})] &= \mathbb{E}[V_{t_u}(\boldsymbol{x} + \mu_{\Delta t} + V\boldsymbol{Z})] \\ &= \mathbb{E}\big[\mathbb{E}[V_{t_u}(\boldsymbol{x} + \mu_{\Delta t} + Z_1\mathbf{1} + V_{:,2:d}\overline{\boldsymbol{Z}})|\overline{\boldsymbol{Z}}]\big]\end{aligned}$$

for a vector of independent random variables $\boldsymbol{Z} \in \mathbb{R}^d$ with $\overline{\boldsymbol{Z}} = (Z_2, \ldots, Z_d)$ and a column vector of ones $\mathbf{1}$. The notation $V_{:,2:d}$ refers to the columns 2 to d of the matrix V. The inner expectation is an univariate quadrature problem in Z_1 and the resulting conditional expectation

$$\mathbb{R}^{d-1} \ni \overline{\boldsymbol{Z}} \mapsto \mathbb{E}[V_{t_u}(\boldsymbol{x} + \mu_{\Delta t} + Z_1\mathbf{1} + V_{:,2:d}\overline{\boldsymbol{Z}})|\overline{\boldsymbol{Z}}]$$

is then a smooth function in the remaining variables. If V_{t_u} is not the payoff function there is unfortunately no closed form expression for the inner expectation as in Proposition 13. This motivates us to perform a numerical version of the smoothing the payoff idea. We propose to use a relatively higher number of nodal points N_1 in the first dimension and a lower number of nodal points $(N_2, \ldots, N_d)$ in the other dimensions in the dynamic Chebyshev algorithm. Due to its smoothness, a lower number of quadrature points should be sufficient to compute the outer expectation accurately. The resulting multivariate dynamic Chebyshev algorithm with numerical smoothing for a Bermudan put option is presented in Algorithm 6 and Algorithm 7.

This idea of numerically smoothing the value function or the payoff is not limited to basket options in a multivariate Black-Scholes model. In fact, whenever $\boldsymbol{X}_{t_{u+1}}|\boldsymbol{X}_{t_u} = \boldsymbol{x}$ is multivariate normally distributed variable we can explore this idea. Further examples are multivariate short rate models such as the two-factor Hull-White/Black-Karasinski model used for the pricing of callable bonds. Another possible example is an equity Black-Scholes model combined with stochastic volatility and stochastic interest rates.

Algorithm 6 Multivariate dynamic Chebyshev - pre-computation

Require: Stock price $\boldsymbol{S}_0 \in \mathbb{R}^d$, positive definite covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$, maturity T, interest rate r, Chebyshev degree $(N_1, \dots, N_d) \in \mathbb{N}^d$, number of time steps n_T

Define $\boldsymbol{\mu} \in \mathbb{R}^d$ with $\boldsymbol{\mu}_i \leftarrow (r - 0.5\Sigma_{i,i})$ and define $\Delta t \leftarrow T/n_T$
Decompose covariance matrix
  Find $\Sigma = VDV^T$ with $D = \operatorname{diag}(\lambda_1^2, \dots, \lambda_d^2)$ using Remark 6
Define $\boldsymbol{z}_0 \leftarrow V^{-1} \cdot (\log(\boldsymbol{S}_0) + \boldsymbol{\mu}T)$
for $i = 1, \dots, d$ **do**
  Fix domain
    $[\underline{z}_i, \overline{z}_i] \leftarrow [\boldsymbol{z}_0^i - k_{dom}\lambda_i, \boldsymbol{z}_0^i + k_{dom}\lambda_i]$
  Compute nodal points
    $z_k^i \leftarrow \underline{z}_i + 0.5(\cos(k\pi/N_i) + 1)(\overline{z}_i - \underline{z}_i)$, for $k = 0, \dots, N_i$
  Compute generalized moments $\Gamma^i \in \mathbb{R}^{(N_i+1)\times(N_i+1)}$
    $\mu_k^i \leftarrow 1 - 2(z_k^i + (V^{-1}\boldsymbol{\mu}\Delta t)_i - \overline{z}^i)/(\underline{z}^i - \overline{z}^i)$ and $\sigma^i \leftarrow \big(2\lambda_i\sqrt{\Delta t}\big)/\big(\overline{z}^i - \underline{z}^i\big)$
    $\Gamma_{j,k}^i \leftarrow \mathbb{E}[T_j(Z_k^i)]$ with $Z_k^i \sim \mathcal{N}\big(\mu_k^i, (\sigma^i)^2\big)$
  Compute quadrature weights
    Define $\mathcal{T}^i \in \mathbb{R}^{(N_i+1)\times(N_i+1)}$ with $\mathcal{T}_{j,k}^i \leftarrow (1/4)\cos(jk\pi/N_i)2^{\mathbb{1}_{0<j<N_i}}2^{\mathbb{1}_{0<k<N_i}}$
    $W^i \leftarrow \mathcal{T}^i \cdot \Gamma^i$
return weights $W^1, \dots, W^d$

5.4.3 Numerical investigation for Bermudan basket options

Next, we investigate the multivariate dynamic Chebyshev algorithm with smoothing in the first component numerically. We consider a basket put option with early-exercise feature for $d = 2$, $d = 3$ and $d = 4$. We investigate the convergence behaviour for an increasing number of nodal points and compare the standard dynamic Chebyshev approach with $N_1 = N_2 = \ldots = N_d$ to a numerical smoothing where N_1 is bigger than $N_2 = \ldots = N_d$. More precisely, we investigate the situation where $N_1 = N_2$, $N_1 = 2N_2$, $N_1 = 3N_2$ and $N_1 = 4N_2$. We refer to the last three ones as dynamic Chebyshev with numerical smoothing.

In our experiments we use the same model parameters as in previous experiments and assume that the underlyings are correlated. We assume that the Bermudan put option is weekly exercisable, i.e. 52 exercise rights per year. For $N_1 = kN_2$, we choose $N_2 \in \{4, 8, 16, 32\}$ if $k = 1, 2$ and $N_2 \in \{4, 8, 16\}$ otherwise. We fix an interpolation domain using 5 times the standard deviation of the underlying. The reference prices are computed using $N_1 = N_2 = 128$ and 6 times the standard deviation.

Moreover, we compare the dynamic Chebyshev approach with and without the smoothing to the least-squares Monte Carlo approach of [88]. We consider a least-squares approach with $1 + d + d^2$ basis functions consisting of a constant, the variable $x_1, \dots, x_d$ and all products $x_i x_j$ for $i, j \in \{1, \dots, d\}$.

Algorithm 7 Multivariate dynamic Chebyshev with numerical smoothing

Require: Stock price $\boldsymbol{S}_0 \in \mathbb{R}^d$, positive definite covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$, maturity T, interest rate r, Chebyshev degree $\boldsymbol{N} = (N_1, \ldots, N_d) \in \mathbb{N}^d$, number of time steps n_T, strike K and set of exercise dates $S_1, \ldots, S_m$

Obtain quadrature weights $W^1, \ldots, W^d$ using Algorithm 6

Define $\Delta t \leftarrow T/n_T$ and $\boldsymbol{\mu} \in \mathbb{R}^d$ with $\mu_i \leftarrow (r - 0.5\Sigma_{i,i})\Delta t$

Payoff values at nodal points

  For $0 \le \boldsymbol{k} \le \boldsymbol{N}$ compute nodal values $\mathcal{G}(\boldsymbol{z_k})$

  $\mathcal{G}(\boldsymbol{z_k}) \leftarrow \Big(K - (1/d)\sum_{i=1}^d \exp\big(\mu_i + \sum_{j=1}^d V_{i,j} z^j_{k_j}\big)\Big)^+$

Initial time step

  $\boldsymbol{\beta} \in \mathbb{R}^d$ with $\beta_i \leftarrow (1/d)\exp(-0.5\Sigma_{i,i}\Delta t)$

  For $0 \le \boldsymbol{k} \le \boldsymbol{N}$ compute nodal values $\mathcal{V}_{t_{n_T-1}}(\boldsymbol{z_k})$

  $h \leftarrow \sum_{i=1}^d \beta_i \exp\big(\sum_{j=2}^d V_{i,j} z^j_{k_j}\big)$

  $\mathcal{V}_{t_{n_T-1}}(\boldsymbol{z_k}) \leftarrow P_{BS}(\exp(z^1_{k_1} + (V^{-1}\boldsymbol{\mu}\Delta t)_1) h e^{\lambda_1^2 \Delta t/2}, e^{-r\Delta t}K, \lambda_1\sqrt{\Delta t})$

  if $t_{n_T-1} \in \{S_1, \ldots, S_m\}$ **then** $\mathcal{V}_{t_{n_T-1}}(\boldsymbol{z_k}) \leftarrow \max\{\mathcal{G}_{\boldsymbol{k}}, \mathcal{V}_{t_{n_T-1}}(\boldsymbol{z_k})\}$

Backward time stepping

for $u = n_T - 2, \ldots, 1$ **do**

  $\mathcal{V}_{t_u} \leftarrow \mathcal{V}_{t_{u+1}} \times_1 W^1 \times_2 \ldots \times_d W^d$

  if $t_u \in \{S_1, \ldots, S_m\}$ **then** $\mathcal{V}_{t_u} \leftarrow \max\{\mathcal{G}, \mathcal{V}_{t_u}\}$

Evaluate option price

  Define $\mathcal{T}^i \in \mathbb{R}^{(N_i+1)\times(N_i+1)}$ with $\mathcal{T}^i_{j,k} \leftarrow (1/4)\cos(jk\pi/N_i) 2^{\mathbb{1}_{0<j<N_i}} 2^{\mathbb{1}_{0<k<N_i}}$

  Coefficients $\mathcal{C}_0 \leftarrow \mathcal{V}_{t_0} \times_1 \mathcal{T}^1 \times_2 \ldots \times_d \mathcal{T}^d$

  For $i = 1, \ldots, d$ compute vector $\boldsymbol{T}^i$ with entries

  $T^i_j \leftarrow T_j\big(1 - 2\big(\sum_{k=1}^d V^{-1}_{i,k}(\log(S^i_0) - \mu_i) - \overline{z}_i\big)/(\overline{z}_i - \underline{z}_i)\big)$

  Price $\leftarrow \mathcal{C}_0 \times_1 T^1 \times_2 \ldots \times_d T^d$

For $d = 2$ we obtain for the basket put option a reference price of 7.103965. Figure 5.21 shows the error decay of the dynamic Chebyshev method (left plot) and a comparison with the least-squares Monte Carlo approach (right plot) for $d = 2$. We observe that the numerical smoothing is able to deliver a higher accuracy for the same total number of nodal points. Using around 1,000 grid points in total leads to an error of 10^{-4} for the numerical smoothing with $N_1 = 4N_2$ and slight below 10^{-2} without this smoothing, i.e. almost two orders of magnitude better. The versions with $N_1 = 2N_2$ and $N_1 = 3N_2$ are between the two extremes. From the right plot in Figure 5.21 we conclude that both versions of the dynamic Chebyshev method are significantly faster than the least-squares Monte Carlo approach. They admit a better convergence rate and the dynamic Chebyshev version with smoothing is at the same time almost three orders of magnitude faster as well as two orders of magnitudes more accurate.

For $d = 3$ we obtain for the basket put option a reference price of 7.038891. Figure 5.22 shows the same experiments for a basket option on three assets. The comparison

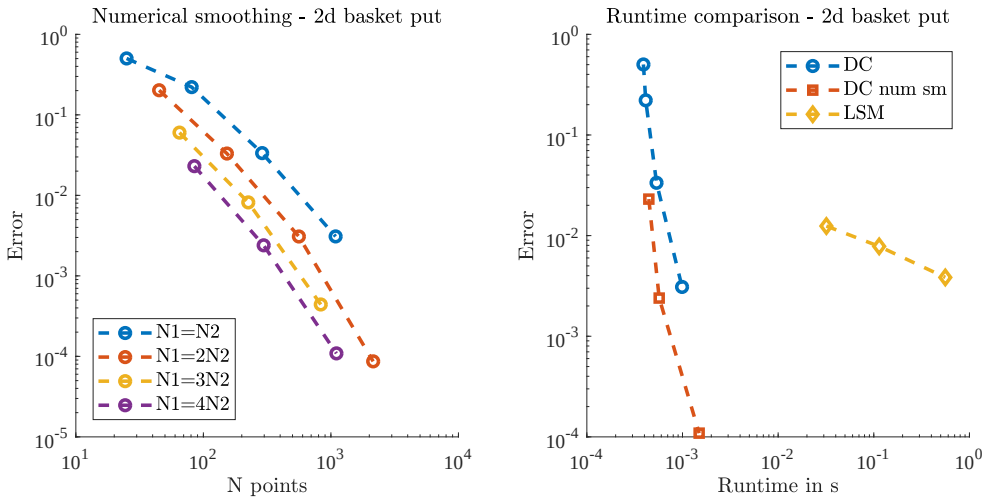

Figure 5.21: Left plot: Error decay of the multivariate dynamic Chebyshev (DC) with smoothing the payoff and different number of nodal points per dimension. Right plot: Comparison of the dynamic Chebyshev method with $N_1 = N_2$ and with $N_1 = 4N_2$ to the least-squares Monte Carlo (LSM) approach.

between the different versions provide similar results as for the bivariate basket option. Overall, the algorithm needs $(1 + N_2)$ times more nodal points and the runtime of the dynamic Chebyshev method in three dimensions scales accordingly. In contrast, the least-squares Monte Carlo method is only moderately effected by the increase of the dimension and its runtime scales linearly in d. The dynamic Chebyshev method in three dimensions is still two orders of magnitude faster than the Monte Carlo approach and at the same time also more accurate.

Since the results for two and three dimensions were promising we will extend the method one step further and we investigate a basket option on four assets. We assume that the covariance matrix in the multivariate Black-Scholes model is given by

$$\Sigma = \begin{pmatrix} 0.0625 & 0.0187 & 0.0281 & 0.0313 \\ 0.0187 & 0.0625 & 0.0500 & 0.0406 \\ 0.0281 & 0.0500 & 0.0625 & 0.0375 \\ 0.0313 & 0.0406 & 0.0375 & 0.0625 \end{pmatrix} T.$$

In this model we obtain a reference price of 6.877431 for the Bermudan put. We slightly modify the experiments and omit the cases where $N = 2N_2$ and $N_2 = 32$ as well as $N_1 = 4N_2$ and $N_2 = 16$. Figure 5.23 shows the resulting convergence analysis and the performance comparison with least-squares Monte Carlo. Similar to the step from

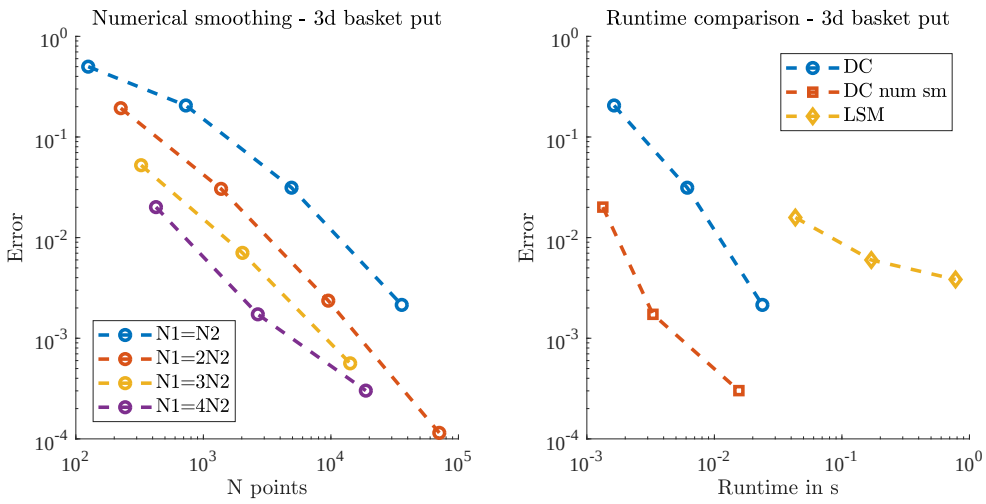

Figure 5.22: Left plot: Error decay of the multivariate dynamic Chebyshev (DC) with smoothing the payoff and different number of nodal points per dimension. Right plot: Comparison of the dynamic Chebyshev method with $N_1 = N_2$ and with $N_1 = 4N_2$ to the least-squares Monte Carlo (LSM) approach.

$d = 2$ to $d = 3$ we see that the number of points and the runtime increase by another factor of N_2. However, the dynamic Chebyshev method with smoothing is still able to outperform the least-squares Monte Carlo approach by delivering a higher accuracy in a shorter runtime.

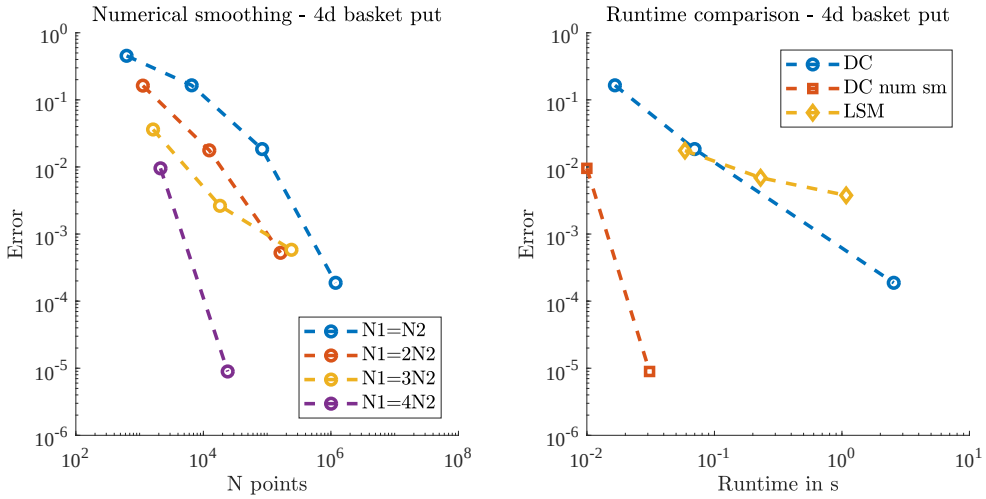

Figure 5.23: Left plot: Error decay of the multivariate dynamic Chebyshev (DC) with smoothing the payoff and different number of nodal points per dimension. Right plot: Comparison of the dynamic Chebyshev method with $N_1 = N_2$ and with $N_1 = 4N_2$ to the least-squares Monte Carlo (LSM) approach.

We have seen that the runtime and the number of nodal points of the dynamic

Chebyshev method increase roughly by one order of magnitude if we add an additional dimensions. The numerical experiments confirm that the method can be efficiently applied in up to four dimensions. Even for $d = 4$ the proposed method was faster than a least-squares Monte Carlo approach. For a pricing method operating on a tensor grid this is a relatively high dimensional extension without applying any additional dimension reduction techniques. The reason behind this is that combination of the smoothing the payoff in the first time step plus the numerical smoothing in the first dimension in the backward induction keeps the number of nodal points in all other dimensions comparably low. For higher dimensions however, a tensor of $(N_1+1)\times(N_2+1)^{d-1}$ elements becomes too large even if N_2 is small.

Comparison with benchmark method

Similarly to the experiments presented in Figure 5.15 for European basket options, we compare the bivariate dynamic Chebyshev method with smooothing to the bivariate COS method of [109]. We consider again the model parameter set (5.11) and price a Bermudan put option with 10 exercise rights. Table 5-K shows the accuracy in absolute terms of the bivariate COS method obtained form [109]. For the dynamic Chebyshev method we obtain a reference price of 6.6113. We use the dynamic Chebyshev without numerical smoothing and $N_1 = N_2$ for $N_2 = 8, 16, 32$ and the one with numerical smoothing $N_1 = 4N_2$ for $N_2 = 4, 8$. In both cases we choose a interpolation domain based on 6 times the standard deviation of the underlying. Figure 5.24 shows the resulting error decay of

$N_1(= N_2)$	40	80	160
total pts	1600	6400	25600
error	$2.33 \cdot 10^{-3}$	$4.66 \cdot 10^{-4}$	$2.86 \cdot 10^{-4}$

Table 5-K: Absolute error of the bivariate COS method for a Bermudan basket put option with 10 exercise rights, see Table 7 in [109].

the dynamic Chebyshev method with and without numerical smoothing, the bivariate COS method (values taken from Table 5-K) and of the least-squares Monte Carlo method. We observe that the dynamic Chebyshev method with numerical smoothing outperforms all three other methods. It admits a faster convergence behaviour and is therefore able to achieve the same accuracy as the 2d COS method using around two order of magnitudes fewer quadrature points. With only 297 total points ($N_1 = 32$ and $N_2 = 8$), the smoothed dynamic Chebyshev method achieves an absolute accuracy $2.4 \cdot 10^{-4}$. In contrast, the dynamic Chebyshev method without numerical smoothing needs 1,089 points for an accuracy of $1.3 \cdot 10^{-2}$ and the bivariate COS method reaches an accuracy of $2.9 \cdot 10^{-4}$ with 25,600 total points. The smoothing seems to work even better for a model where

σ_1 is bigger than σ_2.

As mentioned for the European basket, the bivariate COS method requires the numerical calculation of the coefficients of the bivariate cosine expansion. This causes an additional numerical overhead and yields longer runtimes. Further, we note that the proposed smoothing is only applicable if the underlying is conditional normally distributed whereas the COS method is only relying on the availability of the characteristic function.

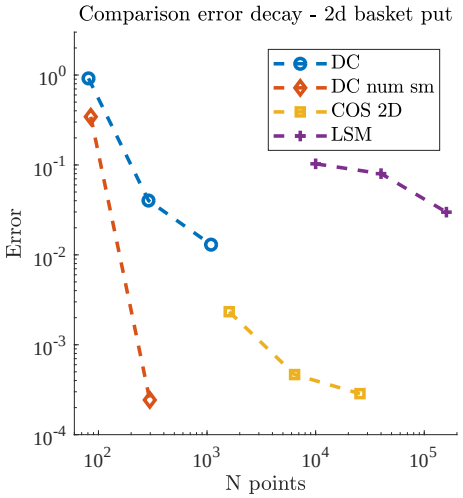

Figure 5.24: Error decay of the bivariate dynamic Chebyshev (DC) with with $N_1 = N_2$ and with $N_1 = 4N_2$ and comparison to the bivariate cosine method (COS 2d) and to the least-squares Monte Carlo (LSM) approach. Error is measured in absolute terms.

5.5 Conclusion and outlook

In this chapter we investigated the pricing of multivariate options that possibly admit an early-exercise feature. We discussed the extension of the dynamic Chebyshev pricing method into a multivariate framework. In general this is difficult and a straightforward application is often not feasible. As a first special case we investigated models where the risk factors are independent. In this case, the computational effort of the pre-computation step is dramatically reduced and scales linearly in the number of dimensions. We showed how models with multivariate normally distributed increments can be transformed into this framework using a decomposition of the covariance matrix. The class of models with multivariate normally distributed increments includes for instance the multivariate Black-Scholes model and short rate models such as a two-factor Hull-White or Black-Karasinski model. A first toy example of a bivariate Bermudan option provided promising results in comparison to a least-squares Monte Carlo algo-

rithm. Furthermore, we have shown that the pricing can be accelerated if the explicit computation of coefficients is omitted. In this case, the dynamic Chebyshev algorithm operates solely on the grid values and each time-step becomes the application of (a) linear operator(s) mapping function values $\widehat{V}_{t_{u+1}}$ to updated values $\widehat{V}_{t_u}$. In one dimension, $\widehat{V}_{t_{u+1}}$ is a vector in $\mathbb{R}^n$ and each time step can be seen as a linear operator $\mathcal{L}_{dC} \in \mathbb{R}^{n\times n}$ with $\widehat{V}_{t_u} = \mathcal{L}_{dC}\widehat{V}_{t_{u+1}}$. In two dimensions, $\widehat{V}_{t_{u+1}}$ is a matrix in $\mathbb{R}^{n\times n}$ and if the risk factors are independent we can write $\widehat{V}_{t_u} = \mathcal{L}^1_{dC}\widehat{V}_{t_{u+1}}\mathcal{L}^2_{dC}$ for two operators $\mathcal{L}^1_{dC}, \mathcal{L}^2_{dC} \in \mathbb{R}^{n\times n}$. The efficiency of the multivariate dynamic Chebyshev algorithm depends critically on the size of the grid of nodal points n^d.

We then considered an application of the bivariate dynamic Chebyshev method: The calibration of a two-factor interest rate/credit model and the pricing of callable bonds. Callable bonds belong to the most relevant fixed income instruments and in 2019, over 70% of the 1.4 trillion USD corporate bond issuance were callable. We showed that our new pricing method is suitable for an efficient calibration of the model to CDS spreads. Once the two-factor model is calibrated the method delivers accurate prices and provides a fast error decay for callable bonds. The empirical order of convergence was better than quadratic which implies that the new approach outperforms the popular and common finite-differences PDE solvers. Furthermore, we empirically verified the stability of the calibration routine with respect to parallel and individual shifts of the interest rate and credit curve. We point out that the calibration of the two-factor model is of relevance on its own since different other credit derivatives could possibly be priced in the same model. Especially, the efficient pricing algorithm for CDS instruments might be interesting for different applications in fixed income markets.

In the remaining two sections of the chapter we took a closer look at basket options in a multivariate Black-Scholes model. We started with the pricing of European basket options and we briefly discussed that the dynamic Chebyshev method can be used as a quadrature techniques for conditional expectations. Then we showed that the calculation of basket option prices can be simplified by using the smoothing the payoff approach proposed in [11]. In a numerical convergence analysis we showed that both, Gauss-Hermite and the dynamic Chebyshev quadrature are able to produce very accurate option prices using only a few points per dimension. The experiments indicate that in lower dimensions up to roughly $d = 5$, the additional effort of an adaptive sparse grid quadrature, as suggested in [11], can be avoided. Due to the smoothness of the integrand, a full-tensor is still of a workable size and procudes very accurate results while being as fast or faster than a Monte Carlo simulation.

In the case of $d = 2$, we compared the new approach to a Fourier type benchmark method, the bivariate COS method of [109]. Due to the increased smoothness and the

dimension reduction to a $d-1$-dimensional integration problem, the new quadrature approaches were able to produce the same accuracy with a significantly lower number of nodal values. Similar comparisons can be done with PDE type benchmark methods, for example methods using radial basis functions (RBF). For $d = 2$ (and $d = 3$) the RBF partition of unity method of [116] and the RBF generated finite differences of [97] are suitable benchmark approaches. Alternative approaches, that can also be extended to higher dimensions, are the hierarchical approximation method of [108] and the PDE method of [121]. One major drawback of PDE methods for European basket options is that they require a time-discretization additionally to the discretization in space. For example, [116] state that their RBF method needs $42 \times 42 = 1764$ points in space and also 10 time steps to reach an accuracy of 10^{-4} for a European basket call option. In our experiments, the dynamic Chebyshev quadrature was usually able to reach the same accuracy with 33 total points and no time steps.

The proposed quadrature approaches for European basket options can then be extended to early exercise options. For the pricing of Bermudan basket options, we generalized the smoothing the payoff idea and added a new numerical smoothing to the dynamic Chebyshev method. Our numerical investigation provides promising results for multivariate early-exercise options on up to 4 assets. The experiments confirm a fast convergence of the resulting pricing method and indicate an efficiency gain in comparison to the least-squares Monte Carlo method. The dynamic Chebyshev method was able to achieve a higher accuracy while also being faster. Moreover, for $d = 2$ we compared our method again to the bivariate COS method of [109] and observed an efficiency gain in our experiment. In the PDE world the above mentioned methods of [116], [97] and [121] are again interesting benchmarks for our method. Considering again the numerical examples in [116], they need also around 46×46 spatial points for an American bivariate basket option to achieve an accuracy of 10^{-4}. However, for stability reasons they require 10,000 time steps and this leads to realtively long runtimes. The dynamic Chebyshev method needs a comparable number of points in space but comes out with only 50 time steps per year. For higher dimensions, [73] propose a new pricing method for Bermudan baskets based on the approach of [108]. Their ansatz uses a principal component analysis and then solves several low-dimensional PDEs instead of one high-dimensional one. [73] report a second-order convergence behaviour of the method but they state that the "convergence behaviour can be somewhat irregular". This approach could still be an interesting benchmark for the dynamic Chebyshev method in four dimensions. It would be interesting to perform a more comprehensive comparison of the different methods in terms of convergence behaviour as well as runtimes. Moreover, improvements of the least-squares Monte Carlo method are suitable candidate for a numerical comparison in higher dimensions. For example, [82] present a pricing method for high-dimensional

American options using machine learning as alternative to the classical least-squares Monte Carlo approach. We expect that for lower dimensions, the dynamic Chebyshev method is favourable but if d becomes too high, regression or machine learning based approaches should be more suitable.

In summary, the multivariate dynamic Chebyshev method is a promising candidate for the pricing of Bermudan basket options in the multivariate Black-Scholes model. In our numerical experiments the method was able to outperform standard benchmark methods such as a least-squares Monte Carlo method by providing more accurate results while also delivering faster runtimes. The numerical smoothing leads to a fast error decay and a comparably low number of nodal points for option pricing problems in up to 4 dimensions. Besides its efficiency, the new method has also the advantage of being simple and easy to implement.

We believe that the presented numerical analysis is a good starting point for further theoretical and empirical investigations. So far, we focused mainly on an empirical convergence analysis and a comparison to different benchmark methods. From a theoretical point of view it would be interesting to extend the error analysis conducted in Chapter 3 to the dynamic Chebyshev algorithm with numerical smoothing. In order to achieve high efficiency, it is important to investigate the optimal relationship between N_1 and N_2 as well as the optimal domain size in more detail. The fixed ratio of $N_1 = 4N_2$ turned out to be suitable for our experiments but might not be the optimal choice. A better theoretical understanding could therefore improve the proposed method further.

Besides the theoretical error analysis, it would be interesting to empirically investigate if the numerical smoothing can be extended to other models or other payoff profiles. The smoothing the payoff of [11] is specifically designed for basket options in the multivariate Black-Scholes model and has limited flexibility. In contrast, we expect that the numerical smoothing offers a higher flexibility and is applicable in a wider range of scenarios. Here, the pricing of a callable bond in a two-factor Hull–White/Black–Karasinski model would be an interesting example. Another example would be the replacement of the basket option payoff by a call or put option on the maximum of d assets.

By design, the new method is a valid candidate for option pricing problems in models with up to five risk factors that are jointly multivariate normally distributed. The method allows us to consider correlation between the risk factors as well as products with early-exercise features. For this framework, different further extensions and modifications of the method might be interesting to investigate. While this thesis focussed mostly on Chebyshev interpolation, other function approximation methods can also be used in a similar way. Here, it would be interesting to compare different approximation techniques

in terms of their convergence behaviour as well as the resulting runtimes. Possible examples are kernel based approximation techniques or radial basis function interpolation that are known to provide good results in multivariate settings. See for example the above mentioned option pricing methods of [116] and [97]. Often these function approximation methods are able to achieve accurate results using only a relatively low number of basis functions. In contrast to the tensor-based Chebyshev interpolation these methods are not necessarily limited to a rectangular interpolation domain. They come however with a higher computational complexity and an efficient implementation can be significantly more challenging. As briefly discussed in [11] for adaptive sparse grids, the choice of a programming language for the implementation can have a significant influence on the comparison. For instance, a method that performs relatively poorly in Matlab might still be highly efficient when implemented in C or C++.

Moreover, we have seen that low-rank tensors in the tensor train format are able to reduce the storage requirements of the full tensor. We did not pursue this approach due to the sub-optimal rank structure that would make a rank adaptive completion algorithm infeasible. It would however be interesting to investigate if other forms of low-rank approximations are more suitable for this particular problem. For example, in the dynamic Chebyshev algorithm it could be beneficial to start with a full tensor in the first time step and then continue with tensors in a (different) low-rank format. For the one-dimensional dynamic Chebyshev method the splitting at the strike K can increase the efficiency of the approach by replacing one interpolation of a high polynomial degree with two interpolations of a lower degree. Similar ideas based on a domain splitting could also be applied in the multivariate dynamic Chebyshev algorithm. For instance, a straightforward approach for the bivariate algorithm with numerical smoothing would be to divide the interpolation domain $[\underline{x}_1, \overline{x}_1] \times [\underline{x}_2, \overline{x}_2]$ into two domains $[\underline{x}_1, k] \times [\underline{x}_2, \overline{x}_2]$ and $[k, \overline{x}_1] \times [\underline{x}_2, \overline{x}_2]$. Overall, we think that the presented dynamic Chebyshev method is an interesting alternative to standard multivariate pricing methods and the method is a promising starting point for further research.

Appendix A

Chebyshev algorithm for the implied volatility

In this appendix, we present additional material on the derivation of the Chebyshev method for the implied volatility as presented in Section 2.4.

Scaling functions

In the following, we derive the scaling functions for each of the three areas. The idea of the scaling is to define functions $\phi_{i,x}$ such that $v(c,x) = \tilde{v}(\tilde{c},x)$ with $\tilde{c} = \phi_{i,x}(c) \in [-1,1]$ and $[-1,1] \ni \tilde{c} \mapsto \tilde{v}(\tilde{c},x)$ is approximately linear. The resulting Chebyshev interpolation of $\tilde{v}$ in $\tilde{c}$ will then be significantly more efficient then a direct interpolation of v in c. A more detailed derivation of the different scaling functions is provided in [68].

Medium volatilities:
For v around the point of inflection, the implied volatility surface is almost linear, see Figure 2.6. Thus, a linear scaling suffices,

$$\phi_{2,x} : [c_1(x), c_2(x)] \to [-1,1], \quad c \mapsto 2\frac{c - c_1(x)}{c_2(x) - c_1(x)} - 1.$$

Clearly, ϕ_2 is analytic and the inverse is given by

$$\phi_{2,x}^{-1}[-1,1] \to [c_1(x), c_2(x)], \quad \tilde{c} \mapsto c_1(x) + \frac{1}{2}(\tilde{c}+1)(c_2(x) - c_1(x)).$$

Low volatilities:
For low volatilities the call price function is very flat, and thus the implied volatility

function as its inverse is steep. In order to find a suitable scaling function, we explore the limit behaviour of the normalized call price. For $v \to 0$ we have by equation (2.8) of [74] that

$$c(x,v) \approx \varphi\Big(\frac{x}{v}\Big)\Big(\frac{v^3}{x^2}\Big),$$

where φ is the density of the standard normal distribution. Inverting the function $c(v) = \varphi\left(\frac{x}{v}\right)$, which has the major effect in the limit, leads to the following transformation

$$\tilde{\phi}_{1,x} : [0, c_1(x)] \to [-1,1]$$
$$c \mapsto \begin{cases} 2\left(-\frac{2}{(x-\delta)^2}\log(c) + \frac{2}{(x-\delta)^2}\log(c_1(x)) + 1\right)^{-\frac{1}{2}} - 1 & \text{if } c > 0 \\ -1 & \text{else.} \end{cases}$$

The parameter $\delta > 0$ ensures the well-definedness for $x = 0$ and the remaining terms are needed to map the interval $[0, c_1(x)]$ to $[-1, 1]$. The transformation $\tilde{\phi}_{1,x}$ is analytic with inverse

$$\tilde{\phi}^{-1}_{1,x} : [-1,1] \to [0, c_1(x)] : \tilde{c} \mapsto \begin{cases} c_1(x) e^{-\frac{2(x-\delta)^2}{(\tilde{c}+1)^2} + \frac{(x-\delta)^2}{2}} & \text{if } \tilde{c} > -1 \\ 0 & \text{else.} \end{cases}$$

Using this transformation the function $v(\tilde{\phi}^{-1}_{1,x}(\tilde{c}), x)$ is approximately linear in $\tilde{c}$. As already mentioned earlier, to guarantee analyticity we restrict the interval $[0, c_1(x)]$ to $[c_{min}(x), c_1(x)]$ for $0 < c_{min}(x) < c_1(x) < e^{\frac{x}{2}}$. Therefore, we define the scaling function for the low volatilities $\phi_{1,x} : [c_{min}(x), c_1(x)] \to [-1,1]$ as $\phi_{1,x}(c) := l(\tilde{\phi}_{1,x}(c))$, where l is the linear transformation that maps $[\tilde{\phi}_{1,x}(c_{min}(x)), 1]$ to $[-1,1]$. The function $c \mapsto \phi_{1,x}(c)$ is analytic on $[c_{min}(x), c_1(x)]$ and its inverse is given by $\phi^{-1}_{1,x}(\tilde{c}) = \tilde{\phi}^{-1}_{1,x}\left(l^{-1}(\tilde{c})\right)$.

High volatilities:
Just as for the low volatilities, the call price function is very flat for high volatilities and thus its inverse becomes steep. As $\lim_{c \to e^{\frac{x}{2}}} v(c,x) = \infty$ the implied volatility function is not bounded and we cap it at some v_{max}. From [74] equation (2.7) we obtain for $v \to \infty$

$$c(x,v) \approx e^{\frac{x}{2}} - \frac{4}{v}\varphi\Big(\frac{v}{2}\Big).$$

A similar transformation as in the case of low volatilities entails improvement. Assume

first that $c_{max}(x) = e^{\frac{x}{2}}$ and define

$$\tilde{\phi}_{3,x} : [c_2(x), e^{\frac{x}{2}}] \to [0, \infty] : c \mapsto \begin{cases} \left(-8 \log \left(\frac{e^{\frac{x}{2}} - c}{e^{\frac{x}{2}} - c_2(x)} \right) \right)^{\frac{1}{2}} & \text{if } c < e^{\frac{x}{2}} \\ \infty & \text{else} \end{cases}$$

with inverse

$$\tilde{\phi}_{3,x}^{-1} : [0, \infty] \to [c_2(x), e^{\frac{x}{2}}] : \tilde{c} \mapsto \begin{cases} e^{\frac{x}{2}} - \left(e^{x/2} - c_2(x) \right) e^{-\frac{\tilde{c}^2}{8}} & \text{if } \tilde{c} < \infty \\ e^{\frac{x}{2}} & \text{else.} \end{cases}$$

Exploiting the limit behaviour of the call price, one can show that for v large enough $\tilde{c} = \tilde{\phi}_{3,x}(c(x, v)) \approx -v$. Hence $v = v(\tilde{\phi}_{3,x}^{-1}(\tilde{c}), x) \approx -\tilde{c}$ which is linear in $\tilde{c}$. For $c_{max}(x) < e^{\frac{x}{2}}$ the transformation is a bijection into a bounded domain which can be normalized to $[-1, 1]$ by the linear transform l that maps $[0, \tilde{\phi}_{3,x}(c_{max}(x))]$ to $[-1, 1]$. Thus $\phi_{3,x}(c) := l(\tilde{\phi}_{3,x}(x))$ and $\phi_{3,x}^{-1}(\tilde{c}) = \tilde{\phi}_{3,x}^{-1}\left(l^{-1}\left(\tilde{c}\right)\right)$ depending on the choice of c_{max}.

Splitting

As a next steps we define the splitting call prices $c_{min}(x), c_1(x), c_2(x)$ and $c_{max}(x)$ and thus the three areas D_1, D_2 and D_3 explicitly. We do this by defining the corresponding volatilities $0 < v_{min}(x) < v_1(x) < v_2(x) < v_{max}(x)$. We want to set the boundaries in such a way that a very large set of parameters is covered and the rate of convergence is about the same for all areas.

Maximal and minimal volatility:
We choose $v_{max} = 6$ as an upper bound for the time scaled volatility. This allows us to include highly volatile markets and long maturities. As a a lower bound we fix $v_{min}(x) = 0.001 - 0.03x$. This choice includes very low volatilities and the corresponding prices $c_{min}(x)$ can still be computed with the machine precision. For instance at $x = \log(e^{rT} S_0/K) = 0$ this choice allows call options with a time to maturity of one day ($T = 1/365$) and an annual Black-Scholes volatility of $\sigma \approx 2\%$. The rate of convergence can be increased further if v_{min} is chosen higher.

Splitting volatilities v_1 and v_2:
We choose v_1 and v_2 according to the properties of the call price function. The call price function has a unique inflection point for $v_c(x) = \sqrt{2|x|}$ where the slope is maximal. [75] proposes the lower bound v_1 as the zeros of the tangent line at this point. The upper bound v_2 is set to be the point where the line hits the maximal call price depending on x. See Figure A.1 for the tangent line and the suggested splitting points $\tilde{v}_1$ and $\tilde{v}_2$.

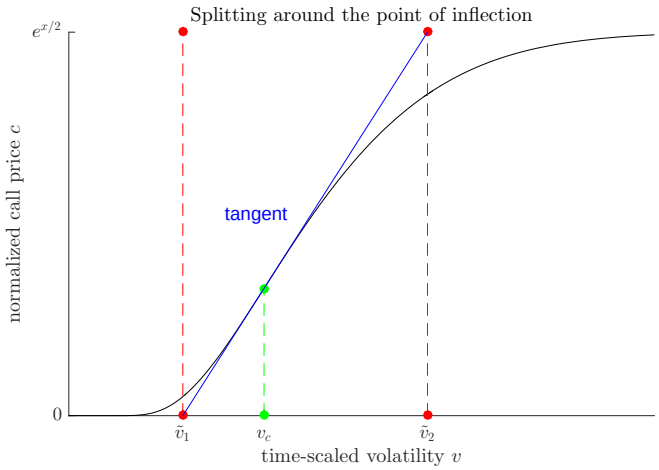

Figure A.1: Definition of the splitting at $\tilde{v}_1$ and $\tilde{v}_2$ by the zeros of the tangent line at the point of inflection v_c.

However, this choice of boundaries has two serious disadvantages. First, the boundary $\tilde{v}_1$ tends to zero, hence for small values of x we obtain $\tilde{v}_1(x) < v_{min}(x)$. Second, the computation of $\tilde{v}_1(x)$ and $\tilde{v}_2(x)$ requires the evaluation of $c(x, v_c)$ and $\frac{\partial}{\partial v}c(x, v_c)$ for each x. For real-time computation on large data sets, this becomes a computational burden. We solve this problem by replacing $\tilde{v}_1$ and $\tilde{v}_2$ with linear approximations. We propose the boundaries $v_1(x) = 0.25 - 0.4x$ and $v_2(x) = 2 - 0.4x$.

Splitting of the low volatility area:
For low volatilities we improve the interpolation by introducing a further splitting in x. Following [68], we divide the area of the low volatilities in an Area I for $x \in [-5, -0.0348]$ and an Area I' for $x \in [-0.0348, 0]$, see Figure 2.7. The empirical results show that this additional splitting further improves the rate of convergence.

Algorithm as pseudocode

Here, we present the algorithms of the Chebyshev method for the implied volatility in form of pseudocodes. Algorithm 8 defines the transformations and scaling functions. Algorithm 9 and Algorithm 10 describe two versions of the offline phase of our approach. Algorithm 9 uses Jäckel's method to compute the implied volatilities at the transformed Chebyshev grid points. Algorithm 10 shows the offline phase using a general root finder to compute the implied volatilities at the transformed grid points. Algorithm 11 presents the Chebyshev implied volatility function using *chebfun2* objects in the four interpolation areas.

Algorithm 8 Initilization

Define normalized call price:
$c(x,v) \leftarrow e^{\frac{x}{2}}\Phi\left(\frac{x}{v}+\frac{v}{2}\right)-e^{-\frac{x}{2}}\Phi\left(\frac{x}{v}-\frac{v}{2}\right)$
Define splitting call prices:
$c_{min}(x) \leftarrow c(x, 0.001-0.03x)$, $c_1(x) \leftarrow c(x, 0.25-0.4x)$, $c_2(x) \leftarrow c(x, 2-0.4x)$, $c_{max}(x) \leftarrow c(x,6)$
Define scaling functions:
$\varphi(x,\underline{x},\overline{x}) \leftarrow 1-2\cdot\frac{\overline{x}-x}{\overline{x}-\underline{x}}$
$\tilde{\phi}_1(x,c) \leftarrow 2\left(-\frac{2}{(x-\delta)^2}\log(c)+\frac{2}{(x-\delta)^2}\log(c_1(x))+1\right)^{-\frac{1}{2}}-1$
$\phi_1(x,c) \leftarrow 2\frac{\tilde{\phi}_1(x,c)-\tilde{\phi}_1(x,c_{min}(x))}{1-\tilde{\phi}_1(x,c_{min}(x))}-1$
$\phi_2(x,c) \leftarrow 2\frac{c-c_1(x)}{c_2(x)-c_1(x)}-1$
$\tilde{\phi}_3(x,c) \leftarrow \left(-8\log\left(\frac{e^{\frac{x}{2}}-c}{e^{\frac{x}{2}}-c_2(x)}\right)\right)^{\frac{1}{2}}$
$\phi_3(x,c) \leftarrow \frac{2\tilde{\phi}_3(x,c)}{\tilde{\phi}_3(x,c_{max}(x))}-1$
Define inverse scaling functions:
$\varphi^{-1}(y,\underline{x},\overline{x}) \leftarrow \overline{x}-0.5(\overline{x}-\underline{x})(1-y)$
$\tilde{\phi}_1^{-1}(x,\tilde{c}) \leftarrow c_1(x)e^{-\frac{2(x-\delta)^2}{(\tilde{c}+1)^2}+\frac{(x-\delta)^2}{2}}$
$\phi_1^{-1}(x,\tilde{c}) \leftarrow \tilde{\phi}_1^{-1}(x,\tilde{\phi}_1(x,c_{min}(x))+0.5(1-\tilde{\phi}_1(x,c_{min}(x)))(1+\tilde{c}))$
$\phi_2^{-1}(x,\tilde{c}) \leftarrow c_1(x)+\frac{1}{2}(\tilde{c}+1)(c_2(x)-c_1(x))$
$\tilde{\phi}_3^{-1}(x,\tilde{c}) \leftarrow e^{\frac{x}{2}}-\left(e^{x/2}-c_2(x)\right)e^{-\frac{\tilde{c}^2}{8}}$
$\phi_3^{-1}(x,\tilde{c}) \leftarrow \tilde{\phi}_3^{-1}(x, 0.5\tilde{\phi}_3(x,c_{max}(x))(1+\tilde{c}))$

Algorithm 9 Pre-computation step (Using Jäckel's method)

Require: $\widehat{v}_{\mathrm{JL}}(x,c)$ (Jäckel's implied vola function), load *chebfun* package, ε error tolerance
Area I
$\tilde{v}_1(\tilde{x},\tilde{c}) \leftarrow \widehat{v}_{\mathrm{JL}}(\varphi^{-1}(\tilde{x},-5,-0.0348),\phi_1^{-1}(\tilde{x},\tilde{c}))$
$I_1(\tilde{x},\tilde{c}) \leftarrow$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_1(\tilde{x},\tilde{c}),\ 'tol'=\varepsilon)$
Area I'
$\tilde{v}_{1+}(\tilde{x},\tilde{c}) \leftarrow \widehat{v}_{\mathrm{JL}}(\varphi^{-1}(\tilde{x},-0.0348,0),\phi_1^{-1}(\tilde{x},\tilde{c}))$
$I_{1+}(\tilde{x},\tilde{c}) \leftarrow$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_{1+}(\tilde{x},\tilde{c}),\ 'tol'=\varepsilon)$
Area II
$\tilde{v}_2(\tilde{x},\tilde{c}) \leftarrow \widehat{v}_{\mathrm{JL}}(\varphi^{-1}(\tilde{x},-5,0),\phi_2^{-1}(\tilde{x},\tilde{c}))$
$I_2(\tilde{x},\tilde{c}) \leftarrow$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_2(\tilde{x},\tilde{c}),\ 'tol'=\varepsilon)$
Area III
$\tilde{v}_3(\tilde{x},\tilde{c}) \leftarrow \widehat{v}_{\mathrm{JL}}(\varphi^{-1}(\tilde{x},-5,0),\phi_3^{-1}(\tilde{x},\tilde{c}))$
$I_3(\tilde{x},\tilde{c}) \leftarrow$ chebfun2$((\tilde{x},\tilde{c})\mapsto\tilde{v}_3(\tilde{x},\tilde{c}),\ 'tol'=\varepsilon)$
Output: Four *chebfun2* objects $I_1(\tilde{x},\tilde{c})$, $I_{1+}(\tilde{x},\tilde{c})$, $I_2(\tilde{x},\tilde{c})$, $I_3(\tilde{x},\tilde{c})$

Algorithm 10 Pre-computation step (Using a rootfinder)

Require: $root(f, tol)$ (rootfinder which takes a function f and error tolerance tol), load *chebfun* package, ε error tolerance

Area I
$\tilde{v}_1(\tilde{x}, \tilde{c}) \leftarrow root(v \mapsto \phi_1^{-1}(\tilde{x}, \tilde{c}) - c(\varphi^{-1}(\tilde{x}, -5, -0.0348), v), \varepsilon)$
$I_1(\tilde{x}, \tilde{c}) \leftarrow$ chebfun2$((\tilde{x}, \tilde{c}) \mapsto \tilde{v}_1(\tilde{x}, \tilde{c}),\ 'tol' = \varepsilon)$
Area I'
$\tilde{v}_{1+}(\tilde{x}, \tilde{c}) \leftarrow root(v \mapsto \phi_1^{-1}(\tilde{x}, \tilde{c}) - c(\varphi^{-1}(\tilde{x}, -0.0348, 0), v), \varepsilon)$
$I_{1+}(\tilde{x}, \tilde{c}) \leftarrow$ chebfun2$((\tilde{x}, \tilde{c}) \mapsto \tilde{v}_{1+}(\tilde{x}, \tilde{c}),\ 'tol' = \varepsilon)$
Area II
$\tilde{v}_2(\tilde{x}, \tilde{c}) \leftarrow root(v \mapsto \phi_2^{-1}(\tilde{x}, \tilde{c}) - c(\varphi^{-1}(\tilde{x}, -5, 0), v), \varepsilon)$
$I_2(\tilde{x}, \tilde{c}) \leftarrow$ chebfun2$((\tilde{x}, \tilde{c}) \mapsto \tilde{v}_2(\tilde{x}, \tilde{c}),\ 'tol' = \varepsilon)$
Area III
$\tilde{v}_3(\tilde{x}, \tilde{c}) \leftarrow root(v \mapsto \phi_3^{-1}(\tilde{x}, \tilde{c}) - c(\varphi^{-1}(\tilde{x}, -5, 0), v), \varepsilon)$
$I_3(\tilde{x}, \tilde{c}) \leftarrow$ chebfun2$((\tilde{x}, \tilde{c}) \mapsto \tilde{v}_3(\tilde{x}, \tilde{c}),\ 'tol' = \varepsilon)$
Output: Four *chebfun2* objects $I_1(\tilde{x}, \tilde{c})$, $I_{1+}(\tilde{x}, \tilde{c})$, $I_2(\tilde{x}, \tilde{c})$, $I_3(\tilde{x}, \tilde{c})$

Algorithm 11 The Chebyshev implied volatility function

Require: Four *chebfun2* objects $I_1(\tilde{x}, \tilde{c})$, $I_{1+}(\tilde{x}, \tilde{c})$, $I_2(\tilde{x}, \tilde{c})$, $I_3(\tilde{x}, \tilde{c})$
Input call price C, strike K, spot S_0, maturity T and rate r
Normalization
$x \leftarrow rT + \log(S_0/K), \quad c \leftarrow C/\sqrt{S_0 e^{-rT} K}$
if $x < 0$ **then** $c \leftarrow c + e^{-x/2} - e^{x/2}, \quad x \leftarrow -x$
Splitting
if $c \le c_{max}(x)$ and $x \le 5$ **then**
    if $c < c_2(x)$ **then**
        if $c < c_1(x)$ and $c \ge c_{min}(x)$ **then**
            if $x < -0.0348$ **then** $Area \leftarrow$ Area I'
            else $Area \leftarrow$ Area I
        else $Area \leftarrow$ Area II
    else $Area \leftarrow$ Area III
else Return c or x is too high
Transformation and Evaluation
Switch cases '$Area$'
Case Area I: $\tilde{x} \leftarrow \varphi(x, -5, -0.0348), \quad \tilde{c} \leftarrow \phi_1(x, c)$ and $v \leftarrow I_1(\tilde{x}, \tilde{c})$
Case Area I': $\tilde{x} \leftarrow \varphi(x, -0.0348, 0), \quad \tilde{c} \leftarrow \phi_1(x, c)$ and $v \leftarrow I_{1+}(\tilde{x}, \tilde{c})$
Case Area II: $\tilde{x} \leftarrow \varphi(x, -5, 0), \quad \tilde{c} \leftarrow \phi_2(x, c)$ and $v \leftarrow I_2(\tilde{x}, \tilde{c})$
Case Area III: $\tilde{x} \leftarrow \varphi(x, -5, 0), \quad \tilde{c} \leftarrow \phi_1(x, c)$ and $v \leftarrow I_3(\tilde{x}, \tilde{c})$
Return: v

Appendix B

Multivariate generalized moments

In this appendix, we extend the one-dimensional recursive formula for the generalized moments into a multivariate setting. We want to compute the multivariate generalized moments

$$\Gamma^d_{\boldsymbol{n}}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \mathbb{E}[T_{\boldsymbol{n}}(\boldsymbol{X}) 1_{\mathcal{I}^d}(\boldsymbol{X})] \qquad \text{with} \qquad \boldsymbol{X} \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

for mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$. Here $\boldsymbol{n}$ is a multi-index with $\boldsymbol{n} \in \mathbb{N}^d$ and $T_{\boldsymbol{n}}$ the multivariate Chebyshev polynomial $T_{\boldsymbol{n}}(\boldsymbol{x}) = \prod_{i=1}^d T_{n_i}(x_i)$. The following theorem provides a recursive formula for the multivariate generalized moments and is based on [79].

Theorem 16. *The generalized moments for a multivariate normally distributed variable $\boldsymbol{X} \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be calculated with the following recursive formula for $n_i \geq 1$*

$$\Gamma^d_{\boldsymbol{n}+e_i}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = 2\big(\boldsymbol{\mu}_i \Gamma^d_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}) + \boldsymbol{\Sigma}_{i,\cdot} \boldsymbol{c}_n\big) - \Gamma^d_{\boldsymbol{n}-e_i}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

with vector $c_{\boldsymbol{n}} \in \mathbb{R}^d$ with entry j given by

$$c_{\boldsymbol{n},j} = (-1)^{n_j} \Gamma^{d-1}_{n(j)}(\tilde{\boldsymbol{\mu}}_j^{-1}, \tilde{\boldsymbol{\Sigma}}_j)\phi_{j,-1} - \Gamma^{d-1}_{n(j)}(\tilde{\boldsymbol{\mu}}_j^{1}, \tilde{\boldsymbol{\Sigma}}_j)\phi_{j,1} + \Gamma^{d,\prime}_{\boldsymbol{n},j}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where $\Gamma^{d,\prime}_{\boldsymbol{n},j}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathbb{E}[\frac{\partial T_{\boldsymbol{n}}(\boldsymbol{X})}{\partial x_j} \mathbb{1}_{\mathcal{I}^d}(\boldsymbol{X})]$ and $\phi_{j,-1} = \phi^1(-1, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j^2)$, $\phi_{j,1} = \phi^1(1, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j^2)$.

Here, ϕ^1 is the density of an univariate normally distributed variable. The formulas for the generalized moments $\Gamma^d_{\boldsymbol{n}+e_i}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for $\|\boldsymbol{n}\|_\infty \leq 1$ can be directly obtained form [79].

Proof. The generalized moment $\Gamma^d_{\boldsymbol{n}+e_i}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is given by

$$\begin{aligned}\Gamma^d_{\boldsymbol{n}+e_i}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \mathbb{E}[T_{\boldsymbol{n}+e_i}(\boldsymbol{X})1_{\mathcal{I}^d}(\boldsymbol{X})] \\ &= 2\mathbb{E}[X_i T_{\boldsymbol{n}}(\boldsymbol{X})1_{\mathcal{I}^d}(\boldsymbol{X})] - \mathbb{E}[T_{\boldsymbol{n}-e_i}(\boldsymbol{X})1_{\mathcal{I}^d}(\boldsymbol{X})], \\ &= 2\mathbb{E}[X_i T_{\boldsymbol{n}}(\boldsymbol{X})1_{\mathcal{I}^d}(\boldsymbol{X})] - \Gamma^d_{\boldsymbol{n}-e_i}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\end{aligned}$$

where we used the recursive structure of the Chebyshev polynomials. This means we need to find an expression for $\mathbb{E}[X_i T_{\boldsymbol{n}}(\boldsymbol{X})1_{\mathcal{I}^d}(\boldsymbol{X})]$. For the density of the multivariate normal distribution ϕ^d holds

$$-\frac{\partial \phi^d(\mathbf{x})}{\partial \mathbf{x}} = \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\phi^d(\mathbf{x}) \in \mathbb{R}^d,$$

where $\frac{\partial \phi^d(\mathbf{x})}{\partial \mathbf{x}}$ is the vector of partial derivatives, see [79]. The j-th component the equation reads as

$$-\frac{\partial \phi^d(\mathbf{x})}{\partial x_j} = \boldsymbol{\Sigma}^{-1}_{j,\cdot}(\mathbf{x} - \boldsymbol{\mu})\phi^d(\mathbf{x}).$$

We multiply each side with the Chebyshev polynomial $T_{\boldsymbol{n}}$ and integrate of $\mathcal{I}^d$. For the left hand side we obtain

$$\begin{aligned}\int_{\mathcal{I}^d} T_{\boldsymbol{n}}(\mathbf{x})\frac{\partial \phi^d(\boldsymbol{x})}{\partial x_j}\mathrm{d}\boldsymbol{x} &= \int_{\mathcal{I}^{d-1}} T_{\boldsymbol{n}(j)}(\boldsymbol{x}_{(j)}) \int_{\mathcal{I}} T_{n_j}(x_j)\frac{\partial \phi^d(\mathbf{x})}{\partial x_j}\mathrm{d}x_j\mathrm{d}\boldsymbol{x}_{(j)} \\ &= \int_{\mathcal{I}^{d-1}} T_{\boldsymbol{n}(j)}(\mathbf{x}_{(j)})\Big[T_{n_j}(x_j)\phi^d(\boldsymbol{x})|^1_{x_j=-1} \\ &\quad - \int_{\mathcal{I}} \frac{\partial T_{n_j}(x_j)}{\partial x_j}\phi^d(\mathbf{x})\mathrm{d}x_j\Big]\mathrm{d}\boldsymbol{x}_{(j)}\end{aligned}$$

using integration by paths. We define

$$\Gamma^{d,\prime}_{\boldsymbol{n},j}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \int_{\mathcal{I}^d} \frac{\partial T_{\boldsymbol{n}}(\boldsymbol{x})}{\partial x_j}\phi^d(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \int_{\mathcal{I}^{d-1}} T_{n(j)}(\mathbf{x}_{(j)}) \int_{\mathcal{I}} \frac{\partial T_{n_j}(x_j)}{\partial x_j}\phi^d(\mathbf{x})\mathrm{d}x_j\mathrm{d}\boldsymbol{x}_{(j)}$$

where $\boldsymbol{x}_{(j)}$ is the $d-1$ dimensional vector of $\boldsymbol{x}$ without the j-th entry. From [79] we obtain the following formula for the density of the multivariate normal distribution

$$\begin{aligned}\phi^d(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})|_{x_j=-1} &= \phi^1(-1, \mu_j, \sigma_j^2)\phi^{d-1}(\mathbf{x}_{(j)}, \tilde{\boldsymbol{\mu}}_j^{-1}, \tilde{\boldsymbol{\Sigma}}_j) \\ \phi^d(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})|_{x_j=1} &= \phi^1(1, \mu_j, \sigma_j^2)\phi^{d-1}(\mathbf{x}_{(j)}, \tilde{\boldsymbol{\mu}}_j^{1}, \tilde{\boldsymbol{\Sigma}}_j)\end{aligned}$$

with modified parameters

$$\tilde{\boldsymbol{\mu}}_j^{-1} = \boldsymbol{\mu}_{(j)} - \boldsymbol{\Sigma}_{(j),j}\frac{1+\mu_j}{\sigma_j^2}, \quad \tilde{\boldsymbol{\mu}}_j^{1} = \boldsymbol{\mu}_{(j)} + \boldsymbol{\Sigma}_{(j),j}\frac{1-\mu_j}{\sigma_j^2}, \quad \tilde{\boldsymbol{\Sigma}}_j = \boldsymbol{\Sigma}_{(j),(j)} - \frac{1}{\sigma_j^2}\boldsymbol{\Sigma}_{(j),j}\boldsymbol{\Sigma}_{j,(j)}.$$

The equations for the density yield

$$\begin{aligned}
&\int_{\mathcal{I}^{d-1}} T_{\boldsymbol{n}(j)}(\mathbf{x}_{(j)})\Big[T_{n_j}(x_j)\phi^d(\mathbf{x})|_{x_j=-1}^{1}\Big]\mathrm{d}\mathbf{x}_{(j)} \\
&= \int_{\mathcal{I}^{d-1}} T_{\boldsymbol{n}(j)}(\mathbf{x}_{(j)})T_{n_j}(1)\phi^1(1,\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j^2)\phi^{d-1}(\mathbf{x}_{(j)},\tilde{\boldsymbol{\mu}}_j^1,\tilde{\boldsymbol{\Sigma}}_j)\mathrm{d}\mathbf{x}_{(j)} \\
&- \int_{\mathcal{I}^{d-1}} T_{\boldsymbol{n}(j)}(\mathbf{x}_{(j)})T_{n_j}(-1)\phi^1(-1,\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j^2)\phi^{d-1}(\mathbf{x}_{(j)},\tilde{\boldsymbol{\mu}}_j^{-1},\tilde{\boldsymbol{\Sigma}}_j)\mathrm{d}\mathbf{x}_{(j)} \\
&= \Gamma^{d-1}_{\boldsymbol{n}(j)}(\tilde{\boldsymbol{\mu}}_j^1,\tilde{\boldsymbol{\Sigma}}_j)\phi^1(1,\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j^2) - T_{n_j}(-1)\Gamma^{d-1}_{\boldsymbol{n}(j)}(\tilde{\boldsymbol{\mu}}_j^{-1},\tilde{\boldsymbol{\Sigma}}_j)\phi^1(-1,\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j^2)
\end{aligned}$$

Overall, we obtain for the left hand side

$$-\int_{\mathcal{I}^d} T_{\boldsymbol{n}}(\mathbf{x})\frac{\partial\phi^d(\mathbf{x})}{\partial x_j}\mathrm{d}\mathbf{x} = (-1)^{n_j}\Gamma^{d-1}_{\boldsymbol{n}(j)}(\tilde{\boldsymbol{\mu}}_j^{-1},\tilde{\boldsymbol{\Sigma}}_j)\phi_{j,-1} - \Gamma^{d-1}_{n(j)}(\tilde{\boldsymbol{\mu}}_j^1,\tilde{\boldsymbol{\Sigma}}_j)\phi_{j,1} + \Gamma^{d,\prime}_{\boldsymbol{n},j}(\boldsymbol{\mu},\boldsymbol{\Sigma})$$

with $\phi_{j,-1} = \phi^1(-1,\mu_j,\sigma_j^2)$, $\phi_{j,1} = \phi^1(1,\mu_j,\sigma_j^2)$. For the right hand side we obtain,

$$\begin{aligned}
\int_{\mathcal{I}^d} T_{\boldsymbol{n}}(\mathbf{x})\big(\boldsymbol{\Sigma}_{j,\cdot}^{-1}(\mathbf{x}-\boldsymbol{\mu})\phi^d(\mathbf{x})\big)\mathrm{d}\mathbf{x} &= \int_{\mathcal{I}^d} T_{\boldsymbol{n}}(\mathbf{x})\big(\sum_{i=1}^{d}\boldsymbol{\Sigma}_{j,i}^{-1}(x_i-\boldsymbol{\mu}_i)\phi^d(\mathbf{x})\big)\mathrm{d}\mathbf{x} \\
&= \sum_{i=1}^{d}\boldsymbol{\Sigma}_{j,i}^{-1}\int_{\mathcal{I}^d} T_{\boldsymbol{n}}(\mathbf{x})x_i\phi^d(\mathbf{x}) - T_{\boldsymbol{n}}(\mathbf{x})\boldsymbol{\mu}_i\phi^d(\mathbf{x})\mathrm{d}\mathbf{x} \\
&= \sum_{i=1}^{d}\boldsymbol{\Sigma}_{j,i}^{-1}\big(\mathbb{E}[T_{\boldsymbol{n}}(\boldsymbol{X})X_i\mathbb{1}_{\mathcal{I}^d}(\boldsymbol{X})] - \boldsymbol{\mu}_i\Gamma^d_{\boldsymbol{n}}(\boldsymbol{\mu},\boldsymbol{\Sigma})\big) \\
&= \boldsymbol{\Sigma}_{j,\cdot}^{-1}\big(\mathbb{E}[T_{\boldsymbol{n}}(\boldsymbol{X})X\mathbb{1}_{\mathcal{I}^d}(\boldsymbol{X})] - \boldsymbol{\mu}\Gamma^d_{\boldsymbol{n}}(\boldsymbol{\mu},\boldsymbol{\Sigma})\big),
\end{aligned}$$

where $\mathbb{E}[T_{\boldsymbol{n}}(X)\boldsymbol{X}\mathbb{1}_{\mathcal{I}^d}]$ is a column vector with i-th entry $\mathbb{E}[T_{\boldsymbol{n}}(\boldsymbol{X})X_i\mathbb{1}_{\mathcal{I}^d}(\boldsymbol{X})]$ and we recall that $\boldsymbol{\mu}$ is a column vector as well. Bringing the right hand side and the left hand side together yields

$$\begin{aligned}
-\frac{\partial\phi^d(\mathbf{x})}{\partial\mathbf{x}} &= \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\phi^d(\mathbf{x}) \\
\implies \qquad \mathbf{c}_{\boldsymbol{n}} &= \boldsymbol{\Sigma}^{-1}\big(\mathbb{E}[T_{\boldsymbol{n}}(\boldsymbol{X})\boldsymbol{X}\mathbb{1}_{\mathcal{I}^d}(\boldsymbol{X})] - \boldsymbol{\mu}\Gamma^d_{\boldsymbol{n}}(\boldsymbol{\mu},\boldsymbol{\Sigma})\big) \\
\implies \qquad \mathbb{E}[T_{\boldsymbol{n}}(\boldsymbol{X})\boldsymbol{X}\mathbb{1}_{\mathcal{I}^d}(\boldsymbol{X})] &= \boldsymbol{\mu}\Gamma^d_{\boldsymbol{n}}(\boldsymbol{\mu},\boldsymbol{\Sigma}) + \boldsymbol{\Sigma}\mathbf{c}_{\boldsymbol{n}}
\end{aligned}$$

and hence

$$\Gamma^d_{\boldsymbol{n}+e_i}(\boldsymbol{\mu},\boldsymbol{\Sigma}) = 2\big(\boldsymbol{\mu}_i\Gamma^d_{\boldsymbol{n}}(\boldsymbol{\mu},\boldsymbol{\Sigma}) + \boldsymbol{\Sigma}_{i,\cdot}\mathbf{c}_{\boldsymbol{n}}\big) - \Gamma^d_{\boldsymbol{n}-e_i}(\boldsymbol{\mu},\boldsymbol{\Sigma})$$

which was our claim.

□

Proposition 14. *The expectations of the derivatives of the Chebyshev polynomials are given by*

$$\Gamma^{d,\prime}_{\boldsymbol{n},j}(\boldsymbol{\mu},\boldsymbol{\Sigma}) = 2n_j \sum_{k_j=0}^{n_j-1}{}' \Gamma^{d}_{n-(n_j-k_j)e_j}(\boldsymbol{\mu},\boldsymbol{\Sigma}) \mathbb{1}_{(n_j-1+k_j)mod2=0}.$$

Proof. For the derivative of the (one-dimensional) Chebyshev polynomial holds $T_n'(x) = 2n \sum_{k=0}^{n-1}{}' T_k(x) \mathbb{1}_{(n-1+k)\mathrm{mod}2=0}$. This yields

$$\begin{aligned}
&\int_{\mathcal{I}^{d-1}} T_{n(j)}(\mathbf{x}_{(j)}) \int_{\mathcal{I}} \frac{\partial T_{n_j}(x_j)}{\partial x_j} \phi^d(\mathbf{x}) \mathrm{d}x_j \mathrm{d}\mathbf{x}_{(j)} \\
&\quad = \int_{\mathcal{I}^d} T_{n(j)}(\mathbf{x}_{(j)}) 2n_j \sum_{k_j=0}^{n_j-1}{}' T_{k_j}(x_j) \mathbb{1}_{(n_j-1+k_j)\mathrm{mod}2=0} \phi^d(\mathbf{x}) \mathrm{d}\mathbf{x} \\
&\quad = 2n_j \sum_{k_j=0}^{n_j-1}{}' \int_{\mathcal{I}^d} T_{n-(n_j-k_j)e_j}(\mathbf{x}) \phi^d(\mathbf{x}) \mathrm{d}\mathbf{x}\ \mathbb{1}_{(n_j-1+k_j)\mathrm{mod}2=0} \\
&\quad = 2n_j \sum_{k_j=0}^{n_j-1}{}' \Gamma^{d}_{n-(n_j-k_j)e_j}(\boldsymbol{\mu},\boldsymbol{\Sigma}) \mathbb{1}_{(n_j-1+k_j)\mathrm{mod}2=0}
\end{aligned}$$

which was our claim. □

Note that for the d-variate generalized moments we require the moments in dimension $d-1$. This dependence on moments in lower dimensions limits the applicability of the approach in high dimensions.

References

[1] Carol Alexander. *Market Risk Analysis, Pricing, Hedging and Trading Financial Instruments*, volume 3. John Wiley & Sons, 2008.

[2] Mark Azoulay, Daniel Härtl, Yuri Mushkin, and Anke Raufuss. FRTB reloaded:the need for a fundamental revamp of trading-risk infrastructure. *McKinsey & Company Risk Management*, 2018.

[3] Brett W Bader and Tamara G Kolda. Algorithm 862: Matlab tensor classes for fast algorithm prototyping. *ACM Transactions on Mathematical Software (TOMS)*, 32(4):635–653, 2006.

[4] Brett W. Bader, Tamara G. Kolda, et al. Matlab tensor toolbox version 3.1. Available online at `https://www.tensortoolbox.org`, June 2019.

[5] Jonas Ballani, Lars Grasedyck, and Melanie Kluge. Black box approximation of tensors in hierarchical Tucker format. *Linear algebra and its applications*, 438(2):639–657, 2013.

[6] Maxime Barrault, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T Patera. An 'empirical interpolation'method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[7] Volker Barthelmann, Erich Novak, and Klaus Ritter. High dimensional polynomial interpolation on sparse grids. *Advances in Computational Mathematics*, 12(4):273–288, 2000.

[8] David S Bates. Jumps and stochastic volatility: Exchange rate processes implicit in Deutsche Mark options. *The Review of Financial Studies*, 9(1):69–107, 1996.

[9] Zachary Battles and Lloyd N Trefethen. An extension of Matlab to continuous functions and operators. *SIAM Journal on Scientific Computing*, 25(5):1743–1770, 2004.

[10] J Baumeister. Inverse problems in finance. In *Recent Developments in Computational Finance: Foundations, Algorithms and Applications*, pages 81–157. World Scientific, 2013.

[11] Christian Bayer, Markus Siebenmorgen, and Raul Tempone. Smoothing the payoff for efficient computation of basket option prices. *Quantitative Finance*, 18(3):491–

505, 2018.

[12] Erhan Bayraktar. A proof of the smoothness of the finite time horizon American put option for jump diffusions. *SIAM Journal on Control and Optimization*, 48(2):551–572, 2009.

[13] Hatem Ben-Ameur, Michele Breton, Lotfi Karoui, and Pierre L'Ecuyer. A dynamic programming approach for pricing options embedded in bonds. *Journal of Economic Dynamics and Control*, 31(7):2212–2233, 2007.

[14] Nicholas H Bingham and Rüdiger Kiesel. *Risk-Neutral Valuation: Pricing and Hedging of Financial Derivatives*. Springer Science & Business Media, 2013.

[15] F. Black and M. Scholes. The pricing of options and other liabilities. *Journal of Political Economy*, 81:637–654, 1973.

[16] Fischer Black and Piotr Karasinski. Bond and option pricing when short rates are lognormal. *Financial Analysts Journal*, 47(4):52–59, 1991.

[17] Alexander Boogert and Cyriel De Jong. Gas storage valuation using a Monte Carlo method. *The Journal of Derivatives*, 15(3):81–98, 2008.

[18] L Bos, S De Marchi, and M Vianello. Polynomial approximation on Lissajous curves in the d-cube. *Applied Numerical Mathematics*, 116:47–56, 2017.

[19] M. J. Brennan and E. S. Schwartz. The valuation of American put options. *The Journal of Finance*, 2(32):449–462, 1977.

[20] Michael J Brennan and Eduardo S Schwartz. Savings bonds, retractable bonds and callable bonds. *Journal of Financial Economics*, 5(1):67–88, 1977.

[21] Damiano Brigo and Fabio Mercurio. *Interest Rate Models–Theory and Practice: With Smile, Inflation and Credit*. Springer Science & Business Media, 2007.

[22] L Brutman. On the Lebesgue function for polynomial interpolation. *SIAM Journal on Numerical Analysis*, 15(4):694–704, 1978.

[23] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[24] Hans-Joachim Bungartz, Alexander Heinecke, Dirk Pflüger, and Stefanie Schraufstetter. Option pricing with a direct adaptive sparse grid approach. *Journal of Computational and Applied Mathematics*, 236(15):3741–3750, 2012.

[25] Olena Burkovska, Maximilian Gaß, Kathrin Glau, Mirco Mahlstedt, Wim Schoutens, and Barbara Wohlmuth. Calibration to American options: numerical investigation of the de-Americanization method. *Quantitative Finance*, 18(7):1091–1113, 2018.

[26] Yongyang Cai and Kenneth L Judd. Shape-preserving dynamic programming. *Mathematical Methods of Operations Research*, 77(3):407–421, 2013.

[27] Marco Caliari, Stefano De Marchi, and Marco Vianello. Bivariate polynomial interpolation on the square at new nodal sets. *Applied Mathematics and Computation*, 165(2):261–274, 2005.

[28] Guglielmo Maria Caporale and Mario Cerrato. Using Chebyshev polynomials to approximate partial differential equations. *Computational Economics*, 35(3):235–

244, 2010.

[29] Peter Carr, Hélyette Geman, Dilip B Madan, and Marc Yor. The fine structure of asset returns: An empirical investigation. *The Journal of Business*, 75(2):305–332, 2002.

[30] Peter Carr and Dilip Madan. Option valuation using the fast Fourier transform. *Journal of Computational Finance*, 2(4):61–73, 1999.

[31] Ron Chan and Nicholas Hale. Hedging and pricing European-type, early-exercise and discrete barrier options using an algorithm for the convolution of Legendre series. *Management Science, Forthcoming*, 2018.

[32] CW Clenshaw. A note on the summation of Chebyshev series. *Mathematics of Computation*, 9(51):118–120, 1955.

[33] John C Cox and Stephen A Ross. The valuation of options for alternative stochastic processes. *Journal of Financial Economics*, 3(1-2):145–166, 1976.

[34] Y d'Halluin, PA Forsyth, KR Vetzal, and G Labahn. A numerical PDE approach for pricing callable bonds. *Applied Mathematical Finance*, 8(1):49–77, 2001.

[35] Victor Dominguez, IG Graham, and VP Smyshlyaev. Stability and error estimates for Filon–Clenshaw–Curtis rules for highly oscillatory integrals. *IMA Journal of Numerical Analysis*, 31(4):1253–1280, 2011.

[36] Paul Doust. Two useful techniques for financial modelling problems. *Applied Mathematical Finance*, 17(3):201–210, 2010.

[37] Tobin A Driscoll, Nicholas Hale, and Lloyd N Trefethen. Chebfun guide. Available online at `https://www.chebfun.org`, 2014.

[38] Darrell Duffie and Kenneth J Singleton. Modeling term structures of defaultable bonds. *The Review of Financial Studies*, 12(4):687–720, 1999.

[39] Bruno Dupire. Pricing with a smile. *Risk*, 7(1):18–20, 1994.

[40] Ernst Eberlein, Kathrin Glau, and Antonis Papapantoleon. Analysis of Fourier transform valuation formulas and applications. *Applied Mathematical Finance*, 17(3):211–240, 2010.

[41] Ernst Eberlein and Jan Kallsen. *Mathematical Finance*. Springer, 2019.

[42] Marcos Escobar, Mirco Mahlstedt, Sven Panz, and Rudi Zagst. Vulnerable exotic derivatives. *The Journal of Derivatives*, 24(3):84–102, 2017.

[43] Fang Fang and Cornelis W Oosterlee. A novel pricing method for European options based on Fourier-cosine series expansions. *SIAM Journal on Scientific Computing*, 31(2):826–848, 2008.

[44] Fang Fang and Cornelis W Oosterlee. Pricing early-exercise and discrete barrier options by Fourier-cosine series expansions. *Numerische Mathematik*, 114(1):27, 2009.

[45] Qian Feng, Shashi Jain, Patrik Karlsson, Drona Kandhai, and Cornelis W Oosterlee. Efficient computation of exposure profiles on real-world and risk-neutral scenarios for Bermudan swaptions. *Journal of Computational Finance*, 20(1):139–

172, 2016.

[46] Leslie Fox and Ian Bax Parker. *Chebyshev Polynomials in Numerical Analysis*. Oxford University Press, 1968.

[47] Maximilian Gaß. *PIDE methods and concepts for parametric option pricing*. PhD thesis, Technische Universität München, 2016.

[48] Maximilian Gaß, Kathrin Glau, Mirco Mahlstedt, and Maximilian Mair. Chebyshev interpolation for parametric option pricing. *arXiv preprint arXiv:1505.04648v2*, 2016.

[49] Maximilian Gaß, Kathrin Glau, Mirco Mahlstedt, and Maximilian Mair. Chebyshev interpolation for parametric option pricing. *Finance and Stochastics*, 22(3):701–731, 2018.

[50] Maximilian Gaß, Kathrin Glau, and Maximilian Mair. Magic points in finance: Empirical integration for parametric option pricing. *SIAM Journal on Financial Mathematics*, 8(1):766–803, 2017.

[51] Jim Gatheral. *The Volatility Surface: A Practitioner's Guide*, volume 357. John Wiley & Sons, 2011.

[52] Walter Gautschi. *Orthogonal Polynomials*. Oxford university press Oxford, 2004.

[53] Robert Geske and Herb E Johnson. The American put option valued analytically. *The Journal of Finance*, 39(5):1511–1524, 1984.

[54] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*, volume 53. Springer Science & Business Media, 2003.

[55] Kathrin Glau, Behnam Hashemi, Mirco Mahlstedt, and Christian Pötz. Spread option in 2D Black-Scholes model. Example for chebfun3 in the chebfun toolbox., 2017. Available online at `http://www.chebfun.org/examples/applics/BlackScholes2D.html`.

[56] Kathrin Glau, Paul Herold, Dilip B Madan, and Christian Pötz. The Chebyshev method for the implied volatility. *Journal of Computational Finance*, 23(3), 2019.

[57] Kathrin Glau, Daniel Kressner, and Francesco Statti. Low-rank tensor approximation for Chebyshev interpolation in parametric option pricing. *arXiv preprint arXiv:1902.04367*, 2019.

[58] Kathrin Glau and Mirco Mahlstedt. Improved error bound for multivariate Chebyshev polynomial interpolation. *International Journal of Computer Mathematics*, 96(11):2302–2314, 2019.

[59] Kathrin Glau, Mirco Mahlstedt, and Christian Pötz. A new approach for American option pricing: The Dynamic Chebyshev method. *SIAM Journal on Scientific Computing*, 41(1):B153–B180, 2019.

[60] Kathrin Glau, Ricardo Pachon, and Christian Pötz. Fast calculation of credit exposures for barrier and Bermudan options using Chebyshev interpolation. *arXiv preprint arXiv:1905.00238*, 2019.

[61] Kathrin Glau, Ricardo Pachon, and Christian Pötz. Speed-up credit exposure

calculations for pricing and risk management. *Quantitative Finance*, pages 1–19, 2020.

[62] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[63] Andrew Green. *XVA: Credit, Funding and Capital Valuation Adjustments*. John Wiley & Sons, 2015.

[64] Jon Gregory. *Counterparty Credit Risk: The New Challenge for Global Financial Markets*, volume 470. John Wiley & Sons, 2010.

[65] Bernard Haasdonk, Julien Salomon, and Barbara Wohlmuth. A reduced basis method for the simulation of American options. In *Numerical Mathematics and Advanced Applications 2011*, pages 821–829. Springer, 2013.

[66] Tinne Haentjens and Karel J in't Hout. ADI schemes for pricing American options under the Heston model. *Applied Mathematical Finance*, 22(3):207–237, 2015.

[67] Patrick S Hagan, Deep Kumar, Andrew S Lesniewski, and Diana E Woodward. Managing smile risk. *The Best of Wilmott*, 1:249–296, 2002.

[68] Paul Herold. Interpolation of implied volatilities via Chebyshev interpolation. Master thesis, Technical University of Munich, 2017.

[69] Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, 6(2):327–343, 1993.

[70] N. Hilber, N. Reich, C. Winter, and C. Schwab. *Computational Methods for Qantitative Finance*. Springer, 2013.

[71] John Hull and Alan White. Pricing interest-rate-derivative securities. *The Review of Financial Studies*, 3(4):573–592, 1990.

[72] John C Hull. *Options Futures and other Derivatives*. Pearson Education India, 2003.

[73] Karel J. In't Hout and Jacob Snoeijer. Numerical valuation of Bermudan basket options via partial differential equations. *arXiv preprint arXiv:1909.01164*, 2019.

[74] Peter Jäckel. By implication. *Wilmott*, 26:60–66, 2006.

[75] Peter Jäckel. Let's be rational. *Wilmott*, 2015(75):40–53, 2015.

[76] Shashi Jain and Cornelis W Oosterlee. The stochastic grid bundling method: Efficient pricing of Bermudan options and their Greeks. *Applied Mathematics and Computation*, 269:412–431, 2015.

[77] Robert Jarrow, Haitao Li, Sheen Liu, and Chunchi Wu. Reduced-form valuation of callable corporate bonds: Theory and evidence. *Journal of Financial Economics*, 95(2):227–248, 2010.

[78] Kenneth L Judd. *Numerical Methods in Economics*. MIT press, 1998.

[79] Raymond Kan and Cesare Robotti. On moments of folded and truncated multivariate normal distributions. *Journal of Computational and Graphical Statistics*, 26(4):930–934, 2017.

[80] Patrik Karlsson, Shashi Jain, and Cornelis W Oosterlee. Counterparty credit ex-

posures for interest rate derivatives using the stochastic grid bundling method. *Applied Mathematical Finance*, 23(3):175–196, 2016.

[81] Rüdiger Kiesel, Jochen Gernhard, and Sven-Olaf Stoll. Valuation of commodity-based swing options. *The Journal of Energy Markets*, 3(3):91, 2010.

[82] Michael Kohler, Adam Krzyżak, and Nebojsa Todorovic. Pricing of high-dimensional American options by neural networks. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 20(3):383–410, 2010.

[83] Ralf Korn, Elke Korn, and Gerald Kroisandt. *Monte Carlo methods and models in finance and insurance*. CRC press, 2010.

[84] Daniel Kressner and Christine Tobler. Algorithm 941: htucker—a matlab toolbox for tensors in hierarchical Tucker format. *ACM Transactions on Mathematical Software (TOMS)*, 40(3):1–22, 2014.

[85] SZ Levendorskiĭ. Pricing of the American put under Lévy processes. *International Journal of Theoretical and Applied Finance*, 7(03):303–335, 2004.

[86] Minqiang Li. Approximate inversion of the Black–Scholes formula using rational functions. *European Journal of Operational Research*, 185(2):743–759, 2008.

[87] Dongjae Lim, Lingfei Li, and Vadim Linetsky. Evaluating callable and putable bonds: an eigenfunction expansion approach. *Journal of Economic Dynamics and Control*, 36(12):1888–1908, 2012.

[88] F. A. Longstaff and E. S. Schwartz. Valuing American Options by Simulation: A Simple Least-Squares Approach. *Review of Financial Studies*, 14(1):113–147, 2001.

[89] Roger Lord, Fang Fang, Frank Bervoets, and Cornelis W Oosterlee. A fast and accurate FFT-based method for pricing early-exercise options under Lévy processes. *SIAM Journal on Scientific Computing*, 30(4):1678–1705, 2008.

[90] Dilip B Madan. Adapted hedging. *Annals of Finance*, 12(3-4):305–334, 2016.

[91] Mirco Mahlstedt. *Complexity Reduction for Option Pricing*. PhD thesis, Technische Universität München, 2017.

[92] Steven Manaster and Gary Koehler. The calculation of implied variances from the Black–Scholes model: A note. *The Journal of Finance*, 37(1):227–230, 1982.

[93] John C Mason and David C Handscomb. *Chebyshev Polynomials*. Chapman and Hall/CRC, 2002.

[94] R. C. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3:125–144, 1976.

[95] Robert C Merton. Theory of rational option pricing. *The Bell Journal of Economics and Management Science*, pages 141–183, 1973.

[96] Attilio Meucci. 'P' versus 'Q': Differences and commonalities between the two areas of quantitative finance. *GARP Risk Professional*, pages 47–50, 2011.

[97] Slobodan Milovanović and Lina von Sydow. Radial basis function generated finite differences for option pricing problems. *Computers & Mathematics with Applica-*

tions, 75(4):1462–1481, 2018.

[98] Alejandro Mosiño. Using Chebyshev polynomials to approximate partial differential equations: A reply. *Computational Economics*, 39(1):13–27, 2012.

[99] Ravi Myneni. The pricing of the American option. *The Annals of Applied Probability*, pages 1–23, 1992.

[100] Daniel B Nelson and Krishna Ramaswamy. Simple binomial processes as diffusion approximations in financial models. *The Review of Financial Studies*, 3(3):393–430, 1990.

[101] Cornelis W Oosterlee and Lech A Grzelak. *Mathematical Modeling and Computation in Finance*. World Scientific, 2019.

[102] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[103] Ricardo Pachón. Numerical pricing of European options with arbitrary payoffs. *International Journal of Financial Engineering*, 5(02):1850015, 2018.

[104] Ricardo Pachón, Rodrigo B Platte, and Lloyd N Trefethen. Piecewise-smooth chebfuns. *IMA journal of numerical analysis*, 30(4):898–916, 2010.

[105] Ricardo Pachón and Lloyd N Trefethen. Barycentric-Remez algorithms for best polynomial approximation in the chebfun system. *BIT Numerical Mathematics*, 49(4):721, 2009.

[106] Goran Peskir and Albert Shiryaev. *Optimal Stopping and Free-Boundary Problems*. Springer, 2006.

[107] Christian Pötz. Chebyshev interpolation for parametric option pricing: Empirical and theoretical investigations. Master thesis, Technical University of Munich, 2016.

[108] Christoph Reisinger and Gabriel Wittum. Efficient hierarchical approximation of high-dimensional option pricing problems. *SIAM Journal on Scientific Computing*, 29(1):440–458, 2007.

[109] Marjon J Ruijter and Cornelis W Oosterlee. Two-dimensional Fourier cosine series expansion method for pricing financial options. *SIAM Journal on Scientific Computing*, 34(5):B642–B671, 2012.

[110] Carl Runge. Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten. *Zeitschrift für Mathematik und Physik*, 46(224-243):20, 1901.

[111] Oliver Salazar Celis. A parametrized barycentric approximation for inverse problems with application to the Black–Scholes formula. *IMA Journal of Numerical Analysis*, 2017.

[112] Robert Schöftner. On the estimation of credit exposures using regression-based Monte Carlo simulation. *The Journal of Credit Risk*, 4(4):37–62, 2008.

[113] Wim Schoutens, Erwin Simons, and Jurgen Tistaert. A perfect calibration! now what? *The best of Wilmott*, page 281, 2003.

[114] Mark Schroder. Computing the constant elasticity of variance option pricing formula. *The Journal of Finance*, 44(1):211–219, 1989.

[115] Rüdiger Seydel. *Tools for Computational Finance*, volume 3. Springer, 2006.

[116] Victor Shcherbakov and Elisabeth Larsson. Radial basis function partition of unity methods for pricing vanilla basket options. *Computers & Mathematics with Applications*, 71(1):185–200, 2016.

[117] Yanbin Shen, Johannes AM Van Der Weide, and Jasper HM Anderluh. A benchmark approach of counterparty credit exposure of Bermudan option under Lévy process: the Monte Carlo-COS method. *Procedia Computer Science*, 18:1163–1171, 2013.

[118] Yaovi Gassesse Siliadin. *Essays on Credit Risk and Callable Bonds Valuation*. HEC Montreal (Canada), 2016.

[119] S Simaitis, CSL de Graaf, N Hari, and D Kandhai. Smile and default: the role of stochastic volatility and interest rates in counterparty credit risk. *Quantitative Finance*, 16(11):1725–1740, 2016.

[120] Sergei Abramovich Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. In *Doklady Akademii Nauk*, volume 148, pages 1042–1045. Russian Academy of Sciences, 1963.

[121] Fazlollah Soleymani. Pricing multi-asset option problems: a Chebyshev pseudospectral method. *BIT Numerical Mathematics*, 59(1):243–270, 2019.

[122] Harvey J Stein. Fixing risk neutral risk measures. *International Journal of Theoretical and Applied Finance*, 19(03):1650021, 2016.

[123] Michael Steinlechner. Riemannian optimization for high-dimensional tensor completion. *SIAM Journal on Scientific Computing*, 38(5):S461–S484, 2016.

[124] Lars Stentoft. Assessing the least squares Monte-Carlo approach to American option valuation. *Review of Derivatives Research*, 7(2):129–168, 2004.

[125] Eitan Tadmor. The exponential accuracy of Fourier and Chebyshev differencing methods. *SIAM Journal on Numerical Analysis*, 23(1):1–10, 1986.

[126] Yung Liang Tong. *The Multivariate Normal Distribution*. Springer Science & Business Media, 1990.

[127] Alex Townsend and Lloyd N Trefethen. An extension of chebfun to two dimensions. *SIAM Journal on Scientific Computing*, 35(6):C495–C518, 2013.

[128] Lloyd N Trefethen. Six myths of polynomial interpolation and quadrature. *Mathematics TODAY*, 2011.

[129] Lloyd N Trefethen. *Approximation Theory and Approximation Practice*. SIAM books, 2013.

[130] Lloyd N Trefethen. Cubature, approximation, and isotropy in the hypercube. *SIAM Review*, 59(3):469–491, 2017.

[131] Lloyd N Trefethen. Multivariate polynomial approximation in the hypercube. *Proceedings of the American Mathematical Society*, 145(11):4837–4844, 2017.

[132] Francesco Giacomo Tricomi. *Vorlesungen über Orthogonalreihen*, volume 76. Springer-Verlag, 1955.

[133] Colin Turfus. Perturbation expansion for Arrow-Debreu pricing with Hull-White

interest rates and Black-Karasinski credit intensity. *Available at SSRN 3287910*, 2019.

[134] Oldrich Vasicek. An equilibrium characterization of the term structure. *Journal of Financial Economics*, 5(2):177–188, 1977.

[135] Lina von Sydow, Lars Josef Höök, Elisabeth Larsson, Erik Lindström, Slobodan Milovanović, Jonas Persson, Victor Shcherbakov, Yuri Shpolyanskiy, Samuel Sirén, Jari Toivanen, et al. BENCHOP–The BENCHmarking project in option pricing. *International Journal of Computer Mathematics*, 92(12):2361–2379, 2015.

[136] Paul Wilmott. *Paul Wilmott Introduces Quantitative Finance*. John Wiley & Sons, 2007.

[137] Shu Yeung Wong. Low-rank tensor approximation methods for financial problems. Master thesis, Technical University of Munich, 2018.

[138] Kosaku Yosida. *Functional Analysis*. Springer, 1980.

[139] Christoph Zenger. Sparse grids. In *Parallel Algorithms for Partial Differential Equations*, 1991.

[140] B Zhang and CW Oosterlee. An efficient pricing algorithm for swing options based on Fourier cosine expansions. *Journal of Computational Finance*, 16(4):1–32, 2013.

[141] Tao Zhou, Akil Narayan, and Zhiqiang Xu. Multivariate discrete least-squares approximations with a new type of collocation grid. *SIAM Journal on Scientific Computing*, 36(5):A2401–A2422, 2014.