Hybrid Monte Carlo methods in computational finance

Leitao Rodriguez, Alvaro

DOI
10.4233/uuid:fc4d5fc5-cee9-44ef-bca1-e69250c1480f

Publication date
2017

Document Version
Final published version

Citation (APA)
Leitao Rodriguez, A. (2017). *Hybrid Monte Carlo methods in computational finance*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:fc4d5fc5-cee9-44ef-bca1-e69250c1480f

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

HYBRID MONTE CARLO METHODS IN COMPUTATIONAL FINANCE

HYBRID MONTE CARLO METHODS IN COMPUTATIONAL FINANCE

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K. C. A. M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 27 juni 2017 om 12:30 uur.

door

Álvaro LEITAO RODRÍGUEZ

Master of Science in Mathematical Engineering
Universiteit van Vigo, Spanje
geboren te Xove, Spanje.

Dit proefschrift is goedgekeurd door de promotor:

Prof. dr. ir. C. W. Oosterlee

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof. dr. ir. C. W. Oosterlee	Technische Universiteit Delft, promotor

Onafhankelijke leden:

Prof. dr. A. Pascucci	Università di Bologna, Italië
Prof. dr. K. In't Hout	Universiteit Antwerpen, België
Prof. dr. B. D. Kandhai	Universiteit van Amsterdam
Prof. dr. ir. A. W. Heemink	Technische Universiteit Delft
Prof. dr. C. Witteveen	Technische Universiteit Delft

Overig lid:

Dr. ir. L. A. Grzelak	Technische Universiteit Delft

This research was partially supported by the EU in the FP7-PEOPLE-2012-ITN Program under Grant Agreement Number 304617 (FP7 Marie Curie Actions, Project Multi-ITN STRIKE – Novel Methods in Computational Finance).

Hybrid Monte Carlo methods in computational finance
Dissertation at Delft University of Technology

Printed by: IPSKAMP printing

ISBN 978-94-028-0681-6

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

To Mela;

To my parents;

Two things are infinite:
the universe and human stupidity;
and I'm not sure about the universe.

Albert Einstein

Summary

Monte Carlo methods are highly appreciated and intensively employed in computational finance in the context of financial derivatives valuation or risk management. The method offers valuable advantages like flexibility, easy interpretation and straightforward implementation. Furthermore, the dimensionality of the financial problem can be increased without reducing the efficiency significantly. The latter feature of Monte Carlo methods is important since it represents a clear advantage over other competing numerical methods. Furthermore, in the case of option valuation problems in multiple dimensions (typically more than five), the Monte Carlo method and its variants become the only possible choices. Basically, the Monte Carlo method is based on the simulation of possible *scenarios* of an underlying process and by then aggregating their values for a final solution. Pricing derivatives on equity and interest rates, risk assessment or portfolio valuation are some of the representative examples in finance, where Monte Carlo methods perform very satisfactorily. The main drawback attributed to these methods is the rather poor balance between computational cost and accuracy, according to the theoretical rate of Monte Carlo convergence. Based on the *central limit theorem*, the Monte Carlo method requires hundred times more scenarios to reduce the error by one order, i.e. the rate of convergence is $\mathcal{O}\left(n^{-\frac{1}{2}}\right)$.

However, the application of basic Monte Carlo methods, usually called *plain* Monte Carlo methods, is often not sufficient to solve modern problems appearing in the financial sector. Problems like the accurate computation of *sensitivities* and *early-exercise* financial derivatives in multiple dimensions are particularly interesting examples. Except for the simplest asset price stochastic models, the generation of the scenarios is not always a trivial task. Although the so-called Taylor-based discretization schemes, like the Euler-Maruyama scheme, are heavily used, more involved mechanisms are required to achieve the goal of accuracy and efficiency in a number of cases. For the computation of the sensitivities and for early-exercise valuation, the use of plain Monte Carlo is either not sufficient or computationally unaffordable.

Therefore, hybrid solution methods, i.e., combining Monte Carlo methods with some other methodologies, can be developed in order to overcome the issues and provide fast, accurate and consistent computational results. This can be done by relying on mathematical, computational approaches, or combinations of both. Hybrid solution methods are becoming essential tools to deal with the increasing complexity in quantitative finance.

In this thesis, we propose Monte Carlo-based hybrid solution methods for some *cutting edge* problems in computational finance. From the mathematical side, we employ Fourier-based techniques that, in combination with the Monte Carlo methods, result in efficient methodologies to simulate and evaluate sensitivities under *stochastic local volatility models*, that are nowadays used for foreign-exchange (FX) rate options. We also employ some recent scientific computing paradigms advances, like high-performance computing on GPUs, to extend the applicability of a Monte Carlo method for pricing

multi-dimensional early-exercise options and thus enable the computation of very high-dimensional problems.

In Chapter 1, a general overview of the methodologies that commonly appear in computational finance is presented. By considering the solution of partial differential equations (PDEs) provided by the *Feynman-Kac theorem* and its variants, mathematical techniques like Fourier inversion, Monte Carlo methods and PDE approaches have been successfully connected to solve financial problems like option pricing and risk management. We provide a brief explanation of the techniques. Other recent developments like data-driven approaches, where a pricing model is derived solely based on the available financial data, or parallel GPU computing are also briefly described, as they are considered in the second part of this thesis.

In Chapter 2, we propose an efficient technique to simulate exactly a representative stochastic local volatility model, i.e. the Stochastic Alpha Beta Rho (SABR) model, considered well-established in FX and interest rate markets. We aim to employ only one time-step in a Monte Carlo procedure. Particularly, we focus on the derivation of the conditional distribution of the appearing *time-integrated variance*, an element which is required in the simulation of the SABR asset dynamics. For that purpose, we compute the marginal distribution by means of Fourier inversion and the use a copula to obtain a multivariate distribution which resembles the conditional time-integrated variance distribution. Being a one time-step method, the method is suitable for the valuation of European options and, due to the presented improvements in terms of performance, can be particularly employed for calibration purposes.

The method introduced in Chapter 2 is generalized towards a multiple time-step version for the simulation of the SABR model in Chapter 3. Although the multiple time-step procedure is conceptually similar, this version entails a new challenge regarding the computational cost. We deal with this challenge by introducing an advanced numerical interpolation which is based on a stochastic collocation technique. The proposed method, which we call the mSABR method, allows for an efficient simulation with large time-steps, and it can be employed to accurately price (*exotic*), non-standard, options under the SABR dynamics. This simulation scheme also performs highly satisfactory even in the context of negative interest rates, that are presently observed in the financial world.

In Chapter 4, we develop a data-driven Fourier-based option valuation method that appears beneficial when calculating the sensitivities of financial derivatives in a general framework. The method is based on the assumption that only asset data samples are available. The resulting method is a data-based generalization of *the COS method*, which is a Fourier inversion method that employs the *characteristic function* of the relevant stochastic asset variables. In our method, which we call the data-driven COS (ddCOS) method, the use of the characteristic function is avoided. Instead, we recover the probability density function from the available data samples, by using concepts from the *statistical learning theory*. Since the ddCOS method is a data-driven procedure it is, thus, generally applicable and can be employed, for example, in combination with the above mentioned SABR simulation method, as described in Chapters 2 and 3.

In Chapter 5, we consider another component in our work, the efficient computation on Graphics Processing Units (GPUs). We present a parallel version of an effi-

cient Monte Carlo-based method for pricing multi-dimensional early-exercise derivatives. Parallelization and GPU computation give an extraordinary gain in terms of computational cost reduction, which allows us to explore new applications of the method. Very high-dimensional problems become tractable and generalizations regarding the early-exercise policy are developed thanks to our GPU implementation.

The work developed in this PhD Thesis is based on journal articles, that have either been published or been submitted during the doctoral research period.

Samenvatting

Monte-Carlomethoden worden zeer gewaardeerd en intensief gebruikt binnen de financiële wiskunde voor het waarderen van financiële derivaten en voor het beheersen van risico's. Monte-Carlosimulatie biedt waardevolle voordelen zoals flexibiliteit, makkelijke interpretatie en eenvoudige implementatie. Bovendien kan de dimensie van het financiële vraagstuk worden verhoogd zonder de efficiëntie significant te verminderen. Dit laatste kenmerk van Monte-Carlomethoden is belangrijk, omdat het een duidelijk voordeel is vergeleken met andere, concurrerende numerieke methoden. Voor het waarderen van opties in meerdere dimensies (typisch meer dan vijf) zijn Monte-Carlosimulatie en zijn varianten zelfs de enige mogelijke keuzes. In principe is Monte-Carlosimulatie gebaseerd op de simulatie van mogelijke *scenario's* van een onderliggend proces, waarbij vervolgens de verkregen scenariowaarden worden gecombineerd om tot de uiteindelijke oplossing te komen. Het prijzen van derivaten met als onderliggende aandeel- en rentekoersen, het beoordelen van risico's en het waarderen van de portefeuille zijn representatieve financiële voorbeelden waarbij Monte-Carlosimulatie zeer goed presteert. Het grootste nadeel dat aan Monte-Carlo wordt toegeschreven is de theoretische snelheid van de convergentie, oftewel het vrij slechte evenwicht tussen de kosten van de berekening en de bijbehorende nauwkeurigheid. Op basis van de *centrale limietstelling* vereist Monte-Carlosimulatie honderd keer meer scenario's om de fout met één orde te verminderen, d.w.z. de convergentiesnelheid is $\mathcal{O}\left(n^{-\frac{1}{2}}\right)$.

Echter, de toepassing van basis Monte-Carlomethoden is vaak niet voldoende om moderne vraagstukken op te lossen die in de financiële sector voorkomen. Problemen zoals de nauwkeurige berekening van de *gevoeligheid* en al dan niet *vervroegd uitoefenen* van financiële derivaten in meerdere dimensies zijn interessante voorbeelden. Alleen bij de eenvoudigste stochastische modellen voor financiële producten is het genereren van de scenario's altijd triviaal. Hoewel discretisatieschema's die op Taylorontwikkeling gebaseerd zijn, zoals het Euler-Maruyama-schema, veel gebruikt worden, zijn er in een aantal gevallen ingewikkeldere technieken vereist om een bepaalde nauwkeurigheid en efficiëntie te bereiken. Voor het berekenen van de gevoeligheid en in het geval van vervroegd uitoefenen is het gebruik van basis Monte-Carlomethoden ofwel niet toereikend of de rekentijden zijn te hoog.

Daarom kunnen hybride oplosmethoden, d.w.z. het combineren van Monte-Carlomethoden met een aantal andere technieken, worden ontwikkeld om deze moeilijkheden te overwinnen en snelle, nauwkeurige en consistente resultaten te behalen. Dit kan gedaan worden door te vertrouwen op analytische en/of numerieke benaderingen. Hybride oplosmethoden zullen in de toekomst essentieel zijn om de toenemende complexiteit in kwantitatieve financiële wiskunde te hanteren.

In dit proefschrift dragen we op Monte-Carlo gebaseerde hybride oplosmethoden aan voor sommige moderne vraagstukken in financiële wiskunde. Vanuit de wiskundige kant gebruiken we Fouriertechnieken die, gecombineerd met de Monte-Carlomethoden, resulteren in efficiënte technieken voor het simuleren en schatten van gevoeligheden

onder *stochastische lokale volatiliteitsmodellen*, die op dit ogenblik veel gebruikt worden voor opties in de valutamarkt (FX). We gebruiken ook enkele recente technologische ontwikkelingen, zoals het snel uitvoeren van complexe berekeningen op de grafische kaart, om zo de toepasbaarheid van Monte-Carlosimulatie uit te breiden voor het berekenen van multidimensionale opties waarbij vervroegd uitoefenen toegestaan is. Dit maakt het numeriek oplossen van zeer hoog-dimensionale problemen mogelijk.

In Hoofdstuk 1 wordt een algemeen overzicht gegeven van methoden die vaak in de financiële wiskunde voorkomen. Door het beschouwen van de oplossing van partiële differentiaalvergelijkingen (PDV's) die door de *Feynman-Kac-stelling* en zijn varianten worden gegeven, zijn wiskundige technieken zoals Fourier inversie, Monte-Carlosimulatie en PDV-benaderingen succesvol gecombineerd om financiële vraagstukken, zoals het prijzen van opties en risicobeheer, op te lossen. We geven een korte toelichting op deze technieken. Andere recente ontwikkelingen, die we beschouwen in het tweede deel van dit proefschrift, worden ook kort beschreven in dit hoofdstuk. Door data gegenereerde benaderingen, waarbij een prijsmodel alleen gevormd wordt door de beschikbare financiële data, en het uitvoeren van parallelle berekeningen op een grafische kaart zijn voorbeelden van zulke ontwikkelingen.

In Hoofdstuk 2 stellen we een efficiënte techniek voor om een representatief stochastisch lokale volatiliteitsmodel exact te simuleren, namelijk het Stochastic Alpha Beta Rho-model (SABR), dat in de FX- en rentemarkten vaak gebruikt wordt. Wij streven ernaar om slechts één stap in een Monte-Carloprocedure te gebruiken. In het bijzonder richten we ons op de afleiding van de voorwaardelijke verdeling van de naar de *tijd geïntegreerde variantie*, een element dat nodig is bij de simulatie van het onderliggende SABR-proces. Daarvoor berekenen we de marginale verdeling door middel van Fouriertechnieken en het gebruik van een copula om een multivariate verdeling te verkrijgen die lijkt op de voorwaardelijke verdeling van de naar de tijd geïntegreerde variantie. Doordat de methode slechts één tijdstap nodig heeft, is het een methode die geschikt is voor het prijzen van Europese opties. Daarnaast hebben we verbeteringen voorgesteld, waardoor de methode ook kan worden gebruikt voor kalibratiedoeleinden.

De methode die in Hoofdstuk 2 wordt geïntroduceerd wordt in Hoofdstuk 3 gegeneraliseerd naar een versie met meerdere tijdstappen voor de simulatie van het SABR-model. Hoewel het gebruik van meerdere tijdstappen qua concept gelijk is, zorgt deze versie voor een nieuwe uitdaging met betrekking tot de rekentijd. Hiervoor introduceren we een geavanceerde numerieke interpolatie die gebaseerd is op stochastische collocatie. De voorgestelde methode, die we de mSABR-methode noemen, zorgt voor een efficiënte simulatie met grote tijdstappen en kan worden gebruikt om nauwkeurig niet-standaard (*exotische*) opties onder het SABR-model te prijzen. Zelfs onder negatieve rentetarieven, welke momenteel in de financiële wereld worden waargenomen, presteert dit simulatieschema zeer goed.

In Hoofdstuk 4 ontwikkelen we een data-gestuurde en op Fouriertechnieken gebaseerde prijsmethode voor opties. Deze methode blijkt in een algemene setting gunstig bij het berekenen van de gevoeligheden van financiële derivaten. De methode is gebaseerd op de veronderstelling dat er alleen gegevens van de onderliggende waarde beschikbaar zijn. De resulterende methode is een op data gebaseerde generalisatie van de *COS-methode*. Dit is een Fouriermethode die de *karakteristieke functie* van het onderlig-

gende proces gebruikt. In onze methode, die we de data-driven COS (ddCOS) methode noemen, wordt het gebruik van de karakteristieke functie vermeden. In plaats daarvan benaderen we de kansdichtheidsfunctie uit de beschikbare gegevens, door gebruik te maken van concepten uit de *statistische leertheorie*. Aangezien de ddCOS-methode een data-gestuurde techniek is, is het dus algemeen toepasbaar en kan het bijvoorbeeld worden gebruikt in combinatie met de SABR-simulatiemethode zoals beschreven in Hoofdstukken 2 en 3.

In Hoofdstuk 5 beschouwen we een ander onderdeel van ons werk, efficiënte berekeningen op de grafische kaart, ook wel Graphics Processing Unit (GPU) genoemd. We presenteren een parallelle versie van een efficiënte en op Monte-Carlo gebaseerde methode voor het prijzen van multidimensionale derivaten waarbij vervroegd uitoefenen toegestaan is. De parallellisatie en het gebruik van de GPU resulteren in een uitzonderlijke besparing in de rekentijd, waardoor we nieuwe toepassingen van de methode kunnen onderzoeken. Zeer hoog-dimensionale problemen worden nu hanteerbaar en generalisaties over het al dan niet vroegtijdig uitoefenen kunnen worden ontwikkeld dankzij onze GPU-implementatie.

Het werk in dit proefschrift is gebaseerd op artikelen die zijn gepubliceerd of ingediend tijdens het promotietraject.

Contents

List of most used abbreviations

AV	Antithetic Variates
bp	Basis points
CDF	Cumulative Distribution Function
CEV	Constant Elasticity of Variance
ChF	Characteristic function
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
ES	Expected Shortfall
FX	Foreign-eXchange
GBM	Geometric Brownian Motion
GOF	Goodness-of-fit
GPGPU	General-Purpose computing on Graphics Processing Units
GPU	Graphics Processing Unit
MCFD	Monte Carlo Finite Difference
MISE	Mean Integrated Squared Error
MSE	Mean Squared Error
PDE	Partial Differential Equation
PDF	Probability Density Function
PIDE	Partial Integro-Differential Equation
RE	Relative Error
SABR	Stochastic Alpha Beta Rho
SCMC	Stochastic Collocation Monte Carlo
SCvM	Smirnov-Cramér-von Mises
SDE	Stochastic Differential Equation
SGBM	Stochastic Grid Bundling Method
SV	Stochastic Volatility
SLV	Stochastic Local Volatility
VaR	Value-at-Risk

CHAPTER 1

Introduction

1.1. INTRODUCTION

Risk management and valuation problems as they appear in quantitative finance can be very challenging from both a mathematical and a computational point of view.

The underlying financial assets are typically governed by random and uncertain movements, that are mathematically modelled by means of *stochastic processes* and corresponding *Stochastic Differential Equations* (SDEs). Except for the most basic and classical models, closed-form expressions of the probability densities, governing these stochastic quantities are not available. For many relevant asset price processes, however, we know the characteristic function (ChF), which is the Fourier transform of the probability density function (PDF) of interest. By Fourier inversion we can thus approximate the governing PDFs.

Pricing financial derivatives like options is one of the most common research areas in computational finance. A derivative is a financial product whose valuation depends on another underlying asset such as stocks, commodities or interest rates. In this thesis, we focus on option valuation. Except for particular and simple cases, pricing an option is not a trivial task, depending on the type of the option and/or the stochastic model for the asset movements.

For any financial institution, risk management forms an essential part of their business. Some of the risks appearing in finance include credit risk, market risk and counterparty risk. Important issues in risk management are described in the *Basel Accords*, whose requirements must be satisfied by the banks and other financial institutions. Once the risk is identified, quantification is necessary. The estimation of accurate risk measures facilitates the interpretation of particular risk factors.

1.2. STOCHASTIC PROCESSES

A basic technique to deal with stochastic processes and SDEs from a mathematical point of view is the well-established *Itô's calculus*. A central concept within the Itô's calculus is the Itô's integral, which defines a stochastic integral with respect to a *Brownian motion*. Based on this, the *Itô's stochastic process*, in integral form, is defined as

$$X(t) - X(s) = \int_s^t a(z, X)\mathrm{d}z + \int_s^t b(z, X)\mathrm{d}W(z)$$

with W a Brownian motion and $a(\cdot)$ and $b(\cdot)$ predictable and integrable functions.

The stochastic processes appearing in computational finance are often Itô's processes. The most basic and extensively used example is the *geometric Brownian mo-*

tion (GBM) where the functions $a(\cdot)$ and $b(\cdot)$ are defined as $a(t,X) = \mu X(t)$ and $b(t,X) = \sigma X(t)$, with μ and σ constants. GBM is used to model the underlying asset prices in the classical Black-Scholes model, leading to log-normal asset dynamics. However, the assumed simplifications (like the constant parameters among others) have often shown to be inconsistent with the financial market behaviour.

An important indicator of the market inconsistency of GBM and the Black-Scholes equation is found in the *implied volatility smile*, which means that the volatility deduced from option quotes in the market is not constant, but appears as a function of the option strike prices and of time. A first attempt was found in the *local volatility models*. The model reproduces the stochastic behaviour of the volatility observed in the market already within the parameter calibration process. Parametric and non-parametric local volatility processes exist, where the parametric version is governed by a functional form with a few free parameters to be calibrated, and the non-parametric version is solely based on option market data. One of the first parametric volatility processes was the *constant elasticity of variance* (CEV) model [31]. The model aims to reproduce the stochastic behaviour of the volatility observed in the market by including an elasticity parameter.

Another particularly interesting asset processes are formed by exponential Lévy processes. Lévy processes are governed by the properties of independent and stationary increments and continuity in probabilistic sense. A useful formula in the context of Lévy processes is the celebrated *Lévy-Khintchine formula* that gives us a representation of the process by its ChF. The ChFs of Lévy processes and of regular affine diffusion processes are known, or can be computed, and by these, many relevant asset price processes can be represented, like jump processes, jump-diffusion or stochastic volatility processes. With these processes, the market observed volatility smile can be modelled.

The adoption of multiple-dimensional stochastic processes forms another successfully attempt in the context of modelling implied volatility smiles, especially by considering the volatility as a stochastic process. This leads to the *stochastic volatility* (SV) models[1] or even *stochastic local volatility* (SLV) models. One of the representative examples is *the Stochastic Alpha Beta Rho (SABR) model* in the Foreign-exchange (FX) and interest rates markets. The SABR model will be intensively studied throughout this thesis, for that, in the following section we briefly describe its main features.

It is important to notice that particularly these SLV models do not immediately result in a ChF or PDF, and as such it is not trivial to perform a rapid calibration of the model for a wide range of model and pricing problem parameters. By means of perturbation theory, closed-form solutions for implied volatilities can be obtained in the form of an analytic expression for a range of parameter values, but, as for many solution originating from asymptotic theory, this expression is not general. Often in practice, contracts are defined whose parameters lie outside the range of validity of the theory. In that case, it is really difficult to perform a fast and accurate calibration.

1.2.1. THE SABR MODEL

As mentioned, the SABR model belongs to the SLV model class. The model was introduced by Hagan et al. [67], aiming to capture the volatility smile and manage the potential risk originating from unexpected future volatility movements. The SABR model is

[1] Some SV models indeed fall in the class of Lévy processes, enabling the use of the Lévy-Khintchine formula.

based on, besides stochasticity in the volatility, a parametric local volatility component in terms of a model parameter, β as a power in S^β. The definition of the SABR model is according to the following system of SDEs,

$$\begin{aligned} \mathrm{d}S(t) &= \sigma(t)S^\beta(t)\mathrm{d}W_S(t), \quad & S(0) &= S_0\exp(rT), \\ \mathrm{d}\sigma(t) &= \alpha\sigma(t)\mathrm{d}W_\sigma(t), & \sigma(0) &= \sigma_0. \end{aligned}$$

with $S(t) = \bar{S}(t)\exp(r(T-t))$ the forward value of the underlying asset $\bar{S}(t)$, r the interest rate, S_0 the spot price at $t_0 = 0$, and T the contract's expiry time. The second stochastic process $\sigma(t)$ represents the stochastic volatility, with $\sigma(0) = \sigma_0$. The two correlated Brownian motion $W_S(t)$ and $W_\sigma(t)$ are correlated with constant correlation coefficient ρ (i.e. $W_S W_\sigma = \rho t$). The model parameters are $\alpha > 0$ (the volatility of the volatility), $0 \le \beta \le 1$ (the elasticity) and ρ (the correlation coefficient).

In the original SABR model paper, the authors provided a closed-form formula for the implied volatility under the SABR dynamics based on asymptotic expansion. After a correction made by Obloj [100], the expression of the so-called *Hagan formula* reads

$$\begin{aligned} \sigma_{imp}(K,S_0,T) = &\frac{1}{\left[1+\frac{(1-\beta)^2}{24}\ln^2\left(\frac{S_0}{K}\right)+\frac{(1-\beta)^4}{1920}\ln^4\left(\frac{S_0}{K}\right)+\cdots\right]}\cdot\left(\frac{\alpha\log\left(\frac{S_0}{K}\right)}{x(z)}\right)\cdot \\ &\left[1+\frac{(1-\beta)^2}{24}\frac{\sigma_0^2}{(KS_0)^{1-\beta}}+\frac{1}{4}\frac{\rho\beta\alpha\sigma_0}{(KS_0)^{(1-\beta)/2}}+\frac{2-3\rho^2}{24}\alpha^2\right]\cdot T+\cdots, \end{aligned}$$

with

$$z = \frac{\alpha\left(S_0^{1-\beta}-K^{1-\beta}\right)}{\sigma_0(1-\beta)}, \quad x(z) = \log\left(\frac{\sqrt{1-2\rho z+z^2}+z-\rho}{1-\rho}\right).$$

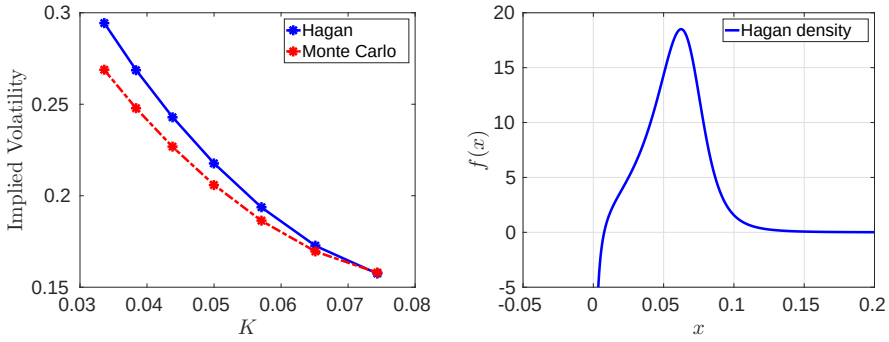
Some terms are very small and can be then neglected, resulting the following implied volatility formula,

$$\sigma_{imp}(K,S_0,T) = \frac{1}{\omega}\left(1+B_1\log\left(\frac{K}{S_0}\right)+B_2\ln^2\left(\frac{K}{S_0}\right)+B_2T\right),$$

where the term B_1, B_2, B_3 and ω are given by

$$\begin{aligned} B_1 &= -\frac{1}{2}\left(1-\beta-\rho\alpha\omega\right), \\ B_2 &= \frac{1}{12}\left((1-\beta)^2+3\left((1-\beta)-\rho\alpha\omega\right)+\left(2-3\rho^2\right)\alpha^2\omega^2\right), \\ B_3 &= \frac{(1-\beta)^2}{24}\frac{1}{\omega^2}+\frac{\beta\rho\alpha}{4}\frac{1}{\omega}+\frac{2-3\rho^2}{24}\alpha^2, \ \omega = \frac{S_0^{1-\beta}}{\sigma_0}. \end{aligned}$$

The Hagan formula is highly appreciated since it facilitates the model calibration, i.e., the determination of the model parameters such that the implied volatility given by the model to the market implied volatility.



(a) Hagan's implied volatility.

(b) Implied density.

Figure 1.1: SABR parameter configuration: $S_0 = 0.05$, $\sigma_0 = 0.05$, $\alpha = 0.4$, $\beta = 0.5$, $\rho = -0.7$ and $T = 7$,

However, the use of an asymptotic approximation makes the Hagan formula for the implied volatility not generally applicable. SABR model parameter configurations exist for which the formula gives inaccurate or even incorrect values. It is well-known that Hagan formula gives rise to problems when the asset quotes are close to zero, where a so-called absorption boundary value must be set. This issue can be seen by constructing the so-called *implied density*, i.e. the PDF associated to the implied volatility given by the Hagan formula. In Figure 1.1, an example of such an incorrect density is shown. In Figure 1.1(a) the implied volatilities given the Hagan formula and also numerically computed values by means of a Monte Carlo method are depicted for different strike prices K. We can see that the Hagan formula is not accurate for very low strikes. The problem become even more clear when representing the implied density, as in Figure 1.1(b), where the PDF exhibits negative values close to the boundary at zero. This cannot be, and implies arbitrage opportunities.

The valuation of European and exotic options under the SABR dynamics is generally not a trivial task. When reliable, the implied volatility given by the Hagan formula can be used in the *Black formula*[2] to obtain European option prices. However, a problem remains in the situations where the Hagan formula is not applicable. Then, Monte Carlo methods are sometimes employed even though an "exact Monte Carlo simulation" of the SABR model, which allows for simulation with only a few large time-steps, is not straightforward. The use of Fourier inversion techniques is limited since the ChF is not available. A PDE formulation for the option price under SABR dynamics has also been developed, enabling the use of PDE numerical techniques. The two-dimensional SABR PDE reads

$$\frac{1}{2}\sigma^2 \bar{S}^{2\beta} D^{2-2\beta} \frac{\partial^2 V}{\partial \bar{S}^2} + \rho\alpha \bar{S}^{\beta} D^{1-\beta} \sigma^2 \frac{\partial^2 V}{\partial \bar{S} \partial \sigma} + \frac{1}{2}\alpha^2 \sigma^2 \frac{\partial^2 V}{\partial \sigma^2} + \frac{\partial V}{\partial t} + r\bar{S}\frac{\partial V}{\partial \bar{S}} - rV = 0,$$

where $V(t, \bar{S}, \sigma)$ is the option price and $D(t, T) := \exp(-r(T-t))$.

[2] The Black-Scholes variant for underlying forwards.

SLV models in general and the SABR model in particular are very well-suited to model so-called *forward starting options*. Forward starting options can be seen as European options, but starting at a predefined time in the future (at $t > t_0$). In that case, the initial asset value is not known, but still the contract *premium* and hedge strategy must be set advance (at today's date). The SLV models are accurate for modelling the so-called *forward volatility*, which helps to determine accurate option prices and hedge strategies for forward starting options. In contrast, plain local volatility models are typically not able to accurately model the forward volatility, as volatility is flattening out in time under these models, and, thus, the volatility smile "flattens".

1.3. PRICING TECHNIQUES

Given a SDE governing the asset movements, the financial derivative price can be represented by the solution of a partial differential equation (PDE) via *Itô's lemma*. By the *payoff* function, a final condition can be prescribed. Due to the celebrated *Feynman-Kac theorem* (and its generalizations towards non-linear equations), the solution of many PDEs appearing in derivative valuation can be written in terms of a probabilistic representation by means of an expectation. The associated risk neutral asset price PDF plays an important role as it forms the basis of the computation of the expectation. In Theorem 1.3.1, the Feynman-Kac version for geometric Brownian motion and the Black-Scholes PDE is given.

Theorem 1.3.1 (Feynman-Kac)**.** *Given the money-savings account, modelled by* $\mathrm{d}B(t) = rB(t)\mathrm{d}t$*, with constant interest rate,* r*, let* $V(t,S)$ *be a sufficiently differentiable function of time* t *and stock price* $S(t)$*. Suppose that* $V(t,S)$ *satisfies the following PDE, with drift term,* $\mu(t,S)$*, and volatility term,* $\sigma(t,S)$*:*

$$\frac{\partial V}{\partial t} + \mu(t,S)\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2(t,S)\frac{\partial^2 V}{\partial S^2} - rV = 0, \tag{1.1}$$

with final condition $h(T,S)$*. The solution for* $V(t,S)$ *at time* $t_0 < T$ *is then given by:*

$$V(t_0,S) = \mathbb{E}^{\mathbb{Q}}\left[\mathrm{e}^{-r(T-t_0)}h(T,S)|\mathscr{F}(t_0)\right],$$

where the expectation is taken under measure $\mathbb{Q}$*, with respect to a process* S*, defined by:*

$$\mathrm{d}S(t) = \mu^{\mathbb{Q}}(t,S)\mathrm{d}t + \sigma(t,S)\mathrm{d}W^{\mathbb{Q}}(t), \quad \textit{for} \quad t > t_0.$$

The expectation in Theorem 1.3.1 can be written in integral form, resulting in the *risk-neutral valuation formula*,

$$V(t_0,S) = \mathrm{e}^{-r(T-t_0)}\int_{\mathbb{R}} h(T,y)f(y|\mathscr{F}(t_0))\mathrm{d}y, \tag{1.2}$$

with r the risk-free rate, T the maturity time, and $f(\cdot)$ the PDF of the underlying process. Many numerical techniques are based on the solution provided by the Feynman-Kac theorem and, particularly, by the risk-neutral valuation formula.

The Feynman-Kac formula can be generalized to more involved types of problems. Suppose, for example, a one-dimensional *semi-linear parabolic PDE* of the form

$$\begin{aligned} &\frac{\partial V}{\partial t} + \mu(t,S)\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2(t,S)\frac{\partial^2 V}{\partial S^2} - g\left(t,S,V,\sigma,\frac{\partial V}{\partial S}\right) = 0, \\ &V(T,S) = h(T,S), \end{aligned} \tag{1.3}$$

where g can be a function of V and its first derivative. This PDE also can also be expressed in a probabilistic representation by means of the following *forward* and *backward* SDEs,

$$\begin{aligned} \mathrm{d}S(t) &= \mu(t,S)\mathrm{d}t + \sigma(t,S)\mathrm{d}W(t), \quad S(0) = S_0 \\ -\mathrm{d}Y(t) &= g(t,S,Y,Z)\mathrm{d}t + Z\mathrm{d}W(t), \quad Y(T) = g(S(T)), \end{aligned}$$

with a final condition for the backward SDE. The solution consist of a pair of processes (Y,Z) as

$$Y(t) = V(t,S), \quad Z(t) = \sigma(t,S)\frac{\partial V}{\partial S}(t,S).$$

Many generalizations, to towards multi-dimensional linear PDEs, partial integro-differential equations (PIDEs), and also towards fully non-linear PDEs have been given in the literature.

1.3.1. MONTE CARLO METHODS

As the integral in Equation 1.2 is typically not solvable in analytic form, numerical-based approaches can be developed. Monte Carlo methods are well-known numerical techniques to evaluate integrals. They are based on the analogy between probability and volume. Suppose we need to compute an integral

$$I := \int_C f(x)\mathrm{d}x,$$

and we have a technique to draw n independent and identically distributed samples in C, $X_1, X_2, \ldots, X_n$. We then define a Monte Carlo estimator as

$$\bar{I}_n := \frac{1}{n}\sum_{j=1}^{n} f(X_j).$$

If f is integrable over C then, by the *strong law of the large numbers*,

$$\bar{I}_n \to I \quad \text{as} \quad n \to \infty,$$

with probability one.

Furthermore, if f is square integrable, we can define

$$s_f := \sqrt{\int_C \left(f(x) - I\right)^2 \mathrm{d}x},$$

the error of the Monte Carlo estimate $I - \bar{I}$ is, by means of the *central limit theorem*, which is assumed normally distributed with mean 0 and standard deviation $s_f/\sqrt{n}$.

The order of convergence of the plain Monte Carlo method is $\mathcal{O}(1/\sqrt{n})$, which is considered slow for many applications.

However, Monte Carlo methods are highly appreciated in computational finance due to their simplicity, flexibility and immediate implementation. One of the most important advantages is that the methods are easily extended to multi-dimensional problems, while keeping the order of convergence. Moreover, the method's simplicity allows to explore different convergence improvement techniques, like *variance reduction techniques* [59] or *multi-level Monte Carlo* acceleration [57].

In the context of the Monte Carlo approximation of an expectation, like in the solution of the PDE given by the Feynman-Kac theorem, an essential part of the method is to generate sample paths. *Random number generators* (RNG) form the basis of these Monte Carlo paths, and they have been studied for many years. Broadly, they can be subdivided into "true", pseudo- and quasi-random generators, and usually generate uniformly distributed samples. This is key, because when uniform samples between 0 and 1 are available, samples from any distribution can be obtained as long as the *quantile function*, i.e., the inverse of the *cumulative distribution function* (CDF), is known. The procedure is then as follows,

$$F_Z(Z) \overset{\mathrm{d}}{=} U \quad \text{thus} \quad Z_i = F_Z^{-1}(U_i),$$

where F_Z is the CDF, $\overset{\mathrm{d}}{=}$ means equality in the distributional sense, $U \sim \mathcal{U}([0,1])$ and U_i is a sample from $\mathcal{U}([0,1])$. The computational efficiency highly depends on the cost of calculating F_Z^{-1}. We will require this distribution inversion approach in the Chapters 2 and 3 to simulate the SABR model by means of exact Monte Carlo simulation.

When the "exact" sample generation is not possible, either because the distribution is not available or the inversion is computationally unaffordable, intermediate steps in the numerical scenario simulation can be introduced by means of a time discretization of the associated SDE. The Taylor-based discretization schemes are widely employed in quantitative finance. The most representative example is the *Euler-Maruyama method*, the generalization of the Euler method to SDEs available in the context of the Itô's calculus. Another well-known Taylor-based simulation scheme is *the Milstein method*, which presents a superior order of convergence to the Euler-Maruyama method.

The application of Monte Carlo methods to European-style options is rather straightforward and well-established. However, the use of Monte Carlo methods for other financial contracts is usually non-trivial. Some representative examples are the efficient Monte Carlo computation of the option sensitivities, the Greeks, [56] and the valuation of early-exercise option contracts [94].

1.3.2. FOURIER INVERSION METHODS

Within the context of the risk-neutral valuation formula, Fourier techniques can be applied, when the integral is solved in Fourier space. Fourier techniques rely on the availability of the ChF, i.e., the Fourier transform of the PDF associated to the asset stochastic

process. They form the so-called *Fourier pair*, as

$$\hat{f}(u) = \int_{\mathbb{R}} \exp(ixu) f(x) \mathrm{d}x,$$
$$f(x) = \frac{1}{2\pi} \int_{\mathbb{R}} \exp(-ixu) \hat{f}(u) \mathrm{d}u,$$

where $\hat{f}$ denotes a Fourier transformed function, i.e., here, the ChF.

Contrary to the PDF, the ChF can be derived for many models in finance, particularly those that are driven by Lévy processes. Therefore, the PDF can be recovered from the ChF via Fourier inversion. Thus, the PDF can be directly employed within the risk-neutral valuation formula in Equation (1.2). However, more efficient Fourier techniques are also available. The starting point is the representation of the unknown PDF in the form of an *orthonormal* series expansion on a finite interval $[a,b]$, $z \in [a,b]$, as

$$f(z) = \frac{1}{b-a} \left(1 + 2 \sum_{k=1}^{\infty} A_k \cdot \psi_k(z) \right),$$

where $\psi_k(\cdot)$ are orthonormal functions and the expansion coefficients A_k can be obtained by

$$A_k := < f, \psi_k > = \int_a^b f(z) \psi_k(z) \mathrm{d}z.$$

By using Parseval's identity, i.e., $< f, \psi_k > = < \hat{f}, \hat{\psi}_k >$, where $\hat{f}$ and $\hat{\psi}_k$ denote the Fourier transformations of f and ψ_k, respectively, the expansion coefficients can be computed based on the ChF.

Fourier inversion methods usually provide a high accuracy at limited computational cost. However, their applicability strongly relies on the availability of the ChF. In some cases when a ChF is not directly available, the ChF can be approximated, but this may hamper the efficiency of the methodology. The curse of dimensionality in the case of multi-dimensional problems will also be problematic when Fourier techniques are employed.

Several pricing techniques have been developed based on a Fourier approach, like the Carr and Madan method [19], Boyarchenko and Levendorskii [13], Lewis [92], Fang and Oosterlee [49] and Ortiz-Gracia and Oosterlee [102].

In this thesis, we mainly employ the COS method [49], which is based on a Fourier cosine series expansion of the PDF and the efficient computation of the expansion coefficients. An important advantage of this method is that closed-form solutions can be derived for many types of option contracts.

1.3.3. PDE METHODS

The financial derivatives valuation problem can be written in PDE form, as in Equation 1.1 for the case of the Black-Scholes model, or as in Equation 1.3. The main advantage of the PDE-based techniques is that the option Greeks can be obtained without substantial additional computational effort, and also exotic options can be cast relatively easily in the PDE framework. In general, the PDEs appearing in finance cannot be solved analytically and numerical methods should be applied. The computational cost and also the

reduced applicability in higher problem dimensions due to the curse of dimensionality are drawbacks to methodologies relying on the PDE approach.

Obtaining the PDE formulation of a general option pricing problem can be challenging depending on the assumed underlying model and the type of option itself. Another important application of PDEs is given by the so-called *Fokker-Planck equation*, also called *forward Kolmogorov equation*, whose solution is the PDF (and its evolution in time) associated to a particular SDE. When involved underlying processes (including jumps, for example) are employed, a generalized version of the Fokker-Planck equation in the form of a PIDE is also available.

Several extensions have been derived for the Black-Scholes equation, where some of the basic model assumptions have been relaxed to introduce, for example, transaction costs or the market's illiquidity. The latter extensions can be included by adding some non-linear terms in the PDE. Other well-known example of non-linear PDEs appearing in finance is given by the *Hamilton-Jacobi-Bellman equation*, which models, for example, optimal asset allocation of portfolio investment problems under constraints. Some recent representative works are [96] or [80].

Regarding numerical techniques for PDEs, the *finite difference* method is the most commonly employed approach. Several finite difference discretization schemes in both time and space have been studied and developed. In [66] the authors employed *alternating direction implicit* schemes for a hybrid asset price model with applications for European and also exotic options. In [125] a new PDE numerical scheme based on Crank-Nicolson and finite elements for interest rate derivatives was presented. To deal with non-linearity, Newton-like methods or penalty methods are state-of-the-art iteration techniques.

A summary of the numerical schemes commonly used in finance is provided in [44, 119, 133]. And there are very many other relevant works in the context of the efficient numerical solution of the PDEs appearing in finance.

1.3.4. HYBRID SOLUTION TECHNIQUES

In practical valuation and risk management applications, we need to deal with rather involved problems where computing the solution in an efficient way is a complex task. The techniques explained so far all have their pros and cons, depending on the problems at hand. Our goal in this thesis is to develop methods that keep the advantages and circumvent the drawbacks of the individual approaches (Fourier, Monte Carlo and PDE) for some specific valuation problems, leading to robust, fast and accurate solutions to these problems. To achieve this, a combination of different numerical approaches may be considered.

In the literature we can find many examples of hybrid solution methods, combining two of the techniques presented in the previous sections or employing advanced methods from other mathematical disciplines. Mentioning a few examples: Gobet et al. explore the use of *Least Squares Monte Carlo* methods on PDEs and on backward SDEs [62]. In [63] Grzelak and Oosterlee proposed a projection method to approximate a ChF of hybrid asset models. Dang et al. presented in [40] a dimension reduction Monte Carlo method, combining PDE methods with Monte Carlo simulation.

The techniques and implementations in this thesis are all Monte Carlo-based meth-

ods. However, as we are dealing with involved problems, the application of the basic version of the Monte Carlo method is not sufficient to provide a satisfactory balance between accuracy and efficiency. We therefore combine the Monte Carlo method components with other techniques like with copulas, stochastic collocation interpolation, Fourier inversion, dynamic programming recursion or regression techniques. An interesting aspect of Monte Carlo methods is the independency of the different computations, which makes the method highly parallelizable which facilitates the use of high-performance computing. In this thesis we have considered GPU computing as well.

1.3.5. DATA IN COMPUTATIONAL FINANCE

The amount of data generated in the financial and insurance sectors have been drastically increasing during the last decades. How to take advantage of *big financial data* is a great challenge, due to the "three V's", volume, variety and velocity.

A common application of financial data is the analysis of the retail business of banks and insurance companies. Some examples are client knowledge, fraud and anomaly detection, mortgage risk prediction or customer evaluation. As the data available is typically heterogeneous and unstructured, *machine learning* and the tools in classification, clustering and regression may be employed highly successfully.

Other interesting applications of big financial data are found in *algorithmic trading*, where investment decisions are carried out by algorithms and thus computers, based on the combination of mathematical models and big data analytics. This leads to a partly or fully automated process, where the profit and velocity are maximized and the human influence (manual errors, emotion, bias, etc.) is minimized. Nowadays, the most popular example is found in *high frequency trading*.

When the data available is structured, mathematical techniques can be explored, like methods for market prediction or risk assessment. However, data can be also interpreted as just samples from an unknown distribution. When the data available is a set of asset values, an approach would be to approximate either the PDF or the CDF solely based on the data, which can then be used within Equation (1.2).

In this thesis, we consider data-based density estimation for the risk neutral asset dynamics. By exploring the connection of the density estimation problem and cosine series expansions, we will develop a data-driven technique that generalizes the COS method.

1.4. GPU COMPUTING

Next to the precision in the financial results, it is also of importance, in many situations, that solutions are obtained in a very short time. Often the calculations appearing in finance can be further accelerated by means of advanced scientific computing tools. In last decades, high performance computing in general and GPU computing in particular have gained popularity in the quantitative finance context. Several large financial institutions have GPUs at their disposal.

The GPU computing, or more specifically, GPGPU (General-Purpose computing on Graphics Processing Units) is a relatively novel approach in the parallel computing. The GPU is used as a co-processor where computationally heavy tasks can be executed with high efficiency. Particular computing architectures with hundreds (and even thousands

in the latest generations) of small cores and a multi-level memory hierarchy give an extraordinary performance on embarrassingly parallel computations.

Different technologies and platforms have appeared in the framework of GPU computing. The Compute Unified Device Architecture (CUDA) [35] from NVIDIA is probably the best known and extended technology. Its main features are the GPU drivers, the programming languages (C, C++ and Fortran) and a complete and mature ecosystem including scientific libraries (cuBLAS, cuSOLVER, cuRAND, etc.), editors (Nsight) and debugging and analysis tools (CUDA-DBG, CUDA-MEMCHECK, Visual profiler, Tau). Furthermore, extensions of CUDA to use it in combination with other languages (Python, Java, .NET, etc.) and scientific platforms (Matlab, Mathematica) have been developed. OpenCL represents the non-proprietary alternative to CUDA, in which NVIDIA is also involved. OpenACC is a directive-based implementation tool that allows non programming experts to achieve significant acceleration of their scientific research with reduced programming effort.

GPU computing appears appropriate for financial calculations since these are typically *compute-bound* and not *memory-bound*. GPUs have an extraordinary computation power, but a limited amount of memory with respect to other high-performance systems (multi-core, multi-node, etc.). A particularly interesting application is formed by the Monte Carlo method. Due to characteristics as the availability of a huge number of independent scenarios, the Monte Carlo estimator fits perfectly in the GPU architecture philosophy. However, when hybrid solution methods are considered, like in this thesis, the corresponding parallelization, and therefore their implementation on GPUs may not be a trivial task. Some techniques are intrinsically sequential and must be adapted or reformulated to take advantage of the GPU architecture. In certain financial applications, the amount of memory is the limiting factor and an efficient management of the memory resources becomes essential to obtain a gain in terms of performance. The use of the highly optimized libraries available on GPUs is strongly recommended when possible.

During the period of the thesis work, we have had access to the Dutch national supercomputer Cartesius [20]. In terms of GPU computing the facilities form one of the most powerful architectures in the world. Thanks to that, we were able to take our implementations to the limit, because of the memory resources and the speed provided by the latest and most powerful generations of GPUs installed in the Cartesius system. For prototyping purposes, we also employed the *Little Green Machine*, an low-consuming and environmental-friendly system equipped with fast and compact GPUs.

1.5. MULTI-ITN-STRIKE PROJECT

The research described in this thesis has been partially carried out in the framework of the EU-funded Multi-ITN-STRIKE Marie Curie project *Novel Methods in Computational Finance*. The consortium was formed by several universities from around Europe: B.U. Wuppertal, C.U. Bratislava, U.P. Valencia, Rousse, I.S.E.G. Lisboa, U.A. Zittau, T.U. Wien, T.U. Delft, Greenwich, Würzburg and Antwerp as a participants and several other universities, research institutions and companies as project associate collaborators.

The goal of this international training network was to deepen the understanding of complex and advanced financial models and to develop effective and robust numerical schemes for linear and non-linear problems arising from the mathematical theory of

pricing financial derivatives and related financial products. This has been accomplished by means of financial modelling, mathematical analysis and numerical simulation, optimal control techniques and validation of models. Twelve *early-stage researchers* (PhD students) and five *experienced researchers* (post-doctoral students) have successfully followed the high level education provided within the project, building up strong collaborations with the network partners.

In the problems studied, the challenge has been expressed by the non-linearity in the financial problem formulation. On the one hand, non-linear generalizations of the Black-Scholes equation (Theorem 1.3.1) have been introduced by considering non-linear payoffs and market frictions. On the other hand, more complicated asset models have been studied in ITN-STRIKE, like the already mentioned SV or SLV models.

The modelling research in the ITN-STRIKE project included transaction costs into the Black-Scholes model, Lévy models [16], Lie Group approaches [11], optimal control and Fokker-Planck equations [54]. The numerical methods investigated in ITN-STRIKE included finite difference schemes for PDEs [28, 46, 71, 73] and also hybrid Monte Carlo methods [88, 89].

Besides the research performed in the project, many scientific events have taken place, providing a complete educational training program which ranged from advanced mathematical concepts (Newton methods, optimal control, Lie Group methods, etc.) to transferable skills (project management, effective writing, cultural awareness, plagiarism) and to computational skills (high-performance computing, GPU programming). Furthermore, the project has provided many networking opportunities in both academic (ECMI 2014, SCF2015, ICCF 2015, ALGORITMY 2016) and industrial (Postbank, MathFinance) environments.

1.6. OUTLINE OF THE THESIS

In this thesis, we develop hybrid solution methods for involved problems in option pricing and risk management.

In Chapter 2, a one time-step Monte Carlo method for the exact simulation of the SABR model is introduced. It relies on an accurate approximation of the conditional distribution of the time-integrated variance, for which Fourier inversion and copulas are combined. Resulting is a fast simulation algorithm for the SABR model. As a one time-step method, it is suitable for European options which can be employed for calibration purposes. We numerically show that our technique overcomes some known issues of the SABR approximated asymptotic formula.

We extend the method presented in Chapter 2 to the multiple time-step case in Chapter 3. To reduce the method's complexity, new components need to be added, like an approximation of the sum of log-normals, stochastic collocation interpolation and a correlation approximation. The resulting method, which we call the *mSABR* method, is highly efficient and competitive, as it requires only a few time-steps to achieve high accuracy. The method is particularly interesting for long-term and exotic options. Furthermore, the current market situation of negative interest rates is also successfully addressed by the mSABR scheme.

In Chapter 4, a data-driven generalization of the well-known COS method is presented, which we call the ddCOS method. The ddCOS method relies on the assumption

that only asset data samples are available and the ChF is not available and no longer required. Whereas the order of convergence provided by the method is according the convergence of Monte Carlo methods, the method is beneficial for the accurate and efficient computation of some option sensitivities. Therefore, the method appears suitable for risk management. We use the ddCOS method for the Greeks computation with the SABR simulation scheme presented in Chapters 2 and 3 in the context of the Delta-Gamma approach.

In Chapter 5, we develop a parallel version of the Stochastic Grid Bundling Method (SGBM), a method to price multi-dimensional early-exercise option contracts. The parallel implementation follows the GPGPU paradigm. A new grid point bundling scheme is proposed, which is highly suitable for parallel systems, increasing the efficiency in the use of memory and workload distribution. This parallel version of SGBM generalizes its applicability, allowing us to solve very high-dimensional problems (up to fifty dimensions). Thanks to the performance of the parallel SGBM, a generally applicable way of computing the early-exercise policy is proposed.

In Chapter 6, the conclusions of the presented work are summarized. We also include an outlook for possible future research lines.

CHAPTER 2

One time-step Monte Carlo simulation of the SABR model

In this chapter, we propose a one time-step Monte Carlo method for the SABR model. We base our approach on an accurate approximation of the CDF for the time-integrated variance (conditional on the SABR volatility), using Fourier techniques and a copula. Resulting is a fast simulation algorithm which can be employed to price European options under the SABR dynamics. Our approach can thus be seen as an alternative to Hagan analytic formula for short maturities that may be employed for model calibration purposes.

2.1. INTRODUCTION

The SABR model [67] is an established SDE system which is often used for interest rates and FX modelling in practice. The model belongs to the SLV models. The idea behind SLV models is that the modelling of volatility is partly done by a local volatility and partly by a stochastic volatility contribution, aiming to preserve the advantages and minimize the disadvantages of the individual models.

In the original paper [67], the authors have provided a closed-form approximation formula for the implied volatility under SABR dynamics. This is important for practitioners, as it can be used highly efficiently within the calibration exercise. However, the closed-form expression is derived by perturbation theory and its applicability is thus not general. The formula is for example not always accurate for small strike values or for long time to maturity options. In [100], Oblój provided a correction to the Hagan formula but the same drawbacks remained. In order to improve the approximation for the SABR implied volatility, different techniques have been proposed in the literature, based on either perturbation theory [69, 136], heat kernel expansions [72, 104] or model map techniques [4].

The objective of a bank is typically to employ the SABR model for pricing exotic derivatives. Before this pricing can take place, the model needs to be calibrated to European options, and, if possible, to some exotic options as well. It is market practice to use Hagan formula to calibrate the SABR model, however, since it is only an approximation it will not always result in a sufficient fit to the market. In other words, a SABR Monte Carlo simulation may give different implied volatilities than the Hagan formula.

In this chapter, we aim to develop a *one time-step* Monte Carlo method, which is accurate for European derivative contracts up to two years. This kind of contract is often traded in FX markets. In particular, we focus on the efficient simulation of SABR's time-integrated variance, conditional on the volatility dynamics. An analytical expression of

This chapter is based on the article "On a one time-step Monte Carlo simulation approach of the SABR model: Application to European options". Published in *Applied Mathematics and Computation*, 293:461–479, 2017 [88].

the conditional distribution of the time-integrated variance is available in differential form [107]. However, this solution is not tractable in our context, due to the instabilities (related to the Hartman-Watson distribution) appearing when numerical integration techniques are applied [74]. Furthermore, the use of numerical methods also implies that sampling the distribution becomes computationally unaffordable. Here, we propose to approximate the distribution by the two marginal distributions involved, i.e. the volatility and time-integrated variance distributions, by using a copula. The copula methodology has been used intensively in the financial world, see, for example, [26], in risk management and option pricing. In the field of risk management, some relevant works are Li [93], Embrechts et al [47], Cherubini and Luciano [24] or Rosenberg and Schuermann [110]. In the case of pricing multi-asset options, some examples of the use of copulas are Cherubini and Luciano [25] or Rosenberg [109].

The approximation of the CDF of the time-integrated variance is obtained by applying a recursive procedure, as described in [137], which was originally developed to price arithmetic Asian options. The derivation is based on the definition of the ChF of the time-integrated variance and the use of Fourier inversion techniques. We adapt the procedure to our problem and improve it in terms of computational costs. Once both marginal distributions are available, we employ the copula technique to get a multivariate distribution which approximates the conditional distribution of the time-integrated variance given the volatility. Several copulas are considered and an analysis of their performance is carried out. The conditional time-integrated variance approximation will be employed in the asset simulation for the complete SABR model.

The chapter is organized as follows. In Section 2.2, the SABR model and the simulation steps are briefly introduced. Section 2.3 describes the procedure to derive the CDF of SABR's time-integrated variance. The copula approach to approximate the conditional time-integrated variance distribution is explained in Section 2.4. In Section 2.5, some numerical experiments are presented. We conclude in Section 2.6.

The one-step methodology restricts the use of the proposed method to option maturities up to two years and European-type options. In the next chapter, we will generalize the methodology to the *multiple time-step* case.

2.2. THE SABR MODEL

The SABR model is based on a parametric local volatility component in terms of a model parameter, β. The formal definition of the SABR model reads

$$\begin{aligned} \mathrm{d}S(t) &= \sigma(t)S^{\beta}(t)\mathrm{d}W_S(t), \quad & S(0) &= S_0\exp(rT), \\ \mathrm{d}\sigma(t) &= \alpha\sigma(t)\mathrm{d}W_\sigma(t), & \sigma(0) &= \sigma_0, \end{aligned} \tag{2.1}$$

where $S(t) = \bar{S}(t)\exp(r(T-t))$ denotes the forward value of the underlying asset $\bar{S}(t)$, with r the interest rate, S_0 the spot price and T the contract's final time. Quantity $\sigma(t)$ denotes the stochastic volatility, with $\sigma(0) = \sigma_0$, $W_S(t)$ and $W_\sigma(t)$ are two correlated Brownian motions with constant correlation coefficient ρ (i.e. $W_S W_\sigma = \rho t$). The open parameters of the model are $\alpha > 0$ (the volatility of the volatility), $0 \leq \beta \leq 1$ (the elasticity) and ρ (the correlation coefficient).

A useful aspect for practitioners is that each model parameter can be assigned to a

specific feature of the market observed implied volatility when it is represented against the strike prices, commonly known as a *volatility smile or skew*. The α parameter mainly affects the smile curvature. Exponent β is connected to the forward process distribution, being normal for $\beta = 0$ and log-normal for $\beta = 1$. In terms of the volatility smile, β also affects the curvature. Correlation parameter ρ rotates the implied volatility curve around the *at-the-money* point, obtaining "more pronounced smile" or "more pronounced skew" shapes.

2.2.1. SABR MONTE CARLO SIMULATION

In the first equation of the SABR model (2.1), the forward asset dynamics are defined as a CEV process [31]. Based on the works of Schroder [117] and Islah [75], an analytic expression for the CDF of the SABR conditional process has been obtained. For some $S(0) > 0$, the conditional CDF of $S(t)$ with an absorbing boundary at $S(t) = 0$, and given the volatility, $\sigma(t)$, and the conditional time-integrated variance, $\int_0^t \sigma^2(s)\mathrm{d}s|\sigma(t)$, reads

$$Pr\left(S(t) \leq K|S(0) > 0, \sigma(t), \int_0^t \sigma^2(s)\mathrm{d}s\right) = 1 - \chi^2(a; b, c), \tag{2.2}$$

where

$$a = \frac{1}{\nu(t)}\left(\frac{S(0)^{1-\beta}}{(1-\beta)} + \frac{\rho}{\alpha}(\sigma(t) - \sigma(0))\right)^2, \qquad c = \frac{K^{2(1-\beta)}}{(1-\beta)^2\nu(t)},$$
$$b = 2 - \frac{1 - 2\beta - \rho^2(1-\beta)}{(1-\beta)(1-\rho^2)}, \qquad \nu(t) = (1-\rho^2)\int_0^t \sigma^2(s)\mathrm{d}s,$$

and $\chi^2(x; \delta, \lambda)$ is the non-central chi-square CDF.

It should be noted that this formula is *exact* for the case $\rho = 0$ and results in an *approximation* otherwise. A shifted process with an approximated initial condition is employed in the derivation for $\rho \neq 0$. Based on Equation (2.2), an "exact" Monte Carlo simulation scheme for the SABR model can be defined based on inverting the conditional SABR CDF, when the conditional time-integrated variance and the volatility are already simulated. This forms the basis of the one time-step SABR approach, where exact simulation is required for accuracy reasons when using only one large time-step (meaning no time discretization). Furthermore, it is well known that the convergence ratio of the Monte Carlo method is $1/\sqrt{n}$, with n the number of paths or scenarios.

According to Equation (2.2), in order to apply an "exact" one time-step Monte Carlo simulation for the SABR dynamics, we need to perform the following steps (with the terminal time T):

- *Simulation of SABR's volatility.* From Equation (2.1), we observe that the volatility process of the SABR model is governed by a log-normal distribution. As is well-known, the solution follows a GBM, i.e. the exact simulation of the volatility at terminal time, $\sigma(T)$, reads

$$\sigma(T) \sim \sigma(0)\exp(\alpha\tilde{W}_\sigma(T) - \frac{1}{2}\alpha^2 T), \tag{2.3}$$

  where $\tilde{W}_\sigma(T)$ is an independent Brownian motion.

- *Simulation of SABR's time-integrated variance, conditional on the terminal value of the volatility, i.e.,* $\int_0^T \sigma^2(s)\mathrm{d}s|\sigma(T)$. Exact simulation of the time-integrated variance in the SABR model is not practically possible since the expression of the conditional distribution given the volatility cannot be efficiently treated. One possibility is to simulate it by a Monte Carlo method, but this implies the use of a nested simulation which can be very expensive or even unaffordable. Another possibility is to find an approximation for the conditional distribution of SABR's time-integrated variance given $\sigma(T)$. The integral $\int_0^T \sigma^2(s)\mathrm{d}s$ can also be approximated by some quadrature rule once $\sigma(0)$ and $\sigma(T)$ (or intermediate values of the volatility process) are available.

- *Simulation of SABR's forward asset process.* We can simulate the forward dynamics by inverting the CDF in Equation (2.2). For this, the conditional SABR's time-integrated variance (besides the volatility) is required. Concerning the forward asset simulation, there is no analytic expression for the inverse SABR distribution so the inversion has to be calculated by means of some numerical approximation. Another possibility is to employ SDE discretization schemes, like Taylor-based schemes (Euler-Maruyama, log-Euler, Milstein or full-truncation) or more involved ones (low-bias or QE schemes). However, the use of such schemes gives rise to several issues that have to be managed, like the discretization error, negativity in the asset values and computational cost.

In Algorithm 1, a schematic representation of the complete one time-step Monte Carlo simulation procedure for the SABR model is presented. We include references to the sections of this chapter where the respective parts of the SABR simulation are discussed.

Algorithm 1: SABR's one time-step Monte Carlo simulation.

Data: $S_0, \sigma_0, \alpha, \beta, \rho, T, n$.
for *Path* $p = 1 \dots n$ **do**
  Simulation of $\tilde{W}_\sigma(t) \sim \mathcal{N}(0, \sqrt{T})$.
  Simulation of $\sigma(T)|\sigma(0)$ (see Equation (2.3)).
  Simulation of $\int_0^T \sigma^2(s)\mathrm{d}s|\sigma(T)$ (see Subsection 2.4.3).
  Simulation of $S(T)|S(0), \sigma(T), \int_0^T \sigma^2(s)\mathrm{d}s$ (see Subsection 2.4.4).

In the above steps of the SABR model exact simulation, the challenging part is the simulation of the time-integrated variance, $\int_0^T \sigma^2(s)\mathrm{d}s$, conditional on $\sigma(T)$ (and $\sigma(0)$). Sometimes moment-matching techniques can be used. Here, however, we propose a computationally efficient approximation, based on a copula multi-variate distribution, to simulate the conditional distribution of $\int_0^T \sigma^2(s)\mathrm{d}s$ given the volatility $\sigma(T)$. Simulation by means of a copula technique requires the CDF of the involved marginal distributions. Although the distribution of $\sigma(T)$ is known, the distribution of $\int_0^T \sigma^2(s)\mathrm{d}s$ needs to be approximated. We propose a method based on a Fourier technique, which was already employed for Asian options in [137]. In Section 2.3, this derivation is presented in detail.

Hereafter, for notational convenience, we will use $Y(T) := \int_0^T \sigma^2(s)\mathrm{d}s$.

2.3. CDF OF SABR'S TIME-INTEGRATED VARIANCE

In this section, we present a procedure to approximate the CDF of the time-integrated variance, $Y(T)$. Since we will work in the log-space, an approximation of the CDF of $\log Y(T)$, $F_{\log Y(T)}$, will be derived. First of all, we approximate $Y(T)$ by its discrete analog, i.e.

$$\int_0^T \sigma^2(s)\mathrm{d}s \approx \sum_{j=1}^{M} \sigma^2(t_j)\Delta t =: \tilde{Y}(T), \tag{2.4}$$

where M is the number of discrete time points[1], i.e., $t_j = j\Delta t$, $j = 1,\ldots,M$ and $\Delta t = \frac{T}{M}$. $\tilde{Y}(T)$ is subsequently transformed to the logarithmic domain, where we aim to find an approximation of $F_{\log \tilde{Y}(T)}$, i.e.

$$F_{\log \tilde{Y}(T)}(x) = \int_{-\infty}^{x} f_{\log \tilde{Y}(T)}(y)\mathrm{d}y, \tag{2.5}$$

with $f_{\log \tilde{Y}(T)}$ the PDF of $\log \tilde{Y}(T)$. $f_{\log \tilde{Y}(T)}$ is, in turn, found by approximating the associated ChF, $\hat{f}_{\log \tilde{Y}(T)}$, and applying a Fourier inversion procedure. The ChF and the desired PDF of $\log \tilde{Y}(T)$ form a so-called Fourier pair. Based on the work in [8] and [137], we develop a recursive procedure to recover the ChF of $f_{\log \tilde{Y}(T)}$. We start by defining the sequence,

$$R_j = \log\left(\frac{\sigma^2(t_j)}{\sigma^2(t_{j-1})}\right) = \log\left(\sigma^2(t_j)\right) - \log\left(\sigma^2(t_{j-1})\right), \tag{2.6}$$

where R_j is the logarithmic increment of $\sigma^2(t)$ between t_j and t_{j-1}, $j = 1,\ldots,M$. As mentioned, in the SABR model context the volatility process follows log-normal dynamics. Since increments of Brownian motion are independent and identically distributed, the R_j are also independent and identically distributed, i.e. $R_j \overset{d}{=} R$. In addition, the ChF of R_j, as a normal increment, is well known and reads, $\forall u, j$,

$$\hat{f}_{R_j}(u) = \hat{f}_R(u) = \exp(-iu\alpha^2\Delta t - 2u^2\alpha^2\Delta t), \tag{2.7}$$

with α as in (2.1) and i the imaginary unit.

By Equation (2.6), we can write $\sigma^2(t_j)$ as

$$\sigma^2(t_j) = \sigma^2(t_0)\exp(R_1 + R_2 + \cdots + R_j), \tag{2.8}$$

because

$$\begin{aligned}\sigma^2(t_j) &= \sigma^2(t_0)\exp\left(\log\left(\frac{\sigma^2(t_1)}{\sigma^2(t_0)}\right) + \log\left(\frac{\sigma^2(t_2)}{\sigma^2(t_1)}\right) + \cdots + \log\left(\frac{\sigma^2(t_j)}{\sigma^2(t_{j-1})}\right)\right)\\ &= \sigma^2(t_0)\exp\left(\log\left(\frac{\sigma^2(t_j)}{\sigma^2(t_0)}\right)\right).\end{aligned}$$

[1] These time points are not to be confused with the Monte Carlo time-steps. We will have only one Monte Carlo time-step. M is the number of points for the discrete approximation of $Y(T)$.

At this point, a backward recursion procedure in terms of R_j will be set up by which we will recover $\hat{f}_{\log \tilde{Y}(T)}$. We define

$$\begin{aligned} Y_1 &= R_M, \\ Y_j &= R_{M+1-j} + Z_{j-1}, \quad j = 2, \ldots, M, \end{aligned} \tag{2.9}$$

with $Z_j = \log(1 + \exp(Y_j))$.

Using the definitions in Equations (2.8) and (2.9), the approximated time-integrated variance can be written as

$$\tilde{Y}(T) = \sum_{i=1}^{M} \sigma^2(t_i)\Delta t = \Delta t \sigma_0^2 \exp(Y_M). \tag{2.10}$$

From Equation (2.10), we determine $\hat{f}_{\log \tilde{Y}(T)}$, as follows

$$\begin{aligned} \hat{f}_{\log \tilde{Y}(T)}(u) &= \mathbb{E}[\exp(iu \log \tilde{Y}(T))] \\ &= \mathbb{E}[\exp\left(iu \log\left(\Delta t \sigma_0^2\right) + iuY_M\right)] \\ &= \exp\left(iu \log\left(\Delta t \sigma_0^2\right)\right) \hat{f}_{Y_M}(u). \end{aligned} \tag{2.11}$$

We have reduced the computation of $\hat{f}_{\log \tilde{Y}(T)}$ to the computation of $\hat{f}_{Y_M}$. As Y_M is defined recursively, its ChF, $\hat{f}_{Y_M}$, can be obtained recursively as well. In Subsection 2.3.1, a procedure for obtaining an approximation for $\hat{f}_{Y_M}$, i.e. $\hat{f}^*_{Y_M}$, is described. According to the definition of the (backward) sequence Y_j in Equation (2.9), the required initial and recursive ChFs are given by the following expressions,

$$\begin{aligned} \hat{f}_{Y_1}(u) &= \hat{f}_{R_M}(u) = \exp(-iu\alpha^2\Delta t - 2u^2\alpha^2\Delta t), \\ \hat{f}_{Y_j}(u) &= \hat{f}_{R_{M+1-j}}(u)\hat{f}_{Z_{j-1}}(u) = \hat{f}_R(u)\hat{f}_{Z_{j-1}}(u), \end{aligned} \tag{2.12}$$

where Equation (2.7) is used in both expressions and the independence of R_{M+1-j} and Z_{j-1} is also employed.

After the computation of $\hat{f}_{Y_M}$, and thus of $\hat{f}_{\log \tilde{Y}(T)}$, the next step is to compute the PDF of $\log \tilde{Y}(T)$ by means of the COS method [49]. For that, we need to determine the support of $\log \tilde{Y}(T)$, which is denoted by interval $[\tilde{a}, \tilde{b}]$. It can be calculated by employing the relation

$$\log(\tilde{Y}(T)) = \log\left(\Delta t \sigma_0^2 \exp(Y_M)\right) = \log\left(\Delta t \sigma_0^2\right) + Y_M.$$

An alternative to the COS method is the so-called SWIFT method [102], based on Shannon wavelets. This technique is efficient especially when long time horizons are considered. For the present application, however, the COS method appears superior in terms of computational time. The use of the one-step SABR simulation will be restricted to exercise times up to two years (details in Subsection 2.4.2).

We recover $f_{\log \tilde{Y}(T)}$ by employing the COS expression, as follows

$$f_{\log \tilde{Y}(T)}(x) \approx \frac{2}{\tilde{b} - \tilde{a}} \sum_{k=0}^{N-1}{}' C_k \cos\left((x - \tilde{a}) \frac{k\pi}{\tilde{b} - \tilde{a}}\right), \tag{2.13}$$

with

$$C_k = \Re\left(\hat{f}_{\log\tilde{Y}(T)}\left(\frac{k\pi}{\tilde{b}-\tilde{a}}\right)\exp\left(-i\frac{\tilde{a}k\pi}{\tilde{b}-\tilde{a}}\right)\right),$$

and

$$\begin{aligned}\hat{f}_{\log\tilde{Y}(T)}\left(\frac{k\pi}{\tilde{b}-\tilde{a}}\right) &= \exp\left(i\frac{k\pi}{\tilde{b}-\tilde{a}}\log\left(\Delta t\sigma_0^2\right)\right)\hat{f}_{Y_M}\left(\frac{k\pi}{\tilde{b}-\tilde{a}}\right)\\ &\approx \exp\left(i\frac{k\pi}{\tilde{b}-\tilde{a}}\log\left(\Delta t\sigma_0^2\right)\right)\hat{f}^*_{Y_M}\left(\frac{k\pi}{\tilde{b}-\tilde{a}}\right),\end{aligned}$$

where N is the number of COS terms and the prime $'$ and $\Re$ symbols in (2.13) indicate division of the first term in the summation by two and taking the real part of the complex-valued expressions in the brackets, respectively.

The CDF $F_{\log\tilde{Y}(T)}$, as in Equation (2.5), is obtained by integrating the corresponding approximated PDF from Equation (2.13), as follows

$$\begin{aligned}F_{\log\tilde{Y}(T)}(x) &= \int_{-\infty}^{x} f_{\log\tilde{Y}(T)}(y)\mathrm{d}y\\ &\approx \int_{\tilde{a}}^{x}\frac{2}{\tilde{b}-\tilde{a}}\sum_{k=0}^{N-1}{}'C_k\cos\left((y-\tilde{a})\frac{k\pi}{\tilde{b}-\tilde{a}}\right)\mathrm{d}y.\end{aligned} \tag{2.14}$$

2.3.1. ChF of Y_M

In the previous section, we defined a procedure to compute an approximation of the CDF of $Y(T)$. This computation relies on the efficient calculation of the ChF of Y_M. In this section, we explain the recursive procedure for approximating $\hat{f}_{Y_M}$ based on the initial ChF, $\hat{f}_{Y_1}$, the ChF of the recursive process, $\hat{f}_R$, and the equality $\hat{f}_{Y_j}(u) = \hat{f}_R(u)\hat{f}_{Z_{j-1}}(u)$ (see Equations (2.6) and (2.12)). Note that this iterative computation needs to be done only once. By definition, the ChF of Z_{j-1} reads

$$\hat{f}_{Z_{j-1}}(u) := \int_{-\infty}^{\infty}(\exp(x)+1)^{iu}f_{Y_{j-1}}(x)\mathrm{d}x.$$

PDF $f_{Y_{j-1}}$ is not known. To approximate it, the Fourier cosine series expansion on $f_{Y_{j-1}}$ is applied. We suppose a bounded the integration range $[a,b]$. The calculation of integration boundaries a and b follows the cumulant-based approach as described in [137] and [49]. After truncation of the integral, we have

$$\hat{f}^*_{Z_{j-1}}(u) = \int_a^b(\exp(x)+1)^{iu}f_{Y_{j-1}}(x)\mathrm{d}x. \tag{2.15}$$

Now we can apply a cosine series expansion to $f_{Y_{j-1}}$, so that $\hat{f}^*_{Z_{j-1}}$ becomes

$$\hat{f}^*_{Z_{j-1}}(u) \approx \frac{2}{b-a}\sum_{l=0}^{N-1}{}'B_l\int_a^b(\exp(x)+1)^{iu}\cos\left((x-a)\frac{l\pi}{b-a}\right)\mathrm{d}x, \tag{2.16}$$

with

$$B_l = \Re\left(\hat{f}^*_{Y_{j-1}}\left(\frac{l\pi}{b-a}\right)\exp\left(-ia\frac{l\pi}{b-a}\right)\right),$$

where N is the number of expansion terms, $\hat{f}^*_{Y_1}(u) = \hat{f}_R(u)$ and $\hat{f}^*_{Y_j}(u) = \hat{f}_R(u)\hat{f}^*_{Z_{j-1}}(u)$.

We wish to compute $\hat{f}^*_{Y_M}(\frac{k\pi}{b-a})$, for $k = 0,\ldots,N-1$. Considering Equation (2.16), we rewrite $\hat{f}^*_{Y_j}(u) = \hat{f}_R(u)\hat{f}^*_{Z_{j-1}}(u)$ in matrix-vector form, as follows

$$\Phi_{j-1} = \mathcal{M} A_{j-1}, \tag{2.17}$$

where

$$\begin{aligned}
&\Phi_{j-1} = \left[\Phi_{j-1}(k)\right]_{k=0}^{N-1}, \quad \Phi_{j-1}(k) = \hat{f}^*_{Z_{j-1}}(u_k), \\
&\mathcal{M} = \left[\mathcal{M}(k,l)\right]_{k=0}^{N-1}, \quad \mathcal{M}(k,l) = \int_a^b (\exp(x)+1)^{iu_k}\cos\left((x-a)u_l\right)\mathrm{d}x, \\
&A_j = \frac{2}{b-a}\left[A_j(l)\right]_{l=0}^{N-1}, \quad A_j(l) = \Re\left(\hat{f}^*_{Y_{j-1}}(u_l)\exp\left(-iau_l\right)\right),
\end{aligned} \tag{2.18}$$

with

$$u_z = \frac{z\pi}{b-a}. \tag{2.19}$$

By the recursion procedure in Equation (2.17), we obtain the approximation $\hat{f}^*_{Y_M}$. Before using this approximation, we must compute matrix $\mathcal{M}$ in Equation (2.18) highly efficiently. This matrix $\mathcal{M}$ has to be computed only once. Its elements are given by

$$I_{\mathcal{M}} := \int_a^b (\exp(x)+1)^{iu_k}\cos\left((x-a)u_l\right)\mathrm{d}x. \tag{2.20}$$

The efficient computation of the integral in Equation (2.20) is a key aspect for the performance of the whole procedure. The number of elements in matrix $\mathcal{M}$ can be large, as it corresponds to the number of elements in the cosine series expansion i.e. N^2. The efficiency also depends on the required accuracy. We discuss the numerical treatment in Subsection 2.3.2.

2.3.2. TREATMENT OF INTEGRAL $I_{\mathcal{M}}$

The efficient calculation of the integral in Equation (2.20) is crucial for an efficient computation of $\hat{f}^*_{Y_M}$ (and thus of $F_{\log\tilde{Y}(T)}$). In this section, we propose several ways to deal with $I_{\mathcal{M}}$. In Subsection 2.3.3, an analysis in terms of accuracy and efficiency for each of the approximations is carried out.

Analytic solution based on ${}_2F_1$. Integral $I_{\mathcal{M}}$ in (2.20) can be rewritten as

$$I_{\mathcal{M}} = \int_a^b \mathrm{e}^{iu_k g(x)}\cos\left((x-a)u_l\right)\mathrm{d}x, \quad g(x) = \log((\exp(x)+1)), \tag{2.21}$$

with u_k and u_l as in (2.19).

By the Euler formula, integral $I_{\mathcal{M}}$ becomes

$$I_{\mathcal{M}} = \frac{1}{2}\left(I^+ + I^-\right), \quad I^{\pm} = \int_a^b \exp(i(u_k g(x) \pm (x-a)u_l))\mathrm{d}x.$$

Integrals I^+ and I^- can be solved analytically in terms of the hypergeometric function ${}_2F_1$. The solution reads

$$I^{\pm} = \mp \frac{i e^{\pm i u_l (x-a)}}{u_l} \, {}_2F_1(-i u_k, \pm i u_l; 1 \pm i u_l; -\exp(x)).$$

Unfortunately, multiple evaluations of the hypergeometric function make the computation rather time-consuming.

Clenshaw-Curtis quadrature. Regarding numerical approximation methods, quadrature rules can be applied to solve integral $I_{\mathcal{M}}$. In our case, the elements in Equation (2.20) can, for example, be approximated numerically by the Clenshaw-Curtis quadrature, i.e.,

$$\int_a^b (\exp(x)+1)^{i u_k} \cos((x-a) u_l) \, \mathrm{d}x \approx (\mathscr{D}^T v)^T w, \tag{2.22}$$

where the matrix element $\mathscr{D}(k,m)$ and the vectors v and w can be computed by the expressions

$$\begin{aligned}
\mathscr{D}(k,m) &= \frac{2}{n_q} \cos\left(\frac{(m-1)(k-1)\pi}{n_q/2}\right) \cdot \begin{cases} 1/2 & \text{if } m = \{1, n_q/2+1\}, \\ 1 & \text{otherwise,} \end{cases} \\
v &:= \left(1, \frac{2}{1-4}, \frac{2}{1-16}, \cdots, \frac{2}{1-(n_q-2)^2}, \frac{2}{1-n_q^2}\right)^T, \\
w_m &:= h\left(\cos\left(\frac{(m-1)\pi}{n_q}\right)\right) + h\left(-\cos\left(\frac{(m-1)\pi}{n_q}\right)\right), \\
h(x) &= \frac{b-a}{2} \cos\left(\left(\frac{b-a}{2}x + \frac{a+b}{2} - a\right) u_l\right) \left(\exp\left(\frac{b-a}{2}x + \frac{a+b}{2}\right) + 1\right)^{i u_k},
\end{aligned}$$

with $k, m = 1, \ldots, (n_q/2+1)$ and with n_q the number of terms in the quadrature. This was the approach taken in [137], with computational complexity $O(n_q N^2)$.

Piecewise linear approximation. In a second numerical approximation, we start with $I_{\mathcal{M}}$ in (2.21) and notice that although function $g(x)$ (the blue curve in Figure 2.1) is not linear, it is smooth and monotonic. Hence, the numerical technique proposed is to define sub-intervals of the integration range $[a,b]$ in which $g(x)$ is constant or almost linear, i.e.,

$$I_{\mathcal{M}} = \sum_{j=1}^{L} \int_{a_j}^{b_j} \exp(i u_k g(x)) \cos((x-a) u_l) \, \mathrm{d}x,$$

so that we can perform a first-order expansion of function $g(x)$ in each sub-interval $[a_j, b_j]$, $j = 1, \ldots, L$. In order to guarantee continuity of the approximation, $g(x)$ is approximated by a linear function in each interval as

$$g(x) \approx c_{1,j} + c_{2,j} x, \quad x \in [a_j, b_j].$$

This gives us an approximation $\tilde{I}_{\mathcal{M}}$, as follows

$$\tilde{I}_{\mathcal{M}} = \sum_{j=1}^{L} \exp(i u_k c_{1,j}) \underbrace{\int_{a_j}^{b_j} \exp(i u_k c_{2,j} x) \cos((x-a) u_l) \, \mathrm{d}x}_{I_L}. \tag{2.23}$$

In each sub-interval $[a_j, b_j]$, we need to determine a simple integral. I_L in Equation (2.23) is known analytically and is given by

$$\begin{aligned} I_L = & \frac{\exp(i a_j c_{2,j} u_k) \left(-i c_{2,j} u_k \cos((a-a_j) u_l) + u_l \sin((a-a_j) u_l)\right)}{(u_l - c_{2,j} u_k)(u_l + c_{2,j} u_k)} \\ & + \frac{i \exp(i b_j c_{2,j} u_k) \left(c_{2,j} u_k \cos((a-b_j) u_l) + i u_l \sin((a-b_j) u_l)\right)}{(u_l - c_{2,j} u_k)(u_l + c_{2,j} u_k)}. \end{aligned}$$

The optimal integration points a_j and b_j in (2.23) are found by differentiation of $g(x)$ w.r.t. x, i.e.

$$G(x) := \frac{\partial g(x)}{\partial x} = \frac{\exp(x)}{\exp(x)+1} = \frac{1}{1+\exp(-x)},$$

giving a relation between the derivative of $g(x)$ and the logistic distribution. This representation indicates that $G(x)$ is the CDF of the logistic distribution with a null location parameter and a scale parameter $s = 1$. In order to determine the optimal integration points so that their positions correspond to the logistic distribution, we need to compute the quantiles. The quantiles of the logistic distribution are analytically available and given by

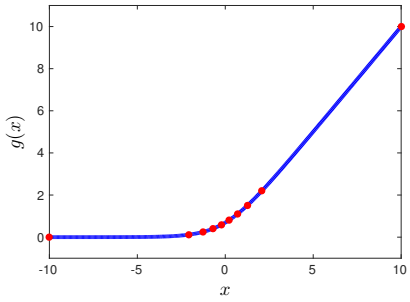
$$q(p) = \log\left(\frac{p}{1-p}\right).$$

The algorithm for the optimal integration points, given integration interval $[a, b]$ and the number of intervals L, is as follows:

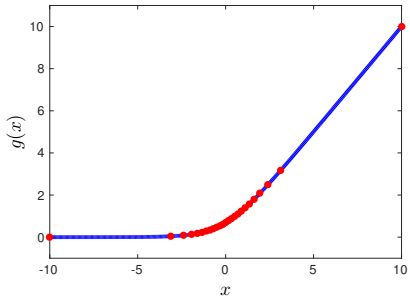
- Determine an interval in the probability space: $P_g = [G(a), G(b)]$.
- Divide the interval P_g equally in L parts: $P_g^j = [p_j, p_{j+1}]$, $j = 0 \ldots L-1$.
- Determine a_j and b_j by calculating the quantiles, $a_j = q(p_j)$ and $b_j = q(p_{j+1})$, $j = 0 \ldots L-1$.

The algorithm above ensures that integration points are optimally distributed. In Figures 2.1(a) and 2.1(b), we present the optimal integration points for $L = 10$ and $L = 25$, respectively. The points are well-positioned in the region of interest, as expected.

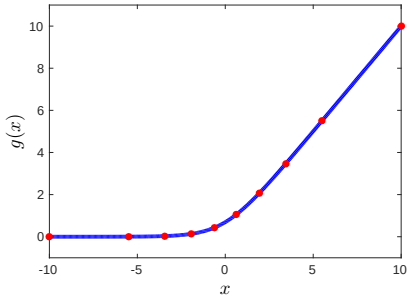
However, for our specific problem, the integration interval can be rather large. As the integration points are densely concentrated in the function's curved part, the errors in the outer sub-intervals dominate the overall error and more points at the beginning and at the end of the interval are required to reduce this error. A more balanced distribution of the integration points can be achieved by introducing a scale parameter, $s \neq 1$, which



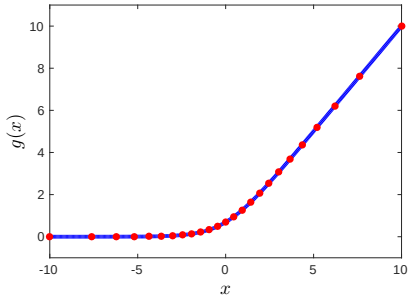
(a) $L = 10$ and $s = 1$



(b) $L = 25$ and $s = 1$



(c) $L = 10$ and $s = 3$



(d) $L = 25$ and $s = 3$

Figure 2.1: Optimal integration points.

	$L=25$	$L=50$	$L=75$	$L=100$
Equidistant	2.4075×10^{-3}	2.6743×10^{-4}	3.1064×10^{-4}	2.9547×10^{-6}
Optimal ($s=1$)	1.3273×10^{-2}	2.5343×10^{-3}	9.5959×10^{-4}	4.7953×10^{-4}
Optimal ($s=3$)	3.4837×10^{-5}	1.1445×10^{-6}	2.1506×10^{-7}	6.7114×10^{-8}

Table 2.1: MSE vs. L for $\tilde{I}_{\mathscr{M}}$ in Equation (2.23).

implies that the integration points correspond to a CDF with increased variance. In Figures 2.1(c) and 2.1(d), the distributions of the points for $s=3$ are shown. In the case of $s\neq1$, the expression for the quantiles reads,

$$q(p)=s\log\left(\frac{p}{1-p}\right).$$

The impact of the optimal point redistribution can be seen in Table 2.1. The mean squared error (MSE), in absolute value, of the piecewise linear approximation is presented for a fixed interval $[-10,30]$ and a fixed number of elements in the cosine expansion, $N=150$, considering different numbers of sub-intervals and different integration point locations (equidistant, optimal with $s=1$ and optimal with $s=3$). We observe that, for the problem at hand, the optimal distribution with $s>1$ gives better accuracy.

2.3.3. ERROR AND PERFORMANCE ANALYSIS

In the sections above, we have proposed a method to compute the approximated CDF for SABR's time-integrated variance. In this section, we perform an error analysis, indicating how to reduce or bound the errors. As usual, there is a relation between the error (or accuracy) and the performance and we cannot discuss them separately. Therefore, we will take into account the computational effort here. The sources of error in our approximation include ϵ_D, the discretization of $Y(T)$ in Equation (2.4), ϵ_T, the truncation in Equations (2.14) and (2.15), ϵ_C, due to the cosine expansion in Equations (2.13) and (2.16) and ϵ_M, the numerical computation of $I_{\mathscr{M}}$ in Equation (2.20).

Error ϵ_D. This error occurs because the SABR's time-integrated variance is approximated by its discrete equivalent, i.e. $Y(T)\approx\tilde{Y}(T)$. Note that the process $Y(t)=\int_0^t\sigma^2(s)\mathrm{d}s$ is the solution of the SDE

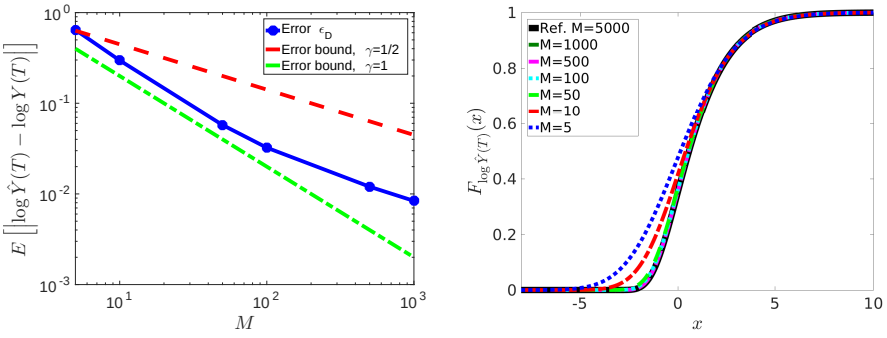
$$\mathrm{d}Y(s)=\sigma^2(s)\mathrm{d}s,\quad Y(0)=0,$$

which, by employing the same time discretization as for $\tilde{Y}(T)$ in Equation (2.4), can be written as

$$\bar{Y}(t_j)-\bar{Y}(t_{j-1})=\sigma^2(t_{j-1})\Delta t.$$

This is essentially the Euler-Maruyama discretization scheme. Indeed, it is easy to see that, if we sum the right-hand side of the equality over the M discrete time points, we recover our discrete approximation $\tilde{Y}(T)$ as follows

$$\sum_{j=1}^{M}\sigma^2(t_{j-1})\Delta t=\sum_{j=1}^{M}\left(\bar{Y}(t_j)-\bar{Y}(t_{j-1})\right)=\bar{Y}(t_M)-\bar{Y}(0)=\tilde{Y}(T).$$



(a) Error ϵ_D (b) $F_{\log \tilde{Y}(T)}(x)$

Figure 2.2: Convergence analysis in terms of the number of discrete points, M.

The error (or convergence) of the Euler-Maruyama discretization scheme has been intensively studied in the literature (see [83], for example). Two errors can be considered when a stochastic process is discretized: *strong* and *weak*. It was proved by Kloeden and Platen in [83] that, under some conditions of regularity, the Euler-Maruyama scheme has strong convergence with order $\gamma = \frac{1}{2}$ and weak convergence with order $\gamma = 1$. In our particular case, we focus on the strong error between $Y(T)$ and $\tilde{Y}(T)$. Then, the error ϵ_D can be computed and bounded by employing the usual strong convergence expression as

$$\epsilon_D := \mathbb{E}\left[\left|\tilde{Y}(T) - Y(T)\right|\right] \leq C\left(\frac{T}{M}\right)^{\gamma},$$

for some constant $C > 0$.

In our actual experiments, we consider $\log \tilde{Y}(T)$ and $\log Y(T)$. This will reduce the error further. In order to numerically show the order of convergence of our approximation, in Figure 2.2(a) the strong error[2] of $\log \tilde{Y}(T)$ with respect to the number of discrete time points, M, is depicted. Additionally, "upper" ($C = 1$ and $\gamma = \frac{1}{2}$) and "lower" ($C = 1$ and $\gamma = 1$) bounds are presented as references. In order to further test the convergence, the resulting CDFs, $F_{\log \tilde{Y}(T)}$, are presented in Figure 2.2(b), again varying the number of discrete time points, M. Both experiments perform as expected.

Error ϵ_T. The use of cosine expansions and the COS method implies a truncation of the integration range to interval $[a, b]$. This gives rise to a truncation error, which is defined by

$$\epsilon_T(H) := \int_{\mathbb{R}\setminus[a,b]} f_H(y)\mathrm{d}y,$$

where H corresponds to $\log \tilde{Y}(T)$ and Y_j, according to Equations (2.14) and (2.15), respectively. By a sufficiently large integration range $[a, b]$, this error can be reduced and does not dominate the overall error.

[2] As $\log Y(T)$ is unknown, the reference value is computed by using a very fine discretization, i.e. $M = 5000$.

Error ϵ_C. The convergence due to the number of terms N in the cosine series expansion was derived in [49]. For any $f(y|x) \in \mathbf{C}^\infty[a,b]$, ϵ_C can be bounded by

$$|\epsilon_C(N,[a,b])| \leq P^*(N)\exp(-(N-1)\nu),$$

being $\nu > 0$ a constant and $P^*(N)$ a term which varies less than exponentially with respect to N. As the PDF of the time-integrated variance is a very smooth function, ϵ_C will decay exponentially with respect to N.

Error ϵ_M. We have described two different numerical methods to compute integral $I_{\mathcal{M}}$: Clenshaw-Curtis quadrature and the piecewise linear approximation.

Regarding the Clenshaw-Curtis quadrature in Equation (2.22) as stated in [137], the error can be bounded by $O((2n_q)^{-m}/m)$ for an m-times differentiable integrand, with n_q the number of quadrature terms. When m is bounded, we achieve algebraic convergence while, otherwise, the error converges exponentially with respect to n_q. In Equation (2.20), the integrand of $I_{\mathcal{M}}$ belongs to $\mathbf{C}^\infty[a,b]$ thus, by Clenshaw-Curtis quadrature, error ϵ_M can be reduced with exponential convergence in n_q.

When the approximation in Equation (2.23) is employed, we observe a reduction of the error by a factor four, when going from $L = 25$ to $L = 50$ integration points in Table 2.1. However, with $L = 100$ the error reduces more drastically due to the choice of optimal integration points.

PERFORMANCE ANALYSIS

We perform an efficiency analysis considering the terms that control the evaluated errors. Error ϵ_T is not taken into account here, as we always employ a sufficiently large integration interval.

Error ϵ_D can be reduced by increasing M in Equation (2.4). We find that the value of M hardly affects the computational cost, so it can be increased without a significant impact on the performance.

Error ϵ_C is governed by the number of series expansion terms, N. As ϵ_C exhibits an exponential decay w.r.t. N, a relatively small value ($N \approx 100$) already results in highly accurate approximations. Larger N values have a great impact on the performance because $I_{\mathcal{M}}$ is calculated $N \times N$ times. As this calculation dominates the computational cost, the execution time increases by $O(N^2)$.

Error ϵ_M can be controlled by the number of quadrature terms, n_q, in the case of the Clenshaw-Curtis method or by the number of sub-intervals, L, when using the piecewise linear approximation. Again, the impact can be significant because $I_{\mathcal{M}}$ is computed many times. To achieve a similar order of accuracy ($\sim 10^{-7}$) in a generic interval $[a,b]$, the piecewise linear approximation is approximately three times faster than the Clenshaw-Curtis quadrature. Therefore, we will use the piecewise linear approximation in the following.

2.4. SIMULATION OF $Y(T)|\sigma(T)$: COPULA APPROACH

The approximated CDF of the marginal distribution of $Y(T)$ (in the log-space, see Section 2.3) is employed to simulate $Y(T)|\sigma(T)$ by means of a *copula*. A d-dimensional

copula is a joint distribution function on a unit hyper-cube $[0,1]^d$ with d marginal distributions. A distribution with a copula contains essentially the same information as the joint distribution, like the marginal properties and the dependence structure, but this information can be decoupled. The importance of copulas is represented by two parts of the Sklar theorem, introduced by Abe Sklar in [122]. The first part states that a copula can be derived from any joint distribution function, and the second part states that any copula can be combined with any set of marginal distributions to result in a multivariate distribution function. More formally, Sklar's theorem reads

- Let F be a joint distribution function and F_j, $j = 1,\dots,d$, be the marginal distributions. Then there exists a copula $C : [0,1]^d \to [0,1]$ such that

$$F(x_1,\dots,x_d) = C(F_1(x_1),\dots,F_d(x_d)),$$

  for all $x_j \in [-\infty,\infty]$. Moreover, if the margins are continuous, then C is unique; otherwise C is uniquely determined on $Ran(F_1) \times \cdots \times Ran(F_d)$, where $Ran(F_j) = F_j([-\infty,\infty])$ is the range of F_j.

- The converse is also true. That is, if C is a copula and $F_1,\dots,F_d$ are univariate distribution functions, then the multivariate function defined by the preceding equation is a joint distribution function with marginal distributions F_j, $j = 1,\dots,d$.

2.4.1. FAMILIES OF COPULAS

Copulas can be classified into so-called *fundamental*, *implicit* and *explicit* copulas. Fundamental copulas are copulas that represent either perfect positive dependence, independence or perfect negative dependence. Implicit copulas are copulas extracted from well-known multivariate distributions, like the normal, Gaussian or Student **t** distributions, that do not have closed-form expressions. Explicit or Archimedean copulas have simple closed-form expressions. The Clayton, Frank and Gumbel copulas are the most commonly employed ones.

Here, we briefly describe these copulas, considering $F_1 \dots, F_d \in [0,1]^d$ as the marginal distributions:

- *Gaussian*: given a correlation matrix, $\mathscr{R} \in \mathbb{R}^{d\times d}$, the Gaussian copula with parameter $\mathscr{R}$ is defined by

$$\mathscr{C}_{\mathscr{R}}(F_1 \dots, F_d) = \Phi_{\mathscr{R}}(\Phi^{-1}(F_1),\dots,\Phi^{-1}(F_d)),$$

  where Φ is the distribution function of a standard normal random variable and $\Phi_{\mathscr{R}}$ is the d-variate standard normal distribution with zero mean vector and covariance matrix $\mathscr{R}$.

- *Student* **t**: given correlation matrix $\mathscr{R} \in \mathbb{R}^{d\times d}$ and the degrees of freedom parameter $\nu \in (0,\infty)$, the **t** copula is defined by

$$\mathscr{C}_{\mathscr{R},\nu}(F_1 \dots, F_d) = \mathbf{t}_{\mathscr{R},\nu}(\mathbf{t}_\nu^{-1}(F_1),\dots,\mathbf{t}^{-1}(F_d)),$$

  where $\mathbf{t}_\nu$ and $\mathbf{t}_{\mathscr{R},\nu}$ are univariate and the d-variate Student **t** distributions, respectively.

- *Archimedean*: is governed by the representation

$$\mathscr{C}_\theta(F_1\ldots,F_d)=\psi_\theta^{-1}(\psi_\theta(F_1)+\cdots+\psi_\theta(F_d)),$$

where $\psi_\theta:[0,1]\to[0,\infty]$ is the so-called generator function which is supposed to be a continuous, strictly decreasing and convex function such that $\psi_\theta(1)=0$, $\psi_\theta(0)=\infty$ and its inverse ψ_θ^{-1} is monotonic on $[0,\infty)$. A variety of Archimedean copulas can be defined, like the following well-known copulas.

The *Clayton* copula, for which the generator function is given by $\psi_\theta(u)=\theta^{-1}(u^{-\theta}-1)$, $\theta>0$, reads

$$\mathscr{C}_\theta(F_1\ldots,F_d)=\left(\sum_{i=1}^{d}F_i^{-\theta}-d+1\right)^{-1/\theta}. \tag{2.24}$$

The *Frank* copula, with generator function given by $\psi_\theta(u)=-\log\left(\frac{\exp(-\theta u)-1}{\exp(-\theta)-1}\right)$, reads

$$\mathscr{C}_\theta(F_1\ldots,F_d)=-\frac{1}{\theta}\log\left(1+\frac{\prod_{i=1}^{d}(\exp(-\theta F_i)-1)}{(\exp(-\theta)-1)^{d-1}}\right), \tag{2.25}$$

being $\theta\in\mathbb{R}\setminus\{0\}$ for $d=2$ and $\theta>0$ for $d\geq 3$.

The *Gumbel* copula, with generator function $\psi_\theta(u)=\big(-\log(u)\big)^\theta$, $\theta>0$, reads

$$\mathscr{C}_\theta(F_1\ldots,F_d)=\exp\left(-\left(\sum_{i=1}^{d}\big(-\log(F_i)\big)^\theta\right)^{1/\theta}\right). \tag{2.26}$$

In order to calibrate parameter θ in (2.24), (2.25) or (2.26) a relation between θ and the rank correlation coefficient Kendall's τ can be employed. This relation is available in closed-form for the Clayton copula, i.e. $\theta=2\tau/(1-\tau)$, and the Gumbel copula, i.e. $\theta=1/(1-\tau)$. There is no analytic expression of θ for the Frank copula.

To apply any of these copulas, an approximation for the correlation between the involved distributions needs to be determined. Here, Pearson's correlation coefficient, $\mathscr{P}$, is chosen since it is employed directly in the Gaussian and Student **t** copulas and a relation with Kendall's τ exists

$$\mathscr{P}=\sin\left(\frac{\pi}{2}\tau\right).$$

In Subsection 2.4.2, an approximation for Pearson's coefficient between $\log Y(T)$ and $\log\sigma(T)$ of the SABR dynamics is derived.

2.4.2. CORRELATION: $\log Y(T)$ AND $\log\sigma(T)$

For the problem at hand, we need to derive an expression for Pearson's correlation coefficient $\mathscr{P}_{\log Y(T),\log\sigma(T)}$. By definition, we have

$$\mathscr{P}_{\log Y(T),\log\sigma(T)}=\operatorname{corr}\left[\log\int_0^T\sigma^2(s)\mathrm{d}s,\log\sigma(T)\right]=\frac{\operatorname{cov}\left[\log\int_0^T\sigma^2(s)\mathrm{d}s,\log\sigma(T)\right]}{\sqrt{\mathbb{V}\mathrm{ar}\left[\log\int_0^T\sigma^2(s)\mathrm{d}s\right]\mathbb{V}\mathrm{ar}\left[\log\sigma(T)\right]}}.$$

We employ the following approximation

$$\log \int_0^T \sigma^2(s)\mathrm{d}s \approx \int_0^T \log\sigma^2(s)\mathrm{d}s = 2\int_0^T \log\sigma(s)\mathrm{d}s,$$

where the logarithm and the integral are interchanged. Since the log function is concave, this approximation forms a lower bound for the true value. This can be seen by applying Jensen's inequality, i.e.

$$\log \int_0^T \sigma^2(s)\mathrm{d}s \geq \int_0^T \log\sigma^2(s)\mathrm{d}s.$$

We will numerically show that, under our model settings with an option contract time to maturity of less than two years, this is a highly satisfactory approximation. The correlation coefficient can now be computed as

$$\mathscr{P}_{\log Y(T),\log\sigma(T)} \approx \frac{\mathrm{cov}\left[\int_0^T \log\sigma(s)\mathrm{d}s, \log\sigma(T)\right]}{\sqrt{\mathbb{V}\mathrm{ar}\left[\int_0^T \log\sigma(s)\mathrm{d}s\right]\mathbb{V}\mathrm{ar}\left[\log\sigma(T)\right]}}.$$

The quantity $\mathbb{V}\mathrm{ar}\left[\log\sigma(T)\right]$ is known by Equation (2.3), and we derive expressions the other two quantities. Starting with the covariance, we have

$$\mathrm{cov}\left[\int_0^T \log\sigma(s)\mathrm{d}s, \log\sigma(T)\right] = \mathbb{E}\left[\left(\int_0^T \log\sigma(s)\mathrm{d}s\right)\log\sigma(T)\right] - \mathbb{E}\left[\int_0^T \log\sigma(s)\mathrm{d}s\right]\mathbb{E}\left[\log\sigma(T)\right].$$

From Equation (2.3), we find

$$\log\sigma(T) = \log\sigma_0 - \frac{1}{2}\alpha^2 T + \alpha W(T) =: \eta(T) + \alpha W(T),$$

and

$$\int_0^T \log\sigma(s)\mathrm{d}s = T\log\sigma_0 - \frac{1}{4}\alpha^2 T^2 + \alpha\int_0^T W(s)\mathrm{d}s =: \gamma(T) + \alpha\int_0^T (T-s)\mathrm{d}W(s).$$

Based on these equations, with

$$\mathbb{E}\left[\log\sigma(T)\right] = \eta(T), \quad \mathbb{E}\left[\int_0^T \log\sigma(s)\mathrm{d}s\right] = \gamma(T),$$

we obtain the following expressions for the required expectation,

$$\begin{aligned}
\mathbb{E}\left[\left(\int_0^T \log\sigma(s)\mathrm{d}s\right)\log\sigma(T)\right] &= \mathbb{E}\left[\left(\gamma(T) + \alpha\int_0^T (T-s)\mathrm{d}W(s)\right)\left(\eta(T) + \alpha W(T)\right)\right] \\
&= \gamma(T)\eta(T) + \alpha^2\mathbb{E}\left[\int_0^T \mathrm{d}W(s)\int_0^T (T-s)\,\mathrm{d}W(s)\right] \\
&= \gamma(T)\eta(T) + \alpha^2\mathbb{E}\left[\int_0^T (T-s)\,\mathrm{d}s\right] = \gamma(T)\eta(T) + \frac{1}{2}\alpha^2 T^2.
\end{aligned}$$

In order to determine the variance of $\int_0^T \log\sigma(s)\mathrm{d}s$, we compute

$$\begin{aligned}\mathbb{E}\left[\left(\int_0^T \log\sigma(s)\mathrm{d}s\right)^2\right] &= \mathbb{E}\left[\left(\gamma(T)+\alpha\int_0^T (T-s)\,\mathrm{d}W(s)\right)^2\right]\\ &= \gamma^2(T)+\alpha^2\mathbb{E}\left[\int_0^T (T-s)^2\,\mathrm{d}s\right]\\ &= \gamma^2(T)+\frac{1}{3}\alpha^2T^3,\end{aligned}$$

and the variance reads

$$\mathbb{V}\mathrm{ar}\left[\int_0^T \log\sigma(s)\mathrm{d}s\right] = \mathbb{E}\left[\left(\int_0^T \log\sigma(s)\mathrm{d}s\right)^2\right] - \left(\mathbb{E}\left[\int_0^T \log\sigma(s)\mathrm{d}s\right]\right)^2 = \frac{1}{3}\alpha^2T^3.$$

Pearson's correlation coefficient is then obtained as

$$\begin{aligned}\mathscr{P}_{\log Y(T),\log\sigma(T)} &\approx \frac{\mathbb{E}\left[\left(\int_0^T \log\sigma(s)\mathrm{d}s\right)\log\sigma(T)\right] - \mathbb{E}\left[\int_0^T \log\sigma(s)\mathrm{d}s\right]\mathbb{E}\left[\log\sigma(T)\right]}{\sqrt{\mathbb{V}\mathrm{ar}\left[\int_0^T \log\sigma(s)\mathrm{d}s\right]\mathbb{V}\mathrm{ar}\left[\log\sigma(T)\right]}}\\ &= \frac{\gamma(T)\eta(T)+\frac{1}{2}\alpha^2T^2-\gamma(T)\eta(T)}{\sqrt{\left(\frac{1}{3}\alpha^2T^3\right)\left(\alpha^2T\right)}} = \frac{\frac{1}{2}\alpha^2T^2}{\sqrt{\frac{1}{3}\alpha^4T^4}} = \frac{\sqrt{3}}{2}.\end{aligned}$$

The approximation obtained is a *constant value*. We will show numerically that, for the problems at hand, this appears to be a very reasonable approximation. The correlation between $\log Y(T)$ and $\log\sigma(T)$ is affected by the maturity time T, and by the *volatility-of-volatility* parameter α. In Figure 2.3, the empirical and approximated (red grid) correlations are depicted for typical values of T and α. Because we focus on short maturity options, we restrict $T\in[0,2]$ and $\alpha\in(0,1]$. The experiment shows that, only in rather extreme cases ($\alpha>0.7$), the differences between the empirical and approximated correlation increase, but remain within five basis points. The approximation is therefore sufficiently accurate for our purposes, since a small error in the correlation (of less than ten basis points) does not affect the performed simulation significantly.

2.4.3. SAMPLING $Y(T)|\sigma(T)$

The following steps describe the complete procedure for the simulation of $Y(T)$ given $\sigma(T)$ by using a bivariate copula approach:

1. Determine $F_{\log\sigma(T)}$ (analytically) and $F_{\log\tilde{Y}(T)}$ via Equation (2.14).
2. Define the bivariate copula distribution by employing the marginal distributions and the correlation approximation (Subsection 2.4.2).
3. Generate correlated uniform samples, $U_{\log\sigma(T)}$ and $U_{\log\tilde{Y}(T)}$ by means of the bivariate copula distribution.
4. From $U_{\log\sigma(T)}$ and $U_{\log\tilde{Y}(T)}$ invert the original marginals, $F_{\log\sigma(T)}$ and $F_{\log\tilde{Y}(T)}$. The resulting quantiles should be correlated under the chosen correlation coefficient.

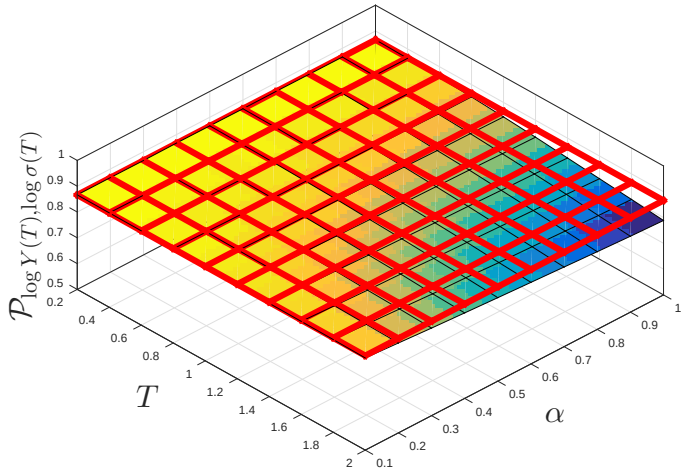


Figure 2.3: Pearson's correlation coefficient: Empirical (surface) vs. approximation (red grid).

5. Finally, the samples of $Y(T)|\sigma(T)$ are obtained by taking exponentials.

The inversion procedure in the fourth step is straightforward in the case of $\sigma(T)$. The case of $\tilde{Y}(T)$ is more involved, and we propose a direct inversion procedure based on linear interpolation. The insight is that, by rotating CDF $F_{\log\tilde{Y}(T)}$, we can interpolate over probabilities instead of $x-$values. This is possible since $F_{\log\tilde{Y}(T)}$ is a smoothly increasing function. The interpolation polynomial provides the quantiles of the original distribution, $F_{\log\tilde{Y}(T)}$, from the probabilities $U_{\log\tilde{Y}(T)}$. This inversion procedure is very fast (at almost no cost) and sufficiently accurate.

2.4.4. SIMULATION OF $S(T)$ GIVEN $S(0)$, $\sigma(T)$ AND $\int_0^T \sigma^2(s)\mathrm{d}s$

We complete the one-step SABR simulation by the conditional sampling of $S(T)$. The most commonly used techniques can be classified into two categories: direct inversion of SABR's CDF given in Equation (2.2) and moment-matching approaches. The direct inversion procedure has a higher computational cost because of the evaluation of the non-central χ^2 distribution. However, some recent developments make this computation affordable. In [21], Chen et al. proposed a forward asset simulation based on a combination of moment-matching (quadratic Gaussian) and enhanced direct inversion procedures. We employ this technique also here. Note, however, that, for some specific values of β, the simulation of the conditional $S(T)$ given $S(0)$, $\sigma(T)$ and $\int_0^T \sigma^2(s)\mathrm{d}s$ gives rise to analytic expressions.

CASE $\beta = 0$

For $\beta = 0$, it is easily seen from Equation (2.1) that the asset process (in integral form) becomes

$$S(T) = S(0) + \rho \int_0^T \sigma(s) \mathrm{d}\tilde{W}_\sigma(s) + \sqrt{1-\rho^2} \int_0^T \sigma(s) \mathrm{d}\tilde{W}_S(s),$$

where $\mathrm{d}\tilde{W}_S(t)$ and $\mathrm{d}\tilde{W}_\sigma(t)$ are independent Brownian motions and with

$$\int_0^T \sigma(s) \mathrm{d}\tilde{W}_\sigma(s) = \frac{1}{\alpha} \left(\sigma(T) - \sigma(0) \right), \tag{2.27}$$

and

$$\int_0^T \sigma(s) \mathrm{d}\tilde{W}_S(s) | \sigma(T) \sim \mathcal{N}\left(0, \sqrt{\int_0^T \sigma^2(s) \mathrm{d}s}\right). \tag{2.28}$$

CASE $\beta = 1$

In the case of $\beta = 1$, the asset dynamics in Equation (2.1) are log-normal and the solution is given by

$$S(T) = S(0) \exp\left(-\frac{1}{2} \int_0^T \sigma^2(s) \mathrm{d}s + \rho \int_0^T \sigma(s) \mathrm{d}\tilde{W}_\sigma(s) + \sqrt{(1-\rho^2)} \int_0^T \sigma(s) \mathrm{d}\tilde{W}_S(s) \right).$$

If we take the log-transform,

$$\log\left(\frac{S(T)}{S(0)} \right) = -\frac{1}{2} \int_0^T \sigma^2(s) \mathrm{d}s + \rho \int_0^T \sigma(s) \mathrm{d}\tilde{W}_\sigma(s) + \sqrt{(1-\rho^2)} \int_0^T \sigma(s) \mathrm{d}\tilde{W}_S(s),$$

and by employing again Equation (2.27) and Equation (2.28), we obtain the distribution of $\log\left(\frac{S(T)}{S(0)}\right) | \int_0^T \sigma^2(s) \mathrm{d}s, \sigma(T), \sigma(0)$, which reads

$$\mathcal{N}\left(-\frac{1}{2} \int_0^T \sigma^2(s) \mathrm{d}s + \frac{\rho}{\alpha} \left(\sigma(T) - \sigma(0) \right), \sqrt{(1-\rho^2) \int_0^T \sigma^2(s) \mathrm{d}s} \right). \tag{2.29}$$

By employing Equation (2.29), the asset dynamics $S(t)$ can be sampled from

$$S(T) \sim S(0) \exp\left(-\frac{1}{2} \int_0^T \sigma^2(s) \mathrm{d}s + \frac{\rho}{\alpha} \left(\sigma(T) - \sigma(0) \right) + X \sqrt{(1-\rho^2) \int_0^T \sigma^2(s) \mathrm{d}s} \right), \tag{2.30}$$

where X is the standard normal.

CASE $\beta \neq 0$, $\beta \neq 1$

As mentioned before, for the generic case of $\beta \in (0,1)$, we employ the enhanced inversion of the SABR asset price distribution in Equation (2.2) as introduced in [21]. We briefly summarize this approach. The asset simulation is performed either by a moment-matched quadratic Gaussian approximation or by an enhanced direct inversion. The authors in [21] proposed a threshold value to choose the suitable technique among these

two. The moment-matching approach relies on the fact that, for $S(0) \gg 0$, the distribution function in Equation (2.2) can be approximated by

$$Pr\left(S(t) \leq K | S(0) > 0, \sigma(t), \int_0^t \sigma^2(s)\mathrm{d}s\right) \approx \chi^2(c; 2-b, a).$$

By definition, the mean, μ and the variance, κ^2, of a generic non-central chi-square distribution $\chi^2(x;\delta,\lambda)$ are $\mu := \delta + \lambda$ and $\kappa^2 := 2(\delta + 2\lambda)$, respectively. A variable $\mathcal{V} \sim \chi^2(x;\delta,\lambda)$ is accurately approximated by a quadratic Gaussian distribution, as follows

$$\mathcal{V} \overset{\mathrm{d}}{\approx} d(e+X)^2, \quad X \sim \mathcal{N}(0,1),$$

with

$$\psi := \frac{\kappa^2}{\mu^2}, \quad e^2 = 2\psi^{-1} - 1 + \sqrt{2\psi^{-1}}\sqrt{2\psi^{-1}-1} \geq 0, \quad d = \frac{\mu}{1+e^2}.$$

Applying this to our case in Equation (2.2), we can sample the conditional asset dynamics, $T > 0$, by

$$S(T) = \left((1-\beta)^2(1-\rho^2)d(e+X)^2 \int_0^T \sigma^2(s)\mathrm{d}s\right)^{\frac{1}{2(1-\beta)}}.$$

The moment-matching approximation is accurately applicable when $0 < \psi \leq 2$ and $\mu \geq 0$ [21]. Otherwise, a direct inversion procedure must be employed. A root-finding Newton method is then used. In order to reduce the number of Newton iterations (and the expensive evaluation of the non-central chi-square CDF), the first iterations are based on an approximated formula for the non-central chi-square distribution, which is based on the normal CDF and derived by Sankaran in [114]. Then, the obtained value is employed as the initial guess for the Newton algorithm. The result is a significant reduction in the number of iterations and, hence, in the computational cost. Furthermore, this root-finding procedure consists of only basic operations, so that the whole procedure can be easily vectorized, leading to a further efficiency improvement. The resulting sampling procedure is robust and efficient.

MARTINGALE CORRECTION

As pointed out in [2] and [21], the exact simulation of the asset price can, in certain SV models, give rise to loss of the martingale property (because of the approximation of a continuous process by its discrete equivalent). This is especially seen, when the size of the time-step is large and for certain configurations of the SABR parameters, like small β values, close-to-zero initial asset values S_0, high *vol-vol* parameter α or large initial volatility σ_0. As we deal with one *large* time-step, a martingale correction needs to be applied. We incorporate a simple but effective numerical martingale correction, as follows

$$\begin{aligned} S(T) &= S(T) - \frac{1}{n}\sum_{p=1}^{n} S_p(T) + \mathbb{E}[S(T)], \\ &= S(T) - \frac{1}{n}\sum_{p=1}^{n} S_p(T) + S_0, \end{aligned}$$

where $S_p(\cdot)$ represents the forward asset value for the $p-$th Monte Carlo path.

Note, however, that, for practical SABR parameters, the martingale error is very small.

	S_0	σ_0	α	β	ρ	T
Set I	1.0	0.5	0.4	0.7	0.0	2
Set II	0.05	0.1	0.4	0.0	−0.8	0.5
Set III	0.04	0.4	0.8	1.0	−0.5	2

Table 2.2: Data sets.

2.5. NUMERICAL EXPERIMENTS

In this section, different numerical experiments will be carried out. In Subsection 2.5.1, we compare the different copulas from Subsection 2.4.1 for the simulation of $Y(T)|\sigma(T)$. After that, in Subsection 2.5.2, we employ the best performing copulas in a SABR pricing experiment. We consider several representative SABR data sets with special characteristics, like a zero correlation (Set I), a normal SABR model (Set II) and a high *volatility-of-volatility* SABR set (Set III). The parameter values are shown in Table 2.2. Other parameters in our one-step SABR method include:

- Number of discrete time points: $M = 1{,}000$.
- Number of COS elements: $N = 150$.
- Number of intervals for piecewise linear approximation of $I_{\mathcal{M}}$: $L = 100$.

The experiments were performed in a computer with CPU Intel Core i7-4720HQ 2.6 GHz and RAM memory of 16 Gigabytes. The employed software package was Matlab r2015b.

2.5.1. COPULA APPROACH ANALYSIS

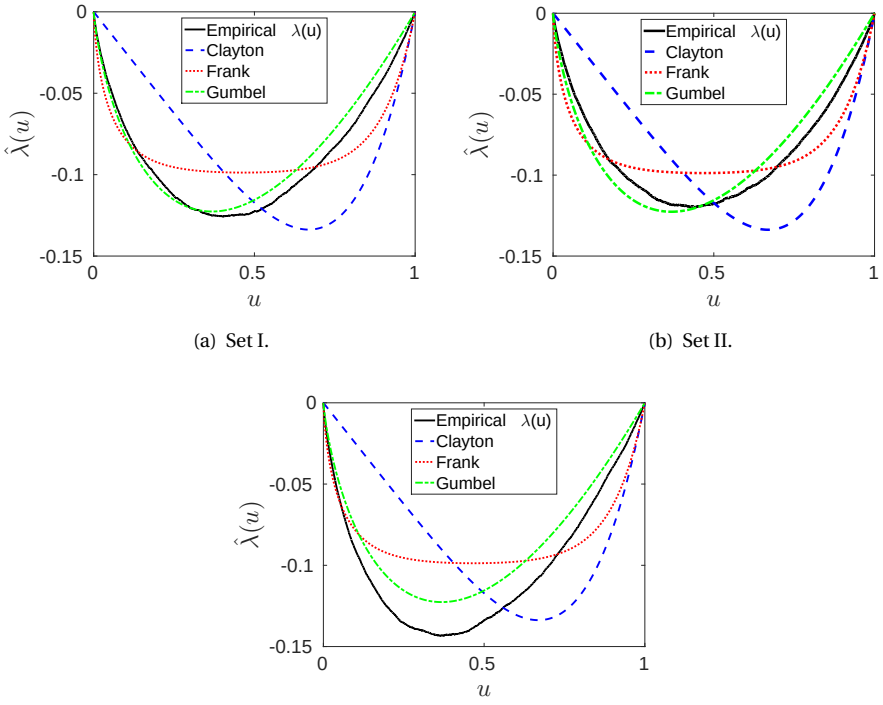
As $Y(T) = \int_0^T \sigma^2(s)\mathrm{d}s$ is a summation of squared log-normal processes, Gaussian-based copulas may fit well. However, we also consider Archimedean copulas as these may be superior. In order to choose an optimal copula, we assess a so-called *goodness-of-fit* (GOF) criterion for copulas. There are different GOF tests in the literature, see an overview in [9], for example. These measures are based on a comparison between some empirical accurate approximations (that are usually computationally expensive) and the analyzed copula. According to the work of Durrleman et al. in [45], we split the analysis, first evaluating the Archimedean copulas and subsequently, after choosing the optimal copula from this class, performing an analysis of the remaining copulas. The GOF testing for Archimedean copulas is a graphic procedure based on Kendall's processes, proposed by Genest and Rivest [55] and Barbe et al. [6]. They defined a function $\mathcal{K}$ by

$$\mathcal{K}(u) = Pr\left[\mathscr{C}(U_1 \ldots, U_d) \leq u\right]$$

where $\mathscr{C}$ is the Archimedean copula and $u \in [0,1]$. A non-parametric estimate[3] of $\mathcal{K}$ is available as

$$\tilde{\mathcal{K}}(u) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{[\vartheta_i \leq u]}, \quad \vartheta_i = \frac{1}{n-1}\sum_{j=1}^{n} \mathbf{1}_{[x_1^j < x_1^i, \ldots, x_d^j < x_d^i]},$$

[3] An analytic solution is known, but impracticable in terms of computational effort.



(a) Set I.

(b) Set II.

(c) Set III.

Figure 2.4: Archimedean GOF test: $\hat{\lambda}(u)$ vs. empirical $\lambda(u)$.

with n the number of samples and x_d^i the i-th sample of the d-th marginal distribution. Since we deal with a bivariate case, we have $d = 2$. The graphical procedure of the GOF test consists in comparing the distance $\hat{\lambda}(u) := u - \tilde{\mathcal{K}}(u)$ and the *empirical* distance, $\lambda(u)$, obtained by using an empirical estimation of $\mathcal{K}(u)$.

In Figure 2.4, the distances $\hat{\lambda}(u)$ between three Archimedean copulas (Clayton, Frank and Gumbel) for the graphical GOF test are depicted. The empirical $\lambda(u)$ (black line) is employed as the reference. The experiment is performed for each data set in Table 2.2. As the measurable quantity, the MSE of $\hat{\lambda}(u) - \lambda(u)$ is presented in Table 2.3. We use $n = 100,000$ simulations.

From the GOF results for the Archimedean copulas, we find that the Gumbel copula

	Clayton	Frank	Gumbel
Set I	1.3469×10^{-3}	2.9909×10^{-4}	5.1723×10^{-5}
Set II	1.0885×10^{-3}	2.1249×10^{-4}	8.4834×10^{-5}
Set III	2.1151×10^{-3}	7.5271×10^{-4}	2.6664×10^{-4}

Table 2.3: MSE of $\hat{\lambda}(u) - \lambda(u)$.

	Gaussian	Student **t** ($\nu = 5$)	Gumbel
Set I	5.0323×10^{-3}	5.0242×10^{-3}	3.8063×10^{-3}
Set II	3.1049×10^{-3}	3.0659×10^{-3}	4.5703×10^{-3}
Set III	5.9439×10^{-3}	6.0041×10^{-3}	4.3210×10^{-3}

Table 2.4: Generic GOF: D_2.

fits best in our framework. Once we have determined the most suitable Archimedean copula, we perform a GOF test including the Gaussian, Student **t** and Gumbel copulas. For an objective comparison, also a measurable GOF technique is applied. Among the several families of available generic GOF tests in the literature, in [9], Berg suggests that methods that rely on the so-called *Deheuvels or empirical copula* [42] give reliable results. Hence, we will perform a test which is based on the distances between the Deheuvels copula, $\mathscr{C}_d$, and the analyzed copula $\mathscr{C}$ (see [45]). By using the discrete L^2 norm, this GOF measure reads

$$D_d(\mathscr{C}_d, \mathscr{C}) = \|\mathscr{C}_d - \mathscr{C}\|_{L^2},$$

where d is the number of random variables to build the copula, $d = 2$. For this GOF test, we reduce the number of samples to $n = 1,000$ due to the computational (time and memory) cost of the Deheuvels copula calculations. The other parameters remain the same. In Table 2.4, the obtained distances, D_2, between the tested copulas (Gaussian, Student **t** and Gumbel) and the Deheuvels copula are shown.

According to the GOF results, the three copulas perform very similarly. When longer maturities are considered, the Gumbel copula exhibits smallest errors in the GOF tests. In terms of speed, the Gaussian copula is around three times faster than the Gumbel copula, although the impact in the overall method is very small. The Student **t** copula is discarded since its accuracy and performance are very similar to the Gaussian copula and the calibration of the ν parameter adds extra complexity. Therefore, as a general strategy, we conclude that the Gumbel copula is the most robust choice. When short maturities are considered, the Gaussian copula may be a satisfactory alternative.

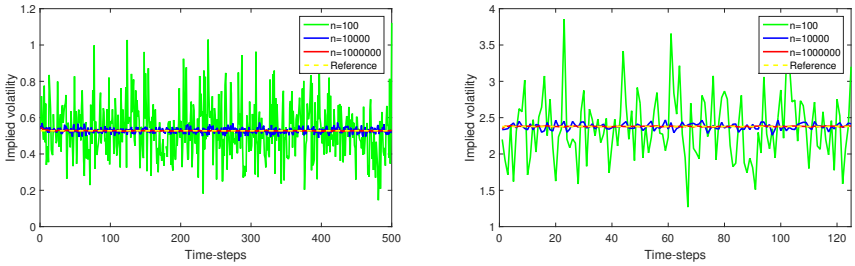
2.5.2. PRICING EUROPEAN OPTIONS BY THE ONE-STEP SABR METHOD

In Subsection 2.2.1, the Monte Carlo simulation of the SABR model was described. Now that all method components have been defined, we wish to test the one-step SABR simulation scheme by pricing European call options.

For the experiment, we consider the data sets in Table 2.2, the strike values K_i are chosen following the expression by Piterbarg, in [106],

$$K_i(T) = S(0)\exp(0.1 \times \sqrt{T} \times \delta_i),$$
$$\delta_i = -1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5.$$

As usual, the option prices are obtained by averaging the maximum of the forward asset values at maturity minus the strike and zero, i.e. $\max(S(T) - K_i(T), 0)$ (call option). Subsequently, the Black-Scholes formula is inverted to determine the *implied volatilities*. Note that, once the forward asset is simulated by our method, we can price options at multiple strikes simultaneously.



(a) Set I (b) Set II

Figure 2.5: Monte Carlo Milstein convergence test. Strike K_1.

We will compare the implied volatilities obtained by means of different approaches: Hagan formula, Monte Carlo simulation with a Milstein discretization and the one-step SABR method. Milstein-based Monte Carlo simulation is employed here as a reference for our one time-step simulation. In [21], the authors have shown that the Milstein scheme is relatively stable when applied for the SABR model compared to the Euler-Maruyama or log-Euler schemes. In Figure 2.5, the convergence of the Milstein Monte Carlo simulation is numerically shown. Figure 2.5(a) gives convergence results for Set I, where an analytic solution is given by Antonov [4] ($\rho = 0$). In Figure 2.5(b), the Monte Carlo convergence for Set II is depicted (in [86], Korn and Tang provide a reference price for $\beta = 0$). In order to have a reliable reference, we set $n = 1,000,000$ paths and $250 \times T$ time-steps for the pricing experiment.

Now, we analyse the accuracy and the performance of the one-step SABR method with the Gaussian and Gumbel copulas. In Table 2.5, the convergence of our method when the number of samples, n, is increased is empirically shown. We present the mean and the standard deviations of the error in basis points between the implied volatilities given by our one-step method and the reference price (Antonov) when Set I and K_1 are employed. We observe a reduction in the error (both mean and standard deviation) according to the expected Monte Carlo ratio ($1/\sqrt{n}$). Regarding the computational time, also in Table 2.5, the execution times of the one-step SABR method are shown. We can see that, up to $n = 100,000$, the number of simulations does not affect the performance. Also when larger numbers of samples are required, the computational time of the one time-step method does not increase drastically. As stated before, sampling with the Gumbel copula is somewhat slower than with the Gaussian copula, but the overall difference is negligible. The parameter M, the number of discrete time points, is not varied here because, as stated in Subsection 2.3.3, it has no significant impact on the execution time and can be set to a very high value.

To further test the one-step SABR method, in Table 2.6, the differences (in basis points) between the obtained implied volatilities with different considered methods and several strikes are presented. The number of paths is set to $n = 1,000,000$.

As the pricing experiments show, our copula-based one-step method achieves a very high accuracy. Furthermore, since it is a one time-step technique, it is a fast (≈ 0.5 *sec.*)

	$n = 1000$	$n = 10000$	$n = 100000$	$n = 1000000$
	Gaussian (Set I, K_1)			
Error	519.58(204.02)	132.39(68.03)	37.42(16.55)	16.23(7.66)
Time	0.3386	0.3440	0.3857	0.5733
	Gumbel (Set I, K_1)			
Error	151.44(199.36)	−123.76(86.33)	34.14(17.03)	11.59(6.58)
Time	0.3492	0.3561	0.3874	0.6663

Table 2.5: Convergence in n: mean and standard deviation of the error (basis points) and time (*sec.*).

Strikes	K_1	K_2	K_3	K_4	K_5	K_6	K_7
	Set I (Reference: Antonov [4])						
Hagan	55.07	52.34	50.08	N/A	47.04	46.26	45.97
MC	23.50	21.41	19.38	N/A	16.59	15.58	14.63
Gaussian	16.23	20.79	24.95	N/A	33.40	37.03	40.72
Gumbel	11.59	15.57	19.12	N/A	25.41	28.66	31.79
	Set II (Reference: Korn [86])						
Hagan	−558.82	−492.37	−432.11	−377.47	−327.92	−282.98	−242.22
MC	5.30	6.50	7.85	9.32	10.82	12.25	13.66
Gaussian	9.93	9.98	10.02	10.20	10.57	10.73	11.04
Gumbel	−9.93	−9.38	−8.94	−8.35	−7.69	−6.83	−5.79
	Set III (Reference: MC Milstein)						
Hagan	287.05	252.91	220.39	190.36	163.87	141.88	126.39
Gaussian	16.10	16.76	16.62	15.22	13.85	12.29	10.67
Gumbel	6.99	3.79	0.67	−2.27	−5.57	−9.79	−14.06

Table 2.6: One-step SABR method with Gaussian and Gumbel copulas - Implied volatility: differences in basis points.

alternative for pricing of European options up to maturities of two years under the SABR model. Our one-step approach works particularly well in the case of low strike values and higher volatilities, that pose some problems for the Hagan implied volatility formula.

2.6. CONCLUSIONS

In this chapter, an efficient method to obtain samples of the SABR dynamics based on the time-integrated variance has been developed. The technique employs a Fourier method to derive the CDF of the time-integrated variance. By means of copulas, the conditional sampling technique is obtained. Its application gives us a fast and accurate *one time-step* Monte Carlo method for the SABR model simulation. By numerical experiments, we have shown that our method does not suffer from well-known problems of the Hagan formula in the case of small strike values and higher volatilities.

A generalization of our one-step SABR method to a multiple time-step version will be presented in the next chapter. This will allow us to deal with problems with longer maturities (more than two years) and also with more involved exotic options (early-exercise and path-dependent options). We need to introduce new method components to deal with the increasing computational complexity in that case.

The hybrid aspect of the method proposed in this chapter is represented by the combined use of Monte Carlo and Fourier techniques. This fact has allowed us to develop an accurate and highly efficient one time-step simulation scheme that can be used for calibration purposes, where Monte Carlo-based methods are typically discarded.

CHAPTER 3

Multiple time-step Monte Carlo simulation of the SABR model

In this chapter, we will present the multiple time-step Monte Carlo simulation technique for pricing options under the SABR model. The proposed method is an extension of the one time-step Monte Carlo method that we proposed in the previous chapter, for pricing European options in the context of the model calibration. A highly efficient method results, with many highly interesting and non-trivial components, like Fourier inversion for the sum of log-normals, stochastic collocation, Gumbel copula, correlation approximation, that are not yet seen in combination within a Monte Carlo simulation. The present multiple time-step Monte Carlo method is especially useful for long-term options and for exotic options.

3.1. INTRODUCTION

The SABR model [67] is an established SDE model which, in practice, is often used for interest rates and FX modelling. It is based on a parametric local volatility component in terms of a model parameter, β, and reads

$$\begin{aligned} \mathrm{d}S(t) &= \sigma(t)S^{\beta}(t)\mathrm{d}W_S(t), \quad & S(0) &= S_0 \exp(rT), \\ \mathrm{d}\sigma(t) &= \alpha\sigma(t)\mathrm{d}W_\sigma(t), & \sigma(0) &= \sigma_0. \end{aligned} \tag{3.1}$$

Here $S(t) = \bar{S}(t)\exp(r(T-t))$ denotes the forward price of the underlying asset $\bar{S}(t)$, with r an interest rate, S_0 the spot price and T the maturity. Further, $\sigma(t)$ represents a stochastic volatility process, with $\sigma(0) = \sigma_0$, $W_S(t)$ and $W_\sigma(t)$ are correlated Brownian motions with constant correlation coefficient ρ (i.e. $W_S W_\sigma = \rho t$). The parameters of the SABR model are $\alpha > 0$ (the volatility of volatility, *vol-vol*), $0 \leq \beta \leq 1$ (the variance elasticity) and ρ (the correlation coefficient).

By the one time-step SABR method in Chapter 2, we can efficiently calculate accurate implied volatilities for any strike, however, only for short time to maturity (less than two years). This fits well to the context of FX markets. Here, we extend the one time-step Monte Carlo method and propose *a multiple time-step Monte Carlo simulation technique* for pricing options under the SABR dynamics. We call it the *mSABR method*. With this new simulation approach, we can price options with longer maturities, payoffs relying on the transitional distributions and exotic options (like path-dependent options) under the SABR dynamics.

This chapter is based on the article "On an efficient multiple time-step Monte Carlo simulation of the SABR model". Published in *Quantitative Finance*, 2017 [89].

Because a robust and efficient SABR simulation scheme may be involved and quite expensive, we aim for using as few time-steps as possible. This fact, together with the long time horizon assumption, implies that we focus on larger time-steps. In this context, the important research line is called *exact simulation*, where, rather than Taylor-based simulation techniques, the PDF of the SDE under consideration is highly accurately approximated. Point-of-departure is Broadie and Kaya's exact simulation for the Heston SV model [15]. That method is however time-consuming because of multiple evaluations of Bessel functions and the use of Fourier inversion techniques on each Monte Carlo path. Inspired by this approach, in [124] and [130], computational improvements were proposed. Furthermore, Andersen in [2] presented two efficient alternatives to the Broadie and Kaya scheme, the so-called Truncated Gaussian (TG) and the Quadratic Exponential (QE) schemes.

For the SABR model, exact Monte Carlo simulation is nontrivial because the forward process in (3.1) is governed by CEV dynamics [31, 117]. In [75], the connection between the CEV and a squared Bessel processes was explained, and an analytic approximation for the SABR conditional distribution based on the non-central χ^2 distribution was presented. The expression relies on the stochastic volatility dynamics, but also on the *time-integrated variance* process (which itself also depends on the volatility).

An (approximately) exact SABR simulation can then be subdivided in the simulation of the volatility process, the simulation of the time-integrated variance (conditional on the volatility process) and the simulation of the underlying CEV process (conditional on the volatility and the time-integrated variance processes). Based on this, Chen et al. proposed in [21] a low-biased Monte Carlo scheme with moment-matching (following the techniques of the QE scheme by Andersen in [2]) and a direct inversion approach. For the simulation of the time-integrated variance, they also employed moment-matching based on conditional moments that were approximated by a small disturbance expansion. In [78], Kennedy and Pham provided the moments of the time-integrated variance for specific SABR models (like the normal, log-normal and displaced diffusion SABR models) and derived an approximation of the SABR distribution. In [95], a summary of different approaches was presented.

In this chapter, we approximate[1] the distribution of the time-integrated variance conditional on the volatility dynamics by joining the two involved marginal distributions, i.e. the stochastic volatility and time-integrated variance distributions, by means of a copula technique. With both marginal distributions available, we use the Gumbel copula to define a multivariate distribution which approximates the conditional distribution of the time-integrated variance given the stochastic volatility.

An approximation of the CDF of the time-integrated variance can be obtained by a recursive procedure, as described in [137], originally employed to price arithmetic Asian options. This iterative technique is based on the derivation of the ChF of the time-integrated variance process and Fourier inversion to recover the PDF.

The fact that we need to apply recursion and compute the corresponding ChF, PDF and CDF of the time-integrated variance for each Monte Carlo sample at each time-step, makes this approach however computationally very expensive (even with the improvements already made in Chapter 2), like the Heston exact simulation. In order to reduce

[1] An analytical expression exists but is not practically tractable.

the use of this procedure as much as possible, highly efficient sampling from the CDF of the time-integrated variance is proposed, which is based on the so-called *Stochastic Collocation Monte Carlo sampler* (SCMC), see [65]. The technique employs a sophisticated interpolation (based on Lagrange polynomials and collocation points) of the CDF under consideration.

As will be seen, the resulting 'almost exact' Monte Carlo SABR simulation method that we present here (the mSABR method), contains several interesting (and not commonly used) components, like the Gumbel copula, a recursion plus Fourier inversion to approximate the CDF of the time-integrated variance, and efficient interpolation by means of the SCMC sampler. The proposed mSABR scheme allows for fast and accurate option pricing under the SABR dynamics, providing a better ratio of accuracy and computational cost than Taylor-based simulation schemes. Compared to the other approaches mentioned in this introduction, we provide a highly accurate SABR simulation scheme, based on only a few time-steps.

The chapter is organized as follows. In Section 3.2, SABR model simulation is briefly introduced. In Section 3.3, the different parts of the multiple time-step copula-based technique are described, including the derivation of the marginal distributions and the application of the copula. Some numerical experiments are presented in Section 3.4. We conclude in Section 3.5.

3.2. 'ALMOST EXACT' SABR SIMULATION

The SABR model with dependent Brownian motions was already given in Equation (3.1). The multiple time-step Monte Carlo SABR simulation is based on the corresponding SDE system with independent Brownian motions $\mathrm{d}\tilde{W}_S(t)$ and $\mathrm{d}\tilde{W}_\sigma(t)$, i.e.

$$\begin{aligned} \mathrm{d}S(t) &= \sigma(t)S^\beta(t)\left(\rho\mathrm{d}\tilde{W}_\sigma(t)+\sqrt{1-\rho^2}\mathrm{d}\tilde{W}_S(t)\right), \quad S(0)=S_0\exp(rT), \\ \mathrm{d}\sigma(t) &= \alpha\sigma(t)\mathrm{d}\tilde{W}_\sigma(t), \qquad\qquad\qquad\qquad\quad \sigma(0)=\sigma_0. \end{aligned} \tag{3.2}$$

The forward dynamics are governed by a CEV process. Based on this fact and the works by Schroder [117] and Islah [75], an analytic approximation for the CDF of the SABR conditional process has been obtained. For some generic time interval $[s,t]$, $0\leq s<t\leq T$, assuming $S(s)>0$, the conditional cumulative distribution for forward $S(t)$ with an absorbing boundary at $S(t)=0$, given $\sigma(s)$, $\sigma(t)$ and $\int_s^t\sigma^2(z)\mathrm{d}z$, is given by

$$Pr\left(S(t)\leq K|S(s)>0,\sigma(s),\sigma(t),\int_s^t\sigma^2(z)\mathrm{d}z\right)=1-\chi^2(a;b,c), \tag{3.3}$$

where

$$\begin{aligned} a &= \frac{1}{\nu(t)}\left(\frac{S(s)^{1-\beta}}{(1-\beta)}+\frac{\rho}{\alpha}(\sigma(t)-\sigma(s))\right)^2, & c &= \frac{K^{2(1-\beta)}}{(1-\beta)^2\nu(t)}, \\ b &= 2-\frac{1-2\beta-\rho^2(1-\beta)}{(1-\beta)(1-\rho^2)}, & \nu(t) &= (1-\rho^2)\int_s^t\sigma^2(z)\mathrm{d}z, \end{aligned}$$

and $\chi^2(x;\delta,\lambda)$ is the non-central chi-square CDF.

This formula is exact in the case of $\rho = 0$ and constitutes an *approximation* otherwise (because the CEV process is approximated by a shifted process with an approximated initial condition for $\rho \neq 0$ in the derivation). Based on Equation (3.3), an *approximately* exact simulation of SABR model is feasible by inverting the conditional SABR cumulative distribution when the conditional time-integrated variance is known, see Section 3.2.1.

3.2.1. SABR MONTE CARLO SIMULATION

In order to apply an 'almost exact' multiple time-step Monte Carlo method for the SABR model, several steps need to be performed, that are described in the following:

- *Simulation of the SABR volatility process, $\sigma(t)$ given $\sigma(s)$.* By Equation (3.2), the stochastic volatility process of the SABR model exhibits a log-normal distribution. The solution is a GBM, i.e. the exact simulation of $\sigma(t)|\sigma(s)$ reads

$$\sigma(t) \sim \sigma(s) \exp\left(\alpha \tilde{W}_\sigma(t) - \frac{1}{2}\alpha^2(t-s)\right). \tag{3.4}$$

- *Simulation of the SABR time-integrated variance process, $\int_s^t \sigma^2(z)\mathrm{d}z|\sigma(s),\sigma(t)$.* This conditional distribution is not efficiently tractable. We will therefore derive an approximation of the conditional distribution of the SABR time-integrated variance given $\sigma(t)$ and $\sigma(s)$. The time-integrated variance sampling can be done by simply inverting it.

- *Simulation of the SABR forward price process.* The forward price $S(t)$ can be simulated by inverting the CDF in Equation (3.3). By this, we avoid negative forward prices in the simulation, as an absorbing boundary at zero is considered. There is no analytic expression for the inverse distribution and therefore this inversion has to be computed by means of some numerical approximation.

The multiple time-step Monte Carlo simulation for the SABR model is thus summarized in Algorithm 2. Input parameters are the initial conditions (S_0 and σ_0), the maturity time, T, the SABR parameters α, β and ρ, the number of Monte Carlo samples (n) and the number of time-steps (m).

Algorithm 2: Multiple time-step SABR Monte Carlo simulation.

Data: $S_0, \sigma_0, T, \alpha, \beta, \rho, n, m$.
$S(0) = S_0 \exp(rT)$, $\sigma(0) = \sigma_0$
for *Path* $p = 1 \ldots n$ **do**
  for *Step* $k = 1 \ldots m$ **do**
    $s = (k-1)\frac{T}{m}$
    $t = k\frac{T}{m}$
    Sampling $\sigma(t)$ given $\sigma(s)$.
    Sampling $\int_s^t \sigma^2(z)\mathrm{d}z$ given $\sigma(s)$ and $\sigma(t)$.
    Sampling $S(t)$ given $S(s)$, $\sigma(s)$, $\sigma(t)$ and $\int_s^t \sigma^2(z)\mathrm{d}z$.

In Section 3.3.4, we describe the procedure to sample $\int_s^t \sigma^2(z)\mathrm{d}z|\sigma(s),\sigma(t)$. This represents the main difference with respect to the technique presented in Chapter 2, where the conditionality is solely on the final volatility (the initial volatility is constant). The double condition implies that the marginal distributions that will be employed within the copula are conditional distributions themselves. Although this is, conceptually, a minor change, it has a drastic impact on the efficiency of the procedure, increasing the computational cost.

As a copula approach is again employed, the CDF of the time-integrated variance given the initial volatility, $\sigma(s)$, (as a marginal distribution) must be derived. In Section 3.3.1, a recursive technique to obtain an approximation of this CDF is presented. Because we need to apply this recursion to approximate the ChF, the PDF and the CDF of $\int_s^t \sigma^2(z)\mathrm{d}z|\sigma(s)$ for each sample of $\sigma(s)$ at each time-step, this multiple time-step approach is very expensive in terms of execution time. To overcome this drawback, an efficient alternative will be employed here, based on Lagrange interpolation, as in the SCMC sampler [65]. In Section 3.2.2, this methodology is briefly described.

3.2.2. STOCHASTIC COLLOCATION MONTE CARLO SAMPLER

In a multiple time-step exact simulation Monte Carlo method we need to perform a significant number of computations each time-step on each Monte Carlo path. This often hampers the applicability of the exact simulation for systems of SDEs. One of the important components of the multiple time-step Monte Carlo SABR simulation is therefore an accurate and highly efficient interpolation, as proposed in the SCMC sampler in [65].

In this section, we will introduce the SCMC technique. The details in the SABR conditional time-integrated variance context will be presented in Section 3.3.2.

The SCMC technique relies on the property that a CDF of a distribution (if it exists) is uniformly distributed. A well-known standard approach to sample from a given distribution, Y, with CDF F_Y, reads

$$F_Y(Y) \overset{\mathrm{d}}{=} U \quad \text{thus} \quad y_n = F_Y^{-1}(u_n),$$

where $\overset{\mathrm{d}}{=}$ means equality in distributional sense, $U \sim \mathscr{U}([0,1])$ and u_n are samples from $\mathscr{U}([0,1])$. The computational cost of this approach highly depends on the cost of the inversion F_Y^{-1}, which is assumed to be expensive.

We therefore consider another, 'cheap', random variable X, whose inversion, F_X^{-1}, is computationally much less expensive. In this framework, the following relation holds

$$F_Y(Y) \overset{\mathrm{d}}{=} U \overset{\mathrm{d}}{=} F_X(X).$$

The samples y_n of Y, and x_n of X, are thus related via the following inversion,

$$y_n = F_Y^{-1}(F_X(x_n)).$$

However, this does not yet imply any improvement since the number of expensive inversions F_Y^{-1} remains the same. The goal is to compute y_n by using a function $g(\cdot) = F_Y^{-1}(F_X(\cdot))$, such that

$$F_X(x) = F_Y(g(x)) \quad \text{and} \quad Y \overset{\mathrm{d}}{=} g(X),$$

where evaluations of function $g(\cdot)$ do not require many inversions $F_Y^{-1}(\cdot)$.

In [65], function $g(\cdot)$ is approximated by means of Lagrange interpolation, which is a well-known interpolation also used in the Uncertainty Quantification (UQ) context. The result is a polynomial, $g_{N_Y}(\cdot)$, which approximates function $g(\cdot) = F_Y^{-1}(F_X(\cdot))$, and the samples y_n can be obtained by employing $g_{N_Y}(\cdot)$ as

$$y_n \approx g_{N_Y}(x_n) = \sum_{i=1}^{N_Y} y_i \ell_i(x_n), \quad \ell_i(x_n) = \prod_{j=1, j\neq i}^{N_Y} \frac{x_n - \bar{x}_j}{\bar{x}_i - \bar{x}_j},$$

where x_n is a vector of samples from X and $\bar{x}_i$ and $\bar{x}_j$ are so-called *collocation points*. N_Y represents the number of collocation points and y_i the exact inversion value of F_Y at the collocation point $\bar{x}_i$, i.e. $y_i = F_Y^{-1}(F_X(\bar{x}_i))$. By applying this interpolation, the number of inversions is reduced and, with only N_Y expensive inversions $F_Y^{-1}(F_X(\bar{x}_i))$, we can generate any number of samples by evaluating the polynomial $g_{N_Y}(x_n)$. This constitutes the 1D version of SCMC sampler in [65].

According to the definition of the SCMC technique, a crucial aspect for the computational cost is parameter N_Y: the fewer collocation points are required, the more efficient will be the sampling procedure. The collocation points must be *optimally* chosen in a way to minimize their number. For any random variable X, the optimal collocation points will be based on the moments of X. The optimal collocation points are here chosen to be Gauss quadrature points that are defined as the zeros of the related orthogonal polynomial. This approach leads to a stable interpolation under the probability distribution of X. The complete procedure to compute the collocation points is described in [65].

In Section 3.3.2, the application of the SCMC technique to generate samples from the CDF of $\int_s^t \sigma^2(z)\mathrm{d}z|\sigma(s)$ is presented. Since we deal with a conditional distribution, the 2D version of SCMC needs to be used.

3.3. COMPONENTS OF THE MSABR METHOD

In this section, we will discuss the different components of the mSABR method. Again, hereafter, we denote the SABR's time-integrated variance process by $Y(s,t) := \int_s^t \sigma^2(z)\mathrm{d}z$. We will explain how to efficiently sample the time-integrated variance given the initial and the final volatility, i.e. $Y(s,t)|\sigma(t),\sigma(s)$, as well as its use in a complete SABR simulation.

In Section 3.3.4, we employ an accurate sampling method based on copula theory [122], which is employed to approximate the required conditional distributions. The copula relies on the availability of the *marginal* distributions to simulate the *joint* distribution. As the marginal distributions, $Y(s,t)|\sigma(s)$ and $\sigma(t)|\sigma(s)$ appear as the natural choices. In Section 3.3.1, a procedure to recover the CDF of the time-integrated variance process given the initial volatility is presented.

3.3.1. CDF OF $\int_s^t \sigma^2(z)\mathrm{d}z|\sigma(s)$ USING THE COS METHOD

We present a technique to approximate the CDF of $Y(s,t)|\sigma(s)$, i.e. $F_{Y|\sigma(s)}$. We will work in the log-space, so an approximated CDF of $\log Y(s,t)|\log\sigma(s)$, $F_{\log Y|\log\sigma(s)}$, will be derived. We have employed this technique in the one time-step SABR method (Section 2.3),

but the multiple time-step version is somewhat more involved. We approximate $Y(s,t)$ by its discrete equivalent, i.e.

$$Y(s,t) := \int_s^t \sigma^2(z)\mathrm{d}z \approx \sum_{j=1}^{M} \Delta t \sigma^2(t_j) =: \tilde{Y}(s,t), \tag{3.5}$$

where M is the number of intermediate or discrete time-points, $\Delta t = \frac{t-s}{M}$ and $t_j = s + j\Delta t$, $j = 1,\ldots,M$. So, we now have m large time-steps, for the multiple time-step Monte Carlo simulation, and M intermediate dates for the time-integrated variance, $M >> m$. We suggest to prescribe Δt, and use $M = \frac{t-s}{\Delta t}$, in order to avoid over-discretization issues. This makes the value of M dependent on the interval size, $t-s$. $\tilde{Y}(T)$ is subsequently transformed to the logarithmic domain, with

$$F_{\log\tilde{Y}|\log\sigma(s)}(x) = \int_{-\infty}^{x} f_{\log\tilde{Y}|\log\sigma(s)}(y)\mathrm{d}y, \tag{3.6}$$

and $f_{\log\tilde{Y}|\log\sigma(s)}$ the PDF of $\log\tilde{Y}(s,t)|\log\sigma(s)$.

Density function $f_{\log\tilde{Y}|\log\sigma(s)}$ is, in turn, recovered by approximating the associated ChF, $\hat{f}_{\log\tilde{Y}|\log\sigma(s)}$, and applying a Fourier inversion procedure.

RECURSIVE PROCEDURE TO RECOVER $\hat{f}_{\log\tilde{Y}|\log\sigma(s)}$

We follow a similar procedure to approximate $\hat{f}_{\log\tilde{Y}|\log\sigma(s)}$ as in Chapter 2. However, as stated before, an extra computational issues will appear. We start again by defining the sequence,

$$R_j = \log\left(\frac{\sigma^2(t_j)}{\sigma^2(t_{j-1})}\right), \quad j = 1,\ldots,M, \tag{3.7}$$

where R_j is the logarithmic increment of $\sigma^2(t)$ between t_j and t_{j-1}, $j = 1,\ldots,M$. As the volatility process follows log-normal dynamics, and increments of Brownian motion are independent and identically distributed, the R_j are also independent and identically distributed, i.e. $R_j \overset{\mathrm{d}}{=} R$. In addition, the ChF of R_j is well-known and reads, $\forall u, j$,

$$\hat{f}_{R_j}(u) = \hat{f}_R(u) = \exp(-iu\alpha^2\Delta t - 2u^2\alpha^2\Delta t), \tag{3.8}$$

with α as in (3.1) and $i = \sqrt{-1}$ the imaginary unit. By the definition of R_j in Equation (3.7), we write $\sigma^2(t_j)$ as

$$\sigma^2(t_j) = \sigma^2(t_0)\exp(R_1 + R_2 + \cdots + R_j). \tag{3.9}$$

At this point, a backward recursion procedure in terms of R_j will be defined by which we can recover $\hat{f}_{\log\tilde{Y}|\log\sigma(s)}$. We define

$$Y_1 = R_M, \quad Y_j = R_{M+1-j} + Z_{j-1}, \quad j = 2,\ldots,M, \tag{3.10}$$

with $Z_j = \log(1+\exp(Y_j))$.

By Equations (3.9) and (3.10), the discrete time-integrated variance can be expressed as

$$\tilde{Y}(s,t) = \sum_{i=1}^{M} \sigma^2(t_i)\Delta t = \Delta t \sigma^2(s) \exp(Y_M). \tag{3.11}$$

From Equation (3.11) and by applying the definition of ChF, we determine $\hat{f}_{\log \tilde{Y}|\log\sigma(s)}$, as follows

$$\begin{aligned}\hat{f}_{\log \tilde{Y}|\log\sigma(s)}(u) &= \mathbb{E}[\exp(iu\log\tilde{Y}(s,t))|\log\sigma(s)] \\ &= \exp\left(iu\log(\Delta t\sigma^2(s))\right)\mathbb{E}[\exp(iuY_M)|\log\sigma(s)] \\ &= \exp\left(iu\log(\Delta t\sigma^2(s))\right)\hat{f}_{Y_M}(u).\end{aligned} \tag{3.12}$$

As Y_M is defined recursively, its ChF can be obtained by a recursion as well. In Sections 2.3.1 and 2.3.2 of Chapter 2, we have already proposed a procedure to efficiently approximate $\hat{f}_{Y_M}$. The methodology is based on a Fourier cosine series expansion and the use of a numerical *piecewise linear* approach in the approximation of a computational expensive integral (Equation (2.20)). The experiments presented in Chapter 2 have empirically shown the good balance between accuracy and computational effort provided by this approach.

RECOVERING $f_{\log \tilde{Y}|\log\sigma(s)}$ BY COS METHOD

Once the approximation of $\hat{f}_{Y_M}$, denoted by $\hat{f}^*_{Y_M}$, has been efficiently derived, we can recover $f_{\log \tilde{Y}|\log\sigma(s)}$ from $\hat{f}_{\log \tilde{Y}|\log\sigma(s)}$ by employing the COS method [49], as follows

$$f_{\log \tilde{Y}|\log\sigma(s)}(x) \approx \frac{2}{\tilde{b}-\tilde{a}} \sum_{k=0}^{N-1\prime} C_k \cos\left((x-\tilde{a})\frac{k\pi}{\tilde{b}-\tilde{a}}\right), \tag{3.13}$$

with

$$C_k = \Re\left(\hat{f}_{\log \tilde{Y}|\log\sigma(s)}\left(\frac{k\pi}{\tilde{b}-\tilde{a}}\right)\exp\left(-i\frac{\tilde{a}k\pi}{\tilde{b}-\tilde{a}}\right)\right),$$

and

$$\begin{aligned}\hat{f}_{\log \tilde{Y}|\log\sigma(s)}\left(\frac{k\pi}{\tilde{b}-\tilde{a}}\right) &= \exp\left(i\frac{k\pi}{\tilde{b}-\tilde{a}}\log\left(\Delta t\sigma^2(s)\right)\right)\hat{f}_{Y_M}\left(\frac{k\pi}{\tilde{b}-\tilde{a}}\right) \\ &\approx \exp\left(i\frac{k\pi}{\tilde{b}-\tilde{a}}\log\left(\Delta t\sigma^2(s)\right)\right)\hat{f}^*_{Y_M}\left(\frac{k\pi}{\tilde{b}-\tilde{a}}\right),\end{aligned}$$

where N is the number of COS terms, $[\tilde{a},\tilde{b}]$ is the support[2] of $\log\tilde{Y}(s,t)|\log\sigma(s)$ and the prime $'$ and $\Re$ symbols in Equation (3.13) mean division of the first term in the summation by two and taking the real part of the complex-valued expressions in the brackets, respectively. CDF $F_{\log \tilde{Y}|\log\sigma(s)}$ can be obtained by integration, plugging the approximated $f_{\log \tilde{Y}|\log\sigma(s)}$ from Equation (3.13) into Equation (3.6).

The ChF $\hat{f}_{\log \tilde{Y}|\log\sigma(s)}$ and PDF $f_{\log \tilde{Y}|\log\sigma(s)}$ need to be derived for each sample of $\log\sigma(s)$ and this makes the computation of $F_{\log \tilde{Y}|\log\sigma(s)}$ (and also its subsequent inversion) very expensive. In order to overcome this issue, the SCMC technique (see Section 3.2.2) will be employed approximating these rather expensive computations by means of an accurate interpolation.

[2] It can be calculated given the support of Y_M, $[a,b]$.

3.3.2. EFFICIENT SAMPLING OF $\log \tilde{Y} | \log \sigma(s)$

By employing the SCMC technique, instead of directly computing $F_{\log \tilde{Y}|\log\sigma(s)}$ for each sample of $\log\sigma(s)$, we only have to compute $F_{\log \tilde{Y}|\log\sigma(s)}$ at the collocation points. In general, only a few collocation points are sufficient to obtain accurate approximations, which is a well-known fact from the UQ research field. This fact allows us to drastically reduce the computational cost of sampling the required distribution.

For the problem at hand, we require samples from the time-integrated variance conditional on the initial volatility, $\log \tilde{Y}(s,t)|\log\sigma(s)$. Therefore, we need to make use of the *2D version of the SCMC technique*. Two levels of collocation points need to be chosen, one for each dimension. If we denote them by $N_{\tilde{Y}}$ and N_σ, respectively, the resulting number of inversions equals $N_{\tilde{Y}} \cdot N_\sigma$. The formal definition of the 2D SCMC technique applied in our context reads

$$y_n|v_n \approx g_{N_{\tilde{Y}},N_\sigma}(x_n) = \sum_{i=1}^{N_{\tilde{Y}}} \sum_{j=1}^{N_\sigma} F^{-1}_{\log \tilde{Y}|\log\sigma(s)=\bar{v}_j}(F_X(\bar{x}_i))\ell_i(x_n)\ell_j(v_n), \tag{3.14}$$

where x_n are the samples from $X \sim \mathcal{N}(0,1)$, which is used as the cheap variable, and v_n the samples of $\log\sigma(s)$; $\bar{x}_i$ and $\bar{v}_j$ are the collocation points for approximating variables $\log Y$ and $\log\sigma(s)$, respectively. The Lagrange polynomials ℓ_i and ℓ_j are defined by

$$\ell_i(x_n) = \prod_{k=1,k\neq i}^{N_{\tilde{Y}}} \frac{x_n - \bar{x}_k}{\bar{x}_i - \bar{x}_k}, \quad \ell_j(v_n) = \prod_{k=1,k\neq j}^{N_\sigma} \frac{v_n - \bar{v}_k}{\bar{v}_i - \bar{v}_k}.$$

The collocation points $\bar{x}_i$ and $\bar{v}_j$ are calculated based on the moments of the random variables. The two involved variables in the application of the 2D SCMC technique are the collocation variables, $X \sim \mathcal{N}(0,1)$, and $\log\sigma(s) \sim \mathcal{N}(\log\sigma_0 + \frac{1}{2}\alpha^2 s, \alpha\sqrt{s})$. The moments of a normal variable, $X \sim \mathcal{N}(\mu_X, s_X)$ are analytically available, and are given by

$$\mathbb{E}\left[X^p\right] = \begin{cases} 0 & \text{if } p \text{ is odd,} \\ s_X^p (p-1)!! & \text{if } p \text{ is even,} \end{cases}$$

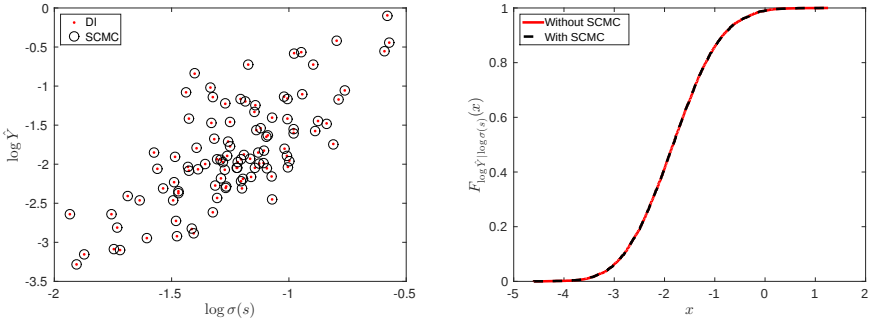
where p is the moment order and the expression !! represents the *double factorial*. In order to compute the optimal collocation points, $\bar{v}_j$, the precalculated collocation points by means of the $\log\sigma(s)$ moments need to be shifted according to the mean, i.e. $\log\sigma_0 + \frac{1}{2}\alpha^2 s$.

We first test numerically the accuracy of the SCMC technique for our problem. In Figure 3.1, two experiments are presented. In Figure 3.1(a), we compare the samples obtained by direct inversion and by the SCMC technique. The fit is highly accurate, already with only seven collocation points. In Figure 3.1(b), the empirical CDFs are depicted, where the excellent match is confirmed.

In order to show the performance of the SCMC technique for the simulation of the distribution of $\log \tilde{Y}|\log\sigma(s)$, the execution times of generating different numbers of samples are presented in Table 3.1.

3.3.3. ERROR ANALYSIS

We have proposed a method to compute the approximated CDF for the conditional time-integrated variance, i.e. $Y(s,t)|\sigma(s)$. In this section, the different sources of error due to



(a) $\log \tilde{Y}|\log\sigma(s)$ - Direct inversion vs. SCMC. (b) $F_{\log \tilde{Y}|\log\sigma(s)}(x)$.

Figure 3.1: (a) Points cloud $\log \tilde{Y}|\log\sigma(s)$ by direct inversion (DI) sampling (red dots) and SCMC sampling (black circles). (b) Empirical CDFs of $\log Y|\log\sigma(s)$ with and without using SCMC.

Samples	Without SCMC	With SCMC		
		$N_{\tilde{Y}} = N_\sigma = 3$	$N_{\tilde{Y}} = N_\sigma = 7$	$N_{\tilde{Y}} = N_\sigma = 11$
100	1.0695	0.0449	0.0466	0.0660
10000	16.3483	0.0518	0.0588	0.0798
1000000	1624.3019	0.2648	0.5882	1.0940

Table 3.1: SCMC time in seconds.

the approximations made in the Sections 3.3.1 and 3.3.2 are analysed, indicating how to reduce or bound the errors. The sources of error in the approximation include ϵ_D, the discretization of $Y(s,t)$ in Equation (3.5), ϵ_T, the truncation in Equations (2.16) and (3.13), ϵ_C, due to the cosine expansion in Equations (2.16) and (3.13), ϵ_M, the numerical computation of $I_{\mathcal{M}}$ in Equation (2.23) and ϵ_S due to the application of SCMC in Equation (3.14). The errors ϵ_T, ϵ_C and ϵ_M were already studied in Section 2.3.3 of Chapter 2. The error ϵ_D is updated according to the multiple time-step version.

Error ϵ_D. This error occurs because SABR's time-integrated variance is approximated by its discrete equivalent, i.e. $Y(s,t) \approx \tilde{Y}(s,t)$. Note that the process $Y(s,t) = \int_s^t \sigma^2(z)\mathrm{d}z$ is the solution of the SDE

$$\mathrm{d}Y(s,z) = \sigma^2(z)\mathrm{d}z, \quad Y(s,s) = 0,$$

which, by employing the same time discretization as for $\tilde{Y}(s,t)$ in Equation (3.5), can be written as

$$\bar{Y}(s,t_j) - \bar{Y}(s,t_{j-1}) = \sigma^2(t_{j-1})\Delta t.$$

This is again the Euler-Maruyama discretization scheme. Indeed, it is easy to see that, if we sum the right-hand side of the equality over the M discrete time points, we

recover the discrete approximation $\tilde{Y}(s,t)$, as follows

$$\sum_{j=1}^{M} \sigma^2(t_{j-1})\Delta t = \sum_{j=1}^{M} \left(\bar{Y}(s,t_j) - \bar{Y}(s,t_{j-1})\right) = \bar{Y}(s,t_M) - \bar{Y}(s,t_0) = \tilde{Y}(s,t).$$

Then, error ϵ_D can be computed and bounded by employing the usual strong convergence expression as

$$\epsilon_D := \mathbb{E}\left[\left|\tilde{Y}(s,t) - Y(s,t)\right|\right] \leq C\left(\frac{t-s}{M}\right)^{\gamma},$$

for some constant $C > 0$.

In the actual experiments, we consider $\log \tilde{Y}(s,t)$ and $\log Y(s,t)$. This will reduce the error further.

Error ϵ_S. This error appears in the application of the interpolation-based SCMC technique to get samples of $\log \tilde{Y}|\log\sigma(s)$. In [65], where the SCMC sampler was proposed, an error analysis of the technique was carried out. Here we summarize the most important results related to our context and refer the reader to the original paper for further details.

To measure the error which results from the collocation method in a more general case (see Section 3.2.2), we can consider either the difference between functions $g(X)$ and $g_N(X)$ (X the cheap variable) or the error associated with the approximated CDF. The first type of error is due to Lagrange interpolation, so the corresponding error estimate is well-known. The error $\epsilon(\xi_n)$ with N_Y collocation points given by the standard Lagrange interpolation reads

$$\epsilon_X(\xi_n) = \left|g(\xi_n) - g_{N_Y}(\xi_n)\right| = \left|\frac{1}{N_Y!}\frac{\partial^{N_Y} g(x)}{\partial x^{N_Y}}\Big|_{x=\xi} \prod_{i=1}^{N_Y}(\xi_n - \bar{x}_i)\right|, \tag{3.15}$$

where $\bar{x}_i$ are the collocation points, and $\tilde{\xi} \in [\min(\mathbf{x}), \max(\mathbf{x})]$, $\mathbf{x} = (\bar{x}_1, \ldots, \bar{x}_{N_{\tilde{Y}}})^T$. The error can be bounded by defining ξ as the point where $\left|\partial^{N_Y} g(x)/\partial x^{N_Y}\right|$ has its maximum. A small probability of large errors *in the tails* can be observed by deriving the error $\epsilon_U(\xi_n)$, by substituting the uniformly distributed random variable u_n in the previous equation, using $\xi_n = F_X^{-1}(u_n)$,

$$\epsilon_U(u_n) = \left|g(F_X^{-1}(u_n)) - g_{N_Y}(F_X^{-1}(u_n))\right| = \left|\frac{1}{N_Y!}\frac{\partial^{N_Y} g(x)}{\partial x^{N_Y}}\Big|_{x=\xi} \prod_{i=1}^{N_Y}(F_X^{-1}(u_n) - \bar{x}_i)\right|.$$

A "close-to-linear" relation between the involved stochastic variables, meaning that $\partial^{N_Y} g(x)/\partial x^{N_Y}$ is small, gives a small approximation error. On the other hand, when approximating the CDF of the expensive variable Y, we have $F_Y(y) = F_Y(g(x)) \approx F_Y(g_{N_Y}(x))$, which is exact at the collocation points $\bar{x}_i$,

$$F_Y(y_i) = F_Y(g(\bar{x}_i)) = F_Y(g_{N_Y}(\bar{x}_i)).$$

A deeper analysis can be found in the original paper of SCMC [65], including a theoretical convergence in L^2.

We have chosen X to be standard normal, and $Y = \log \tilde{Y} | \log \sigma(s)$ can be seen as the logarithm of a sum of log-normal variables. Therefore, it is a "close-to-normal" distribution which can be efficiently approximated by a linear combination of standard normal distributions. In order to measure the error when SCMC is applied to the sampling of $\int_s^t \sigma^2(z)\mathrm{d}z|\sigma(s)$ (see Section 3.3.2), we can consider the difference between the functions $g(x) = F^{-1}_{\log \tilde{Y}|\log\sigma(s)}(F_X(x))$ and $g_{N_{\tilde{Y}},N_\sigma}(x)$ in Equation (3.14). By following the same derivation as in Equation (3.15), we have

$$\epsilon_S := \left| g(\xi_n) - g_{N_{\tilde{Y}},N_\sigma}(\xi_n) \right| = \left| \frac{1}{N_{\tilde{Y}}!} \frac{\partial^{N_{\tilde{Y}}} g(x)}{\partial x^{N_{\tilde{Y}}}} \Big|_{x=\xi} \prod_{i=1}^{N_{\tilde{Y}}} (\xi_n - \bar{x}_i) \prod_{j=1}^{N_\sigma} (\upsilon_n - \bar{\upsilon}_j) \right|,$$

where $N_{\tilde{Y}}$ and N_σ are the number of collocation points in each dimension and $\bar{\upsilon}_j$ are the collocation points corresponding to the conditional random variable $\log\sigma(s)$.

3.3.4. COPULA-BASED SIMULATION OF $\int_s^t \sigma^2(z)\mathrm{d}z|\sigma(t),\sigma(s)$

In this section we present the algorithm for the simulation of the time-integrated variance given $\sigma(t)$ and $\sigma(s)$ by means of a copula. In order to obtain the joint distribution, we require the marginal distributions. In our case, the required CDFs are $F_{\log Y|\log\sigma(s)}$ and $F_{\log\sigma(t)|\log\sigma(s)}$. In Section 3.3.1, an approximated CDF of $\log Y|\log\sigma(s)$, $F_{\log\tilde{Y}|\log\sigma(s)}$, has been derived. Since $\sigma(t)$ follows a log-normal distribution, by definition, $\log\sigma(t)$ is normally distributed, and the conditional process $\log\sigma(t)$ given $\log\sigma(s)$, follows a normal distribution,

$$\log\sigma(t)|\log\sigma(s) \sim \mathcal{N}\left(\log\sigma(s) - \frac{1}{2}\alpha^2(t-s), \alpha\sqrt{t-s}\right). \tag{3.16}$$

PEARSON'S CORRELATION COEFFICIENT

For any copula some measure of the correlation between the marginal distributions is needed. We will employ the Pearson correlation coefficient for $\log Y(s,t)$ and $\log\sigma(t)$. For this quantity, an approximated analytic formula can be derived. By definition, we have

$$\mathscr{P}_{\log Y,\log\sigma(t)} = \mathrm{corr}\left[\log\int_s^t \sigma^2(z)\mathrm{d}z, \log\sigma(t)\right] = \frac{\mathrm{cov}\left[\log\int_s^t \sigma^2(z)\mathrm{d}z, \log\sigma(t)\right]}{\sqrt{\mathbb{V}\mathrm{ar}\left[\log\int_s^t \sigma^2(z)\mathrm{d}z\right]\mathbb{V}\mathrm{ar}\left[\log\sigma(t)\right]}}.$$

We employ the following approximation

$$\log\int_s^t \sigma^2(z)\mathrm{d}z \approx \int_s^t \log\sigma^2(z)\mathrm{d}z = 2\int_s^t \log\sigma(z)\mathrm{d}z,$$

where the logarithm and the integral are interchanged. Since the log function is concave, this approximation forms a lower bound for the true value. This can be seen by applying Jensen's inequality, i.e.

$$\log\int_s^t \sigma^2(z)\mathrm{d}z \geq \int_s^t \log\sigma^2(z)\mathrm{d}z.$$

It has been numerically shown that, under the model settings with an interval size $t-s$ less than two years, the results based on this approximation are highly satisfactory (further details in Chapter 2, Section 2.4.2). The correlation coefficient can then be approximated by

$$\mathscr{P}_{\log Y(T),\log\sigma(t)} \approx \frac{\operatorname{cov}\left[\int_s^t \log\sigma(z)\mathrm{d}z, \log\sigma(t)\right]}{\sqrt{\mathbb{V}\mathrm{ar}\left[\int_s^t \log\sigma(z)\mathrm{d}z\right]\mathbb{V}\mathrm{ar}\left[\log\sigma(t)\right]}}.$$

In order to compute the covariance, we need $\mathbb{E}\left[\left(\int_s^t \log\sigma(z)\mathrm{d}z\right)\log\sigma(t)\right]$, $\mathbb{E}\left[\int_s^t \log\sigma(z)\mathrm{d}z\right]$ and $\mathbb{E}\left[\log\sigma(t)\right]$. From Equation (3.4), the last expectation as well as $\mathbb{V}\mathrm{ar}\left[\log\sigma(t)\right]$ are known. We find that

$$\log\sigma(t) = \log\sigma_0 - \frac{1}{2}\alpha^2 t + \alpha W(t) =: \eta(t) + \alpha W(t),$$

and

$$\begin{aligned}\int_s^t \log\sigma(z)\mathrm{d}z &= \log\sigma_0(t-s) - \frac{1}{4}\alpha^2(t^2-s^2) + \alpha\int_s^t W(z)\mathrm{d}z \\ &=: \gamma(s,t) + \alpha\left(tW(t) - sW(s) - \int_s^t z\mathrm{d}W(z)\right).\end{aligned}$$

Based on these equations, using

$$\mathbb{E}\left[\log\sigma(T)\right] = \eta(t), \quad \mathbb{E}\left[\int_s^t \log\sigma(z)\mathrm{d}z\right] = \gamma(s,t),$$

we obtain the following expressions for the remaining expectation,

$$\begin{aligned}\mathbb{E}\left[\left(\int_s^t \log\sigma(z)\mathrm{d}z\right)\log\sigma(t)\right] &= \mathbb{E}\left[\left(\gamma(s,t) + \alpha\left(tW(t) - sW(s) - \int_s^t z\mathrm{d}W(z)\right)\right)\left(\eta(t) + \alpha W(t)\right)\right] \\ &= \eta(t)\gamma(s,t) + \alpha^2\left(t^2 - s^2 - \mathbb{E}\left[\int_s^t z\mathrm{d}W^2(z)\right]\right) \\ &= \eta(t)\gamma(s,t) + \frac{1}{2}\alpha^2(t^2-s^2).\end{aligned}$$

For the variance of $\int_s^t \log\sigma(z)\mathrm{d}z$, we first compute

$$\begin{aligned}\mathbb{E}\left[\left(\int_s^t \log\sigma(z)\mathrm{d}z\right)^2\right] &= \mathbb{E}\left[\left(\gamma(s,t) + \alpha\left(tW(t) - sW(s) - \int_s^t z\mathrm{d}W(z)\right)\right)^2\right] \\ &= \gamma^2(s,t) + \alpha^2\mathbb{E}\left[(tW(t) - sW(s))^2\right] + \alpha^2\mathbb{E}\left[\left(\int_s^t z\mathrm{d}W(z)\right)^2\right] \\ &\quad - 2\alpha^2\mathbb{E}\left[(tW(t) - sW(s))\int_s^t z\mathrm{d}W(z)\right],\end{aligned}$$

where

$$\begin{aligned}\mathbb{E}\left[(tW(t)-sW(s))^2\right] &= t^2\mathbb{E}\left[(W(t))^2\right]+s^2\mathbb{E}\left[(W(s))^2\right]-2st\mathbb{E}\left[W(s)W(t)\right]\\ &= t^3+s^3-2s^2t,\\ \mathbb{E}\left[\left(\int_s^t z\mathrm{d}W(z)\right)^2\right] &= \int_s^t z^2\mathrm{d}z=\frac{1}{3}(t^3-s^3),\\ \mathbb{E}\left[(tW(t)-sW(s))\int_s^t z\mathrm{d}W(z)\right] &= t\mathbb{E}\left[W(t)\int_s^t z\mathrm{d}W(z)\right]-s\mathbb{E}\left[W(s)\int_s^t z\mathrm{d}W(z)\right]\\ &= t\mathbb{E}\left[\int_s^t \mathrm{d}W(z)\int_s^t z\mathrm{d}W(z)\right]-s\mathbb{E}\left[\int_s^s \mathrm{d}W(z)\int_s^t z\mathrm{d}W(z)\right]\\ &= \frac{1}{2}t(t^2-s^2),\end{aligned}$$

and the variance reads

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left[\int_s^t \log\sigma(z)\mathrm{d}z\right] &= \mathbb{E}\left[\left(\int_s^t \log\sigma(z)\mathrm{d}z\right)^2\right]-\left(\mathbb{E}\left[\int_s^t \log\sigma(z)\mathrm{d}z\right]\right)^2\\ &= \alpha^2\left(t^3+s^3-2s^2t+\frac{1}{3}(t^3-s^3)-t(t^2-s^2)\right)\\ &= \alpha^2\left(\frac{1}{3}t^3+\frac{2}{3}s^3-s^2t\right).\end{aligned}$$

An approximation of the Pearson correlation coefficient is then obtained as

$$\begin{aligned}\mathscr{P}_{\log Y,\log\sigma(t)} &\approx \frac{\eta(t)\gamma(s,t)+\frac{1}{2}\alpha^2(t^2-s^2)-\eta(t)\gamma(s,t)}{\sqrt{\alpha^2\left(\frac{1}{3}t^3+\frac{2}{3}s^3-s^2t\right)\left(\alpha^2t\right)}}\\ &= \frac{t^2-s^2}{2\sqrt{\left(\frac{1}{3}t^4+\frac{2}{3}ts^3-t^2s^2\right)}}.\end{aligned} \tag{3.17}$$

Some copulas do not employ, in their definition, Pearson's coefficient as the correlation measure. For example, Archimedean copulas employ the Kendall's τ rank correlation coefficient. However, a relation between both Pearson's and Kendall's τ coefficients is available,

$$\mathscr{P}=\sin\left(\frac{\pi}{2}\tau\right).$$

GUMBEL COPULA

In this chapter, we will use Archimedean Gumbel copula. The Gaussian copula may seem a natural choice, as $\sigma(t)$ follows a log-normal distribution and $Y(T)=\int_s^t\sigma^2(z)\mathrm{d}z$ can be seen as summation of squared log-normal processes. However, in Section 2.5.1, we have empirically shown that the Gumbel copula performs most accurately in this context for a variety of SABR parameter values. The formal definition of the Archimedean Gumbel copula, considering $F_1\ldots,F_d\in[0,1]^d$ as the marginal distributions, reads

$$C_\theta(F_1\ldots,F_d)=\exp\left(-\left(\sum_{i=1}^d\left(-\log(F_i)\right)^\theta\right)^{1/\theta}\right), \tag{3.18}$$

where $\left(-\log(\cdot)\right)^{\theta}$, $\theta > 0$ is the so-called *generator function* of the Gumbel copula. In order to calibrate parameter θ, a relation between θ and the rank correlation coefficient Kendall's τ can be employed, i.e. $\theta = 1/(1-\tau)$.

So, we have derived approximations for all the components required to apply the copula-based technique for the time-integrated variance simulation. The algorithm to sample $\int_s^t \sigma^2(z)\mathrm{d}z$ given $\sigma(t)$ and $\sigma(s)$ then consists of the following steps:

1. Determine $F_{\log\sigma(t)|\log\sigma(s)}$ by Equation (3.16).
2. Determine $F_{\log\tilde{Y}|\log\sigma(s)}$ by Equation (3.6).
3. Determine the correlation between $\log Y(s,t)$ and $\log\sigma(t)$ by Equation (3.17).
4. Generate correlated uniforms, $U_{\log\sigma(t)|\log\sigma(s)}$ and $U_{\log\tilde{Y}|\log\sigma(s)}$ from the Gumbel copula by Equation (3.18).
5. From $U_{\log\sigma(t)|\log\sigma(s)}$, invert $F_{\log\sigma(t)|\log\sigma(s)}$ to get the samples $\bar{\sigma}_n$ of $\log\sigma(t)|\log\sigma(s)$. This procedure is straightforward since the normal distribution inversion is analytically available.
6. From $U_{\log\tilde{Y}|\log\sigma(s)}$, invert $F_{\log\tilde{Y}|\log\sigma(s)}$ to get the samples $\bar{y}_n$ of $\log\tilde{Y}|\log\sigma(s)$. We propose an inversion procedure based on linear interpolation. First we evaluate the CDF function at some discrete points. Then, the insight is that, by rotating the CDF under consideration, we can interpolate over probabilities. This is possible when the CDF function is a smoothly increasing function. The interpolation polynomial provides the quantiles of the original distribution from some given probabilities. Since $F_{\log\tilde{Y}|\log\sigma(s)}$ is indeed a smooth and increasing function, the interpolation-based inversion is definitely applicable. This procedure together with the use of 2D SCMC sampler (see Section 3.3.2) results in a fast and accurate inversion.
7. The samples σ_n of $\sigma(t)|\sigma(s)$ and y_n of $Y(s,t) = \int_s^t \sigma^2(z)\mathrm{d}z|\sigma(t),\sigma(s)$ are obtained by simply taking exponentials as

$$\sigma_n = \exp(\bar{\sigma}_n), \quad y_n = \exp(\bar{y}_n).$$

3.3.5. SIMULATION OF $S(t)$ GIVEN $S(s)$, $\sigma(s)$, $\sigma(t)$ AND $\int_s^t \sigma^2(z)\mathrm{d}z$

Once the samples from the time-integrated variance are obtained, the simulation of the conditional asset dynamics represents the final step of the mSABR method. For that, we follow the same approach as in Section 2.4.4 of the previous chapter, distinguishing the cases $\beta = 0$, $\beta = 1$ and $\beta \neq 0, \beta \neq 1$.

However, as the mSABR method is a multiple time-step scheme, a generic interval $[s,t]$ (instead of the fixed interval $[0,T]$) is employed and some extra considerations must be taken into account.

When $\beta \neq 1$, i.e. the $S(t)$ process does not follow log-normal dynamics, and negative asset prices can be obtained. This contradicts one of the main assumptions for the validity of the CDF expression for the 'exact' SABR simulation in Equation (3.3). Therefore, in the case of $\beta \neq 1$, the simulation of $S(t)$ must be corrected to be consistent within the

	S_0	σ_0	α	β	ρ	T
Set I [65]	0.5	0.5	0.4	0.5	0.0	4
Set II [21]	0.04	0.2	0.3	1.0	−0.5	5
Set III [4]	1.0	0.25	0.3	0.6	−0.5	20
Set IV [5]	0.0056	0.011	1.080	0.167	0.999	1

Table 3.2: Data sets.

whole simulation procedure. If the price at the initial time, s, is less or equal to zero, i.e. $S(s) \leq 0$, the price $S(t)$ must remain zero for all time. Otherwise, the absorbing boundary at zero needs to be prescribed to avoid negative prices as

$$S(t) = \max(S(t), 0).$$

If negative values are permitted, this must be handled in a different way, as it has been recently studied in [68] or [5], for example.

As was also mentioned in Section 2.4.4, a loss of the martingale property can appear when a continuous process is approximated by its discrete equivalent, especially if the discretization steps are large. In the case of the SABR model, this issue appears in the case of small β and/or S_0 values and high α and/or σ_0. Note however that, in most situations appearing in practice, this martingale error is very small and can be easily controlled by an increasing number of time-steps.

Since our goal is to use as few time-steps as possible, the application of a martingale correction becomes again essential and needs to be considered in each time-step of the simulation. We use the same martingale correction as proposed in Chapter 2.

3.4. NUMERICAL EXPERIMENTS

In this section, we benchmark the mSABR method by pricing options under the SABR dynamics. We will consider several parameter configurations, extracted from the literature (see Table 3.2), a zero-correlation set as in [65] (set I), a set under log-normal dynamics as in [21] (set II), a medium-long time to maturity set as in [4] (set III) and a more extreme set as in [5] (set IV). The strikes, K_i, are chosen following the expression

$$K_i(T) = S(0)\exp(0.1 \times \sqrt{T} \times \delta_i), \quad \delta = \{-1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5\},$$

introduced by Piterbarg in [105].

Other parameters include:

- Step size in Equation (3.5): $\Delta t = 5 \times 10^{-5}$.
- Number of terms in Equations (2.16) and (3.13): $N = 150$.
- Number of intervals for piecewise linear approximation in Equation (2.23): $L = 100$.
- Number of collocation points in Equation (3.14): $N_{\tilde{Y}} = N_\sigma = 3$.

Strikes	K_1	K_2	K_3	K_4	K_5	K_6	K_7
Antonov	73.34%	71.73%	70.17%	N/A	67.23%	65.87%	64.59%
$m = T/4$	73.13%	71.75%	70.41%	69.11%	67.85%	66.64%	65.48%
Error(bp)	−21.51	2.54	24.38	N/A	61.71	76.66	89.26
$m = T/2$	73.30%	71.78%	70.29%	68.86%	67.49%	66.17%	64.93%
Error(bp)	−4.12	4.94	12.71	N/A	25.48	30.40	34.73
$m = T$	73.25%	71.67%	70.14%	68.66%	67.24%	65.89%	64.62%
Error(bp)	−9.56	−5.93	−2.79	N/A	0.92	2.21	3.17
$m = 2T$	73.32%	71.71%	70.16%	68.65%	67.22%	65.85%	64.55%
Error(bp)	−2.08	−1.56	−1.20	N/A	−1.65	−2.35	−3.36
$m = 4T$	73.34%	71.73%	70.18%	68.67%	67.24%	65.87%	64.58%
Error(bp)	0.15	0.58	0.78	N/A	0.43	0.04	−0.48

Table 3.3: Implied volatility, increasing m: Antonov vs. mSABR. Set I.

- Number of samples: $n = 1,000,000$.

The experiments were performed on a computer with CPU Intel Core i7-4720HQ 2.6 GHz and RAM memory of 16 Gigabytes. The employed software package was Matlab r2016b.

As usual, the European option prices are obtained by averaging the maximum of the forward asset values at maturity minus the strike and zero, i.e. $\max(S(T) - K_i(T), 0)$ (call option). Subsequently, the Black-Scholes formula is inverted to determine the *implied volatilities*. Note that, once the forward asset is simulated by the mSABR method, we can price options at multiple strikes simultaneously.

3.4.1. CONVERGENCE TEST

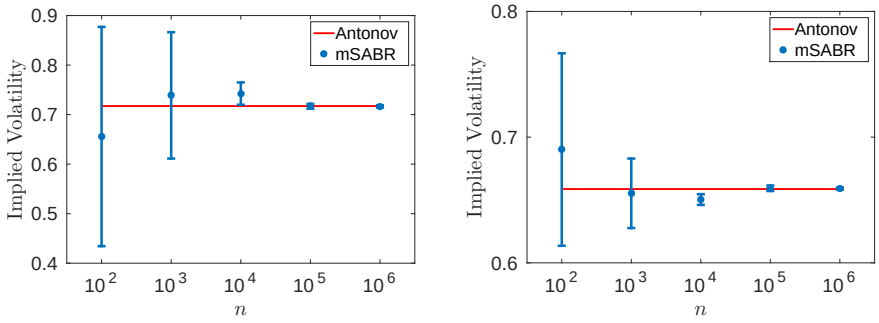
We perform a convergence study in terms of the number of time-steps, m and, subsequently, in terms of the number of samples, n. Set I is considered suitable for this experiment, since an analytic solution is available by Antonov et al. [4] when $\rho = 0$, and it can be used as a reference. Also, as Equation (3.3) is employed to sample the forward asset dynamics, the case $\rho = 0$ results in an *exact simulation*.

In Table 3.3, the implied volatilities obtained by the mSABR method for several choices of the number of time-steps m are presented. We observe by the errors in basis points that, by increasing the number of time-steps, the technique converges very rapidly. Furthermore, only a few time-steps are required to obtain accurate results: with only $m = 4$ time-steps (one per year) the error is already less than ten basis points and with $m = 4T$ (the time-step size is a quarter of year), the error remains below one basis point. Compared to the *low-bias SABR simulation scheme* proposed in [21], the mSABR method is more accurate, since it relies on the *exact* simulation of the time-integrated variance, whereas the low-bias scheme approximates the time-integrated variance by a small disturbance expansion and moment-matching techniques. That scheme performs worse when bigger time-steps are considered.

We also perform a convergence test in the number of samples, n. According to the previous experiment and in order to guarantee that the errors do not increase by the time discretization, we set $m = 4T$. In Table 3.4, the implied volatilities for increasing n are

Strikes	K_1	K_2	K_3	K_4	K_5	K_6	K_7
Antonov	73.34%	71.73%	70.17%	N/A	67.23%	65.87%	64.59%
$n = 10^2$	67.29%	65.55%	63.84%	62.20%	60.63%	59.01%	57.65%
RE	8.24×10^{-2}	8.61×10^{-2}	9.01×10^{-2}	N/A	9.82×10^{-2}	1.04×10^{-1}	1.07×10^{-1}
$n = 10^4$	73.41%	71.87%	70.36%	68.91%	67.51%	66.19%	64.94%
RE	9.65×10^{-4}	1.94×10^{-3}	2.75×10^{-3}	N/A	4.08×10^{-3}	4.93×10^{-3}	5.48×10^{-3}
$n = 10^6$	73.34%	71.73%	70.18%	68.67%	67.24%	65.87%	64.58%
RE	2.04×10^{-5}	8.08×10^{-5}	1.11×10^{-4}	N/A	6.39×10^{-5}	6.07×10^{-6}	7.43×10^{-5}

Table 3.4: Implied volatility, increasing n: Antonov vs. mSABR. Set I.



(a) Strike K_2.

(b) Strike K_6.

Figure 3.2: Convergence of the mSABR method.

presented. By the *relative error* (RE), we can see that, as expected, the error is approximately reduced by a factor of $1/\sqrt{n}$. Furthermore, we wish to show how the number of samples affects the variance of the mSABR method. In Figure 3.2, the standard deviations[3] for several choices of n are presented. Two strikes are evaluated, one *in-the-money* strike, K_2, and one *out-of-the-money* strike[4], K_6. An identical behaviour can be observed for all other strikes as well. Again, we see a fast convergence and variance reduction in terms of n.

3.4.2. PERFORMANCE TEST

Next to the convergence of the mSABR method in terms of m and n, another important aspect is the computational cost. Specifically, we will measure the execution times of an option pricing experiment using different alternative Monte Carlo-based methods and compare them with the mSABR method. Again, parameter Set I is employed since a reference value is available for this case. We consider a plain Monte Carlo Euler-Maruyama (MC Euler) discretization scheme for the forward asset, $S(t)$. Note that the absorbing boundary at zero must be handled with care. Since our main contribution here is the exact simulation of the time-integrated variance, it is natural to also com-

[3] Computed by performing 100 realizations.
[4] We consider European call options.

Error	< 100 bp	< 50 bp	< 25 bp	< 10 bp
MC Euler	6.85(200)	10.71(300)	27.42(800)	42.90(1200)
Y-Euler	2.18(4)	6.55(16)	11.85(32)	45.12(128)
Y-trpz	2.17(3)	4.24(8)	7.25(16)	14.47(32)
mSABR	3.46(1)	2.98(2)	3.72(3)	4.89(4)

Table 3.5: Execution times and time-steps, m (parentheses).

Error	< 100 bp	< 50 bp	< 25 bp	< 10 bp
MC Euler	1.98	3.59	7.37	8.77
Y-Euler	0.63	2.19	3.18	9.22
Y-trpz	0.62	1.42	1.94	2.95

Table 3.6: Speedups provided by the mSABR method.

pare the performance with other, less involved, techniques. We therefore also present a comparison between the mSABR method and two techniques where the time-integrated variance is either approximated by $\int_s^t \sigma^2(z)\mathrm{d}z \approx \sigma^2(s)(t-s)$ (left rectangular rule) or $\int_s^t \sigma^2(z)\mathrm{d}z \approx \frac{1}{2}(\sigma^2(t)+\sigma^2(s))(t-s)$ (trapezoidal rule). We denote these alternatives as the *Y-Euler* and *Y-trpz* schemes, respectively. The simulation of $S(t)$ is, in both cases, carried out as explained in Section 3.3.5. In Table 3.5, we present execution times (in seconds) of the MC Euler, *Y-Euler*, *Y-trpz* and mSABR schemes, to achieve a certain level of accuracy (measured as the error of the implied volatilities in basis points). In addition, the required number of time-steps is presented in parentheses to provide an insight in the efficiency. Furthermore, the speedup obtained by our method is included in Table 3.6. We observe that the mSABR method is superior in terms of the ratio between computational cost and accuracy. The accuracy of Y-Euler or Y-trpz can be improved by adding intermediate time-steps, but then also the computational cost increases.

3.4.3. 'ALMOST EXACT' SABR SIMULATION BY VARYING ρ

As a next experiment we test the stability of the mSABR method when correlation parameter ρ varies from negative to positive values. As stated before, Equation (3.3) is only exact for $\rho = 0$. For that, we use Set II since the conditional forward asset simulation enables a closed-form solution (see Section 2.4.4) for $\beta = 1$. The considered values are $\rho = \{-0.5, 0.0, 0.5\}$. Following the outcome of the previous experiment, we set $m = 4T$. As a reference, a Monte Carlo (MC) simulation with a very fine Euler-Maruyama time discretization ($1000T$ time-steps) in the forward asset process is employed. In Table 3.7, the resulting implied volatilities are provided. The differences between the Monte Carlo volatilities and the ones given by the mSABR method are within ten basis points for all choices of ρ.

3.4.4. PRICING BARRIER OPTIONS

We will price barrier options with the mSABR method. The *up-and-out* call option is considered here, with the barrier level, B, $B > S_0, B > K_i$. The price of this type of barrier

Strikes	K_1	K_2	K_3	K_4	K_5	K_6	K_7
				$\rho = -0.5$			
MC	22.17%	21.25%	20.38%	19.57%	18.88%	18.33%	17.95%
mSABR	22.21%	21.28%	20.39%	19.58%	18.88%	18.32%	17.94%
Error(bp)	3.59	2.86	1.78	0.95	−0.19	−0.96	−1.10
				$\rho = 0.0$			
MC	21.35%	20.96%	20.71%	20.63%	20.71%	20.96%	21.34%
mSABR	21.35%	20.95%	20.69%	20.60%	20.68%	20.93%	21.32%
Error(bp)	0.04	−1.04	−2.51	−3.02	−3.33	−3.19	−2.56
				$\rho = 0.5$			
MC	19.66%	20.04%	20.61%	21.34%	22.20%	23.14%	24.16%
mSABR	19.59%	19.96%	20.54%	21.28%	22.15%	23.11%	24.11%
Error(bp)	−6.93	−7.36	−6.77	−5.53	−4.35	−3.76	−4.05

Table 3.7: Implied volatility, varying ρ: Euler Monte Carlo vs. mSABR. Set II.

option reads

$$V_i(T,S,B) = \exp(-rT)\,\mathbb{E}\left[(S(T)-K_i)\mathbb{1}(\max_{0<t_k\leq T} S(t_k) > B)\right],$$

where $\mathbb{1}(\cdot)$ represents the indicator function and t_k are the predefined times where the barrier condition (whether or not it is hit) is checked.

As a reference, a Monte Carlo method with very fine Euler-Maruyama time discretization is employed, as in the previous experiment. The number of time-steps for the mSABR scheme is again set to $m = 4T$. In Tables 3.8, 3.9 and 3.10, the obtained prices are presented for the parameter sets I, II and III, respectively. The prices are multiplied by a factor of 100. Also, the MSE is shown. We define the MSE as

$$\text{MSE} = \frac{1}{7}\sum_{i=1}^{7}\left(V_i^{MC}(T,S,B) - V_i^{mSABR}(T,S,B)\right)^2,$$

where $V_i^{MC}(T,S,B)$ and $V_i^{mSABR}(T,S,B)$ are the barrier option prices provided by Monte Carlo method and by the mSABR method, respectively.

The resulting accuracy is very satisfactory. As expected, higher option prices are obtained for bigger barrier levels, B, and/or lower strikes, K_i.

3.4.5. NEGATIVE INTEREST RATES

The SABR model is very popular in the interest rate context. One of the most important current features in this market is the occurence of negative interest rate values and strikes. Approaches dealing with this issue have appeared in the literature, like [5] or [68]. To apply the mSABR scheme in the case of negative interest rates, we will use the *shifted SABR model* by Schlenkrich in [116]. The shifted SABR model is defined as follows

$$\begin{aligned} dS(t) &= \sigma(t)(S(t)+\theta)^{\beta} dW_S(t), \\ S(0) &= (S_0+\theta)\exp(rT), \end{aligned}$$

Strikes	K_1	K_2	K_3	K_4	K_5	K_6	K_7
				$B = 1.2$			
MC	6.2868	5.4577	4.6330	3.8263	3.0551	2.3358	1.6869
mSABR	6.3264	5.4890	4.6566	3.8436	3.0677	2.3452	1.6953
MSE	5.3196×10^{-8}						
				$B = 1.5$			
MC	10.2271	9.1568	8.0718	6.9856	5.9142	4.8753	3.8899
mSABR	10.2249	9.1507	8.0609	6.9702	5.8960	4.8572	3.8737
MSE	1.8884×10^{-8}						
				$B = 1.8$			
MC	13.5740	12.3571	11.1118	9.8513	8.5910	7.3477	6.1403
mSABR	13.5915	12.3686	11.1174	9.8507	8.5851	7.3380	6.1276
MSE	1.0851×10^{-8}						

Table 3.8: Pricing barrier options with mSABR: $V_i(T,S,B) \times 100$. Set I.

Strikes	K_1	K_2	K_3	K_4	K_5	K_6	K_7
				$B = 0.08$			
MC	1.1702	0.9465	0.7268	0.5215	0.3423	0.1996	0.0987
mSABR	1.1724	0.9486	0.7285	0.5226	0.3428	0.1997	0.0986
MSE	1.8910×10^{-10}						
				$B = 0.1$			
MC	1.3099	1.0766	0.8462	0.6290	0.4367	0.2794	0.1626
mSABR	1.3092	1.0761	0.8456	0.6282	0.4355	0.2782	0.1618
MSE	7.5542×10^{-11}						
				$B = 0.12$			
MC	1.3521	1.1168	0.8841	0.6644	0.4695	0.3093	0.1891
mSABR	1.3518	1.1166	0.8838	0.6639	0.4686	0.3080	0.1880
MSE	6.3648×10^{-11}						

Table 3.9: Pricing barrier options with mSABR: $V_i(T,S,B) \times 100$. Set II.

Strikes	K_1	K_2	K_3	K_4	K_5	K_6	K_7
				$B = 2.0$			
MC	29.1174	23.4804	17.2273	10.7825	5.0203	1.1750	0.0036
mSABR	29.2346	23.5828	17.3086	10.8327	5.0385	1.1805	0.0036
MSE	4.8146×10^{-7}						
				$B = 2.5$			
MC	41.3833	34.5497	26.8311	18.6089	10.7281	4.4893	0.9434
mSABR	41.3394	34.5097	26.7948	18.5747	10.6943	4.4546	0.9320
MSE	1.2131×10^{-7}						
				$B = 3.0$			
MC	48.5254	41.1652	32.7980	23.7807	14.9344	7.5364	2.6692
mSABR	48.5008	41.1515	32.7888	23.7655	14.9097	7.5117	2.6549
MSE	3.6201×10^{-8}						

Table 3.10: Pricing barrier options with mSABR: $V_i(T,S,B) \times 100$. Set III.

where $\theta > 0$ is a displacement, or shift, in the underlying. The volatility process, $\sigma(t)$, remains invariant (see Equation (3.1)). Parameter θ can be seen as the maximum level of negativity that is expected in the rates. This generalization of the SABR model is widely used by market practitioners due to its simplicity, and, precisely, the advantage of keeping the existing simulation schemes. In mSABR, the assumption $S(t) > 0$ is required to employ the method.

In order to test the mSABR scheme in the context of negative interest rates, we employ the set IV. The SABR model parameters were obtained by Antonov et al. [5] after a calibration process to real data[5] under the shifted SABR model. Note that this configuration corresponds to an extreme situation with high values of the correlation, ρ, and *vol-vol*, α. As a shift parameter, $\theta = 0.02$ is chosen. We compare the *normal implied volatilities*[6] obtained by employing Monte Carlo with very fine ($1000T$) time discretization (as a reference) and the mSABR method with $m = 4T$. In Figure 3.3(a), these curves are depicted and a perfect fit is observed. The strikes, K are chosen to permit negative values. We perform an even more extreme experiment, by setting $S_0 = 0$. The resulting normal implied volatilities are presented in Figure 3.3(b), again with a excellent fit.

Due to the complexity of the negative interest rates experiment and this particular set of parameters ($\rho \approx 1$), we wish to test the performance of our method under these conditions. In Tables 3.11 and 3.12 the execution times obtained are presented. Again, the small number of time-steps due to the "almost" exact simulation of the mSABR method provides an important gain in terms of performance.

3.5. CONCLUSIONS

In this chapter, we have proposed an accurate and robust multiple time-step Monte Carlo method to simulate the SABR model with only a few time-steps. The mSABR method employs many nontrivial components that are not yet seen before in combina-

[5] 1Y15Y Swiss franc swaption from February 10, 2015.
[6] Determined by using the *normal* Bachelier model instead of the *log-normal* Black-Scholes model.

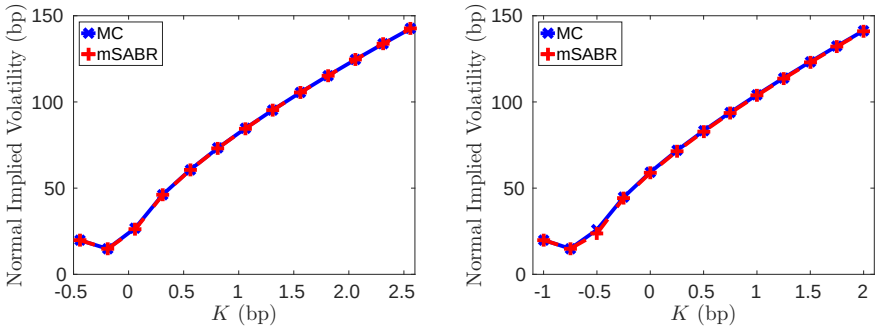


Figure 3.3: The mSABR method in the context of negative interest rates.

Error	< 25 bp	< 10 bp	< 5 bp	< 1 bp
Y-Euler	2.47(4)	9.64(16)	14.32(24)	19.25(32)
Y-trpz	1.32(2)	4.99(8)	9.75(16)	14.52(24)
mSABR	1.61(1)	2.57(2)	3.52(3)	4.57(4)

Table 3.11: Execution times and time-steps, m (parentheses).

Error	< 25 bp	< 10 bp	< 5 bp	< 1 bp
Y-Euler	1.57	3.75	4.07	4.21
Y-trpz	0.82	1.94	2.77	3.17

Table 3.12: Speedups provided by the mSABR method.

tion within a Monte Carlo simulation. A copula methodology to generate samples from the conditional SABR's time-integrated variance process has been used. The marginal distribution of the time-integrated variance has been derived by employing Fourier techniques. We have dealt with an increasing computational complexity, due to the multiple time-steps and the almost exact simulation, by applying an interpolation based on stochastic collocation. This has resulted in an efficient SABR simulation scheme. We have numerically shown the convergence and the stability of the method. In terms of convergence, the mSABR method requires only very few time-steps to achieve high accuracy, due to the focus on almost exact simulation (in contrast to the low-bias SABR scheme [21]). This fact impacts the performance, obtaining an excellent ratio between accuracy and computational cost. As a multiple time-step method, it can be employed to price path-dependent options. As an example, a pricing experiment for barrier options has been carried out. Furthermore, we have shown that the mSABR scheme is also suitable in the context of negative interest rates, in combination with a shifted model.

As a generalization of the technique presented in Chapter 2, the mSABR method combines the Monte Carlo simulation and Fourier inversion, resulting again in a hybrid solution. However, due to the extra complexity introduced by the use of multiple time-steps, an additional component is required to make the technique tractable in terms of computational cost. We have employed an advanced interpolation method based on stochastic collocation, SCMC, which provides an enormous gain regarding the execution time.

CHAPTER 4

The data-driven COS method

In this chapter, we consider a completely different hybrid solution method, which relies on a combination of Monte Carlo methods and Fourier inversion techniques, but employing a data-based approach. We therefore present the data-driven COS method, ddCOS. It is a financial option valuation method which assumes the availability of asset data samples: a ChF of the underlying asset is not required. As such, the method represents a generalization of the well-known COS method[49]. Convergence with respect to the number of asset samples is according the convergence of Monte Carlo methods for pricing financial derivatives. The ddCOS method is particularly interesting for PDF recovery and also for the efficient computation of the option's sensitivities Delta and Gamma. These are often used in risk management, and can be obtained at a higher accuracy with ddCOS than with plain Monte Carlo methods.

4.1. INTRODUCTION

In quantitative finance, statistical distributions are commonly used for the valuation of financial derivatives and within risk management. The underlying assets are often modelled by means of SDEs. Except for the classical and most simple asset models, the corresponding PDF and CDF are typically not known and need to be approximated.

In order to compute derivative prices, and to approximate statistical distributions, Fourier-based methods are often used numerical techniques. They are based on the connection between the PDF and the ChF, which is the Fourier transform of the PDF. The ChF is often available, and sometimes even in closed form, for the broad class of regular diffusions and also for Lévy processes. Some representative efficient Fourier pricing methods include those by Carr and Madan [19], Boyarchenko and Levendorskii [13], Lewis [92] and Fang and Oosterlee [49]. Here, we focus on the COS method from [49], which is based on an approximation of the PDF by means of a cosine series expansion.

Still, however, the asset dynamics for which the ChF are known is not exhaustive, and for many relevant asset price processes we do not have such information to recover the PDF. In recent years several successful attempts have been made to employ Fourier pricing methods without the explicit knowledge of the ChF. In Grzelak and Oosterlee [64], for example, a hybrid model with stochastic volatility and stochastic interest rate was linearized by means of expectation operators to cast the approximate system of SDEs in the framework of affine diffusions. Ruijter and Oosterlee [111] discretized the governing asset SDEs first and then worked with the ChF of the discrete asset process, within the framework of the COS method. Borovykh et al. [12] used the Taylor expansion to derive

This chapter is based on the article "On the data-driven COS method". Submitted for publication, 2017 [91].

a ChF for which they could even price Bermudan options highly efficiently. In this work, we extend the applicability of the COS method to the situation where only data (like samples from an unknown underlying risk neutral asset distribution) are available.

The density estimation problem, using a *data-driven* PDF, has been intensively studied in the last decades, particularly since it is a component in the *machine learning* framework [10]. Basically, density estimators can be classified into parametric and non-parametric estimators. The first type relies on the fact that prior knowledge is available (like moments) to determine the relevant parameters, while for non-parametric estimators the parameters need to be determined solely from the samples themselves. Within this second type of estimators we can find histograms, kernel density and orthogonal series estimators. A thorough description of these estimators is provided in [120]. More recently, some applications in finance have also appeared, see [48, 97, 118], for example.

For the valuation of financial derivatives, we will combine density estimators with Fourier-based methods, so orthogonal series form a natural basis. We will focus on the framework of *statistical learning*, see [131]. In statistical learning, a regularization is employed to derive an expression for the data-driven empirical PDF. By representing the unknown PDF as a cosine series expansion, a closed-form solution of the regularization problem is known [131], which forms the basis of the *data-driven COS method* (ddCOS).

The use of the COS method gives us expressions for option prices and, in particular, for the *option sensitivities or Greeks*. These option Greeks are the derivatives of option price with respect to a variable or parameter. The efficient computation of the Greeks is a challenging problem when only asset samples are available. Existing approaches are based on Monte Carlo-based techniques, like on finite-differences (bump and revalue), pathwise or likelihood ratio techniques, for which details can be found in [59], chapter 7. Several extensions and improvements of these approaches have appeared, for example, based on adjoint formulations [56], the ChF [60, 61], Malliavin calculus [41, 53], algorithmic differentiation [17, 43] or combinations of these [18, 22, 58]. All in all, the computation of the Greeks can be quite involved. The ddCOS method is not directly superior to Monte Carlo methods for option valuation, but it is competitive for the computation of the corresponding sensitivities. We derive simple expressions for the Greeks Delta and Gamma.

The importance of Delta and Gamma in dynamic hedging and risk management is well-known. A useful application is found in the *Delta-Gamma approach* [14] to quantify market risk. The approximation of risk measures like *Value-at-Risk* (VaR) and *Expected Shortfall* (ES) under the Delta-Gamma approach is still nontrivial. Next to Monte Carlo methods, Fourier techniques have been employed in this context, when the ChF of the change in the value of the option portfolio is known (see [23, 121]). For example, the COS method has been applied in [101] to efficiently compute the VaR and ES under the Delta-Gamma approach. The ddCOS method may generalize the applicability to the case where only data is available.

This paper is organized as follows. The ddCOS method, and the origins in statistical learning and Fourier-based option pricing, are presented in Section 4.2. Variance reduction techniques can also be used within the ddCOS method, providing an additional convergence improvement. We provide insight and determine values for the method's open parameters in Section 4.3. Numerical experiments, with a focus on the option Greeks,

are presented in Section 4.4. We conclude in Section 4.5.

4.2. THE DATA-DRIVEN COS METHOD

In this section we will discuss the ddCOS method, in which aspects of the Monte Carlo method, density estimators and the COS method are combined to approximate, in particular, the option Greeks Delta and Gamma. We will focus on European options here.

The COS method in [49] is a Fourier-based method by which option prices and sensitivities can be computed for various options under different models. The method relies heavily on the availability of the ChF, i.e., the Fourier transform of the PDF. In the present work, we assume that only asset samples are available, not the ChF, resulting in the data-driven COS method. It is based on regularization in the context of the statistical learning theory, presented briefly in Section 4.2.2. The connection with the COS method is found in the fact that the data-driven PDF appears as a cosine series expansion.

4.2.1. THE COS METHOD

The starting point for the well-known COS method is the risk-neutral option valuation formula, where the value of a European option at time t, $V(t,x)$, is an expectation under the risk neutral pricing measure, i.e.,

$$V(t,x) = \mathrm{e}^{-r(T-t)}\mathbb{E}\left[h(T,y)|x\right] = \mathrm{e}^{-r(T-t)}\int_{\mathbb{R}} h(T,y)f(y|x)\mathrm{d}y, \tag{4.1}$$

with r the risk-free rate, T the maturity time, and $f(y|x)$ the PDF of the underlying process, and $h(T,y)$ represents the option value at maturity time, being the payoff function. Typically, x and y are chosen to be scaled variables,

$$x := \log\left(\frac{S(0)}{K}\right) \quad \text{and} \quad y := \log\left(\frac{S(T)}{K}\right),$$

where $S(t)$ is the underlying asset process at time t, and K is the strike price.

$f(y|x)$ is unknown in most cases and in the COS method it is approximated, on a finite interval $[a,b]$, by a cosine series expansions, i.e.,

$$f(y|x) = \frac{1}{b-a}\left(A_0 + 2\sum_{k=1}^{\infty} A_k(x)\cdot\cos\left(k\pi\frac{y-a}{b-a}\right)\right),$$
$$A_0 = 1, \quad A_k(x) = \int_a^b f(y|x)\cos\left(k\pi\frac{y-a}{b-a}\right)\mathrm{d}y, \quad k = 1,2,\ldots.$$

By substituting this expression in Equation (4.1), interchanging the summation and integration operators using Fubini's Theorem, and introducing the following definition,

$$H_k := \frac{2}{b-a}\int_a^b h(T,y)\cos\left(k\pi\frac{y-a}{b-a}\right)\mathrm{d}y,$$

we find that the option value is given by

$$V(t,x) \approx \mathrm{e}^{-r(T-t)}\sum_{k=0}^{\infty}{}' A_k(x)H_k, \tag{4.2}$$

where $'$ indicates that the first term is divided by two. So, the product of two real-valued functions in Equation (4.1) is transformed into the product of their cosine expansion coefficients, A_k and H_k. Coefficients A_k can be computed by the ChF and H_k is known analytically (for many types of options).

Closed-form expressions for the option Greeks can also be derived. From the COS option value formula, Δ and Γ are obtained by

$$\begin{aligned}\Delta &= \frac{\partial V(t,x)}{\partial S} = \frac{1}{S(0)}\frac{\partial V(t,x)}{\partial x} \approx \exp(-r(T-t))\sum_{k=0}^{\infty}{}' \frac{\partial A_k(x)}{\partial x}\frac{H_k}{S(0)},\\ \Gamma &= \frac{\partial^2 V(t,x)}{\partial S^2} = \frac{1}{S^2(0)}\left(-\frac{\partial V(t,x)}{\partial x} + \frac{\partial^2 V(t,x)}{\partial x^2}\right)\\ &\approx \exp(-r(T-t))\sum_{k=0}^{\infty}{}' \left(-\frac{\partial A_k(x)}{\partial x} + \frac{\partial^2 A_k(x)}{\partial x^2}\right)\frac{H_k}{S^2(0)}.\end{aligned} \tag{4.3}$$

Due to the rapid decay of the coefficients, $V(t,x)$, Δ and Γ can be approximated with high accuracy by truncating the infinite summation in Equations (4.2) and (4.3) to N terms. Under suitable assumptions, exponential convergence is proved and numerically observed.

4.2.2. STATISTICAL LEARNING THEORY FOR DENSITY ESTIMATION

In the setting of this paper, we assume a vector of n independent and identically distributed (i.i.d.) samples, $X_1, X_2, \ldots, X_n$. Based on these samples, we wish to find an accurate approximation of the PDF estimator, $f_n(x)$, which should approximate PDF $f(x)$.

By definition, the PDF is related to its CDF $F(x)$,

$$\int_{-\infty}^{x} f(y)\mathrm{d}y = F(x). \tag{4.4}$$

Function $F(x)$ is approximated by the empirical approximation,

$$F_n(x) = \frac{1}{n}\sum_{j=1}^{n}\eta(x - X_j), \tag{4.5}$$

where $\eta(\cdot)$ is a step function. This approximation converges to the "true CDF" with rate $\mathcal{O}(1/\sqrt{n})$. Rewriting Equation (4.4) as a linear operator equation, gives us,

$$Cf = F \approx F_n,$$

where the operator $Ch := \int_{-\infty}^{x} h(z)\mathrm{d}z$.

As explained in [131], this matrix equation represents an ill-posed problem, and a *risk functional* should be constructed, with a regularization term, as follows

$$R_{\gamma_n}(f, F_n) = L^2_{\mathscr{H}}(Cf, F_n) + \gamma_n W(f), \tag{4.6}$$

where $L_{\mathscr{H}}$ is a metric of the space $\mathscr{H}$, $\gamma_n > 0$ is a parameter which gives a weight to the regularization term $W(f)$, with $\gamma_n \to 0$ as $n \to \infty$. The solution of $Cf = F_n$ belongs to $\mathscr{D}$, the domain of definition of $W(f)$. Functional $W(f)$ takes real non-negative values in $\mathscr{D}$.

Furthermore, $\mathscr{M}_c = \{f : W(f) \leq c\}$ is a compact set in $\mathscr{H}$ (the space where the solution exists and is unique).

The solution f_n, minimizing the functional in Equation (4.6), converges almost surely to the desired PDF. For the ill-posed density estimation problem, other conditions imposed for consistency (see details in [131], chapter 7), include

$$\begin{aligned} &n\gamma_n \to \infty \ \text{ as } \ n \to \infty, \text{ and} \\ &\frac{n}{\log n}\gamma_n \to \infty \ \text{ as } \ n \to \infty. \end{aligned} \tag{4.7}$$

4.2.3. REGULARIZATION AND FOURIER-BASED DENSITY ESTIMATORS

A relation exists between the regularization approach in Equation (4.6) and Fourier-based PDF approximation, more specifically, cosine series expansion estimators. By specific choices for the metric and the regularization term in Equation (4.6), i.e., $L_{\mathscr{H}} = L_2$, and

$$W(f) = \int_{\mathbb{R}} \left(\int_{\mathbb{R}} \mathscr{K}(z-x) f(x) \mathrm{d}x \right)^2 \mathrm{d}z,$$

with kernel $\mathscr{K}(z-x)$, the functional reads

$$R_{\gamma_n}(f, F_n) = \int_{\mathbb{R}} \left(\int_0^x f(y)\mathrm{d}y - F_n(x) \right)^2 \mathrm{d}x + \gamma_n \int_{\mathbb{R}} \left(\int_{\mathbb{R}} \mathscr{K}(z-x) f(x) \mathrm{d}x \right)^2 \mathrm{d}z. \tag{4.8}$$

Denoting by $\hat{f}(u)$, $\hat{F}_n(u)$ and $\hat{\mathscr{K}}(u)$ the Fourier transforms of $f(x)$, $F_n(x)$ and $\mathscr{K}(x)$, respectively, an expression for $\hat{F}_n(u)$ can be derived by applying Fourier transformation to Equation (4.5),

$$\begin{aligned} \hat{F}_n(u) &= \frac{1}{2\pi} \int_{\mathbb{R}} F_n(x) \mathrm{e}^{-iux} \mathrm{d}x \\ &= \frac{1}{2n\pi} \int_{\mathbb{R}} \sum_{j=1}^{n} \eta(x - X_j) \mathrm{e}^{-iux} \mathrm{d}x = \frac{1}{n} \sum_{j=1}^{n} \frac{\exp(-iuX_j)}{iu}, \end{aligned}$$

where $i = \sqrt{-1}$ is the imaginary unit.

By employing the *convolution theorem* and *Parseval's identity*, Equation (4.8) can be rewritten as

$$R_{\gamma_n}(f, F_n) = \left\| \frac{\hat{f}(u) - \frac{1}{n}\sum_{j=1}^{n} \exp(-iuX_j)}{iu} \right\|_{L_2}^2 + \gamma_n \left\| \hat{\mathscr{K}}(u) \hat{f}(u) \right\|_{L_2}^2.$$

The condition to minimize $R_{\gamma_n}(f, F_n)$ is given by,

$$\frac{\hat{f}(u)}{u^2} - \frac{1}{nu^2} \sum_{j=1}^{n} \exp(-iuX_j) + \gamma_n \hat{\mathscr{K}}(u) \hat{\mathscr{K}}(-u) \hat{f}(u) = 0,$$

which gives us,

$$\hat{f}_n(u) = \left(\frac{1}{1 + \gamma_n u^2 \hat{\mathscr{K}}(u) \hat{\mathscr{K}}(-u)} \right) \frac{1}{n} \sum_{j=1}^{n} \exp(-iuX_j). \tag{4.9}$$

4

When kernel $\mathcal{K}$ is the p-th derivative of the Dirac delta function, i.e., $\mathcal{K}(x) = \delta^{(p)}(x)$, and the desired PDF, $f(x)$, belongs to the class of functions whose p-th derivative ($p \geq 0$) belongs to $L_2(0,\pi)$, the risk functional becomes

$$R_{\gamma_n}(f,F_n) = \int_0^\pi \left(\int_0^x f(y)\mathrm{d}y - F_n(x) \right)^2 \mathrm{d}x + \gamma_n \int_0^\pi \left(f^{(p)}(x) \right)^2 \mathrm{d}x. \tag{4.10}$$

Given a series expansion in *orthonormal functions*, $\psi_1(\theta),\ldots,\psi_k(\theta),\ldots$, the approximation to the unknown PDF will be of the form

$$f_n(\theta) = \frac{1}{\pi} + \frac{2}{\pi} \sum_{k=1}^{\infty} \tilde{A}_k \psi_k(\theta), \tag{4.11}$$

4

with $\tilde{A}_0, \tilde{A}_1, \ldots, \tilde{A}_k, \ldots$ expansion coefficients, defined as $\tilde{A}_k =< f_n, \psi_k >$.

We need to compute the expansion coefficients so that the functional in Equation (4.10) is minimized. The coefficients $\tilde{A}_k$ cannot be directly computed from the definition since the unknown PDF, f_n, is implicitly involved in the expression, i.e.,

$$\begin{aligned} \tilde{A}_k &=< f_n, \psi_k >=< \hat{f}_n, \hat{\psi}_k > \\ &= \int_0^\pi \left(\left(\frac{1}{1+\gamma_n u^2 \hat{\mathcal{K}}(u)\hat{\mathcal{K}}(-u)} \right) \frac{1}{n} \sum_{j=1}^{n} \exp(-iu\theta_j) \right) \cdot \hat{\psi}_k(u)\mathrm{d}u. \end{aligned}$$

Using cosine series expansions, i.e., $\psi_k(\theta) = \cos(k\theta)$, it is well-known that $\hat{\psi}_k(u) = \frac{1}{2}(\delta(u-k)+\delta(u+k))$. This facilitates the computation of the series coefficients, $\tilde{A}_k$, avoiding the calculation of the integral. Thus, the minimum of the functional using cosine series expansions is obtained when

$$\begin{aligned} \tilde{A}_k &= \frac{1}{2n} \left(\left(\frac{1}{1+\gamma_n (-k)^2 \hat{\mathcal{K}}(-k)\hat{\mathcal{K}}(k)} \right) \sum_{j=1}^{n} \exp(ik\theta_j) \right. \\ &\quad \left. + \left(\frac{1}{1+\gamma_n k^2 \hat{\mathcal{K}}(k)\hat{\mathcal{K}}(-k)} \right) \sum_{j=1}^{n} \exp(-ik\theta_j) \right) \\ &= \frac{1}{1+\gamma_n k^2 \hat{\mathcal{K}}(k)\hat{\mathcal{K}}(-k)} \frac{1}{n} \sum_{j=1}^{n} \cos(k\theta_j) \\ &= \frac{1}{1+\gamma_n k^{2(p+1)}} \frac{1}{n} \sum_{j=1}^{n} \cos(k\theta_j), \end{aligned} \tag{4.12}$$

where $\theta_j \in (0,\pi)$ are given samples of the unknown distribution. In the last step, $\hat{\mathcal{K}}(u) = (iu)^p$ is used.

Assuming that the samples are given, the solution contains two free parameters: *regularization parameter* γ_n, and *smoothing parameter* p.

In Section 4.3, we will discuss the impact of the regularization parameter on the convergence to the PDF in terms of the number of samples. We will use $p = 0$ here.

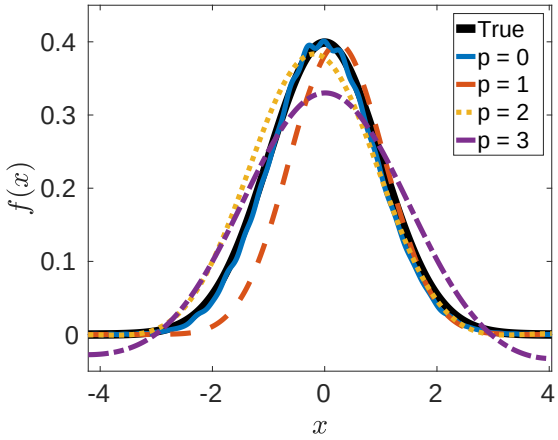


Figure 4.1: Smoothing of the PDF approximation in relation to parameter p.

SMOOTHING PARAMETER EXAMPLE

In order to give an insight in the influence of parameter p on the approximation, in Figure 4.1 standard normal densities obtained for several values of p are shown. With p increasing, the densities get increasingly smoothed and flat. The choice $p = 0$ (regularizing the PDF itself and not imposing regularization upon its derivatives) appears most appropriate in our context of smooth densities.

4.2.4. THE ddCOS METHOD

We are now ready to present the ddCOS method, where we employ the series expansion coefficients from the regularization approach. We replace the A_k-coefficients from Equation (4.2) by those coefficients based on data, $\tilde{A}_k$ in Equation (4.12).

So, suppose we have risk neutral samples (or values) from an underlying asset at a future time t, i.e., $S_1(t), S_2(t), \ldots, S_n(t)$. We compute the value of a European option with maturity time T and strike price K, and require therefore the samples $S_j(T)$. With a logarithmic transformation, we have

$$Y_j := \log\left(\frac{S_j(T)}{K}\right).$$

Before employing these samples in the regularization approach and because the solution is defined in $(0,\pi)$, we need to transform the samples by the following change of variables,

$$\theta_j = \pi \frac{Y_j - a}{b - a},$$

where the boundaries a and b are defined as

$$a := \min_{1\le j\le n}(Y_j), \quad b := \max_{1\le j\le n}(Y_j).$$

The A_k coefficients in Equation (4.2) are replaced by the data-driven $\tilde{A}_k$ in Equation (4.12),

$$A_k \approx \tilde{A}_k = \frac{\frac{1}{n}\sum_{j=1}^{n} \cos\left(k\pi \frac{Y_j - a}{b-a}\right)}{1+\gamma_n k^{2(p+1)}}.$$

The ddCOS pricing formula for European options based on risk neutral data is now obtained as

$$\begin{aligned}\tilde{V}(t,x) &= \mathrm{e}^{-r(T-t)} \sum_{k=0}^{\infty}{}' \frac{\frac{1}{n}\sum_{j=1}^{n} \cos\left(k\pi \frac{Y_j - a}{b-a}\right)}{1+\gamma_n k^{2(p+1)}} \cdot H_k \\ &= \mathrm{e}^{-r(T-t)} \sum_{k=0}^{\infty}{}' \tilde{A}_k H_k.\end{aligned} \tag{4.13}$$

As in the original COS method, we must truncate the infinite sum in Equation (4.13) to a finite number of terms N, i.e.,

$$\tilde{V}(t,x) = \mathrm{e}^{-r(T-t)} \sum_{k=0}^{N}{}' \tilde{A}_k H_k, \tag{4.14}$$

which completes the ddCOS pricing formula.

The samples Y_j should originate from one initial state, i.e. the dependency on the state x is implicitly assumed. In the case of European options this is typically fulfilled. In the Monte Carlo method, for example, all simulated asset paths depart from the same point $S(0)$, so that $x := \log\left(\frac{S(0)}{K}\right)$.

Regarding the Greeks, we can also derive data-driven expressions for the Δ and Γ sensitivities. We first define the corresponding sine coefficients as

$$\tilde{B}_k := \frac{\frac{1}{n}\sum_{j=1}^{n} \sin\left(k\pi \frac{Y_j - a}{b-a}\right)}{1+\gamma_n k^{2(p+1)}}.$$

Taking derivatives in Equation (4.14) w.r.t the samples, Y_j, and following the COS expression for the sensitivities in Equation (4.3), the data-driven Greeks, $\tilde{\Delta}$ and $\tilde{\Gamma}$, can be obtained by

$$\begin{aligned}\tilde{\Delta} &= \mathrm{e}^{-r(T-t)} \sum_{k=0}^{N}{}' \tilde{B}_k \cdot \left(-\frac{k\pi}{b-a}\right) \cdot \frac{H_k}{S(0)}, \\ \tilde{\Gamma} &= \mathrm{e}^{-r(T-t)} \sum_{k=0}^{N}{}' \left(\tilde{B}_k \cdot \frac{k\pi}{b-a} - \tilde{A}_k \cdot \left(\frac{k\pi}{b-a}\right)^2\right) \cdot \frac{H_k}{S^2(0)}.\end{aligned}$$

APPLICATION OF VARIANCE REDUCTION

Because of the focus on asset path data, the ddCOS method is related to the Monte Carlo method. Variance reduction in Monte Carlo methods is typically achieved by the use of variance reduction techniques. The ddCOS method also admits an additional variance reduction, in this case, for the computation of the expansion coefficients, $\tilde{A}_k$. We show how to introduce *antithetic variates (AV)* to our method. Since one of the assumptions for the regularization approach is that the samples are i.i.d., an immediate application

of AV is not possible. Therefore, if we assume that antithetic samples, Y_i', to the original samples Y_i, can be computed without any serious computational effort, a new estimator for the coefficients can be defined as

$$\bar{A}_k := \frac{1}{2}\left(\tilde{A}_k + \tilde{A}_k'\right),$$

where we denote by $\tilde{A}_k'$ the corresponding "antithetic coefficients", obtained by Y_i'. By a similar derivation as for the standard AV technique, it can be proved that the use of coefficients $\bar{A}_k$ will give us a variance reduction compared to using the $\tilde{A}_k$ coefficients. Other variance reduction techniques may also be considered for the ddCOS method under the assumption of i.i.d. samples.

In order to reduce the variance of any estimator, additional information may be introduced. A well-known property to fulfil is the martingale property. To preserve this property, a simple transformation of the samples can be made by

$$\begin{aligned} S(T) &= S(T) - \frac{1}{n}\sum_{j=1}^{n} S_j(T) + \mathbb{E}[S(T)], \\ &= S(T) - \frac{1}{n}\sum_{j=1}^{n} S_j(T) + S(0)\exp(rT). \end{aligned}$$

As this modification is performed over the samples, it can also be used in the context of the ddCOS method.

4.3. CHOICE OF PARAMETERS IN DDCOS METHOD

In this section, the selection and the influence of the regularization parameter γ_n in the ddCOS method is studied.

4.3.1. REGULARIZATION PARAMETER γ_n

The regularization parameter γ_n plays an important role in the empirical PDF f_n. Without the inclusion of the regularization term, the PDF approximation provided by Equation (4.10), would give us a standard orthogonal series estimator. The choice of parameter γ_n impacts the efficiency of the data-driven COS method, since it is related to the required number of data samples, and by reducing the number of samples, the overall computational cost can be reduced.

The first option for choosing the regularization parameter, γ_n, which was proposed in [131], is given by the following rule,

$$\gamma_n = \frac{\log\log n}{n}. \tag{4.15}$$

As proved in [131], this rule provides a robust asymptotic rate of convergence under the assumption of a compactly supported PDF. It implies, with probability one, uniformly converging approximations f_n to the unknown PDF. Note, however, that the regularization parameter does not satisfy the second condition in Equation (4.7).

Although Equation (4.15) ensures an optimal asymptotic convergence in terms of n, it may not be the optimal γ_n-value for density estimation with a given fixed amount

of samples. For that purpose we can exploit the relation between the empirical CDF, $F_n(x)$, and the unknown CDF, $F(x)$. This relation can be modelled by means of different *statistical laws*. Some examples include the Kolmogorov-Smirnov, Anderson-Darling, Kuiper and the Smirnov-Cramér-von Mises laws, by which a measure of the distance, or, *goodness-of-fit*, between $F_n(x)$ and $F(x)$ can be defined.

We are interested in a statistic which has a distribution, independent of the actual CDF and the number of samples n, and consider the Smirnov-Cramér-von Mises (SCvM) statistic [32, 123], defined as

$$\omega^2 := n \int_{\mathbb{R}} (F(x) - F_n(x))^2 \, \mathrm{d}F(x).$$

Based on an approximation of the desired PDF, f_{γ_n} (depending on γ_n) and thus the CDF, F_{γ_n}, we choose the regularization parameter such that F_{γ_n} satisfies the SCvM statistic optimally, by solving (in terms of γ_n) the following equation:

$$\int_{\mathbb{R}} \left(F_{\gamma_n}(x) - F_n(x)\right)^2 \mathrm{d}F(x) = \frac{m_{\omega^2}}{n},$$

where m_{ω^2} is the mean of the SCvM statistic, ω^2. In the one-dimensional case, a simplified expression can be derived [3, 123], i.e.,

$$\sum_{j=1}^{n} \left(F_{\gamma_n}(\bar{X}_j) - \frac{i-0.5}{n}\right)^2 = m_{\omega^2} - \frac{1}{12n}, \tag{4.16}$$

with $\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_n$, the ordered array of samples $X_1, X_2, \ldots, X_n$. It can be proved (details in [131]) that, by solving Equation (4.16) under the assumption that the solution is in the form of a cosine series expansion, a regularization parameter can be determined, which provides an almost optimal rate of convergence in n towards the desired PDF.

Next to the improvement of the method's convergence, the quality of the PDF approximation, in terms of the considered expansion coefficients, is also influenced by the regularization parameter.

In order to assess the impact of γ_n on the quality of approximation, we employ the well-known *Mean Integrated Squared Error* (MISE), which is commonly used in density estimation (also known as the *risk function*). The formal definition reads

$$\mathbb{E}\left[\left\|f_n - f\right\|_2^2\right] = \mathbb{E}\left[\int_{\mathbb{R}} \left(f_n(x) - f(x)\right)^2 \mathrm{d}x\right].$$

In our case, the MISE can be decomposed into two terms (see [87]), as follows,

$$\mathbb{E}\left[\int_0^{\pi} \left(f_n(x) - f(x)\right)^2 \mathrm{d}x\right] = \sum_{k=1}^{N} \mathbb{V}\mathrm{ar}\left[\tilde{A}_k\right] + \sum_{k=N+1}^{\infty} A_k^2,$$

where the A_k are the "true" coefficients from Equation (4.2), and the $\tilde{A}_k$ are from Equation (4.12). The MISE, as it is defined, is the summation of the bias and the variance of the estimator. In the Fourier cosine expansions context, an increasing N implies smaller bias (but bigger variance). The opposite also holds i.e. small N produces more bias and

smaller variance. We need to compute the variance of the data-driven coefficients, $\tilde{A}_k$. By Equation (4.12), we have

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left[\tilde{A}_k\right] &= \mathbb{V}\mathrm{ar}\left[\frac{1}{1+\gamma_n k^{2(p+1)}}\frac{1}{n}\sum_{j=1}^{n}\cos(k\theta_j)\right] \\ &= \frac{1}{\left(1+\gamma_n k^{2(p+1)}\right)^2}\frac{1}{n^2}\sum_{j=1}^{n}\mathbb{V}\mathrm{ar}\left[\cos(k\theta_j)\right].\end{aligned}$$

By basic trigonometric properties, this variance can be computed as

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left[\cos(kx)\right] &= \mathbb{E}\left[\cos^2(kx)\right] - \left(\mathbb{E}\left[\cos(kx)\right]\right)^2 \\ &= \int_0^\pi \cos^2(kx)f(x)\mathrm{d}x - \left(\int_0^\pi \cos(kx)f(x)\mathrm{d}x\right)^2 \\ &= \int_0^\pi \left(\frac{1+\cos(2kx)}{2}\right)f(x)\mathrm{d}x - A_k^2 = \frac{1}{2}+\frac{A_{2k}}{2}-A_k^2,\end{aligned}$$

where the definition of the expansion coefficients is used in steps 2 and 3.

The expression for the variance of $\tilde{A}_k$ then reads

$$\mathbb{V}\mathrm{ar}\left[\tilde{A}_k\right] = \frac{1}{\left(1+\gamma_n k^{2(p+1)}\right)^2}\frac{1}{n}\left(\frac{1}{2}+\frac{1}{2}A_{2k}-A_k^2\right),$$

and the MISE is thus given by

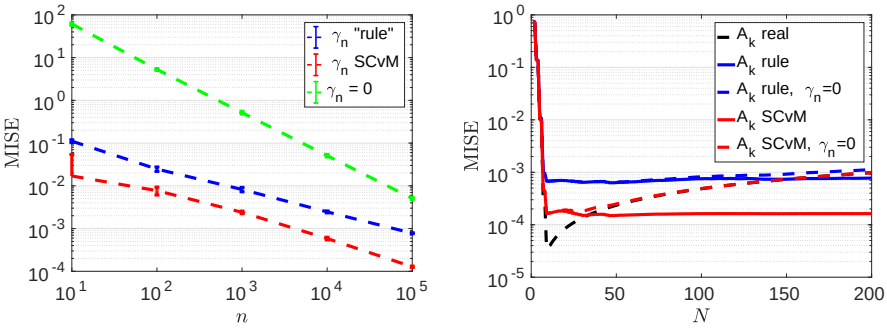
$$\mathrm{MISE} = \frac{1}{n}\sum_{k=1}^{N}\frac{1}{\left(1+\gamma_n k^{2(p+1)}\right)^2}\left(\frac{1}{2}+\frac{1}{2}A_{2k}-A_k^2\right)+\sum_{k=N+1}^{\infty}A_k^2. \tag{4.17}$$

EXAMPLE

The error measure defined in Equation (4.17) is employed to analyse the influence of the regularization. We use the standard normal distribution as a reference test case. The coefficients that are based on the available analytic solution are replaced by the corresponding data-driven coefficients that depend on γ_n.

In Figure 4.2(a), we present the convergence results for different regularization parameters. Next to the rules suggested by Equations (4.15) and (4.16), we also include the case $\gamma_n = 0$ to highlight the benefits of employing the regularization approach. The obtained results confirm the improvements provided by both γ_n-rules, with the almost optimal γ_n given by the SCvM statistic.

A second aspect which is influenced by γ_n is the accuracy with respect to the number of expansion terms N in Equation (4.14). For this, in Figure 4.2(b) we present the MISE for the standard normal distribution when different coefficients A_k are employed: A_k by the γ_n-rule (4.15) (red lines), A_k by the SCvM (4.16) (blue lines) and, as a reference, the A_k-coefficients obtained by the ChF (black dashed line). We notice that, when $\gamma_n = 0$ is used in the MISE formula (all dashed lines), for increasing values of N, the approximation deteriorates, resulting in increasing approximation errors. In contrast, when the corresponding γ_n is used (regular lines), the error stabilizes. Since the number of expansion coefficients is typically chosen high, this property of the regularization approach is useful.



(a) Convergence in terms of n.

(b) Accuracy in terms of N.

Figure 4.2: Influence of γ_n on the convergence w.r.t. the number of samples n and the number of terms N.

OPTIMAL N-VALUES

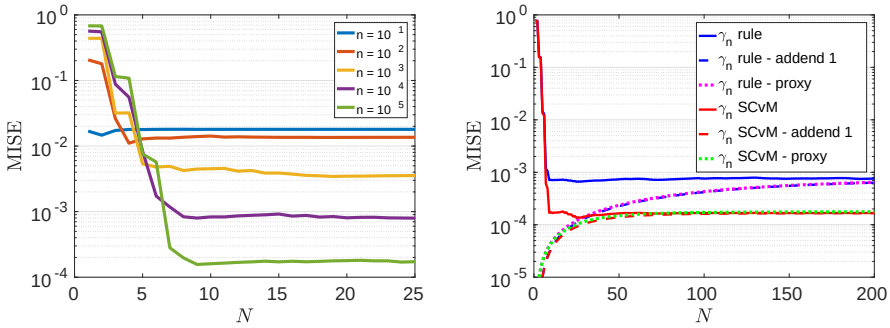
As mentioned, with an increasing number of series expansion coefficients N, the approximation based on the regularization approach does not improve any further. This fact indicates that we need to determine an *optimal* value of N, i.e. the smallest value of N for which the MISE stabilizes, see Figure 4.2(b). Using small N-values is important within the data-driven methodology, since parameter N considerably affects the performance of the method.

We propose an empirical procedure to compute the optimal value of N. The MISE in Equation (4.17) depends on the number of samples n, the number of coefficients N, and the coefficients themselves A_k. Since we wish to compute no more coefficients than necessary, we focus on the parameters n and N. Regarding the influence of the number of samples, see also the curves in Figure 4.3(a), higher n-values also require higher N-values to ensure stabilizing errors in the MISE curve. To determine a relation between n and N, we need to simplify the MISE formula as we desire a closed-form expression. We discard the second part in Equation (4.17), as it goes to zero when N increases. Within the first part, we approximate the quantity $\left(\frac{1}{2}+\frac{1}{2}A_{2k}-A_k^2\right)\approx\frac{1}{2}$. Then, the approximate formula for the MISE is found to be

$$\text{MISE}\approx\frac{1}{n}\sum_{k=1}^{N}\frac{\frac{1}{2}}{\left(1+\gamma_n k^{2(p+1)}\right)^2}=:\text{MISE}_N,$$

where n and N are directly connected, but also γ_n appears. It is possible to prove that the above MISE proxy, MISE_N, is an upper bound for the first part in Equation (4.17).

From Figure 4.3(b), we observe two important facts: the MISE proxy provides a highly satisfactory approximation for the first addend of the MISE, which converges to the MISE when N increases. By combining these two observations, we will employ the MISE proxy to determine the optimal number of terms N. Since the computation of γ_n by Equation (4.16) involves N, we use the case where γ_n is determined by Equation (4.15) (which only depends on n). Figure 4.3(b) shows that the MISE_N (to a different level of accuracy) is very similar in both cases, where the γ_n rule appears conservative, i.e. bigger N-values



(a) MISE for several values of n.

(b) Comparison between MISE and MISE_N.

Figure 4.3: Influence of the number of samples n, and the number of coefficients N in the MISE and in the MISE proxy, MISE_N.

4

are required to reach the non-decreasing error region. The proposed procedure iteratively determines whether or not we reached error stabilization by checking the differences in MISE_N between two consecutive N-values. When this difference is less than a predefined tolerance, ϵ, we have approximated the optimal N-value. Since N should grow with n, we propose to use $\epsilon := \frac{1}{\sqrt{n}}$, i.e. the expected order of accuracy for the PDF approximation should be given in terms of the number of samples.

By collecting all described components, the approximately optimal N-value becomes a function only of n. The iterative methodology is described in Algorithm 3. In Figure 4.4, we observe that the resulting optimal N function is an increasing staircase function (with a predefined floor of $N = 5$).

Algorithm 3: Optimal N-values.

Data: n, γ_n
$N_{min} = 5$
$N_{max} = \infty$
$\epsilon = \frac{1}{\sqrt{n}}$
$\text{MISE}_* = \infty$
for $N = N_{min} : N_{max}$ **do**
  $\text{MISE}_N = \frac{1}{n} \sum_{k=1}^{N} \frac{\frac{1}{2}}{(1+\gamma_n k^{2(p+1)})^2}$
  $\epsilon_N = \frac{|\text{MISE}_N - \text{MISE}_*|}{|\text{MISE}_N|}$
  if $\epsilon_N > \epsilon$ **then**
    $N_{op} = N$
  else
    Break
  $\text{MISE}_* = \text{MISE}_N$

Now, we have described the techniques to determine values for the regularization

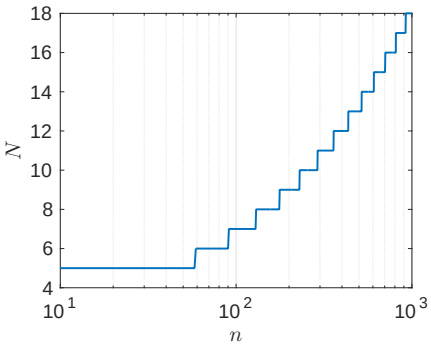


Figure 4.4: Optimal N-values in terms of n, computed by Algorithm 3.

parameter γ_n, and for the number of coefficients, N. By these, the ddCOS method is defined with Monte Carlo samples, or other data values, as the input.

4.4. APPLICATIONS OF THE DDCOS METHOD

In this section, we present some applications of the ddCOS method. The first application is an option pricing experiment, where we show the method's convergence. Subsequently, we present the performance regarding the computation of the Greeks, where ddCOS exhibits a stable convergence and can be employed with involved models, as we only need asset samples. We also compute the Greeks under the SABR model. Once the Greeks have been efficiently approximated, they can be used for the computation of the VaR and ES risk measures within the Delta-Gamma approach. All steps in this methodology can be performed by the ddCOS method.

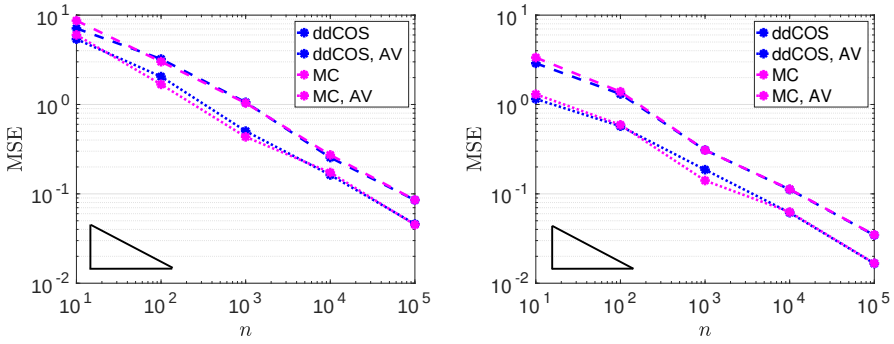
The experiments have been carried out on a computer system with the following characteristics: CPU Intel Core i7-4720HQ 2.6GHz and RAM memory of 16GB RAM. The employed software package is Matlab r2016b.

4.4.1. OPTION VALUATION AND GREEKS

First of all, we numerically test the convergence of the ddCOS method in an option valuation experiment. The GBM asset dynamics are employed, since a reference value for the option value is available by the Black-Scholes formula. The regularization parameter γ_n is set as in Equation (4.15), as for option valuation experiments the difference between this rule and γ_n based on the SCvM statistic in Equation (4.16) is not significant. Moreover, the use of the γ_n rule provides faster ddCOS estimators.

As is common in Monte Carlo experiments, the MSE is considered as the error measure. In the convergence tests, the reported values are computed as the average of 50 experiments.

The expected order of convergence for the option values is $\mathcal{O}(1/\sqrt{n})$, according to the convergence of the empirical CDF towards the true CDF in Equation (4.5). In Section 4.2.4, the application of antithetic variates in the ddCOS framework has been presented. In Figure 4.5, we confirm that this variance reduction technique provides a similar im-



(a) Call: Strike $K = 100$.

(b) Put: Strike $K = 100$.

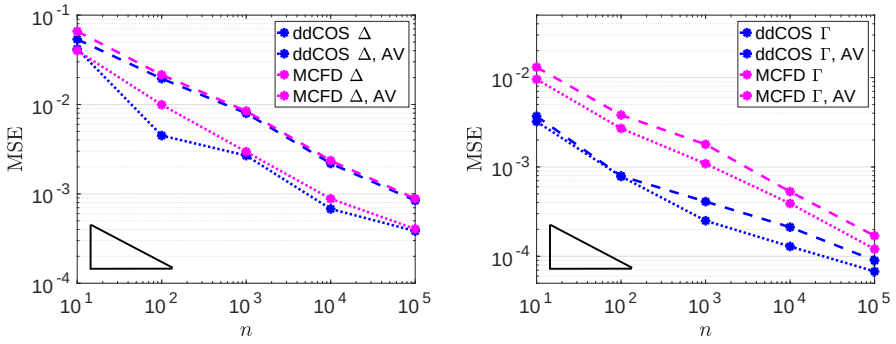
Figure 4.5: Convergence in prices of the ddCOS method: Antithetic Variates (AV); GBM, $S(0) = 100$, $r = 0.1$, $\sigma = 0.3$ and $T = 2$.

provement in terms of precision as when it is applied to the plain Monte Carlo method. Another observation is that, under this particular setting, the estimators (both ddCOS and Monte Carlo) of the put option value result in smaller variances than the call option estimators. In terms of accuracy, it is thus worth computing the put value and then use the *put-call parity* formula for call options. In addition, the use of the put together with the put-call parity is recommended since call payoff functions grow exponentially and may give rise to cancellation errors.

We have empirically shown in Figure 4.5 that the ddCOS method converges to the true price with the expected convergence rate $\mathcal{O}(1/\sqrt{n})$, which resembles the plain Monte Carlo convergence. However, by the ddCOS method, not only the option value but also the sensitivities can readily be obtained. This is an advantage w.r.t Monte Carlo-based methods for estimating sensitivities, where often, additional simulations, intermediate time-steps or prior knowledge are required.

Thus, a similar convergence test is performed for the Δ and Γ sensitivities, see Figure 4.6. As Monte Carlo-based method for the Greeks calculation we consider the *Finite Difference* method (bump and revalue, denoted as MCFD). We have chosen MCFD for the comparison because it is flexible and it does not require prior knowledge. MCFD may require one or two extra simulations, starting at $S(0) + \nu$ and $S(0) - \nu$, and the choice of optimal *shift* parameter, ν, may not be trivial. The reference Delta and Gamma values are given by the Black-Scholes formula. In Figure 4.6 we observe the expected convergence and the reduction in the variance due to the use of AV. In both experiments, while the Δ is very well approximated by the ddCOS and MCFD methods, the second derivative, Γ, appears more complicated for the MCFD method. This fact was already pointed out by Glasserman in [59]. The ddCOS estimator, however, is accurate and stable as it is based on the data-driven PDF and the ddCOS machinery.

Using $n = 100,000$, in Table 4.1 we now compare the Δ and Γ estimations obtained under the GBM dynamics for several strikes. The performance of the ddCOS method is very satisfactory as it is accurate, with small RE (averaged over K) and reproduces the



(a) Δ (Call): Strike $K = 100$.

(b) Γ: Strike $K = 100$.

Figure 4.6: Convergence in Greeks of the ddCOS method: Antithetic Variates (AV); GBM, $S(0) = 100$, $r = 0.1$, $\sigma = 0.3$ and $T = 2$.

reference values very well. The difficulties of the MCFD estimating Γ are more clearly visible.

We wish to test the ddCOS method in a more complex situation, by adding jumps in the form of a Merton jump-diffusion asset price process. To accurately compute the option sensitivities in this case gives rise to difficulties for Monte Carlo-based methods. We perform a similar experiment as before, where now the underlying asset follows the Merton jump-diffusion model, and the obtained Δ and Γ are presented in Table 4.2. In this case, the reference value is provided by the COS method at a high accuracy.

In terms of computational cost, the ddCOS method is a competitive alternative, as additional simulations are not needed. Notice that in these latter experiments AV techniques are not employed. Under the Merton dynamics, the ddCOS method takes 0.1813 seconds and MCFD 0.3149 seconds. In this case, the use of the ddCOS method reduces the computational costs, as the cost of an individual simulation by the Merton model is significantly higher than for GBM asset dynamics.

4.4.2. THE SABR MODEL

The SABR model [67] is interesting within the ddCOS framework since the ChF is not known and, furthermore, the asset path Monte Carlo simulation is not trivial. The model is a stochastic-local volatility model which is widely used in FX modelling, and is given by

$$\begin{aligned} \mathrm{d}S(t) &= \sigma(t)S^{\beta}(t)\mathrm{d}W_S(t), \quad & S(0) &= S_0 \exp(rT), \\ \mathrm{d}\sigma(t) &= \alpha\sigma(t)\mathrm{d}W_\sigma(t), & \sigma(0) &= \sigma_0, \end{aligned} \tag{4.18}$$

where $S(t) = \bar{S}(t)\exp(r(T-t))$ is the forward value of the underlying $\bar{S}(t)$, with r the interest rate, S_0 the spot price and T maturity time. The stochastic volatility process is denoted by $\sigma(t)$, with $\sigma(0) = \sigma_0$, $W_S(t)$ and $W_\sigma(t)$ are two correlated Brownian motions with correlation ρ (i.e. $W_S W_\sigma = \rho t$). The parameters of the SABR model are $\alpha > 0$ (the volatility of the volatility), $0 \leq \beta \leq 1$ (the elasticity) and ρ (the correlation coefficient).

K (% of $S(0)$)	80%	90%	100%	110%	120%
0.1			Δ		
Ref.	0.8868	0.8243	0.7529	0.6768	0.6002
ddCOS	0.8867	0.8240	0.7528	0.6769	0.6002
RE	1.1012×10^{-4}				
MCFD	0.8876	0.8247	0.7534	0.6773	0.6006
RE	7.5168×10^{-4}				
			Γ		
Ref.	0.0045	0.0061	0.0074	0.0085	0.0091
ddCOS	0.0045	0.0062	0.0075	0.0084	0.0090
RE	8.5423×10^{-3}				
MCFD	0.0045	0.0059	0.0071	0.0079	0.0083
RE	4.9554×10^{-2}				

Table 4.1: GBM option Greeks: Call, $S(0) = 100$, $r = 0.1$, $\sigma = 0.3$ and $T = 2$.

K (% of $S(0)$)	80%	90%	100%	110%	120%
			Δ		
Ref.	0.8385	0.8114	0.7847	0.7584	0.7328
ddCOS	0.8383	0.8113	0.7846	0.7585	0.7333
RE	2.7155×10^{-4}				
MCFD	0.8387	0.8118	0.7850	0.7586	0.7330
RE	3.1265×10^{-4}				
			Γ		
Ref.	0.0022	0.0024	0.0027	0.0029	0.0030
ddCOS	0.0022	0.0024	0.0027	0.0029	0.0030
RE	8.2711×10^{-3}				
MCFD	0.0023	0.0026	0.0028	0.0031	0.0033
RE	6.118×10^{-2}				

Table 4.2: Merton jump-diffusion option Greeks: Call, $S(0) = 100$, $r = 0.1$, $\sigma = 0.3$, $\mu_j = -0.2$, $\sigma_j = 0.2$ and $\lambda = 8$ and $T = 2$.

K (% of $S(0)$)	80%	90%	100%	110%	120%
			Δ		
Ref.	0.9914	0.9284	0.5371	0.0720	0.0058
ddCOS	0.9916	0.9282	0.5363	0.0732	0.0058
RE	5.2775×10^{-3}				
MCFD	0.9911	0.9279	0.5368	0.0737	0.0058
RE	5.5039×10^{-3}				

Table 4.3: Greek Δ under the SABR model: Call, $S(0) = 100$, $r = 0$, $\sigma_0 = 0.3$, $\alpha = 0.4$, $\beta = 0.6$, $\rho = -0.25$ and $T = 2$.

K (% of $S(0)$)	80%	90%	100%	110%	120%
			Δ		
Ref.	0.8384	0.7728	0.6931	0.6027	0.5086
ddCOS	0.8364	0.7703	0.6902	0.6006	0.5084
RE	2.7855×10^{-3}				
Hagan	0.8577	0.7955	0.7170	0.6249	0.5265
RE	3.1751×10^{-2}				

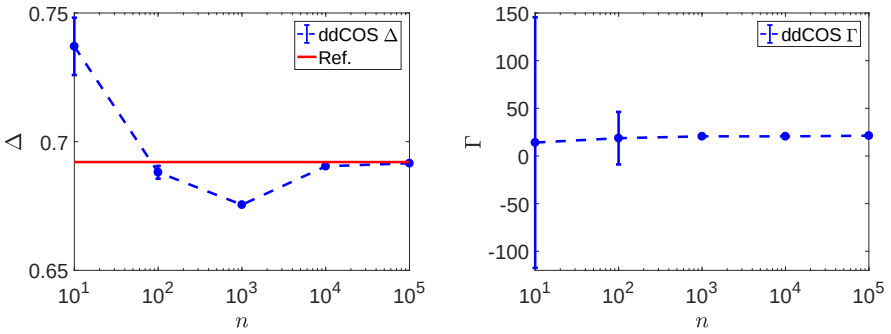
Table 4.4: Greek Δ under SABR model: Call, $S(0) = 0.04$, $r = 0.0$, $\sigma_0 = 0.4$, $\alpha = 0.8$, $\beta = 1.0$, $\rho = -0.5$ and $T = 2$.

The calculation of the Greeks under the SABR model becomes challenging but can be addressed by the ddCOS method. To employ the method, we need samples of the underlying asset at time T. Here, we make use of the one time-step SABR Monte Carlo simulation introduced in Chapter 2. This one time-step SABR simulation is based on the expression for the CDF of the conditional SABR process [75]. Thus, the ddCOS method will be combined with the one time-step SABR simulation to efficiently compute Δ and Γ under the SABR dynamics.

For the numerical experiments, we consider two parameter settings. First of all, a basic parameter set is taken, where the Hagan formula by [67] is valid and can be used as a reference. The results are presented in Table 4.3. For the second test we use a more difficult set of parameters (i.e., Set III in Table 2.2 of Chapter 2), where the Hagan formula does not provide accurate results anymore. In Table 4.4, we observe that the ddCOS provides accurate Δ-values in this case, without any problems. The reference value has been computed by the MCFD in combination with the mSABR simulation scheme from Chapter 3, with a large number of Monte Carlo paths ($n = 10,000,000$) and time steps ($4T$). The convergence in n of the ddCOS Δ estimator under the SABR dynamics is shown in Figure 4.7(a). The calculation of Γ when the underlying is governed by the SABR model is again involved and the MCFD estimation is not reliable. In Figure 4.7(b), the convergence of the ddCOS Γ estimator is presented, where we observe convergence, with impressive variance reduction.

4.4.3. VaR, ES AND THE DELTA-GAMMA APPROACH

In the evaluation of market risk, the computation of risk measures is important, and even mandatory for regulatory purposes to estimate the risk of large losses. With the risk



(a) Δ: Strike $K = 0.04$.

(b) Γ: Strike $K = 0.04$.

Figure 4.7: The ddCOS method and SABR model: Greeks convergence test.

factors denoted by S and a time horizon Δt, we define the change in S at time Δt by ΔS. The variation in S directly affects the value of a portfolio $\Pi(t,S)$, containing derivatives of S. We denote the changes in the value of the portfolio by $\Delta\Pi$, so that the definition of the loss in interval $[t,\Delta t]$ is given by

$$L := -\Delta\Pi = \Pi(t,S) - \Pi(t+\Delta t, S+\Delta S).$$

In order to manage possible large losses, we are interested in the distribution of L, specifically in the CDF, $F_L(x) = Pr(L < x)$, which can be employed to compute the risk measures VaR or ES. The formal definition of the VaR reads

$$Pr(\Delta\Pi < \text{VaR}(q)) = 1 - F_L(\text{VaR}(q)) = q,$$

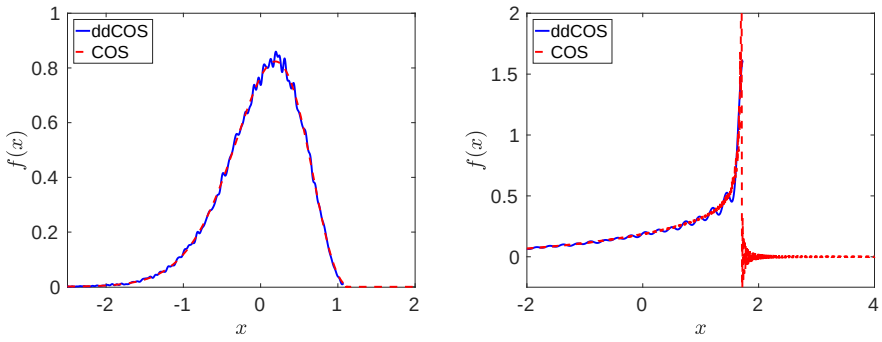
with q a predefined confidence level, whereas, given the VaR, the ES measure is computed as

$$\text{ES} := \mathbb{E}[\Delta\Pi | \Delta\Pi > \text{VaR}(q)].$$

So, VaR is given as a quantile of the loss distribution, while ES is the average of the largest possible losses.

Although simple in definition, the practical computation of these risk measures is a challenging and computationally expensive problem, especially when the changes in Π cannot be assumed linear in S. Then, VaR and ES estimation is often performed by means of an Monte Carlo method. In order to find a balance between accuracy and tractability, one of the most employed methodologies is the Delta-Gamma approximation which combines Monte Carlo path generation, a second-order Taylor expansion and the sensitivities to reduce the computational cost and capture the non-linearity in portfolio changes. The delta-gamma approximation of $\Delta\Pi$ (in the case of only one risk factor) is given by

$$\Delta\Pi \approx \sum_{i=1}^{M} w_i \frac{\partial \Pi_i}{\partial S}\Delta S + \frac{1}{2}\sum_{i=1}^{M} w_i \frac{\partial^2 \Pi_i}{\partial S^2}(\Delta S)^2,$$



(a) PDF Portfolio 1.

(b) PDF Portfolio 2.

Figure 4.8: Recovered densities of L: ddCOS vs. COS.

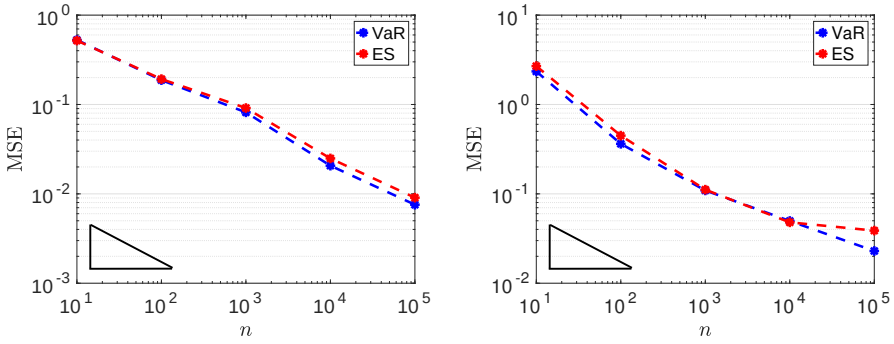
with M the number of assets depending on risk factor S, w_i and Π_i the amount and the value of asset i, respectively. The partial derivatives are evaluated at initial time t. In the case of options contracts, these partial derivatives correspond to the Δ and Γ sensitivities. It is usually assumed that the distribution of ΔS is known (normal, Student's **t**, etc). Then, by applying the Delta-Gamma approach, the distribution of the losses, F_L, and therefore the VaR and the ES are easily calculated.

The use of the ddCOS method in the context of the Delta-Gamma approach generalizes its applicability. Since the use of the ChF is not longer required, we can assume non-trivial dynamics for ΔS, where the use of Fourier inversion methods (as in [101]) would be a limitation (it may be impossible to obtain a ChF). As we have seen, by employing the ddCOS method, Δ and Γ can be computed at once and, therefore, be directly employed within the Delta-Gamma approximation. The ddCOS method can thus be used to recover the distribution of $\Delta\Pi$, whenever samples are available. This can be useful when historical data is employed, and no particular distribution is assumed.

In order to show the performance of the ddCOS method within the Delta-Gamma approach, we first repeat the experiments from [101]. Two portfolios are considered, both with the same composition (one European call and half a European put under the same underlying asset, maturity 60 days and strike $K = 101$) but different time horizons, i.e. 1 day and 10 days. We denote them by Portfolio 1 and Portfolio 2, respectively. The underlying asset follows a GBM with $S(0) = 100$, $r = 0.1$ and $\sigma = 0.3$. Change ΔS is assumed to be normally distributed.

In Figure 4.8 the recovered densities by the COS and ddCOS methods are depicted. An almost perfect fit is observed, with the expected small-sized oscillations in the data-driven approach. Since the computational domain is also driven by data, the ddCOS recovered PDF remains within the defined domain, avoiding incorrect estimations outside the domain (see the COS curve in Figure 4.8(b)).

We also employ the ddCOS method to compute the risk measures VaR and ES. The convergence of VaR and ES in terms of n is presented in Figure 4.9, with reference values provided by [101]. As expected, the convergence rate for both estimators is $\mathcal{O}(1/\sqrt{n})$.



(a) Portfolio 1: $q = 99\%$.

(b) Portfolio 2: $q = 90\%$.

Figure 4.9: VaR and ES convergence in n.

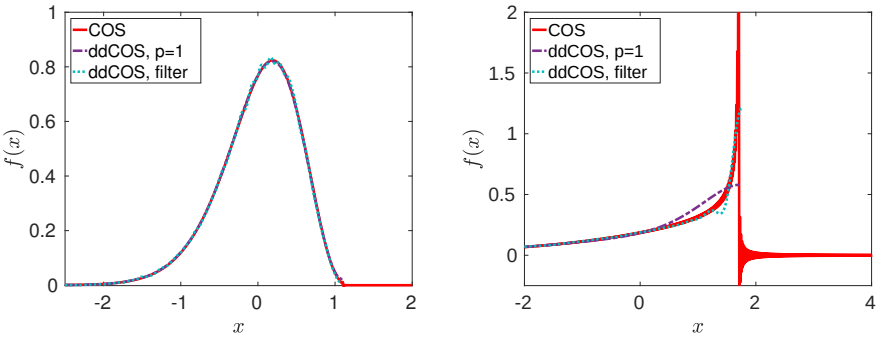
SMOOTHING THE PDF OF L

As seen in Figure 4.8, the densities estimated by the ddCOS method exhibit some artificial oscillations due to the lack of data in particular regions and the so-called *Gibbs phenomenon*. Two possibilities to avoid the appearance of these oscillations are increasing smoothing parameter p, and the application of so-called spectral filters within the ddCOS formula (4.13). By parameter p we can include derivatives of the PDF into the regularization (see Equation (4.10)). We analyze the use of $p = 1$. Filtering was already successfully applied in the context of the Delta-Gamma approximation in [101], based on the work by Ruijter et al. [112], and we refer to the references for the filter details. Adding the filter is almost trivial as it merely implies a multiplication with a specific filter term. Based on the references, we here choose the so-called 6-*th order exponential filter* within the ddCOS formula.

In Figure 4.10, the resulting densities from the application of both alternatives are presented. Whereas both smoothing techniques give highly satisfactory results for Portfolio 1, the spectral filters are superior in the case of Portfolio 2. Based on these tests, we suggest the use of a spectral filter to obtain smooth densities. Note, however, that the application of these smoothing procedures does not give us an improvement in the convergence, which is still dominated by the order of convergence in Equation (4.5).

DELTA-GAMMA APPROACH UNDER THE SABR MODEL

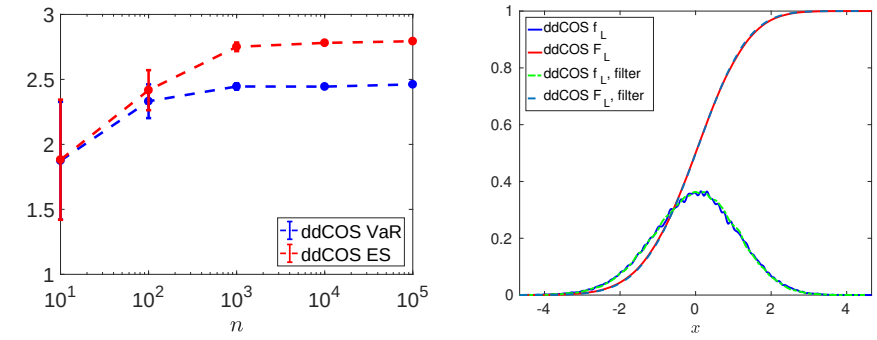
In order to further test the ddCOS method in the context of the Delta-Gamma approach, we now assume the dynamics of the underlying asset and ΔS to be governed by the SABR dynamics. In Figure 4.11(a), the obtained VaR and ES when varying n are presented. No reference is available here, since the MCFD is unstable for the Γ computation under the SABR model. We observe that, already for $n = 1,000$, a stable Γ-value is found and, even more important, the variance is negligible. By the ddCOS method, a closed-form expression for the loss distribution is also obtained. The recovered F_L and the corresponding PDF f_L are depicted in Figure 4.11(b) (for $n = 100,000$), where we also include the resulting densities when employing the exponential spectral filter.



(a) PDF Portfolio 1.

(b) PDF Portfolio 2.

Figure 4.10: Smoothed densities of L.



(a) VaR and ES: $q = 99\%$.

(b) F_L and f_L.

Figure 4.11: Delta-Gamma approach under the SABR model. Setting: $S(0) = 100$, $K = 100$, $r = 0.0$, $\sigma_0 = 0.4$, $\alpha = 0.8$, $\beta = 1.0$, $\rho = -0.5$, $T = 2$, $q = 99\%$ and $\Delta t = 1/365$.

q	10%	30%	50%	70%	90%
VaR	−1.4742	−0.5917	−0.0022	0.5789	1.3862
ES	0.1972	0.5345	0.8644	1.2517	1.8744

Table 4.5: VaR and ES under SABR model. Setting: $S(0) = 100$, $K = 100$, $r = 0.0$, $\sigma_0 = 0.4$, $\alpha = 0.8$, $\beta = 1.0$, $\rho = -0.5$, $T = 2$, and $\Delta t = 1/365$.

In Table 4.5, the VaR and ES under SABR are presented for several choices of q, ranging from 10% to 90%. Again, the results seem to be coherent.

4.5. CONCLUSIONS

In this chapter, the ddCOS method has been introduced. The method extends the COS method applicability to cases when only data samples of the underlying asset are available. The method exploits a closed-form solution, in terms of Fourier cosine expansions, of a PDF. The use of the COS machinery in combination with density estimation allowed us to develop a data-driven method which can be employed for option pricing and risk management. The ddCOS method particularly results in an efficient method for the Δ and Γ sensitivities computation, based solely on the samples. Therefore, it can be employed within the Delta-Gamma approximation for calculating risk measures. Through several numerical examples, we have empirically shown the convergence of our method. In some cases, in order to get monotonic densities, it may be beneficial to add a filter term to the ddCOS method.

A possible future extension may be the use of other basis functions. Haar wavelets are for example interesting since they provide positive densities and allow an efficient treatment of dynamic data.

The hybrid aspect presented in this chapter is expressed as a combination of data-driven density estimation, Fourier series and Monte Carlo sampling. We have thus generalized the COS method to a data-driven approach, the ddCOS method, where the knowledge of the ChF is not required. Therefore, the technique presented here can be employed as a Monte Carlo-based alternative for challenging problems in quantitative finance, like Greek estimation or risk measures computation.

CHAPTER 5

GPU Acceleration of the Stochastic Grid Bundling Method

In this chapter, we move in yet another type of algorithm. Here we focus on a parallel GPU version of the Monte Carlo-based SGBM for pricing multi-dimensional early-exercise options. To extend the method's applicability, the problem dimensions will be increased drastically. This makes SGBM very expensive in terms of computational costs on conventional hardware systems based on CPUs. A parallelization strategy of the method is developed and the GPGPU paradigm is used to reduce the execution time. An improved technique for bundling asset paths, which is more efficient on parallel hardware is introduced. Thanks to the performance of the GPU version of SGBM, a general approach for computing the early-exercise policy is proposed. Comparisons between sequential and GPU parallel versions are presented.

5.1. INTRODUCTION

In recent years, different techniques for pricing *early-exercise option contracts*, as they appear in computational finance, have been developed. In the wake of the recent financial crisis, accurately modelling and pricing these kinds of options gained additional importance, as they also form the basis for incorporation of the so-called *counterparty risk premium* to an option value, which can be seen as an option covering the fact that a counterparty may go into default before the end of a financial contract, and thus cannot pay possible contractual obligations.

The early-exercise pricing methods can be classified according to different kinds of techniques depending on whether they are based on *simulation and regression*, *transition probabilities* or *duality* approaches. Focusing on the first class of methods, Monte Carlo path generation and dynamic programming to determine an optimal early-exercise policy and the option price are typically combined. Some representative methods were developed by Longstaff and Schwartz [94] and Tsitsiklis and Van Roy [128].

The pricing problem becomes significantly more complex when the early-exercise options are *multi-* or even *high*-dimensional. One of the recent simulation-regression pricing techniques for early-exercise (Bermudan) options with several underlying assets, on which we will focus, is the SGBM, proposed by Jain and Oosterlee in [76]. The method is a hybrid of *regression* and *bundling* approaches, as it employs regressed value functions, together with bundling of the state space to approximate continuation values at different time-steps. A high biased *direct estimator* and an early-exercise policy are first computed in SGBM. The early-exercise policy is then used to determine a lower bound to

This chapter is based on the article "GPU Acceleration of the Stochastic Grid Bundling Method for Early-Exercise options". Published in *International Journal of Computer Mathematics*, 92(12):2433–2454, 2015 [90].

the true option price which is called the *path estimator*. The early-exercise policy computation involves the computation of expectations of certain *basis functions*. Usually, these basis functions are chosen by experience in such a way that an analytic solution for the expectations appearing in the pricing method is available.

In this chapter, we extend SGBM's applicability by drastically increasing the problem dimensionality, the number of bundles and, hence, the number of Monte Carlo asset paths. As the method becomes much more time-consuming then, we parallelize SGBM taking advantage of the GPGPU paradigm. It is known that GPGPU is very well suited for financial purposes and Monte Carlo techniques. In the case of pricing early-exercise options, several contributions regarding GPU implementations of different techniques appeared recently, Abbas-Turki and Lapeyre [1], Benguigui and Baude [7], Cvetanoska and Stojanovski [38], Dang et al. [39], Fatica and Phillips [52] and Pagès and Wilbertz [103]. All these papers are based on a combination of a Monte Carlo method and dynamic programming, except the Dang et al. paper, which uses PDE-based pricing methods and the Pagès and Wilbertz paper, which employs Monte Carlo simulation together with a quantization method. Our GPU version of SGBM is based on parallelization in two steps, according to the method's stages, i.e., forward in time (Monte Carlo path generator) followed by the backward stage (early-exercise policy computation). This is a novelty with respect to other methods since, although the parallelization of a Monte Carlo method is immediate, the parallelization of the backward stage is not trivial for other methods, for example not for the Least Squares Monte Carlo Method (LSM) by Longstaff and Schwartz [94] where some load balancing issues for parallel systems can appear as only the in-the-money paths are considered. Due to some timing and distribution issues observed in the original bundling procedure caused by an increasing number of bundles, we present another bundling technique for SGBM which is much more efficient and more suitable on parallel hardware.

Thanks to the performance of our SGBM GPU version, we can explore different possibilities to compute the expectations appearing within the SGBM formulation, generalize towards different option contracts and, in particular, towards more general underlying asset models. A general approach to compute the expectations needed for the early-exercise policy is proposed and evaluated. This approach is based on the computation of a ChF for the discretized underlying stochastic differential system. Once this ChF of the discretized asset dynamics is determined, we can use it for the expectations calculation, for example, for multi-dimensional local volatility asset models.

The chapter is organized as follows. In Section 5.2, the Bermudan option pricing problem is introduced. Section 5.3 describes the SGBM. The new approach to compute the expectations for the early-exercise policy is explained in Section 5.4. Section 5.5 gives details about the GPU implementation. In Section 5.6, numerical results and CPU/GPU time comparisons are shown. Finally, we conclude in Section 5.7.

5.2. BERMUDAN OPTIONS

This section defines the Bermudan option pricing problem and sets up the notations used. A Bermudan option is an option where the buyer has the right to exercise at a number of times, $t \in [t_0 = 0, \ldots, t_m, \ldots, t_M = T]$, before the end of the contract, T. $\mathbf{S}_t := (S_1(t), \ldots, S_d(t)) \in \mathbb{R}^d$ defines a d-dimensional underlying process which here follows the

dynamics given by the system of SDEs

$$\begin{aligned} \mathrm{d}S_1(t) &= \mu_1(\mathbf{S}_t)\mathrm{d}t + \sigma_1(\mathbf{S}_t)\mathrm{d}W_1(t), \\ \mathrm{d}S_2(t) &= \mu_2(\mathbf{S}_t)\mathrm{d}t + \sigma_2(\mathbf{S}_t)\mathrm{d}W_2(t), \\ &\vdots \\ \mathrm{d}S_d(t) &= \mu_d(\mathbf{S}_t)\mathrm{d}t + \sigma_d(\mathbf{S}_t)\mathrm{d}W_d(t), \end{aligned} \tag{5.1}$$

where $W_\delta(t), \delta = 1,2,\ldots,d$, are correlated standard Brownian motions. The instantaneous correlation coefficient between $W_i(t)$ and $W_j(t)$ is $\rho_{i,j}$. The correlation matrix is thus defined as

$$\mathbf{C} = \begin{pmatrix} 1 & \rho_{1,2} & \cdots & \rho_{1,d} \\ \rho_{1,2} & 1 & \cdots & \rho_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1,d} & \rho_{2,d} & \cdots & 1 \end{pmatrix},$$

and the Cholesky decomposition of $\mathbf{C}$ reads

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \rho_{1,2} & L_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1,d} & L_{2,d} & \cdots & L_{d,d} \end{pmatrix}. \tag{5.2}$$

Let $h_t := h(\mathbf{S}_t)$ be an adapted process representing the intrinsic value of the option, i.e. the holder of the option receives $\max(h_t, 0)$, if the option is exercised at time t. With the risk-free savings account process $B(t) = \exp(\int_0^t r_s \mathrm{d}s)$, where r_t denotes the instantaneous risk-free rate, we define

$$D_{t_m} := \frac{B(t_{m-1})}{B(t_m)}.$$

We consider the special case where $r_t = r$ is constant. The problem is then to compute

$$V(t_0, \mathbf{S}_{t_0}) = \max_\tau \mathbb{E}\left[\frac{h(\mathbf{S}_\tau)}{B(\tau)}\right],$$

where τ is a stopping time, taking values in the finite set $\{0, t_1, \ldots, T\}$. The value of the option at the terminal time T is equal to the option's payoff,

$$V(T, \mathbf{S}_T) = \max\left(h(\mathbf{S}_T), 0\right).$$

The conditional continuation value $Q_{t_{m-1}}$, i.e. the expected payoff at time t_{m-1}, is given by

$$Q_{t_{m-1}}(\mathbf{S}_{t_{m-1}}) = D_{t_m}\mathbb{E}\left[V(t_m, \mathbf{S}_{t_m})|\mathbf{S}_{t_{m-1}}\right]. \tag{5.3}$$

The Bermudan option value at time t_{m-1} and state $\mathbf{S}_{t_{m-1}}$ is then given by

$$V(t_{m-1}, \mathbf{S}_{t_{m-1}}) = \max\left(h\left(\mathbf{S}_{t_{m-1}}\right), Q_{t_{m-1}}\left(\mathbf{S}_{t_{m-1}}\right)\right).$$

We are interested in finding the value of the option at the initial state $\mathbf{S}_{t_0}$, i.e. $V(t_0, \mathbf{S}_{t_0})$.

5.3. STOCHASTIC GRID BUNDLING METHOD

The SGBM [76] is a simulation-based Monte Carlo method for pricing early-exercise options (such as Bermudan options). SGBM first generates Monte Carlo paths forward in time, which is followed by determining the optimal early-exercise policy, moving backwards in time in a dynamic programming framework, based on the Bellman principle of optimality. The steps involved in the SGBM algorithm are briefly described in the following paragraphs.

STEP I: GENERATION OF STOCHASTIC GRID POINTS

The grid points in SGBM are generated by Monte Carlo sampling, i.e., by simulating independent copies of sample paths, $\{\mathbf{S}_{t_0}(n), \ldots, \mathbf{S}_{t_M}(n)\}$, $n = 1, \ldots, N$, of the underlying process $\mathbf{S}_t$, all starting from the same initial state $\mathbf{S}_{t_0}$. The n-th grid point at exercise time-step t_m is then denoted by $\mathbf{S}_{t_m}(n)$, $n = 1, \ldots, N$. Depending upon the underlying stochastic process an appropriate discretization scheme (with Δt the discretization step), e.g. the Euler-Maruyama scheme, is used to generate sample paths. Sometimes the diffusion process can be simulated directly, essentially because it appears in closed form, like for the regular multi-dimensional Black–Scholes model.

STEP II: OPTION VALUE AT TERMINAL TIME

The option value at the terminal time $t_M = T$ is given by

$$V(t_M, \mathbf{S}_{t_M}) = \max\left(h(\mathbf{S}_{t_M}), 0\right),$$

with $\max\left(h(\mathbf{S}_{t_M}), 0\right)$ the payoff function.

This relation is used to compute the option value for all grid points at the final time-step.

The following steps are subsequently performed for each time-step, t_m, $m \leq M$, recursively, moving backwards in time, starting from t_M.

STEP III: BUNDLING

The grid points at exercise time t_{m-1} are clustered or *bundled* into $\mathscr{B}_{t_{m-1}}(1), \ldots, \mathscr{B}_{t_{m-1}}(\nu)$ non-overlapping sets or partitions. SGBM employs bundling to approximate the conditional distribution in Equation (5.3) using simulation. The method samples this distribution by bundling the grid points at t_{m-1} and then uses those paths that originate from the corresponding bundle to obtain a conditional sample for time t_m, see also in [76]. Different approaches for partitioning can be considered. Due to its importance in the parallel case, this decision is discussed in more detail in Section 5.3.1.

STEP IV: MAPPING HIGH-DIMENSIONAL STATE SPACE TO A LOW-DIMENSIONAL SPACE

Corresponding to each bundle $\mathscr{B}_{t_{m-1}}(\beta),\ \beta = 1,\dots,\nu$, a parametrized value function $Z : \mathbb{R}^d \times \mathbb{R}^b \mapsto \mathbb{R}$, which assigns values $Z\left(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m}\right)$ to states $\mathbf{S}_{t_m}$, is computed. Here $\alpha^{\beta}_{t_m} \in \mathbb{R}^b$ is a vector of free parameters. The objective is then to choose, for each t_m and β, a parameter vector $\alpha^{\beta}_{t_m}$ so that

$$Z\left(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m}\right) \approx V(t_m, \mathbf{S}_{t_m}).$$

After some approximations to be discussed in Section 5.3.2, $Z\left(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m}\right)$ can be computed by using ordinary least squares regression.

STEP V: COMPUTING THE CONTINUATION AND OPTION VALUES AT t_{m-1}

The continuation values for $S_{t_{m-1}}(n) \in \mathscr{B}_{t_{m-1}}(\beta),\ n = 1,\dots,N,\ \beta = 1,\dots,\nu$, are approximated by

$$\widehat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = \mathbb{E}\left[Z\left(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m}\right) | \mathbf{S}_{t_{m-1}}(n) \right].$$

The option value is then given by

$$\widehat{V}(t_{m-1}, \mathbf{S}_{t_{m-1}}(n)) = \max\left(h\left(\mathbf{S}_{t_{m-1}}(n)\right), \widehat{Q}_{t_{m-1}}\left(\mathbf{S}_{t_{m-1}}(n)\right)\right).$$

The Steps III, IV and V are repeated backward in time, till the initial time, t_0 is reached. Thus, the value $V(t_0, \mathbf{S}_{t_0})$ of the option is approximately computed.

5.3.1. BUNDLING

One of the techniques proposed (in [76]) to partition the asset path data into ν non-overlapping sets at each time t_{m-1} is the *k-means* clustering technique. The algorithm uses an iterative refinement algorithm, where, given an initial guess of cluster means, first of all the algorithm assigns each data item to one specific set (i.e. bundle) by calculating the distance between the item and the cluster mean (under some measure) and subsequently updates the sets and the cluster means. This process is repeated until some stopping condition is satisfied. The procedure needs a pre-bundling step to set approximated initial cluster means. The bundles obtained by using the k-means technique can be very irregular, i.e. the number of data items within each bundle may vary significantly. This fact can be a problem when many bundles are considered, and parallel load-balancing can be an issue too.

Since our goal is to drastically increase the number of bundles used and, in particular, the problem dimensionality, the k-means algorithm becomes too expensive in terms of computational time and memory usage in high dimensions because the iterative process of searching new sets (of high dimension) takes much time and d-dimensional data points for each Monte Carlo path and for each time-step have to be stored. In addition, it may happen that some bundles do not contain a sufficient number of data points to compute an accurate regression function when the number of bundles increases. In order to overcome these two problems of the k-means clustering technique, we propose another bundling technique in the SGBM context which is much more efficient when

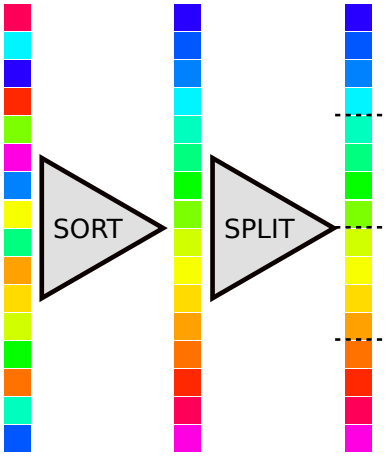


Figure 5.1: Equal partitioning scheme.

taking into account our goal of efficiency in high dimensions: it does not involve an iterative process, distributes the data equally and does not require the storage of the d-dimensional points. The details are given in the next subsection.

EQUAL-PARTITIONING TECHNIQUE

The equal-partitioning bundling technique is particularly well-suited for parallel processing; it involves two steps: sorting and splitting. The general idea is to sort the data first under some convenient criterion and then split the sorted data items into sets (i.e. bundles) of equal size. A schematic representation of this technique is shown in Figure 5.1.

With this simple approach, the drawbacks of the iterative bundling in the case of very high dimensions and an enormous number of bundles can be avoided. The sorting process is independent of the dimension of the problem, more efficient and less time-consuming than an iterative search and, furthermore, it is highly parallelizable. In addition, the storage of all Monte Carlo simulation data points is avoided. Assuming that the criterion is known in advance, we can perform the necessary computations within inside the Monte Carlo generator. That allows us to map the d-dimensional Monte Carlo points to a 1D vector and reduce amount of the stored data, i.e., storing a 2D matrix ($N \times M$) instead of storing a 3D matrix ($N \times M \times d$). The split stage assigns directly the portions of data to bundles which will contain the same number of *similar* (following some criterion) data items. Hence, the regression can be performed accurately even though the number of bundles increases in a significant way. Furthermore, the equally sized bundles allow for a better load balancing within the parallel implementation.

5.3.2. PARAMETERIZING THE OPTION VALUES

As we aim to increase the dimensions of the problem drastically, the option pricing problem may become intractable and requires the approximation of the value function. This can be achieved by a *parametrized value function* $Z : \mathbb{R}^d \times \mathbb{R}^b \mapsto \mathbb{R}$, which assigns a value

$Z(\mathbf{S}_{t_m}, \alpha)$ to state $\mathbf{S}_{t_m}$, where $\alpha \in \mathbb{R}^b$ is a vector of free parameters. The objective is to choose, corresponding to each bundle β at time point t_{m-1}, a parameter vector $\alpha^{\beta}_{t_m} := \alpha$ so that,

$$V(t_m, \mathbf{S}_{t_m}) \approx Z\left(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m}\right).$$

As in the original SGBM paper, we follow the approach of Tsitsiklis and Van Roy [128] and we use basis functions to approximate the values of the options. Hence, two important decisions have to be made: the form of the function Z and the basis functions. For each particular problem we define several basis functions, $\phi_1, \phi_2, \ldots, \phi_b$, that are typically chosen based on experience, as in the case of the LSM method [94], aiming to represent relevant properties of a given state, $\mathbf{S}_{t_m}$. In Section 5.3.2 and in Section 5.4, we present two possible choices for the basis functions, one specific and the other one more generally applicable. In our case, $Z\left(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m}\right)$ depends on $\mathbf{S}_{t_m}$ only through $\phi_k(\mathbf{S}_{t_m})$. Hence, for some function $f : \mathbb{R}^b \times \mathbb{R}^b \mapsto \mathbb{R}$, we can write $Z\left(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m}\right) = f\left(\phi_k(\mathbf{S}_{t_m}), \alpha^{\beta}_{t_m}\right)$, where

$$Z\left(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m}\right) = \sum_{k=1}^{b} \alpha^{\beta}_{t_m}(k)\phi_k(\mathbf{S}_{t_m}). \tag{5.4}$$

An exact computation of the vector of free parameters, $\alpha^{\beta}_{t_m}$, is generally not feasible. Hence, Equation (5.4) can be approximated by

$$Z\left(\mathbf{S}_{t_m}, \widehat{\alpha}^{\beta}_{t_m}\right) = \sum_{k=1}^{b} \widehat{\alpha}^{\beta}_{t_m}(k)\phi_k(\mathbf{S}_{t_m}), \tag{5.5}$$

satisfying

$$\underset{\widehat{\alpha}^{\beta}_{t_m}}{\arg\min} \sum_{n=1}^{|\mathscr{B}_{t_{m-1}}(\beta)|} \left(V(t_m, \mathbf{S}_{t_m}(n)) - \sum_{k=1}^{b} \widehat{\alpha}^{\beta}_{t_m}(k)\phi_k(\mathbf{S}_{t_m}(n))\right)^2,$$

for the corresponding bundle $\mathscr{B}_{t_{m-1}}(\beta)$. The parametrized function in Equation (5.5) is computed by using ordinary least squares regression, so that

$$V(t_m, \mathbf{S}_{t_m}(n)) = Z\left(\mathbf{S}_{t_m}(n), \widehat{\alpha}^{\beta}_{t_m}\right) + \epsilon^{\beta}_{t_m},$$

where $\mathbf{S}_{t_{m-1}}(n) \in \mathscr{B}_{t_{m-1}}(\beta)$. The regression error, $\epsilon^{\beta}_{t_m}$, can be neglected here, as we have a sufficiently large number of paths in each bundle.

COMPUTING THE CONTINUATION VALUE

By employing the parametrized option value function $Z\left(\mathbf{S}_{t_m}, \widehat{\alpha}^{\beta}_{t_m}\right)$ corresponding to bundle $\mathscr{B}_{t_{m-1}}(\beta)$, the continuation values for the grid points that belong to this bundle are approximated by

$$\widehat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = D_{t_{m-1}}\mathbb{E}\left[Z\left(\mathbf{S}_{t_m}, \widehat{\alpha}^{\beta}_{t_m}\right) | \mathbf{S}_{t_{m-1}} = \mathbf{S}_{t_{m-1}}(n)\right],$$

where $\mathbf{S}_{t_{m-1}}(n) \in \mathscr{B}_{t_{m-1}}(\beta)$. Using Equation (5.5), this can be written as

$$\begin{aligned}\widehat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) &= D_{t_{m-1}}\mathbb{E}\left[\left(\sum_{k=1}^{b}\widehat{\alpha}^{\beta}_{t_m}(k)\phi_k(\mathbf{S}_{t_m})\right)|\mathbf{S}_{t_{m-1}} = \mathbf{S}_{t_{m-1}}(n)\right] \\ &= D_{t_{m-1}}\sum_{k=1}^{b}\widehat{\alpha}^{\beta}_{t_m}(k)\mathbb{E}\left[\phi_k(\mathbf{S}_{t_m})|\mathbf{S}_{t_{m-1}} = \mathbf{S}_{t_{m-1}}(n)\right].\end{aligned} \tag{5.6}$$

The continuation value will give us a reference value to compute the early-exercise policy and it will be used by the estimators in Section 5.3.3.

CHOICE OF BASIS FUNCTIONS

The basis functions ϕ_k should be chosen such that the expectations in Equation (5.6), $\mathbb{E}\left[\phi_k(\mathbf{S}_{t_m})|\mathbf{S}_{t_{m-1}} = \mathbf{S}_{t_{m-1}}(n)\right]$, are easy to calculate, i.e. they are preferably known in closed form or otherwise have analytic approximations. The intrinsic value of the option, $h(\cdot)$, is usually an important and useful basis function, especially in high problem dimensions. In this section, we describe the choices of basis functions for different Bermudan basket options with the underlying processes following GBM; we thus choose in Equation (5.1),

$$\mu_\delta(\mathbf{S}_t) = (r_t - q_\delta)S_\delta(t), \qquad \sigma_\delta(\mathbf{S}_t) = \sigma_\delta S_\delta(t),$$

where r_t is the risk-free rate and q_δ and σ_δ, $\delta = 1,2,\ldots,d$, are the dividend yield rates and the volatility, respectively.

In the case of a *geometric* Bermudan basket option, the intrinsic value of the option is defined by

$$h\left(\mathbf{S}_{t_m}\right) = \left(\prod_{\delta=1}^{d} S_\delta(t_m)\right)^{\frac{1}{d}} - K, \quad h\left(\mathbf{S}_{t_m}\right) = K - \left(\prod_{\delta=1}^{d} S_\delta(t_m)\right)^{\frac{1}{d}},$$

for *call* or *put* options, respectively, where K is the strike value of the option contract.

Then, basis functions that make sense are given by

$$\phi_k(\mathbf{S}_{t_m}) = \left(\left(\prod_{\delta=1}^{d} S_\delta(t_m)\right)^{\frac{1}{d}}\right)^{k-1}, \; k = 1,\ldots,b. \tag{5.7}$$

The expectation in Equation (5.6) requires the computation of the expectations of the expression in Equation (5.7), given $\mathbf{S}_{t_{m-1}}$,

$$\mathbb{E}\left[\phi_k(\mathbf{S}_{t_m})|\mathbf{S}_{t_{m-1}}\right] = \left(P_{t_{m-1}}\mathrm{e}^{\left(\bar{\mu}+\frac{(k-1)\bar{\sigma}^2}{2}\right)\Delta t}\right)^{k-1}, \; k = 1,\ldots,b, \tag{5.8}$$

where,

$$P_{t_{m-1}} = \left(\prod_{\delta=1}^{d} S_\delta(t_{m-1})\right)^{\frac{1}{d}}, \; \bar{\mu} = \frac{1}{d}\sum_{\delta=1}^{d}\left(r_t - q_\delta - \frac{\sigma_\delta^2}{2}\right), \; \bar{\sigma}^2 = \frac{1}{d^2}\sum_{i=1}^{d}\left(\sum_{j=1}^{d} L_{ij}^2\right)^2,$$

with L_{ij} – matrix element i, j of $\mathbf{L}$, the Cholesky decomposition of the correlation matrix, given by Equation (5.2).

For the *arithmetic* Bermudan basket options, the intrinsic values of the call and put options are

$$h\left(\mathbf{S}_{t_m}\right) = \left(\frac{1}{d}\sum_{\delta=1}^{d} S_\delta(t_m)\right) - K, \quad h\left(\mathbf{S}_{t_m}\right) = K - \left(\frac{1}{d}\sum_{\delta=1}^{d} S_\delta(t_m)\right).$$

The basis functions are chosen as

$$\phi_k(\mathbf{S}_{t_m}) = \left(\frac{1}{d}\sum_{\delta=1}^{d} S_\delta(t_m)\right)^{k-1}, k = 1, \ldots, b.$$

Again, the continuation value, as given by Equation (5.6), requires us to compute,

$$\mathbb{E}\left[\phi_k(\mathbf{S}_{t_m})|\mathbf{S}_{t_{m-1}}\right] = \mathbb{E}\left[\left(\frac{1}{d}\sum_{\delta=1}^{d} S_\delta(t_m)\right)^{k-1}|\mathbf{S}_{t_{m-1}}\right], k = 1, \ldots, b. \tag{5.9}$$

The expectation in Equation (5.9) can be expressed as a linear combination of moments of the geometric average of the assets [76], i.e.

$$\left(\sum_{\delta=1}^{d} S_\delta(t_m)\right)^k = \sum_{k_1+k_2+\cdots+k_d=k} \binom{k}{k_1, k_2, \ldots, k_d} \prod_{1\le\delta\le d} \left(S_\delta(t_m)\right)^{k_\delta},$$

where,

$$\binom{k}{k_1, k_2, \ldots, k_d} = \frac{k!}{k_1!\,k_2!\cdots k_d!},$$

which can be computed in a straightforward way by Equation (5.8).

5.3.3. ESTIMATING THE OPTION VALUE

The estimation of the option value is the final step in SGBM. In this chapter, we follow the original proposed idea in [76] and we consider the so-called *direct estimator* and *path estimator*, which can give us a confidence interval for the option price. SGBM has been developed as a so-called duality-based method, as originally introduced by Haugh and Kogan [70] and Rogers [108]. Using a duality-based methods an upper bound on the option value for a given exercise policy can be obtained, by adding a non-negative quantity that penalizes potentially incorrect exercise decisions made by the sub-optimal policy. The SGBM direct estimator is typically biased high, i.e., it is often an upper bound estimator. The definition of the direct estimator is

$$\widehat{V}(t_{m-1}, \mathbf{S}_{t_{m-1}}(n)) = \max\left(h\left(\mathbf{S}_{t_{m-1}}(n)\right), \widehat{Q}_{t_{m-1}}\left(\mathbf{S}_{t_{m-1}}(n)\right)\right),$$

where $n = 1, \ldots, N$. The final option value reads

$$\mathbb{E}[\widehat{V}(t_0, \mathbf{S}_{t_0})] = \frac{1}{N}\sum_{n=1}^{N} \widehat{V}(t_0, \mathbf{S}_{t_0}(n)).$$

The direct estimator corresponds to Step V of the initial description in Section 5.3.

Once the optimal early-exercise policy has been obtained, the path estimator, which is typically biased low, can be developed based on the early-exercise policy. The resulting confidence interval is useful, because, depending on the problem at hand, sometimes the path estimator and sometimes the direct estimator appears superior in terms of accuracy. The obtained confidence intervals are generally small, indicating accurate SGBM results. In order to compute the low-biased estimates, we generate a new set of paths, as is common for duality-based Monte Carlo methods, $\mathbf{S}(n) = \{\mathbf{S}_{t_1}(n), \ldots, \mathbf{S}_{t_M}(n)\}$, $n = 1, \ldots, N_{\mathrm{L}}$, using the same scheme as followed for generating the paths in the case of the direct estimator and the bundling stage is performed considering the new set of paths. Along each path, the approximate optimal policy exercises at,

$$\widehat{\tau}^* \left(\mathbf{S}(n)\right) = \min\{t_m : h\left(\mathbf{S}_{t_m}(n)\right) \geq \widehat{Q}_{t_m}\left(\mathbf{S}_{t_m}(n)\right),\, m = 1, \ldots, M\},$$

where $\widehat{Q}_{t_m}(\mathbf{S}_{t_m}(n))$ is previously computed using Equation (5.6). The path estimator is then defined by $v(n) = h\left(\mathbf{S}_{\widehat{\tau}^*(\mathbf{S}(n))}\right)$. Finally, the low-biased estimate given by the path estimator is

$$\underline{V}(t_0, \mathbf{S}_{t_0}) = \lim_{N_{\mathrm{L}}} \frac{1}{N_{\mathrm{L}}} \sum_{n=1}^{N_{\mathrm{L}}} v(n),$$

where $V_{t_0}(\mathbf{S}_{t_0})$ is the true option value.

To summarize and clarify the general idea behind SGBM, in Algorithm 4 we show the corresponding algorithm considering both, direct estimator and path estimator.

Algorithm 4: SGBM.

Data: $\mathbf{S}_{t_0}, K, \mu_\delta, \sigma_\delta, \rho_{i,j}, T, N, M$
Pre-Bundling (only in k-means case).
Generation of the grid points (Monte Carlo). Step I.
Option value at terminal time $t = M$. Step II.
for *Time* $t = (M-1)\ldots 1$ **do**
    Bundling. Step III.
    for *Bundle* $\beta = 1 \ldots \nu$ **do**
        Exercise policy (Regression). Step IV.
        Continuation value. Step V.
        Direct estimator. Step V.
Generation of the grid points (Monte Carlo). Step I.
Option value at terminal time $t = M$. Step II.
for *Time* $t = (M-1)\ldots 1$ **do**
    Bundling. Step III.
    for *Bundle* $\beta = 1 \ldots \nu$ **do**
        Continuation value. Step V.
        Path estimator. Step V.

5.4. CONTINUATION VALUE COMPUTATION: NEW APPROACH

In Section 5.3.2, we showed that, for certain derivative contracts (geometric and arithmetic Bermudan basket options) and underlying processes (GBM), analytic solutions were available for the expectations that we needed to compute within SGBM. However, finding suitable basis functions resulting in analytic expressions for the expectations is not always possible for all underlying models. For that, in this chapter, we propose a different way to compute the expected values in Equation (5.6) which is more generally applicable, for example, we can also work with local volatility models with this approach. The approach consists of, first, discretizing the asset process and then obtaining the multi-variate ChF of the discrete process. This procedure is based on the work of Ruijter and Oosterlee in [111]. Once we have this ChF of the discrete process, we can compute the required expectations for certain choices of basis functions.

5.4.1. DISCRETIZATION, JOINT CHF AND JOINT MOMENTS

Taking into account the correlation between Brownian motions and the Cholesky decomposition given by Equation (5.2) and using an Euler-Maruyama scheme with time-step $\Delta t = t_{m+1} - t_m$, we can discretize the general system of SDEs defined by Equation (5.1), with independent Brownian motions, $\Delta\tilde{W}_\delta(t_{m+1}) = \tilde{W}_\delta(t_{m+1}) - \tilde{W}_\delta(t_m)$, $\delta = 1,2,\ldots,d$, as follows

$$
\begin{aligned}
S_1(t_{m+1}) &= S_1(t_m) + \mu_1(\mathbf{S}_{t_m})\Delta t + \sigma_1(\mathbf{S}_{t_m})\Delta\tilde{W}_1(t_{m+1}),\\
S_2(t_{m+1}) &= S_2(t_m) + \mu_2(\mathbf{S}_{t_m})\Delta t + \rho_{1,2}\sigma_2(\mathbf{S}_{t_m})\Delta\tilde{W}_1(t_{m+1}) + L_{2,2}\sigma_2(\mathbf{S}_{t_m})\Delta\tilde{W}_2(t_{m+1}),\\
&\vdots\\
S_d(t_{m+1}) &= S_d(t_m) + \mu_d(\mathbf{S}_{t_m})\Delta t + \rho_{1,d}\sigma_d(\mathbf{S}_{t_m})\Delta\tilde{W}_1(t_{m+1}) + \cdots + L_{d,d}\sigma_d(\mathbf{S}_{t_m})\Delta\tilde{W}_d(t_{m+1}),
\end{aligned}
$$

where $\mathbf{S}_{t_m} = (S_1(t_m), S_2(t_m), \ldots, S_d(t_m))$.

The d-variate ChF of process $\mathbf{S}_{t_{m+1}}$, given $\mathbf{S}_{t_m}$, is given by

$$
\begin{aligned}
\psi_{\mathbf{S}_{t_{m+1}}}(u_1,u_2,\ldots,u_d|\mathbf{S}_{t_m}) &= \mathbb{E}\left[\exp\left(\sum_{j=1}^{d} i u_j S_j(t_{m+1})\right)|\mathbf{S}_{t_m}\right]\\
&= \mathbb{E}\left[\exp\left(\sum_{j=1}^{d} i u_j \left(S_j(t_m) + \mu_j(\mathbf{S}_{t_m})\Delta t + \sigma_j(\mathbf{S}_{t_m})\sum_{k=1}^{j} L_{k,j}\Delta\tilde{W}_k(t_{m+1})\right)\right)|\mathbf{S}_{t_m}\right]\\
&= \exp\left(\sum_{j=1}^{d} i u_j \left(S_j(t_m) + \mu_j(\mathbf{S}_{t_m})\Delta t\right)\right)\cdot\prod_{k=1}^{d}\left(\mathbb{E}\left[\exp\left(\sum_{j=k}^{d} i u_j L_{k,j}\sigma_j(\mathbf{S}_{t_m})\Delta\tilde{W}_k(t_{m+1})\right)\right]\right)\\
&= \exp\left(\sum_{j=1}^{d} i u_j \left(S_j(t_m) + \mu_j(\mathbf{S}_{t_m})\Delta t\right)\right)\cdot\prod_{k=1}^{d}\left(\psi_{\mathcal{N}(0,\Delta t)}\left(\sum_{j=k}^{d} u_j L_{k,j}\sigma_j(\mathbf{S}_{t_m})\right)\right),
\end{aligned}
$$

where i is the imaginary unit and $\psi_{\mathcal{N}(\mu,\sigma^2)}(u) = \exp\left(i\mu u - 0.5\sigma^2 u^2\right)$ is the ChF of a normal random variable with mean μ and variance σ^2.

The joint moments of the product of several random variables can be obtained by evaluating the following expression (more details in [85] or [129]), based on the ChF of the discrete process

$$\begin{aligned} M_{\mathbf{S}_{t_{m+1}}} &= \mathbb{E}\left[(S_1(t_{m+1}))^{c_1}(S_2(t_{m+1}))^{c_2}\cdots(S_d(t_{m+1}))^{c_d}\,|\mathbf{S}_{t_m}\right] \\ &= (-i)^{c_1+c_2+\cdots+c_d}\left[\frac{\partial^{c_1+c_2+\cdots+c_d}\psi_{\mathbf{S}_{t_{m+1}}}(\mathbf{u}|\mathbf{S}_{t_m})}{\partial u_1^{c_1}\partial u_2^{c_2}\cdots\partial u_d^{c_d}}\right]_{\mathbf{u}=\mathbf{0}}, \end{aligned} \tag{5.10}$$

being $\mathbf{u} = (u_1, u_2, \ldots, u_d)$.

If we take the basis functions as used in Equation (5.5), to be a product of the underlying processes to some power, i.e.,

$$\phi_k(\mathbf{S}_{t_m}) = \left(\prod_{\delta=1}^{d} S_\delta(t_m)\right)^{k-1}, k = 1, \ldots, b, \tag{5.11}$$

the expected value in Equation (5.6) can be easily computed by means of Equation (5.10).

The approximation obtained in this way is, in general, worse than the analytic value because the ChF on which the expectations are based is related to the Euler-Maruyama SDE discretization and thus less accurate. However, since we can increase the number of bundles and, in particular, time-steps drastically (due to our GPU implementation), we can employ a suitable combination of these to price products without analytic solution for the ChF. More involved models for the underlying asset dynamics can be chosen, for which we do not have analytic expressions for these expectations. In Section 5.6.4, more details are given.

5.5. IMPLEMENTATION DETAILS

5.5.1. GPGPU: CUDA

GPGPU is the use of graphics hardware (GPUs) in order to perform computations that are typically performed by central processing units (CPUs). In this sense, the GPU can be seen as a co-processor of the CPU.

CUDA is a parallel computing platform and programming model developed and maintained by NVIDIA (see [35]) for its GPUs. CUDA eases the coding of algorithms by means of an extension of traditional programming languages, like C. In this section, we give some basic details about the platform that are necessary to understand the parallel version of SGBM. More information about CUDA can be obtained from [30, 34, 51, 81, 82, 113, 134].

PROGRAMMING MODEL

The basic operational units in CUDA are the *CUDA threads*. The CUDA threads are processes that are executed in parallel. The threads are grouped into *blocks* of the same size and blocks of threads are grouped into *grids*. Such organization eases the adaptability to different problems. One grid executes special functions called *kernels* so that all threads of the grid execute the same instructions in parallel.

MEMORY HIERARCHY

CUDA threads may access data from multiple memory spaces during their execution. Each thread manages its own *registers* and has private *local memory*. Each thread block has *shared memory* visible to all threads of the block and with the same lifetime as the block. All threads have access to the same *global memory*. There are also two additional read-only memory spaces accessible by all threads: the *constant and texture memory* spaces. In terms of speed, the registers represent the fastest memory (but also the smallest) while the global and local memories are the slowest. In recent GPU architectures (Kepler GK110 and GK110B [79], for example), several levels of cache are included to improve the accessibility.

MEMORY TRANSFERS

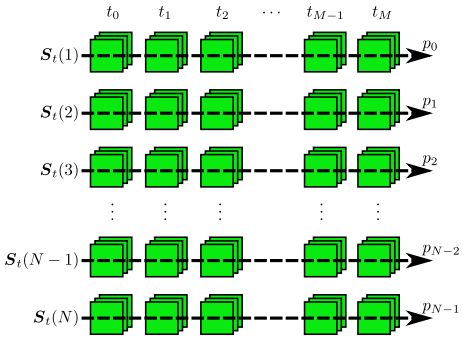
A key aspect in each parallel system is the memory transfer. Specifically in a GPU framework, the transfers between the CPU main memory and the GPU memory space are of great importance for the performance of the implementation. The number of transfers and the amount of data for each transfer are important. CUDA provides different ways to allocate and move data. In that sense, one of the CUDA 5.5 features is Unified Virtual Addressing (UVA) which allows asynchronous transfers and page-locked memory accesses by using a single address space for both CPU and GPU.

5.5.2. PARALLEL SGBM

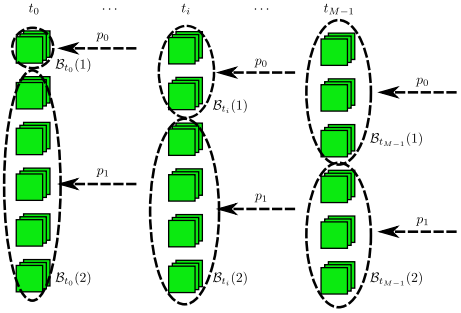
The original SGBM implementation [76] was done in Matlab. The first step of our implementation was to code an efficient sequential version of the method in the C programming language, because it eases the parallel coding in CUDA. In addition, both implementations can be used to compare results and execution times. Once we obtained the C-version, we coded the CUDA-version aiming to parallelize the suitable parts of the method. In addition, we also carried out the implementation of SGBM with the equal-partitioning bundling technique in C and CUDA.

Since SGBM is based on two clearly separated stages, we parallelize them separately. First of all, the Monte Carlo path generation is parallelized (Step I). As is well-known, Monte Carlo methods are very well suitable for parallelization, because of characteristics like a very large number of simulations and data independence. In Figure 5.2(a), we see schematically how the parallelization is done where $p_0, p_1, \ldots, p_{N-1}$ are the CUDA threads. The second main stage of SGBM is the regression and the computation of the continuation and option values (Steps IV and V) in each bundle, backwards in time. Due to the data dependency between time-steps, the way to parallelize this stage of the method is by parallelizing over the bundles, performing the calculations in each bundle in parallel. Schematic and simplified representations with two bundles are given in Figure 5.2(b) and Figure 5.2(c) for the two considered bundling techniques, k-means and equal-partitioning. Note that, actually, several stages of parallelization are performed, one per time-step. Between each parallel stage, the bundling (Step III) is carried out. This step can be also parallelized in the case of equal-partitioning. In the case of k-means, bundling has to be done sequentially.

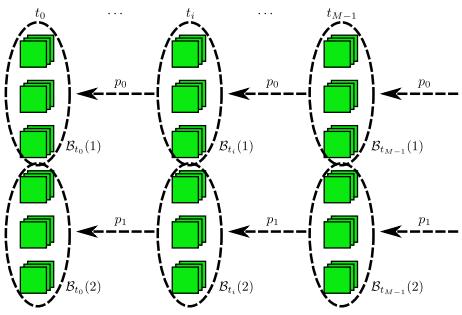
As mentioned, the memory transfers between CPU and GPU are key since we have to move huge amounts of data. An increase of the number of bundles implies a drastic



(a) Monte Carlo stage.



(b) Bundling stage using k-means.



(c) Bundling stage using equal-partitioning.

Figure 5.2: Parallel SGBM. Monte Carlo and bundling stages.

increase of the number of Monte Carlo paths. Data has to be moved to CPU memory between the stages of parallelization, Monte Carlo path simulation and bundle computations. For example, in case of the Monte Carlo scenario generator, we need to store in and move from GPU global memory to CPU memory $N \times M \times d$ doubles[1] (8 bytes).

In the following subsections, we will show more specific details of the CUDA implementation of the two SGBM versions, the original one (with k-means bundling) and the new one (with equal-partitioning).

PARALLEL MONTE CARLO PATHS

In a GPU very large numbers of threads can be managed, so we launch one CUDA thread per Monte Carlo path. The necessary random numbers are obtained "on the fly" in each thread by means of the cuRAND library [37]. In addition, the intrinsic value of the option and also the computation of the expectation in Equation (5.6) are performed within the Monte Carlo generator, decreasing the number of launched loops i.e. we perform all necessary computations for the bundling and pricing stages taking advantage of the parallel Monte Carlo simulation. The intermediate results are stored in an array defined inside the kernel which can be allocated in the registers, speeding up the memory accesses.

In the original implementation (using k-means bundling), we need to store all Monte Carlo data because it will be used during the bundling stage. This is a limiting factor since the maximum memory storage in global memory is easily reached. Furthermore, we have to move each value obtained from GPU global memory to CPU main memory and, with a significant number of paths, time-steps and dimensions, this transfer can become really expensive.

When *equal-partitioning bundling* is considered, the main difference is that we also perform calculations for the sorting criterion inside the Monte Carlo generator, avoiding the storage of the complete Monte Carlo simulation and the transfer of data from GPU global memory to CPU main memory. This approach gives us a considerable performance improvement and allows us to increase drastically the dimensionality and also the number of Monte Carlo paths (depending on the number of bundles).

BUNDLING SCHEMES

For the k-means clustering the computations of the distances between the cluster means and all of the stochastic grid points have been parallelized. However, the other parts must be performed sequentially. The very large number of bundles makes this very expensive, since the bundling must be done in each time-step.

As mentioned, equal-partitioning bundling involves two operations: sorting and splitting. It is well known that efficient sorting algorithms are a challenging research topic (see [84], for example). In parallel systems, like GPUs, this is even more important. In recent years, a great effort has taken place, see, for example, [77], [98], [115] or [126], to adapt classical algorithms to the new parallel systems and developing new parallel techniques for sorting. Several libraries for GPU programming appeared in this field, like Thrust [127], CUB [33], Modern GPU [99], clogs [27] or VexCL [132].

In our work, we take advantage of the CUDA Data-Parallel Primitives Library (CUDPP), described in [36]. CUDPP is a library of data-parallel algorithm primitives that are im-

[1] Double-precision floating-point format.

portant building blocks for a wide variety of data-parallel algorithms, including sorting. The sorting algorithms are based on the work of Satish, Harris and Garland [115]. The authors showed the performance of several sorting algorithms implemented on GPUs. Following the results of their work, we choose the parallel Radix sort which is included in version 2.1 of CUDPP. In addition, CUDPP provides a kernel-level API[2], allowing us to avoid the transfers between stages of parallelization.

Once the sorting stage has been performed, the splitting stage is immediate since the size of the bundles is known, i.e. N/ν. Each CUDA thread manages a pointer which points at the beginning of the corresponding region of global memory for each bundle. The global memory allocation is made for all bundles which means that the bundle's memory areas are adjacent and the accesses are faster.

ESTIMATORS

When the bundling stage is done, the exercise policy and the final option values can be computed by means of direct and path estimators. In order to use the GPU memory efficiently, the obtained Monte Carlo paths (once bundled) are sorted with respect to the bundles and stored in that way. We minimize the global memory accesses and the sorted version is much more efficient because, again, the bundle's memory areas are contiguous. For this purpose, we use again the CUDPP 2.1 library. Note that the data is already sorted in the case of equal-partitioning bundling.

For the direct estimator, one CUDA thread per bundle is launched at each time-step. For each bundle, the regression and option values are calculated on the GPU. All threads collaborate in order to compute the continuation value which determines the early-exercise policy. As we mentioned before, $T/\Delta t$ stages of parallelization are performed, i.e. one per time-step. Hence, in each stage of parallelization, we need to transfer only the data corresponding to the current time-step from CPU memory to GPU global memory. This data cannot be transferred at once because we wish to take advantage of the optimized parallel sorting libraries for the sorting step after the bundling stage. This allows us to minimize and improve the global memory accesses when the sequential bundling is employed, i.e. considering k-means clustering. This fact does not imply a reduction of the performance since the total transferred amount is the same. Note again that, employing equal-partitioning bundling, these transfers are avoided since this bundling technique is fully parallelizable.

Once the early-exercise policy is determined, the path estimator can be executed. For that, a new set of grid points has to be generated following the procedure described in Section 5.5.2. In the case of the path estimator, the parallelization can be done over paths because the early-exercise policy is already known (given by the previous computation of the direct estimator) and it is not needed to perform the regression again. One CUDA thread per path is launched and it computes the optimal exercise time and the cash flows according to the policy. Another sorting stage is needed to assign the paths to the corresponding bundle. Again, we use the sorting functions of CUDPP 2.1 for that purpose.

In the final stage for both estimators, a summation is necessary to determine the final option price. We take advantage of the Thrust library [127] to perform this reduction on

[2] Application Programming Interface.

the GPU.

We present a schematic representation of parallel SGBM in Algorithm 5, highlighting the parts that have been parallelized and considering the equal-partitioning version. The variables payoffData, expData and critData correspond to three matrices in which we store the necessary computations for the pricing, regression and sorting stages, respectively, for each path and each exercise time. The α_t^β values are computed in the regression step and determine the early-exercise policy, which is later used in the path estimator calculation.

Algorithm 5: Parallel SGBM.

Data: $\mathbf{S}_{t_0}, K, \mu_\delta, \sigma_\delta, \rho_{i,j}, T, N, M$
// Generation of the grid points (Monte Carlo). Step I.
// Option value at terminal time $t = M$. Step II.
[payoffData, critData, expData] = MonteCarloGPU($\mathbf{S}_{t_0}, K, \mu_\delta, \sigma_\delta, \rho_{i,j}, T, N, M$);
for *Time* $t = M \dots 1$ **do**
  // Bundling. Step III.
  SortingGPU(critData[t-1]);
  begin CUDAThread per bundle $\beta = 1 \dots \nu$
    // Exercise policy (Regression). Step IV.
    α_t^β = LeastSquaresRegression(payoffData[t]);
    // Continuation value. Step V.
    CV = ContinuationValue(α_t^β, expData[t-1]);
    // Direct estimator. Step V.
    DE = DirectEstimator(CV, payoffData[t-1]);
return *DE*;
// Generation of the grid points (Monte Carlo). Step I.
// Option value at terminal time $t = M$. Step II.
[payoffData, critData, expData] = MonteCarloGPU($\mathbf{S}_{t_0}, K, \mu_\delta, \sigma_\delta, \rho_{i,j}, T, N, M$);
for *Time* $t = M \dots 1$ **do**
  // Bundling. Step III.
  SortingGPU(critData[t-1]);
  begin CUDAThread per path $n = 1 \dots N$
    // Continuation value. Step V.
    CV[n] = ContinuationValue(α_t^β, expData[t-1]);
    // Path estimator. Step V.
    PE[n] = PathEstimator(CV[n], payoffData[t-1]);
return *PE*;

5.6. RESULTS

Experiments were performed on the Accelerator Island system of the Cartesius Supercomputer (more information in [20]) with the following main characteristics:

- Intel Xeon E5-2450 v2.
- NVIDIA Tesla K40m.
- C-compiler: GCC 4.4.7.
- CUDA version: 5.5.

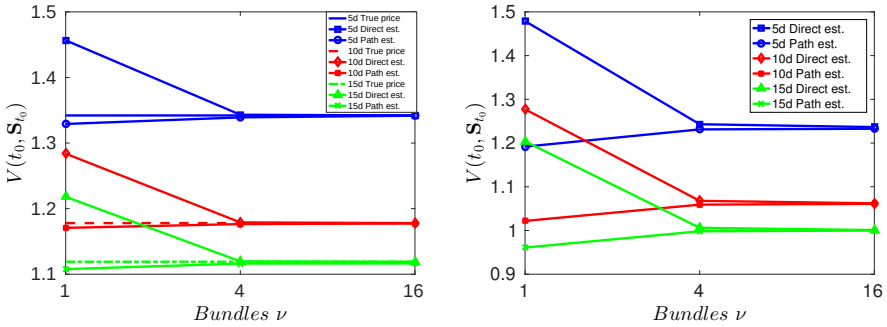
All computations are done in double precision, because a high accuracy is required both in the bundling as well as in the regression computations. We consider the d-dimensional problem of pricing Bermudan basket options with the following characteristics:

- Initial state: $\mathbf{S}_{t_0} = (40, 40, \ldots, 40) \in \mathbb{R}^d$.
- Strike: $K = 40$.
- Risk-free interest rate: $r_t = 0.06$.
- Dividend yield rate: $q_\delta = 0.0$, $\delta = 1, 2, \ldots, d$.
- Volatility: $\sigma_\delta = 0.2$, $\delta = 1, 2, \ldots, d$.
- Correlation: $\rho_{i,j} = 0.25$, $j = 2, \ldots, d$, $i = 1, \ldots, j$.
- Maturity: $T = 1.0$.
- Exercise times: $M = 10$.

Specifically, geometric and arithmetic Bermudan basket put options are chosen in order to show the accuracy and performance. The number of basis functions is taken as $b = 3$. As the stochastic asset model, we first choose the multi-dimensional GBM, and for the discretization scheme we employ the Euler-Maruyama SDE discretization.

5.6.1. EQUAL-PARTITIONING: CONVERGENCE TEST

In the original SGBM paper, the authors have shown the convergence of SGBM using k-means bundling, in dependence of the number of bundles. For the equal-partitioning bundling, we perform a similar convergence study by pricing the previously mentioned options. Regarding the sorting criterion needed for the equal-partitioning technique, we choose the payoff criterion, i.e. we sort the Monte Carlo scenarios following the geometric or arithmetic average of the assets for geometric and arithmetic basket options, respectively. Similar results can be obtained using other criteria like the product or the sum of the assets. In Figure 5.3, we show the convergence of the calculated option prices for both geometric and arithmetic Bermudan basket options with different dimensionalities, i.e. $d = 5$, $d = 10$ and $d = 15$. In the case of the geometric basket option, we can also specify the reference price, because a multi-dimensional geometric basket option can be reformulated as a one-dimensional option pricing problem. In the original paper for SGBM [76], we can also find a comparison with the LSM method [94].



(a) Geometric basket put option.

(b) Arithmetic basket put option.

Figure 5.3: Convergence with equal-partitioning bundling technique. Test configuration: $N = 2^{18}$ and $\Delta t = T/M$.

	Geometric Bermudan basket option					
	k-means			equal-partitioning		
	MC	DE	PE	MC	DE	PE
C	82.42	234.37	203.77	101.77	41.48	59.16
CUDA	1.04	18.69	12.14	0.63	4.66	1.29
Speedup	79.25	12.88	16.78	161.54	8.90	45.86
	Arithmetic Bermudan basket option					
	k-means			equal-partitioning		
	MC	DE	PE	MC	DE	PE
C	78.86	226.23	203.49	79.22	39.64	58.65
CUDA	1.36	17.89	11.74	0.83	4.14	1.20
Speedup	57.98	12.64	17.33	95.44	9.57	48.87

Table 5.1: SGBM stages time (s) for the C and CUDA versions. Test configuration: $N = 2^{22}$, $\Delta t = T/M$, $d = 5$ and $\nu = 2^{10}$.

5.6.2. BUNDLING TECHNIQUES: K-MEANS VS. EQUAL-PARTITIONING

With the convergence of the equal-partitioning technique shown numerically, we now increase drastically the number of bundles and, hence, the number of Monte Carlo paths. For the two presented bundling techniques, we perform a time comparison between the C and CUDA implementations for multi-dimensional geometric and arithmetic Bermudan basket options. First of all, in Table 5.1, we show the execution times for the different SGBM stages, i.e. Monte Carlo path generation (MC), direct estimator computation (DE) and path estimator computation (PE), for the C and CUDA versions. We can see an overall improvement of the timing results for SGBM based on equal-partitioning bundling due to more efficient direct and path estimators, which are around four times faster. The performance of the CUDA versions is remarkable, reaching a speedup of 160 for the Monte Carlo simulation.

	Geometric Bermudan basket option					
	k-means			equal-partitioning		
	$d=5$	$d=10$	$d=15$	$d=5$	$d=10$	$d=15$
C	604.13	1155.63	1718.36	303.26	501.99	716.57
CUDA	35.26	112.70	259.03	8.29	9.28	10.14
Speedup	17.13	10.25	6.63	36.58	54.09	70.67
	Arithmetic Bermudan basket option					
	k-means			equal-partitioning		
	$d=5$	$d=10$	$d=15$	$d=5$	$d=10$	$d=15$
C	591.91	1332.68	2236.93	256.05	600.09	1143.06
CUDA	34.62	126.69	263.62	8.02	11.23	15.73
Speedup	17.10	10.52	8.48	31.93	53.44	72.67

Table 5.2: SGBM total time (s) for the C and CUDA versions. Test configuration: $N = 2^{22}$, $\Delta t = T/M$ and $\nu = 2^{10}$.

In Table 5.2, the total execution times for the $d = 5$, $d = 10$ and $d = 15$ problems are presented. We observe a significant acceleration of the CUDA versions for both bundling techniques, with a special improvement in the case of the equal-partitioning. This is because the iterative process of k-means bundling penalizes parallelism and memory transfers, especially when the dimensionality increases, while equal-partitioning handles these issues in a more efficient way.

5.6.3. HIGH-DIMENSIONAL PROBLEMS

The second goal is to drastically increase the problem dimensionality for Bermudan basket option pricing. In the case of the k-means bundling algorithm, this is not possible because of memory limitations. However, we save memory using the equal-partitioning technique which enables us to increase the problem dimensions. We can reduce the total number of Monte Carlo paths, since by equal-partitioning each bundle has the same number of paths. This gives us accurate regression values in all bundles. The SGBM prices for geometric and arithmetic Bermudan basket put options are shown in Table 5.3. Again, the reference price for the geometric Bermudan basket option is obtained by the COS method [50] and we present the prices given by the direct estimator (DE) and path estimator (PE).

In Table 5.4, the execution times for pricing geometric and arithmetic Bermudan basket put options in different dimensions and with different numbers of bundles, ν, are shown. Note that the number of bundles hardly influences the execution times. With the equal-partitioning technique, the performance is mainly dependent on the number of paths and the dimensionality. For that reason, we can exploit the GPU parallelism arriving at a speedup of around 75 for the 50-dimensional problem in the case of geometric Bermudan basket option and around 100 for the 50-dimensional arithmetic Bermudan basket option.

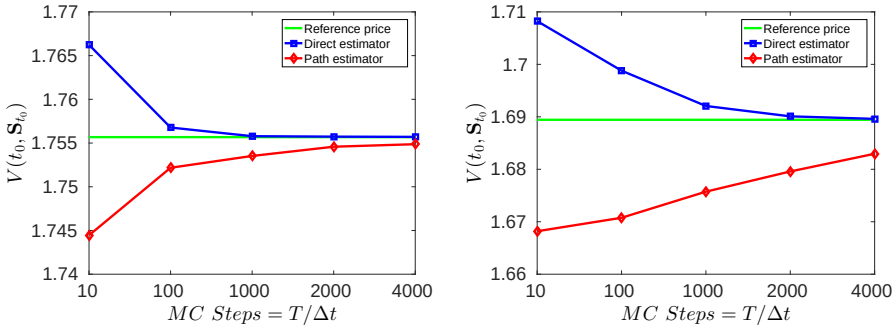
Taking into account the accuracy and the performance of our CUDA implementation, we can even increase the dimension of the problem to higher values.

	Geometric Bermudan basket option		
	$d = 30$	$d = 40$	$d = 50$
COS	1.057655	1.041889	1.032343
SGBM DE	1.057657	1.041888	1.032339
SGBM PE	1.057341	1.041545	1.031797
	Arithmetic Bermudan basket option		
	$d = 30$	$d = 40$	$d = 50$
SGBM DE	0.937436	0.921009	0.911023
SGBM PE	0.934359	0.919695	0.909646

Table 5.3: Option price for a high-dimensional problem with equal-partitioning. Test configuration: $N = 2^{20}$, $\Delta t = T/M$ and $\nu = 2^{10}$.

	Geometric Bermudan basket option					
	$\nu = 2^{10}$			$\nu = 2^{14}$		
	$d = 30$	$d = 40$	$d = 50$	$d = 30$	$d = 40$	$d = 50$
C	337.61	476.16	620.11	337.06	475.12	618.98
CUDA	4.65	6.18	8.08	4.71	6.26	8.16
Speedup	72.60	77.05	76.75	71.56	75.90	75.85
	Arithmetic Bermudan basket option					
	$\nu = 2^{10}$			$\nu = 2^{14}$		
	$d = 30$	$d = 40$	$d = 50$	$d = 30$	$d = 40$	$d = 50$
C	993.96	1723.79	2631.95	992.29	1724.60	2631.43
CUDA	11.14	17.88	26.99	11.20	17.94	27.07
Speedup	89.22	96.41	97.51	88.60	96.13	97.21

Table 5.4: SGBM total time (s) for a high-dimensional problem with equal-partitioning. Test configuration: $N = 2^{20}$ and $\Delta t = T/M$.



(a) Geometric basket put option.

(b) Arithmetic basket put option.

Figure 5.4: CEV model convergence, $\gamma = 1.0$. Test configuration: $N = 2^{16}$, $\nu = 2^{10}$ and $d = 2$.

5

5.6.4. EXPERIMENTS WITH MORE GENERAL APPROACH

Parallel SGBM can thus be used to efficiently price high-dimensional Bermudan basket options under GBM or other processes for which the ChF is available. However, as presented in Section 5.4, we can use SGBM also when the ChF is not available. In this case, we first discretize the SDE system and then determine the ChF, which changes with position and/or time. In order to get accurate approximations when using this ChF of the discrete process, we need to improve the approximation of the expectations resulting from the use of the basis functions in Equation (5.11). For that, we can increase the number of time-steps, i.e., we can reduce the differences between Monte Carlo paths within each bundle (with this, the regression becomes more accurate). We perform a convergence test to show this behaviour. For the experiment, we choose the GPU version of SGBM with equal-partitioning bundling, because this is the most efficient implementation and the performance is independent of the number of bundles. An increasing number of time-steps makes the C version too expensive again.

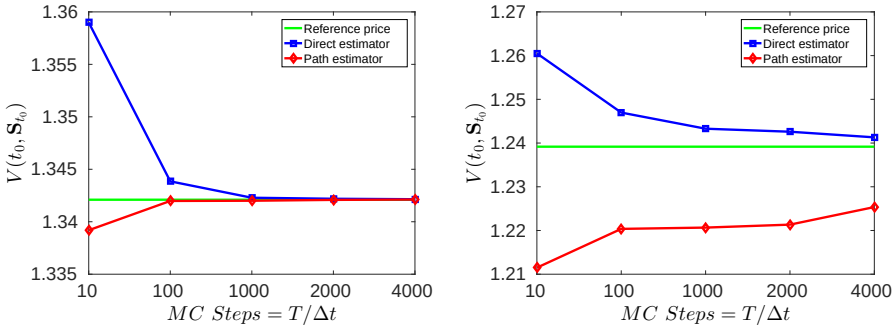
We consider a parametric local volatility model called the CEV model [31]. This model can be obtained by substituting

$$\mu_\delta(\mathbf{S}_t) = (r_t - q_\delta)S_\delta(t), \qquad \sigma_\delta(\mathbf{S}_t) = \sigma_\delta\,(S_\delta(t))^\gamma, \tag{5.12}$$

in Equation (5.1). $\gamma \in [0,1]$ is a free parameter called the variance elasticity, and q_δ and σ_δ, $\delta = 1,2,\ldots,d$, are constant dividend yield and volatility, respectively.

The computation of the derivatives in Equation (5.10) is performed by using Wolfram Mathematica 8 [135].

In Figure 5.4, a convergence test regarding the number of Monte Carlo time-steps is presented. A 2-dimensional problem is considered and $\gamma = 1.0$ is used in Equation (5.12) to compare our approximation with the reference price given by the COS method (in the case of a geometric Bermudan basket put option) and the original SGBM following GBM (for an arithmetic Bermudan basket put option). The figure shows that we can get improved approximations of the option price by increasing the number of Monte Carlo time-steps.



(a) Geometric basket put option.

(b) Arithmetic basket put option.

Figure 5.5: CEV model convergence, $\gamma = 1.0$. Test configuration: $N = 2^{16}$, $\nu = 2^{10}$ and $d = 5$.

	Geometric Bermudan basket option			
	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1.0$
SGBM DE	0.001420	0.055636	0.411066	1.755705
SGBM PE	0.001395	0.055620	0.410758	1.754869
	Arithmetic Bermudan basket option			
	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1.0$
SGBM DE	0.001417	0.055340	0.404410	1.688609
SGBM PE	0.001346	0.055249	0.400400	1.682956

Table 5.5: CEV option pricing. Test configuration: $N = 2^{16}$, $\Delta t = T/4000$, $\nu = 2^{10}$ and $d = 2$.

In Figure 5.5, the same convergence test as $d = 2$ case is presented for a 5-dimensional problem. Again, we can see that the approximations are better when the number of Monte Carlo time-steps is increased. In both cases, $d = 2$ and $d = 5$, the direct estimator gives a better approximation compared to the reference price. It is also observed that the direct and path estimators for the new approach perform similarly as in the case of the original SGBM and they can provide a confidence interval for the option price.

Once we find an appropriate combination of the number of Monte Carlo paths, bundles and time-steps, we can use our parallel SGBM method to price products under local volatility dynamics. In Tables 5.5 and 5.6, the results of pricing 2-dimensional and 5-dimensional geometric and arithmetic Bermudan basket put options are presented. We take a different values for γ in the CEV model here. Note that we did not encounter any problems with the Euler-Maruyama discretization of the CEV dynamics in our test cases. The execution times (s) are around 120 and 150 for 2-dimensional and 5-dimensional problems, respectively.

5.7. CONCLUSIONS

In this chapter, we have presented an efficient implementation of the SGBM on a GPU architecture. Through the GPU parallelism, we could speed up the execution times when

	Geometric Bermudan basket option			
	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1.0$
SGBM DE	0.000291	0.029395	0.276030	1.342147
SGBM PE	0.000274	0.029322	0.275131	1.342118
	Arithmetic Bermudan basket option			
	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1.0$
SGBM DE	0.000289	0.029089	0.267943	1.241304
SGBM PE	0.000288	0.028944	0.267214	1.225359

Table 5.6: CEV option pricing. Test configuration: $N = 2^{16}$, $\Delta t = T/4000$, $\nu = 2^{10}$ and $d = 5$.

the number of bundles and the dimensionality increase drastically. In addition, we have proposed a parallel bundling technique which is more efficient in terms of memory use and more suitable on parallel systems. These two improvements enable us to extend the method's applicability and explore more general ways to compute the continuation values. A general approach for approximating expectations based on the ChF of the discrete process was presented. This approach allowed us to use SGBM for underlying asset models for which the continuous ChF is not available.

Several results under the CEV local volatility model were presented to show the performance and the accuracy of the new proposed techniques.

Compared with other GPU parallel implementations of early-exercise option pricing methods, our parallel SGBM is very competitive in terms of computational time and can solve very high-dimensional problems. Furthermore, we provided a new way to parallelize the backward stage, according to the bundles, which gave us a remarkable performance improvement.

We think that the parallel SGBM technique presented forms a fine basis to deal with Credit Valuation Adjustment of portfolios with financial derivatives in the near future.

The hybrid component in this chapter is represented by the SGBM itself (it combines Monte Carlo, dynamic programming, bundling and local regression approaches) and the efficient parallel implementation of the method on GPU-based systems. The application of parallel computing to early-exercise option pricing methods is very challenging. Here, we have taken advantage of the particular features of SGBM to develop an efficient parallel GPU version, allowing the use of the method for very high-dimensional problems and under more involved dynamics. In the latter, another interesting component is employed, i.e. the ChF of the discrete process.

CHAPTER 6

Conclusions and Outlook

6.1. CONCLUSIONS

In this thesis, several numerical methods have been presented to deal with some of the challenging problems appearing in computational finance. We have based our computational techniques on the combination of Monte Carlo methods and other advanced methodologies, resulting in robust and efficient hybrid methods. We have proposed an "almost exact" simulation technique for the SABR model, which is a standard model in the FX and interest rate markets. The data-driven COS (ddCOS) method has been introduced in the thesis, which may be a useful method for applications in risk management. We have also developed a parallel GPU version of the Stochastic Grid Bundling Method, which opens up the applicability towards very high-dimensional problems. In the following paragraphs, we briefly summarize the most important findings in the thesis.

In Chapter 2, a one time-step Monte Carlo method to simulate the SABR dynamics has been developed. It is based on an efficient computational procedure for the time-integrated variance. The technique employs a Fourier method to approximate the marginal distribution of the time-integrated variance. Then, we have used a copula to approximate the conditional distribution (time-integrated variance conditional on the initial volatility). Resulting is a fast and accurate one time-step Monte Carlo method for the SABR model, that may be used in the context of pricing European options with a contract maturity of up to two years. By several numerical tests, we have shown that our technique does not suffer from the well-known issues of the closed-form Hagan formula. The latter formula is not accurate when small strike values and high volatilities are considered. As an alternative to this formula, our method may be suitable for calibration purposes.

In Chapter 3, we have defined an accurate and robust multiple time-step Monte Carlo method to simulate the SABR model with only a few time-steps, the so-called mSABR method. The technique represents an extension of the method presented in Chapter 2. Within the mSABR method we have incorporated several non-trivial method components, like the use of a copula, the stochastic collocation interpolation technique and a correlation approximation. The copula is employed to approximate the conditional time-integrated variance process. The marginal distribution of the time-integrated variance has been derived by means of Fourier techniques and numerical integration. Due to the use of multiple time-steps, the straightforward generation of Monte Carlo paths becomes computationally unaffordable. To deal with this computational cost issue, we have applied an efficient sampling technique based on stochastic collocation interpolation. The mSABR method appears an attractive SABR simulation technique with an impressive ratio of accuracy and performance. The convergence and the sta-

bility of the method have been numerically assessed, whereby the mSABR method requires only a few time-steps to achieve high accuracy. By the performed experiments, we have shown that mSABR scheme reduces computational costs compared to other Monte Carlo-based approaches when accurate results are required. As a multiple time-step Monte Carlo method, it can be applied to pricing path-dependent options, like barrier options. Furthermore, we have considered the extreme situation of negative interest rates, for which the mSABR method (in combination with a shifted model) also performs highly satisfactory.

In Chapter 4, we have presented the data-driven extension of the well-known COS method, the so-called ddCOS method. This new technique generalizes the applicability of the COS method towards cases where only data samples of the underlying risk neutral asset dynamics are available. Therefore, a pre-defined stochastic model, with a corresponding ChF, is not longer a requirement for this Fourier-based method. The method relies on a closed-form expression in terms of Fourier cosine expansions for the density estimation problem as it appears in statistical learning. With this Fourier based empirical density approximation, the COS method machinery can be efficiently exploited, resulting in a data-driven method which can be used for option valuation and also for risk management. The expected rate of convergence of the ddCOS method has been numerically confirmed. The method is particularly suitable for the accurate computation of option sensitivities, specifically for the option Δ and Γ. Therefore, it can be employed within the Delta-Gamma approximation context for the calculation of risk measures, solely based on asset samples. This fact has enabled us to explore more involved dynamics like those given by the SABR model. We have taken advantage of the simulation schemes developed in Chapters 2 and 3 to test the ddCOS method. For monotonic data-driven densities, the use of filters within the ddCOS method has appeared to be natural and appropriate.

In Chapter 5, a parallel GPU implementation of the Stochastic Grid Bundling Method, SGBM, has been presented. By means of the GPU parallelism, we have been able to drastically reduce the computational effort for the valuation of early-exercise options. The particular method components of the SGBM enables us to parallelize the nontrivial backward stage, where parallelization has taken place according to the bundles that are defined at each time step within SGBM. This represents a novelty with respect to other GPU parallel implementations of early-exercise option pricing methods. It allowed us to explore different configurations of bundles and to increase the option pricing problem dimensionality. We have introduced a different bundling technique, which is particularly efficient in terms of memory use and workload balancing. The convergence of this technique has been numerically shown. We have developed a different approach to computing the appearing conditional expectations, based on the ChF of the discrete underlying process. This allowed us to use SGBM also for models for which the ChF is not available. We have experienced with the CEV local volatility model (which is also a part of the SABR model), with highly satisfactory results in terms of performance and accuracy.

6.2. OUTLOOK

We think that the challenges in computational finance will necessarily require hybrid and interdisciplinary computational approaches to come up with robust, accurate and efficient solutions.

In Chapters 2 and 3, we have chosen components of the COS methodology as the Fourier inversion technique to approximate the cumulative distribution function of the time-integrated variance. Many alternatives can also be found in the literature. A particular interesting alternative is the so-called *SWIFT* method [102], which is based on Shannon wavelets. The locality features of these basis functions make the technique particularly suitable for the case of options with long maturities. The use of the SWIFT method within the mSABR method should, for example, give fast and accurate approximations. By this, CPU time of the mSABR method may be further reduced.

The use of the clever interpolation based on stochastic collocation already gave us a remarkable improvement in terms of computational cost in Chapter 3. In order to further reduce execution times, the use of high-performance computing also appears to be an interesting option in this context. In Chapter 5, GPU computing has been successfully employed. The parallelization may also be directly applied to the SCMC method or to the sample generation within the mSABR method.

In the ddCOS method presented in Chapter 4, we have exploited the relation between the Fourier cosine expansions and the density estimation from the statistical learning framework. Again, the use of other basis functions, like wavelets, may form a possible extension. Specifically, Haar wavelets have particularly features like locality and compact support, guaranteeing positive densities and allowing an efficient treatment of dynamic data.

Another natural extension for the ddCOS method would be to consider density estimation in higher dimensions. While in the context of the regularization approach, this can be a challenging task, the connection between the ddCOS method and the empirical ChF may form an interesting line of research to explore.

As mentioned, the ddCOS method has been derived in the framework of statistical learning for density estimation. The density estimation problem has been intensively studied from different points of view in the recent years. Particularly interesting is the aspect of the machine learning approach and the algorithms originating from it, like *support vector machines*, *neural networks* or *deep learning*. It would be highly interesting to develop the ddCOS method further in the context of machine learning and data science, especially because so much data is generated within the financial world.

In Chapter 5, finally, we have mainly worked with very basic dynamics for the underlying asset, i.e., geometric Brownian motion (GBM). In [29], the authors have considered the Merton jump-diffusion model, deriving analytic formulas for the early-exercise computation. Due to the increased asset path generation complexity, the gain provided by a GPU parallel version should be even higher than in the GBM case. The inclusion of several underlying models may give the opportunity of applying parallel SGBM towards more complex problems, like the problem of Credit Valuation Adjustment of a financial portfolio in the context of counterparty credit risk management.

The future for hybrid solution methods, data-driven technologies and parallel computing seems bright within finance, insurance and economics.

References

[1] Lokman A. Abbas-Turki and Bernard Lapeyre. American options pricing on multi-core graphic cards. In *2009 International Conference on Business Intelligence and Financial Engineering*, pages 307–311. IEEE, 2009.

[2] Leif B. G. Andersen. Efficient simulation of the Heston stochastic volatility model. *Journal of Computational Finance*, 11(3):1–22, 2008.

[3] Theodore W. Anderson. On the Distribution of the Two-Sample Cramér-von Mises Criterion. *Annals of Mathematical Statistics*, 33(3):1148–1159, 1962.

[4] Alexandre Antonov, Michael Konikov, and Michael Spector. SABR spreads its wings. *Risk Magazine*, pages 58–63, August 2013.

[5] Alexandre Antonov, Michael Konikov, and Michael Spector. The free boundary SABR: natural extension to negative rates. *Risk Magazine*, pages 58–63, August 2015.

[6] Philippe Barbe, Christian Genest, Kilani Ghoudi, and Bruno Rémillard. On Kendall's process. *Journal of Multivariate Analysis*, 58(2):197–229, 1996.

[7] Michael Benguigui and Françoise Baude. Fast American basket option pricing on a multi-GPU cluster. In *Proceedings of the 22nd High Performance Computing Symposium*, pages 1–8, Tampa, FL, USA, Apr 2014. 8 pages.

[8] Eric Benhamou. Fast Fourier Transform for discrete Asian options. *Journal of Computational Finance*, 6(1):49–68, 2002.

[9] Daniel Berg. Copula goodness-of-fit testing: an overview and power comparison. *The European Journal of Finance*, 15(7-8):675–701, 2009.

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[11] Ljudmila A. Bordag. *Geometrical properties of differential equations: applications of Lie group analysis in financial mathematics*. World Scientific Publishing Co, 2015.

[12] Anastasia Borovykh, Andrea Pascucci, and Cornelis W. Oosterlee. Pricing Bermudan Options Under Local Lévy Models with Default. *Journal of Mathematical Analysis and Applications*, 450(2):929–953, 2017.

[13] Svetlana Boyarchenko and Sergei Levendorskii. Efficient variations of the Fourier transform in applications to option pricing. *Journal of Computational Finance*, 18(2):57–90, 2014.

[14] Mark Britten-Jones and Stephen M. Schaefer. Non-linear Value-at-Risk. *Review of Finance*, 2(2):161–187, 1999.

[15] Mark Broadie and Özgür Kaya. Exact simulation of stochastic volatility and other affine jump diffusion processes. *Operations Research*, 54(2):217–231, March 2006.

[16] Nicola Cantarutti, João Guerra, Manuel Guerra, and Maria do Rosário Grossinho. Option pricing in exponential Lévy models with transaction costs, 2016. Available at arXiv: https://arxiv.org/abs/1611.00389v3.

[17] Luca Capriotti. Fast Greeks by algorithmic differentiation. *Journal of Computational Finance*, 14(3):3–35, 2011.

[18] Luca Capriotti and Michael B. Giles. Algorithmic differentiation: adjoint Greeks made easy. *Risk Magazine*, 25(10).

[19] Peter Carr and Dilip B. Madan. Option valuation using the Fast Fourier transform. *Journal of Computational Finance*, 2:61–73, 1999.

[20] Cartesius webpage. https://www.surfsara.nl/systems/cartesius.

[21] Bin Chen, Cornelis W. Oosterlee, and Hans van der Weide. A low-bias simulation scheme for the SABR stochastic volatility model. *International Journal of Theoretical and Applied Finance*, 15(2):1250016–1 – 1250016–37, 2012.

[22] Nan Chen and Paul Glasserman. Malliavin Greeks without Malliavin calculus. *Stochastic Processes and their Applications*, 117(11):1689–1723, 2007. Recent Developments in Mathematical Finance: Special issue based on the CCCP Meeting, April 2006, New York, NY.

[23] Rongda Chen and Lean Yu. A novel nonlinear Value-at-Risk method for modeling risk of option portfolio with multivariate mixture of normal distributions. *Economic Modelling*, 35:796–804, 2013.

[24] Umberto Cherubini and Elisa Luciano. Value-at-Risk trade-off and capital allocation with copulas. *Economic notes*, 30(2):235–256, 2001.

[25] Umberto Cherubini and Elisa Luciano. Bivariate option pricing with copulas. *Applied Mathematical Finance*, 9(2):69–85, 2002.

[26] Umberto Cherubini, Sabrina Mulinacci, Fabio Gobbi, and Silvia Romagnoli. *Dynamic copula methods in finance*, volume 625. John Wiley & Sons, 2011.

[27] Clogs webpage. http://sourceforge.net/projects/clogs/.

[28] Rafael Company, Vera Egorova, Lucas Jódar, and Carlos Vázquez. Computing American option price under regime switching with rationality parameter. *Computers and Mathematics with Applications*, 72(3):741 – 754, 2016.

[29] Fei Cong and Cornelis W. Oosterlee. Pricing Bermudan options under Merton jump-diffusion asset dynamics. *International Journal of Computer Mathematics*, 92(12):2406–2432, 2015.

[30] Shane Cook. *CUDA programming: a developer's guide to parallel computing with GPUs*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2013.

[31] John C. Cox. Note on option pricing I: Constant Elasticity of Variance diffusions. *Journal of Portfolio Management*, 22:15–17, 1996.

[32] Harald Cramér. On the composition of elementary errors. *Scandinavian Actuarial Journal*, 1928(1):13–74, 1928.

[33] CUB webpage. http://nvlabs.github.io/cub/.

[34] CUDA programming guide. http://docs.nvidia.com/cuda/cuda-c-programming-guide/.

[35] CUDA webpage. http://www.nvidia.com/object/cuda_home_new.html.

[36] CUDPP webpage. http://cudpp.github.io/.

[37] cuRAND webpage. https://developer.nvidia.com/curand.

[38] Verche Cvetanoska and Toni Stojanovski. Using high performance computing and Monte Carlo simulation for pricing American options. *CoRR*, abs/1205.0106, 2012.

[39] Duy-Minh Dang, Christina C. Christara, and Kenneth R. Jackson. An efficient graphics processing unit-based parallel algorithm for pricing multi-asset American options. *Concurrency and Computation: Practice and Experience*, 24(8):849–866, 2012.

[40] Duy-Minh Dang, Kenneth R. Jackson, and Mohammadreza Mohammadi. Dimension and variance reduction for Monte Carlo methods for high-dimensional models in finance. *Applied Mathematical Finance*, 22(6):522–552, 2015.

[41] Mark H.A. Davis and Martin P. Johansson. Malliavin Monte Carlo Greeks for jump diffusions. *Stochastic Processes and their Applications*, 116(1):101–129, 2006.

[42] Paul Deheuvels. La fonction de dépendance empirique et ses propriétés, un test non paramétrique d'indépendance. *Bulletin de l'Académie Royale de Belgique, Classe des Sciences (5)*, 65:274–292, 1979.

[43] Jacques du Toit and Uwe Naumann. Adjoint Algorithmic Differentiation Tool Support for Typical Numerical Patterns in Computational Finance. *To appear in Journal of Computational Finance*, 2017. Available at NAG Technical Reports: https://www.nag.co.uk/doc/techrep/pdf/tr3_14.pdf.

[44] Daniel J. Duffy. *Finite difference methods in financial engineering - a partial differential equation approach*. John Wiley & Sons Ltd, 2006.

[45] Valdo Durrleman, Ashkan Nikeghbali, and Thierry Roncalli. Which copula is the right one?, 2000. Available at SSRN: http://ssrn.com/abstract=1032545.

[46] Vera N. Egorova, Shi-Hau Tan, Choi-Hong Lai, Rafael Company, and Lucas Jódar. Moving boundary transformation for American call options with transaction cost: finite difference methods and computing. *International Journal of Computer Mathematics*, 94(2):345–362, 2017.

[47] Paul Embrechts, Alexander McNeil, and Daniel Straumann. Correlation: pitfalls and alternatives. *Risk Magazine*, 12:69–71, 1999.

[48] Bert Van Es, Peter Spreij, and Harry Van Zanten. Nonparametric volatility density estimation. *Bernoulli*, 9(3):451–465, 2003.

[49] Fang Fang and Cornelis W. Oosterlee. A novel pricing method for European options based on Fourier-cosine series expansions. *SIAM Journal on Scientific Computing*, 31:826–848, 2008.

[50] Fang Fang and Cornelis W. Oosterlee. Pricing early-exercise and discrete barrier options by Fourier-cosine series expansions. *Numerische Mathematik*, 114(1):27–62, 2009.

[51] Rob Farber. *CUDA Application, design and development*. Elsevier, 2011.

[52] Massimiliano Fatica and Everett Phillips. Pricing American options with least squares Monte Carlo on GPUs. In *Proceedings of the 6th Workshop on High Performance Computational Finance*, WHPCF '13, pages 5:1–5:6, New York, NY, USA, 2013. ACM.

[53] Eric Fournié, Jean-Michel Lasry, Jérôme Lebuchoux, Pierre-Louis Lions, and Nizar Touzi. Applications of Malliavin calculus to Monte Carlo methods in finance. *Finance and Stochastics*, 3(4):391–412, 1999.

[54] Beatrice Gaviraghi, Andreas Schindele, Mario Annunziato, and Alfio Borzì. On optimal sparse-control problems governed by jump-diffusion processes. *Applied Mathematics*, 7:1978–2004, 2016.

[55] Christian Genest and Louis-Paul Rivest. Statistical inference procedures for bivariate Archimedean copulas. *Journal of the American Statistical Association*, 88(423):1034–1043, 1993.

[56] Michael B. Giles. Smoking adjoints: fast Monte Carlo Greeks. *Risk Magazine*, 19(1):88–92.

[57] Michael B. Giles. Multi-level Monte Carlo path simulation. *Operations Research*, 56(3):607–617, 2008.

[58] Michael B. Giles. Vibrato Monte Carlo Sensitivities. In Pierre L' Ecuyer and Art B. Owen, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2008*, pages 369–382. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[59] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Applications of Mathematics : stochastic modelling and applied probability. Springer, 2004.

[60] Paul Glasserman and Zongjian Liu. Sensitivity estimates from characteristic functions. *Operations Research*, 58(6):1611–1623, 2010.

[61] Paul Glasserman and Zongjian Liu. Estimating greeks in simulating Lévy-driven models. *Journal of Computational Finance*, 14(2):3–56, 2011.

[62] Emmanuel Gobet, José G. López-Salas, Plamen Turkedjiev, and Carlos Vázquez. Stratified regression Monte-Carlo scheme for semilinear PDEs and BSDEs with large scale parallelization on GPUs. *SIAM Journal on Scientific Computing*, 38(6):C652–C677, 2016.

[63] Lech A. Grzelak and Cornelis W. Oosterlee. On the Heston model with stochastic interest rates. *SIAM Journal on Financial Mathematics*, 2(1):255–286, 2011.

[64] Lech A. Grzelak and Cornelis W. Oosterlee. An equity-interest rate hybrid model with stochastic volatility and the interest rate smile. *Journal of Computational Finance*, 15(4):45–77, 2012.

[65] Lech A. Grzelak, Jeroen A. S. Witteveen, M. Suárez-Taboada, and Cornelis W. Oosterlee. The Stochastic Collocation Monte Carlo sampler: highly efficient sampling from "expensive" distributions, 2015. Available at SSRN: https://ssrn.com/abstract=2529691.

[66] Tinne Haentjens and Karel J. In 't Hout. Alternating direction implicit finite difference schemes for the Heston–Hull–White partial differential equation. *Journal of Computational Finance*, 16:83–110, 2012.

[67] Patrick S. Hagan, Deep Kumar, Andrew S. Lesniewski, and Diana E. Woodward. Managing smile risk. *Wilmott Magazine*, pages 84–108, 2002.

[68] Patrick S. Hagan, Deep Kumar, Andrew S. Lesniewski, and Diana E. Woodward. Arbitrage-Free SABR. *Wilmott Magazine*, (69):60–75, 2014.

[69] Patrick S. Hagan, Andrew S. Lesniewski, and Diana E. Woodward. Probability distribution in the SABR model of stochastic volatility. In *Large Deviations and Asymptotic Methods in Finance*, pages 1–35, 2005.

[70] Martin B. Haugh and Leonid Kogan. Pricing American options: a duality approach. *Operations Research*, 52(2):258–270, 2004.

[71] Christian Hendricks, Christof Heuer, Matthias Ehrhardt, and Michael Günther. High-order ADI finite difference schemes for parabolic equations in the combination technique with application in finance. *Journal of Computational and Applied Mathematics*, 316:175 – 194, 2017.

[72] Pierre Henry-Labordère. A general asymptotic implied volatility for stochastic volatility models, 2005. Available at arXiv: http://arxiv.org/pdf/cond-mat/0504317v2.pdf.

[73] Karel in 't Hout and Radoslav Valkov. Numerical study of splitting methods for American option valuation, 2016. Available at arXiv: https://arxiv.org/abs/1610.09622.

[74] Kazuyuki Ishiyama. Methods for evaluating density functions of exponential functionals represented as integrals of geometric Brownian motion. *Methodology and Computing in Applied Probability*, 7(3):271–283, 2005.

[75] Othmane Islah. Solving SABR in exact form and unifying it with LIBOR market model, 2009. Available at SSRN: http://ssrn.com/abstract=1489428.

[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] Bilal Jan, Bartolomeo Montrucchio, Carlo Ragusa, Fiaz Gul Khan, and Omar Khan. Fast parallel sorting algorithms on GPUs. *International Journal of Distributed and Parallel systems*, 14(1), 2012.

[78] Joanne E. Kennedy and Duy Pham. On the Approximation of the SABR with mean reversion model: a Probabilistic Approach. *Applied Mathematical Finance*, 21(5):451–481, 2014.

[79] Kepler architecture webpage. http://www.nvidia.com/object/nvidia-kepler.html.

[80] Soňa Kilianova and Daniel Ševčovič. A transformation method for solving the Hamilton-Jacobi-Bellman equation for a constrained dynamic stochastic optimal allocation problem. *The ANZIAM Journal*, 55(01):14–38, 2013.

[81] Volodymyr Kindratenko, editor. *Numerical computations with GPUs*. Springer, 2014.

[82] David B. Kirk and Wen-mei W. Hwu. *Programming massively parallel processors: a hands-on approach*. Elsevier, 2010.

[83] Peter E. Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*. Springer–Verlag, 1992.

[84] Donald E. Knuth. *The art of computer programming, volume 3: (2nd Ed.) Sorting and searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.

[85] Hisashi Kobayashi, Brian L. Mark, and William Turin. *Probability, random processes, and statistical analysis*. Cambridge, 2012.

[86] Ralf Korn and Songyin Tang. Exact analytical solution for the normal SABR model. *Wilmott Magazine*, 2013(66):64–69, 2013.

[87] R. A. Kronmal and M. E. Tarter. The estimation of probability densities and cumulatives by Fourier series methods. *Journal of the American Statistical Association*, 63(323):925–952, 1968.

[88] Álvaro Leitao, Lech A. Grzelak, and Cornelis W. Oosterlee. On a one time-step Monte Carlo simulation approach of the SABR model: application to European options. *Applied Mathematics and Computation*, 293:461–479, 2017.

[89] Álvaro Leitao, Lech A. Grzelak, and Cornelis W. Oosterlee. On an efficient multiple time step Monte Carlo simulation of the SABR model. *Quantitative Finance*, 2017.

[90] Álvaro Leitao and Cornelis W. Oosterlee. GPU Acceleration of the Stochastic Grid Bundling Method for Early-Exercise options. *International Journal of Computer Mathematics*, 92(12):2433–2454, 2015.

[91] Álvaro Leitao, Cornelis W. Oosterlee, Luis Ortiz-Gracia, and Sander M. Bohte. On the data-driven COS method. *Submitted for publication*, 2017.

[92] Alan L. Lewis. A Simple Option Formula for General Jump-Diffusion and Other Exponential Levy Processes, 2001. Available at SSRN: https://ssrn.com/abstract=282110.

[93] David X. Li. On default correlation: a copula function approach. *Journal of Fixed Income*, 9:43–54, 2000.

[94] Francis A. Longstaff and Eduardo S. Schwartz. Valuing American options by simulation: a simple least-squares approach. *Review of Financial Studies*, 14(1):113–47, 2001.

[95] Roger Lord and Adam Farebrother. Fifty shades of SABR simulation, 2014. Presentation at 10th Fixed Income Conference, WBS Training, Barcelona, Spain. Available at http://www.rogerlord.com/fiftyshadessabrwbs.pdf.

[96] K. Ma and Peter A. Forsyth. Numerical solution of the Hamilton-Jacobi-Bellman formulation for continuous time mean variance asset allocation under stochastic volatility. *Journal of Computational Finance*, 20(1):1–37, 2016.

[97] Paul Malliavin and Maria Elvira Mancino. A Fourier transform method for nonparametric estimation of multivariate volatility. *Annals of Statistics*, 37(4):1983–2010, 2009.

[98] Marko J. Misic and Milo V. Tomasevic. Data sorting using Graphics Processing Units. *Telfor Journal*, 4(1), 2012.

[99] Modern GPU webpage. http://nvlabs.github.io/moderngpu/.

[100] Jan Obloj. Fine-tune your smile: Correction to Hagan et al, 2008. Available at arXiv: http://arxiv.org/abs/0708.0998v3.

[101] Luis Ortiz-Gracia and Cornelis W. Oosterlee. Efficient VaR and Expected Shortfall computations for nonlinear portfolios within the delta-gamma approach. *Applied Mathematics and Computation*, 244:16–31, 2014.

[102] Luis Ortiz-Gracia and Cornelis W. Oosterlee. A highly efficient Shannon wavelet inverse Fourier technique for pricing European options. *SIAM Journal on Scientific Computing*, 38(1):118–143, 2016.

[103] Gilles Pagès and Benedikt Wilbertz. GPGPUs in computational finance: massive parallel computing for American style options. *Concurrency and Computation: Practice and Experience*, 24(8):837–848, 2012.

[104] Louis Paulot. Asymptotic implied volatility at the second order with application to the SABR model, 2009. Available at SSRN: http://ssrn.com/abstract=1413649.

[105] Vladimir Piterbarg. A multi-currency model with FX volatility skew, 2005. Available at SSRN: http://ssrn.com/abstract=685084.

[106] Vladimir Piterbarg. Smiling hybrids. *Risk Magazine*, 19(5):66–70, 2006.

[107] Nicolas Privault and Jiadong Yu. Stratified approximations for the pricing of options on average. *Journal of Computational Finance*, 19(4):95–113, 2016.

[108] L. C. Rogers. Monte Carlo valuation of American options. *Mathematical Finance*, 12(3):271–286, 2002.

[109] Joshua V. Rosenberg. Nonparametric pricing of multivariate contingent claims. *Journal of Derivatives*, 10(3):9–26, 2003.

[110] Joshua V. Rosenberg and Til Schuermann. A general approach to integrated risk management with skewed, fat-tailed risks. *Journal of Financial Economics*, 79(3):569–614, 2006.

[111] Maria J. Ruijter and Cornelis W. Oosterlee. Numerical Fourier Method and Second-order Taylor Scheme for Backward SDEs in Finance. *Applied Numerical Mathematics*, 103(C):1–26, May 2016.

[112] Maria J. Ruijter, Mark Versteegh, and Cornelis W. Oosterlee. On the application of spectral filters in a Fourier option pricing technique. *Journal of Computational Finance*, 19(1):75–106, 2014.

[113] Jason Sanders and Edward Kandrot. *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley, 2011.

[114] Munuswamy Sankaran. Approximations to the non-central chi-square distribution. *Biometrika*, 50(1-2):199–204, 1963.

[115] Nadathur Satish, Mark Harris, and Michael Garland. Designing efficient sorting algorithms for manycore GPUs. In *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium*, 2009.

[116] Sebastian Schlenkrich, André Miemiec, and Tilman Wolff-Siemssen. Low strike extrapolation for SABR. *Wilmott Magazine*, 2014.

[117] Mark Schroder. Computing the Constant Elasticity of Variance option pricing formula. *Journal of Finance*, 44(1):211–219, 1989.

[118] Rituparna Sen and Changie Ma. Forecasting density function: application in finance. *Mathematical Finance*, 5:433–447, 2015.

[119] Rüdiger U. Seydel. *Tools for computational finance*. Springer–Verlag, 2009.

[120] Bernard W. Silverman. *Density estimation for statistics and data analysis*. Chapman & Hall, London, 1986.

[121] Johannes V. Siven, Jeffrey T. Lins, and Anna Szymkowiak-Have. Value-at-Risk computation by Fourier inversion with explicit error bounds. *Finance Research Letters*, 6(2):95–105, 2009.

[122] Abe Sklar. Fonctions de répartition à *n* dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8:229–231, 1959.

[123] Nikolai V. Smirnov. *Theory of Probability and Mathematical Statistics (selected works)*. Nauka, Moscow, 1970.

[124] Robert D. Smith. An almost exact simulation method for the Heston model. *Journal of Computational Finance*, 11(1):115–125, 2007.

[125] Maria Suárez-Taboada and Carlos Vázquez. Numerical solution of a PDE model for a ratchet-cap pricing with BGM interest rate dynamics. *Applied Mathematics and Computation*, 218(9):5217–5230, 2012.

[126] Krishnahari Thouti and S.R. Sathe. An OpenCL method of parallel sorting algorithms for GPU architecture. *International Journal of Experimental Algorithms*, 3(1), 2012.

[127] Thrust webpage. http://thrust.github.io/.

[128] John N. Tsitsiklis and Benjamin Van Roy. Regression methods for pricing complex American-style options. *IEEE transactions on Neural networks*, 12(4):694–703, July 2001.

[129] Nikolai G. Ushakov. *Selected topics in characteristic functions*. Modern Probability and Statistics. Mouton De Gruyter, 1999.

[130] Alexander van Haastrecht and Antoon A.J. Pelsser. Efficient, almost exact simulation of the Heston stochastic volatility model. *International Journal of Theoretical and Applied Finance*, 13(01):1–43, 2010.

[131] Vladimir N. Vapnik. *Statistical learning theory*. Wiley-Interscience, 1998.

[132] VexCL webpage. http://ddemidov.github.io/vexcl/.

[133] Paul Wilmott, Sam Howison, and Jeff Dewynne. *The mathematics of financial derivatives : a student introduction*. Cambridge University Press, Cambridge, UK, New York, 1995.

[134] Nicholas Wilt. *The CUDA handbook: a comprehensive guide to GPU programming*. Addison-Wesley, 2013.

[135] Wolfram Mathematica webpage. http://www.wolfram.com/mathematica/.

[136] Qi Wu. Series expansion of the SABR joint density. *Mathematical Finance*, 22(2):310–345, 2012.

[137] Bowen Zhang and Cornelis W. Oosterlee. Efficient pricing of European-style Asian options under exponential Lévy processes based on Fourier cosine expansions. *SIAM Journal on Financial Mathematics*, 4(1):399–426, 2013.

Curriculum Vitæ

Álvaro LEITAO RODRÍGUEZ

08-01-1985 Born in Xove, Spain.

EDUCATION

1997–2001 Secondary school
I.E.S. Illa de Sarón, Xove, Spain

2001–2003 Baccalaureate
I.E.S. María Sarmiento, Viveiro, Spain

2004–2010 Bachelor Degree in Technical Engineering of Computer Science
University of A Coruña, A Coruña, Spain

2009–2011 Master Degree in Mathematical Engineering
University of Vigo, Vigo, Spain

2011–2013 Researcher
M2NICA research group, Department of Mathematics
University of A Coruña, A Coruña, Spain

2013–2017 PhD Researcher
Scientific Computing group
Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

2017 PhD. Applied Mathematics
Delft University of Technology, Delft, The Netherlands
Thesis: Hybrid Monte Carlo methods in computational finance
Promotor: Prof. dr. ir. Cornelis W. Oosterlee

List of Publications

Journal articles:

4. **A. Leitao**, **C. W. Oosterlee**, **L. Ortiz-Gracia** and **S. M. Bohte**, On the data-driven COS method, Submitted for publication, 2017.

3. **A. Leitao**, **L. A. Grzelak** and **C. W. Oosterlee**, On an efficient multiple time-step Monte Carlo simulation of the SABR model, Quantitative Finance, 2017.

2. **A. Leitao**, **L. A. Grzelak** and **C. W. Oosterlee**, On a one time-step Monte Carlo simulation approach of the SABR model: Application to European options, Applied Mathematics and Computation, 293:461–479, 2017.

1. **A. Leitao** and **C. W. Oosterlee**, GPU Acceleration of the Stochastic Grid Bundling Method for Early-Exercise options, International Journal of Computer Mathematics, 92(12):2433–2454, 2015.

Chapters in books:

2. **A. Leitao**, **L. A. Grzelak** and **C. W. Oosterlee**, A highly efficient numerical method for the SABR model, STRIKE - Novel Methods in Computational Finance. To be published in 2017.

1. **A. Leitao** and **C. W. Oosterlee**, Modern Monte Carlo methods and GPU computing, STRIKE - Novel Methods in Computational Finance. To be published in 2017.

List of Attended Conferences

Presentations:

8. MathFinance conference, Frankfurt, Germany, April 2017.
7. Summer school on Quantitative methods for Risk management in Finance and Insurance, A Coruña, Spain, July 2016.
6. The 19th European Conference on Mathematics for Industry, Santiago de Compostela, Spain, June 2016.
5. SIAM conference on Uncertainty Quantification, Lausanne, Switzerland, April 2016.
4. International Conference on Computational Finance, Greenwich-London, United Kingdom, December 2015.
3. Stochastics & Computational Finance: from academia to industry, Lisbon, Portugal, July 2015.
2. Python for finance: an Introduction, Lisbon, Portugal, December 2014.
1. The 18th European Conference on Mathematics for Industry, Taormina, Italy, June 2014.

Conference papers:

2. **A. Leitao**, **L. A. Grzelak** and **C. W. Oosterlee**, Efficient multiple time-step simulation of the SABR Model, In Proceedings of the 19th European Conference on Mathematics for Industry, 2016, Springer Heidelberg.
1. **A. Leitao** and **C. W. Oosterlee**, On a GPU acceleration of the Stochastic Grid Bundling Method, In Proceedings of the 18th European Conference on Mathematics for Industry, 2014, Springer Heidelberg.

Acknowledgement

First and foremost, my most sincere gratitude goes to my supervisor and promotor prof. Kees Oosterlee. His guidance and advice have been essential for the successful completion of this thesis. Kees never hesitates to share his knowledge and experience with his students, keeping the motivation and the learning interest in the research process. Furthermore, Kees directly offered me this opportunity, for which, I will always be grateful.

The last four years have been one of the nicest experience in my life which I had the pleasure to share with lots of excellent people. The (former) students and collaborators of Kees form a very friendly and wonderful group, not only cooperating and helping each other but also having fun together. Many thanks to Andrea, Anton, Fei, Gemma, Ki Wai, Marjon, Peiyao, Prashant, Qian, Sander, Shashi, Shuaiqiang and Zaza (dank u wel voor de Nederlandse vertaling!), and the Spanish community, Luis, María, Marta and Paco. The colleagues from the CWI, with whom, I have enjoyed the informal discussions about several, sometimes crazy, topics during lunches and next to the coffee machine: Anne, Barry, Bart, Benjamin, Bram, Chris, Daan, Daniel, Debarati, Folkert, Fredrik, Inti, Jan, Jan Willem, Jeroen (×2), Joost, Keith, Krzysztof, Laurent, Nick, Sangeetika, Sirshendu, Svetlana, Wander, Willem Jan, Xiaodong, Yous and Zhichao. With many of them, I have also enjoyed practising basketball, football, squash or table-tennis. The help of the CWI staff, particularly Nada and Duda, is very appreciated.

My PhD was partially carried out in the framework of a European Multi-ITN project called STRIKE. I would like to specially thank the coordinators, Matthias Ehrhardt and Jan ter Maten for their impressive work. I have spent a very nice time with my colleagues in the project, the "STRIKErs" Zuzana, Zé, Pedro, Vera, Walter, Nicola, Ivan, Lara, Giang, Shih-Hau, Beatrice, Radoslav, Christof, Silvie and Fazlollah, during the countless project events. I also had a great time with Anastasia, Enrico, José Germán, Maarten, Matthieu, Sidy and Zuzana in different conferences around Europe. It was really a pleasure to have met all of them. I want to particularly express my gratitude to prof. Carlos Vázquez Cendón who encouraged me to take this important step in my career.

I would also like to thank the members of my defence committee for reading my dissertation and participating in my defence. Special thanks to Lech Grzelak for his constant support and helpful comments during my PhD research.

Finally, I would like to express the most special gratitude to my family.

Gracias en primeiro lugar a meus pais pola educación que me deron, da que me sinto moi orgulloso. Sempre me animaron e me apoiaron nos momentos importantes, dándome os mellores consellos. Moitas gracias a miña nai por todo o seu cariño. E moitas gracias a meu pai, a quen, despois de nove anos sen él, recordo cada día.

E por suposto, moitísimas gracias a Mela. Ela foi o meu gran apoio nos últimos catro anos (e nos once anteriores), nos que soubemos desfrutar e aprender dunha situación excepcional. A súa axuda, apoio e comprensión foron imprescindibles para acadar este logro. Xuntos afrontaremos o que o futuro nos depare.

Álvaro Leitao Rodríguez,
Amsterdam, June 2017