Efficient Pricing of Early–Exercise and Exotic Options Based on Fourier Cosine Expansions

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 woensdag 4 juli 2012 om 10.00 uur

door

BOWEN ZHANG
Master of Science in Applied Mathematics,

geboren te Beijing, China

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
Prof.dr. K. In't Hout,	Universiteit Antwerpen, Belgium
Prof.dr. R. Seydel,	Universität zu Köln, Germany
Prof.dr. M. Vanmaele,	Universiteit Gent, Belgium
Prof.dr. R.J.A. Laeven,	Universiteit van Amsterdam, the Netherlands
Prof.dr.ir. A. W. Heemink,	Technische Universiteit Delft, the Netherlands
Prof.dr.ir. C. Vuik,	Technische Universiteit Delft, the Netherlands

Efficient Pricing of Early–Exercise and Exotic Options Based on Fourier Cosine Expansions.
Dissertation at Delft University of Technology.

ISBN 978–94–6203–073–2

Cover design by Bowen Zhang.
Thesis printed in the Netherlands by Wöhrmann Print Service.

Acknowledgements

This dissertation concludes my PhD research at Delft University of Technology, from September 2008 to March 2012. I would like to attribute the success of this thesis to a number of people for their support and advice.

First of all, my greatest thanks goes to my supervisor Prof. Kees Oosterlee, for trusting me in this PhD position, as well as for his countless help and patience with my research in the last four years.

Next, I would like to express my gratitude to Fang Fang, for her help and encouragement throughout my PhD period, as a friend, colleague and as the developer of the COS method.

My sincere gratitude goes to Dr. Hans van der Weide for his help for the last six years, from the beginning of my MSc period, when I took the course reliability analysis, taught by him and Prof. van Noortwijk, till the end of my PhD project, when his advices and help contributes a lot to my last paper on American-style Asian options.

I would like to express my appreciation of Prof. Piet Hemker, for fruitful discussion about my first Asian option paper.

I am indebted to Lech Grzelak for his contribution to the paper with commodity option pricing, as well as the proofreading of the revisions.

Special thanks goes to Kees Lemmens for introducing me into the wonderful world of GPU computing.

I would like to thank all my friends and colleague from TU Delft, CWI and back in China, in particular, Xinzheng Huang, Liangyue Ji, Corneliu Cofaru, Hisham bin Zubair, and Fred Vermolen, for their friendship and encouragement throughout my PhD period.

A special thanks goes to my boyfriend, Catalin, as without his love, support, understanding, encouragement, and extreme patience, none of my publications, nor this thesis would ever been finished.

I dedicate this PhD thesis to my parents for their endless love.

Summary

Efficient pricing of early–exercise and exotic options based on Fourier cosine expansions

Bowen Zhang

In the financial world, two tasks are of prime importance: model calibration and portfolio hedging. For both tasks, efficient option pricing is necessary, particularly for the calibration where many options with different strike prices and different maturities need to be priced at the same time. Therefore, a fast yet accurate pricing method is a necessity for banks and trading companies.

Nowadays three groups of pricing methods are being used in the financial industry and academia, that is, Monte–Carlo methods, partial (integro-)differential equation (PIDE) methods, and numerical integration methods, where the option price is modeled as the discounted expected value of the payoff at maturity. The latter type of methods is attractive from both practice and research point of view, as the fast computational speed, especially for plain vanilla options, makes it useful for calibration at financial institutions. Usually numerical integration techniques are combined with the Fast Fourier transform or Hilbert transform, and therefore, the numerical integration methods are often referred to as the 'transform methods'. Representatives of transform methods are the Carr–Madan method [16], the CONV method [50] and the Hilbert transform method [36].

A recent contribution to the transform method category is the COS method proposed in [34, 35], that is, an option pricing method based on the Fourier cosine expansions. It departs from a truncated risk–neutral formula, in which the conditional density function is recovered in terms of its characteristic function, by Fourier cosine expansions. This method can be used for asset processes as long as the characteristic function of the conditional density function is known, or can be approximated. For processes where the density function and its derivatives are continuous functions with respect to

the underlying asset, the COS method exhibits an exponential convergence rate.

Our research work is based on the COS method, which has been used for vanilla European option pricing [34], vanilla early–exercise option pricing and barrier option pricing [35]. The motivation of this thesis is to further improve the robustness of the COS method, make it efficient for non–Lévy models, and extend it to different types of exotic options.

The point of departure of this thesis is to improve the robustness of the COS method for call option pricing with early-exercise features, as presented in Chapter 1, where the call option prices are obtained from put option prices, in combination with the put–call parity and put–call duality relations, which are incorporated into our pricing algorithm at each early–exercise date to recover the Fourier coefficients and to compute the continuation value. The robustness of the pricing methods is demonstrated by error analysis, as well as by a series of numerical examples.

In Chapter 2, the acceleration of option pricing by the COS method on the Graphics Processing Unit (GPU) is presented. After a brief discussion of the GPU and its potential for option pricing, we will study different ways of GPU implementation, followed by three examples of GPU acceleration, the so-called multiple strike option pricing, option pricing under hybrid models where the characteristic function is derived from a Riccati ODE system, and the example of Bermudan option pricing. Influence of data transfer between host and device is also discussed in this chapter.

Extension of COS method to early–exercise option pricing with an Ornstein–Uhlenbeck (OU) process is explained in Chapter 3. OU processes for commodity derivatives, either with or without seasonality functions, are non–Lévy processes and more computationally expensive within the COS framework, as compared to Lévy processes. First of all, an accurate pricing algorithm is given, which can be used for all OU processes with different types of seasonality functions. Then, based on a detailed error analysis, a more efficient pricing method is proposed, which reduces the computing time from seconds to milliseconds. However, this new method is not advocated for all parameter settings. The conditions under which the basis point accuracy can be ensured is derived by error analysis. In the numerical part, the accuracy and efficiency of these two pricing methods are compared, and the conditions we derived from error analysis are further verified by several numerical experiments.

In Chapter 4, we present an efficient pricing method for American–style swing options, based on Fourier cosine expansions. Here we assume that the holder of the swing option has the right, but not the obligation, to buy or sell a certain amount of commodity, such as gas and electricity, at any time before the expiry of the option, and more than once. Moreover, a recovery time is added between two consecutive exercises in which exercise is not allowed. Our pricing method is based on the Bellman principle, leading to

a backward recursion procedure in which the optimal exercise regions are determined at each time step, after which the Fourier coefficients can be recovered recursively. Our method performs well for different underlying processes, different swing contracts and different types of recovery time.

The pricing methods for European and early–exercise Asian options (ASCOS) are shown respectively in Chapters 5 and 6. In Chapter 5, we present an efficient option pricing method for Asian options written on different types of averaged asset prices, but without early–exercise features. In our method, the characteristic function of the average asset is recursively recovered, with the help of Fourier expansions and Clenshaw–Curtis quadrature. Then it is used in the risk–neutral formula to get the Asian option price. Exponential convergence rate is observed for most Lévy processes, which is also supported by a detailed error analysis. Advantages of our pricing algorithm are that as the number of monitoring dates increases, the method stays robust and the computing time does not increase significantly, as shown in the numerical results.

Our pricing method for early–exercise Asian options is presented in Chapter 6. In this case, the Fourier cosine coefficients of the option price are recursively recovered by Fourier transform and Clenshaw–Curtis quadrature. Then these coefficients are inserted into the risk–neutral formula, which, in the early–exercise Asian case, is a two–dimensional integration, to get the option value. The *chain rule* from probability theory is also needed in our algorithm to factorize the joint conditional density functions. An exponential convergence rate in the option price, as derived in a detailed error analysis, is observed from various numerical experiments. Factors of approximately hundred of speedup are achieved on the GPU.

Conclusions and insight into future research are to be found in Chapter 7. In this thesis, efficient pricing methods for different early–exercise and exotic options, based on the Fourier cosine expansions, are presented, followed by an error analysis and numerical results, from which we see that the COS method is an efficient, robust and flexible method for pricing different types of option products, for different asset models, and is suitable for GPU acceleration. It is a promising tool for financial calibration and dynamic hedging in practice.

Samenvatting

Efficiënte waardering van opties met vervroegde uitoefeningsmogelijkheid en exotische opties gebaseerd op Fourier cosinusexpansies

Bowen Zhang

In de financiële wereld zijn twee taken bij het waarderen van derivaten van groot belang: model kalibratie en portefeuille hedgen. Voor beide taken zijn efficiënte optiewaarderingstechnieken nodig, maar in het bijzonder voor het kalibreren, waarin vele opties met verschillende uitoefenprijzen en looptijden tegelijkertijd geprijsd moeten worden. Een snelle en accurate waarderingsmethode is daarom noodzakelijk voor banken en handelsbedrijven.

In het algemeen worden drie typen waarderingsmethoden gebruikt in de financiële industrie en in de academische wereld: Monte–Carlo simulatietechnieken, partiële–(integro) differentiaalvergelijkingsmethoden, en numerieke integratie, waarbij de optieprijs als de verdisconteerde verwachtingswaarde van de uitbetaling op de uitoefendatum geschreven wordt.

Numerieke integratie is aantrekkelijk, zowel vanuit praktisch als ook vanuit academisch oogpunt bekeken, vanwege de uiterst snelle rekentijd, vooral voor *plain vanilla* opties. Meestal worden de numerieke integratietechnieken gecombineerd met een transformatie naar het Fourierdomein, of met de Hilberttransformatie, en daarom worden deze methoden vaak 'transformatiemethoden' genoemd. Vertegenwoordigers van deze klasse zijn de Carr–Madan methode [16], de CONV methode [50] en de Hilbert Transformatiemethode [36].

Een recente bijdrage aan de klasse van transformatiemethoden is de COS methode, voorgesteld in [34, 35]. Dit is een optiewaarderingsmethode gebaseerd op Fourier cosinusexpansies en op een versie van de risico–neutrale waarderingsformule, waarin de kansdichtheidsfunctie benaderd wordt in termen van zijn karakteristieke functie. Deze methode is bruikbaar voor aandeelprocessen waarvan de karakteristieke functie bekend of afleidbaar is.

Voor processen waarvan de kansdichtheidsfunctie en zijn afgeleiden continue functies zijn, heeft de COS methode een exponentiële convergentiesnelheid.

Het onderzoek in dit proefschrift is gebaseerd op de COS methode, die reeds gebruikt werd voor Europese optiewaardering [34], en de waardering van Bermuda opties met vervroegde uitoefeningsmogelijkheden en barrieropties [35]. In dit proefschrift wordt de robuustheid van de COS methode verder verbeterd, en de methode wordt efficiënter gemaakt voor 'niet–Lévy' modellen. Verder wordt de toepasheidheid veralgemeniseerd naar verschillende exotische opties.

Het startpunt van dit proefschrift is het verbeteren van de robuustheid van de COS methode voor de waardering van koopopties met vervroegde uitoefeningsmogelijkheid, gepresenteerd in Hoofdstuk 1. De koopoptieprijzen worden van de verkoopoptieprijzen afgeleid, in combinatie met *put–call pariteits-* en put–call dualiteitsrelaties. De robuustheid van de waarderingsmethode wordt aangetoond met een foutenanalyse, en met een aantal numerieke voorbeelden.

In Hoofdstuk 2, wordt de versnelling van optiewaardering met de COS methode op de *grafische kaart (de GPU)* gepresenteerd. Na een korte discussie over de GPU en zijn potentie voor optiewaardering, bestuderen we verschillende manieren van GPU implementatie, en geven we drie voorbeelden van GPU versnelling: de optiewaardering met meerdere uitoefenprijzen, optiewaardering onder hybride modellen waarbij de karakteristieke functie vanuit een Riccati ODE systeem bepaald wordt, en Bermuda optiewaardering. De invloed van communicatie tussen de CPU en de GPU wordt ook in deze hoofdstuk besproken.

De COS methode wordt in Hoofdstuk 3 voor opties met vervroegde uitoefeningsmogelijkheid onder een Ornstein–Uhlenbeck (OU) proces efficiënter gemaakt. OU processen worden vaak gekozen voor derivaten die op grondstoffen geschreven worden. Dit zijn geen Lévy processen en COS berekeningen zijn in dat geval duurder, vergeleken met Lévy processen. Er wordt een efficiënt en nauwkeurig algoritme gepresenteerd, dat ook bruikbaar is voor processen met verschillende seizoensafhankelijke functies, zoals we die vaak bij energiederivaten tegenkomen. De rekentijd wordt van seconden tot milliseconden gereduceerd. De voorwaarden, waaraan de parameterwaarden van het OU proces moeten voldoen om een basispuntprecisie te bereiken, worden met behulp van foutenanalyse afgeleid.

In hoofdstuk 4 wordt een efficiënte waarderingsmethode voor Amerikaanse *swing opties* gepresenteerd, gebaseerd op Fourier cosinusexpansies. Hier nemen wij aan dat de koper van een *swing optie* het recht, maar niet de verplichting heeft, tijdens de looptijd van de optie, hoeveelheden van de onderliggende, als gas of elektriciteit, extra te kopen of te verkopen tegen een van tevoren vastgestelde prijs. De *swing optie* kan meer dan één keer uitgeoefend worden. Daarnaast wordt een *hersteltijd* toegevoegd tussen opeenvolgende uitoefendata, waarin geen uitoefening van de optie toegestaan is.

Het waarderingsalgoritme is op het Bellman principe gebaseerd, dat tot een achterwaartse recursie leidt, waarin de optimale uitoefeningswaarden op elke uitoefentijdstip bepaald worden met behulp van de Newton methode. De nieuwe methode werkt goed voor verschillende onderliggende processen, verschillende swingoptiecontracten, en voor verschillende typen hersteltijden.

Nieuwe waarderingsmethoden voor Aziatische opties (de ASCOS methode genaamd), met of zonder vervroegde uitoefeningsmogelijkheden, worden in Hoofdstuk 5 en 6 gepresenteerd. In Hoofdstuk 5, presenteren we een efficiënte optiewaarderingsmethode voor Aziatische opties die op verschillende typen gemiddelde aandeelprijzen geschreven worden, zonder de vervroegde uitoefeningsmogelijkheid. In de methode wordt de karakteristieke functie van de gemiddelde aandeelprijs recursief berekend met behulp van Fourierexpansie en Clenshaw–Curtis kwadratuur. Vervolgens wordt de karakteristieke functie in de risico–neutrale waarderingsformule gesubstitueerd om de prijs van de Aziatische optie te berekenen. Voor gladde kansdichtheidsfuncties tonen wij aan via een foutenanalyse dat de convergentie exponentieel is. Voordelen van onze waarderingsmethode, zoals in de numerieke resultaten te zien is, zijn dat de methode robuust blijft en dat de rekentijd niet stijgt als het aantal tijdstippen waarop het gemiddelde gebaseerd is, toeneemt.

De ASCOS waaderingsmethode voor Aziatische opties met vervroegde uitoefeningsmogelijkheid wordt in Hoofdstuk 6 gepresenteerd. Ook in dit geval worden de Fourier coëfficiënten van de optieprijzen recursief berekend, met behulp van Fouriertransformatie en Clenshaw–Curtis kwadratuur. Vervolgens worden deze Fourier coëfficiënten in de risico–neutrale waarderingsformule, die in dit geval door een tweedimensionale integraal beschreven wordt, ingezet om de optieprijzen te verkrijgen. De *kettingregel* vanuit de kansrekening is essentieel in ons algoritme. Exponentiële convergentie van de optieprijs, bevestigd met een foutenanalyse, wordt met numerieke experimenten bevestigd voor meerdere Lévy processen. Een rekenversnelling van onze waarderingsmethode met een factor honderd wordt op de GPU bereikt.

De conclusies van de proefschrift en inzichten voor toekomstig onderzoek worden in Hoofdstuk 7 samengevat. In dit proefschrift worden dus efficiënte waarderingsmethoden voor verschillende exotische opties middels Fourier cosinusexpansies gepresenteerd, gevolgd door foutenanalyse en numerieke resultaten. We zien dat de COS methode een efficiënte, robuuste en flexibele waarderingsmethode voor verschillende typen opties, en voor verschillende aandeelmodellen is, geschikt voor implementatie op de GPU.

CONTENTS

List of Symbols

r	Interest rate.
q	Dividend rate.
σ	Volatility of the underlying process.
$Re(\cdot)$	Taking the real part of the input argument.
$Im(\cdot)$	Taking the imaginary part of the input argument.
i	$\sqrt{-1}$, the unit of the imaginary part of a complex number.
T	Maturity time of the option.
K	Strike price of an options.
N	The number of terms in the Fourier cosine expansions.
$\mathcal{M}$	For an early–exercise option $\mathcal{M}$ denotes the number of early–exercise dates. For an Asian option M denotes the number of monitoring–dates.
n_q	The number of terms in the Clenshaw–Curtis quadrature.
$f(y\|x)$	The conditional density function of y given x.
$\varphi(u;x,t)$	The conditional characteristic function, given state variable x and time interval t.
$\phi(u;t)$	The Lévy component in the characteristic function. Also used for unconditioned characteristic function.
ξ_n	The n^{th} cumulant of the underlying process.
S_t	Stock price at time t.
S_0	Stock price at initial time.
S_m	The stock price at early–exercise date t_m. Only used for early–exercise options.
x, y	State variables, usually denote the log–asset values at consecutive time steps for an early–exercise option. In chapter 1 and 2, $x = \log(S_{m-1}/K)$, $y = \log(S_m/K)$; In chapter 3 and 4, $x = \log(S_{m-1})$, $y = \log(S_m)$; In chapter 6, $x = \log(S_m/S_{m-1})$, $y = \sum_{j=0}^{'m} S_j/S_0$.
x_m^*	Early–exercise point at the m^{th} early–exercise date, where the continuation value equals the payoff.

$v(x,t)$	Option price with state value x at time t.
$c(x,t)$	Continuation value with state value x at time t.
$g(x)$	Payoff function with state value x. In chapter 4 and 6, where we study exotic options, the payoff functions has more input arguments.
V_k	The k^{th} Fourier cosine coefficient of option price at maturity for an European option.
$V_k(t_m)$	The k^{th} Fourier cosine coefficient of the option price at m^{th} early–exercise date of an early–exercise option over the whole asset domain.
$C_k(x_1, x_2, t_m)$	The k^{th} Fourier cosine coefficient of the continuation value at m^{th} early–exercise date for an early–exercise option, over the asset region $[x_1, x_2]$.
$G_k(x_1, x_2)$	The k^{th} Fourier cosine coefficient of the payoff function over the asset region $[x_1, x_2]$.

CHAPTER 1

Introduction

This chapter serves as an introduction for the thesis. It also contains essentially the contents of paper [72].

Numerical integration methods are traditionally very efficient for the valuation of single asset European options. They are also referred to as "transform methods" as a transformation, often to the Fourier domain, is combined with numerical integration [16, 34, 51]. The transform methods can readily be used with asset price models for which the characteristic function (i.e., the Fourier transform of the probability density function) is available.

Next to Fourier-based transform methods, techniques based on the Gauss or the Hilbert Transform have also been introduced [11, 12, 36, 66]. A contribution of our research group to the development of the transform methods is the COS method [34, 35], which is based on Fourier cosine expansions and converges exponentially in the number of terms in the Fourier cosine expansion.

Transform methods have also been generalized to pricing options with early-exercise features. The key idea is to set up a time lattice on each early-exercise date and view the option as "European style" between two adjacent lattices. Pricing an early-exercisable option usually involves two steps: recovery of the probability density function and computation of the integral that appears in the risk-neutral valuation formula. Some of the existing methods employ quadrature rules in both steps, see for example [50, 33, 29, 30, 44]. We will detail the generalization of the COS method to pricing Bermudan options here.

When pricing call options with the COS method, the method's accuracy may exhibit sensitivity regarding the choice of the domain size in which the series expansion is defined. A call payoff grows exponentially with the

log-stock price which may introduce significant cancellation errors for *large domain sizes*. Put options do not suffer from this, as their payoff value is bounded by the strike value. For pricing European calls, one can employ the well-known put–call parity or put–call duality and price calls via puts. Here, we generalize this concept, so that we can also apply the put-call parity or put–call duality when pricing *Bermudan call options*.

The purpose of the present chapter is two-fold. First of all, we present the COS method, focusing on options with early-exercise features, like Bermudan and American options. Secondly, we present a novel component for the *robust* pricing of call options, where we use the put–call parity and the put–call duality relations for the valuation of Bermudan call options.

The outline of this chapter is as follows: we start with a brief introduction of exponential Lévy asset dynamics, which will also be used in upcoming chapters in this thesis, given in Section 1.1. Then we will introduce the COS method for European options as well as early–exercise options in Section 1.2, and discuss the choice of computational domain. In Section 1.3 an error analysis for the COS method is included for call options. The generalization of the put–call parity and put–call duality is presented in Section 1.4. Section 1.5 then presents a variety of numerical results, confirming the robustness of the introduced version of the COS valuation method for Bermudan options.

1.1 Exponential Lévy Asset Dynamics

An asset is modeled here by an exponential Lévy process (e.g. Geometric Brownian Motion, the Variance Gamma (VG) model [15], the CGMY model [53], the Normal Inverse Gaussian model [3], ...).

The asset price can be written as an exponential function of Lévy process, L_t as follows:

$$S_t = S_0 \exp(L_t). \tag{1.1}$$

For ease of exposure we assume that the asset pays a continuous stream of dividends, measured by the dividend rate, q. In addition, we assume the existence of a bank account, B_t, which evolves according to $dB_t = rB_t dt$, with r being the (deterministic) risk-free rate. Recall that a process L_t on $(\Omega, \mathcal{J}, P)$, with $L_0 = 0$, is a Lévy process if it has independent increments, stationary increments, and it is stochastically continuous, i.e., for any $t \geq 0$ and $\epsilon > 0$ we have

$$\lim_{s \to t} \mathbb{P}(|L_t - L_s| > \epsilon) = 0. \tag{1.2}$$

A Lévy process can be characterized by a triplet (μ, σ, ν) with $\mu \in \mathbb{R}, \sigma \geq 0$ and ν a measure satisfying $\nu(0) = 0$ and

$$\int_{\mathbb{R}} \min(1, |x|^2)\nu(dx) < \infty. \tag{1.3}$$

In terms of this triplet the *characteristic function* of the Lévy process equals:

$$\begin{aligned}\phi(u;t) &= \mathbb{E}[\exp(iuL_t)] \\ &= \exp\left(t(i\mu u - \frac{1}{2}\sigma^2 u^2 + \int_{\mathbb{R}}(e^{iux} - 1 - iux\mathbf{1}_{[|x|<1]}\nu(dx))\right),\end{aligned} \tag{1.4}$$

the celebrated Lévy-Khinchine formula. As is common in most models nowadays we assume that Equation (1.1) is formulated directly under the risk-neutral measure. To ensure that the reinvested relative price, $e^{qt}S_t/B_t$, is a martingale under the risk-neutral measure, we need to ensure that

$$\phi(-i,t) = \mathbb{E}[\exp(L_t)] = e^{(r-q)t}, \tag{1.5}$$

which is satisfied if the drift μ is chosen as:

$$\mu = r - q - \frac{1}{2}\sigma^2 - \int_{\mathbb{R}}(e^x - 1 - x\mathbf{1}_{[|x|<1]})\nu(dx). \tag{1.6}$$

Based on Equation (1.1) we define:

$$\log(S_t/K) = \log(S_0/K) + L_t := x + L_t.$$

The characteristic function of $\log(S_t/K)$ is denoted by $\varphi(u,x;t)$ and reads:

$$\varphi(u;x,t) := e^{iux}\phi(u;t) = e^{iux}\mathbb{E}(exp(iuL_t)). \tag{1.7}$$

Characteristic functions for several exponential Lévy processes are available in [23, 57]. Here we give two examples of Lévy processes which we will encounter in upcoming chapters, the CGMY model and NIG model.

1.1.1 Examples of Lévy processes and characteristic functions

One problem with the Geometric Brownian Motion (GBM) model is that it is not able to reproduce the volatility skew or smile present in most financial markets. Over the past few years it has been shown that several exponential Lévy models are, at least to some extent, able to reproduce the skew or smile. One particular model is the *CGMY model* [14]. The underlying Lévy process is characterized by the triple $(\mu,\sigma,\nu_{\text{CGMY}})$, where the Lévy density is specified as:

$$\nu_{\text{CGMY}}(x) = \begin{cases} C\dfrac{\exp(-G|x|)}{|x|^{1+Y}} & \text{if} \quad x<0 \\ C\dfrac{\exp(-M|x|)}{|x|^{1+Y}} & \text{if} \quad x>0. \end{cases} \tag{1.8}$$

with parameters C, G, M and Y. Conveniently, the characteristic function of the log-asset price can be found in closed-form as:

$$\varphi(u;x_0,t) = \exp\left(iu(x_0+\mu t) - \frac{1}{2}u^2\sigma^2 t\right)\phi_{\text{CGMY}}(u;t), \tag{1.9}$$

with $x_0 = \log(S_0)$ and

$$\phi_{\text{CGMY}}(u;t) = \exp\left(tC\Gamma(-Y)\left((M-iu)^Y - M^Y + (G+iu)^Y - G^Y\right)\right),$$

where $\Gamma(x)$ is the gamma function. When $C = 0$ the model reduces to the *GBM model*.

The *Normal Inverse Gaussian* (NIG) process [3] is a variance-mean mixture of a Gaussian distribution with an inverse Gaussian. The pure jump characteristic function of the NIG model reads

$$\phi_{\text{NIG}}(u;t) = \exp\left(t\delta\left(\sqrt{\alpha^2-\beta^2} - \sqrt{\alpha^2-(\beta+iu)^2}\right)\right),$$

with $\alpha, \delta > 0$ and $\beta \in (-\alpha, \alpha - 1)$. The α-parameter controls the steepness of the density; β is a skewness parameter: $\beta > 0$ implies a density skew to the right, $\beta < 0$ a density skew to the left, and $\beta = 0$ implies the density is symmetric around 0. δ is a scale parameter in the sense that the re–scaled parameters $\alpha \to \alpha\delta$ and $\beta \to \beta\delta$ are invariant under location-scale changes of x.

1.2 The Fourier Cosine Method (COS)

The Fourier cosine pricing method, *the COS method*, is based on the risk-neutral option valuation formula (discounted expected payoff approach):

$$v(x,t_0) = e^{-r\Delta t}\int_{-\infty}^{\infty} v(y,T)f(y|x)dy, \tag{1.10}$$

where $v(x,t_0)$ is the present option value, r the interest rate, $\Delta t = T - t_0$ and x, y can be any monotone function of the underlying asset at initial time t_0 and the expiration date T. Function $v(y,T)$, which equals payoff function $g(y)$, is known, but the transitional density function, $f(y|x)$ in (1.10), typically is not.

To employ the COS method, we first truncate the integration range to $[a,b]$

$$v(x,t_0) = e^{-r\Delta t}\int_{a}^{b} v(y,T)f(y|x)dy. \tag{1.11}$$

The size of the truncated domain can be determined with the help of the cumulants [34] [1], discussed in Section 1.2.1.

Then we approximate the conditional density function on the truncated domain, by a truncated Fourier cosine expansion, which recovers the conditional density function from its characteristic function as follows:

$$f(y|x) \approx \frac{2}{b-a}\sum_{k=0}^{N-1}{}' Re\left(\varphi(\frac{k\pi}{b-a};x,\Delta t)\exp\left(-i\frac{ak\pi}{b-a}\right)\right)\cos\left(k\pi\frac{y-a}{b-a}\right), \tag{1.12}$$

[1] For example so that $|\int_{\mathbb{R}} f(y|x)dy - \int_a^b f(y|x)dy| < TOL$.

with $\varphi(u; x, t)$ the characteristic function of $f(y|x)$, defined as

$$\varphi(u; x, t) = \mathbb{E}(e^{iuY}|X = x, T - t_0 = t). \tag{1.13}$$

Moreover, Re means taking the real part of the input argument and the prime at the sum symbol indicates that the first term in the expansion is multiplied by one-half. X, Y are the state variable (here the log–asset) at t_0 and T, respectively.

Replacing $f(y|x)$ by its approximation (1.12) in Equation (1.10) and interchanging integration and summation gives the COS formula for computing the values of European options:

$$v(x, t_0) = e^{-r\Delta t} \sum_{k=0}^{N-1}{}' \, Re(\varphi(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi \frac{a}{b-a}}) V_k, \tag{1.14}$$

where:

$$V_k = \frac{2}{b-a} \int_a^b v(y, T) \cos(k\pi \frac{y-a}{b-a}) dy, \tag{1.15}$$

are the Fourier cosine coefficients of $v(y, T)$, that are available in closed form for several payoff functions, like for plain vanilla puts and calls, but also for example for discontinuous payoffs like for digital options.

It was found by a rigorous analysis in [34], that, with integration interval $[a, b]$ chosen sufficiently wide, the series truncation error dominates the overall error. For conditional density functions $f(y|x) \in C^\infty([a, b] \subset \mathbb{R})$, the method converges exponentially; otherwise convergence is algebraically [35].

Formula (1.14) also forms the basis for the pricing of Bermudan options [35].

1.2.1 Truncation Range and Put–Call Relations

The choice of integration range, $[a, b]$, is quite important. An interval which is chosen too small or too wide will lead to significant integration-range errors.

We use the definition of the integration (also called truncation) range as given [34] and we center the domain at $x_0 := \log(S_0/K)$, i.e.

$$[a, b] := \left[(\xi_1 + x_0) - L\sqrt{\xi_2 + \sqrt{\xi_4}}, \quad (\xi_1 + x_0) + L\sqrt{\xi_2 + \sqrt{\xi_4}} \right], \tag{1.16}$$

with $L \in [6, 12]$ depending on a *user-defined tolerance level*, TOL and $\xi_1, \ldots, \xi_4$ being the cumulants of the underlying stochastic process. The error connected to the size of the domain decreases exponentially with L.

Given the characteristic function, the cumulants, as defined in [23], can be computed via

$$\xi_n(X) = \frac{1}{i^n} \frac{\partial^n (t\Psi(u))}{\partial u^n} \bigg|_{u=0},$$

where $t\Psi(u)$ is the logarithm of the characteristic function of $\log(S_t/S_0)$, which is $\phi(u;t)$, i.e.

$$\phi(u;t) = e^{t\Psi(u)}, \quad t \geq 0.$$

However, when pricing *call options*, the solution's accuracy exhibits sensitivity regarding the size of this truncated domain. This holds specifically for call options under fat-tailed distributions, like under certain Lévy jump processes, or for options with a very long time to maturity [2]. A call payoff grows exponentially in log–stock price which may introduce cancellation errors for large domain sizes. A put option does not suffer from this (see [35]), as their payoff value is bounded by the strike value. In [34], European call options were therefore priced by means of European put option computations, in combination with the *put-call parity*:

$$v^{call}(x,t) = v^{put}(x,t) + S_t e^{-q(T-t)} - Ke^{-r(T-t)}, \tag{1.17}$$

where $v^{call}(x,t)$ and $v^{put}(x,t)$ are the call and put option prices, respectively, and q is again the dividend rate.

Alternatively, one can use the *put–call duality* relation (see also [55]):

$$v^{call}(S,K,r,q,t,\nu) = v^{put}(K,S,q,r,t,e^{-x}\nu(-dx)), \tag{1.18}$$

where [3] measure $\nu(dx)$ is the same as in (1.4) and (1.6). In the case that

$$\nu(dx) = e^{-x}\nu(-dx)$$

is satisfied (for Lévy processes without any jumps, for example), Eqn. (1.18) simplifies:

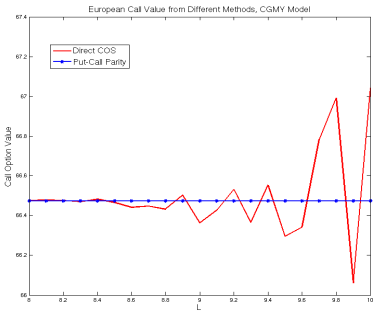
$$v^{call}(S,K,r,q) = v^{put}(K,S,q,r).$$

Figures 1.1 and 1.2 present European call option values under the infinite activity Lévy CGMY jump model, see [53]. The option values obtained by pricing call options directly by the COS method (solid lines) are compared to the values calculated with the put–call parity and put–call duality relations (dotted lines), for different values of parameter L, which determines the sizes of the truncated domain in (1.16). Reference solutions are obtained on a very fine grid.

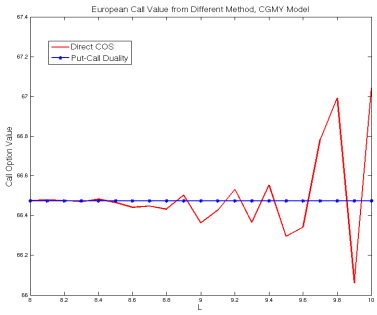
The asset price parameters read $S_0 = 100, K = 110, r = 0.1, q = 0.05$, and the CGMY parameters are chosen as $C = 1, G = 5, M = 5$. For Figure 1.1, with the remaining CGMY parameter $Y = 1.5$, and with $T = 5$, the reference value for the European option is $66.474333\ldots$ and in Figure 1.2 we set $Y = 1.98$, and $T = 0.1$ for which the reference value is $86.826264\ldots$.

[2] This is mainly the case when we consider real options or insurance products with a long life time.

[3] Here we have a long list of arguments, as they are important for the use of the put-call duality.

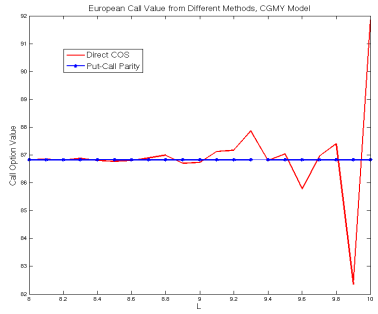


(a) Put-Call Parity

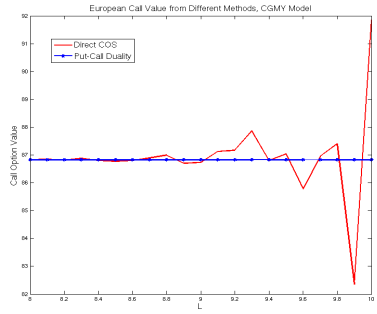


(b) Put-Call Duality

Figure 1.1: Comparison of European call option values, directly obtained by the COS method, with those obtained by the put–call parity and the put–call duality, CGMY model, $Y = 1.5, T = 5$, $L \in [8, 10]$.



(a) Put-Call Parity



(b) Put-Call Duality

Figure 1.2: Comparison of European call option values, directly obtained by the COS method, with those obtained by the put–call parity or put–call duality, CGMY model, $Y = 1.98, T = 0.1$, $L \in [8, 10]$.

As shown in Figures 1.1 and 1.2, the errors appearing, when call prices are directly computed with the COS method, increase for large Y- and T-values, since then the probability density function of the underlying is governed by fat tails. The errors grow drastically as L, i.e. the size of the computational domain, increases. It seems that the choice $L = 6$ results in accurate values in these tests, but this choice is heuristic.

The option prices obtained by the put–call parity or the put–call duality do not deviate from the reference solutions in both test cases, for all integration ranges. The parity and duality lead to robust formulas for pricing European call options by the COS method.

1.2.2 Pricing Early-Exercise Options

A Bermudan option can be exercised at pre-specified dates before maturity. The holder receives the exercise payoff when she exercises the option. We have again t_0 as initial time and $\{t_1, \cdots, t_{\mathcal{M}}\}$ the collection of exercise dates with $\Delta t := (t_m - t_{m-1})$, $t_0 < t_1 < \cdots < t_{\mathcal{M}} = T$. The pricing formula for a Bermudan option with $\mathcal{M}$ exercise dates then reads, for $m = \mathcal{M}, \mathcal{M} - 1, \ldots, 2$:

$$\begin{cases} c(x, t_{m-1}) &= e^{-r\Delta t} \int_{\mathbb{R}} v(y, t_m) f(y|x) dy, \\ v(x, t_{m-1}) &= \max\left(g(x), c(x, t_{m-1})\right), \end{cases} \tag{1.19}$$

followed by

$$v(x, t_0) = e^{-r\Delta t} \int_{\mathbb{R}} v(y, t_1) f(y|x) dy. \tag{1.20}$$

Functions $v(x,t)$, $c(x,t)$ and $g(x)$ are the option value, the continuation value and the payoff at time t, respectively. For call and put options, $g(x) \equiv v(x,T)$, with

$$v(x,T) = \max\left[\alpha K(e^x - 1), 0\right], \quad \alpha = \begin{cases} 1 & \text{for a call,} \\ -1 & \text{for a put,} \end{cases} \tag{1.21}$$

where x and y are state variables at consecutive early–exercise dates t_{m-1} and t_m, defined as

$$x := \log(S_{m-1}/K) \quad \text{and} \quad y := \log(S_m/K).$$

Pricing Bermudan Options by the COS Method

The continuation value in (1.19) can be calculated by means of the COS formula. For exponential Lévy processes it reads:

$$c(x, t_{m-1}) = e^{-r\Delta t} \sum_{k=0}^{N-1}{}' \operatorname{Re}\left\{\phi\left(\frac{k\pi}{b-a}; \Delta t\right) e^{ik\pi\frac{x-a}{b-a}}\right\} V_k(t_m), \tag{1.22}$$

where $\phi(u;t) := \varphi(u;0,t)$, as defined in (1.7).

The technique of pricing Bermudan options by the COS method is based on the computation of the Fourier cosine coefficients of the option value at t_1, $V_k(t_1)$, which are then inserted into (1.20) to get the option value $v(x,t_0)$ by means of the COS formula. The derivation of an induction formula for $V_k(t_1)$, backwards in time, was the basis of the work in [35]. It is briefly explained here.

First, the *early-exercise point*, x_m^*, at time t_m, which is the point where the continuation value equals the payoff, i.e., $c(x_m^*, t_m) = g(x_m^*)$, is determined for example by Newton's method.

Based on x_m^*, we can split $V_k(t_m)$ in Eqn. (1.22) into two parts: One on the interval $[a, x_m^*]$ and another on $(x_m^*, b]$, i.e.

$$V_k(t_m) = \begin{cases} C_k(a, x_m^*, t_m) + G_k(x_m^*, b), & \text{for a call,} \\ G_k(a, x_m^*) + C_k(x_m^*, b, t_m), & \text{for a put,} \end{cases} \tag{1.23}$$

for $m = \mathcal{M} - 1, \mathcal{M} - 2, \cdots, 1$, and at $t_{\mathcal{M}} = T$,

$$V_k(t_{\mathcal{M}}) = \begin{cases} G_k(0, b), \text{ for a call,} \\ G_k(a, 0), \text{ for a put.} \end{cases} \tag{1.24}$$

Here C_k and G_k are the Fourier coefficients for the continuation value and payoff function, respectively, which read,

$$G_k(x_1, x_2) := \frac{2}{b-a} \int_{x_1}^{x_2} g(x) \cos\left(k\pi \frac{x-a}{b-a}\right) dx, \tag{1.25}$$

and

$$C_k(x_1, x_2, t_m) := \frac{2}{b-a} \int_{x_1}^{x_2} c(x, t_m) \cos\left(k\pi \frac{x-a}{b-a}\right) dx. \tag{1.26}$$

For $k = 0, 1, \cdots, N-1$ and $m = 1, 2, \cdots, \mathcal{M}$, the $G_k(x_1, x_2)$ in (1.25) admit analytic solutions, and the challenge is to compute the C_k-coefficients efficiently.

We can generally write characteristic functions as:

$$\varphi(u; x, \tau) = e^{iux\beta} \phi(u; \tau), \tag{1.27}$$

with $\phi(u; \tau)$ not depending on x.

By (1.27), we can distinguish basically two types of stochastic processes in view of their characteristic functions. The first type, governed by $\beta = 1$, which corresponds to a process with independent increments, includes the exponential Lévy processes, for which the characteristic function can thus be written in the form $\varphi(u; x, \tau) = e^{iux} \phi(u; \tau)$. Examples for these are the log-versions of Geometric Brownian Motion, jump-diffusion processes of Kou [47] and Merton [54], infinite activity Lévy processes [23], like Variance-Gamma (VG) [15], Normal Inverse Gaussian (NIG) [3] or CGMY [53].

For the second type of processes, $\phi(u; x, t)$ *cannot* be written as the product of e^{iux} and a function independent of x. An example is the OU mean reverting process, for which $\beta = e^{-\kappa\tau}$ in (1.27), with κ a mean reversion parameter.

In the lemma to follow we will see that characteristic functions of the first type ($\beta = 1$) are beneficial for pricing Bermudan options by the COS method as the Fast Fourier Transform can be applied.

Lemma 1.2.1 (Efficient Computation)**.** *The terms $C_k(x_1, x_2, t_m)$ can be computed in $O(N \log_2 N)$ operations, if the stochastic process for the underlying is governed by general characteristic function (1.27) with parameter $\beta = 1$.*

Proof. At times t_m, $m = \mathcal{M} - 1, \cdots, 1$, from Equations (1.19) and (1.22), we obtain an approximation for $c(x, t_m)$, the continuation value at t_m, which is inserted into (1.26). Interchanging summation and integration gives the following coefficients, $C_k(x_1, x_2, t_m)$:

$$C_k(x_1, x_2, t_m) := e^{-r\Delta t} \sum_{j=0}^{N-1}{}' Re\left(\phi\left(\frac{j\pi}{b-a}; \Delta t \right) V_j(t_{m+1}) \cdot H_{k,j}(x_1, x_2) \right), \tag{1.28}$$

where $\phi(u; \Delta t)$ comes from the general expression for the characteristic function (1.27). To get $C_k(x_1, x_2, t_m)$, the following integrals need to be computed:

$$H_{k,j}(x_1, x_2) = \frac{2}{b-a} \int_{x_1}^{x_2} e^{ij\pi \frac{\beta x - a}{b-a}} \cos(k\pi \frac{x-a}{b-a}) dx,$$

with β defined in (1.27).

By basic calculus, we can split $H_{k,j}(x_1, x_2)$ into two parts as

$$H_{k,j}(x_1, x_2) = -\frac{i}{\pi}(H^s_{k,j}(x_1, x_2) + H^c_{k,j}(x_1, x_2)),$$

where

$$H^c_{k,j}(x_1, x_2) = \begin{cases} \frac{(x_2 - x_1)\pi i}{b-a}, & \text{if } k = j = 0, \\ \frac{1}{(j\beta + k)} \left[\exp\left(\frac{((j\beta + k)x_2 - (j+k)a)\pi i}{b-a} \right) - \right. & \\ \left. \exp\left(\frac{((j\beta + k)x_1 - (j+k)a)\pi i}{b-a} \right) \right], & \text{otherwise.} \end{cases} \tag{1.29}$$

and

$$H^s_{k,j}(x_1, x_2) = \begin{cases} \frac{(x_2 - x_1)\pi i}{b-a}, & \text{if } k = j = 0, \\ \frac{1}{(j\beta - k)} \left[\exp\left(\frac{((j\beta - k)x_2 - (j-k)a)\pi i}{b-a} \right) - \right. & \\ \left. \exp\left(\frac{((j\beta - k)x_1 - (j-k)a)\pi i}{b-a} \right) \right], & \text{otherwise.} \end{cases} \tag{1.30}$$

Matrices H^s and H^c have a Toeplitz and Hankel structure, respectively, *if* $H^s_{k,j}(x_1, x_2) = H^s_{k+1,j+1}(x_1, x_2)$ and $H^c_{k,j}(x_1, x_2) = H^c_{k+1,j-1}(x_1, x_2)$, *which is the case for* $\beta \equiv 1$. In other words, pricing Bermudan options can be done highly efficiently when exponential Lévy asset price models are employed. Then, the Fast Fourier Transform can be applied directly for matrix-vector multiplication [35], and the resulting computational complexity of $C_k(x_1, x_2, t_m)$ is $O(N \log_2 N)$. □

We would obtain terms of the form $j\beta - k, j\beta + k$ in the matrix elements in (1.29) and (1.30), instead of terms with $j - k, j + k$ if $\beta \neq 1$ in (1.27). Terms with β not being an integer hamper an efficient computation of the matrix-vector products, leading to computations with $O(N^2)$ complexity.

American Options

For the valuation of American options by the COS method, there are basically two approaches. One is to approximate an American option by a Bermudan option with many exercise opportunities, the other is to use repeated Richardson extrapolation on a series of Bermudan options with an increasing number of exercise opportunities. Here we will focus on the extrapolation-based method, which has been described in detail in [20], although the approach dates back to [38].

Let here $\hat{v}(\mathcal{M})$ be the price of a Bermudan option with $\mathcal{M}$ exercise dates with a maturity of T years where the exercise dates are $\Delta t = T/\mathcal{M}$ years apart. It is assumed that $\hat{v}(\mathcal{M})$ can be expanded as:

$$\hat{v}(\mathcal{M}) = v_{AM} + \sum_{i=1}^{\infty} a_i (\Delta t)^{\gamma_i}, \tag{1.31}$$

with $0 < \gamma_i < \gamma_{i+1}$; v_{AM} is the American option value. Classical extrapolation procedures assume that the exponents γ_i are known, which means that we can use $n + 1$ Bermudan prices with varying Δt to eliminate the n leading order terms in (1.31). The prices of American options can be obtained by applying repeated Richardson extrapolation on the values of a few Bermudan options with small $\mathcal{M}$. We use the following 4-point repeated Richardson extrapolation scheme,

$$\hat{v}_{AM}(\mathcal{M}) = \frac{1}{21}\left(64\hat{v}(8\mathcal{M}) - 56\hat{v}(4\mathcal{M}) + 14\hat{v}(2\mathcal{M}) - \hat{v}(\mathcal{M})\right), \tag{1.32}$$

where $\hat{v}_{AM}(\mathcal{M})$ denotes the approximated value of the American option [4].

1.3 Error Analysis

In this subsection we give error analysis for the COS pricing method, *focusing on Bermudan call options*. First, we analyze the local error, i.e., the error in the continuation values at each time step. A similar error analysis has been performed in [34], where, however, the influence of the call payoff function on the global error convergence was omitted. Here, we study the influence of the payoff function and the integration range on the error convergence.

[4] Without any dividend payments, of course, the American call option value is equal to the European call option value.

It has been shown, [34], that the error in the continuation value, calculated by the COS method, consists of three parts, denoted by ϵ_1, ϵ_2 and ϵ_3, respectively.

Error ϵ_1 is the integration range error

$$|\epsilon_1(x,[a,b])| = e^{-r\Delta t}\int_{\mathbb{R}\setminus[a,b]} v(y,T)f(y|x)dy,$$

which depends on the payoff function and the integration range.

Error ϵ_2 is the series truncation error on $[a,b]$, which depends on the smoothness of the probability density function of the underlying processes:

$$\epsilon_2\left(x;N,[a,b]\right) := e^{-r\Delta t}\sum_{k=N}^{\infty}\text{Re}\left\{e^{-ik\pi\frac{a}{b-a}}\int_a^b e^{i\frac{k\pi}{b-a}y}f(y|x)dy\right\}\cdot V_k. \quad (1.33)$$

For probability density functions $f(y|x)\in\mathbf{C}^{\infty}[a,b]$, we have

$$|\epsilon_2(x,N,[a,b])| < P\exp(-(N-1)\nu),$$

where N is the number of terms in the Fourier cosine expansions, $\nu>0$ is a constant and P is a term which varies less than exponentially with respect to N. When the probability density function has a discontinuous derivative, then the Fourier cosine expansions converge algebraically,

$$|\epsilon_2(x,N,[a,b])| < \frac{P}{(N-1)^{\beta-1}},$$

where P is a constant and $\beta\geq 1$ is the algebraic index of convergence.

Error ϵ_3 is the error related to the approximation of the Fourier cosine coefficients of the density function in terms of its characteristic function, which reads

$$|\epsilon_3(x,N,[a,b])| = e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\int_{\mathbb{R}\setminus[a,b]} e^{ik\pi\frac{y-a}{b-a}}f(y|x)dy)V_k.$$

It can be shown that

$$|\epsilon_3(x,N,[a,b])| < e^{-r\Delta t}Q_1\int_{\mathbb{R}\setminus[a,b]} f(y|x)dy,$$

where Q_1 is a positive constant.

We denote by

$$\begin{aligned} I_1 &= \int_{\mathbb{R}\setminus[a,b]} v(y,T)f(y|x)dy,\\ I_2 &= \int_{\mathbb{R}\setminus[a,b]} f(y|x)dy, \end{aligned}$$

so that $\epsilon_1 = e^{-r\Delta t} I_1$, $\epsilon_3 < e^{-r\Delta t} Q_1 I_2$. Integral I_1 then depends on the payoff function and the integration range, whereas I_2 depends only on the integration range.

We start with a discussion about the influence of the payoff function on the error convergence and then we analyze the influence of L in (1.16).

For an option with a bounded payoff function, such as a put option, we have $\forall y$, $v(y,T) \leq Q_2$, so that it follows directly that

$$I_1 \leq Q_2 I_2, \tag{1.34}$$

and both ϵ_1 and ϵ_3 can be controlled by means of parameter L. This was the basis for the detailed error analysis for Bermudan put options in [35].

However, in the case of an unbounded payoff, for instance, a call option, we have:

$$\begin{aligned} I_1 &= \int_{R\setminus[a,b]} v(y,T) f(y|x) dy \geq \int_b^\infty v(y,T) f(y|x) dy \\ &= \int_b^\infty (Ke^y - K)^+ f(y|x) dy \geq K(e^b - 1) \int_b^\infty f(y|x) dy \end{aligned} \tag{1.35}$$

Note that we assume that $b \geq 0$, as otherwise for all $y \in [a,b]$, $v(y,T) = 0$ and the option value is also zero.

Function $\int_b^\infty f(y|x)dy$ is bounded by $0 < \int_b^\infty f(y|x)dy < 1$.

Denoting by $Q_3 \triangleq K \int_b^\infty f(y|x)dy$ then

$$I_1 \geq Q_3(e^b - 1).$$

Function $e^b - 1$ will, however, not decrease to zero as N, the number of terms in the Fourier cosine expansion, goes to infinity. Furthermore, the larger the integration range, the larger the value $e^b - 1$, i.e. the error in the option price. Given the fact that $\epsilon_1 = e^{-r\Delta t} Q_3(e^b - 1)$, the global error in the call option price may increase as the integration range $[a,b]$ (or L) increases. This implies that when we directly use the COS formula for a call option, the value may diverge, depending on the decay rate of $f(y|x)$. This is not the case if a very small integration range (or a very small value of L) is used, but by this error ϵ_3 may increase. This is the next topic in the error analysis.

To study the influence of truncation on the error convergence, we start the analysis with the Black–Scholes model. From the cumulative density function (which is known analytically) it follows that with $L = 6$, we find $I_2 = 1.9732 \times 10^{-9}$ and with $L = 8$, we have $I_2 = 1.3323 \times 10^{-15}$, so that with $L \in [6,8]$ the errors ϵ_1 and ϵ_3 can be controlled. Incorporating jumps in a Lévy model gives rise to a slightly larger value of L. As shown in [35], an integration range with $L \in [8,10]$ is sufficient for most of the Lévy processes with $T > 0.1$ to bound I_2 (but not always for I_1).

In general, from Chebyshev's inequality we know that for any random variable, X, with expected value μ and finite variance σ and for any real number $k > 0$, $Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$, which implies

$$\begin{aligned} I_2(x_0) &= \int_{\mathbb{R}\setminus[a,b]} f(y|x_0)dy = Pr(|X_T - (\xi_1 + x_0)| \geq L(\xi_2 + \sqrt{\xi_4})) \\ &\leq Pr(|X_T - (\xi_1 + x_0)| \geq L(\xi_2)) \leq \frac{1}{L^2}. \end{aligned}$$

Therefore for all processes and model parameters, I_2 decays at least algebraically with algebraic index $n \geq 2$.

1.4 Pricing Bermudan Call Options Using Put-Call Relations

In this section, we present two techniques to deal efficiently with the inaccurate pricing by the COS method for Bermudan call options. With our improved method, the Fourier cosine coefficients of call options need not to be calculated directly at each time step, which will eliminate the error related to the unbounded payoff of call options. In Section 1.4.1 we discuss the use of the *put-call parity relation*, and in Section 1.4.2 we explain the use of the *put-call duality relation*. These techniques are accompanied by error analysis in Section 1.4.3.

1.4.1 The Put–Call Parity

Here we give details of the use of the *European* put-call parity for the robust pricing of *Bermudan* call options by means of the COS method.

At each time step we need to calculate the continuation value based on the Fourier coefficients of the call option payoff. The continuation value is then used to determine the early–exercise points, and to recover the Fourier cosine coefficients for a next time step. In these steps, the influence of an exponentially–increasing payoff can be significant, similar as for European call options. Here, we modify the pricing algorithm for Bermudan call options employing put–call parity (1.17).

We denote the Fourier cosine coefficients for a put and a call option at $t_{\mathcal{M}} = T$ by $V_k^{put}(t_{\mathcal{M}})$ and $V_k^{call}(t_{\mathcal{M}})$, respectively. By (1.17) we then find

$$\begin{aligned} & e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}; \Delta t) e^{i(x-a)\frac{k\pi}{b-a}}) \cdot V_k^{call}(t_{\mathcal{M}}) \qquad (1.36) \\ = \; & S_t e^{-q\Delta t} - K e^{-r\Delta t} + e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}; \Delta t) e^{i(x-a)\frac{k\pi}{b-a}}) \cdot V_k^{put}(t_{\mathcal{M}}). \end{aligned}$$

We have $V_k^{put}(t_{\mathcal{M}}) = G_k^{put}(a,0)$ and $V_k^{call}(t_{\mathcal{M}}) = G_k^{call}(0,b)$, where G_k^{put} and G_k^{call} are the Fourier cosine coefficients for the respective payoffs. So, we can write (1.36) as:

$$e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a};\Delta t)e^{i(x-a)\frac{k\pi}{b-a}})G_k^{call}(0,b) = Se^{-q\Delta t} - Ke^{-r\Delta t} + e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a};\Delta t)e^{i(x-a)\frac{k\pi}{b-a}})G_k^{put}(a,0). \quad (1.37)$$

Equation (1.37) will be used in the backward recursion.

At $t = \mathcal{M}-1$, we first determine the early–exercise point, $x^*_{\mathcal{M}-1}$, by Newton's method, for which the functions $c, g, \partial c/\partial x, \partial g/\partial x$ are required. The continuation value for the call option now reads, using (1.36):

$$\begin{aligned} c(x,t_{\mathcal{M}-1}) &= e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a};\Delta t)e^{i(x-a)\frac{k\pi}{b-a}})V_k^{put}(t_{\mathcal{M}}) \\ &+ Ke^x e^{-q\Delta t} - Ke^{-r\Delta t}, \end{aligned} \quad (1.38)$$

with $x = \log(S/K)$, and similarly we find:

$$\begin{aligned} \frac{\partial c}{\partial x} &= e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a};\Delta t)e^{i(x-a)\frac{k\pi}{b-a}}(\frac{ik\pi}{b-a}))V_k^{put}(t_{\mathcal{M}}) \\ &+ Ke^x e^{-q\Delta t}. \end{aligned} \quad (1.39)$$

With $x \geq 0$, we have $g(x) = Ke^x - K$ and $\partial g/\partial x = Ke^x$, whereas for $x < 0$ both the payoff and its derivative are zero, for all time steps.

With the early-exercise point determined, we need to compute the values,

$$V_k^{call}(t_{\mathcal{M}-1}) := C_k^{call}(a, x^*_{\mathcal{M}-1}, t_{\mathcal{M}-1}) + G_k^{call}(x^*_{\mathcal{M}-1}, b). \quad (1.40)$$

Application of (1.38) gives us:

$$\begin{aligned} C_k^{call}(a, x^*_{\mathcal{M}-1}, t_{\mathcal{M}-1}) &= \frac{2}{b-a}\int_a^{x^*_{\mathcal{M}-1}} c(x,t_{\mathcal{M}-1})\cos(k\pi\frac{x-a}{b-a})dx \\ &= \frac{e^{-r\Delta t}}{\pi} Im(H^c + H^s)u + \frac{2}{b-a}Ke^{-q\Delta t}\chi(a, x^*_{\mathcal{M}-1}) \\ &- \frac{2}{b-a}Ke^{-r\Delta t}\psi(a, x^*_{\mathcal{M}-1}) \end{aligned} \quad (1.41)$$

where Im means taking the imaginary part of the input argument, vector u consists of values:

$$u_j = \phi(\frac{k\pi}{b-a};\Delta t)V_j^{put}(t_{\mathcal{M}}), j = 1, \cdots, N-1,$$

and $u_0 = \frac{1}{2}\phi(0;\Delta t)V_0^{put}(t_{\mathcal{M}})$. Matrices H^c, H^s are as in Equations (1.29) and (1.30), with $\beta = 1$. Moreover,

$$\begin{aligned}\chi(x_1,x_2) &= \int_{x_1}^{x_2} e^x \cos(\frac{k\pi(x-a)}{b-a})dx,\\ \psi(x_1,x_2) &= \int_{x_1}^{x_2} \cos(\frac{k\pi(x-a)}{b-a})dx,\end{aligned} \tag{1.42}$$

both of which have an analytic solution.

We further have $G_k^{call}(x^*_{\mathcal{M}-1}, b) = G_k^{call}(0,b) - G_k^{call}(0, x^*_{\mathcal{M}-1})$, and $\forall x \in (0, x^*_{\mathcal{M}-1})$ the payoff of a call option is less than the continuation value. Therefore, $G_k^{call}(0, x^*_{\mathcal{M}-1})$ can be calculated directly and it will remain accurate, independent of the choice of integration range. Quantity $G_k^{call}(0,b)$ will be replaced by $G_k^{put}(a,0)$ via (1.37).

We now arrive at the following lemma:

Lemma 1.4.1. *Quantities $c(x,t_m)$, x_m^*, and $C_k^{call}(a, x_m^*, t_m)$ can be recovered from $C_k^{call}(a, x_{m+1}^*, t_{m+1})$ in an accurate way for $m = \mathcal{M}-2, \ldots, 1$, with a computational complexity of $O(N \log_2 N)$ at each time step. $C_k^{call}(a, x_1^*, t_1)$ is then finally also recovered in a robust way.*

Proof. At the time steps t_m, $m = \mathcal{M}-2, \cdots, 1$, the continuation value reads

$$\begin{aligned}c(x,t_m) &= e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a};\Delta t)e^{i(x-a)\frac{k\pi}{b-a}})V_k^{call}(t_{m+1})\\ &= e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a};\Delta t)e^{i(x-a)\frac{k\pi}{b-a}})(C_k^{call}(a, x_{m+1}^*, t_{m+1})\\ &- G_k^{call}(0, x_{m+1}^*)) + e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a};\Delta t)e^{i(x-a)\frac{k\pi}{b-a}})G_k^{call}(0,b)\\ &= e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a};\Delta t)e^{i(x-a)\frac{k\pi}{b-a}})(C_k^{call}(a, x_{m+1}^*, t_{m+1})\\ &+ G_k^{put}(a,0) - G_k^{call}(0, x_{m+1}^*)) + Ke^x e^{-q\Delta t} - Ke^{-r\Delta t},\end{aligned} \tag{1.43}$$

where the last step is from (1.37). Derivative $\partial c/\partial x$ can be obtained similarly. Together with $g(x)$ and $\partial g/\partial x$, they are used to determine early–exercise point x_m^* at t_m.

Furthermore,

$$\begin{aligned}C_k^{call}(a, x_m^*, t_m) &= \frac{e^{-r\Delta t}}{\pi} Im(H^c + H^s)u + \frac{2}{b-a}Ke^{-q\Delta t}\chi(a, x_m^*)\\ &- \frac{2}{b-a}Ke^{-r\Delta t}\psi(a, x_m^*),\end{aligned}$$

where H^c, H^s are as defined earlier in Equations (1.29) and (1.30) with $\beta = 1$ and vector u consists of elements:

$$u_j = \phi(\frac{k\pi}{b-a}; \Delta t)(C_j^{call}(a, x_{m+1}^*, t_{m+1}) + G_j^{put}(a,0) - G_j^{call}(0, x_{m+1}^*)), \quad (1.44)$$

and

$$u_0 = \frac{1}{2}\phi(0; \Delta t)(C_0^{call}(a, x_{m+1}^*, t_{m+1}) + G_0^{put}(a,0) - G_0^{call}(0, x_{m+1}^*)). \quad (1.45)$$

Regarding the computational costs, at each time step $C_k^{call}(a, x_m^*, t_m)$ needs to be calculated once. Therefore we have the same computational complexity as the original COS method, which is $O(\mathcal{M}-1)N\log_2 N$.

Finally, the two terms $G_k^{put}(a,0)$ and $G_k^{call}(0, x_m^*)$ at t_m admit analytic solutions.

At t_0 we have

$$\begin{aligned} v(x,t_0) &= e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}; \Delta t)e^{i(x-a)\frac{k\pi}{b-a}})V_k(t_1) \\ &= e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}; \Delta t)e^{i(x-a)\frac{k\pi}{b-a}})(C_k^{call}(a, x_1^*, t_1) \\ &+ G_k^{put}(a,0) - G_k^{call}(0, x_1^*)) + Ke^x e^{-q\Delta t} - Ke^{-r\Delta t}, \end{aligned} \quad (1.46)$$

where the last step follows from (1.37) and we complete the robust and efficient pricing of Bermudan options via the put-call parity relation. □

1.4.2 The Put–Call Duality

In this section, we discuss a second possibility to price a Bermudan call with the help of the pricing formula for a put. It is based on the put-call duality from [55].

In the COS pricing formula (1.14), $r, q, \nu(dx)$ are essential in the definition of the characteristic function ϕ, whereas S and K enter the formula for the Fourier cosine coefficients, V_k. Therefore, we use in this section the notation $\phi := \phi(u, t, r, q, \nu)$. Moreover, we use $V_k^{call}(t_m, S, K)$, $V_k^{put}(t_m, S, K)$, $V_k(t_m)$ to denote the Fourier cosine coefficients of European call options (with stock price S and strike price K), of European put options and the Fourier cosine coefficients of a Bermudan option at t_m, respectively. We also denote $e^{-x}\nu(dx)$ by $\tilde{\nu}(dx)$.

We start at $t_{\mathcal{M}} = T$. From $t_{\mathcal{M}}$ to $t_{\mathcal{M}-1}$ the direct application of (1.18)

gives us

$$
\begin{aligned}
c(x, t_{\mathcal{M}-1}) &= \mathrm{e}^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}, \Delta t, r, q, \nu) e^{i(x-a)\frac{k\pi}{b-a}}) V_k^{call}(t_{\mathcal{M}}, S, K) \\
&= \mathrm{e}^{-q\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}, \Delta t, q, r, \tilde{\nu}) e^{i(-x-a)\frac{k\pi}{b-a}}) V_k^{put}(t_{\mathcal{M}}, K, S)
\end{aligned}
\tag{1.47}
$$

where $V_k^{call}(t_{\mathcal{M}}, S, K) = G_k^{call}(0, b)$, and

$$
\begin{aligned}
V_k^{put}(t_{\mathcal{M}}, K, S) &= \frac{2}{b-a} \int (S - Se^y) \cos(k\pi \frac{y-a}{b-a}) dy \\
&= \frac{2K}{b-a} e^x \int (1 - e^y) \cos(k\pi \frac{y-a}{b-a}) dy = e^x G_k^{put}(a, 0).
\end{aligned}
$$

Note that for both S and K as state variables in the put–call duality formulation, integration ranges need to be defined. We set $a = \min(a_S, a_K)$, $b = \max(b_S, b_K)$. The use of "$-x$" in the second equation in (1.47) appears because the state variable $\log(K/S) = -\log(S/K) = -x$.

At $t_{\mathcal{M}-1}$ the continuation value and its derivative read:

$$
c(x, t_{\mathcal{M}-1}) = \mathrm{e}^{-q\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}, \Delta t, q, r, \tilde{\nu}) e^{i(-x-a)\frac{k\pi}{b-a}}) e^x G_k^{put}(a, 0),
\tag{1.48}
$$

$$
\begin{aligned}
\frac{\partial c(x, t_{\mathcal{M}-1})}{\partial x} &= \mathrm{e}^{-q\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}, \Delta t, q, r, \tilde{\nu}) e^{i(-x-a)\frac{k\pi}{b-a}} (-\frac{ik\pi}{b-a})) \\
&\cdot \quad e^x G_k^{put}(a, 0) + c(x, t_{\mathcal{M}-1}),
\end{aligned}
$$

which are used to calculate the early–exercise point $x^*_{\mathcal{M}-1}$ by Newton's method, so that

$$
\begin{aligned}
V_k(t_{\mathcal{M}-1}) &= C_k(a, x^*_{\mathcal{M}-1}) + G_k^{call}(x^*_{\mathcal{M}-1}, b) \\
&= C_k(a, x^*_{\mathcal{M}-1}, t_{\mathcal{M}-1}) - G_k^{call}(0, x^*_{\mathcal{M}-1}) + G_k^{call}(0, b)
\end{aligned}
\tag{1.49}
$$

Now, $\forall x \in (0, x^*_{\mathcal{M}-1})$ the payoff of the call option is less than the continuation value. Therefore, $G_k^{call}(0, x^*_{\mathcal{M}-1})$ can be calculated directly and it will be accurate with respect to the size of the integration range; $G_k^{call}(0, b)$ can be replaced by $G_k^{put}(a, 0)$, in a similar way as (1.47).

The computation of C_k represents again the main part of the algorithm. First, we demonstrate how to compute $C_k(x_1, x_2, t_{\mathcal{M}-1})$ in (1.49) with the help of the Fast Fourier Transform (FFT), then we will show that for all

$m = \mathcal{M}-2, \cdots, 1$, $C_k(x_1, x_2, t_m)$ can be recovered from $C_k(x_1, x_2, t_{m+1})$. We denote $D(x_1, x_2) := \{D_k(x_1, x_2)\}_{k=0}^{N-1}$, with

$$D_k(x_1, x_2) = e^{-q\Delta t} Re(\sum_{j=0}^{N-1}{}' \phi(\frac{j\pi}{b-a}, \Delta t, q, r, \tilde{\nu}) G_j^{put}(a,0) J_{k,j}(x_1, x_2)), \tag{1.50}$$

in which

$$J_{k,j}(x_1, x_2) := \frac{2}{b-a} \int_{x_1}^{x_2} e^{ij\pi} \frac{\beta x - a}{b-a} \cos(k\pi \frac{x-a}{b-a}) dx.$$

where now $\beta = -1 - \frac{i(b-a)}{j\pi}$, which is different from $\beta = 1$. However, this β-value still results in a sum of a Toeplitz plus Hankel matrix.

Application of (1.26) and (1.48) gives $C_k(x_1, x_2, t_{\mathcal{M}-1}) = D_k(x_1, x_2)$, $\forall k = 0, \cdots, N-1$.

First, we study the structure of $J_{k,j}$ then we compute $D(x_1, x_2)$. From (1.29) and (1.30), we find that

$$J_{k,j}(x_1, x_2) = -\frac{i}{\pi}(J_{k,j}^c(x_1, x_2) + J_{k,j}^s(x_1, x_2)),$$

with

$$\begin{aligned}
J_{k,j}^s(x_1, x_2) &= \frac{(-1)}{(j-k) + \frac{i}{\pi}(b-a)} (\exp(x_2) \exp(-\frac{(j-k)x_2\pi i}{b-a}) \\
&- \exp(x_1) \exp(-\frac{(j-k)x_1\pi i}{b-a})) \exp(-\frac{(j+k)a\pi i}{b-a}) \\
&= \frac{(-1)}{(j-k) + \frac{i}{\pi}(b-a)} (\exp(x_2) \frac{1}{\exp(\frac{(j-k)x_2\pi i}{b-a})} \\
&- \exp(x_1) \frac{1}{\exp(\frac{(j-k)x_1\pi i}{b-a})}) \exp(\frac{(j-k)a\pi i}{b-a}) \frac{1}{\exp(\frac{2ja\pi i}{b-a})},
\end{aligned}$$

and

$$\begin{aligned}
J_{k,j}^c(x_1, x_2) &= \frac{(-1)}{(j+k) + \frac{i}{\pi}(b-a)} (\exp(x_2) \exp(-\frac{(j+k)x_2\pi i}{b-a}) \\
&- \exp(x_1) \exp(-\frac{(j+k)x_1\pi i}{b-a})) \exp(-\frac{(j-k)a\pi i}{b-a}) \\
&= \frac{(-1)}{(j+k) + \frac{i}{\pi}(b-a)} (\exp(x_2) \frac{1}{\exp(\frac{(j+k)x_2\pi i}{b-a})} \\
&- \exp(x_1) \frac{1}{\exp(\frac{(j+k)x_1\pi i}{b-a})}) \exp(\frac{(j+k)a\pi i}{b-a}) \frac{1}{\exp(\frac{2ja\pi i}{b-a})}.
\end{aligned}$$

We denote $\mathbf{u} := \{u_j\}_{j=0}^{N-1}$ with

$$\begin{aligned} u_j &= \phi(\frac{j\pi}{b-a}, \Delta t, q, r, \tilde{\nu}) G_j^{put}(a,0) \frac{1}{\exp(\frac{2ja}{b-a}\pi i)}, \\ u_0 &= \frac{1}{2}\phi(0, \Delta t, q, r, \tilde{\nu}) G_0^{put}(a,0), \end{aligned}$$

and we have

$$D = \frac{e^{-q\Delta t}}{\pi} Im\{(J^c + J^s)u\},$$

where J^s is a Toeplitz matrix and J^c is a Hankel matrix.

Matrix-vector multiplications can be performed highly efficiently then, with the help of the FFT.

With the use of the Fast Fourier and Inverse Fast Fourier Transforms, the computational complexity of $C_k(a, x^*_{\mathcal{M}-1}, t_{\mathcal{M}-1})$ is $O(N \log_2 N)$.

We then have the following lemma:

Lemma 1.4.2. *For $m = \mathcal{M}-2, \cdots, 1$, $c(x,t_m)$, $x^*_m, C_k(a, x^*_m, t_m)$ can all be recovered from $C_k(a, x^*_{m+1}, t_{m+1})$ with computational complexity $O(N \log_2 N)$ at each time step. $C_k(a, x^*_1, t_1)$ is recovered at the final step.*

Proof. For any $m = \mathcal{M}-2, \cdots, 1$, the continuation value reads:

$$\begin{aligned} c(x,t_m) &= e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}, \Delta t, r, q, \nu) e^{i(x-a)\frac{k\pi}{b-a}} V_k(t_{m+1})) \\ &= e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}, \Delta t, r, q, \nu) e^{i(x-a)\frac{k\pi}{b-a}}) \cdot \\ &\quad (C_k(a, x^*_{m+1}, t_{m+1}) - G_k^{call}(0, x^*_{m+1}) + G_k^{call}(0,b)) \\ &= e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}, \Delta t, r, q, \nu) e^{i(x-a)\frac{k\pi}{b-a}}) \\ &\cdot \quad (C_k(a, x^*_{m+1}, t_{m+1}) - G_k^{call}(0, x^*_{m+1})) \qquad (1.51) \\ &+ \quad e^{-q\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}, \Delta t, q, r, \tilde{\nu}) e^{i(-x-a)\frac{k\pi}{b-a}}) e^x G_k^{put}(a,0). \end{aligned}$$

The last step is from (1.18) and (1.47) and the fact that $V_k^{put}(K,S) = e^x G_k^{put}(a,0)$. $G_k^{call}(0, x^*_{m+1})$ and $G_k^{put}(a,0)$ can be calculated directly from their analytic solutions.

By (1.51) the continuation value, $c(x,t_m)$, is recovered from $C_k(a, x^*_{m+1}, t_{m+1})$ and $\partial c(x,t_m)/\partial x$ is directly calculated with (1.51).

The continuation value and its derivative are then used in the Newton method to find early–exercise point x^*_m, which splits $V_k(t_m)$ as follows:

$$V_k(t_m) = C_k(a, x^*_m, t_m) - G_k^{call}(0, x^*_m) + G_k^{call}(0,b).$$

From (1.51) we have that

$$\begin{aligned} C_k(a, x_m^*, t_m) &= \frac{2}{b-a}\int_a^{x_m^*} c(x, t_m)\cos(k\pi\frac{x-a}{b-a})dx \\ &= \frac{e^{-r\Delta t}}{\pi} Im((H^c(a, x_m^*) + H^s(a, x_m^*))u^1) + \\ &\quad \frac{e^{-q\Delta t}}{\pi} Im((J^c(a, x_m^*) + J^s(a, x_m^*))u^2), \end{aligned} \tag{1.52}$$

where we have four matrix-vector multiplications, instead of the usual two.

Matrices H^c and H^s are defined in (1.29) and (1.30), respectively, with $\beta = 1$. Moreover, we have in (1.52):

$$\begin{aligned} u_0^1 &= \frac{1}{2}\phi(0, \Delta t, r, q, \nu)(C_0(a, x_{m+1}^*, t_{m+1}) - G_0^{call}(0, x_{m+1}^*)), \\ u_j^1 &= \phi(\frac{j\pi}{b-a}, \Delta t, r, q, \nu)(C_j(a, x_{m+1}^*, t_{m+1}) - G_j^{call}(0, x_{m+1}^*)), j = 1, \cdots, N-1, \\ u_0^2 &= \frac{1}{2}\phi(0, \Delta t, q, r, \tilde{\nu})G_0^{put}(a, 0). \\ u_j^2 &= \phi(\frac{j\pi}{b-a}, \Delta t, q, r, \tilde{\nu})G_j^{put}(a, 0)\frac{1}{\exp(\frac{2ja}{b-a}\pi i)}, j = 1, \cdots, N-1, \end{aligned}$$

H^c and J^c are Hankel matrices, H^s and J^s are Toeplitz matrices. Therefore, the Fast Fourier Transform can be employed to compute $C_k(a, x_m^*, t_m)$, $m = \mathcal{M} - 2, \cdots, 1$ and the computational complexity at each time step is $O(N \log_2 N)$.

From (1.51) and (1.52), $\forall m = \mathcal{M}-2, \cdots, 1$, $c(x, t_m)$, x^* and $C_k(a, x^*, t_m)$ can be recovered from $C_k(a, x_{m+1}^*, t_{m+1})$ with the help of the Fast Fourier Transform, which finishes the proof. □

With $C_k(a, x_1^*, t_1)$ known, the call option price then reads:

$$\begin{aligned} v(x_0, t_0) &= e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}, \Delta t, r, q, \nu)e^{i(x_0-a)\frac{k\pi}{b-a}})V_k(t_1) \\ &= e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}, \Delta t, r, q, \nu)e^{i(x_0-a)\frac{k\pi}{b-a}}) \\ &\quad \cdot (C_k(a, x_1^*, t_1) - G_k^{call}(0, x_1^*)) \\ &\quad + e^{-q\Delta t}\sum_{k=0}^{N-1}{}' Re(\phi(\frac{k\pi}{b-a}, \Delta t, q, r, \tilde{\nu})e^{i(-x_0-a)\frac{k\pi}{b-a}})e^{x_0}G_k^{put}(a, 0). \end{aligned} \tag{1.53}$$

1.4.3 Error analysis with the put-call relations

As shown in the previous sections, European put option values, combined with the put–call parity or the put–call duality relations, are used to price

European call options with the COS method. We denote by v_{call} and v_{put} the exact European call and put option values, respectively, and by $\hat{v}_{put}$ the put option value obtained by the COS method. Then, from the put–call parity, we have, $\forall x, t$,

$$\begin{aligned}\epsilon_{call}(x,t) &= v^{call}(x,t) - \hat{v}^{call}(x,t)\\ &= v^{put}(x,t) + Ke^{x}e^{-q(T-t)} - Ke^{-r(T-t)}\\ &- (\hat{v}^{put}(x,t) + Ke^{x}e^{-q(T-t)} - Ke^{-r(T-t)})\\ &= v^{put}(x,t) - \hat{v}^{put}(x,t) = \epsilon_{put}(x,t),\end{aligned}$$

whereas for the put–call duality, we find:

$$\begin{aligned}\epsilon_{call} &= v^{call}(S,K,r,q,t,v) - \hat{v}^{call}(S,K,r,q,t,v)\\ &= v^{put}(K,S,q,r,e^{-x}\nu(-dx)) - \hat{v}^{put}(K,S,q,r,e^{-x}\nu(-dx)) = \epsilon_{put}.\end{aligned}$$

So, by means of the put–call relations, the error of the European call option equals that of a put option. As for put options the payoff is bounded, we have from (1.34):

$$|\epsilon_1(x,[a,b])| = e^{-r\Delta t}I_1 \leq e^{-r\Delta t}Q_2 I_2. \tag{1.54}$$

The error can be controlled if the integration range is sufficiently large (which is our next issue). The above error analysis also implies that the error in the continuation value of an early–exercise call option, as calculated from the put–call relation at each time step, equals the error in the continuation value of the corresponding put option.

The integration range is defined as in (1.16) and can be controlled by parameter L.

After discussing the influence of the payoff and integration range on the error convergence separately in the previous section, we give a remark on the interaction of them on the error convergence of ϵ_1.

Remark 1.4.1 (Interaction of Payoff and Truncation Range on ϵ_1)**.** *From (1.35) we see that*

$$\epsilon_1 = e^{-r\Delta t}I_1 \geq e^{-r\Delta t}K(e^{b}-1)\int_b^{\infty} f(y|x)dy.$$

For the Black–Scholes model and other underlying processes for which the density function decays very fast both at left and right tails, the fast decay in $\int_b^{\infty} f(y|x)dy$ can compensate the exponential increase in $e^b - 1$. On the other hand, for underlying processes with fat tails, for instance, the CGMY model with Y close to 2, or with a long maturity, the error decay rate with respect to L is not so high and we require a larger integration range. In these cases the increase in $e^b - 1$ may give rise to divergence of the call value and the put–call parity or the put–call duality should be used for robust and accurate option values. *This is further illustrated by numerical examples in Section 1.5.*

1.5 Numerical Examples

In this section we will show the method's accuracy, efficiency and robustness by a series of numerical examples. The CPU used is an Intel(R) Core(TM)2 Duo CPU E6550 (2.33GHz Cache size 4MB) with an implementation in Matlab 7.7.0.

We use as reference values the Bermudan option prices obtained by the robust version of the COS method, with a very fine grid (with $N = 2^{14}$).

In the experiments, we will use the CGMY model, with test parameters $Y = 0.5$, $Y = 1.5$ and $Y = 1.98$; the remaining CGMY parameters are chosen as $[C, M, G] = [1, 5, 5]$. Other parameters include: $r = 0.1, q = 0.02, S_0 = 100, K = 110$. We set $\mathcal{M} = 10$ and maturity $T = 1$. Computational time and the absolute error in the option value are displayed in Tables 1.1 to 1.3. From these tables we see that for $Y = 0.5$ $N = 256$ is sufficient while for $Y = 1.5$ and $Y = 1.98$ it is $N = 128$. When $Y > 1$, which implies that the process has infinite activity, the error in the option price is of order 10^{-12}. From the tables we see that the methods with both the put–call parity and the put–call duality converge very well within milliseconds. The CPU time when using the put–call duality is approximately twice the time with put–call parity, because with the put–call duality we need to calculate two matrix-vector products with Hankel and Toeplitz matrices at each time step.

	N	64	128	256	512
Parity:	abs.err	2.9497e-004	1.0586e-005	8.5622e-007	1.1607e-007
	ms.	4.959	6.819	10.484	18.878
Duality:	abs.err	3.7177e-002	8.5904e-005	5.8262e-005	6.4494e-006
	ms.	8.000	12.105	19.778	35.554

Table 1.1: Absolute error and CPU time (in milliseconds) for the CGMY model, $Y = 0.5$. COS pricing with the put-call relations.

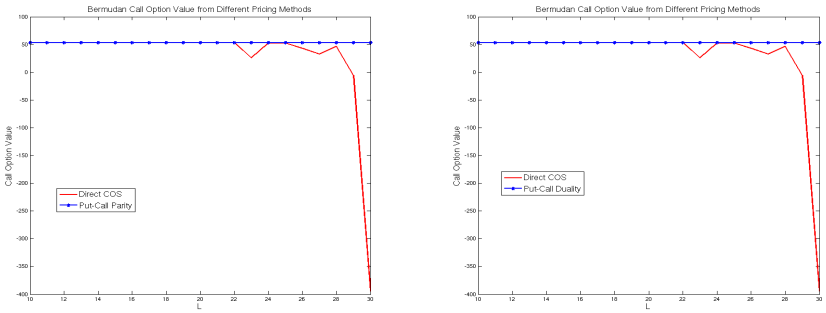
	N	32	64	128	256
Parity:	abs.err	7.7799e-003	1.8691e-005	2.2737e-012	5.6843e-014
	ms.	3.735	4.699	6.760	10.527
Duality:	abs.err	2.8937e-002	1.3074e-002	5.8769e-007	7.9581e-013
	ms.	5.839	8.009	12.078	20.016

Table 1.2: Absolute error and CPU time (in milliseconds) for the CGMY model, $Y = 1.5$. COS pricing with the put-call relations.

	N	32	64	128	256
Parity:	abs.err	4.0414e-001	3.8936e-004	1.1369e-013	$< 1e-016$
	ms.	3.690	4.831	6.664	10.577
Duality:	abs.err	1.5431e-001	3.4510e-006	1.4495e-011	6.9207e-012
	ms.	7.927	12.034	19.643	35.400

Table 1.3: Absolute error and CPU time (in milliseconds) for the CGMY model, $Y = 1.98$. COS pricing with the put-call relations.

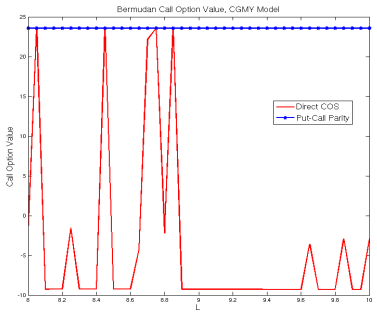
Figure 1.3 compares Bermudan call option values under the GBM model, with $S_0 = 100, K = 80, r = 0.1, \sigma = 0.2$, obtained directly by the COS method with the values obtained via the put–call parity or the put–call duality, and with reference values. The dividend rate is $q = 0.02$, and the reference value is $53.355758\ldots$. For very large values, $L > 20$, the option values obtained by the COS method (without the put-call relations) differ dramatically from the reference values. Pricing is robust, with respect to the size of the integration interval when the put–call parity and the put–call duality are applied, as then accurate call prices are obtained for any value of L, see Figure 1.3.



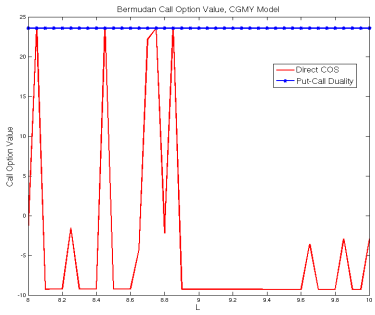
(a) Put–Call Parity (b) Put–Call Duality

Figure 1.3: Bermudan call option values with varying L-values, GBM model, $r = 0.1, q = 0.02, \sigma = 0.2, T = 10, \mathcal{M} = 50$, $L \in [10, 30]$.

We again consider the CGMY model, for which Figure 1.4 shows Bermudan pricing results for $Y = 0.5$ and $r = 0.1, q = 0.02$. The other parameter values are as in the previous experiments. The reference value is $23.574835\ldots$. Compared to Figure 1.3, the error in Bermudan call option values under this CGMY parameter set is significantly larger than under the GBM model. However, combined with the put–call parity or the put–call duality, the option prices converge in a robust way to the reference value, for all L.



(a) Put–Call Parity

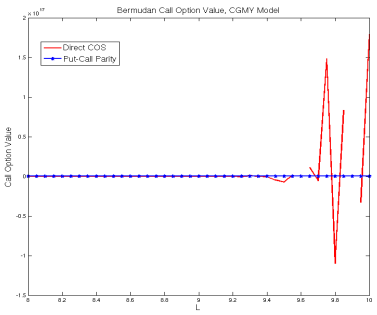


(b) Put–Call Duality

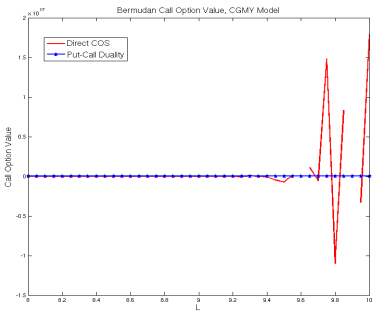
Figure 1.4: Bermudan call option values with varying L-values, CGMY model with $q = 0.02, Y = 0.5, \mathcal{M} = 24$, $L \in [8, 10]$.

With parameter Y close to 2 in CGMY, the Bermudan call prices, computed directly by the COS method are subject to cancellation errors even for small sizes of the computational domain and small maturity dates, as shown in Figure 1.5. Here the reference value for the Bermudan call is $99.053582\ldots$. With T and $\mathcal{M}$ increasing, the error also increases. The COS method with the put–call parity or the put–call duality remains however robust also for these parameter values.

Comparing Figures 1.5 and 1.4, we see that as Y increases, which implies a fatter tail in the probability density function of the underlying, the error in the call price obtained by the COS method with respect to large computational domain sizes increases drastically.



(a) Put–Call Parity



(b) Put–Call Duality

Figure 1.5: Bermudan call option values with varying L-values, CGMY model with $q = 0.05, Y = 1.98, \mathcal{M} = 10$, $L \in [8, 10]$.

1.5.1 American Options

Finally, we price an American call option by the 4-point Richardson extrapolation using (1.32) with Bermudan options. We use the CGMY model with $Y = 1.5$ and 1.98, and $q = 0.05$, and compare American calls for which the Bermudan calls in the extrapolation are priced directly by the COS method with those computed using the put–call parity or the put–call duality. In the COS method we used $N = 1024$ in the case with $Y = 1.98, \mathcal{M} = 32$ ($\mathcal{M}$ as in (1.32)); in all other cases, $N = 512$ is sufficient for convergence. The number of Newton iterations is set to 5 (as in [35]).

The accuracy of the American prices depends on parameter $\mathcal{M}$ in the extrapolation formula (1.32). The results obtained are in Tables 1.4 and 1.5 with CPU time in seconds. In these tables the American option prices are accurate and robust when the put–call parity or the put–call duality was used in the COS pricing procedure.

$\mathcal{M}$	Put-Call Parity		Put-Call Duality		direct COS method	
	option value	time (sec.)	option value	time (sec.)	option value	time (sec.)
8	44.0934	0.243	44.0934	0.501	58.3396	0.238
16	44.0933	0.489	44.0933	1.002	56.6221	0.428
32	44.0936	0.998	44.0934	2.014	-5.3915e+02	0.840

Table 1.4: American call option values and CPU time (in seconds) by Richardson extrapolation, under the CGMY model with, $Y = 1.5, q = 0.05$, $\mathcal{M}$ in Eq. (1.32).

$\mathcal{M}$	Put-Call Parity		Put-Call Duality		direct COS method	
	option value	time (sec.)	option value	time (sec.)	option value	time (sec.)
8	99.1739	0.244	99.1739	0.497	-2.2964e+48	0.221
16	99.1739	0.520	99.1739	0.987	5.0141e+46	0.460
32	99.1738	0.976	99.1738	3.761	2.1427e+53	0.820

Table 1.5: American call option values and CPU times (in seconds) by Richardson extrapolation, under the CGMY model with $Y = 1.98, q = 0.05$, $\mathcal{M}$ in Eq. (1.32).

1.6 Conclusions and Discussion

In this chapter, we have discussed the COS option pricing method, based on Fourier cosine expansions, for European options and Bermudan options.

The method can be used whenever the characteristic function of the underlying price process is available. It is especially efficient for exponential Lévy processes.

The COS formula for European options from [34] can be used for pricing Bermudan options, if the series coefficients of the option values at the first early-exercise date are known. These coefficients can be recursively recovered from those of the payoff function. The computational complexity is $O((\mathcal{M}-1)N\log_2 N)$, for Bermudan options under Lévy processes with $\mathcal{M}$ exercise dates. The COS method exhibits an exponential convergence in N for density functions in $\mathrm{C}^\infty[a,b]$ and an impressive computational speed. With a limited number, N, of Fourier cosine coefficients, it produces highly accurate results. We have also presented error analysis for this method, showing that convergence for put options is easily obtained, whereas the unbounded payoff function for calls may hamper the robust convergence. The convergence of directly applying the COS method to call options depends of the choice of the integration range. Robust pricing, insensitive of the choice of the size of the integration range, is achieved for call options, when the put-call parity or the put–call duality relation is applied. The use of these relations for call options with early exercise features has been explained in detail. It results in a robust pricing technique for Bermudan and American options, independent of the size of the computational domain.

Pricing American options can be done by a Richardson extrapolation method on Bermudan options with a varying number of exercise dates.

CHAPTER 2

Acceleration of the COS Option Pricing Technique on Graphics Processing Units

This chapter contains essentially the content of paper [69].

2.1 Introduction

In this chapter we deal with the highly efficient pricing of options on stocks or other assets. It is particularly necessary when the asset price models are calibrated to real market data. Option values, with many different parameter values for the underlying asset price process, are then computed thousands of times in order to fit the mathematical model. In this chapter, we show that it may make sense to perform this task on a Graphics Processing Unit-based computer.

Integration-based option pricing with the help of the Fast Fourier Transform is common practice for calibration (see, for example [16], [21], [26], [43]).

The COS method, as presented in Subsection 1.2, is applicable if the characteristic function of the stochastic asset price process (i.e. the Fourier transform of the conditional density function) is available. This is certainly the case for state-of-the-art asset price models, like the Lévy jump processes and the Heston stochastic volatility process, which we use in the current chapter. However, also more involved *hybrid* stochastic processes, for example for equity and interest rates can be considered, as long as we can get to a characteristic function. For such asset models with stochastic volatility *and* stochastic interest rate, like the Heston-Hull-White or the Heston–Gaussian

two-factor model, the analytic characteristic function is typically not available. However, after some appropriate reformulations of the SDE system (see, for example [39]) the coefficients of the characteristic function can be found as the solution of a Riccati system of ordinary differential equations (ODEs), as described in [31]. These ODE systems can be solved numerically by means of an explicit Runge–Kutta method or by other ODE solvers. We will show that this task can also be performed efficiently on a GPU using CUDA [75].

In practice, the option values obtained from a mathematical model should be consistent with market option prices. Usually, options with many different strike prices are needed for calibration. In the COS method, European option prices *for a vector of strikes* can be computed in one computation, which accelerates the calibration procedure significantly.

To further accelerate the calibration procedure, two approaches directly come into mind. The easiest is to purchase a faster CPU! As an example, Table 2.1 compares error convergence and CPU times between the CPU used in [34] (CPU 1: Intel(R) Pentium(R) 4 CPU, 2.80GHz with cache size 1MB) and a faster CPU (CPU 2: Intel(R) Core(TM)2 Duo CPU, E6550 @ 2.33GHz Cache size 4MB) for European calls under a Geometric Brownian Motion asset process. Time is in milliseconds[1].

N	16	32	64
ms(cpu1)	0.337	0.388	0.506
ms(cpu2)	0.1032	0.1503	0.2270
max.abs.error	0.0059	9.1396e-08	1.4211e-14

Table 2.1: Comparison of cpu times between different CPUs.

The faster CPU gives a satisfactory acceleration, but one needs to wait (sometimes up to two years) for an acceleration by a factor two.

Another possibility to accelerate the pricing engine is to run the program, or parts of it, on the popular Graphics Processing Unit, which supports parallel computation. Executing a code on a GPU is worthwhile if:

1. A program can be divided into several independent parts;
2. A program does not contain many sequential parts;
3. A program does not require much memory transfer from host to device or vice versa.

We will study the suitability of the GPU for the important task of calibration, which is traditionally done with European options. In that respect, it

[1] 1 millisecond=10^{-3} second

is a challenge to accelerate the pricing of European options. A well-known insight in Numerical Mathematics and Scientific Computing is that it is often easy to accelerate a sub-optimal solver on novel hardware, but that it is hard to improve the computing times for optimal solvers.

In this chapter, we will demonstrate a significant performance improvement on the GPU, due to parallelization, when pricing European options for several options simultaneously, i.e. with *multiple strikes*.

Earlier work in the direction of GPU acceleration of an integration-based option pricing method [61] achieved an impressive speed-up on the GPU for options with early-exercise features. By a relatively large number of space and time points the advantages of the GPU implementation were shown. As the COS method exhibits an exponential rate of convergence, a very large number of computations is not necessary to get accurate option values. We will focus on a speed-up on the GPU with not-more-than-necessary terms in the Fourier cosine expansion.

For certain modern option products it also makes sense to calibrate to barrier options, or other, liquid financial products. The pricing of barrier options with the COS method is closely connected to the example of pricing Bermudan options in the present chapter, see [35].

The outline of the current chapter is as follows. In Section 2.2, after a short introduction on option pricing with multiple strike prices and the Heston model, we will explain why the COS method can be well implemented on the GPU. For European options different ways of implementation are described in Section 2.3. Section 2.4 presents the speed-up for multi-strike European options on the GPU. The acceleration of an explicit Runge–Kutta method for numerically solving systems of Riccati ODEs to approximate a characteristic function (if it is not available in closed form) is presented in Section 2.4.3. Section 2.5 gives pricing results for Bermudan options. Here the influence of the number of terms in the Fourier cosine expansion as well as the number of early exercise dates on the GPU speedup is discussed.

The GPU we work with is an NVIDIA GeForce 9800 GX2, which has two graphics processing units (GPUs) and 1 GB of memory (512 MB for each GPU); the CPU on the same computer, needed for data transfer, is an AMD Athlon(tm)64 X2 Dual Core Processor 4600+ (cache size 512 KB, 2412.364MHz).

The results obtained are compared with timings on a CPU from an Intel(R) Core(TM)2 Duo CPU E6550 (@ 2.33GHz Cache size 4MB).

2.2 COS Pricing Method and Advantage of the GPU

In Subsection 1.2 an introduction of the COS pricing method for European options and Bermudan options was presented. In this section we discuss the

COS method for options focusing on pricing multiple strikes simultaneously and explain why a GPU is advantageous when pricing an option.

2.2.1 Pricing of European Options with Multi–Strike Features

With $X_t = \log(S_t/K)$, the solution for Equation (1.15) can be written as

$$V_k = U_k K, \tag{2.1}$$

where

$$U_k = \begin{cases} \frac{2}{b-a}(\chi_k(0,b) - \psi_k(0,b)), & \text{for a call,} \\ \frac{2}{b-a}(\psi(a,0) - \chi(a,0)), & \text{for a put,} \end{cases} \tag{2.2}$$

with

$$\chi_k(x_1,x_2) := \int_{x_1}^{x_2} e^x \cos\left(k\pi\frac{x-a}{b-a}\right) dx, \tag{2.3}$$

$$\psi_k(x_1,x_2) := \int_{x_1}^{x_2} \cos\left(k\pi\frac{x-a}{b-a}\right) dx. \tag{2.4}$$

Now, the pricing formula (1.14) reads:

$$v(x,t_0) = Ke^{-r\Delta t} Re(\sum_{k=0}^{N-1}{}' \varphi(\frac{k\pi}{b-a}; x, \Delta t) \cdot e^{-ik\pi\frac{a}{b-a}} U_k). \tag{2.5}$$

Let's assume that we deal with a vector of strikes, $\mathbf{K} = [\mathbf{K}(1), \cdots, \mathbf{K}(P)]^T$, where P is the number of strikes. In this setting, x and the integration upper and lower bounds, a and b, are functions of $\mathbf{K}$, which implies that they are also vectors with length P. For a vectorised version of the COS method, enabling an efficient computation of multi-strike options, we define the following matrices:

- Φ is a $(P \times N)$-matrix with elements

  $$\Phi(j,k{+}1) = \varphi\left(\frac{k\pi}{b(j)-a(j)}; x(j), \Delta t\right) \exp\left(-ik\pi\frac{a(j)}{b(j)-a(j)}\right) U_k(a(j),b(j)),$$

  $j = 1, \cdots, P, k = 0, \cdots, N-1.$

- Λ is a $(P \times P)$ diagonal matrix with $\Lambda(j,j) = \mathbf{K}(j),\ j = 1, \cdots, P.$

With these matrices, the formula for pricing options with multi–strike features reads:

$$v(\mathbf{x}, t_0) = e^{-r\Delta t}\ \Lambda\ Re(\Phi\ I), \tag{2.6}$$

where I is a $(N \times 1)$-vector with its first element equal to 0.5 and all the other elements equal to 1. This way option values for many strikes can be computed simultaneously.

2.2.2 Underlying Asset Processes

In this chapter we discuss two different underlying asset processes, the CGMY process, a Lévy jump process, and the Heston stochastic volatility process. For efficient use of the COS method, the characteristic function for these processes is required. A discussion on the CGMY can be found in Subsection 1.1.1. In the Heston stochastic volatility model, the underlying and the volatility are modeled by the following stochastic differential equations,

$$\begin{aligned} dx_t &= (r - \frac{1}{2}\mu_t)dt + \sqrt{\mu_t}dW_{1,t}, \\ d\mu_t &= \lambda(\bar{\mu} - \mu_t)dt + \eta\sqrt{\mu_t}dW_{2,t}, \end{aligned} \tag{2.7}$$

where x_t and μ_t denote the log–asset price process and the variance of the asset price process, respectively. Parameters $\lambda, \bar{\mu}, \eta$ represent the speed of mean–reversion, the long–term mean value of variance and the volatility of volatility parameters, respectively. Moreover, $W_{1,t}$ and $W_{2,t}$ are Brownian motions, correlated with correlation coefficient ρ.

For the log-asset price in the Heston model an analytic characteristic function can be found, which reads:

$$\begin{aligned} \varphi(u;\mu_0,\Delta t) &= \exp(iur\Delta t + \frac{\mu_0}{\eta^2}(\frac{1-e^{-D\Delta t}}{1-Ge^{-D\Delta t}})(\lambda - i\rho\eta u - D)) \cdot \\ &\quad \exp(\frac{\lambda\bar{\mu}}{\eta^2}(\Delta t(\lambda - i\rho\eta u - D) - 2\log\left(\frac{1-Ge^{-D\Delta t}}{1-G}\right))), \end{aligned}$$

with $D = \sqrt{(\lambda - i\eta\rho u)^2 + (u^2 + iu)\eta^2}$ and $G = \frac{\lambda - i\eta\rho u - D}{\lambda - i\eta\rho u + D}$.

For the value of D, we take the square root whose real part is non-negative.

2.2.3 Advantage of COS method on GPU

Different operations in the COS method are run in MATLAB and C on the CPU, and also in CUDA on the GPU. From Tables 2.2 to 2.4 we see that the GPU is significantly faster than the CPU (either Matlab or C), especially for functions like $\sin(x), \cos(x), \exp(x)$. This is due to its parallel features, where operations on each element of a vector can be done independently.

Note that in this chapter, when we mention GPU time, it is the "time of a GPU round trip", where the time for the memory transfer from the host to the device and vice versa is included.

The comparison of CPU and GPU time for the Fourier and inverse Fourier transform is shown in Table 6.

Table 2.5 shows that the GPU is significantly faster than the CPU when dealing with Fourier and inverse Fourier transform in C, but, compared to

size of x	CPU(Matlab)	CPU(C)	GPU
65536	0.004811	0.005558	0.000672
262144	0.019119	0.022127	0.002650
1048576	0.075415	0.088729	0.008016

Table 2.2: Comparison of CPU time and GPU time for $exp(x)$.

size of x	CPU(Matlab)	CPU(C)	GPU
65536	0.008033	0.003585	0.000739
262144	0.032880	0.014503	0.003250
1048576	0.132002	0.057822	0.008042

Table 2.3: Comparison of CPU time and GPU time for $sin(x)$.

size of x	CPU(Matlab)	CPU(C)	GPU
65536	0.008048	0.003636	0.000729
262144	0.032933	0.014597	0.003252
1048576	0.132410	0.059626	0.008036

Table 2.4: Comparison of CPU time and GPU time for $cos(x)$.

size of x	CPU(Matlab)	CPU(C)	GPU
1024	0.000087	0.000440	0.000182
4096	0.000311	0.000764	0.000412
16384	0.001253	0.002087	0.000744

Table 2.5: Comparison of CPU time and GPU time for $ifft(fft(x))$.

MATLAB, it is only faster when the size of the vector is large. Moreover, due to an unscaled inverse Fourier transform on the GPU, the result of $ifft(fft(x))$ is $length(x) \cdot x$ and we need an additional operation in CUDA compared to MATLAB to scale the result of $ifft(fft(x))$ to x. However, because the GPU is more advantageous than the CPU in most of the time consuming parts, it is expected to outperform the CPU for the COS algorithm.

2.3 European Options

A European option can be viewed as a special case of a Bermudan option with only one possible exercise date (the expiry time). For European options, the Fourier and inverse Fourier transform operations are not needed.

2.3.1 Different Ways of GPU Implementation

In this section, we discuss different ways of implementation on a GPU. Consider a simple case where we need to price one vanilla option.

From (1.14) the COS algorithm can be decomposed into two steps, i.e., computations on each element of a vector, i.e. $Re(\exp(-ik\pi\frac{a}{b-a})\varphi(\frac{k\pi}{b-a};x,T-t_0)V_k)$, which can be parallelized; and the summation of vector elements.

We consider three ways of GPU implementation:

1. Directly run the whole code on the GPU, referred to as GPU1;

2. All operations related to each vector element are parallelized on the GPU, whereas the summation is performed on the CPU. This hybrid GPU/CPU way of implementation is referred to as GPU2;

3. When summing up the elements of a vector, we can split the vector in two vectors, each of size $N/2$, and sum up these two on the GPU. The procedure is repeated until $N = 1$. The summation of pairs of elements can be parallelized on the GPU this way. The number of operations for the summation can be reduced from N to $\log_2(N)$, referred to as GPU3.

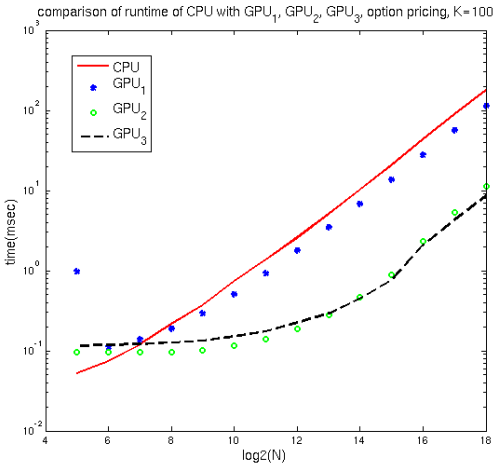


Figure 2.1: Comparison of different GPU implementations.

Figure 2.1 presents a comparison of the time consumed by the above mentioned three ways of GPU implementation and also the CPU time. Clearly, GPU1 is faster than the CPU only when N, the number of terms in the Fourier cosine expansion, is large, whereas the implementations GPU2 and GPU3 are faster than either the CPU implementation or GPU1. Moreover, as N increases, the speedup of GPU2 and GPU3 also increases. GPU3 is

slightly slower than GPU2 for small N, but when N is very large, GPU3 beats GPU2.

Unlike the CPU or GPU1, the time for GPU2 and GPU3 does not increase much as N increases, until $N \approx 2^{16}$.

For Bermudan options, implementation GPU3 is preferred, since the complete code then runs on the GPU and data transfer can be reduced. With GPU2, we would need to transfer data at each time step, which consumes time.

In this chapter we will use implementation GPU3 for all numerical examples to follow.

2.3.2 Numerical Example

We take as an example the CGMY model with $Y = 1.5$. The other parameters are chosen as $S_0 = 100, r = 0.1, K = 80, C = 1, M = 5, G = 5$. Table 2.6 compares time and accuracy of the CPU and the GPU results, with time measured in milliseconds. Table 2.6 shows that with $N = 1024$

N	CPU(time)	GPU(time)	CPU(value)	GPU(value)
256	0.193...	0.182	27.974744	27.974733
1024	0.691...	0.433	27.974744	27.974733

Table 2.6: Comparison of time and precision for CPU and GPU implementation of the COS method for a single European option.

the GPU implementation is 1.5 times faster than running the code on the CPU. However, it is not necessary to take such a large value of N in the COS method in practice, as with $N = 256$ the option values already differ less than one basis point. Therefore, in the present setting (COS method, small values of N, one option price) the use of a GPU is not really advantageous. However, when dealing with multiple strikes, presented in Section 2.4, many more computations are needed, so that the GPU performance will be more profound.

2.4 Multiple Strike Option Pricing

In this section we focus on pricing European options, but now with multiple strikes, on the GPU. The parameters used, next to $S_0 = 100$, in the CGMY process and for the Heston process are:

$$\begin{aligned} &\text{CGMY}: && r = 0.1, C = 1, G = 5, M = 5, Y = 1.5, T = 1; \\ &\text{Heston}: && r = 0.040, \lambda = 1.577, \eta = 0.575, \mu_0 = 0.018, \rho = -0.57, T = 10. \end{aligned}$$

We price European call options with vectors of strikes, as shown in Table 2.7.

Number of strikes	Strike values
3 strikes	$\mathbf{K} = 80, 100, 120$
5 strikes	$\mathbf{K} = 80, 90, 100, 110, 120$
9 strikes	$\mathbf{K} = 80, 85, \cdots, 115, 120$
13 strikes	$\mathbf{K} = 70, 75, \cdots, 125, 130$
17 strikes	$\mathbf{K} = 60, 65, \cdots, 135, 140$
21 strikes	$\mathbf{K} = 50, 55, \cdots, 145, 150$

Table 2.7: Number of strikes used in the numerical examples.

To efficiently implement (2.6) on a GPU, we first divide the P–axis and the N–axis in different blocks and threads, as shown in Figure 2.2.

```
int i=blockIdx.x*blockDim.x+threadIdx.x;
int j=blockIdx.y*blockDim.y+threadIdx.y;
```

Figure 2.2: Blocks and threads.

Then, the elements of the $(P \times N)$-matrix can be calculated *simultaneously*, as illustrated in Figure 2.3.

When performing the summation on each row of the matrix, as the final step in (2.6), we divide the $(P \times N)$-matrix into smaller sub-matrices. For instance, with $N = 128$, and 21 strikes, the corresponding 21×128-matrix can be subdivided into fifty–six matrices of size 3×16. The elements of these smaller sub-matrices are copied to shared memory, as shown in Figure 2.4.

Here $aBegin$ is the first location of As, i.e. the blocks with shared memory, and tx, ty are the thread indices of As. Then, as we run the program, data transfer only happens within the shared memory, and not in the global memory, which saves us a lot of GPU time.

2.4.1 Convergence and Precision

Tables 2.8 and 2.9 present the convergence behaviour and the precision of option values with 5 strikes and 21 strikes, respectively, for the two underlying processes. Time is again measured in milliseconds. Option values obtained with $N = 2^{16}$, and in double precision, are taken as the reference values. We calculate the maximum absolute error, for varying values of N, over the strike vectors.

Both the GPU and CPU timing results are shown to be extremely fast,

```
v[i*Ndim+j]=exp(-r*T)*K[i]*real(cf[i*Ndim+j]
           *exp((x[i]-a[i])*omega[i*Ndim+j])*U[i*Ndim+j]);
```

Figure 2.3: Parallelization of the COS method for options with multiple strike values.

```
__shared__ float As[BLOCK_SIZE_K][BLOCK_SIZE_N];
As[tx][ty]=A[aBegin+N*tx+ty];
__syncthreads();
```

Figure 2.4: Data transfer from global memory to shared memory.

as we need only $N = 64$ for the CGMY process and $N = 128$ for Heston's model to obtain converged option values on the GPU and the CPU. However, the execution time on the GPU is significantly smaller than on the CPU. As we are in the milliseconds range, one might question the relevance of this gain in speed. However, within a calibration setting option prices have to be computed very many times, which immediately turns a small gain into a significant profit. The advantage of the use of the GPU becomes more pronounced when the value of N and the number of strikes, $\mathbf{K}(P)$, increase, since then more arithmetic operations are required. As shown in Tables 2.8 and 2.9, the acceleration on the GPU for Heston's model increases for 5

strikes from 12 to 21, as N increases from 128 to 256. For 21 strikes, the speedup on the GPU is a factor 37 for $N = 128$ and 47 for $N = 256$. Since more computations are necessary to evaluate the characteristic function of the Heston model, the speedup on the GPU for the Heston model is higher than for the CGMY model. However, due to the fact that the GPUs used in this test give computed values in single precision, round-off errors can build up during the computation of the characteristic function on the GPU. A larger maximum absolute error for the Heston model is therefore observed in the tables.

CGMY model				
	N	32	64	128
MATLAB	ms	0.413230	0.745590	1.388770
	max.abs.err	1.3409e-05	$< 10^{-14}$	$< 10^{-14}$
GPU	ms	0.141144	0.143051	0.152826
	max.abs.err	0.000027	0.000034	0.000034
Heston model				
	N	64	128	256
MATLAB	ms	1.206600	1.958680	3.873950
	max.abs.err	4.2839e-04	2.2218e-08	$< 10^{-14}$
GPU	ms	0.154972	0.159979	0.182867
	max.abs.err	0.000534	0.000104	0.000104

Table 2.8: Convergence and maximum absolute error when pricing a vector of 5 strikes.

CGMY model				
	N	32	64	128
MATLAB	ms	1.335130	2.690250	5.340340
	max.abs.err	1.3409e-05	$< 10^{-14}$	$< 10^{-14}$
GPU	ms	0.154018	0.169992	0.200987
	max.abs.err	0.000053	0.000053	0.000053
Heston model				
	N	64	128	256
MATLAB	ms	3.850890	7.703350	15.556240
	max.abs.err	6.0991e-04	2.7601e-08	$< 10^{-14}$
GPU	ms	0.177860	0.209093	0.333786
	max.abs.err	0.000534	0.000144	0.000144

Table 2.9: Convergence and maximum absolute error when pricing a vector of 21 strikes.

Figure 2.5 presents the speed-up obtained on the GPU for different num-

bers of strikes, with $N = 128, 512, 2048$. It displays a speedup of 30 to 40 for 21 strikes on the GPU with $N = 128$ and a speedup of 60 to 70 for 13 and 17 strikes with $N = 512$. The GPU appears to be a very satisfactory architecture for pricing options at multiple strikes.

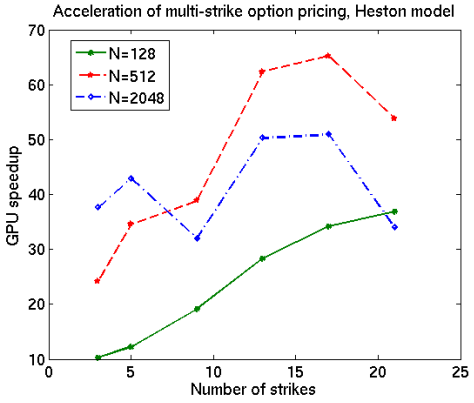


Figure 2.5: GPU speed-up for Heston model with different numbers of strikes, $N = 128, 512, 2048$.

Figure 2.5 also displays the influence of data transfer on the GPU time. The data transfer time is the time to transfer the input (in this case the value of the strikes) from the CPU to the GPU at the beginning plus the time to transfer the output (the option values) back to the CPU after the computations have finished.

GPU time is the sum of the data transfer and the computation time. For larger input sizes the GPU is advantageous regarding the computations, but the data transfer time increases. The influence of data transfer time is more pronounced when the number of terms in the cosine expansion, N, reaches 1000.

For a closer look, we refer to Table 2.10, from which we see that due to the parallelization the GPU computation time hardly changes when N increases from 2048 to 8192. On the other hand, data transfer time is more significant with increasing N, which gives a smaller acceleration for large N and many strike values, as also shown in Figure 2.5.

2.4.2 Option Pricing with Short Maturity Times

In certain practical applications, profit opportunities may arise from the calibration of Levy processes for European option contracts with a very short time to maturity.

N	2048	4096	8192
Data transfer time excluded	0.086069	0.087023	0.084877
Data transfer time included	0.493050	1.350164	4.626989

Table 2.10: Influence of data transfer on the computational time on a GPU.

In this section, we report on the recent implementation of the COS method on a different GPU architecture, a Tesla C1060, which also supports double precision. The true advantages of this GPU for the COS method implementation are for short maturity options, since we then need a larger value of N (many terms in a cosine expansion) for convergence with the COS method, thus more computations need to be performed.

In Table 2.11, the computational time in milliseconds is recorded on this GPU and on the CPU for options under the CGMY model. A vector of strikes, $\mathbf{K} = 90, 92, \cdots, 110$, is priced simultaneously. The maximum absolute error [2] over all strikes for short maturities is also presented. The model parameters used are $C = 0.5, G = 10, M = 30, Y = 0.5$ (Set 1) and $C = 0.25, G = 5, M = 50, Y = 1.5$ (Set 2).

Set 1 with $T = 0.1$ (one month)				
	N	512	1024	2048
MATLAB	ms	5.9980	13.4580	27.9570
	max.abs.err	2.4248e-04	4.0405e-06	1.1445e-07
GPU	ms	0.188828	0.279903	0.492811
	max.abs.err	0.000255	0.000025	0.000025
Set 2 with $T = 0.0001$ (one hour)				
	N	1024	2048	4096
MATLAB	ms	9.4810	26.9409	60.3240
	max.abs.err	1.6419e-04	7.5000e-08	4.8746e-16
GPU	ms	0.280857	0.494957	1.353979
	max.abs.err	0.000019	0.000010	0.000010

Table 2.11: Convergence and maximum absolute error when pricing a vector of 11 strikes.

For the experiments in Tables 2.8 and 2.9 we reached convergence with $N = 128$ and we got speedups of 9 and 26 on the GPU for 5 and 21 strikes, respectively. In Table 2.11, $N = 1024$ and $N = 2048$ were used in the two experiments, respectively, for convergence at short maturities, and the Tesla GPU then achieves speedups of 48 and 54.

[2] Reference values have been obtained with $N = 2^{20}$.

2.4.3 Riccati ODEs and Characteristic Function

In this subsection, we will implement an explicit Runge-Kutta ODE solver to determine the characteristic function for Heston's model. Riccati ODEs arise in the determination of a characteristic function. Sometimes, as in the case of the Heston model, the characteristic function is known analytically. In the cases for which we cannot find an analytic expression, we may resort to the numerical solution of the Riccati ODEs. A numerical procedure is for example necessary for systems of stochastic differential equations, that include stochastic interest rate *and* stochastic volatility. In the numerical procedure operations like taking the square root of a complex number are not needed, in contrast to the evaluation of the analytic solution. For the numerical ODE solver the influence of single precision arithmetic is therefore less pronounced, and we even obtain a higher accuracy than with the analytic characteristic function.

From [31] we know that the characteristic function for the Heston model (2.7) is of the following form:

$$\varphi(u; x_0, \mu_0, t) = e^{A(u,t)+B_\mu(u,t)\mu_0+B_x(u,t)x_0}, \tag{2.8}$$

with the coefficients $A(u,t)$, $B_x(u,t)$ and $B_\mu(u,t)$ given by the following Riccati ODEs:

$$\begin{cases} \frac{\partial}{\partial t}B_x(u,t) = 0, & B_x(u,0) = iu, \\ \frac{\partial}{\partial t}B_\mu(u,t) = 0.5\eta^2\mu_t^2 - (\lambda - i\rho\eta u)\mu_t - 0.5iu - 0.5u^2, & B_\mu(u,0) = 0, \\ \frac{\partial}{\partial t}A(u,t) = \lambda\bar{\mu}\mu_t + i(r-q)u, & A(u,0) = 0. \end{cases} \tag{2.9}$$

It is easy to see from (2.9) that $B_x(u,t) = iu$; $A(u,t)$ and $B_\mu(u,t)$ are here solved numerically by the explicit fourth order Runge–Kutta method (RK4), and inserted in the general characteristic function (2.8), based on which we then employ the COS method.

Tables 2.12 shows the corresponding timing results on the GPU and the CPU for 5 and 21 strikes, respectively. Compared to the results in Tables 2.8 and 2.9, a higher speed-up is achieved on the GPU. For 5 strikes, with $N = 128$ and $N = 256$, the GPU timings are around 50 times faster than the CPU results. For 21 strikes the acceleration on the GPU is a factor of 100. Note that for 21 strikes, due to the increased data transfer, the speedup reduces as N increases, but due to the exponential convergence rate, $N = 128$ is sufficient for convergence.

Of course, option pricing with an analytic characteristic function is still fastest, but the numerical solution of Riccati ODEs can be performed highly efficiently on these units.

5 strikes				
	N	64	128	256
MATLAB	ms	37.9491	50.5196	81.1083
	max.abs.err	4.2848e-04	8.4949e-08	1.0650e-07
GPU	ms	1.091957	1.121998	1.253843
	max.abs.err	0.000443	0.000013	0.000013
21 strikes				
MATLAB	ms	84.9870	145.0005	268.2299
	max.abs.err	6.0979e-04	1.5607e-07	1.2847e-07
GPU	ms	1.228094	1.402855	2.689838
	max.abs.err	0.000611	0.000037	0.000037

Table 2.12: Convergence and maximum absolute error for 5 and 21 strikes, Heston model's characteristic function obtained by RK4.

2.5 Bermudan Options

In this section we consider the pricing of Bermudan options with a discrete number of early exercise points. Also here we do not focus on large values of N or on many time points, $\mathcal{M}$. With $N = 160$ in the COS method, we obtain the price of a Bermudan option with an error smaller than 10^{-9}. Moreover, Bermudan options with up to 64 time points ($\mathcal{M} = 64$) are typically sufficient to get a very satisfactory approximation of the value of an American options (that can be exercised at any time before expiry) by a repeated Richardson extrapolation, see [50].

We will show in this section that by exploiting the parallelism of a GPU the COS algorithm is somewhat faster than on the CPU with a realistic number of terms, N. The reduction of the total execution time is however not as impressive as in the previous sections.

We use as a numerical example a Bermudan put option under the CGMY process. The model parameters are $S_0 = 100, K = 80, \mathcal{M} = 10, C = 1, M = 5, G = 5, Y = 1.5$. Tables 2.13 and 2.14 compare time and accuracy on the GPU and CPU.

With N relatively small, the Bermudan option price converges well and the resulting error is small. However, as mentioned before, as N gets larger, which is probably necessary for other types of options, the advantage of using the GPU will become more pronounced.

We increase the number of exercise dates, $\mathcal{M}$, to 20, 40 and 80 and compare the GPU and CPU execution times. The model parameters are as before. We use $N = 512$. The timing results are listed in Table 2.15.

With $N = 512$ the GPU is twice as fast as the CPU for different numbers of exercise dates. The number of exercise dates does not influence the

N	CPU(time)	GPU(time)
256	13.15...	11.02
512	24.69...	13.69
1024	46.65...	27.34

Table 2.13: Comparison between CPU and GPU execution times for the CGMY model, for a Bermudan option; different numbers of terms in the Fourier cosine expansion.

N	CPU(value)	GPU(value)
256	28.829781987399432	28.829739
512	28.829781987399425	28.829721
1024	28.829781987399404	28.829756

Table 2.14: Precision on the GPU, Bermudan put option, for different numbers of terms in the Fourier cosine expansion.

$\mathcal{M}$	CPU(time)	GPU(time)
20	51.04...	26.09
40	104.00...	50.88
80	210.20...	100.54

Table 2.15: Comparison of CPU and GPU times for the CGMY model, Bermudan option with a different numbers of exercise dates.

speedup of the GPU. This is because the algorithm can not be parallelized in time, as we use values at t_{m+1} to calculate values at t_m in the backward recursion procedure. This results in a recursion of $m = \mathcal{M} - 1, \cdots, 1$ in the CUDA implementation.

2.6 Conclusions

The COS method is a highly efficient pricing method for European and early–exercise options. It is a challenge to implement an efficient method, which requires a small number of terms in the Fourier cosine expansion and a small number of exercise dates to approximate the value of an American option, efficiently on a GPU.

We implemented the COS algorithm on the GPU using CUDA. The GPU timings and option values were compared to those obtained on the CPU. To exploit the advantages of a GPU, we split a vector (or matrix) and perform summation in parallel, which enhanced the GPU performance.

A highly satisfactory performance on the GPU was observed, especially in the case of multiple strike European option computations. Then, we find GPU speedups ranging from 10 to 100, depending on the form of the characteristic function and on the number of strikes that are computed simultaneously.

Although computation on a GPU to date still exhibits some disadvantages, such as a somewhat time-consuming memory transfer, it is a highly promising architecture for option pricing.

For involved SDE asset price models, the characteristic function must be determined numerically by a Riccati ODE solver, as an analytic expression is not available. Based on the results in this chapter, the numerical ODE treatment by means of a fourth order Runge-Kutta method is also highly efficient on a GPU.

A GPU-based architecture may serve very well as a calibration engine in option pricing.

A comparison between a GPU and a single CPU may not be completely fair. Instead we should have compared with a multi-core CPU machine (if it would have been available to us). This would have improved the CPU performance.

However, it should be noted that with a multi-core CPU, also the data transfer time between the CPU and the GPU will improve. Since data transfer time dominates the total execution time so far, we also expect an increased speedup of the total GPU time compared to the times reported in this chapter.

CHAPTER 3

Efficient Pricing of Commodity Options with Early–Exercise under the Ornstein–Uhlenbeck Process

This chapter contains essentially the contents of paper [71].

3.1 Introduction

Computational Finance is one of those mathematical areas in which stochastic modeling and numerical mathematics are closely intertwined. Efficient numerical pricing methods are for example required for financial derivatives, on stocks, interest rates, credit or commodities, all governed by stochastic differential equations. In this chapter we focus on a numerical pricing technique for commodity derivatives that can be exercised before the expiration date of the contract.

Movements in the commodity markets expose participants to different types of risks. An obvious way for market players to control their exposure to price and volume fluctuations is by buying or selling derivatives written on the underlying products. Bermudan options and also swing options, which allow one to buy or sell extra quantities of a commodity, are commonly sold derivatives with early-exercise features.

Significant contributions have been made in modeling commodity processes by Schwartz and collaborators in [52, 58, 59, 60], where the authors used a model of Ornstein-Uhlenbeck type [63], which accounts for the mean reversion of prices, combined with a deterministic seasonality component. Often these processes are combined with independent jump components

(this extension of the models studied here is straightforward). In this chapter we deal with the COS method [34, 35] for pricing early-exercise options under the stochastic processes for commodities. In [35] it was shown that the COS method can price the early-exercise and barrier options with exponential convergence under various Lévy jump models. The computational complexity for pricing a Bermudan option with $\mathcal{M}$ exercise dates was $O((\mathcal{M}-1)N\log_2(N))$, where N denotes the number of terms in the Fourier-cosine expansion.

In the present chapter we show that this complexity cannot be easily achieved in the case of mean reverting processes of Ornstein-Uhlenbeck (OU) type. We therefore introduce an *approximation of the original characteristic function*, so that the COS pricing algorithm remains highly efficient for early-exercise commodity options under Ornstein-Uhlenbeck processes, but, at the same time, the error in the option prices should be controlled by means of error analysis.

This chapter is organized as follows. Details of the OU processes and computational complexity are presented in Section 3.2. In Section 3.3 the approximate OU model is introduced. It is followed by a detailed error analysis in Section 3.4; In Section 3.5 numerical results are presented. Finally conclusions are summarized in Section 3.6.

3.2 Problem Definition

3.2.1 The Ornstein-Uhlenbeck Process

Stochastic processes for commodities are characterized by the properties of mean reversion and seasonality. If for any reason the price of a certain commodity falls significantly due to overproduction, then market participants expect the price to rise eventually as producers decrease their supply. Moreover, incorporation of seasonality in the model is necessary since energy consumption (the use of electricity, for example) differs in different seasons of the year.

We first look at the OU process without seasonality. The logarithm of the commodity price, $X_t = \log S_t$, is modeled by an Ornstein-Uhlenbeck mean reverting process. We define this process here under the so-called equivalent martingale measure $\mathbb{Q}$

$$\mathrm{d}X_t = \kappa(\overline{x}^{\mathbb{Q}} - X_t)\mathrm{d}t + \sigma \mathrm{d}W_t^{\mathbb{Q}}, \text{with } X_0 = x_0, \tag{3.1}$$

with a Brownian motion, $W_t^{\mathbb{Q}}$, under measure $\mathbb{Q}$, and the parameters κ and σ represent the speed of mean reversion and volatility of the underlying process, respectively. Under this measure, the parameter $\overline{x}^{\mathbb{Q}} := \overline{x} - \lambda$ with λ the market price of risk, and $\overline{x}$ the long-term mean value of the underlying process. In the derivations to follow, we will just use $\overline{x}$ to denote $\overline{x}^{\mathbb{Q}}$.

X_t is normally distributed, i.e.: $X_t \sim \mathcal{N}(\mathbb{E}(X_t), \mathbb{V}\text{ar}(X_t))$, with:

$$\begin{aligned} \mathbb{E}(X_t|\mathcal{F}_0) &= x_0 \mathrm{e}^{-\kappa t} + \overline{x}\left(1 - \mathrm{e}^{-\kappa t}\right), \\ \mathbb{V}\text{ar}(X_t|\mathcal{F}_0) &= \frac{\sigma^2}{2\kappa}\left(1 - \mathrm{e}^{-2\kappa t}\right). \end{aligned}$$

Modeling energy prices by mean reversion is well supported by empirical studies of the price behavior, as described in [7]. General diffusion models that incorporate mean reversion go a long way in capturing the nature of energy prices; notably their tendency to randomly oscillate away from, and over time back towards, a price level determined by the cost of production. These models have gained a more wide-spread acceptance among market practitioners as progress is made in the techniques to estimate the mean reversion level and the mean reversion rates.

We are interested in the characteristic function, $\mathbb{E}\left(\mathrm{e}^{iuX_T}|\mathcal{F}_t\right)$, related to model (3.1). Based on [31] the characteristic function is of the form $\varphi(u; x_0, \tau) = \mathrm{e}^{x_0 B(u,\tau) + A(u,\tau)}$ where $\tau := T - t$; $A(u,\tau)$ and $B(u,\tau)$ satisfy the following set of ODEs:

$$\begin{cases} B'(u,\tau) = & -\kappa B, & B(u,0) = iu, \\ A'(u,\tau) = & \kappa\overline{x}B + \frac{1}{2}\sigma^2 B^2, & A(u,0) = 0, \end{cases}$$

and the prime $'$ denotes the derivative w.r.t. τ. For the solution we find:

$$\begin{cases} B(u,\tau) &= iu\mathrm{e}^{-\kappa\tau}, \\ A(u,\tau) &= \frac{1}{4\kappa}\left(\mathrm{e}^{-2\kappa\tau} - \mathrm{e}^{-\kappa\tau}\right)\left(u^2\sigma^2 + u\mathrm{e}^{\kappa\tau}\left(u\sigma^2 - 4i\kappa\overline{x}\right)\right). \end{cases} \tag{3.2}$$

Then the characteristic function of OU process reads:

$$\varphi(u; x_0, \tau) = e^{iux_0 e^{-\kappa\tau} + A(u,\tau)}. \tag{3.3}$$

3.2.2 Incorporation of Seasonality Component

More realistic stochastic processes modeling commodity prices include a seasonality component. As presented in [17, 52] we choose a commodity price process, S_t, written as:

$$S_t = \mathrm{e}^{\zeta(t) + Y_t} = \Xi(t)\mathrm{e}^{Y_t}, \text{ with } S_0 = \Xi(0), \tag{3.4}$$

where $\Xi(t) \equiv \mathrm{e}^{\zeta(t)}$ is a deterministic function which describes the seasonality effect and Y_t is a stochastic zero-level-mean reverting process given by:

$$\mathrm{d}Y_t = -\kappa Y_t \mathrm{d}t + \sigma \mathrm{d}W_t^y, \text{ with } Y_0 \equiv y_0 = 0,$$

with W_t^y a Brownian motion, κ corresponds to the speed of mean reversion and σ determines the volatility. By applying Itô's lemma to Equation (3.4),

and adding and subtracting $\kappa\zeta(t)$, we obtain the dynamics for S_t of the form:

$$\mathrm{d}S_t = \left(\frac{1}{2}\sigma^2 - \kappa(Y_t + \zeta(t)) + \kappa\zeta(t) + \zeta'(t)\right) S_t\mathrm{d}t + \sigma S_t \mathrm{d}W_t^y,$$

$\zeta'(t)$ being the derivative of $\zeta(t)$. This equals:

$$\mathrm{d}S_t = \left(\frac{1}{2}\sigma^2 - \kappa \log S_t + \kappa\zeta(t) + \zeta'(t)\right) S_t\mathrm{d}t + \sigma S_t \mathrm{d}W_t^y.$$

Setting $\theta(t) := \zeta(t) + \left(\frac{1}{2}\sigma^2 + \zeta'(t)\right)/\kappa$, we arrive at the following process:

$$\mathrm{d}S_t = \kappa(\theta(t) - \log S_t)S_t\mathrm{d}t + \sigma S_t \mathrm{d}W_t^y.$$

By taking the log-transform of the stock price, $X_t = \log S_t$, one recognizes the model to be a mean reverting Hull-White model [41], i.e.:

$$\mathrm{d}X_t = \kappa\left(\widetilde{\theta}(t) - X_t\right)\mathrm{d}t + \sigma \mathrm{d}W_t^y, \text{ with } X_0 \equiv x_0 = \log S_0, \tag{3.5}$$

where $\widetilde{\theta}(t) = \theta(t) - \sigma^2/2\kappa$. The choice $\widetilde{\theta}(t) = \bar{x}$ gives the same process as (3.1). This model is very similar to the model in [10] for electricity prices. The OU process, X_t in (3.5), admits the solution:

$$X_t = x_0 \mathrm{e}^{-\kappa t} + \kappa \int_0^t \mathrm{e}^{-\kappa(t-s)}\widetilde{\theta}(s)\mathrm{d}s + \sigma \int_0^t \mathrm{e}^{-\kappa(t-s)}\mathrm{d}W_t^y,$$

and is thus normally distributed, i.e., $X_t \sim \mathcal{N}(\mathbb{E}(X_t), \mathbb{V}\mathrm{ar}(X_t))$, with:

$$\begin{aligned}
\mathbb{E}(X_t|\mathcal{F}_0) &= x_0\mathrm{e}^{-\kappa t} + \kappa\int_0^t \mathrm{e}^{-\kappa(t-s)}\widetilde{\theta}(s)\mathrm{d}s, \\
\mathbb{V}\mathrm{ar}(X_t|\mathcal{F}_0) &= \frac{\sigma^2}{2\kappa}\left(1 - \mathrm{e}^{-2\kappa t}\right).
\end{aligned}$$

For process $X_t = \log S_t$, as given by Equation (3.5), we find the following ODEs for the characteristic function:

$$\begin{cases}
B'(u,\tau) = & -\kappa B & \text{with } B(u,0) = iu, \\
A'(u,\tau) = & \kappa\widetilde{\theta}(t)B(u,\tau) + \frac{1}{2}B^2(u,\tau)\sigma^2 & \text{with } A(u,0) = 0,
\end{cases}$$

where $\tau = T - t$ for a European-style derivative, and $\tau = t_{m+1} - t, t \in [t_m, t_{m+1}]$, for a Bermudan option, between any consecutive exercise dates. For the solution we find $B(u,\tau) = iu\mathrm{e}^{-\kappa\tau}$ and $A(u,\tau)$ contains function $\widetilde{\theta}(t)$, which is given by:

$$\widetilde{\theta}(t) = \theta(t) - \frac{1}{2}\frac{\sigma^2}{\kappa} = \frac{1}{\kappa}\zeta'(t) + \zeta(t).$$

The ODE for $A(u,\tau)$ admits the following solution:

$$\begin{aligned}
A(u,\tau) &= \int_0^\tau A'(u,s)ds \\
&= iu\int_0^\tau (\zeta'(T-s) + \kappa\zeta(T-s))e^{-\kappa s}ds + \frac{1}{4\kappa}u^2\sigma^2(e^{-2\kappa\tau} - 1).
\end{aligned} \tag{3.6}$$

3.2.3 Computational Complexity

The conditional characteristic function between two time points $s < t$ is defined by

$$\begin{aligned}\varphi(u; x, t-s) &:= \mathbb{E}(e^{iuX_t}|X_s) \\ &= \int_{\mathbb{R}} e^{iuy} f(X_t = y|X_s = x, t-s)dy \qquad (3.7) \\ &= e^{iux} \int_{\mathbb{R}} e^{iu(y-x)} f(X_t = x + (y-x)|X_s = x, t-s)dy.\end{aligned}$$

We transform $z := y - x$, which gives

$$\varphi(u; x, t-s) = e^{iux} \int_{\mathbb{R}} e^{iuz} f(X_t - X_s = z|X_s = x, t-s)dz. \qquad (3.8)$$

For underlying processes with independent and stationary increments, like the exponential Lévy processes, the density $f(X_t - X_s = z|X_s = x, t-s)$ only depends on $\Delta t = t-s$, and is independent of x. However, OU processes are not defined by the property of independent and stationary increments as the increment does not only depend on Δt, but also on x. In other words, its characteristic function is of the form (1.27) with $\beta \neq 1$.

From Lemma 1.2.1 we know that if $\beta = 1$ in the general characteristic function form (1.27), efficient computation can be ensured for pricing early–exercise options with the COS method, as the Fourier cosine coefficients of the continuation value can be calculated by the FFT, and the computational complexity at each exercise date is $O(N \log_2 N)$, where N is the number of terms in the Fourier cosine expansions.

However, for the OU models, we have $\beta = e^{-\kappa \Delta t}$, with and without seasonality functions, and we obtain terms of the form $j\beta - k, j\beta + k$ in the matrix elements in (1.29) and (1.30), instead of terms with $j - k, j + k$, for the Lévy jump processes. In particular, the term $1/(j\beta \pm k)$ cannot be decomposed in terms $j \pm k$, j and/or k, so that we cannot formulate H_s, H_c in terms of Hankel and Toeplitz matrices. This hampers an efficient computation of the matrix-vector products, leading to $O(N^2)$ computations at each time step.

In the next section we will therefore introduce *an approximate OU model* for which the efficient pricing technique with FFT can be applied.

3.3 An Approximate OU Model

In this section we introduce an approximation for the characteristic function of the OU processes from Section 3.2, so that the performance of the COS method for Bermudan options can be improved in terms of computational complexity. The aim is to make use of the FFT algorithm *as much as*

possible with a modified characteristic function. Without loss of generality, we will focus on Bermudan put options here.

The original characteristic functions of the OU processes with and without seasonality can be written as:

$$\varphi_{ou}(u;x,\Delta t) = e^{iux}e^{A(u,\Delta t)-iux(1-e^{-\kappa\Delta t})} =: e^{iux}\psi(u;x,\Delta t). \tag{3.9}$$

Without seasonality we have:

$$A(u,\Delta t) = \frac{1}{4\kappa}(e^{-2\kappa\Delta t} - e^{-\kappa\Delta t})(u^2\sigma^2 + ue^{\kappa\Delta t}(u\sigma^2 - 4i\kappa\bar{x})),$$

and with seasonality:

$$A(u,\Delta t) = iu\int_0^{\Delta t}(\zeta'(T-s) + \kappa\zeta(T-s))e^{-\kappa s}ds + \frac{1}{4\kappa}u^2\sigma^2(e^{-2\kappa\Delta t} - 1).$$

The Fast Fourier Transform can be used in the computation of Fourier coefficients $C_k(x_1, x_2, t_m)$, when $\psi(u;x,\Delta t)$ in (3.9) is approximated by a function, which does not contain variable x. The (approximate) characteristic function is then of the form (1.27) with $\beta = 1$. We denote by $\varphi_{app}(u;x,\Delta t)$ this approximate characteristic function. The approximation suggested here is as follows

$$\varphi_{app}(u;x,\Delta t) := e^{iux}\psi(u;\mathbb{E}(x|\mathcal{F}_0),\Delta t). \tag{3.10}$$

In other words, equation (3.8) is replaced by

$$\varphi_{app}(u;x,t-s) := e^{iux}\int_{\mathbb{R}} e^{iuz}f(X_t - X_s = z|X_s = \mathbb{E}(X_s|\mathcal{F}_0), t-s)dz.$$

This approximation may not be accurate for all model parameters when pricing Bermudan options. Comparison of the original characteristic function (3.9) with this approximation (3.10) gives us that

$$\varphi_{app}(u;x,\Delta t) = \varphi_{ou}(u;x,\Delta t)e^{iu\epsilon_1}, \tag{3.11}$$

where ϵ_1 reads

$$\epsilon_1 = (x - \mathbb{E}(x))(1 - e^{-\kappa\Delta t}). \tag{3.12}$$

Based on (3.11) the approximation proposed may only be considered accurate for sets of model parameters for which ϵ_1 in (3.12) is less than a prescribed tolerance level. This tolerance level is defined so that the Bermudan option prices resulting from the approximate characteristic function at each time step are accurate up to a basis point compared to the option prices obtained by the original characteristic function of the OU process.

The tolerance level for ϵ_1, as well as the requirements that model parameters should satisfy so that the approximate characteristic function (3.10) is accurate and thus the Fast Fourier Transform can be used at each time step, are determined by an error analysis in the next section.

3.4 Error analysis

Our aim is to keep the error, defined as the difference between Bermudan option values calculated with the original characteristic function and those obtained with $X_s = x$ approximated by $\mathbb{E}(X_s)$ in $\psi(u; x, \Delta t)$ in (3.9), less than a basis point, which is 1/100-th of a percentage point. We here discuss how the error in the option value can be controlled and the basis point precision can be achieved.

We first introduce the following notation:

- $\epsilon_c(x,t)$ is the error in the continuation value $c(x,t)$ at time t.
- $\epsilon_x(t)$ is the error in early–exercise point x_t^* at time t.
- $\epsilon_V(t)$ is the error in V_k at time t, i.e. the error in the Fourier-cosine coefficients of option value $v(x,t)$.

We focus here on the error in Bermudan option values resulting from our approximate characteristic function. For the basic convergence analysis of the COS pricing method we refer the reader to [34] and [35].

3.4.1 The first step in the backward recursion

The first step in the backward recursion is from $t_{\mathcal{M}} \equiv T$ to $t_{\mathcal{M}-1}$. The *error* in the characteristic function, due to the approximation, gives an error in the continuation value, $c(x, t_{\mathcal{M}-1})$, as well as a *shift* in the early-exercise point, $x^*_{t_{\mathcal{M}-1}}$. These errors contribute to $\epsilon_V(t_{\mathcal{M}-1})$, the error in $V_k(t_{\mathcal{M}-1})$.

The connection between the error in the continuation value and the error in the characteristic function is presented in the following lemma.

Lemma 3.4.1. *The error in continuation value reads*

$$\epsilon_c(x, t_{\mathcal{M}-1}) = c(x + e^{\kappa \Delta t}\epsilon_1, t_{\mathcal{M}-1}) - c(x, t_{\mathcal{M}-1}), \tag{3.13}$$

with ϵ_1 defined in (3.12).

Proof. Applying (3.9) and (3.11) gives us:

$$\begin{aligned} \varphi_{app}(u; x, \Delta t) &= \exp(iuxe^{-\kappa\Delta t} + A(u, \Delta t) + iue^{-\kappa\Delta t}(e^{\kappa\Delta t}\epsilon_1)) \\ &= \exp(iu(x + e^{\kappa\Delta t}\epsilon_1)e^{-\kappa\Delta t} + A(u, \Delta t)) \\ &= \varphi_{ou}(u; x + e^{\kappa\Delta t}\epsilon_1, \Delta t). \end{aligned} \tag{3.14}$$

By substituting (3.14) in the COS pricing formula for continuation value (1.22), we obtain:

$$\hat{c}(x, t_{\mathcal{M}-1}) = c(x + e^{\kappa\Delta t}\epsilon_1, t_{\mathcal{M}-1}),$$

which results in:

$$\epsilon_c(x, t_{\mathcal{M}-1}) = c(x + e^{\kappa\Delta t}\epsilon_1, t_{\mathcal{M}-1}) - c(x, t_{\mathcal{M}-1}).$$

□

Then, we have the following corollary.

Corollary 3.4.1. *For put options, if $\epsilon_1 > 0$, then $\epsilon_c(x, t_{\mathcal{M}-1}) < 0\ \forall x$, and subsequently $\epsilon_x(t_{\mathcal{M}-1}) > 0$, and vice versa if $\epsilon_1 < 0$.*

Proof. The continuation value, $c(x,t)$, is a decreasing function for put options. This implies that, for $\epsilon_1 > 0$,

$$\epsilon_c(x, t_{\mathcal{M}-1}) := c(x + e^{\kappa\Delta t}\epsilon_1, t_{\mathcal{M}-1}) - c(x, t_{\mathcal{M}-1}) < 0.$$

In this case, we have that at the early-exercise point related to the original characteristic function, $x^*_{t_{\mathcal{M}-1}}$:

$$\hat{c}(x^*_{t_{\mathcal{M}-1}}, t_{\mathcal{M}-1}) < c(x^*_{t_{\mathcal{M}-1}}, t_{\mathcal{M}-1}) = g(x^*_{t_{\mathcal{M}-1}}, t_{\mathcal{M}-1}).$$

So, the continuation value is smaller than the payoff. Therefore, the approximate early–exercise point is larger than the original $x^*_{t_{\mathcal{M}-1}}$ and thus $\epsilon_x(t_{\mathcal{M}-1}) > 0$.

For $\epsilon_1 < 0, \forall x$, the proof that for $\epsilon_c(x, t_{\mathcal{M}-1}) > 0\ \ \forall x$, and that $\epsilon_x(t_{\mathcal{M}-1}) < 0$ goes similarly. □

The upper bounds of $|\epsilon_c(x, t_{\mathcal{M}-1})|$ and $|\epsilon_x(t_{\mathcal{M}-1})|$ are found in the next lemma.

Lemma 3.4.2. *For all x in the range of integration $[a,b]$, we have that*

$$|c(x + e^{\kappa\Delta t}\epsilon_1, t) - c(x,t)| \leq e^a e^{\kappa\Delta t}|\epsilon_1| =: \hat{\epsilon}_c, \forall t, \tag{3.15}$$

which implies:

$$|\epsilon_c(x, t_{\mathcal{M}-1})| \leq \hat{\epsilon}_c,\ \forall x \in [a,b]. \tag{3.16}$$

Error $|\epsilon_x(t_{\mathcal{M}-1})|$ can furthermore be bounded in terms of $\hat{\epsilon}_c$.

Proof. Application of Lagrange's mean value theorem, gives

$$\Big|c(x + e^{\kappa\Delta t}\epsilon_1, t) - c(x,t)\Big| = e^{\kappa\Delta t}\Big|\epsilon_1\Big|\Big|\frac{\partial c(x,t)}{\partial x}|_{x=\delta_0}\Big|, \tag{3.17}$$

where $\delta_0 \in (x, x + e^{\kappa\Delta t}\epsilon_1)$. The function $\partial c(x,t)/\partial x$ is a non–positive and non–decreasing [1] function for $x \geq \log(K)$ for Bermudan put options, which goes to zero as $x \to \infty$. Therefore, when $x \geq \log(K)$, we have

$$\max_{x\in[a,b]}\Big|\frac{\partial c(x,t)}{\partial x}\Big| = \Big|\frac{\partial c(x,t)}{\partial x}|_{x=a}\Big|. \tag{3.18}$$

[1] It is non–decreasing is because the payoff of a put option is convex, which implies that the gamma, $\partial c^2(x,t)/\partial x^2$, is non–negative indicating a non-decreasing first derivative $\partial c(x,t)/\partial x$ (for a long position in the option).

We denote this derivative of the continuation value at a by $c'(a,t)$, and thus,

$$|c(x + e^{\kappa\Delta t}\epsilon_1, t) - c(x,t)| \leq e^{\kappa\Delta t}|\epsilon_1||c'(a,t)|. \tag{3.19}$$

If $a < log(K)$, then Eq. (3.18) is not valid and the upper bound is $|c'(K,t)|$. However, it overestimates the error in the option value (that is, in Eq. (3.45)). For example, if x is very small then for the complete integration range $[a,b]$ we will deal with the payoff (not the continuation value) and the error is zero. Therefore we still use $|c'(a,t)|$ for $a < log(K)$, which works well in all our numerical experiments.

Furthermore, at each time step we have $|c'(x,t)| \leq |g'(x,t)|$ for $x < log(K)$. So, if $a < log(K)$ we have

$$|c'(a,t)| \leq |g'(a,t)| = e^a. \tag{3.20}$$

For deep out-the-money (OTM) options, where $a \geq log(K)$, we find

$$|c'(a,t)| \leq |c'(log(K)-1,t)| \leq |g'(log(K)-1,t)| = e^{log(K)-1} \leq e^a. \tag{3.21}$$

Summarizing, we find for all cases:

$$|c'(a,t)| \leq e^a. \tag{3.22}$$

Substitution of (3.22) in (3.19) gives us that for $\forall x \in [a,b]$ and for $\forall t$,

$$|c(x + e^{\kappa\Delta t}\epsilon_1, t) - c(x,t)| \leq e^a e^{\kappa\Delta t}|\epsilon_1| =: \hat{\epsilon}_c,$$

which implies $|\epsilon_c(x, t_{\mathcal{M}-1})| \leq \hat{\epsilon}_c$. We now look at the error in the early–exercise point at $t_{\mathcal{M}-1}$. Assume points $x^*_{t_{\mathcal{M}-1}}$ and $x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1})$ are the early–exercise points obtained from the original and the approximate characteristic functions, respectively. It follows that

$$\begin{aligned} c(x^*_{t_{\mathcal{M}-1}}, t_{\mathcal{M}-1}) &= g(x^*_{t_{\mathcal{M}-1}}, t_{\mathcal{M}-1}), \\ \hat{c}(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}) &= g(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}). \end{aligned}$$

Therefore

$$\begin{aligned} & g(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}) - c(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}) \\ = \;& \hat{c}(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}) - c(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}) \\ =: \;& \epsilon_c(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}). \end{aligned}$$

We introduce a new function $h(x,t) := g(x,t) - c(x,t)$, so that we have

$$\begin{aligned} h(x^*_{t_{\mathcal{M}-1}}, t_{\mathcal{M}-1}) &= 0, \\ h(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}) &= \epsilon_c(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}). \end{aligned}$$

Application of (3.16) gives:

$$\begin{aligned} & |h(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}) - h(x^*_{t_{\mathcal{M}-1}}, t_{\mathcal{M}-1})| \\ = \; & |\epsilon_c(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1})| \leq \hat{\epsilon}_c. \end{aligned} \tag{3.23}$$

Using Lagrange's mean value theorem for (3.23) gives us

$$|\epsilon_x(t_{\mathcal{M}-1})||h'(\overline{\delta}, t_{\mathcal{M}-1})| \leq \hat{\epsilon}_c,$$

for some $\overline{\delta} \in (x^*_{t_{\mathcal{M}-1}}, x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}))$.

The fact that there is one early–exercise point for plain Bermudan put options implies that

$$|h(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}) - h(x^*_{t_{\mathcal{M}-1}}, t_{\mathcal{M}-1})| > 0.$$

If there is an error in the early–exercise point, i.e. if $\epsilon_x(t_{\mathcal{M}-1}) \neq 0$, we have that $|h'(\overline{\delta}, t_{\mathcal{M}-1})| > 0$, so that

$$|\epsilon_x(t_{\mathcal{M}-1})| \leq \frac{\hat{\epsilon}_c}{|h'(\overline{\delta}, t_{\mathcal{M}-1})|}. \tag{3.24}$$

Hence $|\epsilon_x(t_{\mathcal{M}-1})|$ is bounded in terms of $\hat{\epsilon}_c$. If $\hat{\epsilon}_c$ tends to zero, then $|\epsilon_x(t_{\mathcal{M}-1})|$ also tends to zero. □

At the other time points, $m = 0, \ldots, \mathcal{M}-2$, the upper bound for the shift in the early–exercise point can also be determined in terms of the error in the continuation value. However, unlike time step $t_{\mathcal{M}-1}$, the error in the continuation value is not only related to the approximate characteristic function, but also to the error in $V_k(t)$,

$$V_k(t) := \int_a^b \max(c(x,t), g(x,t)) \cos(k\pi \frac{x-a}{b-a}) dx.$$

We first have a look at the error in $V_k(t_{\mathcal{M}-1})$ from (1.23) in the following lemma. We will need the results in the lemma to derive upper bounds in the lemmas to follow.

Lemma 3.4.3. *For $\epsilon_x(t_{\mathcal{M}-1}) > 0$, two points, $\delta_1 \in (x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), b)$ and $\delta_2 \in (x^*_{t_{\mathcal{M}-1}}, x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}))$ exist, so that*

$$\begin{aligned} \epsilon_V(t_{\mathcal{M}-1}) \; = \; & \epsilon_c(\delta_1, t_{\mathcal{M}-1}) I_k(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), b) + \\ & (g(\delta_2, t_{\mathcal{M}-1}) - c(\delta_2, t_{\mathcal{M}-1})) I_k(x^*_{t_{\mathcal{M}-1}}, x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1})). \end{aligned} \tag{3.25}$$

where

$$I_k(x_1, x_2) = \frac{2}{b-a} \int_{x_1}^{x_2} \cos(k\pi \frac{x-a}{b-a}) dx, \tag{3.26}$$

can be viewed as the (analytically available) Fourier-cosine coefficient of an option with value:

$$v_I(x, x_1, x_2) = \begin{cases} 1 & \text{if } x \in [x_1, x_2], \\ 0 & \text{otherwise.} \end{cases} \tag{3.27}$$

Moreover, we have that $|\epsilon_c(\delta_1, t_{\mathcal{M}-1})| \leq \hat{\epsilon}_c$, *and*

$$|g(\delta_2, t_{\mathcal{M}-1}) - c(\delta_2, t_{\mathcal{M}-1})| \leq \hat{\epsilon}_c. \tag{3.28}$$

Proof. Here we assume that both early–exercise points, for the original and approximate OU processes, lie in the integration range. If either x^* or $x^* + \epsilon_x$ lies outside range $[a, b]$, it is set equal to the nearest boundary point. V_k can thus be split as in (1.23) depending on the early–exercise point.

Let us first analyze the case of a *positive error*, $\epsilon_x(t_{\mathcal{M}-1})$, in the early–exercise point at $t_{\mathcal{M}-1}$. Between $x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1})$ and b, for the original and the approximate OU processes, we use the continuation value. The error in $V_k(t_{\mathcal{M}-1})$ in this interval originates from the error in the continuation value, $\epsilon_c(x, t_{\mathcal{M}-1})$. This error, which is denoted by $\epsilon_{V_1}(t_{\mathcal{M}-1})$, reads:

$$\epsilon_{V_1}(t_{\mathcal{M}-1}) = \frac{2}{b-a} \int_{x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1})}^{b} \epsilon_c(x, t_{\mathcal{M}-1}) \cos(k\pi \frac{x-a}{b-a}) dx.$$

By application of the first mean value theorem for integration, there exists a $\delta_1 \in (x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), b)$, so that

$$\begin{aligned} \epsilon_{V_1}(t_{\mathcal{M}-1}) &= \frac{2\epsilon_c(\delta_1, t_{\mathcal{M}-1})}{b-a} \int_{x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1})}^{b} \cos(k\pi \frac{x-a}{b-a}) dx \\ &= \epsilon_c(\delta_1, t_{\mathcal{M}-1}) I_k(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), b), \end{aligned} \tag{3.29}$$

with I_k from (3.26). From (3.16) we have that $|\epsilon_c(\delta_1, t_{\mathcal{M}-1})| \leq \hat{\epsilon}_c$.

Between a and $x^*_{t_{\mathcal{M}-1}}$, for both the original and approximate OU processes, we take the payoff function which for a put option reads $g(x, t) = \max(K - e^x, 0)$. There is no error in the payoff function, hence no error in $V_k(t_{\mathcal{M}-1})$ along this part of the x–axis.

Between $x^*_{t_{\mathcal{M}-1}}$ and $x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1})$, with the original OU process we use continuation value, $c(x, t_{\mathcal{M}-1})$. However, due to the shift in the early–exercise point we have the payoff $g(x, t_{\mathcal{M}-1})$ instead, when using the approximate OU process. This leads to an error in $V_k(t_{\mathcal{M}-1})$, denoted by $\epsilon_{V_2}(t_{\mathcal{M}-1})$, which reads:

$$\epsilon_{V_2}(t_{\mathcal{M}-1}) = \frac{2}{b-a} \int_{x^*_{t_{\mathcal{M}-1}}}^{x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1})} (g(x, t_{\mathcal{M}-1}) - c(x, t_{\mathcal{M}-1})) \cos(k\pi \frac{x-a}{b-a}) dx.$$

By application of again the first mean value theorem for integration, there exists a $\delta_2 \in (x^*_{t_{\mathcal{M}-1}}, x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}))$, so that

$$\begin{aligned}\epsilon_{V_2}(t_{\mathcal{M}-1}) &= (g(\delta_2, t_{\mathcal{M}-1}) - c(\delta_2, t_{\mathcal{M}-1})) \\ &\cdot \quad \frac{2}{b-a}\int_{x^*_{t_{\mathcal{M}-1}}}^{x^*_{t_{\mathcal{M}-1}}+\epsilon_x(t_{\mathcal{M}-1})} \cos(k\pi\frac{x-a}{b-a})dx \\ &= (g(\delta_2, t_{\mathcal{M}-1}) - c(\delta_2, t_{\mathcal{M}-1}))I_k(x^*_{t_{\mathcal{M}-1}}, x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1})).\end{aligned} \tag{3.30}$$

For a put option, $\forall t$, $\forall x > x^*_t$, $c(x,t) - g(x,t) > 0$, and function $c(x,t) - g(x,t)$ is non–decreasing [2] between x^*_t and $x^*_t + \epsilon_x(t)$. This implies:

$$\begin{aligned}|g(\delta_2, t_{\mathcal{M}-1}) &- c(\delta_2, t_{\mathcal{M}-1})| = c(\delta_2, t_{\mathcal{M}-1}) - g(\delta_2, t_{\mathcal{M}-1}) \\ &\leq c(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}) - g(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1}) \\ &= |\epsilon_c(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), t_{\mathcal{M}-1})| \leq \hat{\epsilon}_c.\end{aligned}$$

The last step is from (3.16).

Adding up (3.29) and (3.30) gives

$$\begin{aligned}\epsilon_V(t_{\mathcal{M}-1}) &= \epsilon_{V_1}(t_{\mathcal{M}-1}) + \epsilon_{V_2}(t_{\mathcal{M}-1}) \\ &= \epsilon_c(\delta_1, t_{\mathcal{M}-1})I_k(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), b) + \\ &\quad (g(\delta_2, t_{\mathcal{M}-1}) - c(\delta_2, t_{\mathcal{M}-1}))I_k(x^*_{t_{\mathcal{M}-1}}, x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1})).\end{aligned}$$

□

Remark 3.4.1. *The case when $\epsilon_x(t_{\mathcal{M}-1}) < 0$ goes similarly. It can then be proved that points $\delta_1 \in (x^*_{t_{\mathcal{M}-1}}, b)$ and $\delta_2 \in (x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), x^*_{t_{\mathcal{M}-1}})$ exist, so that*

$$\begin{aligned}\epsilon_V(t_{\mathcal{M}-1}) &= \epsilon_c(\delta_1, t_{\mathcal{M}-1})I_k(x^*_{t_{\mathcal{M}-1}}, b) + \\ &\quad (\hat{c}(\delta_2, t_{\mathcal{M}-1}) - g(\delta_2, t_{\mathcal{M}-1}))I_k(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), x^*_{t_{\mathcal{M}-1}}).\end{aligned} \tag{3.31}$$

Moreover, $|\epsilon_c(\delta_1, t_{\mathcal{M}-1})| \leq \hat{\epsilon}_c$, and $|\hat{c}(\delta_2, t_{\mathcal{M}-1}) - g(\delta_2, t_{\mathcal{M}-1})| \leq \hat{\epsilon}_c$.

3.4.2 Further steps in the backward recursion

We analyze the case $t = t_{\mathcal{M}-2}$ in the following lemma.

Lemma 3.4.4. $|\epsilon_c(x, t_{\mathcal{M}-2})| \leq \hat{\epsilon}_c(1 + e^{-r\Delta t})$, $\forall x \in [a, b]$.

[2] For put options in log-scale when $x \leq 0$, function $c(x,t) - g(x,t)$ is non–decreasing and for $x \geq 0$, $c(x,t) - g(x,t)$ is non–increasing. For a put option early–exercise points are negative. Therefore, between the two early–exercise points, x^*_t and $x^*_t + \epsilon_x(t)$, $c(x,t) - g(x,t)$ is non–decreasing.

Proof. We have, at $t = t_{\mathcal{M}-2}$,

$$\begin{aligned}
|\epsilon_c(x, t_{\mathcal{M}-2})| &:= |\hat{c}(x, t_{\mathcal{M}-2}) - c(x, t_{\mathcal{M}-2})| \\
&= |e^{-r\Delta t} {\sum_{k=0}^{N-1}}' Re(\varphi_{app}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi \frac{a}{b-a}}) \\
&\cdot \quad (V_k(t_{\mathcal{M}-1}) + \epsilon_V(t_{\mathcal{M}-1})) \qquad (3.32) \\
&- \quad e^{-r\Delta t} {\sum_{k=0}^{N-1}}' Re(\varphi_{ou}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi \frac{a}{b-a}}) V_k(t_{\mathcal{M}-1})|.
\end{aligned}$$

Application of (3.14) gives:

$$\begin{aligned}
|\epsilon_c(x, t_{\mathcal{M}-2})| &\leq |e^{-r\Delta t} {\sum_{k=0}^{N-1}}' Re(\varphi_{app}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi \frac{a}{b-a}}) \epsilon_V(t_{\mathcal{M}-1})| \\
&+ |e^{-r\Delta t} {\sum_{k=0}^{N-1}}' Re(\varphi_{ou}(\frac{k\pi}{b-a}; x + e^{\kappa\Delta t}\epsilon_1, \Delta t) e^{-ik\pi \frac{a}{b-a}}) V_k(t_{\mathcal{M}-1}) \\
&- e^{-r\Delta t} {\sum_{k=0}^{N-1}}' Re(\varphi_{ou}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi \frac{a}{b-a}}) V_k(t_{\mathcal{M}-1})| \qquad (3.33) \\
&\leq |e^{-r\Delta t} {\sum_{k=0}^{N-1}}' Re(\varphi_{app}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi \frac{a}{b-a}}) \epsilon_V(t_{\mathcal{M}-1})| + \hat{\epsilon}_c.
\end{aligned}$$

The last step is from (3.15).

By application of Lemma 3.4.3, Equation (3.28) we have, for $\epsilon_x(t_{\mathcal{M}-1}) > 0$,

$$\begin{aligned}
& |e^{-r\Delta t} {\sum_{k=0}^{N-1}}' Re(\varphi_{app}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi \frac{a}{b-a}}) \epsilon_V(t_{\mathcal{M}-1})| \\
\leq \quad & |\epsilon_c(\delta_1, t_{\mathcal{M}-1})| \cdot |e^{-r\Delta t} {\sum_{k=0}^{N-1}}' Re(\varphi_{app}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi \frac{a}{b-a}}) \\
\cdot \quad & I_k(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), b)| + |g(\delta_2, t_{\mathcal{M}-1}) - c(\delta_2, t_{\mathcal{M}-1})| \cdot \\
& |e^{-r\Delta t} {\sum_{k=0}^{N-1}}' Re(\varphi_{app}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi \frac{a}{b-a}}) I_k(x^*_{t_{\mathcal{M}-1}}, x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}))| \\
\leq \quad & \hat{\epsilon}_c e^{-r\Delta t} {\sum_{k=0}^{N-1}}' Re(\varphi_{app}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi \frac{a}{b-a}}) (I_k(x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}), b) + \\
& I_k(x^*_{t_{\mathcal{M}-1}}, x^*_{t_{\mathcal{M}-1}} + \epsilon_x(t_{\mathcal{M}-1}))) \\
= \quad & \hat{\epsilon}_c v_I(x + e^{\kappa\Delta t}\epsilon_1, x^*_{t_{\mathcal{M}-1}}, b), \qquad (3.34)
\end{aligned}$$

where we have used the fact that option values, represented by the cosine series with $I_k(\cdot,\cdot)$, are positive. From (3.27) we have

$$\begin{aligned} v_I(x+e^{\kappa\Delta t}\epsilon_1, x^*_{t_{\mathcal{M}-1}}, b) &= e^{-r\Delta t}\int_{\mathbb{R}} f(y|x+e^{\kappa\Delta t}\epsilon_1,\Delta t)I(y)dy \qquad (3.35)\\ &\leq e^{-r\Delta t}\int_{\mathbb{R}} f(y|x+e^{\kappa\Delta t}\epsilon_1,\Delta t)dy = e^{-r\Delta t}. \end{aligned}$$

Substitution of (3.35) in (3.34) gives us

$$|e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\varphi_{app}(\frac{k\pi}{b-a};x,\Delta t)e^{-ik\pi\frac{a}{b-a}})\epsilon_V(t_{\mathcal{M}-1})| \leq \hat{\epsilon}_c e^{-r\Delta t}. \qquad (3.36)$$

By using (3.36) in (3.33), we obtain

$$|\epsilon_c(x,t_{\mathcal{M}-2})| \leq \hat{\epsilon}_c + \hat{\epsilon}_c e^{-r\Delta t} = \hat{\epsilon}_c(1+e^{-r\Delta t}).$$

When $\epsilon_x(t_{\mathcal{M}-1}) < 0$, it follows similarly, from (3.33) and Remark 3.4.1, that

$$\begin{aligned} |\epsilon_c(x,t_{\mathcal{M}-2})| &\leq \hat{\epsilon}_c + |e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re(\varphi_{app}(\frac{k\pi}{b-a};x,\Delta t)e^{-ik\pi\frac{a}{b-a}})\epsilon_V(t_{\mathcal{M}-1})|\\ &\leq \hat{\epsilon}_c + |\epsilon_c(\delta_1,t_{\mathcal{M}-1})|\cdot|v_I(x+e^{\kappa\Delta t}\epsilon_1, x^*_{t_{\mathcal{M}-1}},b)| +\\ &\quad |\hat{c}(\delta_2,t_{\mathcal{M}-1}) - g(\delta_2,t_{\mathcal{M}-1})|\\ &\quad \cdot\ |v_I(x+e^{\kappa\Delta t}\epsilon_1, x^*_{t_{\mathcal{M}-1}}+\epsilon_x(t_{\mathcal{M}-1}), x^*_{t_{\mathcal{M}-1}})|\\ &\leq \hat{\epsilon}_c(1+e^{-r\Delta t}). \qquad (3.37) \end{aligned}$$

□

The relations in (3.16) and Lemma 3.4.4 serve as the first steps in a mathematical induction proof to find an upper bound for $\epsilon_c(t_0)$, the error in the Bermudan option price at t_0, as follows:

Theorem 3.4.1. *If we assume,* $\forall x \in [a,b], j \in \{1,\ldots,\mathcal{M}-1\}$,

$$\epsilon_c(x,t_{\mathcal{M}-j}) \leq \hat{\epsilon}_c \sum_{l=1}^{j} e^{-r(l-1)\Delta t}, \qquad (3.38)$$

then, it follows that, $\forall x$,

$$\epsilon_c(x,t_{\mathcal{M}-(j+1)}) \leq \hat{\epsilon}_c \sum_{l=1}^{j+1} e^{-r(l-1)\Delta t}. \qquad (3.39)$$

Proof. At $t_{\mathcal{M}-(j+1)}$, we have

$$\begin{aligned}
& |\epsilon_c(x, t_{\mathcal{M}-(j+1)})| = |\hat{c}(x, t_{\mathcal{M}-(j+1)}) - c(x, t_{\mathcal{M}-(j+1)})| \\
= \; & |e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\varphi_{app}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi\frac{a}{b-a}})(V_k(t_{\mathcal{M}-j}) + \epsilon_V(t_{\mathcal{M}-j})) \\
- \; & e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\varphi_{ou}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi\frac{a}{b-a}}) V_k(t_{\mathcal{M}-j})| \\
\leq \; & |e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\varphi_{app}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi\frac{a}{b-a}}) \epsilon_V(t_{\mathcal{M}-j})| + \hat{\epsilon}_c, \qquad (3.40)
\end{aligned}$$

where the last step follows from (3.15).

With arguments as in Lemma 3.4.3 and its proof, we have that for $\epsilon_x(t_{\mathcal{M}-j}) > 0$, values $\delta_1 \in (x^*_{t_{\mathcal{M}-j}} + \epsilon_x(t_{\mathcal{M}-j}), b)$ and $\delta_2 \in (x^*_{t_{\mathcal{M}-j}}, x^*_{t_{\mathcal{M}-j}} + \epsilon_x(t_{\mathcal{M}-j}))$ exist, so that,

$$\begin{aligned}
\epsilon_V(t_{\mathcal{M}-j}) \; = \; & \epsilon_c(\delta_1, t_{\mathcal{M}-j}) I_k(x^*_{t_{\mathcal{M}-j}} + \epsilon_x(t_{\mathcal{M}-j}), b) \\
+ \; & (g(\delta_2, t_{\mathcal{M}-j}) - c(\delta_2, t_{\mathcal{M}-j})) I_k(x^*_{t_{\mathcal{M}-j}}, x^*_{t_{\mathcal{M}-j}} + \epsilon_x(t_{\mathcal{M}-j})).
\end{aligned}$$

With the induction assumptions:

$$|\epsilon_c(\delta_1, t_{\mathcal{M}-j})| \leq \hat{\epsilon}_c \sum_{l=1}^{j} e^{-r(l-1)\Delta t}, \qquad (3.41)$$

and

$$|g(\delta_2, t_{\mathcal{M}-j}) - c(\delta_2, t_{\mathcal{M}-j})| \leq |\epsilon_c(x^*_{t_{\mathcal{M}-j}} + \epsilon_x(t_{\mathcal{M}-j}), t_{\mathcal{M}-j})| \leq \hat{\epsilon}_c \sum_{l=1}^{j} e^{-r(l-1)\Delta t}, \qquad (3.42)$$

and by similar arguments as in Lemma 3.4.4 and its proof, we have that for positive errors in the early–exercise point at $t_{\mathcal{M}-j}$,

$$\begin{aligned}
& |e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\varphi_{app}(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi\frac{a}{b-a}}) \epsilon_V(t_{\mathcal{M}-j})| \\
\leq \; & \hat{\epsilon}_c \sum_{l=1}^{j} e^{-r(l-1)\Delta t} v_I(x + e^{\kappa\Delta t}\epsilon_1, \min(x^*_{t_{\mathcal{M}-j}}, x^*_{t_{\mathcal{M}-j}} + \epsilon_x(t_{\mathcal{M}-j})), b) \\
\leq \; & \hat{\epsilon}_c \sum_{l=1}^{j} e^{-r(l-1)\Delta t} e^{-r\Delta t} = \hat{\epsilon}_c \sum_{l=2}^{j+1} e^{-r(l-1)\Delta t}. \qquad (3.43)
\end{aligned}$$

Therefore, we find that, for $\forall x \in [a,b]$:

$$|\epsilon_c(x, t_{\mathcal{M}-(j+1)})| \leq \hat{\epsilon}_c \sum_{l=2}^{j+1} e^{-r(l-1)\Delta t} + \hat{\epsilon}_c = \hat{\epsilon}_c \sum_{l=1}^{j+1} e^{-r(l-1)\Delta t}. \tag{3.44}$$

□

Remark 3.4.2. *When $\epsilon_x(t_{\mathcal{M}-j}) < 0$, the proof goes similarly, and we can find that $\delta_1 \in (x^*_{t_{\mathcal{M}-j}}, b)$ and $\delta_2 \in (x^*_{t_{\mathcal{M}-j}} + \epsilon_x(t_{\mathcal{M}-j}), x^*_{t_{\mathcal{M}-j}})$ exist, so that*

$$\begin{aligned} \epsilon_V(t_{\mathcal{M}-j}) &= \epsilon_c(\delta_1, t_{\mathcal{M}-j}) I_k(x^*_{t_{\mathcal{M}-j}}, b) \\ &+ (\hat{c}(\delta_2, t_{\mathcal{M}-j}) - g(\delta_2, t_{\mathcal{M}-j})) I_k(x^*_{t_{\mathcal{M}-j}} + \epsilon_x(t_{\mathcal{M}-j}), x^*_{t_{\mathcal{M}-j}}). \end{aligned}$$

With the induction assumptions,

$$|\epsilon_c(\delta_1, t_{\mathcal{M}-j})| \leq \hat{\epsilon}_c \sum_{l=1}^{j} e^{-r(l-1)\Delta t},$$

and

$$|\hat{c}(\delta_2, t_{\mathcal{M}-j}) - g(\delta_2, t_{\mathcal{M}-j})| \leq |\epsilon_c(x^*_{t_{\mathcal{M}-j}}, t_{\mathcal{M}-j})| \leq \hat{\epsilon}_c \sum_{l=1}^{j} e^{-r(l-1)\Delta t},$$

we can then also prove the inequalities (3.43) and (3.44) to hold.

It follows directly from (3.16), Lemma 3.4.4 and Theorem 3.4.1 that at t_0, for any x, the error in the Bermudan option price satisfies:

$$|\epsilon_c(x, t_0)| \leq \hat{\epsilon}_c \sum_{l=1}^{\mathcal{M}} e^{-r(l-1)\Delta t} = \hat{\epsilon}_c \frac{1 - e^{-rT}}{1 - e^{-r\Delta t}} = |\epsilon_1| e^{\kappa \Delta t} e^{a} \frac{1 - e^{-rT}}{1 - e^{-r\Delta t}}.$$

To obtain accuracy up to one basis point[3] for Bermudan options, with the approximate OU process, we prescribe that for all x

$$|\epsilon_c(x, t_0)| < 4 \cdot 10^{-5},$$

which is equivalent to

$$|\epsilon_1| < e^{-\kappa \Delta t} e^{-a} \frac{1 - e^{-r\Delta t}}{1 - e^{-rT}} \cdot 4 \cdot 10^{-5}.$$

Therefore, the approximate characteristic function (3.10) and thus the Fast Fourier Transform can be applied for pricing Bermudan options, if

$$|\epsilon_1| := |x - \mathbb{E}(x)|(1 - e^{-\kappa \Delta t}) < e^{-\kappa \Delta t} e^{-a} \frac{1 - e^{-r\Delta t}}{1 - e^{-rT}} \cdot 4 \cdot 10^{-5} =: TOL. \tag{3.45}$$

[3] To ensure the basis point precision the error in the option price should be less than 10^{-4}. We also consider the influence of rounding up errors, and set therefore $4 \cdot 10^{-5}$ here.

Finally, we need an approximation for $|\epsilon_1|$ in practice. Note that

$$0 \leq (\mathbb{E}|(x - \mathbb{E}(x))|)^2 \leq \mathbb{E}|x - \mathbb{E}(x)|^2 = \mathbb{V}\text{ar}(x) \leq \frac{\sigma^2}{2\kappa}.$$

So,

$$\mathbb{E}(|x - \mathbb{E}(x)|) \leq \frac{\sigma}{\sqrt{2\kappa}}. \tag{3.46}$$

In our implementation we use the upper bound of this expected value to estimate $|\epsilon_1|$ and apply the Fast Fourier Transform with the approximate characteristic function if $\frac{\sigma}{\sqrt{2\kappa}}(1 - e^{-\kappa\Delta t})$ is below the tolerance level defined by (3.45).

3.5 Numerical Results

The FFT can be applied in the parameter range for which

$$\hat{\epsilon}_1 := \frac{\sigma}{\sqrt{2\kappa}}(1 - e^{-\kappa\Delta t}) < TOL,$$

with tolerance level TOL defined in (3.45). In the so-called "non–FFT range", where $\hat{\epsilon}_1 > TOL$, we use the characteristic function of the original OU process to ensure accurate Bermudan option prices.

MATLAB 7.7.0 has been used for all numerical experiments and the CPU is an Intel(R) Core(TM)2 Duo CPU E6550 (@ 2.33GHz Cache size 4MB). CPU time is recorded in seconds.

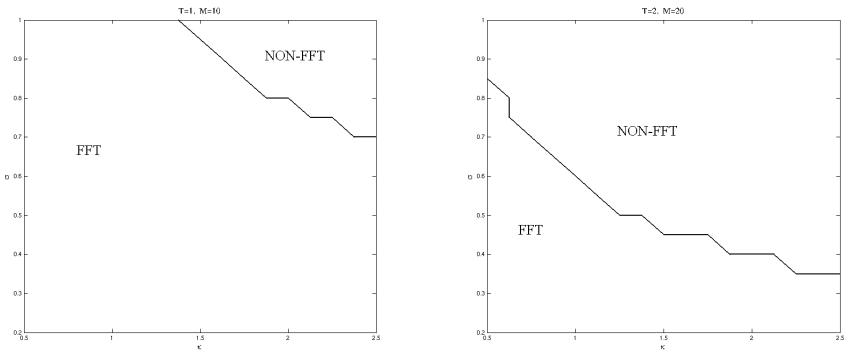
Figure 3.1 compares the FFT ranges with a fixed Δt, but with different maturities, T. With Δt fixed, $\hat{\epsilon}_1$ remains the same for the same κ and σ for all maturities. However, as T increases, the tolerance level (3.45) decreases so that we find a tighter criterion regarding the use of the FFT.

As σ (y–axis in Figure 3.1) increases to certain values, we cannot employ the approximate OU model anymore, as $\hat{\epsilon}_1$ increases, resulting in a large error in the Bermudan option price.

An increase in parameter κ (x–axis in Figure 3.1) also leads to a decrease of the size of the FFT range, see Figure 3.1, due to an increase in the value of $\hat{\epsilon}_1$ and a reduction in the tolerance level (3.45), see Figure 3.2, where $\hat{\epsilon}_1 - TOL$ is plotted as a function of κ.

Figure 3.3 then presents the FFT and non–FFT ranges, with $\mathcal{M} = 5$ and $\mathcal{M} = 50$ for $T = 1$.

As parameter $\mathcal{M}$, the number of exercise dates, increases, the range in which the FFT can be applied expands. However, the influence of $\mathcal{M}$ on the error and the tolerance level (3.45) is different for small and large model parameters. This is illustrated in Figure 3.4. For small κ and σ (example in Figure 3.4a), $\hat{\epsilon}_1 - TOL$ is an increasing function of $\mathcal{M}$, whereas for large model parameters (see Figure 3.4b), $\hat{\epsilon}_1 - TOL$ decreases as $\mathcal{M}$ increases.



(a) $T = 1, \mathcal{M} = 10$ (b) $T = 2, \mathcal{M} = 20$

Figure 3.1: FFT and non–FFT parameter ranges for different maturities, with $\Delta t = 0.1$.

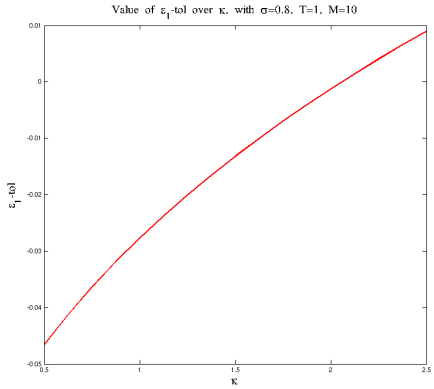
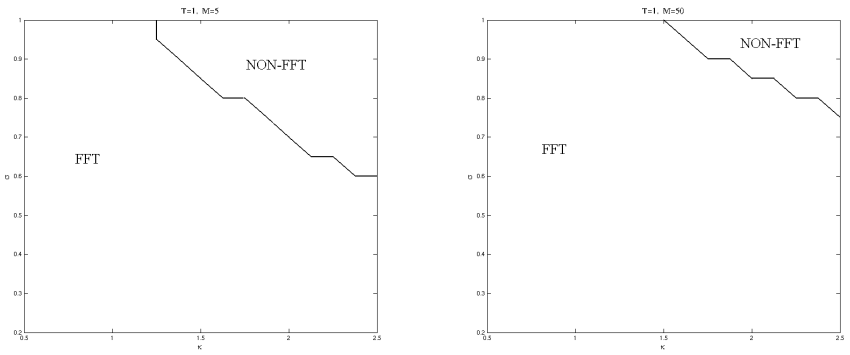


Figure 3.2: Value of $\hat{\epsilon_1} - TOL$ over κ, with $T = 1, \mathcal{M} = 10, \sigma = 0.8$.

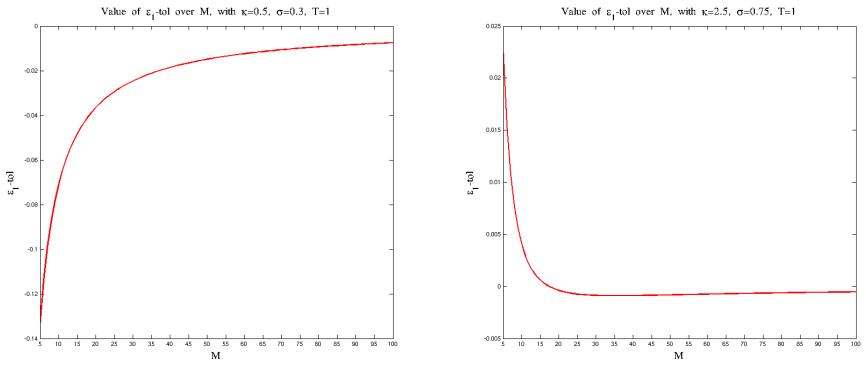
This can be detailed by the derivative of $\hat{\epsilon_1} - TOL$ over $\mathcal{M}$ (in Figure 3.5). In both cases, the quantity goes to zero as $\mathcal{M}$ increases, which implies that function $\hat{\epsilon_1} - TOL$ converges. For small parameters, $\hat{\epsilon_1} - TOL$ converges fast, so that these sets fall in the FFT range (see Figure 3.3). On the other hand, as Figure 3.5b shows, with large model parameters, $\hat{\epsilon_1} - TOL > 0$ when $\mathcal{M}$ is small. For large parameter values, the approximate model can thus be used for large values of $\mathcal{M}$. This insight is particularly helpful for parameter sets at boundary of the FFT and non–FFT ranges, as will be illustrated in the next subsection.

In our next tests we randomly choose different model parameters and check whether the numerical experiments are in accordance with our error analysis. The range of parameters is $\kappa \in [0.5, 2.5]$, $\sigma \in [0.2, 0.8]$, $\mathcal{M} \in \{5, , \ldots, 20\}$, $T \in [1, 2]$, and we use seasonality function $\Xi(t) =$



(a) $\mathcal{M} = 5$ (b) $\mathcal{M} = 50$

Figure 3.3: FFT and non–FFT parameter ranges for different numbers of early–exercise dates, with $T = 1$.

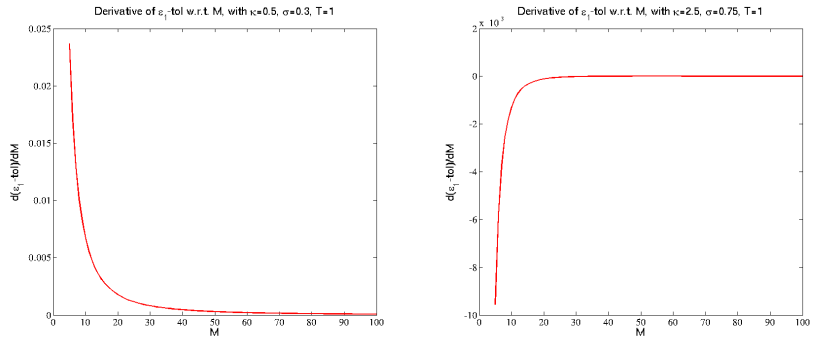


(a) $\kappa = 0.5, \sigma = 0.3$ (b) $\kappa = 2.5, \sigma = 0.75$

Figure 3.4: Value of $\hat{\epsilon_1} - TOL$ over $\mathcal{M}$, with (a) small value of κ and σ and (b) large value of κ and σ.

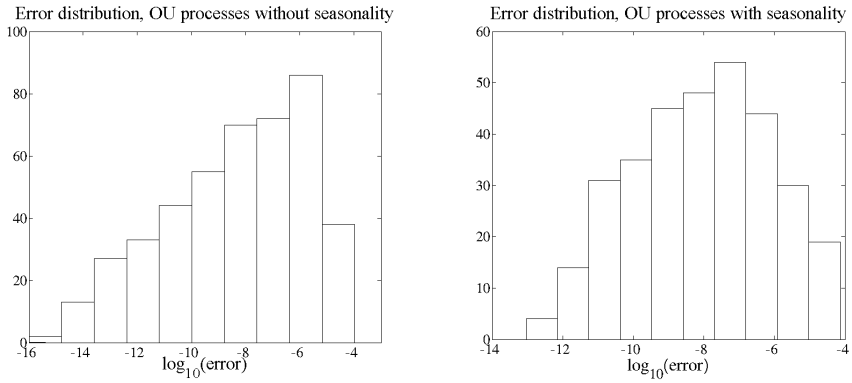
$a_1 + a_2 \sin(a_3 t)$ with $a_1 = 3, 5, 10$, $a_2 \in [0.5, 2]$ and $a_3 \in [0.5, 2]$. Figure 3.6a presents results for the OU process without seasonality, whereas Figure 3.6b shows results for the approximate OU process with seasonality. The x–axes in the figures represent the logarithms of the error in the Bermudan option price.

In these numerical simulations we only consider the continuation values for the Bermudan options, as there is no error in the payoff function $g(x, t)$ neither in its Fourier-cosine coefficients G_k. For all parameter sets considered, the error is of order 10^{-4} or less, which implies that our approximation is accurate up to one basis point.



(a) $\kappa = 0.5, \sigma = 0.3$ (b) $\kappa = 2.5, \sigma = 0.75$

Figure 3.5: Derivative of $\hat{\epsilon_1} - TOL$ with respective to $\mathcal{M}$.



(a) Without Seasonality (b) With Seasonality

Figure 3.6: Simulation result for OU processes: (a) without seasonality and (b) with seasonality.

3.5.1 CPU Time and Accuracy

We perform some more experiments checking the validity of the error analysis.

We first consider the OU process without seasonality and choose the following four sets of model parameters for the numerical examples and set $T = 1$ and $\mathcal{M} = 10, 20, 50$:

1. $\kappa = 0.301, \sigma = 0.334$. This parameter set originates from commodity price calibration in [25].

2. $\kappa = 1, \sigma = 0.5$. This parameter set lies "in the FFT range" for $\mathcal{M} = 10, 20, 50$, as in Figure 3.3.

3. $\kappa = 2, \sigma = 0.7$. This parameter set is at the boundary of the FFT range

(but still inside the FFT range) for $\mathcal{M} = 10, 20, 50$, see Figure 3.3.

4. $\kappa = 2.5, \sigma = 1$. This parameter set lies outside the FFT parameter range for $\mathcal{M} = 10, 20, 50$, see Figure 3.3.

For each parameter set, the CPU time, in seconds, as well as the error are recorded. We set $N = 512$ for which we are sure that convergence is achieved in space when $\mathcal{M} = 20$ and $\mathcal{M} = 50$.

The numerical results are listed in Table 3.1, where *CPU time 1* and *CPU time 2* are the run-times of the Bermudan COS method with the original and the approximate OU model, respectively [4]. Moreover, the log_{10}*(error)* quantity in the table represents the logarithm of the absolute error in the Bermudan option price from the approximate model.

	Parameter	Set 1	Set 2	Set 3	Set 4
$\mathcal{M} = 10$	log_{10}(error)	-10.7784	-7.3434	-4.9392	-2.3424
	CPU time 1	15.2889	15.2108	15.2691	16.9608
	CPU time 2	0.0063	0.0064	0.0065	0.0064
$\mathcal{M} = 20$	log_{10}(error)	-10.9893	-7.0835	-4.4139	-1.9757
	CPU time 1	31.8807	32.2614	32.3253	35.6780
	CPU time 2	0.0120	0.0122	0.0124	0.0124
$\mathcal{M} = 50$	log_{10}(error)	-11.2600	-6.9289	-4.1043	-1.7612
	CPU time 1	81.9564	82.8565	91.6191	91.6244
	CPU time 2	0.0287	0.0296	0.0301	0.0300

Table 3.1: CPU time and error for the two OU processes and different model parameters.

In Table 3.1 it is shown that for all parameter sets in the FFT range (sets 1 to 3), we can confirm the basis point precision. Moreover, the CPU time drops from seconds to milli–seconds if the FFT can be applied. As κ and σ increase (from sets 1 to 4), the error increases and the size of the FFT range reduces. This is in agreement with our analysis. For parameter set 4, for instance, only the use of the original characteristic function ensures the basis point precision.

Table 3.2 gives the results of the Delta value, the first derivative of the Bermudan option value with respect of the underlying stock price at $t = 0$. Here parameter sets 1–3 in the FFT range are used. From Table 3.2 we see that for these parameters, where according to our error analysis the approximation can be used, also the Delta value is within basis point precision.

[4] The FFT is used with the approximate OU model.

	Parameter	Set 1	Set 2	Set 3
$\mathcal{M} = 10$	log_{10}(error)	-10.4071	-7.2635	-5.1049
	CPU time 1	16.1995	15.5985	15.6600
	CPU time 2	0.0065	0.0065	0.0066
$\mathcal{M} = 20$	log_{10}(error)	-10.6045	-6.9894	-4.5673
	CPU time 1	32.8128	33.1084	32.8150
	CPU time 2	0.0122	0.0125	0.0126
$\mathcal{M} = 50$	log_{10}(error)	-10.4156	-6.8262	-4.2511
	CPU time 1	85.3691	84.5079	84.5922
	CPU time 2	0.0291	0.0301	0.0306

Table 3.2: CPU time and error for the two OU processes and different model parameters.

3.5.2 Probability Density Function of ϵ_1

In this subsection we take a closer look at the error in the characteristic function, ϵ_1:

$$\varphi_{app}(u; x, \Delta t) = \varphi_{ou}(u; x, \Delta t)e^{iu\epsilon_1}. \tag{3.47}$$

We have already seen that $\epsilon_1 := (x - \mathbb{E}(x))(1 - e^{-\kappa\Delta t})$. It is a normally distributed process, with $\mathbb{E}(\epsilon_1) = 0$ and $\mathbf{V}ar(\epsilon_1) = \frac{\sigma^2}{2\kappa}(1-e^{-2\kappa t})(1-e^{-\kappa\Delta t})$, because the OU process is also normally distributed.

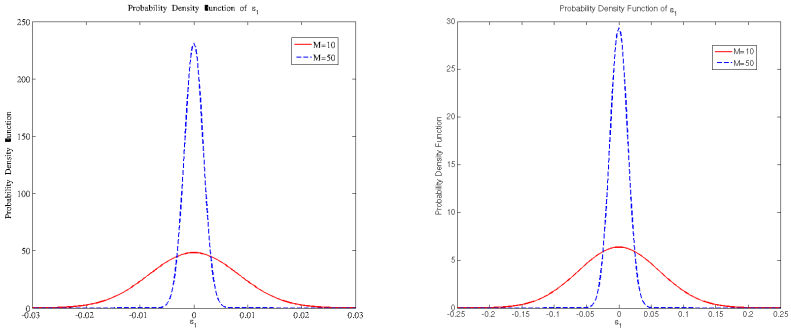
Larger values of parameter t will give rise to larger variance in ϵ_1, with fixed value for Δt. Therefore we analyze here the error ϵ_1 at time point $T - \Delta t$, which gives us the largest variance in the backward recursion. The probability density functions for ϵ_1 with $T = 1$ and $T = 2$ are shown in Figures 3.7 and 3.8, respectively. We have chosen the parameter sets used earlier, i.e., parameter set 1 with $\kappa = 0.301$ and $\sigma = 0.334$ (well in the FFT range) and set 3 with $\kappa = 2$ and $\sigma = 0.7$ (at the boundary of the FFT range) with $T = 1$. For $T = 2$ we used $\sigma = 0.4$ in set 3, so that this set also falls in the FFT range for this test.

For the OU processes with and without seasonality, when model parameters are fixed, the density function of ϵ_1 is essentially the same (with the same variance).

From these figures we see that with T fixed, larger values for $\mathcal{M}$ result in smaller errors in ϵ_1. Moreover, small values for κ and σ also bring smaller errors in characteristic function, compared to the larger model parameters.

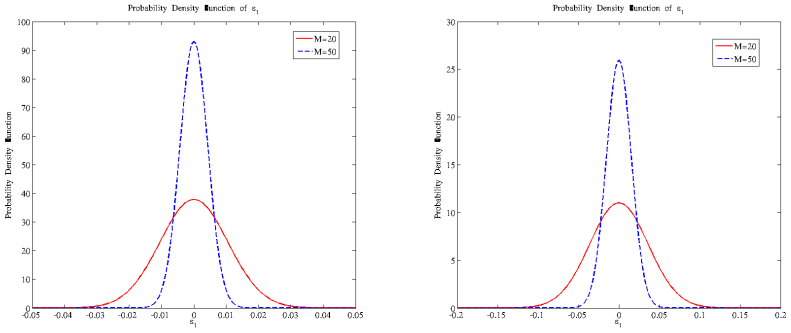
3.5.3 Early-Exercise Points

In this subsection we compare the early–exercise points obtained from the original and approximate characteristic functions, $\varphi_{ou}(u; x, \Delta t)$ and $\varphi_{app}(u; x, \Delta t)$,



(a) $\kappa = 0.301, \sigma = 0.334$ (b) $\kappa = 2, \sigma = 0.7$

Figure 3.7: Probability density function of ϵ_1 with $T = 1$.



(a) $\kappa = 0.301, \sigma = 0.334$ (b) $\kappa = 2, \sigma = 0.4$

Figure 3.8: Probability density function of ϵ_1 with $T = 2$.

respectively. We present early–exercise point $x^*_{t=\Delta t}$, i.e. the value at the last time step of the backward recursion, with $T = 1$ and $\mathcal{M} = 10, 20, 50$.

We use here parameter set 1 ($\kappa = 0.301, \sigma = 0.334$) and parameter set 3 ($\kappa = 2, \sigma = 0.7$, at the boundary of the FFT parameter range).

The results are shown in Table 3.3.

$\mathcal{M}$	$\mathcal{M} = 10$	$\mathcal{M} = 20$	$\mathcal{M} = 50$
Set 1, $\varphi_{ou}(u; x, \Delta t)$	3.1438	3.1489	3.1516
Set 1, $\varphi_{app}(u; x, \Delta t)$	3.1277	3.0742	3.0836
Set 3, $\varphi_{ou}(u; x, \Delta t)$	3.3039	3.2822	3.2684
Set 3, $\varphi_{app}(u; x, \Delta t)$	2.8212	2.8271	2.8084

Table 3.3: Early–exercise points at $t = \Delta t$ for Bermudan put options; parameter sets 1 and 3, $T = 1$, different values for $\mathcal{M}$ (no seasonality).

In Table 3.3 we see an increasing error in the early–exercise point, especially for parameters near the boundary of the range of κ- and σ-values for which the FFT can still be applied. With small parameter values for κ and σ, the error is relatively small (0.08 in Table 3.3).

Obviously, the Bermudan option prices with the approximate process are much more accurate than the values of the corresponding early-exercise points. This can be understood from Equation (3.24) in our error analysis. There the error in the continuation value is divided by a small number (as $f'(x,t)$ is defined as the derivative of the difference of the continuation value and the payoff), resulting in a bigger error in ϵ_x, which is the error in the early-exercise points.

3.5.4 Seasonality Experiment

Now we end with some test cases with seasonality. Our aim is to show that the approximation works well for different seasonality functions. Two seasonality functions are used:

1. Seasonality Function 1: $\Xi(t) = 5 + \sin(t)$,
2. Seasonality Function 2: $\Xi(t) = 3 + 4\cos(0.25t)$.

We check two parameter sets at the boundary of the FFT range, see Figure 3.1 and Figure 3.3.

1. Parameter Set 1: $\kappa = 1.5, \sigma = 0.8, T = 1$.
2. Parameter Set 2: $\kappa = 0.85, \sigma = 0.7, T = 2$.

CPU time as well as the log–absolute error in the option price from our approximate model for different $\mathcal{M}$ are presented in Table 3.4. For the parameter sets at the boundary, we cannot achieve basis point precision for small values of $\mathcal{M}$. However, the error drops below the tolerance level as $\mathcal{M}$ increases, which is in accordance with our analysis.

3.6 Conclusion

In this chapter, we derive a characteristic function for an approximation of the well-known OU process. This approximation enables us to apply the Fast Fourier Transform when pricing Bermudan options by means of the COS method. The approximate process may be employed if the error generated by the approximation is less than a prescribed tolerance level. We would like to ensure that the Bermudan option prices are accurate up to a basis point. This tolerance level is determined by a detailed error analysis. In various numerical experiments it is demonstrated that the characteristic function for the approximate process, in combination with the tolerance level, predicts

Sea.Fun.	Parameter	Set 1 $\mathcal{M} = 10$	Set 1 $\mathcal{M} = 50$	Set 2 $\mathcal{M} = 20$	Set 2 $\mathcal{M} = 40$
1	log_{10}(error)	-3.2217	-4.0466	-3.6035	-4.1734
	CPU time 1	15.7927	84.6543	33.1099	69.5744
	CPU time 2	0.0230	0.0769	0.0457	0.0702
2	log_{10}(error)	-3.5567	-4.4686	-3.4631	-4.1343
	CPU time 1	15.5535	84.7788	34.7118	74.6010
	CPU time 2	0.0232	0.0774	0.0454	0.0690

Table 3.4: CPU time and error with parameter set at boundary of FFT and non–FFT ranges.

well for which model parameter ranges, numbers of early–exercise dates and seasonality functions the FFT can be safely applied. Moreover also the value of Delta is obtained with basis point precision in the FFT range. For the model parameter sets for which the error is below the tolerance level and our approximation can thus be applied, we have reduced the computational time for pricing of Bermudan options under the OU process from seconds to milliseconds.

CHAPTER 4

An Efficient Pricing Algorithm for Swing Options Based on Fourier Cosine Expansions

This chapter contains essentially the contents of paper [70].

4.1 Introduction

Swing options are often encountered derivatives in the world of commodities. A swing option usually consists of two contract parts: a future part and a swing part. The future contract guarantees that the option seller delivers certain amounts of a commodity (base load) to the option buyer at certain times, $T_0 < T_1 \leq T_2 \cdots \leq T_N \leq T$, with T the maturity time. The swing part gives the option buyer the right to order extra or deliver back certain amounts. Usually, the motivation behind the purchase of a swing option is to hedge the uncertainty in the future demand of a commodity. The future part of a swing option can be priced as the discounted expected price of the underlying commodity at the delivery times, whereas the swing part, the focus of the present chapter, can vary in contract complexity and is most interesting from a numerical point-of-view.

In the literature the swing option is often modeled as a Bermudan-style option with swing actions being allowed at the (fixed) delivery times of the base load, combined with some constraints. Pflug and Broussev [56] model the bid and ask prices as the least acceptable contract price and the maximal expected profit over demand patterns, respectively, and those prices are determined by stochastic programming. They present an algorithm to find the equilibrium prices from a game theoretic point-of-view.

Jaillet, Ronn and Tompaidis [42] use a trinomial model where a so-called usage level is discretized. Their model is a multiple layer tree which captures the information of the number of exercise rights remaining, the total amount exercised, and the price scenario. By a swing action one moves from one tree to another. A discrete binomial methodology is also applied by Lari-Lavassani, Simchi and Ware [48], where a transition probability matrix is used to calculate the expected profit, to be maximized over different swing actions at each time step.

Carmona and Touzi [13] view swing options as American-style contingent claims with multiple exercise opportunities and address the problem from the perspective of multiple optimal stopping problems, dealt with by means of Monte Carlo methods and Malliavin calculus. They focus on the Black-Scholes dynamics. Zeghal and Mnif [67] extend that method to Lévy processes.

Unlike the models in which swing actions are only allowed at discrete times, Dahlgren [25] proposed a continuous time model to price the commodity-based swing options. Here the option buyer can exercise the swing option any time before expiry, and more than once, with an upper bound for the maximum amount of additional commodity that can be ordered or delivered back (specified in the contract). After a swing action, the option buyer cannot exercise again unless a recovery time, $\tau_R(D)$, has elapsed, where D represents the amount of commodity. This recovery time can be constant, or dependent on the amount of the last swing action. Dahlgren [25] connected the price of the swing option to a system of discrete variational inequalities of Hamilton-Jacobi-Bellman-type, that is solved by means of finite elements and a projected successive over-relaxation (PSOR) algorithm [24]. A combination of dynamic programming and a finite difference approximation of the resulting partial integro-differential equation (PIDE) under Lévy jump processes has been presented in [46].

The purpose of the present chapter is to develop an efficient alternative solution method for the continuous time model in [25], which is at least competitive with PIDE solvers or Monte Carlo methods in terms of efficiency, accuracy and flexibility. Our solution method for the swing option is based on dynamic programming, backward recursion and Fourier cosine expansions. For the dynamics of the underlying prices, we employ the Ornstein-Uhlenbeck mean reverting process, commonly used in commodity derivatives, and the CGMY Lévy jump process [14]. The present work can be seen as a generalization, in terms of the financial products, of the work in [34, 35].

This chapter is organized as follows. Details of swing options are presented in Section 4.2. In Section 4.3, our contribution to pricing swing options is described in detail. We consider both constant and state-dependent recovery times. Numerical results are presented in Section 4.4. We focus in this chapter on the algorithmic description, which is somewhat technical at

places. An error analysis is not included here, but it is included in [34, 35] for European and Bermudan options, which are the building blocks of the present swing option algorithm.

4.2 Details of the Swing Option

In our discussion, we ignore the future part of the swing option and concentrate on the swing part. Whenever we mention the term 'swing option', it indicates the swing part of the option.

4.2.1 Contract Details

Our assumptions for the swing option are listed below.

- We adopt the concept of *recovery time*, denoted by $\tau_R(D)$, which means that if the option buyer has already exercised the swing option with an amount D at time point t, she has to wait $\tau_R(D)$ time before a next swing action can be conducted. Two different models of recovery time will be considered:
  - Constant recovery time: If $D \neq 0$, $\tau_R(D) \equiv C$, where C is constant.
  - State-dependent recovery time: Here the recovery time is assumed to be an increasing function of D, independent of time t, i.e. $\tau_R(D) = f(D)$.

  Moreover, $\tau_R(D) = 0$ if and only if $D = 0$, and this holds for both types of recovery time.
- A swing option can be exercised at any time after a recovery time delay until the expiry date T. It implies that we deal with an American-style continuous problem.
- With the constraint of recovery time, a swing option can be exercised *more than once* before expiry.
- The amount of commodity at each swing action, D, is assumed to range from $-L, \cdots, -1, 0, 1, \cdots, L$, where a negative amount implies back delivery and a positive amount means ordering. The upper bound, L, is necessary as otherwise it may be optimal to order or deliver back an infinite amount of commodity, and thus receive an unrealistic profit.
- The price the option holder has to pay for *ordering* extra units of the commodity is given by:

$$\begin{cases} S & \text{if } S \leq K_a \\ K_a & \text{if } K_a \leq S \leq S_{max} \\ S - (S_{max} - K_a) & \text{if } S \geq S_{max}, \end{cases}$$

Here $S := S_t$ is the price of the underlying commodity, based on an SDE for S_t, and the values of the strikes K_a and S_{max} are specified in the contract.

- The price the option holder will receive for *delivering back* units of the commodity is

$$\begin{cases} K_d - S_{min} + S & \text{if } S \leq S_{min} \\ K_d & \text{if } S_{min} \leq S \leq K_d \\ S & \text{if } S \geq K_d, \end{cases}$$

where the values of the strikes K_d and S_{min} are also specified in the contract.

Based on the last two assumptions, the payoff function of a swing option is of the form:

$$\begin{aligned} g(S,T,D) = \; & D \cdot (\max(S - K_a, 0) - \max(S - S_{max}, 0) \\ & + \max(K_d - S, 0) - \max(S_{min} - S, 0)) \,, \end{aligned} \tag{4.1}$$

with $S = S_T$. This implies that there can be no profit unless the price of the underlying fluctuates below or above the thresholds K_d or K_a. The two other thresholds, S_{min} and S_{max}, are defined to protect an option writer against extreme fluctuations, see [25]. Figure 4.1 shows an example of the payoff for varying S and D.

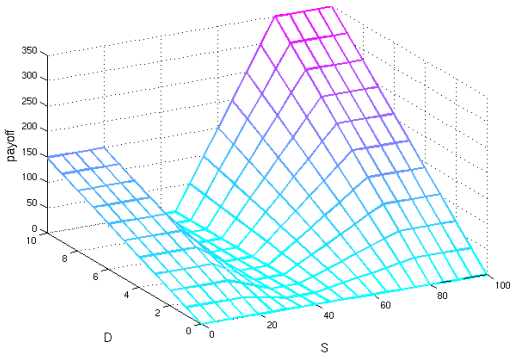


Figure 4.1: Example of a payoff of a swing option with $S_{min} = 20, K_d = 35, K_a = 45$, and $S_{max} = 80$, and S and D varying.

4.2.2 Pricing Details

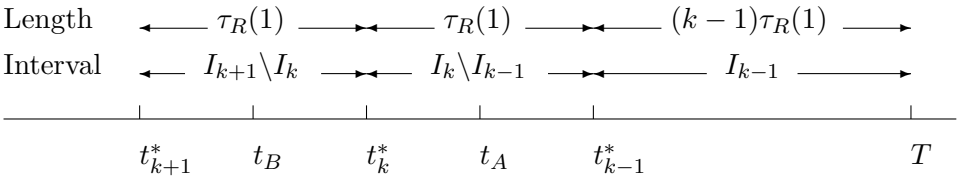
Assume that the first possible time at which a swing action is allowed [1] is T_0: $0 < T_0 < T$. Let

$$n_s := min\{n | n \in \mathbb{N}_+, n \geq (T - T_0)/\tau_R(1)\}, \tag{4.2}$$

where $\tau_R(1)$ is the recovery time when $D = 1$. Then n_s represents the maximum number of swing actions that can be performed in the interval $[T_0, T]$.

We set $t_n^* := T - n\tau_R(1)$, so that t_n^* is the last point in time for which we can have $n + 1$ swing actions, $n = 1, \cdots, n_s - 1$. Moreover, let $I_n = (t_n^*, T]$ and $I_{n_s} = [T_0, T]$ be defined as shown in Figure 4.2 [2].

On I_1, there is only one opportunity left for a swing action, which implies that the recovery time has no further influence for the future. Hence, if it is profitable to exercise the swing option during $(t_1^*, T]$ one should exercise the maximum possible amount, L. In this time interval the only issue which needs to be decided is the optimal exercise time. So, the problem is equivalent to an American-style option pricing problem, and the swing option value for any $t \in (t_1^*, T]$ is equal to the value of an American option, starting from t and expiring at T, with payoff $g(S, t', L)$, $t' \in (t, T]$.



t_A: Option holder can exercise at maximum k times.

t_B: Option holder can exercise at maximum $k + 1$ times.

Figure 4.2: Division of the time axis and the maximum remaining number of swing rights.

At any time $t \in I_{n+1}\backslash I_n$, where $t \neq t_n^*$, $n = 1, \cdots, n_s - 1$, see Figure 4.2,

[1] If $T_0 > T$ we deal with a futures contract, and with $T_0 = T$ the price of the swing option is just the payoff, $g(S, T, 0)$, if a swing action is not profitable, and $g(S, T, L)$ otherwise.

[2] A division of the time interval into portions $I_{n+1}\backslash I_n$ was first proposed by M. Dahlgren in [25]. Our analysis is based on the appendix in [25].

the option holder basically has two options: Either exercise the swing option at any time in $[t, t_n^*]$ or not exercise until t_n^+, the time point immediately after t_n^*.

Note here that the length of interval $I_{n+1}\backslash I_n$ equals $\tau_R(1)$, the recovery time for $D = 1$. It is therefore not possible to exercise more than once within $I_{n+1}\backslash I_n$. In the case of exercise, the problem reduces to the decision of the optimal exercise time within $I_{n+1}\backslash I_n$. So, for each possible amount, D, the problem is equivalent to an American-style option problem, starting at $t \in I_{n+1}\backslash I_n$ and ending at t_n^*, with payoff

$$\overline{g}(S, t', D) = g(S, t', D) + \phi_D^{t'}(S, t'), \quad t' \in [t, t_n^*],\ t \in I_{n+1}\backslash I_n, \tag{4.3}$$

where

$$\phi_D^{t'}(S, t') = e^{-r\tau_R(D)}\mathbb{E}_{S,t'}(v(S, t' + \tau_R(D))), \tag{4.4}$$

and $\mathbb{E}_{S,t'}$ represents the conditional expectation of $v(S, t' + \tau_R(D))$ given $S(t')$.

For each possible value of D, i.e., $D = -L, \cdots, L$, we compute the corresponding value of the swing option at t, assuming that D commodities are bought/sold within $I_{n+1}\backslash I_n$, by an American-style option pricing method. After taking the maximum over all values of D, we obtain the swing option value at $t \in I_{n+1}\backslash I_n$ with $t \neq t_n^*$ *if exercise takes place before* t_n^+. We denote the corresponding option value by $v_1(S, t)$.

On the other hand, if the option holder decides *not to exercise* before t_k^+ she has an option worth the discounted expected value:

$$v_2(S, t) = e^{-r(t_n^+ - t)}\mathbb{E}_{S,t}(v(S, t_n^+)), t \in I_{n+1}\backslash I_n, \tag{4.5}$$

where

$$v(S, t_n^+) = v(S, t_n^* + \delta),\ 0 < \delta << 1.$$

The value $v(S, t_n^+)$ with $t_n^+ \in I_n\backslash I_{n-1}$, has already been obtained at the latest step in the backward recursion. After another, European-type, backward recursion procedure (4.5), value $v_2(S, t)$ is obtained. From the view of a profit maximizing agent, we find that

$$v(S, t) = \max\left(v_1(S, t), v_2(S, t)\right), \quad t \in I_{n+1}\backslash I_n.$$

Moreover, at each t_n^*, the last time point to perform $n + 1$ swing actions, which is also in $I_{n+1}\backslash I_n$, the option value is the maximum of the payoff $\overline{g}(S, t_n^*, D)$ from (4.3), over all values of D, and the value of $v(S, t_n^+)$.

Finally, for $t \in [0, T_0)$, a time interval in which swing actions are not yet allowed, we have

$$v(S, t) = e^{-r(T_0 - t)}\mathbb{E}_{S,t}(v(S, T_0)),$$

which is computed by one step of a European option pricing algorithm.

This concludes the global description of the algorithm for the swing option pricing method.

Summarizing, we can distinguish two major parts in the pricing algorithm:

- For $t \in (t_1^*, T]$, we are faced with an American option pricing problem with payoff $g(S,t,D)$, given by (4.1), which can take five different forms in five different regions of the spot price of the underlying (see Figure 4.1). As mentioned, if it is profitable to exercise the swing option in this time interval, then $D_{opt} = L$. Hence the swing option price is the maximum of $g(S,t,L)$ and the continuation value.

- For the other time regions, $t \in [T_0, t_1^*)$, we compute the following two quantities and compare them within each time region $I_{n+1} \backslash I_n$:

  - The value of an American option, $v_1(S,t)$, with payoff $\bar{g}(S,t,D) := g(S,t,D) + \phi_D^t(S,t)$, as in (4.3), and ϕ_D^t as in (4.4).
  - The discounted value $v_2(S,t) = \mathbb{E}_{S,t}(v(S,t_n^+))$.

  For the values $v(S,t_n^+)$ we only have to calculate the value of $v_1(S,t_n^+)$. This is due to the fact that the discounted value of $\mathbb{E}_{S,t}(v(S,t_{n-1}^+))$ equals $\phi_{D=1}^{t_n^+}(S,t_n^+)$, which is less than (or equal to) the payoff with $D = 1$ (since g is non-negative), and thus less than (or equal to) the corresponding American option value, $v_1(S,t_n^+)$.

4.2.3 Commodity Processes

The commodity underlying the swing option is modeled by a stochastic differential equation for the log–asset process. State variables x and y are defined as the logarithm of the asset price S_t at t_{m-1} and t_m, respectively:

$$x := \log(S_{m-1}) \quad \text{and} \quad y := \log(S_m),$$

respectively. Consequently, (4.1) can be rewritten (keeping the same notation, g, for the function based on the log–asset) as

$$\begin{aligned} g(x,t,D) := \quad & D \cdot (\max(e^x - K_a, 0) - \max(e^x - S_{max}, 0) \\ & + \max(K_d - e^x, 0) - \max(S_{min} - e^x, 0)) \,. \end{aligned} \tag{4.6}$$

Function $\overline{g}$ from Equation (4.3) can be generalized accordingly, also keeping the same notation, $\bar{g}$, for the function based on the log–asset.

Two underlying processes are considered in this chapter, an exponential Ornstein-Uhlenbeck (OU) mean reverting process and a CGMY Lévy jump process. An introduction on the OU and CGMY processes as well as the expression of the corresponding characteristic functions, have been given in detail in Subsection 3.2 and Subsection 1.1.1, respectively. Summarizing,

we deal here with two characteristic functions of the form given in (1.27) in which, for the OU process, $\beta = \exp(-\kappa\Delta t)$, whereas for general Lévy processes, we find $\beta = 1$.

4.3 Fourier Cosine Algorithm for Swing Options

In Section 4.2, we argued that the price of a swing option can be obtained by a series of Bermudan- and American-style option pricing procedures. For pricing we will work with the COS algorithm, which can be applied to processes for which the characteristic function is available. In the following subsections we generalize the COS algorithm to pricing swing options.

Remark 4.3.1. *Subscript n in t_n^*, as well as in t_n^+, decreases, from $n_s - 1$ to 1, if we move forward in time, with t from 0 to T, see Figure 4.2. In contrast, subscript m, denoting the early-exercise dates, in t_m (without $*$) increases and goes from 1 to $\mathcal{M}$, if we move forward in time. Further, there are $N_R = \tau_R(1)/\Delta t \equiv \tau_R(1)\mathcal{M}/T$ early-exercise dates in each time interval $I_{n+1}\backslash I_n$, i.e. between time points t_{n+1}^* and t_n^*.*

4.3.1 Algorithm for the Final Time Interval, $t \in I_1$

We start the detailed description of our pricing algorithm for swing options by considering the last time interval, defined as I_1, see Figure 4.2.

As mentioned in Subsection 4.2.2, in I_1, the swing option is equivalent to an American option. We can thus generalize the algorithm based on the Fourier cosine expansions for Bermudan options to the swing option payoff and combine it with a 4-point repeated Richardson extrapolation to obtain an approximation of an American option price.

Fourier Cosine Coefficients

At $t_{\mathcal{M}} = T$, we have for the Fourier cosine coefficients of the swing option value:

$$V_k(t_{\mathcal{M}}) \quad = \quad G_k(a, \log(K_d), D) + G_k(\log(K_a), b, D), \tag{4.7}$$

with $D = L$, and a, b as in (1.11). Here

$$G_k(x_1, x_2, D) = \frac{2}{b-a}\int_{x_1}^{x_2} g(x, t_{\mathcal{M}}, D)\cos(k\pi\frac{x-a}{b-a})dx \tag{4.8}$$

is the Fourier cosine coefficient of the swing option payoff.
In detail, we find, with $D = L$:

$$\begin{aligned} V_k(t_{\mathcal{M}}) &= \frac{2L}{b-a}\left((K_d - S_{min})\psi_k(a, \log(S_{min}))\right. \\ &+ K_d\psi_k(\log(S_{min}), \log(K_d)) - \chi_k(\log(S_{min}), \log(K_d)) \\ &+ \chi_k(\log(K_a), \log(S_{max})) - K_a\psi_k(\log(K_a), \log(S_{max})) \\ &+ \left.(S_{max} - K_a)\psi_k(\log(S_{max}), b)\right), \end{aligned} \tag{4.9}$$

with

$$\begin{aligned} \chi_k(x_1, x_2) &= \frac{1}{1 + (\frac{k\pi}{b-a})^2}\left(\cos(k\pi\frac{x_2 - a}{b-a})e^{x_2} - \cos(k\pi\frac{x_1 - a}{b-a})e^{x_1}\right. \\ &+ \left.\frac{k\pi}{b-a}\left(\sin(k\pi\frac{x_2 - a}{b-a})e^{x_2} - \sin(k\pi\frac{x_1 - a}{b-a})e^{x_1}\right)\right), \end{aligned} \tag{4.10}$$

and

$$\psi_k(x_1, x_2) = \left(\sin(k\pi\frac{x_2 - a}{b-a}) - \sin(k\pi\frac{x_1 - a}{b-a})\right)\frac{b-a}{k\pi}, \quad (k \neq 0), \tag{4.11}$$

and for $k = 0$, $\psi_k(x_1, x_2) = x_2 - x_1$.

At each time step, t_m, $m = \mathcal{M} - 1, \cdots, 2$, as in the case of a regular Bermudan option, the log-asset values for which the payoff equals the continuation value are determined by Newton's method. Based on these values we can determine in which asset region we should take the continuation value as the option value, and in which region we should take the payoff. In the case of the swing option, there are *two* early-exercise points at each time step, as it is profitable to exercise the option when the underlying is less than K_d or larger than K_a. We denote the lower and upper early-exercise points for time t_m by x_m^d and x_m^a, respectively. To determine the two early-exercise points by Newton's method, we need the values of $c(x, t_m), g(x, t_m, D), \partial c(x, t_m)/\partial x$, and $\partial g(x, t_m, D)/\partial x$, with the help of the following formulae:

$$c(x, t_m) = e^{-r\Delta t}\sum_{k=0}^{N-1}{}' \operatorname{Re}\left\{\varphi(\frac{k\pi}{b-a}; x, \Delta t)e^{-ik\pi\frac{a}{b-a}}\right\} V_k(t_{m+1}), \tag{4.12}$$

$$\frac{\partial c(x, t_m)}{\partial x} = e^{-r\Delta t}\sum_{k=0}^{N-1}{}' \operatorname{Re}\left\{\varphi(\frac{k\pi}{b-a}; x, \Delta t)\cdot i\beta\frac{k\pi}{b-a}\cdot e^{-ik\pi\frac{a}{b-a}}\right\} V_k(t_{m+1}), \tag{4.13}$$

with conditional characteristic functions $\varphi(\omega; x, \Delta t)$ in (4.12) and (4.13) defined in (1.13). Function g is defined in (4.6) and its derivative is given

by the following expression:

$$\frac{\partial g(x,t_m,D)}{\partial x} = \begin{cases} -De^x, & \text{if } \log(S_{min}) \le x \le \log(K_d), \\ De^x, & \text{if } \log(K_a) \le x \le \log(S_{max}), \\ 0, & \text{otherwise.} \end{cases} \tag{4.14}$$

Once x_m^d and x_m^a have been determined, we split the Fourier coefficients V_k into three parts, for $m = \mathcal{M}-1, \cdots, 1$:

$$V_k(t_m) = G_k(a, x_m^d, D) + C_k(x_m^d, x_m^a, t_m) + G_k(x_m^a, b, D),$$

with the Fourier cosine coefficient of the continuation value given by:

$$C_k(x_1, x_2, t_m) = \frac{2}{b-a}\int_{x_1}^{x_2} c(x,t_m)\cos(k\pi\frac{x-a}{b-a})dx, \tag{4.15}$$

and $c(x,t_m)$ defined in (4.12), so that the value of $V_k(t_m)$ is obtained from $V_k(t_{m+1})$.

From basic calculus we have that, if $x_m^d < \log(S_{min})$,

$$G_k(a, x_m^d, D) = D\cdot\frac{2}{b-a}(K_d - S_{min})\psi_k(a, x_m^d), \tag{4.16}$$

and otherwise,

$$\begin{aligned} G_k(a, x_m^d, D) &= D\cdot\frac{2}{b-a}((K_d - S_{min})\psi_k(a, \log(S_{min})) \\ &+ K_d\psi_k(\log(S_{min}), x_m^d) - \chi_k(\log(S_{min}), x_m^d)). \end{aligned} \tag{4.17}$$

If $x_m^a > \log(S_{max})$, we have

$$G_k(x_m^a, b, D) = D\cdot\frac{2}{b-a}(S_{max} - K_a)\psi(x_m^a, b), \tag{4.18}$$

and otherwise,

$$\begin{aligned} G_k(x_m^a, b, D) &= D\cdot\frac{2}{b-a}(\chi_k(x_m^a, \log(S_{max})) - K_a\psi_k(x_m^a, \log(S_{max})) \\ &+ (S_{max} - K_a)\psi_k(\log(S_{max}), b)), \end{aligned} \tag{4.19}$$

where χ_k and ψ_k are defined by (4.10) and (4.11), respectively.

Next we discuss the computation of $C_k(x_m^d, x_m^a, t_m)$ in (4.15). From the proof of Lemma 1.2.1, we know that the value of $C_k(x_1, x_2, t_m)$ is computed as

$$\begin{aligned} C_k(x_1, x_2, t_m) &= -\frac{i}{\pi}\cdot e^{-r\Delta t}\sum_{l=0}^{N-1}{}' \mathrm{Re}\{\phi(\frac{l\pi}{b-a}; \Delta t)V_l(t_{m+1})\cdot \\ &\quad (H_{k,l}^c(x_1,x_2) + H_{k,l}^s(x_1,x_2))\} \\ &= \frac{e^{-r\Delta t}}{\pi}\mathrm{Im}\left\{(H_{k,l}^c + H_{k,l}^s)\mathbf{u}\right\}, \end{aligned} \tag{4.20}$$

with $\phi(u;\Delta t)$ defined in (1.27) and from (1.29) and (1.30) we see that the matrix elements of $H^c_{k,l}(x_1,x_2)$ and $H^s_{k,l}(x_1,x_2)$ are given by

$$H^c_{k,l}(x_1,x_2) = \begin{cases} \dfrac{(x_2-x_1)\pi i}{b-a}, & \text{if } k=l=0, \\ \dfrac{1}{(l\beta+k)} \left[\exp\left(\dfrac{((l\beta+k)x_2-(l+k)a)\pi i}{b-a}\right) - \right. & \\ \left. \exp\left(\dfrac{((l\beta+k)x_1-(l+k)a)\pi i}{b-a}\right)\right], & \text{otherwise.} \end{cases} \tag{4.21}$$

and

$$H^s_{k,l}(x_1,x_2) = \begin{cases} \dfrac{(x_2-x_1)\pi i}{b-a}, & \text{if } k=l=0, \\ \dfrac{1}{(l\beta-k)} \left[\exp\left(\dfrac{((l\beta-k)x_2-(l-k)a)\pi i}{b-a}\right) - \right. & \\ \left. \exp\left(\dfrac{((l\beta-k)x_1-(l-k)a)\pi i}{b-a}\right)\right], & \text{otherwise.} \end{cases} \tag{4.22}$$

where $\beta = 1$ for Lévy processes and $\beta = e^{-\kappa\tau}$ for OU processes. Moreover, $\mathrm{Im}\{\cdot\}$ in (4.20) denotes taking the imaginary part of the input argument, and the expression of vector $\mathbf{u}$ is as follows.

$$\mathbf{u} := \{u_l\}_{l=0}^{N-1}, \quad u_l := \phi\left(\frac{l\pi}{b-a};\Delta t\right) V_l(t_{m+1}),\ u_0 = \frac{1}{2}\varphi\left(0;\Delta t\right) V_0(t_{m+1}). \tag{4.23}$$

From Lemma 1.2.1, we know that for all Lévy processes we need $O(N\log_2 N)$ computations to calculate the Fourier coefficients $C_k(x_1,x_2,t_m)$ and for OU processes the computational complexity is $O(N^2)$. The computation of $G_k(x_1,x_2)$ is linear in N.

Although the algorithm above is only the first step towards solving the pricing problem, it can also be viewed as the complete algorithm for swing options if the option holder is only allowed to conduct a swing action once.

4.3.2 Algorithm for Interval $t \in I_{n_s} \backslash I_1$

Recall that n_s represents the upper bound for the number of swing rights that can be exercised, as defined in (4.2). In the time interval $I_{n_s}\backslash I_1$, the option holder has more than one possibility to exercise the swing option. Therefore, apart from the exercise time, the optimal amount of commodity to be exercised, D, should also be determined, due to its influence on the recovery time.

Remark 4.3.2. *In our discussion we deal with the following three functions:*

- $c(x,t_m)$*, the continuation value, which is typically continuous and differentiable. Moreover, its derivative is usually also continuous.*

- *$g(x, t_m, D)$, the payoff, which is continuous and piecewise differentiable (see Figure 4.1).*

- *$v(x, t_m)$, the option value, which is piecewise continuous in time. $v(x, t)$ jumps at t_n^* where the number of swing rights is decreased by 1.*

Note that the equality $v(t_n^*) = v(t_n^+)$ may not hold, since the number of possible exercise times is reduced by 1 from t_n^* to t_n^+ . The definition of t_n^+ simply implies that, numerically, we use $t_n^+ = t_n^*$. Therefore numerically $t_n^* - t = t_n^+ - t$, and $c(x, t_n^*) = c(x, t_n^+)$, but $v(x, t_n^*) \geq v(x, t_n^+)$.

Under these assumptions we have that

$$e^{-r(t_n^*-t)}\mathbb{E}_{x,t}(v(x, t_n^*)) \geq e^{-r(t_n^+-t)}\mathbb{E}_{x,t}(v(x, t_n^+)).$$

Model Analysis

By Q and Q_n we denote (in this chapter) the continuous interval $\{(x, t)|x \geq 0, t \in [T_0, t_1^*]\}$ and the discrete set $\{(x, t)|x \geq 0, t \in [T_0, t_1^*], t \equiv t_n^* := T - n\tau_R(1), n = 1, \cdots, n_s - 1\}$, respectively.

The swing option value for $(x, t) \in Q\backslash Q_n$ is then given by

$$v(x, t) = \max(\max_D \tilde{v}_{AM}(\bar{g}(x, t, D)), e^{-r(t_n^+-t)}\mathbb{E}_{x,t}(v(x, t_n^+))), \quad (x, t) \in Q\backslash Q_n \tag{4.24}$$

where $\tilde{v}_{AM}(\bar{g}(x, t, D))$ represents the value of an American-style option in any interval $I_{n+1}\backslash I_n$ with payoff $\bar{g}(x, t, D) = g(x, t, D) + \phi_D^t(x, t)$.

The quantity $e^{-r(t_n^+-t)}\mathbb{E}_{x,t}(v(x, t_n^+))$ represents the value of a European option, which cannot be larger than the American option. The term

$$e^{-r(t_n^+-t)}\mathbb{E}_{x,t}(v(x, t_n^+))$$

is therefore implicitly already included in the first term in (4.24), so that we find, for (4.24),

$$\begin{aligned} v(x, t) &= \max_D \tilde{v}_{AM}(g(x, t, D) + \phi_D^t(x, t)) \\ &= \max_D(\max(g(x, t, D) + \phi_D^t(x, t), c(x, t))) \\ &= \max(\max_D g(x, t, D) + \phi_D^t(x, t), c(x, t)), \quad (x, t) \in Q\backslash Q_n, \end{aligned} \tag{4.25}$$

where $c(x, t)$ is the continuation value. Therefore, the price for $(x, t) \in Q\backslash Q_n$ is reduced to the maximum of American option values over D, i.e. $v_1(x, t)$ as defined in Section 4.2.2.

On the other hand, for $(x, t_n^*) \in Q_n$, the value $v(x, t_n^*)$ is defined by

$$v(x, t_n^*) = \max(\max_D \bar{g}(x, t_n^*, D), v(x, t_n^+)). \tag{4.26}$$

After application of (4.25) to the right-hand side of (4.26), we can rewrite (4.26) as

$$v(x,t_n^*) = \max(\max_D \bar{g}(x,t_n^*,D), \max_D \bar{g}(x,t_n^+,D), c(x,t_n^+)), \tag{4.27}$$

where we assume $c(x,t_n^+) = c(x,t_n^*)$, and $\bar{g}$ is as in (4.3),(4.4).

If $(x,t_n^* + \tau_R(D)) \in Q \backslash Q_n$, with the number of exercise possibilities the same for $t_n^* + \tau_R(D)$ and $t_n^+ + \tau_R(D)$, we have $v(x,t_n^* + \tau_R(D)) = v(x,t_n^+ + \tau_R(D))$. If $t_n^* + \tau_R(D) \in Q_n$, we have $v(x,t_n^* + \tau_R(D)) \geq v(x,t_n^+ + \tau_R(D))$.

So, $v(x,t_n^* + \tau_R(D)) \geq v(x,t_n^+ + \tau_R(D))$ for any x. Thus, from (4.4) we have $\phi_D^{t_n^*}(x,t_n^*) \geq \phi_D^{t_n^+}(x,t_n^+)$. Equation (4.27) is now given by:

$$v(x,t_n^*) = \max(\max_D g(x,t_n^*,D) + \phi_D^{t_n^*}(x,t_n^*), c(x,t_n^*)). \tag{4.28}$$

As a result, from (4.25) and (4.28), we find that for all $t \in [T_0, t_1^*]$:

$$v(x,t) = \max(\max_D g(x,t,D) + \phi_D^t(x,t), c(x,t)). \tag{4.29}$$

Equation (4.29) tells us that the swing option is an American-style option with recovery time and multiple exercise opportunities. Its pricing algorithm is therefore different from a standard American option. Instead of taking the maximum of the payoff and the continuation value, we take the maximum of the resulting payoff for all possible values of D, and the continuation value from the previous time step. Another difference is that for any amount, D, the payoff also includes the term $\phi_D^t(x,t)$ from an earlier time step.

It is easy to determine the value of $g(x,t,D)$ for any x,t,D according to (4.6). We therefore focus on the values $\phi_D^t(x,t)$ and $c(x,t)$, which are both obtained in the recursion of Fourier cosine coefficients V_k. To calculate $c(x,t_m)$, one only needs the values of $V_k(t_{m+1})$, like in the case of a Bermudan option. However, to compute the value of $\phi_D^t(x,t)$ we need the coefficients $V_k(t + \tau_R(D))$, which depend on the function for the recovery time.

Remark 4.3.3. *In time interval $t \in [0,T_0]$ swing actions are not yet allowed. Therefore, we have:*

$$v(t,x) = e^{-r(T_0-t)} \sum_{k=0}^{N-1}{}' \operatorname{Re}\left\{ \varphi(\frac{k\pi}{b-a}; x, \Delta t) e^{-ik\pi \frac{a}{b-a}} \right\} V_k(T_0),$$

where $V_k(T_0)$ is obtained by a backward recursion procedure.

4.3.3 The Early-Exercise Points

In this section we consider the state-dependent recovery time, $\tau_R(D)$, which is assumed to be an increasing function of D.

The option value is obtained by means of a backward recursion on $V_k(t_m), m = \mathcal{M}-1, \cdots, 1$. At each time step, as shown in Section 4.3.2, the payoff, $\bar{g}(x, t_m, D)$, for all possible values of D and the continuation value, $c(x, t_m)$, are compared. The largest value represents the swing option value at t_m. We therefore need to identify the following regions in our pricing domain:

- $A_D, D = 1, \cdots, L$: the regions in which exercising the swing option with D commodity units will result in the highest profit $g(x, t_m, D) + \phi_D^{t_m}(x, t_m)$.
- A_c: The region in which $c(x, t)$ is the maximum. In other words, with the commodity price in A_c, it is profitable *not* to exercise the swing option.

With these regions determined, the Fourier cosine coefficients, $V_k(t_m)$, for the swing option can be determined with a splitting, as follows,

$$\begin{aligned} V_k(t_m) &= \frac{2}{b-a}\left(\int_{A_c} c(x, t_{m+1}) \cos(\frac{k\pi(x-a)}{b-a})dx \right. \\ &+ \left. \sum_{D=1}^{L} \int_{A_D} g(x, t_m, D) \cos(\frac{k\pi(x-a)}{b-a})dx \right). \end{aligned} \tag{4.30}$$

We now describe the procedure to determine the different regions A_c and $A_D, D = 1, \cdots, L$. As an example, let us first look at the payoff functions for *two* values $D = D_1$ and $D = D_2$ where $D_1 > D_2$, shown in Figure 4.3. Points

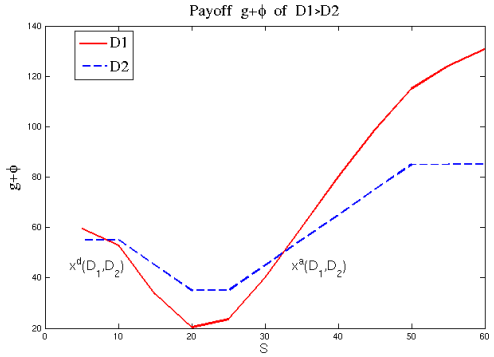


Figure 4.3: Payoff function $g + \phi$ for two different D.

$x^d(D_1, D_2)$ and $x^a(D_1, D_2)$ denote the "early-exercise points", where the strategy of exercising D_1 or D_2 units results in the same $\bar{g}$-values. Between $x^d(D_1, D_2)$ and $x^a(D_1, D_2)$, the value for D_2 is largest, in other words, it is profitable to exercise a smaller amount of commodity. Beyond $x^d(D_1, D_2)$ and $x^a(D_1, D_2)$, it is profitable to exercise the larger amount D_1.

Remark 4.3.4. *An explanation of the behavior of the two payoff functions in Figure 4.3 is as follows. The payoff, $\overline{g}(x,t,D)$, is the sum of $g(x,t,D)$ and $\phi_D^t(x,t)$. For D increasing, the true payoff $g(x,t,D)$ increases, but the quantity $\phi_D^t(x,t)$ decreases because of the time penalty due to the longer recovery.*

For $0 < D_2 < D_1$, we have

$$\begin{aligned} \overline{g}(x,t,D_1) - \overline{g}(x,t,D_2) &= g(x,t,D_1) - g(x,t,D_2) \\ &- (\phi_{D_2}^t(x,t) - \phi_{D_1}^t(x,t)). \end{aligned} \tag{4.31}$$

With the underlying between K_d and K_a, we have $g(x,t,D_1) = 0$, $g(x,t,D_2) = 0$ and $\phi_{D_2}^t(x,t) > \phi_{D_1}^t(x,t)$. Therefore, $\overline{g}(x,t,D_1) < \overline{g}(x,t,D_2)$ and it is more profitable to exercise the smaller amount D_2.

From (4.6) we find for the derivative:

$$\begin{aligned} \frac{\partial g}{\partial D} &= (\max(e^x - K_a, 0) - \max(e^x - S_{max}, 0) \\ &+ \max(K_d - e^x, 0) - \max(S_{min} - e^x, 0)) \equiv g(x,t,1), \end{aligned}$$

which increases as $S = e^x$ decreases/increases beyond K_d or K_a, until payoff $g(x,t,1)$ reaches its upper bound, see Figure 4.1. Therefore, if, before x reaches $\log(S_{\min})$ or $\log(S_{\max})$, we have for some x, $\frac{\partial g(x,t,D)}{\partial D} > \frac{\partial \phi_D^t(x,t)}{\partial D}$, this implies that the payoff function $g(x,t,D)$ is more sensitive with respect to variation in D than function $\phi_D^t(x,t)$, and it is thus more profitable to exercise at the larger amount D_1.

Based on the insight in Remark 4.3.4, let us look at a second example with $L = 4$ and we will determine A_2, i.e. the region where it is most profitable to exercise two units. The example is detailed in Figure 4.4, where the relation between the payoffs for any two different amounts of commodity is graphically sketched. Figure 4.4 is a one-dimensional picture with only the x–axis, which consists of different sections, where purchasing two different amounts of commodity are compared on each horizontal line in Figure 4.4.

In the figure, "0", denotes the continuation value $c(x,t)$. Point sets $x^d(2,D_j)$ and $x^a(2,D_j)$, $j = 0,1,3,4$, are the two sets of points for which $D = D_j$ gives the same payoff value as $D = 2$. In order to determine the region A_2, we need to find the sub-regions in which $D = 2$ gives the largest payoff compared to the other D-values.

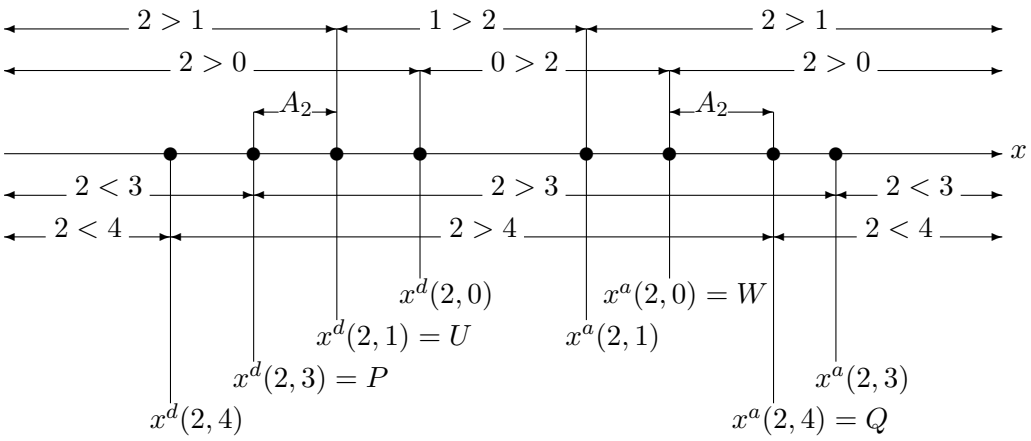


Figure 4.4: An example to illustrate the exercise region A_2 with $L = 4$.

The value $D = 2$ returns a larger value than $c(x,t)$, if $x < x^d(2,0)$ or $x > x^a(2,0)$; Similarly, $D = 2$ returns a larger value than $D = 1$, if $x < x^d(2,1)$ or $x > x^a(2,1)$. So, $D = 2$ returns larger values than both $c(x,t)$ and $D = 1$, if x is either smaller than both $x^d(2,0)$ and $x^d(2,1)$, or larger than both $x^a(2,0)$ and $x^a(2,1)$. To determine these regions we compute the following early-exercise points (see again Figure 4.4 for the values of U and W for this example):

- $U := \min(x^d(2,0), x^d(2,1)) \equiv x^d(2,1)$,
- $W := \max(x^a(2,0), x^a(2,1)) \equiv x^a(2,0)$.

$D = 2$ now returns a larger value for $x < U$ or $x > W$.

We proceed in the same spirit: To make sure that $D = 2$ returns larger values than $D = 3$ and $D = 4$, x should be *larger* than both $x^d(2,3)$ and $x^d(2,4)$, or *smaller* than both $x^a(2,3)$ and $x^a(2,4)$. This is again related to the global behavior of the payoff functions with $D_1 > D_2$, as in Figure 4.3. Therefore we calculate

- $P := \max(x^d(2,3), x^d(2,4)) \equiv x^d(2,3)$
- $Q := \min(x^a(2,3), x^a(2,4)) \equiv x^a(2,4)$

Now $D = 2$ returns a larger value than $D = 3$ and $D = 4$ for $x > P$ or $x < Q$.

So, $D = 2$ returns the largest value, if $P < x < U$ or $W < x < Q$; Therefore, $A_2 = [P,U] \cup [W,Q]$, as shown in Figure 4.4.

More generally, for each $D = 1, \cdots, L$, we determine:

$$P_D = \max_{j>D} x^d(D,j),\ Q_D = \min_{j>D} x^a(D,j),\ U_D = \min_{j<D} x^d(D,j),\ W_D = \max_{j<D} x^a(D,j),$$

and set $A_D = [P_D, U_D] \cup [W_D, Q_D]$. Here P_D, Q_D represent the early-exercise interval boundaries, within which exercising D units of commodity returns a larger payoff than exercising more units. U_D, W_D are the left and right boundary, respectively, beyond which exercising D units returns a larger value than when fewer or no units are exercised. Similarly, we have

$$\begin{aligned} A_L &= [a, \min_{j<L} x^d(L,j)] \cup [\max_{j<L} x^a(L,j), b], \\ A_c &= [\max_{j>0} x^d(0,j), \min_{j>0} x^a(0,j)]. \end{aligned}$$

All early-exercise points, $x^d(D,j), x^a(D,j),\ j = 0, \ldots, L$, are computed by Newton's method.

With the regions A_c and $A_D, D = 1, \cdots, L$, fixed, Equation (4.30) can be rewritten as:

$$\begin{aligned} V_k(t_m) &= C_k(\max_{j=1,\cdots,L} x^d(0,j), \min_{j=1,\cdots,L} x^a(0,j), t_m) + \sum_{D=1}^{L} G_k(P_D, U_D, D) \\ &+ \sum_{D=1}^{L} G_k(W_D, Q_D, D) + G_k(a, \min_{j=0,\cdots,L-1} x^d(L,j), L) \\ &+ G_k(\max_{j=0,\cdots,L-1} x^a(L,j), b, L). \end{aligned} \tag{4.32}$$

The computation of $C_k(x_1, x_2, t_m)$ in (4.32) is as in (4.20). The G_k differ from the expressions (4.16) ,..., (4.19), which will be described in detail in Subsection 4.3.3.

In the Newton procedure to find the points $x^d(D_i, D_j)$ and $x^a(D_i, D_j)$ we need to find the values of $c(x, t_m), g(x, t_m, D), \partial c/\partial x$ and $\partial g/\partial x$ as in Subsection 4.3.1. The values of $\phi_D^{t_m}(x, t_m)$ and $\partial \phi_D^{t_m}/\partial x$ are found by:

$$\begin{aligned} \phi_D^{t_m}(x, t_m) &= e^{-r\tau_R(D)} {\sum_{k=0}^{N-1}}' \mathrm{Re}\left\{ \varphi(\frac{k\pi}{b-a}; x, \tau_R(D)) e^{-ik\pi \frac{a}{b-a}} \right\} \cdot V_k(t_m + \tau_R(D)), \\ \frac{\partial \phi_D^{t_m}}{\partial x} &= e^{-r\tau_R(D)} {\sum_{k=0}^{N-1}}' \mathrm{Re}\left\{ \varphi(\frac{k\pi}{b-a}; x, \tau_R(D)) \cdot i\beta \frac{k\pi}{b-a} e^{-ik\pi \frac{a}{b-a}} \right\} \cdot \\ & \quad V_k(t_m + \tau_R(D)). \end{aligned}$$

Remark 4.3.5 (Computation of $V_k(t_m + \tau_R(D))$)**.** *To calculate $V_k(t_m + \tau_R(D))$, we determine a time step, Δt, so that $T - t$ and $\tau_R(D)$ are both time points. So, we set $\mathcal{M} = T - t/\Delta t, N_D = \tau_R(D)/\Delta t,\ D = 1, \cdots, L$. For $t_m + \tau_R(D) = t_m + N_D \Delta t \le T$, the value $V_k(t_m + \tau_R(D)) = V_k(t_{m+N_D})$. The values $V_k(t_m + \tau_R(D)) = 0$ for all k if $t_m + N_D \Delta t > T$. In that case, $\phi_D^{t_m}$ and $\partial \phi_D^{t_m}/\partial x$ are zero, as they are linear combinations of $V_k(t_m + \tau_R(D))$. In this setting, $V_k(t_m)$ and $V_k(t_m + \tau_R(D)), D = 1, \cdots, L$ can be determined in one recursion, in which the intermediate values of V_k need to be stored for later use.*

Calculation of $G_k(x_1, x_2, D)$

The terms G_k in (4.30) are split into two parts, i.e.

$$G_k(x_1, x_2, D) = G_{k,g}(x_1, x_2, D) + G_{k,c}(x_1, x_2, D),$$

with $G_{k,g}$ from an instantaneous profit $g(x, t_m, D)$, and $G_{k,c}$ the part generated by $\phi_D^{t_m}(x, t_m)$, i.e., the continuation value from time point $t_m + \tau_R(D)$, as defined in (4.4).

Equations (4.16) and (4.17) can be used to compute the Fourier terms $G_{k,g}(a, \min_{j<L} x^d(L, j), L)$ and $G_{k,g}(P_D, U_D, D), D = 1, \cdots, L$, unless $P_D > \log(S_{min})$ where we use,

$$G_{k,g}(P_D, U_D, D) = D \cdot \frac{2}{b-a}(K_d \psi_k(P_D, U_D) - \chi_k(P_D, U_D)).$$

Similarly, the quantities $G_k(\max_{j<L} x^a(L, j), b, L)$ and $G_k(W_D, Q_D, D), D = 1, \cdots, L$ can be computed by (4.18) and (4.19), unless $Q_D < \log(S_{max})$ for which we have

$$G_{k,g}(W_D, Q_D, D) = D \cdot \frac{2}{b-a}(\chi_k(W_D, Q_D) - K_a \psi_k(W_D, Q_D)).$$

Finally, the quantity $G_{k,c}(x_1, x_2, D)$ can be obtained by (4.20), replacing Δt and $V_l(t_{m+1})$ by $\tau_R(D)$ and $V_l(t_m + \tau_R(D))$, respectively.

Remark 4.3.6 (Early-Exercise Points and Convergence)**.** *The accurate determination of the early-exercise points, and the consistent pricing of Bermudan-style swing options forms the basis for the valuation of the American-style swing options by means of the Richardson extrapolation scheme. Only with an accurate location of the early-exercise points we can benefit from extrapolation techniques which rely heavily on (consistent) asymptotic expansions.*

The main components of the swing option pricing algorithm here are those that have also been used for pricing Bermudan and barrier options with Fourier cosine expansions in [35]. The convergence of the swing option algorithm is therefore expected to be the same as that for Bermudan options, which has been studied in detail in [35].

Remark 4.3.7 (Constant recovery time)**.** *If the recovery time does not depend on D, we call the recovery time constant. This can be viewed as a special case of the pricing method discussed above. As additional profit is not related to an extra penalty, if it is profitable to exercise the swing option, we have $D_{opt} \equiv L$ from a profit maximizing point-of-view. Hence, at any point in time, we have either $D = 0$, or $D = L$.*

Newton's method is now applied to determine two early-exercise points x_m^d and x_m^a, so that

$$c(x_m^d, t_m) = g(x_m^d, t_m, L) + \phi_L^{t_m}(x_m^d, t_m),$$

and

$$c(x_m^a, t_m) = g(x_m^a, t_m, L) + \phi_L^{t_m}(x_m^a, t_m),$$

with $D = L$ *and* $\tau_R(D)$ *constant. Then,* $V_k(t_m)$ *is split into three parts,*

$$V_k(t_m) = G_k(a, x_m^d, L) + C_k(x_m^d, x_m^a, t_m) + G_k(x_m^a, b, L),$$

that can be calculated as in the case of state-dependent recovery time.

4.4 Numerical Results

In this section we demonstrate the performance of our pricing algorithm for swing options with constant and dynamic recovery times. The CPU used is an Intel (R) Core (TM) 2 Duo CPU E6550 2.33GHz, Cache size 4MB, and the algorithm is programmed in MATLAB 7.5. The two sub-sections to follow present results with two different types of recovery time:

- Constant recovery time is in Subsection 4.4.1: If $D \neq 0$, we set $\tau_R(D) = \frac{1}{4}$, as in [25]. In other words, the option holder needs to wait 3 months between two consecutive swing actions, independent of the time point of exercise or the size D.
- State-dependent recovery time is in Subsection 4.4.2: We assume $\tau_R(D) = D/12$ which implies that if the option holder exercises the swing option with D units, she has to wait D months before the option can be exercised again.

Parameter sets used for numerical examples are (unless stated otherwise):

$$\text{CGMY} \qquad C = 1, G = 5, M = 5, Y = 1.5, r = 0.05, \tag{4.33}$$

$$\text{OU}: \qquad \kappa = 0.301, \bar{x} = 3.150, \sigma = 0.334, r = 0.05, \tag{4.34}$$

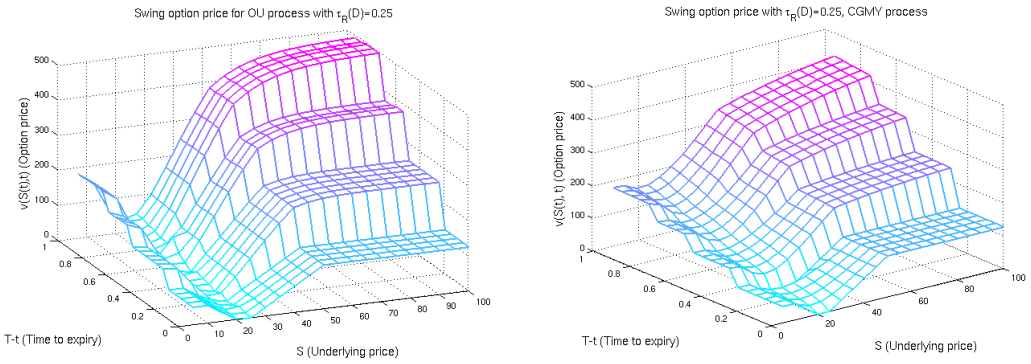
where for the OU process the value of $\bar{x}$ is defined under the Q-measure. The value set for the OU process is as in [25]. The values for CGMY, in particular $Y > 1$ (infinite activity jump process) are known to be particularly difficult for PIDE solvers. We will see here that these CGMY parameters do not pose any problem for the swing option COS method.

In the numerical experiments we further choose $S_{min} = 10, K_d = 20, K_a = 25, S_{max} = 50, T_0 = 0$. The choice $T_0 = 0$ does not pose any restrictions on the algorithm, as we can simply change it to any $T_0 > 0$.

4.4.1 Constant Recovery Time

First of all, American-style swing option values under the CGMY and OU processes, with $L = 5$, are presented in Figure 4.5, with as independent variables S and t; $v(S, t)$ is the swing option value. Jumps in the swing

option values are observed at $t = 0.25, t = 0.5$ and $t = 0.75$. This can be explained by the fact that at these time points the maximum number of times the holder can exercise, n_s, is reduced by one. For instance, time point $t = 0.5$ is the last time point at which an option holder can exercise up to three times. For any $t > T - 0.5$, the holder cannot exercise more than twice.



(a) OU (b) CGMY

Figure 4.5: American-style swing option values under the OU and CGMY processes with constant recovery time, $\tau_R(D) = 0.25$.

Due to the constant recovery time, we should exercise $L = 5$ units whenever it is profitable to exercise. Hence for $S > 50$, with $K_a = 25$, the profit would be $L \cdot (50 - 25) = 125$. When $T - t \in [0.75, 1)$, we have at maximum four possibilities to exercise, which is the reason for option values as high as 500 in Figure 4.5.

Next, we discuss the convergence behavior of the option values over N, the number of terms in the Fourier cosine series. The CGMY and OU processes are used with the parameters in (4.33), (4.34). The remaining parameters are $\tau_R = 0.25, T = 1, \mathcal{M} = 12$ and $S_0 = 8$.

In Table 4.1 it is shown that the swing option pricing algorithm for the CGMY and OU processes, with the parameters chosen, take 0.024 and 0.19 seconds, respectively, to converge to one basis point accuracy. The CPU time is higher for the OU process as its computational complexity is of higher order than for the Lévy processes. The convergence behavior for both processes is very similar, as shown in Table 4.1.

An American option can be viewed as a Bermudan option with $\mathcal{M} \to \infty$. In Table 4.2 the performance of two methods to approximate an American-style swing option is compared. One method is the direct approximation by means of Bermudan-style options by increasing $\mathcal{M}$, whereas the second method is based on the repeated Richardson 4-point extrapolation technique (1.32) on Bermudan-style swing options with four different numbers

N	64	96	128	160	192
CGMY option value	190.2045	229.0515	229.0515	229.0515	229.0515
CPU time (sec.)	0.0191	0.0234	0.0266	0.0304	0.0402
N	64	96	128	160	192
OU option value	225.9100	225.9100	225.9100	225.9100	225.9100
CPU time (sec.)	0.1891	0.3994	0.7147	1.1638	1.7590

Table 4.1: Swing option prices and CPU time under the CGMY and the OU process, with parameter sets (4.33), (4.34).

of exercise opportunities. In Table 4.2, the column denoting "$P(N/2)$" gives the computed values of the Bermudan-style options with $\mathcal{M} = N/2$. For the values obtained with the Richardson extrapolation we use $\mathcal{M} = 16$ in (1.32) (so, $2\mathcal{M} = 32, 4\mathcal{M} = 64, 8\mathcal{M} = 128$).

The CGMY model is used here with the parameters r, C, G, M, Y, from (4.33) , and $T = 0.5, S_0 = 8, S_{min} = 10, S_{max} = 50, K_d = 20, K_a = 25$. As illus-

$n = \log_2 N$	$P(N/2)$		Richardson	
	option value	CPU time	option value	CPU time
7	137.423	0.27	137.395	0.59
8	137.408	0.53	137.390	0.99
9	137.399	2.00	137.390	1.79
10	137.394	8.39	137.390	3.40
11	137.392	39.55	137.390	6.68
12	137.391	203.27	137.390	13.21

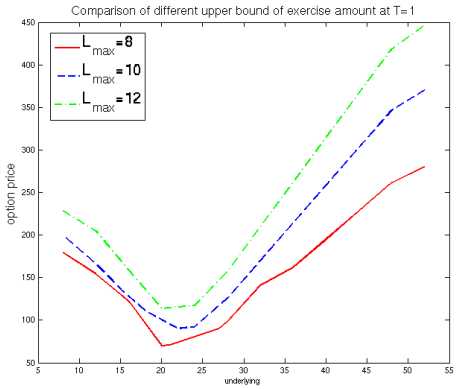
Table 4.2: Convergence over $\mathcal{M}$ and comparison between two approximation methods for American-style swing option.

trated in Table 4.2, to converge to an error of $O(10^{-4})$, one would require 203 seconds with the direct approximation method, and approximately one second with the extrapolation technique. The convergence observed here is in accordance with the behavior observed in [35].

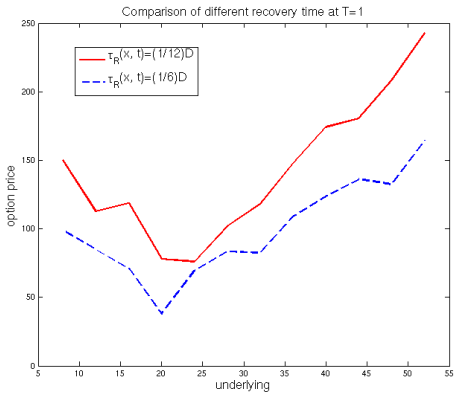
4.4.2 State-Dependent Recovery Time

We now consider the case where the recovery time, τ_R, depends on the amount D. We first use the CGMY model with the parameters from (4.33). Figure 4.6a compares the swing option prices with three upper bounds for D: $L = 8, 10, 12$. A higher upper bound results in higher option values, because a higher upper bound implies more possibilities for an option holder at each exercise date.

Figure 4.6b illustrates the influence of the recovery time on the swing option value. Here we compare $\tau_R(D) = \frac{1}{12}D$ with $\tau_R(D) = \frac{1}{6}D$, which



(a) Varying amount L



(b) Varying recovery time $\tau_R(D)$

Figure 4.6: CGMY process, $T-t=1$; Figure (a): Different values for L, and fixed $\tau_R(D,t)=\frac{1}{12}D$; Figure (b): Different Recovery time, and fixed $L=5$.

corresponds to one month (solid line) or two months (dashed line) penalty time for each unit exercised. Figure 4.6b shows that longer recovery time leads to lower option prices. In other words, if one can wait after exercising one can pay less for the swing option [3].

Table 4.3 shows how the option value and optimal value of D (i.e., D_{opt}) change over time. Here we take $L=8$, and $S_0=8$, a case where the option is deep in-the-money. As expected, jumps in the optimal D-values are observed at $t_n^*=T-n\tau_R(1)$.

Recovery time $\tau_R(D)=\frac{1}{12}D$ implies that if we exercise n or fewer units at t_n^*, we can exercise once more before expiry T, whereas if we exercise

[3]Similarly, smaller recovery times result in higher option prices with *constant recovery time*.

more than n units, we cannot exercise again before T. In other words, at t_n^*, $\phi_D^t > 0$ for $D \leq n$ and $\phi_D^t = 0$, otherwise.

Note that at the time points $t = T$ and $t = T - 1/24$, the optimal value equals $D_{opt} = L = 8$. For $t = T$ this is due to the arbitrage-free condition and the profit maximization principle, whereas for $T = t - 1/24$ the time left is so small that, in our present setting, there is only one opportunity left for a swing action ($\phi_D^t = 0$ for all D, n). One should then choose the largest D-value allowed for an optimal profit.

T-t	option value	D_{opt}	T-t	option value	D_{opt}
0	80	8	8/24	110.587	4
1/24	80	8	9/24	111.556	4
2/24	85.489	1	10/24	120.572	5
3/24	85.794	1	11/24	121.806	5
4/24	92.441	2	12/24	130.769	6
5/24	93.116	2	13/24	132.224	6
6/24	101.058	3	14/24	141.051	7
7/24	102.371	3	15/24	142.690	7

Table 4.3: D_{opt} over time $L = 8, S_0 = 8, \tau_R(D) = \frac{D}{12}$.

Figure 4.7 shows how D_{opt} changes with respect to the underlying price, with $L = 8, t = 0, \tau_R(D) = \frac{1}{12}D$. As S goes beyond K_d and K_a, D_{opt} tends to increase, because in this region the payoff, $g(x, t, D)$, dominates in the term $g(x, t, D) + \phi_D^t(x, t)$. Between $S = 20$ and $S = 25$, $D_{opt} = 0$, since $g(x, t, D) = 0$ for all $D > 0$ in this interval.

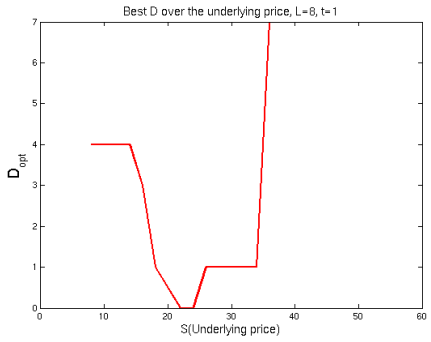


Figure 4.7: D_{opt} over underlying price, $L = 8, t = 0, \tau_R(D) = \frac{1}{12}D$

In a next experiment, the convergence of the swing option value with respect to parameter N, with the corresponding CPU time, for the CGMY and the OU processes, with $S_0 = 8, T = 1, M = 12$ and different upper bounds L, are presented in Tables 4.4 and 4.5, respectively. With $N = 256$

the swing option prices are accurate up-to a basis point for both stochastic processes. Tables 4.4 and 4.5 also indicate that the algorithm is flexible with respect to the variation in parameter L. Large L-values result in higher CPU times, because a larger number of early-exercise points needs to be determined, and many C_k- and G_k-terms have to be computed.

	N	128	256	512
L=5	option price	153.6884	150.0041	150.0041
	CPU time	0.3293	0.4569	0.7731
L=8	option price	177.2750	179.5152	179.5152
	CPU time	0.6914	1.1369	1.9020
L=10	option price	199.4206	199.6870	199.6870
	CPU time	1.0625	1.6609	2.9439

Table 4.4: Swing option values for CGMY process, dynamic recovery time, $S_0 = 8, T = 1, t = 0$.

	N	96	128	160
L=5	option price	145.5943	153.1150	153.1150
	CPU time	0.5256	0.7854	1.0180
L=8	option price	172.0567	172.0567	172.0567
	CPU time	0.9182	1.2263	1.5297
L=10	option price	196.9790	196.9790	196.9790
	CPU time	1.3039	1.6746	2.0252

Table 4.5: Swing option values for OU process, dynamic recovery time, $S_0 = 8, T = 1, t = 0$.

In the final experiment, we compare for American-style swing options with the state-dependent recovery time, the approximation obtained by the 4–point Richardson extrapolation with the direct approximation, obtained with Bermudan option values with increasing $\mathcal{M}$-values. We use the CGMY model with $Y = 0.5$ (other parameters as in (4.33)). Table 4.6 shows that the 4–point Richardson extrapolation is much more efficient than the direct approximation method, and that both methods converge to the same swing option values. Larger values of $\mathcal{M}$ give the same extrapolation result.

4.5 Conclusions

We have presented an efficient, flexible and robust pricing algorithm for swing options with early-exercise features, based on Fourier cosine series expansions and backward recursion. The algorithm performs nicely for different types of swing contract with flexibility in the upper bounds for the amount that can be exercised and recovery times. The pricing technique

Bermudan approximation $\mathcal{M} = N/2$			Richardson approximation $\mathcal{M}$=6 in Equation (1.31)		
N	option value	CPU time	N	option value	CPU time
128	93.9501	5.7391	64	93.9710	1.6077
256	93.9710	20.1821	128	93.9707	2.3621
512	93.9707	77.0859	256	93.9707	3.9196

Table 4.6: Comparison between two approximation methods for American-style swing options, CGMY model, $S_0 = 10, L = 5, Y = 0.5$.

is valid under different stochastic commodity processes, such as the CGMY process, other Lévy processes, as well as the OU process.

For the Lévy processes, the Fast Fourier Transform can be applied in the backward recursion procedure. This gives Bermudan-style swing option prices that are accurate up to a basis point in milli–seconds for constant recovery times, and in a fraction of a second ($L = 5$) to 1.7 seconds ($L = 10$) for the dynamic recovery time.

For OU processes, despite of the higher computational complexity (compared to Lévy processes), swing option prices can be obtained with basis point precision in less than 0.2 seconds for constant recovery times and within 2 seconds for dynamic recovery times. This is due to the exponential convergence rate of Fourier cosine series expansions.

The Richardson 4-point extrapolation technique has been used for pricing the American-style swing options in an efficient way.

CHAPTER 5

Efficient Pricing of Asian Options under Lévy Processes Based on Fourier Cosine Expansions Part I: European–Style Products

This chapter contains essentially the contents of paper [73].

5.1 Introduction

Asian options, introduced in 1987, belong to the class of path–dependent options. Their payoff is typically based on a geometric or arithmetic average of underlying asset prices at monitoring dates before maturity. The number of monitoring dates can be finite (*discretely–monitored*) or infinite (*continuously–monitored*). Volatility inherent in an asset is reduced due to the averaging feature, leading to cheaper options compared to plain vanilla option equivalents.

For geometric Asian options a closed–form solution under the Black–Scholes model has been presented in [45]. Other Lévy asset models have been studied in [37], resulting in an efficient valuation method based on the Fast Fourier Transform.

For arithmetic Asian options the prices have to be approximated numerically. Monte Carlo methods have been applied for this task, for example in [45]. An efficient PDE method for arithmetic Asian options which works particularly well for short maturities has been presented in [64].

Advanced pricing methods for options on the arithmetic average are based on a recursive integration procedure, in which the probability density

function of the log-return of the sum of asset prices is approximated, see [18, 4, 49, 37]. In [18, 4] an FFT and inverse FFT have been incorporated in the procedure to approximate the governing densities. The study in [18] focused on log–normally distributed underlying processes and required a fine grid to approximate the probability density function. This method is extended to more general densities in [4], where the size of the grid was reduced by re–centering the probability densities at each monitoring step, resulting in reduced CPU time. In [37] the FFT was used to approximate the density of the increment between consecutive monitoring dates, in combination with a series of recursive quadrature rules. The total computational complexity in [37] was $O(\mathcal{M}n^2)$, where $\mathcal{M}$ is the number of monitoring dates and n the number of points used in the quadrature. A recent contribution was presented in [19], where discretely sampled Asian options were priced via backward price convolutions.

Another pricing approach can be found in [27], where the governing densities were computed by a special Laplace inversion, for guaranteed return rate products, that can be seen as generalized discretely sampled Asian options. Alternatively, upper and lower bounds of the Asian option prices have been determined, for example in [49], for Lévy asset processes.

In this chapter we propose a different pricing method for Asian options and we name it the *ASCOS* method (Asian cosine method), as it is related to the COS method from [34, 35], see also Chapter 1. The method is also inspired by the work in [37], but there are some significant differences. Instead of recursively recovering the probability density of the logarithm of the sum of asset prices, as in [37], we recover the corresponding characteristic function by means of Fourier cosine expansions. The transitional density function is then in turn approximated in terms of the conditional characteristic function by a Fourier cosine expansion. The characteristic function for a Lévy process is known analytically and a Fourier cosine expansion most often exhibits exponential convergence.

Furthermore, the Clenshaw–Curtis quadrature rule is applied in the ASCOS method to approximate certain integrals appearing. We will perform an extensive error analysis to confirm the exponential convergence for Asian options.

The ASCOS pricing method can thus be seen as an efficient alternative to the FFT and convolution methods in [18, 37, 4, 49]. The method remains robust as the number of monitoring dates, $\mathcal{M}$, increases for arithmetic Asian options. In Section 5.2, the ASCOS method to price geometric Asian options under Lévy processes (discretely and continuously monitored) is presented. The pricing algorithm for arithmetic Asian options is then detailed in Section 5.3. A detailed error analysis is given in Section 5.4 and numerical results are presented in Section 5.5. We compare our results to those presented in [37]. The ASCOS method is extended to pricing American–style Asian options in the next chapter.

In this chapter, we focus on a fixed–strike Asian options. The extension to floating–strike Asian options follows directly from the symmetry between floating–strike and fixed–strike Asian options, as explained in [40] and [32].

5.2 ASCOS method for European-style geometric Asian options

The ASCOS pricing technique for geometric and arithmetic Asian options is described in Sections 5.2 and 5.3, respectively. In our method, the characteristic function of the geometric or arithmetic mean value of the underlying is recovered, which is then used to calculate the Asian option value by Fourier cosine expansions. For geometric Asian options, the characteristic function of the geometric mean can be calculated directly, as we will see below. The payoff function of a geometric Asian option with $\mathcal{M}$ monitoring dates and a fixed strike reads:

$$v(S,T) \equiv g(S) = \begin{cases} \max((\prod_{j=0}^{\mathcal{M}} S_j)^{\frac{1}{\mathcal{M}+1}} - K, 0), & \text{for a call,} \\ \max(K - (\prod_{j=0}^{\mathcal{M}} S_j)^{\frac{1}{\mathcal{M}+1}}, 0). & \text{for a put.} \end{cases}$$

Here S, K, $g(S)$ denote the stock price, the strike price and the payoff function, respectively.

For geometric Asian options, the characteristic function of the geometric mean can be calculated directly. The underlying process is transformed to the logarithm domain and we use the notation:

$$y := \log((\prod_{j=0}^{\mathcal{M}} S_j)^{\frac{1}{\mathcal{M}+1}}) = \frac{1}{\mathcal{M}+1} \sum_{j=0}^{\mathcal{M}} \log(S_j) =: \frac{1}{\mathcal{M}+1} \sum_{j=0}^{\mathcal{M}} x_j. \quad (5.1)$$

In order to use the Fourier cosine expansion, we need to determine the conditional characteristic function of y given x_0. From the definition of

characteristic function we have:

$$\begin{aligned}
\varphi(u;x_0,\Delta t) &= \mathbb{E}[\exp(iuy)|\mathcal{F}_0] = \mathbb{E}[\exp(iu\frac{1}{\mathcal{M}+1}\sum_{j=0}^{\mathcal{M}} x_j)|\mathcal{F}_0] \\
&= \mathbb{E}[\exp(iu(x_0+\frac{1}{\mathcal{M}+1}(\mathcal{M}(x_1-x_0)+(\mathcal{M}-1)(x_2-x_1) \\
&+ \cdots\{(\mathcal{M}-2)(x_3-x_2)+\ldots \\
&+ 2(x_{\mathcal{M}-1}-x_{\mathcal{M}-2})+(x_{\mathcal{M}}-x_{\mathcal{M}-1}))\}|\mathcal{F}_0))] \\
&= e^{iux_0}\mathbb{E}\left[\exp\left(i(u\frac{\mathcal{M}}{\mathcal{M}+1})(x_1-x_0)\right)|\mathcal{F}_0\right] \\
&\cdot \mathbb{E}\left[\exp\left(i(u\frac{\mathcal{M}-1}{\mathcal{M}+1})(x_2-x_1)\right)|\mathcal{F}_0\right]\cdot \\
&\quad \cdots\cdot\mathbb{E}\left[\exp\left(i(u\frac{1}{\mathcal{M}+1})(x_{\mathcal{M}}-x_{\mathcal{M}-1})\right)|\mathcal{F}_0\right]. \qquad (5.2)
\end{aligned}$$

The last step is due to the fact that Lévy processes have independent increments. A Lévy process also has stationary increments, which implies that the increments x_1-x_0, x_2-x_1, $\cdots$, $x_{\mathcal{M}}-x_{\mathcal{M}-1}$ are identically distributed, and they are all independent of x_0. Denoting the (identical) characteristic functions of these increments by $\varphi(u,t)$, and substitution of $\varphi(u,t)$ into (5.2) gives the characteristic function of y given x_0:

$$\varphi(u;x_0,\Delta t) = e^{iux_0}\cdot\prod_{j=1}^{\mathcal{M}}\phi\left(u\frac{\mathcal{M}+1-j}{\mathcal{M}+1};\frac{T-t_0}{\mathcal{M}}\right), \qquad (5.3)$$

where $\phi(u;t)$ is as defined in (1.27).

Substitution of characteristic function (5.3) into (1.14) results in the *ASCOS pricing formula for European-style geometric Asian options*, with the underlying asset modeled by a Lévy process:

$$v(x_0,t_0) = e^{-r\Delta t}\sum_{k=0}^{N-1}{}'\,Re\left(\varphi(\frac{k\pi}{b-a};x_0,\Delta t)e^{-ik\pi\frac{a}{b-a}}\right)V_k, \qquad (5.4)$$

where

$$V_k = \begin{cases} \frac{2}{b-a}(\chi_k(\log(K),b)-K\psi_k(\log(K),b)), & \text{for a call,} \\ \frac{2}{b-a}(K\psi_k(a,\log(K))-\chi_k(a,\log(K))), & \text{for a put,} \end{cases}$$

with

$$\begin{aligned}
\chi_k(x_1,x_2) &:= \int_{x_1}^{x_2} e^y\cos\left(k\pi\frac{y-a}{b-a}\right)dy, \\
\psi_k(x_1,x_2) &:= \int_{x_1}^{x_2}\cos\left(k\pi\frac{y-a}{b-a}\right)dy, \qquad (5.5)
\end{aligned}$$

which are known analytically.

The computational complexity to get the characteristic function for each $u = k\pi/b - a,\ k = 0, \cdots, N-1$ is linear in $\mathcal{M}$, so that $O(\mathcal{M}N)$ computations are required. The complexity of the work in (5.4) is linear in N, so that the total computational complexity of the method is $O(\mathcal{M}N)$.

For geometric Asian options there is no error in deriving the characteristic function by (5.2) and (5.3). The only errors made are due to the COS formula (5.4). Detailed error analysis of the COS method for European options can be found in [34]. The ASCOS pricing method for geometric Asian options under Lévy processes is thus expected to have an exponential convergence rate in the number of cosine terms, for all density functions that satisfy $f(y|x) \in C^{\infty}([a,b] \subset \mathbb{R})$.

5.3 ASCOS method for arithmetic Asian options

For arithmetic Asian options, the characteristic function of the arithmetic mean will be derived recursively by Fourier cosine expansions and Clenshaw–Curtis quadrature. The Fourier cosine expansion is used each time step (i.e. at each monitoring date), whereas the Clenshaw–Curtis quadrature rule is used once at the beginning of the computation. In Subsection 5.2 the characteristic function of the geometric average was recovered, which was explicitly a function of $x_0 = \log(S_0)$, as $x_0 + ... + x_T$ is a function of x_0, so that the characteristic function took the form $\varphi(u; x_0, \Delta t)$. In the present section, we recover the characteristic function of the sum of Lévy asset price increments, which is independent of x_0. Therefore, we write the characteristic function here in the form $\phi(u; \Delta t)$, like the Levy component $\phi(u,t)$ defined in (1.27), rather than $\varphi(u; x_0, \Delta t)$.

The payoff function of an arithmetic Asian options reads:

$$v(S,T) \equiv g(S) = \begin{cases} \max(\frac{1}{\mathcal{M}+1}\sum_{j=0}^{\mathcal{M}} S_j - K, 0), & \text{for a call,} \\ \max(K - \frac{1}{\mathcal{M}+1}\sum_{j=0}^{\mathcal{M}} S_j, 0), & \text{for a put.} \end{cases} \tag{5.6}$$

We denote by n_q the number of terms in the Clenshaw–Curtis quadrature (q stands for quadrature).

We first explain the recursion procedure to recover the characteristic function of the arithmetic mean value of the underlying, where we denote by:

$$R_j := \log\left(\frac{S_j}{S_{j-1}}\right), \quad j = 1, \cdots, \mathcal{M}. \tag{5.7}$$

For Lévy processes, the increments $R_j,\ j = 1, \cdots, \mathcal{M}$ are identically and independently distributed, so that $R_j \overset{d}{=} R$. Then, $\forall u, j$, we have $\phi_{R_j}(u; \Delta t) =$

$\phi_R(u;\Delta t)$. Characteristic function $\phi_R(u;\Delta t)$ is known in closed form for different Lévy processes.

A stochastic process, Y_j, is introduced where $Y_1 = R_{\mathcal{M}}$ and for $j = 2,\cdots,\mathcal{M}$, we have

$$Y_j := R_{\mathcal{M}+1-j} + \log(1+\exp(Y_{j-1})). \tag{5.8}$$

We denote by $Z_j := \log(1+\exp(Y_j)), \forall j$, so that (5.8) is rewritten as

$$Y_j := R_{\mathcal{M}+1-j} + Z_{j-1}. \tag{5.9}$$

In this setting Y_j admits the form

$$Y_j = \log\left(\frac{S_{\mathcal{M}-j+1}}{S_{\mathcal{M}-j}} + \frac{S_{\mathcal{M}-j+2}}{S_{\mathcal{M}-j}} + \cdots + \frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}}\right), \tag{5.10}$$

and we have that

$$\frac{1}{\mathcal{M}+1}\sum_{j=0}^{\mathcal{M}} S_j = \frac{(1+\exp(Y_{\mathcal{M}}))S_0}{\mathcal{M}+1}. \tag{5.11}$$

Convolution scheme (5.9) has already been used in [18, 4, 37], in combination with other numerical methods, to recover the probability density function of $Y_{\mathcal{M}}$. Here, however, we will recover the *characteristic function of* $Y_{\mathcal{M}}$ instead, by a forward recursion procedure, which is then used in turn to recover the density of the European-style arithmetic mean of the underlying process in the risk–neutral formula (5.12). The arithmetic Asian option value is now defined as:

$$v(x_0,t_0) = e^{-r\Delta t}\int_{-\infty}^{\infty} v(y,T) f_{Y_{\mathcal{M}}}(y)dy. \tag{5.12}$$

By (5.11), $v(y,T)$ in (5.12) is of the following form:

$$v(y,T) = \begin{cases} \left(\dfrac{S_0(1+\exp(y))}{\mathcal{M}+1} - K\right)^+, & \text{for a call,} \\ \left(K - \dfrac{S_0(1+\exp(y))}{\mathcal{M}+1}\right)^+, & \text{for a put.} \end{cases}$$

5.3.1 Recovery of characteristic function

To recover the characteristic function of $Y_{\mathcal{M}}$, i.e. $\phi_{Y_{\mathcal{M}}}(u;\Delta t)$, we start with Y_1, for which the characteristic function reads:

$$\phi_{Y_1}(u;\Delta t) = \phi_R(u;\Delta t). \tag{5.13}$$

Then, at time step t_j, $j = 2,\cdots,\mathcal{M}$, $\phi_{Y_j}(u;\Delta t)$ can be recovered in terms of $\phi_{Y_{j-1}}(u;\Delta t)$. This is done by application of (5.9) and the fact that Lévy

processes have independent increments. This implies that $\forall j$, $R_{\mathcal{M}+1-j}$ and Z_{j-1} are independent, which gives us:

$$\phi_{Y_j}(u;\Delta t) = \phi_{R_{\mathcal{M}+1-j}}(u;\Delta t)\phi_{Z_{j-1}}(u;\Delta t) = \phi_R(u;\Delta t)\phi_{Z_{j-1}}(u;\Delta t). \quad (5.14)$$

From the definition of characteristic function, we have

$$\phi_{Z_{j-1}}(u;\Delta t) = \mathbb{E}[e^{iu\log(1+\exp(Y_{j-1}))}] = \int_{-\infty}^{\infty}(e^x+1)^{iu}f_{Y_{j-1}}(x)dx. \quad (5.15)$$

To apply the Fourier cosine series expansion to *approximate* the characteristic function, we first truncate the integration range

$$\hat{\phi}_{Z_{j-1}}(u;\Delta t) = \int_a^b (e^x+1)^{iu}f_{Y_{j-1}}(x)dx. \quad (5.16)$$

If we define the following error

$$\epsilon_T(X) := \int_{\mathbb{R}\setminus[a,b]} f_X(x)dx,$$

then, as $\forall j, u \in \mathbb{R}$,

$$|(e^x+1)^{iu}| = |\cos(u\log(1+e^x) + i\sin(u\log(1+e^x)))| = 1, \quad (5.17)$$

the error in (5.16) can be bounded by:

$$|\int_{\mathbb{R}\setminus[a,b]}(e^x+1)^{iu}f_{Y_{j-1}}(x)dx| \le \int_{\mathbb{R}\setminus[a,b]} f_{Y_{j-1}}(x)dx := \epsilon_T(Y_{j-1}). \quad (5.18)$$

We apply the Fourier cosine expansion to approximate $f_{Y_{j-1}}(x)$, giving:

$$\begin{aligned}\hat{\phi}_{Z_{j-1}}(u;\Delta t) &= \frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\hat{\phi}_{Y_{j-1}}(\frac{l\pi}{b-a};\Delta t)\exp(-ia\frac{l\pi}{b-a})\right) \\ &\cdot \int_a^b (e^x+1)^{iu}\cos\left((x-a)\frac{l\pi}{b-a}\right)dx, \end{aligned} \quad (5.19)$$

where $\hat{\phi}_{Y_{j-1}}$ is an approximation of $\phi_{Y_{j-1}}$.

In this way, $\hat{\phi}_{Z_{j-1}}$ is recovered in terms of $\hat{\phi}_{Y_{j-1}}$. Application of (5.14) gives us an approximation $\hat{\phi}_{Y_j}(u;\Delta t)$ for any u. Equation (5.19) can be written in matrix–vector form:

$$\Phi_{j-1} = \mathcal{H}A_{j-1}, \quad (5.20)$$

using:

$$\begin{aligned}
\Phi_{j-1} &= (\Phi_{j-1}(k))_{k=0}^{N-1}, \quad \Phi_{j-1}(k) = \hat{\phi}_{Z_{j-1}}(u_k;\Delta t),\\
u_k &= \frac{k\pi}{b-a},\ k=0,\cdots,N-1,\\
\mathcal{H} &= (\mathcal{H}(k,l))_{k,l=0}^{N-1}, \quad \mathcal{H}(k,l) = \int_a^b (e^x+1)^{iu_k}\cos((x-a)u_l)dx,\\
A_j &= \frac{2}{b-a}(A_j(l))_{l=0}^{N-1}, \quad A_j(l) = Re(\hat{\phi}_{Y_{j-1}}(u_l;\Delta t)\exp(-iau_l)).
\end{aligned}$$

By the recursion procedure in (5.14) and (5.20), the characteristic function, $\phi_{Y_{\mathcal{M}}}(u;\Delta t)$, can be approximated by $\hat{\phi}_{Y_{\mathcal{M}}}(u;\Delta t)$ efficiently. Application of (1.14) in (5.12) finally gives us the European-style arithmetic Asian option value:

$$\hat{v}(x,t_0) = e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re\left(\hat{\phi}_{Y_{\mathcal{M}}}(\frac{k\pi}{b-a};\Delta t)e^{-ik\pi\frac{a}{b-a}}\right)V_k, \tag{5.21}$$

in which

$$V_k = \begin{cases} \frac{2}{b-a}\left(\frac{S_0}{\mathcal{M}+1}\chi_k(x^*,b) + (\frac{S_0}{\mathcal{M}+1}-K)\psi_k(x^*,b)\right), & \text{for a call,}\\ \frac{2}{b-a}\left((K-\frac{S_0}{\mathcal{M}+1})\psi(a,x^*) - \frac{S_0}{\mathcal{M}+1}\chi(a,x^*)\right), & \text{for a put.}\end{cases} \tag{5.22}$$

Functions $\chi_k(x_1,x_2)$, $\psi_k(x_1,x_2)$ are as in (5.5), and $x^* = \log(\frac{K(\mathcal{M}+1)}{S_0}-1)$.

5.3.2 Integration range

Here we explain how to determine integration range $[a,b]$, so that the error $\epsilon_T(Y_{j-1})$, $j=2,\cdots,\mathcal{M}$ in (5.18) as well as truncation error $\epsilon_T(Y_{\mathcal{M}})$ in (5.21) can be controlled. In [34, 35], the integration range for each $Y_j, j=1,\cdots,\mathcal{M}$, was determined by means of the cumulants, as:

$$[\xi_1(Y_j) - L\sqrt{\xi_2(Y_j)+\sqrt{\xi_4(Y_j)}}, \xi_1(Y_j) + L\sqrt{\xi_2(Y_j)+\sqrt{\xi_4(Y_j)}}], \tag{5.23}$$

where $\xi_1(Y_j), \xi_2(Y_j), \xi_4(Y_j)$ are the first, second and fourth cumulant of Y_j, respectively. It is rather expensive to determine these cumulants, and therefore we propose a different integration range, which is very similar to (5.23). For a Lévy process, the cumulants of the increments, $\log(S_l/S_k)$ $\forall l > k$, are linearly increasing functions of $t := (l-k)\Delta t$. Therefore, for

$Y_j, j = 1, \cdots, \mathcal{M}$, as defined in (5.10), we have

$$\begin{aligned}
&\xi_1(\log(j\frac{S_{\mathcal{M}-j+1}}{S_{\mathcal{M}-j}})) \le \xi_1(Y_j) \le \xi_1(\log(j\frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}})),\\
&0 \le \xi_2(Y_j) \le \xi_2(\log(j\frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}})),\\
&0 \le \xi_4(Y_j) \le \xi_4(\log(j\frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}})).
\end{aligned}$$

Denoting

$$\begin{aligned}
a_j &:= \xi_1(\log(j\frac{S_{\mathcal{M}-j+1}}{S_{\mathcal{M}-j}})) - L\sqrt{\xi_2(\log(j\frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}})) + \sqrt{\xi_4(\log(j\frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}}))}},\\
b_j &:= \xi_1(\log(j\frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}})) + L\sqrt{\xi_2(\log(j\frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}})) + \sqrt{\xi_4(\log(j\frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}}))}},
\end{aligned} \tag{5.24}$$

we can define a suitable interval $[a_j, b_j]$. Note that the cumulants of $\log(j\frac{S_{\mathcal{M}-j+1}}{S_{\mathcal{M}-j}})$ and $\log(j\frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}})$ in (5.24) are known in closed form for Lévy processes, that is, for $n = 1$

$$\begin{aligned}
\xi_1(\log(j\frac{S_{\mathcal{M}-j+1}}{S_{\mathcal{M}-j}})) &= log(j) + \xi_1(R),\\
\xi_1(\log(j\frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}})) &= log(j) + j\xi_1(R),
\end{aligned}$$

and $\forall n \ge 2$,

$$\begin{aligned}
\xi_n(\log(j\frac{S_{\mathcal{M}-j+1}}{S_{\mathcal{M}-j}})) &= \xi_n(R),\\
\xi_n(\log(j\frac{S_{\mathcal{M}}}{S_{\mathcal{M}-j}})) &= j\xi_n(R),
\end{aligned}$$

with R the logarithm of the increment of a Lévy process.

In order to compute the integration in (5.19) only once, we adopt the following integration range:

$$[a, b] := [\min_{j=1,\cdots,\mathcal{M}} a_j, \max_{j=1,\cdots,\mathcal{M}} b_j], \tag{5.25}$$

for all time steps, so that the truncation error $\epsilon_T(Y_j), \forall j$ can be controlled easily.

In accordance with [34, 35], we will use $L = 10 \sim 12$ in (5.24), in our numerical experiments.

5.3.3 Clenshaw–Curtis quadrature

We discuss the efficient computation of matrix $\mathcal{H}$ in (5.20). An important feature is that matrix $\mathcal{H}$ remains constant for all time steps $t_j, j = 1, \cdots, \mathcal{M}-1$, so that we need to calculate it only once. Its elements are given by:

$$\mathcal{H}(k,l) = \int_a^b (e^x+1)^{iu_k} \cos((x-a)u_l)dx, \quad k,l = 0, \cdots N-1, \tag{5.26}$$

which can be rewritten in terms of incomplete Beta functions, as follows

$$\begin{aligned} & \int_a^b (e^x+1)^{i\frac{k\pi}{b-a}} \cos((x-a)\frac{l\pi}{b-a})dx \\ = \; & \frac{1}{2}e^{-\frac{l(ia+\pi)}{d}}(e^{\frac{2ial}{d}}(-\beta(-e^a, -\frac{il}{d}, 1+\frac{ik}{d}) + \beta(-e^b, -\frac{il}{d}, 1+\frac{ik}{d})) \\ + \; & e^{\frac{2l\pi}{d}}(-\beta(-e^a, \frac{il}{d}, 1+\frac{ik}{d}) + \beta(-e^b, \frac{il}{d}, 1+\frac{ik}{d}))), \end{aligned} \tag{5.27}$$

where $i = \sqrt{-1}$, $d = \frac{b-a}{\pi}$ and $\beta(x,y,z)$ is the incomplete Beta function

$$\beta(x,y,z) = \int_0^x t^{y-1}(1-t)^{z-1}dt.$$

The computation of the incomplete Beta functions in (5.27) is involved with these complex-valued arguments. Here, Eq. (5.26) is approximated numerically by the Clenshaw–Curtis quadrature rule, which is based on an expansion of the integrand in terms of Chebyshev polynomials (as proposed in [22]; more information can be found in [9]).

Although both the Clenshaw–Curtis and the Gaussian quadrature rules exhibit an exponential convergence for the integrand in (5.26), the Clenshaw–Curtis quadrature is preferred here since it is computationally cheaper. The weights and nodes of the Clenshaw-Curtis quadrature are easy to determine. Moreover, Clenshaw–Curtis quadrature is *a nested integration rule*, where the nodes for a small value of N are also nodes for larger N-values.

To use the Clenshaw–Curtis rule for (5.26), we first change the integration interval from $[a,b]$ to $[-1,1]$

$$\begin{aligned} & \int_a^b (e^x+1)^{iu_k} \cos\left((x-a)u_l\right)dx = \\ & \int_{-1}^1 \frac{b-a}{2}\left(\exp\left(\frac{b-a}{2}x + \frac{a+b}{2}\right)+1\right)^{iu_k} \cos\left(\left(\frac{b-a}{2}x + \frac{a+b}{2} - a\right)u_l\right)dx. \end{aligned}$$

The integral can then be approximated as follows

$$\int_a^b (e^x+1)^{iu_k} \cos((x-a)u_l)dx \approx (D^T d)^T y =: w^T y, \tag{5.28}$$

where D is an $(n_q/2+1)\times(n_q/2+1)$ matrix, whose elements read

$$D(k,n)=\frac{2}{n_q}\cos\left(\frac{(n-1)(k-1)\pi}{n_q/2}\right)\cdot\begin{cases}1/2, & \text{if } n=\{1,n_q/2+1\},\\ 1, & \text{otherwise.}\end{cases} \tag{5.29}$$

The vector d and the elements y_n in $y=\{y_n\}_{n=0}^{n_q/2}$ are defined as:

$$\begin{aligned} d &:= \left(1,\frac{2}{(1-4)},\frac{2}{(1-16)},\cdots,\frac{2}{(1-(n_q-2)^2)},\frac{1}{(1-n_q^2)}\right)^T,\\ y_n &:= f(\cos(\frac{n\pi}{n_q}))+f(-\cos(\frac{n\pi}{n_q})), \end{aligned} \tag{5.30}$$

where in our case

$$f(x)=\frac{b-a}{2}\left(\exp(\frac{b-a}{2}x+\frac{a+b}{2})+1\right)^{iu_k}\cos\left((\frac{b-a}{2}x+\frac{a+b}{2}-a)u_l\right).$$

For all (k,l), the vector $w=D^Td$ remains the same, so that it needs to be computed only once, for all (k,l). Because D^Td is a type I discrete cosine transform, the computational complexity is $O(n_q\log_2 n_q)$. Elements y_n must be calculated for each pair (k,l), with complexity $O(n_q)$ and the computational complexity, for all (k,l), is therefore $O(n_qN^2)$. When using the Clenshaw–Curtis quadrature rule to compute matrix $\mathcal{H}$ (only once, used for all time steps), the total computational complexity is thus $O(n_q\log_2 n_q)+O(n_qN^2)$.

Furthermore, at each time step t_j, we need $O(N^2)$ computations for the matrix–vector multiplication (5.20) and $O(N)$ computations to obtain $\hat{\phi}_{Y_j}$ by equation (5.13) or (5.14). The computational complexity for this task is thus $O(\mathcal{M}N^2)$.

The overall computational complexity of our method for arithmetic Asian options is then $O(n_q\log_2 n_q)+O(n_qN^2)+O(\mathcal{M}N^2)$. The number N^2 is in practice much larger than $\log_2 n_q$. The overall complexity is then of order $O((n_q+\mathcal{M})N^2)$.

We will show in the section on error analysis for arithmetic Asian options that for most of the Lévy processes, the Fourier cosine expansion exhibits an exponential convergence rate with respect to N. For the integrand in (5.26) the Clenshaw–Curtis quadrature converges exponentially with respect to n_q. Therefore, the ASCOS pricing method is an efficient alternative to the method proposed in [37], which requires $O(\mathcal{M}\bar{N}^2)$ computations ($\bar{N}$ being the number of points used in the quadrature in [37]), with $\bar{N}>n_q$, and $\bar{N}>N$, for the same level of accuracy. Our pricing method is especially advantageous when the number of monitoring dates, $\mathcal{M}$, increases. The method is summarized below.

ASCOS Algorithm: Pricing European-style arithmetic Asian options.

Initialization

- Use Clenshaw–Curtis quadrature (5.28) to compute $\mathcal{H} = (\mathcal{H}(k,l))$, $k,l = 0,\cdots,N-1$, with $\mathcal{H}$ in (5.20), (5.26).
- Compute $\phi_R(u_k;\Delta t), k = 0,\cdots,N-1$.
- Set $\phi_{Y_1}(u_k;\Delta t) = \phi_R(u_k;\Delta t)$.

Main Loop to Recover $\hat{\phi}_{Y_{\mathcal{M}}}$: For $j = 2$ to $\mathcal{M}$,

- Compute the vector Φ_{j-1} with elements $\hat{\phi}_{Z_{j-1}}(u_k;\Delta t)$, $k = 0,\cdots,N-1$ using (5.20).
- Recover $\hat{\phi}_{Y_j}(u_k;\Delta t)$, $k = 0,\cdots,N-1$ using (5.14).

Final step:

- Compute $\hat{v}(x_0,t_0)$ by inserting $\hat{\phi}_{Y_{\mathcal{M}}}(u_k;\Delta t)$, $k = 0,\cdots,N-1$ into (5.21).

5.3.4 Extensions

In a series of remarks, we now discuss some other generalizations of the ASCOS method.

Remark 5.3.1 (Continuously–monitored Asian options)**.** *The option values of continuously–monitored arithmetic Asian options, with payoff*

$$v(S,T) = g(S) = \begin{cases} \left(\frac{1}{T}\int_0^T S(t)dt - K\right)^+, & \text{for a call,} \\ \left(K - \frac{1}{T}\int_0^T S(t)dt\right)^+, & \text{for a put,} \end{cases}$$

can be obtained from discretely–monitored arithmetic Asian option prices by a four-point Richardson extrapolation.

Let $\hat{v}(\mathcal{M})$ denote the computed value of a discretely–monitored Asian option with $\mathcal{M}$ monitoring dates. The continuously–monitored Asian option value, denoted by $\hat{v}_\infty$, is approximated by a four-point Richardson extrapolation scheme, as follows:

$$\hat{v}_\infty(d) = \frac{1}{21}(64\hat{v}(2^{d+3}) - 56\hat{v}(2^{d+2}) + 14\hat{v}(2^{d+1}) - \hat{v}(2^d)). \tag{5.31}$$

The same technique can be applied for continuously monitored geometric Asian options.

Remark 5.3.2 (Asian options on the harmonic average)**.** *Asian options with a payoff based on the harmonic average, i.e. on $\mathcal{M}/(\sum_{j=1}^{\mathcal{M}} 1/S_j)$, can be*

priced in a similar fashion as explained above by the ASCOS method. First, we recover the characteristic function of a variable $y = \log(\sum_{j=1}^{m} S_0/S_j)$ *recursively; then we insert the approximation into the COS pricing formula. We define* $\bar{R}_j = \log(S_{j-1}/S_j)$*. Starting with* $Y_1 = \log(\bar{R}_{\mathcal{M}})$*, we find,* $\forall j, u$*:*

$$\phi_{\bar{R}_j}(u;\Delta t) = \mathbb{E}[e^{iu\log(\frac{S_{j-1}}{S_j})}] = \mathbb{E}[e^{i(-u)\log(\frac{S_j}{S_{j-1}})}] = \phi_{R_j}(-u;\Delta t), \quad (5.32)$$

with ϕ_{R_j} *available in closed form for Lévy processes. For this reason,* $\phi_{Y_1}(u;\Delta t)$ *is also known analytically.*

For $j = 2, \cdots, \mathcal{M}$ *we then define* $Y_j := \bar{R}_{\mathcal{M}+1-j} + Z_{j-1}$*, where* $Z_j := \log(1+\exp(Y_j))$*. In this setting we have* $Y_{\mathcal{M}} \equiv \log(\sum_{j=1}^{m} S_0/S_j)$*.*

Again, $\bar{R}_{\mathcal{M}+1-j}$ *and* Z_{j-1} *are independent at each time step, due to the properties of Lévy processes. Therefore*

$$\phi_{Y_j}(u;\Delta t) = \phi_{\bar{R}_{\mathcal{M}+1-j}}(u;\Delta t)\phi_{Z_{j-1}}(u;\Delta t),\ \forall u,$$

where $\phi_{\bar{R}_{\mathcal{M}+1-j}}(u;\Delta t)$ *is known analytically from (5.32) and* $\phi_{Z_{j-1}}(u;\Delta t)$ *can be recovered, as* $\hat{\phi}_{Z_{j-1}}(u;\Delta t)$ *from* $\hat{\phi}_{Y_{j-1}}(u;\Delta t)$ *by Fourier cosine expansions and Clenshaw–Curtis quadrature, as in (5.19). We thus approximate the characteristic function of* $Y_{\mathcal{M}}$ *and the fixed strike Asian option value is given by:*

$$\hat{v}(x,t_0) = e^{-r\Delta t}\sum_{k=0}^{N-1}{}' Re\left(\hat{\phi}_{Y_{\mathcal{M}}}(\frac{k\pi}{b-a};\Delta t)e^{-ik\pi\frac{a}{b-a}}\right)V_k,$$

in which

$$V_k = \begin{cases} \frac{2}{b-a}(\mathcal{M}S_0\bar{\chi}_k(x^*,b) - K\psi_k(x^*,b)), & \text{for a call,} \\ \frac{2}{b-a}(K\psi(a,x^*) - \mathcal{M}S_0\bar{\chi}(a,x^*)), & \text{for a put,} \end{cases}$$

where $x^* = \log(\mathcal{M}S_0/K)$*,* $\bar{\chi}(x_1,x_2) := \int_{x_1}^{x_2} e^{-y}\cos(k\pi\frac{y-a}{b-a})dy$*, and* $\psi_k(x_1,x_2)$ *is defined in (5.5).*

Finally, the symmetry between floating and fixed–strike Asian options also holds for Asian options on the harmonic average, so that floating strike options can be valued as well.

Remark 5.3.3 (A special case: the forward contract)**.** *A forward contract, as often encountered in commodity markets, may be defined by the payoff:*

$$g(S) = \frac{1}{\mathcal{M}+1}\sum_{j=0}^{\mathcal{M}} S_j - K. \quad (5.33)$$

The contract value then reads

$$\begin{aligned} v(x_0,t_0) &= e^{-r\Delta t}\mathbb{E}\left[\frac{1}{\mathcal{M}+1}\sum_{j=0}^{\mathcal{M}} S_j - K\right] \\ &= e^{-r\Delta t}\left(\frac{S_0}{\mathcal{M}+1}\mathbb{E}[e^{Y_{\mathcal{M}}}] + (\frac{S_0}{\mathcal{M}+1} - K)\right), \end{aligned} \tag{5.34}$$

where the last step follows from (5.11). The expected value of $\exp(Y_{\mathcal{M}})$ *can be obtained by a forward recursion procedure. At each monitoring date,* t_j*, we have from (5.9) that*

$$\mathbb{E}[e^{Y_j}] = \mathbb{E}[e^{R_{\mathcal{M}+1-j}}(1+e^{Y_{j-1}})]. \tag{5.35}$$

For Lévy processes $R_{\mathcal{M}+1-j}$ *and* $(1+\exp(Y_{j-1}))$ *are independent and* $R_j \overset{d}{=} R$*,* $\forall j$*, so that equation (5.35) reads:*

$$\mathbb{E}[e^{Y_j}] = \mathbb{E}[e^R](1+\mathbb{E}[e^{Y_{j-1}}]),\ \forall j, \tag{5.36}$$

with $\mathbb{E}[e^{Y_1}] \equiv \mathbb{E}[e^R]$*. The value of* $\mathbb{E}[e^R]$ *reads*

$$\mathbb{E}[e^R] = \int_{-\infty}^{\infty} e^y f_R(y)dy = \sum_{k=0}^{N-1}{}' Re\left(\phi_R(\frac{k\pi}{b-a};\Delta t)e^{-ik\pi\frac{a}{b-a}}\right)\chi_k(a,b),$$

where function $\chi_k(x_1,x_2)$ *is defined in (5.5) and* ϕ_R *is the characteristic function of* R*, which is available for various Lévy processes.*

The $\mathbb{E}[e^R]$*-term needs to be calculated only once, with* $O(N)$ *complexity. In the recursion procedure to get the forward value, we use (5.36)* $\mathcal{M}-1$ *times and (5.34) once. Therefore, the total computational complexity is* $O(N)+O(\mathcal{M})$*, and exponential convergence is expected for probability density functions belonging to* $C^\infty[a,b]$*.*

5.4 Error analysis for arithmetic Asian options

Here we give an error analysis of the ASCOS method for arithmetic Asian options. We first discuss, in general terms, three types of error occurring, i.e., the truncation error, ϵ_T, the error of the Fourier cosine expansion, ϵ_F, and the error from the use of the Clenshaw–Curtis quadrature, ϵ_Q.

The truncation error was defined as

$$\epsilon_T(Y_j) := \int_{\mathbb{R}\setminus[a,b]} f_{Y_j}(y)dy, \quad j=1,\cdots,\mathcal{M}, \tag{5.37}$$

and it decreases as interval $[a,b]$ increases. In other words, for a sufficiently large integration range $[a,b]$, this part of the error won't dominate the overall error of the arithmetic Asian option price.

Regarding the error of the Fourier cosine expansions, we know from [34] that for $f(y|x) \in \mathbf{C}^\infty[a,b]$, it can be bounded by

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

with $\nu > 0$ a constant and a term $P^*(N)$, which varies less than exponentially with respect to N.

When the probability density function has a discontinuous derivative, the error can be bounded by

$$|\epsilon_F(N,[a,b])| \leq \frac{\bar{P}^*(N)}{(N-1)^{\beta-1}},$$

where $\bar{P}^*(N)$ is a constant and $\beta \geq 1$.

Error ϵ_F decays thus either exponentially with respect to N, if the density function $f(y|x) \in \mathbf{C}^\infty[a,b]$, or algebraically.

Let us now have a look at the error from the Clenshaw–Curtis quadrature, which we use to approximate

$$I := \int_a^b (e^x+1)^{iu_k}\cos((x-a)u_l)dx, \tag{5.38}$$

by $\hat{I} := w^T y$ in (5.28). In other words, $\epsilon_q = I - \hat{I}$.

According to [62, 65], the Clenshaw–Curtis quadrature rule exhibits an error which can be bounded by $O((2n_q)^{-k}/k)$, for a k–times differentiable integrand. When k is bounded, we have algebraic convergence; otherwise the error converges exponentially with respect to n_q, see also [8]. The integrand in (5.38) belongs to $C^\infty[a,b]$, as all derivatives are continuous on any interval $[a,b]$, confirming that, for the integrand in (5.38), we will have exponential convergence with respect to n_q.

5.4.1 Error propagation in the characteristic functions

The following lemma is used in the error analysis.

Lemma 5.4.1. *For any random variable, X, and any $u \in \mathbb{R}$, the characteristic function can be bounded by $|\phi_X(u;\Delta t)| \leq 1$.*

Proof. For any X and u, the characteristic function $\phi_X(u;\Delta t)$ is defined by:

$$\phi_X(u;\Delta t) = \mathbb{E}[e^{iuX}] = \int_{-\infty}^{\infty} e^{iux} f(x)dx.$$

We have

$$|\phi_X(u;\Delta t)| \leq \int_{-\infty}^{\infty} |e^{iux}| f(x)dx,$$

and thus:

$$|\phi_X(u;\Delta t)| \leq \int_{-\infty}^{\infty} f(x)dx = 1.$$

□

Now we start with the error analysis, and denote by $\epsilon(\hat{\phi}_{Y_m}(u;\Delta t))$ and $\epsilon(\hat{\phi}_{Z_m}(u;\Delta t))$, $m = 1,\cdots,\mathcal{M}$, the errors in $\hat{\phi}_{Y_m}(u;\Delta t)$ and $\hat{\phi}_{Z_m}(u;\Delta t)$, respectively. From (5.21) the error in the arithmetic Asian option price, denoted by ϵ, is given by

$$\begin{aligned}
\epsilon &= e^{-r\Delta t}\int_{-\infty}^{\infty} v(y,T) f_{Y_{\mathcal{M}}}(y)dy - e^{-r\Delta t}{\sum_{k=0}^{N-1}}' Re\left(\hat{\phi}_{Y_{\mathcal{M}}}(\frac{k\pi}{b-a};\Delta t)e^{-ik\pi\frac{a}{b-a}}\right)V_k \\
&= e^{-r\Delta t}\int_{-\infty}^{\infty} v(y,T) f_{Y_{\mathcal{M}}}(y)dy - e^{-r\Delta t}{\sum_{k=0}^{N-1}}' Re\left(\phi_{Y_{\mathcal{M}}}(\frac{k\pi}{b-a};\Delta t)e^{-ik\pi\frac{a}{b-a}}\right)V_k \\
&+ e^{-r\Delta t}{\sum_{k=0}^{N-1}}' Re\left((\phi_{Y_{\mathcal{M}}}(\frac{k\pi}{b-a};\Delta t) - \hat{\phi}_{Y_{\mathcal{M}}}(\frac{k\pi}{b-a};\Delta t))e^{-ik\pi\frac{a}{b-a}}\right)V_k \\
&= \epsilon_{\cos} + e^{-r\Delta t}{\sum_{k=0}^{N-1}}' Re\left(\epsilon(\hat{\phi}_{Y_{\mathcal{M}}}(\frac{k\pi}{b-a};\Delta t))e^{-ik\pi\frac{a}{b-a}}\right)V_k,
\end{aligned}$$

where V_k is known analytically and $\epsilon_{\cos}$ is the error resulting from the use of the COS pricing method. From [34] we know that for a sufficiently large truncation range $[a,b]$, we have $\epsilon_{\cos} = O(\epsilon_F)$ and thus

$$\epsilon = O(\epsilon_F) + e^{-r\Delta t}{\sum_{k=0}^{N-1}}' Re\left(\epsilon(\hat{\phi}_{Y_{\mathcal{M}}}(\frac{k\pi}{b-a};\Delta t))e^{-ik\pi\frac{a}{b-a}}\right)V_k. \tag{5.39}$$

The remaining part of the error (5.39), which we need to estimate, is $\epsilon(\hat{\phi}_{Y_{\mathcal{M}}}(u;\Delta t))$. This is done by mathematical induction. We first estimate the error in $\hat{\phi}_{Y_1}(u;\Delta t)$ and $\hat{\phi}_{Y_2}(u;\Delta t)$ and then use an induction step to bound the error in $\hat{\phi}_{Y_{\mathcal{M}}}(u;\Delta t)$.

Characteristic function $\phi_{Y_1}(u;\Delta t)$ is known analytically from (5.13), so that $\epsilon(\hat{\phi}_{Y_1}(u;\Delta t)) = 0,\ \forall u$.

The error in $\hat{\phi}_{Z_1}(u;\Delta t)$ consists of three parts. The first part is the error due to the truncation of the integration range as in (5.16). The second part is due to the approximation of $f_{Y_1}(x)$ by the Fourier cosine expansion in (5.19). The third part is due to the use of the Clenshaw–Curtis quadrature rule to

approximate the integral in (5.19). Summing up, we have:

$$\begin{aligned}
\epsilon(\hat{\phi}_{Z_1}(u;\Delta t)) &= \int_{-\infty}^{\infty}(e^x+1)^{iu}f_{Y_1}(x)dx - \int_a^b (e^x+1)^{iu}f_{Y_1}(x)dx \\
+ &\int_a^b (e^x+1)^{iu}f_{Y_1}(x)dx - \frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\phi_{Y_1}(\frac{l\pi}{b-a};\Delta t)\exp(-ia\frac{l\pi}{b-a})\right) I \\
+ &\frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\phi_{Y_1}(\frac{l\pi}{b-a};\Delta t)\exp(-ia\frac{l\pi}{b-a})\right)(I-\hat{I}) \qquad (5.40)\\
= &\int_{\mathbb{R}\setminus[a,b]}(e^x+1)^{iu}f_{Y_1}(x)dx + \epsilon_F \\
+ &\frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\phi_{Y_1}(\frac{l\pi}{b-a};\Delta t)\exp(-ia\frac{l\pi}{b-a})\right)\epsilon_q.
\end{aligned}$$

The lemma below gives an upper bound for the local error.

Lemma 5.4.2. *We define by*

$$\begin{aligned}
e_j : &= \int_{\mathbb{R}\setminus[a,b]}(e^x+1)^{iu}f_{Y_j}(x)dx + \epsilon_F \\
&+ \frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\phi_{Y_j}(\frac{l\pi}{b-a};\Delta t)\exp(-ia\frac{l\pi}{b-a})\right)\epsilon_q, \qquad (5.41)
\end{aligned}$$

then, with integration range $[a,b]$ sufficiently wide, we have

$$|e_j| \le \bar{P}(N,n_q)(|\epsilon_F| + \frac{2}{b-a}N|\epsilon_q|),\ \forall j,$$

where $\bar{P}(N,n_q) > 0$ varies less than ϵ_F and ϵ_q, with respect to N, n_q.

Proof. Application of (5.18) gives us that, $\forall j, u \in \mathbb{R}$,

$$|\int_{\mathbb{R}\setminus[a,b]}(e^x+1)^{iu}f_{Y_j}(x)dx| \le \epsilon_T(Y_j), \qquad (5.42)$$

with $\epsilon_T(Y_j)$ defined in (5.37). Substitution into (5.41), results in

$$|e_j| \le |\epsilon_T(Y_j)| + |\epsilon_F| + \frac{2}{b-a}\sum_{l=0}^{N-1}{}' |Re\left(\phi_{Y_j}(\frac{l\pi}{b-a};\Delta t)\exp(-ia\frac{l\pi}{b-a})\right)||\epsilon_q|.$$

From Lemma 5.4.1, it follows that, $\forall j, l$, $|\phi_{Y_j}(l\pi/b-a)| \le 1$, and

$$|\exp(-ia\frac{l\pi}{b-a})| = |\cos\left(-a\frac{l\pi}{b-a}\right) + i\sin\left(-a\frac{l\pi}{b-a}\right)| = 1,\ \forall l,$$

so that $|Re\left(\phi_{Y_j}\left(l\pi/(b-a)\right)\exp\left(-ial\pi/(b-a)\right)\right)| \leq 1,\ \forall j,l.$

For $[a,b]$ sufficiently wide, ϵ_F dominates the expression $\epsilon_F+\epsilon_T$, so that we find, $\forall j$:

$$|e_j| \leq \bar{P}(N,n_q)\left(|\epsilon_F|+\frac{2}{b-a}\sum_{l=0}^{N-1}{}'|\epsilon_q|\right) = \bar{P}(N,n_q)\left(|\epsilon_F|+\frac{2}{b-a}N|\epsilon_q|\right), \tag{5.43}$$

where $\bar{P}(N,n_q)>0$ varies less than ϵ_F and ϵ_q with respect to N,n_q. □

Using the notation:

$$\epsilon_L := |\epsilon_F|+\frac{2}{b-a}N|\epsilon_q|, \tag{5.44}$$

we can write $|e_j| \leq \bar{P}(N,n_q)\epsilon_L,\ \forall j$. Application of Lemma 5.4.2 and (5.44) to (5.40) gives

$$|\epsilon(\hat{\phi}_{Z_1}(u;\Delta t))| = |e_1| \leq \bar{P}(N,n_q)\epsilon_L.$$

We continue with the error in $\hat{\phi}_{Y_2}(u;\Delta t)$. From (5.14) we have that

$$\begin{aligned}\epsilon(\hat{\phi}_{Y_2}(u;\Delta t)) &= \epsilon(\hat{\phi}_{Z_1}(u;\Delta t))\phi_R(u;\Delta t)\\ &= e_1\phi_R(u;\Delta t) = e_1\phi_{Y_1}(u;\Delta t),\ \forall u.\end{aligned} \tag{5.45}$$

Applying Lemma 5.4.1 and Lemma 5.4.2 to (5.45) results in

$$|\epsilon(\hat{\phi}_{Y_2}(u;\Delta t))| = |e_1||\phi_{Y_1}(u;\Delta t)| \leq |e_1| \leq \bar{P}(N,n_q)\epsilon_L. \tag{5.46}$$

Next, we arrive at the induction step, described in the lemma below.

We use the common notation $\epsilon = O(\bar{g}(a_1,\cdots,a_n))$ to indicate that a $Q>0$ exists, so that $|\epsilon| = Q|\bar{g}(a_1,\cdots,a_N)|$, with Q constant or varying less than function $\bar{g}(\cdot)$ with respect to parameters $a_1,\cdots,a_N$.

Lemma 5.4.3. *For $m=3,\cdots,\mathcal{M}$, assuming that*

$$\epsilon(\hat{\phi}_{Y_{m-1}}(u;\Delta t)) = \bar{P}(N,n_q)\sum_{j=1}^{(m-1)-1}\phi_{Y_j}(u;\Delta t)e_{(m-1)-j},\ \forall u, \tag{5.47}$$

where $\bar{P}(N,n_q)$ is a term which varies less than exponentially with respect to N and n_q, we have

$$\epsilon(\hat{\phi}_{Y_m}(u;\Delta t)) = O(\sum_{j=1}^{m-1}\phi_{Y_j}(u;\Delta t)e_{m-j}),\ \forall u, \tag{5.48}$$

and thus

$$|\epsilon(\hat{\phi}_{Y_m}(u;\Delta t))| = O(m-1)\epsilon_L. \tag{5.49}$$

Proof. We find that for $m = 3, \cdots, \mathcal{M}$, and $\forall u$:

$$
\begin{aligned}
& \epsilon(\hat{\phi}_{Z_{m-1}}(u;\Delta t)) \\
= & \int_{-\infty}^{\infty}(e^x+1)^{iu} f_{Y_{m-1}}(x)dx - \frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\hat{\phi}_{Y_{m-1}}(\frac{l\pi}{b-a};\Delta t)\exp(-ia\frac{l\pi}{b-a})\right)\hat{I} \\
= & \int_{-\infty}^{\infty}(e^x+1)^{iu} f_{Y_{m-1}}(x)dx - \int_a^b (e^x+1)^{iu} f_{Y_{m-1}}(x)dx \\
+ & \int_a^b (e^x+1)^{iu} f_{Y_{m-1}}(x)dx - \frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\phi_{Y_{m-1}}(\frac{l\pi}{b-a};\Delta t)\exp(-ia\frac{l\pi}{b-a})\right) I \\
+ & \frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\phi_{Y_{m-1}}(\frac{l\pi}{b-a};\Delta t)\exp(-ia\frac{l\pi}{b-a})\right)(I-\hat{I}) \\
+ & \frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left((\phi_{Y_{m-1}}(\frac{l\pi}{b-a};\Delta t) - \hat{\phi}_{Y_{m-1}}(\frac{l\pi}{b-a};\Delta t))\exp(-ia\frac{l\pi}{b-a})\right)\hat{I} \\
= & \int_{\mathbb{R}\setminus[a,b]}(e^x+1)^{iu} f_{Y_{m-1}}(x)dx + \epsilon_F \\
+ & \frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\phi_{Y_{m-1}}(\frac{l\pi}{b-a};\Delta t)\exp(-ia\frac{l\pi}{b-a})\right)\epsilon_q \\
+ & \frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\epsilon(\phi_{Y_{m-1}}(\frac{l\pi}{b-a};\Delta t))\exp(-ia\frac{l\pi}{b-a})\right)\hat{I} \\
= & e_{m-1} + \frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\epsilon(\phi_{Y_{m-1}}(\frac{l\pi}{b-a};\Delta t))\exp(-ia\frac{l\pi}{b-a})\right)\hat{I}. \qquad (5.50)
\end{aligned}
$$

Substitution of (5.47) into (5.50) gives

$$
\begin{aligned}
& \epsilon(\hat{\phi}_{Z_{m-1}}(u;\Delta t)) \\
= & e_{m-1} + \bar{P}(N,n_q)\sum_{j=1}^{(m-1)-1}\frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\phi_{Y_j}(\frac{l\pi}{b-a};\Delta t)e_{(m-1)-j}\exp(-ia\frac{l\pi}{b-a})\right)\hat{I} \\
= & e_{m-1} + \bar{P}(N,n_q)\sum_{j=1}^{(m-1)-1} e_{(m-1)-j}(\frac{2}{b-a}\sum_{l=0}^{N-1}{}' Re\left(\phi_{Y_j}(\frac{l\pi}{b-a};\Delta t)\exp(-ia\frac{l\pi}{b-a})\right)\hat{I}) \\
= & e_{m-1} + \bar{P}(N,n_q)\sum_{j=1}^{(m-1)-1} e_{(m-1)-j}\hat{\phi}_{Z_j}(u;\Delta t).
\end{aligned}
$$

The error in $\hat{\phi}_{Y_m}(u;\Delta t)$, $\forall u$, is found to be

$$\begin{aligned}
\epsilon(\hat{\phi}_{Y_m}(u;\Delta t)) &= \phi_R(u;\Delta t)\epsilon(\hat{\phi}_{Z_{m-1}}(u;\Delta t)) \\
&= \phi_R(u;\Delta t)e_{m-1} + \bar{P}(N,n_q)\sum_{j=1}^{(m-1)-1} e_{(m-1)-j}\phi_R(u;\Delta t)\hat{\phi}_{Z_j}(u;\Delta t) \\
&= \phi_R(u;\Delta t)e_{m-1} + \bar{P}(N,n_q)\sum_{j=1}^{(m-1)-1} e_{(m-1)-j}\hat{\phi}_{Y_{j+1}}(u;\Delta t) \\
&= \phi_{Y_1}(u;\Delta t)e_{m-1} + \bar{P}(N,n_q)\sum_{j=2}^{m-1} e_{m-j}\hat{\phi}_{Y_j}(u;\Delta t) \\
&= O(\sum_{j=1}^{m-1}\phi_{Y_j}(u;\Delta t)e_{m-j}) + O(e_k e_l), \quad k,l \in 1,\cdots,m-1.
\end{aligned}$$

From Lemma 5.4.2 we see that $|e_j| = O(|\epsilon_F| + |\epsilon_q|)$, $\forall j$, if N and n_q increase simultaneously. Error ϵ_F decays exponentially with respect to N, and ϵ_q decays exponentially with respect to n_q, so that e_j decays exponentially and the quadratic term, $e_k e_l$, converges to zero faster than e_j. We thus have that

$$\epsilon(\hat{\phi}_{Y_m}(u;\Delta t)) = O(\sum_{j=1}^{m-1}\phi_{Y_j}(u;\Delta t)e_{m-j}),$$

and application of Lemmas 5.4.1 and 5.4.2 gives, $\forall u \in \mathbb{R}$,

$$|\sum_{j=1}^{m-1}\phi_{Y_j}(u;\Delta t)e_{m-j}| \leq \sum_{j=1}^{m-1}|\phi_{Y_j}(u;\Delta t)||e_{m-j}| \leq \bar{P}(N,n_q)(m-1)\epsilon_L,$$

where $\bar{P}(N,n_q)$ varies less than ϵ_F and ϵ_q with respect to N, n_q, respectively. So,

$$|\epsilon(\hat{\phi}_{Y_m}(u;\Delta t))| = O((m-1)\epsilon_L), \tag{5.51}$$

which concludes the proof. □

As a result of the lemma above, we have, $\forall u$,

$$\epsilon(\hat{\phi}_{Y_{\mathcal{M}}}(u;\Delta t)) = O(\sum_{j=1}^{\mathcal{M}-1}\phi_{Y_j}(u;\Delta t)e_{m-j}), \tag{5.52}$$

and

$$|\epsilon(\hat{\phi}_{Y_{\mathcal{M}}}(u;\Delta t))| = O((\mathcal{M}-1)\epsilon_L). \tag{5.53}$$

Remark 5.4.1 (Error of $\hat{\phi}_{Y_{\mathcal{M}}}$). *Application of (5.53) and (5.44) results in*

$$|\epsilon(\hat{\phi}_{Y_{\mathcal{M}}}(u;\Delta t))| = O((\mathcal{M}-1)(|\epsilon_F| + \frac{2}{b-a}N|\epsilon_q|)),\ \forall u.$$

When the number of monitoring dates, $\mathcal{M}$, increases, larger values of N and n_q are necessary to reach a specified level of accuracy.

Moreover, when a large value of N is necessary for accuracy, we should also increase n_q to control the error. When N and n_q both increase, the expression $|N\epsilon_q|$ converges exponentially to zero[1]*, and we have that*

$$|\epsilon(\hat{\phi}_{Y_{\mathcal{M}}}(u;\Delta t))| = O((\mathcal{M}-1)(|\epsilon_F| + |\epsilon_q|)),\ \forall u.$$

5.4.2 Error in the option price

We now focus on the error in the arithmetic Asian option price. After application of (5.52) in (5.39) the error reads

$$\epsilon = O(\epsilon_F) + O(\sum_{j=1}^{\mathcal{M}-1} e_{m-j}\exp(-r\Delta t)\sum_{k=0}^{N-1}{}' Re(\phi_{Y_j}(\frac{k\pi}{b-a};\Delta t)e^{-ik\pi\frac{a}{b-a}})V_k). \tag{5.54}$$

When replacing $e^{-r\Delta t}V_k$ (V_k as defined in (5.22)) by the following term:

$$e^{-r\Delta t_j}W_k^j := e^{-r\Delta t_j}\begin{cases} \frac{2}{b-a}(\frac{S_0}{j+1}\chi_k(x^*,b) + (\frac{S_0}{j+1} - K)\psi_k(x^*,b)), & \text{for a call,} \\ \frac{2}{b-a}((K - \frac{S_0}{j+1})\psi(a,x^*) - \frac{S_0}{j+1}\chi(a,x^*)), & \text{for a put,} \end{cases} \tag{5.55}$$

with $\Delta t_j := j\Delta t/\mathcal{M}$, the expression

$$\sum_{j=1}^{\mathcal{M}-1} e_{m-j}\exp(-r\Delta t)\sum_{k=0}^{N-1}{}' Re(\phi_{Y_j}(\frac{k\pi}{b-a};\Delta t)e^{-ik\pi\frac{a}{b-a}})V_k,\ \forall j,k,$$

remains of the same order, regarding N and n_q.

The error in (5.54) therefore satisfies

$$\epsilon = O(\epsilon_F) + O(\sum_{j=1}^{\mathcal{M}-1} e_{m-j}e^{-r\Delta t_j}\sum_{k=0}^{N-1}{}' Re(\phi_{Y_j}(\frac{k\pi}{b-a};\Delta t)e^{-ik\pi\frac{a}{b-a}})W_k^j).$$

We now can write for the overall error:

$$\epsilon = O(\epsilon_F) + O(\sum_{j=1}^{\mathcal{M}-1} e_{m-j}A(S_0,\Delta t_j)),$$

[1] Note that N varies linearly, but ϵ_q decays exponentially, so that $N|\epsilon_q|$ also decays exponentially.

where $A(S_0, \tau)$ stands for the Asian option value with initial underlying price S_0 and time to maturity τ. Then

$$|\epsilon| = O(|\epsilon_F|) + O(\sum_{j=1}^{\mathcal{M}-1} |e_{m-j}| A(S_0, \Delta t_j)).$$

By Lemma 5.4.2 we find

$$|\epsilon| = O(|\epsilon_F|) + O((|\epsilon_F| + \frac{2}{b-a} N|\epsilon_q|) \sum_{j=1}^{\mathcal{M}-1} A(S_0, \Delta t_j)). \tag{5.56}$$

Volatility inherent in an Asian option is smaller than that of an equivalent vanilla European option, due to the averaging feature. This makes Asian options cheaper than their plain vanilla equivalents. In other words, with the same maturity, the value of an Asian option, $A(S_0, \tau)$, is less or equal to that of the corresponding vanilla European option, denoted by $E(S_0, \tau)$, written on the same underlying asset. The European option value will be used as upper bound for the corresponding arithmetic Asian option value in (5.56) and we have:

$$|\epsilon| = O(|\epsilon_F|) + O((|\epsilon_F| + \frac{2}{b-a} N|\epsilon_q|) \sum_{j=1}^{\mathcal{M}-1} E(S_0, \Delta t_j)). \tag{5.57}$$

We assume that

$$\max_{j=1,\cdots,\mathcal{M}-1} E(S_0, j\Delta t_j) = E(S_0, \Delta t_{j^*}),$$

so that the error in the Asian option price satisfies

$$|\epsilon| = O(|\epsilon_F|) + O((|\epsilon_F| + \frac{2}{b-a} N|\epsilon_q|)(\mathcal{M}-1) E(S_0, \Delta t_{j^*})). \tag{5.58}$$

What remains is an upper bound for the plain vanilla European option value, $E(S_0, (\mathcal{M}-1)\Delta t_{j^*})$, which is given as follows.

Result 5.4.1. *The value of a plain vanilla European call option can be bounded by*

$$v_C(S_0, \tau) \leq S_0 e^{-q\tau},$$

with S_0, τ, q the initial underlying price, the time to maturity and the dividend rate, respectively.

The value of a vanilla European put option can be bounded by

$$v_P(S_0, \tau) \leq K e^{-r\tau},$$

with K, r the strike price and the interest rate, respectively.

Summarizing, the error in the arithmetic Asian option with $\mathcal{M}$ monitoring dates can be approximated by:

$$|\epsilon| \sim \begin{cases} O((|\epsilon_F| + \frac{2}{b-a}N|\epsilon_q|)(\mathcal{M}-1)S_0 e^{-q\Delta t_{j^*}}), & \text{for a call,} \\ O((|\epsilon_F| + \frac{2}{b-a}N|\epsilon_q|)(\mathcal{M}-1)Ke^{-r\Delta t_{j^*}}), & \text{for a put.} \end{cases} \tag{5.59}$$

For $f(y|x) \in \mathbf{C}^{\infty}[a,b]$, ϵ_F and ϵ_q converge exponentially with respect to N and n_q, respectively. Therefore, as N and n_q both increase, the error in the Asian option price decreases exponentially:

$$|\epsilon| \leq \bar{P}(N, n_q)(\exp(-(N-1)\nu_F) + \exp(-(n_q-1)\nu_q)),$$

where $\bar{P}(N, n_q)$ is a term which varies less than exponentially with respect to N and n_q, and $\nu_F > 0, \nu_q > 0$.

When the probability density function has a discontinuous derivative, the error in the Asian option price converges algebraically.

5.5 Numerical results

In this section numerical results for Asian options under the Black-Scholes (BS), CGMY [14] and Normal Inverse Gaussian (NIG) [3] models are presented. We use the same parameter sets as in [37], based on three test cases:

- *BS case:* $r = 0.0367, \sigma = 0.17801$.
- *CGMY case:* $r = 0.0367, C = 0.0244, G = 0.0765, M = 7.5515,$ $Y = 1.2945$.
- *NIG case:* $r = 0.0367, \alpha = 6.1882, \beta = -3.8941, \delta = 0.1622$.

These parameters have been obtained by calibration (see [37]). The characteristic functions for these processes were presented in Subsection 1.1.1. In all numerical examples we set time to maturity $T - t_0 = 1$, and $S_0 = 100$. Strike price, K, and the number of monitoring dates, $\mathcal{M}$, vary among the different experiments.

MATLAB 7.7.0 is used and the CPU is an Intel(R) Core(TM)2 Duo CPU E6550 (@ 2.33GHz Cache size 4MB). CPU time is recorded in seconds.

The absolute error that we report below is defined as the absolute value of the difference between the approximate solution at t_0 and S_0, and a reference value which is computed by the ASCOS method with a large number of terms in the Fourier cosine expansions. The values have also been compared to reference values in the literature. With our own reference values however we can compare up to a higher accuracy.

5.5.1 Geometric Asian options

First of all, we confirm the exponential convergence of the ASCOS method for geometric Asian options under the Black–Scholes model, for which an analytic result is available, in Figure 5.1. For increasing N-values the error decreases exponentially.

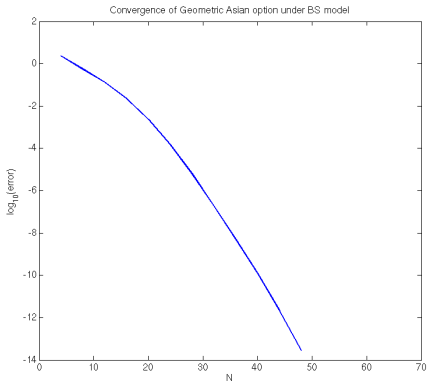


Figure 5.1: Convergence of geometric Asian options under the BS model with $\mathcal{M} = 250, S_0 = 100, K = 90$.

The performance of the ASCOS pricing method for the NIG and CGMY test cases is presented in Table 5.1. Geometric Asian call option prices with 12, 50 and 250 monitoring dates are shown. Reference values are taken from ASCOS computations with $N = 4096$. In all examples our method also gives the same option prices, up to a basis point, as those presented in [37].

From Table 5.1 we see that the option prices have converged up to basis point precision with $N = 128$ and $N = 512$, respectively, for the NIG and CGMY test cases. Exponential convergence is observed for these Lévy processes and, as a result, geometric Asian options can be priced within milliseconds by the ASCOS method. In a comparison with the results in [37], we found that our timing results are approximately 100 times faster for the NIG test case and 20 times for the CGMY case.

Table 5.2 presents the convergence behavior when continuously–monitored geometric Asian options ($\mathcal{M} = \infty$) are approximated by discretely–monitored geometric Asian options combined with the 4–point Richardson extrapolation (5.31). Here d is as defined in (5.31), that is, discretely–monitored Asian options with $2^d, 2^{d+1}, 2^{d+2}, 2^{d+3}$ monitoring dates are used to approximate the continuously–monitored Asian option. The reference values have been obtained by employing the ASCOS method with $N = 4096, \mathcal{M} = 512$.

The discretely–monitored Asian prices with 4, 8, 16 and 32 monitor-

NIG model				
$\mathcal{M}$		$N = 64$	$N = 128$	$N = 192$
12	abs.error	1.42e-04	2.81e-05	1.33e-08
	CPU time	4.9e-04	7.7e-04	8.3e-04
50	abs.error	1.23e-04	3.07e-05	1.24e-08
	CPU time	9.3e-04	1.4e-03	2.1e-03
250	abs.error	1.13e-04	3.13e-05	2.11e-08
	CPU time	3.1e-03	5.8e-03	8.2e-03
CGMY model				
$\mathcal{M}$		$N = 256$	$N = 512$	$N = 1024$
12	abs.error	2.1e-03	9.87e-06	6.27e-11
	CPU time	2.7e-03	4.1e-03	9.9e-03
50	abs.error	1.20e-02	1.24e-05	6.71e-11
	CPU time	1.2e-02	1.7e-02	4.3e-02
250	abs.error	1.16e-02	3.65e-05	3.84e-11
	CPU time	0.050	0.10	0.22

Table 5.1: Convergence of geometric Asian options for the NIG and CGMY test cases with $S_0 = 100, K = 110$.

	NIG		CGMY	
d	abs.error	CPU time	abs.error	CPU time
1	3.78e-04	0.0018	2.06e-04	0.0120
2	5.92e-05	0.0023	1.21e-04	0.0247
3	3.31e-05	0.0052	5.71e-05	0.0499

Table 5.2: Convergence of geometric Asian options for the NIG and CGMY cases with $S_0 = 100, K = 110$. For the NIG model we use $N = 128$, for the CGMY model $N = 512$.

ing dates, i.e., $d = 2$ have converged to the continuously–monitored Asian price in Table 5.2. We need approximately 2 and 25 milliseconds to get the continuously–monitored Asian option prices for the NIG and CGMY test cases, respectively. As compared to [37], we achieve a speedup of 20 for the NIG test and the CPU time for the CGMY case is approximately one–third of that in [37].

5.5.2 Arithmetic Asian options

First, the error in an arithmetic Asian option under the Black–Scholes model with 50 monitoring dates is presented in Figure 5.1, where at the y–axis we have the logarithm (basis 10) of the absolute error in the Asian option price

and at the x–axis the value of index d, where $N = 64d$ and $n_q = 100d$. The reference value is obtained by the ASCOS method with $N = 4096$ (the resulting values are as in [37]). Exponential convergence in the arithmetic Asian price with respect to N and n_q, increasing simultaneously, is observed in Figure 5.2.

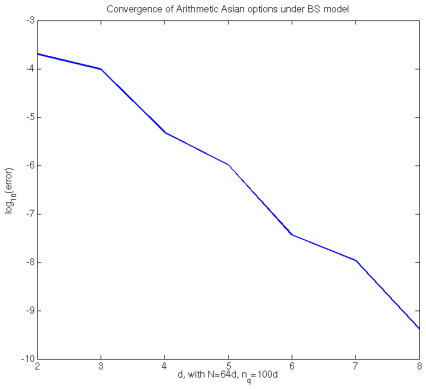


Figure 5.2: Convergence of arithmetic Asian options for the BS test case with $\mathcal{M} = 50, S_0 = 100, K = 90$.

Table 5.3 then presents the convergence and the CPU time of an arithmetic Asian option for the NIG test case with $\mathcal{M} = 12$ and $\mathcal{M} = 50$ (monthly and weekly-monitored, respectively). Reference values are again obtained by setting $N = 4096$. Exponential convergence can be seen in Table 5.3, as the error decreases exponentially, when n_q and N increase linearly. The Asian options for $\mathcal{M} = 50$ converge up to basis point precision with $N = 128$ and $n_q = 200$, where the CPU time is approximately 2.5 seconds. Higher order accuracy can be achieved as N and n_q increase, but the CPU time grows with respect to $n_q N^2$.

The speed of convergence is *not* influenced significantly by an increase in the number of monitoring dates, $\mathcal{M}$, neither is the CPU time.

$\mathcal{M}$	time and error	$N = 128$ $n_q = 200$	$N = 256$ $n_q = 400$	$N = 384$ $n_q = 600$
12	abs.error	2.0e-3	1.71e-4	5.16e-6
	CPU time	2.41	15.13	46.09
50	abs.error	2.26e-4	6.94e-5	2.17e-6
	CPU time	2.43	15.16	46.22

Table 5.3: Convergence of arithmetic Asian options for the NIG test case with $S_0 = 100, K = 110$.

Furthermore, the convergence remains robust when the number of monitoring dates increases, which gave rise to convergence difficulties for other pricing methods. A larger number of Fourier cosine terms is required (thus resulting in a larger CPU time) as compared to monthly or weekly-monitored examples. This can be seen in Table 5.4, where arithmetic Asian options for the NIG and CGMY test cases, with 250 monitoring dates (daily-monitored), are presented. With $N = 256, n_q = 400$ and $N = 320, n_q = 500$, we find converged option prices (up to basis point precision) for the NIG and CGMY cases, respectively.

Due to the exponential convergence rate of the Clenshaw–Curtis quadrature and the Fourier cosine expansion, the number of terms needed remains limited, which influences the CPU time positively. In [37] an accuracy of $O(10^{-3})$ was reached in approximately 210 seconds for the same CGMY test case with $\mathcal{M} = 250$. The ASCOS method reaches $O(10^{-4})$ accuracy in approximately 27 seconds.

A comparison of the CPU times in Tables 5.3 and 5.4 shows that the ASCOS CPU time does not increase from $\mathcal{M} = 12$ to $\mathcal{M} = 250$, because the quadrature rule, which dominates the CPU time, is used only once. This is especially beneficial for pricing continuously-monitored Asian options.

NIG	time and error	$N = 128$ $n_q = 200$	$N = 256$ $n_q = 400$	$N = 512$ $n_q = 800$
	abs.error	7.8e-3	9.33e-5	6.94e-7
	CPU time	2.42	15.23	104.28
CGMY	time and error	$N = 256$ $n_q = 400$	$N = 320$ $n_q = 500$	$N = 384$ $n_q = 600$
	abs.error	1.6e-3	4.69e-4	8.96e-5
	CPU time	14.92	26.61	44.41

Table 5.4: Convergence of arithmetic Asian options for the NIG and CGMY test cases with $S_0 = 100, K = 110, \mathcal{M} = 250$.

In Table 5.5 we finally compute continuously–monitored arithmetic Asian call options under the NIG model with $S_0 = 100$ and different strikes, by the repeated Richardson extrapolation, based on discretely–monitored arithmetic Asian call options (5.31). The option prices converge somewhat slower with respect to parameter d, as compared to the geometric Asian case. However, the CPU time of the ASCOS method does not increase when d increases, so that we can use a larger value for d, for instance $d = 6$ ($\mathcal{M} = 64, 128, 256, 512$) and obtain accurate results.

d	$K = 90$		$K = 100$	
	Option value	CPU time	Option value	CPU time
4	12.6748	60.05	5.1191	60.01
5	12.6744	60.13	5.1186	59.94
6	12.6743	60.35	5.1185	60.17

Table 5.5: Convergence of arithmetic Asian options under the NIG model with $S_0 = 100, N = 256, n_q = 400$.

5.6 Conclusions

In this chapter, we proposed an efficient pricing method for European-style Asian options, the ASCOS method, based on Fourier cosine expansions and Clenshaw–Curtis quadrature. The method performs well for different Lévy processes, different parameter values and different numbers of Asian option monitoring dates. The method is accompanied by a detailed error analysis, giving evidence for an exponential convergence rate for geometric and arithmetic Asian options. Due to the exponential convergence, our pricing method is highly efficient and significant speedup has been achieved compared to competitor pricing methods.

The ASCOS method performs in a robust manner when the number of monitoring dates increases, and, interestingly, the CPU time does not increase significantly. This makes the pricing method especially advantageous for weekly- and even daily-monitored arithmetic Asian options, as well as for continuously–monitored Asian options whose value is approximated by discretely–monitored Asian options in combination with Richardson extrapolation.

CHAPTER 6

Efficient Pricing of Asian Options under Lévy Processes Based on Fourier Cosine Expansions Part II: Early–Exercise Features and GPU Implementation

This chapter contains essentially the contents of paper [74].

6.1 Introduction

An Asian option is a special type of exotic option, introduced in Japan, in 1987. Because the contract description (i.e. the pay-off function) is based on geometric or arithmetic averages of the underlying stock price at monitoring dates, rather than just on the present asset price, this exotic option is also called *path-dependent*. The number of monitoring dates can be finite (so-called discretely–monitored Asian options) or infinite (continuously–monitored Asian options). Asian options are popular, because averages typically move in a more stable way than individual asset prices, and the volatility, inherent in asset prices, is reduced due to the averaging feature, so that Asian option holders may pay lower prices for these contracts, compared to plain European options.

There is not much information on early-exercise Asian option products in the present markets. We may encounter them in the commodity market, and variants in the equity market are so-called American options with an Asian

tail (meaning that the final part of the contract time is based on averaged asset prices rather than on plain assets). In the academic literature, important contributions [28, 5] have been presented, when pricing these Asian options by partial differential and partial integro-differential equations (PDEs and PIDEs, respectively). In [28], for example, a semi–Lagrangian time-stepping method was used to solve the P(I)DE in a time-stepping procedure. The method worked particularly well for American-style Asian options under a jump–diffusion model.

In [73], European-style Asian options were priced by means of Fourier cosine expansions (as in the COS method [34]) and Clenshaw–Curtis quadrature. The method was named the *Asian cosine method* (ASCOS). This new pricing method can be seen as an efficient alternative to Fast Fourier Transform (FFT) and convolution methods, as in [18, 37, 4, 49], for pricing European–style Asian options under Lévy processes.

In this chapter, which is the continuation work of chapter 5, we propose an efficient version of the ASCOS pricing method for early–exercise Asian options, again based on Fourier expansions, Fast Fourier Transform (FFT) and Clenshaw–Curtis quadrature. In the 2D ASCOS method the option price is calculated based on two dimensions of uncertainty, i.e. the uncertainty in the asset process, as well as in the *averaged asset process* over time. The risk–neutral formula then becomes a two–dimensional integration, based on which the continuation value is approximated at each time step. By application of the *chain rule* from probability theory, the joint conditional density function in the risk–neutral formula can be factorized into two marginal conditional density functions that are approximated by Fourier cosine expansions. To calculate the option price, we need to recursively recover the Fourier coefficients with the help of Fourier cosine expansions and Clenshaw–Curtis quadrature. The FFT is used to accelerate the algorithm. The computational complexity of our pricing method for Asian options, with $\mathcal{M}$ early–exercise dates, is $O((\mathcal{M}-1)n_q N_1 N_2 \log_2 N_2)$, with N_1, N_2 the number of Fourier cosine terms in the expansions for the density functions of the asset and the averaged asset price, respectively, and n_q the number of terms in the Clenshaw-Curtis quadrature.

Exponential convergence in the option price, with respect to n_q, N_1, N_2, is obtained for most Lévy processes, for which we give an error analysis, combined with numerical examples. The 2D method is presented in Section 6.4, followed by an error analysis in Section 6.5. Numerical results are given in Section 6.6, where the efficiency and accuracy of the pricing methods are presented. Implementation has taken place on the Graphics Processing Unit (GPU). It may be interesting to see that this computer architecture improves the pricing speed drastically when pricing arithmetic Asian options with early-exercise features.

We start, however, in Section 6.3, with another, alternative Asian pricing method, which is only accurate in the case of a large number of early–exercise

dates. It has a reduced computational cost of $O((\mathcal{M}-1)n_q N^2)$, with N the number of terms in a 1D Fourier cosine expansion. The approximation error appearing from the approximations in this alternative method converges to zero only when the number of early–exercise dates tends to infinity.

6.2 Early-exercise Asian options under Lévy processes

In this chapter the underlying asset, S_t, is assumed to be an exponential function of a Lévy process, L_t, i.e. $S_t = S_0 \exp(L_t)$. Lévy process, L_t, with initial condition $L_0 = 0$, has independent and stationary increments and is stochastically continuous. For any $s < t$, and $\forall \epsilon > 0$, we have

$$\lim_{s \to t} \mathbb{P}(|L_t - L_s| > \epsilon) = 0.$$

The (conditional) probability density function is not known for many relevant Lévy asset processes. However, its *Fourier transform*, the (conditional) characteristic function, $\phi_{Y_m|Y_{m-1}}(\cdot)$, is often available, for example, by the Lévy-Khinchine theorem for underlying Lévy processes. Our pricing algorithm is based on this Fourier transform.

In this chapter, we deal with early–exercise options in which the contract function at each exercise date is a function of the averaged underlying asset price, up to that date. *Early-exercise* implies that the option may be exercised prior to the expiration date. Let t_0 denote the initial time and $\mathcal{T} = \{t_1, \cdots, t_{\mathcal{M}}\}$ be the collection of all exercise dates with $\Delta t := (t_m - t_{m-1})$, $t_0 < t_1 < \cdots < t_{\mathcal{M}} = T$, and assume that the early–exercise dates and the monitoring dates of the Asian options are the same.

We focus on arithmetic averaging, as it is mathematically more challenging, and on *fixed–strike Asian options*, with payoff functions defined by

$$g(S, t_m) = \begin{cases} \max(\frac{1}{m+1} \sum_{j=0}^{m} S_j - K, 0), & \text{for a call,} \\ \max(K - \frac{1}{m+1} \sum_{j=0}^{m} S_j, 0), & \text{for a put,} \end{cases}$$

These payoff functions change from time step to time step, due to the averaging feature.

6.3 A first Asian pricing method (for $\mathcal{M} \to \infty$)

We present here a first pricing algorithm for early–exercise arithmetic Asian options, *which is only accurate for a large number of early-exercise dates*, as we will proof in the subsection to follow.

The pricing formula for an early–exercise Asian option with $\mathcal{M}$ exercise dates then reads, for $m = \mathcal{M}, \mathcal{M}-1, \ldots, 2$:

$$\begin{cases} c(y_{m-1}, t_{m-1}) &= e^{-r\Delta t} \int_{\mathbb{R}} v(y_m, t_m) f(y_m|y_{m-1}) dy_m, \\ v(y_{m-1}, t_{m-1}) &= \max\left(g(y_{m-1}, t_{m-1}), c(y_{m-1}, t_{m-1})\right), \end{cases} \tag{6.1}$$

followed by

$$v(y_0, t_0) = e^{-r\Delta t} \int_{\mathbb{R}} v(y_1, t_1) f(y_1|y_0) dy_1. \tag{6.2}$$

Here, y_m is the state variable at time step t_m, and $v(x,t)$, $c(x,t)$ and $g(x,t)$ are the option value, the continuation value and the payoff at time t, respectively. $v(S, t_{\mathcal{M}}) = g(S, t_{\mathcal{M}})$ is the payoff function at final time, $t_{\mathcal{M}} = T$. Function $f(y_m|y_{m-1})$ is the conditional density of y_m given y_{m-1}. Interest rate r is assumed to be constant here.

In the risk–neutral formula (6.1) the continuation value is computed at each time step as the discounted expected value of the option price at a next time step. Moreover, to avoid arbitrage opportunities, the option value at each time step cannot be less than the payoff of the option, which is the second equation in (6.1).

In [34, 35] the COS method was developed for the computation of continuation value $c(y_{m-1}, t_{m-1})$ and option price $v(y_0, t_0)$, for *vanilla* (i.e. non-path dependent) Bermudan options, under the assumption that the characteristic function of the underlying Lévy asset price process is known.

The pricing algorithm for early–exercise arithmetic Asian options in this section can be seen as a generalization of the COS method for Bermudan options (a general 2D pricing algorithm will be presented in Section 6.4).

Here, we define

$$Y_m := \log\left(\frac{S_1}{S_0} + \frac{S_2}{S_0} + \cdots + \frac{S_m}{S_0}\right), \; m = \mathcal{M}, \mathcal{M}-1, \cdots, 1. \tag{6.3}$$

Based on this variable, the payoff function is given by

$$g(y_m, t_m) = \begin{cases} \left(\dfrac{S_0(1+e^{y_m})}{m+1} - K\right)^+, & \text{for a call,} \\ \left(K - \dfrac{S_0(1+e^{y_m})}{m+1}\right)^+, & \text{for a put,} \end{cases} \tag{6.4}$$

After truncation of the integration range in (6.1) and (6.2), from $\mathbb{R}$ to $[a,b]$, we approximate the conditional density function in terms of its characteristic function, via a Fourier cosine expansion. For $m = \mathcal{M}-1, \cdots, 1$ the continuation value is approximated by Fourier cosine expansions, as

$$\hat{c}(y_{m-1}, t_{m-1}) = e^{-r\Delta t} \sum_{k=0}^{N-1}{}' \, Re\left(\hat{\varphi}_{Y_m|Y_{m-1}}\left(\frac{k\pi}{b-a}; y_{m-1}, \Delta t\right) e^{-ik\pi\frac{a}{b-a}}\right) \hat{V}_k(t_m), \tag{6.5}$$

where $\hat{c}, \hat{V}, \hat{\phi}$ indicate that these are numerical approximations. The prime at the sum symbol indicates that the first term in the summation is weighted by one-half. Conditional characteristic function $\hat{\varphi}_{Y_m|Y_{m-1}}(u; y_{m-1}, \Delta t)$, in (6.5), will be derived in Subsection 6.3.1. The Fourier cosine coefficients of the option price at t_m in (6.5) are defined by

$$\hat{V}_k(t_m) = \int_a^b \hat{v}(y_m, t_m) \cos(k\pi \frac{y_m - a}{b-a}) dy_m, \tag{6.6}$$

The target option price is obtained by computing

$$\hat{v}(y_0, t_0) = e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re\left(\phi_{Y_1|Y_0}(\frac{k\pi}{b-a}; y_0, \Delta t) e^{-ik\pi \frac{a}{b-a}} \right) \hat{V}_k(t_1), \tag{6.7}$$

where for arithmetic Asian options under Lévy processes, $\phi_{Y_1|Y_0}$ is known analytically.

Based on the conditional characteristic function for Y_m in (6.3), the early–exercise arithmetic Asian option value can be calculated by a *backward recursion* on the Fourier coefficients, $\hat{V}_k(t_m)$, as defined in (6.6). Then, the option price is obtained by inserting the value of $\hat{V}_k(t_1)$ into (6.7).

At maturity time, $t_{\mathcal{M}}$, the option value equals the payoff, and Fourier coefficients, $V_k(t_{\mathcal{M}})$, read:

$$V_k(t_{\mathcal{M}}) = \begin{cases} \frac{2}{b-a}(\frac{S_0}{\mathcal{M}+1}\chi_k(y^*_{\mathcal{M}}, b) + (\frac{S_0}{\mathcal{M}+1} - K)\psi_k(y^*_{\mathcal{M}}, b)), & \text{for a call,} \\ \frac{2}{b-a}((K - \frac{S_0}{\mathcal{M}+1})\psi_k(a, y^*_{\mathcal{M}}) - \frac{S_0}{\mathcal{M}+1}\chi_k(a, y^*_{\mathcal{M}})), & \text{for a put,} \end{cases} \tag{6.8}$$

with $y^*_{\mathcal{M}} = \log(\frac{K(\mathcal{M}+1)}{S_0} - 1)$, the value for which the payoff is nonzero, and

$$\psi_j(y_l, y_u) := \int_{y_l}^{y_u} \cos\left(j\pi \frac{y - \delta_n}{b_2 - \delta_n} \right) dy, \tag{6.9}$$

and

$$\chi_j(y_l, y_u) := \int_{y_l}^{y_u} e^y \cos\left(j\pi \frac{y - \delta_n}{b_2 - \delta_n} \right) dy. \tag{6.10}$$

At recursion step $t_m, m = \mathcal{M} - 1, \cdots, 1$, as a first step in the algorithm, the so-called *early–exercise point*, y^*_m, for which $c(y^*_m, t_m) = g(y^*_m, t_m)$, is determined by Newton's method, as the derivatives of the continuation value and the payoff function, with respect to y_m, can be easily derived. Based on this, we can split $V_k(t_m)$, as follows

$$V_k(t_m) = \begin{cases} C_k(a, y^*_m, t_m) + G_k(y^*_m, b, t_m), & \text{for a call,} \\ G_k(a, y^*_m, t_m) + C_k(y^*_m, b, t_m), & \text{for a put,} \end{cases}$$

where C_k, G_k are Fourier cosine coefficients of the continuation value and payoff at t_m, respectively. Coefficients G_k are of the form

$$G_k(t_m, y_l, y_u) = \frac{2}{b-a}\int_{y_l}^{y_u} g(y, t_m)\cos(k\pi\frac{y-a}{b-a})dy, \tag{6.11}$$

where for a call, $[y_l, y_u] = [y_m^*, b]$, and for a put, $[y_l, y_u] = [a, y_m^*]$. Inserting (6.4) into (6.11) gives us

$$G_k(t_m) = \frac{2}{b-a}\begin{cases} \frac{S_0}{m+1}\chi_k(y_m^*, b) + (\frac{S_0}{m+1} - K)\psi_k(y_m^*, b), & \text{for a call,} \\ \left(K - \frac{S_0}{m+1}\right)\psi_k(a, y_m^*) - \frac{S_0}{m+1}\chi_k(a, y_m^*), & \text{for a put.} \end{cases}$$

Coefficients C_k, approximated by $\hat{C}_k$, defined as

$$\hat{C}_k(y_l, y_u, t_m) = \frac{2}{b-a}\int_{y_l}^{y_u} \hat{c}(y, t_m)\cos(k\pi\frac{y-a}{b-a})dy, \tag{6.12}$$

are computed numerically. In the subsection to follow, we will show that, $\forall u \in \mathbb{R}$, the conditional characteristic function $\varphi_{Y_m|Y_{m-1}}(u; y_{m-1}, \Delta t)$ can be approximated by

$$\hat{\varphi}_{Y_m|Y_{m-1}}(u; y_{m-1}, \Delta t) \approx (1 + e^{Y_{m-1}})^{iu}\phi_Z(u; \Delta t), \tag{6.13}$$

and that the error from approximation (6.13) converges to zero only when $\mathcal{M}$ goes to infinity. The distribution of Z is identical to that of the logarithm of increment between (any) two consecutive time steps of a Lévy process.

Applying (6.5) and (6.13) in (6.12), gives us

$$\begin{aligned} \hat{C}_k(y_l, y_u, t_m) &= \int_{y_l}^{y_u} \hat{c}(y_m, t_m)\cos(k\pi\frac{y_m-a}{b-a})dy_m \qquad (6.14) \\ &= e^{-r\Delta t}Re\left(\sum_{j=0}^{N-1}{}' \phi_Z(\frac{j\pi}{b-a}; \Delta t)e^{-ij\pi\frac{a}{b-a}}\hat{V}_j(t_{m+1})\mathcal{H}_{k,j}(y_l, y_u)\right), \end{aligned}$$

where function $\mathcal{H}_{k,j}(y_l, y_u)$ is given by

$$\mathcal{H}_{k,j}(y_l, y_u) = \frac{2}{b-a}\int_{y_l}^{y_u} (1 + e^y)^{i\frac{j\pi}{b-a}}\cos(k\pi\frac{y-a}{b-a})dy. \tag{6.15}$$

With $\hat{C}(y_l, y_u, t_m) := \{\hat{C}_k(y_l, y_u, t_m)\}_{k=0}^{N-1}$, Equation (6.14) can be written in matrix–vector multiplication form, as

$$\hat{C}(y_l, y_u, t_m) = e^{-r\Delta t}Re(\mathcal{H}\cdot\mathbf{u}), \tag{6.16}$$

with $\mathcal{H} := \{\mathcal{H}_{k,j}\}_{k,j=0}^{N-1}$, $\mathbf{u} := \{\mathbf{u}_j\}_{j=0}^{N-1}$, and

$$\begin{aligned} \mathbf{u}_0 &= \frac{1}{2}\phi_Z(0; \Delta t)\hat{V}_0(t_{m+1}), \\ \mathbf{u}_j &= \phi_Z(\frac{j\pi}{b-a}; \Delta t)e^{-ij\pi\frac{a}{b-a}}\hat{V}_j(t_{m+1}),\ (j \neq 0). \end{aligned} \tag{6.17}$$

Integral $\mathcal{H}_{k,j}(y_l, y_u)$ in (6.15) can be rewritten in terms of Beta functions, however, the calculation of these Beta functions, with complex-valued arguments for all k, j, is computationally expensive. Therefore, as in [73], we will use the Clenshaw–Curtis quadrature rule to calculate the integrals $\mathcal{H}_{k,j}(y_l, y_u)$ in (6.15).

By recursion, we get the $\hat{V}_k(t_1)$-coefficients, and the value of an early–exercise arithmetic Asian option is given by

$$\hat{v}(y_0, t_0) = e^{-r\Delta t} \sum_{k=0}^{N-1}{}' Re(\phi_{\log(\frac{S_1}{S_0})}(\frac{k\pi}{b-a}; \Delta t) e^{-ik\pi\frac{a}{b-a}}) \hat{V}_k(t_1),$$

where $y_0 = \log(S_0)$ and $\phi_{\log(\frac{S_1}{S_0})}(u; t)$, $\forall u, t$, is known analytically for most Lévy processes.

At each time step the computational complexity is $O(n_q N^2)$ and $O(N)$ to compute the $\hat{C}_k$- and $\hat{G}_k$-terms, respectively, where n_q denotes the number of terms in the discrete cosine expansion of the Clenshaw–Curtis quadrature. In total, $O((\mathcal{M}-1) n_q N^2)$ computations are required for early-exercise arithmetic Asian options under Lévy processes.

Remark 6.3.1 (Error Analysis)**.** *With the conditional characteristic function well approximated, the error in the option price propagates basically in the same way as for a plain vanilla Bermudan option, for which we refer to [35], where a detailed error analysis was given.*

6.3.1 Characteristic function of the first pricing method

The fact that the pricing method explained is only highly accurate for $\mathcal{M} \to \infty$ will be presented in this section, where we derive the conditional characteristic function.

First, we give two lemmas which will be used later in our analysis.

Lemma 6.3.1. *A conditional characteristic function, $\varphi_{Y|X}(u; x, t)$, satisfies, $\forall X, Y$ with X and $Y - X$ independent, and $\forall u \in \mathbb{R}$,*

$$\varphi_{Y|X}(u; x, t) = e^{iuX} \phi_Z(u; t),$$

where Z and $Y - X$ are identically distributed, that is, $Z \overset{d}{=} Y - X$.

Proof. From the definition, we have

$$\varphi_{Y|X}(u; x, t) = \mathbb{E}(e^{iuY}|X) = e^{iuX} \mathbb{E}(e^{iu(Y-X)}|X).$$

With the notion of independence of X and $Y - X$, it follows that:

$$e^{iuX} \mathbb{E}(e^{iu(Y-X)}|X) = e^{iuX} \mathbb{E}(e^{iu(Y-X)}) = e^{iuX} \phi_{Y-X}(u; t) = e^{iuX} \phi_Z(u; t),$$

which proves the lemma. □

Lemma 6.3.2. *For random variables X, Y, with a bijective and bi-measurable function $h : \mathbb{R} \to \mathbb{R}$, we have that, $\forall u \in \mathbb{R}$,*

$$\varphi_{Y|X}(u; x, t) = \varphi_{Y|h(X)}(u; h(x), t). \tag{6.18}$$

Proof. The proof is straightforward as the σ-fields generated by X and $h(X)$ coincide. □

As a consequence of the Lemmas 6.3.2 and 6.3.1, we find that, for random variables X, Y and bijective function $h : \mathbb{R} \to \mathbb{R}$, with $h(X)$ and $Y - h(X)$ independent,

$$\varphi_{Y|X}(u; x, t) = e^{iuh(X)}\phi_Z(u; t), \ \forall u \in \mathbb{R} \tag{6.19}$$

where $Z \stackrel{d}{=} Y - h(X)$.

The basis for the efficient 1D Asian pricing algorithm is in the following lemma, which describes a relation between the characteristic functions of the state variables at consecutive time steps t_m and t_{m-1}, for Lévy processes.

Lemma 6.3.3. *If we define*

$$Y_m := \log(\frac{S_1}{S_0} + \frac{S_2}{S_0} + \cdots + \frac{S_m}{S_0}), \ m = \mathcal{M}, \mathcal{M} - 1, \cdots, 1,$$

then, at time step t_m, $m = \mathcal{M}, \cdots, 2$, we have, $\forall u \in \mathbb{R}$, in the case of a Lévy process, S_t, that

$$\phi_{Y_m}(u; t) = \phi_{\log(1+e^{Y_{m-1}})}(u; t)\phi_{Z_m}(u; t), \tag{6.20}$$

where

$$Z_m := \log(\frac{S_m}{S_{m-1}}). \tag{6.21}$$

Proof. For all $u \in \mathbb{R}$, we have

$$Y_m = \log(1 + e^{W_{m-1}}) + Z_1,$$

with

$$W_{m-1} := \log(\frac{S_2}{S_1} + \frac{S_3}{S_1} + \cdots + \frac{S_m}{S_1}).$$

Note that W_{m-1} and Z_1 are independent, as Lévy processes are defined by the property of independent increments. Therefore, $\forall u \in \mathbb{R}$,

$$\phi_{Y_m}(u; t) = \phi_{\log(1+e^{W_{m-1}})}(u; t)\phi_{Z_1}(u; t).$$

Moreover, a Lévy process has stationary increments, which implies that, $\forall u, m$, $\phi_{Z_1}(u; t) = \phi_{Z_m}(u; t)$, so that we find

$$\phi_{Y_m}(u; t) = \phi_{\log(1+e^{W_{m-1}})}(u; t)\phi_{Z_m}(u; t).$$

Now, we only need to prove that, $\forall u \in \mathbb{R}$,

$$\phi_{\log(1+e^{Y_{m-1}})}(u;t) = \phi_{\log(1+e^{W_{m-1}})}(u;t). \tag{6.22}$$

Here, $e^{Y_{m-1}}$ and $e^{W_{m-1}}$ can be rewritten as follows

$$\begin{aligned} e^{Y_{m-1}} &= e^{Z_1} + e^{Z_1+Z_2} + \cdots + e^{Z_1+\cdots+Z_{m-1}}, \\ e^{W_{m-1}} &= e^{Z_2} + e^{Z_2+Z_3} + \cdots + e^{Z_2+\cdots+Z_m}, \end{aligned}$$

where Z_m is defined in (6.21). For a Lévy process all $Z_j, j = 1, \cdots, \mathcal{M}$, are identically and independently distributed, so that $e^{Y_{m-1}} \overset{d}{=} e^{W_{m-1}}$, and $\forall u$, $\phi_{e^{Y_{m-1}}}(u;t) = \phi_{e^{W_{m-1}}}(u;t)$.

Equation (6.22) can be proved by the fact that for any two random variables, X, Y, if we have $\phi_X(u;t) = \phi_Y(u;t), \quad \forall u \in \mathbb{R}$, then, for any bijective and bi–measurable function $h : \mathbb{R} \to \mathbb{R}$, we have

$$\phi_{h(X)}(u;t) = \phi_{h(Y)}(u;t). \tag{6.23}$$

This concludes the proof. □

From (6.20) and since Y_{m-1} and Z_m are independent variables, as we work with Lévy processes, we have

$$\phi_{Y_m}(u;t) = \phi_{\log(1+e^{Y_{m-1}})+Z_m}(u;t), \forall u, t,$$

which implies that Y_m and $\log(1 + e^{Y_{m-1}}) + Z_m$ are identically distributed, i.e. $Y_m \overset{d}{=} \log(1 + e^{Y_{m-1}}) + Z_m$.

Let

$$\bar{Y}_m := \log(1 + e^{Y_{m-1}}) + Z_m,$$

so that $Y_m \overset{d}{=} \bar{Y}_m$. Conditional characteristic function $\varphi_{\bar{Y}_m|Y_{m-1}}(u; y_{m-1}, \Delta t)$ is known in closed form, that is, we set $h(Y_{m-1}) := \log(1 + e^{Y_{m-1}})$, and, from (6.19), $\forall y_{m-1}, u \in \mathbb{R}$, we find

$$\begin{aligned} \varphi_{\bar{Y}_m|Y_{m-1}}(u; y_{m-1}, \Delta t) &= \varphi_{\bar{Y}_m|h(Y_{m-1})}(u; h(y_{m-1}), \Delta t) \\ &= e^{iuh(Y_{m-1})}\phi_{\bar{Y}_m - h(Y_{m-1})}(u; \Delta t) \\ &= (1 + e^{Y_{m-1}})^{iu}\phi_{Z_m}(u; \Delta t). \end{aligned} \tag{6.24}$$

Our aim is to approximate the conditional characteristic function of Y_m, given Y_{m-1}, in terms of the conditional characteristic function in (6.24). Both Y_m and $\bar{Y}_m$ can be decomposed in terms of the increments between consecutive time steps, as follows

$$\begin{aligned} Y_m &= \log\Big(\sum_{j=1}^{m}(\Pi_{l=1}^{j}\frac{S_l}{S_{l-1}})\Big), \\ \bar{Y}_m &= \log\Big(\sum_{j=1}^{m-1}(\Pi_{l=1}^{j}\frac{S_l}{S_{l-1}}) + 1\Big) + \log(\frac{S_m}{S_{m-1}}). \end{aligned}$$

These increments are identically and independently distributed, depending only on the model parameters and Δt. Therefore, Y_m and $\bar{Y}_m$ are both functions of Δt. In the following lemma we will show that as $\mathcal{M} \to \infty$, that is, as Δt goes to zero, we have $Y_m \to \bar{Y}_m$, $\forall m = 1, \cdots$, and we can use $\varphi_{\bar{Y}_m|Y_{m-1}}(u; y_{m-1}, \Delta t)$ to approximate $\varphi_{Y_m|Y_{m-1}}(u; y_{m-1}, \Delta t)$ at each time step.

Lemma 6.3.4. *As $\Delta t \to 0$, that is, as $t_k - t_{k-1} \to 0$, $\forall k = 1, \cdots, \mathcal{M}$, we have that, $\forall m = 1, \cdots, \mathcal{M}$,*

$$\log(\sum_{j=1}^{m}(\Pi_{l=1}^{j}\frac{S_l}{S_{l-1}})) \to \log(\sum_{j=1}^{m-1}(\Pi_{l=1}^{j}\frac{S_l}{S_{l-1}}) + 1) + \log(\frac{S_m}{S_{m-1}}).$$

In other words, as $\mathcal{M} \to \infty$, then, $Y_m \to \bar{Y}_m$, $\forall m = 1, \cdots, \mathcal{M}$.

Proof.

$$\begin{aligned}
\exp(Y_m) - \exp(\bar{Y}_m) &= (\frac{S_1}{S_0} - \frac{S_m}{S_{m-1}}) + (\frac{S_2}{S_1} - \frac{S_m}{S_{m-1}})\frac{S_1}{S_0} \\
&+ (\frac{S_3}{S_2} - \frac{S_m}{S_{m-1}})\frac{S_2}{S_0} + \cdots + (\frac{S_{m-1}}{S_{m-2}} - \frac{S_m}{S_{m-1}})\frac{S_{m-2}}{S_0} \\
&= \sum_{j=1}^{m-1}(\frac{S_j}{S_{j-1}} - \frac{S_m}{S_{m-1}})\frac{S_{j-1}}{S_0}. \qquad (6.25)
\end{aligned}$$

The mean and variance of $\log(S_j/S_{j-1})$, $j = 1, \cdots, \mathcal{M}$, under a Lévy process, are of the form $J_\mu \Delta t$ and $J_\sigma \Delta t$, where J_μ, J_σ are constants, independent of Δt. Therefore, as $\mathcal{M} \to \infty$, $\Delta t \to 0$, the mean and variance of $\log(S_j/S_{j-1})$ will tend to zero, so that $\log(S_j/S_{j-1}) \to 0$.

Function $\exp(x), x \in \mathbb{R}$, is a continuous function with respect to x, therefore, we have $S_j/S_{j-1} \to 1$, $\forall j = 1, \cdots, \mathcal{M}$, and thus

$$\frac{S_j}{S_{j-1}} - \frac{S_m}{S_{m-1}} \to 0, \forall j. \qquad (6.26)$$

Moreover, the term $\frac{S_{j-1}}{S_0}$ is independent of the term $\frac{S_j}{S_{j-1}} - \frac{S_m}{S_{m-1}}$, so that, from (6.25), we find

$$\exp(Y_m) - \exp(\bar{Y}_m) \to 0,$$

or, in other words, $\exp(Y_m) \to \exp(\bar{Y}_m)$ as $\mathcal{M} \to \infty$, and we can conclude that $Y_m \to \bar{Y}_m$, because $\log(x)$ is a continuous function in $x > 0$.

This concludes the proof. □

As an approximation, we can thus use

$$\begin{aligned}
\hat{\varphi}_{Y_m|Y_{m-1}}(u; y_{m-1}, \Delta t) &\approx \varphi_{\bar{Y}_m|Y_{m-1}}(u; y_{m-1}, \Delta t) \\
&= (1 + e^{Y_{m-1}})^{iu}\phi_{Z_m}(u; \Delta t). \qquad (6.27)
\end{aligned}$$

For Lévy processes all Z_m-terms are identically distributed. Thus

$$\phi_{Z_m}(u;t) =: \phi_Z(u;t),$$

and we replace $\phi_{Z_m}(u;t)$ by $\phi_Z(u;t)$ in (6.27).

This concludes our discussion of the first pricing method. Numerical experiments, comparing the performance of this method with that of the 2D method presented in the section to follow, for a small and large number of early-exercise dates, will be presented in the section with numerical experiments.

6.4 The 2D ASCOS method for early-exercise Asian options

In this section we present *a 2D pricing algorithm for early–exercise Asian options*, which can be used for all Lévy processes with any number of early–exercise dates. Calculations of the continuation value and the Fourier coefficients at each time step are discussed, respectively, in Subsections 6.4.1 and 6.4.2. The method appears to be more robust than the method from the previous section, but also somewhat more expensive.

6.4.1 Continuation value

At each time step $m = \mathcal{M}, \cdots, 1$, we use in this case the variables

$$y_m := \frac{S_1}{S_0} + \cdots + \frac{S_m}{S_0}, \quad x_m := \log(\frac{S_m}{S_0}),$$

and we have

$$y_m = y_{m-1} + e^{x_m}. \tag{6.28}$$

From the risk–neutral evaluation formula, where the continuation value is derived as the discounted expected option price at the next time step, we now use a 2D version, as follows, for $m = \mathcal{M}, \cdots, 1$,

$$\begin{aligned} c(y_{m-1}, x_{m-1}, t_{m-1}) &= e^{-r\Delta t}\mathbb{E}(v(y_m, x_m, t_m)|y_{m-1}, x_{m-1}) \\ &= e^{-r\Delta t}\int_{\mathbb{R}}\int_{\exp(x_m)}^{+\infty} v(y_m, x_m, t_m) f(y_m, x_m|y_{m-1}, x_{m-1}) dy_m dx_m. \end{aligned} \tag{6.29}$$

where the integration range of y_m comes from (6.28) and that $y_{m-1} \geq 0$.

Truncating the integration range, gives us

$$\hat{c}(y_{m-1}, x_{m-1}, t_{m-1}) = e^{-r\Delta t}\int_{a_1}^{b_1}\int_{\exp(x_m)}^{b_2} v(y_m, x_m, t_m) f(y_m, x_m|y_{m-1}, x_{m-1}) dy_m dx_m, \tag{6.30}$$

where $[a_1, b_1]$ and $[\exp(x_m), b_2]$ are the integration ranges for x_m and y_m, respectively. Integration range $[a_1, b_1]$ is calculated the same way as presented in [35], and the calculation of b_2 will be explained in Subsection 6.4.4. By applying the *chain rule* to the joint conditional density function in (6.30), we find

$$\begin{aligned} f(y_m, x_m|y_{m-1}, x_{m-1}) &= f(y_m|x_m, y_{m-1}, x_{m-1}) \cdot f(x_m|y_{m-1}, x_{m-1}) \\ &= f(y_m|x_m, y_{m-1}) \cdot f(x_m|x_{m-1}). \end{aligned} \tag{6.31}$$

By inserting (6.31) into (6.30), the risk–neutral formula becomes

$$\begin{aligned} &\hat{c}(y_{m-1}, x_{m-1}, t_{m-1}) = \\ &e^{-r\Delta t} \int_{a_1}^{b_1} \int_{\exp(x_m)}^{b_2} v(y_m, x_m, t_m) f(y_m|x_m, y_{m-1}) \cdot f(x_m|x_{m-1}) dy_m dx_m \end{aligned} \tag{6.32}$$

Although the conditional density function is not known analytically for many Lévy processes, the corresponding characteristic function is. Based on this, we approximate the conditional density function by a truncated Fourier cosine expansion based on the characteristic function, as follows,

$$\begin{aligned} \hat{f}(x_m|x_{m-1}) &= \frac{2}{b_1 - a_1} {\sum_{k=0}^{N_1-1}}' Re \left(\phi_{x_m - x_{m-1}}(\frac{k\pi}{b_1 - a_1}; \Delta t) \cdot \right. \\ &\left. \exp\left(ik\pi \frac{x_{m-1} - a_1}{b_1 - a_1}\right) \right) \cos(k\pi \frac{x_m - a_1}{b_1 - a_1}), \end{aligned} \tag{6.33}$$

and

$$\begin{aligned} \hat{f}(y_m|x_m, y_{m-1}) &= {\sum_{j=0}^{N_2-1}}' \frac{2}{b_2 - \exp(x_m)} Re \left(\exp\left(i \frac{j\pi}{b_2 - \exp(x_m)} y_{m-1}\right) \right) \\ &\cos(j\pi \frac{y_m - \exp(x_m)}{b_2 - \exp(x_m)}). \end{aligned} \tag{6.34}$$

where (6.34) is based on Equation (6.28). Note that for Lévy processes, defined by independent and identical increments, the (unconditional) characteristic functions of all increments of consecutive time steps, i.e. $\phi_{x_m - x_{m-1}}(u; \Delta t)$, are the same, for all time steps, and are known analytically [1]. Therefore, we use the notation $\phi(u; \Delta t) := \phi_{x_m - x_{m-1}}(u; \Delta t)$ for all time steps.

By replacing the two density functions in (6.32) by their approximations in (6.33) and (6.34), and then interchanging the order of summation and

[1] Compared to the previous section and the 1D pricing method, we have reduced the number of arguments for $\phi(\cdot)$ from three to two. So, for the conditional characteristic function we have used $\phi(u; x, t)$, whereas for the unconditional characteristic function, or if we deal with independent increments, as in the present section, we use $\phi(u; t)$.

integration, we obtain,

$$
\begin{aligned}
&\hat{c}(y_{m-1}, x_{m-1}, t_{m-1}) = e^{-r\Delta t}\frac{2}{b_1-a_1}\sum_{k=0}^{N_1-1}{}'\sum_{j=0}^{N_2-1}{}'\int_{a_1}^{b_1}\int_{\exp(\delta_n)}^{b_2}\hat{v}(y_m, x_m, t_m)\cdot \\
&Re\left(\phi(\frac{k\pi}{b_1-a_1};\Delta t)\exp\left(ik\pi\frac{x_{m-1}-a_1}{b_1-a_1}\right)\right)\cos(k\pi\frac{x_m-a_1}{b_1-a_1})\cdot \\
&\frac{2}{b_2-\exp(x_m)}Re\left(\exp\left(i\frac{j\pi}{b_2-\exp(\delta_n)}y_{m-1}\right)\right)\cos(j\pi\frac{y_m-\exp(x_m)}{b_2-\exp(x_m)})dy_m dx_m \\
= \;& e^{-r\Delta t}\frac{2}{b_1-a_1}\sum_{k=0}^{N_1-1}{}'\sum_{j=0}^{N_2-1}{}'Re\left(\phi(\frac{k\pi}{b_1-a_1};\Delta t)\exp\left(ik\pi\frac{x_{m-1}-a_1}{b_1-a_1}\right)\right)\cdot \\
&Re\left(\int_{a_1}^{b_1}\frac{2}{b_2-\exp(x_m)}\exp\left(i\frac{j\pi}{b_2-\exp(x_m)}y_{m-1}\right)\cos(k\pi\frac{x_m-a_1}{b_1-a_1})\cdot\right. \\
&\left.\int_{\exp(\delta_n)}^{b_2}\hat{v}(y_m, x_m, t_m)\cos(j\pi\frac{y_m-\exp(x_m)}{b_2-\exp(x_m)})dy_m dx_m\right). \qquad (6.35)
\end{aligned}
$$

For the integration over x_m in (6.35) numerical approximation is required, for which Clenshaw–Curtis quadrature is employed here. Function

$$
\frac{2}{b_2-\exp(x_m)}\exp\left(i\frac{j\pi}{b_2-\exp(x_m)}y_{m-1}\right)\cos(k\pi\frac{x_m-a_1}{b_1-a_1})\cos(j\pi\frac{y_m-\exp(x_m)}{b_2-\exp(x_m)})
$$

is smoothly varying[2] in x_m and the same is true for the option value, $\hat{v}(y_m, x_m, t_m)$, for all $m < \mathcal{M}$. At $t_{\mathcal{M}} = T$, $v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}})$ is only a function of $y_{\mathcal{M}}$, i.e. $v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) \equiv g(y_{\mathcal{M}}, t_{\mathcal{M}})$. Because of these properties, we expect an exponential convergence for this quadrature.

Note that both the Clenshaw–Curtis and Gaussian quadrature rules exhibit exponential convergence for the integral under consideration, however, the Clenshaw–Curtis quadrature appears to be computationally somewhat cheaper. The weights and nodes of the Clenshaw-Curtis quadrature are easy to calculate and they form a nested sequence. We refer the reader to [22] and [9] for more information about Clenshaw–Curtis quadrature.

[2]That is, the function is countinuous in x_m and so its derivatives with resepct to x_m.

In detail, for the approximation by Clenshaw–Curtis quadrature, we have

$$\begin{aligned}
&\int_{a_1}^{b_1} \frac{2}{b_2-\exp(x_m)} \exp\left(i\frac{j\pi}{b_2-\exp(x_m)}y_{m-1}\right)\cos(k\pi\frac{x_m-a_1}{b_1-a_1}) \\
&\int_{\exp(\delta_n)}^{b_2} \hat{v}(y_m,x_m,t_m)\cos(j\pi\frac{y_m-\exp(x_m)}{b_2-\exp(x_m)})dy_m dx_m \\
\approx\ & \frac{b_1-a_1}{2}\sum_{n=1}^{n_q+2} w_n \frac{2}{b_2-\exp(\delta_n)} \exp\left(i\frac{j\pi}{b_2-\exp(\delta_n)}y_{m-1}\right)\cos(k\pi\frac{\delta_n-a_1}{b_1-a_1}) \\
\cdot\ & \int_{\exp(\delta_n)}^{b_2} \hat{v}(y_m,\delta_n,t_m)\cos(j\pi\frac{y_m-\exp(\delta_n)}{b_2-\exp(\delta_n)})dy_m,
\end{aligned}$$

where

$$\delta_n = \begin{cases} \dfrac{b_1-a_1}{2}\cos(\dfrac{n\pi}{n_q})+\dfrac{b_1+a_1}{2}, & n=\{0,\cdots,n_q/2\}, \\ \dfrac{a_1-b_1}{2}\cos\left(\dfrac{(n-(n_q/2+1))\pi}{n_q}\right)+\dfrac{b_1+a_1}{2}, & n=\{n_q/2+1,\cdots,n_q+1\} \end{cases} \tag{6.36}$$

and w is an (n_q+2)-vector, defined as $w := \{w_n\}_{n=1}^{n_q+2} = [D^T d; D^T d]$, with D an $(\frac{n_q}{2}+1)\times(\frac{n_q}{2}+1)$-matrix, with elements

$$D(n_1,n_2) = \frac{2}{n_q}\cos\left(\frac{(n_1-1)(n_2-1)\pi}{n_q/2}\right)\cdot\begin{cases} 1/2, & n_2=\{1,n_q/2+1\}, \\ 1, & \text{otherwise}, \end{cases}$$

and vector d reads

$$d = (1, \frac{2}{1-4}, \frac{2}{1-16}, \cdots, \frac{2}{1-(n_q-2)^2}, \frac{1}{1-n_q^2})^T.$$

Note that, $\forall k,j,m$, the values of δ_n, w_n are the same, in other words, they only need to be calculated once and can be used for all k,j and for all time steps.

Inserting (6.36) in (6.35) gives us the formula for the continuation value at each time step, as follows

$$\begin{aligned}
&\hat{c}(y_{m-1},x_{m-1},t_{m-1}) = e^{-r\Delta t}\sum_{k=0}^{N_1-1}{}'\sum_{j=0}^{N_2-1}{}' Re\left(\phi(\frac{k\pi}{b_1-a_1};\Delta t)\exp\left(ik\pi\frac{x_{m-1}-a_1}{b_1-a_1}\right)\right) \\
\cdot\ & Re\left(\sum_{n=1}^{n_q+2}\frac{2}{b_2-\exp(\delta_n)}w_n\exp\left(i\frac{j\pi}{b_2-\exp(\delta_n)}y_{m-1}\right)\cos(k\pi\frac{\delta_n-a_1}{b_1-a_1})\right. \\
\cdot\ & \left.\int_{\exp(\delta_n)}^{b_2}\hat{v}(y_m,\delta_n,t_m)\cos(j\pi\frac{y_m-\exp(\delta_n)}{b_2-\exp(\delta_n)})dy_m\right).
\end{aligned}$$

By denoting

$$\hat{V}_{n,j}(t_m) := \int_{\delta_n}^{b_2} \hat{v}(y_m, \delta_n, t_m) \cos(j\pi \frac{y_m - \exp(\delta_n)}{b_2 - \exp(\delta_n)}) dy_m, \tag{6.37}$$

with δ_n as in (6.36), we have

$$\hat{c}(y_{m-1}, x_{m-1}, t_{m-1}) = e^{-r\Delta t} \sum_{k=0}^{N_1-1}{}' \sum_{j=0}^{N_2-1}{}' Re\left(\phi(\frac{k\pi}{b_1 - a_1}; \Delta t) \exp\left(ik\pi \frac{x_{m-1} - a_1}{b_1 - a_1}\right)\right) \cdot$$
$$Re\left(\sum_{n=1}^{n_q+2} \frac{2}{b_2 - \exp(\delta_n)} w_n \exp\left(i \frac{j\pi}{b_2 - \exp(\delta_n)} y_{m-1}\right) \cos(k\pi \frac{\delta_n - a_1}{b_1 - a_1}) \hat{V}_{n,j}(t_m)\right). \tag{6.38}$$

From (6.37) we see that the computational complexity to compute the continuation value at each time step is $O(N_1 N_2 n_q)$.

The 2D pricing algorithm is based on backward recursion of the Fourier coefficients $\hat{V}_{n,j}(t_m)$, defined in (6.37). The early–exercise Asian option price, $\hat{v}(x_0, t_0) = \hat{c}(y_0, x_0, t_0)$ is obtained by taking $m = 1$ and inserting $\hat{V}_{n,j}(t_1)$ in (6.38). In the next subsection we will show that the $V_{n,j}(t_{\mathcal{M}})$ are known analytically. For $m = \mathcal{M} - 1, \cdots, 1$, $\hat{V}_{n,j}(t_m)$ can be recovered from $\hat{V}_{n,j}(t_{m+1})$.

6.4.2 Fourier coefficients

At maturity time, $t_{\mathcal{M}} = T$, the option value equals the payoff, so that $\forall n, j$,

$$V_{n,j}(t_{\mathcal{M}}) := \int_{\exp(\delta_n)}^{b_2} g(y_{\mathcal{M}}, t_{\mathcal{M}}) \cos(j\pi \frac{y_{\mathcal{M}} - \exp(\delta_n)}{b_2 - \exp(\delta_n)}) dy_{\mathcal{M}},$$

where, $\forall m = 1, \cdots, \mathcal{M}$,

$$g(y_m, t_m) = \begin{cases} (\frac{S_0(1+y_m)}{m+1} - K)^+, & \text{for a call,} \\ (K - \frac{S_0(1+y_m)}{m+1})^+, & \text{for a put.} \end{cases} \tag{6.39}$$

Thus, the Fourier coefficients at maturity read

$$V_{n,j}(t_{\mathcal{M}}) = \begin{cases} \frac{S_0}{\mathcal{M}+1} \varsigma_j(y^*_{\mathcal{M},n}, b_2) + (\frac{S_0}{\mathcal{M}+1} - K)\psi_j(y^*_{\mathcal{M},n}, b_2), & \text{for a call,} \\ (K - \frac{S_0}{\mathcal{M}+1})\psi_j(\exp(\delta_n), y^*_{\mathcal{M},n}) - \frac{S_0}{\mathcal{M}+1}\varsigma_j(\exp(\delta_n), y^*_{\mathcal{M},n}), & \text{for a put,} \end{cases}, \tag{6.40}$$

where $y^*_{\mathcal{M},n} \equiv \frac{K(\mathcal{M}+1)}{S_0} - 1$, $\psi_j(y_l, y_u)$ is as in (6.9), and

$$\varsigma_j(y_l, y_u) = \int_{y_l}^{y_u} y \cos\left(j\pi \frac{y - \exp(\delta_n)}{b_2 - \exp(\delta_n)}\right) dy. \tag{6.41}$$

Both $\psi_j(y_l, y_u)$ and $\varsigma_j(y_l, y_u)$ are known analytically, and so is $V_{n,j}(t_{\mathcal{M}})$. The recursive step is presented in the following result.

Result 6.4.1. *For $t_m, m = \mathcal{M}-1.\cdots, 1$, the continuation value, $c(y_m, x_m, t_m)$, and the Fourier cosine coefficients, $V_{n,j}(t_m)$, can be obtained from $V_{n,j}(t_{m+1})$. By this approach, the Fourier coefficients $V_{n,j}(t_1)$ are recovered recursively.*

Proof. For $m = \mathcal{M}-1.\cdots, 1$, first of all, the early–exercise points, $y^*_{m,n}$, for which $c(y^*_{m,n}, \delta_n, t_m) = g(y^*_{m,n}, t_m)$, δ_n as in (6.36), need to be determined by means of Newton's method. Here, the payoff function is calculated in (6.39), and the continuation value is derived via

$$\begin{aligned}&\hat{c}(y_m, \delta_n, t_m) = \\ &e^{-r\Delta t}\sum_{k=0}^{N_1-1}{}'\sum_{j=0}^{N_2-1}{}' Re\left(\phi(\frac{k\pi}{b_1-a_1};\Delta t)\exp\left(ik\pi\frac{\delta_n-a_1}{b_1-a_1}\right)\right)\cdot \\ &Re\left(\sum_{p=1}^{n_q+2}\frac{2}{b_2-\exp(\delta_p)}w_p\exp\left(i\frac{j\pi}{b_2-\exp(\delta_p)}y_m\right)\cos(k\pi\frac{\delta_p-a_1}{b_1-a_1})\hat{V}_{p,j}(t_{m+1})\right).\end{aligned} \tag{6.42}$$

which is directly obtained from (6.38). Furthermore, the derivative of the continuation value and that of the payoff function with respect to y_m can be easily computed by (6.42) and (6.39). From [35] we know that typically after approximately five Newton iterations, the error in $y^*_{m,n}$ is $O(10^{-10})$.

As a next step, $\hat{V}_{n,j}(t_m)$ is split by means of these early–exercise points, as follows

$$\hat{V}_{n,j}(t_m) = \begin{cases}\hat{C}_{n,j}(\exp(\delta_n), y^*_{m,n}, t_m) + G_{n,j}(y^*_{m,n}, b_2, t_m), & \text{for a call,}\\ G_{n,j}(\exp(\delta_n), y^*_{m,n}, t_m) + \hat{C}_{n,j}(y^*_{m,n}, b_2, t_m), & \text{for a put,}\end{cases} \tag{6.43}$$

where $\hat{C}_{n,j}, G_{n,j}$ are Fourier cosine coefficients of the continuation value and payoff at t_m, respectively. Coefficient $G_{n,j}$ is of the form,

$$G_{n,j}(t_m) = \begin{cases}\frac{S_0}{m+1}\varsigma_j(y^*_{m,n}, b_2) + (\frac{S_0}{m+1}-K)\psi_j(y^*_{m,n}, b_2), & \text{for a call,}\\ (K-\frac{S_0}{m+1})\psi_j(\exp(\delta_n), y^*_{m,n}) - \frac{S_0}{m+1}\varsigma_j(\exp(\delta_n), y^*_{m,n})), & \text{for a put,}\end{cases} \tag{6.44}$$

and coefficient $\hat{C}_k$, defined by

$$\hat{C}_{n,j}(y_l, y_u, t_m) = \int_{y_l}^{y_u}\hat{c}(y_m, \delta_n, t_m)\cos(j\pi\frac{y_m-\exp(\delta_n)}{b_2-\exp(\delta_n)})dy_m, \tag{6.45}$$

with integration range $[y_l, y_u] \in [\delta_n, b_2]$, is computed numerically.

By substituting for $\hat{c}(y_m, \delta_n, t_m)$ in (6.45) its expression in (6.42) and interchanging integration and summation, we obtain

$$\begin{aligned}
&\hat{C}_{n,j}(y_l, y_u, t_m) = \\
&e^{-r\Delta t} \sum_{k=0}^{N_1-1}{}' \sum_{l=0}^{N_2-1}{}' Re\left(\phi(\frac{k\pi}{b_1-a_1}; \Delta t) \exp\left(ik\pi\frac{\delta_n - a_1}{b_1-a_1}\right)\right) \\
\cdot\ &Re\left(\sum_{p=1}^{n_q+2} \Lambda(k,l,p) \int_{y_l}^{y_u} \exp\left(i\frac{l\pi}{b_2-\exp(\delta_p)} y_m\right) \cos(j\pi\frac{y_m - \exp(\delta_p)}{b_2-\exp(\delta_p)}) dy_m\right),
\end{aligned} \tag{6.46}$$

where

$$\Lambda(k,l,p) := \frac{2}{b_2-\exp(\delta_p)} w_p \cos(k\pi\frac{\delta_p - a_1}{b_1-a_1}) \hat{V}_{p,l}(t_{m+1}). \tag{6.47}$$

The integral in (6.46) is known analytically. We have, $\forall y_l, y_u, l, j,\ j, l = 0, \cdots, N_2 - 1,\ j \neq l$,

$$\begin{aligned}
&\int_{y_l}^{y_u} \exp\left(i\frac{l\pi}{b_2-\exp(\delta_p)} y_m\right) \cos(j\pi\frac{y_m - \exp(\delta_p)}{b_2-\exp(\delta_p)}) dy_m \\
=\ & \frac{1}{j^2-l^2}\frac{d-c}{\pi}(\exp\left(\frac{il\pi}{b_2-\exp(\delta_p)} y_u\right) \sin(j\pi\frac{y_u - \exp(\delta_p)}{b_2-\exp(\delta_p)}) \\
-\ & \exp\left(\frac{il\pi}{b_2-\exp(\delta_p)} y_l\right) \sin(j\pi\frac{y_l - \exp(\delta_p)}{b_2-\exp(\delta_p)}) \\
+\ & il(\exp\left(\frac{il\pi}{b_2-\exp(\delta_p)} y_u\right) \cos(j\pi\frac{y_u - \exp(\delta_p)}{b_2-\exp(\delta_p)}) - \exp\left(\frac{il\pi}{b_2-\exp(\delta_p)} y_l\right) \\
\cdot\ & \cos(j\pi\frac{y_l - \exp(\delta_p)}{b_2-\exp(\delta_p)}))),
\end{aligned}$$

and, if $j = l, j \neq 0, l \neq 0$,

$$\begin{aligned}
&\int_{y_l}^{y_u} \exp\left(i\frac{l\pi}{b_2-\exp(\delta_p)} y_m\right) \cos(j\pi\frac{y_m - \exp(\delta_p)}{b_2-\exp(\delta_p)}) dy_m \\
&= \frac{\exp\left(il\pi\frac{\exp(\delta_p)}{b_2-\exp(\delta_p)}\right)}{2}(y_u - y_l) + (-\frac{i}{\pi})\frac{b_2-\exp(\delta_p)}{2} \exp\left(il\pi\frac{\exp(\delta_p)}{b_2-\exp(\delta_p)}\right) \cdot \\
&\frac{\exp(i(j+l)\frac{y_u-\exp(\delta_p)}{b_2-\exp(\delta_p)}\pi) - \exp(i(j+l)\frac{y_l-\exp(\delta_n)}{b_2-\exp(\delta_p)}\pi)}{j+l},
\end{aligned}$$

and, finally, for $l = j = 0$,

$$\int_{y_l}^{y_u} \exp\left(i\frac{l\pi}{b_2-\exp(\delta_p)} y_m\right) \cos(j\pi\frac{y_m - \exp(\delta_p)}{b_2-\exp(\delta_p)}) dy_m = y_u - y_l.$$

Therefore, Fourier coefficients $\hat{C}_{n,j}(y_l, y_u, t_m)$ can be calculated directly from (6.46) without additional numerical techniques.

From (6.42) and (6.46) we can observe that the continuation value as well as the Fourier coefficients at $t_m, m = \mathcal{M}-1, \cdots, 1$, can be recovered from the Fourier coefficients at t_{m+1}. This concludes the proof and the $V_{n,j}(t_1)$, $\forall n, j$ are recovered at the end of backward recursion. □

The value of the Asian option with early-exercise features is then obtained by inserting $V_{n,j}(t_1)$ into (6.38).

6.4.3 Computational complexity and Fast Fourier Transform

Newton's method is applied to determine the $y^*_{m,n}$-values with $n = 1, \cdots, n_q+2$. For this purpose the continuation value $\hat{c}(y_m, \delta_n, t_m)$ must be computed by (6.42). Term

$$Re\left(\sum_{p=1}^{n_q+2} \frac{2}{b_2 - \exp(\delta_p)} w_p \exp\left(i\frac{j\pi}{b_2 - \exp(\delta_p)} y_m\right) \cos\left(k\pi \frac{\delta_p - a_1}{b_1 - a_1}\right) \hat{V}_{p,j}(t_{m+1})\right).$$

in (6.42) is calculated once and can be reused in all iteration steps and for all δ_n. Therefore, we perform $O(N_1 N_2 n_q)$ computations to determine $y^*_{m,1}$, and to compute $y^*_{m,n}$, $n = 2, \cdots, n_q+2$, only $O(N_1 N_2)$ computations are needed. We end up with $O(N_1 N_2 n_q)$ compuations to determine all the early–exercise points.

Furthermore, to compute $\hat{C}_{n,j}(y_l, y_u, t_m)$ each time step we perform $O(N_1 N_2 n_q)$ computations, as the integration in (6.46) has an analytically known solution. We need to calculate $\hat{C}_{n,j}(y_l, y_u, t_m)$ for each value of n and j. Term

$$Re\left(\sum_{p=1}^{n_q+2} \Lambda(k,l,p) \int_{y_l}^{y_u} \exp\left(i\frac{l\pi}{b_2 - \exp(\delta_p)} y_m\right) \cos\left(j\pi \frac{y_m - \exp(\delta_p)}{b_2 - \exp(\delta_p)}\right) dy_m\right) \tag{6.48}$$

in (6.46) need not be re–computed for different n, and we have $O(N_1 N_2 n_q)$ computations in total for all values of n, with $n = 1, \cdots, n_q+2$. To determine all Fourier coefficients, $\hat{C}_{n,j}(y_l, y_u, t_m)$, with $j = 0, \cdots, N_2 - 1$, we require in total $O(N_1 N_2^2 n_q)$ computations, at each time step.

We need to repeat all the computations for time steps, $m = \mathcal{M}-1, \cdots, 1$, so that the overall computational complexity for the pricing technique is $O((\mathcal{M}-1) N_1 N_2^2 n_q)$.

The Fast Fourier Transform (FFT) *can however be employed* to reduce

this computational complexity. Equation (6.46) can be rewritten, $\forall k, p$, as

$$\begin{aligned}\hat{C}_{n,j}^{k,p} &= e^{-r\Delta t}\sum_{l=0}^{N_2-1}{}' Re\left(\phi(\frac{k\pi}{b_1-a_1};\Delta t)\exp\left(ik\pi\frac{\delta_n-a_1}{b_1-a_1}\right)\right)\\ &\cdot\ Re\left(\Lambda(k,l,p)\int_{y_l}^{y_u}\exp\left(i\frac{l\pi}{b_2-\exp(\delta_p)}y_m\right)\cos(j\pi\frac{y_m-\exp(\delta_p)}{b_2-\exp(\delta_p)})dy_m\right),\end{aligned} \tag{6.49}$$

If we denote vector $\hat{\mathbf{C}}_n^{k,p} := \{\hat{C}_{n,j}^{k,p}\}_{j=0}^{N_2-1}$, then it is well-known, that,

$$\begin{aligned}\hat{\mathbf{C}}_n^{k,p} &= \frac{e^{-r\Delta t}}{\pi}Re(\phi(\frac{k\pi}{b_1-a_1};\Delta t)\exp\left(ik\pi\frac{\exp(\delta_n)-a_1}{b_1-a_1}\right))\\ &\quad Im((H^c(y_l,y_u)+H^s(y_l,y_u))\mathbf{u}),\end{aligned} \tag{6.50}$$

where $Im(\cdot)$ denotes taking the imaginary part of the input argument, and

$$\mathbf{u} = \Lambda(k,l,p)\exp\left(\frac{il\pi\exp(\delta_p)}{b_2-\exp(\delta_p)}\right).$$

Moreover, H^c and H^s have a Hankel and Toeplitz structure, respectively, with elements as follows,

$$H_{j,l}^c(x_1,x_2) = \begin{cases} \dfrac{(x_2-x_1)\pi i}{b_2-\exp(\delta_p)}, & \text{if } j=l=0,\\ \dfrac{1}{(l+j)}\left[\exp\left(\dfrac{((l+j)x_2-(l+j)\exp(\delta_p))\pi i}{b_2-\exp(\delta_p)}\right)-\right. & \\ \left.\exp\left(\dfrac{((l+j)x_1-(l+j)\exp(\delta_p))\pi i}{b_2-\exp(\delta_p)}\right)\right], & \text{otherwise,}\end{cases} \tag{6.51}$$

and

$$H_{j,l}^s(x_1,x_2) = \begin{cases} \dfrac{(x_2-x_1)\pi i}{b_2-\exp(\delta_p)}, & \text{if } j=l=0,\\ \dfrac{1}{(l-j)}\left[\exp\left(\dfrac{((l-j)x_2-(l-j)\exp(\delta_p))\pi i}{b_2-\exp(\delta_p)}\right)-\right. & \\ \left.\exp\left(\dfrac{((l-j)x_1-(l-j)\exp(\delta_p))\pi i}{b_2-\exp(\delta_p)}\right)\right], & \text{otherwise.}\end{cases} \tag{6.52}$$

From [35] we know that the FFT can be used to calculate matrix–vector multiplications in (6.50).

To compute vector $\hat{\mathbf{C}}_n^{k,p}$ for each pair of (k,p), with $k=0,\cdots,N_1-1$, $p=1,\cdots,n_q+2$, $O(N_2\log_2 N_2)$ computations are performed. Therefore in total we need $O(N_1N_2\log_2 N_2n_q)$ computations to compute $\hat{\mathbf{C}}_n^{k,p}$ for all k,p. Furthermore, term $Im((H^c+H^s)\mathbf{u})$ can be reused for all n, with $n=1,n_q+2$, and in total, $O(N_2\log_2 N_2)$ computations are needed for all Fourier coefficients.

At the final stage of the algorithm, we need to add up all $k \times p$ elements, that is

$$\hat{C}_{n,j}(y_l, y_u, t_m) = \sum_{k=0}^{N_1-1}{}' \sum_{p=1}^{n_q+2} \hat{C}_{n,j}^{k,q}(y_l, y_u, t_m), \tag{6.53}$$

with $\hat{C}_{n,j}^{k,q}(y_l, y_u, t_m)$ defined in (6.49) and computed by (6.50).

Define from (6.50) that

$$\begin{aligned} A_1(k,n) &:= \frac{e^{-r\Delta t}}{\pi} Re(\phi(\frac{k\pi}{b_1 - a_1}; \Delta t) \exp\left(ik\pi \frac{\exp(\delta_n) - a_1}{b_1 - a_1}\right)) \\ A_2(k,p) &:= Im((H^c(y_l, y_u) + H^s(y_l, y_u))\mathbf{u}) \end{aligned}$$

and (6.53) can be computed in an efficient way as summarized below.

ALGORITHM: Efficient computation of (6.53)

For $j = 0, \cdots, N_2 - 1$, compute

- Step 1: Compute
$$A_2(k) := \sum_{p=1}^{n_q+2} A_2(k,p)$$
.
- Step 2: For $n = 1, \cdots, n_q + 2$, compute
$$\hat{C}_{n,j} := \sum_{k=0}^{N_1-1}{}' A_1(k,n) * A_2(k)$$
.

For each j, with $j = 0, \cdots, N_2 - 1$, we have $O(n_q)$ computations for step 1, and $O(N_1 n_q)$ computations for step 2. Therefore in total, $O(N_1 N_2 n_q)$ computations are needed for summation (6.53). By the use of the FFT, the computational complexity at each time step is then reduced to $O(N_1 N_2 \log_2 N_2 n_q)$.

The overall 2D ASCOS pricing algorithm is summarized below.

ASCOS ALGORITHM: Pricing early–exercise arithmetic Asian options.

Initialization

- For $n = 1, \cdots, n_q + 2$, $j = 0, \cdots, N_2 - 1$, compute $V_{n,j}(t_{\mathcal{M}})$ from (6.40).

Main Loop to Recover $\hat{V}_{n,j}(t_m)$: For $m = \mathcal{M} - 1$ to 1,

- Determine the early–exercise points, $y^*_{m,n}$, for $n = 1, \cdots, n_q + 2$, with $\hat{c}(y^*_{m,n}, \delta_n, t_m) = g(y^*_{m,n}, t_m)$, by Newton's method. Continuation value and payoff function are given by (6.42) and (6.39) , respectively.
- Compute the Fourier coefficients $\hat{V}_{n,j}(t_m)$.
  - For $k = 0, \cdots, N_1 - 1$, compute each column of matrix $\hat{\mathbf{C}}^k_n := \{\hat{C}^k_{n,j}\}_{j=0}^{N_2}$ by (6.50) with the help of Fast Fourier Transform.
  - Compute $\hat{C}_{n,j}(t_m)$, $\forall n, j$, from (6.53).
  - Compute $G_{n,j}(t_m)$, $\forall n, j$, from (6.44).
  - Calculate the Fourier coefficients $\hat{V}_{n,j}(t_m)$ by inserting $\hat{C}_{n,j}(t_m)$ and $G_{n,j}(t_m)$ into (6.43).

Final step:

- Compute the early–exercise Asian option value, $\hat{v}(x_0, t_0)$, by inserting $\hat{V}_{n,j}(t_1)$ in (6.38).

6.4.4 Integration range of Y_m

Here we explain how to determine the upper bound b_2, so that the truncation error in Y_m, with integration range $[\exp(x_m), b_2]$ can be controlled. First of all, we derive the integration range for $\log(Y_m)$ and after that the range for Y_m. From [34, 35], we know that a suitable integration range for $\log(Y_m)$ can be determined by means of the cumulants, as follows

$$[\ell, \upsilon] \approx \Big[(\xi_1(\log(Y_m)) - L\sqrt{\xi_2(\log(Y_m)) + \sqrt{\xi_4(\log(Y_m))}}\,, \\ \xi_1(\log(Y_m)) + L\sqrt{\xi_2(\log(Y_m)) + \sqrt{\xi_4(\log(Y_m))}}\Big], \quad (6.54)$$

and the integration range of Y_m at t_m can then be set to $[e^{x_m}, e^{\upsilon}]$. By $\xi_n(X)$, we denote the n^{th} cumulant of X, computed via

$$\xi_n(X) := \frac{1}{i^n}\frac{\partial^n(t\Phi(\mathbf{u}))}{\partial \mathbf{u}^n}|\mathbf{u} = 0,$$

where $t\Phi(\mathbf{u})$ is the exponent of the characteristic function, $\phi(\mathbf{u}; t)$, i.e.

$$\phi(\mathbf{u}; t) = e^{t\Phi(\mathbf{u})}.$$

For arithmetic Asian options, it is however expensive to calculate these cumulants, and therefore we propose *another integration range* for the arithmetic case, which is very similar to that in (6.54). For a Lévy process, the cumulants of any increment, $\log(S_l/S_k)$, $\forall l > k$, are linearly increasing functions of $(l-k)\Delta t$, so that, for all Y_m, $m = 1, \cdots, \mathcal{M}$, we have

$$\xi_1(\log(m\frac{S_1}{S_0})) \leq \xi_1(\log(Y_m)) \leq \xi_1(\log(m\frac{S_m}{S_0})),$$
$$0 \leq \xi_2(\log(Y_m)) \leq \xi_2(\log(m\frac{S_m}{S_0})), \quad 0 \leq \xi_4(\log(Y_m)) \leq \xi_4(\log(m\frac{S_m}{S_0})).$$

and we will use the integration boundaries

$$\ell := \xi_1(\log(m\frac{S_1}{S_0})) - L\sqrt{\xi_2(\log(m\frac{S_m}{S_0})) + \sqrt{\xi_4(\log(m\frac{S_m}{S_0}))},}$$
$$\upsilon := \xi_1(\log(m\frac{S_m}{S_0})) + L\sqrt{\xi_2(\log(m\frac{S_m}{S_0})) + \sqrt{\xi_4(\log(m\frac{S_m}{S_0}))}.} \quad (6.55)$$

Interval $[\ell, \upsilon]$ from (6.55) will be the integration range for $\log(Y_m)$.

Note that the cumulants of $\log(m\frac{S_1}{S_0})$ and $\log(m\frac{S_m}{S_0})$ in (6.55) are known in closed form for Lévy processes, as for $n = 1$, we have $\xi_1(\log(m\frac{S_1}{S_0})) = log(m)+\xi_1(R)$, $\xi_1(\log(m\frac{S_m}{S_0})) = log(m)+m\xi_1(R)$, and for $n \geq 2$, $\xi_n(\log(m\frac{S_1}{S_0})) = \xi_n(R)$, $\xi_n(\log(m\frac{S_m}{S_0})) = m\xi_n(R)$. Here, parameter R denotes the logarithm of the increment between any two consecutive time steps of a Lévy process.

From [34, 35] we know that with $L \approx 10$, the integration range ensures highly accurate option prices for most Lévy processes. With a wider integration range $[\ell, \upsilon]$, the error will be smaller, but an increasing number of Fourier cosine terms may need to be used (which makes it more costly). Adaptation of parameter L for very short term, or very long term options is easily possible.

Integration range of Y_m at t_m is then taken as $[e^{x_m}, e^{\upsilon}]$.

6.5 Error analysis

Here, we give a detailed error analysis of the 2D ASCOS method for early–exercise arithmetic Asian options from Section 6.4. We identify three different types of errors, for which we first introduce some notation.

The *truncation error*, ϵ_T, for any random variable, Z, with integration range $[a, b]$, is defined as

$$\epsilon_T(Z; [a, b]) := \int_{\mathbf{R}\setminus[a,b]} f_Z(z)dz, \quad (6.56)$$

and it decreases as the integration range $[a, b]$ increases. In other words, for a sufficiently large integration range, this error won't dominate the total error in the arithmetic Asian option price.

For Y_m we truncate one side of the integration range, and the truncation error reads

$$\epsilon_T(Y_m; b_2) := \int_{b_2}^{+\infty} f_{Y_m}(y)dy, \tag{6.57}$$

The *error due to the number of terms used in the Fourier cosine expansion* is denoted by ϵ_F. We know, from [34], that for $f_Z(z) \in \mathbf{C}^\infty[a, b]$, this error can be bounded by

$$|\epsilon_F(Z; N)| \leq P^*(N) \exp(-(N-1)\nu_F), \tag{6.58}$$

with $\nu_F > 0$ a constant and a term $P^*(N)$, which varies less than exponentially with respect to N. Note that, although the upper bound of ϵ_F is not a function of the underlying state variable Z, the state variable still appears as an input argument, because the smoothness of the density function influences the convergence behavior.

When the probability density function has a discontinuous derivative, the error can be bounded by

$$|\epsilon_F(Z; N)| \leq \frac{\bar{P}^*(N)}{(N-1)^{\beta-1}},$$

where $\bar{P}^*(N)$ is a constant and $\beta \geq 1$.

Error ϵ_F thus decays either exponentially with respect to N, if the density function $f(z) \in \mathbf{C}^\infty[a, b]$, or otherwise algebraically.

We denote the error from the Clenshaw–Curtis quadrature (6.36) by ϵ_q. From [73] we know that for integrands belonging to $C^\infty[a, b]$, which is the case here, error ϵ_q decays exponentially, i.e.,

$$|\epsilon_q(n_q)| \leq P(n_q) \exp(-(n_q - 1)\nu_q), \tag{6.59}$$

with $\nu_q > 0$ a constant and a term $P(n_q)$, which varies less than exponentially with respect to n_q.

Note that the value of the averaged underlying price, y_m, $\forall t_m$, does not influence the smoothness of the density function of the underlying process. In fact, it can be recursively proved that if $f(x_j), j = 1, \cdots, \mathcal{M}$ is smooth then $f(y_j), j = 1, \cdots, \mathcal{M}$, with $y_j = \sum_{i=1}^{j} \exp(x_i)$ is also smooth, so that it does not influence the convergence speed negatively.

We further denote by $\epsilon(\hat{c}(y_m, x_m, t_m))$, $\epsilon(V_{n,j}(t_m))$ and $\epsilon^*_{m,n}$, the errors in the continuation value, in the Fourier coefficients and in the early–exercise points, $y^*_{m,n}$, at time step t_m, respectively.

Our error analysis is based on backward recursion, i.e. first of all we analyze the error in the continuation value, $\hat{c}(y_{\mathcal{M}-1}, x_{\mathcal{M}-1}, t_{\mathcal{M}-1})$, in Subsection 6.5.1, after which the error propagation throughout the time steps $t_m, m = \mathcal{M} - 2, \cdots, 1$ is discussed in Subsection 6.5.2.

6.5.1 Initial error

In this subsection, the error from (6.29) to (6.35) is discussed. At $t_{\mathcal{M}-1}$, Eqns. (6.29) and (6.35) can be rewritten, respectively, as

$$c(y_{\mathcal{M}-1}, x_{\mathcal{M}-1}, t_{\mathcal{M}-1}) = \tag{6.60}$$
$$e^{-r\Delta t}\int_{\mathbb{R}}\int_{\exp(x_{\mathcal{M}})}^{+\infty} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) f(y_{\mathcal{M}}|x_{\mathcal{M}}, y_{\mathcal{M}-1}) f(x_{\mathcal{M}}|x_{\mathcal{M}-1}) dy_{\mathcal{M}} dx_{\mathcal{M}},$$

and

$$\begin{aligned}\hat{c}(y_{\mathcal{M}-1}, x_{\mathcal{M}-1}, t_{\mathcal{M}-1}) &= e^{-r\Delta t}\int_{a_1}^{b_1}\int_{\exp(x_{\mathcal{M}})}^{b_2} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) \hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}}, y_{\mathcal{M}-1}) \\ &\cdot \hat{f}(x_{\mathcal{M}}|x_{\mathcal{M}-1}) dy_{\mathcal{M}} dx_{\mathcal{M}},\end{aligned} \tag{6.61}$$

where $\hat{f}(x_{\mathcal{M}}|x_{\mathcal{M}-1})$ and $\hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}}, y_{\mathcal{M}-1})$ are defined in (6.33) and (6.34), respectively.

Then, the error, which we denote by $\tilde{\epsilon}$, consists of two parts, that is, $\tilde{\epsilon} := \epsilon_I + \epsilon_{II}$, with

$$\begin{aligned}\epsilon_I &:= e^{-r\Delta t}\int_{\mathbb{R}}\int_{\exp(x_{\mathcal{M}})}^{+\infty} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) f(y_{\mathcal{M}}|x_{\mathcal{M}}, y_{\mathcal{M}-1}) dy_{\mathcal{M}} f(x_{\mathcal{M}}|x_{\mathcal{M}-1}) dx_{\mathcal{M}} \\ &- e^{-r\Delta t}\int_{\mathbb{R}}\int_{\exp(x_{\mathcal{M}})}^{b_2} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) \hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}}, y_{\mathcal{M}-1}) dy_{\mathcal{M}} f(x_{\mathcal{M}}|x_{\mathcal{M}-1}) dx_{\mathcal{M}},\end{aligned} \tag{6.62}$$

and

$$\begin{aligned}\epsilon_{II} &:= e^{-r\Delta t}\int_{\exp(x_{\mathcal{M}})}^{b_2}\int_{\mathbb{R}} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) \hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}}, y_{\mathcal{M}-1}) f(x_{\mathcal{M}}|x_{\mathcal{M}-1}) dx_{\mathcal{M}} dy_{\mathcal{M}} \\ &- e^{-r\Delta t}\int_{\exp(x_{\mathcal{M}})}^{b_2}\int_{a_1}^{b_1} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) \hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}}, y_{\mathcal{M}-1}) \hat{f}(x_{\mathcal{M}}|x_{\mathcal{M}-1}) dx_{\mathcal{M}} dy_{\mathcal{M}}.\end{aligned} \tag{6.63}$$

We use the notation ϵ_{cos} to denote the error of one step of the COS method [34],

$$\begin{aligned}\epsilon_{cos}(X_{\mathcal{M}}) &:= \int_{\mathbb{R}} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) f(x_{\mathcal{M}}|x_{\mathcal{M}-1}) dx_{\mathcal{M}} \\ &- \int_{a_1}^{b_1} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) \hat{f}(x_{\mathcal{M}}|x_{\mathcal{M}-1}) dx_{\mathcal{M}},\end{aligned}$$

and

$$\begin{aligned}\epsilon_{cos}(Y_{\mathcal{M}}) &:= \int_{\exp(x_{\mathcal{M}})}^{+\infty} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) f(y_{\mathcal{M}}|x_{\mathcal{M}}, y_{\mathcal{M}-1}) dy_{\mathcal{M}} \\ &- \int_{\exp(x_{\mathcal{M}})}^{b_2} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) \hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}}, y_{\mathcal{M}-1}) dy_{\mathcal{M}}.\end{aligned}$$

The first part of the error (6.62) then reads

$$\epsilon_I = e^{-r\Delta t}\epsilon_{cos}(Y_{\mathcal{M}}) \int_{\mathbb{R}} f(x_{\mathcal{M}}|x_{\mathcal{M}-1})dx_{\mathcal{M}} = e^{-r\Delta t}\epsilon_{cos}(Y_{\mathcal{M}}). \tag{6.64}$$

To compute the second part of the error, in (6.63), first of all, from (6.34) we have that $\forall y_{\mathcal{M}} \in [\exp(x_{\mathcal{M}}), b_2]$,

$$|\hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}}, y_{\mathcal{M}-1})| \leq \frac{2}{b_2 - \exp(x_{\mathcal{M}})} N_2 \leq \frac{2}{b_2 - \exp(a_1)} N_2.$$

Then, ϵ_{II}, in (6.63), can be written as

$$\begin{aligned} |\epsilon_{II}| &\leq e^{-r\Delta t} \frac{2N_2}{b_2 - \exp(a_1)} \Big| \int_{\exp(x_{\mathcal{M}})}^{b_2} \Big(\int_{\mathbb{R}} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) f(x_{\mathcal{M}}|x_{\mathcal{M}-1}) dx_{\mathcal{M}} \\ &- \int_{a_1}^{b_1} v(y_{\mathcal{M}}, x_{\mathcal{M}}, t_{\mathcal{M}}) \hat{f}(x_{\mathcal{M}}|x_{\mathcal{M}-1}) dx_{\mathcal{M}} \Big) dy_{\mathcal{M}} \Big| \\ &\leq e^{-r\Delta t} 2N_2 |\epsilon_{cos}(X_{\mathcal{M}})| \end{aligned} \tag{6.65}$$

We will now use the common notation $\epsilon(x) = O(\varsigma)$, $\forall x \in \mathbb{R}$, if $Q > 0$ exists, so that $|\epsilon(x)| \leq Q|\varsigma|$.

We then have $\epsilon_{II} = O(N_2 \epsilon_{cos}(X_{\mathcal{M}}))$.

From [34] we know that $\forall Z, a, b, N$, $\epsilon_{cos}(Z) = O(\epsilon_T(Z; [a, b])) + \epsilon_F(Z; N)$, and a similar analysis can be performed for a one–side truncated variable $Y_{\mathcal{M}}$, then, from (6.64) and (6.65), we obtain

$$\begin{aligned} \tilde{\epsilon} = \epsilon_I + \epsilon_{II} &= O(N_2(\epsilon_T(X_{\mathcal{M}}; [a_1, b_1]) + \epsilon_F(X_{\mathcal{M}}; N_1)) + \\ &\quad \epsilon_T(Y_{\mathcal{M}}; b_2) + \epsilon_F(Y_{\mathcal{M}}; N_2)), \end{aligned} \tag{6.66}$$

which is the error made up to Eq. (6.35).

At $t_{\mathcal{M}-1}$, the Fourier coefficients of the option value, $V_{n,j}(t_{\mathcal{M}})$ are known analytically. Therefore, the error from Eq. (6.35) to Eq. (6.38) is only due to approximation (6.36), where the Clenshaw–Curtis quadrature was used. For each j, k the error in the approximated integration is $O(\epsilon_q(n_q))$, with ϵ_q defined in (6.59). Thus, the error in (6.38) is found to be $\tilde{\epsilon} + O(N_1 N_2 \epsilon_q(n_q))$, with $\tilde{\epsilon}$ in (6.66). Summarizing, the error in the continuation value, $\hat{c}(y_{\mathcal{M}-1}, x_{\mathcal{M}-1}, t_{\mathcal{M}-1})$, is found to be

$$\begin{aligned} \epsilon(\hat{c}(\quad y_{\mathcal{M}-1} \quad , x_{\mathcal{M}-1}, t_{\mathcal{M}-1})) &= O(N_2(\epsilon_T(X_{\mathcal{M}}; [a_1, b_1]) + \epsilon_F(X_{\mathcal{M}}; N_1)) \\ + \quad \epsilon_T(Y_{\mathcal{M}}; b_2) + \epsilon_F(Y_{\mathcal{M}}; N_2) &+ N_1 N_2 \epsilon_q(n_q)). \end{aligned} \tag{6.67}$$

With integration ranges $[a_1, b_1]$ and b_2 carefully chosen, truncation errors $\epsilon_T(X_{\mathcal{M}}; [a_1, b_1])$ and $\epsilon_T(Y_{\mathcal{M}}; b_2)$ will not be the dominant parts of error (6.67). For a smooth density function of $X_{\mathcal{M}}$ ($f(X_{\mathcal{M}}) \in C^\infty$), it can

be proved that the density function of $Y_{\mathcal{M}}$ is also smooth, and that the error in the continuation value decays to zero exponentially, with respect to N_1, N_2, n_q. In detail, inserting (6.58) and (6.59) into (6.67) gives us

$$\begin{aligned} |\epsilon(\hat{c}(\quad y_{\mathcal{M}-1} \quad , x_{\mathcal{M}-1}, t_{\mathcal{M}-1}))| \leq P^*(N_1, N_2, n_q)(\exp(-(N_1-1)\nu_1) \\ + \quad \exp(-(N_2-1)\nu_2) + \exp(-(n_q-1)\nu_q)), \end{aligned} \tag{6.68}$$

where $P^*(N_1, N_2, n_q)$ is a term which varies less than exponentially with respect to N_1, N_2, n_q.

If the density of $X_{\mathcal{M}}$ is not smooth, then the error converges exponentially to zero with respect to n_q and algebraically with respect to N_1 and N_2.

6.5.2 Error propagation

Regarding the propagation of the error through time, we state the following lemma:

Lemma 6.5.1 (Error propagation)**.** *For $m = \mathcal{M}-2, \cdots, 0$, assuming that at time step t_{m+1}, $\forall y_{m+1}, x_{m+1}$,*

$$\begin{aligned} |\epsilon(\hat{c}(y_{m+1}, x_{m+1}, t_{m+1}))| \quad &\leq \quad P(N_1, N_2, n_q)(\exp(-(N_1-1)\nu_1) \\ &+ \quad \exp(-(N_2-1)\nu_2) + \exp(-(n_q-1)\nu_q)), \end{aligned} \tag{6.69}$$

where $P(N_1, N_2, n_q)$ is a term which varies less than exponentially with respect to N_1, N_2, n_q, then, at time step t_m, we can show that, $\forall y_m, x_m$,

$$\begin{aligned} |\epsilon(\hat{c}(y_m, x_m, t_m))| \quad &\leq \quad \bar{P}(N_1, N_2, n_q)(\exp(-(N_1-1)\nu_1) \\ &+ \quad \exp(-(N_2-1)\nu_2) + \exp(-(n_q-1)\nu_q)), \end{aligned} \tag{6.70}$$

where $\bar{P}(N_1, N_2, n_q)$ is a term which varies less than exponentially with respect to N_1, N_2, n_q.

Proof. This is a proof based on mathematical induction.

First, we compute the error in the Fourier coefficients, $\hat{V}_{n,j}(t_{m+1})$, after which we analyze the error in $\hat{c}(y_m, x_m, t_m)$.

Error $\epsilon(\hat{V}_{n,j}(t_{m+1}))$ consists of two parts, the error in the Fourier cosine coefficients of the continuation value, and the error due to an incorrect value of the early–exercise point. Without loss of generality, we consider a call option with a positive-valued error in the early–exercise points. The analysis of the error propagation for other cases (negatively-valued error, put option)

goes similarly. For a call option, with $\epsilon^*_{m+1,n} > 0$, we have

$$\begin{aligned}
& \epsilon(\hat{V}_{n,j}(t_{m+1})) = (C_{n,j}(\exp(\delta_n), y^*_{m+1,n}, t_{m+1}) - \hat{C}_{n,j}(\exp(\delta_n), y^*_{m+1,n}, t_{m+1})) \\
+ \quad & (G_{n,j}(y^*_{m+1,n}, y^*_{m+1,n} + \epsilon^*_{m+1,n}, t_{m+1}) - \hat{C}_{n,j}(y^*_{m+1,n}, y^*_{m+1,n} + \epsilon^*_{m+1,n}, t_{m+1})) \\
= \quad & \int_{\exp(\delta_n)}^{y^*_{m+1,n}} \epsilon(\hat{c}(y_{m+1}, \delta_n, t_{m+1})) \cos\left(j\pi \frac{y_{m+1} - \exp(\delta_n)}{b_2 - \exp(\delta_n)}\right) dy_{m+1} \\
+ \quad & \int_{y^*_{m+1,n}}^{y^*_{m+1,n}+\epsilon^*_{m+1,n}} (g(y_{m+1}, t_{m+1}) - \hat{c}(y_{m+1}, \delta_n, t_{m+1})) \cos\left(j\pi \frac{y_{m+1} - \exp(\delta_n)}{b_2 - \exp(\delta_n)}\right) dy_{m+1},
\end{aligned} \tag{6.71}$$

with $g(y_{m+1}, t_{m+1})$ defined in (6.39).

The error in continuation value $\hat{c}(y_m, x_m, t_m)$ is composed of two parts, $\epsilon(\hat{c}(y_m, x_m, t_m)) := e_I + e_{II}$, where error e_I is the part in which $\epsilon(\hat{V}_{n,j}(t_{m+1}))$ has not yet been considered. It is derived similarly as the error at $t_{\mathcal{M}-1}$ (in Subsection 6.5.1). We find

$$\begin{aligned}
e_I \quad := \quad & O(N_2(\epsilon_T(X_{m+1}; [a_1, b_1]) + \epsilon_F(X_{m+1}; N_1)) \\
+ \quad & \epsilon_T(Y_{m+1}; b_2) + \epsilon_F(Y_{m+1}; N_2) + N_1 N_2 \epsilon_q(n_q)).
\end{aligned} \tag{6.72}$$

Error e_{II} is the additional error with $\epsilon(\hat{V}_{n,j}(t_{m+1}))$ taken into consideration,

$$\begin{aligned}
e_{II} \quad := \quad & e^{-r\Delta t} {\sum_{k=0}^{N_1-1}}' {\sum_{j=0}^{N_2-1}}' Re\left(\phi(\frac{k\pi}{b_1 - a_1}; \Delta t) \exp\left(ik\pi \frac{x_m - a_1}{b_1 - a_1}\right)\right) \cdot \\
& Re\left(\sum_{n=1}^{n_q+2} \frac{2}{b_2 - \exp(\delta_n)} w_n \exp\left(i \frac{j\pi}{b_2 - \exp(\delta_n)} y_m\right) \cos(k\pi \frac{\delta_n - a_1}{b_1 - a_1}) \right. \\
& \left. \epsilon(\hat{V}_{n,j}(t_{m+1}))\right).
\end{aligned}$$

To analyze these errors, we define a European option, v_α, from t_m to t_{m+1}, with payoff function

$$v_\alpha(y_{m+1}, x_{m+1}, t_{m+1}, L_1, L_2) := \begin{cases} 1, & \text{if} \quad y_{m+1} \in [L_1, L_2], \\ 0, & \text{otherwise}, \end{cases}$$

so that the option value at t_m, $\forall L_1, L_2 \in [\exp(x_{m+1}), +\infty]$, can be writ-

ten as

$$\begin{aligned} v_\alpha(y_m, x_m, t_m, L_1, L_2) &= e^{-r\Delta t} \int_{\mathbb{R}} \int_{\exp(x_{m+1})}^{+\infty} v(y_{m+1}, x_{m+1}, t_{m+1}, L_1, L_2) \\ &\cdot \quad f(y_{m+1}, x_{m+1}, t_{m+1}|y_m, x_m) dy_{m+1} dx_{m+1} \\ &= e^{-r\Delta t} \int_{\mathbb{R}} \int_{L_1}^{L_2} f(y_{m+1}, x_{m+1}, t_{m+1}|y_m, x_m) dy_{m+1} dx_{m+1} \\ &\leq e^{-r\Delta t} \int_{\mathbb{R}} \int_{\mathbb{R}} f(y_{m+1}, x_{m+1}, t_{m+1}|y_m, x_m) dy_{m+1} dx_{m+1} \\ &= e^{-r\Delta t} \end{aligned}$$

and its approximation, by using (6.38), reads

$$\begin{aligned} & \hat{v}_\alpha(y_m, x_m, t_m, L_1, L_2) \\ = \; & e^{-r\Delta t} \sum_{k=0}^{N_1-1\prime} \sum_{j=0}^{N_2-1\prime} Re\left(\phi(\frac{k\pi}{b_1 - a_1}; \Delta t) \exp\left(ik\pi \frac{x_m - a_1}{b_1 - a_1}\right) \right) \cdot \\ & Re\left(\sum_{n=1}^{n_q+2} \frac{2}{b_2 - \exp(\delta_n)} w_n \exp\left(i \frac{j\pi}{b_2 - \exp(\delta_n)} y_m\right) \cos(k\pi \frac{\delta_n - a_1}{b_1 - a_1}) \right. \\ & \left. \int_{L_1}^{L_2} \cos(j\pi \frac{y_{m+1} - \exp(\delta_n)}{b_2 - \exp(\delta_n)}) \right) ., \end{aligned}$$

from which it follows that, $\forall L_1 \leq L_2 \leq L_3$,

$$\hat{v}_\alpha(y_m, x_m, t_m, L_1, L_2) + \hat{v}_\alpha(y_m, x_m, t_m, L_2, L_3) = \hat{v}_\alpha(y_m, x_m, t_m, L_1, L_3). \tag{6.73}$$

The value of $\hat{v}_\alpha$ can bounded, as

$$\begin{aligned} \hat{v}_\alpha(y_m, x_m, t_m, L_1, L_2) &\leq v_\alpha(y_m, x_m, t_m, L_1, L_2) + |\epsilon(v_\alpha(y_m, x_m, t_m, L_1, L_2))| \\ &= e^{-r\Delta t} + O(|e_I|), \end{aligned} \tag{6.74}$$

where the last step is because the error from approximation (6.38) at t_m is of the same order as e_I.

Inserting (6.71) into (6.73), then using (6.69) and (6.73), gives us

$$\begin{aligned} |\epsilon(\hat{c}(y_m, x_m, t_m))| &\leq |e_I| + P(N_1, N_2, n_q)(\exp(-(N_1 - 1)\nu_1) + \exp(-(N_2 - 1)\nu_2) \\ &+ \exp(-(n_q - 1)\nu_q))\hat{v}_\alpha(y_m, x_m, t_m, \exp(\delta_n), y^*_{m+1,n}) \\ &+ \max_n |g(\zeta_n, t_{m+1}) - \hat{c}(\zeta_n, \delta_n, t_{m+1})| \\ &\cdot \quad \hat{v}_\alpha(y_m, x_m, t_m, y^*_{m+1,n}, y^*_{m+1,n} + \epsilon^*_{m+1,n}), \end{aligned} \tag{6.75}$$

based on Lagrange's mean value theorem, with $\zeta_n \in (y^*_{m+1,n}, y^*_{m+1,n} + \epsilon^*_{m+1,n})$.

For a call option, with $\zeta_n \in (y^*_{m+1,n}, y^*_{m+1,n} + \epsilon^*_{m+1,n})$, we then have $\forall n$,

$$\begin{aligned} |g(\zeta_n, t_{m+1}) - \hat{c}(\zeta_n, \delta_n, t_{m+1})| &= \hat{c}(\zeta_n, \delta_n, t_{m+1}) - g(\zeta_n, t_{m+1}) \\ &\leq \hat{c}(y^*_{m+1,n}, \delta_n, t_{m+1}) - g(y^*_{m+1,n}, t_{m+1}) \\ &= \hat{c}(y^*_{m+1,n}, \delta_n, t_{m+1}) - c(y^*_{m+1,n}, \delta_n, t_{m+1}) \\ &= \epsilon(\hat{c}(y^*_{m+1,n}, \delta_n, t_{m+1})). \end{aligned} \tag{6.76}$$

Inserting (6.76) in (6.75), and using (6.69) and (6.73), gives us

$$\begin{aligned} |\epsilon(\hat{c}(y_m, x_m, t_m))| \leq\ & |e_I| + P(N_1, N_2, n_q)(\exp(-(N_1 - 1)\nu_1) + \exp(-(N_2 - 1)\nu_2) \\ +\ & \exp(-(n_q - 1)\nu_q))\hat{v}_\alpha(y_m, x_m, t_m, \exp(\delta_n), y^*_{m+1,n} + \epsilon^*_{m+1,n}). \end{aligned} \tag{6.77}$$

Finally, by using (6.74) and (6.72) in (6.77), and then inserting (6.58) and (6.59), we reach the conclusion that if $[a_1, b_1]$, b_2 are carefully chosen, then the truncation errors $\epsilon_T(X_{m+1}; [a_1, b_1])$ and $\epsilon_T(Y_{m+1}; b_2)$ will not be the dominant parts of error (6.72), and we obtain

$$\begin{aligned} |\epsilon(\hat{c}(y_m, x_m, t_m))| \leq\ & \bar{P}(N_1, N_2, n_q)(\exp(-(N_1 - 1)\nu_1) \\ +\ & \exp(-(N_2 - 1)\nu_2) + \exp(-(n_q - 1)\nu_q)), \end{aligned}$$

where $\bar{P}(N_1, N_2, n_q)$ is a term which varies less than exponentially with respect to N_1, N_2, n_q. This concludes the proof. □

In the case of put options or negative-valued errors in the early–exercise points, a similar error expression as in (6.77) can be derived, by a very similar analysis.

6.6 Numerical results

In this section we perform experiments with two different Lévy processes, the Black-Scholes (BS) and the Normal Inverse Gaussian (NIG) processes. We will present numerical results for the two methods presented. Reference values are derived by our 2D version of the ASCOS method, with $N_1 = N_2 = (n_q/2) + 1 = 4096$. When increasing the values of $\mathcal{M}, N_1, N_2, n_q$ in the numerical experiments, the 2D ASCOS method gives the same American Asian option values for the BS model as the values in [28] (in the accuracy given in the reference, which is 10^{-4}).

The same model parameters, as used in [37] for pricing European-style Asian options, are also used here:

- *BS:* $r = 0.0367, \sigma = 0.178$;
- *NIG:* $r = 0.0367, \sigma = 0.178, \alpha = 6.188, \beta = -3.894, \delta = 0.1622$.

Two types of processors, a CPU (Central Processing Unit), and a GPU (Graphics Processing Unit) with double precision are used and compared to obtain the numerical ASCOS results. On the CPU, an Intel(R) Core(TM)2 Duo CPU E6550 (@ 2.33GHz Cache size 4MB), the algorithm is implemented in MATLAB 7.7.0. On the GPU, a Tesla C2070 GPU with 6GB memory, we coded in Compute Unified Device Architecture (CUDA) [76]. Computing time is recorded in seconds.

In this section, the notation 'first method' and '2D method' refer to the pricing methods proposed in Section 6.3 and Section 6.4, respectively.

Remark 6.6.1 (Data transfer)**.** *Data transfer between the GPU and the CPU is the bottleneck for most GPU implementations. However, in our GPU implementation, we code in such a way that, no matter the size of the problem, only one number needs to be transferred between the CPU and the GPU, which is the option price and we transfer it back to the CPU at the end of the computations. As the size of the problem increases, there will be no extra burden of data transfer.*

6.6.1 GPU implementation and acceleration

A GPU is an SIMT (Single Instruction, Multiple Threads) machine. In other words, the same command can be executed simultaneously for each data element on each thread on the GPU. Therefore, GPU processing is advantageous for problems that can be expressed in the form of data–parallel computations.

In both early-exercise ASCOS algorithms we proceed from time step to time step sequentially, however, there are certain parts of the algorithms for which parallelization is possible. For instance, in the first method, the integrals (6.15), for all k, l, can be computed independently of each other, and in the 2D method, the early–exercise points, $y^*_{m,n}$, can also be calculated independently for each $na - value$. The Fourier coefficients, that represent a vector in the first method and a matrix in the 2D method, are computed simultaneously on the GPU at each time step.

In both methods we need to perform matrix–vector multiplications, that result in $O(N^2)$ computations for the first method, and result in $O(n_q N_2)$ computations with the 2D method. In these operations, the summation in each row must be done sequentially. Two techniques can however be used to accelerate the GPU computation of a summation. First of all, we can sum up each row in a pairwise fashion, that is, we split the vector into two parts and add up the two sub–vectors simultaneously on the GPU. This process is repeated until we reach vector sizes of one element, being the sum of all elements of the original vector. A second way to accelerate the process is to use the shared–memory within each block, which significantly reduces the data–communication time on the GPU.

As an example, Table 6.1 presents the error and the GPU speedup, compared to the CPU implementation, when pricing an early–exercise arithmetic Asian option with $\mathcal{M} = 2$ using the 2D method. A speedup factor between 30 and 300 is achieved on the GPU. When the problem size increases, an even higher GPU speedup is expected, since then option pricing will be computationally more intensive and the advantages of parallelization are more profound. When the number of early–exercise dates increases, the GPU as well as the CPU times will increase linearly.

All further numerical experiments will be performed on the GPU.

$N_1 = N_2 = (n_q/2) + 1$	128	256	512
abs.error	1.2134e-01	4.6379e-06	1.3043e-08
GPU speedup	30.4	139.6	341.0

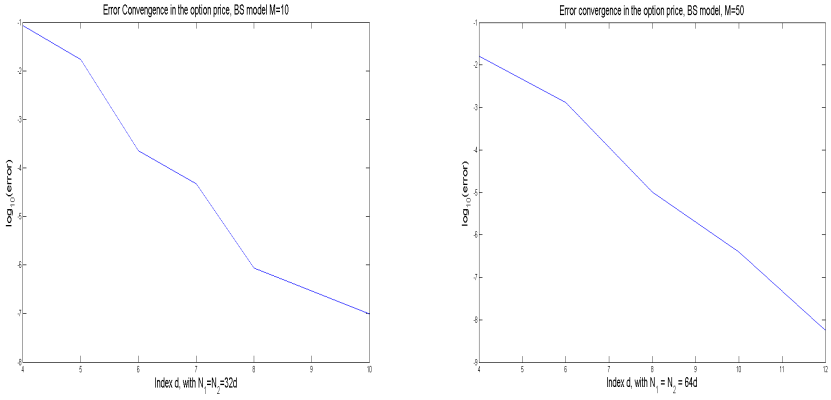
Table 6.1: GPU speedup for Bermudan Asian options, BS model, $\mathcal{M} = 2, S_0 = 100, K = 100$.

6.6.2 Arithmetic Asian options on the GPU

Error convergence of early–exercise arithmetic Asian options under the NIG model, with 10 and 50 early–exercise dates, using the 2D ASCOS method, are presented in Figure 6.1. The horizontal axis presents index d, where in Figure 6.1(a), $N_1 = N_2 = 32d$, $(n_q/2)+1 = 256$, and in Figure 6.1(b), we use $N_1 = N_2 = 64d$, $(n_q/2) + 1 = 512$. The vertical axis shows the logarithm of the absolute error. For $\mathcal{M} = 10$ as well as $\mathcal{M} = 50$ an exponential convergence is observed: When N_1 and N_2 increase linearly, the logarithm of the error in the option price decreases accordingly.

When comparing the two plots in Figure 6.1, we see that with an increasing number of early–exercise dates we require larger values for N_1, N_2 and n_q to reach the same level of accuracy. With smaller time steps, Δt, the conditional density function between consecutive time steps tends to be peaked, and an accurate approximation by means of cosine expansions then requires an increasing number of terms. The need for a larger value of n_q comes from the fact that the error of the Clenshaw–Curtis quadrature is observed in each term of (6.38) and there are $N_1 N_2$-terms in total. Therefore, larger values for N_1 and N_2 give rise to a larger n_q-value to ensure the accuracy.

Tables 6.2 and 6.3 present the convergence behavior and computing time for the NIG model with $\mathcal{M} = 10, 50$, respectively, with the performance of both pricing methods presented. From Table 6.2 we see that when $\mathcal{M} = 10$, due to the error of the first method with a small number of exercise dates, the option price does not converge to the reference value with the first method. On the other hand, the first method is significantly faster than the 2D



(a) $\mathcal{M} = 10$ (b) $\mathcal{M} = 50$

Figure 6.1: 2D ASCOS error convergence for early–exercise arithmetic Asian options with different numbers of early–exercise dates, NIG model, $S_0 = 100, K = 110$.

method. The first method exhibits a reduced computation complexity, by a factor $O(log_2 N_2)$, and the GPU speedup is higher when implementing the first method, as with the 2D method, there is an N_1 by $\mathcal{M}$ loop in the CUDA code.

First method			
$N_1 = N_2 = (n_q/2) + 1$	128	192	256
abs.error	3.3236e-01	3.1511e-01	3.1641e-01
GPU time	0.28	0.53	0.87
2D method			
$N_1 = N_2 = (n_q/2) + 1$	256	384	512
abs.error	1.4213e-04	3.1444e-07	2.2129e-09
GPU time	4.76	9.05	31.25

Table 6.2: Convergence and computation time of early-exercise arithmetic Asian put options, under the NIG model, with $\mathcal{M} = 10, S_0 = 100$ (time in seconds).

As $\mathcal{M}$ increases, as shown in Table 6.3, the error in the first method gets much smaller, and the option prices gradually converges to the reference value. This is consistent with our analysis.

First method	$N_1, N_2 = 256$	$N_1, N_2 = 512$	$N_1, N_2 = 768$
	$(n_q/2) + 1 = 256$	$(n_q/2) + 1 = 512$	
abs.error	1.7165e-02	1.4364e-03	4.4992e-05
GPU time	1.52	9.40	9.64
2D method	$N_1, N_2 = 256$	$N_1, N_2 = 512$	$N_1, N_2 = 768$
	$(n_q/2) + 1 = 256$	$(n_q/2) + 1 = 512$	
abs.error	3.8746e-03	1.0809e-04	4.8980e-07
GPU time	20.0	160.7	399.5

Table 6.3: Convergence and computation time of early-exercise arithmetic Asian put options, under the NIG model with $\mathcal{M} = 50, S_0 = 100$ (time in seconds).

6.7 Conclusions

In this chapter, we have developed an efficient pricing method for Asian options with early–exercise features for arithmetic averages, based on a two-dimensional risk–neutral formula. As an alternative, especially for a large number of exercise dates, a 1D pricing method based on the approximation of the conditional characteristic functions, is proposed, which can be used for very frequently exercised Asian options at a reduced amount of computations. Both methods are based on Fourier cosine expansions and Clenshaw–Curtis quadrature, and, depending on the smoothness of the density function, may give rise to exponential error convergence. The convergence behavior of the 2D ASCOS method is supported by a detailed error analysis, as well as by various numerical experiments. The flexibility and robustness of the 2D pricing method for different Lévy models and different numbers of early–exercise dates is shown in the numerical experiments. In particular, the Graphics Processing Unit, which supports parallel computing, turns out to be very efficient for the computation of arithmetic Asian option values. The speedup on the GPU is high as there are many "parallel" computations and not much data transfer.

CHAPTER 7

Conclusions and Outlook

7.1 Conclusions

In this thesis we have presented efficient pricing methods for early–exercise options and exotic options, using Fourier cosine expansions, in combinations with other numerical methods. Our work is based on the COS method, proposed in [34, 35]. First of all, robust pricing is achieved for both European and Bermudan-style call options, with and without dividend rate, by the use of put–call relations to recover the Fourier coefficients at each time step. The improved methods are particularly advantageous when pricing call options with a long maturity and in the case of fat tail distribution in the underlying process.

A pricing algorithm for early–exercise options under the Ornstein-Uhlenbeck (OU) model has been developed which can also be used for seasonality functions. A more efficient pricing algorithm is proposed, based on an approximation of the OU conditional characteristic function, so that the Fourier coefficients can be calculated by the FFT and the computing time can be reduced from seconds to milliseconds. However, this fast algorithm can only be used for certain model parameter ranges. By a detailed error analysis, we derived the conditions that model parameters need to satisfy for the new pricing method to ensure basis point precision. Furthermore, we have developed efficient and robust pricing algorithms for swing options with early–exercise features. The algorithm works efficiently for different types of swing contracts with different flexible contract choices.

The Asian option has been studied in detail and pricing methods for both European- and American-style Asian options, written on different types of averages, are developed, as presented in the last two chapters. For the European-style Asian options, our pricing method is based on the recovery

of the characteristic function of the average asset. Fourier expansions and Clenshaw–Curtis quadrature are used and exponential convergence is illustrated by detailed error analysis and numerical experiments. An important feature of our pricing method is that the computational time does not increase significantly as the number of monitor dates increases, which makes our method especially advantageous when pricing continuously–monitored Asian options.

Our pricing method for Asian options with early–exercise features, in Chapter 6, is based on a two–dimensional integration and the backward recovery of the Fourier coefficients, where FFT has been used. Our method is flexible and robust for Lévy processes and the error in the option price decays exponentially, as shown by both the error analysis and numerical examples.

Another contribution in this thesis is that we have achieved a fine speedup for the task of pricing options on the Graphics Processing Unit (GPU). The GPU tends to be advantageous when intensive computation is performed, such as in option pricing under hybrid stochastic models, option pricing with multiple strike prices and the pricing for early–exercise Asian options.

The option pricing methods presented in this thesis can be applied to underlying processes for which the characteristic function is known or can be approximated.

7.2 Outlook

The pricing algorithms for underlying processes under the OU model for commodity derivatives can be made more realistic by including a stochastic process to model spikes in prices and more realistic types of seasonality functions. It is generally expected that a research focus on energy derivatives, where we may encounter illiquid markets and significant spreads, may bring substantial advances to this area.

The pricing methods for Asian options, as proposed in Chapters 5 and 6, and the different backward pricing procedures respectively for European and early–exercise style products also gave insight into pricing of other –related– exotic options, like lookback and Australian options. Furthermore, the method for pricing of American Asian options, based on two-dimensional integration, can also form the basis for the pricing of early–exercise options under hybrid models, with stochastic volatility and stochastic interest rate, for example.

The option pricing algorithms based on Fourier cosine expansions of the conditional density function can be used for more complex models, such as inflation models where domestic and foreign interest rate are both assumed stochastic with mean reversion feature and stochastic volatility. Hybrid models may also be used for the quantification of Credit Value Adjustment

(CVA) of portfolios of exotic options, which is of importance for the banks. A GPU implementation is expected to be highly beneficial, when dealing with hybrid models. Another extension is to incorporate the GPU implementation and acceleration into the calibration and hedging procedures, in combination with a CPU based optimization kernel.

BIBLIOGRAPHY

[1] A. Almendral and C. W. Oosterlee. On American options under the Variance Gamma process. *Appl. Math. Finance*, 14:131–152, 2007.

[2] J. Andreasen and M. Dahlgren At the flick of a switch. *Energy risk, 71–75*, February, 2006.

[3] O. E. Barndorff-Nielsen, Normal inverse Gaussian distributions and stochastic volatility. *Scand. J. Statist.*, 24:1–13, 1997.

[4] E. Benhamou, Fast Fourier Transform for Discrete Asian Options. *J. Computational Finance.* 6, 49–68, 2002.

[5] A. Bermúdez, M.R. Nogueiras, and C. Vázquez, Numerical solution of variational inequalities for pricing Asian options by high order Lagrange-Galerkin methods. *Applied Num. Math.* 56: 1256-1270, 2006.

[6] T. BJÖRK, Arbitrage theory in continuous time. *Oxford Univ. Press*, 1998.

[7] C. Blanco and D. Soronow , Mean reverting processes – Energy price processes used for derivatives pricing and risk management, *Commodities Now, Energy Pricing*, 2001.

[8] J. P. Boyd, Exponentially convergent Fourier-Chebyshev quadrature schemes on bounded and infinite intervals. *J. Scient. Comput.* 2(2): 99-109, 1987.

[9] J. P. Boyd, *Chebychev and Fourier Spectral Methods*, 2nd ed., Dover, New York, 2001.

[10] N. Branger, O. Reichmann, and M. Wobben, Pricing electricity derivatives, Working Paper, 2009.

[11] M. Broadie and Y. Yamamoto, Application of the fast Gauss transform to option pricing. *Management Sci.*, 49:1071–1088, 2003.

[12] M. Broadie and Y. Yamamoto, A double–exponential fast Gauss transform for pricing discrete path–dependent options. *Operations Research*, 53, 2005.

[13] R. Carmona and N. Touzi Optimal multiple stopping and valuation of swing options, *Mathematical Finance 18, 2, pp. 239-268.* , April, 2008.

[14] P. P. Carr, H. Geman, D. B. Madan, and M. Yor. The fine structure of asset returns: An empirical investigation. *J. Business*, 75:305–332, 2002.

[15] P. R. Carr, D. B. Madan, and E. C. Chang, The Variance Gamma process and option pricing, *European Finance Review*, 2 (1998) 79–105, 1998.

[16] P.P. Carr and D.B.Madan. Option valuation using Fast Fourier Transformation. *J. Comp.Finance*, 2: 61-73,1999.

[17] U. Cartea and M.G. Figueroa, Pricing in electricity markets: A mean reverting jump diffusion model with seasonality, *Applied Mathematical Finance*, 12(4) (2005) 313–335.

[18] A. Carverhill and L. Clewlow , Flexible Convolution, *From Black Scholes to Black Holes*, 165–171, 1992.

[19] A. Cerny and I. Kyriakou, An Improved Convolution Algorithm for Discretely Sampled Asian Options. *Quantitative Finance,* 11(3), 381–389, 2011.

[20] S-L Chung, C-C Chang, and R. C. Stapleton, Richardson extrapolation technique for pricing American-style options. *J. Futures Markets*,27(8): 791-817, 2007.

[21] K. Chourdakis, Option pricing using the fractional FFT. *J. Comp.Finance*, 8(2):1–18, 2004.

[22] C. W. Clenshaw and A. R. Curtis , A method for numerical integration on an automatic computer, *Numer. Mathematik* 2: 197–205. 1960.

[23] R. Cont and P. Tankov, Financial Modelling with Jump Processes. *Chapman and Hall*, Boca Raton, FL, 2004.

[24] C. Cryer, The solution of a quadratic programming problem using systematic overrelaxation. *SIAM J. Control*, 9, 385–392, 1971.

[25] M.Dahlgren, A continuous time model to price commodity-based swing option., *Review of derivatives research,* 8,27–47, 2005.

[26] M. A. H. Dempster and S. S. G. Hong, Spread option valuation and the Fast Fourier Transform. *Techn. Rep. WP* 26\2000*, the Judge Inst. Manag. Studies, Univ. Cambridge*, 2000.

[27] P. den Iseger and E. Oldenkamp, Pricing Guaranteed Return Rate Products and Discretely Sampled Asian Options. *J. Computational Finance*, Vol.9 - No.3, 2006.

[28] Y. D'Halluin, P. A. Forsyth, and G. Labahn, A semi–Lagrangian approach for American Asian options under jump diffusion. *SIAM J. Sci. Comput.* 27, 315–345, 2005.

[29] P. W. Duck, A. D. Andricopoulos, M. Widdicks and D. P. Newton, Universal option valuation using quadrature methods. *J. Fin. Economics*, 67:447–471, 2003.

[30] P. W. Duck, A. D. Andricopoulos, M. Widdicks, and D. P. Newton, Extending quadrature methods to value multi-asset and complex path dependent options. *J. Fin. Economics*, 83:471–500, 2007.

[31] Duffie D., Pan J., Singleton K., Transform analysis and asset pricing for affine jump-diffusions, Econometrica, 68 (2000) 1343–1376.

[32] E. Eberlein and A. Papapantoleon, Equivalence of floating and fixed strike Asian and lookback options, *Stochastic Processes and their Applications*, 115, 31–40, 2005.

[33] A. Eydeland, A fast algorithm for computing integrals in function spaces: financial applications. *Computational Economics*, 7, 1994.

[34] F. Fang and C. W. Oosterlee, A novel option pricing method based on Fourier-cosine series. *SIAM J. Sci. Comput.*, 31, 2008.

[35] F. Fang and C. W. Oosterlee, Pricing early-exercise and discrete barrier options by Fourier-cosine series expansions. *Numerische Mathematik*, 114, 2009.

[36] L. Feng and V. Linetsky, Pricing discretely monitored barrier options and defaultable bonds in Lévy process models: a fast Heston transform approach. *Math. Finance*, 18, 2008.

[37] G. Fusai and A. Meucci, Pricing discretely monitored Asian options under Lévy processes. *J. Banking and Finance.* 32, 2076–2088, 2008.

[38] R. Geske and H. Johnson, The American put valued analytically. *J. of Finance*, 39, 1984.

[39] L. A. Grzelak and C. W. Oosterlee, On the Heston model with stochastic interest rates *SIAM J. Financial Math.* 2: 255-286, 2011.

[40] V. Henderson and R. Wojakowski, on the equivalence of floating and fixed–strike Asian options, *J, Appl, Prob.*, 39 (2), 391–394, 2002.

[41] J. Hull and A. White , Pricing interest-rate derivative securities, *Rev. Fin. Studies*, 3 (1990) 573–592.

[42] P. Jaillet, E.I. Ronn, and S. Tompaidis Valuation of commodity-based swing option. *Management science*, December, 2003.

[43] K. Jackson, S. Jaimungal, and V. Surkov Option pricing with regime switching Lévy processes using Fourier space time–stepping. *Proc. 4th IASTED Intern. Conf. Financial Engin. Applic.*, pages 92–97, 2007.

[44] S. Jaimungal, K. Jackson, and V. Surkov, Option pricing with regime switching Lévy processes using Fourier space time-stepping. *Proc. 4th IASTED Intern. Conf. Financial Engin. Applic.*, 2007.

[45] A. G. Z. Kemna and A. C. F. Vorst, A pricing method for options based on average asset values, *J. Banking and Finance*, 14(1), 113–129, 1990.

[46] M. Kjaer Pricing of swing options in a mean–reverting model with jumps. *Quantitative analytics, Barclays capital*, January, 2007.

[47] S. G. Kou, A jump diffusion model for option pricing, *Management Science*, 48(8) (2002) 1086–1101.

[48] A. Lari–Lavassani, M. Simchi and A. Ware A discrete valuation of swing options. *Canadian applied mathematics quarterly*, Volume 9, Number 1, Spring 2001.

[49] D. Lemmens, L. Liang, J. Tempere, and A. De Schepper, Pricing bounds for discrete arithmetic Asian options under Lévy models. *Physica A: Statistical Mechanics and its Applications.* Vol. 389, Issue 22: 5193–5207, 2010.

[50] R. Lord, F. Fang, F. Bervoetsand C. W. Oosterlee, A fast and accurate FFT-based method for pricing early-exercise options under Lévy processes. *SIAM J. Sci. Comput.*, 30, 2008.

[51] R. Lord and C. Kahl, Optimal Fourier inversion in semi-analytical option pricing. *J. Comp. Finance*, 10, 2007.

[52] J. J. Lucia and E. S. Schwartz, Electricity prices and power derivatives, Evidence from the nordic power exchange, *Review of Derivatives Research*, 5 (2002) 5–50.

[53] D. B. Madan, P. P. Carr, H. Geman, and M. Yor, The fine structure of asset returns: An empirical investigation. *J. Business.*, 75, 2002.

[54] R. Merton, Option pricing when underlying stock returns are discontinuous, *J. Financial Economics*, 3 (1976) 125–144.

[55] E. Mordecki and J. Fajardo, Symmetry and duality in leacutevy markets. *Quantitative Finance*, 6, 2006.

[56] G.V. Pflug and N. Broussev Electricity swing option: Behavioral models and pricing. *European journal of operational research*, 2008.

[57] W. Schoutens, Lévy processes in finance: Pricing financial derivatives, *Wiley*, ISBN: 0-470-85156-2. 2003.

[58] E. S. Schwartz, The stochastic behavior of commodity prices: Implications for valuation and hedging, *Journal of Finance*, 54(3) (1997) 923–973.

[59] E. S. Schwartz, Valuing long-term commodity assets, *Financial Management*, 27 (1998) 57–66.

[60] E. S. Schwartz and J. E. Smith, Short-term variations and long-term dynamics in commodity prices, *Management Science*, 46 (2000) 893-911.

[61] V. Surkov, Parallel option pricing with Fourier space time–stepping method on graphics processing units. *Parallel Computing*, 36(7): 372-380, 2010.

[62] L. N. Trefethen, 'Is Gauss quadrature better than Clenshaw-Curtis?', *SIAM Review*, 50 (1), 67–87, 2008.

[63] G.E.Uhlenbeck and L.S.Ornstein, On the theory of Brownian motion. *Phys Rev,* 36:823–41, 1930.

[64] J. Vecer, A new PDE approach for pricing arithmetic average Asian options. *J. Computational Finance*, 4(4), 105–113, 2001.

[65] J.A.C. Weideman and L. N. Trefethen, The kink phenomenon in Fejér and Clenshaw–Curtis quadrature, *Numer. Mathematik*, 107 Issue 4, 2007.

[66] Y. Yamamoto, Double-exponential fast Gauss transform algorithms for pricing discrete lookback options. *Publ. Res. Inst. Math. Sci.*, 41:989–1006, 2005.

[67] A. B. Zeghal and M. Mnif Optimal multiple stopping and valuation of swing options in Lévy models. *International Journal of Theoretical and Applied Finance, 1267–1297*, 2006.

[68] B. Zhang and C. W. Oosterlee, Option pricing with COS method on Graphics Processing Unit, *Proceedings of the 23rd IEEE International Parallel & Distributed Processing Symposium,* 25–29, May, 2009.

[69] B. Zhang and C. W. Oosterlee, Acceleration of Option Pricing Technique on Graphics Processing Units, *Concurrency and Computation: Practice and Experience*, Wiley–Blackwell, online version available in February 2012, journal version to appear in 2012.

[70] B. Zhang and C. W. Oosterlee, An efficient pricing algorithm for swing options based on fourier cosine expansions. *Journal of Computational Finance*, to appear in 2012.

[71] B. Zhang, L. A. Grzelak and C. W. Oosterlee, Efficient pricing of commodity options with early–exercise under the Ornstein–Uhlenbeck process, *Applied Numerical Mathematics*, Volume 62 and Issue 2, 2012.

[72] B. Zhang and C. W. Oosterlee, Fourier cosine expansions and put call relations for Bermudan options. *Numerical Methods in Finance*, pp 325–352, 2011.

[73] B. Zhang and C. W. Oosterlee, Efficient Pricing of Asian Options under Lévy Processes based on Fourier Cosine Expansions *Part I: European-Style Products*, submitted in 2011, and available as TU Delft DIAM report 11–11.

[74] B. Zhang and C. W. Oosterlee, Efficient Pricing of Asian Options under Lévy Processes based on Fourier Cosine Expansions *II: Early–Exercise Features and GPU Implementation*, submitted in 2012.

[75] NVIDIA Cuda Compute Unified Device Architecture, Programming Guide, Version 1.0, 2007.

[76] NVIDIA CUDA Programming Guide, Version 4.0, 2011.

Curriculum vitae

Bowen Zhang was born in 1983 in Beijing, China. In 2001 she started her Bachelor's study at Beijing University of Technology, with specialization Applied Mathematics. She received her Bachelor of Science degree in 2005, together with a minor diploma in Economics. During the period of her Bachelor studies, she won the first prize in the China National Contest of Mathematical Modelling in the autumn of 2004. From 2005 to 2006 she worked as a computer engineer at the China International Intellec–Tech Corporation. In September 2006, she came to the Netherlands for her Masters study at Delft University of Technology (TU Delft), in the research group 'Risk and Environmental Modeling', under the supervision of Prof. R. Cooke. After receiving her Master of Science degree in August 2008, she continued with her PhD research at TU Delft, starting September 2008, under the supervision of Prof. C. W. Oosterlee. Her PhD project focuses on the efficient pricing of early–exercise options and exotic options based on the Fourier cosine expansions, in combination with parallel computing on the Graphics Processing Unit. In 2010, she has also worked in Paris, implementing the COS method in Premia, a French financial software package for option pricing, hedging and financial model calibration. After her PhD, she will work as Quantitative Analyst at The Royal Bank of Scotland (RBS), Amsterdam.

She has a diploma for Staatsexamen Nederlands als Tweede Taal (NT2).

List of publications

- Journal papers:

  1. B. Zhang, C.W. Oosterlee, An Efficient Pricing Method for Asian Options under Levy processes. Part II: Early-Exercise Features and GPU implementation, submitted in 2012.
  2. B. Zhang, C.W. Oosterlee, Efficient Pricing of Asian Options under Levy Processes based on Fourier Cosine Expansions. Part I: European-Style Products, submitted in 2011.
  3. B. Zhang, C.W. Oosterlee, An Efficient Pricing Algorithm For Swing Options Based On Fourier Cosine Expansions, *Journal Of Computational Finance*, to appear.
  4. B. Zhang, C.W. Oosterlee, Acceleration of Option Pricing Technique on Graphics Processing Units, *Concurrency and Computation: Practice and Experience*, Wiley–Blackwell, online version available on 6 February 2012, journal version to appear in 2012.
  5. B. Zhang, L.A. Grzelak, and C.W. Oosterlee, Efficient Pricing of Commodity Options with Early-Exercise under the Ornstein-Uhlenbeck Process, *Applied Numerical Mathematics*, pp. 91–111, Volume 62, Issue 2, 2012.

- Book Chapters:

  1. B. Zhang and C.W. Oosterlee, Fourier Cosine Expansions and the Put-Call Relations for Bermudan Options, *Numerical Methods in Finance*, pp. 323–350, Springer Proceedings in Mathematics, Volume 12, 2012.

Proceedings and Presentations

- Presentations

  1. Speaker at SIAM conference on Financial Mathematics & Engineering, Minneapolis, Minnesota, United States, July 9–11, 2012,
  2. Speaker at 11th Winter school on Mathematical Finance, Luteren, the Netherlands, January 23–25, 2012.
  3. Speaker at Quantitative Methods in Finance Conference, Sydney, Australia, December 14-17, 2011.
  4. Speaker at Reunion de livraison de Premia 13, Inria, Paris, France, March 17, 2011.
  5. Speaker at 6th World Congress of the Bachelier Finance Society, Toronto, Canada, June 22–26, 2010.
  6. Speaker at the 23rd IEEE International Parallel & Distributed Processing Symposium, 25-29, May, 2009, Rome, Italy.

- Proceedings

  1. Anthony Thornton, Bowen Zhang et.al Modeling and optimization of Algae Growth. *In proceedings of the 72nd European Study Group Mathematics with Industry, January 25–29, 2010, Centrum Wiskunde & Informatica (CWI), Amsterdam.*
  2. Bowen Zhang and Cornelis W. Oosterlee, Option pricing with COS method on Graphics Processing Unit, *In proceedings of the 23rd IEEE International Parallel & Distributed Processing Symposium, 25-29, May, 2009,* ISBN: 978-1-4244-3750-4 ISSN 1530-2075.