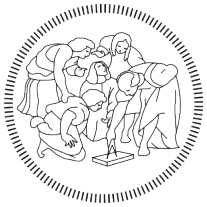
Politecnico di Milano
School of Industrial and Information Engineering
Master of Science in Mathematical Engineering:
Quantitative Finance



POLITECNICO
MILANO 1863

COS method for option pricing under a regime-switching model with time-changed Lévy processes

---

Supervisor: Daniele Marazzina

Master thesis by:
Trebbi Aurora
ID: 913416

Academic year: 2019-2020

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Daniele Marazzina, who with his courses, in particular "Computational finance", has been fundamental in transmitting me interest in mathematical finance and especially for having offered me his valuable advice throughout the process of this dissertation.

A special thanks goes to my family. To my father, for being present in every choice of my life, for his unconditional love and for being my role model. To my mother, for always believing in me. To my brother, for listening to my doubts, even knowing nothing about the subject, just to be supportive. Finally, to my beloved grandmothers, Cesarina and Agata, both of them for bringing me luck at each exam in their own way and in particular, Cesarina, for being the only person willing to remain in religious silence just to let me study.

Last, but not least, I would like to thank all my friends, those I have known for many years and those I met during this course of study, for sharing with me goals and failures and distracting me on weekends. In particular Chiara, I could not wish for a better roommate, and Lucrezia, for being always by my side, no matter what.

Abstract

A significant task for a financial engineer is to continuously look for both models that better describe reality and more efficient pricing methods for derivatives. For such a reason, this dissertation focus on the application of the COS method under a regime-switching model. The method is called the COS method, since it is based on the use of Fourier cosine series expansion. Moreover, we adopt the theory of regime-switching models since they allow to reproduce the evolution over time of market conditions by including the possibility of switches among market scenarios (e.g. bull or bear market) in the model parameters. The COS method enables switches among different models without any additional cost. Its efficiency is further shown implementing a numerical test to compare its performances with respect to other well-known methods. Its versatility, instead, is shown by the multiplicity of financial options to which it can be applied, starting from the basic vanilla options, up to the path-dependent options.

Contents

List of Figures

List of Tables

Introduction

A derivative is a financial instrument the value of which depends on other underlying entities, such as assets, indices, interest rates... Options are just a class of derivatives and in general there exists a great variety of these, ranging from the simplest ones, which are the so called *vanilla options*, to the most complex ones, the so called *exotic options*.
Vanilla options include call and put options. A call (put) option is a contract that gives the owner the right but not the obligation to buy (sell) the underlying asset at a prespecified price: the strike price. Defining S_T as the underlying value at the option maturity T and K as the strike price, the call option has payoff $(S_T - K)^+$. Similarly, put option payoff is $(K - S_T)^+$.
Belonging to the class of exotic options there are the *path dependent options*, the values of which depend not only on the underlying value at maturity, but also on the path history of the underlying price during the option's life. This leads us to another important distinction: *European options* which can be exercised only at maturity and *American options*, which instead can be exercised at any time up to maturity.
Option pricing is one of the main duties of a financial engineer. Analytical prices are often unknown and therefore increasingly efficient numerical methods are sought.
The purpose of this dissertation is to present the application of the COS method, known to be fast, accurate and versatile, under a regime-switching model. This method, originally introduced in [7], has been extended to the regime-switching model in [17]. Here we expand the discussion to Asian options, taking as starting point [20].

We present the content of the dissertation in more detail:

Chapter 1 provides the basics of computational finance. It gives an overview of the most popular models adopted for describing the dynamics of assets prices. Starting from the well known Black-Scholes model, we arrive to the more realistic Lévy processes, distinguishing between finite activity processes, such as *Merton* and *Kou*, and infinite activity ones, *VG*, *NIG*, *CGMY* and *Meixner*. Then, in order to better

describe the features of the historical financial series, stochastic time-changed Lévy and Heston models are tackled. Finally, in order to be able to price financial derivatives, we review the concept of risk-neutral valuation.

Chapter 2 presents the theory of regime-switching models, the great benefit of which is to fully represent the stochastic features of financial markets by allowing switches in the parameters of the underlying process. Switches that are modeled via a continuous time, finite state Markov chain. Then, the notation that will be adopted for the whole dissertation is presented and the pricing problem seen in the previous chapter is readapted to this new framework.

Chapter 3 deals with adapting the COS method for option pricing introduced in [7] to the regime-switching framework, as it is done in [17]. Here, we describe in detail all the steps required by the method for the valuation of European options. Finally we propose a remark. With a numerical test we show that, under certain conditions, using the COS method for pricing call options can result in significant errors, thus, whenever possible, the use of the well known put-call parity relation is suggested.

Chapter 4 aims to show that the financial world is in fact best described when combining different time-changed Lévy models with regime-switching. In order to do so, a calibration of the various models previously presented and also of some of their possible combinations is set up. Hence, first of all, the problem is mathematically formalized and later implemented exploiting real market data.

Chapter 5 aims to verify the performances of the COS method with respect to other popular methods in literature, specifically the Lattice method, the Fourier space time stepping method and the Partial differential equation method. For each of them, a brief theoretical explanation is presented and the formulas used for pricing are derived. Then, a numerical test shows that the efficiency of the COS method with respect to the other methods is already achieved by pricing options under a two-state Black-Scholes model with constant interest rate and dividend yield, which is the simplest of the regime-switching models and thus all methods should perform at their best.

Chapter 6 is the first in a series of chapters which aspire to extend the COS method to pricing various types of options. Here digital and butterfly options are addressed and since they differ from the vanilla options only in the payoff, their derivation is straightforward. The chapter ends with a numerical test showing that the method remains fast and efficient.

Chapter 7 extends the COS method to Bermudan options, which differ from vanilla options as they can be early exercised on a set of prespecified dates. Therefore, a backward in time recursive procedure to find the series coefficients of the option values at the first early-exercise date is needed. Finally a numerical test is proposed.

Chapter 8 follows up the previous chapter. Indeed, barrier options have a payoff depending upon the underlying price touching a preset barrier level and thus even this type of options requires a backward in time recursive procedure. A numerical test is then proposed once again.

Chapter 9 describes an approximation technique to extend the pricing to continuously monitored options, the Richardson extrapolation, and verifies its efficiency in a numerical test.

Chapter 10 focus on the application of the COS method to pricing Asian options, path-dependent options which, differently from Bermudan and barrier options, require a forward in time recursive procedure. This to demonstrate the great versatility of the COS method which can be easily extended to various types of options.

As regards Appendices, Appendix A displays the formulas for the cumulants of the various models taken into consideration, which are needed in implementing the COS method. Appendix B contains the data used in the calibration tests and Appendix C reports the most important MATLAB® codes developed to get the results discussed throughout the dissertation.

Chapter 1

Preliminary notions

1.1 Basic tools

In this section we present the main mathematical notions that are needed in order to price the various options by referring to [6].
For the basics of probability, such as sigma-algebras, probability measure/space, stochastic process, we refer to [11].

Definition 1.1.1 ***Characteristic function***
Let X be an $\mathbb{R}^d$-valued random variable. Its characteristic function φ_X defined on $\mathbb{R}^d$ is

$$\forall u \in \mathbb{R}^d,\ \varphi_X(u) = \mathbb{E}[e^{\mathtt{i}u\cdot X}] = \int_{\mathbb{R}^d} e^{\mathtt{i}u\cdot x} f(x)dx \tag{1.1}$$

where f is the density function of X and $\mathtt{i}$ denotes the *imaginary unit*, which is defined solely by $\mathtt{i}^2 = -1$.
Since φ_X is continuous in zero, we can write it as

$$\varphi_X(u) = e^{\psi_X(u)} \tag{1.2}$$

ψ_X is called the characteristic exponent of X.

Proposition 1.1.1 ***Characteristic function and moments***

1. *If $\mathbb{E}[|X^n|] < \infty$ then φ_X has n continuous derivatives at $u = 0$ and*

$$\forall k = 1, \dots, n, \quad m_k \equiv \mathbb{E}[X^k] = \frac{1}{\mathtt{i}^k}\frac{\partial^k \varphi_X}{\partial u^k}(0) \tag{1.3}$$

2. *If φ_X has $2n$ continuous derivatives at $u = 0$ then $\mathbb{E}[|X^{2n}|] < \infty$ and*

$$\forall k = 1, \dots, 2n, \quad m_k \equiv \mathbb{E}[X^k] = \frac{1}{\mathtt{i}^k}\frac{\partial^k \varphi_X}{\partial u^k}(0) \tag{1.4}$$

3. *X possesses finite moments of all orders iff $u \mapsto \varphi_X(u)$ is C^∞ at $u = 0$. Then the moments of X are related to the derivatives of φ_X by*

$$m_n \equiv \mathbb{E}[X^n] = \frac{1}{\mathtt{i}^n} \frac{\partial^n \varphi_X}{\partial u^n}(0) \tag{1.5}$$

Definition 1.1.2 ***Moment generating function***
The moment generating function of $\mathbb{R}^d$-valued random variable X is the function M_X defined by

$$\forall u \in \mathbb{R}^d, \ M_X(u) = \mathbb{E}[e^{u \cdot X}] \tag{1.6}$$

When M_X is well defined, it can be formally related to the characteristic function by

$$M_X(u) = \varphi_X(-\mathtt{i}u) \tag{1.7}$$

If the moment generating function M_X of a random variable X on $\mathbb{R}$ is defined on a neighborhood of zero, then in particular all moments of X are finite and can be recovered from the derivatives of M in the following manner

$$m_n = \frac{\partial^n M_X}{\partial u^n}(0) \tag{1.8}$$

Moreover, the characteristic exponent of X defined in (1.2) is also the *cumulant generating function*. In particular, we have the following definition.

Definition 1.1.3 ***Cumulants***
The cumulants of X are defined by

$$c_n(X) = \frac{1}{\mathtt{i}^n} \frac{\partial^n \psi_X}{\partial u^n}(0) \tag{1.9}$$

By exploiting Proposition 1.1.1 and equations (1.7) and (1.8), we can rewrite the formula for computing the cumulants as

$$c_n(X) = \frac{\partial^n \log(\varphi_X(-\mathtt{i}u))}{\partial u^n}(0) \tag{1.10}$$

This reformulation will be useful in Chapter 3.

Definition 1.1.4 ***Brownian motion***
$W = (\Omega, \mathcal{F}, (\mathcal{F}_t)_{t \geq 0}, (W_t)_{t \geq 0}, \mathbb{P})$ stochastic process is a Wiener process, also known as Brownian motion, if

- $W_0 = 0$ *a.s.*
- $W_t - W_s \perp\!\!\!\perp \mathcal{F}_s, \ \forall s \in [0, t]$
- $W_t - W_s \sim N(0, t - s)$

Definition 1.1.5 ***Martingale***
A stochastic process $M = (M_t)_{t\geq 0}$ *valued on* $\mathbb{R}$ *is a martingale if*

- $M_t \in L^1(\Omega, \mathcal{F}, \mathbb{P}) \quad \forall t$
- $\mathbb{E}[M_t|\mathcal{F}_s] = M_s, \ \forall s \in [0,t]$

1.2 Black-Scholes

The price of an option depends, as mentioned before, on the values of the underlying asset. Hence, modeling the underlying dynamics is a fundamental step in derivative pricing.
In 1973 Black and Scholes proposed a model for the underlying stock dynamics that became the reference for the financial world. Specifically, they considered a financial market consisting of only two assets: a riskless asset B_t and a risky one S_t. The former has the following dynamics

$$\begin{cases} dB_t = rB_t dt \\ B_0 = 1 \end{cases} \tag{1.11}$$

where r is the risk-free interest rate and we assume it to be constant. If we also assume that B_t can deliver a continuous dividend yield δ, then equation (1.11) can be generalized to

$$\begin{cases} dB_t = (r-\delta)B_t dt \\ B_0 = 1 \end{cases} \tag{1.12}$$

which has solution $B_t = e^{(r-\delta)t}$.
The risky asset, instead, follows the dynamics of a geometric Brownian motion, i.e.

$$\begin{cases} dS_t = \mu S_t dt + \sigma S_t dW_t \\ S_0 > 0 \ \text{ given} \end{cases} \tag{1.13}$$

where W_t is a Brownian motion defined on the historical probability measure $\mathbb{P}$. Moreover, μ is the *drift* term and σ is the *diffusion* term and they represent the deterministic and the stochastic part of the process, respectively.
The solution of (1.13) is given by

$$S_t = S_0 e^{\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t} \tag{1.14}$$

which follows a log-normal distribution.
In Black-Scholes model, other than the probability measure $\mathbb{P}$, in order to price

options a so-called *risk-neutral* probability measure $\mathbb{Q}$ is introduced. We will cover this in detail in Section 1.6, here we just need to know that under this measure the evolution over time of the risky asset corresponds on average to that of the riskless one, i.e. we have the following the dynamics

$$\begin{cases} dS_t = (r-\delta)S_t dt + \sigma S_t dW_t \\ S_0 > 0 \quad \text{given} \end{cases} \tag{1.15}$$

We have to specify that here W_t is defined on $\mathbb{Q}$, hence it is not the same Brownian motion as in (1.13). However, for convenience, we use the same symbol.
Then by comparing (1.13) and (1.15) it follows that $\mu = r - \delta$.
This financial model has been very successful since it is the simplest model that imposes positive prices and also provides closed formulas for the valuation of European options. In the case of European call option, the valuation formula extended to the case in which the underlying pays a continuous dividend yield δ [1] is given by

$$C(S_0, K, t, r, \delta, \sigma) = S_0 e^{-\delta t} N(d_1) - K e^{-rt} N(d_2) \tag{1.16}$$

where

$$d_{1,2} = \frac{\log\left(\frac{S_0}{K}\right) + \left(r - \delta \pm \frac{\sigma^2}{2}\right)t}{\sigma\sqrt{t}} \tag{1.17}$$

For the standard proof see [2].
However, it is now a fact that Black-Scholes model is not suitable for option pricing in the real economy, indeed its assumptions lead it to be an extreme simplification of reality. Specifically, by analyzing the historical financial series, some empirical facts common to a wide set of financial assets came out. These are known as *stylized facts* (we refer to [5] and [6]) and highlight the limitations of Black-Scholes model

1. **heavy tailedness and negative skewness** the empirical distribution shows heavier tails than those of a log-normal distribution and a negative skewness. This leads to a problem of underestimation of extreme events (in particular the negative ones)

2. **volatility clustering** high-volatility events tend to cluster in time

3. **leverage effect** the volatility of an asset is negatively correlated with the return of that asset. It is intuitive that when a stock price falls, the market becomes uncertain about the future of that quoted company and thus the perceived riskiness increases

[1] this extension of BS is known as *Garman-Kohlhagen* model

4. **volatility smile/surface** σ does not remain constant over time and it depends also on the strike K, thus plotting σ as a function of K (and T) we obtain the typical volatility smile (surface)

5. **non continuity** jumps in the price dynamics are often observed

Unfortunately, it is not possible to have a stochastic process that includes all the properties listed above. Anyway, to overcome these problems, many alternative models have been proposed over the years and some of the most popular ones in literature will be analyzed in the following sections.

1.3 Jump processes

The Lévy processes are stochastic processes incorporating jumps in the stock price dynamics in order to account for its fat tailed and leptokurtic distribution. In this section, we present the main concepts behind the vast theory of these processes. For a detailed discussion, refer to [6].

1.3.1 Poisson process

The Poisson process is a fundamental example of stochastic process with discontinuous trajectories and will be used as a building block for more complex jump processes.
Before defining the Poisson process, we need to recall the concepts of exponential random variable:

$$Y \sim \mathcal{E}(\lambda) \qquad f(y) = \lambda e^{-\lambda y} \mathbb{1}_{y \geq 0} \tag{1.18}$$

and of Poisson random variable:

$$N \in \mathbb{N} \text{ r.v.}, \quad N \sim P(\lambda) \qquad \mathbb{P}(N = n) = e^{-\lambda} \frac{\lambda^n}{n!}, \quad \forall n \in \mathbb{N} \tag{1.19}$$

Now we can give the following definition.

Definition 1.3.1 ***Poisson Process***
Let $(\tau_i)_{i \geq 1}$ be a sequence of independent exponential random variables with parameter λ and $T_n = \sum_{i=1}^{n} \tau_i$. The process $(N_t, t \geq 0)$ defined by

$$N_t = \sum_{n \geq 1} \mathbb{1}_{t \geq T_n} \tag{1.20}$$

is called a Poisson process with intensity λ

The Poisson process is therefore defined as a counting process: it counts the number of random times (T_n) which occur between 0 and t, where $(T_n - T_{n-1})_{n \geq 1}$ is an i.i.d. sequence of exponential variables. Moreover, among its properties there are:

- $\forall \omega$, the sample path $t \to N_t(\omega)$ is piecewise constant and increases by jumps of size 1
- if t is fixed, $N_t \sim P(\lambda t)$
- N_t is cadlag: its paths are right continuous and left limited
- its density is $\mathbb{P}(N_t = n) = e^{-\lambda t}\frac{(\lambda t)^n}{n!}$, $\forall n \in \mathbb{N}$
- its characteristic function is $\varphi(u) = \mathbb{E}[e^{\mathbf{i}uN_t}] = e^{\lambda t(e^{\mathbf{i}u}-1)}$, $\forall u \in \mathbb{R}$
- independent increments: $t_1 < \cdots < t_n,\ N_{t_n} - N_{t_{n-1}} \perp\!\!\!\perp N_{t_{n-1}} - N_{t_{n-2}} \dots$
- homogeneous increments: $\forall t > s,\ N_t - N_s \sim N_{t-s}$

In general, N_t is not a martingale, but to make it one we can define

$$\tilde{N}_t = N_t - \mathbb{E}[N_t] = N_t - \lambda t \tag{1.21}$$

$\tilde{N}_t$ is called *compensated Poisson process*.
Its characteristic function is $\varphi(u) = e^{\lambda t(e^{\mathbf{i}u} - \mathbf{i}u - 1)}$.

1.3.2 Lévy processes

To build a jump process to better describe the stock dynamics, so far we have the following ingredients: W_t and N_t. However, by taking N_t the jump sizes will always remain equal to 1. To obtain different jump sizes, we need a different process: the *compound Poisson process*, which is a particular case of a Lévy process as stated in Proposition 1.3.2.

Definition 1.3.2 ***Lévy process***
A stochastic process $(X_t)_{t\geq 0}$ defined on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with values in $\mathbb{R}^d$ is a Lévy process if it possesses the following properties:

- *cadlag: right continuous and left limited trajectories*
- $X_0 = 0$
- *independent increments:* $0 < t_1 < \cdots < t_n,\ \ X_{t_n} - X_{t_{n-1}} \perp\!\!\!\perp X_{t_{n-1}} - X_{t_{n-2}} \dots$
- *stationary increments:* $X_{t+h} - X_t \sim X_h - X_0 = X_h$
- *stochastic continuity:* $\forall \epsilon > 0,\ \lim_{h\to 0} \mathbb{P}(|X_{t+h} - X_t| \geq \epsilon) = 0$

Proposition 1.3.1 ***Characteristic function of a Lévy process***
Let $(X_t)_{t\geq 0}$ be a Lévy process on $\mathbb{R}^d$. There exists a continuous function $\psi : \mathbb{R}^d \to \mathbb{R}$ called the characteristic exponent of X, such that:

$$\mathbb{E}[e^{\mathrm{i}u\cdot X_t}] = e^{t\psi(u)}, \quad u \in \mathbb{R}^d \tag{1.22}$$

The Wiener process is a Lévy process itself, but it is the only one with continuous trajectories. Also Poisson processes are Lévy processes.
Black-Scholes model in the stock price is not a Lévy, indeed we know that its dynamics is given by (1.13) but $S_0 \neq 0$ otherwise $S_t \equiv 0$. However, we can take a Lévy process X_t and define $S_t = S_0 e^{X_t}$, where X_t has driftless characteristic exponent equal to $\psi(u) = -\frac{\sigma^2 u^2}{2}$.

Definition 1.3.3 ***Compound Poisson***
A compound Poisson process with intensity $\lambda > 0$ and jump size distribution f is a stochastic process X_t defined as

$$X_t = \sum_{i=1}^{N_t} Y_i \tag{1.23}$$

where jumps sizes Y_i are i.i.d. with distribution f and (N_t) is a Poisson process with intensity λ, independent from $(Y_i)_{i\geq 1}$

The Poisson process itself can be seen as a compound Poisson process such that $Y_i \equiv 1$.

Proposition 1.3.2 *$(X_t)_{t\geq 0}$ is compound Poisson process if and only if it is a Lévy process and its sample paths are piecewise constant functions*

Proposition 1.3.3 *Let $(X_t)_{t\geq 0}$ be a compound Poisson process on $\mathbb{R}^d$. Its characteristic function has the following representation*

$$\mathbb{E}[e^{\mathrm{i}u\cdot X_t}] = \exp\left\{t\lambda \int_{\mathbb{R}^d} (e^{\mathrm{i}u\cdot x} - 1) f(dx)\right\}, \quad \forall u \in \mathbb{R}^d \tag{1.24}$$

where λ denotes the jump intensity and f the jump size distribution

Introducing a new measure $\nu(A) = \lambda f(A)$, we can rewrite the formula (1.24) as

$$\mathbb{E}[e^{\mathrm{i}u\cdot X_t}] = \exp\left\{t \int_{\mathbb{R}^d} (e^{\mathrm{i}u\cdot x} - 1) \nu(dx)\right\}, \quad \forall u \in \mathbb{R}^d \tag{1.25}$$

ν is called the Lévy measure of process $(X_t)_{t\geq 0}$, it is a positive measure on $\mathbb{R}^d$ but not a probability measure since $\int \nu(dx) = \lambda \neq 1$.
To every cadlag process and in particular to every compound Poisson process $(X_t)_{t\geq 0}$

on $\mathbb{R}^d$ one can associate a random measure on $\mathbb{R}^d \times [0,\infty)$, describing the jumps of X: for any measurable set $B \subset \mathbb{R}^d \times [0,\infty)$

$$J_X(B) = \#\{(t, X_t - X_{t^-}) \in B\} \tag{1.26}$$

$\forall A \subset \mathbb{R}^d$ measurable, if $B = [t_1, t_2] \times A$ this measure counts the number of jumps occurring between t_1 and t_2 and having their sizes belonging to A.
In the case of compound Poisson process with intensity λ and jump size distribution f, it can be proved that J_X has intensity measure $\mu(dx \times dt) = \nu(dx)dt = \lambda f(dx)dt$. This suggests an alternative interpretation of the Lévy measure of a compound Poisson process as the average number of jumps per unit time.

Definition 1.3.4 ***Lévy measure***
Let $(X_t)_{t\geq 0}$ be a Lévy process on $\mathbb{R}^d$. The measure ν on $\mathbb{R}^d$ defined by

$$\nu(A) = \mathbb{E}[\#\{t \in [0,1] : \Delta X_t \neq 0, \Delta X_t \in A\}], \quad A \in \mathcal{B}(\mathbb{R}^d) \tag{1.27}$$

is called the Lévy measure of X: $\nu(A)$ is the expected number, per unit time, of jumps whose size belongs to A

This definition implies that every compound Poisson process can be represented as

$$X_t = \sum_{s\in[0,t]} \Delta X_s = \int_{[0,t]\times\mathbb{R}^d} x J_x(ds \times dx) \tag{1.28}$$

where J_X is a Poisson random measure with intensity measure $\nu(dx)dt$. Hence, every piecewise constant Lévy process X_t^0 can be be rewritten as the sum of its jumps.
Given a Brownian motion with drift and diffusion $\mu t + \sigma W_t$, independent from X^0, the sum

$$X_t = \mu t + \sigma W_t + X_t^0 = \mu t + \sigma W_t + \sum_{i=1}^{N_t} Y_i \tag{1.29}$$

defines another Lévy process know as *Jump diffusion model*. However the Lévy measure ν, due to the cadlag property, is finite for any compact set A such that $\{0\} \notin A$, but it is not necessarily a finite measure: if $\{0\} \in A$, X_t may have an infinite number of infinitesimal jumps. In this case the sum of jumps becomes an infinite series and its convergence imposes some conditions on ν, under which we obtain a decomposition of X_t similar to the one above.

Proposition 1.3.4 ***Lévy-Itô decomposition***
Let $(X_t)_{t\geq 0}$ be a Lévy process on $\mathbb{R}^d$ and ν its Lévy measure.

- *for any measurable compact set $A \subset \mathbb{R}^d \backslash \{0\}$: $\nu(A) < \infty$*
- $\int_{|x|\leq 1} |x|^2 \nu(dx) < \infty$

- $\int_{|x|>1} \nu(dx) < \infty$
- $\exists \mu \in \mathbb{R}^d$ *and a brownian motion* $(W_t)_{t\geq 0}$ *with covariance matrix* A *such that*

$$X_t = \mu t + W_t + X_t^l + \lim_{\epsilon \to 0} \tilde{X}_t^\epsilon \tag{1.30}$$

where

$$X_t^l = \int_{|x|>1,\ s\in[0,t]} x J_X(ds \times dx)$$
$$\tilde{X}_t^\epsilon = \int_{\epsilon\leq|x|\leq 1,\ s\in[0,t]} x\big(J_X(ds \times dx) - \nu(dx)ds\big)$$

This decomposition entails that for every Lévy process there exist a vector μ, a positive definite matrix A and a positive measure ν that uniquely determine its distribution. The triplet (μ, A, ν) is called *characteristic triplet* of the Lévy process. The following theorem express the relation existing between the characteristic triplet and the characteristic exponent of a Lévy process.

Theorem 1.3.1 ***Lévy-Khinchin representation***
Let $(X_t)_{t\geq 0}$ *be a Lévy process on* $\mathbb{R}^d$ *with characteristic triplet* (μ, A, ν)*. Then*

$$\mathbb{E}[e^{\mathtt{i}u\cdot X_t}] = e^{t\psi(u)},\ u \in \mathbb{R}^d \tag{1.31}$$
$$\text{with}\ \ \psi(u) = \mathtt{i}\mu \cdot u - \frac{1}{2}u \cdot Au + \int_{\mathbb{R}^d} (e^{\mathtt{i}u\cdot x} - 1 - iu \cdot x\mathbb{1}_{|x|\leq 1})\nu(dx) \tag{1.32}$$

Moreover, from the Lévy-Itô decomposition it follows also that every Lévy process is the combination of a continuous part $X_t^c = \mu t + W_t$ and a discontinuous part, composed by a finite sum of big jumps X_t^l and a possibly infinite sum of small jumps $\tilde{X}_t^\epsilon$.
Hence, we can define the following distinctions:

- if $\int_{|x|\leq 1} \nu(dx) < \infty$ we have a *finite activity* Lévy process, otherwise an *infinite activity*
- if $\int_{|x|\leq 1} |x|\nu(dx) < \infty$ we have a *finite variation* Lévy process, otherwise an *infinite variation*

In other words, finite activity Lévy processes are essentially diffusion processes punctuated by jumps at random intervals. The distribution of their jump sizes is known and they are composed by a Brownian motion plus a compound Poisson. In this category, the most popular processes are the ones of *Merton* and *Kou*. The infinite

activity Lévy processes, instead, present an infinite number of infinitesimal jumps, thus the distribution of their jump sizes does not exist (jumps arrive infinitely often). Moreover they do not necessarily contain a Brownian component (in that case the characteristic triplet becomes $(\mu, 0, \nu)$) as the dynamics of jumps is already rich enough to generate nontrivial small time behaviour. Many models from this class can be constructed via Brownian subordination, it is the case of *Variance Gamma*, *Normal Inverse Gaussian* and *CGMY* processes.

1.3.3 Building Lévy processes

As we mentioned above, one way to construct a Lévy process consists in the Brownian subordination. A *subordinator* is essentially a Lévy process $(X_t)_{t\geq 0}$ such that $X_t \geq 0 \ \ \forall t$.
Let S_t be a subordinator and let W_t be a Brownian motion independent from S. Subordinating Brownian motion with drift $X_t = \theta t + \sigma W_t$ by the process S we obtain a new Lévy process $Y_t = \theta S_t + \sigma W(S_t) = X_{S_t}$. This process is still a Brownian motion but observed in the stochastic time scale given by S_t. The financial interpretation is that of *business time*: time speeds up or slows down during periods of high or low business activity, respectively.
Let us consider the *tempered α-stable* subordinator, which is a three-parameter process with Lévy measure

$$\rho(x) = \frac{ce^{-\lambda x}}{x^{1+\alpha}} \mathbb{1}_{x>0} \tag{1.33}$$

where $c > 0$ is the intensity of the jump size, $\lambda > 0$ governs the decay of big jumps and $0 \leq \alpha < 1$ governs the importance of small jumps. Usually it is convenient to use the following reparametrization

$$\rho(x) = \frac{1}{\Gamma(1-\alpha)} \left(\frac{1-\alpha}{\nu}\right)^{1-\alpha} \frac{e^{-(1-\alpha)x/\nu}}{x^{1+\alpha}} \tag{1.34}$$

where ν is the variance of the subordinator at time 1 and it actually determines how random the time change is.
By time changing an independent Brownian motion (with drift θ and volatility σ) by a tempered α-stable subordinator, we get the so-called *normal tempered stable process* the characteristic exponent of which, in general case, is

$$\psi(u) = \frac{1-\alpha}{\nu\alpha} \left\{ 1 - \left(1 + \frac{\nu(-\mathtt{i}\theta u + u^2\sigma^2/2)}{1-\alpha} \right)^{\alpha} \right\} \tag{1.35}$$

The choice $\alpha = 0$ corresponds to a *Variance-Gamma* process (VG) and $\alpha = 0.5$ to a *Normal Inverse Gaussian* (NIG). In the following sections, we will see some details about the most popular Lévy models in literature.

1.3.4 Merton

It is a finite activity Lévy process, with the form (1.29) where the jumps in the log-price X_t folllow a Gaussian distribution $Y_i \sim N(\tilde{\mu}, \tilde{\sigma}^2)$. Overall it has 4 parameters:

- $\sigma \in [0, 1]$ diffusion volatility
- $\lambda > 0$ jump intensity
- $\tilde{\mu} \in \mathbb{R}$ mean jump size
- $\tilde{\sigma} \in [0, 1]$ standard deviation of jump size

The characteristic exponent is given by

$$\psi(u) = -\frac{\sigma^2 u^2}{2} + \lambda\left(e^{-\frac{\tilde{\sigma}^2 u^2}{2} + \mathtt{i}\tilde{\mu}u} - 1\right) \tag{1.36}$$

1.3.5 Kou

It is a finite activity Lévy process, with the form (1.29) where the jumps in the log-price X_t follow an asymmetric exponential distribution with density

$$f(dx) = [p\lambda_+ e^{-\lambda_+ x}\mathbb{1}_{x>0} + (1-p)\lambda_- e^{-\lambda_- |x|}\mathbb{1}_{x<0}]dx \tag{1.37}$$

Thus, overall there are 5 parameters:

- $\sigma \in [0, 1]$ diffusion volatility
- $\lambda > 0$ jump intensity
- $\lambda_+ > 0$ governs the decay of the tails for the distribution of positive jump sizes
- $\lambda_- > 0$ as λ_+ but for negative jump sizes
- $p \in [0, 1]$ represents the probability of an upward jump

The characteristic exponent is given by

$$\psi(u) = -\frac{\sigma^2 u^2}{2} + \mathtt{i}u\lambda\left(\frac{p}{\lambda_+ - \mathtt{i}u} - \frac{1-p}{\lambda_- + \mathtt{i}u}\right) \tag{1.38}$$

1.3.6 VG

It is an infinite activity Lévy process, obtained as a normal tempered stable process with $\alpha = 0$. Overall it has 3 parameters:

- $\theta \in \mathbb{R}$ drift of the Brownian motion such that $1 - \nu\left(\theta + \frac{\sigma^2}{2}\right) > 0$

- $\sigma \in [0, 1]$ volatility of the Brownian motion
- $\nu > 0$ variance of the subordinator

Substituting $\alpha = 0$ in the equation (1.35), its characteristic exponent becomes

$$\psi(u) = -\frac{1}{\nu} \log \left(1 - \mathtt{i}u\theta\nu + \frac{\sigma^2 u^2 \nu}{2}\right) \tag{1.39}$$

1.3.7 NIG

It is an infinite activity Lévy process, obtained as a normal tempered stable process with $\alpha = 0.5$. Overall it has 3 parameters:

- θ drift of the Brownian motion
- σ volatility of the Brownian motion
- ν variance of the subordinator

Substituting $\alpha = 0.5$ in the equation (1.35), its characteristic exponent becomes

$$\psi(u) = \frac{1}{\nu}\left(1 - \sqrt{1 - 2\mathtt{i}u\theta\nu + \sigma^2 u^2 \nu}\right) \tag{1.40}$$

However, there is also available another parametrization (see [1]) such that the NIG process Y_t can be written as

$$Y_t = X_{S_t} \tag{1.41}$$

where X_t is a Brownian motion with drift β and diffusion coefficient 1 and where S_t is an Inverse Gaussian Lévy process with parameters γ and $\sqrt{\omega^2 - \beta^2}$, where $\omega > 0$ and $|\beta| < \omega$. In this case, the expression for the characteristic exponent is

$$\psi(u) = \gamma\left(\sqrt{\omega^2 - \beta^2} - \sqrt{\omega^2 - (\beta + \mathtt{i}u)^2}\right) \tag{1.42}$$

1.3.8 CGMY

It is a pure jump process, i.e. it does not contain Brownian part and it is obtained by generalizing the VG process. Overall it has 4 parameters:

- $C > 0$ measure of the level of activity
- $G > 0$ controls the rate of exponential decay on the right of the Lévy density
- $M > 0$ as G but regarding the left of the Lévy density

- $Y < 2$ determines the path behaviour: if $Y < 0$ the paths have finite jumps in any finite interval, if not the process has infinite activity. Moreover, if $Y \in [1, 2)$ the process is of infinite variation and if $Y = 0$ the process reduces to VG

Its characteristic exponent is given by

$$\psi(u) = C\Gamma(-Y)\Big((M - \mathtt{i}u)^Y - M^Y + (G + \mathtt{i}u)^Y - G^Y\Big) \tag{1.43}$$

For further details, see [4] or [16].

1.3.9 Meixner

As CGMY, it is a pure jump process. Overall there are 3 parameters: $\alpha > 0$, $|\beta| < \pi$, $\eta > 0$. Its characteristic exponent is given by

$$\psi(u) = 2\eta \log\left(\frac{\cos(\beta/2)}{\cosh((\alpha u - \mathtt{i}\beta)/2)}\right) \tag{1.44}$$

1.4 Stochastic time-changed Lévy processes

Lévy models can reflect stylized facts like negative skewness and heavy tailedness. However, the volatility clustering effect cannot be explained by the Lévy processes because of their stationary increment property. A solution to this is to allow the parameters to modify the timely evolution of volatility. Hence, the Lévy models with stochastic time were introduced in [3].
Stochastic time-changed Lévy models are simply Lévy processes subordinated or time changed by stochastic clocks. Let us consider the Lévy process X_t and the instantaneous rate of time change y_t such that the new clock is given by its integral

$$Y_t = \int_0^t y_s ds \tag{1.45}$$

For simplicity define the time-changed Lévy $Z_t = X_{Y_t}$, thus the stochastic time-changed risk-neutral asset price process is given by

$$S_t = S_0 \frac{e^{(r-\delta)t}}{\mathbb{E}[e^{Z_t}]} e^{Z_t} \tag{1.46}$$

By defining $s_t = \log(S_t/S_0)$ and noticing that $\mathbb{E}[e^{Z_t}] = \varphi_Z(-\mathtt{i})$ we can rewrite (1.46) as

$$s_t = (r - \delta)t + Z_t - \log \varphi_Z(-\mathtt{i}) \tag{1.47}$$

Hence, the characteristic function of s_t driven by a stochastic clock is given by

$$\varphi_s(u, t) = \mathbb{E}[e^{\mathtt{i}us_t}] = e^{\mathtt{i}u[(r-\delta)t - \log \varphi_Z(-\mathtt{i})]}\mathbb{E}[e^{\mathtt{i}uZ_t}] \tag{1.48}$$

where the characteristic function of Z_t is obtained simply as follows

$$\mathbb{E}[e^{\mathtt{i}uZ_t}] = \varphi_Y(-\mathtt{i}\psi_X(u), t, y_0) \tag{1.49}$$

Therefore the risk-neutral characteristic exponent of s_t is

$$\psi_s(u,t) = \mathtt{i}u(r-\delta)t + \log\left(\frac{\varphi_Y(-\mathtt{i}\psi_X(u), t, y_0)}{\varphi_Y(-\mathtt{i}\psi_X(-\mathtt{i}), t, y_0)^{\mathtt{i}u}}\right) \tag{1.50}$$

For the computations, we use two stochastic clocks very popular in literature: the Cox-Ingersoll-Ross (CIR) and the Gamma-Ornstein-Uhlenbeck (Γ-OU) processes.

1.4.1 CIR time change

The Cox–Ingersoll–Ross (CIR) process y_t solves the SDE

$$dy_t = \kappa(\eta - y_t)dt + \lambda\sqrt{y_t}dW_t \tag{1.51}$$

where κ is the rate of mean reversion, η is the long-run rate of time change and λ governs the volatility of the time change. The presence of the square root of y_t in the diffusion term excludes negative values for y_t. The characteristic function of the integrated CIR process is given by

$$\varphi^{\text{CIR}}(u,t) = \mathbb{E}[\exp(\mathtt{i}uY_t)|y_0] = \frac{\exp(\kappa^2\eta t/\lambda^2)\exp(2y_0\mathtt{i}u/(\kappa+\gamma\coth(\gamma t/2)))}{(\cosh(\gamma t/2)+\kappa\sinh(\gamma t/2)/\gamma)^{2\kappa\eta/\lambda^2}} \tag{1.52}$$

where $\gamma = \sqrt{\kappa^2 - 2\lambda^2\mathtt{i}u}$

1.4.2 Γ-OU time change

The Γ-OU process y_t solves the SDE

$$dy_t = -\lambda y_t dt + dz_{\lambda t} \tag{1.53}$$

where z is a subordinator and $\lambda > 0$. The characteristic function of the Γ-OU stochastic clock Y_t is given by

$$\begin{aligned}\varphi^{\Gamma\text{-OU}}(u,t) &= \mathbb{E}[\exp(\mathtt{i}uY_t)|y_0] \\ &= \exp\left(\frac{\mathtt{i}uy_0}{\lambda}(1-e^{-\lambda t}) + \frac{a_1\lambda}{\mathtt{i}u-\lambda a_2}\left(a_2\log\left(\frac{a_2}{a_2-\mathtt{i}u\lambda^{-1}(1-e^{-\lambda t})}\right)-\mathtt{i}ut\right)\right)\end{aligned} \tag{1.54}$$

1.5 Heston

In the previous section we introduced the stochastic time-changed Lévy processes in order to account for the effects of bursts of volatility. Another model used with this aim is the Heston model (1993), which belongs to the class of *stochastic volatility models*. Hence, it does not require a stochastic time change but it is the volatility parameter to be stochastic.
It is defined by the coupled two-dimensional SDE

$$\begin{cases} dX_t/X_t = \sqrt{V_t}dW_t^1 \\ dV_t = \kappa(\theta - V_t)dt + \epsilon\sqrt{V_t}dW_t^2 \end{cases} \tag{1.55}$$

where X_t represents an asset price process and V_t the instantaneous variance of relative changes to X_t, in the sense that the quadratic variation of dX_t/X_t over $[t, t+dt]$ is $V_t dt$. Moreover,

- $V_0 > 0$ spot variance
- $\theta > 0$ mean
- $\kappa > 0$ mean reversion speed
- $\epsilon > 0$ vol-of-vol
- W^1 and W^2 scalar Brownian motions such that $dW_t^1 dW_t^2 = \rho dt$ with correlation $\rho \in [-1, 1]$

The characteristic function is available in closed form. Its risk-neutral version is given by

$$\varphi(u,t) = e^{A(u,t)+B(u,t)V_t+\mathtt{i}uX_t} \tag{1.56}$$

where

$$A(u,t) = \frac{\kappa\theta}{\epsilon^2}\left((\kappa - \rho\epsilon u\mathtt{i} - D)t - 2\log\left(\frac{Ge^{-Dt}-1}{G-1}\right)\right) \tag{1.57}$$

$$B(u,t) = \frac{\kappa - \rho\epsilon u\mathtt{i} - D}{\epsilon^2}\left(\frac{1-e^{-Dt}}{1-Ge^{-Dt}}\right) \tag{1.58}$$

$$G = \frac{\kappa - \rho\epsilon u\mathtt{i} - D}{\kappa - \rho\epsilon u\mathtt{i} + D} \tag{1.59}$$

$$D = \sqrt{(\kappa - \rho\epsilon u\mathtt{i})^2 + u(\mathtt{i}+u)\epsilon^2} \tag{1.60}$$

V_t follows the dynamics of a Cox-Ingersoll-Ross process (CIR), which does not guarantee that V_t is greater than 0, indeed V_t can become null with a non-zero probability.

However, there exists a condition, sufficient but not necessary, for which V_t is always positive

$$2\kappa\theta \geq \epsilon^2 \tag{1.61}$$

This is known as *Feller's condition*.

1.6 Option pricing

We are now able to deal with the pricing problem, which is the aim of this dissertation. The concepts below are introduced to actually use the models previously presented. For a complete discussion, refer to [2].

Definition 1.6.1 ***Equivalent measures***
Given $\mathbb{P}$, $\mathbb{Q}$ measures on Ω we say that $\mathbb{Q} \sim \mathbb{P}$ if

$$\forall A \in \Omega, \quad \mathbb{Q}(A) > 0 \;\Leftrightarrow\; \mathbb{P}(A) > 0$$

Definition 1.6.2 ***Equivalent martingale measure***
Let $\mathbb{P}$ be the real measure, $\mathbb{Q}$ is an EMM iff

- $\mathbb{Q} \sim \mathbb{P}$
- *S_t/B_t is a martingale under $\mathbb{Q}$*

where B_t and S_t are the risky and riskless assets introduced in Section 1.2. EMM is often referred to as just "a martingale measure".

Theorem 1.6.1 ***First fundamental theorem***
The market is arbitrage free iff $\exists\mathbb{Q}$ martingale measure

An arbitrage opportunity is the possibility to make a riskless profit without net investment of capital. The principle of no arbitrage states that a model of a financial market should not allow for arbitrage possibilities. This theorem allows to obtain the formula of *risk neutral valuation*, so called since it is the kind of formula which would be used for valuing a derivative in a risk neutral world.

Theorem 1.6.2 ***Risk neutral valuation formula***
Given the market model (B, S) and an option with payoff $\Phi(S_T)$, the pricing formula takes the form

$$V(s,t) = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}[\Phi(S_T)|S_t = s] \tag{1.62}$$

where $\mathbb{Q}$ is a (not necessarily unique) martingale measure

Another important notion is that of completeness of the market. In simple terms, the market is said to be *complete* if any derivative product can be replicated from more basic instruments, such as cash and the underlying asset. The unique existence of $\mathbb{Q}$ is connected with this concept by the following theorem.

Theorem 1.6.3 ***Second fundamental theorem***
Assume that the market is arbitrage free. Then the market is complete iff $\mathbb{Q}$ is unique

In order to use equation (1.62), we need to determine $\mathbb{Q}$ such that S_t/B_t, i.e. the discounted stock price process that pays a continuous dividend yield, is a martingale. In particular, for Lévy models, the following proposition holds.

Proposition 1.6.1 *Let $(X_t)_{t\geq 0}$ be a Lévy process with characteristic triplet (μ, σ^2, ν) verifying*

$$\int_{|y|\geq 1} e^y \nu(dy) < \infty$$

Then $Y_t = e^{X_t}$ is a martingale iff

$$\psi(-\mathtt{i}) = \mu + \frac{\sigma^2}{2} + \int_{-\infty}^{\infty} (e^x - 1 - x\mathbb{1}_{|x|\leq 1})\nu(dx) = 0$$

Indeed Y_t is a martingale iff $\mathbb{E}[Y_t] = \mathbb{E}[Y_0] = \mathbb{E}[e^{X_0}] = 1$, but $\mathbb{E}[e^{X_t}] = e^{t\psi(-\mathtt{i})}$.
Therefore, we take X_t Lévy process and we define $S_t = S_0 e^{X_t}$. Then, since we want $e^{-(r-\delta)t} S_0 e^{X_t}$ to be a martingale, it follows from the above proposition that this is equivalent to require

$$\psi(-\mathtt{i}) = r - \delta \tag{1.63}$$

Hence, the idea is to define $\bar{X}_t$ such that

$$\begin{aligned} &\bar{X}_t \quad \text{with} \quad \big(0, \sigma^2, \nu\big) \\ &X_t \quad \text{with} \quad \big(r - \delta - \psi_{\bar{X}}(-\mathtt{i}), \sigma^2, \nu\big) \\ &\psi_X(u) = \mathtt{i}u\big(r - \delta - \psi_{\bar{X}}(-\mathtt{i})\big) + \psi_{\bar{X}}(u) \end{aligned}$$

Chapter 2

Regime switching models

In Chapter 1 we already discussed the need to better fit market data. Lévy processes, stochastic time-changed Lévy processes and stochastic volatility models where developed with this aim. However, to have a better view of the financial world, it is important to also consider its continuous changes over time, for instance periods of high and low volatility alternate unpredictably. This is the reason why regime switching models have been introduced: indeed they exploit a continuous time, finite state Markov chain to allow switches of model parameters which account for the influence of major economic factors on asset price dynamics in a parsimonious manner. Before beginning the discussion, we need to introduce the main concepts behind a Markov chain (refer to [12] and [18]).

2.1 Markov chain

A Markov chain is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. The formal definition is the following.

Definition 2.1.1 *A random process $(\alpha_t)_{t\geq 0}$ taking values in the state space $\mathcal{J}$, either finite or countable, is said to be a Markov chain if $\forall j \in \mathcal{J}$ and $\forall s \in [0,t]$ we have*

$$\mathbb{P}(\alpha_t = j \mid \alpha_r : r \leq s) = \mathbb{P}(\alpha_t = j \mid \alpha_s) \tag{2.1}$$

Let us denote the probability of transition from the state i to the state j $(i, j \in \mathcal{J})$ in the time interval $[s, t+s]$ by

$$p_{ij}(s, t+s) = \mathbb{P}(\alpha_{t+s} = j \mid \alpha_s = i) \tag{2.2}$$

If it holds true that $p_{ij}(s, t+s) = p_{ij}(0, t)$ for any $i, j \in \mathcal{J}$ and $s, t \geq 0$, then the Markov chain is said to be *homogeneous* and in this case we define $p_{ij}(s, t+s) := p_{ij}(t)$. In our discussion, we will assume Markov chains to be homogeneous.

2.1.1 Transition and Intensity matrices

A fundamental step in the study of continuous-time Markov chains is the introduction of the so called *transition rates*. For fixed t, the (possible infinite) matrix $\boldsymbol{P}(t) = (p_{ij}(t))$ is called *transition matrix* and it satisfies the following properties:

(a) $p_{ij}(t) \geq 0, \quad \forall i, j \in \mathcal{J}$

(b) $\sum_{j\in\mathcal{J}} p_{ij}(t) = 1, \quad \forall i \in \mathcal{J}$

(c) $p_{ij}(t+s) = \sum_{k\in\mathcal{J}} p_{ik}(t)p_{kj}(s), \quad \forall i, j \in \mathcal{J},\ \forall t, s \geq 0$

The last identity is usually referred to as the *Chapman-Kolmogorov* equation.
We also assume that $p_{ij}(t)$ are continuous for $t \geq 0$ and that

(d) $\lim_{t\to 0} p_{ij}(t) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$

It turns out that conditions (a)-(d) imply a great deal more than might be expected. One of these results is that $p_{ij}(t)$ are differentiable at $t = 0$, more precisely we have the following theorem.

Theorem 2.1.1 *If the transition probabilities are continuous, then the following limits exist*

$$\text{if } i \neq j,\ q_{ij} = \lim_{t\to 0^+} \frac{p_{ij}(t)}{t} \qquad q_{ii} = \lim_{t\to 0^+} \frac{p_{ii}(t) - 1}{t} \tag{2.3}$$

where $0 \leq q_{ij} < \infty$ *for* $i \neq j$ *and* $q_{ii} \leq 0$ *(possibly* $q_{ii} = -\infty$*).*
If the state space $\mathcal{J}$ *is finite, then* $q_{ii} > -\infty$ *and the following equations hold true*

$$\frac{d}{dt}p_{ij}(t) = \sum_{k\in\mathcal{J}} p_{ik}(t)q_{kj}, \quad p_{ij}(0) = \delta_{ij} \tag{2.4}$$

$$\frac{d}{dt}p_{ij}(t) = \sum_{k\in\mathcal{J}} q_{ik}p_{kj}(t), \quad p_{ij}(0) = \delta_{ij} \tag{2.5}$$

q_{ij} *are called transition rates*

The matrix $\boldsymbol{Q} = (q_{ij})_{i,j\in\mathcal{J}}$ is called *(infinitesimal) generator* of the Markov chain or simply *intensity matrix*.
Thanks to the differentiability of $p_{ij}(t)$ and the homogeneity of the Markov chain, we have that

$$\mathbb{P}(\alpha_{t+h} = j \mid \alpha_t = i) = \begin{cases} p_{ij}(h) = q_{ij}h + o(h), & \text{if } i \neq j \\ p_{ii}(h) = 1 + q_{ii}h + o(h), & \text{if } i = j \end{cases} \tag{2.6}$$

The above structure implies the relation $q_{ii} = -\sum_{j \neq i} q_{ij}$.
Finally notice that equations (2.4) and (2.5), known as forward and backward *Kolmogorov* equations, respectively, can be rewritten in matrix form, thus taking for instance the forward equation and a finite number of states, this is equivalent to

$$\boldsymbol{P}'(t) = \boldsymbol{P}(t)\boldsymbol{Q} \tag{2.7}$$

which has solution

$$\boldsymbol{P}(\Delta t) = \begin{bmatrix} p_{11} & \cdots & p_{1J} \\ \vdots & \ddots & \vdots \\ p_{J1} & \cdots & p_{JJ} \end{bmatrix} = \boldsymbol{P}(0)e^{\boldsymbol{Q}\Delta t} = e^{\boldsymbol{Q}\Delta t} \tag{2.8}$$

where we used property (d), which says that $\boldsymbol{P}(0) = \boldsymbol{I}$, the identity matrix.
Finally, given the notion of transition rates, one can prove that the waiting time in state $i \in \mathcal{J} = \{1, \dots, J\}$ is exponentially distributed with parameter $-q_{ii}$. On the other hand, given that the Markov chain exits the state i, the event of transition to state j occurs with probability $q_{ij}/(-q_{ii})$.

2.2 Notation and pricing formula

We can now use finite-state continuous time Markov chain to regulate the transition from one market state to another. First, let us fix some notation that we will use throughout the dissertation by referring to [17].
Define $\mathcal{J} = \{1, 2, \dots, J\}$ as the state space, being every state characterized by its own parameters $(r_i, \delta_i, \dots)$ with $i \in \mathcal{J}$, and let $\alpha_{t\{t\in[0,T]\}}$ be the Markov chain taking values $i \in \mathcal{J}$ and $\boldsymbol{Q} := \{q_{ij}, 1 \leq i, j \leq J\} \in \mathbb{R}^{J\times J}$ its associated intensity matrix.
Let us take a regime-switching exponential Lévy model $S_t = S_0 e^{X_t}$, $S_0 > 0$, where X_t is a Lévy process in which the evolution of asset prices is influenced by the Markov chain α_t. If α_t is in state i, then X_t^i is a Lévy process with characteristic triplet $(\mu_i, \sigma_i^2, \nu(i, z))$ and characteristic exponent $\psi_i(u)$. Hence, all the results listed in Chapter 1 continue to be true, provided of course to adapt the notation to this new framework. For instance, the Lévy-Khinchin formula for $\psi_i(u)$ introduced in Theorem 1.3.1 becomes

$$\psi_i(u) = \mathtt{i}\mu_i u - \frac{1}{2}\sigma_i^2 u^2 + \int_{\mathbb{R}} \left(e^{\mathtt{i}uz} - 1 - \mathtt{i}uz\mathbb{1}_{|z|\leq 1}\right)\nu(i, dz) \tag{2.9}$$

However, we have to consider all the states $i \in \mathcal{J}$ simultaneously since we have an only underlying asset which can jump between different regime states not known a priori. This will lead to a matrix form for the characteristic exponent, which will be a function of the single ψ_i, as we will see later on.

In case of a plain vanilla derivative, the risk-neutral valuation formula is then generalized to

$$\begin{aligned} v_i(x,t) &= \mathbb{E}^{\mathbb{Q}}\left[\exp\left(-\int_t^T r(\alpha_s)ds \right)\Phi(X_T)\middle| X_t = x, \alpha_t = i \right] \\ &= \int_{\mathbb{R}} \exp\left(-\int_t^T r(\alpha_s)ds \right) f(y|x,\alpha_t = i)\Phi(y)dy \end{aligned} \tag{2.10}$$

where $\mathbb{Q}$ is the risk-neutral measure, $\Phi(X_T)$ is the payoff function of our financial derivative and $f(y|x,\alpha_t = i)$ is the transition density function of the terminal return X_T conditional that we are in economic state i. The transition density function is often unknown, but it can be recovered from the inverse Fourier integral using the characteristic function which is often available.

Therefore, it is natural to consider Fourier transform methods for option pricing under such models. Among these, there are the method of Fourier space time stepping (FST) (discussed in Chapter 5) and the COS method.

Based on any Lévy process with characteristic triplet $(\mu_i, \sigma_i^2, \nu(i,z))$, the resulting system of PIDEs under the risk-neutral measure for the European option $v(X^i,t)$ is given by

$$\begin{cases} \frac{\partial}{\partial t}v_i(x,t) + \mathcal{L}_i v_i(x,t) - (r_i - q_{ii})v_i(x,t) = 0 \\ v_i(x,T) = \Phi(X_T) \end{cases} \tag{2.11}$$

where $v_i(x,t) \in C^{2,1}$, $i = 1,2,\ldots,J$ denotes the option price at time t, with log-spot equal to x and conditional on the state $\alpha_t = i$ at time t and $\mathcal{L}_i$ is the generator of the regime-switching Lévy process given as

$$\begin{aligned} \mathcal{L}_i v_i(x,t) =& \mu_i \frac{\partial}{\partial x}v_i(x,t) + \frac{1}{2}\sigma_i^2 \frac{\partial^2}{\partial x^2}v_i(x,t) + \sum_{j\neq i} q_{ij}v_j(x,t) \\ &+ \int_{\mathbb{R}\setminus\{0\}} \left(v_i(x+z,t) - v_i(x,t) - z\mathbb{1}_{|z|\leq 1}\frac{\partial}{\partial x}v_i(x,t) \right)\nu(i,dz) \end{aligned} \tag{2.12}$$

We recall that μ_i under the risk-neutral measure can be deduced following the same procedure as in Section 1.6, i.e. introducing another Lévy process $\bar{X}_t^i$ with characteristic triplet $(0, \sigma_i^2, \nu(i,z))$ and characteristic exponent $\bar{\psi}_i(u)$, it follows that

$$\mu_i = r_i - \delta_i - \bar{\psi}_i(-\mathtt{i}) \tag{2.13}$$

By applying the Fourier transform $\mathcal{F}$ to (2.11), [10] and [17] show that the characteristic exponent of the Lévy process can be factored out of $\mathcal{F}$ so that the system of PIDEs reduces to a system of ODEs:

$$\begin{cases} (\partial_t + \psi_i(u) - (r_i - q_{ii}))F_i + \sum_{j\neq i} q_{ij}F_j = 0 \\ F_i(u,T) = \mathcal{F}[\Phi(X_T)](u) \end{cases} \tag{2.14}$$

where $F_i(u,t) = \mathcal{F}[v_i](u,t)$.
In matrix form, equation (2.14) can be displayed as

$$\begin{cases} (\partial_t + \mathbf{\Psi}(u))\bar{F}(u,t) = 0 \\ \bar{F}(u,T) = \mathcal{F}[\Phi(X_T)](u)\bar{1}' \end{cases} \tag{2.15}$$

where $\bar{F}$ is a column vector of F_i's and $\bar{1}' = [1,\ldots,1]' \in \mathbb{R}^{J\times 1}$.
The matrix characteristic exponent $\mathbf{\Psi} \in \mathbb{R}^{J\times J}$ is defined as

$$\mathbf{\Psi}(u) = \begin{bmatrix} \psi_1(u) - r_1 + q_{11} & q_{12} & q_{13} & \cdots \\ q_{21} & \psi_2(u) - r_2 + q_{22} & q_{23} & \cdots \\ \vdots & & \ddots & \\ q_{J1} & q_{J2} & \cdots & \psi_J(u) - r_J + q_{JJ} \end{bmatrix} \tag{2.16}$$

By knowing the final condition, the solution of the system of ODEs in equation (2.15) can easily be found by

$$\bar{F}(u,t) = \boldsymbol{\varphi}_\tau(u)\bar{F}(u,T) \tag{2.17}$$

where $\boldsymbol{\varphi}_\tau(u)$ is the characteristic function of the J regime-switching Lévy process X_t such that $\boldsymbol{\varphi}_\tau(u) = \exp\{\tau\mathbf{\Psi}(u)\}$ is a $J \times J$ matrix with $\tau = T - t$. Option prices can then be obtained by taking a reverse Fourier transform. For European options

$$\bar{v} = \mathcal{F}^{-1}[\boldsymbol{\varphi}_\tau(u)\bar{F}(u,T)] \tag{2.18}$$

where $\bar{v} = [v_1, v_2, \ldots, v_J]'$ is the vector of regime-dependent option values.

Chapter 3

COS method

In this chapter we present the application of the Fourier series expansion of cosines in the context of numerical integration as a more efficient alternative method to those based on FFT, indeed equation (2.18) converges slowly whereas the COS method is proven to have exponential convergence.
The COS method was initially introduced in [7] and generalized in [8], however here we extend it to price options under the regime-switching model, as it is done in [17].

3.1 Fourier Cosine series

Let $f : [0, \pi] \to \mathbb{R}$, its cosine expansion is given by

$$f(\theta) = \sum_{k=0}^{\infty}{}' A_k \cdot \cos(k\theta) \tag{3.1}$$

with

$$A_k = \frac{2}{\pi} \int_0^{\pi} f(\theta) \cos(k\theta) d\theta \tag{3.2}$$

where $\sum'$ indicates that the first term in the summation is weighted by one half.
For functions with a generic domain $f : [a, b] \to \mathbb{R}$, the Fourier-cosine series expansion can easily be obtained via a change of variables:

$$\theta := \frac{x-a}{b-a}\pi \qquad x = \frac{b-a}{\pi}\theta + a \tag{3.3}$$

Thus

$$f(x) = \sum_{k=0}^{\infty}{}' A_k \cdot \cos\left(k\pi \frac{x-a}{b-a}\right) \tag{3.4}$$

$$A_k = \frac{2}{b-a} \int_a^b f(x) \cos\left(k\pi \frac{x-a}{b-a}\right) dx \tag{3.5}$$

3.2 COS method under regime-switching

Here we refer to [17]. Consider a set of prespecified monitoring dates $\{t_0, t_1, \ldots, t_M\}$, where $0 = t_0 < t_1 < \ldots t_m < \cdots < t_M = T$, with $\Delta t = t_m - t_{m-1}$ and T being the final time. Also define

$$\tau^i = \int_{t_{m-1}}^{t_m} \mathbb{1}_{\alpha_s = i}\, ds \tag{3.6}$$

to denote the occupation time of the Markov chain α_t in state i during the interval $[t_{m-1}, t_m]$. Finally τ is the sum of all τ^i and is equal to the time step Δt.
The starting point for pricing European options with the COS method, as with many other pricing methods of the financial world, is the use of the risk-neutral valuation formula, which in this case reads

$$\bar{h}(x, t_{m-1}) = \mathbb{E}^{\mathbb{Q}}[\bar{v}(X_{t_m}, t_m)|X_{t_{m-1}} = x] = \int_{\mathbb{R}} \mathbf{F}(y|x, \tau)\bar{v}(y, t_m)dy \tag{3.7}$$

where $\bar{v}$ and $\bar{h}$ denote the vector of regime dependent option values and continuation values, respectively, and x and y are the state variables at time t_{m-1} and t_m.
As mentioned before, the matrix transition density function $\mathbf{F}$, such that $\mathbf{F}(y|x, \tau) = diag\big(\bar{f}(y|x, \tau)\big)$ with $\bar{f}(y|x, \tau) \in \mathbb{R}^{J\times 1}$, usually is not known, on the other hand the characteristic function is often available. It is why we exploit the relation between transition density and characteristic function, which is given by

$$\bar{f}(y|x, \tau) = \frac{1}{2\pi} \int_{\mathbb{R}} \boldsymbol{\varphi}_\tau(u|x)e^{-\mathtt{i}uy}\bar{1}'du \tag{3.8}$$

The Fourier cosine series of formula (3.4) can be used only for functions with finite support, thus exploiting the fact that the density rapidly decays to zero as $y \to \pm\infty$, we can truncate the integration range without losing significant accuracy to $[a, b] \in \mathbb{R}$ obtaining

$$\bar{h}_1(x, t_{m-1}) = \int_a^b \mathbf{F}(y|x, \tau)\bar{v}(y, t_m)dy \tag{3.9}$$

Adapting the Fourier cosine series expansion of formula (3.4), we have

$$\bar{h}_1(x, t_{m-1}) = \int_a^b {\sum_{k=0}^{\infty}}' \mathbf{A}_k(x) \cos\left(k\pi\frac{y-a}{b-a}\right)\bar{v}(y, t_m)dy \tag{3.10}$$

where

$$\mathbf{A}_k(x) = \frac{2}{b-a}\int_a^b \mathbf{F}(y|x, \tau) \cos\left(k\pi\frac{y-a}{b-a}\right)dy \tag{3.11}$$

is the diagonal matrix of the series coefficient of the Fourier cosine expansion of the transition density function and belongs to $\mathbb{R}^{J\times J}$.
Then, we interchange summation and integration and define

$$V_k^i(t_m) := \frac{2}{b-a}\int_a^b \cos\left(k\pi\frac{y-a}{b-a}\right)v^i(y, t_m)dy \tag{3.12}$$

$\bar{V}_k(t_m) = [V_k^1(t_m), V_k^2(t_m), \ldots, V_k^J(t_m)]'$.
Thus, also truncating the infinite series, we obtain

$$\bar{h}_2(x, t_{m-1}) = \frac{1}{2}(b-a)\sum_{k=0}^{N-1}{}' \mathbf{A}_k(x)\bar{V}_k(t_m) \tag{3.13}$$

where $\bar{h}_2$ is an approximation to $\bar{h}_1$ which converges exponentially or algebraically with increasing N as proven in the error analysis section of [17].
Additionally, we have that $\mathbf{A}_k(x)$ in equation (3.11) can be approximated by $\mathbf{D}_k(x)$, i.e.

$$\mathbf{D}_k(x) = \frac{2}{b-a}\,\mathrm{Re}\left(e^{-\mathtt{i}k\pi\frac{a}{b-a}}\boldsymbol{\varphi}_\tau\Big(\frac{k\pi}{b-a}\Big|x\Big)\right) \tag{3.14}$$

Indeed

$$\begin{aligned}
\mathbf{D}_k(x) &= \frac{2}{b-a}\,\mathrm{Re}\left(e^{-\mathtt{i}k\pi\frac{a}{b-a}}\boldsymbol{\varphi}_\tau\Big(\frac{k\pi}{b-a}\Big|x\Big)\right)\\
&= \frac{2}{b-a}\,\mathrm{Re}\left(e^{-\mathtt{i}k\pi\frac{a}{b-a}}\int_{\mathbb{R}}\mathbf{F}(y|x,\tau)e^{\mathtt{i}y\frac{k\pi}{b-a}}dy\right)\\
&= \frac{2}{b-a}\,\mathrm{Re}\left(\int_{\mathbb{R}}\mathbf{F}(y|x,\tau)e^{\mathtt{i}k\pi\frac{y-a}{b-a}}dy\right)\\
&= \frac{2}{b-a}\,\mathrm{Re}\left(\int_{\mathbb{R}}\mathbf{F}(y|x,\tau)\Big(\cos\Big(k\pi\frac{y-a}{b-a}\Big)+\mathtt{i}\sin\Big(k\pi\frac{y-a}{b-a}\Big)\Big)dy\right)\\
&= \frac{2}{b-a}\int_{\mathbb{R}}\mathbf{F}(y|x,\tau)\cos\Big(k\pi\frac{y-a}{b-a}\Big)dy\\
&\approx \frac{2}{b-a}\int_a^b\mathbf{F}(y|x,\tau)\cos\Big(k\pi\frac{y-a}{b-a}\Big)dy = \mathbf{A}_k(x)
\end{aligned}$$

The characteristic function for Lévy processes $\boldsymbol{\varphi}_\tau(u|x)$ can be represented as

$$\boldsymbol{\varphi}_\tau(u|x) := \boldsymbol{\varphi}(u)e^{\mathtt{i}ux}, \quad u \in \mathbb{R} \tag{3.15}$$

such that the approximation in equation (3.13) can be simplified to

$$\bar{h}_3(x, t_{m-1}) = \sum_{k=0}^{N-1}{}'\,\mathrm{Re}\left(e^{\mathtt{i}k\pi\frac{x-a}{b-a}}\boldsymbol{\varphi}\Big(\frac{k\pi}{b-a}\Big)\right)\bar{V}_k(t_m) \tag{3.16}$$

Define

$$\bar{\Lambda}_k(t_m) := \boldsymbol{\varphi}\Big(\frac{k\pi}{b-a}\Big)\bar{V}_k(t_m) \tag{3.17}$$

Then, the COS formula for the European option in state $i \in J$ is given by

$$v^i(x, t_0) = \sum_{k=0}^{N-1}{}'\,\mathrm{Re}\left(e^{\mathtt{i}k\pi\frac{x-a}{b-a}}\Lambda_k^i(t_M)\right) \tag{3.18}$$

where $\Lambda_k^i(t_M)$ is the ith element of $\bar{\Lambda}_k(t_M)$ in equation (3.17) for $i = 1, 2, \ldots, J$.

3.3 Coefficients $\bar{V}_k$ for Plain Vanilla Options

In order to use equation (3.18) to price options, first we need to compute $\bar{V}_k$. Let us denote the log-asset prices by $x := \log(S_0/K)$ and $y := \log(S_T/K)$, with S_t the underlying price at time t and K the strike price.
The payoff of European options in terms of log-asset prices is given by

$$\Phi(y, t_M) = v^i(y, t_M) = [\alpha \cdot K(e^y - 1)]^+ \quad \text{with} \quad \alpha = \begin{cases} 1 & \text{for a call} \\ -1 & \text{for a put} \end{cases} \tag{3.19}$$

Let us define the functions

$$\begin{aligned} \chi_k(c,d) :&= \int_c^d e^y \cos\left(k\pi \frac{y-a}{b-a}\right) dy \\ &= \frac{1}{1+\left(\frac{k\pi}{b-a}\right)^2} \left[\cos\left(k\pi \frac{d-a}{b-a}\right) e^d - \cos\left(k\pi \frac{c-a}{b-a}\right) e^c \right. \\ &\left. + \frac{k\pi}{b-a} \sin\left(k\pi \frac{d-a}{b-a}\right) e^d - \frac{k\pi}{b-a} \sin\left(k\pi \frac{c-a}{b-a}\right) e^c \right] \end{aligned} \tag{3.20}$$

$$\begin{aligned} \xi_k(c,d) :&= \int_c^d \cos\left(k\pi \frac{y-a}{b-a}\right) dy \\ &= \begin{cases} \left[\sin\left(k\pi \frac{d-a}{b-a}\right) - \sin\left(k\pi \frac{c-a}{b-a}\right) \right] \frac{b-a}{k\pi} & k \neq 0 \\ d - c & k = 0 \end{cases} \end{aligned} \tag{3.21}$$

(for the complete proof refer to [7]).
Thus, using these functions and (3.19) into (3.12), we obtain

$$\begin{aligned} V_k^i(t_M) &= \frac{2}{b-a} \int_a^b \cos\left(k\pi \frac{y-a}{b-a}\right) [\alpha \cdot K(e^y - 1)]^+ dy \\ &= \begin{cases} \frac{2}{b-a} K[\chi_k(0,b) - \xi_k(0,b)] & \text{for a call} \\ \frac{2}{b-a} K[\xi_k(a,0) - \chi_k(a,0)] & \text{for a put} \end{cases} \end{aligned} \tag{3.22}$$

3.4 Truncation range

The choice of the two bounds a and b for the integration range is very important. It is intuitive that choosing an interval too small or too large can lead to significant errors. Extending to the regime-switching framework what is reported in [7], we define a_i and b_i, $i \in \mathcal{J}$, such that

$$\begin{aligned} a_i &= x_0 + c_1^i - L\sqrt{c_2^i + \sqrt{c_4^i}} \\ b_i &= x_0 + c_1^i + L\sqrt{c_2^i + \sqrt{c_4^i}} \end{aligned} \tag{3.23}$$

$$a = \min_i\{a_i\} \quad \text{and} \quad b = \max_i\{b_i\} \tag{3.24}$$

where $x_0 = \log(S_0/K)$, L is a constant parameter (typically $L = 10$) and c_n^i denotes the nth cumulant of X_t^i. The cumulants of a general Lévy process X^i can be obtained by adapting the equations (1.9) and (1.10) to the framework of regime-switching Lévy models, i.e. using either

$$c_n^i(X) = \frac{1}{\mathtt{i}^n}\frac{\partial^n(t\psi_i(u))}{\partial u^n}\bigg|_{u=0} \tag{3.25}$$

or

$$c_n^i(X) = \frac{\partial^n f^i(u)}{\partial u^n}\bigg|_{u=0} \tag{3.26}$$

where $f^i(u) = \log(\varphi_i(-\mathtt{i}u))$.

We report in Appendix A the cumulants of the Lévy processes presented in Section 1.3. However, for the stochastic time-changed Lévy processes, it may be more difficult to find the derivatives analytically. Formulas reveal to be complicated and tediously long. Hence, we calculate them numerically using finite differences. Therefore, we need the 1^{st}, 2^{nd} and 4^{th} finite differences denoted by $d_n f^i(\cdot)$, which for some small $h > 0$ are derived from (3.26) as

$$d_1 f^i(u) = \frac{f^i(u+h) - f^i(u-h)}{2h}$$
$$d_2 f^i(u) = \frac{f^i(u+h) - 2f^i(u) + f^i(u-h)}{h^2}$$

$$\begin{aligned} d_4 f^i(u) =& \frac{f^i(u+3h) - 2f^i(u+2h) + 4f^i(u)}{4h^4} \\ & \frac{-f^i(u+h) - f^i(u-h) - 2f^i(u-2h) + f^i(u-3h)}{4h^4} \end{aligned}$$

Then the cumulants c_1^i, c_2^i and c_4^i are $d_1 f^i(0)$, $d_2 f^i(0)$ and $d_4 f^i(0)$, respectively.

3.5 Matrix exponentiation

In order to evaluate (3.17) for all $k = 0, \ldots, N-1$, it is convenient to consider the matrix formulation

$$\bar{\Lambda}(t_m) = \boldsymbol{\varphi}_\tau(\bar{u})\bar{V}(t_m) = \exp\{\tau\boldsymbol{\Psi}(\bar{u})\}\bar{V}(t_m) \tag{3.27}$$

where $\bar{u} = [u_0, u_1, \ldots, u_{N-1}]'$ with $u_k = \frac{k\pi}{b-a}$. Here $\boldsymbol{\varphi}_\tau(\bar{u})$ and $\boldsymbol{\Psi}(\bar{u})$ are $JN \times JN$ matrices and q_{ij} in equation (2.16) is multiplied by the identity matrix on $\mathbb{R}^{N\times N}$.

Moreover, $\bar{\Lambda}(t_m)$ is reordered such that

$$\bar{\Lambda}(t_m) = [\Lambda_0^1(t_m), \Lambda_1^1(t_m), \dots, \Lambda_{N-1}^1(t_m), \Lambda_0^2(t_m), \dots, \Lambda_{N-1}^J(t_m)]'$$
$$\bar{V}(t_m) = [V_0^1(t_m), V_1^1(t_m), \dots, V_{N-1}^1(t_m), V_0^2(t_m), \dots, V_{N-1}^J(t_m)]'$$

Evaluating this matrix exponential is not an easy task. Indeed the `expm` function in MATLAB® which is based on Padé approximants has $O(N^3)$ complexity. Hence, this procedure for high values of N can become computationally demanding. It is why [17] propose alternative methods: an exact formula for the two-state case and a very efficient approximation for generic $J > 2$ states.
However, in this dissertation, we will focus only on the two-state case since this already provides good fit to market data and the increasing of J leads to a larger number of parameters involved which causes instability, imprecisions, making the additional computational load not worth the effort. Nevertheless, for the sake of completeness of analysis, we use a notation for generic J states.

3.5.1 Two-state case

When we have $J = 2$ states we can compute the matrix exponential with an exact formula. Indeed we have

$$\boldsymbol{\Psi}(u_k) = \begin{bmatrix} \tilde{\psi}_1(u_k) & q_{12} \\ q_{21} & \tilde{\psi}_2(u_k) \end{bmatrix} \tag{3.28}$$

where $\tilde{\psi}_i(u_k) = \psi_i(u_k) - r_i + q_{ii}$, $i = 1, 2$. Its eigenvalues are given by

$$s_{1,2}(u_k) = \frac{1}{2}\left(\tilde{\psi}_1(u_k) + \tilde{\psi}_2(u_k) \pm \sqrt{\tilde{\psi}_1(u_k)^2 - 2\tilde{\psi}_1(u_k)\tilde{\psi}_2(u_k) + \tilde{\psi}_2(u_k)^2 + 4q_{12}q_{21}}\right)$$

Then

$$\boldsymbol{\varphi}_\tau(u_k) = e^{\tau\boldsymbol{\Psi}(u_k)} = e^{s_2(u_k)\tau}\boldsymbol{I} + \frac{e^{s_2(u_k)\tau} - e^{s_1(u_k)\tau}}{s_2(u_k) - s_1(u_k)}(\boldsymbol{\Psi}(u_k) - s_2(u_k)\boldsymbol{I}) \tag{3.29}$$

with $\boldsymbol{I}$ identity matrix in $\mathbb{R}^{2\times 2}$.
This will hold for all values of u_k such that $\boldsymbol{\varphi}_\tau(\bar{u})$ in equation (3.27) becomes a block diagonal matrix, i.e.

$$\boldsymbol{\varphi}_\tau(\bar{u}) = \begin{bmatrix} \boldsymbol{\varphi}_\tau(\bar{u})_{1,1} & \boldsymbol{\varphi}_\tau(\bar{u})_{1,2} \\ \boldsymbol{\varphi}_\tau(\bar{u})_{2,1} & \boldsymbol{\varphi}_\tau(\bar{u})_{2,2} \end{bmatrix} \tag{3.30}$$

where $\boldsymbol{\varphi}_\tau(\bar{u})_{i,j}$ has non-zero elements $[\boldsymbol{\varphi}_\tau(u_0)_{i,j}, \boldsymbol{\varphi}_\tau(u_1)_{i,j}, \dots, \boldsymbol{\varphi}_\tau(u_{N-1})_{i,j}]'$ along its main diagonal only.

3.6 Put-Call parity

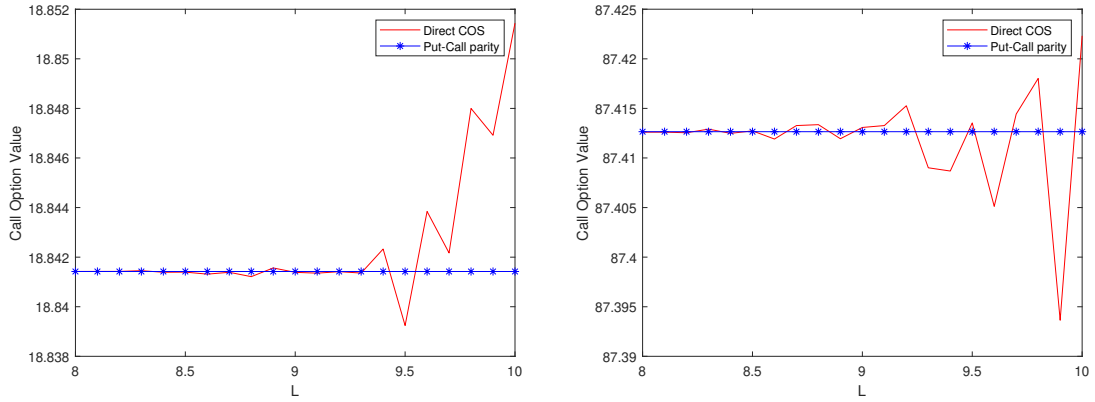
As stated in [7], when pricing call options with the COS method, the solution's accuracy exhibits sensitivity regarding the size of the truncated domain, i.e. the choice of parameter L in (3.23). This holds specifically for call options under certain Lévy jump processes with fat-tailed distributions. Indeed their unbounded payoff may introduce a significant cancellation error for large values of L.
In detail, assuming $K(e^b - 1) \geq 0$, the truncation error in the COS method for a call option is given by

$$\begin{aligned}
\bar{\epsilon} &= \bar{h}(x,T) - \bar{h}_1(x,T) \\
&= \int_{\mathbb{R}\setminus[a,b]} \mathbf{F}(y|x,\tau)\bar{v}(y,T)dy \\
&\geq \int_b^{+\infty} \mathbf{F}(y|x,\tau)\bar{v}(y,T)dy \\
&= \int_b^{+\infty} \mathbf{F}(y|x,\tau)K(e^y - 1)\bar{1}'dy \\
&\geq K(e^b - 1)\int_b^{+\infty} \mathbf{F}(y|x,\tau)\bar{1}'dy
\end{aligned}$$

Thus it is intuitive that the larger $[a,b]$, the larger $K(e^b - 1)$, which grows exponentially with respect to b. Hence, although the value of $\int_b^{+\infty} \mathbf{F}(y|x,\tau)\bar{1}'dy$ decreases as the integration domain increases, the total error might increase.
Since put options are not affected by this, for pricing call options one can exploit the well known *Put-Call parity* which reads

$$v^{\text{call}}(x,t_0) = v^{\text{put}}(x,t_0) + S_0 e^{-\delta T} - Ke^{-rT} \tag{3.31}$$

Let us stress that, intuitively, this relation can be used even in regime-switching models only if r and δ remain constant in the different regimes. To see in practice what we just described, we propose the example presented in [21] adapted to the regime-switching framework.
We consider a two-state CGMY model, with Y values close to 2, so as to have a distribution with very heavy tails. The parameters are: $S_0 = 100,\ K = 100,\ T = 0.1,\ r = 0.1,\ \delta = 0.05,\ C = 1,\ G = 5,\ M = 5,\ Y = [1.5, 1.98]',\ q_{12} = 0.5974,\ q_{21} = 1e-5$. To price numerically the call option, we implement both the direct COS method and the one taking into account the put-call parity. By plotting the results against different values of $L \in [8, 10]$ we obtain the following figures.



(a) State 1 (b) State 2

Figure 3.1: Comparison of European call option values, directly obtained by the COS method, with those obtained by the put-call parity

We can clearly see that the call prices obtained by the put-call parity do not deviate from the reference solutions in both states, for all integration ranges. The parity leads to robust formulas for pricing European call options by the COS method, and hence, whenever we encounter r and δ constant, we will apply this relation.

Chapter 4

Calibration

The calibration problem is of considerable interest in computational finance and beyond. In the financial world, it consists in estimating the risk-neutral parameters of a certain model which give the model prices consistent with the market prices. Thus, it allows you to compare the different pricing models we have seen so far in order to choose the one which best fits the market data.
Therefore, a calibration process requires an objective function to be minimised. In our case, this is given by the so called *Root Mean Square Error*

$$\min_{\theta\in\mathcal{A}} RMSE(\theta) = \min_{\theta\in\mathcal{A}} \sqrt{\sum_{\text{options}} \frac{(\text{Market price} - \text{Model price}(\theta))^2}{\text{number of options}}} \tag{4.1}$$

where θ is the vector of model parameters and $\mathcal{A}$ is its admissibility domain.
To implement the procedure described, we use the `lsqnonlin` MATLAB® function which solves nonlinear least squares problems. However, this function finds a local solution, not a global one, and thus the choice of the starting values for the parameters is really crucial and must be prudent.
To compute the option prices we use the COS method with $N = 2^{12}$. Moreover, we remark the fact that, since our datasets contain call option prices, due to the possible convergence problems highlighted in Section 3.6, we made use of the put-call parity relation.

4.1 Calibration results

First, we set up a calibration procedure considering as dataset the table of call options on the S&P 500 index (taken from [16]) which is reported in Appendix B. Below we report the obtained calibration results in descending order of RMSE.

Table 4.1: Calibration Lévy processes

Model	Parameters	RMSE
BS	$\sigma = 0.1806$	6.7382
VG	$\sigma = 0.1791,\ \nu = 0.7206,\ \theta = -0.1370$	3.5574
Meixner	$\alpha = 0.3903,\ \beta = -1.4913,\ \eta = 0.3558$	3.2366
NIG	$\omega = 6.3199,\ \beta = -3.9686,\ \gamma = 0.1638$	3.0983
Merton	$\sigma = 0.1383,\ \tilde{\mu} = -9.9977,\ \tilde{\sigma} = 0.0767, \lambda = 0.0351$	2.1669
CGMY	$C = 0.0278,\ G = 0.0027,\ M = 2.1152,\ Y = 0.7871$	2.0324
Kou	$\sigma = 5e-7,\ p = 0.9440,\ \lambda_+ = 11.3157,\ \lambda_- = 0.0178,\ \lambda = 0.8775$	1.7831
Heston	$V_0 = 0.0225,\ \theta = 0.1339,\ \kappa = 0.3705,\ \epsilon = 0.3044,\ \rho = -0.7611$	0.8106

Table 4.2: Calibration time-changed Lévy processes

Model	Parameters	RMSE
VG-CIR	$\sigma = 0.1613,\ \nu = 0.0809,\ \theta = -0.1251,$ $k = 0.6198,\ \eta = 1.5649,\ \lambda = 2.0359$	0.5057
Meixner-CIR	$\alpha = 0.1225,\ \beta = -0.5906,\ \eta = 3.3813,$ $k = 0.5747,\ \eta_{CIR} = 1.5736,\ \lambda = 1.9509$	0.4929
NIG-CIR	$\omega = 18.6177,\ \beta = -4.8502,\ \gamma = 0.4709,$ $k = 0.5461,\ \eta = 1.5738,\ \lambda = 1.8866$	0.4848
VG-GOU	$\sigma = 0.1565,\ \nu = 0.0857,\ \theta = -0.2099,$ $\lambda = 1.2553,\ a_1 = 0.5860,\ a_2 = 0.6376$	0.4464
CGMY-CIR	$C = 0.0071,\ G = 0.0598,\ M = 11.4255,\ Y = 1.6826,$ $k = 0.3914,\ \eta = 1.3985,\ \lambda = 1.3589$	0.4367
Meixner-GOU	$\alpha = 0.1107,\ \beta = -0.9843,\ \eta = 3.6324,$ $\lambda = 1.1588,\ a_1 = 0.5880,\ a_2 = 0.6498$	0.4221
NIG-GOU	$\omega = 22.4157,\ \beta = -9.5143,\ \gamma = 0.4848,$ $\lambda = 1.0760,\ a_1 = 0.5919,\ a_2 = 0.6790$	0.4053
CGMY-GOU	$C = 0.0257,\ G = 2.6531,\ M = 32.7961,\ Y = 1.4708,$ $\lambda = 0.8943,\ a_1 = 0.6190,\ a_2 = 0.9747$	0.3649

Table 4.3: Calibration regime-switching Lévy processes

Model	Parameters	RMSE
2RS BS	$\sigma_1 = 0.1486\ \sigma_2 = 0.2115,$ $q_{12} = 1.512,\ q_{21} = 1e-5$	5.8866
2RSMM VG & NIG	$\sigma_1 = 0.1522,\ \nu_1 = 0.0605,\ \theta_1 = -0.2685,$ $\omega_2 = 6.8699,\ \beta_2 = -6.6748,\ \gamma_2 = 0.0662,$ $q_{12} = 0.7837,\ q_{21} = 1e-5$	0.3856

Model	Parameters	RMSE
2RS VG	$\sigma_1 = 0.1541\ \nu_1 = 0.0614,\ \theta_1 = -0.2544,$	0.3758
	$\sigma_2 = 0.2299,\ \nu_2 = 5.1315,\ \theta_2 = -0.2061,$	
	$q_{12} = 0.7289,\ q_{21} = 1e-5$	
2RS Meixner	$\alpha_1 = 0.0905,\ \beta_1 = -1.0184,\ \eta_1 = 5.1949,$	0.3651
	$\alpha_2 = 0.5751,\ \beta_2 = -2.7545,\ \eta_2 = 0.0769,$	
	$q_{12} = 0.7082,\ q_{21} = 1e-5$	
2RSMM	$C_1 = 0.0762,\ G_1 = 7.2908,\ M_1 = 40.6165,$	0.3492
CGMY & Meixner	$Y_1 = 1.2709,\ \alpha_2 = 0.6198,\ \beta_2 = -2.8190,$	
	$\eta_2 = 0.0651, q_{12} = 0.6037,\ q_{21} = 1e-5$	
2RSMM	$C_1 = 0.0686,\ G_1 = 6.9063,\ M_1 = 38.2677,$	0.3460
CGMY & VG	$Y_1 = 1.2911,\ \sigma_2 = 0.2577,\ \nu_2 = 6.7102,$	
	$\theta_2 = -0.2361,\ q_{12} = 0.5752,\ q_{21} = 1e-5$	
2RS NIG	$\omega_1 = 26.3844,\ \beta_1 = -12.3251,\ \gamma_1 = 0.5226,$	0.3437
	$\omega_2 = 6.0287,\ \beta_2 = -6.0127,\ \gamma_2 = 0.0499,$	
	$q_{12} = 0.5611,\ q_{21} = 0.0512$	
2RS CGMY	$C_1 = 0.9026,\ G_1 = 14.4446\ M_1 = 37.9758,$	0.3389
	$C_2 = 0.0541,\ G_2 = 0.0127,\ M_2 = 10.5510,$	
	$Y = 0.6591,\ q_{12} = 0.5974,\ q_{21} = 1e-5$	
2RS Kou	$\sigma_1 = 0.1240,\ p_1 = 3e-13,\ \lambda_{+,1} = 20.5976,$	0.3254
	$\lambda_{-,1} = 20.0841,\ \lambda_1 = 2.7386,\ \sigma_2 = 0.0412,$	
	$p_2 = 1e-5,\ \lambda_{+,2} = 1.0010,\ \lambda_{-,2} = 1.5197,$	
	$\lambda_2 = 0.3333,\ q_{12} = 0.5416, q_{21} = 0.0160$	
2RS Merton	$\sigma_1 = 0.1301,\ \tilde{\mu}_1 = -0.1521,\ \tilde{\sigma}_1 = 3e-8,$	0.3133
	$\lambda_1 = 0.5060,\ \sigma_2 = 0.0282,\ \tilde{\mu}_2 = -0.9951,$	
	$\tilde{\sigma}_2 = 0.6580,\ \lambda_2 = 0.2351, q_{12} = 0.4646,\ q_{21} = 0.2036$	

Table 4.4: Calibration regime-switching time-changed Lévy processes

Model	Parameters	RMSE
2RSMM VG-GOU	$\sigma_1 = 0.1610,\ \nu_1 = 0.0599,\ \theta_1 = -0.1560,$	0.3066
& VG-CIR	$\lambda_1 = 0.4844,\ a_{1,1} = 1.2459,\ a_{2,1} = 0.4058,$	
	$\sigma_2 = 0.0790,\ \nu_2 = 0.2969,\ \theta_2 = -0.2484,$	
	$k_2 = 0.5854,\ \eta_2 = 1e-5,\ \lambda_2 = 0.7392,$	
	$q_{12} = 2.4981,\ q_{21} = 1.5414$	
2RS VG-CIR	$\sigma_1 = 0.1293,\ \nu_1 = 0.0385,\ \theta_1 = -0.2786,$	0.2556
	$k_1 = 0.3073,\ \eta_1 = 2.5613,\ \lambda_1 = 1.3381$	
	$\sigma_2 = 0.2837,\ \nu_2 = 0.0010,\ \theta_2 = -0.2537,$	

Model	Parameters	RMSE
	$k_2 = 5.9938,\ \eta_2 = 0.1079,\ \lambda_2 = 4.4056$	
	$q_{12} = 3.1516,\ q_{21} = 1.5949$	
2RS VG-GOU	$\sigma_1 = 0.1429,\ \nu_1 = 0.0001,\ \theta_1 = -0.0992,$	0.2475
	$\lambda_1 = 0.3358,\ a_{1,1} = 9.6476,\ a_{2,1} = 0.5059$	
	$\sigma_2 = 0.0304,\ \nu_2 = 0.0699,\ \theta_2 = -0.5887,$	
	$\lambda_2 = 1.2465,\ a_{2,1} = 0.2465,\ a_{2,2} = 1.4037$	
	$q_{12} = 9.8471,\ q_{21} = 0.9211$	
2RSMM CGMY-CIR	$C_1 = 0.0165,\ G_1 = 1.3122,\ M_1 = 19.0381,$	0.2472
& Meixner GOU	$Y_1 = 1.5301,\ k_1 = 0.2497,\ \eta_1 = 1.2176,$	
	$\lambda_1 = 1.2889,\ \alpha_2 = 0.0908, \beta_2 = 0.7954,$	
	$\eta_2 = 10.9211,\ \lambda_2 = 5.4519,\ a_{1,2} = 3e-5,$	
	$a_{2,2} = 0.0823,\ q_{12} = 0.6748,\ q_{21} = 1.3954$	
2RSMM CGMY-CIR	$C_1 = 0.0147,\ G_1 = 1.6194,\ M_1 = 20.6193,$	0.2435
& CGMY-GOU	$Y_1 = 1.5567,\ k_1 = 0.0765,\ \eta_1 = 7.7933,$	
	$\lambda_1 = 1.3810,\ C_2 = 0.0928,\ G_2 = 6.6012,$	
	$M_2 = 21.2152,\ Y_2 = 1.3066,\ \lambda_2 = 6.1880,$	
	$a_{1,2} = 0.3100,\ a_{2,2} = 2.1783$	
	$q_{12} = 1.6291,\ q_{21} = 0.8706$	
2RS NIG-CIR	$\omega_1 = 36.4911,\ \beta_1 = -5.6275,\ \gamma_1 = 0.7611,$	0.2381
	$k_1 = 0.3610,\ \eta_1 = 5.9673,\ \lambda_1 = 2.9410$	
	$\omega_2 = 28.6197,\ \beta_2 = -17.1623,\ \gamma_2 = 0.6258,$	
	$k_2 = 3.4886,\ \eta_2 = 0.3190,\ \lambda_2 = 0.1403$	
	$q_{12} = 6.1739,\ q_{21} = 4.4264$	
2RS NIG-GOU	$\omega_1 = 60.5808,\ \beta_1 = 35.8057,\ \gamma_1 = 0.7786,$	0.2162
	$\lambda_1 = 5.5575,\ a_{1,1} = 2e-5,\ a_{2,1} = 0.1058,$	
	$\omega_2 = 60.2640,\ \beta_2 = -47.4782, \gamma_2 = 0.5530,$	
	$\lambda_2 = 1.0145,\ a_{1,2} = 0.5245,\ a_{2,2} = 1.5544,$	
	$q_{12} = 19.0685,\ q_{21} = 4.5030$	

By analyzing these results, we can immediately see that, coming as no surprise, Black-Scholes model leads to larger mispricing than any other model. By introducing Lévy models, we can already observe a remarkable improvement but only incorporating a stochastic volatility, as with the Heston model or even better with the time-changed Lévy models, this improvement becomes really substantial. And we can do even better, by allowing the possibility of switching regime.

Indeed, as outlined in Chapter 2, with the regime-switching models we can have a more complete view of the financial world since they reflect the continuous changes

of market conditions. Moreover, another interesting aspect of regime-switching consists of allowing switches not only in the model parameters, but also in the models as well. And here comes one of the strengths of the COS method: for how it is built, it can easily take into account changes in the models without any additional cost. Finally, as mentioned before, we will consider only the case $J = 2$, but even with such a small J we still get various possible combinations of different models, thus we considered only a set of these and below we report the figures relating to the more significant steps that allowed to arrive to the best calibrated model between the ones considered, which revealed to be the 2RS NIG-GOU[1].

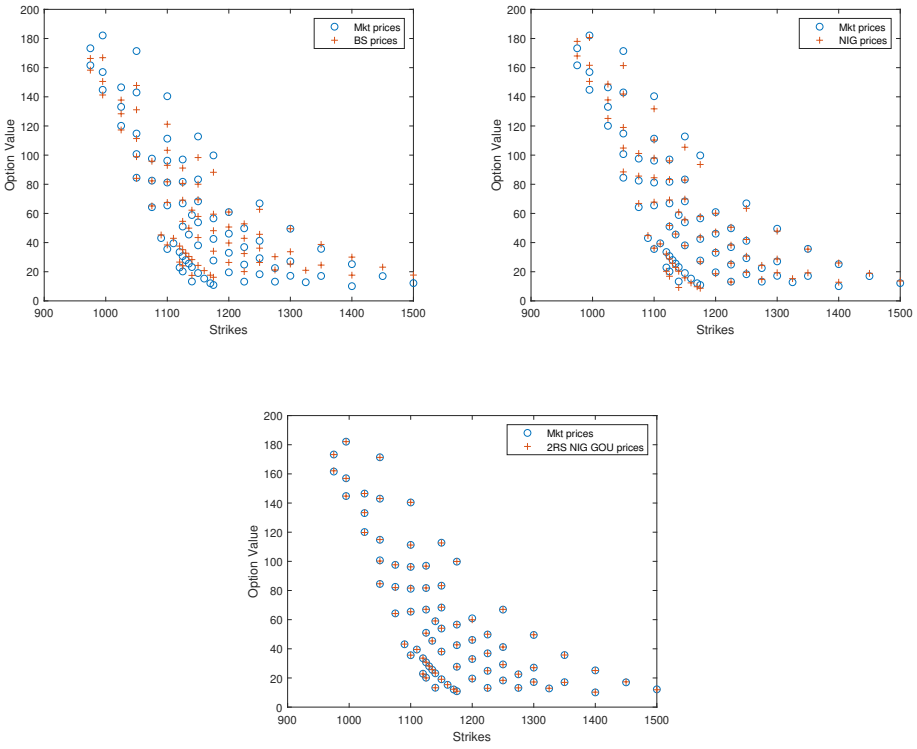


Figure 4.1: Differences between some of the calibrated models

In order to prove the robustness of the result, another calibration procedure should be implemented, but this time taking data during a crisis period. Nowadays, we are in the middle of the *coronavirus* pandemic, which is causing the worst global economic crisis since the *Great Depression*. Hence, we download from Thomson Reuters website a current dataset of European call options on the AAPL stock, considering the most liquid ones (those with a high open interest value) with different

[1]We don't exclude the possibility that a better model can be found since we focused only on a set of all the possible combinations of two different models. However, we can consider ourselves satisfied being the final RMSE equal to 0.2162

maturities. As regards the value of the interest rate r, we point out that r actually depends on maturity, but this is not true in our models. We must therefore download the yield curve and choose a constant value coherent with the maturities taken. Below we report the results of this second calibration.

Table 4.5: Calibration Lévy processes

Model	Parameters	RMSE
BS	$\sigma = 0.3669$	0.3697
NIG	$\omega = 879.4969,\ \beta = -859.8439,\ \gamma = 1.1526$	0.2786
CGMY	$C = 33.2156,\ G = 22.2316,\ M = 1740.9071,\ Y = 0.2679$	0.2777
Meixner	$\alpha = 0.0020,\ \beta = -3.0888,\ \eta = 49.3772$	0.2768
VG	$\sigma = 0.0375,\ \nu = 0.0101,\ \theta = -3.7144$	0.2768
Kou	$\sigma = 0.0281,\ p = 4e-5,\ \lambda_+ = 95.0217,\ \lambda_- = 36.7421,\ \lambda = 94.9603$	0.2755
Merton	$\sigma = 0.0770,\ \tilde{\mu} = -0.0852,\ \tilde{\sigma} = 6e-7, \lambda = 18.6672$	0.2700
Heston	$V_0 = 0.1592,\ \theta = 0.1387,\ \kappa = 1.9015,\ \epsilon = 0.7047,\ \rho = -0.2366$	0.1547

Table 4.6: Calibration time-changed Lévy processes

Model	Parameters	RMSE
CGMY-CIR	$C = 0.0376,\ G = 0.9799,\ M = 11.4161,\ Y = 1.7105,$ $k = 0.6888,\ \eta = 0.3414,\ \lambda = 0.2964$	0.1533
VG-CIR	$\sigma = 0.3721,\ \nu = 0.0209,\ \theta = -1.0998,$ $k = 0.5333,\ \eta = 4e-7,\ \lambda = 0.2657$	0.1506
NIG-CIR	$\omega = 20.2734,\ \beta = -8.1150,\ \gamma = 2.5672,$ $k = 0.5336,\ \eta = 5e-7,\ \lambda = 0.2582$	0.1502
Meixner-CIR	$\alpha = 0.1799,\ \beta = -0.5915,\ \eta = 9.7034,$ $k = 2.7377,\ \eta_{CIR} = 0.6705,\ \lambda = 0.9018$	0.1497
CGMY-GOU	$C = 0.1442,\ G = 3.8267,\ M = 18.2318,\ Y = 1.4640,$ $\lambda = 0.6714,\ a_1 = 1.1595,\ a_2 = 4.4645$	0.1471
Meixner-GOU	$\alpha = 0.1001,\ \beta = -1.1519,\ \eta = 22.8780,$ $\lambda = 0.6533,\ a_1 = 1.1983,\ a_2 = 6.1784$	0.1462
VG-GOU	$\sigma = 0.3604,\ \nu = 0.0159,\ \theta = -1.4357,$ $\lambda = 0.6482,\ a_1 = 1.1541,\ a_2 = 5.8326$	0.1461
NIG-GOU	$\omega = 26.0200,\ \beta = -12.0298,\ \gamma = 2.9702,$ $\lambda = 0.6633,\ a_1 = 1.3111,\ a_2 = 6.8134$	0.1459

Table 4.7: Calibration regime-switching Lévy processes

Model	Parameters	RMSE
2RS BS	$\sigma_1 = 0.3974\ \sigma_2 = 0.3146,$	0.3031
	$q_{12} = 1.7908,\ q_{21} = 1.2040$	
2RS CGMY	$C_1 = 8.0519,\ G_1 = 16.2283\ M_1 = 47.1655,$	0.1310
	$C_2 = 0.0269,\ G_2 = 0.0246,\ M_2 = 20.7043,$	
	$Y = 0.5749,\ q_{12} = 0.5758,\ q_{21} = 1.2042$	
2RS Meixner	$\alpha_1 = 0.0536,\ \beta_1 = -1.7583,\ \eta_1 = 45.9504,$	0.1306
	$\alpha_2 = 0.0513,\ \beta_2 = -3.1306,\ \eta_2 = 0.0177,$	
	$q_{12} = 0.5240,\ q_{21} = 1.1217$	
2RSMM	$C_1 = 0.5281,\ G_1 = 8.9699,\ M_1 = 368.6334,$	0.1304
CGMY & Meixner	$Y_1 = 1.3096,\ \alpha_2 = 0.1877,\ \beta_2 = -3.0833,$	
	$\eta_2 = 0.0199, q_{12} = 0.5588,\ q_{21} = 1.2144$	
2RSMM	$\sigma_1 = 0.2099,\ \nu_1 = 0.0071,\ \theta_1 = -4.0745,$	0.1302
VG & NIG	$\omega_2 = 1.6276,\ \beta_2 = -1.0598,\ \gamma_2 = 0.1562,$	
	$q_{12} = 0.8469,\ q_{21} = 0.4127$	
2RS NIG	$\omega_1 = 1857.3471,\ \beta_1 = -1834.9438,\ \gamma_1 = 1.1215,$	0.1302
	$\omega_2 = 1.6007,\ \beta_2 = -1.6050,\ \gamma_2 = 0.1497,$	
	$q_{12} = 0.8374,\ q_{21} = 0.4620$	
2RS Kou	$\sigma_1 = 0.2752,\ p_1 = 1e-5,\ \lambda_{+,1} = 55.2008,$	0.1300
	$\lambda_{-,1} = 25.1872,\ \lambda_1 = 27.9020,\ \sigma_2 = 0.0710,$	
	$p_2 = 7e-6,\ \lambda_{+,2} = 1.0003,\ \lambda_{-,2} = 3.7271,$	
	$\lambda_2 = 0.7076,\ q_{12} = 0.7903, q_{21} = 0.5965$	
2RSMM	$C_1 = 0.7017,\ G_1 = 9.9899,\ M_1 = 460.4624,$	0.1300
CGMY & VG	$Y_1 = 1.2542,\ \sigma_2 = 0.2975,\ \nu_2 = 5.1175,$	
	$\theta_2 = -0.1136,\ q_{12} = 0.7052,\ q_{21} = 0.8730$	
2RS VG	$\sigma_1 = 0.1440\ \nu_1 = 0.0065,\ \theta_1 = -4.6519,$	0.1297
	$\sigma_2 = 0.3307,\ \nu_2 = 2.1893,\ \theta_2 = -0.1294,$	
	$q_{12} = 0.8487,\ q_{21} = 0.4128$	
2RS Merton	$\sigma_1 = 0.2033,\ \tilde{\mu}_1 = -0.0818,\ \tilde{\sigma}_1 = 0.0097,$	0.1294
	$\lambda_1 = 17.6795,\ \sigma_2 = 2e-5,\ \tilde{\mu}_2 = -0.1782,$	
	$\tilde{\sigma}_2 = 0.3683,\ \lambda_2 = 0.6980,\ q_{12} = 0.7540,\ q_{21} = 0.5086$	

Table 4.8: Calibration regime-switching time-changed Lévy processes

Model	Parameters	RMSE
2RSMM CGMY-CIR	$C_1 = 0.0779,\ G_1 = 1.6966,\ M_1 = 10.9642,$	0.1372
& Meixner GOU	$Y_1 = 1.5576,\ k_1 = 0.3183,\ \eta_1 = 1.3491,$	
	$\lambda_1 = 0.4094,\ \alpha_2 = 0.1141, \beta_2 = 1.2509,$	

Model	Parameters	RMSE
	$\eta_2 = 5.0622,\ \lambda_2 = 1.2483,\ a_{1,2} = 0.0014,$	
	$a_{2,2} = 0.3108,\ q_{12} = 0.8264,\ q_{21} = 0.4472$	
2RS NIG-CIR	$\omega_1 = 17.5165,\ \beta_1 = -1.2189,\ \gamma_1 = 2.7341,$	0.1348
	$k_1 = 1.2828,\ \eta_1 = 3.9711,\ \lambda_1 = 1.6565$	
	$\omega_2 = 25.1103,\ \beta_2 = -17.1673,\ \gamma_2 = 1.5345,$	
	$k_2 = 1.7949,\ \eta_2 = -0.1190,\ \lambda_2 = 0.0288$	
	$q_{12} = 18.3695,\ q_{21} = 6.1298$	
2RSMM VG-GOU	$\sigma_1 = 0.3554,\ \nu_1 = 0.0116,\ \theta_1 = -1.7154,$	0.1270
& VG-CIR	$\lambda_1 = 0.0359,\ a_{1,1} = 2.8973,\ a_{2,1} = 2.9873,$	
	$\sigma_2 = 0.0163,\ \nu_2 = 0.8791,\ \theta_2 = -0.5853,$	
	$k_2 = 5.8691,\ \eta_2 = 4e-5,\ \lambda_2 = 0.1660,$	
	$q_{12} = 0.6789,\ q_{21} = 1.4390$	
2RS NIG-GOU	$\omega_1 = 23.6859,\ \beta_1 = -6.2431,\ \gamma_1 = 3.4914,$	0.1255
	$\lambda_1 = 20.8632,\ a_{1,1} = 3.0541,\ a_{2,1} = 3.3499,$	
	$\omega_2 = 81.3249,\ \beta_2 = -24.6381, \gamma_2 = 0.1219,$	
	$\lambda_2 = 0.0984,\ a_{1,2} = 9.7914,\ a_{2,2} = 0.1896,$	
	$q_{12} = 0.5045,\ q_{21} = 0.6046$	
2RS VG-GOU	$\sigma_1 = 0.1840,\ \nu_1 = 0.0073,\ \theta_1 = -4.0415,$	0.1166
	$\lambda_1 = 0.0421,\ a_{1,1} = 9.8060,\ a_{2,1} = 15.4299$	
	$\sigma_2 = 0.6464,\ \nu_2 = 0.8073,\ \theta_2 = -0.3477,$	
	$\lambda_2 = 4.2687,\ a_{2,1} = 1e-5,\ a_{2,2} = 10.3190$	
	$q_{12} = 1.3212,\ q_{21} = 2.3879$	
2RSMM CGMY-CIR	$C_1 = 1.1804,\ G_1 = 11.2176,\ M_1 = 255.5387,$	0.1144
& CGMY-GOU	$Y_1 = 1.1519,\ k_1 = 6.3486,\ \eta_1 = 0.4934,$	
	$\lambda_1 = 0.5214,\ C_2 = 0.0994,\ G_2 = 75.8750,$	
	$M_2 = 71.1163,\ Y_2 = 1.3590,\ \lambda_2 = 0.0022,$	
	$a_{1,2} = 3.7557,\ a_{2,2} = 0.0942\ q_{12} = 4.7010,\ q_{21} = 3.4896$	
2RS VG-CIR	$\sigma_1 = 0.0009,\ \nu_1 = 0.0060,\ \theta_1 = -5.3094,$	0.1121
	$k_1 = 5.6021,\ \eta_1 = 0.7059,\ \lambda_1 = 0.5938$	
	$\sigma_2 = 0.1631,\ \nu_2 = 0.0350,\ \theta_2 = -0.1399,$	
	$k_2 = 0.3245,\ \eta_2 = 5.0823,\ \lambda_2 = 0.5504$	
	$q_{12} = 3.9621,\ q_{21} = 1.2086$	

By comparing the results of the two calibrations, we can immediately see that in the first case we went from an RMSE equal to 6.7382 in Black-Scholes to an RMSE equal to 0.2162 in the best regime-switching model, while in the second case Black-Scholes and the best model present RMSEs of 0.3697 and 0.1121, respectively. Such

a discrepancy is due to how the error has been defined in formula (4.1): indeed if instead of the RMSE, we used the following relative error

$$Err(\theta) = \frac{||\text{Market prices} - \text{Model prices}(\theta)||_2}{||\text{Model prices}(\theta)||_2} \tag{4.2}$$

we would find that for instance the errors for Black-Scholes in the two calibrations of 6.7382 and 0.3697 would become respectively equal to 0.0897 and 0.0237 and therefore comparable.
Hence, we can draw the same conclusions as before, with the only difference that here the best calibrated model is the 2RS VG-CIR.

Chapter 5

Method's efficiency and comparisons

In this chapter, our goal is to verify the efficiency of the COS method with respect to other well known methods in literature, specifically:

- Lattice method
- Fourier space time stepping (FST) method
- Partial differential equation (PDE) method

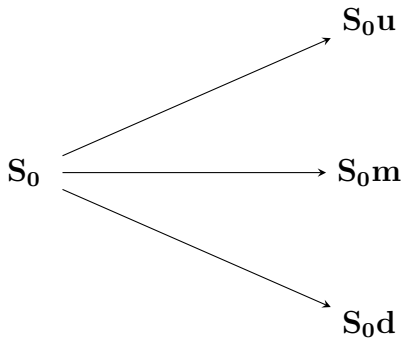
To this end, we review the example presented in [17]. We consider a two-state Black-Scholes model using the same input data and parameters found by the first calibration procedure described in the previous chapter, i.e.

$$r = 0.019,\ \delta = 0.012,\ T = 1.192,\ S_0 = 1124.47,\ K = 1125,$$
$$\sigma_1 = 0.1486,\ \sigma_2 = 0.2115,\ q_{12} = 1.512,\ q_{21} = 1e-5$$

and we price a European call option with the aforementioned methods. All the results are compared with the analytic option values of [14] given as: 92.63798967107 and 105.89517232057 for state 1 and state 2, respectively.
Let us see in detail how these methods work.

5.1 Lattice method

Here we refer to [19].
The lattice method consists in a fast and simple trinomial tree model. With respect to the original CRR binomial tree, an extra branch is introduced so that the stock price is allowed to remain unchanged, or go up or go down by a ratio.
Below we illustrate the situation in the first time interval Δt



with $u = e^{\sigma\sqrt{\Delta t}}$, $m = 1$ and $d = 1/u$.
The trinomial tree we propose is a combining tree with the idea that, if the regime state changes, instead of increasing the number of branches (which include the different rates, dividend yields and volatilities of each regime state), we change the probability, so the tree is still combining.
Taking up the notation defined in Chapter 2, assume that there are J states. $\forall i \in \mathcal{J}$, denote by π_u^i, π_m^i, π_d^i the risk-neutral probabilities corresponding to when the stock price increases, remains the same and decreases, respectively. Then, we have the following set of equations

$$\pi_u^i + \pi_m^i + \pi_d^i = 1 \tag{5.1}$$
$$\pi_u^i u + \pi_m^i + \pi_d^i d = e^{(r_i - \delta_i)\Delta t} \tag{5.2}$$
$$(\pi_u^i + \pi_d^i)\sigma^2 \Delta t = \sigma_i^2 \Delta t \tag{5.3}$$

In order to ensure σ to be greater than all σ_i, define

$$\sigma = \max_{1 \le i \le J} \sigma_i + (\sqrt{1.5} - 1)\bar{\sigma} \tag{5.4}$$

where $\bar{\sigma}$ is the arithmetic mean of σ_i.
Then define $\lambda_i = \sigma/\sigma_i$, so $\lambda_i > 1$ and the set of risk-neutral probabilities can be found in this way

$$\pi_m^i = 1 - \frac{1}{{\lambda_i}^2} \tag{5.5}$$
$$\pi_u^i = \frac{e^{(r_i - \delta_i)\Delta t} - d - \pi_m^i(1-d)}{u - d} \tag{5.6}$$
$$\pi_d^i = \frac{u - e^{(r_i - \delta_i)\Delta t} - \pi_m^i(u-1)}{u - d} \tag{5.7}$$

After the whole lattice is constructed, the main idea of the pricing method is presented here.

Let T be the maturity of the option, N the number of time steps and $\Delta t = T/N$. At time step t, there are $2t+1$ nodes in the lattice. Let $S_{t,n}$ and $V^j_{t,n}$ be the stock price and the option value, respectively, at the nth node at time step t under the jth regime state.
Now we need the transition probability matrix in order to find the price of the derivative at each node by iteration. And from Section 2.1.1 we know that, given the intensity matrix $\boldsymbol{Q}$, the transition probability matrix, denoted by $\boldsymbol{P}$, can be found as in (2.8).
Considering that $V^i_{N,n} = [\alpha \cdot (S_{N,n} - K)]^+$ for all states i where $S_{N,n} = S_0 u^{N+1-n}$ and using the following equation recursively:

$$V^i_{t,n} = e^{-r_i \Delta t}\left[\sum_{j=1}^{J} p_{ij}(\pi^i_u V^j_{t+1,n+2} + \pi^i_m V^j_{t+1,n+1} + \pi^i_d V^j_{t+1,n})\right] \tag{5.8}$$

the price of the option under all regimes can be obtained.

5.2 FST method

Here we follow [10] and [13], adapting the discussion to the regime-switching framework. The main idea behind this method consists in starting from the pricing problem expressed as a partial integro differential equation (PIDE) and then pass to Fourier space, where the problem can be reduced to solve ordinary differential equations (ODE).
To numerically solve (2.18) we consider a grid for the set $\Omega = [0,T] \times [x_{min}, x_{max}]$ which is $\{t_m | m = 0, \dots, M\} \times \{x_n | n = 0, \dots, N-1\}$ where $t_m = m\Delta t$ with $\Delta t = T/M$ and $x_n = x_{min} + n\Delta x$ with $\Delta x = (x_{max} - x_{min})/(N-1)$.
We consider also a grid for the frequency domain: $w_n = -w_{max} + n\Delta w$, $n = 0, 1, \dots, N$ with $w_{max} = \pi/\Delta x$ and $\Delta w = 2w_{max}/N$.
Selecting the grid is no easy task. We choose the boundary points such that first, given the interest in pricing in a neighborhood of $x = 0$, $x_{min} = -x_{max}$ and then such that the interval is large enough to contain all information necessary for pricing the option but at the same time small enough to maintain numerical accuracy. The same holds true for choosing w_{max}.
We also take into account the suggestions reported in [10] i.e. $x_{max} \in [2,5]$ works well for diffusion models with low volatility and short maturity, while $x_{max} \in [4,8]$ is preferable for models with a large volatility term or a dominant jump component.
For each regime state $i \in \mathcal{J}$, let us define $v^m_{i,n} := v_i(t_m, x_n)$ and $\hat{v}^m_{i,n} := \hat{v}_i(t_m, w_n)$ the corresponding point transformed in the Fourier space.

Using this notation, we have

$$\begin{aligned}\hat{v}_{i,n}^m = \mathcal{F}[v_i](t_m, w_n) &\approx \sum_{k=0}^{N-1} v_{i,k}^m e^{-\mathtt{i} w_n x_k} \Delta x \\ &= e^{-\mathtt{i} w_n x_{min}} \Delta x \sum_{k=0}^{N-1} v_{i,k}^m e^{-\mathtt{i} nk/N} \\ &= \alpha_n \, \mathrm{FFT}[\bar{v}_i^m](n) \end{aligned} \tag{5.9}$$

with $\alpha_n = e^{-\mathtt{i} w_n x_{min}} \Delta x$ and $\mathrm{FFT}[\bar{v}_i^m](n)$ denoting the nth component of the discrete Fourier transform (DFT) of the vector $\bar{v}_i^m$, where

$$\begin{aligned}\bar{v}^m &= [v_{1,0}^m, v_{1,1}^m, \ldots, v_{1,N}^m, v_{2,0}^m, \ldots, v_{J,N}^m]' \\ \bar{v}_i^m &= [v_{i,0}^m, v_{i,1}^m, \ldots, v_{i,N}^m]'\end{aligned}$$

Thus, we have

$$v_{i,n}^m = \mathrm{FFT}^{-1}[\alpha^{-1} \hat{v}_i^m](n) \tag{5.10}$$

Combining equations (5.9) and (5.10) with equation (2.18), we get

$$\begin{aligned}\bar{v}_i^{m-1} &= \mathrm{FFT}^{-1}[\alpha^{-1} \cdot \hat{v}_i^{m-1}] \\ &= \mathrm{FFT}^{-1}[\alpha^{-1} \cdot (\boldsymbol{\varphi}(w) \hat{v}^m)_i] \\ &= \mathrm{FFT}^{-1}[\cancel{\alpha^{-1}} \cdot \cancel{\alpha} \cdot (\boldsymbol{\varphi}(w) \, \mathrm{FFT}[\bar{v}^m])_i] \end{aligned} \tag{5.11}$$

where

$$(\boldsymbol{\varphi}(w) \, \mathrm{FFT}[\bar{v}^m])_i = \sum_{j=1}^{J} \boldsymbol{\varphi}(w)_{i,j} \, \mathrm{FFT}[\bar{v}_j^m] \tag{5.12}$$

Notice that for European options $M = 1$ so we can simplify the procedure by taking only one time step.

5.3 PDE method

In the numerical example we are considering, we are in Black-Scholes framework. For this reason its pricing formula will be simplified to a PDE, not a PIDE anymore. From equation (2.11), we can recover the PDE equation considering that now, the generator of the regime-switching Lévy process is given as

$$\mathcal{L}_i v_i(x,t) = \mu_i \frac{\partial}{\partial x} v_i(x,t) + \frac{1}{2} \sigma_i^2 \frac{\partial^2}{\partial x^2} v_i(x,t) + \sum_{j \neq i} q_{ij} v_j(x,t) \tag{5.13}$$

Thus, the PDE is given by

$$\begin{cases} \frac{\partial}{\partial t} v_i(x,t) + \mathcal{L}_i v_i(x,t) - (r_i - q_{ii}) v_i(x,t) = 0 \\ v_i(x,T) = \Phi(X_T) \end{cases} \tag{5.14}$$

where $v_i(x,t) \in C^{2,1}$, $i = 1, 2, \ldots, J$ denotes the option price at time t, with log-spot equal to x and conditional on the state $\alpha_t = i$ at time t.
In order to simplify notation, we already consider the two-state case and given that for Black-Scholes $\mu_i = r_i - \delta_i - \frac{1}{2}\sigma_i^2$, $\forall x \in \mathbb{R}$ and $t \in [0,T]$ we obtain 2 PDEs

$$\begin{cases} \frac{\partial}{\partial t}v_1(x,t) + \mu_1 \frac{\partial}{\partial x}v_1(x,t) + \frac{1}{2}\sigma_1^2 \frac{\partial^2}{\partial x^2}v_1(x,t) - (r_1 - q_{11})v_1(x,t) + q_{12}v_2(x,t) = 0 \\ v_1(x,T) = \Phi(X_T) \end{cases} \tag{5.15}$$

$$\begin{cases} \frac{\partial}{\partial t}v_2(x,t) + \mu_2 \frac{\partial}{\partial x}v_2(x,t) + \frac{1}{2}\sigma_2^2 \frac{\partial^2}{\partial x^2}v_2(x,t) - (r_2 - q_{22})v_2(x,t) + q_{21}v_1(x,t) = 0 \\ v_2(x,T) = \Phi(X_T) \end{cases} \tag{5.16}$$

To solve numerically the PDEs, we use the *finite difference method*. In literature other more efficient methods can be found, however they are less intuitive and here we use this method only as a comparison to COS method. The finite difference method is based on the approximation of derivatives. We recall that, given a regular function $g : \mathbb{R} \to \mathbb{R}$, there are three possible approximations of the first order derivative in the stock:

$$g'(x) = \frac{g(x+h) - g(x)}{h} + O(h) \tag{5.17}$$

$$g'(x) = \frac{g(x) - g(x-h)}{h} + O(h) \tag{5.18}$$

$$g'(x) = \frac{g(x+h) - g(x-h)}{h} + O(h^2) \tag{5.19}$$

known as forward, backward and central finite differences, respectively. In order to have a better accuracy, we will use the central ones.
The second order derivative, instead, is approximated by

$$g''(x) = \frac{g(x+h) - 2g(x) + g(x-h)}{h^2} + O(h^2) \tag{5.20}$$

Even as regards the discretization of the derivative over time there are different methods:

- Explicit Euler scheme, which uses the forward first order derivative
- Implicit Euler scheme, which uses the backward first order derivative
- Theta method, which uses a linear combination with coefficient $\theta \in [0,1]$ of forward and backward first order derivatives

The theta method with $\theta = \frac{1}{2}$ is known as *Crank-Nicolson*.
Now we can proceed to the discretization of the problem expressed by the 2 PDEs.

Let us introduce an uniform grid in log-stock and time domain:

$$x_k = x_{min} + k\Delta x, \quad k = 0, \dots, N \quad \text{with} \quad \Delta x = \frac{x_{max} - x_{min}}{N} \tag{5.21}$$

$$t_l = l\Delta t, \quad l = 0, \dots, M \quad \text{with} \quad \Delta t = T/M \tag{5.22}$$

where considering that the underlying dynamics under Black-Scholes in state $i \in \{1, 2\}$ is

$$S_T \sim S_0 e^x = S_0 e^{\left(r_i - \delta_i - \frac{\sigma_i^2}{2}\right)T + \sigma_i \sqrt{T} Z} \quad \text{with} \quad Z \sim N(0, 1) \tag{5.23}$$

and moreover $\mathbb{P}(Z > 6) = \mathbb{P}(Z < 6) \approx 10^{-8}$, we can take as a rule of thumb:

$$\begin{cases} x^i_{min} = \left(r_i - \delta_i - \frac{\sigma_i^2}{2}\right)T - 6\sqrt{T}\sigma \quad i = 1, 2 \\ x^i_{max} = \left(r_i - \delta_i - \frac{\sigma_i^2}{2}\right)T + 6\sqrt{T}\sigma \quad i = 1, 2 \end{cases} \rightarrow \begin{cases} x_{min} = \min(x^1_{min}, x^2_{min}) \\ x_{max} = \max(x^1_{max}, x^2_{max}) \end{cases}$$

Then, by defining $v_i^{k,l} = v_i(x_k, t_l)$ and using the theta method we obtain

$$\begin{cases} \frac{v_1^{k,l+1} - v_1^{k,l}}{\Delta t} + (1-\theta)\left[\mu_1 \frac{v_1^{k+1,l} - v_1^{k-1,l}}{2\Delta x} + \frac{\sigma_1^2}{2}\frac{v_1^{k+1,l} - 2v_1^{k,l} + v_1^{k-1,l}}{\Delta x^2} - (r_1 - q_{11})v_1^{k,l} + q_{12}v_2^{k,l}\right] + \\ \theta\left[\mu_1 \frac{v_1^{k+1,l+1} - v_1^{k-1,l+1}}{2\Delta x} + \frac{\sigma_1^2}{2}\frac{v_1^{k+1,l+1} - 2v_1^{k,l+1} + v_1^{k-1,l+1}}{\Delta x^2} - (r_1 - q_{11})v_1^{k,l+1} + q_{12}v_2^{k,l+1}\right] = 0 \\ v_1^{k,M} = \Phi(X_k) \quad k = 0, \dots, N \\ \text{boundary conditions} \end{cases} \tag{5.24}$$

boundary conditions:
for call: $v_1^{0,l} = 0, \quad v_1^{N,l} = S_0 e^{x_{max}} - Ke^{-r_1(T-t_l)}$
for put: $v_1^{0,l} = Ke^{-r_1(T-t_l)} - S_0 e^{x_{min}}, \quad v_1^{N,l} = 0$

A similar equation is obtained for the state 2.
Let us define for $i, j \in \{1, 2\}, i \neq j$ the following matrices

$$\boldsymbol{M}_i^1 = \begin{bmatrix} \beta_i & \gamma_i & & \\ \alpha_i & \beta_i & \gamma_i & \\ & \alpha_i & \ddots & \ddots \\ & & \ddots & \ddots \end{bmatrix}_{N-1 \times N-1} \qquad \boldsymbol{M}_i^2 = \begin{bmatrix} \hat{\beta}_i & \hat{\gamma}_i & & \\ \hat{\alpha}_i & \hat{\beta}_i & \hat{\gamma}_i & \\ & \hat{\alpha}_i & \ddots & \ddots \\ & & \ddots & \ddots \end{bmatrix}_{N-1 \times N-1}$$

$$\boldsymbol{M}_i^3 = (1-\theta)\begin{bmatrix} q_{ij} & & & \\ & q_{ij} & & \\ & & \ddots & \\ & & & q_{ij} \end{bmatrix}_{N-1 \times N-1} \qquad \boldsymbol{M}_i^4 = -\theta\begin{bmatrix} q_{ij} & & & \\ & q_{ij} & & \\ & & \ddots & \\ & & & q_{ij} \end{bmatrix}_{N-1 \times N-1}$$

$$\bar{V}_i^1 = \begin{bmatrix} v_i^{1,l} \\ v_i^{2,l} \\ \vdots \\ v_i^{N,l} \end{bmatrix}_{N-1\times 1} \qquad \bar{V}_i^2 = \begin{bmatrix} v_i^{1,l+1} \\ v_i^{2,l+1} \\ \vdots \\ v_i^{N,l+1} \end{bmatrix}_{N-1\times 1} \qquad \bar{BC}_i = \begin{bmatrix} -\alpha_i v_i^{0,l} + \hat{\alpha}_i v_i^{0,l+1} \\ 0 \\ \vdots \\ 0 \\ -\gamma_i v_i^{N,l} + \hat{\gamma}_i v_i^{N,l+1} \end{bmatrix}_{N-1\times 1}$$

with

$$\alpha_i = (1-\theta)\left[-\frac{\mu_i}{2\Delta x} + \frac{\sigma^2}{2\Delta x^2}\right] \qquad \hat{\alpha}_i = -\theta\left[-\frac{\mu_i}{2\Delta x} + \frac{\sigma^2}{2\Delta x^2}\right]$$

$$\beta_i = -\frac{1}{\Delta t} + (1-\theta)\left[-\frac{\sigma^2}{\Delta x^2} - (r+q_{ij})\right] \qquad \hat{\beta}_i = -\frac{1}{\Delta t} - \theta\left[-\frac{\sigma^2}{\Delta x^2} - (r+q_{ij})\right]$$

$$\gamma_i = (1-\theta)\left[\frac{\mu_i}{2\Delta x} + \frac{\sigma^2}{2\Delta x^2}\right] \qquad \hat{\gamma}_i = -\theta\left[\frac{\mu_i}{2\Delta x} + \frac{\sigma^2}{2\Delta x^2}\right]$$

Hence, we can solve the coupled PDEs in the following way

$$\begin{cases} \boldsymbol{M}_1^1\bar{V}_1^1 + \boldsymbol{M}_1^3\bar{V}_2^1 = \boldsymbol{M}_1^2\bar{V}_1^2 + \boldsymbol{M}_1^4\bar{V}_2^2 + \bar{BC}_1 \\ \boldsymbol{M}_2^1\bar{V}_2^1 + \boldsymbol{M}_2^3\bar{V}_1^1 = \boldsymbol{M}_2^2\bar{V}_2^2 + \boldsymbol{M}_2^4\bar{V}_1^2 + \bar{BC}_2 \end{cases} \tag{5.25}$$

Then, define

$$\bar{D}_1 := \boldsymbol{M}_1^2\bar{V}_1^2 + \boldsymbol{M}_1^4\bar{V}_2^2 \tag{5.26}$$

$$\bar{D}_2 := \boldsymbol{M}_2^2\bar{V}_2^2 + \boldsymbol{M}_2^4\bar{V}_1^2 \tag{5.27}$$

which at time l are known since we proceed backward in time.
Thus, in order to find the solution for state 1, we multiply the first equation in (5.25) by $\boldsymbol{M}_2^1$ and the second one by $\boldsymbol{M}_1^3$ and subtracting them we have

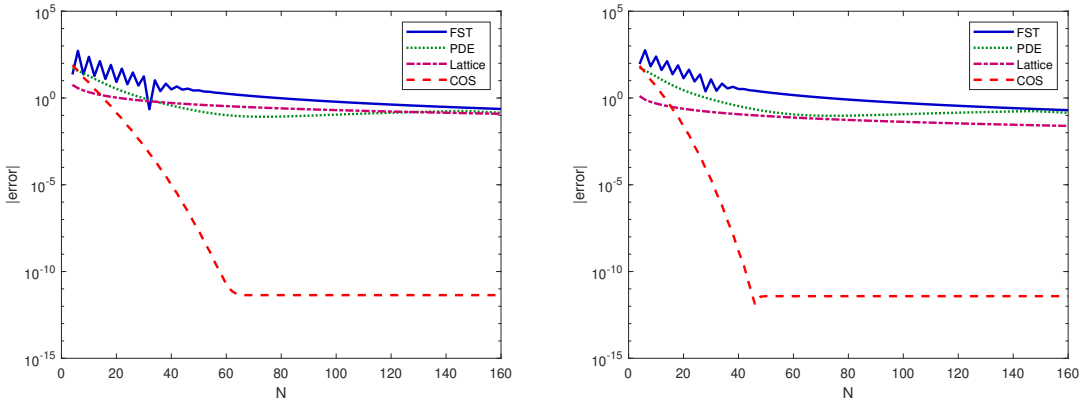
$$\begin{aligned} \boldsymbol{M}_2^1(\boldsymbol{M}_1^1\bar{V}_1^1 + \boldsymbol{M}_1^3\bar{V}_2^1) &= \boldsymbol{M}_2^1(\bar{D}_1 + \bar{BC}_1) \quad - \\ \boldsymbol{M}_1^3(\boldsymbol{M}_2^1\bar{V}_2^1 + \boldsymbol{M}_2^3\bar{V}_1^1) &= \boldsymbol{M}_1^3(\bar{D}_2 + \bar{BC}_2) \quad = \end{aligned}$$

$$(\boldsymbol{M}_2^1\boldsymbol{M}_1^1 - \boldsymbol{M}_1^3\boldsymbol{M}_2^3)\bar{V}_1^1 = \boldsymbol{M}_2^1(\bar{D}_1 + \bar{BC}_1) - \boldsymbol{M}_1^3(\bar{D}_2 + \bar{BC}_2)$$

where we used the fact that $\boldsymbol{M}_2^1\boldsymbol{M}_1^3 = \boldsymbol{M}_1^3\boldsymbol{M}_2^1$ since $\boldsymbol{M}_1^3$ is a diagonal matrix.
Thus, by inverting the last equation we can find $\bar{V}_1^1$.
In order to find $\bar{V}_2^1$ repeat the same procedure but multiply the first equation in (5.25) by $\boldsymbol{M}_2^3$ and the second one by $\boldsymbol{M}_1^1$.

5.4 Numerical results

It is now time to implement all the described methods and in order to compare them, we plot the errors against the number of grid points.



(a) State 1 (b) State 2

Figure 5.1: Log-plot of absolute error against N for the European calls under the Black-Scholes regime-switching model

Table 5.1: Cpu times for the different methods against N

	Time (s)			
N	PDE	Lattice	FST	COS
100	0.1534	0.0496	0.0228	0.0160
120	0.2664	0.1688	0.0713	0.0565
140	0.3012	0.1882	0.0889	0.0621
160	0.4265	0.2713	0.1396	0.0640

We can clearly see that the COS method is by far the more efficient method. Not only in terms of accuracy, but also in terms of computational time. However, we have to make some clarifications:

- the reference solutions are given up to the 11th decimal digit, hence we find that the COS results are already in accord with the reference solutions for $N = 62$ in state 1 and $N = 46$ in state 2
- as regards the implementation of the FST method under regime-switching, in order to efficiently obtain the matrix exponential to evaluate the characteristic function we use again the formula (3.29)
- the PDE approach is supposed to work fine for great values of N, for instance $N = 1000$, thus for the range considered we cannot completely trust this method. Anyway this is a result itself, because in order to have better solutions we should increase N and this implies a greater computational load, confirming the efficiency of the COS method

Chapter 6

Digital and butterfly options

In the previous chapter we showed the efficiency of the COS method in pricing vanilla options. Now our aim is to extend the procedure to price other very popular financial derivatives, as it is done in [17]. Thus, let us start with the so called *digital* and *butterfly options*. They differ from the vanilla options only in the payoff, hence the COS method will remain the same as long as we consider the new cosine series coefficients. A digital call option is an option with payoff 0 if $S_T \leq K$ and K if $S_T > K$, the vice versa holds for a digital put. Therefore, the cosine series coefficients at time T are given by

$$\begin{aligned} V_k^i(T) &= \frac{2}{b-a}\int_a^b \cos\left(k\pi\frac{y-a}{b-a}\right) K \mathbb{1}_{[\alpha\cdot K(e^y-1)]^+} dy \\ &= \begin{cases} \frac{2}{b-a} K \xi_k(0,b) & \text{for a call} \\ \frac{2}{b-a} K \xi_k(a,0) & \text{for a put} \end{cases} \end{aligned} \tag{6.1}$$

The butterfly call (put) option, instead, is a combination of four options for which we have two long position calls (puts) with exercise price K_1 and K_3 and two short position calls (puts) with exercise price $K_2 = (K_1 + K_3)/2$. Therefore, the payoff of a butterfly call option at time T is given by

$$\Phi(X_T) = K_1(e^{y_1} - 1)^+ - 2K_2(e^{y_2} - 1)^+ + K_3(e^{y_3} - 1)^+$$

where $y_j = \log(S_T/K_j)$ for $j = 1, 2, 3$. Again, the cosine series coefficients are known analytically so that by calling $x_j = \log(K_j/p)$ we have

$$V_k^i(T) = \begin{cases} \frac{2}{b-a}\Big[\Big\{p\chi_k(x_1,b) - K_1\xi_k(x_1,b)\Big\} - 2\Big\{p\chi_k(x_2,b) - K_2\xi_k(x_2,b)\Big\} \\ \quad + \Big\{p\chi_k(x_3,b) - K_3\xi_k(x_3,b)\Big\}\Big] \quad \text{for a call} \\ \frac{2}{b-a}\Big[\Big\{K_1\xi_k(a,x_1) - p\chi_k(a,x_1)\Big\} - 2\Big\{K_2\xi_k(a,x_2) - p\chi_k(a,x_2)\Big\} \\ \quad + \Big\{K_3\xi_k(a,x_3) - p\chi_k(a,x_3)\Big\}\Big] \quad \text{for a put} \end{cases} \tag{6.2}$$

where p is a scale parameter and we take $p = 100$. In this case, we use the same definition of the truncation range as in equation (3.23) but now the interval is centered at $x_0 = \log(S_0/p)$.

6.1 Numerical Example

We propose here the following numerical example taken from [17]. Consider a two regime jump-diffusion Merton model with parameters

$$\begin{aligned} &S_0 = 100,\ K = 100, K_1 = 90,\ K_3 = 110,\ T = 1,\\ &r = [0.05, 0.06]',\ \delta = [0.03, 0.05]',\ \sigma = [0.1, 0.2]',\\ &\tilde{\mu} = [0, 0]',\ \tilde{\sigma} = [0.2, 0.2]',\ \lambda = [0.1, 0.2]' \end{aligned}$$

$$Q = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

The reference call values in state 1 are given in [17] up to the 10th decimal digit. We summarize in the following table the obtained errors and CPU times.

Table 6.1: Errors and cpu time for European, digital and butterfly call options in a two-regime jump-diffusion Merton model

	European Call		Digital Call		Butterfly Call	
N	Error	Time (s)	Error	Time (s)	Error	Time (s)
32	0.0908	0.0531	0.1219	0.0224	0.0870	0.0301
64	1.672e-04	0.0547	4.9930e-04	0.0225	1.1914e-04	0.0313
128	<1e-10	0.0551	<1e-10	0.0226	<1e-10	0.0331
Ref. val.	6.4077976575		49.4890166846		2.7062546784	

These results verify our claim, i.e. the COS method is really fast end efficient even with other type of options.

Chapter 7

Bermudan options

A Bermudan option is an option that offers the possibility of early exercise. It falls in between the American and the European option, in fact the owner can exercise it on a set of prespecified dates.
Let us consider the set of possible early exercise dates to be $\{t_0, t_1, \ldots, t_M\}$ where $0 = t_0 < t_1 < \ldots t_m < \cdots < t_M = T$, with $\Delta t = t_m - t_{m-1}$ and T being the final time. Note that by making $M \to \infty$ and therefore passing from discrete to continuous in terms of exercise opportunities, the price should converge to that of the corresponding American option.
Now let us see the implementation of its pricing algorithm following [17].

7.1 Pricing formula

From its description, it can be easily understood that the value of the Bermudan option at one of its possible exercise dates is given by

$$v^i(x, t_m) = \begin{cases} \Phi(x, t_M), & \text{for } m = M \\ \max(h^i(x, t_m), \Phi(x, t_m)), & \text{for } m = M-1, \ldots, 1 \end{cases} \tag{7.1}$$

where $i = 1, \ldots, J$ is the regime state and the continuation value $h^i(x, t_m)$ is given by

$$h^i(x, t_m) = \int_{\mathbb{R}} v^i(y, t_{m+1}) f(y|x, \tau^i) dy \tag{7.2}$$

In matrix-vector form, the initial price of the Bermudan option $\bar{v}(x, t_0)$ can be approximated by

$$\bar{v}(x, t_0) = \sum_{k=0}^{N-1}{}' \operatorname{Re}\left(e^{\mathbf{i}k\pi \frac{x-a}{b-a}} \boldsymbol{\varphi}\Big(\frac{k\pi}{b-a}\Big)\right) \bar{V}_k(t_1) \tag{7.3}$$

where $\bar{V}_k(t_1) = [V_k^1(t_1), \ldots, V_k^J(t_1)]'$ and $\bar{v}(x, t_0) = [v^1(x, t_0), \ldots, v^J(x, t_0)]'$

7.2 Coefficients $\bar{V}_k$

From the discussion above it follows immediately that for the valuation of the Bermudan option it is necessary to proceed backward in time and compare, in each regime and at each time step, the payoff and the continuation value. In this way, we determine the actual regime-dependent early exercise points denoted by $[x_1^*, x_2^*, \dots, x_J^*]$ and such that $h^i(x_i^*, t_m) = \Phi(x_i^*, t_m)$.
Substituting equation (7.1) in the general equation (3.12) of the Fourier cosine series, we have that

- at time t_M, $x_i^* = 0$ and the Fourier cosine coefficients are exact as in equation (3.22)
- for $m = M-1, M-2, \dots, 1$

$$V_k^i(t_m) = \frac{2}{b-a}\int_a^b \cos\left(k\pi\frac{y-a}{b-a}\right)\max(h^i(y,t_m), \Phi(y,t_m))dy \tag{7.4}$$

Let us focus on the case $m = M-1, M-2, \dots, 1$.
Once we find x_i^* at time t_m, we can split the integral, which defines $V_k^i(t_m)$, into two parts: one on the interval $[a, x_i^*]$, and the second on $(x_i^*, b]$, i.e.

$$V_k^i(t_m) = \begin{cases} \mathcal{G}_k^i(x_i^*, b) + \mathcal{H}_k^i(a, x_i^*, t_m) & \text{for a call} \\ \mathcal{G}_k^i(a, x_i^*) + \mathcal{H}_k^i(x_i^*, b, t_m) & \text{for a put} \end{cases} \tag{7.5}$$

where, considering $[c_i, d_i] \in [a, b]$,

$$\begin{aligned} \mathcal{G}_k^i(c_i, d_i) &= \frac{2}{b-a}\int_{c_i}^{d_i} \cos\left(k\pi\frac{x-a}{b-a}\right)\Phi(x,t_m)dx \\ &= \begin{cases} \frac{2}{b-a}K[\chi_k(c_i,d_i) - \xi_k(c_i,d_i)] & \text{for a call} \\ \frac{2}{b-a}K[\xi_k(c_i,d_i) - \chi_k(c_i,d_i)] & \text{for a put} \end{cases} \end{aligned} \tag{7.6}$$

and

$$\mathcal{H}_k^i(c_i, d_i, t_m) = \frac{2}{b-a}\int_{c_i}^{d_i} \cos\left(k\pi\frac{x-a}{b-a}\right)h^i(x,t_m)dx \tag{7.7}$$

7.3 Computation of $\mathcal{H}_k^i(c_i, d_i, t_m)$

For the computation of $\mathcal{H}_k^i(c_i, d_i, t_m)$ we refer to [8], adapting the discussion to the regime-switching framework.
For $l = 0, \dots, N-1$ using equation (3.17), inserting the ith component of (3.16) in equation (7.7) and interchanging summation and integration we obtain

$$\mathcal{H}_k^i(c_i, d_i, t_m) = \mathrm{Re}\left(\sum_{k=0}^{N-1}{}' \Lambda_l^i(t_{m+1}) M_{k,l}(c_i, d_i)\right) \tag{7.8}$$

where

$$M_{k,l}(c_i, d_i) := \frac{2}{b-a} \int_{c_i}^{d_i} e^{\mathtt{i}l\pi \frac{x-a}{b-a}} \cos\left(k\pi \frac{x-a}{b-a}\right) dx \tag{7.9}$$

With fundamental calculus, we can rewrite $M_{k,l}$ as

$$M_{k,l}(c_i, d_i) = -\frac{\mathtt{i}}{\pi}(M_{k,l}^c(c_i, d_i) + M_{k,l}^s(c_i, d_i)) \tag{7.10}$$

where

$$M_{k,l}^c(c_i, d_i) := \begin{cases} \frac{d_i - c_i}{b-a}\pi\mathtt{i} & k = l = 0 \\ \frac{\exp\left(\mathtt{i}(l+k)\frac{(d_i-a)\pi}{b-a}\right) - \exp\left(\mathtt{i}(l+k)\frac{(c_i-a)\pi}{b-a}\right)}{l+k} & \text{otherwise} \end{cases} \tag{7.11}$$

and

$$M_{k,l}^s(c_i, d_i) := \begin{cases} \frac{d_i - c_i}{b-a}\pi\mathtt{i} & k = l \\ \frac{\exp\left(\mathtt{i}(l-k)\frac{(d_i-a)\pi}{b-a}\right) - \exp\left(\mathtt{i}(l-k)\frac{(c_i-a)\pi}{b-a}\right)}{l-k} & k \neq l \end{cases} \tag{7.12}$$

Substituting equation (7.10) into equation (7.8), we have

$$\mathcal{H}_k^i(c_i, d_i, t_m) = \mathrm{Re}\left(\sum_{k=0}^{N-1}{}' \Lambda_l^i(t_{m+1})\Big(-\frac{\mathtt{i}}{\pi}(M_{k,l}^c(c_i, d_i) + M_{k,l}^s(c_i, d_i))\Big)\right) \tag{7.13}$$

In matrix-vector product form (7.13) reads

$$\bar{\mathcal{H}}^i(c_i, d_i, t_m) = \frac{1}{\pi}\,\mathrm{Im}\{(\mathbf{M}^c(c_i, d_i) + \mathbf{M}^s(c_i, d_i))\bar{u}^i\} \tag{7.14}$$

where $\mathrm{Im}\{\cdot\}$ denotes the imaginary part, and $\bar{u}^i$ collect the terms

$$u_l^i := \begin{cases} \frac{1}{2}\Lambda_l^i(t_{m+1}), & l = 0 \\ \Lambda_l^i(t_{m+1}), & l = 1, \ldots, N-1 \end{cases} \tag{7.15}$$

Moreover, $\mathbf{M}^c(c_i, d_i) := \{M_{k,l}^c(c_i, d_i)\}_{k,l=0}^{N-1}$ and $\mathbf{M}^s(c_i, d_i) := \{M_{k,l}^s(c_i, d_i)\}_{k,l=0}^{N-1}$. However, the matrix-vector product in (7.14) requires $O(N^2)$ computations and thus we follow the procedure described in [8], which develops an FFT-based algorithm to reduce the computational complexity to $O(N\log_2(N))$.
We have that $\mathbf{M}^c(c_i, d_i)$ and $\mathbf{M}^s(c_i, d_i)$ are $N \times N$ matrices, specifically a Hankel matrix and a Toeplitz matrix, respectively

$$\mathbf{M}^c(c_i, d_i) = \begin{bmatrix} m_0 & \ldots & m_{N-1} \\ m_1 & \ldots & m_N \\ \vdots & & \vdots \\ m_{N-1} & \ldots & m_{2N-2} \end{bmatrix} \quad \mathbf{M}^s(c_i, d_i) = \begin{bmatrix} m_0 & \ldots & m_{N-1} \\ m_{-1} & \ldots & m_{N-2} \\ \vdots & & \vdots \\ m_{1-N} & \ldots & m_0 \end{bmatrix} \tag{7.16}$$

with

$$m_l := \begin{cases} \frac{d_i - c_i}{b-a}\pi\mathtt{i} & l = 0 \\ \frac{\exp\left(\mathtt{i}l\frac{(d_i-a)\pi}{b-a}\right) - \exp\left(\mathtt{i}l\frac{(c_i-a)\pi}{b-a}\right)}{l} & l \neq 0 \end{cases} \tag{7.17}$$

Due to the special structure of these matrices, we can rewrite the matrix-vector product into circular convolutions. Specifically:

- the product $\mathbf{M}^s(c_i, d_i)\bar{u}^i$ is equal to the first N elements of $\bar{m}_s \circledast \bar{u}^i_s$ where

$$\bar{m}_s = [m_0, m_{-1}, m_{-2}, \ldots, m_{1-N}, 0, m_{N-1}, m_{N-2}, \ldots, m_1]' \tag{7.18}$$

$$\bar{u}^i_s = [u^i_0, u^i_1, \ldots, u^i_{N-1}, 0, \ldots, 0]' \tag{7.19}$$

- the product $\mathbf{M}^c(c_i, d_i)\bar{u}^i$ is equal to the first N elements of $\bar{m}_c \circledast \bar{u}^i_c$ where

$$\bar{m}_c = [m_{2N-1}, m_{2N-2}, \ldots, m_1, m_0]' \tag{7.20}$$

$$\bar{u}^i_c = [0, \ldots, 0, u^i_0, u^i_1, \ldots, u^i_{N-1}]' \tag{7.21}$$

And these can be efficiently dealt with by the FFT algorithm, indeed given two vectors $\bar{x}$ and $\bar{y}$ we can write

$$\bar{x} \circledast \bar{y} = \mathcal{D}^{-1}(\mathcal{D}(\bar{x}) \cdot \mathcal{D}(\bar{y}))$$

where $\mathcal{D}$ is the discrete Fourier transform (DFT).
Notice that we can obtain the DFT of $\bar{u}_c$ from the one of $\bar{u}_s$ exploiting the shift property of DFTs, i.e. $\mathcal{D}(\bar{u}_c) = \overline{sgn} \cdot \mathcal{D}(\bar{u}_s)$ with $\overline{sgn} = [1, -1, 1, -1, \ldots]'$.
Finally, let us stress that, in order to reduce the computational time required to construct $\bar{m}_s$ and $\bar{m}_c$, we can compute the factors $\exp\left(\mathtt{i}l\frac{(d_i-a)\pi}{b-a}\right)$ and $\exp\left(\mathtt{i}l\frac{(c_i-a)\pi}{b-a}\right)$ $(l = 0, 1, \ldots, N-1)$ only once exploiting some special properties of the m_l's:

- $m_{-l} = -\overline{m_l}$ $\quad$ ($\overline{m_l}$ denotes the complex conjugate of m_l)
- $m_{l+N} = \frac{\exp\left(\mathtt{i}N\frac{(d_i-a)\pi}{b-a}\right)\cdot\exp\left(\mathtt{i}l\frac{(d_i-a)\pi}{b-a}\right) - \exp\left(\mathtt{i}N\frac{(c_i-a)\pi}{b-a}\right)\cdot\exp\left(\mathtt{i}l\frac{(c_i-a)\pi}{b-a}\right)}{l+N}, \quad l \neq 0$

7.4 Numerical Example

We consider once again the Merton's jump-diffusion Markov-modulated model, with parameters

$$S_0 = 100,\ K = 100,\ T = 0.5,\ r = [0.05, 0.06]',$$
$$\delta = [0.05, 0.07]',\ \sigma = [0.1, 0.2]',\ \tilde{\mu} = [0, 0]',$$
$$\tilde{\sigma} = [0.3, 0.4]',\ \lambda = [0.1, 0.2]',\ q_{12} = q_{21} = 1$$

and we price a Bermudan put option with $M = 10$ exercise dates but different values for N. Then we focus on the prices in state 2 and we consider as reference solution the one obtained with $N = 2^{16}$, which is: 6.249854030604797. The results are summarized in the following table.

Table 7.1: Errors and cpu time for Bermudan put option in a two-regime jump-diffusion Merton model

Bermudan Put			
N	Price	Error	Time (s)
64	6.259773455002508	0.0099	0.0562
128	6.250000729430543	1.4670e-04	0.0713
256	6.249854354808557	3.2420e-07	0.0808
512	6.249854030599596	5.2012e-12	0.0965
Ref. val. 6.249854030604797			

We then implement the FST method and we find out that we manage to reach almost the same level of accuracy of the COS method by taking $N = 2^{23}$ and $x_{max} = 4$. Indeed in this case, the FST price is 6.249854030824808 with an error of $2.2001e{-}10$. This further confirms the supremacy of the COS method over the FST method.

Chapter 8

Barrier options

Barrier options are financial derivatives the payoffs of which depend upon the underlying asset crossing a certain barrier level. Firstly, there is a major distinction that needs to be addressed: in/out barrier options. In the case of the underlying price touching the barrier, the former become valid whereas the latter invalid. Then, another important distinction is between up, down and knock barrier options. To explain it, let us focus on the out barrier options

- **Up & Out** the option ceases to exist when the underlying price rises above a specific level
- **Down & Out** the option ceases to exist when the underlying price falls below a specific level
- **Knock & Out** the option ceases to exist when the underlying price exits a region bounded by both barriers (Up and Down)

In barrier options are defined similarly, with the only difference that the barrier crossing makes the option valid. As it is commonly done in literature, we will price only out contracts, since the in contract value can be obtained with the so called in-out parity, that is

$$C = C_{IN} + C_{OUT} \tag{8.1}$$

where C is the price of an European option, C_{IN} and C_{OUT} are the prices of the barrier options with the same level. Moreover, we will not consider knock barriers since their derivation is straightforward.
As in the case of American and Bermudan options, even Barrier options can be of continuous or discrete type, which differ in how often the underlying is monitored.

8.1 Pricing formula

Here we continue to refer to [17]. As with Bermudan options, even in this case the pricing method requires a backward in time recursive procedure. However, we do not need anymore to compare the payoff and the continuation value in each regime and at each time step, but attention must be paid to the fact that the continuation value must include information regarding a possible crossing of the barrier level. Nevertheless, notice that barrier levels are already known beforehand implying the fact that pricing Barrier options is much easier than pricing Bermudan ones.
Let us consider, for instance, an up-and-out option. Its payoff function is given by

$$v^i(x,T) = \Big[[\alpha \cdot (S_T - K)]^+ - Rb\Big] \mathbb{1}_{S_t<B} + Rb \tag{8.2}$$

where $\alpha = 1$ for a call and $\alpha = -1$ for a put and Rb is a *rebate*, which is a fixed fee to be paid to the option holder in the case that the barrier event occurs.
Let $x_b = \log(B/K)$ and $m = M-1,\dots,1$ for each regime $i \in \mathcal{J}$. The price of the option monitored M times satisfies the following system

$$\begin{cases} \bar{h}(x,t_{m-1}) = \int_{\mathbb{R}} \mathbf{F}(y|x,\tau)\bar{v}(y,t_m)dy \\ \bar{v}(x,t_{m-1}) = \begin{cases} \bar{R}b & x \geq x_b \\ \bar{h}(x,t_{m-1}) & x < x_b \end{cases} \end{cases} \tag{8.3}$$

Based on the derivation for Bermudan options, we have the following lemma.

Lemma 8.1.1 ***Backward Induction for Discrete Barrier Options***
By backward recursion we find the following solution for discretely monitored barrier options:
For $m = M-1, M-2,\dots,1$,

$$V_k^i(t_m) = \mathcal{H}_k^i(a,x_b,t_m) + \frac{2}{b-a} Rb\xi_k(x_b,b) \tag{8.4}$$

with $\mathcal{H}_k^i(a,x_b,t_m)$ *and* $\xi_k(x_b,b)$ *given by* (7.14) *and* (3.21)*, respectively.*
If $x_b < 0$*, we have*

$$V_k^i(t_M) = \begin{cases} 2Rb\xi_k(x_b,b)/(b-a) & \textit{for a call} \\ G_k^i(a,x_b) + 2Rb\xi_k(x_b,b)/(b-a) & \textit{for a put} \end{cases} \tag{8.5}$$

For $x_b \geq 0$*, we find*

$$V_k^i(t_M) = \begin{cases} G_k^i(0,x_b) + 2Rb\xi_k(x_b,b)/(b-a) & \textit{for a call} \\ G_k^i(a,0) + 2Rb\xi_k(x_b,b)/(b-a) & \textit{for a put} \end{cases} \tag{8.6}$$

with $G_k^i(c_i,d_i)$ *given by equation* (7.6).
A similar recursion formula for a down-and-out option can be derived easily.

8.2 Numerical Example

We continue the example of Section 7.4 considering now an up & out call barrier in state 2 with $M = 12$ monitoring dates (i.e. monthly monitored), barrier level $B = 120$ and rebate $Rb = 0$. Again we consider as reference solution the one obtained with $N = 2^{16}$, which is 2.277022824693888. Below we report the table with the results obtained.

Table 8.1: Errors and cpu time for up & out call option in a two-regime jump-diffusion Merton model

UO Barrier Call			
N	Price	Error	Time (s)
64	2.303984051645670	0.0270	0.0211
128	2.275748862495687	0.0013	0.0236
256	2.276939342532470	8.3482e-05	0.0248
512	2.277022821405835	3.2881e-09	0.0331
Ref. val. 2.277022824693888			

Then, if we implement again the FST method with $N = 2^{23}$ and $x_{max} = 4$ we obtain a price of 2.277023091369360, with an error equal to $2.6668e-07$, leading to the same conclusions made in the previous example.
Finally, let us notice that in this example we are considering a call option but we cannot adapt put-call parity to barrier options because in this example r and δ assume different values in the 2 regimes. However, this is not even necessary, since Merton distribution has not fat tails and thus we can reasonably assume that if there were any numerical errors these would be minimal.

Chapter 9

American and continuously monitored barrier options

As we already anticipated in the previous sections, American option prices can be obtained by Bermudan options by increasing the number of exercise dates. The continuous barrier option can also be approximated in a similar fashion based on a set of discretely monitored barrier options. However, these procedures can be time consuming and computationally expensive, it is why we resort to the application of Richardson extrapolation with s stages ([15]).
Richardson extrapolation requires prior knowledge of the rate of convergence which, from theory, we know to be 1 in the case of convergence of the discrete Bermudan to American and 1/2 in the case of the discrete barrier converging to the continuous barrier. The s-stage extrapolation tableau is then given by

$$\begin{aligned} v^i_{l,0} &= v^i(M = 2^{l-1}M_0) \\ v^i_{l,j} &= \frac{(2^{\text{rate}})^j v^i_{l+1,j-1} - v^i_{l,j-1}}{(2^{\text{rate}})^j - 1} \end{aligned} \tag{9.1}$$

where $j = 1, \ldots, s-1$, $l = 1, \ldots, s$ and $v^i_{l,0}$ is the price of a discrete Bermudan or barrier option with M monitoring dates in regime $i \in \mathcal{J}$. We start with an initial number of monitoring dates M_0 and then we double M at each stage.
Regarding the choice of the number of stages, as we will see in the numerical example, we have verified that the suggestions reported in [17] to use $s = 4$ for American options and $s = 6$ for the continuous barrier options are optimal, so we adopt them. Considering for instance $s = 4$, the computations can be conveniently set up in the following scheme.

l	$v^i_{l,0}$	$v^i_{l,1}$	$v^i_{l,2}$	$v^i_{l,3}$
1	$v^i_{1,0}$	$v^i_{1,1}$	$v^i_{1,2}$	$v^i_{1,3}$
2	$v^i_{2,0}$	$v^i_{2,1}$	$v^i_{2,2}$	
3	$v^i_{3,0}$	$v^i_{3,1}$		
4	$v^i_{4,0}$			

where, in the case rate $= 1$ we have

$$v^i_{1,1} = 2v^i_{2,0} - v^i_{1,0} \qquad v^i_{1,2} = \frac{4v^i_{2,1} - v^i_{1,1}}{3} \qquad v^i_{1,3} = \frac{8v^i_{2,2} - v^i_{1,2}}{7}$$

$$v^i_{2,1} = 2v^i_{3,0} - v^i_{2,0} \qquad v^i_{2,2} = \frac{4v^i_{3,1} - v^i_{2,1}}{3}$$

$$v^i_{3,1} = 2v^i_{4,0} - v^i_{3,0}$$

Hence, we finally get the formula for the value of an American option in the state i using 4-point Richardson extrapolation scheme

$$v^i_{1,3} = \frac{1}{21}\left[64v^i_{4,0} - 56v^i_{3,0} + 14v^i_{2,0} - v^i_{1,0}\right] \tag{9.2}$$

Similarly, doing the computations for $s = 6$ and rate $= 1/2$ we obtain

$$\begin{aligned} v^i_{1,5} =& \frac{1}{3(4\sqrt{2}-1)(2\sqrt{2}-1)(\sqrt{2}-1)}\left[128\sqrt{2}v^i_{6,0} - 32(7+3\sqrt{2})v^i_{5,0}\right. \\ &\left. + 24(7+3\sqrt{2})v^i_{4,0} - 6(7\sqrt{2}+6)v^i_{3,0} + (7\sqrt{2}+6)v^i_{2,0} - v^i_{1,0}\right] \end{aligned} \tag{9.3}$$

9.1 Numerical Example

In order to test the accuracy of Richardson extrapolation formulas derived above, we consider once again the same data as the numerical examples in Section 7.4 and 8.2. For convenience we report them here

$$\begin{aligned} &S_0 = 100,\ K = 100,\ T = 0.5,\ r = [0.05, 0.06]',\\ &\delta = [0.05, 0.07]',\ \sigma = [0.1, 0.2]',\ \tilde{\mu} = [0, 0]',\\ &\tilde{\sigma} = [0.3, 0.4]',\ \lambda = [0.1, 0.2]', q_{12} = q_{21} = 1 \end{aligned}$$

with barrier level $B = 120$, rebate $Rb = 0$.
While we vary the number of monitoring dates M, we decide to fix the number of grid points to $N = 2^{12}$. Below we summarize the results obtained for state 2 in a table.

Table 9.1: Errors and cpu time for the American put option and the up & out continuously monitored call barrier option in the state 2

		Extrapolation		
M	Error	Price	Error	Time (s)
Bermudan approximation to American in state 2				
16	6.0981e-04	6.250840779	4.6193e-06	0.4273
32	3.0250e-04	6.250837644	1.4840e-06	0.8340
64	1.5036e-04	6.250836690	5.2997e-07	1.5584
128	7.4860e-05	6.250836356	1.9633e-07	3.9975
		Ref. val. 6.2508361596		
Discrete barrier approximation to continuous barrier in state 2				
64	0.1821	1.890757303	3.6082e-04	0.5733
128	0.1310	1.890401669	5.1843e-06	1.1149
256	0.0938	1.890395982	5.0248e-07	2.7230
512	0.0669	1.890396474	1.1247e-08	5.0355
		Ref. val. 1.8903964848		

In Richardson extrapolation, M represents the maximum number of monitoring dates which appears in its formula.
From the table, it is evident that, for the same M, in order to price options in continuous time, the extrapolation approach (on the right) allows to obtain a more precise result than simply considering the discrete Bermudan or barrier option (on the left). In particular, this can be seen in the barrier option for $M = 512$, where the extrapolation returns an error of $1.1247e-08$ against an error of 0.0669 of the discrete barrier approximation. Moreover, by looking at the results of the discrete Bermudan option we can see that the more M increases, the more the error becomes negligible confirming what we claimed at the beginning of Chapter 7, i.e. as $M \to \infty$ the price converges to the American option price.

Now all that remains is to discuss the choice of the number of stages to use in the extrapolation formula. We fix for instance $M = 128$ and we repeat the pricing for different values of s. Below we report the table of the obtained errors with respect to the reference values, from which we can clearly see that for the American option, there is no such difference between the errors and thus $s = 4$ is a good choice. Instead, for the continuous barrier option, the errors have different orders of magnitude depending on s implying that $s = 4$ is not enough but should be

increased, in our case to $s = 6$.

Table 9.2: Errors and cpu time against s

	American Put		Continuous Barrier Call	
$M = 128$	Error	Time (s)	Error	Time (s)
$s = 3$	2.6666e-07	4.0200	1.4210e-04	1.4179
$s = 4$	1.9633e-07	3.9339	7.5194e-05	1.3610
$s = 5$	1.7409e-07	4.4089	2.9534e-06	1.3073
$s = 6$	1.6466e-07	3.3975	5.1843e-06	1.0458

Chapter 10

Asian options

In this chapter our goal is to give an innovative contribution to this dissertation consisting in developing the COS method under regime-switching for Asian options. To this aim, we refer to [20] but we extend the procedure to the regime-switching framework.
Asian options are financial derivatives belonging to the class of exotic options. Specifically, they are path-dependent options where the payoff depends on the average of the underlying asset prices during option's life. The most popular types of averages considered are geometric average and arithmetic average.
Let us consider the set of monitoring dates to be $\{t_0, t_1, \ldots, t_M\}$ where $0 = t_0 < t_1 < \ldots t_m < \cdots < t_M = T$, with $\Delta t = t_m - t_{m-1}$ and T being the final time. If $M \to \infty$, we pass from discrete to continuous time. Then

- geometric average in discrete time

$$A(T) = \left(\prod_{m=0}^{M} S_m \right)^{\frac{1}{M+1}} \tag{10.1}$$

- geometric average in continuous time

$$A(T) = \exp\left(\frac{1}{T} \int_0^T \log(S_t) dt \right) \tag{10.2}$$

- arithmetic average in discrete time

$$A(T) = \frac{1}{M+1} \sum_{m=0}^{M} S_m \tag{10.3}$$

- arithmetic average in continuous time

$$A(T) = \frac{1}{T} \int_0^T S_t dt \tag{10.4}$$

Another important distinction that needs to be addressed is between floating-strike and fixed-strike Asian options, which differ in how the payoff function is defined. The floating-strike Asian option has payoff

$$v^i(S,T) = \Big[\alpha \cdot \Big(S_T - A(T)\Big)\Big]^+ \tag{10.5}$$

where $i = 1, \dots, J$ is the regime state and $\alpha = 1$ for a call, $\alpha = -1$ for a put. While the fixed-strike has payoff

$$v^i(S,T) = \Big[\alpha \cdot \Big(A(T) - K\Big)\Big]^+ \tag{10.6}$$

We will focus on fixed-strike Asian options.

10.1 Geometric Asian options

10.1.1 Coefficients $\bar{V}_k$

Defining

$$y = \log\left(\left(\prod_{m=0}^{M} S_m\right)^{\frac{1}{M+1}}\right) = \frac{1}{M+1}\sum_{m=0}^{M}\log(S_m) = \frac{1}{M+1}\sum_{m=0}^{M} x_m \tag{10.7}$$

We can rewrite the payoff in (10.6) in terms of y in the following way

$$v^i(y, t_M) = [\alpha \cdot (e^y - K)]^+ \tag{10.8}$$

Therefore, the cosine series coefficients are given by

$$\begin{aligned} V_k^i(t_M) &= \frac{2}{b-a}\int_a^b \cos\left(k\pi\frac{y-a}{b-a}\right) v^i(y,t_M)dy \\ &= \frac{2}{b-a}\int_a^b \cos\left(k\pi\frac{y-a}{b-a}\right) [\alpha\cdot(e^y-K)]^+ dy \\ &= \begin{cases} \frac{2}{b-a}[\chi_k(\log(K),b) - K\xi_k(\log(K),b)] & \text{for a call} \\ \frac{2}{b-a}[K\xi_k(a,\log(K)) - \chi_k(a,\log(K))] & \text{for a put} \end{cases} \end{aligned} \tag{10.9}$$

10.1.2 Pricing formula

In order to use the Fourier cosine expansion, we need to determine the conditional characteristic function of y given x_0. This for geometric Asian options can be computed directly.

First of all, since the increments of a Lévy process are stationary and independent,

we can denote the (identical) characteristic functions of the increments $x_1 - x_0$, $x_2 - x_1, \ldots, x_M - x_{M-1}$ by $\boldsymbol{\phi}(u, \Delta t)$, i.e.

$$\boldsymbol{\phi}(u, \Delta t) = \mathbb{E}[\exp(\mathtt{i}u \log(S_m/S_{m-1}))] = \mathbb{E}[\exp(\mathtt{i}u(x_m - x_{m-1}))] \tag{10.10}$$

We recall that for $J = 2$, $\boldsymbol{\phi}(u, \Delta t)$ will have the same structure as equation (3.30). Based on [9], we can express y in (10.7) as

$$\begin{aligned} y &= \frac{1}{M+1}\sum_{m=0}^{M}\log(S_m) = \log(S_0) + \frac{1}{M+1}\sum_{m=1}^{M}\sum_{j=1}^{m} x_j \\ &= x_0 + \frac{1}{M+1}\sum_{m=1}^{M} x_m \sum_{j=m}^{M} 1 = x_0 + \frac{1}{M+1}\sum_{m=1}^{M}(M+1-m)x_m \end{aligned}$$

Hence, the characteristic function of y given x_0 in the state i can be computed exploiting the properties of conditional expectation

$$\begin{aligned} \varphi^i(u|x_0) &= \mathbb{E}\Big[e^{\mathtt{i}ux_0} e^{\mathtt{i}u\sum_{m=1}^{M}\frac{M+1-m}{M+1}x_m}\Big|\alpha_0 = i\Big] \\ &= e^{\mathtt{i}ux_0}\cdot\mathbb{E}\Big[\prod_{m=1}^{M} e^{\mathtt{i}u\frac{M+1-m}{M+1}x_m}\Big|\alpha_0 = i\Big] \\ &= e^{\mathtt{i}ux_0}\cdot\prod_{m=1}^{M}\mathbb{E}\Big[e^{\mathtt{i}u\frac{M+1-m}{M+1}x_m}\Big|\alpha_0 = i\Big] \\ &= e^{\mathtt{i}ux_0}\cdot\prod_{m=1}^{M}\phi^i\Big(u\frac{M+1-m}{M+1}, \Delta t\Big) \end{aligned} \tag{10.11}$$

where

$$\phi^i(u, \Delta t) = \sum_{j=1}^{J}\boldsymbol{\phi}(u, \Delta t)_{i,j} \tag{10.12}$$

Therefore, the geometric Asian option price in state i can be derived by

$$v^i(x, t_0) = \sum_{k=0}^{N-1}{}' \operatorname{Re}\left(e^{-\mathtt{i}k\pi\frac{a}{b-a}}\Lambda_k^i(t_M|x)\right) \tag{10.13}$$

with

$$\Lambda_k^i(t_m|x) = \varphi^i\Big(\frac{k\pi}{b-a}\Big|x\Big)\cdot V_k^i(t_m) \tag{10.14}$$

10.2 Arithmetic Asian option

10.2.1 Coefficients $\bar{V}_k$

The payoff function in state i of an arithmetic Asian option with M monitoring dates and a fixed strike reads as

$$v^i(S, t_M) = \left[\alpha\cdot\left(\left(\frac{1}{M+1}\sum_{m=0}^{M} S_m\right) - K\right)\right]^+ \tag{10.15}$$

In this case, the characteristic function of the arithmetic mean cannot be computed directly. Therefore, it will be derived recursively by Fourier cosine expansions and Clenshaw-Curtis quadrature.
We begin the recursion procedure by denoting

$$R_m = \log\left(\frac{S_m}{S_{m-1}}\right) = x_m - x_{m-1}, \qquad m = 1, \dots, M \tag{10.16}$$

From Lévy properties, we know that these increments are identically and independently distributed, so that $R_m \overset{d}{=} R$. Thus, it follows that $\boldsymbol{\varphi}_{R_m}(u) = \boldsymbol{\varphi}_R(u)\ \forall u, m$. Characteristic function $\boldsymbol{\varphi}_R(u)$ is known in closed form for different Lévy processes and for $J = 2$ is given by the procedure described in Section 3.5.1.
Then, we introduce the stochastic process Y_m, where $Y_1 := R_M$ and for $m = 2, \dots, M$ we have

$$Y_m := R_{M+1-m} + \log(1 + \exp(Y_{m-1})) = R_{M+1-m} + Z_{m-1} \tag{10.17}$$

where we defined $Z_m := \log(1 + \exp(Y_m))$, $\forall m$.
In this setting Y_m admits the form

$$Y_m = \log\left(\frac{S_{M+1-m}}{S_{M-m}} + \frac{S_{M+2-m}}{S_{M-m}} + \cdots + \frac{S_M}{S_{M-m}}\right) \tag{10.18}$$

and we have that

$$\frac{1}{M+1}\sum_{m=0}^{M} S_m = \frac{(1 + \exp(Y_M))S_0}{M+1} \tag{10.19}$$

Substituting (10.19) in (10.15) we can rewrite the payoff in the following way

$$v^i(y, t_M) = \left[\alpha \cdot \left(\frac{S_0(1 + \exp(y))}{M+1} - K\right)\right]^+ \tag{10.20}$$

Hence, the cosine series coefficients are given by

$$\begin{aligned} V_k^i(t_M) &= \frac{2}{b-a}\int_a^b \cos\left(k\pi\frac{y-a}{b-a}\right) v^i(y, t_M)\, dy \\ &= \frac{2}{b-a}\int_a^b \cos\left(k\pi\frac{y-a}{b-a}\right)\left[\alpha \cdot \left(\frac{S_0(1 + \exp(y))}{M+1} - K\right)\right]^+ dy \\ &= \begin{cases} \frac{2}{b-a}\left[\frac{S_0}{M+1}\chi_k(x^*, b) - \left(K - \frac{S_0}{M+1}\right)\xi_k(x^*, b)\right] & \text{for a call} \\ \frac{2}{b-a}\left[\left(K - \frac{S_0}{M+1}\right)\xi_k(a, x^*) - \frac{S_0}{M+1}\chi_k(a, x^*)\right] & \text{for a put} \end{cases} \end{aligned} \tag{10.21}$$

where $x^* = \log\left(\frac{K(M+1)}{S_0} - 1\right)$.

10.2.2 Pricing formula

In order to recover the characteristic function of Y_M, i.e. $\boldsymbol{\varphi}_{Y_M}(u)$, we start with Y_1, for which the characteristic function reads as

$$\boldsymbol{\varphi}_{Y_1}(u) = \boldsymbol{\varphi}_R(u) \tag{10.22}$$

Then, for $m = 2, \ldots, M$, $\boldsymbol{\varphi}_{Y_m}(u)$ can be recovered in terms of $\boldsymbol{\varphi}_{Y_{m-1}}(u)$. This is done by application of (10.17) and the fact that Lévy processes have independent increments.
This implies that, $\forall m$, R_{M+1-m} and Z_{m-1} are independent in each state i, which gives

$$\varphi^i_{Y_m}(u) = \varphi^i_{R_{M+1-m}}(u) \cdot \varphi^i_{Z_{m-1}}(u) = \varphi^i_R(u) \cdot \varphi^i_{Z_{m-1}}(u) \tag{10.23}$$

where

$$\varphi^i_{Y_m}(u) = \sum_{j=1}^{J} \boldsymbol{\varphi}_{Y_m}(u)_{i,j} \qquad \varphi^i_R(u) = \sum_{j=1}^{J} \boldsymbol{\varphi}_R(u)_{i,j} \tag{10.24}$$

From the definition of characteristic function, we have

$$\begin{aligned}\varphi^i_{Z_{m-1}}(u) &= \mathbb{E}\left[e^{\mathtt{i}u\log(1+\exp(Y_{m-1}))}|\alpha_{M+1-m} = i\right] = \int_{-\infty}^{\infty} e^{\mathtt{i}u\log(1+e^y)} f^i_{Y_{m-1}}(y)dy \\ &= \int_{-\infty}^{\infty} (e^y+1)^{\mathtt{i}u} f^i_{Y_{m-1}}(y)dy \end{aligned} \tag{10.25}$$

To apply the Fourier cosine series expansion to approximate the characteristic function, we first truncate the integration range, i.e.

$$\varphi^i_{Z_{m-1}}(u) = \int_a^b (e^y+1)^{\mathtt{i}u} f^i_{Y_{m-1}}(y)dy \tag{10.26}$$

Then we apply the Fourier cosine expansion to approximate $f^i_{Y_{m-1}}(y)$, giving

$$\begin{aligned}\varphi^i_{Z_{m-1}}(u) =& \frac{2}{b-a}\sum_{l=0}^{N-1}{}' \operatorname{Re}\left(e^{-\mathtt{i}l\pi\frac{a}{b-a}}\varphi^i_{Y_{m-1}}\left(\frac{l\pi}{b-a}\right)\right) \\ &\cdot \int_a^b (e^y+1)^{\mathtt{i}u}\cos\left(l\pi\frac{y-a}{b-a}\right)dy \end{aligned} \tag{10.27}$$

Equation (10.27) can be written in matrix-vector form as

$$\boldsymbol{\varphi}^i_{m-1} = \mathcal{M}A^i_{m-1} \tag{10.28}$$

where

$$\varphi^i_{m-1} = (\varphi^i_{m-1}(k))_{k=0}^{N-1} \quad \varphi^i_{m-1}(k) = \varphi^i_{Z_{m-1}}(u_k)$$
$$u_k = \frac{k\pi}{b-a} \quad k = 0, \ldots, N-1$$
$$\mathcal{M} = (\mathcal{M}(k,l))_{k,l=0}^{N-1} \quad \mathcal{M}(k,l) = \int_a^b (e^y+1)^{\mathtt{i}u_k} \cos((y-a)u_l)dy$$
$$A^i_m = \frac{2}{b-a}(A^i_m(l))_{l=0}^{N-1} \quad A^i_m(0) = 0.5A^i_m(0) \quad A^i_m(l) = \mathrm{Re}\left(e^{-\mathtt{i}au_l}\varphi^i_{Y_{m-1}}(u_l)\right)$$

Hence, the COS formula for the arithmetic Asian option price in state i reads as

$$v^i(x,t_0) = \sum_{k=0}^{N-1}{}' \mathrm{Re}\left(e^{-\mathtt{i}k\pi\frac{a}{b-a}}\Lambda^i_k(t_M)\right) \tag{10.29}$$

with

$$\Lambda^i_k(t_m) = \varphi^i_{Y_m}\left(\frac{k\pi}{b-a}\right) \cdot V^i_k(t_m) \tag{10.30}$$

10.2.3 Truncation range

In the previous paragraph, we derived the characteristic function of Y_m, $m = 1, \ldots, M$, independently of x_0. With this in mind, the expression for the integration range in (10.27) can be found following the discussion of Section 3.4, i.e.

$$\begin{aligned} a_{i,m} &= c^i_1(Y_m) - L\sqrt{c^i_2(Y_m) + \sqrt{c^i_4(Y_m)}} \\ b_{i,m} &= c^i_1(Y_m) + L\sqrt{c^i_2(Y_m) + \sqrt{c^i_4(Y_m)}} \end{aligned} \tag{10.31}$$

$$a = \min_i\left(\min_m\{a_{i,m}\}\right) \quad \text{and} \quad b = \max_i\left(\max_m\{b_{i,m}\}\right) \tag{10.32}$$

However, it is rather expensive to determine these cumulants here, and therefore we adopt the procedure described in [20] extending it to the regime-switching framework.

For each Y_m, $m = 1, \ldots, M$, we have

$$c^i_1\left(m\frac{S_{M+1-m}}{S_{M-m}}\right) \le c^i_1\big(\exp(Y_m)\big) \le c^i_1\left(m\frac{S_M}{S_{M-m}}\right)$$
$$0 \le c^i_2\big(\exp(Y_m)\big) \le c^i_2\left(m\frac{S_M}{S_{M-m}}\right)$$
$$0 \le c^i_4\big(\exp(Y_m)\big) \le c^i_4\left(m\frac{S_M}{S_{M-m}}\right)$$

Therefore, a truncation range for Y_m can be defined as

$$a_{i,m} = c_1^i\left(\log\left(m\frac{S_{M+1-m}}{S_{M-m}}\right)\right) - L\sqrt{c_2^i\left(\log\left(m\frac{S_M}{S_{M-m}}\right)\right) + \sqrt{c_4^i\left(\log\left(m\frac{S_M}{S_{M-m}}\right)\right)}}$$
$$b_{i,m} = c_1^i\left(\log\left(m\frac{S_M}{S_{M-m}}\right)\right) + L\sqrt{c_2^i\left(\log\left(m\frac{S_M}{S_{M-m}}\right)\right) + \sqrt{c_4^i\left(\log\left(m\frac{S_M}{S_{M-m}}\right)\right)}} \tag{10.33}$$

Note that in principle, given a random variable Z, we have that $\forall i \in \mathcal{J},\ \forall n \geq 1$, $\log(c_n^i(Z)) \neq c_n^i(\log(Z))$, but this does not influence the fact that as $L \to \infty$ the truncation error goes to zero.
The cumulants of $\log\left(m\frac{S_{M+1-m}}{S_{M-m}}\right)$ and $\log\left(m\frac{S_M}{S_{M-m}}\right)$ in (10.33) are known in closed form for exponential Lévy asset price processes, since

$$c_1^i\left(\log\left(m\frac{S_{M+1-m}}{S_{M-m}}\right)\right) = \log(m) + c_1^i(R),\ c_1^i\left(\log\left(m\frac{S_M}{S_{M-m}}\right)\right) = \log(m) + mc_1^i(R)$$
$$\forall n \geq 2\ \ c_n^i\left(\log\left(m\frac{S_{M+1-m}}{S_{M-m}}\right)\right) = c_n^i(R),\ \ c_n^i\left(\log\left(m\frac{S_M}{S_{M-m}}\right)\right) = mc_n^i(R)$$

with R the logarithm of the increment of an exponential Lévy process, between any two consecutive time steps. These expressions are based on $\log(mZ) = \log(m) + \log(Z)$ and on the fact that for an exponential Lévy asset price process, the cumulants of the log-asset returns, $\log(S_l/S_k)\ \ \forall l > k$, are linearly increasing functions of $t = (l-k)\Delta t$.
Hence, we set

$$[a,b] = \left[\min_i\left(\min_m\{a_{i,m}\}\right), \max_i\left(\max_m\{b_{i,m}\}\right)\right] \tag{10.34}$$

10.2.4 Clenshaw-Curtis quadrature

Here we discuss the computation of $\mathcal{M}$ needed in equation (10.28), which can be done efficiently using Clenshaw-Curtis quadrature, as pointed out in [20]. Notice that $\mathcal{M}$ remains constant for all time steps $t_m,\ m = 1,\ldots,M-1$ and for all states $i \in \mathcal{J}$, so that we need to compute it only once.
Its elements are given by

$$\mathcal{M}(k,l) = \int_a^b (e^y+1)^{\mathbf{i}u_k}\cos((y-a)u_l)dy \qquad k,l = 0,\ldots,N-1 \tag{10.35}$$

To use the Clenshaw-Curtis rule, we first change the integration interval from $[a,b]$ to $[-1,1]$

$$\begin{aligned}
&\int_a^b (e^y+1)^{\mathtt{i}u_k}\cos((y-a)u_l)dy \\
&= \int_{-1}^1 \frac{b-a}{2}\left(\exp\left(\frac{b-a}{2}y+\frac{a+b}{2}\right)+1\right)^{\mathtt{i}u_k}\cos\left(\left(\frac{b-a}{2}y+\frac{a+b}{2}-a\right)u_l\right)dy \\
&= \int_{-1}^1 f(y)dy
\end{aligned} \tag{10.36}$$

The integral can then be approximated as follows

$$\int_a^b (e^y+1)^{\mathtt{i}u_k}\cos((y-a)u_l)dy \approx (D^T d)^T y =: w^T y \tag{10.37}$$

where, denoting the number of quadrature points by n_q, D is an $(n_q/2+1)\times(n_q/2+1)$ matrix and

$$D(k,n) = \frac{2}{n_q}\cos\left(\frac{2(n-1)(k-1)\pi}{n_q}\right)\cdot\begin{cases}1/2 & \text{if } n=\{1,n_q/2+1\}\\ 1 & \text{otherwise}\end{cases} \tag{10.38}$$

$$d := \left(1,\frac{2}{(1-4)},\frac{2}{(1-16)},\dots,\frac{2}{(1-(n_q-2)^2)},\frac{1}{(1-n_q^2)}\right)^T \tag{10.39}$$

$$y=\{y_n\}_{n=0}^{n_q/2} \qquad y_n := f\left(\cos\left(\frac{n\pi}{n_q}\right)\right)+f\left(-\cos\left(\frac{n\pi}{n_q}\right)\right) \tag{10.40}$$

10.3 Numerical Example

Unfortunately, the study of Asian options in regime-switching is not widespread in literature yet, so there are no prices to compare our results with. Therefore, to check the correctness of the methods just described, we set up a Monte Carlo procedure as explained in the following section. In doing this, we take the two-state Black-Scholes model, since it has the simplest dynamics and we introduce Monte Carlo for comparison purposes only, so we want to keep a code as readable and fast as possible. However, provided to know the model dynamics, it can be generalized to any model that we presented in Chapter 1, and in principle even to a generic number J of states but procedure would become tediously long, both to implement and to run. As regards the COS method for the arithmetic Asian option, following the suggestions reported in [20], we set $n_q = \frac{25}{16}N$.
As parameters, we choose those obtained by the first calibration procedure described in Chapter 4, i.e.

$$\begin{aligned}
&r=0.019,\ \delta=0.012,\ T=1.192,\ S_0=1124.47,\ K=1125,\\
&\sigma_1=0.1486,\ \sigma_2=0.2115,\ q_{12}=1.512,\ q_{21}=1e-5
\end{aligned}$$

Moreover, we consider a weekly monitoring, $M = 52$, but different values for N. In reporting the results obtained, we focus on the prices in state 1 since they are the most significant, given that $q_{12} > 0$.

Table 10.1: Prices and cpu time for geometric and arithmetic Asian put in a two-regime Black-Scholes model with the COS method

	Geometric Asian Put		Arithmetic Asian Put	
N	Price	Time (s)	Price	Time (s)
32	40.8441	0.1226	41.2714	0.1744
64	40.4938	0.1304	39.3426	0.8831
128	40.4937	0.1366	39.3304	7.1326

As regards the Monte Carlo method, we choose a number of simulation equal to 10^5 obtaining

Table 10.2: Prices, 95% confidence intervals and cpu time for geometric and arithmetic Asian put in a two-regime Black-Scholes model with the MC method

	Price	95% C.I.	Time (s)
Geom. Asian Put	40.4907	[40.3169,40.6645]	5.5605
Arithm. Asian Put	39.3353	[39.1643,39.5064]	4.5367

We can notice that COS results are consistent with MC ones. Moreover, we can consider ourselves already satisfied with $N = 64$, indeed COS prices are inside the confidence intervals of MC and cpu times are really low, proving the efficiency of COS method. Finally, let us stress that if one were required to compute the price of a continuously monitored Asian option, one could apply the Richardson extrapolation, similarly to what was done for the Bermudan and barrier options.

Having checked the correctness of the method for Asian options under regime-switching, we can now consider a more interesting model than the basic 2RS Black-Scholes thus we take the usual 2RS Merton with the following parameters

$$\begin{aligned} &S_0 = 100,\ K = 100,\ T = 0.5,\ r = [0.05, 0.06]',\\ &\delta = [0.05, 0.07]',\ \sigma = [0.1, 0.2]',\ \tilde{\mu} = [0, 0]',\\ &\tilde{\sigma} = [0.3, 0.4]',\ \lambda = [0.1, 0.2]',\ q_{12} = q_{21} = 1 \end{aligned}$$

and we price Asian put options in state 2 with $M = 52$ monitoring dates.

Table 10.3: Prices and cpu time for geometric and arithmetic Asian put in a two-regime jump-diffusion Merton model with the COS method

	Geometric Asian Put		Arithmetic Asian Put	
N	Price	Time (s)	Price	Time (s)
32	4.2891057049	0.1394	4.4253507337	0.1642
64	3.9766991861	0.1453	3.8628489039	0.8904
128	3.9680721978	0.1587	3.8308815987	8.3996
256	3.9680718567	0.1615	3.8309342233	108.5708

We can see that as N increases, prices tend to converge and $N = 128$ can already be considered a good compromise between precision of the result and computational cost.

10.3.1 Monte Carlo under a two-regime model

In finance, Monte Carlo method can be used to numerically compute the expected value which appears in the risk-neutral valuation formula by relying on repeated random sampling. Indeed, it consists in simulating, a large number of times, the option payoff and then in taking the mean of the discounted payoffs in order to reproduce the expected value.

For convenience, we focus on the case of our interest: the two-state Black-Scholes model. Extending the discussion in Section 1.2 to regime-switching, we have that the underlying dynamics in state $i \in \mathcal{J}$ is given by

$$S_t = S_0 e^{X_t^i} \tag{10.41}$$

where

$$dX_t^i = \left(r_i - \delta_i - \frac{\sigma_i^2}{2}\right)dt + \sigma_i dW_t \tag{10.42}$$

Integrating both terms between two subsequent monitoring dates, we get

$$\begin{aligned}
\int_{t_{m-1}}^{t_m} dX_s^i &= \int_{t_{m-1}}^{t_m} \left(r_i - \delta_i - \frac{\sigma_i^2}{2}\right)ds + \int_{t_{m-1}}^{t_m} \sigma_i dW_s \\
X_{t_m}^i - X_{t_{m-1}}^i &= \left(r_i - \delta_i - \frac{\sigma_i^2}{2}\right)\Delta t + \sigma_i (W_{t_m} - W_{t_{m-1}}) \\
&\sim \left(r_i - \delta_i - \frac{\sigma_i^2}{2}\right)\Delta t + \sigma_i \sqrt{\Delta t} Z_m
\end{aligned} \tag{10.43}$$

with $\{Z_m\}_{m=1,\dots,M}$ standard normals i.i.d. and discount factor $D(t_{m-1}, t_m) = \exp(-r_i \cdot (t_m - t_{m-1}))$.

In order to understand which state i to consider, we have to determine the time

instants in which regime switches occur. At the end of Section 2.1.1 we pointed out that the Markov chain remains in each state i for a period of time $\mathcal{T}_i \sim \mathcal{E}(-q_{ii})$. These $\mathcal{T}_i$ may not coincide with the monitoring dates t_m, and thus for each $\mathcal{T}_i$ we identify m^* such that $t_{m^*-1} < \mathcal{T}_i < t_{m^*}$.
Hence, assuming w.l.o.g. that initially we are in state 1, we draw $\mathcal{T}_1$ exponentially distributed and implement equation (10.43) for state 1 until t_{m^*-1}, then we divide $\Delta t = t_{m^*} - t_{m^*-1}$ into two parts, $\Delta t_1 = \mathcal{T}_1 - t_{m^*-1}$ and $\Delta t_2 = t_{m^*} - \mathcal{T}_1$, resulting in

$$
\begin{aligned}
X^1_{\mathcal{T}_1} - X^1_{t_{m^*-1}} &\sim \left(r_1 - \delta_1 - \frac{\sigma_1^2}{2}\right)\Delta t_1 + \sigma_1\sqrt{\Delta t_1}Z_{m^*} \\
X^1_{t_{m^*}} - X^1_{\mathcal{T}_1} &\sim \left(r_2 - \delta_2 - \frac{\sigma_2^2}{2}\right)\Delta t_2 + \sigma_2\sqrt{\Delta t_2}Z_{\text{switch}} \\
D(t_{m^*-1}, t_{m^*}) &= e^{-r_1\cdot(\mathcal{T}_1 - t_{m^*-1}) - r_2\cdot(t_{m^*} - t_{m^*-1})}
\end{aligned}
$$

with Z_{switch} standard normal.
Then, the same procedure is repeated until we reach T and all the algorithm just described is iterated for a great number of times, typically $N_{\text{sim}} = 10^5$.
Having simulated the dynamics of the Lévy process in each time step Δt and for each iteration $n \in \{1, \dots, N_{\text{sim}}\}$, we can now compute the Monte Carlo price as

$$
\frac{1}{N_{\text{sim}}} \sum_{n=1}^{N_{\text{sim}}} D^n \left[\alpha \cdot (A^n(T) - K)\right]^+ \tag{10.44}
$$

where $A^n(T)$ is the nth geometric/arithmetic average and

$$
D^n = \prod_{m=1}^{M} D^n(t_{m-1}, t_m) \tag{10.45}
$$

Antithetic variables

Now we want to go one step further with the Monte Carlo method, considering a technique of variance reduction: the so called *Antithetic Variables technique* (AV). Indeed from theory we know that the error for the MC is an estimation of the unbiased standard deviation of the MC price, so if we manage to reduce this standard deviation we will have a smaller error and thus a better precision of the price estimate. The basic idea behind this method is that, in order to estimate $\mathbb{E}[X]$ with X being a random variable with law f, one must consider other two random variables X_1, X_2 distributed as X and then take Y defined as

$$
Y := \frac{X_1 + X_2}{2} \tag{10.46}
$$

So one gets

$$\mathbb{E}[Y] = \frac{\mathbb{E}[X_1] + \mathbb{E}[X_2]}{2} = \mathbb{E}[X] \tag{10.47}$$

$$\mathbb{V}\text{ar}[Y] = \frac{1}{4}\,\mathbb{V}\text{ar}[X_1] + \frac{1}{4}\,\mathbb{V}\text{ar}[X_2] + \frac{1}{2}\,\mathbb{C}\text{ov}(X_1, X_2) = \frac{1}{2}\Big(\mathbb{V}\text{ar}[X] + \mathbb{C}\text{ov}(X_1, X_2)\Big) \tag{10.48}$$

And if we take X_1, X_2 s.t. $\mathbb{C}\text{ov}(X_1, X_2) < 0$, from equation (10.48) we obtain

$$\mathbb{V}\text{ar}[Y] \leq \mathbb{V}\text{ar}[X]$$

Thus it is clear that is more convenient to estimate $\mathbb{E}[Y]$, instead of $\mathbb{E}[X]$.
In our case we consider X being the discounted payoff. Then, in order to satisfy the following conditions

$$\begin{cases} X_1, X_2 \sim X \\ \mathbb{C}\text{ov}(X_1, X_2) < 0 \end{cases}$$

we take $X_1 = X$ and we construct X_2 starting from X but considering all the random samples from the normal distribution with opposite sign.

Conclusion

In this dissertation, we studied the problem of option pricing under a regime-switching framework using the COS method. This method, based on the Fourier cosine series coefficients, can be used for a various set of options, of which we have covered only a few of the most popular in literature. Before describing its application to exotic options, we set up two calibration procedures with quoted prices of vanilla options, since they are the most liquid contracts. This is done to know the parameters of the various models considered and thanks to the calibrations we found out that regime-switching time-changed Lévy models are the best suited to describe market data. However, if these models account for several regime switches, they will involve a large number of parameters making the method slower and less accurate. For such a reason, we limited our numerical tests to the case of a maximum of 2 different regimes. This choice led us to another advantage. In order to apply the COS formula, the characteristic function for the underlying price process must be known and in general its derivation is not immediate since it requires a matrix exponentiation. But in the two-state case we have an analytical formula available. Using the calibrated parameters, we proceeded to the pricing of the various options, including the continuous path dependent options, such as American and continuous barrier options, for which we used the s-stage Richardson extrapolation reaching a high accuracy. Finally, the method confirmed to outperform the most popular methods in literature, showing excellent convergence properties.

Appendix A

Cumulants

Table A.1: Cumulants

Model	Cumulants
BS	$c_1 = \mu T$
	$c_2 = \sigma^2 T$
	$c_4 = 0$
Merton	$c_1 = (\mu + \lambda\tilde{\mu})T$
	$c_2 = (\sigma^2 + \lambda\tilde{\mu}^2 + \lambda\tilde{\sigma}^2)T$
	$c_4 = \lambda(\tilde{\mu}^4 + 6\tilde{\sigma}^2\tilde{\mu}^2 + 3\tilde{\sigma}^4)T$
Kou	$c_1 = \left(\mu + \lambda\left(\frac{p}{\lambda_+} - \frac{1-p}{\lambda_-}\right)\right)T$
	$c_2 = \left(\sigma^2 + 2\lambda\left(\frac{p}{\lambda_+^2} + \frac{1-p}{\lambda_-^2}\right)\right)T$
	$c_4 = 24\lambda\left(\frac{p}{\lambda_+^4} + \frac{1-p}{\lambda_-^4}\right)T$
VG	$c_1 = (\mu + \theta)T$
	$c_2 = (\sigma^2 + \nu\theta^2)T$
	$c_4 = 3(\sigma^4\nu + 2\theta^4\nu^3 + 4\sigma^2\theta^2\nu^2)T$
NIG	$c_1 = (\mu + \gamma\beta(\omega^2 - \beta^2)^{-\frac{1}{2}})T$
	$c_2 = \gamma\omega^2(\omega^2 - \beta^2)^{-\frac{3}{2}}T$
	$c_4 = 3\gamma\omega^2(\omega^2 + 4\beta^2)(\omega^2 - \beta^2)^{-\frac{7}{2}}T$
CGMY	$c_1 = \mu T + CT\Gamma(1-Y)(M^{Y-1} - G^{Y-1})$
	$c_2 = CT\Gamma(2-Y)(M^{Y-2} + G^{Y-2})$
	$c_4 = CT\Gamma(4-Y)(M^{Y-4} + G^{Y-4})$
Meixner	$c_1 = (\mu + \eta\alpha\tan(\beta/2))T$
	$c_2 = \frac{\eta\alpha^2}{2}(\cos(\beta/2))^{-2}T$
	$c_4 = \frac{\eta\alpha^4}{4}(\cos(\beta/2))^{-2}(1 + 3(\tan(\beta/2))^2)T$

Model	Cumulants
Heston	$c_1 = (r-\delta)T + \frac{\theta - V_0}{2\kappa}(1-e^{-\kappa T}) - \frac{1}{2}\theta T$
	$c_2 = \frac{1}{8\kappa^3}(\kappa\epsilon T e^{-\kappa T}(V_0-\theta)(8\kappa\rho - 4\epsilon) + 8\kappa\rho\epsilon(1-e^{-\kappa T})(2\theta - V_0)$
	$+2\kappa\theta T(-4\kappa\rho\epsilon + \epsilon^2 + 4\kappa^2) + \epsilon^2((\theta - 2V_0)e^{-2\kappa T}$
	$+\theta(6e^{-\kappa T} - 7) + 2V_0) + 8\kappa^2(V_0 - \theta)(1 - e^{-\kappa T}))$
	$c_4 = 0$

Appendix B

Call Option Prices

In the following table, taken from [16], we report the data used in the first calibration procedure. Specifically, 75 call option prices on the S&P 500 Index at the close of the market on 18 April 2002. On that day, the S&P 500 Index closed at 1124.47 with $r = 1.9\%$ and $\delta = 1.2\%$.

Table B.1: Call option prices on the S&P 500 index closed on 18 April 2002

Strike	May 2002 T=0.088	June 2002 T=0.184	Sep. 2002 T=0.436	Dec. 2002 T=0.692	March 2003 T=0.936	June 2003 T=1.192	Dec. 2003 T=1.708
975			161.60	173.30			
995			144.80	157.00		182.10	
1025			120.10	133.10	146.50		
1050		84.50	100.70	114.80		143.00	171.40
1075		64.30	82.50	97.60			
1090	43.10						
1100	35.60		65.50	81.20	96.20	111.30	140.40
1110		39.50					
1120	22.90	33.50					
1125	20.20	30.70	51.00	66.90	81.70	97.00	
1130		28.00					
1135		25.60	45.50				
1140	13.30	23.20		58.90			
1150		19.10	38.10	53.90	68.30	83.30	112.80
1160		15.30					
1170		12.10					
1175		10.90	27.70	42.50	56.60		99.80

Strike	May 2002 T=0.088	June 2002 T=0.184	Sep. 2002 T=0.436	Dec. 2002 T=0.692	March 2003 T=0.936	June 2003 T=1.192	Dec. 2003 T=1.708
1200			19.60	33.00	46.10	60.90	
1225			13.20	24.90	36.90	49.80	
1250				18.30	29.30	41.20	66.90
1275				13.20	22.50		
1300					17.20	27.10	49.50
1325					12.80		
1350						17.10	35.70
1400						10.10	25.20
1450							17.00
1500							12.20

The next table, instead, contains the data used in the second calibration procedure. It has been downloaded from Thomson Reuters website and presents 59 call option prices on the AAPL stock at the close of the market on 5 November 2020. On that day, the AAPL stock closed at 114.95 with $r = 0.15\%$ and $\delta = 0.71\%$.

Table B.2: Call option prices on the AAPL stock closed on 5 November 2020

Strike	Nov. 2020 T=0.041	Dec. 2020 T=0.118	March 2021 T=0.367	June 2021 T=0.616	Sep. 2021 T=0.866	Jan. 2022 T=1.211	June 2022 T=1.614
87.5		27.75				33.15	
90				28.33		31.48	
92.5				26.44			
95		20.53		24.47			
97.5	17.65	18.42		23.20			
100	15.30	16.25	19.58	21.70	23.10	25.25	27.52
102.5	12.85	14.20					
105	10.75	12.20				22.47	
107.5		10.22					
110	6.80	8.70	12.96	15.45		19.85	
112.5	5.10			14.35		18.64	
115	3.65		10.30	13.05			
117.5	2.50						

Strike	Nov. 2020 T=0.041	Dec. 2020 T=0.118	March 2021 T=0.367	June 2021 T=0.616	Sep. 2021 T=0.866	Jan. 2022 T=1.211	June 2022 T=1.614
120		3.55	8.20		13.10	15.55	18.20
125		2.05	6.51	9.00		13.71	
130		1.14		7.57		12.05	
135			3.80	6.13		10.65	
140				5.05	7.05	9.45	
145			2.20			8.11	
150				3.40		7.30	
155				2.86			
160			1.04				

Appendix C

Matlab codes

C.1 Auxiliary functions

C.1.1 Characteristic Exponents

```
function psi=CharExp(model,u,T,r,delta,param)
 % INPUTS
 % model:              Chosen model of the underlying dynamics
 % u:                  Valuation point of the characteristic exponent
 % T:                  Time-to-maturity
 % r:                  Risk-free interest rate
 % delta:              Dividend yield
 % param:              Vector with the parameters of the model
 %
 % FUNCTIONS
 % CharExpAux:         Given psi without drift, it returns psi(u,T)
 % CharExpCIR:         Given psi, it applies the time change
 %                     with CIR clock
 % CharExpGOU:         Given psi, it applies the time change
 %                     with Gamma OU clock
 % CharExpHeston:      Returns directly psi(u,T) for the Heston model
 %
 % OUTPUT
 % psi:                Characteristic exponent in the form psi(u,T),
 %                     s.t. the corresponding characteristic function
 %                     can be written as: phi(u)=exp(psi(u,T))

 switch(model)
     case 'BS'
         sigma=param;
         psi=@(u) -sigma^2/2*u.^2;
         psi=CharExpAux(psi,u,T,r,delta);
```

```
28      case 'Merton'
29          sigma=param(1); mu_tilde=param(2);
30          sigma_tilde=param(3); lambda=param(4);
31          psi=@(u) -sigma^2/2*u.^2+lambda* ...
32              (exp(1i*u*mu_tilde-sigma_tilde^2*u.^2/2)-1);
33          psi=CharExpAux(psi,u,T,r,delta);
34      case 'Kou'
35          sigma=param(1); p=param(2); lambdap=param(3);
36          lambdam=param(4); lambda=param(5);
37          psi=@(u) -sigma^2/2*u.^2+1i*u*lambda.* ...
38              (p./(lambdap-1i*u)-(1-p)./(lambdam+1i*u));
39          psi=CharExpAux(psi,u,T,r,delta);
40      case {'VG','VG_CIR','VG_GOU'}
41          sigma=param(1); nu=param(2); theta=param(3);
42          psi=@(u) (-1/nu)*log(1-1i*u*theta*nu+sigma^2*nu*u.^2/2);
43          if strcmp(model,'VG')
44              psi=CharExpAux(psi,u,T,r,delta);
45          elseif strcmp(model,'VG_CIR')
46              k=param(4); eta=param(5); lambda=param(6);
47              psi=CharExp_CIR(psi,u,T,r,delta,k,eta,lambda);
48          else
49              lambda=param(4); a1=param(5); a2=param(6);
50              psi=CharExp_GOU(psi,u,T,r,delta,lambda,a1,a2);
51          end
52      case {'NIG','NIG_CIR','NIG_GOU'}
53          omega=param(1); beta=param(2); gammaNIG=param(3);
54          psi=@(u) gammaNIG*(sqrt(omega^2-beta^2)- ...
55              sqrt(omega^2-(beta+1i*u).^2));
56          if strcmp(model,'NIG')
57              psi=CharExpAux(psi,u,T,r,delta);
58          elseif strcmp(model,'NIG_CIR')
59              k=param(4); eta=param(5); lambda=param(6);
60              psi=CharExp_CIR(psi,u,T,r,delta,k,eta,lambda);
61          else
62              lambda=param(4); a1=param(5); a2=param(6);
63              psi=CharExp_GOU(psi,u,T,r,delta,lambda,a1,a2);
64          end
65      case {'CGMY','CGMY_CIR','CGMY_GOU'}
66          C=param(1); G=param(2); M=param(3); Y=param(4);
67          psi=@(u) C*gamma(-Y)*((M-1i*u).^Y-M^Y+(G+1i*u).^Y-G^Y);
68          if strcmp(model,'CGMY')
69              psi=CharExpAux(psi,u,T,r,delta);
70          elseif strcmp(model,'CGMY_CIR')
71              k=param(5); eta=param(6); lambda=param(7);
72              psi=CharExp_CIR(psi,u,T,r,delta,k,eta,lambda);
73          else
```

```
74              lambda=param(5); a1=param(6); a2=param(7);
75              psi=CharExp_GOU(psi,u,T,r,delta,lambda,a1,a2);
76          end
77      case {'Meixner','Meixner_CIR','Meixner_GOU'}
78          alpha=param(1); beta=param(2); eta=param(3);
79          psi=@(u) 2*eta*log(cos(beta/2)./ ...
80              cosh((alpha*u-1i*beta)./2));
81          if strcmp(model,'Meixner')
82              psi=CharExpAux(psi,u,T,r,delta);
83          elseif strcmp(model,'Meixner_CIR')
84              k=param(4); etaCIR=param(5); lambda=param(6);
85              psi=CharExp_CIR(psi,u,T,r,delta,k,etaCIR,lambda);
86          else
87              lambda=param(4); a1=param(5); a2=param(6);
88              psi=CharExp_GOU(psi,u,T,r,delta,lambda,a1,a2);
89          end
90      case 'Heston'
91          psi=CharExp_Heston(u,T,r,delta,param(1),param(2), ...
92              param(3),param(4),param(5));
93      otherwise
94          warning('method not found')
95  end
96
97  end
98  %% List of Characteristic Exponents
99  function psi=CharExpAux(psi,u,T,r,delta)
100  psi_u=psi(u);
101  psi_i=-psi(-1i);
102  mu=r-delta+psi_i;
103  psi=(1i*u*mu+psi_u)*T;
104 end
105
106 function psi=CharExp_Heston(u,T,r,delta,V0,theta,k,epsilon,rho)
107  alfa=-0.5*(u.*u+u*1i);
108  beta=k-rho*epsilon*u*1i;
109  epsilon2=epsilon*epsilon;
110  gamma=0.5*epsilon2;
111  D=sqrt(beta.*beta-4.0*alfa.*gamma);
112  bD=beta-D;
113  eDt=exp(-D*T);
114  G=bD./(beta+D);
115  B=(bD./epsilon2).*((1.0-eDt)./(1.0-G.*eDt));
116  psi=(G.*eDt-1.0)./(G-1.0);
117  A=((k*theta)/(epsilon2))*(bD*T-2.0*log(psi));
118  psi=A+B*V0+1i*u*((r-delta)*T);
119 end
```

```
120
121 function psi=CharExp_CIR(psi,u,T,r,delta,k,eta,lambda)
122  psi_u=-1i*psi(u);
123  psi_i=-1i*psi(-1i);
124  y0=1;
125  gamma_u=sqrt(k^2-2*lambda^2*1i*psi_u);
126  gamma_i=sqrt(k^2-2*lambda^2*1i*psi_i);
127  log_phi_u=k^2*eta*T*lambda^(-2)+2*y0*1i*psi_u./ ...
128      (k+gamma_u.*coth(gamma_u*T/2))-log(cosh(gamma_u*T/2)+ ...
129      k*sinh(gamma_u*T/2)./gamma_u)*(2*k*eta*lambda^(-2));
130  log_phi_i=k^2*eta*T*lambda^(-2)+2*y0*1i*psi_i./ ...
131      (k+gamma_i*coth(gamma_i*T/2))-log(cosh(gamma_i*T/2)+ ...
132      k*sinh(gamma_i*T/2)/gamma_i)*(2*k*eta*lambda^(-2));
133  psi=1i*u*((r-delta)*T-log_phi_i)+log_phi_u;
134 end
135
136 function psi=CharExp_GOU(psi,u,T,r,delta,lambda,a1,a2)
137  psi_u=-1i*psi(u);
138  psi_i=-1i*psi(-1i);
139  y0=1;
140  log_phi_u=1i*psi_u*y0*lambda^(-1)*(1-exp(-lambda*T))+ ...
141      a1*lambda./(1i*psi_u-lambda*a2).*(a2*log(a2./(a2-1i*psi_u* ...
142      lambda^(-1)*(1-exp(-lambda*T))))-1i*psi_u*T);
143  log_phi_i=1i*psi_i*y0*lambda^(-1)*(1-exp(-lambda*T))+ ...
144      a1*lambda./(1i*psi_i-lambda*a2).*(a2*log(a2./(a2-1i*psi_i* ...
145      lambda^(-1)*(1-exp(-lambda*T))))-1i*psi_i*T);
146  psi=1i*u*((r-delta)*T-log_phi_i)+log_phi_u;
147 end
```

C.1.2 Cumulants

```
1  function c=ComputeCumulants(model,T,r,delta,param)
2   % INPUTS
3   % model:              Chosen model of the underlying dynamics
4   % T:                  Time-to-maturity
5   % r:                  Risk-free interest rate
6   % delta:              Dividend yield
7   % param:              Vector with the parameters of the model
8   %
9   % OUTPUT
10  % c:                  Vector with the 1st, 2nd and 4th cumulants
11
12  switch (model)
13      case 'BS'
14          sigma=param;
```

```
15          mu=r-delta-sigma^2/2;
16          c1=mu*T;
17          c2=sigma^2*T;
18          c4=0;
19      case 'Merton'
20          sigma=param(1); mu_tilde=param(2);
21          sigma_tilde=param(3); lambda=param(4);
22          mu=r-delta-sigma^2/2-lambda* ...
23              (exp(mu_tilde+sigma_tilde^2/2)-1);
24          c1=(mu+lambda*mu_tilde)*T;
25          c2=(sigma^2+lambda*mu_tilde^2+lambda*sigma_tilde^2)*T;
26          c4=lambda*(mu_tilde^4+6*sigma_tilde^2*mu_tilde^2+ ...
27              3*sigma_tilde^4)*T;
28      case 'Kou'
29          sigma=param(1); p=param(2); lambdap=param(3);
30          lambdam=param(4); lambda=param(5);
31          mu=r-delta-sigma^2/2-lambda* ...
32              (p/(lambdap-1)-(1-p)/(lambdam+1));
33          c1=(mu+lambda*(p/lambdap-(1-p)/lambdam))*T;
34          c2=(sigma^2+2*lambda*(p/(lambdap^2)+(1-p)/(lambdam^2)))*T;
35          c4=24*lambda*(p/(lambdap^4)+(1-p)/(lambdam^4))*T;
36      case 'VG'
37          sigma=param(1); nu=param(2); theta=param(3);
38          mu=r-delta+(1/nu)*log(1-theta*nu-nu*sigma^2/2);
39          c1=(mu+theta)*T;
40          c2=(sigma^2+nu*theta^2)*T;
41          c4=3*(sigma^4*nu+2*theta^4*nu^3+4*sigma^2*theta^2*nu^2)*T;
42      case 'NIG'
43          omega=param(1); beta=param(2); gammaNIG=param(3);
44          mu=r-delta-gammaNIG*(sqrt(omega^2-beta^2)- ...
45              sqrt(omega^2-(beta+1)^2));
46          c1=(mu+gammaNIG*beta/sqrt(omega^2-beta^2))*T;
47          c2=gammaNIG*omega^2*(omega^2-beta^2)^(-3/2)*T;
48          c4=3*gammaNIG*omega^2*(omega^2+4*beta^2)* ...
49              (omega^2-beta^2)^(-7/2)*T;
50      case 'CGMY'
51          C=param(1); G=param(2); M=param(3); Y=param(4);
52          mu=r-delta-C*gamma(-Y)*((M-1)^Y-M^Y+(G+1)^Y-G^Y);
53          c1=mu*T+C*T*gamma(1-Y)*(M^(Y-1)-G^(Y-1));
54          c2=C*T*gamma(2-Y)*(M^(Y-2)+G^(Y-2));
55          c4=C*T*gamma(4-Y)*(M^(Y-4)+G^(Y-4));
56      case 'Meixner'
57          alpha=param(1); beta=param(2); eta=param(3);
58          mu=r-delta-2*eta*log(cos(beta/2)/cos((alpha+beta)/2));
59          c1=mu*T+eta*alpha*tan(beta/2)*T;
60          c2=eta*T*alpha^2/(2*(cos(beta/2))^2);
```

```
        c4=eta*T*alpha^4/(4*(cos(beta/2))^2)*(1+3*(tan(beta/2))^2);
    case 'Heston'
        V0=param(1); theta=param(2); k=param(3);
        epsilon=param(4); rho=param(5);
        c1=(r-delta)*T+0.5*((1-exp(-k*T))* ...
            (theta-V0)/k-theta*T);
        c2=1/(8*k^3)*(k*epsilon*T*exp(-k*T)* ...
            (V0-theta)*(8*k*rho-4*epsilon) ...
            +8*k*rho*epsilon*(1-exp(-k*T))*(2*theta-V0) ...
            +2*k*theta*T*(-4*k*rho*epsilon+epsilon^2+4*k^2) ...
            +epsilon^2*((theta-2*V0)*exp(-2*k*T) ...
            +theta*(6*exp(-k*T)-7)+2*V0) ...
            +8*k^2*(V0-theta)*(1-exp(-k*T)));
        c4=0;
    otherwise    % use finite difference
        h=1e-2;
        psi=@(u) CharExp(model,-1i*u,T,r,delta,param);
        f=psi(0);
        fh=psi(h);
        f_h=psi(-h);
        f2h=psi(2*h);
        f_2h=psi(-2*h);
        f3h=psi(3*h);
        f_3h=psi(-3*h);
        c1=(fh-f_h)/(2*h);
        c2=(fh-2*f+f_h)/(h^2);
        c4=(f3h-2*f2h+4*f-fh-f_h-2*f_2h+f_3h)/(4*h^4);
end
c=[c1,c2,c4];

end
```

C.1.3 Functions χ_k, ξ_k

```
function [chi,xi]=CosineCoeff(k,c,d,a,b)
% INPUTS
% k:                Grid index
% [c,d]:            Interval of interest
% [a,b]:            Truncated domain
%
% OUTPUTS
% chi,xi:           Auxiliar functions needed in the computation
%                   of the Fourier cosine series coefficients

u=k*pi/(b-a);
```

```
 x1=(d-a)*u;
 x2=(c-a)*u;
 xi=(sin(x1)-sin(x2))./u;
 xi(1)=d-c;
 chi=(cos(x1)*exp(d)-cos(x2)*exp(c)+ ...
     u.*(sin(x1)*exp(d)-sin(x2)*exp(c)))./(1+u.^2);

end
```

C.1.4 Coefficients $\bar{V}_k$

```
function Vk=VkVanilla(k,x1,x2,a,b,K,alpha)
 % INPUTS
 % k:                Grid index
 % [x1,x2]:          Exercise interval
 % [a,b]:            Truncated domain
 % K:                Strike price
 % alpha:            1 for a Call, -1 for a Put
 %
 % FUNCTION
 % CosineCoeff:      Function that returns the auxiliar functions
 %                   chi,xi needed to compute Vk
 %
 % OUTPUT
 % Vk:               Fourier cosine series coefficients
 %                   for the Vanilla Option

 [chi,xi]=CosineCoeff(k,x1,x2,a,b);
 Vk=2/(b-a)*K*alpha*(chi-xi);

end
```

```
function Vk=VkDigital(k,x1,x2,a,b,K)
 % INPUTS
 % k:                Grid index
 % [x1,x2]:          Exercise interval
 % [a,b]:            Truncated domain
 % K:                Strike price
 %
 % FUNCTION
 % CosineCoeff:      Function that returns the auxiliar functions
 %                   chi,xi needed to compute Vk
 %
 % OUTPUT
 % Vk:               Fourier cosine series coefficients
```

```
 %                    for the Digital option

 [~,xi]=CosineCoeff(k,x1,x2,a,b);
 Vk=2/(b-a)*K*xi;

end
```

```
function Vk=VkButterfly(k,x1,x2,x3,x,a,b,K1,K2,K3,p,alpha)
 % INPUTS
 % k:                 Grid index
 % [x1,x2,x3,x]:      Exercise points
 % [a,b]:             Truncated domain
 % [K1,K2,K3]:        Strike prices
 % p:                 Scale parameter
 % alpha:             1 for a Call, -1 for a Put
 %
 % FUNCTION
 % CosineCoeff:       Function that returns the auxiliar functions
 %                    chi,xi needed to compute Vk
 %
 % OUTPUT
 % Vk:                Fourier cosine series coefficients
 %                    for the Butterfly Option

 if alpha==1
     [chi1,xi1]=CosineCoeff(k,x1,x,a,b);
     [chi2,xi2]=CosineCoeff(k,x2,x,a,b);
     [chi3,xi3]=CosineCoeff(k,x3,x,a,b);
 else
     [chi1,xi1]=CosineCoeff(k,x,x1,a,b);
     [chi2,xi2]=CosineCoeff(k,x,x2,a,b);
     [chi3,xi3]=CosineCoeff(k,x,x3,a,b);
 end
 Vk=2/(b-a)*alpha*(p*chi1-K1*xi1- ...
     2*(p*chi2-K2*xi2)+p*chi3-K3*xi3);

end
```

```
function Vk=VkAsianGeom(k,x1,x2,a,b,K,alpha)
 % INPUTS
 % k:                 Grid index
 % [x1,x2]:           Exercise interval
 % [a,b]:             Truncated domain
 % K:                 Strike price
 % alpha:             1 for a Call, -1 for a Put
 %
```

```
 % FUNCTION
 % CosineCoeff:        Function that returns the auxiliar functions
 %                     chi,xi needed to compute Vk
 %
 % OUTPUT
 % Vk:                 Fourier cosine series coefficients
 %                     for the Asian Option

 [chi,xi]=CosineCoeff(k,x1,x2,a,b);
 Vk=2/(b-a)*alpha*(chi-K*xi);

end
```

```
function Vk=VkAsianArithm(k,x1,x2,a,b,K,S0,M,alpha)
 % INPUTS
 % k:                  Grid index
 % [x1,x2]:            Exercise interval
 % [a,b]:              Truncated domain
 % K:                  Strike price
 % S0:                 Spot price
 % M:                  Number of monitoring dates
 % alpha:              1 for a Call, -1 for a Put
 %
 % FUNCTION
 % CosineCoeff:        Function that returns the auxiliar functions
 %                     chi,xi needed to compute Vk
 %
 % OUTPUT
 % Vk:                 Fourier cosine series coefficients
 %                     for the arithmetic Asian Option

 [chi,xi]=CosineCoeff(k,x1,x2,a,b);
 Vk=2/(b-a)*alpha*(S0/(M+1)*chi-(K-S0/(M+1))*xi);

end
```

C.1.5 Computation of $\bar{\mathcal{H}}^i$

```
function H=ComputeH(x1,x2,a,b,N,u)
 % INPUTS
 % [x1,x2]:            Interval of interest
 % [a,b]:              Truncated domain
 % N:                  Number of grid points
 % u:                  Lambdak
 %
```

```
% OUTPUT
% H:                  Continuation value

exp2=exp(1i*(1:N)'*(x2-a)/(b-a)*pi);
exp1=exp(1i*(1:N)'*(x1-a)/(b-a)*pi);
m=zeros(3*N-1,1);                   % m=[m_{1-N},...,m_{N-1},
                                    %      m_{N},...,m_{2N-1}]
m(N)=(x2-x1)/(b-a)*pi*1i;           % m_0
m(N+1:2*N)=(exp2-exp1)./(1:N)';     % [m_1,...,m_{N-1},m_N]
m(1:N-1)=-conj(flipud(m(N+1:2*N-1)));
m(2*N+1:3*N-1)=(exp2(N)*exp2(1:N-1)-exp1(N)*exp1(1:N-1))./ ...
    (N+1:2*N-1)';
m_s=[m(N:-1:1);0;m(2*N-1:-1:N+1)]; % m_s=[m_0,m_{-1},...,m_{1-N},
                                    %       0,m_{N-1},...,m_1]'
m_c=m(3*N-1:-1:N);                  % m_c=[m_{2N-1},...,m_1,m_0]'

u_s=[u;zeros(N,1)];                 % u_s=[u_0,u_1,...,u_{N-1},
                                    %       0,...,0]'
sgn=ones(2*N,1);
sgn(2*(1:N))=-1;
xi_s=ifft((fft(m_s)).*fft(u_s));
Msu=xi_s(1:N);
xi_c=ifft((fft(m_c)).*sgn.*fft(u_s));
Mcu=flipud(xi_c(1:N));
H=1/pi*imag(Msu+Mcu);

end
```

C.1.6 Computation of $\mathcal{M}$

```
function Mkl=ComputeM(a,b,uk,ul,nq)
% INPUTS
% [a,b]:              Truncated domain
% uk:                 k*pi/(b-a)
% ul:                 l*pi/(b-a)
% nq:                 number of quadrature points
%
% OUTPUT
% Mkl:                M(k,l)=\int_a^b (e^y+1)^{1i*uk} cos((y-a)ul)dy
%                     using Clenshaw-Curtis quadrature

dim=nq/2+1;
n=(0:dim-1)';
D=2/nq*cos(n*n'*2*pi/nq);
D(:,1)=D(:,1)*0.5;
```

```
 D(:,dim)=D(:,dim)*0.5;
 nn=2:2:nq;
 d=2./(1-nn.^2);
 d=[1;d(1:end-1)';d(end)*0.5];
 f=@(x) (b-a)/2*(exp((b-a)/2.*x+(a+b)/2)+1).^(1i*uk).* ...
     cos(((b-a)/2.*x+(a+b)/2-a)*ul);
 y=f(cos(n*pi/nq))+f(-cos(n*pi/nq));
 Mkl=(D'*d)'*y;

end
```

C.1.7 Put-Call parity

```
function Prices=PutCall_parity(J,Prices,S0,T,r,delta,K)
 % INPUTS
 % J:                Number of states
 % Prices:           Put option prices in the J states
 % S0:               Spot price
 % T:                Time-to-maturity
 % r:                Risk-free interest rate constant
 %                   for all the J states
 % delta:            Dividend yield constant
 %                   for all the J states
 % K:                Strike price
 %
 % OUTPUT
 % Prices:           Call option prices obtained with PC parity

 for j=1:J
     Prices(j)=Prices(j)+S0*exp(-delta*T)-K*exp(-r*T);
 end

end
```

C.1.8 Characteristic function under 2 regimes switching

```
function [phi11,phi12,phi21,phi22]=CharFun_2RS(models,u,T,r, ...
    delta,Q,params)
 % INPUTS
 % models:           String vector containing the chosen models
 %                   of the underlying dynamics in the two states
 % u:                Valuation point
 % T:                Time-to-maturity
```

```
% r:                Vector with the risk-free interest rates
%                   in the 2 states
% delta:            Vector with the dividend yields
%                   in the 2 states
% Q:                Intensity matrix
% params:           Cell array with the parameters of
%                   [model1,model2]
%
% FUNCTION
% CharExp:          Returns the characteristic exponent psi(u,T)
%
% OUTPUTS
% phiij             i,j in [1,2]. They are vectors in order to get
%                   phi=[diag(phi11),diag(phi12); ...
%                   diag(phi21),diag(phi22)]

psi1_tilde=1/T*CharExp(models(1),u,T,r(1),delta(1), ...
    params{1})-r(1)+Q(1,1);
psi2_tilde=1/T*CharExp(models(2),u,T,r(2),delta(2), ...
    params{2})-r(2)+Q(2,2);
s1=1/2*(psi1_tilde+psi2_tilde+sqrt(psi1_tilde.^2 ...
        -2*psi1_tilde.*psi2_tilde+psi2_tilde.^2+ ...
        4*Q(1,2)*Q(2,1)));
s2=1/2*(psi1_tilde+psi2_tilde-sqrt(psi1_tilde.^2 ...
        -2*psi1_tilde.*psi2_tilde+psi2_tilde.^2+ ...
        4*Q(1,2)*Q(2,1)));
phi11=exp(s2*T)+(exp(s2*T)-exp(s1*T))./(s2-s1).* ...
    (psi1_tilde-s2);
phi12=(exp(s2*T)-exp(s1*T))./(s2-s1)*Q(1,2);
phi21=(exp(s2*T)-exp(s1*T))./(s2-s1)*Q(2,1);
phi22=exp(s2*T)+(exp(s2*T)-exp(s1*T))./(s2-s1).* ...
    (psi2_tilde-s2);

end
```

C.1.9 Black-Scholes dynamics under 2 regimes switching

```
function [XT,XTAV,D]=simulateBS_2RS(Nsim,M,T,r,delta,Q,sigma,i)
% INPUTS
% Nsim:             Number of simulations
% M:                Number of time steps
% T:                Time-to-maturity
% r:                Vector with the risk-free interest rates
%                   in the 2 states
% delta:            Vector with the dividend yields
```

```
%                   in the 2 states
% Q:                Intensity matrix
% sigma:            Cell array with the volatilities of
%                   [model1,model2]
% i:                Initial regime state
%
% OUTPUTS
% XT:               Lévy process under 2RS Black & Scholes
% XTAV:             Antithetic variable of XT
% D:                Total discount factor

dT=T/M;
Z=randn(Nsim,M);
XT=zeros(Nsim,M+1);
XTAV=zeros(Nsim,M+1);
D=ones(Nsim,1);
drift=zeros(2,1);
drift(1)=r(1)-delta(1)-sigma{1}^2/2;
drift(2)=r(2)-delta(2)-sigma{2}^2/2;
for n=1:Nsim
    k=i;
    waitingTime=exprnd(1/abs(Q(k,k)));
    for m=1:M
        if waitingTime<=m*dT && waitingTime>(m-1)*dT
            t=waitingTime-(m-1)*dT;
            XT(n,m+1)=XT(n,m)+drift(k)*t+ ...
                sigma{k}*sqrt(t)*Z(n,m);
            XTAV(n,m+1)=XTAV(n,m)+drift(k)*t- ...
                sigma{k}*sqrt(t)*Z(n,m);
            D(n)=D(n)*exp(-r(k)*t);
            if k==1
                k=2;
            else
                k=1;
            end
            t=m*dT-waitingTime;
            Zswitch=randn;
            XT(n,m+1)=XT(n,m+1)+drift(k)*t+ ...
                sigma{k}*sqrt(t)*Zswitch;
            XTAV(n,m+1)=XTAV(n,m+1)+drift(k)*t- ...
                sigma{k}*sqrt(t)*Zswitch;
            D(n)=D(n)*exp(-r(k)*t);
            if k==1
                k=2;
            else
                k=1;
```

```
55              end
56              waitingTime=m*dT+exprnd(1/abs(Q(k,k)));
57          else
58              XT(n,m+1)=XT(n,m)+drift(k)*dT+ ...
59                  sigma{k}*sqrt(dT)*Z(n,m);
60              XTAV(n,m+1)=XTAV(n,m)+drift(k)*dT- ...
61                  sigma{k}*sqrt(dT)*Z(n,m);
62              D(n)=D(n)*exp(-r(k)*dT);
63          end
64      end
65  end
66
67  end
```

C.2 COS method

C.2.1 Vanilla options

```
1   function Price=COS_Vanilla(N,L,alpha,model,S0,T,r,delta,K,param)
2   % INPUTS
3   % N:                Number of grid points
4   % L:                Parameter needed in domain truncation
5   % alpha:            1 for a Call, -1 for a Put
6   % model:            Chosen model of the underlying dynamics
7   % S0:               Spot price
8   % T:                Time-to-maturity
9   % r:                Risk-free interest rate
10  % delta:            Dividend yield
11  % K:                Strike price
12  % param:            Vector with the parameters of the model
13  %
14  % FUNCTIONS
15  % ComputeCumulants: Compute the cumulants for the chosen model
16  % VkVanilla:        Returns the Fourier cosine series
17  %                   coefficients for Vanilla options
18  % CharExp:          Returns the characteristic exponent psi(u,T)
19  %
20  % OUTPUT
21  % Price:            Price of the Vanilla option obtained
22  %                   with the COS method
23
24  x0=log(S0/K);
25  c=ComputeCumulants(model,T,r,delta,param);
26  a=x0+c(1)-L*sqrt(c(2)+sqrt(c(3)));
```

```
b=x0+c(1)+L*sqrt(c(2)+sqrt(c(3)));
k=(0:N-1)';
Vk=VkVanilla(k,a,0,a,b,K,-1);            % Put option
u=k*pi/(b-a);
phik=exp(CharExp(model,u,T,r,delta,param));
Lambdak=phik.*Vk;
Lambdak(1)=0.5*Lambdak(1);
Price=exp(-r*T)*sum(real(exp(1i*u*(x0-a)).*Lambdak));
% Call price with Put-Call parity
if alpha==1
    Price=Price+S0*exp(-delta*T)-K*exp(-r*T);
end

end
```

C.2.2 Bermudan options

```
function Price=COS_Bermudan(N,M,L,alpha,model,S0,T,r,delta,K,param)
% INPUTS
% N:                Number of grid points
% M:                Number of dates of early exercise
% L:                Parameter needed in domain truncation
% alpha:            1 for a Call, -1 for a Put
% model:            Chosen model of the underlying dynamics
% S0:               Spot price
% T:                Time-to-maturity
% r:                Risk-free interest rate
% delta:            Dividend yield
% K:                Strike price
% param:            Vector with the parameters of the model
%
% FUNCTIONS
% ComputeCumulants: Compute the cumulants for the chosen model
% VkVanilla:        Returns the Fourier cosine series
%                   coefficients for Vanilla options
% ComputeH:         Returns the continuation value
% CharExp:          Returns the characteristic exponent psi(u,T)
%
% OUTPUT
% Price:            Price of the Bermudan option
%                   obtained with the COS method

dT=T/M;
x0=log(S0/K);
c=ComputeCumulants(model,T,r,delta,param);
```

```
a=x0+c(1)-L*sqrt(c(2)+sqrt(c(3)));
b=x0+c(1)+L*sqrt(c(2)+sqrt(c(3)));
k=(0:N-1)';
u=k*pi/(b-a);
phik=exp(CharExp(model,u,dT,r,delta,param));
if alpha==1
    Gk=@(x) VkVanilla(k,x,b,a,b,K,alpha);
    ContValue=@(x,y) exp(-r*dT)*ComputeH(a,x,a,b,N,y);
else
    Gk=@(x) VkVanilla(k,a,x,a,b,K,alpha);
    ContValue=@(x,y) exp(-r*dT)*ComputeH(x,b,a,b,N,y);
end
xstark=zeros(M,1);
Hk=0;
for m=M-1:-1:1
    Vk=Gk(xstark(m+1))+Hk;
    Lambdak=phik.*Vk;
    Lambdak(1)=0.5*Lambdak(1);
    g=@(x) exp(-r*dT)*sum(real(exp(1i*(x-a)*u).*Lambdak))- ...
        alpha*K*(exp(x)-1);
    xstark(m)=fzero(g,xstark(m+1));
    Hk=ContValue(xstark(m),Lambdak);
end
Vk=Gk(xstark(1))+Hk;
Lambdak=phik.*Vk;
Lambdak(1)=0.5*Lambdak(1);
Price=exp(-r*dT)*sum(real(exp(1i*u*(x0-a)).*Lambdak));

end
```

C.2.3 Barrier options

```
function Price=COS_BarrierUO(N,M,B,Rb,L,alpha,model,S0,T,r, ...
    delta,K,param)
% INPUTS
% N:                Number of grid points
% M:                Number of monitoring dates
% B:                Barrier
% Rb:               Rebate
% L:                Parameter needed in domain truncation
% alpha:            1 for a Call, -1 for a Put
% model:            Chosen model of the underlying dynamics
% S0:               Spot price
% T:                Time-to-maturity
% r:                Risk-free interest rate
```

```
% delta:              Dividend yield
% K:                  Strike price
% param:              Vector with the parameters of the model
%
% FUNCTIONS
% ComputeCumulants:  Compute the cumulants for the chosen model
% VkVanilla:         Returns the Fourier cosine series
%                    coefficients for Vanilla options
% ComputeH:          Returns the continuation value
% CharExp:           Returns the characteristic exponent psi(u,T)
% CosineCoeff:       Function that returns the auxiliar functions
%                    chi,xi needed to compute Vk
%
% OUTPUT
% Price:             Price of the Up & Out Barrier option
%                    obtained with the COS method

dT=T/M;
x0=log(S0/K);
xb=log(B/K);
c=ComputeCumulants(model,T,r,delta,param);
a=x0+c(1)-L*sqrt(c(2)+sqrt(c(3)));
b=x0+c(1)+L*sqrt(c(2)+sqrt(c(3)));
k=(0:N-1)';
u=k*pi/(b-a);
phik=exp(CharExp(model,u,dT,r,delta,param));
if alpha==1
    Vk=@(x) VkVanilla(k,x,xb,a,b,K,alpha);
    if xb>=0
        Vk=Vk(0);
    else
        Vk=0;
    end
else
    Vk=@(x) VkVanilla(k,a,x,a,b,K,alpha);
    if xb>=0
        Vk=Vk(0);
    else
        Vk=Vk(xb);
    end
end
ContValue=@(y) exp(-r*dT)*ComputeH(a,xb,a,b,N,y);
[~,xik]=CosineCoeff(k,xb,b,a,b);
Gk=2/(b-a)*Rb*xik;
Vk=Vk+Gk;
for m=M-1:-1:1
```

```
60      Lambdak=phik.*Vk;
61      Lambdak(1)=0.5*Lambdak(1);
62      Vk=ContValue(Lambdak)+Gk;
63  end
64  Lambdak=phik.*Vk;
65  Lambdak(1)=0.5*Lambdak(1);
66  Price=exp(-r*dT)*sum(real(exp(1i*u*(x0-a)).*Lambdak));
67
68 end
```

```
1  function Price=COS_BarrierDO(N,M,B,Rb,L,alpha,model,S0,T,r, ...
2      delta,K,param)
3  % INPUTS
4  % N:                  Number of grid points
5  % M:                  Number of monitoring dates
6  % B:                  Barrier
7  % Rb:                 Rebate
8  % L:                  Parameter needed in domain truncation
9  % alpha:              1 for a Call, -1 for a Put
10 % model:              Chosen model of the underlying dynamics
11 % S0:                 Spot price
12 % T:                  Time-to-maturity
13 % r:                  Risk-free interest rate
14 % delta:              Dividend yield
15 % K:                  Strike price
16 % param:              Vector with the parameters of the model
17 %
18 % FUNCTIONS
19 % ComputeCumulants:  Compute the cumulants for the chosen model
20 % VkVanilla:          Returns the Fourier cosine series
21 %                     coefficients for Vanilla options
22 % ComputeH:          Returns the continuation value
23 % CharExp:           Returns the characteristic exponent psi(u,T)
24 % CosineCoeff:       Function that returns the auxiliar functions
25 %                     chi,xi needed to compute Vk
26 %
27 % OUTPUT
28 % Price:              Price of the Down & Out Barrier option
29 %                     obtained with the COS method
30
31 dT=T/M;
32 x0=log(S0/K);
33 xb=log(B/K);
34 c=ComputeCumulants(model,T,r,delta,param);
35 a=x0+c(1)-L*sqrt(c(2)+sqrt(c(3)));
36 b=x0+c(1)+L*sqrt(c(2)+sqrt(c(3)));
```

```
k=(0:N-1)';
u=k*pi/(b-a);
phik=exp(CharExp(model,u,dT,r,delta,param));
if alpha==1
    Vk=@(x) VkVanilla(k,x,b,a,b,K,alpha);
    if xb>=0
        Vk=Vk(xb);
    else
        Vk=Vk(0);
    end
else
    Vk=@(x) VkVanilla(k,x,0,a,b,K,alpha);
    if xb>=0
        Vk=Vk(a);
    else
        Vk=Vk(xb);
    end
end
ContValue=@(y) exp(-r*dT)*ComputeH(xb,b,a,b,N,y);
[~,xik]=CosineCoeff(k,a,xb,a,b);
Gk=2/(b-a)*Rb*xik;
Vk=Vk+Gk;
for m=M-1:-1:1
    Lambdak=phik.*Vk;
    Lambdak(1)=0.5*Lambdak(1);
    Vk=ContValue(Lambdak)+Gk;
end
Lambdak=phik.*Vk;
Lambdak(1)=0.5*Lambdak(1);
Price=exp(-r*dT)*sum(real(exp(1i*u*(x0-a)).*Lambdak));

end
```

C.2.4 Asian options

```
function Price=COS_AsianGeom(N,M,L,alpha,model,S0,T,r,delta,K,param)
% INPUTS
% N:                Number of grid points
% M:                Number of monitoring dates
% L:                Parameter needed in domain truncation
% alpha:            1 for a Call, -1 for a Put
% model:            Chosen model of the underlying dynamics
% S0:               Spot price
% T:                Time-to-maturity
% r:                Risk-free interest rate
```

```
% delta:              Dividend yield
% K:                  Strike price
% param:              Vector with the parameters of the model
%
% FUNCTIONS
% ComputeCumulants:  Compute the cumulants for the chosen model
% VkAsianGeom:       Returns the Fourier cosine series
%                    coefficients for Geometric Asian options
% CharExp:           Returns the characteristic exponent psi(u,T)
%
% OUTPUT
% Price:             Price of the Geometric Asian option
%                    obtained with the COS method

dT=T/M;
x0=log(S0);
c=ComputeCumulants(model,T,r,delta,param);
a=x0+c(1)-L*sqrt(c(2)+sqrt(c(3)));
b=x0+c(1)+L*sqrt(c(2)+sqrt(c(3)));
k=(0:N-1)';
if alpha==1
    Vk=VkAsianGeom(k,log(K),b,a,b,K,1);
else
    Vk=VkAsianGeom(k,a,log(K),a,b,K,-1);
end
u=k*pi/(b-a);
m=1:M;
psik=CharExp(model,u*(M+1-m)./(M+1),dT,r,delta,param);
phik=exp(1i*u*x0).*exp(sum(psik,2));
Lambdak=phik.*Vk;
Lambdak(1)=0.5*Lambdak(1);
Price=exp(-r*T)*sum(real(exp(-1i*u*a).*Lambdak));

end
```

```
function Price=COS_AsianArithm(N,M,L,alpha,model,S0,T, ...
    r,delta,K,param)
% INPUTS
% N:                  Number of grid points
% M:                  Number of monitoring dates
% L:                  Parameter needed in domain truncation
% alpha:              1 for a Call, -1 for a Put
% model:              Chosen model of the underlying dynamics
% S0:                 Spot price
% T:                  Time-to-maturity
% r:                  Risk-free interest rate
```

```
% delta:              Dividend yield
% K:                  Strike price
% param:              Vector with the parameters of the model
%
% FUNCTIONS
% ComputeCumulants:  Compute the cumulants for the chosen model
% VkAsianArithm:     Returns the Fourier cosine series
%                    coefficients for Arithmetic Asian options
% ComputeM:          Returns M using Clenshaw-Curtis quadrature
% CharExp:           Returns the characteristic exponent psi(u,T)
%
% OUTPUT
% Price:             Price of the Arithmetic Asian option
%                    obtained with the COS method

dT=T/M;
c=ComputeCumulants(model,dT,r,delta,param);
a=zeros(M,1);
b=zeros(M,1);
for m=1:M
    a(m)=log(m)+c(1)-L*sqrt(m*c(2)+sqrt(m*c(3)));
    b(m)=log(m)+m*c(1)+L*sqrt(m*c(2)+sqrt(m*c(3)));
end
a=min(a);
b=max(b);
k=(0:N-1)';
if alpha==1
    Vk=VkAsianArithm(k,log(K*(M+1)/S0-1),b,a,b,K,S0,M,1);
else
    Vk=VkAsianArithm(k,a,log(K*(M+1)/S0-1),a,b,K,S0,M,-1);
end
u=k*pi/(b-a);
nq=25/16*N;
MM=zeros(N,N);
for j=1:N
    for l=1:N
        MM(j,l)=ComputeM(a,b,u(j),u(l),nq);
    end
end
phikR=exp(CharExp(model,u,dT,r,delta,param));
phikY=phikR;
for m=2:M
    A=2/(b-a)*real(phikY.*exp(-1i*a*u));
    A(1)=0.5*A(1);
    phikZ=MM*A;
    phikY=phikR.*phikZ;
```

```
 end
 Lambdak=phikY.*Vk;
 Lambdak(1)=0.5*Lambdak(1);
 Price=exp(-r*T)*sum(real(exp(-1i*u*a).*Lambdak));

end
```

C.2.5 2RS Vanilla options

```
function Prices=COS_Vanilla_2RS(N,L,alpha,models,S0,T,r,delta, ...
    K,Q,params)
 % INPUTS
 % N:                 Number of grid points
 % L:                 Parameter needed in domain truncation
 % alpha:             1 for a Call, -1 for a Put
 % models:            String vector containing the chosen models
 %                    of the underlying dynamics in the two states
 % S0:                Spot price
 % T:                 Time-to-maturity
 % r:                 Vector with the risk-free interest rates
 %                    in the 2 states
 % delta:             Vector with the dividend yields
 %                    in the 2 states
 % K:                 Strike price
 % Q:                 Intensity matrix
 % params:            Cell array with the parameters of
 %                    [model1,model2]
 %
 % FUNCTIONS
 % ComputeCumulants:  Compute the cumulants for the chosen model
 % VkVanilla:         Returns the Fourier cosine series
 %                    coefficients for Vanilla options
 % CharFun_2RS:       Returns the vectors necessary to build the
 %                    block diagonal matrix phi(u,T)
 %
 % OUTPUT
 % Prices:            Prices in the 2 states of the Vanilla
 %                    option obtained with the COS method
 %                    under a regime-switching model

 x0=log(S0/K);
 c1=ComputeCumulants(models(1),T,r(1),delta(1),params{1});
 c2=ComputeCumulants(models(2),T,r(2),delta(2),params{2});
 a1=x0+c1(1)-L*sqrt(c1(2)+sqrt(c1(3)));
 b1=x0+c1(1)+L*sqrt(c1(2)+sqrt(c1(3)));
```

```
a2=x0+c2(1)-L*sqrt(c2(2)+sqrt(c2(3)));
b2=x0+c2(1)+L*sqrt(c2(2)+sqrt(c2(3)));
a=min(a1,a2);
b=max(b1,b2);
k=(0:N-1)';
u=k*pi/(b-a);
[phik11,phik12,phik21,phik22]=CharFun_2RS(models,u,T,r, ...
    delta,Q,params);
if alpha==1
    Vk=VkVanilla(k,0,b,a,b,K,1);
else
    Vk=VkVanilla(k,a,0,a,b,K,-1);
end
Lambdak1=(phik11+phik12).*Vk;
Lambdak2=(phik21+phik22).*Vk;
Lambdak1(1)=0.5*Lambdak1(1);
Lambdak2(1)=0.5*Lambdak2(1);
Prices=zeros(2,1);
Prices(1)=sum(real(exp(1i*u*(x0-a)).*Lambdak1));
Prices(2)=sum(real(exp(1i*u*(x0-a)).*Lambdak2));

end
```

C.2.6 2RS Digital and Butterfly options

```
function Prices=COS_Digital_2RS(N,L,alpha,models,S0,T,r,delta, ...
    K,Q,params)
% INPUTS
% N:                Number of grid points
% L:                Parameter needed in domain truncation
% alpha:            1 for a Call, -1 for a Put
% models:           String vector containing the chosen models
%                   of the underlying dynamics in the two states
% S0:               Spot price
% T:                Time-to-maturity
% r:                Vector with the risk-free interest rates
%                   in the 2 states
% delta:            Vector with the dividend yields
%                   in the 2 states
% K:                Strike price
% Q:                Intensity matrix
% params:           Cell array with the parameters of
%                   [model1,model2]
%
% FUNCTIONS
```

```
% ComputeCumulants:  Compute the cumulants for the chosen model
% VkDigital:         Returns the Fourier cosine series
%                    coefficients for Digital options
% CharFun_2RS:       Returns the vectors necessary to build the
%                    block diagonal matrix phi(u,T)
%
% OUTPUT
% Prices:            Prices in the 2 states of the Digital
%                    option obtained with the COS method
%                    under a regime-switching model

x0=log(S0/K);
c1=ComputeCumulants(models(1),T,r(1),delta(1),params{1});
c2=ComputeCumulants(models(2),T,r(2),delta(2),params{2});
a1=x0+c1(1)-L*sqrt(c1(2)+sqrt(c1(3)));
b1=x0+c1(1)+L*sqrt(c1(2)+sqrt(c1(3)));
a2=x0+c2(1)-L*sqrt(c2(2)+sqrt(c2(3)));
b2=x0+c2(1)+L*sqrt(c2(2)+sqrt(c2(3)));
a=min(a1,a2);
b=max(b1,b2);
k=(0:N-1)';
u=k*pi/(b-a);
[phik11,phik12,phik21,phik22]=CharFun_2RS(models,u,T,r, ...
    delta,Q,params);
if alpha==1
    Vk=VkDigital(k,0,b,a,b,K);
else
    Vk=VkDigital(k,a,0,a,b,K);
end
Lambdak1=(phik11+phik12).*Vk;
Lambdak2=(phik21+phik22).*Vk;
Lambdak1(1)=0.5*Lambdak1(1);
Lambdak2(1)=0.5*Lambdak2(1);
Prices=zeros(2,1);
Prices(1)=sum(real(exp(1i*u*(x0-a)).*Lambdak1));
Prices(2)=sum(real(exp(1i*u*(x0-a)).*Lambdak2));

end
```

```
function Prices=COS_Butterfly_2RS(N,L,alpha,models,S0,T,r,delta, ...
    K1,K2,K3,p,Q,params)
% INPUTS
% N:                 Number of grid points
% L:                 Parameter needed in domain truncation
% alpha:             1 for a Call, -1 for a Put
% models:            String vector containing the chosen models
```

```
8  %                    of the underlying dynamics in the two states
9  % S0:                Spot price
10 % T:                 Time-to-maturity
11 % r:                 Vector with the risk-free interest rates
12 %                    in the 2 states
13 % delta:             Vector with the dividend yields
14 %                    in the 2 states
15 % [K1,K2,K3]:        Strike prices
16 % p:                 Scale parameter
17 % Q:                 Intensity matrix
18 % params:            Cell array with the parameters of
19 %                    [model1,model2]
20 %
21 % FUNCTIONS
22 % ComputeCumulants:  Compute the cumulants for the chosen model
23 % VkButterfly:       Returns the Fourier cosine series
24 %                    coefficients for Butterfly options
25 % CharFun_2RS:       Returns the vectors necessary to build the
26 %                    block diagonal matrix phi(u,T)
27 %
28 % OUTPUT
29 % Prices:            Prices in the 2 states of the Butterfly
30 %                    option obtained with the COS method
31 %                    under a regime-switching model
32
33 x0=log(S0/p);
34 c1=ComputeCumulants(models(1),T,r(1),delta(1),params{1});
35 c2=ComputeCumulants(models(2),T,r(2),delta(2),params{2});
36 a1=x0+c1(1)-L*sqrt(c1(2)+sqrt(c1(3)));
37 b1=x0+c1(1)+L*sqrt(c1(2)+sqrt(c1(3)));
38 a2=x0+c2(1)-L*sqrt(c2(2)+sqrt(c2(3)));
39 b2=x0+c2(1)+L*sqrt(c2(2)+sqrt(c2(3)));
40 a=min(a1,a2);
41 b=max(b1,b2);
42 k=(0:N-1)';
43 u=k*pi/(b-a);
44 [phik11,phik12,phik21,phik22]=CharFun_2RS(models,u,T,r, ...
45     delta,Q,params);
46 if alpha==1
47     Vk=VkButterfly(k,log(K1/p),log(K2/p),log(K3/p),b,a,b, ...
48         K1,K2,K3,p,alpha);
49 else
50     Vk=VkButterfly(k,log(K1/p),log(K2/p),log(K3/p),a,a,b, ...
51         K1,K2,K3,p,alpha);
52 end
53 Lambdak1=(phik11+phik12).*Vk;
```

```
 Lambdak2=(phik21+phik22).*Vk;
 Lambdak1(1)=0.5*Lambdak1(1);
 Lambdak2(1)=0.5*Lambdak2(1);
 Prices=zeros(2,1);
 Prices(1)=sum(real(exp(1i*u*(x0-a)).*Lambdak1));
 Prices(2)=sum(real(exp(1i*u*(x0-a)).*Lambdak2));

end
```

C.2.7 2RS Bermudan options

```
function Prices=COS_Bermudan_2RS(N,M,L,alpha,models,S0,T,r, ...
    delta,K,Q,params)
 % INPUTS
 % N:                 Number of grid points
 % M:                 Number of time steps
 % L:                 Parameter needed in domain truncation
 % alpha:             1 for a Call, -1 for a Put
 % models:            String vector containing the chosen models
 %                    of the underlying dynamics in the two states
 % S0:                Spot price
 % T:                 Time-to-maturity
 % r:                 Vector with the risk-free interest rates
 %                    in the 2 states
 % delta:             Vector with the dividend yields
 %                    in the 2 states
 % K:                 Strike price
 % Q:                 Intensity matrix
 % params:            Cell array with the parameters of
 %                    [model1,model2]
 %
 % FUNCTIONS
 % ComputeCumulants:  Compute the cumulants for the chosen model
 % VkVanilla:         Returns the Fourier cosine series
 %                    coefficients for Vanilla options
 % ComputeH:          Returns the continuation value
 % CharFun_2RS:       Returns the vectors necessary to build the
 %                    block diagonal matrix phi(u,T)
 %
 % OUTPUT
 % Prices:            Prices in the 2 states of the Bermudan
 %                    option obtained with the COS method
 %                    under a regime-switching model

 dT=T/M;
```

```
x0=log(S0/K);
c1=ComputeCumulants(models(1),T,r(1),delta(1),params{1});
c2=ComputeCumulants(models(2),T,r(2),delta(2),params{2});
a1=x0+c1(1)-L*sqrt(c1(2)+sqrt(c1(3)));
b1=x0+c1(1)+L*sqrt(c1(2)+sqrt(c1(3)));
a2=x0+c2(1)-L*sqrt(c2(2)+sqrt(c2(3)));
b2=x0+c2(1)+L*sqrt(c2(2)+sqrt(c2(3)));
a=min(a1,a2);
b=max(b1,b2);
k=(0:N-1)';
u=k*pi/(b-a);
[phik11,phik12,phik21,phik22]=CharFun_2RS(models,u,dT,r, ...
   delta,Q,params);
if alpha==1
    Gk=@(x) VkVanilla(k,x,b,a,b,K,alpha);
    ContValue=@(x,y) ComputeH(a,x,a,b,N,y);
else
    Gk=@(x) VkVanilla(k,a,x,a,b,K,alpha);
    ContValue=@(x,y) ComputeH(x,b,a,b,N,y);
end
xstark=zeros(M,2);
Hk1=zeros(N,1);
Hk2=zeros(N,1);
for m=M-1:-1:1
    Vk1=Gk(xstark(m+1,1))+Hk1;
    Vk2=Gk(xstark(m+1,2))+Hk2;
    Lambdak1=phik11.*Vk1+phik12.*Vk2;
    Lambdak2=phik21.*Vk1+phik22.*Vk2;
    Lambdak1(1)=0.5*Lambdak1(1);
    Lambdak2(1)=0.5*Lambdak2(1);
    g1=@(x) sum(real(exp(1i*(x-a)*u).*Lambdak1))- ...
        alpha*K*(exp(x)-1);
    g2=@(x) sum(real(exp(1i*(x-a)*u).*Lambdak2))- ...
        alpha*K*(exp(x)-1);
    xstark(m,1)=fzero(g1,xstark(m+1,1));
    xstark(m,2)=fzero(g2,xstark(m+1,2));
    Hk1=ContValue(xstark(m,1),Lambdak1);
    Hk2=ContValue(xstark(m,2),Lambdak2);
end
Vk1=Gk(xstark(1,1))+Hk1;
Vk2=Gk(xstark(1,2))+Hk2;
Lambdak1=phik11.*Vk1+phik12.*Vk2;
Lambdak2=phik21.*Vk1+phik22.*Vk2;
Lambdak1(1)=0.5*Lambdak1(1);
Lambdak2(1)=0.5*Lambdak2(1);
Prices=zeros(2,1);
```

```
 Prices(1)=sum(real(exp(1i*u*(x0-a)).*Lambdak1));
 Prices(2)=sum(real(exp(1i*u*(x0-a)).*Lambdak2));

end
```

C.2.8 2RS Barrier options

```
function Prices=COS_BarrierUO_2RS(N,M,B,Rb,L,alpha,models,S0,T, ...
    r,delta,K,Q,params)
 % INPUTS
 % N:                  Number of grid points
 % M:                  Number of monitoring dates
 % B:                  Barrier
 % Rb:                 Rebate
 % L:                  Parameter needed in domain truncation
 % alpha:              1 for a Call, -1 for a Put
 % models:             String vector containing the chosen models
 %                     of the underlying dynamics in the two states
 % S0:                 Spot price
 % T:                  Time-to-maturity
 % r:                  Vector with the risk-free interest rates
 %                     in the 2 states
 % delta:              Vector with the dividend yields
 %                     in the 2 states
 % K:                  Strike price
 % Q:                  Intensity matrix
 % params:             Cell array with the parameters of
 %                     [model1,model2]
 %
 % FUNCTIONS
 % ComputeCumulants:  Compute the cumulants for the chosen model
 % VkVanilla:         Returns the Fourier cosine series
 %                     coefficients for Vanilla options
 % ComputeH:          Returns the continuation value
 % CharFun_2RS:       Returns the vectors necessary to build the
 %                     block diagonal matrix phi(u,T)
 % CosineCoeff:       Function that returns the auxiliar functions
 %                     chi,xi needed to compute Vk
 %
 % OUTPUT
 % Prices:             Prices in the 2 states of the Up & Out
 %                     Barrier option obtained with the COS method
 %                     under a regime-switching model

 dT=T/M;
```

```
x0=log(S0/K);
xb=log(B/K);
c1=ComputeCumulants(models(1),T,r(1),delta(1),params{1});
c2=ComputeCumulants(models(2),T,r(2),delta(2),params{2});
a1=x0+c1(1)-L*sqrt(c1(2)+sqrt(c1(3)));
b1=x0+c1(1)+L*sqrt(c1(2)+sqrt(c1(3)));
a2=x0+c2(1)-L*sqrt(c2(2)+sqrt(c2(3)));
b2=x0+c2(1)+L*sqrt(c2(2)+sqrt(c2(3)));
a=min(a1,a2);
b=max(b1,b2);
k=(0:N-1)';
u=k*pi/(b-a);
[phik11,phik12,phik21,phik22]=CharFun_2RS(models,u,dT,r, ...
   delta,Q,params);
if alpha==1
    Vk=@(x) VkVanilla(k,x,xb,a,b,K,alpha);
    if xb>=0
        Vk=Vk(0);
    else
        Vk=0;
    end
else
    Vk=@(x) VkVanilla(k,a,x,a,b,K,alpha);
    if xb>=0
        Vk=Vk(0);
    else
        Vk=Vk(xb);
    end
end
ContValue=@(y) ComputeH(a,xb,a,b,N,y);
[~,xik]=CosineCoeff(k,xb,b,a,b);
Gk=2/(b-a)*Rb*xik;
Vk1=Vk+Gk;
Vk2=Vk+Gk;
for m=M-1:-1:1
    Lambdak1=phik11.*Vk1+phik12.*Vk2;
    Lambdak2=phik21.*Vk1+phik22.*Vk2;
    Lambdak1(1)=0.5*Lambdak1(1);
    Lambdak2(1)=0.5*Lambdak2(1);
    Vk1Aux=ContValue(Lambdak1)+Gk;
    Vk2Aux=ContValue(Lambdak2)+Gk;
    Vk1=Vk1Aux;
    Vk2=Vk2Aux;
end
Lambdak1=phik11.*Vk1+phik12.*Vk2;
Lambdak2=phik21.*Vk1+phik22.*Vk2;
```

```
 Lambdak1(1)=0.5*Lambdak1(1);
 Lambdak2(1)=0.5*Lambdak2(1);
 Prices=zeros(2,1);
 Prices(1)=sum(real(exp(1i*u*(x0-a)).*Lambdak1));
 Prices(2)=sum(real(exp(1i*u*(x0-a)).*Lambdak2));

end
```

```
function Prices=COS_BarrierDO_2RS(N,M,B,Rb,L,alpha,models,S0,T, ...
    r,delta,K,Q,params)
 % INPUTS
 % N:                  Number of grid points
 % M:                  Number of monitoring dates
 % B:                  Barrier
 % Rb:                 Rebate
 % L:                  Parameter needed in domain truncation
 % alpha:              1 for a Call, -1 for a Put
 % models:             String vector containing the chosen models
 %                     of the underlying dynamics in the two states
 % S0:                 Spot price
 % T:                  Time-to-maturity
 % r:                  Vector with the risk-free interest rates
 %                     in the 2 states
 % delta:              Vector with the dividend yields
 %                     in the 2 states
 % K:                  Strike price
 % Q:                  Intensity matrix
 % params:             Cell array with the parameters of
 %                     [model1,model2]
 %
 % FUNCTIONS
 % ComputeCumulants:  Compute the cumulants for the chosen model
 % VkVanilla:         Returns the Fourier cosine series
 %                     coefficients for Vanilla options
 % ComputeH:          Returns the continuation value
 % CharFun_2RS:       Returns the vectors necessary to build the
 %                     block diagonal matrix phi(u,T)
 % CosineCoeff:       Function that returns the auxiliar functions
 %                     chi,xi needed to compute Vk
 %
 % OUTPUT
 % Prices:             Prices in the 2 states of the Down & Out
 %                     Barrier option obtained with the COS method
 %                     under a regime-switching model

 dT=T/M;
```

```
x0=log(S0/K);
xb=log(B/K);
c1=ComputeCumulants(models(1),T,r(1),delta(1),params{1});
c2=ComputeCumulants(models(2),T,r(2),delta(2),params{2});
a1=x0+c1(1)-L*sqrt(c1(2)+sqrt(c1(3)));
b1=x0+c1(1)+L*sqrt(c1(2)+sqrt(c1(3)));
a2=x0+c2(1)-L*sqrt(c2(2)+sqrt(c2(3)));
b2=x0+c2(1)+L*sqrt(c2(2)+sqrt(c2(3)));
a=min(a1,a2);
b=max(b1,b2);
k=(0:N-1)';
u=k*pi/(b-a);
[phik11,phik12,phik21,phik22]=CharFun_2RS(models,u,dT,r, ...
   delta,Q,params);
if alpha==1
    Vk=@(x) VkVanilla(k,x,b,a,b,K,alpha);
    if xb>=0
        Vk=Vk(xb);
    else
        Vk=Vk(0);
    end
else
    Vk=@(x) VkVanilla(k,x,0,a,b,K,alpha);
    if xb>=0
        Vk=Vk(a);
    else
        Vk=Vk(xb);
    end
end
ContValue=@(y) ComputeH(xb,b,a,b,N,y);
[~,xik]=CosineCoeff(k,a,xb,a,b);
Gk=2/(b-a)*Rb*xik;
Vk1=Vk+Gk;
Vk2=Vk+Gk;
for m=M-1:-1:1
    Lambdak1=phik11.*Vk1+phik12.*Vk2;
    Lambdak2=phik21.*Vk1+phik22.*Vk2;
    Lambdak1(1)=0.5*Lambdak1(1);
    Lambdak2(1)=0.5*Lambdak2(1);
    Vk1Aux=ContValue(Lambdak1)+Gk;
    Vk2Aux=ContValue(Lambdak2)+Gk;
    Vk1=Vk1Aux;
    Vk2=Vk2Aux;
end
Lambdak1=phik11.*Vk1+phik12.*Vk2;
Lambdak2=phik21.*Vk1+phik22.*Vk2;
```

```
 Lambdak1(1)=0.5*Lambdak1(1);
 Lambdak2(1)=0.5*Lambdak2(1);
 Prices=zeros(2,1);
 Prices(1)=sum(real(exp(1i*u*(x0-a)).*Lambdak1));
 Prices(2)=sum(real(exp(1i*u*(x0-a)).*Lambdak2));

end
```

C.2.9 2RS Asian options

```
function Prices=COS_AsianGeom_2RS(N,M,L,alpha,models,S0,T, ...
    r,delta,K,Q,params)
 % INPUTS
 % N:                  Number of grid points
 % M:                  Number of monitoring dates
 % L:                  Parameter needed in domain truncation
 % alpha:              1 for a Call, -1 for a Put
 % models:             String vector containing the chosen models
 %                     of the underlying dynamics in the two states
 % S0:                 Spot price
 % T:                  Time-to-maturity
 % r:                  Vector with the risk-free interest rates
 %                     in the 2 states
 % delta:              Vector with the dividend yields
 %                     in the 2 states
 % K:                  Strike price
 % Q:                  Intensity matrix
 % params:             Cell array with the parameters of
 %                     [model1,model2]
 %
 % FUNCTIONS
 % ComputeCumulants:  Compute the cumulants for the chosen model
 % VkAsianGeom:       Returns the Fourier cosine series
 %                     coefficients for Geometric Asian options
 % CharFun_2RS:       Returns the vectors necessary to build the
 %                     block diagonal matrix phi(u,T)
 %
 % OUTPUT
 % Prices:             Prices in the 2 states of the Geometric
 %                     Asian option obtained with the COS method
 %                     under a regime-switching model

 dT=T/M;
 x0=log(S0);
 c1=ComputeCumulants(models(1),T,r(1),delta(1),params{1});
```

```
c2=ComputeCumulants(models(2),T,r(2),delta(2),params{2});
a1=x0+c1(1)-L*sqrt(c1(2)+sqrt(c1(3)));
b1=x0+c1(1)+L*sqrt(c1(2)+sqrt(c1(3)));
a2=x0+c2(1)-L*sqrt(c2(2)+sqrt(c2(3)));
b2=x0+c2(1)+L*sqrt(c2(2)+sqrt(c2(3)));
a=min(a1,a2);
b=max(b1,b2);
k=(0:N-1)';
u=k*pi/(b-a);
if alpha==1
    Vk=VkAsianGeom(k,log(K),b,a,b,K,1);
else
    Vk=VkAsianGeom(k,a,log(K),a,b,K,-1);
end
phik1=ones(N,1);
phik2=ones(N,1);
for m=1:M
    [p11,p12,p21,p22]=CharFun_2RS(models, ...
        u*(M+1-m)./(M+1),dT,r,delta,Q,params);
    phik1=phik1.*(p11+p12);
    phik2=phik2.*(p21+p22);
end
Lambdak1=exp(1i*u*x0).*phik1.*Vk;
Lambdak2=exp(1i*u*x0).*phik2.*Vk;
Lambdak1(1)=0.5*Lambdak1(1);
Lambdak2(1)=0.5*Lambdak2(1);
Prices=zeros(2,1);
Prices(1)=sum(real(exp(-1i*u*a).*Lambdak1));
Prices(2)=sum(real(exp(-1i*u*a).*Lambdak2));

end
```

```
function Prices=COS_AsianArithm_2RS(N,M,L,alpha,models,S0,T, ...
    r,delta,K,Q,params)
% INPUTS
% N:                Number of grid points
% M:                Number of monitoring dates
% L:                Parameter needed in domain truncation
% alpha:            1 for a Call, -1 for a Put
% models:           String vector containing the chosen models
%                   of the underlying dynamics in the two states
% S0:               Spot price
% T:                Time-to-maturity
% r:                Vector with the risk-free interest rates
%                   in the 2 states
% delta:            Vector with the dividend yields
```

```
% in the 2 states
% K: Strike price
% Q: Intensity matrix
% params: Cell array with the parameters of
% [model1,model2]
%
% FUNCTIONS
% ComputeCumulants: Compute the cumulants for the chosen model
% VkAsianArithm: Returns the Fourier cosine series
% coefficients for Arithmetic Asian options
% ComputeM: Returns M using Clenshaw-Curtis quadrature
% CharFun_2RS: Returns the vectors necessary to build the
% block diagonal matrix phi(u,T)
%
% OUTPUT
% Prices: Prices in the 2 states of the Arithmetic
% Asian option obtained with the COS method
% under a regime-switching model

dT=T/M;
c1=ComputeCumulants(models(1),dT,r(1),delta(1),params{1});
c2=ComputeCumulants(models(2),dT,r(2),delta(2),params{2});
a1=zeros(M,1);
a2=zeros(M,1);
b1=zeros(M,1);
b2=zeros(M,1);
for m=1:M
    a1(m)=log(m)+c1(1)-L*sqrt(m*c1(2)+sqrt(m*c1(3)));
    b1(m)=log(m)+m*c1(1)+L*sqrt(m*c1(2)+sqrt(m*c1(3)));
    a2(m)=log(m)+c2(1)-L*sqrt(m*c2(2)+sqrt(m*c2(3)));
    b2(m)=log(m)+m*c2(1)+L*sqrt(m*c2(2)+sqrt(m*c2(3)));
end
a1=min(a1);
a2=min(a2);
b1=max(b1);
b2=max(b2);
a=min(a1,a2);
b=max(b1,b2);
k=(0:N-1)';
u=k*pi/(b-a);
if alpha==1
    Vk=VkAsianArithm(k,log(K*(M+1)/S0-1),b,a,b,K,S0,M,1);
else
    Vk=VkAsianArithm(k,a,log(K*(M+1)/S0-1),a,b,K,S0,M,-1);
end
nq=25/16*N;
```

```
MM=zeros(N,N);
for j=1:N
    for l=1:N
        MM(j,l)=ComputeM(a,b,u(j),u(l),nq);
    end
end
[phikR11,phikR12,phikR21,phikR22]=CharFun_2RS(models,u,dT,r, ...
    delta,Q,params);
phikR1=phikR11+phikR12;
phikR2=phikR21+phikR22;
phikY1=phikR1;
phikY2=phikR2;
for m=2:M
    A1=2/(b-a)*real(phikY1.*exp(-1i*a*u));
    A2=2/(b-a)*real(phikY2.*exp(-1i*a*u));
    A1(1)=0.5*A1(1);
    A2(1)=0.5*A2(1);
    phikZ1=MM*A1;
    phikZ2=MM*A2;
    phikY1=phikR1.*phikZ1;
    phikY2=phikR2.*phikZ2;
end
Lambdak1=phikY1.*Vk;
Lambdak2=phikY2.*Vk;
Lambdak1(1)=0.5*Lambdak1(1);
Lambdak2(1)=0.5*Lambdak2(1);
Prices=zeros(2,1);
Prices(1)=sum(real(exp(-1i*u*a).*Lambdak1));
Prices(2)=sum(real(exp(-1i*u*a).*Lambdak2));

end
```

C.3 Different methods

C.3.1 Lattice method

```
function Prices=Lattice_Vanilla_2RS(N,alpha,S0,T,r,delta,K,Q,params)
% INPUTS
% N:                Number of time steps
% alpha:            1 for a Call, -1 for a Put
% S0:               Spot price
% T:                Time-to-maturity
% r:                Vector with the risk-free interest rates
%                   in the 2 states
```

```
% delta:              Vector with the dividend yields
%                     in the 2 states
% K:                  Strike price
% Q:                  Intensity matrix
% params:             Cell array with the parameters of
%                     [model1,model2]
%
% OUTPUTS
% Prices:             Prices in the 2 states of the Vanilla
%                     option obtained with the Lattice method
%                     under a regime-switching model

dt=T/N;
sigma1=params{1};
sigma2=params{2};
sigma=max(sigma1,sigma2)+(sqrt(1.5)-1)*(sigma1+sigma2)/2;
lambda1=sigma/sigma1;
lambda2=sigma/sigma2;
u=exp(sigma*sqrt(dt));
d=1/u;
pm_1=1-1/(lambda1^2);
pm_2=1-1/(lambda2^2);
pu_1=(exp((r(1)-delta(1))*dt)-d-pm_1*(1-d))/(u-d);
pu_2=(exp((r(2)-delta(2))*dt)-d-pm_2*(1-d))/(u-d);
pd_1=(u-exp((r(1)-delta(1))*dt)-pm_1*(u-1))/(u-d);
pd_2=(u-exp((r(2)-delta(2))*dt)-pm_2*(u-1))/(u-d);
P=expm(Q*dt);

% Lattice generation
Tree1='';
Tree2='';
for t=1:N+1
    Tree1{t}=zeros(1,2*t-1);
    Tree2{t}=zeros(1,2*t-1);
end
for n=1:length(Tree1{end})
    Tree1{end}(n)=max(alpha*(S0*u^(N+1-n)-K),0);
    Tree2{end}(n)=max(alpha*(S0*u^(N+1-n)-K),0);
end

% Backward in time procedure
for t=N:-1:1
    for n=1:length(Tree1{t})
        Tree1{t}(n)=exp(-r(1)*dt)*(P(1,1)*(pu_1*Tree1{t+1}(n)+ ...
            pm_1*Tree1{t+1}(n+1)+pd_1*Tree1{t+1}(n+2))+P(1,2)* ...
            (pu_1*Tree2{t+1}(n)+pm_1*Tree2{t+1}(n+1)+pd_1* ...
```

```
            Tree2{t+1}(n+2)));
        Tree2{t}(n)=exp(-r(2)*dt)*(P(2,1)*(pu_2*Tree1{t+1}(n)+ ...
            pm_2*Tree1{t+1}(n+1)+pd_2*Tree1{t+1}(n+2))+P(2,2)* ...
            (pu_2*Tree2{t+1}(n)+pm_2*Tree2{t+1}(n+1)+pd_2* ...
            Tree2{t+1}(n+2)));
    end
end
Prices=zeros(2,1);
Prices(1)=Tree1{1}(1);
Prices(2)=Tree2{1}(1);

end
```

C.3.2 FST method

```
function Prices=FST_Vanilla_2RS(N,L,alpha,models,S0,T,r,delta, ...
    K,Q,params)
% INPUTS
% N:                Number of grid points
% L:                Parameter needed in choosing the grid domain
% alpha:            1 for a Call, -1 for a Put
% models:           String vector containing the chosen models
%                   of the underlying dynamics in the two states
% S0:               Spot price
% T:                Time-to-maturity
% r:                Vector with the risk-free interest rates
%                   in the 2 states
% delta:            Vector with the dividend yields
%                   in the 2 states
% K:                Strike price
% Q:                Intensity matrix
% params:           Cell array with the parameters of
%                   [model1,model2]
%
% FUNCTIONS
% CharFun_2RS:      Returns the vectors necessary to build the
%                   block diagonal matrix phi(u,T)
%
% OUTPUT
% Prices:           Prices in the 2 states of the Vanilla
%                   option obtained with the FST method
%                   under a regime-switching model

xmax=L;
xmin=-xmax;
```

```
dx=(xmax-xmin)/(N-1);
x=xmin:dx:xmax;
wmax=pi/dx;
wmin=-wmax;
dw=2*wmax/N;
w=[0:dw:wmax,wmin+dw:dw:-dw];
[phi11,phi12,phi21,phi22]=CharFun_2RS(models,w,T,r, ...
    delta,Q,params);
S=K*exp(x);
v_opt=max(alpha*(S-K),0);
v_opt1=real(ifft(fft(v_opt).*(phi11+phi12)));
v_opt2=real(ifft(fft(v_opt).*(phi21+phi22)));
Prices=zeros(2,1);
Prices(1)=interp1(S,v_opt1,S0,'pchip');
Prices(2)=interp1(S,v_opt2,S0,'pchip');

end
```

```
function Prices=FST_Digital_2RS(N,L,alpha,models,S0,T,r, ...
    delta,K,Q,params)
% INPUTS
% N:                Number of grid points
% L:                Parameter needed in choosing the grid domain
% alpha:            1 for a Call, -1 for a Put
% models:           String vector containing the chosen models
%                   of the underlying dynamics in the two states
% S0:               Spot price
% T:                Time-to-maturity
% r:                Vector with the risk-free interest rates
%                   in the 2 states
% delta:            Vector with the dividend yields
%                   in the 2 states
% K:                Strike price
% Q:                Intensity matrix
% params:           Cell array with the parameters of
%                   [model1,model2]
%
% FUNCTIONS
% CharFun_2RS:      Returns the vectors necessary to build the
%                   block diagonal matrix phi(u,T)
%
% OUTPUT
% Prices:           Prices in the 2 states of the Digital
%                   option obtained with the FST method
%                   under a regime-switching model

```

```
xmax=L;
xmin=-xmax;
dx=(xmax-xmin)/(N-1);
x=xmin:dx:xmax;
wmax=pi/dx;
wmin=-wmax;
dw=2*wmax/N;
w=[0:dw:wmax,wmin+dw:dw:-dw];
[phi11,phi12,phi21,phi22]=CharFun_2RS(models,w,T,r, ...
    delta,Q,params);
S=K*exp(x);
if alpha==1
    v_opt=K.*(S>K);
else
    v_opt=K.*(S<K);
end
v_opt1=real(ifft(fft(v_opt).*(phi11+phi12)));
v_opt2=real(ifft(fft(v_opt).*(phi21+phi22)));
Prices=zeros(2,1);
Prices(1)=interp1(S,v_opt1,S0,'pchip');
Prices(2)=interp1(S,v_opt2,S0,'pchip');

end
```

```
function Prices=FST_Bermudan_2RS(N,M,L,alpha,models,S0,T,r, ...
    delta,K,Q,params)
% INPUTS
% N:                Number of grid points
% M:                Number of time steps
% L:                Parameter needed in domain truncation
% alpha:            1 for a Call, -1 for a Put
% models:           String vector containing the chosen models
%                   of the underlying dynamics in the two states
% S0:               Spot price
% T:                Time-to-maturity
% r:                Vector with the risk-free interest rates
%                   in the 2 states
% delta:            Vector with the dividend yields
%                   in the 2 states
% K:                Strike price
% Q:                Intensity matrix
% params:           Cell array with the parameters of
%                   [model1,model2]
%
% FUNCTION
% CharFun_2RS:      Returns the vectors necessary to build the
```

```
% block diagonal matrix phi(u,T)
%
% OUTPUT
% Prices:           Prices in the 2 states of the Bermudan
%                   option obtained with the FST method
%                   under a regime-switching model

dT=T/M;
xmax=L;
xmin=-xmax;
dx=(xmax-xmin)/(N-1);
x=xmin:dx:xmax;
wmax=pi/dx;
wmin=-wmax;
dw=2*wmax/N;
w=[0:dw:wmax,wmin+dw:dw:-dw];
[phi11,phi12,phi21,phi22]=CharFun_2RS(models,w,dT,r, ...
    delta,Q,params);
S=K*exp(x);
payoff=max(0,alpha*(S-K));
v_opt1=payoff;
v_opt2=payoff;
for i=M-1:-1:0
    v_opt1Aux=real(ifft(fft(v_opt1).*phi11+fft(v_opt2).*phi12));
    v_opt2Aux=real(ifft(fft(v_opt1).*phi21+fft(v_opt2).*phi22));
    v_opt1=max(v_opt1Aux,payoff);
    v_opt2=max(v_opt2Aux,payoff);
end
Prices=zeros(2,1);
Prices(1)=interp1(S,v_opt1,S0,'pchip');
Prices(2)=interp1(S,v_opt2,S0,'pchip');

end
```

```
function Prices=FST_Barrier_2RS(N,M,B,Rb,L,alpha,models,S0,T,r, ...
    delta,K,type,Q,params)
% INPUTS
% N:                Number of grid points
% M:                Number of monitoring dates
% B:                Barrier
% Rb:               Rebate
% L:                Parameter needed in choosing the grid domain
% alpha:            1 for a Call, -1 for a Put
% models:           String vector containing the chosen models
%                   of the underlying dynamics in the two states
% S0:               Spot price
```

```
% T:                 Time-to-maturity
% r:                 Vector with the risk-free interest rates
%                    in the 2 states
% delta:             Vector with the dividend yields
%                    in the 2 states
% K:                 Strike price
% type:              'Up' or 'Down'
% Q:                 Intensity matrix
% params:            Cell array with the parameters of
%                    [model1,model2]
% FUNCTION
% CharFun_2RS:       Returns the vectors necessary to build the
%                    block diagonal matrix phi(u,T)
%
% OUTPUTS
% Prices:            Prices in the 2 states of the Out
%                    Barrier option obtained with the FST method
%                    under a regime-switching model

dT=T/M;
xmax=L;
xmin=-xmax;
dx=(xmax-xmin)/(N-1);
x=xmin:dx:xmax;
wmax=pi/dx;
wmin=-wmax;
dw=2*wmax/N;
w=[0:dw:wmax,wmin+dw:dw:-dw];
[phi11,phi12,phi21,phi22]=CharFun_2RS(models,w,dT,r, ...
    delta,Q,params);
S=K*exp(x);
logB=log(B/S0);
if strcmp(type,'Up')
    payoff=max(alpha*(S-K),0).*(x<logB)+Rb*(x>=logB);
else
    payoff=max(alpha*(S-K),0).*(x>logB)+Rb*(x<=logB);
end
v_opt1=payoff;
v_opt2=payoff;
for i=M-1:-1:0
    v_opt1_aux=real(ifft(fft(v_opt1).*phi11+fft(v_opt2).*phi12));
    v_opt2_aux=real(ifft(fft(v_opt1).*phi21+fft(v_opt2).*phi22));
    if strcmp(type,'Up')
        v_opt1=v_opt1_aux.*(x<logB)+Rb*(x>=logB);    % Up & Out
        v_opt2=v_opt2_aux.*(x<logB)+Rb*(x>=logB);
    else
```

```
59          v_opt1=v_opt1_aux.*(x>logB)+Rb*(x<=logB);     % Down & Out
60          v_opt2=v_opt2_aux.*(x<logB)+Rb*(x>=logB);
61      end
62  end
63  Prices=zeros(2,1);
64  Prices(1)=interp1(S,v_opt1,S0,'pchip');
65  Prices(2)=interp1(S,v_opt2,S0,'pchip');
66
67  end
```

C.3.3 PDE method

```
1   function Prices=PDE_Vanilla_2RS(N,M,alphaCP,S0,T,r,delta, ...
2       K,Q,Theta,params)
3   % INPUTS
4   % N:                  Number of grid points
5   % M:                  Number of time steps
6   % alphaCP:            1 for a Call, -1 for a Put
7   % S0:                 Spot price
8   % T:                  Time-to-maturity
9   % r:                  Vector with the risk-free interest rates
10  %                     in the 2 states
11  % delta:              Vector with the dividend yields
12  %                     in the 2 states
13  % K:                  Strike price
14  % Q:                  Intensity matrix
15  % Theta:              0 for Forward Euler, 0.5 for Crank Nicolson,
16  %                     1 for Backward Euler
17  % params:             Cell array with the parameters of
18  %                     [model1,model2]
19  %
20  % OUTPUTS
21  % Prices:             Prices in the 2 states of the Vanilla
22  %                     option obtained with the PDE method
23  %                     under a regime-switching model
24
25  sigma1=params{1};
26  sigma2=params{2};
27
28  % Grids
29  xmin1=(r(1)-delta(1)-sigma1^2/2)*T-6*sigma1*sqrt(T);
30  xmax1=(r(1)-delta(1)-sigma1^2/2)*T+6*sigma1*sqrt(T);
31  xmin2=(r(2)-delta(2)-sigma2^2/2)*T-6*sigma2*sqrt(T);
32  xmax2=(r(2)-delta(2)-sigma2^2/2)*T+6*sigma2*sqrt(T);
33  xmin=min(xmin1,xmin2);
```

```
34 xmax=max(xmax1,xmax2);
35 dx=(xmax-xmin)/N;
36 dt=T/M;
37 x=linspace(xmin,xmax,N+1)';
38
39 % Matrices
40 alpha=@(r,delta,sigma,const) const* ...
41     (-(r-delta-sigma^2/2)/(2*dx)+sigma^2/(2*dx^2));
42 alpha1=alpha(r(1),delta(1),sigma1,1-Theta);
43 alpha2=alpha(r(2),delta(2),sigma2,1-Theta);
44 alphac1=alpha(r(1),delta(1),sigma1,-Theta);
45 alphac2=alpha(r(2),delta(2),sigma2,-Theta);
46
47 beta=@(r,sigma,const,lambda) -1/dt+const* ...
48     (-sigma^2/dx^2-(r+lambda));
49 beta1=beta(r(1),sigma1,1-Theta,Q(1,2));
50 beta2=beta(r(2),sigma2,1-Theta,Q(2,1));
51 betac1=beta(r(1),sigma1,-Theta,Q(1,2));
52 betac2=beta(r(2),sigma2,-Theta,Q(2,1));
53
54 gamma=@(r,delta,sigma,const) const* ...
55     ((r-delta-sigma^2/2)/(2*dx)+sigma^2/(2*dx^2));
56 gamma1=gamma(r(1),delta(1),sigma1,1-Theta);
57 gamma2=gamma(r(2),delta(2),sigma2,1-Theta);
58 gammac1=gamma(r(1),delta(1),sigma1,-Theta);
59 gammac2=gamma(r(2),delta(2),sigma2,-Theta);
60
61 A_1=alpha1*ones(N-1,1); A_2=alpha2*ones(N-1,1);
62 B_1=beta1*ones(N-1,1); B_2=beta2*ones(N-1,1);
63 C_1=gamma1*ones(N-1,1); C_2=gamma2*ones(N-1,1);
64 M1_1=spdiags([A_1 B_1 C_1],[-1 0 1],N-1,N-1);
65 M1_2=spdiags([A_2 B_2 C_2],[-1 0 1],N-1,N-1);
66
67 A_1=alphac1*ones(N-1,1); A_2=alphac2*ones(N-1,1);
68 B_1=betac1*ones(N-1,1); B_2=betac2*ones(N-1,1);
69 C_1=gammac1*ones(N-1,1); C_2=gammac2*ones(N-1,1);
70 M2_1=spdiags([A_1 B_1 C_1],[-1 0 1],N-1,N-1);
71 M2_2=spdiags([A_2 B_2 C_2],[-1 0 1],N-1,N-1);
72
73 M3_1=diag((1-Theta)*Q(1,2)*ones(N-1,1));
74 M3_2=diag((1-Theta)*Q(2,1)*ones(N-1,1));
75 M4_1=diag((-Theta)*Q(1,2)*ones(N-1,1));
76 M4_2=diag((-Theta)*Q(2,1)*ones(N-1,1));
77
78 % Backward in time Loop
79 V_1=max(alphaCP*(S0*exp(x(2:end-1))-K),0);
```

```
80  V_2=max(alphaCP*(S0*exp(x(2:end-1))-K),0);
81  BC_1=zeros(N-1,1);
82  BC_2=zeros(N-1,1);
83  for l=M-1:-1:0
84      if alphaCP==1
85          BC_1(end)=-gamma1*(S0*exp(xmax)-K*exp(-r(1)*(T-l*dt)))...
86              +gammac1*(S0*exp(xmax)-K*exp(-r(1)*(T-(l+1)*dt)));
87          BC_2(end)=-gamma2*(S0*exp(xmax)-K*exp(-r(2)*(T-l*dt)))...
88              +gammac2*(S0*exp(xmax)-K*exp(-r(2)*(T-(l+1)*dt)));
89      else
90          BC_1(1)=-alpha1*(K*exp(-r(1)*(T-l*dt))-S0*exp(xmin))...
91             +alphac1*(K*exp(-r(1)*(T-(l+1)*dt))-S0*exp(xmin));
92          BC_2(1)=-alpha2*(K*exp(-r(2)*(T-l*dt))-S0*exp(xmin))...
93              +alphac2*(K*exp(-r(2)*(T-(l+1)*dt))-S0*exp(xmin));
94      end
95      D_1=M2_1*V_1+M4_1*V_2;
96      D_2=M2_2*V_2+M4_2*V_1;
97      V_1=(M1_2*M1_1-M3_1*M3_2)\(M1_2*(D_1+BC_1)-M3_1*(D_2+BC_2));
98      V_2=(M3_2*M3_1-M1_1*M1_2)\(M3_2*(D_1+BC_1)-M1_1*(D_2+BC_2));
99  end
100 Prices=zeros(2,1);
101 Prices(1)=interp1(S0*exp(x(2:end-1)),V_1,S0,'spline');
102 Prices(2)=interp1(S0*exp(x(2:end-1)),V_2,S0,'spline');
103
104 end
```

Bibliography

[1] Bandorff-Nielsen, O.E., Normal inverse Gaussian distributions and stochastic volatility modelling. Scand. J. Statist., 1997, Vol. 24, 1-13.

[2] Björk, T., Arbitrage Theory in Continuous Time. Oxford Finance Series. Oxford University Press, Incorporated, 2009.

[3] Carr, P., Geman, H., Madan, D.B. and Yor, M., Stochastic Volatility for Lévy Processes. Math. Finance, 2003, 13, 345-382.

[4] Carr, P., Geman, H., Madan, D.B. and Yor, M., The fine structure of asset returns: An empirical investigation. Journal of Business, 2002, 75, 305-332.

[5] Cont, R., Empirical properties of asset returns: stylized facts and statistical issues. Quantitative Finance Volume 1, 223-236, 2001.

[6] Cont, R. and Tankov P., Financial Modelling with Jump Processes. Chapman & Hall/CRC Financial Mathematics Series, 2004.

[7] Fang, F. and Oosterlee, C.W., A novel pricing method for European options based on Fourier-cosine series expansions. SIAM J. Sci. Comput., 2008, 31, 826-848.

[8] Fang, F. and Oosterlee, C.W., Pricing early-exercise and discrete barrier options by Fourier-cosine series expansions. Numer. Math., 2009, 114, 27-62.

[9] Fusai, G. and Meucci, A., Pricing discretely monitored Asian options under Lévy processes, J. Bank. Finance, 2008, 32, 2076-2088.

[10] Jackson, K.R., Jaimungal, S. and Surkov, V., Fourier space time stepping for option pricing with Lévy models. J. Comput.Finance, 2008, 12, 1-29.

[11] Jacod, J. and Protter, Ph., Probability Essentials, Springer, 2 ed., 2004.

[12] Karlin, S. and Taylor, H. M., A Second Course in Stochastic Processes. Academic Press, New York, 1981.

[13] Kienitz, J. and Wetterau, D., Financial Modelling: Theory, Implementation and Practice with MATLAB Source. Wiley Finance, 2012.

[14] Naik, V., Option valuation and hedging strategies with jumps in the volatility of asset returns. J. Finance, 1993, 48, 1969-1984.

[15] Chang, C.-C., Chung S.-L. and Stapleton R.C., Richardson extrapolation technique for pricing American-style options. J. Futures Markets, 27(8): 791-817, 2007.

[16] Schoutens, W., Lévy Processes in Finance: Pricing Financial Derivatives, 2003 (Wiley).

[17] Tour, G., Thakoor, N., Khaliq, A. Q. M. and D. Y. Tangman, COS method for option pricing under a regime-switching model with time-changed Lévy processes. Quantitative Finance, 2018,18:4, 673-692.

[18] Yin, G. and Zhang, Q., Continuous-time markov chains and applications: a singular perturbation approach. Springer, 1998.

[19] Yuen, F.L. and Yang, H., Option pricing with regime switching by trinomial tree method. J. Comput. Appl. Math., 2010, 233, 1821-1833.

[20] Zhang, B. and Oosterlee, C.W., Efficient Pricing of European-Style Asian Options under Exponential Lévy Processes Based on Fourier Cosine Expansions. SIAM J. Financ. Math., 2013, Vol. 4, 399-426.

[21] Zhang, B. and Oosterlee, C.W., Fourier Cosine expansions and put-call relations for Bermudan options. Numer. Methods Finance, 2012, 12, 323-350.