University of Bergen

Likelihood Estimation of Jump-Diffusions

Extensions from Diffusions to Jump-Diffusions, Implementation with Automatic Differentiation, and Applications

Author:
Berent Å. S. Lunde

Supervisor:
Prof. Hans J. Skaug

A thesis submitted in partial fulfillment for the degree of Master of Science

in the

Faculty of Mathematics and Natural Sciences
Department of Mathematics

June 1, 2016

Abstract

This thesis considers the problem of likelihood-based parameter estimation for time-homogeneous jump-diffusion processes. The problem is that there often is no analytic solution to the stochastic differential equations driving the process. Thus, the transition density of the process is unknown. In this thesis we build on the solution presented in Preston and Wood (2012), where the transition density of a time-homogeneous diffusion process is approximated by a saddlepoint approximation based on the approximated solution following from discretization schemes, which in turn stems from an Itô-Taylor expansion of the stochastic differential equation. The mathematical tools for understanding the method in Preston and Wood (2012) and the extended methods to jump-diffusions are developed. We reproduce the results found here, and extend the analysis with maximum likelihood estimation for benchmark processes such as the geometric Brownian motion, the Ornstein-Uhlenbeck process, the Cox-Ingersoll-Ross process, and the Merton model. We also investigate the use of the renormalized saddlepoint approximation in the context of maximum likelihood estimation. The implementation of the methods is carried out with the newly released parallel programming package, Template Model Builder, which uses automatic differentiation among other things. We therefore give an introduction to the basics of automatic differentiation in the context of our computational problems, and also extend the Template Model Builder package to e.g. allow for complex numbers. Thereafter we apply the methods developed in previous chapters to the analysis of stock prices modelled as nonlinear stochastic differential equations, with and without jumps. Finally we briefly analyse some models for stochastic volatility.

Contents

List of Figures

List of Tables

Abbreviations

AD	**A**utomatic **D**ifferentiation
AELD	**A**bsolute **E**rror of the **L**og **D**ensity
CGF	**C**umulative **G**enerating **F**unction
CIR	**C**ox-**I**ngersoll-**R**oss
DFT	**D**iscrete **F**ourier **T**ransform
FFT	**F**ast **F**ourier **T**ransform
FGL	**F**ourier **G**auss **L**aguerre
GBM	**G**eometric **B**rownian **M**otion
GMR	**G**eneral **M**ean **R**everting process
IFT	**I**nverse **F**ourier **T**ransform
iid	**i**ndependent and **i**dentically **d**istributed
ITSPA	**I**tô-**T**aylor **S**addle**P**oint **A**pproximation
JNLL	**J**oint **N**egative **L**og-**L**ikelihood
MGF	**M**oment **G**enerating **F**unction
MJD	**M**erton **J**ump **D**iffusion
MLE	**M**aximum **L**ikelihood **E**stimation
OU	**O**rnstein-**U**hlenbeck
SDE	**S**tochastic **D**ifferential **E**quation
SPA	**S**addle**P**oint **A**pproximation
TMB	**T**emplate **M**odel **B**uilder

Chapter 1

Introduction

Itô calculus, first proposed by Kiyoshi Itô and popularized by the elegant solution to the problem of pricing options proposed in Black and Scholes (1973), has become the focus of many studies. It has applications to many fields of research, such as physics and chemistry, but is perhaps mostly reckoned with in the context of mathematical finance. Stochastic differential equations, governing the Itô process, are used to model a wide variety of objects in mathematical finance, from stock prices to stochastic volatility, as in the Heston stochastic volatility model (Heston, 1993). A good introduction to the subject can be found in Øksendal (2003). Stochastic differential equations are often problematic in the sense that it is, in general, not known how to solve them analytically. This is a problem for pricing formulas in finance, for simulation of the process, and for inference about parameters. A solution is to estimate the solution with a series expansion of the driving equation, similar to that of the familiar Taylor expansions, indeed these series expansions are called *Itô-Taylor expansions*. A rigorous development and application of the Itô-Taylor expansions can be found in Kloeden and Platen (1992).

Models in finance are often modelled under the hypothesis that markets are efficient. This implies that all valuable information for a stock is already embedded in the stock price. In practice, the expectation of future values (and all other aspects) of the price of the stock is the same whether you condition on the current state, or whether you condition on all previous states. For these reasons, models will typically have the Markov property, and in the case of continuous time models, they are typically specified by the time-homogeneous stochastic differential equation:

$$dX_t = \mu(X_t;\theta)dt + \sigma(X_t;\theta)dW_t. \tag{1.0.1}$$

Given discrete observations $\{X_{t_i}\}$ of the process, where i ranges from $i = 1, \dots, n$, and due to the Markov property of the Itô process, the log-likelihood can be written as:

$$l(\theta|x_{t_1}, ..., x_{t_n}) = \sum_{i=2}^{n} \log p(x_{t_i}|x_{t_{i-1}}, \theta), \tag{1.0.2}$$

where

$$p(x_{t_i}|x_{t_{i-1}}, \theta) = \frac{d}{dx_{t_i}} P(X_{t_i} < x_{t_i}|X_{t_{i-1}} = x_{t_{i-1}}, \theta) \tag{1.0.3}$$

is the all important transition density (Preston and Wood, 2012; Lindström, 2007).

The problem of doing inference about parameters of processes where the transition density is not known, has a variety of solutions. Likelihood-based estimation needs the transition density to build the likelihood, and various methods have therefore been developed, either to approximate it or to find it exactly. Preston and Wood (2012) place these approaches into three categories. The first approach involves obtaining the transition density via the Kolmogorov equations for the transition density (Lindström, 2007). The second involves simulating the process, either approximately (Durham and Gallant, 2002) or exact (Beskos et al., 2006). A third alternative is to replace the continuous process with a discrete approximation where it is possible to find the transition density (Shoji and Ozaki, 1998; Aït-Sahalia, 1999). This third solution is well applicable when the time steps between the observations are small, and this bodes well for financial data.

In this thesis we follow and extend the work done in Preston and Wood (2012), which falls into the third category: replacing the continuous process with a discrete version. The method in this paper can be broken down into the following three steps:

1. Develop an Itô-Taylor expansion of the sample path.
2. Calculate the moment generating function of the retained terms in the expansion.
3. Approximate the inverse Fourier transform by a saddlepoint approximation to the transition density.

The extension of this method to a time-homogeneous jump-diffusion process is possible due to the independence between the jump components and the pure diffusion parts. Such an extension is already done by Zhang and Schmidt (2016), where the approximation in step 3 is carried out by the FFT algorithm.

The two chapters following the introduction are concerned with the mathematical tools necessary to understand the method in Preston and Wood (2012), which we shall call the *Itô-Taylor saddlepoint approximation* (ITSPA). In chapter 2 we explain the Itô-Taylor expansions, used for expanding the sample path of the process, and in chapter 3 we discuss approximations of the inverse Fourier transform of the characteristic function, such as the saddlepoint approximation. We continue in chapter 4 with the extensions of the ITSPA to jump-diffusions, and the refinement of the method in Zhang and Schmidt (2016). In chapter 5 we discuss the benefits of the newly developed Template Model Builder (TMB) package, which by utilizing the techniques of automatic differentiation allows for exact numerical derivatives of the likelihood, making the optimization faster and more robust. We also extend the TMB package with some features, allowing us to implement the methods in chapter 4. Chapter 6 presents numerical results for benchmark processes using the implemented methods. Plotted transition densities and numerical results from likelihood-based analysis are used to compare the discretization schemes, the methods, and refinements such as renormalization of the saddlepoint approximation. Chapter 7 is devoted to two case studies: the study of stock prices as a nonlinear process versus a linear process, and comparisons of stochastic volatility models. In chapter 8 we conclude and comment on the results.

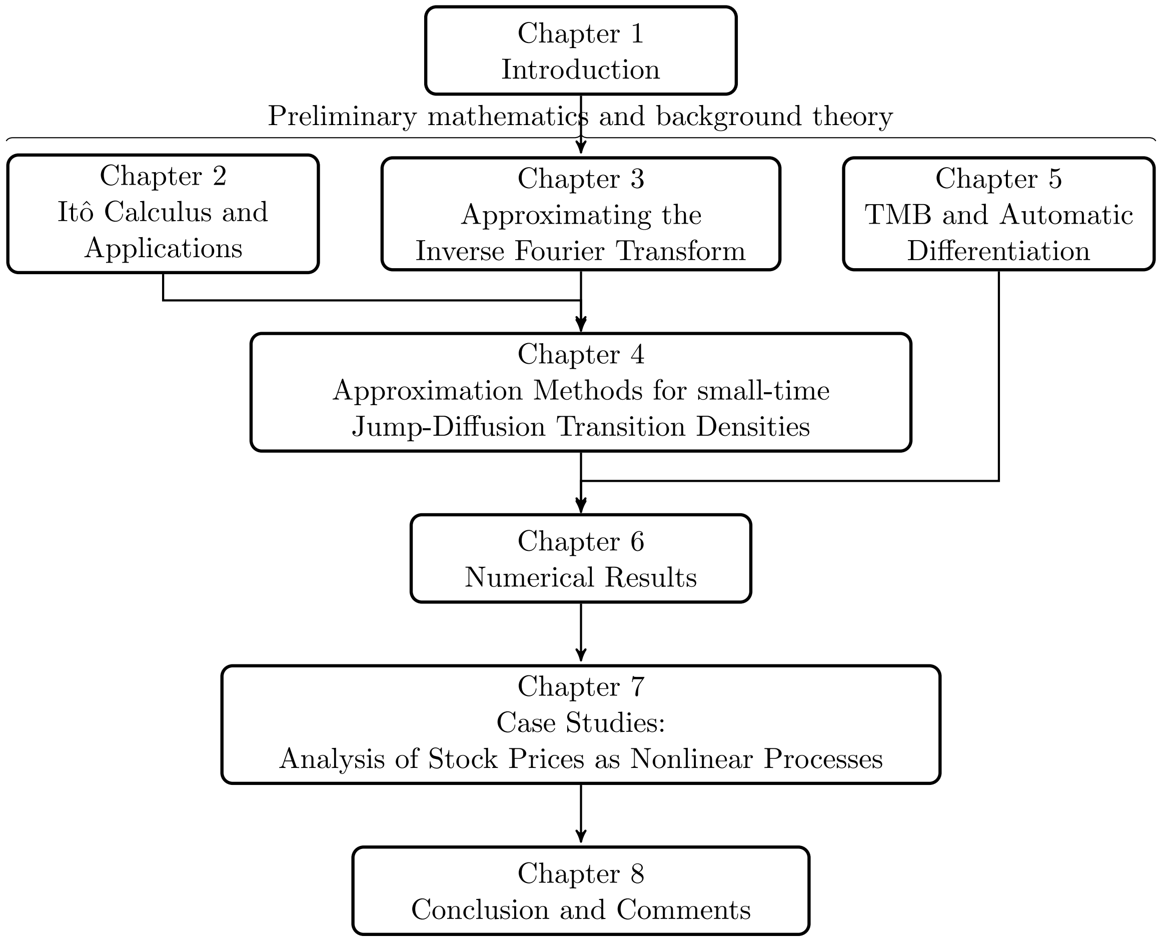
Chapter 1
Introduction
Preliminary mathematics and background theory
Chapter 2
Itô Calculus and Applications
Chapter 3
Approximating the Inverse Fourier Transform
Chapter 5
TMB and Automatic Differentiation
Chapter 4
Approximation Methods for small-time Jump-Diffusion Transition Densities
Chapter 6
Numerical Results
Chapter 7
Case Studies: Analysis of Stock Prices as Nonlinear Processes
Chapter 8
Conclusion and Comments

FIGURE 1.1: Structure of the thesis.

Chapter 2

Itô Calculus and Applications

In this chapter we will go through the first part of the mathematical preliminaries necessary to understand the Itô-Taylor expansions. We start with the building block of Itô calculus, the *Brownian motion*. Further we discuss the Itô integral, stochastic differential equations and Itô's lemma. Some applications of Itô's lemma are shown, solving the stochastic differential equations for benchmark processes that will be used later in this thesis. The extension to jump-diffusion is discussed in the context of the Merton jump-diffusion model. We continue with explaining the concept of the Itô-Taylor expansions. Throughout this thesis we shall limit ourselves to one-dimensional problems.

2.1 A Brief Introduction to Itô Calculus

Consider the *stochastic differential equation* (SDE) (1.0.1), which is shorthand for the integral equation,

$$X_t = X_{t_0} + \int_{t_0}^{t} \mu(X_s;\theta)ds + \int_{t_0}^{t} \sigma(X_s;\theta)dW_s. \tag{2.1.1}$$

The first integral is a standard deterministic integral, while the second is the *Itô Integral*. To define the Itô integral, we first need to define the "noise" term W_t. The building block of *Itô Calculus* is the *Brownian motion*, W_t, defined as follows:

Definition 2.1.1 (standard Brownian motion). A stochastic process, $\{W_t : 0 \leq t \leq \infty\}$, is a standard Brownian motion if

1. $W_0 = 0$,
2. it has continuous sample paths,
3. it has independent, normally-distributed increments,
4. $W_{t+s} - W_t \sim N(0, s)$ for $s > 0$.

From now on, a Brownian motion is assumed to be a standard Brownian motion if nothing else is mentioned, and it is denoted by W_t.

When defining an integral of a function $g(x)$ with respect to a Brownian motion, care must be taken of where to evaluate the function. The standard way of defining an integral is to define a partition from the respective integration area, evaluate the function at some point in each subinterval, sum up the function value times the length of each subinterval and evaluate the limit. It can be shown that, for a nondeterministic integrator, it is not trivial where one chooses to evaluate the function in the subinterval. Itô's choice is to evaluate at the left endpoint of each subinterval. We now define the Itô integral:

Theorem 2.1. *Let f be a right-continuous, adapted, and locally bounded process (Øksendal, 2003, p. 25), let $\{\phi_n\}$ be a sequence of elementary functions:*

$$\phi(t, w) = \sum_j e_j(w) \mathbb{1}_{[t_j, t_{j+1})}(t), \tag{2.1.2}$$

so that

$$\mathrm{E}\left[\int_S^T \left(f(t, w) - \phi_n(t, w)\right)^2 dt\right] \to 0, \tag{2.1.3}$$

as $n \to \infty$. Further, let W_t be a standard Brownian motion and define the integral

$$\int_S^T \phi_n(t, w) dW_t(w) = \sum_{j \geq 0} \phi_j(w) \left[W_{t_{j+1}}(w) - W_{t_j}(w)\right]. \tag{2.1.4}$$

Then the Itô integral of $f(t,w)$ is defined by

$$I[f](w) = \int_S^T f(t,w)dW_t(w) = \lim_{n\to\infty} \int_S^T \phi_n(t,w)dW_t(w), \tag{2.1.5}$$

which can be shown to converge in probability (Øksendal, 2003).

For properties of the Itô integral, see Øksendal (2003, Chapter 3.2, p. 30). For the remainder of this thesis we shall adapt the notation W_t, instead of $W_t(w)$.

We then state one of the most important theorems in Itô calculus, which is repeatedly used when developing the stochastic analogy to Taylor series, *Itô's Lemma.* Consider a general SDE,

$$dX_t = \mu\left(t, X_t\right) dt + \sigma\left(t, X_t\right) dW_t, \tag{2.1.6}$$

and its solution X_t. To handle another stochastic process, defined as a function of the solution, $Y_t = f(t, X_t)$, we need a stochastic analogue to the chain rule. This is precisely what Itô's Lemma gives us.

Theorem 2.2. *Let X_t be an Itô process satisfying the SDE*

$$dX_t = \mu_t dt + \sigma_t dW_t. \tag{2.1.7}$$

If $f(t,x)$ is a twice continuously differentiable function on $[0,\infty)\times\mathbb{R}$, then

$$Y_t = f(t, X_t) \tag{2.1.8}$$

is again an Itô process, and

$$dY_t = \frac{\partial f}{\partial t}(t,X_t)dt + \frac{\partial f}{\partial x}(t,X_t)dX_t + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(t,X_t)\left(dX_t\right)^2, \tag{2.1.9}$$

where $(dX_t)^2$ is computed according to the following rules:

$$dtdt = dtdW_t = dW_tdt = 0,\ dW_tdW_t = dt. \tag{2.1.10}$$

(Øksendal, 2003)

2.2 A Brief Introduction to Jump-Diffusions

A more general process than the Itô process can be obtained by incorporating a jump process component. To this end we first define the *Poisson process.*

Definition 2.2.1. If a stochastic process $\{N(t)\}_{t\geq 0}$ has the following properties:

1. $N(0) = 0$,
2. the process has stationary increments,
3. $P(N(h) = 1) = \lambda h + o(h)$, $h \to 0$ for some $\lambda > 0$,
4. $P(N(h) \geq 2) = o(h)$, $h \to 0$ for some $\lambda > 0$,

where $\lim_{h\to 0} \frac{o(h)}{h} = 0$, then $\{N(t)\}_{t\geq 0}$ is a Poisson process. For convenience, we shall often denote this simply by N_t.

Let N_t denote a Poisson process with intensity $\lambda > 0$, independent of the Brownian motion W_t. We could then express the governing dynamics of a more general process with the SDE

$$dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t + c(t, X_{t^-}, \xi_{N_{t^-}+1})dN_t \tag{2.2.1}$$

for $t \geq 0$, with initial value X_0, and with c determining the jump size in the case of a jump (Platen and Bruti-Liberati, 2010). The subscript in N_{t^-} is used to indicate the left limit of the interval before the N_t'th jump occurs. Such a process is known as a *jump-diffusion* process.

For our purpose we consider models where $c(X_{t^-}, \xi_{N_{t^-}+1})dN_t = dY_t$ and Y_t is a compounded Poisson process, such that

$$Y_t = \sum_{k=1}^{N_t} Z_k, \tag{2.2.2}$$

and thus

$$dY_t = Z_{N_{t^-}+1} dN_t, \tag{2.2.3}$$

where the Z_i's are i.i.d. and independent of the Poisson process.

Due to the jump part of the SDE, we need an extension of Itô's lemma 2.2. This can be found in Tankov (2003) and reads:

Theorem 2.3. *A stochastic process Y_t defined as a function $Y_t = f(t, X_t)$ of a general jump-diffusion process*

$$X_t = X_0 + \int_0^t b_s ds + \int_0^t \sigma_s dW_s + \sum_{i=1}^{N_t} \Delta X_{\tau_i},$$

will be governed by the following SDE

$$\begin{aligned} dY_t = &\left(\frac{\partial f(t, X_t)}{\partial t} + b_t \frac{\partial f(t, X_t)}{\partial x} + \frac{\sigma_t^2}{2} \frac{\partial^2 f(t, X_t)}{\partial x^2} \right) dt \\ &+ \sigma_t \frac{\partial f(t, X_t)}{\partial x} dW_t + \left[f\left(X_{t^-} + \Delta X_t \right) - f\left(X_{t^-} \right) \right] dN_t, \end{aligned} \tag{2.2.4}$$

assuming coefficients b_t and σ_t are sufficiently smooth.

Here, the time τ_i is the i'th jump time, such that $\Delta X_{\tau_i} = X_{\tau_i} - X_{\tau_i^-}$.

For a rigorous development of jump-diffusion processes we refer to Tankov (2003).

2.3 Benchmark Processes

We will now consider some standard applications of Itô's lemma 2.2 to some benchmark processes. We will use these processes throughout the thesis for testing accuracy of estimation methods.

2.3.1 Geometric Brownian Motion

In the derivation of the famous Black and Scholes formula geometric Brownian motion (GBM) is used to model the dynamics of asset prices. Suppose a stock price, X_t, satisfies the SDE

$$dX_t = \mu_t X_t dt + \sigma_t X_t dW_t. \tag{2.3.1}$$

To solve the SDE, define $Y_t = f(t, X) = \log(X_t)$ and apply Itô's lemma (2.2).The partial derivatives are:

$$\frac{\partial f}{\partial t}(t, X_t) = 0,\ \frac{\partial f}{\partial x}(t, X_t) = \frac{1}{X_t},\ \frac{\partial^2 f}{\partial^2 x}(t, X_t) = -\frac{1}{X_t^2}. \tag{2.3.2}$$

Then:

$$dY_t = \left(\frac{\mu_t X_t}{X_t} - \frac{\sigma_t^2 X_t^2}{2X_t^2}\right) dt + \frac{\sigma_t X_t}{X_t} dW_t, \tag{2.3.3}$$

or in integral form:

$$Y_t = X_0 + \int_0^t \left(\mu_s - \frac{\sigma_s^2}{2}\right) ds + \int_0^t \sigma_s dW_s. \tag{2.3.4}$$

Therefore, by definition of Y_t, we have

$$X_t = X_0 \exp\left[\int_0^t \left(\mu_s - \frac{\sigma_s^2}{2}\right) ds + \int_0^t \sigma_s dW_s\right], \tag{2.3.5}$$

Note that if μ_t and σ_t are constants, i.e.: $\mu_t = \mu$ and $\sigma_t = \sigma$, we have

$$X_t = X_0 \exp\left[\left(\mu - \frac{\sigma^2}{2}\right) t + \sigma W_t\right] \Rightarrow Y_t = \log(X_t) \sim N\left[\left(\mu - \frac{\sigma^2}{2}\right) t, \sigma^2 t\right].$$

We note that holding the drift and diffusion coefficient constant is what is done in the derivation of the Black and Scholes formula (Black and Scholes, 1973).

2.3.2 Ornstein-Uhlenbeck Process

The Ornstein-Uhlenbeck process (OU) is uniquely defined by the SDE

$$dX_t = \kappa(\mu - X_t)dt + \sigma dW_t. \tag{2.3.6}$$

It is a "mean reverting process", which means that over time the process tends to drift towards its long-term mean (Rampertshammer, 2007), it is often used to model interest rates, currency exchange rates, and stochastic volatility. To solve the SDE, we apply Itô's lemma to the function $f(t, x) = xe^{\kappa t}$ to obtain

$$\begin{aligned} df(t, X_t) &= \kappa X_t e^{\kappa t} dt + e^{\kappa t} dX_t \\ &= e^{\kappa t}\kappa\mu dt + \sigma e^{\kappa t} dW_t. \end{aligned} \tag{2.3.7}$$

Writing this equation in integral form yields

$$f(t, X_t) = X_0 + \int_0^t e^{\kappa s}\kappa\mu ds + \int_0^t \sigma e^{\kappa s} dW_s, \tag{2.3.8}$$

and hence, from the definition of $f(t, X_t)$, we have our solution

$$X_t = X_0 e^{-\kappa t} + \mu(1 - e^{-\kappa t}) + e^{-\kappa t}\int_0^t \sigma e^{\kappa s} dW_s, \tag{2.3.9}$$

which is normal, from the properties of the Itô integral. The first moment is readily calculated to be

$$\mathrm{E}[X_t] = X_0 e^{-\kappa t} + \mu(1 - e^{-\kappa t}), \tag{2.3.10}$$

and by using the Itô isometry we can calculate the covariance function:

$$\begin{aligned}\text{Cov}(X_s, X_t) &= \text{E}[(X_s - \text{E}[X_s])(X_t - \text{E}[X_t])] \\ &= \text{E}\left[\int_0^s \sigma e^{\kappa(u-s)} dW_u \int_0^t \sigma e^{\kappa(v-t)} dW_v\right] \\ &= \sigma^2 e^{-\kappa(s+t)} \text{E}\left[\int_0^s e^{\kappa u} dW_u \int_0^t e^{\kappa v} dW_v\right] \\ &= \frac{\sigma^2}{2\kappa} e^{-\kappa(s+t)} \left(e^{2\kappa \min\{s,t\}} - 1\right). \qquad (2.3.11)\end{aligned}$$

Choosing $\min\{s,t\} = s$, we obtain

$$\text{Cov}(X_s, X_t) = \frac{\sigma^2}{2\kappa} \left(e^{-\kappa(t-)} - e^{-\kappa(t+s)}\right), \qquad (2.3.12)$$

and we find that X_t is normal with expectation $X_0 e^{-\kappa t} + \mu(1 - e^{-\kappa t})$ and variance $\frac{\sigma^2}{2\kappa}\left(1 - e^{-2\kappa t}\right)$.

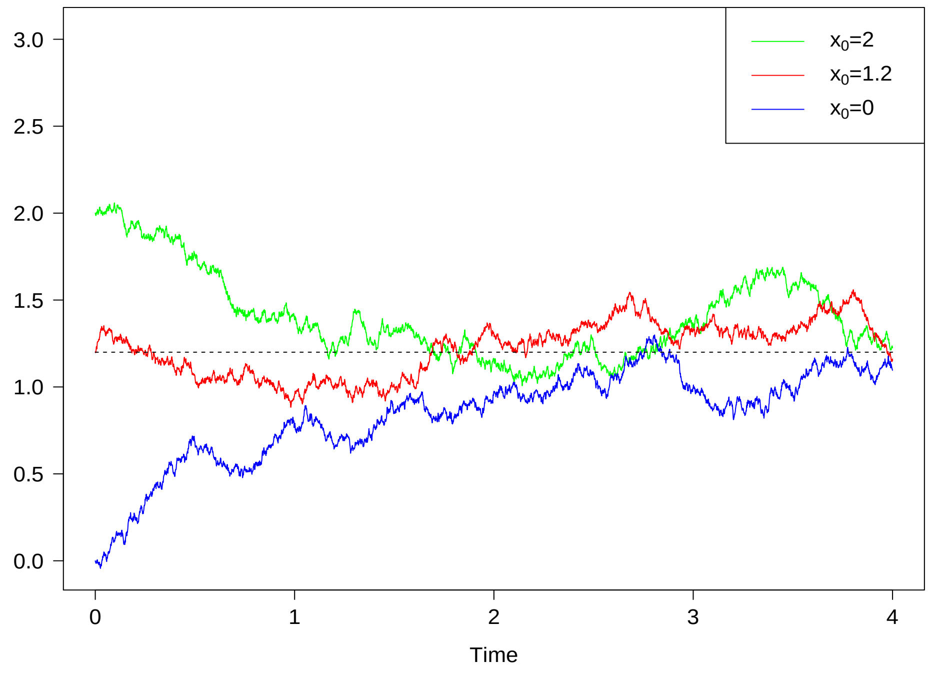

FIGURE 2.1: Three sample paths of different OU processes with $\kappa = 1$, $\mu = 1.2$, and $\sigma = 0.3$, but with different initial values. Simulation was done in R, using the solution of the SDE.

2.3.3 Cox-Ingersoll-Ross Process

As stated earlier, the OU process is often used to model interest rates. However, as we have seen, the solution is normal and can therefore take negative values. This is for obvious reasons not a good model for interest rates. The Cox-Ingersoll-Ross process (CIR) takes non-negative or only positive values depending on the parameters. Like the OU process it is often used to model interest rates (Cox et al., 1985), but also volatility, as is the case in the Heston stochastic volatility model (Heston, 1993).

The CIR process is defined by the SDE

$$dY_t = \kappa(\alpha - Y_t)dt + \sigma\sqrt{Y_t}dW_t. \tag{2.3.13}$$

We will here show that for some parameter choices, the CIR process can be defined in terms of OU processes[1]. Consider a d-dimensional vector of Brownian motions $(W_1, W_2, ..., W_d)^T$ and constants $\kappa > 0$, $\sigma > 0$, then for $j = 1, 2, ..., d$ let X_j be the solution to the SDE

$$dX_t^{(j)} = -\frac{1}{2}\kappa X_t^{(j)}dt + \frac{1}{2}\sigma dW_t^{(j)}, \tag{2.3.14}$$

which is an OU process with drift $-\frac{1}{2}\kappa X_t^{(j)}$ and diffusion $\frac{1}{2}\sigma$. Define the function

$$f(x_1, x_2, ..., x_d) = x_1^2 + x_2^2 + \cdots + x_d^2, \tag{2.3.15}$$

then the partial derivatives are $\frac{\partial f}{\partial x_i} = 2x_i$ and

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \begin{cases} 2 & \text{if } i = j \\ 0 & \text{if } i \neq j. \end{cases} \tag{2.3.16}$$

[1] The essence of the following account is taken from `http://janroman.dhis.org/finance/Books%20Notes%20Thesises%20etc/Shrive%20Finance/chap31.pdf`. In textbooks we have found no equivalent account.

Applying this to our OU processes using the multidimensional version of Itô's lemma 2.2 that can be found in most textbooks on the subject (e.g. Øksendal (2003)), we obtain

$$\begin{aligned}
Y_t &= \sum_{i=1}^{d} \frac{\partial f}{\partial x_i} dX_t^{(i)} + \frac{1}{2}\sum_{i=1}^{d} \frac{\partial^2 f}{\partial x_i \partial x_j} dX_t^{(i)} dX_t^{(i)} \\
&= \sum_{i=1}^{d} dX_t^{(i)} \left(-\frac{1}{2}\kappa X_t^{(i)} dt + \frac{1}{2}\sigma dW_t^{(i)} \right) + \sum_{i=1}^{d} \frac{1}{4}\sigma^2 dW_t^{(i)} dW_t^{(i)} \\
&= -\kappa Y_t dt + \sigma \sum_{i=1}^{d} X_t^{(i)} dW_t^{(i)} + \frac{d\sigma^2}{4} dt \\
&= \left(\frac{d\sigma^2}{4} - \kappa Y_t \right) dt + \sigma \sqrt{Y_t} \sum_{i=1}^{d} \frac{X_t^{(i)}}{\sqrt{f(t)}} dW_t^{(i)}. \qquad (2.3.17)
\end{aligned}$$

Defining

$$W_t = \sum_{i=1}^{d} \int_0^t \frac{X_s^{(i)}}{\sqrt{f(s)}} dW_s^{(i)}, \qquad (2.3.18)$$

and by noticing that

1. W_t is a martingale,
2. $dW_t = \sum_{i=1}^{d} \frac{X_t^{(i)}}{\sqrt{f(t)}} dW_t^{(i)}$,
3. $dW_t dW_t = dt$,

then W_t is a Brownian motion due to the fact that a martingale is a Brownian motion if and only if its quadratic variation is equal to the length of the time interval. We can now write

$$dY_t = \left(\frac{d\sigma^2}{4} - \kappa Y_t \right) dt + \sigma \sqrt{Y_t} dW_t, \qquad (2.3.19)$$

which is a CIR process (2.3.13) with $\alpha = \frac{d\sigma^2}{4\kappa}$. Thus, in the special case when d is an integer, we have the representation $Y_t = f(t) = \sum_{i=1}^{d} X_i^2(t)$. It is often required that $d \geq 2$, since this makes it impossible for the CIR process to take on the value zero (Cox et al., 1985).

Let us now derive the distribution of the CIR process in the special case of an integer valued d. Given starting values $t \geq 0,\ f(0) \geq 0$,

$$X_0^{(1)} = X_0^{(2)} = \cdots X_0^{(d-1)} = 0,\ X_0^{(d)} = \sqrt{Y_0}.$$

From the solution of the SDE governing the OU process (2.3.9), we have

$$X_t^{(i)} \sim N\left(0, \frac{\sigma^2}{4\kappa}\left(1-e^{-\kappa t}\right)\right) \tag{2.3.20}$$

and

$$X_t^{(d)} \sim N\left(e^{-\frac{1}{2}\kappa t}\sqrt{Y_t}, \frac{\sigma^2}{4\kappa}\left(1-e^{-\kappa t}\right)\right), \tag{2.3.21}$$

and write $\rho(t,t)$ for the variances. Then we have by definition

$$\frac{Y_t}{\rho(t,t)} = \sum_{i=1}^{d-1}\left(\frac{X_t^{(i)}}{\sqrt{\rho(t,t)}}\right)^2 + \left(\frac{X_t^{(d)}}{\sqrt{\rho(t,t)}}\right)^2. \tag{2.3.22}$$

The first term is chi-square distributed with $d-1$ degrees of freedom, and the second term is independently noncentral chi-square distributed with one degree of freedom and noncentrality parameter $\lambda = \frac{e^{-\kappa t}\sqrt{Y_0}}{\rho(t,t)}$. Finally, we now have the representation

$$Y_t | Y_0 = g(\chi) = \frac{\chi_t}{2c}, \tag{2.3.23}$$

where χ_t is distributed according to the noncentral chi-squared distribution as mentioned above, and $c = 2\kappa/(\sigma^2(1-e^{-\kappa t}))$. This representation holds even when d is not an integer (Cox et al., 1985). Using this, and the density of a noncentral chi-squared random variable (3.4.2), we find the distribution of the future values of the CIR process:

$$\begin{aligned} f_{Y_t}(y_t) &= f_{Y_t}\left(g^{-1}(y_t)\right)\left|\frac{\partial}{\partial y_t} g^{-1}(y_t)\right| \\ &= ce^{-\frac{2cy_t + 2cy_0 e^{-\kappa t}}{2}}\left(\frac{2cy_t}{2cy_0 e^{-\kappa t}}\right)^{\frac{\kappa\alpha}{\sigma^2}-\frac{1}{2}} I_{\frac{2\kappa\alpha}{\sigma^2}-1}\left(\sqrt{2cy_0 e^{-\kappa t} 2cy_t}\right) \\ &= ce^{-u-v}\left(\frac{v}{u}\right)^{\frac{q}{2}} I_q\left(2\sqrt{uv}\right), \end{aligned} \tag{2.3.24}$$

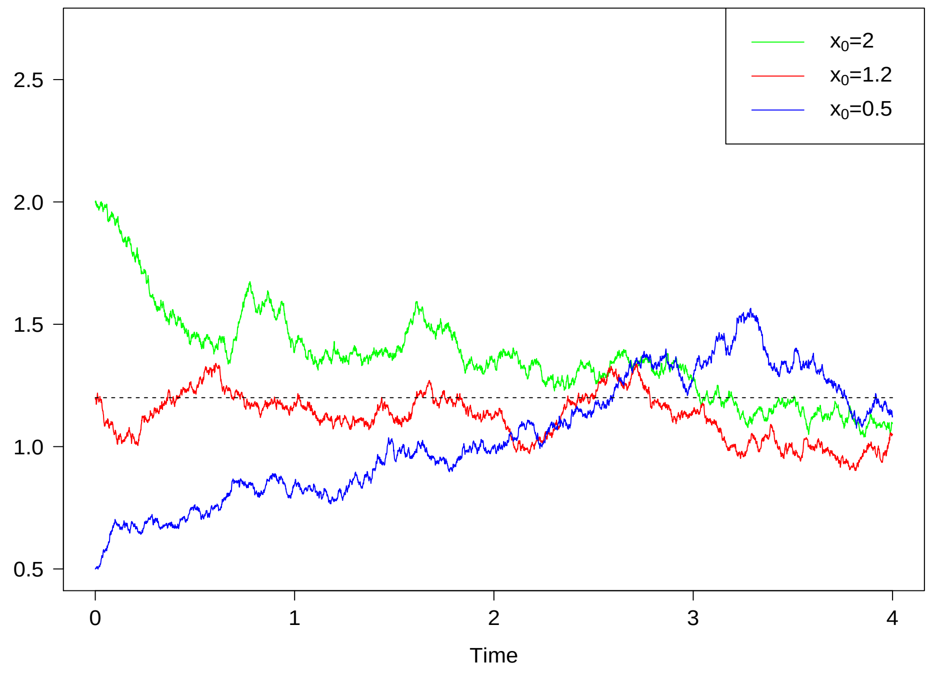

FIGURE 2.2: Three sample paths of different CIR-processes with $\kappa = 1$, $\alpha = 1.2$, and $\sigma = 0.2$, but with different initial values. Simulation was done in R, using the relationship with the non-central chi-squared distribution.

where $q = 2\kappa\alpha/\sigma^2 - 1$, $u = cY_0e^{-\kappa t}$, and $v = cY_t$.

2.3.4 Merton Jump-Diffusion

Merton jump-diffusion (MJD) is a jump-diffusion process (2.2.1) for stock prices, and the straight forward extension of the GBM diffusion to a jump-diffusion. It was presented in the paper Merton (1976) as an alternative to the GBM model for stock prices, as a way of incorporating larger price jumps (in the context of observed prices) than the lognormal distribution allows. The MJD is governed by the SDE

$$\frac{dS_t}{S_{t^-}} = (r - \lambda k)dt + \sigma dW_t + (Y_t - 1)dN_t, \tag{2.3.25}$$

where r is the instantaneous expected return on the asset, σ the instantaneous volatility if a jump does not occur, and k the expected relative jump size. The assumptions on W_t and N_t are as in 2.2.1. Y_t is the "absolute price jump size", meaning that, if a jump occurs, then

S_{t^-} jumps to $Y_t S_t$. It is assumed to be lognormally distributed with $\log(Y_t) \sim N(\mu, \nu^2)$. $Y_t - 1$ then becomes the relative price jump size, since $\frac{Y_t S_t - S_t}{S_t} = Y_t - 1$ (Matsuda, 2004).

To solve this SDE, we investigate the stochastic process Z_t defined as $Z_t = \log(S_t)$, using Itô's lemma for jump-diffusions (2.3). We have

$$\begin{aligned} dZ_t &= \left(\frac{(r - \lambda k) S_t}{S_t} - \frac{\sigma^2 S_t^2}{2} \frac{1}{S_t^2} \right) dt + \frac{\sigma S_t}{S_t} dW_t + [\log Y_t S_t - \log S_t] \, dN_t \\ &= \left(r - \lambda k - \frac{\sigma^2}{2} \right) dt + \sigma dW_t + \log Y_t dN_t, \end{aligned} \tag{2.3.26}$$

which means that log returns have the representation

$$Z_t = Z_0 + \left(r - \lambda k - \frac{\sigma^2}{2} \right) t + \sigma W_t + \sum_{i=1}^{N_t} \log Y_t. \tag{2.3.27}$$

Using the law of total probability, we find that the probability density function for the log returns are a Poisson-weighted mixture of normal distributions:

$$P(Z_t - Z_0 = z) = \sum_{i=0}^{\infty} P(N_t = i) P(Z_t - Z_0 = z | N_t = i) \tag{2.3.28}$$

(Matsuda, 2004).

For our purpose of calculating the inverse Fourier transform and carrying out saddlepoint approximations via the MGF, we shall now derive the MGF for the compounded Poisson process. Let $X = \sum_{i=0}^{N_t} \xi_i$, where all $\xi_i \sim N(\mu, \nu^2)$ are independent of one another. Then from the definition of the MGF we have:

$$\begin{aligned} M_X(s) &= \mathrm{E}\left[\exp\left\{ s \sum_{i=0}^{N_t} \xi_i \right\} \right] \\ &= \mathrm{E}\left[\mathrm{E}\left[\exp\left\{ s n_t \xi_i \right\} \middle| \, N_t = n \right] \right] \\ &= \sum_{k=0}^{\infty} \frac{e^{-\lambda t} (\lambda t)^k}{k!} \left(\mathrm{E}\left[e^{\{s \xi_i\}} \right] \right)^k \\ &= \sum_{k=0}^{\infty} \frac{e^{-\lambda t} (\lambda t)^k}{k!} \left(e^{s\mu + \frac{1}{2} \nu^2 s^2} \right)^k \\ &= \exp\left\{ \lambda t \left(e^{s\mu + \frac{1}{2} \nu^2 s^2} - 1 \right) \right\}. \end{aligned} \tag{2.3.29}$$

Due to the independence between the diffusion and the jump part of the SDE, we can easily find the MGF for the logarithmic stock price:

$$M_{Z_t} = M_{Z_{t^-}} M_X, \tag{2.3.30}$$

where $M_{Z_{t^-}}$ is the MGF for a normal random variable with mean $Z_0 + \left(r - \lambda k - \frac{\sigma^2}{2}\right) t$ and variance $\sigma^2 t$.

For the purpose of *identifiability*: that distinct parameters correspond to distinct distributions, we state the following result.

Lemma 2.4. *Consider a compounded Poisson process, X, with normally distributed jumps with parameters μ and ν^2 which are $o\left(\frac{1}{\lambda}\right)$ functions of λ. Further, define*

$$\mu^* = \lim_{\lambda \to \infty} \lambda \mu(\lambda) < \infty \tag{2.3.31}$$

and

$$\nu^{2*} = \lim_{\lambda \to \infty} \lambda \nu^2(\lambda) < \infty. \tag{2.3.32}$$

Then

$$\lim_{\lambda \to \infty} X \sim N\left(t\mu^*, t\nu^{2*}\right). \tag{2.3.33}$$

Proof. If μ and ν^2 are $o\left(\frac{1}{\lambda}\right)$ functions of λ, then the characteristic function converges to the characteristic function of a normal distribution. The characteristic function of X is the complex valued function

$$\phi_X = M_X(is) = \exp\left(\lambda t \left(e^{is\mu - \frac{1}{2}s^2\nu^2} - 1\right)\right). \tag{2.3.34}$$

Let X_n be a sequence of compound Poisson processes with parameters λ_n, μ_n, and ν_n^2. The first central moments of X are $\mathrm{E}[X] = \lambda t \mu$ and $\mathrm{Var}[X] = \lambda t(\mu^2 + \nu^2)$. Consider the standardized random variable $Z_n = (X_n - \lambda t\mu)/\sqrt{\lambda t(\mu^2+\nu^2)}$. We then have:

$$\begin{aligned}
\phi_{Z_n}(s) &= e^{-is\mu_n\sqrt{\frac{\lambda_n t}{\mu_n^2+\nu_n^2}}}\phi_{X_n}\left(\frac{s}{\sqrt{\lambda_n t(\mu_n^2+\nu_n^2)}}\right) \\
&= e^{-is\mu_n\sqrt{\frac{\lambda_n t}{\mu_n^2+\nu_n^2}}} e^{\lambda_n t\left(\frac{i\mu_n s}{\sqrt{\lambda_n t(\mu_n^2+\nu_n^2)}} - \frac{\nu_n^2 s^2}{2\lambda_n t(\mu_n^2+\nu_n^2)} - \frac{\mu_n^2 s^2}{2\lambda_n t(\mu_n^2+\nu_n^2)} + o\left(\frac{1}{\lambda_n^{3/2}}\right)\right)} \\
&= e^{-\frac{1}{2}s^2 + o\left(\frac{1}{\sqrt{\lambda_n}}\right)}, \qquad (2.3.35)
\end{aligned}$$

from which we see that Z_n is standard normally distributed if $\lambda_n \to \infty$. We therefore have the representation

$$\lim_{\lambda_n\to\infty} X_n = \lim_{\lambda_n\to\infty} \sqrt{\lambda_n t\,(\mu_n^2+\nu_n^2)}Z + \lambda_n t\mu_n, \qquad (2.3.36)$$

where Z is standard normal. Now, demanding that the expectation and variance is bounded, we have the conditions

1. $\lim_{\lambda_n\to\infty} \lambda_n\left(\mu_n^2+\nu_n^2\right) < \infty$,
2. $\lim_{\lambda_n\to\infty} \lambda_n\mu_n < \infty$.

The second condition implies that μ_n tends to zero at least as fast as $o\left(\frac{1}{\lambda_n}\right)$. This must also hold for ν_n^2, considering the first condition. Defining μ^* and ν^{2*}, we then have directly from 2.3.36:

$$\lim_{\lambda_n\to\infty} X_n = \sqrt{\nu_n^{2*}t}Z + t\mu^* \sim N(t\mu^*, t\nu^{2*}), \qquad (2.3.37)$$

in distribution, which is our desired result. □

Theorem 2.5. *Let μ^* and ν^{2*} be as in lemma 2.4, and assume the Merton model for stock prices. Then we have for logarithmic returns:*

$$\lim_{\lambda\to\infty} \log\left(\frac{S_t}{S_0}\right) \sim N\left(\left(r - \frac{1}{2}\sigma^2 - \frac{1}{2}\nu^{2*}\right)t, \left(\sigma^2+\nu^{2*}\right)t\right). \qquad (2.3.38)$$

The Merton model is therefore non-identifiable (when the number of jumps grows large).

Proof. The variance follows directly from the fact that the jumps, the number of jumps, and the Brownian motion are independent. The expectation follows when considering the λk, where $k = e^{\mu+\frac{1}{2}\nu^2} - 1$ in the drift term of the MJD model for logarithmic stock prices:

$$\begin{aligned}
\lim_{\lambda\to\infty} \lambda k &= \lim_{\lambda\to\infty} \lambda e^{\mu+\frac{1}{2}\nu^2} - 1 \\
&= \lim_{\lambda\to\infty} \lambda \left(1 + \mu + \frac{1}{2}\nu^2 + \frac{\left(\mu + \frac{1}{2}\nu^2\right)^2}{2} + \ldots - 1 \right) \\
&= \mu^* + \frac{1}{2}\nu^{2*}. \qquad (2.3.39)
\end{aligned}$$

Now, since the compounded Poisson process in lemma 2.4 has drift $t\mu^*$, this cancels with the part in the drift component and we easily obtain the desired result. □

Although non-identifiability is a problem, it certainly is interesting that defining log-returns as a pure compounded Poisson process (which might be natural for tick-by-tick data) in this way will lead to the same (normal) distribution as the GBM model on a larger time scale.

2.4 Itô-Taylor Expansions

We now consider a somewhat different application of Itô's lemma (2.2). It can be used in a recursive manner to obtain expansions, similar to the familiar Taylor expansions, for diffusion processes governed by a SDE. These expansions are called Itô-Taylor expansions and are very valuable in simulation and approximation of solutions of SDEs. We will limit ourselves to sketch how the expansions can be obtained in the case of an one-dimensional SDE. We do however note that expansions exist for multidimensional diffusion processes, and also for SDEs with jumps, see Platen and Bruti-Liberati (2010). For an in-depth study of the Itô-Taylor expansions and their use, we refer to Kloeden and Platen (1992), from which the material in the following section is taken.

Consider the one-dimensional SDE

$$dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t. \qquad (2.4.1)$$

Applying Itô's lemma (2.2) with a function f, we obtain:

$$dY_t = \left(\frac{\partial f}{dt} + \mu\frac{\partial f}{dx^i} + \frac{1}{2}\sigma^2\frac{\partial^2 f}{dx^i x^i}\right)dt + \sigma\frac{\partial f}{dx^i}dW_t, \tag{2.4.2}$$

and choose $f(t,x) = x$ so that

$$dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t, \tag{2.4.3}$$

which in integral form reads

$$X_t = X_{t_0} + \int_{t_0}^{t}\mu(t, X_t)ds + \int_{t_0}^{t}\sigma(t, X_t)dW_s. \tag{2.4.4}$$

Before obtaining the Itô-Taylor series expansions, we define the following operators, to be used throughout the section:

$$L^0 = \frac{\partial}{\partial t} + \mu\frac{\partial}{\partial x} + \frac{1}{2}\sigma^2\frac{\partial^2}{dx^2}, \tag{2.4.5}$$

$$L^1 = \sigma\frac{\partial}{\partial x}. \tag{2.4.6}$$

Applying these operators in theorem 2.2, we get Itô's lemma in operator form:

$$Y_t = Y_{t_0} + \int_{t_0}^{t} L^0 f ds + \int_{t_0}^{t} L^1 f dW_s. \tag{2.4.7}$$

Applying this twice to the right-hand side of equation (2.4.4), first with $f(t,x)$ as $\mu(t,x)$ and then with $f(t,x) = \sigma(t,x)$, we have

$$\begin{aligned} X_t = X_{t_0} &+ \int_{t_0}^{t}\left[\mu(t_0, X_{t_0}) + \int_{t_0}^{s} L^0\mu(\tau, X_\tau)d\tau\right. \\ &\left.+ \int_{t_0}^{s} L^1\mu(\tau, X_\tau)dW_\tau\right] ds + \int_{t_0}^{t}\left[\sigma(t_0, X_{t_0})\right. \\ &\left.+ \int_{t_0}^{s} L^0\sigma(\tau, X_\tau)d\tau + \int_{t_0}^{s} L^j\sigma(\tau, X_\tau)dZ^j_\tau\right] dW_s. \end{aligned} \tag{2.4.8}$$

Let $I_{i_1,i_2,\dots,i_k}$ represent the multiple Itô integral, defined as

$$I_\alpha = \begin{cases} 1 & \text{if } k = 0, \\ \int_{t_0}^t I_{\alpha^-} ds & \text{if } k \geq 1 \text{ and } \alpha_k = 0, \\ \int_{t_0}^t I_{\alpha^-} dW_s & \text{if } k \geq 1 \text{ and } \alpha_k = 1, \end{cases}$$

where $\alpha = (\alpha_1, \alpha_2, ..., \alpha_k)^T$ is a k-dimensional vector of zeros and ones, and α^- denotes the multi-index that can be obtained by deleting the last component of α. For example,

$$I_{0,1,0} = \int_0^t I_{0,1} ds_1 = \int_0^t \int_0^{s_1} I_0 ds_2 ds_1 = \int_0^t \int_0^{s_1} \int_0^{s_2} ds_3 dW_{s_2} ds_1. \tag{2.4.9}$$

Applying this notation, we have

$$\begin{aligned} X_t = X_{t_0} &+ \mu(t_0, X_{t_0}) I_0 + \sigma(t_0, X_{t_0}) I_1 \\ &+ \int_{t_0}^t \left[\int_{t_0}^s L^0 \mu(\tau, X_\tau) d\tau + \int_{t_0}^s L^1 \mu(\tau, X_\tau) dE_\tau \right] ds \\ &+ \int_{t_0}^t \left[\int_{t_0}^s L^0 \sigma(\tau, X_\tau) d\tau + \int_{t_0}^s L^1 \sigma(\tau, X_\tau) dW_\tau \right] dZ_s. \end{aligned} \tag{2.4.10}$$

We continue with the application of Itô's lemma in operator form (2.4.7), now applied to the functions $L^0\mu$, $L^1\mu$, $L^0\sigma$, and $L^1\sigma$, and again to the newly obtained functions $L^0L^0\mu$, $L^1L^0\mu$, $L^0L^1\mu$, $L^1L^1\mu$, $L^0L^0\sigma$, $L^1L^0\sigma$, $L^0L^1\sigma$, and $L^1L^1\sigma$, to acquire the Itô-Taylor expansion

$$\begin{aligned} X_t = X_{t_0} &+ \mu(t_0, X_{t_0}) I_0 + \sigma(t_0, X_{t_0}) I_1 \\ &+ L^0\mu(t_0, X_{t_0}) I_{0,0} + L^1\mu(t_0, X_{t_0}) I_{1,0} \\ &+ L^0\sigma(t_0, X_{t_0}) I_{0,1} + L^1\sigma(t_0, X_{t_0}) I_{1,1} \\ &+ L^0L^0\mu(t_0, X_{t_0}) I_{0,0,0} + L^1L^0\mu(t_0, X_{t_0}) I_{1,0,0} \\ &+ L^0L^1\mu(t_0, X_{t_0}) I_{0,1,0} + L^1L^1\mu(t_0, X_{t_0}) I_{1,1,0} \\ &+ L^0L^0\sigma(t_0, X_{t_0}) I_{0,0,1} + L^1L^0\sigma(t_0, X_{t_0}) I_{1,0,1} \\ &+ L^0L^1\sigma(t_0, X_{t_0}) I_{0,1,1} + L^1L^1\sigma(t_0, X_{t_0}) I_{1,1,1} + R, \end{aligned} \tag{2.4.11}$$

where R denotes the remainder term.

Chapter 3

Approximating the Inverse Fourier Transform

Consider the case where one has time series data generated by a stochastic process. For likelihood-based inference, one needs the transition density to build the likelihood function. But this transition density is not always readily available from the model. This chapter considers the instances where the *characteristic function*, the *moment generating function* (MGF), or the *cumulative generating function* (CGF) of the one step transition is available, but not the transition density. By definition, the transition density is the *inverse Fourier transform* (IFT) of the characteristic function. This transformation can be approximated by the *discrete Fourier transform* (DFT). Another possible solution is to estimate the transition density with a saddlepoint approximation (SPA), which is derived from the MGF or the CGF. In this chapter we discuss these two approximations of the IFT, to obtain the transition density.

3.1 The Fourier Transform

The Fourier transform has applications to a large variety of research such as signal analysis, quantum physics, and probability theory. It relates to probability theory since the characteristic function and the probability density function of a random variable form a Fourier pair. The definition of a Fourier pair is as follows (Kleppe, 2006):

Definition 3.1. Consider the function f on $[-\infty, \infty]$. The *Fourier transform* of f is the function $\hat{f}$ such that

$$\hat{f}(s) = \int_{-\infty}^{\infty} f(x) e^{isx} dx. \tag{3.1.1}$$

Conversely, we say that f is the *inverse Fourier transform* of $\hat{f}$ and satisfy

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(s) e^{-isx} ds. \tag{3.1.2}$$

We say that the functions f and $\hat{f}$ form a *Fourier pair.*

From definition 3.1 we see that the probability density function and the characteristic function of a random variable X form a Fourier pair if we let $f(x) = 0$ when x is outside the probability space of X. We formulate this as a theorem:

Theorem 3.2. *For a random variable X, the probability density function $f_X(x)$ and the characteristic function $\phi_X(s)$ of X form a Fourier pair.*

Proof.

$$\phi_X(s) = \mathrm{E}\left[e^{isX}\right] = \int_{-\infty}^{\infty} e^{isx} f_X(x) dx = \hat{f}_X. \tag{3.1.3}$$

□

In this thesis, we work with random variables for which the probability density is unknown in closed form, but where the characteristic function (or at least an approximation) is. The inverse Fourier transform then has to be approximated numerically. For such instances, the following lemma comes in handy.

Lemma 3.3. *The characteristic function is conjugate symmetric (Kleppe, 2006):*

$$\phi_X(s) = \overline{\phi_X(-s)}. \tag{3.1.4}$$

Proof. From the definition of the characteristic function, we have:

$$\begin{aligned}\phi_X(s) &= \int_{-\infty}^{\infty} f_X(x)e^{ixs}dx \\ &= \int_{-\infty}^{\infty} f_X(x)cos(ixs)dx + i\int_{-\infty}^{\infty} f_X(x)sin(ixs)dx \\ &= \int_{-\infty}^{\infty} f_X(x)cos(-ixs)dx - i\int_{-\infty}^{\infty} f_X(x)sin(-ixs)dx \\ &= \overline{\int_{-\infty}^{\infty} f_X(x)e^{-ixs}dx} \\ &= \overline{\phi_X(-s)}. \end{aligned} \tag{3.1.5}$$

□

The usefulness of lemma 3.3 is due to the fact that, the probability density being a real valued function, the integral can be simplified:

Theorem 3.4. *The inverse Fourier transform of the characteristic function can be simplified as follows:*

$$f_X(x) = \frac{1}{\pi}\int_0^{\infty} \Re\left(\phi_X(s)e^{-isx}\right) ds. \tag{3.1.6}$$

Proof. Since the probability density function is real, it holds that

$$f_X(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \Re\left(\phi_X(s)e^{-isx}\right) ds. \tag{3.1.7}$$

Now, by investigating the integrand, we have by lemma 3.3:

$$\Re\left\{\phi_X(s)e^{-isx}\right\} = \Re\left\{\overline{\overline{\phi_X(s)e^{-isx}}}\right\} = \Re\left\{\overline{\overline{\phi_X(s)}e^{-isx}}\right\} = \Re\left\{\phi_X(-s)e^{isx}\right\}, \tag{3.1.8}$$

which means that the real part of the original integrand is symmetric, implying

$$f_X(x) = \frac{1}{\pi}\int_0^{\infty} \Re\left(\phi_X(s)e^{-isx}\right) ds. \tag{3.1.9}$$

□

As noted earlier, the Fourier transform often has to be approximated. This can be done by the efficient *fast forward Fourier transform* (FFT) algorithm, which utilizes properties such as the one discussed above. For the instances where we do not have the probability density function readily available, we estimate it using the FFT algorithm already implemented in R, and use this as our benchmark instead of the exact. In chapter 6 we shall use our own algorithm that approximates the IFT directly with Gauss-Laguerre quadrature, also making use of theorem 3.4.

3.2 Derivation of the Saddlepoint Approximation

Instead of approximating the IFT directly, there exist approximations that can lead to closed form expressions. In this thesis we consider the *saddlepoint approximation* (SPA) to the density $f_X(x)$. The SPA is often stated in terms of the mean of i.i.d. random variables, where the SPA is the leading term of an asymptotic expansion (similar to the Laplace approximation) (Butler, 2007). We shall, however, limit ourselves to the case of $n = 1$, or, in other words, of only one continuous random variable.

Theorem 3.5. *For a continuous random variable X with CGF K_X and unknown density f_X, the saddlepoint density approximation to $f_X(x)$ is given by*

$$spa\left(f_X; x\right) = \frac{1}{\sqrt{2\pi K_X''(\hat{s})}} \exp\left\{K_X(\hat{s}) - \hat{s}x\right\}, \tag{3.2.1}$$

where $\hat{s} = \hat{s}(x)$ is the saddlepoint, that is the unique solution to the equation

$$K_X'(\hat{s}) = x, \tag{3.2.2}$$

referred to as the saddlepoint equation or the inner problem (Butler, 2007).

Proof. For a random variable X, the *moment generating function* (MGF) $M_X(s)$ is defined as

$$M_X(s) = \int_{-\infty}^{\infty} e^{sx} f_X(x)dx, \tag{3.2.3}$$

where $f_X(x)$ is the probability density function of X. By using the Fourier inversion formula, we can obtain the density f from the MGF:

$$\begin{aligned} f_X(x) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} M_X(is)\exp\{-isx\}dt \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp\{K_X(is) - isx\}\,ds, \end{aligned} \tag{3.2.4}$$

(Goutis and Casella, 1999).

First, we apply a change of variable, $u = it$ to the integral 3.2.4:

$$f_X(x) = \frac{1}{2\pi i}\int_{-i\infty}^{i\infty} e^{K(u)-ux}du, \tag{3.2.5}$$

and note that the value of the integral is unchanged if we integrate through a line parallel to the imaginary axis:

$$\begin{aligned} f_X(x) &= \frac{1}{2\pi i}\int_{\tau-i\infty}^{\tau+i\infty} e^{K_X(u)-ux}du \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{K_X(\tau+iv)-(\tau+iv)x}dv. \end{aligned} \tag{3.2.6}$$

Further, we expand the inner part of the exponential about the point τ, and obtain:

$$K_X(\tau+iv) - (\tau+iv)x = K_X(\tau) - \tau x + \left(K'_X(\tau) - x\right) iv + \sum_{j\geq 2}\frac{K_X^{(j)}(\tau)(iv)^j}{j!}, \tag{3.2.7}$$

and by choosing τ to be the saddlepoint, $\hat{s}$, the second term in the expansion disappears. Then, from using the transformation $y = \sqrt{K''_X(\hat{s})}v$, we have for the right hand side

$$\frac{\exp\{K_X(\hat{s}) - \hat{s}x\}}{2\pi\sqrt{K''_X(\hat{s})}}\int_{-\infty}^{\infty} e^{-\frac{1}{2}y^2+O(y^3)}dy. \tag{3.2.8}$$

Neglecting the $O(y^3)$ term, and from noting that the integrand then is the integrand of the standard normal density which we evaluate, we obtain our desired result (3.5). $\square$

The SPA is a powerful tool to compute accurate approximations to the densities of random variables, but it comes to the cost of computing $\hat{s}(x)$, $K_X(\hat{s}(x), x)$, and $\frac{\partial^2}{\partial s^2}K_X(s,x)\Big|_{\hat{s}(x)}$ for each new value of x. In the next section we will look at some of the properties and the drawbacks of the SPA.

3.3 Renormalization of the Saddlepoint Approximation

The perhaps most serious problem with the SPA is, that for models deviating from the Gaussian, it does not integrate to 1 (w.r.t. x). Indeed, it is the case that the SPA is only exact up to a multiplicative constant for the normal, gamma, and inverse Gaussian densities (Kolassa, 2006). In applications of the SPA, such as maximum likelihood estimation (MLE), where the model deviates substantially from a Gaussian model, the likelihood estimates with the current SPA will not be accurate enough (Kleppe and Skaug, 2008).

One way to deal with this problem is, then, to develop an alternative SPA with non-Gaussian leading terms (Kleppe and Skaug, 2008; Aït-Sahalia et al., 2006). However, in small dimensions it is feasible to do a *renormalization* of the SPA. This basically means to multiply the SPA with a constant c^{-1} so that c is the integral of the SPA (w.r.t. x) over the whole area. The renormalized SPA is the function

$$\mathrm{rnspa}\,(f_X;x) = \frac{\mathrm{spa}\,(f_X;x)}{\int \mathrm{spa}\,(f_X;x)\,dx}. \tag{3.3.1}$$

The integral in the denominator usually has to be evaluated numerically. We note that the increase in accuracy comes to the cost of numerically evaluating this integral, bearing in mind the original cost of evaluating the SPA.

3.4 Example: Noncentral Chi-Squared

An interesting application of the SPA, proposed in Goutis and Casella (1999), is the *non-central chi-squared* distribution, related to the CIR process (2.3.3). The probability density function is:

$$f_X(x;k,\lambda) = \sum_{i=0}^{\infty} \frac{x^{k/2+i-1}e^{-x/2}}{\Gamma(k/2+i)2^{k/2+i}} \frac{\left(\frac{\lambda}{2}\right)^i e^{-\left(\frac{\lambda}{2}\right)}}{i!}, \tag{3.4.1}$$

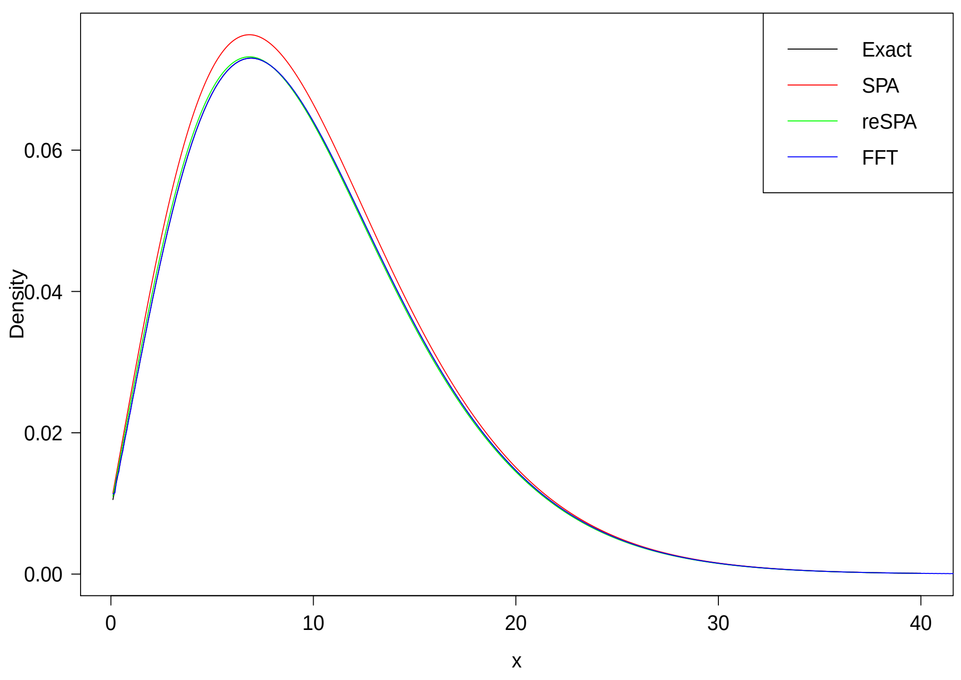

FIGURE 3.1: Exact and estimated probability density function for the non-central chi-square distribution for parameters $k = 2$ and $\lambda = 8$.

an infinite sum of central chi-squared densities weighted with Poisson probabilities. Another way to write this density, which we will exploit later, is:

$$f_X(x; k, \lambda) = \frac{1}{2} e^{-\frac{x+\lambda}{2}} \left(\frac{x}{\lambda}\right)^{\frac{k}{4}-\frac{1}{2}} I_{\frac{k}{2}-1}\left(\sqrt{\lambda x}\right), \tag{3.4.2}$$

where $I_\nu(x)$ is the modified Bessel function of the first kind. If one does not want to deal with infinite sums, one could exploit that the MGF if quite simple:

$$M_X(t) = \frac{e^{\frac{\lambda t}{1-2t}}}{(1-2t)^{\frac{k}{2}}}. \tag{3.4.3}$$

We solve the saddlepoint equation to obtain

$$\hat{\tau} = \frac{-k + 2x - \sqrt{k^2 + 4\lambda x}}{4x}, \tag{3.4.4}$$

which we insert into our expression for the SPA (3.5). The exact density[1] , the SPA, the renormalized SPA, and the density using the FFT of the characteristic function are plotted in figure 3.1. We here see that the SPA lies closely to that of the exact density, and this is even more so the case for the renormalized version. It is very hard to distinguish the renormalized SPA, the density obtained via the FFT, and the exact density using R from one another by just using the naked eye.

3.5 Example: Compounded Poisson Process

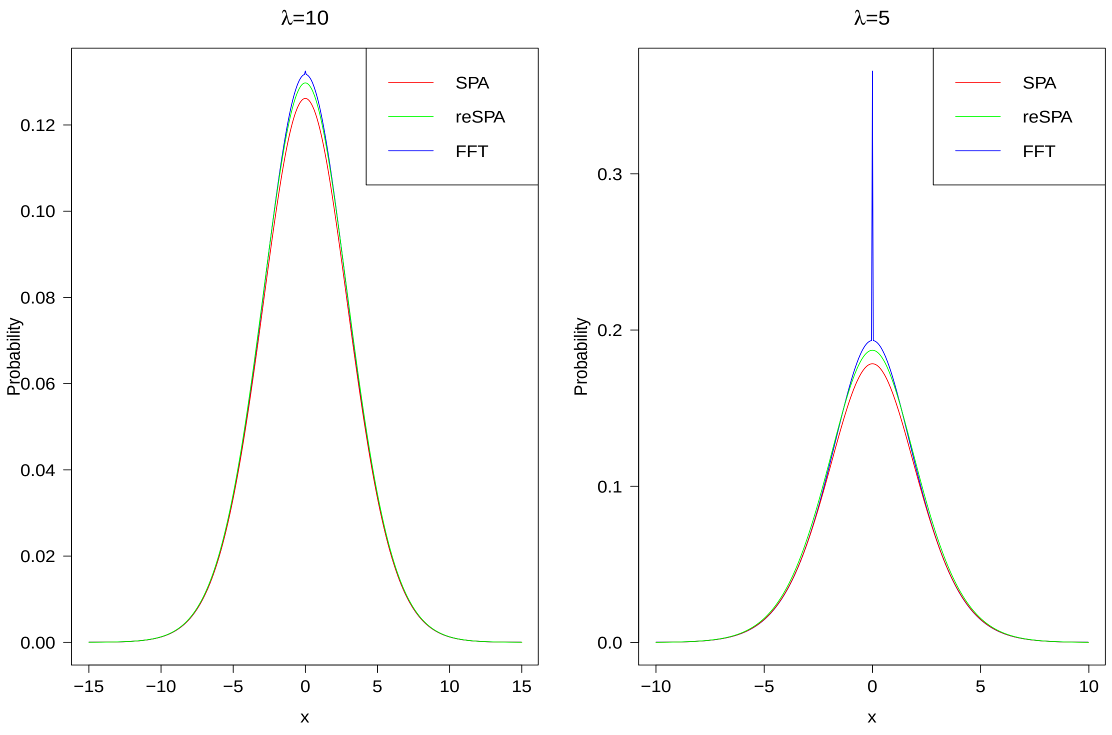

Figure 3.2: Exact (FFT) and approximated (SPA and reSPA) transition probabilities for the compounded Poisson process with different values of λ: $\lambda = 10$ and $\lambda = 5$. The jumps were standard normal and the time interval was set to $t = 1$.

For the previous example, the SPA provides a reasonable approximation of the density, but for different processes this might not be the case. One such instance is the compounded

[1] The density plot labelled as the exact density is obtained via the dchisq(x, k, λ) function in R. The R function calculates the density as a Poisson mixture of central chi-squares (approximation of equation (3.4.1)), and hence it is not exact (R Core Team, 2015). However, for our purpose with a small non-centrality parameter, this gives an accurate result which we use as our benchmark and label as exact.

Poisson process, defined by equation (2.2.2). The CGF can be derived from the MGF (2.3.29) and reads

$$K_X(s) = \lambda t \left(M_Y(s) - 1\right), \tag{3.5.1}$$

where Y is the distribution of the iid jumps. Taking these to be normally distributed with expectation μ and variance ν^2, we arrive at:

$$K_X(s) = \lambda t \left(e^{s\mu + \frac{1}{2}\nu^2 s^2} - 1\right). \tag{3.5.2}$$

The inner problem can then be solved using a Newton-type algorithm. The SPA and the renormalized SPA are plotted in figure 3.2, together with the estimated density using the FFT algorithm with the characteristic function. We assume that the latter is reasonably exact.

In figure 3.2 we find that for $\lambda = 10$, both the SPA and the renormalized SPA are quite accurate, similar to that of the example with the non-central chi-square distribution. As λ decreases, they both become more inaccurate at the centre of the density. An interesting observation concerns the high accuracy of the SPA in the tails. In risk management, different risk measures such as *value at risk* and *expected shortfall* (see e.g. McNeil et al. (2005, chapter 2.2)) concern themselves with the behaviour in the tail. Since the compounded Poisson process is a popular model especially in insurance for the cumulative amount of claims, but also for other applications such as credit risk (Gerhold et al., 2010), it is conceivable that the SPA might have important applications in this respect.

Chapter 4

Approximation Methods for small-time Jump-Diffusion Transition Densities

In this chapter we follow Preston and Wood (2012) closely. The first section considers discretization schemes that are retained terms from the Itô-Taylor expansions. We present three such schemes, the Euler scheme, the Milstein scheme, and a third scheme that under suitable conditions may attain strong order of convergence 1.5. The second section considers approximation methods for small-time jump-diffusion processes. Here, three methods are proposed.

4.1 Discretization Schemes

We will now construct numerical integration schemes based upon the Itô-Taylor approximations developed in section 2.4. The first obstacle is the calculation of the first Itô integrals (2.4). We here state the results. The proof of the ones involving Brownian motions can be found in appendix A. For the first integrals, with α having two or fewer indices, we have

$$\begin{aligned}
I_0 &= \int_0^t ds = t, \\
I_1 &= \int_0^t dW_s = W_t = J_1, \\
I_{0,0} &= \int_0^t \int_0^{s_1} ds_2 ds_1 = \frac{1}{2}t^2, \\
I_{1,0} &= \int_0^t \int_0^{s_1} dW_{s_2} ds_1 = \int_0^t W_{s_1} ds_1 = J_2, \\
I_{0,1} &= \int_0^t \int_0^{s_1} ds_2 dW_{s_1} = \int_0^t s_1 dW_{s_1} = tJ_1 - J_2, \\
I_{1,1} &= \int_0^t \int_0^{s_1} dW_{s_2} dW_{s_1} = \frac{1}{2}\left(J_1^2 - t\right), \qquad (4.1.1)
\end{aligned}$$

where the J_i's are defined as $J_1 = W_t$ and $J_2 = \int_0^t W_s ds$. We note that the vector $(J_1, J_2)^T$ has mean and covariance matrix

$$\mathrm{E}\begin{pmatrix} J_1 \\ J_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mathrm{Var}\begin{pmatrix} J_1 \\ J_2 \end{pmatrix} = \begin{pmatrix} t & \frac{1}{2}t^2 \\ \frac{1}{2}t^2 & \frac{1}{3}t^3, \end{pmatrix} \qquad (4.1.2)$$

respectively. In addition, J_1 and J_2 are Gaussian (Preston and Wood, 2012).

Now, while discussing and comparing schemes, we need some sort of measure to discuss their accuracy. In the deterministic case we usually compare the obtained approximation with the exact solution. In the SDE case, there are two ways of measuring accuracy, strong order and weak order convergence. We note that both the convergence criteria reduce to the normal convergence criterion in the deterministic case, if the diffusion coefficient is zero. In this thesis we shall use the strong order convergence to measure accuracy for our Itô-Taylor schemes.

Definition 4.1.1 (strong order convergence). We say that a time discrete approximation X^δ converges strongly with order $\gamma > 0$ at time t if there exists a positive constant C, not depending on δ, and a finite $\delta_0 > 0$ so that

$$\epsilon(\delta) = E\left(\left|X_t - X^\delta(t)\right|\right) \leq C\delta^\gamma. \qquad (4.1.3)$$

(Kloeden and Platen, 1992)

There exist conditions under which an Itô-Taylor expansion attains a given order of strong convergence. These can be found in Kloeden and Platen (1992, chapter 5). For the multiple Itô integrals 2.4 I_α, let $l(\alpha)$ denote the number of components in α and let $n(\alpha)$ denote the number of zero components. In the case of an one-dimensional SDE, all non-zero components are one. Then a scheme attains strong convergence of order γ if it includes all terms with α satisfying $l(\alpha) + n(\alpha) \leq 2\gamma$ (Preston and Wood, 2012).

We will present three different schemes, the Euler-Maruyama scheme of strong order 0.5, the Milstein scheme of strong order 1.0, and a third scheme that can attain strong order 1.5. These schemes are presented in Preston and Wood (2012). In addition to these three schemes, there is a fourth one of strong order 2.0 presented here. But this scheme involves a transformation to obtain unit diffusion. This is problematic for the intended extension to a more general jump-diffusion process and is therefore not considered here.

Following Preston and Wood (2012), we will consider time-homogeneous processes so that the drift and diffusion coefficients will only depend upon the state X_t of the process at time t. We use μ and σ to denote the drift and diffusion processes evaluated at the left point of the time interval, and primes to indicate derivatives. As an example:

$$\mu = \mu\left(X_t\right)\big|_{X_{t_0}}, \text{ and } \sigma'' = \frac{\partial^2}{\partial x^2}\sigma\left(X_t\right)\Big|_{X_{t_0}}. \tag{4.1.4}$$

4.1.1 Scheme 1: The Euler-Maruyama Scheme

The *Euler-Maruyama Scheme* is the easiest and most used discretization based on the Itô-Taylor expansions (2.4). It attains strong order of convergence 0.5 in general, but if the diffusion coefficient is constant, it attains strong order 1.0 (Kloeden and Platen, 1992). It provides good numerical results in the cases of simple processes with nearly constant drift and diffusion coefficients. However, for processes with nonlinear coefficients, higher order schemes might be preferred. In the one-dimensional case, it is of the form

$$X_t = X_{t_0} + \mu I_0 + \sigma I_1, \tag{4.1.5}$$

which is Gaussian due to the I_1-term. It has expectation $X_{t_0} + \mu t$, variance $\sigma^2 t$, and MGF

$$M_{X_t} = \exp\left\{ s\left(X_{t_0} + \mu t\right) + \frac{s^2\sigma^2 t}{2} \right\}. \tag{4.1.6}$$

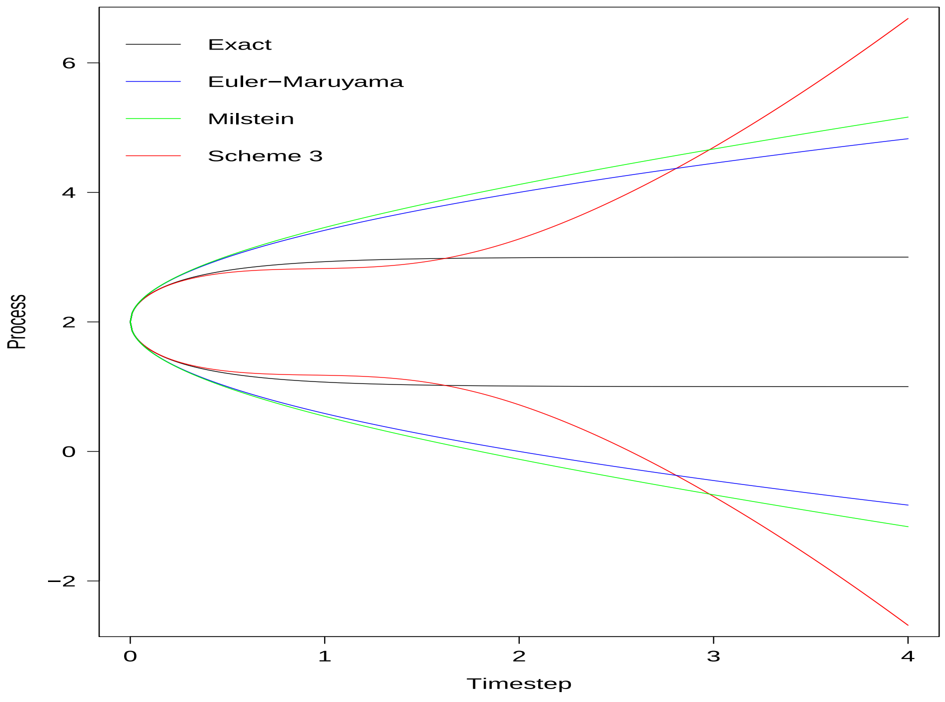

FIGURE 4.1: Exact and approximate prediction bands ($\mathrm{E}[r_t|r_0] \pm SD[r_t|r_0]$) for the CIR process with starting value $r_0 = 2$ and parameters $\kappa = 1$, $\alpha = 2$, and $\sigma = 1$.

4.1.2 Scheme 2: The Milstein Scheme

The *Milstein Scheme* is slightly more complicated than the Euler-Maruyama scheme and might be regarded as the next step, since it contains one more term from the Itô-Taylor expansions. It attains strong order of convergence 1.0 in all cases, not depending on the drift or diffusion coefficients as the Euler-Maruyama scheme. It reads

$$X_t = X_{t_0} + \mu I_0 + \sigma I_1 + \sigma\sigma' I_{1,1}. \tag{4.1.7}$$

We see that when the diffusion coefficient is constant, the Milstein scheme reduces to the Euler-Maruyama scheme (4.1.5).

Theorem 4.1.2. The MGF of the Milstein scheme (4.1.7) is given by the following equation

$$M_{X_t}(s) = \frac{\exp\left\{\frac{c_1^2 s^2 t}{2-4sc_2 t}\right\}}{\sqrt{1-2tc_2 s}} \exp\left\{s\left(X_{t_0}+c_3\right)\right\}, \tag{4.1.8}$$

where

$$c_1 = \sigma,\ c_2 = \frac{1}{2}\sigma\sigma',\ c_3 = \left(\mu - \frac{1}{2}\sigma\sigma'\right)t. \tag{4.1.9}$$

Proof. To find the MGF, we rearrange the scheme to

$$X_t - X_{t_0} - c_3 = c_1 J_1 + c_2 J_1^2, \tag{4.1.10}$$

where the constants c_1, c_2, and c_3 are as in (4.1.9). We now seek the MGF of the right side of equation (4.1.10). Let $Z \sim N(0,t)$, then the MGF of $c_1 Z + c_2 Z^2$ is defined as

$$\begin{aligned}
M_{c_1 Z + c_2 Z^2}(s) &= \mathrm{E}\left[e^{(c_1 Z + c_2 Z^2)s}\right] \\
&= \int_{-\infty}^{\infty} e^{\left(c_1 z + c_2 z^2\right)s} \frac{1}{\sqrt{2\pi t}} e^{-\frac{z^2}{2t}} dz \\
&= \frac{1}{\sqrt{2\pi t}} \int_{-\infty}^{\infty} \exp\left\{-\left(\frac{1}{2t} - c_2 s\right) z^2 + c_1 s z\right\} dz \\
&= \frac{1}{\sqrt{2\pi t}} \int_{-\infty}^{\infty} \exp\left\{-\left(\frac{1}{2t} - c_2 s\right)(z-q)^2 + \frac{c_1^2 s^2 t}{2(1-2sc_2 t)}\right\} dz \\
&= \exp\left\{\frac{c_1^2 s^2 t}{2-4sc_2 t}\right\} \frac{1}{\sqrt{2\pi t}} \int_{-\infty}^{\infty} \exp\left\{-\frac{(z-q)^2}{2t(1-2tc_2 s)^{-1}}\right\} dz \\
&= \frac{\exp\left\{\frac{c_1^2 s^2 t}{2-4sc_2 t}\right\}}{\sqrt{1-2tc_2 s}} \frac{1}{\sqrt{2\pi t}} \frac{1}{(1-2tc_2 s)^{-\frac{1}{2}}} \int_{-\infty}^{\infty} \exp\left\{-\frac{(z-q)^2}{2t(1-2tc_2 s)^{-1}}\right\} dz \\
&= \frac{\exp\left\{\frac{c_1^2 s^2 t}{2-4sc_2 t}\right\}}{\sqrt{1-2tc_2 s}},
\end{aligned} \tag{4.1.11}$$

where $q = c_1 st/(1-2sc_2 t)$. The desired MGF is then found by multiplying the MGF for the right hand side and the MGF of $X_{t_0} + c_3$. □

4.1.3 Scheme 3: The Itô-Taylor Scheme of Strong Order 1.5

A scheme that can attain strong order 1.5, found in Preston and Wood (2012) reads

$$\begin{aligned} X_t &= X_{t_0} + \mu I_0 + \sigma I_1 + \sigma\sigma' I_{1,1} \\ &\quad + \left(\mu\mu' + \frac{1}{2}\sigma^2\mu''\right) I_{0,0} + \left(\mu\sigma' + \frac{1}{2}\sigma^2\sigma''\right) I_{0,1} + \sigma\mu' I_{1,0}, \end{aligned} \tag{4.1.12}$$

or, in a different form:

$$X_t - X_{t_0} - c_4 = c_1 J_1 + c_2 J_1^2 + c_3 J_2, \tag{4.1.13}$$

where

$$\begin{aligned} c_1 &= \sigma + \left(\mu\sigma' + \frac{1}{2}\sigma^2\sigma''\right) t, \\ c_2 &= \frac{1}{2}\sigma\sigma', \\ c_3 &= \sigma\mu' - \mu\sigma' - \frac{1}{2}\sigma^2\sigma'', \\ c_4 &= \left(\mu - \frac{1}{2}\sigma\sigma'\right) t + \left(\mu\mu' + \frac{1}{2}\sigma^2\mu''\right)\frac{1}{2}t^2. \end{aligned} \tag{4.1.14}$$

(Preston and Wood, 2012)

Preston and Wood (2012) then finds the MGF to be

$$M_{X_t}(s) = \frac{\exp\left\{\frac{\left(6c_1^2+6c_1c_3t+2c_3^2t^2-c_3^2t^3sc_2\right)ts^2}{12-24sc_2t}\right\}}{\sqrt{1-2tc_2s}} \exp\left\{s\left(X_{t_0}+c_4\right)\right\}. \tag{4.1.15}$$

We note that when the diffusion coefficient $\sigma(t, X_t)$ is constant, this scheme is Gaussian, having mean $\mu t + \left(\mu\mu' + \frac{1}{2}\sigma^2\mu''\right)\frac{1}{2}t^2$ and variance $\left(\sigma^2 + \sigma^2\mu' t + \frac{1}{3}(\sigma\mu' t)^2\right)t$. It will then have strong order of convergence 1.5, but in the case of a nonconstant diffusion term, it has strong order of convergence 1.0.

4.2 Approximation Methods for small-time Jump-Diffusion Transition Densities

The MGFs of the discretization schemes in the previous section were used in Preston and Wood (2012) to estimate the transition densities for time-homogeneous diffusion processes by calculating the SPA of the estimation to the diffusion process. In the following theorem we extend this method to a time-homogeneous jump-diffusion process. A note on notation: we will refer to the transition density of X_t, which is the probability density function of the random variable $X_t|X_{t_0}$. We will however omit the conditioning in further notation.

Theorem 4.1. *Let X_t be a jump-diffusion process, where the diffusion part and the jumps are independent of one another. Further, let $\widetilde{X}_{t^-}$ denote the approximate solution to the pure diffusion, based on a discretization scheme in section (4.1). Then an approximation to the jump-diffusion is:*

$$X_t \approx \widetilde{X}_t = \widetilde{X}_{t^-} + \sum_{i=1}^{N_t} Z_i. \tag{4.2.1}$$

An approximation to the transition density of X_t then follows from the saddlepoint approximation (3.5) to the transition density of the approximation in 4.2.1, which we shall call the Itô-Taylor saddlepoint approximation (ITSPA):

$$f_{X_t}(x) \approx spa\left(f_{\widetilde{X}_t}; x\right), \tag{4.2.2}$$

where

$$K_{\widetilde{X}_t}(\hat{s}) = K_{\widetilde{X}_{t^-}}(\hat{s}) + K_{Y_t}(\hat{s}) = K_{\widetilde{X}_{t^-}}(\hat{s}) + \lambda t\left(M_Z - 1\right), \tag{4.2.3}$$

and M_Z is the MGF of the iid jump magnitudes.

Proof. Follows from the independence between the diffusion part and the jump part of the SDE. □

In chapter 3 we calculated the SPA for a compounded Poisson process. While this works well by high jump intensity, it performs poorly by estimating the probability of states that are possible without jumps and by low intensity. For small-time transition densities it is natural to assume a fairly small jump intensity, and for these reasons the following method is preferable.

Theorem 4.2. *Let X_t, $\widetilde{X}_t$, $\widetilde{X}_{t^-}$ and Y_t be as in theorem 4.1. The Itô-Taylor saddlepoint approximation mixture (mITSPA) to the transition density of X_t is as follows:*

$$mspa\left(f_{\widetilde{X}_t}; x\right) = spa\left(f_{\widetilde{X}_{t^-}}; x\right) e^{-\lambda t} + spa\left(f_{\widetilde{X}_t^*}; x\right)\left(1 - e^{-\lambda t}\right), \tag{4.2.4}$$

where

$$\widetilde{X}_t^* = \widetilde{X}_{t^-} + \sum_{i=1}^{N_t^*} Z_i, \tag{4.2.5}$$

and N_t^* has a zero-truncated Poisson distribution with intensity λt and defined by $N_t^* = N_t | N_t > 0$. The related CGF of the compounded zero-truncated Poisson process Y^* is given by:

$$K_{Y^*}(s) = \lambda t \left(M_Z(s) - 1\right) + \log\left(1 - e^{-\lambda t M_Z(s)}\right) - \log\left(1 - e^{-\lambda t}\right). \tag{4.2.6}$$

Proof.

$$\begin{aligned} f_{X_t}(x) &= \sum_{i=0}^{\infty} f_{X_t|N_t=i}(x) P(N_t = i) \\ &= f_{X_t|N_t=0}(x) P(N_t = 0) + f_{X_t|N_t>0}(x) P(N_t > 0) \\ &\approx \text{spa}\left(f_{\widetilde{X}_{t^-}}; x\right) e^{-\lambda t} + \text{spa}\left(f_{\widetilde{X}_t^*}; x\right)\left(1 - e^{-\lambda t}\right). \end{aligned} \tag{4.2.7}$$

Now, the only thing left is the derivation of the CGF of Y_t^*. From the definition of the MGF we have:

$$\begin{aligned} M_{Y_t^*} &= \mathrm{E}\left[e^{sY_t^*}\right] = \mathrm{E}\left[\exp\left\{s\sum_{i=1}^{N_t^*} Z_i\right\}\right] = \mathrm{E}\left[\mathrm{E}\left[e^{snZ}|N_t^* = n > 0\right]\right] \\ &= \sum_{k=1}^{\infty} \frac{(\lambda t)^k e^{-\lambda t}}{k!\,(1-e^{\lambda t})} \mathrm{E}\left[e^{sZ}\right]^k = \sum_{k=0}^{\infty} \frac{(\lambda t)^k e^{-\lambda t}}{k!\,(1-e^{\lambda t})} M_Z(s)^k - \frac{e^{-\lambda t}}{1-e^{-\lambda t}} \\ &= \frac{e^{\lambda t(M_Z(s)-1)} - e^{-\lambda t}}{1-e^{-\lambda t}}. \end{aligned} \tag{4.2.8}$$

The CGF then follows as the logarithm of the MGF, and by using the identity $\log(a-b) = \log(a) + \log(1-b/a)$. □

In the case of a pure diffusion process, the mITSPA reduces to the ITSPA.

Our third and final method stems from Zhang and Schmidt (2016). It is simply the approximated IFT of the characteristic function corresponding to one of the discretization schemes using the Gauss-Laguerre method.

The Gauss-Laguerre method of order n approximates exponentially weighted integrals in the following way (Press et al., 1992):

$$\int_0^{\infty} e^{-x} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i), \tag{4.2.9}$$

where x_i is the i'th root of the Laguerre polynomial of order n defined recursively by

$$(i+1)L_{i+1} = (1-x+2i)L_i - iL_{i-1},\ L_0 = 1,\ L_1 = 1-x, \tag{4.2.10}$$

and w_i are the weights given by

$$w_i = \frac{x_i}{(n+1)^2\left[L_{n+1}(x_i)\right]^2}. \tag{4.2.11}$$

Theorem 4.3. *Let X_t, $\widetilde{X}_t$, $\widetilde{X}_{t-}$, and Y_t be as in theorem 4.1. The Fourier-Gauss-Laguerre (FGL) approximation to the transition density of X_t is given by:*

$$fgl\left(f_{\widetilde{X}_t}; x\right) = \frac{1}{\pi}\sum_{j=1}^{n} w_j \Re\left(\phi_{\widetilde{X}_t}(s_j)e^{s_j - is_jx}\right), \tag{4.2.12}$$

where w_j and s_j are the weights and the abscissa respectively in the Gauss-Laguerre method of order n. The characteristic function is found by multiplying the characteristic function for one of the discretizations $\widetilde{X}_t^-$ of the diffusion part and the characteristic function for the compounded Poisson process Y_t:

$$\phi_{\widetilde{X}_t}(s) = \phi_{\widetilde{X}_{t-}}(s)\phi_{Y_t}(s). \tag{4.2.13}$$

Proof. From theorem 3.4, we have

$$\begin{aligned} f_{X_t}(x) &= \frac{1}{\pi}\int_0^\infty \Re\left(\phi_{X_t}(s)e^{-isx}\right) ds = \frac{1}{\pi}\int_0^\infty e^{-s}\Re\left(\phi_{X_t}(s)e^{s-isx}\right) ds \\ &\approx \frac{1}{\pi}\int_0^\infty e^{-s}\Re\left(\phi_{\widetilde{X}_t}(s)e^{s-isx}\right) ds \approx \frac{1}{\pi}\sum_{i=j}^{n} w_j\Re\left(\phi_{\widetilde{X}_t}(s_j)e^{s_j-is_jx}\right). \end{aligned} \tag{4.2.14}$$

□

The original method of Zhang and Schmidt (2016) suggests approximating the original IFT (3.1) of $\phi_{\widetilde{X}_t}$ by using the trapezoid method and the FFT algorithm for efficiently obtaining the transition density. While this might be efficient for estimating the transition density, it was shown to be computationally costly and to not produce accurate results when doing likelihood-based inference for the real data in chapter 7. When constructing the likelihood, the quantities of interest are the transition probabilities, not the whole densities. We therefore suggest the FGL over directly approximating the IFT with a DFT using the FFT, which proved to be computationally costly to a lesser degree and which provided us with accurate results. The implementation of these methods with automatic differentiation is discussed in chapter 5 and numerical results in chapter 6.

Chapter 5

TMB and Automatic Differentiation

In this section we wish to explain *automatic differentiation* (AD), also called *algorithmic differentiation*, and to motivate its use through packages such as *Template Model Builder* (**TMB**). We start with outlining in what situations it might be applicable and how it diverges from standard tools such as divided differences and symbolic differentiation. We then look into the theory of AD and its implementation in software, considering dual numbers, forward mode, and reverse mode. We then proceed to discuss the TMB package in R, utilizing **CppAD**, which is an AD package for C++.

5.1 Motivation for Automatic Differentiation

In the natural sciences many problems revert to the problem of calculating derivatives. Popular and easily implemented methods such as the Euler and Newton methods require first order derivatives, while optimization problems typically require the Hessian matrix in addition to the gradient. For higher order approximation, higher-order derivatives are preferable.

In the context of our problem of parameter estimation, the second alternative, namely optimization problems usually requiring both the gradient and the Hessian matrix of the objective function, is relevant. The usual solutions to these problems are of three kinds:

1. Numerical differentiation

   Divided differences is very easy to implement, using the already implemented objective function f. However, the method has the disadvantage of $O(n)$ evaluations of $f \in \mathbb{R}^n$. It also has the problem of truncation error versus round-off error, and of becoming increasingly inaccurate for higher-order derivatives.

2. Symbolic differentiation

   Symbolic differentiation is indeed completely mechanical in the sense that it takes advantage of repeated use of the chain rule. A disadvantage is that it requires relatively large amounts of memory. It also requires implementation of the obtained expression.

3. Differentiation by hand

   This method has the benefit of producing efficient derivative code, but for many applications where there is a need for computing derivatives of order n when n is large, or when the objective function is tedious, it is not feasible (Radul, 2013).

In recent years an intriguing alternative has gained increasing popularity: automatic differentiation (AD). Given a computer algorithm defining a function, AD is a set of techniques used to evaluate numerically the derivatives of that function (Kristensen et al., 2015). In fact, derivatives of any order of the function can be calculated exactly.

5.2 A Brief Introduction to Automatic Differentiation

AD theory is based on the property of programming languages such as C++ of decomposing expressions into elementary operations. Elementary differentiation rules can then be applied to the elementary operations and evaluated, and the derivatives are bound together using the chain rule. This is a strategy similar to that of symbolic differentiation, but it differs in the sense that AD generates evaluations and not formulas. As soon as an intermediate expression can be evaluated, it is evaluated (Tucker, 2010). We illustrate AD by an example:

Step	Operation	Value	Derivatives
1	$t_1 = sx$	sx	$\frac{\partial t_1}{\partial s} = x$
2	$t_2 = \frac{s^2\sigma^2}{2} - t_1$	$\frac{s^2\sigma^2}{2} - sx$	$\frac{\partial t_2}{\partial s} = \sigma^2 s - \frac{\partial t_1}{\partial s}$
3	$t_3 = s\mu + t_2$	$s\mu + \frac{s^2\sigma^2}{2} - sx$	$\frac{\partial t_3}{\partial s} = \mu + \frac{\partial t_2}{\partial s}$
4	$g = t_3$	$s\mu + \frac{s^2\sigma^2}{2} - sx$	$\frac{\partial g}{\partial t_3} = 1$

TABLE 5.1: Algorithm broken down into elementary operations

Consider the inner problem (3.2.2)

$$\arg\min_s \{K_X(s) - sx\}. \tag{5.2.1}$$

Taking X to be normal $X \sim N(\mu, \sigma^2)$, we have the solution $\hat{s} = \frac{x-\mu}{\sigma^2}$ from solving the equation $K'_X(s) - x = 0$. However, the inner problem often has to be solved numerically when the CGF is more complicated, and for such instances AD comes in handy. We can implement the objective function $g = K_X(s) - sx$ as a computer algorithm, and, utilizing AD theory, we can construct a robust optimization algorithm minimizing the objective function with respect to s. An illustration of how the implemented objective function might be broken down is to be found in table 5.1. The derivatives can then be combined:

$$g'(s) = \frac{\partial g}{\partial t_3}\frac{\partial t_3}{\partial s} = \mu + \frac{\partial t_2}{\partial s} = \mu + \sigma^2 s - \frac{\partial t_1}{\partial s} = \mu + \sigma^2 s - x. \tag{5.2.2}$$

Another illustrative example, considering linear regression, is given in Fournier et al. (2012).

To discuss implementation of AD, we first consider the case of first order AD for functions $f : \mathbb{R} \to \mathbb{R}$. We extend all real numbers with a second component: $f(x_0) \to (f(x_0), f'(x_0))$. The idea is, then, that the first component holds the value of the function, and the second component holds the derivative at x_0. This idea of *dual numbers* can be implemented with an AD data type. Arithmetic for this data type $\vec{u} = (u, u')$ might look like:

- $\vec{u} + \vec{v} = (u + v, u' + v')$,
- $\vec{u} - \vec{v} = (u - v, u' - v')$,
- $\vec{u} \times \vec{v} = (uv, uv' + u'v)$,

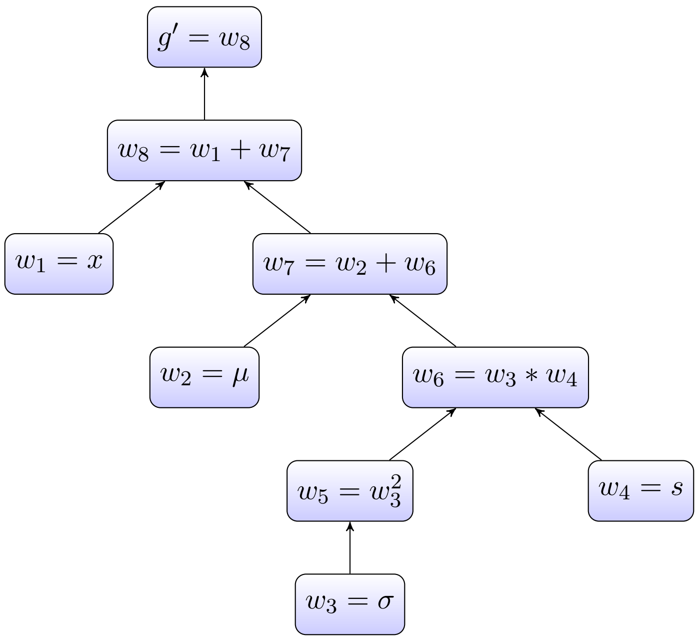

FIGURE 5.1: Computational graph of the derivative of the inner problem $g'(s)$ for a normal random variable.

- $\frac{\vec{u}}{\vec{v}} = \left(\frac{u}{v}, \frac{u'v - uv'}{v^2}\right)$.

One then only has to define how the data type should treat constants and the independent variable x:

$$\hat{x} = (x, 1) \text{ and } \hat{c} = (c, 0),$$

Using the usual rules of differentiation, one can then extend the data type to functions using the chain rule:

$$\hat{f}(\hat{u}) \to \hat{f}(u, u') = (f(u), u'f'(u)).$$

If the user declares a variable of this specific type and then proceeds to initialize it with a rather lengthy expression, we have already seen how a programming language breaks up the expression into elementary operations (table 5.1). Having the AD arithmetic defined in the language will then ensure that we end up with the correct values for the user-defined variable and its derivative.

This idea might of course be extended to a data type which, in addition to the type already discussed, has a third component holding the value of the second derivative $f''(x_0)$. However, a more effective approach to higher-order AD can be obtained through the calculus of Taylor series (Griewank and Walther, 2008). The arithmetic of the AD data type can be integrated using *operator overloading*. While this is not possible in all programming languages, it is possible in C++, and is used by the AD package **CppAD** (Bell, 2005).

Our final note on the principles of AD considers how to recurse through the chain rule. In our example with the inner problem, we have shown how the derivatives might be combined in order to find the derivative of the inner problem. The way we recursed through the derivatives is known as *forward mode*. But this is just one way to combine derivatives. In practice, forward mode is combined with *reverse mode* for calculation of higher-order derivatives. Let g' denote $\frac{\partial g}{\partial s}$, and consider the computational graph of g' in figure 5.1. Reverse mode will then pass through the computational graph, starting at the node representing g', until it reaches the node representing s in the following way:

$$\begin{aligned}\frac{\partial g'}{\partial s} &= \frac{\partial w_8}{\partial s}\left(\frac{\partial g'}{\partial w_8}\right) = \frac{\partial w_7}{\partial s}\left(\frac{\partial w_8}{\partial w_7}\frac{\partial g'}{\partial w_8}\right) \\ &= \frac{\partial w_6}{\partial s}\left(\frac{\partial w_7}{\partial w_6}\frac{\partial w_8}{\partial w_7}\frac{\partial g'}{\partial w_8}\right) = \frac{\partial w_4}{\partial s}\left(\frac{\partial w_6}{\partial w_4}\frac{\partial w_7}{\partial w_6}\frac{\partial w_8}{\partial w_7}\frac{\partial g'}{\partial w_8}\right) \\ &= 1\left(\sigma^2 \times 1 \times 1 \times 1\right) = \sigma^2. \end{aligned} \tag{5.2.3}$$

For a thorough introduction of the techniques and principles of AD, we refer to Griewank and Walther (2008).

5.3 TMB and CppAD

When implementing and using the ITSPA (6), we have utilized the R package *Template Model Builder* (**TMB**) recently released on CRAN. **TMB** enables easy and efficient access to parallel computations with R and C++. The user will define an objective function in C++, for example the joint likelihood, and then everything else is done in R (data management, Laplace approximation and optimization). The Laplace approximation is performed by the use of **CHOLMOD**, available in R through the **Matrix** package. It also lets the user take

Code	Description	AD-mode	Starting point	result
1	T1: Code for $f(u, \theta)$		θ	f
2	T2: AD of Code 1	Forward	θ	$\nabla_\theta f$
3	T3: AD of Code 2	Reverse	$\nabla_\theta f$	$\mathbf{H}$

TABLE 5.2: **CppAD** steps to calculate tapes T1-T3 (Kristensen et al., 2015; Skaug and Fournier, 2006).

advantage of the general purpose AD package **CppAD** (Bell, 2005) in C++, to calculate first, second, and possibly third order derivatives of the objective function (Kristensen et al., 2015).

The objective function is a function $f(u, \theta) : \mathbb{R}^{m+n} \to \mathbb{R}$, which is implemented by the user in C++. During the initial phase of the execution of the program, **CppAD** creates three tapes, T1-T3. A "tape" refers to a representation of some implemented function. Such a tape can be illustrated by a computational graph, similar to table 5.1 or figure 5.1. In **TMB**, the tape T1 refers to the representation of the user-implemented objective function, T2 refers to the gradient of the objective function, and T3 to the Hessian matrix. T2 is generated by a forward pass through T1, and T3 is generated through a reverse pass through T2 (Kristensen et al., 2015). This is illustrated in table 5.2, taken from Skaug and Fournier (2006). The computational graph of $g'(s)$ (figure 5.1) can be viewed as the tape T2 after a forward pass through the computational graph of $g(s)$.

5.3.1 Example: Cox-Ingersoll-Ross Maximum Likelihood Estimation

We will now show how **TMB** can be used for MLE [1]. From section 2.3.3, we know that future values of the CIR process, X_t, given a previous time point X_0, have the density

$$f_{X_t}(x) = ce^{-u-v}\left(\frac{v}{u}\right)^{\frac{q}{2}} I_q\left(2\sqrt{uv}\right),$$

where $c = 2\kappa/(\sigma^2(1-e^{-\kappa t}))$, $q = 2\kappa\alpha/\sigma^2 - 1$, $u = cX_0e^{-\kappa t}$, and $v = cX_t$. Consider observations of the CIR process at discrete time points $t_0, t_1, \ldots t_n$, and that we wish to estimate the

[1] The following example is modelled after the tutorial example on the **TMB** home page: `https://github.com/kaskr/adcomp/wiki/Tutorial`.

parameters of the process. We can then implement the joint negative log-likelihood (negative of equation (1.0.2)) as the objective function in C++. Using the **TMB** package, we pass the observed data points and some initial parameters from R to the objective function in C++. The tapes T1-T3 are created, and the function value l_θ, the gradient ∇l_θ, and the Hessian matrix $\mathbf{H}_{l_\theta}$ are returned from C++ and made available in R.

The C++ part of the **TMB** code is as follows: .

We first link the **TMB** library and then declare the objective function in the standard way of **TMB**:

```
#include <TMB.hpp>      // Links in the TMB libraries
template<class Type>
Type objective_function<Type>::operator() ()
```

where the data type `Type` is the AD special type in **TMB**, used for all variables except for `int` for integers. The data and parameters needed for the JNLL:

```
DATA_VECTOR(x);            // Sample path transmitted from R
DATA_SCALAR(dt);           // Equidistant timestep from R
PARAMETER_VECTOR(param);   // Parameters, initial guess from R
```

are read from R when the function is called. Note that the usual data types such as float and double are not used in **TMB** code.

Variables used in the JNLL of the CIR process are declared and initialized:

```
Type kappa=param[0], alpha=param[1], sigma=param[2];
Type c = 2*kappa/(sigma*sigma*(1-exp(-kappa*dt)));
Type q = 2*kappa*alpha/pow(sigma,2) - 1;
Type JNLL = 0;             // Declare and initialize the JNLL
```

including the JNLL itself. We then proceed to evaluate the JNLL, defined as the negative of the JLL (1.0.2), looping through the sample path:

```
for(int i = 0; i<(x.size()-1); i++){
        Type u=c*x[i]*exp(-kappa*dt);
        Type v=c*x[i+1];
        Type pCIR = c * exp(-u-v) * pow((v/u),(q/2)) *
```

```
                    besselI(Type(2*sqrt(u*v)),q);
          JNLL -= log(pCIR);
}
```

Finally we return the JNLL:

```
return JNLL;
```

.

The objective function is called in the R part of the code. We first generate data:

```
x<-cirProcess(50, 1000, 2, 1, .5, 1, 2)
```

The function `cirProcess(`$t, n, \kappa, \alpha, \sigma, x_{t_0}$`,seed)` simulates a CIR process exactly using the noncentral chi square random number generator in R. We then proceed to compile the C++ part of the code, stored in `cirJNLLexact.cpp`, and dynamically link it into R:

```
library(TMB)
compile("cirJNLLexact.cpp")
dyn.load(dynlib("cirJNLLexact"))
```

We construct an R object, `obj`, using the linked code:

```
obj<-MakeADFun(data=list(x=x,dt=50/1000),parameters=list(param=c(5,2,1)))
```

This object holds the initial parameters used when calling the code, the function value of the JNLL, the gradient, and the Hessian matrix:

```
obj$par
param  param  param
5      2      1
obj$fn(c(3,2,1))
[1] -338.4826
obj$gr(c(3,2,1))
          [,1]      [,2]      [,3]
[1,] 104.0886  360.1807  307.2538
obj$he(c(3,2,1))
          [,1]        [,2]       [,3]
```

```
[1,]    52.23106   266.0956  -310.4137
[2,]   266.09557   443.2943  -874.4130
[3,]  -310.41251  -874.4114  1181.7715
```

We then find the maximum of the JLL by minimizing the JNLL, using the *nlminb()* function in R:

```
opt<-nlminb(obj$par,obj$fn, obj$gr, obj$he)
outer mgc:  1069.415
⋮
outer mgc:  0.00540083
```

Finally, we can evaluate the standard error of the parameters:

```
rep<-sdreport(obj)
outer mgc:  0.00540083
⋮
outer mgc:  7.647844
rep
sdreport(.) result
      Estimate Std. Error
param 1.917462 0.29077243
param 1.089665 0.03958950
param 0.513891 0.01205538
Maximum gradient component: 0.00540083
```

One of the benefits of parallel computation with R and C++ is that R is based on C++, and functions in R are written in C++ located in the RMath library. Since R is running in the background, **TMB** can access the RMath library and extract functions from within. This is done with functions such as `ppois(`n, λ`)`. However, these functions only hold the function values, and therefore the first order derivatives have to be specified in terms of functions known to **TMB**. The `besselI(`x, ν`)` is not implemented in **TMB**. But it is needed for the density of the non-central chi-square and the CIR process. The code that makes the modified Bessel function of the first kind available to the **TMB** user can be found in appendix B.

5.4 Implementation of Approximation Methods

The implementation of approximated JNLLs and the exact ones is carried out in **TMB**. This approach to implementation benefits us with the speed of C++, the R-like syntax, easy access to probability density functions, and the possibility of creating very robust optimization code in R, having access to the exact values of the function, the gradient, and the Hessian matrix for all the JNLLs.

In the case of the pure diffusion processes, the implementation of the JNLL using the exact transition densities is similar to the example used to discuss MLE in **TMB** (5.3.1). We note that the R-like syntax and the implemented functions in **TMB**, such as `dnorm`$(x, \mu, \sigma, \log)$ and the log-normal density, here come into their own.

The ITSPA (4.1) is implemented in a similar manner as the exact likelihood (see section 5.3.1). The (non trivial) difference is inside the for-loop when building the JNLL, we use the ITSPA to the transition density. A nice feature of **TMB**, which we have made use of here, is the possibility to utilize AD inside the C++ part of the program through the functions `autodiff::gradient` and `autodiff::hessian`. We utilize this when calculating derivatives used in an iterative solver that finds the saddlepoint numerically. To illustrate this, we here provide the code: .

```
template<class Type, class Functor>
Type SP(Functor f, vector<Type> s, int niter) {
// Minimize the function, f, with respect to s
for (int i = 0; i<niter; i++){
        vector<Type> g = autodiff::gradient(f, s);
        matrix<Type> H = autodiff::hessian(f, s);
        s -= atomic::matinv(H) * g;
}
return s[0];
}
```

Figure 5.2 illustrates how the ITSPA is obtained in C++. Data and parameters are passed from R to the objective function in C++. It creates the CGF object which has the CGF functional value K_{X_t} as an operator with respect to s embedded in the structure. The inner

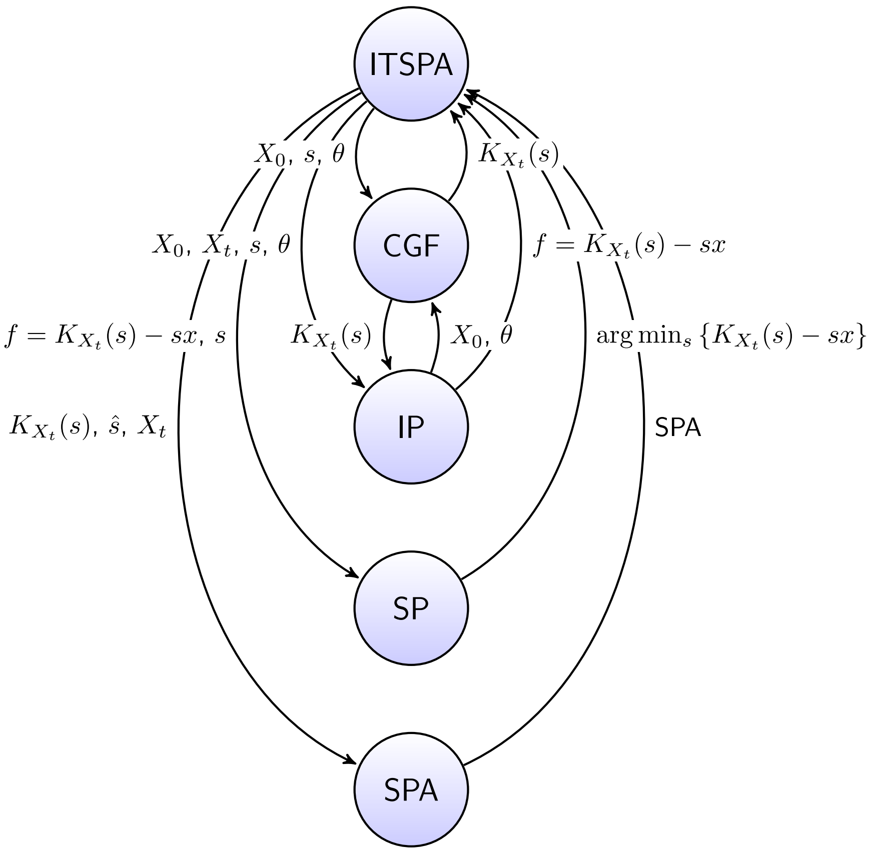

FIGURE 5.2: Illustration of the Itô-Taylor saddlepoint approximation implemented in C++. The CGF and inner problem (IP) are constructed, and the saddlepoint (SP) and the SPA are calculated on this basis.

problem object is created, which contains $K_{X_t}(s) - sx$ as an operator. The inner problem is then passed to the saddlepoint function which optimizes the inner problem with respect to s, and returns $\arg\min_s \{K_{X_t}(s) - sx\}$, using an algorithm utilizing AD. Having the CGF and the saddlepoint, we then pass these objects to the SPA function, which calculates the SPA, using AD to calculate the Hessian matrix of the CGF. The ITSPA is then returned from the objective function to R. The mITSPA (4.2) also utilizes this structure in obtaining the ITSPA.

In chapter 6 we shall find that a renormalization is necessary when working with the jump-diffusion models. In general, the ITSPA will not be easy to integrate exactly, but exact quadrature rules can in some cases be applied, as is the case for the CIR process. We have, however, estimated the integral via the *composite Simpson's rule*:

$$\int_a^b \hat{f}(x)dx \approx \frac{\Delta x_t}{3}\left[\hat{f}(X_0) + 2\sum_{j=1}^{\frac{n}{2}-1}\hat{f}(x_{2j}) + 4\sum_{j=1}^{\frac{n}{2}}\hat{f}(x_{2j-1}) + \hat{f}(x_n)\right],$$

where $x_j = a + jh$ for $j = 0, \ldots, n$ and Δx is the equidistant step length.

The nice feature of **TMB** of allowing AD inside the C++ code comes in handy when we find suitable starting and endpoints, (a, b), of the integral. We wish to integrate over the area where there is the most probability mass, which can be found through the possibility of calculating the moments via the derivatives of the CGF or the MGF. A natural estimate of an interval containing (most of) the probability mass could be $\mathrm{E}[X] \pm k\mathrm{SD}[X]$, where k is some constant.

The FGL (4.3) method requires the values of the characteristic function which is a complex valued function. As mentioned, **TMB** supports the AD data type `Type` and `int`, together with vectors and matrices based on these, which are all real valued. We therefore implement a third templated complex data type, `cType` based upon `Type`. Operators in C++ are overloaded to handle complex arithmetic, compound assignments, and unary operations. In addition, standard functions such as `abs` and `arg`, together with exponential functions, power functions, trigonometric functions, and hyperbolic functions were implemented. The full overview can be found in table 5.3. The abscissas and weights (4.2.11) for Gauss-Laguerre quadrature was calculated directly in C++, using the algorithm in Press et al. (1992, chapter 4, p. 152).

cType<Type>				
Type	Name	Arguments	Return value	Explanation
		Member functions		
`cType`	`(constructor)`	x, y	$x+iy$	Constructs a complex number
`cType`	`operator +=`	w	$z+w$	Compound assignments
`cType`	`operator -=`	w	$z-w$	
`cType`	`operator *=`	w	$z*w$	
`cType`	`operator /=`	w	z/w	
		Non-member functions		
`cType`	`operator +`	z	z	Unary operators
`cType`	`operator -`	z	$-z$	
`cType`	`operator +`	z, w	$z+w$	Complex arithmetic
`cType`	`operator -`	z, w	$z-w$	
`cType`	`operator *`	z, w	$z*w$	
`cType`	`operator /`	z, w	z/w	
`bool`	`operator ==`	z, w	`true` or `false`	Comparison operators
`bool`	`operator !=`	z, w	`true` or `false`	
`Type`	`abs`	z	$\|z\|$	Absolute value of z
`Type`	`arg`	z	θ	Phase angle of z in $(-\pi,\pi]$
`cType`	`conj`	z	$\overline{z}$	Complex conjugate
`Type`	`real`	z	$\Re(z)$	Real part of z
`Type`	`imag`	z	$\Im(z)$	Imaginary part of z
`cType`	`exp`	z	e^z	Complex exponential
`cType`	`log`	z	$\log z$	Complex logarithm
`cType`	`pow`	z, w	z^w	Power of a complex number
`cType`	`sqrt`	z	$\sqrt{z}$	Square root of a complex number
`cType`	`sin`	z	$\sin(z)$	Trigonometric functions
`cType`	`cos`	z	$\cos(z)$	
`cType`	`tan`	z	$\tan(z)$	
`cType`	`asin`	z	$\arcsin(z)$	
`cType`	`acos`	z	$\arccos(z)$	
`cType`	`atan`	z	$\arctan(z)$	
`cType`	`sinh`	z	$\sinh(z)$	Hyperbolic functions
`cType`	`cosh`	z	$\cosh(z)$	
`cType`	`tanh`	z	$\tanh(z)$	

Table 5.3: Complex AD data type: `cType<Type>`. Here x and y denote real numbers of type `Type`, while z, w, and i denote complex numbers of type `cType` and the imaginary unit, respectively.

Chapter 6

Numerical Results

In this chapter we present numerical results for the methods described in chapter 4. We test the accuracy of the methods by calculating and plotting transition densities, in addition to likelihood-based inference for processes with known solutions and which can be simulated exactly. The error in the estimated transition densities are measured using the absolute error of the log density. The processes that are considered are the GBM (2.3.1), the OU process (2.3.2), the CIR process (2.3.3), and the MJD model for log returns (2.3.4). The first section considers the transition densities, the second considers likelihood-based analysis.

6.1 Approximation of Transition Densities

In the following, we present some numerical results from estimating transition densities with the ITSPA (4.1), the mITSPA (4.2), and the FGL (4.3) methods. We consider the CIR process (2.3.3), also considered in Preston and Wood (2012), and the MJD model (2.3.4) for log-returns as our test processes for a pure diffusion and a jump-diffusion process respectively. For the CIR process, we already have the exact transition density, for the MJD model we obtain the transition density via the FGL method, which we take as our benchmark because the characteristic function is known in closed form. The error is measured using

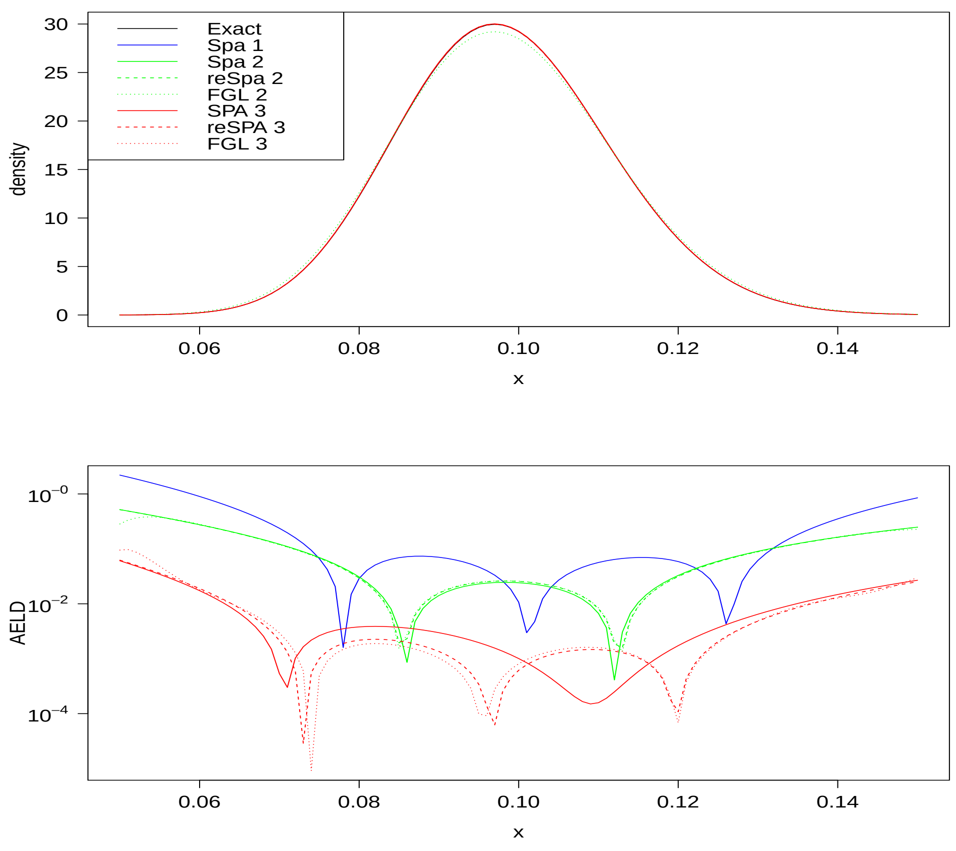

FIGURE 6.1: (Top) Exact and estimated transition densities for the CIR process with starting value $x_0 = 0.1$, parameters $\kappa = 0.5$, $\alpha = 0.06$, and $\sigma = 0.15$, and with timestep $t = 1/12$.
(Bottom) AELD plotted for the same values of the estimated transition densities.

the absolute error of the log density (AELD), defined as

$$\text{AELD}(x_t|x_0,\theta) = \left|\log \hat{f}_X t(x_t|x_0,\theta) - f_X t(x_t|x_0,\theta)\right|. \tag{6.1.1}$$

From figure 6.1 we see that all approximations are close to the exact transition density. For this particular example, it is evident that scheme 3 (4.1.3) outperforms the Euler-Maruyama scheme (4.1.5) and the Milstein scheme (4.1.7). In view of the AELD, the renormalized version of scheme 3 seems to provide the best approximation for most points. There are, however, points where the Milstein scheme and even the Euler-Maruyama scheme outperform the others. This stems from the fact that at some point $\hat{f} - f$ will change sign and make the error practically zero. For both figure 6.1 and figure 6.2, the FGL is practically

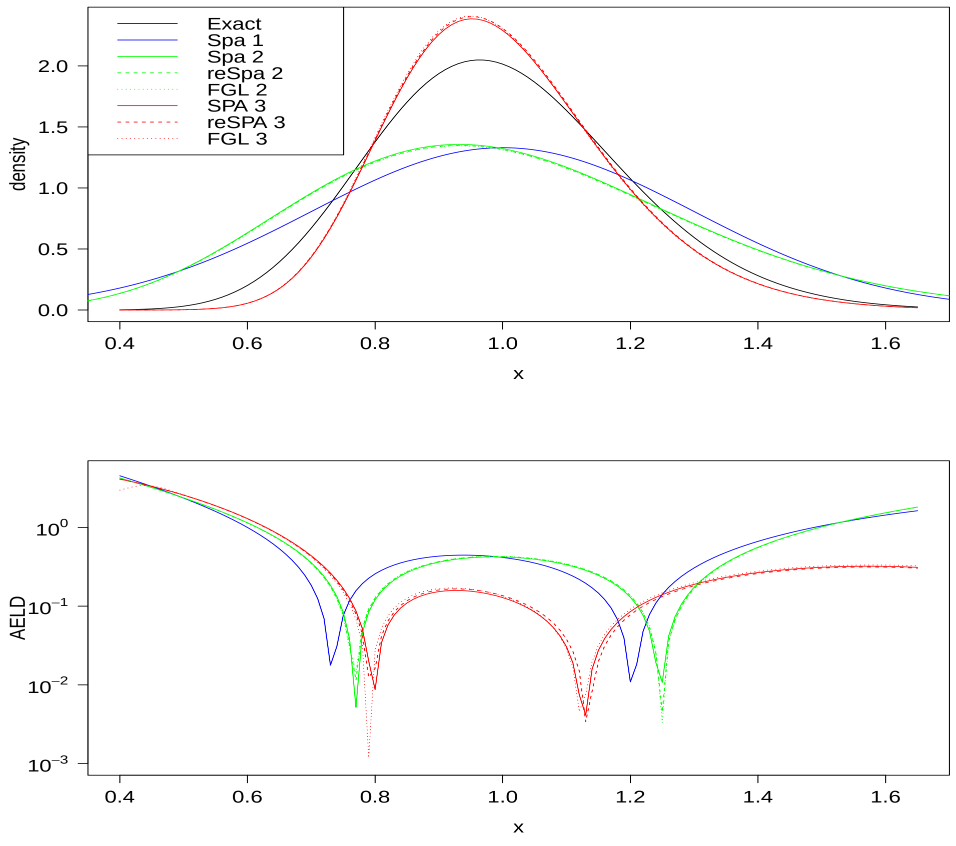

FIGURE 6.2: (Top) Exact and estimated transition densities for the CIR process with starting value $x_0 = 1$, parameters $\kappa = 1$, $\alpha = 1$, and $\sigma = 0.3$, and with timestep $t = 1$. (Bottom) AELD plotted for the same values of the estimated transition densities.

the same as the renormalized SPA. This suggests that the error stems from the Itô-Taylor approximation, and not from the saddlepoint approximation, when considering processes without jumps. This seems to hold for both the Milstein scheme and scheme 3.

For a different (and perhaps rather artificial) set of parameters, we see that the AELD in figure 6.2 is much larger than for the previous example. The shape of the estimated transition densities for both scheme 1 and scheme 2 seems to deviate from the exact density. The shape provided by the estimation via scheme 3 seems to be more in accordance with the exact density. For the left tail, the AELD measurements are very similar, while for the right tail scheme 3 provides a better estimate. Scheme 3 outperforms the others clearly where the density has the most mass. Thus, in general, scheme 3 provides the better estimate, but

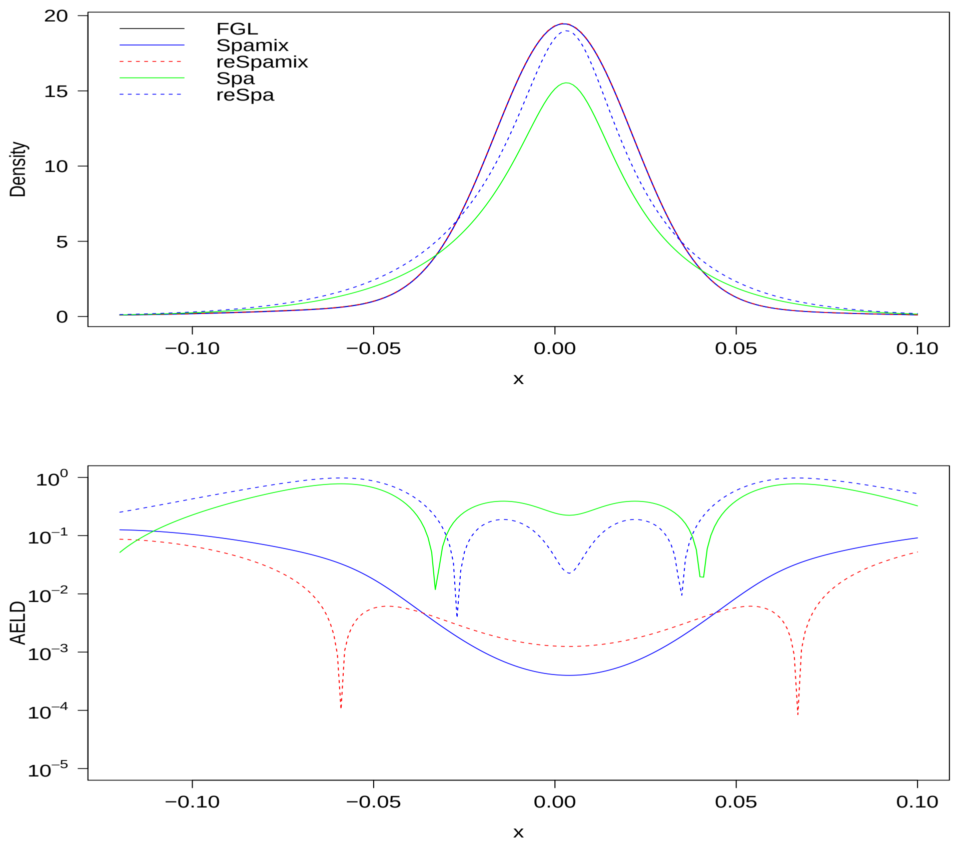

Figure 6.3: (Top) Exact and estimated transition densities for the MJD model (for log-returns) with timestep $t = 1/250$ and parameters $r = 0.55$, $\sigma = 0.2$, $\lambda = 18$, $\mu = -0.01$, and $\nu = 0.04$.
(Bottom) AELD plotted for the same values of the estimated transition densities.

there exist points (close to where $\hat{f} - f$ changes sign) where scheme 1 and scheme 2 perform better.

In figure 6.3 we see the exact and estimated transition densities for the MJD model for log-returns. For this particular example, the drift and diffusion coefficients are constants, and all schemes will therefore provide exact estimates for this instance. As noted earlier in this section, we here take the FGL approximation as our benchmark. It is evident from both the transition density plot and the AELD that the ITSPA and also its renormalized version perform poorly compared to the other methods. The ITSPA and the renormalized version seem to have fatter tails and sharper peaks compared with the others. The peak of the renormalized version is also very close to the peak of the exact density, as can be

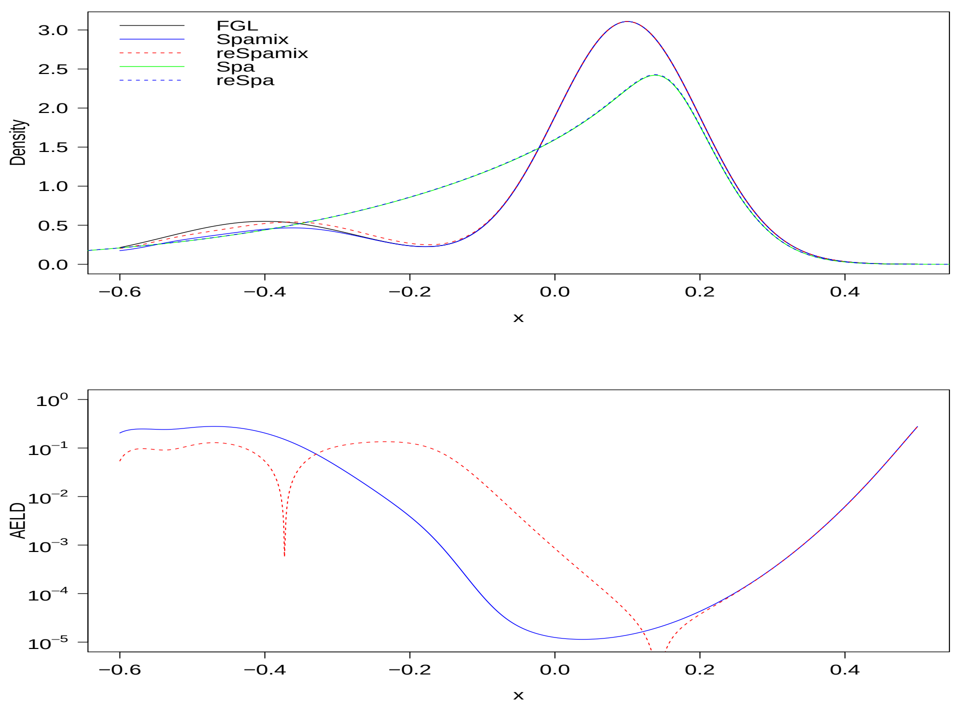

FIGURE 6.4: Example of bimodal transition density for the MJD model (for log-returns) with timestep $t = 1/4$ and parameters $r = 0.03$, $\sigma = 0.2$, $\lambda = 1$, $\mu = -0.5$, and $\nu = 0.1$.

seen from the AELD changing sign at this point. The reason why the mITSPA is preferable to the ITSPA is evident from both the plot of the transition density and the AELD. The renormalized version of the mITSPA is the following approximation:

$$\text{remspa}\left(f_{\widetilde{X}_t}; x\right) = \text{spa}\left(f_{\widetilde{X}_{t-}}; x\right) e^{-\lambda t} + \frac{\text{spa}\left(f_{\widetilde{X}_t^*}; x\right)}{\int \text{spa}\left(f_{\widetilde{X}_t^*}; x\right) dx} \left(1 - e^{-\lambda t}\right), \qquad (6.1.2)$$

so the SPA for $\widetilde{X}_t^*$ is renormalized, but not the pure diffusion part. The reason why we do not renormalize the whole expression is evident from figure 6.4. For some parameters, the MJD model will be bimodal, which is a shape the ITSPA is not able to replicate. Both the mITSPA and its renormalized version perform very well for the right part (to the right of -0.1), but the mITSPA underestimates the impact of jumps. The renormalized version handles this problem well. With these results in mind, we continue with the mITSPA and the DFT as approximation methods in the next section and in chapter 7.

Method		Parameters		
		μ	σ	$l(\hat{\theta}; \mathbf{x})$
Exact	Estimate	0.04551693	0.20474323	2040.185
	Standard error	0.118292400	0.005289691	
ITSPA				
Scheme 1	Estimate	0.04552182	0.20480000	2040.051
	Standard error	0.118320241	0.005291158	
Scheme 2	Estimate	0.04553684	0.20479341	2040.232
	Standard error	0.118321415	0.005292543	
Scheme 3	Estimate	0.04551767	0.20475610	2040.232
	Standard error	0.118299834	0.005289801	
reITSPA				
Scheme 2	Estimate	0.04552938	0.20478127	2040.235
	Standard error	0.118314384	0.005291601	
Scheme 3	Estimate	0.04552016	0.20474397	2040.235
	Standard error	0.118292829	0.005288862	
FGL				
Scheme 2	Estimate	0.04552548	0.20478043	2040.185
	Standard error	0.118313815	0.005291554	
Scheme 3	Estimate	0.04552666	0.20474278	2040.185
	Standard error	0.118292071	0.005288793	

TABLE 6.1: Parameter estimates for the GBM, with $\Delta t = 1/250$ and $n = 750$. True parameters are $\mu = 0.1$ and $\sigma = 0.2$.

6.2 Approximation methods applied to likelihood-based analysis

In the following we present likelihood-based estimation of parameters using the methods previously discussed. This is compared to estimation based on using the exact transition densities, or the FGL in the case of the Merton model. The estimation results for the GBM, the OU process, the CIR process, and the Merton model are found in tables 6.1, 6.2, 6.3, and 6.4, respectively. The data used for the estimation were generated using the solutions of the SDEs in chapter 2.3. The data generated were generated with timesteps $\Delta t = 1/250$ for

Method		Parameters			
		κ	α	σ	$l(\hat{\theta}; \mathbf{x})$
Exact	Estimate	0.6371062	0.4767801	0.2056826	1030.498
	Standard error	0.149601549	0.041683387	0.005564301	
ITSPA					
Scheme 1	Estimate	0.6204889	0.476780	0.2003414	1030.498
	Standard error	0.141866267	0.041683387	0.005279164	
Scheme 2	Estimate	0.6204889	0.4767801	0.2003414	1030.498
	Standard error	0.141866267	0.041683387	0.005279164	
Scheme 3	Estimate	0.6374181	0.4767801	0.2057819	1030.498
	Standard error	0.14982442	0.04168339	0.00557789	
FGL					
Scheme 2	Estimate	0.6205907	0.4755344	0.1999412	1030.6
	Standard error	0.141603579	0.041653429	0.005401377	
Scheme 3	Estimate	0.6375256	0.4755344	0.2053718	1030.6
	Standard error	0.149548434	0.041653428	0.005697267	

TABLE 6.2: Parameter estimates for the OU process with $\Delta t = 1/12$ and $n = 720$. True parameters are $\kappa = 0.5$, $\alpha = 0.5$, and $\sigma = 0.2$.

the GBM, $\Delta t = 1/52$ for the CIR process, $\Delta t = 1/12$ for the OU process, and $\Delta = 1/250$ for the Merton model, mimicking daily, weekly, and monthly observations for financial data.

Considering the point estimates of the parameters and the uncertainty of these estimates, all the methods and schemes produce good results compared to the estimates based on using the exact transition densities. In addition, the evaluations of the log-likelihood functions in their respective optima are practically the same. This tells us that the renormalization of the ITSPA does not have a large and beneficial effect when one is working with pure diffusions. Likelihood-based inference with the ITSPA without renormalization is therefore possible and quite accurate for pure diffusions. This is also reasonable in view of the estimated transition densities (figures 6.1 and 6.2) in section 6.1: when the timestep is small, even the Euler-Maruyama scheme (4.1.5), which is normal, provides a reasonable approximation. The SPA is exact for a normal random variable, and it is therefore reasonable to assume that transition densities from higher-order schemes will be approximated accurately with the SPA, since the true (and estimated) transition densities will be close to normal. Renormalization does not

Method		Parameters			
		κ	α	σ	$l(\hat{\theta}; \mathbf{x})$
Exact	Estimate	2.6364617	1.0590417	0.5287758	434.2396
	Standard error	0.55194190	0.04644282	0.01766123	
ITSPA					
Scheme 1	Estimate	2.5979632	1.0586450	0.5018126	434.2881
	Standard error	0.49744663	0.04470062	0.01586857	
Scheme 2	Estimate	2.4631499	1.0590665	0.5033727	434.2972
	Standard error	0.49845482	0.04734476	0.01598954	
Scheme 3	Estimate	2.6323468	1.0591018	0.5305189	434.7113
	Standard error	0.55584460	0.04667515	0.01792541	
reITSPA					
Scheme 2	Estimate	2.4630148	1.0589346	0.5029161	433.8832
	Standard error	0.49801072	0.04729622	0.01594586	
Scheme 3	Estimate	2.6198579	1.0593195	0.5297991	434.2021
	Standard error	0.55459115	0.04684416	0.01784453	
FGL					
Scheme 2	Estimate	2.4625853	1.0845777	0.5032077	433.8103
	Standard error	0.49791928	0.04912341	0.01597505	
Scheme 3	Estimate	2.6189290	1.0592701	0.5297533	434.1375
	Standard error	0.55448542	0.04685458	0.01783991	

Table 6.3: Parameter estimates for the CIR process with $\Delta t = 1/52$ and $n = 624$. True parameters are $\kappa = 2$, $\alpha = 1$, and $\sigma = 0.5$.

seem to be crucial here. In table 6.4, we estimated parameters for $T = 1$, $T = 3$, and $T = 5$. As mentioned, renormalization of the mITSPA seems to be necessary both for parameter estimates and especially the value of the likelihood.

In table 6.5 we compare the speed (in milliseconds) of the different methods relative to each other. The methods involving saddlepoint approximations without renormalization (ITSPA and mITSPA) are faster than saddlepoint approximations with renormalization and also faster than the FGL method, which is second in speed. These results are valid for both the CIR process and the MJD model. It is also interesting to note how much more time the calculation of the gradient and the Hessian matrix is consuming than the log-likelihood

Method		Parameters					
		r	σ	λ	μ	ν	$l(\hat{\theta}; \mathbf{x})$
				$T = 1$			
mitspa	est	0.72074	0.30700	12.86635	0.00132	0.06056	596.015
	se	0.37759	0.01912	8.82091	0.02037	0.01959	
remitspa	est	0.72172	0.30249	16.55132	0.00040	0.05570	596.398
	se	0.38265	0.02103	12.93950	0.01719	0.01938	
FGL	est	0.72003	0.30272	16.43486	0.00045	0.05430	596.336
	se	0.37553	0.02060	12.27560	0.01644	0.01830	
				$T = 3$			
mitspa	est	0.38524	0.30172	24.74671	-0.01911	0.06328	1717.037
	se	0.25461	0.01067	5.65762	0.00889	0.00779	
remitspa	est	0.36125	0.29691	29.68622	-0.01735	0.05953	1720.067
	se	0.26023	0.01110	7.26270	0.00816	0.00734	
FGL	est	0.38405	0.29776	28.79432	-0.01692	0.05882	1719.595
	se	0.25321	0.01102	6.92008	0.00790	0.00729	
				$T = 5$			
mitspa	est	0.59237	0.30154	30.57034	0.00051	0.04735	2897.868
	se	0.17890	0.00959	6.83584	0.00484	0.00515	
remitspa	est	0.59498	0.28530	52.00543	0.00060	0.03925	2904.966
	se	0.18215	0.01220	15.40468	0.00339	0.00509	
FGL	est	0.59175	0.28959	45.15461	0.00054	0.04058	2903.782
	se	0.17819	0.01099	11.49991	0.00360	0.00457	

Table 6.4: Parameter estimates (est) with standard deviations (se) for the MJD model for log-returns with $T = 1$, $T = 3$, and $T = 5$, and $\Delta t = 1/250$. True parameters are $r = 0.4$, $\sigma = 0.3$, $\lambda = 30$, $\mu = -0.01$, and $\nu = 0.05$.

Method	l_θ			$\nabla_\theta l$			$\mathbb{H}_\theta$		
	Q_1	Q_2	Q_3	Q_1	Q_2	Q_3	Q_1	Q_2	Q_3
					CIR				
ITSPA									
scheme 1	1.874	1.888	1.952	7.063	7.196	7.496	51.071	51.400	52.249
scheme 2	3.169	3.182	3.220	11.270	11.483	11.606	72.521	73.114	73.738
scheme 3	4.559	4.689	4.841	15.091	15.445	15.799	94.724	95.580	97.269
reITSPA									
scheme 2	134.809	137.473	141.612	457.098	509.000	584.649	6292.039	6298.884	6301.882
scheme 3	173.784	174.192	175.156	591.581	607.021	619.613	4275.437	4563.417	5471.84
FGL									
scheme 2	7.417	7.489	7.536	16.14098	16.445	16.777	78.665	78.742	79.279
scheme 3	24.527	24.666	25.682	56.484	57.116	58.173	273.357	273.790	274.896
					MJD				
mSPA	9.376	9.488	9.711	35.335	35.574	35.986	389.262	396.939	410.138
remSPA	150.614	151.196	152.677	539.840	541.689	543.621	8255.338	12120.45	12138
FGL	15.985	16.336	16.666	35.778	36.031	36.344	217.680	219.360	220.330

Table 6.5: Microbenchmarking approximation methods for the CIR process and the MJD model for log returns. The time (in milliseconds) it takes to evaluate the log likelihood, the gradient, and the Hessian matrix for the approximation methods. Evaluation of each expression was replicated 100 times, and the quartiles for each method and expression are shown in the table. The data used were simulated exactly, for the CIR process: 500 equidistant data points with $T = 20$, for the MJD model: 750 equidistant data points with $T = 3$. Parameters for both processes were set to the same values as the ones used for testing accuracy in tables 6.3 and 6.4. The ITSPA used 3 Newton iterations to find the saddlepoint, the renormalization used 45 points with $k = 4$ (see section 5.4). The mITSPA used 4 Newton iterations to find the saddlepoint, the renormalization used 25 points with $k = 2.8$. The FGL for the CIR process was used with $n = 60$, and for the MJD model: $n = 70$.

function. Having compared both speed and accuracy, it seems natural to suggest the ITSPA method over the others when working with pure diffusion processes. In addition to being efficient and accurate, it seems to be even more stable than the FGL method, when working with the real data and diffusion models in chapter 7. However, for jump-diffusion processes, the FGL method is preferable compared to the other methods. It is not as fast as the mITSPA, but it provides better accuracy. Compared to the renormalized mITSPA, it is faster, more accurate and also allows for higher jump intensities without crashing.

Chapter 7

Case Studies: Analysis of Stock Prices as Nonlinear Processes

In this section we analyse financial data, using the approximation methods presented in chapter 4. The first section concerns a brief presentation of some background theory and stylized facts for financial data, such as non-constant volatility and heavy-tailed return series. The standard GBM model has been extended with certain features, such as jumps (the MJD model 2.3.4) and stochastic volatility (the Heston or Bates models), to incorporate some of these traits of financial data. But to our knowledge nonlinearity in the price process has not been thoroughly investigated. An exception from this is the *constant elasticity of variance* model (CEV) Cox (1975), which allows for nonlinearity in the diffusion component of the SDE. The aim of this chapter is to investigate whether such nonlinearity is appropriate or not. The first section considers background theory. The likelihood-based analysis for stock prices modelled with nonlinear SDEs with and without jumps is carried out in section 2. In section 3 we briefly compare some models for stochastic volatility.

7.1 Background Theory

Most financial models have their basis in the *efficient market hypothesis* (EMH). This hypothesis states that markets are informatively efficient, in the sense that all available information is incorporated into asset prices. Therefore it is in this context impossible to consistently achieve returns in excess of average market returns on a risk-adjusted basis (Brealey et al., 2012, Chapter 13, p. 317). EMH depends upon several assumptions made about the market and its participants. One of the most important (and frequently discussed) of those is the rationality assumption made about agents in the markets. It can be formalized in terms of Bayesian statistics:

1. Agents hold a prior probability belief over states of the world.
2. Agents obtain new information about individual stocks or about macroeconomic events.
3. Agents update their prior probability belief to form a posterior probability belief using Bayes' law.

Stock price models such as the GBM (2.3.1) and the Merton model (2.3.4) are compatible with EMH.

To discuss the appropriateness of EMH, some *stylized facts* for financial data are needed, based upon inferences drawn from empirical observations concerning log-returns for equities, indices, exchange rates, and commodity prices (McNeil et al., 2005):

1. Return series are not iid, although they show little serial correlation.
2. Series of absolute or squared returns show profound serial correlation.
3. Conditional expected returns are close to zero.
4. Volatility appears to vary over time.
5. Return series are heavy-tailed.
6. Extreme returns appear in clusters.

In the framework of the EMH, large price jumps and rare events are often incorporated using the "Black Swan" concept developed by Nassim Taleb (Taleb, 2010). This can be incorporated in the GBM model by extending the SDE with a jump component, leading to e.g. the Merton model if jumps are lognormally distributed. Stochastic volatility has been incorporated via choosing the instantaneous variance in the GBM model to follow a CIR process (2.3.3), this is known as the Heston model (Heston, 1993). The combination of the Heston model and the Merton model is known as the Bates stochastic volatility jump-diffusion model (Bates, 1996). However, some of the other facts are difficult to deal with in the framework of EMH. If log-returns are not iid, then the random walk hypothesis breaks down, and if log-returns are iid and heavy-tailed, the GBM model is not a suitable model. We do however note that for longer time intervals such as months and years, return series seem to behave more as iid random variables (McNeil et al., 2005).

A somewhat different approach to financial modelling is proposed in Johansen et al. (2000); Sornette and Andersen (2002); Lin et al. (2009). They view financial markets as complex systems where investors interact with each other. The perhaps most interesting part is the notion of nonlinear behaviour of stock prices due to positive reinforcement or herding behaviour (violating the rationality assumption) leading to crashes as critical points. According to Johansen et al. (2000), the easiest way to describe a mimicking process, S_t, is in accordance with the equation

$$dS_t = rS^{\alpha}, \tag{7.1.1}$$

where $\alpha > 1$ (Johansen et al., 2000). It is then possible to use this description to extend standard stock price models. The "natural" extension of the geometric Brownian motion,

$$dS_t = rS_t^{\alpha}dt + \sigma S_t^{\alpha}dW_t, \tag{7.1.2}$$

is proposed in Sornette and Andersen (2002), but is there not further investigated. Instead they propose a similar model containing parts "as a convenient device to simplify the Itô calculation of these stochastic differential equations" (Sornette and Andersen, 2002). Such additions with no economic interpretation might be considered unaesthetic and makes the model less attractive. But with the methods discussed in chapter 4, we can extend and

Model name	Drift component	Diffusion component	Jump component
GBM	rS_t	σS_t	None
CEV	rS_t	σS_t^α	None
nlModel 1	rS_t^α	σS_t^α	None
nlModel 2	rS_t^α	σS_t^β	None
MJD	$(r - \lambda\hat{k})S_t$	σS_t	Log-normal
CEVJD	$(r - \lambda\hat{k})S_t$	σS_t^α	Log-normal

TABLE 7.1: Stock price models considered in the preceding section. The first nonlinear model (nlModel 1) is the model described by equation (7.1.2), the second nonlinear model (nlModel 2) is a refinement where the exponent is allowed to be different for the drift and diffusion coefficients. The constant elasticity of volatility jump-diffusion (CEVJD) model is the CEV model extended with a log-normal jump component.

analyse the standard stock-price models to nonlinear models in a natural way. A model similar to (7.1.2) is the CEV model, where only the diffusion part is allowed to be nonlinear. Table 7.1 gives an overview over the models to be investigated.

7.2 Analysis of Stock Prices as Nonlinear Processes

It is interesting to investigate whether stock prices emit nonlinear behaviour ($\alpha \neq 1$). Using stock price data of daily returns ($\Delta t = 1/250$) on the Shanghai Securities Exchange (SSE) from 03.01.2005 until 16.10.2007, the Dow Jones Industrial Average (DJIA) from 29.04.1925 until 03.09.1929, and data from every other day ($\Delta t = 1/125$) on the Standard & Poors 500 (S&P500) from 11.10.1990 until 24.03.2000, we estimate parameters for the models in table 7.1 for different time periods and bubbles. In addition, we evaluate the hypothesis $H_0 : \alpha = 1$ against the alternative hypothesis $H_1 : \alpha \neq 1$ by calculating twice the log of the likelihoods' ratio (denoted by D), i.e.:

$$D = 2\left(l(\hat{\theta}_1; \mathbf{x}) - l(\hat{\theta}_0; \mathbf{x})\right), \tag{7.2.1}$$

where the subscripts denote the respective hypothesis, exploiting that the GBM and the MJD are special cases ($\alpha = 1$) of their nonlinear counterparts.

Model		Parameters							Statistics		
		r	σ	α	β	λ	μ	ν	$l(\hat{\theta};\mathbf{x})$	D	p-value
						SSE bubble of 07					
GBM	est	0.6268	0.2629						1796.7		0.0013
	se	0.1604	0.0071								
CEV	est	0.4718	0.0118	1.4072					1826.2	59	0.0011
	se	0.1478	0.0021	0.0244							
nlModel 1	est	0.0249	0.0112	1.4132					1827.6	61.8	0.0030
	se	0.0082	0.0019	0.0235							
nlModel 2	est	0.0001	0.0120	2.0744	1.4046				1828.9	64.4	0.0020
	se	0.0005	0.0022	0.3920	0.0247						
MJD	est	0.6261	0.1694			92.1	-0.0039	0.0202	1840.9	88.4	0.7645
	se	0.1584	0.0272			64.9	0.0027	0.0051			
CEVJD	est	0.4356	0.0128	1.3769		13.6	-0.0094	0.0345	1851.8	110.2	0.0179
	se	0.1525	0.0033	0.0345		8.2	0.0086	0.0085			
						S&P500 Dot-com bubble					
GBM	est	0.1817	0.1399						3536.6		0.0021
	se	0.0460	0.0020								
CEV	est	0.1798	0.0095	1.4093					3587.4	101.6	0.0087
	se	0.0420	0.0010	0.0175							
nlModel 1	est	0.0133	0.0097	1.4072					3587.2	101.2	0.0098
	se	0.0034	0.0010	0.0177							
nlModel 2	est	0.1855	0.0095	0.9950	1.4094				3587.4	101.6	0.0088
	se	0.7987	0.0010	0.6829	0.0175						
MJD	est	0.1797	0.0731			118.0	0.0002	0.0108	3580.8	88.4	0.9661
	se	0.0452	0.0110			38.1	0.0005	0.0012			
CEVJD	est	0.1801	0.0019	1.6170		31.1	0.0020	0.0140	3625.0	176.8	0.5466
	se	0.0783	0.0006	0.0185		66.3	0.0042	0.0103			
						DJIA 1929 bubble					
GBM	est	0.2328	0.1476						4224.8		0.000009
	se	0.0647	0.0028								
CEV	est	0.2205	0.0101	1.5056					4255.1	60.6	0.000015
	se	0.0620	0.0011	0.0227							
nlModel 1	est	0.0160	0.0101	1.5051					4255.3	61.0	0.000027
	se	0.0048	0.0011	0.0228							
nlModel 2	est	0.0024	0.0072	1.7331	1.5038				4255.3	61.0	0.000025
	se	0.0134	0.0006	1.0447	0.01688						
MJD	est	0.2223	0.0961			104.9	-0.0033	0.0104	4295.8	142	0.9153
	se	0.0647	0.0057			25.1	0.0008	0.0008			
CEVJD	est	0.2234	0.0061	1.5490		49.4	-0.0047	0.0130	4315.3	181	0.3694
	se	0.0618	0.0011	0.0394		8.9003	0.0002	0.0006			

TABLE 7.2: Parameter estimates (est) with standard deviations (se) for the (log) stock price models, in addition to the likelihood values, the D statistic (7.2.1) where the GBM is the model under the null hypothesis, and the p-values from the Kolmogorov-Smirnov test for uniformity after the transformation (7.2.2).

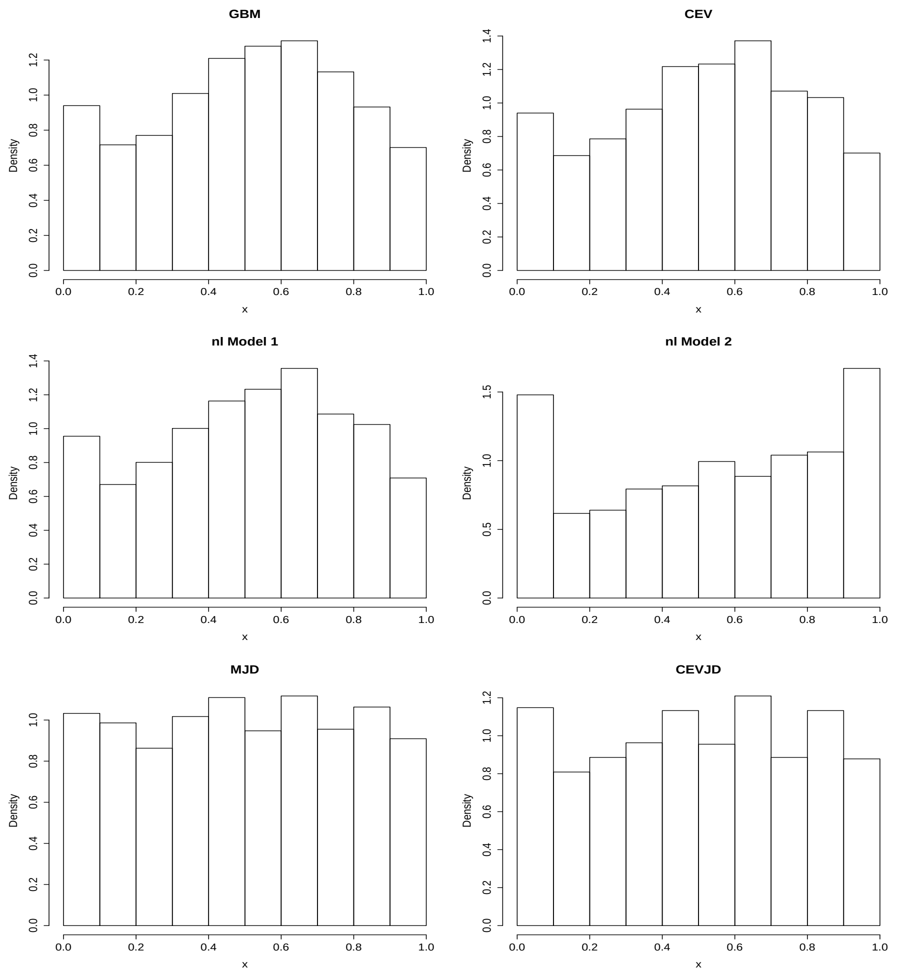

FIGURE 7.1: Goodness-of-fit diagnostic for different models. Histograms of the quantity (7.2.2) should be compared to a uniform distribution.

As a diagnostic (goodness of fit) to test whether the models are at all reasonable models for logarithmic stock prices, we transform the data in the following way: given a model m for the logarithmic stock prices $x_1, \ldots, x_n$, the data are transformed according to

$$y_i = F^m_{X_t}(x_i) \tag{7.2.2}$$

for $i = 2, \ldots, n$, where $F^m_{X_t}$ is the distribution function of $X_{t_i}|X_{t_{i-1}}$, given the model m. Under the assumption that the data indeed follow the model m, we can test the transformed data for uniformity on the interval $[0, 1]$ using the Kolmogorov-Smirnov test. This test will however not be perfectly accurate, as it requires observations of iid random variables. Our

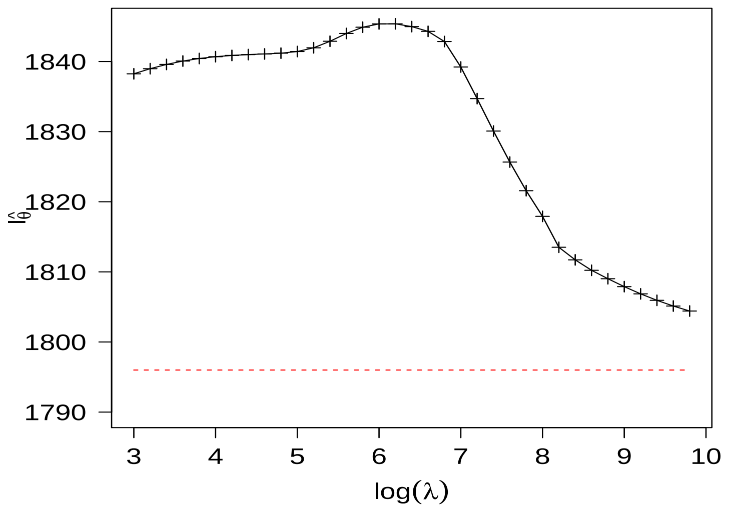

FIGURE 7.2: Profile likelihood versus logarithmic jump intensities together with the value of the optimized GBM log-likelihood (dashed line).

original observations form a time series, and are not iid.

All the data were transformed to logarithmic indices. The models must then be transformed using Itô's lemma 2.3. Using the ITSPA (4.1) with scheme 3 for the pure diffusion models, and the FGL method (4.3) also with scheme 3 for the jump-diffusions, we then estimate parameters and evaluate the log-likelihoods at the optima. The results can be found in table 7.2, showing significant support in favour of the nonlinear models. Comparing the D statistic with the chi-square distribution, we see that both the addition of nonlinearity and of jumps are significant improvements. The likelihood of the CEV model compared to the nonlinear models (nlModel 1 and nlModel 2) are very close to one another. From this we can not suggest that agents in the markets have so-called bounded rationality or herding behaviour as described in Sornette and Andersen (2002), since the nonlinearity parameter in the drift part of the SDE is not a significant addition to the model. We therefore have chosen the simplest model (CEV) and have extended it with a jump component, leading to what we call the *constant elasticity of variance jump-diffusion* model (CEVJD). This was done in order to see if the data continued to emit nonlinear behaviour, even when allowing for jumps. And for all three datasets this was indeed the case.

The MJD model aims at modelling relatively rare events leading to abnormal changes in the stock price. According to this model, the jump intensity λ should be fairly low, since by definition rare events do not happen very often. For all three datasets, the $\hat{\lambda}_{ml}$ estimates for the MJD model are around 100, but for the CEVJD, the estimated jump intensities are 13.6, 31.1, and 49.4. A possible interpretation is the following: the behaviour that is captured as nonlinearity in the CEVJD model is so significant for the value of the log-likelihood of the MJD model, that instead of modelling rare events, the behaviour is captured in the jump component as several small jumps. The estimated jump sizes μ and variances ν^2 are also greater in absolute value for the CEVJD model than for the MJD model, which supports our interpretation.

The ITSPA for diffusions and the FGL method for jump-diffusions (both with scheme 3) were chosen on the basis of their speed, their stability and their accuracy. As mentioned in chapter 6, the SPA methods (without renormalization) are faster than the FGL methods (see table 6.5), and for diffusion processes without jumps they also seem to be more stable than the FGL methods (and also SPA with renormalization). However, when working with jump-diffusion processes, the FGL method is more accurate than the SPA method (renormalization is needed, see chapter 6), reasonably fast (faster than SPA with renormalization), and it seems to be quite stable (more stable than SPA with renormalization).

As a side note, we have plotted the profile likelihood (MJD log-likelihood) versus the fixed values of the logarithmic lambda in figure 7.2. It seems that the value of the profile likelihood tends towards the value of the GBM optimized likelihood, when the jump intensity grows large. This is in accordance with theorem 2.5.

From the p-values from the Kolmogorov-Smirnov test in table 7.2 and the histograms in figure 7.1 (bearing in mind the inaccuracy described earlier), we see that the data transformed with models that include the possibility of jumps are closer to a uniform distribution on $[0, 1]$. On the 95% confidence level, the CEVJD model is rejected for the SSE data, while the MJD is not. Indeed, the p-values for transformed data under the MJD model are greater than those for the CEVJD. This is interesting, because the MJD is a special case of the CEVJD model, and this must imply that optimizing the value of the likelihood is not equivalent to optimizing for uniformity for the transformed data.

7.3 Stochastic Volatility Models

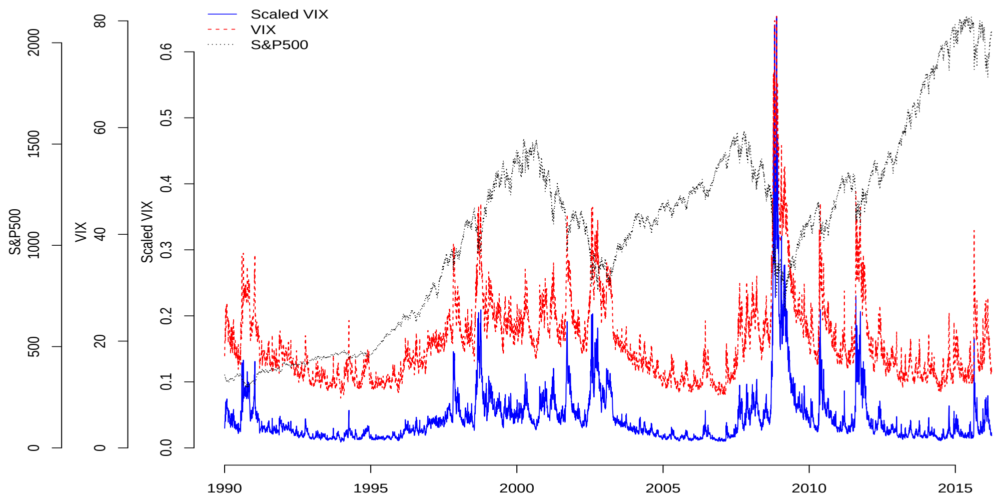

FIGURE 7.3: The scaled VIX for daily variance plotted together with the original VIX and the underlying S&P500 indices.

.

The volatility for log-returns has for some time been known not to be constant, indeed this is one of the stylized facts 7.1 and what we tried to incorporate in the CEV and the CEVJD models. Several stochastic volatility models already exist to incorporate this feature. We have studied some of the more popular ones, where the variance follows a CIR process (Heston, 1993), the continuous time GARCH model (Brockwell et al., 2006), the 3/2 model, and the OU process. The model specifications can be found in table 7.3. Using the ITSPA with the Euler scheme, we estimate parameters and evaluate the log-likelihood at the respective optima. We also consider a more general model which has the model SDE specification

$$dV_t = \kappa(\alpha - V_t)dt + \sigma V_t^{\delta} dW_t, \tag{7.3.1}$$

for which we see that the OU, CIR, and continuous time GARCH(1,1) are special cases ($\delta = 0$, $\delta = 0.5$, and $\delta = 1$ respectively). We refer to this process as the *general mean*

Model				Estimates				Statistic
Name	Drift	Diffusion		κ	α	σ	δ	$l(\hat{\theta}, \mathbf{x})$
OU	$\kappa(\alpha - V_t)$	σ	est	7.46228	0.04550	0.18001		20308
			se	0.76305	0.00469	0.00158		
CIR	$\kappa(\alpha - V_t)$	$\sigma\sqrt{V_t}$	est	3.34275	0.04543	0.49902		24585
			se	0.73762	0.00617	0.00433		
GARCH(1,1)	$\kappa(\alpha - V_t)$	σV_t	est	2.22330	0.05407	2.13335		26096
			se	0.81318	0.01295	0.01855		
3/2 model	$V_t(\alpha - \kappa V_t)$	$\sigma V_t^{\frac{3}{2}}$	est	86.37495	5.96746	12.26975		25646
			se	18.13629	0.65073	0.10669		
GMR	$\kappa(\alpha - V_t)$	σV_t^{δ}	est	2.19812	0.05449	2.99164	1.10223	26133
			se	0.84034	0.01386	0.12376	0.01202	

TABLE 7.3: Stochastic volatility models and their respective parameter estimates using the scaled VIX data.

reverting process (GMR).

To perform likelihood-based inference, we use the 6613 daily observations ($\Delta t = 1/250$) of the VIX-index for the S&P500 from January 1990 to March 2016 as our observations of implied volatility. The VIX is an approximation of expected yearly volatility in percentage. We therefore transform it to an approximation of expected yearly variance, using the function $f(x) = \left(\frac{x}{100}\right)^2$. Seeing that in the calculation of the VIX, the approximated monthly expected variance is calculated first, this transformation is just a transformation back to the original approximated variance and therefore reasonable to apply. The VIX, the scaled VIX and the underlying S&P500 index quotes are plotted in figure 7.3. For calculation of the VIX and its relation with variance swaps we refer to Carr and Madan (1998), Carr and Wu (2005), and Exchange (2009).

The aim of this section is to find the MLE of δ in the GMR model. This was found to be 1.102 with standard error 0.012. Of all the special cases, the continuous time GARCHA(1,1) is the one closest to this result. However, the value of the log-likelihood function for the GMR model has a significant increase over that of the GARCH(1,1) (and also for all the other models). The GMR therefore seems to be the preferable model for the stochastic variance in stochastic volatility models for stock prices.

Chapter 8

Conclusion and Comments

In this thesis we have considered the problem of likelihood-based analysis for a somewhat general time-homogeneous jump-diffusion process. In chapters 2 and 3 we have presented brief introductions to the preliminary mathematical theory needed for the approximation methods in chapter 4, as well as to the benchmark processes such as the GBM, the OU process, the CIR process, and the MJD model. We have also given limiting theorems for the compound Poisson process and the MJD model in lemma 2.4 and theorem 2.5.

The essential outcome of chapter 4 is the three approximation methods to the transition density of a jump-diffusion. The methods are tested for the benchmark processes in chapter 6, where we first plotted transition densities for the CIR and MJD processes with different sets of parameters. The ITSPA to the transition density of a jump-diffusion performed poorly, and we therefore rejected the method. We then performed likelihood-based analysis with simulated data for all of the benchmark processes, comparing them with likelihood-based analysis using the exact transition densities. All the discretization schemes performed well, and the SPA was shown to produce very similar results to that of a DFT when considering pure diffusions. For the jump-diffusions, we found that a renormalization of the SPA in the jump component of the mITSPA is necessary, both for parameter estimates and for the value of the log-likelihood. We have also tested the speed of the methods, and have found that the ITSPA and mITSPA are the fastest for diffusions and jump-diffusions, respectively.

Chapter 5 deals with the theory of AD, and the newly released programming package TMB. We have here presented examples relating to the computational problems in this thesis, to illustrate the benefits of AD and TMB. Small extensions such as the inclusion of the modified Bessel function of the first kind and the log-normal density were implemented in TMB. A larger extension was that of the templated complex data type cType, which allowed us to implement the FGL method in TMB.

In chapter 7 we have considered two case studies. In the first of these, the ITSPA and FGL methods were used in order to investigate whether nonlinearity and jumps are significant additions to standard stock price models such as the GBM. The ITSPA was chosen for diffusion processes and the FGL for jump-diffusions (both with scheme 3), on the basis of speed, stability and accuracy (see table 6.5 and the discussion in chapter 7). To this end we proposed three models, two nonlinear pure diffusion models (nlModel 1 and nlModel 2) and the CEVJD model, which we compared with existing models. The statistical evidence (values of the D statistics and p-values from the Kolmogorov-Smirnov test on the transformed data) seems to point to the answer "yes" regarding both the question of addition of nonlinearity in the diffusion part and the question of inclusion of jumps. – In the second case study, we have considered mean reverting processes as models for instantaneous variance in stochastic volatility models. We here propose a more general model, the GMR model. The results from the likelihood analysis point to the GMR model as being a statistical significant extension compared to the standard models, with the continuous time GARCH(1,1) model as the closest one of the standard models.

To finish off, we shall here mention some possibilities for further work:

1. An extension of the mITSPA and the FGL methods to several dimensions. This should be possible, considering that the Itô-Taylor expansions are available for multidimensional Itô-processes (Kloeden and Platen, 1992), which also holds true for the SPA (Kleppe and Skaug, 2008). The multidimensional Milstein scheme and its characteristic function are already calculated in Zhang and Schmidt (2016), upon which the FGL is based. A nontrivial question is: which numerical integration routine should be used for the renormalization? Quadrature rules might be a natural answer both with respect to accuracy and with respect to speed.

2. An extension of the methods to a more general jump-diffusion process, where the jump part of the SDE may be allowed to take a more general form.

3. A comparative study of the methods presented in this thesis, the closely related method in Zhang and Schmidt (2016), the method in Varughese (2013), and other approximation methods.

4. A more extensive study of nonlinearity in financial markets. Can the behaviour that is captured as nonlinearity in the CEV and CEVJD models be explained solely by stochastic volatility and jumps (e.g. the Bates model (Bates, 1996))? What implications does nonlinearity in the price process have for the pricing of derivatives?

5. The study of a more general mean reverting jump-diffusion process as a model for stochastic volatility, an extension of the *basic affine jump-diffusion* process. E.g:

$$dV_t = \kappa(\alpha - V_t)dt + \sigma V_t^{\delta} dW_t + dJ_t, \tag{8.0.1}$$

where J_t is a compounded Poisson process with gamma distributed jumps.

Appendix A

Multiple Itô Integrals

The evaluations of the integrals (4.1.1) involved in the development of the discretization schemes in section 4.1 were presented without proofs. For the non-trivial ones, we here show how they can be calculated.

The first integral of concern is the integral $I_{1,0}$. The calculation involves using the Fubini theorem for stochastic integrals (see e.g. Bjork (2009, p. 477)) for switching the order of integration:

$$\begin{aligned} I_{1,0} &= \int_0^t \int_0^{s_1} dW_{s_2} ds_1 = \int_0^t W_{s_1} ds_1 \\ &= \int_0^t \int_0^t \mathbb{1}_{[0,s_1]}(s_2) dW_{s_2} ds_1 = \int_0^t \int_0^t \mathbb{1}_{[0,s_1]}(s_2) ds_1 dW_{s_2} \\ &= \int_0^t t - s_2 dW_{s_2} \sim N\left(0, \int_0^t (t-s_2)^2 ds_2\right) \\ &\sim N\left(0, \frac{1}{3}t^3\right). \end{aligned} \tag{A.0.1}$$

The second integral of interest is the integral $I_{0,1}$. We here wish to show that the equation $I_{0,1} = tJ_1 - J_2$, where J_1 and J_2 are as defined in section 4.1, is valid. Define Y by $Y_t = tW_t$. Then we have $Y_t = f(t, X_t)$, where $f(t,x) = tx$, $X_t = W_t$, and $Y_0 = 0$. The partial derivatives are $\frac{\partial f}{\partial t} = x$, $\frac{\partial f}{\partial x} = t$, and $\frac{\partial^2 f}{\partial^2 x} = 0$. We also trivially have $dX = dW$. Itô's lemma then gives

$$dY = W_t dt + t dW_t, \tag{A.0.2}$$

which in integral form yields

$$tW_t = Y_t = \int_0^t W_s ds + \int_0^t s dW_s = W_t + \int_0^t s dW_s. \tag{A.0.3}$$

From this we see that the equation holds.

The third and final integral, $I_{1,1}$, is commonly used as an example to illustrate the Itô integral and can be found in most textbooks on the subject. It can of course be computed directly from the definition, but also via an application of Itô's lemma, similar to that of $I_{0,1}$. We first calculate the inner integral, and then we follow Bjork (2009) and the application of Itô's lemma found there:

$$I_{1,1} = \int_0^t \int_0^{s_1} dW_{s_2} dW_{s_1} = \int_0^t W_{s_1} dW_{s_1}. \tag{A.0.4}$$

Define $Y_t = W_t^2$, then $Y_0 = 0$ and Y can be written as $Y_t = f(t, X_t)$, where $X_t = W_t$ and f is a function such that $f(t,x) = x^2$. The partial derivatives of f are $\frac{\partial f}{\partial t} = 0$, $\frac{\partial f}{\partial x} = 2x$, and $\frac{\partial^2 f}{\partial^2 x} = 2$. From Itô's lemma we then have

$$dY_t = 2XdX + \frac{1}{2}2(dX)^2 = dt + 2W_t dW_t, \tag{A.0.5}$$

since $dX = dW$. In integral form this reads

$$W_t^2 = Y_t = t + 2\int_0^t W_s dW_s, \tag{A.0.6}$$

which trivially implies that $I_{1,1} = \frac{1}{2}\left(J_1^2 - t\right)$.

Our final consideration is the covariance between J_1 and J_2,

$$Cov(J_1, J_2) = Cov\left(W_t, \int_0^t W_s ds\right) = Cov\left(\int_0^t dW_s, \int_0^t \int_0^{s_1} dW_{s_2} ds_1\right). \tag{A.0.7}$$

Applying the Fubini theorem as for $I_{1,0}$ (A.0.1), and by the properties of the Itô integral (2.1), we obtain

$$Cov(J_1, J_2) = Cov\left(\int_0^t dW_s, \int_0^t (t-s)dW_s\right) = \int_0^t 1 * (t-s)ds = \frac{1}{2}t^2. \qquad \text{(A.0.8)}$$

Appendix B

Code Snippets

B.1 Additions to the TMB Package

For practical purposes (cf. section 5.3.1), the modified Bessel function of the first kind and the log-normal density were made available to the **TMB** user. Since R is running in the background and R is based on C++, the function values can be drawn from R since both functions are already implemented in R. The partial derivatives of the function must then be implemented manually. For these purposes, the first code snippet (B.1) was placed inside the file "atomic_math.hpp" in the "include" folder of the **TMB** package. In addition, to ease user implementation, the second code (B.2) was placed inside the file "convenience.hpp" of the same folder. We note that for the modified Bessel function of the first kind, a finite difference approximation of the derivative with respect to ν was used, due to the complicated expression of this term. This goes against the exactness of AD, but was tested and found to work well in practice.

.

```
TMB_ATOMIC_VECTOR_FUNCTION(
// ATOMIC_NAME
besselI
,
// OUTPUT_DIM
1
,
// ATOMIC_DOUBLE
ty[0] = Rmath::Rf_bessel_i(tx[0], tx[1], 1.0 /* Not scaled */ );
,
// ATOMIC REVERSE
Type value = ty[0];
Type x = tx[0];
Type nu = tx[1];
CppAD::vector<Type> arg(2);
arg[0] = x;
arg[1] = nu + Type(1);
px[0] = (besselI(arg)[0] + value * (nu / x)) * py[0];
arg[1] = nu + Type(0.000001);
px[1] = ((besselI(arg)[0] - besselI(tx)[0]) / Type(0.0001))* py[0];
)

TMB_ATOMIC_VECTOR_FUNCTION(
// ATOMIC_NAME
dlnorm
,
// OUTPUT_DIM
1
,
// ATOMIC_DOUBLE
ty[0] = Rmath::Rf_dlnorm(tx[0],tx[1],tx[2],0 /* log=FALSE */ );
,
// ATOMIC_REVERSE
px[0] = -ty[0]/tx[0] * (1-(log(tx[0])-tx[1]) / (tx[2]*tx[2])) * py[0];
px[1] = ty[0] * (log(tx[0])-tx[1]) / (tx[2]*tx[2]) * py[0];
px[2] = -ty[0] / tx[2] * (1 - ((log(tx[0])-tx[1]) *
                (log(tx[0])-tx[1]) / (tx[2]*tx[2]))) * py[0];
)
```

Listing B.1: Addition to atomic_math.hpp

```
template<class Type>
```

```
Type besselI(Type x, Type nu) {
CppAD::vector<Type> tx(2);
tx[0] = x;
tx[1] = nu;
return atomic::besselI(tx)[0];
}

template<class Type>
Type dlnorm(Type x, Type mu, Type sigma){
CppAD::vector<Type> tx(3);
tx[0] = x;
tx[1] = mu;
tx[2] = sigma;
return atomic::dlnorm(tx)[0];
}
```

Listing B.2: Addition to convenience.hpp

.

```
/**
* Makes the following available for the TMB user:
* The "cType<Type>" complex AD data type.
* The arithmetic follows standard arithmetic for complex variables.
* It is defined to work together with the standard TMB data type "Type".
* Constructor: cType<Type> z; defaults to 0+0*i.
*              cType<Type> z((Type)Re,(Type)Im) = Re + i*Im.
* Compound assignements (+=, -=, *=, /=).
* Relational and comparison operators (==, !=).
* Standard functions for complex variables (abs, arg, conj).
* Exponential functions (exp, log).
* Power functions (pow, sqrt).
* Trigonometric functions (sin, cos, tan, asin, acos, atan, sinh, cosh, tanh).
*
*/

template<class Type>
struct cType{
```

```
// MEMBER FUNCTIONS
Type r, i;

// Constructor
cType(void) {r=0;i=0;}
cType(Type r_, Type i_) : r(r_), i(i_) {}

// Compound assignements
cType& operator =(const cType& c){
r = c.r;
i = c.i;
return *this;
}
cType& operator +=(const Type& t){
r = r + t;
return *this;
}
cType& operator +=(const cType& c){
r = r + c.r;
i = i + c.i;
return *this;
}
cType& operator -=(const Type& t){
r = r - t;
return *this;
}
cType& operator -=(const cType& c){
r = r - c.r;
i = i - c.i;
return *this;
}
cType& operator *=(const Type t){
r = r*t;
i = i*t;
return *this;
}
cType& operator *=(const cType c){
Type tmp_r, tmp_i;
tmp_r = r*c.r - i*c.i;
tmp_i = r*c.i + i*c.r;
r = tmp_r, i=tmp_i;
return *this;
}
cType& operator /=(const Type t){
r = r / t;
i = i / t;
return *this;
}
cType& operator /=(const cType c){
Type div = c.r*c.r + c.i*c.i, tmp_r, tmp_i;
tmp_r = (r*c.r + i*c.i)/div;
tmp_i = (i*c.r - r*c.i)/div;
r = tmp_r, i = tmp_i;
return *this;
}
};

// NON-MEMBER FUNCTIONS

// Arithmetic
template<class Type>
cType<Type> operator +(const cType<Type>& c, const Type& t){
cType<Type> res = c;
return res+=t;
}
template<class Type>
cType<Type> operator +(const Type& t, const cType<Type>& c){
cType<Type> res = c;
return res+=t;
}
template<class Type>
cType<Type> operator +(const cType<Type>& c_1, const cType<Type>& c_2){
cType<Type> res = c_1;
return res += c_2;
}
template<class Type>
cType<Type> operator -(const cType<Type>& c, const Type& t){
```

```
cType<Type> res = c;
return res-=t;
}
template<class Type>
cType<Type> operator -(const Type& t, const cType<Type>& c){
cType<Type> res(t,0);
return res -= c;
}
template<class Type>
cType<Type> operator -(const cType<Type>& c_1, const cType<Type>& c_2){
cType<Type> res = c_1;
return res -= c_2;
}
template<class Type>
cType<Type> operator *(const cType<Type>& c, const Type& t){
cType<Type> res = c;
return res *= t;
}
template<class Type>
cType<Type> operator *(const Type& t, const cType<Type>& c){
cType<Type> res(t,0);
return res *= c;
}
template<class Type>
cType<Type> operator *(const cType<Type>& c_1, const cType<Type>& c_2){
cType<Type> c1 = c_1, c2=c_2;
return c1 *= c2;
}
template<class Type>
cType<Type> operator /(const cType<Type>& c, const Type& t){
cType<Type> res = c;
return res /= t;
}
template<class Type>
cType<Type> operator /(const Type& t, const cType<Type>& c){
cType<Type> res(t,0);
return res /= c;
}
template<class Type>
```

```
cType<Type> operator /(const cType<Type>& c_1, const cType<Type>& c_2){
cType<Type> res = c_1;
return res /= c_2;
}

// Relational and comparison operators
template<class Type>
bool operator ==(const cType<Type>& lhs, const cType<Type>& rhs){
cType<Type> c_1=lhs, c_2 = rhs;
if(c_1.r == c_2.r && c_1.i == c_2.i) {return true;}
else{return false;}
}
template<class Type>
bool operator ==(const cType<Type>& lhs, const Type& rhs){
cType<Type> c_1=lhs, c_2(rhs,0);
if(c_1.r == c_2.r && c_1.i == c_2.i) {return true;}
else{return false;}
}
template<class Type>
bool operator ==(const Type& lhs, const cType<Type>& rhs){
cType<Type> c_1(lhs,0), c_2=rhs;
if(c_1.r == c_2.r && c_1.i == c_2.i) {return true;}
else{return false;}
}
template<class Type>
bool operator !=(const cType<Type>& lhs, const cType<Type>& rhs){
cType<Type> c_1=lhs, c_2 = rhs;
return !(c_1==c_2);
}
template<class Type>
bool operator !=(const cType<Type>& lhs, const Type& rhs){
cType<Type> c_1=lhs, c_2(rhs,0);
return !(c_1==c_2);
}
template<class Type>
bool operator !=(const Type& lhs, const cType<Type>& rhs){
cType<Type> c_1(lhs,0), c_2=rhs;
return !(c_1==c_2);
}
```

```
// Standard functions
template<class Type>
Type abs(const cType<Type>& z){
cType<Type> c = z;
Type abs = sqrt(c.r*c.r + c.i*c.i);
return abs;
}
template<class Type>
Type arg(const cType<Type>& z){ // Returns Arg(z)
cType<Type> c = z;
return atan2(c.i,c.r);
}
template<class Type>
cType<Type> conj(const cType<Type>& z) {
cType<Type> c = z;
c.i = - c.i;
return c;
}

// Exponential functions
template<class Type>
cType<Type> exp(const cType<Type>& z) {
cType<Type> c = z;
Type temp = exp(c.r)*cos(c.i);
c.i = exp(c.r)*sin(c.i);
c.r = temp;
return c;
}
template<class Type>
cType<Type> log(const cType<Type>& z){
cType<Type> c = z;
Type r = abs(z), theta = arg(z);
c.r = log(r);
c.i = theta;
return c;
}

// Power functions
```

```
template<class Type>
cType<Type> pow(const cType<Type>& z1, const cType<Type>& z2){
cType<Type> c1=z1, c2=z2, c3;
Type a=c1.r,b=c1.i, c=c2.r,d=c2.i;
c3.r = pow((a*a+b*b),(c/2))*exp(-d*arg(c1))*
(cos(c*arg(c1)+0.5*d*log(a*a+b*b)));
c3.i = pow((a*a+b*b),(c/2))*exp(-d*arg(c1))*
(sin(c*arg(c1)+0.5*d*log(a*a+b*b)));
return c3;
}
template<class Type>
cType<Type> pow(const cType<Type>& z, const Type& t){
cType<Type> c1=z, c2(t,0);
c1 = pow(c1,c2);
return c1;
}
template<class Type>
cType<Type> pow(const Type& t, const cType<Type>& z){
cType<Type> c1=z, c2(t,0);
c1 = pow(c2,c1);
return c1;
}
template<class Type>
cType<Type> sqrt(const cType<Type>& z){
cType<Type> c1=z, c2(0.5,0);
c1 = pow(c1,c2);
return c1;
}

// Trigonometric functions
template<class Type>
cType<Type> sin(const cType<Type>& z){
cType<Type> c = z, i(0,1);
c = (exp(i*c) - exp((-(Type)1)*i*c)) / ((Type)2 * i);
return c;
}
template<class Type>
cType<Type> cos(const cType<Type>& z){
cType<Type> c = z, i(0,1);
c = (exp(i*c) + exp(c/i)) / ((Type)2);
return c;
}
template<class Type>
cType<Type> tan(const cType<Type>& z){
cType<Type> c = z;
c = sin(c) / cos(c);
return c;
}
template<class Type>
cType<Type> asin(const cType<Type>& z){
cType<Type> c = z, i(0,1);
c = (-(Type)1)*i*log( i*c + sqrt((Type)1 - (c*c)) );
return c;
}
template<class Type>
cType<Type> acos(const cType<Type>& z){
cType<Type> c = z;
c = asin((-(Type)1 ) * c) + (Type)(M_PI/2);
return c;
}
template<class Type>
cType<Type> atan(const cType<Type>& z){
cType<Type> c = z, i(0,1);
c = (-(Type)(0.5))*i*(log((Type)1 - (i*c)) - log((Type)1 + (i*c) ) );
return c;
}
template<class Type>
cType<Type> sinh(const cType<Type>& z){
cType<Type> c = z, i(0,1);
c = sin((-(Type)1)*i*c) * i;
return c;
}
template<class Type>
cType<Type> cosh(const cType<Type>& z){
cType<Type> c = z, i(0,1);
c = cos(c/i);
return c;
}
```

```
template<class Type>
cType<Type> tanh(const cType<Type>& z){
cType<Type> c = z;
c = sinh(c) / cosh(c);
return c;
}
```

Listing B.3: Creation of the complex cType data type in TMB

Bibliography

Aït-Sahalia, Y. (1999). Transition densities for interest rate and other nonlinear diffusions. *The Journal of Finance*, 54(4):1361–1395.

Aït-Sahalia, Y., Yu, J., et al. (2006). Saddlepoint approximations for continuous-time Markov processes. *Journal of Econometrics*, 134(2):507–551.

Bates, D. S. (1996). Jumps and stochastic volatility: Exchange rate processes implicit in Deutsche mark options. *Review of Financial Studies*, 9(1):69–107.

Bell, B. (2005). Cppad: A package for c++ algorithmic differentiation. https://www.coin-or.org/CppAD.

Beskos, A., Papaspiliopoulos, O., Roberts, G. O., and Fearnhead, P. (2006). Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):333–382.

Bjork, T. (2009). *Arbitrage Theory in Continuous Time*. Oxford University Press.

Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *The Journal of Political Economy*, pages 637–654.

Brealey, R. A., Myers, S. C., Allen, F., and Mohanty, P. (2012). *Principles of corporate finance*. Tata McGraw-Hill Education.

Brockwell, P., Chadraa, E., Lindner, A., et al. (2006). Continuous-time GARCH processes. *The Annals of Applied Probability*, 16(2):790–826.

Butler, R. W. (2007). *Saddlepoint approximations with applications*, volume 22. Cambridge University Press.

Carr, P. and Madan, D. (1998). Towards a theory of volatility trading. *Volatility: New estimation techniques for pricing derivatives*, (29):417–427.

Carr, P. and Wu, L. (2005). A tale of two indices. *Available at SSRN 871729.*

Cox, J. (1975). Notes on option pricing 1: Constant elasticity of variance diffusions. *Unpublished note, Stanford University, Graduate School of Business.*

Cox, J. C., Ingersoll, J. E., and Ross, S. A. (1985). A theory of the term structure of interest rates. *Econometrica*, 53(2):385–407.

Durham, G. B. and Gallant, A. R. (2002). Numerical techniques for maximum likelihood estimation of continuous-time diffusion processes. *Journal of Business & Economic Statistics*, 20(3):297–338.

Exchange, C. B. O. (2009). The cboe volatility index-vix. *White Paper*, pages 1–23.

Fournier, D. A., Skaug, H. J., Ancheta, J., Ianelli, J., Magnusson, A., Maunder, M. N., Nielsen, A., and Sibert, J. (2012). Ad model builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. *Optimization Methods and Software*, 27(2):233–249.

Gerhold, S., Schmock, U., and Warnung, R. (2010). A generalization of Panjer's recursion and numerically stable risk aggregation. *Finance and Stochastics*, 14(1):81–128.

Goutis, C. and Casella, G. (1999). Explaining the saddlepoint approximation. *The American Statistician*, 53(3):216–224.

Griewank, A. and Walther, A. (2008). *Evaluating derivatives: principles and techniques of algorithmic differentiation.* Siam.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6(2):327–343.

Johansen, A., Ledoit, O., and Sornette, D. (2000). Crashes as critical points. *International Journal of Theoretical and Applied Finance*, 3(02):219–255.

Kleppe, T. S. (2006). Numerical path integration for Lévy driven stochastic differential equations.

Kleppe, T. S. and Skaug, H. J. (2008). Building and Fitting Non-Gaussian Latent Variable Models via the Moment-Generating Function. *Scandinavian Journal of Statistics*, 35(4):664–676.

Kloeden, P. and Platen, E. (1992). *Numerical Solution of Stochastic Differential Equations.* Applications of mathematics: stochastic modelling and applied probability. Springer.

Kolassa, J. E. (2006). *Series approximation methods in statistics*, volume 88. Springer Science & Business Media.

Kristensen, K., Nielsen, A., Berg, C. W., Skaug, H., and Bell, B. (2015). Tmb: Automatic Differentiation and Laplace Approximation. *arXiv preprint arXiv:1509.00660.*

Lin, L., Ren, R., and Sornette, D. (2009). A consistent model of 'explosive' financial bubbles with mean-reversing residuals. *Swiss Finance Institute Research Paper*, (09-14).

Lindström, E. (2007). Estimating parameters in diffusion processes using an approximate maximum likelihood approach. *Annals of Operations Research*, 151(1):269–288.

Matsuda, K. (2004). Introduction to Merton jump diffusion model. *Department of Economics. The Graduate Center, The City University of New York.*

McNeil, A., Frey, R., and Embrechts, P. (2005). Quantitative risk management: Concepts, techniques, and tools.

Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1):125–144.

Øksendal, B. (2003). *Stochastic differential equations.* Springer.

Platen, E. and Bruti-Liberati, N. (2010). *Numerical solution of stochastic differential equations with jumps in finance.* Springer Science & Business Media.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). Numerical recipes in c. *Cambridge: Cambridge University.*

Preston, S. and Wood, A. T. (2012). Approximation of transition densities of stochastic differential equations by saddlepoint methods applied to small-time Ito–Taylor sample-path expansions. *Statistics and Computing*, 22(1):205–217.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Radul, A. (2013). Introduction to automatic differentiation. `http://alexey.radul.name/ideas/2013/introduction-to-automatic-differentiation/`. [Online; accessed August 15, 2013].

Rampertshammer, S. (2007). An Ornstein-Uhlenbeck framework for pairs trading. *Preprint. Available at http://www. ms. unimelb. edu. au/publications/RampertshammerStefan. pdf*.

Shoji, I. and Ozaki, T. (1998). Estimation for nonlinear stochastic differential equations by a local linearization method 1. *Stochastic Analysis and Applications*, 16(4):733–752.

Skaug, H. J. and Fournier, D. A. (2006). Automatic approximation of the marginal likelihood in non-Gaussian hierarchical models. *Computational Statistics & Data Analysis*, 51(2):699–709.

Sornette, D. and Andersen, J. V. (2002). A nonlinear super-exponential rational model of speculative financial bubbles. *International Journal of Modern Physics C*, 13(02):171–187.

Taleb, N. N. (2010). *The black swan:: The impact of the highly improbable fragility*, volume 2. Random House.

Tankov, P. (2003). *Financial modelling with jump processes*, volume 2. CRC Press.

Tucker, W. (2010). Automatic differentiation - lecture no 1. `https://www.sintef.no/globalassets/project/evitameeting/2010/ad2010.pdf`.

Varughese, M. M. (2013). Parameter estimation for multivariate diffusion systems. *Computational Statistics & Data Analysis*, 57(1):417–428.

Zhang, L. and Schmidt, W. M. (2016). An approximation of small-time probability density functions in a general jump diffusion model. *Applied Mathematics and Computation*, 273:741–758.