University of Cape Town

Doctoral Thesis

Non-Linear Diffusion Processes and Applications

Author:
Etienne A. D. Pienaar

Supervisor:
Dr. Melvin M. Varughese

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy
in Statistics

December 9, 2016

"All possible definitions of probability fall short of the actual practice."

W. Feller

UNIVERSITY OF CAPE TOWN

Abstract

Faculty of Science
Department of Statistical Sciences

Doctor of Philosophy

Non-Linear Diffusion Processes and Applications

by Etienne A. D. Pienaar

Diffusion models are useful tools for quantifying the dynamics of continuously evolving processes. Using diffusion models it is possible to formulate compact descriptions for the dynamics of real-world processes in terms of stochastic differential equations. Despite the flexibility of these models, they can often be extremely difficult to work with. This is especially true for non-linear and/or time-inhomogeneous diffusion models where even basic statistical properties of the process can be elusive. As such, we explore various techniques for analysing non-linear diffusion models in contexts ranging from conducting inference under discrete observation and solving first passage time problems, to the analysis of jump diffusion processes and highly non-linear diffusion processes. We apply the methodology to a number of real-world ecological and financial problems of interest and demonstrate how non-linear diffusion models can be used to better understand such phenomena. In conjunction with the methodology, we develop a series of software packages that can be used to accurately and efficiently analyse various classes of non-linear diffusion models.

Acknowledgements

I would like to express my sincere gratitude to Dr Melvin Varughese for providing support, guidance, insight, patience, and perhaps most importantly for believing in me and granting me the freedom to explore research avenues which would otherwise have been written off as naively ambitious. With this, I extend my gratitude to Dr Varughese and Assoc. Prof. Francesca Little for facilitating the process by which I was granted the opportunity to pursue a PhD.

I am also eternally grateful to my family, Attie, Maureen, Lizelle Diwan, Marné, and Dian. Without their unfailing love and support, this task would most certainly have been insurmountable.

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Contents

List of Figures

List of Tables

Abbreviations

AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
BAJD	Basic Affine Jump Diffusion
BPF	Bloom Potential Function
CBOE	Chicago Board Options Exchange
CGF	Cumulant Generating Function
CIR	Cox–Ingersoll–Ross (process)
DIC	Deviance Information Criterion
FFT	Fast Fourier Transform
GQD	Generalised Quadratic Diffusion
HMM	Hidden Markov Model
LHS	Left-Hand Side
MCMC	Markov Chain Monte Carlo
MGF	Moment Generating Function
MOL	Method of lines
MVN	Multivariate Normal
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PDDE	Partial Difference Differential Equation
QER	Quarterly Earnings Report
RHS	Right-Hand Side
RWMH	Random Walk Metropolis–Hastings
SDE	Stochastic Differential Equation
SPX	S&P 500 Index
VIX	S&P 500 Volatility Index
VXAZN	Amazon Equity Volatility Index
VXGOG	Google Equity Volatility Index

To my Mother and Father.

Preface

This thesis is divided into five chapters, each dealing with different aspects relating to the analysis of diffusion processes. Although each chapter has distinct research goals, the content is developed in such a way that each chapter may build on elements researched in the chapters that precede it. In the first chapter, we introduce computationally efficient numerical techniques for the analysis of non-linear diffusions in the context of an ecological application. Specifically, we use a data-imputation scheme in order to perform inference on a set of non-linear time-inhomogeneous diffusion models of the population-environment dynamics of *Emiliania huxleyi* – a coccolithophorid species known to exhibit bloom-dynamics. Based on the resulting analysis we develop a technique for measuring bloom-risk for the species based on the dynamics of the underlying diffusion model by using a numerical technique to calculate the marginal transitional density of the model process under parameter uncertainty. Based on the methodology outlined in this chapter, we develop a software package which can be used to analyse highly non-linear diffusion processes.

In the second chapter, we focus on the development of a software package for conducting inference and analysis on non-linear diffusion processes. Specifically, we aim to develop tools for analysing non-linear diffusion models as efficiently as possible and address various shortcomings in the methodology of the previous chapter. We start by developing a class of quadratic diffusion processes for which we can calculate accurate approximations to the transitional density whilst maintaining a high degree of freedom with respect to the model specification in terms of its time-inhomogeneity. Based on this architecture, we develop a novel software package for the R language which provides computationally efficient routines for analysing quadratic diffusion models. In particular, the architecture allows us to delegate computationally demanding elements of the algorithm to the C++ language within R, thus allowing for efficient implementation of the methodology whilst retaining the accessibility of the R interface and resources.

In Chapter 3 we analyse first passage time problems for non-linear diffusion processes. By their very nature, first passage time problems are extremely difficult to solve, and closed-form solutions to first passage time problems for non-linear diffusions are even rarer than closed-form expressions for transitional densities. Indeed, the difficulties associated with solving first passage time problems for non-linear diffusions stem not only from the intractability of the process dynamics but also from the nature of the stopping rule which gives rise to the first passage time event. As such, we focus on two types of first passage time problem. The first concerns the calculation of the first passage time density for non-linear scalar diffusion processes passing through a time-varying threshold function. The second focuses on time-homogeneous first passage time problems for varying dimensions where the event of interest concerns the escape of the diffusion from an arbitrarily shaped region in the state-space. In each case, we revisit the software packages developed earlier and show how the methodology can be used in order to solve such first passage time problems.

Chapter 4 focuses on the analysis of non-linear jump diffusion processes, an important class of models which generalises standard diffusion processes by allowing the trajectory of the diffusion to be influenced by a jump process. Jump diffusion processes have important applications in numerous fields of science and address a number of deficiencies associated with pure diffusion processes. Although the generalisation of diffusion processes to jump diffusion processes serve to improve the flexibility of diffusion models and allows for more realistic models of real-world processes, they are significantly more difficult to analyse. Building on the analysis of the preceding chapters, we develop a scheme for analysing jump diffusion processes by deriving a computationally efficient algorithm for approximating the transitional densities of non-linear jump diffusion processes with state-dependent and/or stochastic intensity. Using this scheme we analyse a financial time series and explore some theoretical applications of the methodology. Finally, building on the architecture of software developed in the preceding chapters we develop an R package for the analysis of quadratic jump diffusion models and show how the software can be applied to the analysis of data from real-world systems.

Finally, in Chapter 5 we delve into future research topics that have stemmed from the foregoing research. Specifically, we show partially completed work on non-linear Markov-switching diffusion models and proceed to show preliminary work on the analysis of such models. We also show preliminary work on an interesting non-standard first passage time problem with strong application potential in the field of ecology. We then give an overview of the research conducted here and provide some concluding remarks on the methodology.

Chapter 1

Data-Imputation and the Method of Lines

1.1 Diffusion processes

Diffusion processes are a class of continuous-time Markov processes that are formulated in terms of stochastic differential equations (SDEs). As such, diffusion processes can be seen as extensions of ordinary differential equations (ODEs) whereby the trajectory of the process is not only driven by a change in time, but also by a continuous-time stochastic process, namely Brownian motion. By allowing the process to be driven by Brownian motion, the trajectory of the process evolves stochastically over time with continuous sample paths. Diffusion processes thus give one the means to model dynamical processes in terms of differential equations which simultaneously account for both deterministic and stochastic behaviour.

Let $\mathbf{X}_t$ be the k dimensional state vector of the diffusion process at time t, then the dynamics of the process $\mathbf{X}_t$ is governed by the SDE:

$$d\mathbf{X}_t = \boldsymbol{\mu}(\mathbf{X}_t, t)dt + \boldsymbol{\sigma}(\mathbf{X}_t, t)d\mathbf{B}_t, \tag{1.1.1}$$

where $\boldsymbol{\mu}(\mathbf{X}_t, t) = \{\mu_i(\mathbf{X}_t, t), i = 1, ..., k\}$ gives the drift vector of the process, $\boldsymbol{\sigma}(\mathbf{X}_t, t) = \{\sigma_{ij}(\mathbf{X}_t, t), i, j = 1, ..., k\}$ gives the diffusion matrix of the process, $\boldsymbol{\sigma}(\mathbf{X}_t, t)\boldsymbol{\sigma}(\mathbf{X}_t, t)' = (\gamma_{ij}(\mathbf{X}_t, t))_{k\times k}$ where $'$ denotes matrix transposition gives the instantaneous covariance matrix of the process and $\mathbf{B}_t = \{B_t^{(j)}, j = 1, ..., k\}$

denotes a k-dimensional vector of Brownian motions or Wiener processes (throughout the present text we adopt the former naming convention). Each Brownian motion $B_t^{(j)}$ has the fundamental properties:

- $B_0^{(j)} = 0$,
- non-overlapping increments $B_t^{(j)} - B_s^{(j)}$ for $s < t$ are independent,
- and $B_t^{(j)} - B_s^{(j)} \sim \mathrm{N}(0, t-s)$ where $\mathrm{N}(0, \phi)$ is the Normal distribution with standard deviation $\sqrt{\phi}$ and mean 0.

Equation 1.1.1 thus relates how an instantaneous change in the state of the process is governed both by deterministic forces through the drift coefficients, as well as stochastic forces through the diffusion coefficients of the process. In order to see how these quantities relate to the dynamics of the process, one can express the drift and diffusion coefficients in terms of the instantaneous moments of the process:

$$\lim_{h \to 0} \frac{E[X_{t+h}^{(i)} - X_t^{(i)} | \mathbf{X}_t]}{h} = \mu_i(\mathbf{X}_t, t) \tag{1.1.2}$$

and

$$\lim_{h \to 0} \frac{E[(X_{t+h}^{(i)} - X_t^{(i)})(X_{t+h}^{(j)} - X_t^{(j)}) | \mathbf{X}_t]}{h} = \gamma_{ij}(\mathbf{X}_t, t). \tag{1.1.3}$$

Although it is easy to interpret a given diffusion process in terms of its instantaneous dynamics, practical applications of diffusion models rely on the analysis of diffusion processes over larger time scales. As such, the analysis of diffusion processes is usually concerned with solving Equation 1.1.1 over an applicable transition horizon. That is, we analyse the trajectory of the process $\mathbf{X}_t$ at time t from a known state $\mathbf{X}_s$ at time $s < t$, which is given by the integral equation:

$$\mathbf{X}_t = \mathbf{X}_s + \int_s^t \boldsymbol{\mu}(\mathbf{X}_u, u) du + \int_s^t \boldsymbol{\sigma}(\mathbf{X}_u, u) d\mathbf{B}_u. \tag{1.1.4}$$

Naturally, the existence of a solution to Equation 1.1.1 is dictated by the behaviour of the drift and diffusion coefficients. Although determining the existence of solutions to stochastic differential equations presents a challenging mathematical problem, various results have been derived with regard to the existence and uniqueness of such solutions. For example, under a known initial distribution for the process, if the coefficients of Equation 1.1.1 are Lipschitz continuous, and both $\boldsymbol{\mu}(\mathbf{X}_t, t)$ and $\boldsymbol{\sigma}(\mathbf{X}_t, t)$ are defined so as to not permit degeneracy of the diffusion over the applicable transition horizon, a 'pathwise unique' solution of

the SDE exists (Yamada *et al.*, 1971). For purposes of this thesis, we shall assume that the initial value of the process – and thus the initial distribution – is always known. Furthermore, we shall assume that the drift and diffusion coefficients, $\mu_i(\mathbf{X}_t, t)$ and $\sigma_{ij}(\mathbf{X}_t, t)$, are locally Lipschitz and satisfy the so-called linear growth conditions. Essentially, the former assumption relates to the continuity of the coefficients (as functions of $\mathbf{X}_t$) and assures that the coefficients do not exhibit (locally) unbounded rates of change, whilst the latter relates to the boundedness of the process trajectory. Combined, these assumptions fulfil the requirements for the existence of a solution to the SDE in so far as it is required for the methods that we will use to analyse such processes.

Since diffusion processes evolve stochastically over time, a principal constituent in the analysis of diffusion processes is the distribution of the state vector $\mathbf{X}_t$ over time. Let $\{S \subseteq \mathbb{R}^k, \mathcal{X}, F\}$ denote a probability space where $S \subseteq \mathbb{R}^k$ denotes the sample space, $\mathcal{X}$ denotes a σ-algebra of subsets of S, and F denotes a probability measure on events in $\mathcal{X}$, then the transition probability density function $f(\mathbf{X}_t|\mathbf{X}_s)$ of moving from state $\mathbf{X}_s$ at time s to state $\mathbf{X}_t$ at time t is given by the solution to the Kolmogorov forward equation (see, e.g., Aït-Sahalia, 2008):

$$\begin{aligned}\frac{\partial f(\mathbf{X}_t|\mathbf{X}_s)}{\partial t} = &-\sum_{i=1}^{k} \frac{\partial}{\partial X_t^{(i)}}\Big[\mu_i(\mathbf{X}_t, t) f(\mathbf{X}_t|\mathbf{X}_s)\Big] \\ &+ \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k} \frac{\partial^2}{\partial X_t^{(i)} \partial X_t^{(j)}}\Big[\gamma_{ij}(\mathbf{X}_t, t) f(\mathbf{X}_t|\mathbf{X}_s)\Big],\end{aligned} \tag{1.1.5}$$

where $X_t^{(i)}$ denotes the i-th element of $\mathbf{X}_t$ and the initial conditions of the transition density are given by the Dirac delta function, $f(\mathbf{x}_s|\mathbf{X}_s) = \delta(\mathbf{x}_s - \mathbf{X}_s)$ where

$$\delta(\mathbf{y}) = \begin{cases} \infty & \text{if } \mathbf{y} = \mathbf{0}, \\ 0 & \text{otherwise.} \end{cases} \tag{1.1.6}$$

Depending on the context, Equation 1.1.5 may also be referred to as the Fokker-Planck equation[1]. The Dirac delta initial condition implies that the trajectory of the probability density evolves over time from an infinite point-mass with unit measure (note that strictly, as it is presented here, this is not a rigorous definition of the Dirac delta function) at $\mathbf{X}_s$ (since $\mathbf{X}_s$ is occupied with certainty) and subsequently *flows* over the state space in accordance with the differential terms in Equation 1.1.5. It thus follows that both the local and long-term dynamics of $\mathbf{X}_t$ are dictated by the drift vector $\boldsymbol{\mu}(\mathbf{X}_t, t)$ and diffusion matrix $\boldsymbol{\sigma}(\mathbf{X}_t, t)$. That is, the shape of the transitional density over the state space changes in accordance

[1] With reference to the naming conventions, the equations can be seen to appear in Kolmogoroff (1931), Planck (1917) and Fokker (1914), for example.

with the functional form of the drift and diffusion terms. Despite diffusion models often being constructed under simple functional forms for the drift and diffusion coefficients, finding analytical expressions that satisfy Equation 1.1.5 proves to be extremely difficult. Consequently, the primary difficulty with the analysis of diffusion models stems from not being able to express the dynamics of a model process probabilistically over finite transition horizons. Although it is well known that for sufficiently short time horizons the transitional density is distributed according to the multivariate Normal distribution

$$f(\mathbf{X}_t|\mathbf{X}_s) \approx \text{MVN}\Big(\mathbf{X}_s + \boldsymbol{\mu}(\mathbf{X}_s, s)(t-s), \boldsymbol{\sigma}(\mathbf{X}_s, s)\boldsymbol{\sigma}(\mathbf{X}_s, s)'(t-s)\Big), \quad (1.1.7)$$

where $\text{MVN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the multivariate Normal distribution with location vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, the shape of the transitional density over larger transition horizons may differ vastly from that of the Normal distribution. For example, consider a non-linear diffusion with dynamics given by the SDE:

$$\begin{aligned} dX_t &= X_t(1 - X_t^2)dt + 0.5dB_t^{(1)} \\ dY_t &= Y_t(1 - Y_t^2)dt + 0.5dB_t^{(2)}. \end{aligned} \quad (1.1.8)$$

Figure 1.1.1 illustrates the evolution of the transition density of Equation 1.1.8 over time, evaluated using a numerical techniques that will be outlined later in the present chapter. Clearly, although the initial shape of the transitional density is approximately Normal, as time progresses the shape of the transitional density evolves into a multimodal surface, reflecting the non-linear nature of the underlying model.

Indeed, the transitional density forms an integral part of numerous forms of analysis. In the context of modelling time series, it is useful to formulate a diffusion model by specifying the coefficients $\boldsymbol{\mu}(\mathbf{X}_t, t)$ and $\boldsymbol{\sigma}(\mathbf{X}_t, t)$ in terms of a set of parameters $\boldsymbol{\theta}$. Assuming that the continuous process $\{\mathbf{X}_t : t \geq 0\}$ is observed only at discrete epochs, given by the ordered set $\mathbf{T}_S = \{t_n : n = 1, ..., N\}$ resulting in the set of observations $\mathbf{D}_S = \{\mathbf{X}_{t_n} : n = 1, \ldots, N\}$, the likelihood can be evaluated from the transition density by way of the Markov property [2]:

$$L(\boldsymbol{\theta}|\mathbf{D}_S) \propto \prod_{n=1}^{N-1} f(\mathbf{X}_{t_{n+1}}|\mathbf{X}_{t_n}). \quad (1.1.9)$$

where $\boldsymbol{\theta}$ denotes a parameter vector on which the process depends.

[2] Note that we follow the convention that the initial distribution has negligible effect on the evaluation of the likelihood, ergo the proportional relation as opposed to exact relation (Aït-Sahalia, 2008).

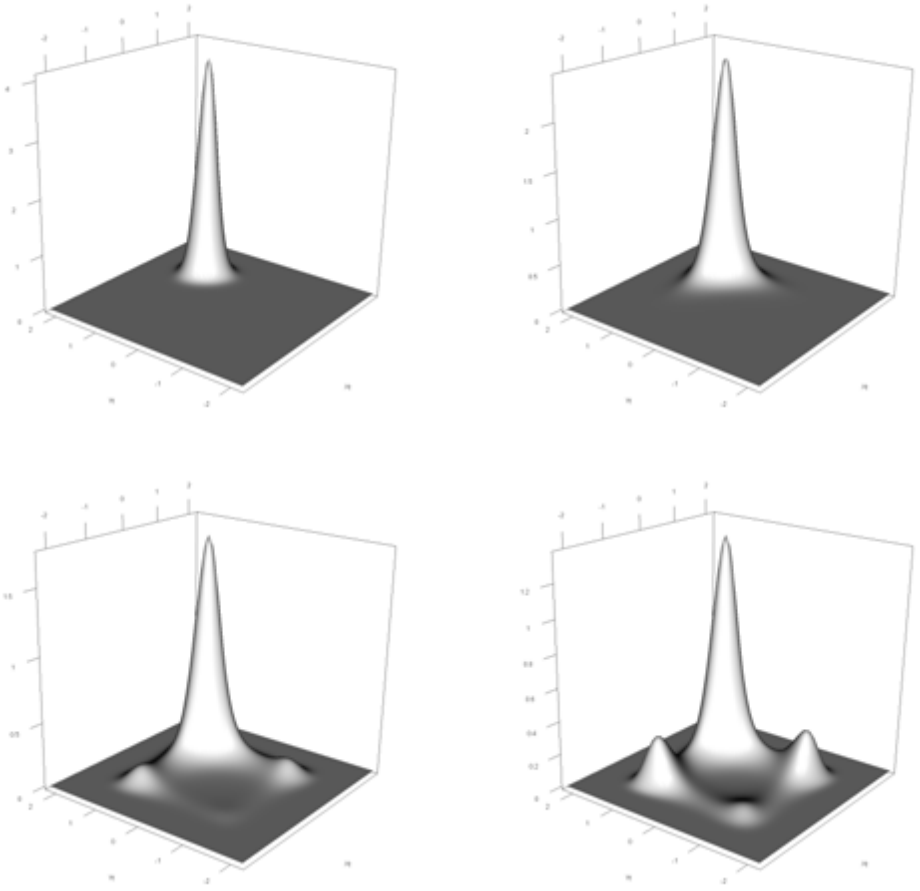


FIGURE 1.1.1: Evolution of the transition density of Equation 1.1.8 at times $t - s = \{0.25, 1, 5, 10\}$. Although the shape of the transition density is approximately Normal during the initial phases of the evolution, the cubic drift structure results in a transitional density with four distinct modes. R code: Supplementary materials, Section 1.1.

Unfortunately, since the transitional density is not available analytically in general, the likelihood function cannot always be evaluated directly using Equation 1.1.9 and one often has to resort to numerical techniques in order to evaluate the likelihood function. Fortunately, for diffusion processes, an alternative expression is available for the likelihood. Under the assumption that the process is observed continuously (i.e., at all points along its trajectory) one may apply the Radon-Nikodym derivative in order to arrive at Girsanov's formula:

$$\begin{aligned}\frac{d\mathbb{P}^{\theta}}{d\mathbb{Q}^{\theta}} &= L_t(\boldsymbol{\theta}|\mathbf{D}_S^C) \\ &= \exp\Bigg(\int_0^t \boldsymbol{\mu}(\mathbf{X}_u,u)'[\boldsymbol{\sigma}(\mathbf{X}_u,u)\boldsymbol{\sigma}(\mathbf{X}_u,u)']^{-1}\mathbf{d}\mathbf{X}_u \\ &\qquad -\frac{1}{2}\int_0^t \boldsymbol{\mu}(\mathbf{X}_u,u)'[\boldsymbol{\sigma}(\mathbf{X}_u)\boldsymbol{\sigma}(\mathbf{X}_u)']^{-1}\boldsymbol{\mu}(\mathbf{X}_u,u)du\Bigg),\end{aligned} \tag{1.1.10}$$

where $\mathbf{D}_s^C = \{\mathbf{X}_t : t \geq 0\}$ denotes the uncountably infinite set of continuously observed data points (in contrast to the discrete set $\mathbf{D}_S$), $\mathbb{P}^{\theta}$ is the law of $\mathbf{X}_t$ for $\boldsymbol{\theta}$ and finally with respect to $\mathbb{Q}^{\theta}$, $\mathbf{X}_t$ is the drift-free diffusion $d\mathbf{X}_t = \boldsymbol{\sigma}(\mathbf{X}_t,t)\mathbf{d}\tilde{\mathbf{B}}_t$ where $\tilde{\mathbf{B}}_t$ is a Brownian motion with respect to $\mathbb{Q}^{\theta}$ (see, e.g., Kloeden *et al.*, 1996). As with SDEs, analytical solutions to the stochastic integrals required to evaluate Equation 1.1.10 are difficult to find. Moreover, Equation 1.1.10 is a continuous time representation of the likelihood, thus being disparate with the notion of discrete observation. Despite this, Kloeden *et al.* (1996) derived estimators of the parameter vector $\boldsymbol{\theta}$ by the analytic maximisation of Equation 1.1.10, thereafter evaluating discrete approximations of the stochastic integrals appearing in the parameter estimators at the observed data points. For example, for the scalar diffusion process:

$$dX_t = cX_t^3dt + dB_t, \tag{1.1.11}$$

a maximum likelihood estimator of the parameter c can be derived as:

$$\begin{aligned}\hat{c} &= \frac{\int_s^t X_t^3 dX_t}{\int_s^t X_t^6 dt} \\ &\approx \frac{\sum_n X_{t_n}^3(X_{t_{n+1}} - X_{t_n})}{\sum_n X_{t_n}^6(t_{n+1}-t_n)}.\end{aligned} \tag{1.1.12}$$

Note that the last step in Equation 1.1.12 applies a low-order Wagner-Platen expansion (i.e., a discrete approximation) to the stochastic integrals contained in the true estimator. The estimator may thus be improved by increasing the order of the expansion so as to more accurately approximate the stochastic integrals in the estimator. However, since the quality of such discrete approximations deteriorates quickly for large time-lapses between observations $(t_{n+1} - t_n)$, the

estimator may be inherently biased unless the data is of adequate resolution (Pedersen, 1995). In order to surmount this resolution threshold one may resort to data-imputation in order to artificially improve the data resolution, thus rendering the short-time approximations valid. The idea of developing data-imputation schemes for discretely observed diffusion processes can be traced to a number of authors, with perhaps the most popular paradigms following from the work of Elerian *et al.* (2001), Eraker (2001), Roberts and Stramer (2001) and Chib *et al.* (2004) among others.

In the following section, the method of lines (a numerical technique for solving partial differential equations) is introduced and applied to the Kolmogorov forward equation. It is also shown how this method may be adapted to evaluate the marginal transition density of a particular element of a multivariate diffusion process. Subsequently, in Section 1.3 we demonstrate techniques that may be used in order to perform likelihood inference on discretely observed diffusions using equations 1.1.9 and 1.1.10. Throughout the analysis, we discuss various technical and practical pitfalls that arise within the methodology and subsequently develop strategies that aim to make the analysis feasible in real world applications. In Section 1.4, we demonstrate the methodology by way of an application to the modelling of population dynamics in an ecological setting. Finally, in Section 1.5 we outline the development of a software package which provides routines for implementing the techniques developed in this chapter.

1.2 Approximating the transitional density of a diffusion process

Calculating an analytical expression for the transitional density of a diffusion process can be extremely difficult and is, more often than not, impossible. As such, one often has to resort to numerical techniques in order to analyse the transitional density of a diffusion process. Since the transitional density is formulated in terms of a partial differential equation (PDE), given by the Kolmogorov equation, one may employ various techniques for calculating numerical solutions to PDEs in order to evaluate the transitional density. In this section, we detail one such numerical technique and proceed to derive results that become useful for approximating the transitional density over large transition horizons.

1.2.1 The method of lines

In order to evaluate the transitional density of a diffusion process, we may resort to direct numerical techniques in order to solve the Kolmogorov equation. By discretizing the spatial domain and applying finite difference approximations to the spatial derivatives $\frac{\partial}{\partial X_t^{(i)}}$ and $\frac{\partial^2}{\partial X_t^{(i)} \partial X_t^{(j)}}$ in Equation 1.1.5, we can formulate a system of ordinary differential equations that approximate the time evolution of the transition density at fixed points in the support of the transition density. For Equation 1.1.5, the scheme yields the system:

$$\begin{aligned} \frac{\partial f_{i_1,\ldots,i_k}(t)}{\partial t} = &- \sum_{r=1}^{k} \Delta_r^q [\mu_r(\mathcal{L}, t) \cdot \mathbf{f}(t)]_{i_1,\ldots,i_k} \\ &+ \frac{1}{2} \sum_{r=1}^{k} \sum_{s=1}^{k} \Delta_{r,s}^q [\gamma_{r,s}(\mathcal{L}, t) \cdot \mathbf{f}(t)]_{i_1,\ldots,i_k}, \end{aligned} \tag{1.2.1}$$

where $\mathcal{L}$ is an appropriately chosen k-dimensional lattice on a section of the domain $\mathcal{X}$ of the process, $f_{i_1,\ldots,i_k}(t)$ denotes the density at the $(i_1, \ldots, i_k)$-th lattice point at time t, and Δ_r^q and $\Delta_{r,s}^q$ are finite difference operators of order q with respect to the dimensions r and r, s respectively. Here $\mathbf{f}(t) = (f_{i_1,\ldots,i_k}(t))$ denotes the lattice in the form of a multidimensional array (e.g., in two dimensions it will be a matrix). This is the so-called method of lines (MOL) (see, e.g, Hamdi *et al.*, 2007): a computationally efficient alternative to the well known Crank-Nicolson method often used to solve partial differential equations numerically. The lattice $\mathcal{L}$ can be constructed by creating an $d_1 \times d_2 \times \ldots \times d_k$ array of coordinates with limits on each dimension defined by $\{[x_{min}^{(r)}, x_{max}^{(r)}] : r = 1, 2, \ldots k\}$. $\mathcal{L}$ thus represents regularly spaced points in the hypercube $[x_{min}^{(1)}, x_{max}^{(1)}] \times [x_{min}^{(2)}, x_{max}^{(2)}] \times \ldots \times [x_{min}^{(k)}, x_{max}^{(k)}]$ where the vertices of the lattice $\mathcal{L}$ are formed by d_r regularly spaced points along each segment $[x_{min}^{(r)}, x_{max}^{(r)}]$. The right-hand side of the system in Equation 1.2.1 is then defined by operation of a local finite difference approximation to the spatial derivatives in Equation 1.1.5. In order to solve the system in Equation 1.2.1, we require appropriate boundary conditions. In order to mimic the Dirac delta initial conditions we shall assume that the lattice points are chosen such that one of the vertices fall on the initial value $\mathbf{X}_s$ of each transition horizon. Let $(x_{i_1}^{(1)}, x_{i_2}^{(2)}, \ldots, x_{i_k}^{(k)})$ denote the $(i_1, i_2, \ldots, i_k)$-th vertex on the lattice $\mathcal{L}$ then we set

$$f_{i_1,\ldots,i_k}(s) = \begin{cases} \prod_{r=1}^{k} \frac{1}{(x_{i_r+1}^{(r)} - x_{i_r-1}^{(r)})/2} & \text{if } (x_{i_1}^{(1)}, x_{i_2}^{(2)}, \ldots, x_{i_k}^{(k)}) = \mathbf{X}_s, \\ 0 & \text{otherwise.} \end{cases} \tag{1.2.2}$$

By evaluating the system of ODEs implied by Equation 1.2.1 from time s up to t, $f_{i_1,\dots,i_k}(t)$ approximates the transitional density for the coordinates $(x_{i_1}^{(1)}, x_{i_2}^{(2)}, \dots, x_{i_k}^{(k)})$ at time t. Using the lattice points as a reference, we may subsequently evaluate the transitional density at any point within the limits of $\mathcal{L}$ by interpolating between the values $f_{i_1,\dots,i_k}(t)$. Consider for example Equation 1.1.8: By approximating the transitional density with a large system of differential equations, we may approximate the transitional density over the applicable transition horizon at fixed points on a square lattice of the state-space. From Figure 1.2.1, we can see how the various modes of the transitional density manifest over time as the local ODEs assume increasing values around the modes of the density. Subsequently, should we wish to evaluate the transitional density at states that do not fall on the pre-determined lattice, we may approximate the transitional density by interpolating from the ODEs at neighbouring nodes of the lattice.

Using this technique, we may calculate various useful quantities. For example, we may approximate the likelihood using Equation 1.1.9 by substituting $f(\mathbf{X}_{t_{n+1}}|\mathbf{X}_{t_n})$ with the numerical approximant evaluated at the coordinates $\mathbf{X}_{t_{n+1}}$. That is, set

$$L(\boldsymbol{\theta}|\mathbf{D}_S) \approx \prod_{i=1}^{N-1} \mathcal{I}_{\mathbf{X}_{t_{n+1}}}(\{f_{i_1,\dots,i_k}(t_{n+1})\}), \tag{1.2.3}$$

where $\mathcal{I}_{\mathbf{u}}(Z)$ interpolates the coordinate $\mathbf{u}$ on the k-dimensional lattice Z. Here we take the notation $\{f_{i_1,\dots,i_k}(t_{n+1})\}$ to mean that the array is populated at each $i_1, \dots, i_k$-node by the approximate transition density evaluated at time t_{n+1} subject to the initial condition $\mathbf{X}_{t_i}$ i.e., $f_{i_1,\dots,i_k}(t_i)$ is calculated according to Equation 1.2.2.

Naturally, the quality of the approximation under the method of lines depends on the various constituents of the algorithm. Indeed, the system of ODEs used to approximate the PDE are determined directly by the specification of the finite difference operators applied to the spatial derivatives in the Kolmogorov equation. Consequently, the numerical properties of the scheme may vary in accordance with the order and specification (forward, backwards, and central difference) finite difference approximation being used. For purposes of applying the method of lines to the Kolmogorov equation, it suffices to use first order central difference operators (see Appendix A.1). In addition to setting up the appropriate system of ODEs, the algorithm requires a method for solving said system of ODEs. This involves using numerical techniques for solving ordinary differential equations. For these purposes, we approximate a solution to the transitional density by using a high order Runge-Kutta method with a sufficiently fine time-step. Although the fine time step is typically required to ensure that the system of ODEs is

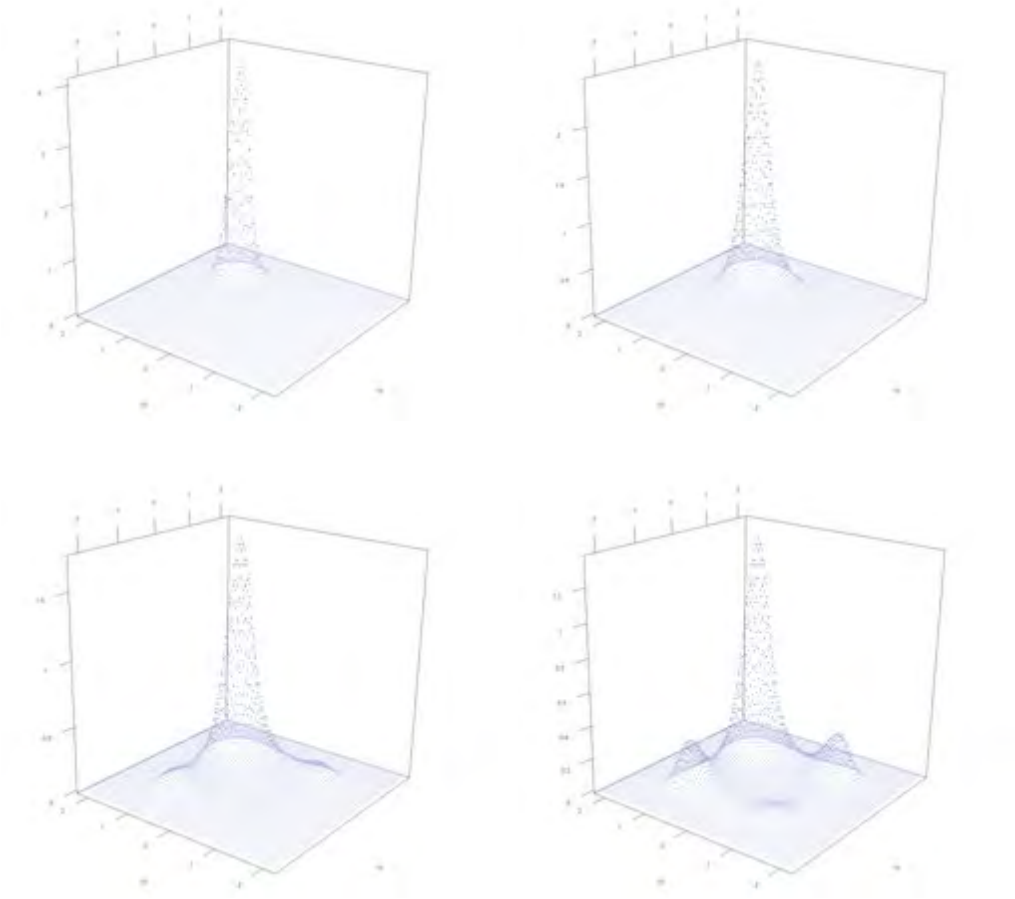


FIGURE 1.2.1: Transition density of Equation 1.1.8 at times $t - s = \{0.25, 1, 5, 10\}$ expressed on an equispaced, finite lattice of the state space. Each point represents the approximate transition density value calculated at that node using the method of lines. Points are coloured according to their magnitude (z-axis value). R code: Supplementary materials, Section 1.2.

approximated accurately, in this context it is also important to ensure that the numerical solution of the system of ODEs implied by the discretization scheme remains stable. Instability is caused by the ratio between the time step in the numerical method used to solve the ODEs and the grid spacings $x_{i+1}^{(1)} - x_i^{(1)}$ and $x_{j+1}^{(2)} - x_j^{(2)}$, being too large. This is discussed further in Appendix A.2.

1.2.2 Hybrid PDEs and the marginal Kolmogorov equation

When analysing the transitional densities of multivariate diffusion systems, one is tasked with the difficult problem of calculating the solution of a multivariate partial differential equation with singular boundary conditions. As the dimensions of the process increase, calculating the transitional density using numerical techniques becomes increasingly difficult. Indeed, the so-called 'curse of dimensionality' has a severe effect on the performance of algorithms such as the method of lines. For example, applying finite-difference schemes of Equation 1.2.1 requires that the transitional density be approximated at $\prod_{i=1}^{k} d_i$ nodes on a lattice of the state-space, where d_i is the number of abscissae in each dimension and k is the dimension of the diffusion process. Consequently, the computational efficiency of the method of lines decreases exponentially as the number of dimensions of the process of interest increases. Depending on the context, however, one is often only interested in the probabilistic evolution of a single dimension of a process. For these purposes, it can be useful to analyse any one of the marginal transition densities of a multivariate diffusion process. That is, given a k-dimensional diffusion process, $\mathbf{X}_t$, we wish to find the transitional density of say the r-th coordinate, $X_t^{(r)}$. Should an analytical expression for the transitional density exist, this could be achieved by integrating over the residual dimensions of the process in order to arrive at an expression for the marginal transition density. When the transitional density is not available analytically, we can apply the same principle and solve the transitional density numerically in multiple dimensions and numerically integrate out the residual dimensions of the density function. Citing the aforementioned curse of dimensionality, this would be extremely costly from a computational perspective. Thus, it would be useful to derive an expression that governs the evolution of the marginal transitional density. Subsequently, we can treat the marginal transitional density in a similar fashion to a scalar diffusion process. Fortunately, this can be achieved by way of carefully manipulating the Kolmogorov equation.

Formally, the transitional density $g(X_t^{(r)}|\mathbf{X}_s)$ of the r-th marginal coordinate, $X_t^{(r)}$, of the vector process $\mathbf{X}_t$, starting in state $\mathbf{X}_s$ at time $s < t$ is governed by

the equation:

$$\begin{aligned}\frac{\partial}{\partial t}g(X_t^{(r)}|\mathbf{X}_s) = &-\frac{\partial}{\partial X_t^{(r)}}\Big[E_{\mathbf{Y}_t|X_t^{(r)}}\big(\mu_r(X_t^{(r)},\mathbf{Y}_t,t)\big)g(X_t^{(r)}|\mathbf{X}_s)\Big]\\ &+\frac{1}{2}\frac{\partial}{\partial^2(X_t^{(r)})^2}\Big[E_{\mathbf{Y}_t|X_t^{(r)}}\big(\gamma_{rr}(X_t^{(r)},\mathbf{Y}_t,t)\big)g(X_t^{(r)}|\mathbf{X}_s)\Big],\end{aligned} \tag{1.2.4}$$

with the boundary condition $g(x_s|\mathbf{X}_s) = \delta(x_s - X_s^{(r)})$, where $\delta(.)$ is the Dirac delta function, $\mathbf{Y}_t = \{X_t^{(1)}, X_t^{(2)}, \ldots, X_t^{(k)}\}' \setminus X_t^{(r)}$, and $E_{\mathbf{Y}_t|X_t^{(r)}}$ denotes the expectation of $\mathbf{Y}_t$ conditional on $X_t^{(r)}$.

To see how this result arises, consider the multivariate Kolmogorov equation:

$$\begin{aligned}\frac{\partial f(\mathbf{X}_t|\mathbf{X}_s)}{\partial t} = &-\sum_{i=1}^{k}\frac{\partial}{\partial X_t^{(i)}}\Big[\mu_i(\mathbf{X}_t,t)f(\mathbf{X}_t|\mathbf{X}_s)\Big]\\ &+\frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{\partial^2}{\partial X_t^{(i)}\partial X_t^{(j)}}\Big[\boldsymbol{\gamma}_{ij}(\mathbf{X}_t,t)f(\mathbf{X}_t|\mathbf{X}_s)\Big],\end{aligned} \tag{1.2.5}$$

First, we integrate over the residual coordinates $\mathbf{Y}_t$:

$$\begin{aligned}&\frac{\partial}{\partial t}\int f(\mathbf{X}_t|\mathbf{X}_s)d\mathbf{Y}_t =\\ &-\int\sum_{i=1}^{k}\frac{\partial}{\partial X_t^{(i)}}\Big[\mu_i(\mathbf{X}_t,t)f(\mathbf{X}_t|\mathbf{X}_s)\Big]d\mathbf{Y}_t\\ &+\int\frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{\partial^2}{\partial X_t^{(i)}\partial X_t^{(j)}}\Big[\gamma_{ij}(\mathbf{X}_t,t)f(\mathbf{X}_t|\mathbf{X}_s)\Big]d\mathbf{Y}_t.\end{aligned} \tag{1.2.6}$$

Note that here, $d\mathbf{Y}_t$ does not represent a process but an integration variable – the nomenclature is preserved simply for ease of notation. Subsequently, by conditioning on the marginal we have:

$$\begin{aligned}&\frac{\partial}{\partial t}\int f(\mathbf{Y}_t|\mathbf{X}_s,X_t^{(r)})g(X_t^{(r)}|\mathbf{X}_s)d\mathbf{Y}_t =\\ &-\int\sum_{i=1}^{k}\frac{\partial}{\partial X_t^{(i)}}\Big[\mu_i(\mathbf{X}_t,t)f(\mathbf{Y}_t|\mathbf{X}_s,X_t^{(r)})g(X_t^{(r)}|\mathbf{X}_s)\Big]d\mathbf{Y}_t\\ &+\int\frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{\partial^2}{\partial X_t^{(i)}\partial X_t^{(j)}}\Big[\gamma_{ij}(\mathbf{X}_t,t)f(\mathbf{Y}_t|\mathbf{X}_s,X_t^{(r)})g(X_t^{(r)}|\mathbf{X}_s)\Big]d\mathbf{Y}_t\end{aligned} \tag{1.2.7}$$

Finally, by rearranging the differential and integral operators we may derive:

$$\begin{aligned}
&\frac{\partial}{\partial t} g(X_t^{(r)}|\mathbf{X}_s) = \\
&- \frac{\partial}{\partial X_t^{(r)}} \left[\int \mu_r(X_t^{(r)}, \mathbf{Y}_t, t) f(\mathbf{Y}_t|\mathbf{X}_s, X_t^{(r)}) d\mathbf{Y}_t g(X_t^{(r)}|\mathbf{X}_s) \right] \\
&+ \frac{1}{2} \frac{\partial^2}{\partial (X_t^{(r)})^2} \left[\int \gamma_{rr}(X_t^{(r)}, \mathbf{Y}_t, t) f(\mathbf{Y}_t|\mathbf{X}_s, X_t^{(r)}) d\mathbf{Y}_t g(X_t^{(r)}|\mathbf{X}_s) \right] \\
&- \sum_{i \neq r} \frac{\partial}{\partial X_t^{(i)}} \left[\int \mu_i(X_t^{(r)}, \mathbf{Y}_t, t) f(\mathbf{Y}_t|\mathbf{X}_s, X_t^{(r)}) d\mathbf{Y}_t g(X_t^{(r)}|\mathbf{X}_s) \right] \\
&+ \frac{1}{2} \sum_{i \neq r} \sum_{j \neq r} \frac{\partial^2}{\partial X_t^{(i)} \partial X_t^{(j)}} \left[\int \boldsymbol{\gamma}_{ij}(X_t^{(r)}, \mathbf{Y}_t, t) f(\mathbf{Y}_t|\mathbf{X}_s, X_t^{(r)}) d\mathbf{Y}_t g(X_t^{(r)}|\mathbf{X}_s) \right],
\end{aligned} \tag{1.2.8}$$

which leads to the result:

$$\begin{aligned}
\frac{\partial}{\partial t} g(X_t^{(r)}|\mathbf{X}_s) = &\frac{\partial}{\partial X_t^{(r)}} \left[E_{\mathbf{Y}_t|X_t^{(r)}} \big(\mu_r(X_t^{(r)}, \mathbf{Y}_t, t) \big) g(X_t^{(r)}|\mathbf{X}_s) \right] \\
&+ \frac{1}{2} \frac{\partial^2}{\partial (X_t^{(r)})^2} \left[E_{\mathbf{Y}_t|X_t^{(r)}} \big(\gamma_{rr}(X_t^{(r)}, \mathbf{Y}_t, t) \big) g(X_t^{(r)}|\mathbf{X}_s) \right] \\
&+ 0 + 0.
\end{aligned} \tag{1.2.9}$$

Equation 1.2.4 thus constitutes a hybrid PDE which consists of a partial differential equation that depends on the evolution of the moments of the residual coordinates conditioned on the marginal dimension. That is, if we can evaluate the conditional moments of elements of the drift and diffusion of the coordinate of interest, we can evaluate the marginal density of that coordinate using a PDE with only one spatial dimension. Consequently, using numerical techniques such as the method of lines, we may circumvent a great deal of computational overhead by evaluating Equation 1.2.4 as opposed to the full transitional density and subsequently integrating out the residual dimensions.

1.3 Multivariate data-imputation

1.3.1 A Brownian bridge data-imputation scheme

Roberts and Stramer (2001) introduce a novel Bayesian data-imputation scheme for performing inference on non-linear scalar diffusion models for discretely observed processes (see also Dellaportas *et al.* (2006) and Beskos *et al.* (2009)).

Later the methodology was extended for a class of multivariate diffusion models by Kalogeropoulos *et al.* (2011). In the present context, the method is appealing for a number of reasons: It is applicable for a wide class of diffusion models, it is computationally efficient and robust to long transition horizons between data points, and completely circumvents the need to evaluate the transitional density. The intuition of the scheme is to induce near continuous simulated sample paths that pass through existing data points in order to exploit Girsanov's formula for the likelihood. These paths are constructed using discretely sampled Brownian bridges, tethered to successive data points. Let B_t be standard Brownian motion, then the Brownian bridge b_t, is defined as:

$$b_t = B_t - tB_1, \quad t \in [0,1], B_0 = 0. \tag{1.3.1}$$

The trajectory resulting from Equation 1.3.1, with endpoints tethered to the coordinates $(0,0)$ and $(1,0)$, can then be mapped onto arbitrary points in the space-time domain, for example the coordinates (s,A) and (t,B) for $t > s$, using a linear transform. Using this, the scheme attempts to artificially enhance the data resolution by simulating Brownian bridges that are tethered to transformed values of the data points as illustrated in Figure 1.3.1, and in so doing makes it possible to evaluate the stochastic integrals in Equation 1.1.10 via quadrature. However, for a k-dimensional system of SDEs, in order to facilitate augmentation, Girsanov's formula is modified by factorizing the dominating measure $\mathbb{Q}^\theta$ with respect to a k-dimensional Lebesgue measure $\text{Leb}_k(.)$:

$$\frac{d\mathbb{P}^\theta}{d\mathbb{Q}^\theta} = \frac{d\mathbb{P}^\theta}{d\big(\mathbb{Q}^\theta_{\mathbf{X}} \times \text{Leb}_k(\mathbf{X})\big) \times h(\mathbf{X}, \boldsymbol{\theta})}. \tag{1.3.2}$$

where $h(\mathbf{X}, \boldsymbol{\theta})$ is the marginal density of $\mathbf{X}_t$ w.r.t. the Lebesgue measure and $\mathbb{Q}^\theta_{\mathbf{X}}$ is $\mathbb{Q}^\theta$ conditioned on the observations $\mathbf{X}$ (Kalogeropoulos *et al.*, 2011). The goal of this factorization is to separate the contribution to the likelihood of the simulated sample paths and the observed trajectory. For example, $\mathbb{Q}^\theta_{\mathbf{X}}$ here represents the distribution of the simulated paths between successive observations whilst $h(\mathbf{X}, \boldsymbol{\theta})$ denotes the distribution of the actual observations. Unfortunately, since the change of measure that gives rise to Girsanov's formula depends on the parameters of the diffusion model, the factorization is not valid when the diffusion matrix of the model in question depends on the parameter vector $\boldsymbol{\theta}$. As such, Kalogeropoulos *et al.* (2011) advocate the use of the Lamperti transform (Aït-Sahalia, 2008) in order to transform the model process (under the assumption that the model is indeed reducible) into one of unit diffusion:

$$\mathbf{Y}_t = \Gamma(\mathbf{X}_t) = \int^{\mathbf{X}_t} \boldsymbol{\sigma}(\mathbf{u})^{-1} d\mathbf{u}. \tag{1.3.3}$$

It is worth noting that reducibility plays a key role in the implementation of the data-imputation algorithm. Specifically, in the context of multivariate diffusions, this places constraints on the specification of the diffusion matrix. Consequently, when using the data-imputation algorithm, care needs to be taken to ensure that the model specification is indeed valid under the methodology. Aït-Sahalia (2008) develops a necessary and sufficient condition for the reducibility of a multivariate diffusion system.

After performing the Lamperti transform, the likelihood can then be evaluated as:

$$\frac{d\mathbb{P}^{\theta}}{d\{\mathbb{Q}^{\theta}_{\mathbf{X}} \times Leb_k(\mathbf{X})\}} = L^{\Gamma}_t(\boldsymbol{\theta}|\mathbf{Z},\mathbf{X}) \times \prod_{n=1}^{N-1} \mathrm{MVN}_k(\Gamma(\mathbf{X}_{t_{n+1}}) - \Gamma(\mathbf{X}_{t_n}), \mathbf{0}, (t_{n+1}-t_n)I_{k\times k})J(\mathbf{X}_{t_n}), \tag{1.3.4}$$

Here $\mathrm{MVN}_k(\mathbf{x}, \mathbf{0}, \boldsymbol{\Sigma})$ denotes the zero-mean multivariate normal distribution of dimension k with covariance matrix $\boldsymbol{\Sigma}$ evaluated at $\mathbf{x}$, $I_{k\times k}$ denotes the $k \times k$ identity matrix, $J(\mathbf{X}_t) = |\boldsymbol{\Sigma}^{-1}(\mathbf{X}_t)|$ denotes the Jacobian of the transformation, and $\mathbf{Z}$ denotes a set of paths imputed on $\mathbf{X}$. The superscript Γ is to indicate that $L^{\Gamma}_t(\boldsymbol{\theta}|\mathbf{Z},\mathbf{X})$ is Girsanov's formula evaluated under the transformed process. A consequence of the Lamperti transform is its effect on the drift vector of the target diffusion model. Itô's lemma for multivariate SDEs gives the drift under the reduced diffusion process as:

$$\mu^{\Gamma}_r(\mathbf{X}_t, t) = \sum_{i=1}^{k} \mu_i(\mathbf{X}_t, t)\frac{\partial \Gamma_r(\mathbf{X}_t)}{\partial X^{(i)}_t} + \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k} \boldsymbol{\gamma}_{ij}(\mathbf{X}_t)\frac{\partial^2 \Gamma_r(\mathbf{X}_t)}{\partial X^{(i)}_t \partial X^{(j)}_t} \tag{1.3.5}$$

for $r \in \{1, 2, \ldots, k\}$, where again we use the superscript Γ to distinguish the drift under the transformed diffusion from that of the original diffusion model. As per Kalogeropoulos *et al.* (2011) $\mathbf{X}_t$ may be replaced with the inverse transform in order to write the diffusion in terms of the transformed values $\mathbf{Y}_t$. However, for notational purposes, we preserve writing the process in terms of $\mathbf{X}_t$. Although the former is arguably preferable under most circumstances, we find it easier to convey the methodology in terms of the original variables. Indeed, mathematically it makes no difference whether one works in terms of the transformed or untransformed diffusion so long as one keeps track of the effect of the transformation since it depends on the parameter vector.

In order to augment the observed data $\mathbf{D}_S = \{X_{t_n} : n = 1, ..., N\}$ with artificial sample paths which are generated using Brownian bridges, we are required to simulate trajectories for Equation 1.3.1 discretely at m interim points between

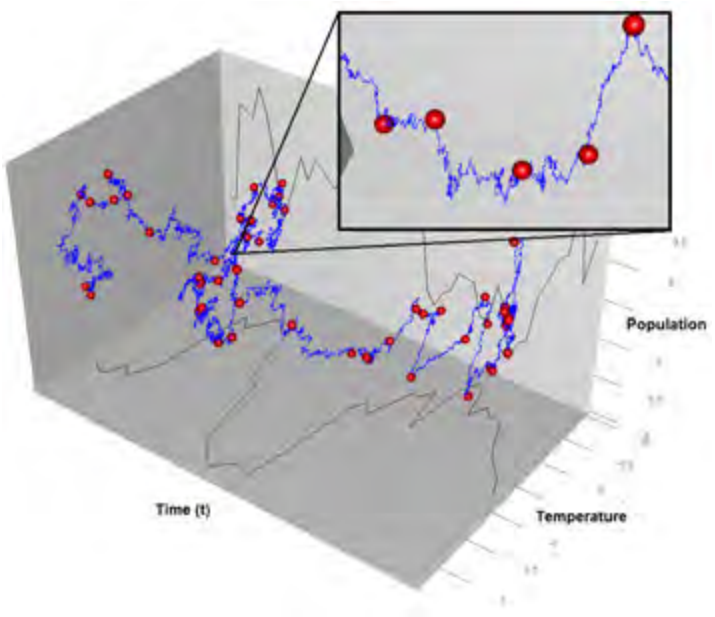


FIGURE 1.3.1: Imputed Brownian bridges (blue) over a subset of the dataset to be analysed later (red spheres). Individual time series are drawn on their respective planes (black).

successive data coordinates under the Lamperti transform. Let $P^{(m)}_{t_n,t_{n+1}}(\mathbf{y})$ denote a single simulation of a Brownian bridge with endpoints tethered to the coordinates (t_n, y_{t_n}) and $(t_{n+1}, y_{t_{n+1}})$. Furthermore, let $\mathcal{Q}^{[m]}_S = \{\mathbf{Q}_{\tau_r} : r = 1, 2, \ldots, (N-1)(m+1)+1\}$ denote a set of imputed data at time points $t \in \boldsymbol{\tau} = \{\tau_r : r = 1, \ldots, (N-1)(m+1)+1, \tau_1 = t_1, \tau_{m+2} = t_2, \ldots\}$. Then $\mathcal{Q}^{[m]}_S$ can be constructed as:

$$\mathcal{Q}^{[m]}_S = \Gamma^{-1}(U^{[m]}(\Gamma(\mathbf{D}_S)), \tag{1.3.6}$$

where $U^{[m]}(A)$ defines an imputation function that has the action of simulating a k-dimensional Brownian bridge over each successive pair of coordinates in A. That is, for $A = \{\mathbf{y}_n : n = 1, 2, \ldots, N\}$, $U^{[m]}(\mathbf{y}_n) = P^{(m)}_{t_n,t_{n+1}}(\mathbf{y}_n)$, $U^{[m]}(\{\mathbf{y}_n, \mathbf{y}_{n+1}\}) = P^{(m)}_{t_n,t_{n+1}}(\mathbf{y}_n) \cup P^{(m)}_{t_{n+1},t_{n+2}}(\mathbf{y}_{n+1})$ and so on. Note that, since we introduce m additional values for each pair of successive observations, the revised set of time-indices for $\mathbf{Q}_t$ is given by $t \in \boldsymbol{\tau} = \{\tau_r : r = 1, \ldots, (N-1)(m+1)+1, \tau_1 = t_1, \tau_{m+2} = t_2, \ldots\}$. Note also that the unit diffusion bridges $U^{[m]}(\Gamma(\mathbf{D}_S))$ are back-transformed to give sample paths in terms of the untransformed variables $\mathbf{X}_t$. Were one to write the algorithm in terms of the transformed diffusion, this inversion would occur when the drift of the revised diffusion is expressed in terms of the transformed variable $\mathbf{Y}_t = \Gamma(\mathbf{X}_t)$.

When m is large enough, sample paths of the augmented data become near continuous on $[t_1, t_N]$ and we may use a discrete proxy to the stochastic integrals in Equation 1.1.10. Since quadrature rules for stochastic integrals do not behave the same as their deterministic counterparts, we use a Euler scheme for the integrals in Equation 1.1.10, as it follows from the strong order 0.5 Wagner-Platen expansion (Kloeden *et al.*, 1996). Although the resulting integration kernel is identical to that of the left end point Riemann sum used in deterministic calculus, it is important to note that the direction of time plays an important role in the behaviour of quadrature rules in stochastic calculus. For example, using a right end point Riemann sum will result in estimates identical in absolute value to those produced by the left point scheme, but negative where the later is positive and *vice versa*.

By imputing multiple Brownian bridges on discretely observed data points, one may evaluate the likelihood of a given diffusion model using Girsanov's formula. Since evaluating the likelihood in this way requires simulation of the imputed paths, inferring model parameters requires wrapping the algorithm in a Markov chain Monte Carlo (MCMC) procedure. We formalise the data-imputation procedure within the Metropolis-Hastings algorithm as follows:

1. Initialize the parameter set $\boldsymbol{\theta}^{old} = \boldsymbol{\theta}^{start}$. Initialize the imputed paths $\left\{\mathcal{Q}_S^{[m]}, \boldsymbol{\theta}^{old}\right\}^{old} = \Gamma^{-1}(U^{[m]}(\{\Gamma(\mathbf{X}_{t_n}) : n = 1, \ldots, N\}))$. The notation $\ldots \mathcal{Q}_S^{[m]}, \boldsymbol{\theta}^{old} \ldots$ signifies that the the initial set of imputed trajectories depend on the parameter vector $\boldsymbol{\theta}^{old}$, whilst the enclosure $\left\{\ldots\right\}^{old}$ indicates which string of Brownian bridges the transformation is applied to, e.g. $\left\{\mathcal{Q}_S^{[m]}, \boldsymbol{\theta}^{a}\right\}^{old}$ and $\left\{\mathcal{Q}_S^{[m]}, \boldsymbol{\theta}^{b}\right\}^{old}$ use the same string of simulated Brownian motions but are transformed under different parameter vectors.

2. Update the drift and diffusion parameters of the process as follows: Propose a new state for the parameter vector $\boldsymbol{\theta}$ by drawing from a suitable proposal distribution, $q(\boldsymbol{\theta}^{new}|\boldsymbol{\theta}^{old})$. Then set $\boldsymbol{\theta}^{old} = \boldsymbol{\theta}^{new}$ with probability:

$$\min\left(\frac{L^{\star}(\boldsymbol{\theta}^{new}, \mathbf{D}_S, \left\{\mathcal{Q}_S^{[m]}, \boldsymbol{\theta}^{new}\right\}^{old})}{L^{\star}(\boldsymbol{\theta}^{old}, \mathbf{D}_S, \left\{\mathcal{Q}_S^{[m]}, \boldsymbol{\theta}^{old}\right\}^{old})} \frac{q(\boldsymbol{\theta}^{old}|\boldsymbol{\theta}^{new})}{q(\boldsymbol{\theta}^{new}|\boldsymbol{\theta}^{old})} \frac{\pi(\boldsymbol{\theta}^{new})}{\pi(\boldsymbol{\theta}^{old})}, 1\right), \quad (1.3.7)$$

where $\pi(\boldsymbol{\theta})$ denotes a prior density on the parameter vector and from equations 1.1.10 and 1.3.2 it follows that the likelihood is approximated by:

$$L^{\star}(\boldsymbol{\theta}, \mathbf{D}_S, \mathcal{Q}_S^{[m]}) = \\ \exp\Bigg(\sum_{r=1}^{(N-1)(m+1)} \boldsymbol{\mu}^{\Gamma}(\mathbf{Q}_{\tau_r}, \tau_r)' \Delta\boldsymbol{\Gamma}(\mathbf{Q}_{\tau_r}) - \frac{1}{2} \sum_{r=1}^{(N-1)(m+1)} \boldsymbol{\mu}^{\Gamma}(\mathbf{Q}_{\tau_r}, \tau_r)' \mu^{\Gamma}(\mathbf{Q}_{\tau_r}, \tau_r) \delta_r \Bigg) \\ \times \prod_{n=1}^{N-1} \Big[\mathrm{MVN}_k(\Delta\Gamma(\mathbf{X}_{t_n}), \mathbf{0}_{k\times 1}, (t_{n+1} - t_n) I_{k\times k}) \times J(\mathbf{X}_{t_n}) \Big]. \tag{1.3.8}$$

where $\delta_r = \tau_{r+1} - \tau_r$ and Δ is the standard first difference operator, i.e., $\Delta\Gamma(\mathbf{X}_{t_n}) = \Gamma(\mathbf{X}_{t_{n+1}}) - \Gamma(\mathbf{X}_{t_n})$. Note again that $\big\{\mathcal{Q}_S^{[m]}, \boldsymbol{\theta}^{new}\big\}^{old}$ is to indicate that the simulated bridges are the same as those of $\big\{\mathcal{Q}_S^{[m]}, \boldsymbol{\theta}^{old}\big\}^{old}$, but the transformation $\Gamma(.)$ takes place under the parameters $\boldsymbol{\theta}^{new}$ in the former. Also, the superscript $\star$ is used to distinguish the quantity $L^{\star}(.)$ from that of 'the likelihood' in the formal sense that the acceptance probability manifests as a ratio of the quantities in Equation 1.3.4 (after accounting for the reduction to unit diffusion) where the measure $\mathbb{Q}^{\theta}$ depends on the parameters of the process. However, this is a somewhat technical matter which is more of conceptual consequence rather than a practical issue, so for brevity, we shall still refer to the quantity $L^{\star}(.)$ as 'the likelihood' for the remainder of this chapter.

3. Update the imputed paths by sampling a proposal string of Brownian bridges $\big\{\mathcal{Q}_S^{[m]}, \boldsymbol{\theta}^{old}\big\}^{new}$ (here $\boldsymbol{\theta}^{old}$ is the updated parameter vector): By the Markov property, we may update imputed paths by breaking up the discretized integrals in the likelihood into non-overlapping intervals delimited by the observations $\mathbf{D}_S$. That is, the likelihood may be broken up into its partial sums $\{l_n(\boldsymbol{\theta}, \mathbf{D}_S, \mathcal{Q}_S^{[m]}) : n = 1, \ldots, N-1\}$ where

$$l_n(\boldsymbol{\theta}, \mathbf{D}_S, \mathcal{Q}_S^{[m]}) = \\ \exp\Bigg(\sum_{r=n\times m+1}^{(n+1)\times m} \boldsymbol{\mu}^{\Gamma}(\mathbf{Q}_{\tau_r}, \tau_r)' \Delta\boldsymbol{\Gamma}(\mathbf{Q}_{\tau_r}) - \frac{1}{2} \sum_{r=n\times m+1}^{(n+1)\times m} \boldsymbol{\mu}^{\Gamma}(\mathbf{Q}_{\tau_r}, \tau_r)' \boldsymbol{\mu}^{\Gamma}(\mathbf{Q}_{\tau_r}, \tau_r) \delta_r \Bigg). \tag{1.3.9}$$

Note the product term in the likelihood of step 2 falls away since it does not depend on the imputed paths $\mathbf{Q}_t$. One may update individual bridges by accepting each section of the proposed string of Brownian bridges with the set with probability:

$$\min\left(\frac{l_n(\boldsymbol{\theta}, \mathbf{D}_S, \big\{\mathcal{Q}_S^{[m]}, \boldsymbol{\theta}^{old}\big\}^{new})}{l_n(\boldsymbol{\theta}, \mathbf{D}_S, \big\{\mathcal{Q}_S^{[m]}, \boldsymbol{\theta}^{old}\big\}^{old})}, 1 \right). \tag{1.3.10}$$

Note that the values of $\boldsymbol{\theta}^{old}$ used here are from the outcome of step 2. That is, in this step the parameters used in the transformation remain the same whilst the simulated bridges assume differing trajectories.

4. Repeat from step 2 until a satisfactory number of $\boldsymbol{\theta}$ have been drawn.

A final point of consideration with regard to the algorithm is the simulation technique used in step 3: Any simulation scheme for $U^{[m]}(.)$ should suffice, however for large m and k the computational efficiency of standard algorithms may be compromised. Indeed, with respect to computational efficiency, this step will account for the bulk of the computational overhead. As such, it is recommended that a fast simulation scheme is used during the data augmentation stage of the algorithm. In the section that follows, we develop a vectorized Brownian bridge simulation scheme, tailored to the algorithm and designed to maximise the efficiency with which updates of MCMC procedure can be carried out by simultaneously simulating of all $N-1$ Brownian bridges as a string of Brownian bridges passing through fixed points. This, in turn, allows one to use the partial sums in step 3 in order to update all individual sections of the string whilst leaving the imputation algorithm itself unaltered.

1.3.2 Vectorized simulation of Brownian bridges.

A cornerstone of the data-imputation algorithm is the use of Brownian bridges to construct data strings tethered to the discretely observed data points which take the place of missing sample trajectories. In order to minimise the computational overhead of *in-turn* simulation of the Brownian bridges over successive data points we outline a vectorized algorithm for simulating coupled strings of Brownian bridges. By simulating one long string of coupled Brownian bridges, it is possible to improve the computational efficiency of the data-imputation procedure at the imputation step. By breaking up the likelihood into the respective contributions of each imputed data string and subsequently calculating a vector of acceptance probabilities using the string of Brownian bridges (as opposed to iterating through and updating each string individually) we may eliminate one level of looping from the algorithm and significantly improve the computational efficiency of the data-imputation procedure.

We outline the scheme as follows: Suppose we want to simulate $N-1$ Brownian bridges over time nodes $\mathbf{T}_S = \{t_n, n = 1, ..., N\}$ simultaneously, starting and ending at nodes, say $\mathbf{y} = \{y_{t_n} : i = 1, \ldots, N\}$. Let each each pair of consecutive time points $\{t_n, t_{n+1}\} \in \mathbf{T}_S$ be discretized into a sequence of m interim time points, $\boldsymbol{\omega}_{t_n} = \{t^*_{n,1} = t_n, t^*_{n,2} = t_n + (t_{n+1} - t_n)/(m+1), ..., t^*_{n,m+2} = t_{n+1}\}$.

$\boldsymbol{\omega}_{t_n}$ thus gives the time index of the n-th transition horizon. It is important to note that, although the bridges are simulated simultaneously, they are generated independently (apart from the overlapping start/end-points) on each transition horizon. For these purposes, we construct two time indexes. First, we set:

$$\boldsymbol{\tau} = \left\{ \bigcup_{n=1}^{N-1} \boldsymbol{\omega}_{t_n} \right\} \setminus \left\{ \bigcap_{n=1}^{N-1} \boldsymbol{\omega}_{t_n} \right\}, \tag{1.3.11}$$

where $\setminus$ is the set-minus operator, which corresponds to a sequence of m imputed points for all consecutive transition horizons. Then by collecting all $N-1$ sets of $\boldsymbol{\omega}_{t_n}$, we construct:

$$\boldsymbol{\tau}^* = \bigcup_{n=1}^{N-1} \boldsymbol{\omega}_{t_n}. \tag{1.3.12}$$

Thus we have $\tau_1^* = t_1, \tau_{m+2}^* = t_2$, and we restart from t_2 such that $\tau_{m+3}^* = t_2$, and so forth. Here, the first index corresponds to the appropriate time index for an imputed dataset, whilst the latter index acts as an interim index used in the construction of the bridges.

The vectorized algorithm for simulating a single instance of a string of Brownian bridges that pass through the points $\mathbf{y}$ then follows:

1. Simulate $(m+2)(N-1)$ random deviates $\{r_{i+1} \sim N(0, \tau_{i+1}^* - \tau_i^*) : i = 1, ..., (N-1)(m+2)\}$ and set $r_1 = 0$.

2. Let
$$\mathbf{r}^{[m]} = \left\{ \sum_{j=1}^{s} r_j : s = 1, ..., (N-1)(m+2)+1 \right\}, \tag{1.3.13}$$
and translate this sequence into $N-1$ independent Brownian motions through
$$\mathbf{B}^{[m]} = \mathbf{r}^{[m]} - \mathbf{r}_{S_1}^{[m]}, \tag{1.3.14}$$
where S_1 is a set of repeated indexes: $S_1 = \{1 \cdot \mathbb{1}_{1\times(m+2)}, (m+3) \cdot \mathbb{1}_{1\times(m+2)}, ..., ((N-2)(m+2)+1) \cdot \mathbb{1}_{1\times(m+2)}\}$ and $\mathbb{1}_{1\times(m+2)}$ denotes a row vector of $m+2$ ones. S_1 thus repeats the starting index of each transition horizon after including m interim points. This produces $N-1$ independent Brownian motions where each Brownian motion consists of $m+2$ simulated points starting at t_n and ending at t_{n+1} for $n = 1, 2, \ldots, N-1$.

3. $\mathbf{B}^{[m]}$ may then be translated into a string of Brownian bridges by the transformation:
$$\mathbf{d}^{[m]} = \mathbf{B}^{[m]} - \frac{\boldsymbol{\tau}^*}{\boldsymbol{\tau}_{S_2}^* - \boldsymbol{\tau}_{S_1}^*} \cdot \mathbf{B}_{S_2}^{[m]}, \tag{1.3.15}$$

where S_2 is the set of repeated indexes $S_2 = \{(m+2)\cdot \mathbb{1}_{1\times(m+2)}, 2(m+2)\cdot \mathbb{1}_{1\times(m+2)}, ..., (N-1)(m+2)\cdot \mathbb{1}_{1\times(m+2)}\}$. S_2 thus repeats the final index of each individual transition horizon after including m interim points. Up to and including this step, the appropriate time index for the sequences is given by $\boldsymbol{\tau}^*$.

4. Finally, we may translate the endpoints of the bridges to start and end at arbitrary consecutive points $\{y_{t_n} : i = 1, ..., N\}$ by the relation:

$$\mathbf{q}_m = \left\{ \mathbf{d}^{[m]} + \frac{(\boldsymbol{\tau}^*_{S_2} - \boldsymbol{\tau}^*)}{\boldsymbol{\tau}^*_{S_2} - \boldsymbol{\tau}^*_{S_1}} \cdot \mathbf{z}^+ + \frac{(\boldsymbol{\tau}^* - \boldsymbol{\tau}^*_{S_1})}{\boldsymbol{\tau}^*_{S_2} - \boldsymbol{\tau}^*_{S_1}} \cdot \mathbf{z}^- \right\}_{-S_3}, \tag{1.3.16}$$

where $-S_3$ indicates the action of removing the elements S_3 from the sequence in brackets, $\mathbf{z}^-$ repeats the first $N-1$ entries of the vector $\mathbf{y}$ $m+2$ times, $\mathbf{z}^+$ repeats the last $N-1$ entries of $\mathbf{y}$, and $S_3 = S_1 \setminus 1$ (i.e. S_1 minus the first entry). This final operation essentially removes the duplicate end and start points of the simulation so that the resulting sequence consists of m simulated interim points between the nodes $\{(t_n, y_{t_n}) : n = 1, 2, \ldots, N-1\}$. $\mathbf{q}_m$ thus gives a sequence of Brownian bridges that have been strung together with endpoints coinciding with the coordinates (t_n, y_{t_n}). Note that $\mathbf{q}_m$ is time-indexed by the sequence $\boldsymbol{\tau} = \{\tau_r : r = 1, \ldots, (N-1)(m+1)+1, \tau_1 = t_1, \tau_{m+2} = t_2, \ldots\}$.

Using the vectorized algorithm we may generate multivariate Brownian bridges by repeating steps 1–4 successively over each dimension of a given observation sequence. That is, for each dimension j of the state vector in $\Gamma(\mathbf{D}_S)$, one may set $\mathbf{y} = \Gamma^{(j)}(\mathbf{D}_S)$ (where the superscript (j) denotes the j-th entry of the vector $\Gamma(.)$) in turn, after which the collectively simulated 'strings' $\mathbf{Q}_t^{[m]} = \{\mathbf{q}_m^1, \mathbf{q}_m^2, \ldots, \mathbf{q}_m^k\}'$ for $k = \dim(\mathbf{X}_t)$, reflect a single simulation of $U^{[m]}(\Gamma(\mathbf{D}_S))$. Here the superscripts denote the dimension to which each $\mathbf{q}_m$ pertains. Figure 1.3.1 illustrates the application of the vectorized simulation scheme to a subset of the dataset analysed in Section 1.4.1.

Although the scheme is designed to work in conjunction with the data-imputation procedure, it can perhaps be visualised best by simulating a string of bridges on a deterministic curve. For example, consider a deterministic vector of nodes in a three-dimensional space defined by the relation:

$$\mathbf{y} = \{(\sin(0.5\pi t_i), \cos(0.5\pi t_i), 0.5t_i) : i = 1, 2, \ldots, N\}. \tag{1.3.17}$$

Figure 1.3.2 illustrates a single simulation of 100 Brownian bridges at a resolution of $m = 100$ for $\mathbf{y}$ evaluated at $t_1 = 0, t_2 = 1/10, t_2 = 2/10, \ldots, t_{101} = 10$. Using this it can be seen how the endpoints of the simulated bridges are tethered to

FIGURE 1.3.2: Strings of three-dimensional Brownian bridges on a helix with $m = 100$. Nodes (black spheres) are calculated using Equation 1.3.17 for $t_1 = 0, t_2 = 1/10, t_2 = 2/10, \ldots, t_{101} = 10$. R code: Supplementary materials, Section 1.3.

successive nodes of $\mathbf{y}$ such that the trajectory of the sequence of bridges follow the path of Equation 1.3.17.

1.3.3 Calculating pseudo-AIC statistics for the data-imputation procedure

An important part of any statistical analysis is the process of selecting a suitable model from a plethora of candidates that aim to describe the dynamics of the phenomena of interest. Unfortunately, in the field of diffusion processes, the methodology for likelihood-based model selection is less established than other disciplines of time series analysis. Furthermore, under alternative descriptions of the likelihood such as Equation 1.1.10, calculating statistics such as the Akaike information criterion (AIC) by applying the appropriate quadrature rules on the data directly (without imputation) would require an unrealistically high data resolution in order to provide a sufficiently accurate statistic. Although the

imputation procedure circumvents the resolution issue with respect to inferring model parameters, simply comparing likelihood values for competing models obtained during separate MCMC runs under each model would not suffice since the revised likelihood values cannot be disentangled from the imputed data, which would necessarily be different for each run. This could be solved by adding a reversible jump step wherein the imputation algorithm may alternate between different models nested within a more general equation as in Dellaportas *et al.* (2006), for example. Unfortunately, such a scheme presents a number of challenges in the present context as when the data is relatively sparse, achieving desirable MCMC characteristics can be difficult. Instead, we suggest a strategy whereby the parameter estimates are calculated using the data-imputation procedure for each amongst a set of models separately and subsequently plugged into a numerical evaluation of the likelihood under Equation 1.1.9. This may be achieved by solving Equation 1.1.5 using standard (but more computationally expensive) numerical techniques, wherein the transitional density is evaluated numerically and subsequently plugged into Equation 1.1.9. In this way, we may exploit the computational efficiency of the imputation scheme in order to calculate parameter values whilst only having to evaluate the likelihood numerically once in order to calculate an approximate AIC statistic for a given diffusion model. Following Section 1.2, we can use the method of lines in order to approximate the transitional density over successive observation horizons in order to calculate an approximate likelihood function as per Equation 1.2.3. Let $\hat{L}(\boldsymbol{\theta}|\mathbf{D}_S)$ denote the approximate likelihood function calculated using the method of lines, then for a given model specification and estimate of the parameter vector, $\tilde{\boldsymbol{\theta}}$, we calculate a pseudo-AIC statistic using the approximate likelihood as:

$$\mathrm{AIC}^{\star}(\tilde{\boldsymbol{\theta}}) = -2\log(\hat{L}(\tilde{\boldsymbol{\theta}}|\mathbf{D}_S)) + 2 \times \dim(\tilde{\boldsymbol{\theta}}). \tag{1.3.18}$$

We use the notation $\tilde{\boldsymbol{\theta}}$ here in order to make the distinction that the parameter estimate used in the calculation is not necessarily the quantity that maximises $\hat{L}(\boldsymbol{\theta}|\mathbf{D}_S)$ directly – hence the prefix *pseudo*. The premise behind Equation 1.3.18 is that, provided that it is evaluated using parameters which are sufficiently close to the theoretical estimate calculated by maximising Equation 1.1.9, the statistic retains the ability to distinguish between the fit of competing models. By combining parameter estimates calculated using the data-imputation procedure with the method of lines, we may thus fit various diffusion models to a given dataset and subsequently calculate pseudo-AIC statistics in order to compare model fit.

We conclude this section with some remarks on the method of lines as applied to the calculation of Equation 1.3.18: It can easily be argued that since the method of lines can be used to approximate the likelihood directly, it stands to reason

that one can calculate maximum likelihood estimates directly from $\hat{L}(\boldsymbol{\theta}|\mathbf{D}_S)$. However, as we will demonstrate later in this chapter, although the method of lines is a comparatively efficient means for solving PDEs, repeated evaluation of the likelihood using the method of lines will be orders of magnitude slower than the imputation scheme. This follows since the resolution of the lattice $\mathcal{L}$ required to produce a sufficiently accurate approximation is tied both to the functional forms of the drift and diffusion components as well as the parameter values at which they are evaluated. Consequently, the maximum threshold for time step size is dependent on the parameters of the diffusion and care needs to be taken in order to ensure that the parameters of the approximation are chosen so as to produce an accurate solution. Consequently, the mechanics of the scheme in the context of inference can be quite cumbersome to work with and quickly becomes infeasible in the desktop computing environment.

1.4 Application: Using a diffusion model to analyse an ecological time series

Diffusion processes have been successfully applied in ecological contexts (Varughese, 2009, 2011) and important overlapping fields such as biology (Frank and Beek, 2001) and climate modelling (Majda, 1999). The flexibility of diffusion processes as a modelling tool allows one to adapt a given model to incorporate environmental characteristics into the analysis. As such SDEs provide a compact description of multidimensional dynamical processes – an advantageous attribute in the modelling of potentially complex ecological interactions. The incorporation of an infinitesimal random component to a system of ODEs can result in vastly different behaviour of solutions to those predicted by their deterministic counterparts (Mao *et al.*, 2002). This results in more realistic models of natural phenomena in the presence of stochastic environmental factors. For example Preisler (2004) and Brillinger (2003) used bivariate diffusion models in order to describe physical processes such as animal movement whilst simultaneously accounting for spatial restrictions on the process trajectory. Although ecological applications of SDEs typically employ linear diffusion models, the use of diffusion models is not limited to this domain. Indeed, the famous non-linear Lotka-Volterra differential equations have been extended to their SDE counterparts by Gard and Kannan (1976).

Plankton form an integral part of marine ecosystems and count among the largest biomasses on earth. As such changes in plankton abundance can often have profound effects on the environment (and *vice versa*) and the stability of ecosystems. Previously, Varughese and Pienaar (2013) used a non-linear,

tri-variate diffusion to model plankton species competition in the presence of environmental fluctuations. In this section, we study the dynamics of *Emiliania huxleyi* – an abundant species of phytoplankton that can be found throughout the world's oceans. Using the data-imputation scheme in conjunction with the method of lines, we fit various diffusion models to abundance data for *E. huxleyi*. Subsequently, we show how the mechanics of the resulting diffusion model can be used in order to predict the risk of the species undergoing a 'bloom'. These are instances where the abundance of a plankton species can increase by several orders of magnitude within short periods of time and after which the population can just as rapidly decline back to their original levels. Such blooms are often harmful to the ecosystem and can indeed also have an economic impact due to their effect on fisheries. Consequently, measuring the risk of bloom events using a diffusion model can be a very useful tool in the management of the environment.

1.4.1 A diffusion model for *Emiliania huxleyi* abundance

Though much effort goes into the monitoring of terrestrial populations, advances in ocean monitoring systems for marine ecology allow scientists to more accurately keep track of species abundance and ambient variables in marine ecosystems. In the present study, we focus on *Emiliania huxleyi*, a cosmopolite coccolithophorid known to exhibit bloom dynamics (Paasche, 2001). The ubiquitous nature of *E. huxleyi* in marine ecosystems makes it an ideal species for investigating population dynamics (Paasche, 2001). Advanced studies on the factors that allow and contribute to extreme phases in the abundance of the species have been carried out both in their natural habitats (Head *et al.*, 1998) and under laboratory conditions (Buitenhuis, 1999; Engel, 2004). Subsequently, many theories have been developed around the mechanisms that drive *E. huxleyi* abundances.

We demonstrate the application of a diffusion model to the species-environment dynamics of *E. huxleyi* by modelling field samples, taken from the web-archives of the Western Channel Observatory[3] (WCO). The WCO is a leading institute in marine research, collating a wide array of environmental observations and facilitating the collaboration of various scientific disciplines in order to better understand marine dynamics. Initiatives by the WCO have resulted in an unparalleled effort to encapsulate marine and environmental observation into a globally accessible format. We source abundance data for *E. huxleyi* (Widdicombe *et al.*, 2010) as well as surface temperature (Smyth, 2011) from conductivity, temperature, and depth (CTD) water sampler archives, at irregular time intervals for the period beginning 2002 to end 2009. Figure 1.4.1 gives a time series

[3]Instructions for obtaining the data can be found at .

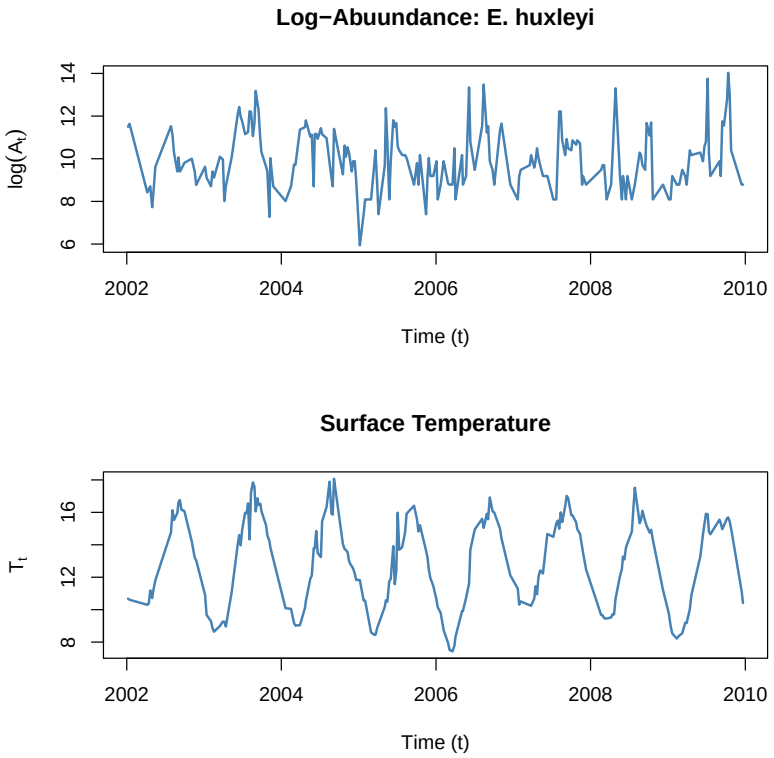


FIGURE 1.4.1: Log-scaled species abundances for *E. huxleyi* (top) and sea surface temperatures (bottom) for the period beginning 2002 to end 2009.

plot of scaled population abundances for *E. huxleyi* and corresponding surface temperatures. Though observations are made at irregular intervals, the data resolution is such that, on average, an observation is made around once every two weeks. Since diffusion models operate continuously in time, it is important to note the unit of time measurement used in the analysis. For the current dataset, we shall assume that one unit of time corresponds to a single month in a calendar year.

We postulate a model for the population-environment dynamics of *E. huxleyi* by defining a system of SDEs that attempt to capture the dynamics of its population abundance in relation to surface temperature. In light of the non-linear behaviour of the species abundance, we aim to produce a model that replicates the salient features of the observed dynamics. Since the observed abundances vary over orders of magnitude, we transform the data using $P_t = \log(A_t)$ where A_t is the observed cell concentration in #cells/litre. Furthermore, in order to avoid excessively 'stretched' distributions, we adopt the so-called slow-fast equation

methodology of Berglund and Gentz (2003) and subsequently scale temperatures to match the magnitude of P_t: $T_t^* = T_t/c$, where $c = median(T_{t_n})/median(P_{t_n})$. This scaling is by no means a limitation of the analysis as we can easily translate backwards to a model in terms of the 'raw' observations T_t by inversion of the scale factor and the application of Itô's lemma. By transforming the data, we model the population-environment dynamics of *E. huxleyi* with respect to surface temperatures using a bivariate diffusion model of the form:

$$\begin{aligned} dP_t &= \mu_1(P_t, T_t^*, t)dt + \sigma_1(P_t)dB_t^{(1)}, \\ dT_t^* &= \mu_2(P_t, T_t^*, t)dt + \sigma_2(T_t^*)dB_t^{(2)}. \end{aligned} \tag{1.4.1}$$

By identifying the appropriate form for the drift and diffusion components $\mu_1(P_t, T_t^*)$, $\sigma_1(P_t)$ etc. under the data, we may draw conclusions as to the nature of the population-environment dynamics of *E. huxleyi* with respect to surrounding water temperatures. For these purposes, we formulate a number of competing models using various functional forms for the drift and diffusion terms of Equation 1.4.1. Subsequently, we compare the performances of these models in order to select a system which adequately approximates the underlying process. This is done by calculating the pseudo-AIC statistic in Equation 1.3.18 using parameter estimates calculated from the data-imputation procedure under a given model.

Since the focus of the study is to identify temperature effects on the population levels of *E. huxleyi*, it is important to find an appropriate scalar diffusion model for the temperature dynamics as we will introduce interaction terms when formulating a diffusion model of the joint dynamics of temperature and abundance levels. As expected, the temperature series oscillates between low temperatures in winter months and high temperatures in summer months. In order to account for this, we model the temperature series using diffusion models of the form:

$$dT_t^* = (\mu(T_t^*) + h(t, A, B))dt + \sigma(T_t^*)dB_t \tag{1.4.2}$$

where

$$h(t, A, B) = A\sin\left(\frac{2\pi t}{12}\right) + B\cos\left(\frac{2\pi t}{12}\right) \tag{1.4.3}$$

for some parameters A and B. The term $h(t, A, B)$ thus serves to account for the time deterministic oscillation present in the series whilst $\mu(T_t^*)$ and $\sigma(T_t^*)$ dictate the behaviour of the stochastic components the process. For the present dataset, we compare five different models for the temperature time series:

Model 1:

$$dT_t^* = (\theta_1(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5 dB_t. \tag{1.4.4}$$

Model 2:

$$dT_t^* = (\theta_1(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5\sqrt{T_t^*}dB_t. \tag{1.4.5}$$

Model 3:

$$dT_t^* = (\theta_1(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5 T_t^* dB_t. \tag{1.4.6}$$

Model 4:

$$dT_t^* = (\theta_1 T_t^*(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5 dB_t. \tag{1.4.7}$$

Model 5:

$$dT_t^* = (\theta_1 T_t^*(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5 T_t^* dB_t. \tag{1.4.8}$$

Note that all five candidate models are of mean reverting form. That is, the drift is chosen so as to predict a stable long-run average for the temperature time series regardless of whether the force of reversion is dependent on the state of the model process. For example, temperature models 1–3 exhibit linear mean reversion whereby, given initial value T_s^* at time s, for some sufficiently large threshold s^* the model process stabilises with expectation

$$E[T_t^*|t \geq s^*] = \theta_2 + \theta_1^{-1}\int^t h(u, \theta_3, \theta_4)du, \tag{1.4.9}$$

regardless of any state dependence in the diffusion term $\sigma(T_t^*)$. In contrast, models 4 and 5 exhibit non-linear reversion whereby the speed of reversion scales with the current level of the process. That is the reversion force $(\theta_2 - T_t^*)$ scales according to $\theta_1 T_t^*$. For such a model the long-run stable dynamics depend on the higher order moments of the process, and thus exhibits dependence on the form of the diffusion term $\sigma(T_t^*)$. Although analysis of the long-run dynamics of non-linear processes is much more complicated than the linear case, the models may still predict stable long-run dynamics whilst mimicking the oscillatory behaviour of the observed series through the term $h(u, \theta_3, \theta_4)$.

In order to conduct inference on the temperature time series, we apply the data-imputation scheme of Section 1.3 for each of the five candidate models. In each instance we ran the scheme (using symmetric proposal distributions, i.e., random walk Metropolis-Hastings) for 150 000 updates using a burn-in period of 50 000 updates at an imputation resolution of $m = 50$ interim points for each pair of successive observations ($N = 214 \Rightarrow 213$ transitions). For each model, we calculate approximate pseudo-AIC values numerically based on the resulting parameter estimates using the methodology of Section 1.3.3. Tables 1.4.1 and

1.4.2 give the parameter estimates, corresponding 90% credibility intervals, and approximate pseudo-AIC values for each of the five candidate temperature models.

In order to verify the analysis, we perform an independent check on the parameter estimates and model fit statistics by using the cumulant truncation procedure of Varughese (2013) (we cover this procedure extensively in Chapter 2) in order to sample parameters for each model, again using 150 000 updates and a burn-in period of 50 000 updates. The resulting estimates and 90% credibility intervals are given in tables 1.4.3 and 1.4.4. We also compare model selection results by calculating the approximate deviance information criterion (DIC) and the effective number of parameters for each model under the cumulant truncation procedure.

Par.	Model 1		Model 2		Model 3	
θ_1	2.11	(1.61, 2.68)	1.76	(1.30, 2.25)	1.37	(1.02, 1.77)
θ_2	9.35	(9.22, 9.48)	9.34	(9.22, 9.46)	9.36	(9.22, 9.52)
θ_3	-3.87	(-5.14,-2.74)	-3.14	(-4.21,-2.15)	-2.33	(-3.27,-1.46)
θ_4	-4.41	(-5.38,-3.58)	-3.85	(-4.62,-3.08)	-3.27	(-3.97,-2.65)
θ_5	1.26	(1.14, 1.38)	0.38	(0.34, 0.42)	0.12	(0.11, 0.13)
AIC*	334.02		321.80		315.69	

TABLE 1.4.1: Estimated parameter values and 90% credibility intervals for models 1–3 (see equations 1.4.4–1.4.8) calculated using the data-imputation scheme. Pseudo-AIC statistics are calculated using these parameter estimates in conjunction with the methodology of Section 1.3.3. R code: Supplementary materials, Section 1.4.

Par.	Model 4		Model 5	
θ_1	0.17	(0.13, 0.22)	0.16	(0.13, 0.21)
θ_2	9.77	(9.62, 9.91)	9.78	(9.65, 9.91)
θ_3	-2.51	(-3.52,-1.62)	-2.26	(-3.23,-1.48)
θ_4	-3.47	(-4.21,-2.84)	-3.26	(-3.89,-2.66)
θ_5	1.20	(1.09, 1.33)	0.12	(0.10, 0.13)
AIC*	353.62		**310.55**	

TABLE 1.4.2: Estimated parameter values and 90% credibility intervals for models 4–5 (see equations 1.4.4–1.4.8) calculated using the data-imputation scheme. Pseudo-AIC statistics are calculated using these parameter estimates in conjunction with the methodology of Section 1.3.3. R code: Supplementary materials, Section 1.4.

Par.	Model 1		Model 2		Model 3	
θ_1	2.69	(2.06, 3.37)	2.29	(1.85, 2.93)	1.95	(1.4, 2.72)
θ_2	9.33	(9.23, 9.42)	9.34	(9.25, 9.45)	9.34	(9.23, 9.45)
θ_3	-5.06	(-6.57,-3.72)	-4.24	(-5.62,-3.13)	-3.62	(-5.42,-2.24)
θ_4	-5.41	(-6.33,-4.4)	-4.74	(-5.54,-3.92)	-4.24	(-5.57,-3.33)
θ_5	1.37	(1.21, 1.53)	0.41	(0.37, 0.47)	0.13	(0.11, 0.15)
DIC	333.40		321.07		317.20	
pD	4.85		4.63		5.10	

TABLE 1.4.3: Estimated parameter values and 90% credibility intervals for models 1–3 (see equations 1.4.4–1.4.8) calculated using the cumulant truncation procedure. DIC values and the effective number of parameters are computed from the MCMC output. R code: Supplementary materials, Section 1.5.

Par.	Model 4		Model 5	
θ_1	0.18	(0.13, 0.23)	0.19	(0.14, 0.24)
θ_2	9.77	(9.65, 9.89)	9.76	(9.63, 9.89)
θ_3	-2.62	(-3.61,-1.59)	-2.66	(-3.67,-1.81)
θ_4	-3.65	(-4.42,-2.92)	-3.62	(-4.34,-3.02)
θ_5	1.23	(1.11, 1.35)	0.12	(0.11, 0.13)
DIC	353.99		**309.85**	
pD	4.90		4.89	

TABLE 1.4.4: Estimated parameter values and 90% credibility intervals for models 4–5 (see equations 1.4.4–1.4.8) calculated using the cumulant truncation procedure. DIC values and the effective number of parameters are computed from the MCMC output. R code: Supplementary materials, Section 1.5.

Interestingly the data-imputation scheme and cumulant truncation procedure produce comparable parameter estimates and model fit statistics. Indeed the pseudo-AIC and DIC statistics match very closely between the two schemes, which is encouraging given the relatively sparse data. For the temperature time series, models 3 and 5 are clear favourites based on the minimum AIC/DIC criteria. The results suggest that the temperature time series exhibits non-linear drift with state dependent volatility.

We now turn our focus to the abundance data for *E. huxleyi*. By modelling the joint series as a bivariate diffusion model, we may postulate various forms of interaction between the dimensions of the diffusion model so as to gain insight into the presence and/or behaviour of temperature effects on population levels of *E. huxleyi*. However, in order for the model to be a realistic approximation to the observed process, we incur some precluding assumptions about the joint dynamics. Firstly, irrespective of whether a significant interaction is indeed

present, population levels should be predicted to sustain for a reasonable amount of time. Note that in this context the statement 'self-sustaining' does not preclude the inability to go extinct: A diffusion model exhibiting 'self-sustaining' dynamics may still reach the zero bound in finite time with a non-zero probability. Secondly, we assume that surface temperatures measured at the depth of the current dataset are unaffected by the presence of *E. huxleyi*. That is, we do not expect the presence of *E.huxleyi* to have a noticeable effect on the dynamics of surface temperatures at the resolution of the present dataset. Consequently, no interaction terms will be included in the temperature component of the model. Under these assumptions, we postulate a number of diffusion models for the joint series by alternating between different forms of drift and diffusion terms for the abundance component of Equation 1.4.1:

Model 1:

$$\begin{aligned} dP_t &= \theta_6(\theta_7 - P_t)dt + \theta_8 dB_t^{(1)} \\ dT_t^* &= (\theta_1 T_t^*(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5 T_t^* dB_t^{(2)}. \end{aligned} \tag{1.4.10}$$

Model 2:

$$\begin{aligned} dP_t &= \theta_6(\theta_7 - P_t)dt + \theta_8\sqrt{P_t} dB_t^{(1)} \\ dT_t^* &= (\theta_1 T_t^*(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5 T_t^* dB_t^{(2)}. \end{aligned} \tag{1.4.11}$$

Model 3:

$$\begin{aligned} dP_t &= \theta_6(\theta_7 - P_t)dt + \theta_8 P_t dB_t^{(1)} \\ dT_t^* &= (\theta_1 T_t^*(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5 T_t^* dB_t^{(2)}. \end{aligned} \tag{1.4.12}$$

Model 4:

$$\begin{aligned} dP_t &= (\theta_6(\theta_7 - P_t) + \theta_9 T_t^*)dt + \theta_8\sqrt{P_t} dB_t^{(1)} \\ dT_t^* &= (\theta_1 T_t^*(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5 T_t^* dB_t^{(2)}. \end{aligned} \tag{1.4.13}$$

Model 5:

$$\begin{aligned} dP_t &= (\theta_6 P_t(\theta_7 - P_t) + \theta_9 T_t^*)dt + \theta_8\sqrt{P_t} dB_t^{(1)} \\ dT_t^* &= (\theta_1 T_t^*(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5 T_t^* dB_t^{(2)}. \end{aligned} \tag{1.4.14}$$

Model 6:

$$\begin{aligned} dP_t &= (\theta_6(\theta_7 - P_t) + \theta_9 (T_t^*)^2)dt + \theta_8\sqrt{P_t} dB_t^{(1)} \\ dT_t^* &= (\theta_1 T_t^*(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5 T_t^* dB_t^{(2)}. \end{aligned} \tag{1.4.15}$$

Model 7:

$$\begin{aligned} dP_t &= (\theta_6(\theta_7 - P_t) + \theta_9 P_t T_t^*)dt + \theta_8\sqrt{P_t}dB_t^{(1)} \\ dT_t^* &= (\theta_1 T_t^*(\theta_2 - T_t^*) + h(t, \theta_3, \theta_4))dt + \theta_5 T_t^* dB_t^{(2)}. \end{aligned} \tag{1.4.16}$$

For the bivariate time series, models 1–3 will serve as baseline models in the sense that temperature levels are assumed to have no effect on abundance levels for *E. huxleyi*. Although these models assume that the dimensions of the process are not coupled, the various volatility specifications will still give an indication of the strength of density dependence in the population volatility. Throughout the model space, the term $\theta_6(\theta_7 - P_t)$ serves as the population regulation mechanism whereby the growth rate $\theta_6\theta_7$ is regulated by the term $-\theta_6 P_t$. Thus, when population levels exceed a threshold of θ_7 (on the log-scale) the growth rate is increasingly stunted. Furthermore, as population levels drop below the threshold θ_7, the rate of demise of the population decreases and the population is presumed to revive. Finally, by including various forms of interaction terms in the drift of the abundance dimension in models 4–7, these intrinsic factors are assumed to be augmented by the effects of changes in surrounding water temperatures. Depending on the intensity and nature of the augmentation – which is dictated by the magnitude of the coefficients of the interaction terms – the effect of variations in water temperature may serve to either inhibit or stimulate population growth. By comparing the performance of the bivariate models 1–7 we may test for the nature and presence of such effects on the abundance of *E. huxleyi* and subsequently find a suitable candidate model for Equation 1.4.1.

	Model 1		Model 2		Model 3		Model 4	
θ_1	0.17	(0.13, 0.21)	0.15	(0.12, 0.19)	0.18	(0.14, 0.23)	0.17	(0.13, 0.22)
θ_2	9.77	(9.65, 9.90)	9.77	(9.65, 9.90)	9.82	(9.65, 10.00)	9.76	(9.63, 9.89)
θ_3	-2.39	(-3.32, -1.59)	-2.06	(-2.91, -1.37)	-2.67	(-3.64, -1.80)	-2.36	(-3.44, -1.54)
θ_4	-3.37	(-4.05, -2.70)	-3.14	(-3.69, -2.63)	-3.56	(-4.20, -3.00)	-3.34	(-4.06, -2.74)
θ_5	0.12	(0.11, 0.13)	0.11	(0.10, 0.12)	0.12	(0.11, 0.13)	0.12	(0.11, 0.13)
θ_6	1.63	(1.27, 2.02)	1.74	(1.32, 2.21)	-0.02	(-0.08, 0.05)	2.02	(1.51, 2.55)
θ_7	9.81	(9.52, 10.07)	9.81	(9.54, 10.07)	2.84	(0.37, 4.81)	7.99	(6.75, 9.31)
θ_8	2.54	(2.39, 2.67)	0.84	(0.77, 0.92)	0.23	(0.22, 0.25)	0.85	(0.77, 0.93)
θ_9	.		.		.		0.40	(0.09, 0.69)
AIC*	978.61		976.02		982.4		**970.58**	

TABLE 1.4.5: Estimated parameter values and 90% credibility intervals for models 1–4 (see equations 1.4.10–1.4.16) calculated using the data-imputation scheme. Pseudo-AIC statistics are calculated using these parameter estimates in conjunction with the methodology of Section 1.3.3. R code: Supplementary materials, Section 1.6.

Par.	Model 5		Model 6		Model 7	
θ_1	0.17	(0.12, 0.22)	0.16	(0.12, 0.21)	0.16	(0.12, 0.20)
θ_2	9.78	(9.64, 9.91)	9.77	(9.63, 9.89)	9.78	(9.66, 9.91)
θ_3	-2.43	(-3.37, -1.51)	-2.34	(-3.19, -1.53)	-2.28	(-3.01, -1.52)
θ_4	-3.40	(-4.12, -2.69)	-3.34	(-3.98, -2.73)	-3.27	(-3.79, -2.77)
θ_5	0.12	(0.10, 0.13)	0.12	(0.11, 0.13)	0.12	(0.11, 0.13)
θ_6	0.16	(0.12, 0.20)	2.03	(1.58, 2.50)	2.35	(1.68, 3.10)
θ_7	7.71	(5.55, 9.41)	8.78	(8.16, 9.47)	8.38	(7.52, 9.30)
θ_8	0.85	(0.77, 0.92)	0.85	(0.77, 0.94)	0.85	(0.78, 0.93)
θ_9	0.38	(0.12, 0.70)	0.02	(0.01, 0.04)	0.04	(0.01, 0.07)
AIC*	973.35		970.7		970.86	

TABLE 1.4.6: Estimated parameter values and 90% credibility intervals for models 5–7 (see equations 1.4.10–1.4.16) calculated using the data-imputation scheme. Pseudo-AIC statistics are calculated using these parameter estimates in conjunction with the methodology of Section 1.3.3. R code: Supplementary materials, Section 1.6.

	Model 1		Model 2		Model 3		Model 4	
θ_1	0.18	(0.14, 0.23)	0.21	(0.16, 0.25)	0.20	(0.15, 0.27)	0.20	(0.15, 0.26)
θ_2	9.76	(9.64, 9.87)	9.76	(9.65, 9.88)	9.77	(9.67, 9.87)	9.76	(9.63, 9.89)
θ_3	-2.52	(-3.35, -1.75)	-3.06	(-4.01, -2.14)	-2.98	(-4.21, -1.90)	-2.91	(-4.10, -1.94)
θ_4	-3.67	(-4.45, -3.07)	-4.04	(-4.83, -3.29)	-3.93	(-4.79, -3.11)	-3.88	(-4.64, -3.13)
θ_5	0.12	(0.11, 0.13)	0.13	(0.11, 0.14)	0.12	(0.11, 0.13)	0.12	(0.11, 0.14)
θ_6	2.11	(1.59, 2.83)	2.23	(1.59, 2.99)	2.09	(1.66, 2.62)	2.85	(2.09, 3.77)
θ_7	9.84	(9.63, 10.06)	9.83	(9.59, 10.07)	9.75	(9.51, 10.05)	7.93	(7.12, 8.83)
θ_8	2.85	(2.58, 3.15)	0.91	(0.81, 1.01)	0.28	(0.25, 0.30)	0.96	(0.83, 1.09)
θ_9	.		.		.		0.59	(0.25, 0.90)
DIC*	976.77		973.40		980.32		**966.51**	
pD	7.73		7.61		9.08		7.56	

TABLE 1.4.7: Estimated parameter values and 90% credibility intervals for models 1–4 (se equations 1.4.10–1.4.16) calculated using the cumulant truncation procedure. DIC values and the effective number of parameters are computed from the MCMC output. R code: Supplementary materials, Section 1.7.

Par.	Model 5		Model 6		Model 7	
θ_1	0.05	(0.04, 0.07)	0.20	(0.15, 0.24)	0.21	(0.17, 0.28)
θ_2	9.88	(9.85, 9.91)	9.75	(9.65, 9.85)	9.77	(9.64, 9.87)
θ_3	0.08	(-0.18, 0.33)	-2.86	(-3.61,-2.04)	-3.16	(-4.39,-2.27)
θ_4	-1.83	(-2.02,-1.62)	-3.84	(-4.47,-3.25)	-4.14	(-5.05,-3.53)
θ_5	0.11	(0.10, 0.11)	0.12	(0.11, 0.13)	0.13	(0.11, 0.14)
θ_6	0.16	(0.13, 0.19)	6.30	(5.04, 7.80)	3.43	(2.87, 4.20)
θ_7	8.13	(7.86, 8.35)	8.25	(7.84, 8.61)	7.92	(7.30, 8.42)
θ_8	0.82	(0.80, 0.83)	1.35	(1.19, 1.53)	0.95	(0.86, 1.06)
θ_9	0.30	(0.23, 0.36)	0.11	(0.10, 0.11)	0.07	(0.04, 0.11)
DIC*	970.25		966.86		966.78	
pD	7.43		8.05		8.17	

TABLE 1.4.8: Estimated parameter values and 90% credibility intervals for models 5–7 (see equations 1.4.10–1.4.16) calculated using the cumulant truncation procedure. DIC values and the effective number of parameters are computed from the MCMC output. R code: Supplementary materials, Section 1.7.

Tables 1.4.5 and 1.4.6, give resulting parameter estimates and 90% credibility intervals calculated using the data-imputation scheme whilst tables 1.4.7 and 1.4.8 does the same for the cumulant truncation procedure. Pseudo-AIC statistics and DIC statistics for each method are also given for each model under the respective inference schemes. In each case 150 000 updates were made with a burn-in period of 50 000 iterations. In the case of the data-imputation procedure, we used an imputation resolution of $m = 50$ interim points for each pair of successive observations. Average acceptance rates for the Brownian bridge updates are satisfactory (given both the time-inhomogeneity of the models, and the relatively poor data resolution), ranging from 75% to 85% with rates dropping slightly for models 4–7. Computation times ranged from around 1 hour and 10 minutes to 1 hour and 20 minutes under the data-imputation scheme and around 30 minutes to 1 hour for the cumulant truncation procedure on a 2.66GHz Intel i5-480M processor. A direct comparison of the computational efficiency of the two algorithms is difficult since in general the efficiency of the cumulant truncation procedure varies with the complexity of a given diffusion model, whereas the computational overhead of the imputation procedure is primarily dictated by the imputation resolution. Indeed, the cumulant truncation procedure as implemented here uses a combination of C++ and R code, whereas the imputation scheme was executed in R alone. Again both methods produce comparable parameter estimates for the models fitted to the joint time series. Although the pseudo-AICs, in absolute magnitude, are slightly higher than the DIC values calculated under the cumulant truncation procedure, the relative differences between models are again remarkably similar. Models 4, 6 and 7 produce very similar model fit statistics with Model 4 having the minimal pseudo-AIC. Comparing pseudo-AIC statistics to those of the reference models 1, 2 and 3, there is significant evidence that temperature does indeed affect *E. huxleyi* abundance levels. The DIC criteria

calculated using the cumulant truncation procedure verifies the results. Note however that under both methods the resulting pseudo-AIC and DIC statistics of models 4, 6 and 7 are very close in absolute value. It may indeed be the case that although there is an identifiable interaction term, there may not be enough data to clearly distinguish between linear and non-linear interactions. That said, a linear model constitutes a simpler model of the observed dynamics. Consequently, we select Model 4 on grounds of parsimony.

Although a diffusion model itself can provide insight into the behaviour of real world processes, the value of diffusion models extends beyond providing a compact description of the observed dynamics. In the following section, we show how the analysis may be extended by using parameter draws from the MCMC output in conjunction with the proposed model of population-environment dynamics in order to make useful inference on extreme population events for *E. huxleyi*.

1.4.2 Bloom potential function

Diffusion models enable one to formulate compact probabilistic models of multidimensional dynamical systems observed in the real world. Indeed the ultimate goal of most studies in the application of diffusion processes is to forecast the behaviour of an observed process with regard to some predefined future event. For example, in ecology, we may wish to assess the likely values of future population abundances or times at which extreme population numbers are likely to be observed. *E. huxleyi* is well known for forming very large 'blooms' wherein populations undergo short explosive phases resulting in large cell concentrations at the ocean surface. Although the species itself is microscopic, the phenomena can result in spectacular visual and chemical changes in the environment. Indeed, the effects are prominent enough that images taken from lower earth orbit have been used to assess the presence and extent of population blooms (Tyrrell and Merico, 2004). Naturally, such bloom events have a measurable impact on the environment in general. It is thus important to understand and quantify such events in order to better manage the environment. Given an appropriate statistical model of the population-environment dynamics of *E. huxleyi*, we may formulate various measures of such risks based on historical data so as to extrapolate what can be expected in the near future.

By inferring an appropriate diffusion model of the *E. huxleyi* dataset we can develop a measure of expected abnormal abundances from the model transitional density by borrowing from the concept of the conditional tail expectation. As such we define the bloom potential function (BPF) of a population P_t, as the

quantity:

$$BPF_{\boldsymbol{\theta}}(t, \omega_t) = E[P_t | P_t > \omega_t, P_s = p_s, \mathbf{F}_s], \quad t > s \tag{1.4.17}$$

where ω_t is some threshold abundance chosen *a priori* to the analysis and $\mathbf{F}_s$ gives initial value(s) for the remaining factor(s) in the model (in this case, temperature). The BPF thus measures expected population levels (log-population in the present study) above a given threshold ω_t. Associated with this threshold is the probability of observing values in excess of ω_t, which depends on how the threshold is defined. For example, if ω_t is a fixed value the probability associated with the threshold event will vary over time. This follows since the transitional density varies over time irrespective of whether the model is time-homogeneous or not. Alternatively, one may take the classical approach and define the threshold in terms of the probability of exceeding the threshold at any given time, in which case ω_t will vary in accordance with the transitional density. For our purposes, we use the classical convention wherein ω_t is defined as the upper 95% quantile of the log-abundance of *E. huxleyi* predicted by the diffusion model:

$$\omega_t = \min\{p_t : \text{Prob}(P_t \leq p_t) \geq 0.95\}. \tag{1.4.18}$$

Once an appropriate form has been chosen for ω_t we can calculate the BPF under a given model and parameter set. If the parameters of the diffusion are known exactly, we can calculate the BPF as a deterministic curve. However, in the context of inference one has to account for uncertainty in the parameters of the underlying diffusion model. Consequently, the BPF function will instead have a distribution determined by the variability in the parameter set. For the *E. huxleyi* we can for example sample parameters of Model 4 directly from the MCMC output and thus, by repeatedly evaluating the BPF using these parameter samples, calculate the distribution of the BPF.

Although the principles underlying the calculation of the BPF are quite simple, the physical calculation of the BPF warrants discussion. For example, the simplest strategy for calculating the BPF would be to use brute force simulation: This would consist of simulating trajectories of the diffusion over the desired transition horizon and subsequently recording the observed expectation under a given threshold regime for ω_t. Although this technique is the least costly in terms of mathematical complexity and programming time, applying brute force simulation techniques can become extremely costly in a serial computing environment – keeping in mind that the simulations will have to be carried out repeatedly for a number of parameter samples. As such we advocate calculating the BPF directly. Unfortunately, direct calculation of the BPF leads to significant mathematical difficulties: In order to evaluate the bloom potential for *E. huxleyi* directly, we need to evaluate the marginal transitional density under Equation 1.4.1. The

resulting marginal can then be used to evaluate Equation 1.4.17:

$$BPF_{\boldsymbol{\theta}}(t,\omega_t) = \int_{\omega_t}^{\infty} u g_{P_t|P_t>\omega_t}(u|P_s,T_s^*)du, \tag{1.4.19}$$

where $g_{P_t|P_t>\omega_t}(p_t|P_s,T_s^*)$ is the marginal density of P_t conditioned on the event that it is greater than ω_t at time t, given that the joint process started in (P_s,T_s^*) at time s. Using the method of lines, we may numerically evaluate the joint transition density of a given diffusion over arbitrarily large time horizons. Then, from the approximate joint density function, Equation 1.4.17 can be calculated by first integrating over the scaled temperature dimension of the bivariate transition density approximation and subsequently evaluating the BPF. Using the notation of Section 1.2.1, we construct a lattice $\mathcal{L}$ consisting of $d_1 \times d_2$ nodes with limits $[x_{min}^{(1)}, x_{max}^{(1)}]$ and $[x_{min}^{(2)}, x_{max}^{(2)}]$ in the $P_t \times T_t^*$-plane. Then, under left-endpoint quadrature the calculation becomes:

$$\begin{aligned} &BPF_{\boldsymbol{\theta}}(t,\omega_t) \approx \\ &\frac{\sum_{i=0}^{d_1-1} x_i^{(1)} \left[\sum_{j=0}^{d_2-1} f_{i,j}(t)(x_{j+1}^{(2)} - x_j^{(2)})\right](x_{i+1}^{(1)} - x_i^{(1)})\mathbb{1}(x_i^{(1)} \geq \omega_t)}{\sum_{i=0}^{d_1-1} \left[\sum_{j=0}^{d_2-1} f_{i,j}(t)(x_{j+1}^{(2)} - x_j^{(2)})\right](x_{i+1}^{(1)} - x_i^{(1)})\mathbb{1}(x_i^{(1)} \geq \omega_t)}, \end{aligned} \tag{1.4.20}$$

where $f_{i,j}(t)$ is the joint transition density approximation evaluated using the method of lines at the i,j-th lattice point. Figure 1.4.2 illustrates the evolution of the contours of the joint transition density of Equation 1.4.1, calculated using the method of lines under the dynamics and estimated parameters for Model 4. Under the dynamics of Model 4, the density can be seen to exhibit periodic increases in population levels as a result of the effect that the sinusoidal component of T_t^* has on the population abundance through its drift. Thus, although the BPF only focuses on the population component of the diffusion model, the bloom potential will intrinsically depend on the dynamics of the temperature component under the present model. Due to changes in the probability density over time (note how the density changes from the bottom to the top of Figure 1.4.2) the expected population levels above the 95% quantile changes over time in accordance with the shift in density over time – reflecting the sinusoidal behaviour of the diffusion model.

Although calculating the bloom potential function from the joint density is relatively straightforward, and provides improved computational efficiency over simulation, for certain classes of diffusion it is possible to reduce the computational complexity of the bloom potential function in order to further improve computational efficiency. Noting that calculation of the BPF requires only the marginal transition density of the diffusion model, we make use of the marginal

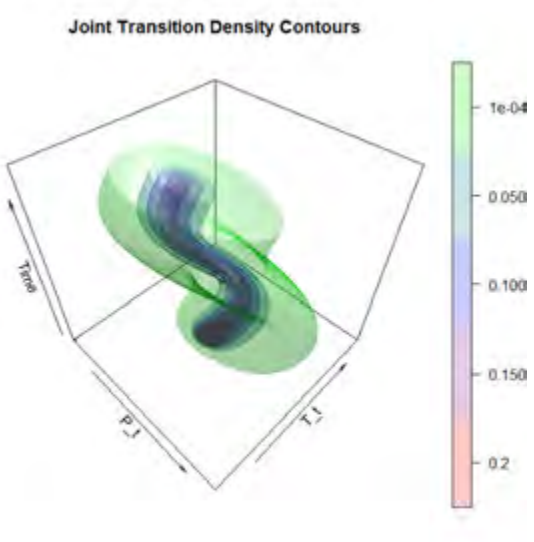


FIGURE 1.4.2: Time-evolution of arbitrary contour levels of the transition density of Equation 1.4.13 calculated using the parameter estimates from the data-imputation procedure. The transition density evolves from a point mass and subsequently propagates density over the support in accordance with Equation 1.1.5. Under the dynamics of Equation 1.4.13 the density periodically expands and contracts due to the sinusoidal drift and drift interaction. R code: Supplementary materials, Section 1.8.

Kolmogorov equation derived in Section 1.2.2 in order to evaluate the bloom potential function: Using the template of Equation 1.4.1, the marginal transitional density of the population component, $f^{(1)}(P_t|P_s, T_s^*)$, is governed by the hybrid PDE:

$$\begin{aligned}\frac{\partial}{\partial t} f^{(1)}(P_t|P_s, T_s^*) = &- \frac{\partial}{\partial P_t}\Big[\psi^\star(P_t, t) f^{(1)}(P_t|P_s, T_s^*)\Big] \\ &+ \frac{1}{2}\frac{\partial^2}{\partial P_t^2}\Big[\nu^\star(P_t, t) f^{(1)}(P_t|P_s, T_s^*)\Big],\end{aligned} \tag{1.4.21}$$

where

$$\begin{aligned}\psi^\star(P_t, t) &= \mathrm{E}_{T_t^*|P_t}\big[\mu_1(P_t, T_t^\star, t)\big], \\ \nu^\star(P_t, t) &= \mathrm{E}_{T_t^*|P_t}\big[\sigma_1^2(P_t, T_t^\star, t)\big].\end{aligned} \tag{1.4.22}$$

Equation 1.4.21 is thus a PDE resembling the Kolmogorov forward equation for a scalar diffusion, but with the addition that $\psi^\star$ and $\nu^\star$ depend on the evolution of the conditional expectation of the temperature terms in the drift and diffusion

terms of the population component. The role of the conditional expectation is to account for the effect of T_t^* on P_t in order to extract the marginal transitional density of P_t without calculating the joint transition density. For multivariate diffusion models with diagonal diffusion matrices and interaction terms which are polynomial in T_t^*, the moment trajectories can be shown to be governed by a system of ODEs. Thus, Equation 1.4.21 expresses the evolution of the marginal transitional density in terms of a PDE and a system of ODEs which describe the moment equations of the temperature component of the model – hence the designation *hybrid*. For example, under the dynamics of Model 4, the resulting hybrid PDE is given by:

$$\begin{aligned}\frac{\partial}{\partial t}f^{(1)}(P_t|P_s,T_s^*) = &-\frac{\partial}{\partial P_t}\left[(\theta_6(\theta_7-P_t)+\theta_9 m_1(t))f^{(1)}(P_t|P_s,T_s^*)\right] \\ &+\frac{1}{2}\frac{\partial^2}{\partial P_t^2}\left[\theta_8^2 P_t f^{(1)}(P_t|P_s,T_s^*)\right]\end{aligned} \tag{1.4.23}$$

where

$$\begin{aligned}\frac{\partial}{\partial t}m_i(t) =&i(\theta_1\theta_2 m_i(t)-\theta_1 m_{i+1}(t)+h(t,\theta_3,\theta_4)m_{i-1}(t)) \\ &+\theta_5^2\frac{i(i-1)}{2}m_i(t)\text{Ind}(i\geq 2)\end{aligned} \tag{1.4.24}$$

for $i = 1, 2, \ldots$. The Dirac delta initial condition for the population component is retained as $f^{(1)}(p_s|P_s,T_s^*) = \delta(p_s - P_s)$, whilst the initial values of the moment equations are given by $m_i(s) = (T_s^\star)^i$. By truncating the system in Equation 1.4.24 under the assumption of cumulant neglect i.e., setting

$$m_q(t) = \sum_{r=1}^{q-1}\binom{q-1}{r-1}\kappa_r(t)m_{q-r}(t) \tag{1.4.25}$$

for some truncation order $q > 2$, where $\kappa_r(t)$ is calculated recursively using the relation:

$$\kappa_r(t) = m_r(t) - \sum_{l=1}^{r-1}\binom{r-1}{l-1}\kappa_l(t)m_{r-l}(t), \tag{1.4.26}$$

we can evaluate the moment equations in conjunction with Equation 1.4.23 in order to calculate the marginal transition density. By applying the method of lines to Equation 1.4.21 we can thus reduce numerical evaluation of the marginal density to a system of $d_1 + q$ ODEs from the previous $d_1 \times d_2$ under the joint density approximant. Consequently, calculation of the bloom potential function takes significantly less time under Equation 1.4.21 in comparison to calculating it from Equation 1.1.5. Using the marginal density approximation, the BPF

calculation becomes:

$$\begin{aligned} BPF_{\boldsymbol{\theta}}(t,\omega_t) &= \int_{\omega_t}^{\infty} u g_{P_t|P_t>\omega_t}(u_t|x_s)du \\ &\approx \frac{\sum_{i=0}^{d_1-1} x_i^{(1)} f_i^{(1)}(t)(x_{i+1}^{(1)} - x_i^{(1)})\mathbb{1}(x_i^{(1)} \geq \omega_t)}{\sum_{i=0}^{d_1-1} f_i^{(1)}(t)(x_{i+1}^{(1)} - x_i^{(1)})\mathbb{1}(x_i^{(1)} \geq \omega_t)}. \end{aligned} \tag{1.4.27}$$

Applying this technique to the *E. huxleyi* dataset, we are able to efficiently calculate the distribution of the BPF under Model 4 using parameter samples from the output of the data-imputation procedure. Setting $\omega_t = \min\{p_t : \text{Prob}(P_t \leq p_t) \geq 0.95\}$ we evaluate expected extreme abundances of *E. huxleyi* for the two years following the final time series observation. Interestingly, on the original cell count scale (i.e., transforming back from the log-scale), the mean BPF indicates expected extreme abundances of around 600 000 cells/litre during the peak of the temperature cycle and around 200 000 cells/litre during the trough of the temperature cycle. Accounting for uncertainty in the parameters of the model, however, these figures can be as high as 1 100 000 and 400 000 cells/litre respectively – using the upper 95%-quantile of the BPF distribution as a conservative BPF measure. Even at the 5%-quantile of the BPF distribution extreme counts cycle between around 300 000 and 150 000 cells/litre. With regard to the interpretation of these figures, it is important to note that, although we have fixed the probability of extreme events, the risks associated with these events are not fixed. This follows from the fact that the so-called risk in this context lies with the values of the cell counts. Consequently, although we have fixed the probability associated with an extreme event under the diffusion model, a higher risk may be associated with peaks of the temperature cycle than with troughs of the temperature cycle. Using the BPF as a guide, high-risk phases occur during the months of August to October in the *E. huxleyi* dataset.

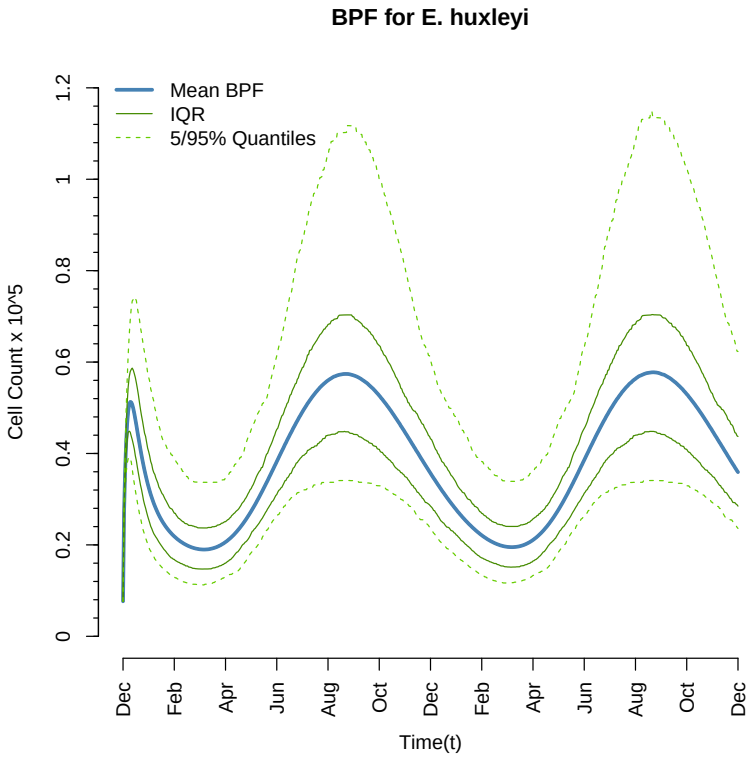


FIGURE 1.4.3: BPF on the exponential scale calculated over 2 years following final time series observation using 500 samples of the parameter vector. Mean BPF (blue), inter quartile range (green) and 95% quantiles (dashed green) are shown. R code: Supplementary materials, Section 1.8.

1.5 Software: The DiffusionRimp package

In the preceding sections, we explored the various facets of conducting inference and analysis on diffusion models using the data-imputation procedure of Kalogeropoulos *et al.* (2011) in conjunction with numerical techniques for calculating the transitional density. Although the methodology as presented here is developed with an aim toward analysing an ecological dataset, the methodology can be applied in various other scientific settings. With a mind to making the algorithms more accessible, we have developed a software package for the R-language, **DiffusionRimp** (Pienaar and Varughese, 2015b), which provides routines for performing data-imputation, calculating pseudo-AICs, and calculating

approximate density functions using the method of lines for scalar and bivariate non-linear diffusion processes.

1.5.1 Interface

In order to analyse a given diffusion model in the software environment, we require an interface that allows the user to define a diffusion model in a way which can be interpreted by the software. Whilst there are numerous ways in which this can be achieved, it is also important that the interface mimics how hand-written models are communicated. Since diffusion models are defined by assigning a functional form to the drift and diffusion coefficients of the SDE (Equation 1.1.1), we emulate this in the software environment by letting the user assign coefficients using lexical elements typically used in written forms of diffusion models. For example, the drift and diffusion coefficients typically consist of three elements: Spatial variables (X_t, Y_t, etc.), a temporal component (i.e., some function of time (t)), and finally the parameters of the process (for example, α, β etc.). Naturally, depending on the particular methodology being used and the type of analysis being conducted, various restrictions may apply to the specification of these coefficients. Fortunately, since both the data-imputation scheme and the method of lines are quite general in the sense that very little restrictions are placed on the specification of the drift and diffusion coefficients, the methodology can easily be adopted in the software environment using this interface. As such, the **DiffusionRimp** package operates by scanning the workspace for functions with pre-determined names that give the functional form of the drift and diffusion coefficients under a given template SDE. Depending on the routine being employed, the user may then define a diffusion model by assigning the desired functional form to the coefficients of the diffusion model in terms of lexical elements emulating those in the written form. For example, the diffusion model

$$dX_t = \alpha X_t(\beta + \sin(2\pi(t - 0.25)) - X_t^2) + \sigma(1 + 0.25\sin(3\pi t))dB_t \qquad (1.5.1)$$

may be defined in R using the code:

```
R> # Define the drift coefficient:
R> mu  <- function(X, t){alpha*X*(beta + sin(2*pi*(t - 0.25)) - X^2)}
R>
R> # Define the diffusion coefficient:
R> sig <- function(X, t){sigma*(1 + 0.25*sin(3*pi*t))}
R>
R> # Call some routine:
R> res <- MOL.density(Xs, Xt, s, t, delt, N)
```

where $\mu(X_t, t) =$ `mu(X, t)` and $\sigma(X_t, t) =$ `sig(X, t)` in the template equation:

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dB_t. \tag{1.5.2}$$

Using this interface, the user can treat the coefficients of the diffusion as stand-alone R-functions which can be used as such throughout the workspace. From this, a particular routine can be applied by letting it scan the workspace for the drift and diffusion coefficients and subsequently pass the peripheral parameters specific to that routine as parameters. Alternatively, we can pass the coefficients of the model as arguments to the routine of interest, as in:

```
R> # Call some routine:
R> res <- MOL.density(Xs, Xt, s, t, delt, N,
+          drift = "alpha*X*(beta + sin(2*pi*(t - 0.25)) - X^2)",
+          diffusion = "sigma*(1 + 0.25*sin(3*pi*t))")
```

Superficially this appears to be significantly more simple, however, this interface becomes quite cumbersome as model complexity increases and excessively long text expressions need to be passed as arguments to the desired routine. This is especially true in the multivariate case. As such, we prefer the functional interface. In the sections that follow, we demonstrate by way of practical examples how the software may be used to apply the data-imputation procedure and the method of lines to the analysis of diffusion processes.

1.5.2 Outline of the package

The **DiffusionRimp** package consists of various routines for analysing diffusion processes. Below follows an outline of the routines and their respective functions (functions that use C++ are indicated with an asterisk):

`RS.impute`: Perform inference on a scalar diffusion model using the random walk Metropolis-Hastings algorithm under the data-imputation scheme.

`BiRS.impute`: Perform inference on a bivariate diffusion model using the random walk Metropolis-Hastings algorithm under the data-imputation scheme.

`MOL.density`: Calculate the transitional density of a scalar diffusion model using the method of lines.

`BiMOL.density`: Calculate the transitional density of a bivariate diffusion model using the method of lines.

`MOL.passage`: Calculate the first passage time density of a scalar time-homogeneous diffusion model with fixed upper and lower thresholds (we defer discussion of first passage time problems to a later chapter in this thesis).

`BiMOL.passage`: Calculate the first passage time density of a time-homogeneous bivariate diffusion model with static perimeter (we defer discussion of first passage time problems to a later chapter in this thesis).

`MOL.aic`*: Calculate an approximate AIC value for a scalar diffusion model using the method of lines.

`BiMOL.aic`*: Calculate an approximate AIC value for a bivariate diffusion model using the method of lines.

1.5.3 Example applications

In the examples that follow we demonstrate how **DiffusionRimp** package is used in practice. The package can be found on GitHub at `https://github.com/eta21` and the Comprehensive R Archive Network (CRAN) at `https://cran.r-project.org/package=DiffusionRimp`. In addition to the examples showcased here, detailed examples can be found within the package vignettes on the package CRAN page or by running the command: `browseVignettes("DiffusionRimp")` after loading the package.

1.5.3.1 Calculate the transitional density of a highly non-linear bivariate diffusion process

Using the method of lines, it is possible to analyse highly non-linear time-inhomogeneous diffusion processes. By approximating a solution to Equation 1.1.5 using the method of lines, we can gain insight into the probabilistic evolution of a diffusion model. For example, consider a diffusion process with dynamics given by the SDE:

$$\begin{aligned} dX_t &= \alpha_x(X_t(\beta_x - X_t^2) + \sin(0.5\pi t)Y_t)dt + \sigma_x dB_t^{(1)} \\ dY_t &= \alpha_y(Y_t(\beta_y - Y_t^2) - \sin(0.5\pi t)X_t)dt + \sigma_y dB_t^{(2)}. \end{aligned} \tag{1.5.3}$$

Equation 1.5.3 has a number of interesting properties: Firstly, by calculating the zeroes of the drift vector for fixed time t, it can be seen that for positive α_x and α_y, the process exhibits drift forces in the direction of four distinct modes depending on which quadrant of the axes the process is located in. Secondly,

as time progresses these nodes change position periodically. To see how this manifests in the transitional density, we use the `BiMOL.density()` function in order to calculate the approximate transitional density under the method of lines. Here, the prefix `BiMOL` indicates that the function applies a **bi**variate implementation of the **m**ethod **o**f **l**ines. The first step in using the method of lines is to define the model, model parameters, and initial conditions of the model. Assuming that $\{\alpha_x, \beta_x, \sigma_x, \alpha_y, \beta_y, \sigma_y\} = \{1, 1, 0.5, 1, 1, 0.5\}$ with initial values $\{X_s, Y_s\} = \{1, 1\}$, we can define the model in R using:

```
R>  library("DiffusionRimp")
R>
R>  # Define the model parameters:
R>  alpha.x <- 1; beta.x <- 1; sigma.x <- 0.5; Xs <- 1;
R>  alpha.y <- 1; beta.y <- 1; sigma.y <- 0.5; Ys <- 1;
R>
R>  # Define drift and diffusion terms:
R>  mu1   <- function(X, Y, t){alpha.x*X*(beta.x - X^2) + sin(0.5*pi*t)*Y}
R>  mu2   <- function(X, Y, t){alpha.x*Y*(beta.x - Y^2) - sin(0.5*pi*t)*X}
R>  sig11 <- function(X, Y, t){sigma.x}
R>  sig22 <- function(X, Y, t){sigma.y}
```

The next step is to define the peripherals of the numerical approximation. This consists of defining the limits of the xy-lattice, the number of nodes on each side of the square lattice, the transition horizon, and the step size on the transition horizon for the numerical solution of the ordinary differential equations that approximate the transitional density surface. In R:

```
R> # Peripheral parameters of the problem:
R> s     <- 0            # Starting time
R> t     <- 10           # Final time
R> Xlim  <- c(-2.2, 2.2) # Limits in X-dimension
R> Ylim  <- c(-2.2, 2.2) # Limits in Y-dimension
R> Nodes <- 51           # How abscissae in each dimension
R> delt  <- 0.01         # Time step size
R>
R>  # Run the method of lines:
R>  res <- BiMOL.density(Xs, Ys, s, t, Xlim, Ylim, Nodes ,delt)
```

Here, the `BiMOL.density()` function sets up a 51×51 square lattice on the section $[-2.2., 2.2] \times [-2.2., 2.2]$, after which the lattice is *shifted* so that the initial value falls on the lattice point. Subsequently, by setting up a 51×51 system of ODEs, the transitional density is approximated over the transition horizon $[s, t]$ by solving the system of ODEs using increments of size `delt`. The resulting calculations are then returned in the form of a list containing elements such as the density approximation, the time lattice etc. The transitional density can then be visualised using standard perspective plots, heat maps or contour plots.

Figure 1.5.1 illustrates the evolution of the transitional density of Equation 1.5.3 at various points along the transition horizon.

Although a highly non-linear model such as Equation 1.5.3 can be challenging to analyse, the method of lines can be applied to even more challenging problems. Consider for example a diffusion model with dynamics governed by the SDE:

$$\begin{aligned} dX_t &= -\alpha_x X_t \sin(X_t \pi)dt + \sigma_x dB_t^{(1)} \\ dY_t &= -\alpha_y Y_t \sin(Y_t \pi)dt + \sigma_y dB_t^{(2)}. \end{aligned} \tag{1.5.4}$$

Figure 1.5.2 illustrates the transitional density of Equation 1.5.4 for the parameter set $\{\alpha_x, \sigma_x, \alpha_y, \sigma_y\} = \{1, 0.7, 1, 0.7\}$ with initial values $\{X_s, Y_s\} = \{0.5, 0.5\}$. The transitional density is calculated by using 201×201 points on a lattice on the region $[-4, 4] \times [-4, 4]$ using a time step size of 0.002. Here, the dynamics of the process lead to a complex transitional density surface with numerous peaks and valleys. Furthermore, as time progresses the number of modes of the transitional density increases. This serves to emphasize the fact that, depending on the specification of the diffusion model, it can be extremely difficult to derive a closed form transitional density function or likewise express the transitional density in terms of a known distribution.

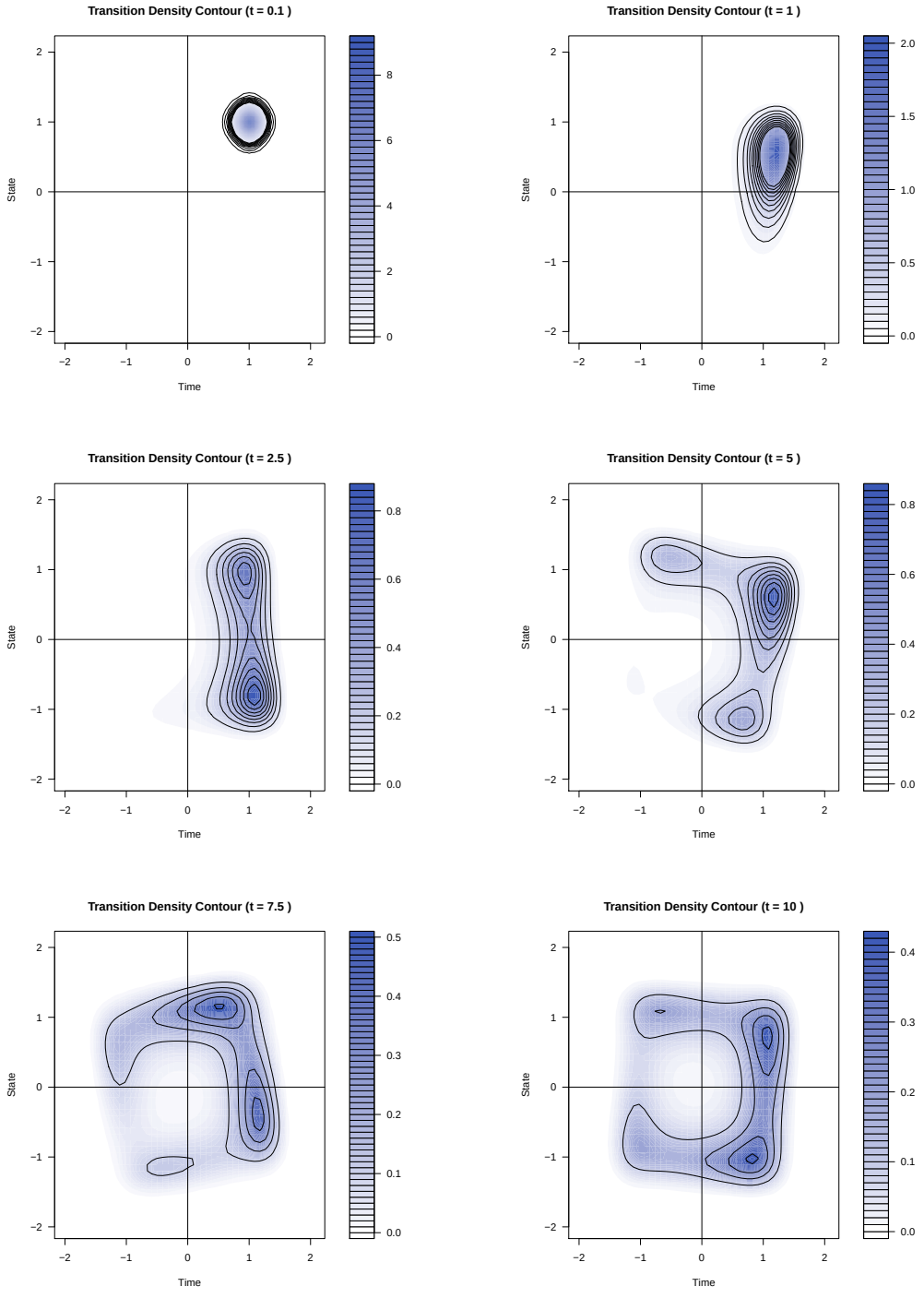


FIGURE 1.5.1: Contourplot of the transitional density of Equation 1.5.3 at times $t = 0.1, 1, 2.5, 5, 7.5$ and 10. Initially, the transitional density appears unimodal. As time progresses the transitional density becomes multimodal. The sinusoidal terms in the drift alternate the spin of the process trajectory between clockwise and anti-clockwise over time. R code: Supplementary materials, Section 1.9.

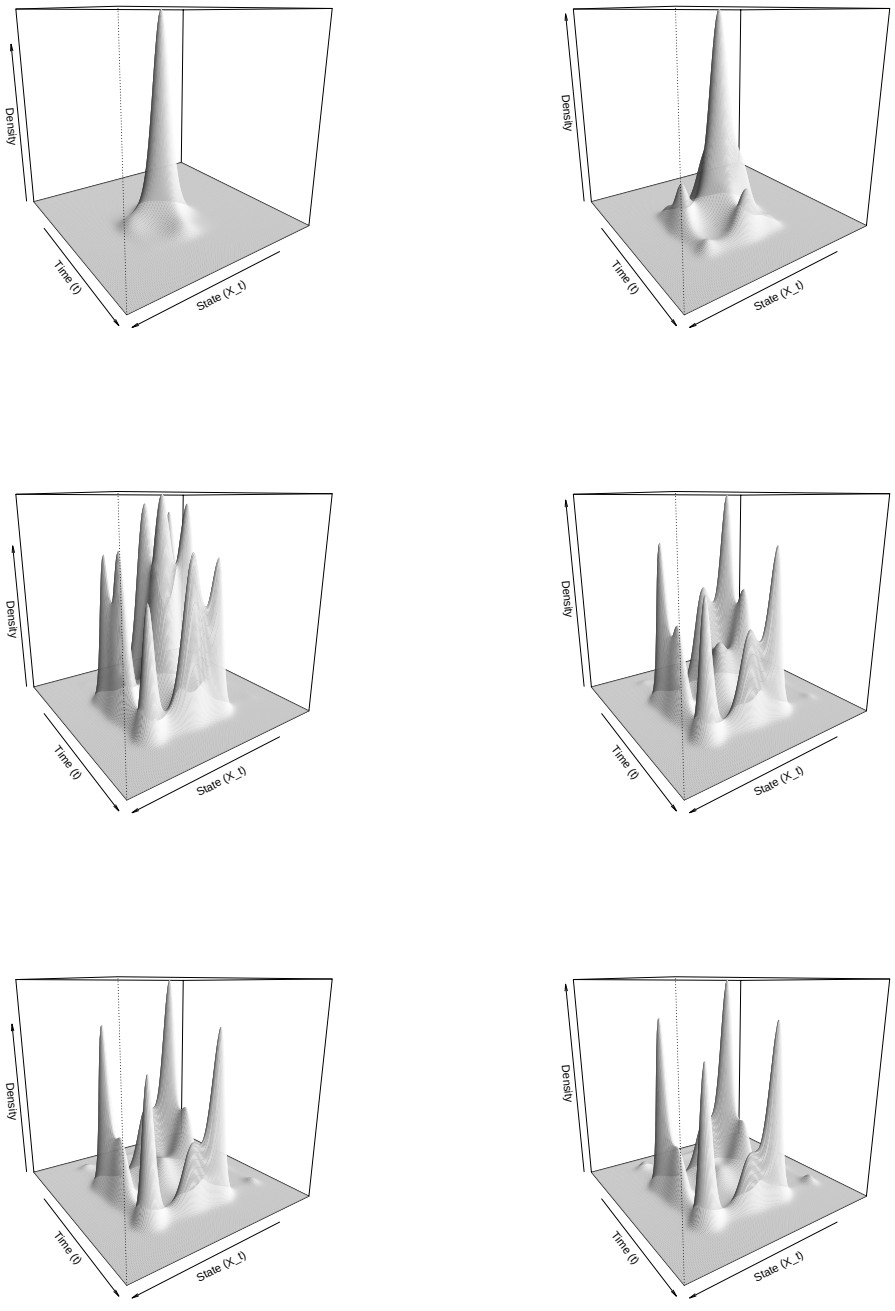


FIGURE 1.5.2: Perspective plot of the transitional density of Equation 1.5.4 at times $t = 0.5, 1, 2, 3, 4$ and 5. Again the transitional density starts out as unimodal and as time progresses the transitional density becomes multimodal. In this case, the transitional density has numerous modes of varying magnitude. R code: Supplementary materials, Section 1.9.

1.5.3.2 Estimate the parameters of a non-linear, time-inhomogeneous diffusion process via data-imputation

One of the primary motivations for building the **DiffusionRimp** package is to perform inference using the data-imputation scheme outlined in Section 1.3. By using the vectorized Brownian bridge simulation scheme, this can be achieved with a reasonable degree of efficiency within the R programming environment. To illustrate the workings of the algorithm, we analyse a simulated dataset using the `RS.impute()` function – a routine that implements the data-imputation procedure for scalar diffusions. The `RS` prefix is a shorthand for 'Roberts and Stramer', the original authors of the algorithm. For purposes of the simulation study, we have included a simulated dataset for a 'double-well' diffusion with dynamics:

$$dX_t = \theta_1 X_t(\theta_2 + \theta_3 \sin(0.5\pi t) - X_t^2)dt + \theta_4 dB_t, \tag{1.5.5}$$

sampled at equispaced time points $t_1 = 0, t_2 = 0.25, t_3 = 0.5, \ldots, t_{201} = 50$ under the parameter set $\{\theta_1, \theta_2, \theta_3, \theta_4\} = \{1, 1, 1, 0.5\}$. Figure 1.5.4 illustrates the simulated trajectory.

Since the data-imputation algorithm requires that the diffusion being fitted is reducible, the interface for defining a model in the workspace deviates slightly from that of the rest of the package: For the purposes of the imputation algorithm, various pre-defined diffusion structures are provided (as opposed to letting the user arbitrarily define the functional form of the diffusion coefficient). This ensures that the user does not specify an irreducible diffusion and that the correct transformations are applied during implementation of the algorithm. Consequently, the model is defined by assigning a drift function as usual and subsequently specifying the diffusion structure by passing an argument to the `RS.impute()` function. In R:

```
R> data(DoubleWell)
R> x <- DoubleWell
R>
R> # Define the drift function:
R> mu <- function(X,t,theta)
+  {
+     theta[1]*X*(theta[2] + theta[3]*sin(0.5*pi*t) - X^2)
+  }
R>
R> # Define starting parameters for the imputation procedure:
R> burns   <-  10000                    # Number of updates to burn
R> updates <-  50000                    # Number of updates
R> theta   <-  c(5, 5, 5, 5)            # Starting parameters
R> sds     <-  c(0.2, 0.2, 0.2, 0.03)   # Proposal standard deviations
R> m       <-  25                       # Imputation resolution
R>
```

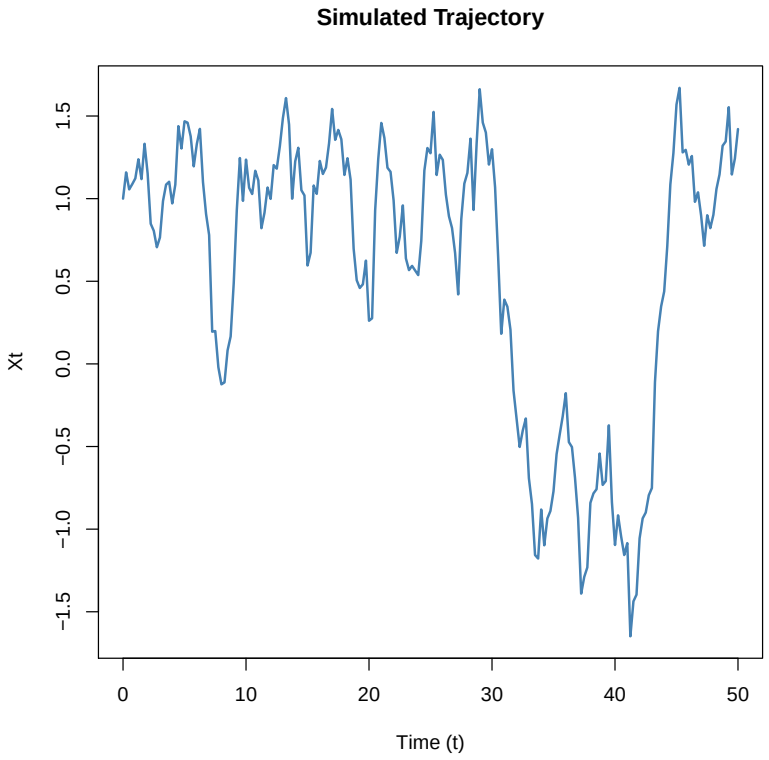


FIGURE 1.5.3: Simulated trajectory of Equation 1.5.5 sampled at equispaced time points over time. R code: Supplementary materials, Section 1.10.

```
R> res <- RS.impute(X = x$X_t, time = x$t, m, theta, sds, diff.type = 1, burns,
    updates)
```

Here, the term `diff.type = 1` is used to specify the diffusion coefficient $\sigma(X_t, t) = \theta_5$ for the model.

By passing values to the arguments of `RS.impute()` giving the time series (`X` and `time`), imputation resolution (`m`), starting parameters (`theta`), symmetric proposal standard deviations (`sds`), and dimensions for the MCMC chain (`burns` and `updates`), the data-imputation scheme will start running. Here, the argument `diff.type = 1` specifies a diffusion structure of the form $\sigma(X_t, t) = \theta_4$ where the index of the parameter $\theta_{\cdot}$ is calculated by counting the number of parameters in the drift function and adding one. Consequently, the dimension of the starting parameter vector and vector of proposal standard deviations have to reflect this (i.e., `length(theta)` $= 4$). At the time of this writing, `RS.impute()`

supports arguments `diff.type = 1`, `diff.type = 2`, and `diff.type = 3` corresponding to diffusion structures of the form $\sigma(X_t, t) = \theta_.$, $\sigma(X_t, t) = \theta_.\sqrt{X_t}$, and $\sigma(X_t, t) = \theta_. X_t$ respectively, allowing for the specification of simple forms of state-dependent volatility. Note that, despite the constraints of reducibility, the set of available specifications can easily be extended to more general cases. Upon further development, such specifications will be included accordingly.

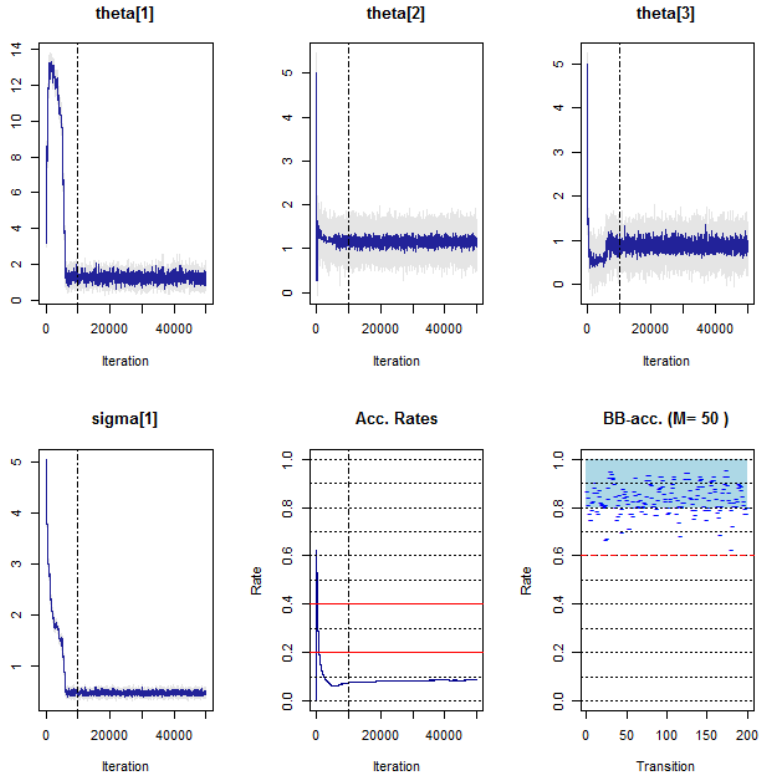


FIGURE 1.5.4: Trace plot output along with rejected proposals (light gray) for the parameter chains calculated using an imputed resolution of $m = 50$ points per transition horizon. The vertical dashed lines indicate the end of the burn-in period. In addition, a trace plot of the acceptance rate of the RWMH algorithm is drawn with guide lines ranging from 0% to 100% in increments of 10% (20% and 40% highlighted in red). Finally, a plot of the average acceptance rate for imputations on each transition horizon is indicated. 60% guide (dashed red) is indicated in addition to a target range of $\geq 80\%$. R code: Supplementary materials, Section 1.10.

After running the data-imputation algorithm, the function will create a trace plot of the parameter chains and the acceptance rate for the Metropolis updates

(Figure 1.5.4). In addition, the average acceptance rate for each individual imputed bridge is calculated and indicated. The latter plot can be used to identify sections of the time series where the sampling rate for imputed bridges is low. This may be due to factors such as low data resolution or when the dynamics of the Brownian bridges are highly disparate with that of the model over the applicable transition horizons. For example, if a Brownian bridge is used to impute a missing trajectory for a sinusoidal model over a large transition horizon, one can expect the acceptance rate to be low. To see how the algorithm operates, one can pass an argument `imputation.plot = TRUE` to `RS.impute()` which will create a live plot of the imputed trajectories superimposed over the observed trajectory as in Figure 1.5.5.

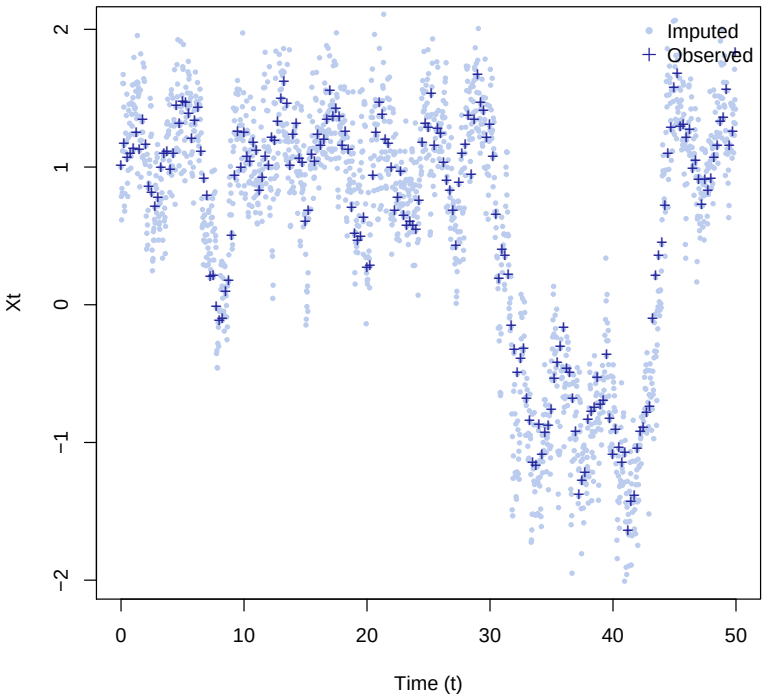


FIGURE 1.5.5: Snapshot of the observed time series (dark blue crosses) and imputed trajectories (light blue dots) captured during the burn-in phase of the imputation procedure. The imputed trajectories can be viewed live by passing the argument `imputation.plot = TRUE` to `RS.impute()`. R code: Supplementary materials, Section 1.10.

Subsequently, in order to calculate parameter estimates from the imputation procedure, one can pass the model object to the `RS.estimates()` function:

```
R> RS.estimates(res, thin = 100, burns, corrmat = TRUE)
```

```
$estimates
         Estimate Lower_CI Upper_CI
theta[1]    1.274    1.007    1.570
theta[2]    1.163    1.063    1.255
theta[3]    0.868    0.719    1.024
theta[4]    0.457    0.416    0.499

$corrmat
         theta[1] theta[2] theta[3] theta[4]
theta[1]     1.00     0.32    -0.45     0.41
theta[2]     0.32     1.00    -0.47     0.03
theta[3]    -0.45    -0.47     1.00    -0.18
theta[4]     0.41     0.03    -0.18     1.00
```

Comparing the output to the true parameter set, the calculated estimates are reasonably close to the actual values. By repeating the experiment for numerous simulated trajectories, it can be seen that the algorithm typically produces accurate parameter estimates.

1.5.3.3 *E. huxleyi* data revisited

Using the **DiffusionRimp** package, one can easily fit diffusion models to real-world datasets. Indeed, the analysis of Section 1.4.1 was conducted using the imputation routines in conjunction with the pseudo-AIC functions in a similar fashion to the previous example. To demonstrate how this was achieved, we revisit the dataset. To start, we fit the SDE:

$$dX_t = \left(\theta_1 X_t(\theta_2 - X_t) + \theta_3 \sin\left(\frac{2\pi t}{12}\right) + \theta_4 \cos\left(\frac{2\pi t}{12}\right)\right)dt + \theta_3 X_t dB_t \quad (1.5.6)$$

to the scaled temperature time series (previously denoted $T_t^\star$). In R:

```
R> m       <- 50
R> updates <- 150000
R> burns   <- 50000
R> theta   <- c(0.5, 10, -0.5, -0.5, 0.5)
R> sds     <- c(0.03, 0.09, 0.52, 0.38, 0.01)
R>
R> mu <- function(X,t,theta)
+ {
+   theta[3]*sin(2*pi*0.08333333*t)+theta[4]*cos(2*pi*0.08333333*t)+
+   theta[1]*theta[2]*X-theta[1]*X*X
```

```
+ }
R> model_5 <- RS.impute(X, time, m, theta, sds, diff.type = 3, burns, updates)
```

where `X` is a vector of scaled temperature time series values and `time` is the corresponding numerical vector of times at which the observations are made. Here, the first observation time is set to zero and time is measured in months. Using the `RS.estimates()` function, we can compare the resulting parameter estimates to that of Table 1.4.2:

```
R> # Calculate par. estimates and correlation matrix from the MCMC output:
R> res <- RS.estimates(model_5, thin = 200, burns = burns, corrmat = TRUE)
R> res
```

```
$estimates
         Estimate Lower_CI Upper_CI
theta[1]    0.170    0.123    0.224
theta[2]    9.796    9.672    9.914
theta[3]   -2.347   -3.486   -1.479
theta[4]   -3.393   -4.226   -2.719
theta[5]    0.118    0.108    0.131

$corrmat
         theta[1] theta[2] theta[3] theta[4] theta[5]
theta[1]     1.00     0.14    -0.98    -0.96     0.51
theta[2]     0.14     1.00    -0.19    -0.13     0.28
theta[3]    -0.98    -0.19     1.00     0.93    -0.47
theta[4]    -0.96    -0.13     0.93     1.00    -0.43
theta[5]     0.51     0.28    -0.47    -0.43     1.00
```

Naturally, the next step in the process is to calculate a pseudo-AIC statistic for the model. This can be achieved using the `MOL.aic()` function. The `MOL.aic()` function applies the method of lines in order to approximate the transitional density over successive transition horizons of a given time series. By iteratively evaluating the density approximation, an approximate likelihood function can then be calculated based on the structure of Equation 1.1.9, from which an approximate AIC value can be calculated. Given a parameter vector, we can evaluate the pseudo-AIC for Equation 1.5.6 under the scaled temperature data in R using the code:

```
R> delt <- 0.001    # Step size
R> N    <- 201      # Number of nodes on the lattice
R> lims <- c(2, 15) # Spatial limits for the lattice
R>
R> mu     <- function(X,t,theta)
+ {
+   theta[3]*sin(2*pi*0.08333333*t) + theta[4]*cos(2*pi*0.08333333*t)+
+   theta[1]*theta[2]*X - theta[1]*pow(X,2)
```

```
+ }
R> model5.aic <- MOL.aic(X, time, delt, lims, N, theta= res$estimates[,1], diff.
    type = 3)
R> model5.aic$AIC
```

```
[1] 310.6262
```

Here, the variables `delt`, `N`, and `lims` respectively give the time step size, the number of nodes in the spatial discretization and the limits of the spatial discretization for the method of lines. That is, the transitional density will be approximated on the domain $[2, 15]$ using a 201-dimensional system of ODEs which is solved numerically using a time step of 0.001 time units for each pair of successive observations. Although, we pass the fixed lattice implied by these parameters to the function, the `MOL.aic()` function will shift the lattice appropriately for each observation before during calculation of the likelihood function. Figure 1.5.6 illustrates graphical output for pseudo AIC calculation: At each step of the calculation a plot is drawn indicating on which transition horizon the density approximation is calculated as well as the resulting density approximation in conjunction with contribution to the likelihood (i.e., the value of the density at the subsequent observation).

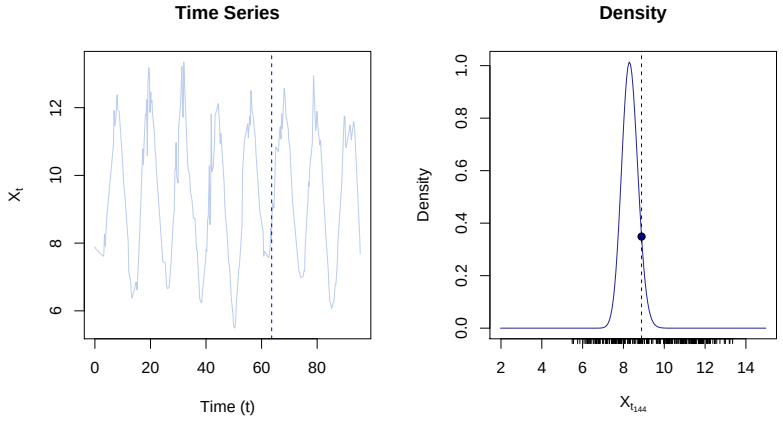


FIGURE 1.5.6: Snapshot of the pseudo-AIC calculation for a scalar diffusion model. `MOL.aic()` evaluates the transitional density (right) under the parameter vector `theta` for each observed transition horizon (left) (i.e., at succesive values of `X` – as indicated by the vertical line super imposed on the transitional density) using the method of lines in order to approximate the AIC of a given model. R code: Supplementary materials, Section 1.11.

Although fitting multivariate diffusions is somewhat more involved, the procedure is more or less similar. For example, following from (bivariate) Model 4 in Section 1.4.1 we can fit the SDE:

$$\begin{aligned} dX_t &= (\theta_6 X_t(\theta_7 - P_t) + \theta_7 Y_t)dt + \theta_8\sqrt{X_t}dB_t^{(1)} \\ dY_t &= \left(\theta_1 Y t(\theta_2 - Y_t) + \theta_3 \sin\left(\frac{2\pi t}{12}\right) + \theta_4 \cos\left(\frac{2\pi t}{12}\right)\right)dt + \theta_9 Y_t dB_t^{(2)}, \end{aligned} \tag{1.5.7}$$

where X_t denotes log-scaled abundances for E. *huxleyi* and Y_t denotes the scaled temperature time series, using the code:

```
R> mu1   <- function(X,Y,t,theta){theta[5]*X*(theta[6] - X) + theta[7]*Y}
R> mu2   <- function(X,Y,t,theta)
+  {
+    theta[1]*theta[2]*Y - theta[1]*Y^2 +
+    theta[3]*sin(0.5235988*t) + theta[4]*cos(0.5235988*t)
+  }
R>
R> theta <- c(0.8, 10, -1, -0.5, 0.5, 7, 0.01 ,1.5, 0.3)
R> sds   <- c(0.03, 0.07, 0.52, 0.40, 0.04, 0.77, 0.14, 0.07, 0.01)/3
R> model_4 <- BiRS.impute(X, time, m, theta, sds, diff.type = c(2, 3), burns,
    updates)
R>
R> res <- RS.estimates(model_4, thin = 200, burns = burns, corrmat = FALSE)
R> res
```

```
         Estimate Lower_CI Upper_CI
theta[1]    0.168    0.129    0.212
theta[2]    9.785    9.653    9.944
theta[3]   -2.352   -3.238   -1.513
theta[4]   -3.384   -3.995   -2.807
theta[5]    2.337    2.196    2.511
theta[6]    8.044    6.926    9.266
theta[7]    0.447    0.175    0.699
theta[8]    0.909    0.843    0.987
theta[9]    0.117    0.108    0.130
```

From this, we can once again calculate the corresponding pseudo-AIC statistic using the method of lines. In this case, we use the `BiMOL.aic()` function:

```
R> mu1   <- function(X,Y,t,theta){theta[5]*(theta[6] - X) + theta[7]*Y}
R> mu2   <- function(X,Y,t,theta)
+  {
+    theta[1]*theta[2]*Y - theta[1]*pow(Y,2) +
+    theta[3]*sin(0.5235988*t) + theta[4]*cos(0.5235988*t)
+  }
R>
R> delt  <- 1/1000    # Step size
R> N     <- 121       # Number of nodes on the lattice in each dimension
```

```
R> xlims <- c(2, 17)  # X-limits for the lattice
R> ylims <- c(2, 17)  # Y-limits for the lattice
R> model4.aic <- BiMOL.aic(X, time, delt, xlims, ylims, N, theta = res[ ,1],
    diff.type = c(2, 3))
R>
R> model4.aic$AIC
```

```
[1] 969.2975
```

Here, we use a 121×121 system of ODEs on the region $[2, 17] \times [2, 17]$ in order to approximate the transitional density over each transition horizon. Solving this system of ODEs numerically, we calculate the contribution to the likelihood at successive observations by interpolating the density value from those evaluated at nodes on the lattice which surround the observation. Figure 1.5.7 illustrates the procedure for a single observation under the estimated parameter set.

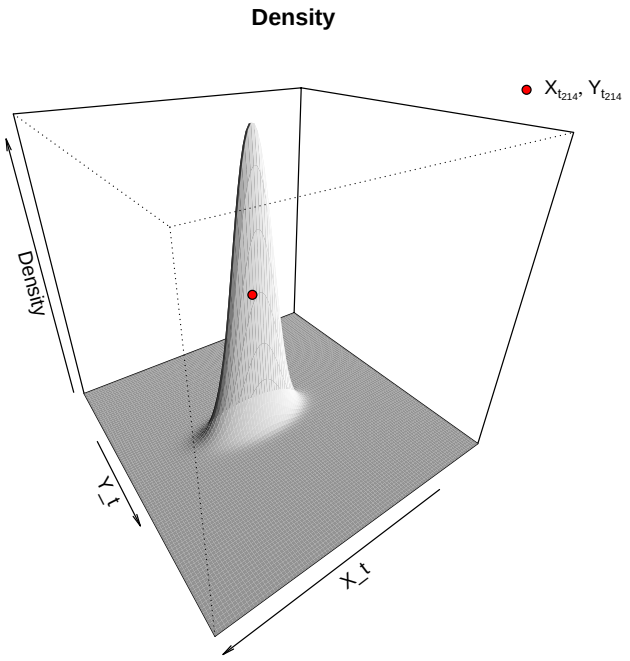


FIGURE 1.5.7: Snapshot of the pseudo-AIC calculation for a bivariate diffusion model. As in the scalar case, `BiMOL.aic()` evaluates the transitional density under the parameter vector `theta` for each observed transition horizon using the method of lines in order to approximate the AIC of a given model. At each iteration of the calculation, the contribution to the likelihood is calculated by interpolating from the approximate transitional density values at nodes which surround the current observation (red dot). R code: Supplementary materials, Section 1.11.

1.6 Chapter summary

We focus on combining numerical techniques for solving the Kolmogorov equations with the data-imputation scheme in order to efficiently perform inference on non-linear diffusion models. Using these techniques we are able to fit various models to population and temperature time series of the species *E. huxleyi* and subsequently perform model selection by calculating pseudo-AIC statistics. Based on the resulting model dynamics we establish dependence between population growth and water temperature. Using the probabilistic evolution of the model

process as a basis we develop a temporal risk measure of the bloom potential for the species in order to calculate phases in which blooms are likely to occur. By deriving a PDE for the marginal transitional density we develop a computationally efficient strategy for evaluating the bloom potential function whilst accounting for parameter uncertainty. With an aim to widening the scope of the analysis conducted here, we develop software routines that implement the techniques outlined here and construct **DiffusionRimp** – a package for performing inference and analysis on scalar and bivariate non-linear diffusion processes in R using the data-imputation scheme and the method of lines. Using the **DiffusionRimp** package, the methodology can be applied to complex problems such as the ecological time series analysed here without having to deal with the challenge of having to set up the data-imputation algorithm or the method of lines (although the latter is significantly less intricate). We show how the package can be used to analyse highly non-linear diffusions and illustrate by way of a simulated dataset the workings of the data-imputation scheme.

In order to plot the way forward, we identify some shortcomings of with the methodology outlined here. The principal motivation behind the use of the data-imputation scheme is the efficiency with which one can calculate parameter estimates. Although the data-imputation scheme is very effective in this regard, the scope of the scheme is not completely general. For example, it relies on the principal of reducibility for the imputation to be feasible. As such the model space is somewhat restricted with respect to the diffusion specification and thus excludes a class of models that may otherwise be useful for modelling purposes in practice, for example, non-linear stochastic volatility models. Although it is possible to apply the data-imputation procedure in special cases of irreducible models (see for example Kalogeropoulos *et al.* (2011), where the method is applied to a stochastic volatility model), such a strategy requires significant tailoring of the algorithm. Another perhaps more subtle consideration with respect to the imputation scheme relates to approximating missing trajectories with Brownian bridges. In practice, using standard Brownian bridges to impute missing trajectories suffices for most applications. However, depending on the model specification and the resolution of the dataset, one may see low acceptance rates for imputation updates over transitions where the imputed bridges approximate the dynamics of the diffusion model poorly. For example, if on a given transition horizon, the process is predicted to oscillate significantly due to some drift time-dependency, the imputed trajectories may have difficulty replicating this behaviour leading to low acceptance rates for the imputed trajectories. Thus, although the paths that are imputed may still be accurate, low acceptance rates may affect the quality of the parameter chains calculated using the Metropolis-Hastings algorithm. Although this can be remedied using more complex Bridge imputation structures

this may diminish the performance of the algorithm since such schemes are not as easily vectorized.

In the chapter that follows, we address the limitations of the methodology presented here by developing a class of models with an aim to maximising computational efficiency under a transition density approximant whilst still making it possible to conduct analysis on time-inhomogeneous non-linear diffusion models.

Chapter 2

Generalised Quadratic Diffusions in the Software Environment

2.1 Introduction

Although numerous innovative strategies have been developed for performing inference and analysis on non-linear diffusion models, the mathematics that underpins these methods can be daunting and often require a good understanding of technical material from outside the discipline of pure statistics or the context of the desired application. Indeed, the use of these strategies has mostly been limited to fields like mathematical finance and physics which exhibit some theoretical overlap with stochastic calculus. Whilst the application of non-linear diffusion models has propagated in recent years, the growth has been somewhat stunted by the computational complexity of the existing methodologies – at least from the perspective of researchers in non-statistical or mathematical fields. Consequently, demand has arisen for software that makes non-linear diffusion models more accessible in less mathematically focused sciences. In response to this, we develop methodology for the analysis of diffusion models specifically with an aim to implementation in the software environment.

Currently, there are a number of excellent R packages for the analysis of diffusion processes. Although there are some overlapping points of interest, these packages cover a number of topics. For example, the **Sim.DiffProc** (Boukhetala *et al.*, 2011; Guidoum and Boukhetala, 2015) provides a comprehensive list of advanced routines for the simulation of scalar and multivariate diffusion processes in both

Itô and Stratonovich form. This includes the simulation of bridge-processes and first passage times. In addition, **Sim.DiffProc** also provides useful routines for fitting diffusion models using pseudo-likelihood methods based on small time approximations of the transition density. Another package that covers the simulation and estimation of diffusion models is the **sde** (Iacus, 2015) package (see also Iacus (2009)). In addition to providing built-in functions for the evaluation of transitional densities for well-known analytically tractable diffusion models, the package covers (among others, some of which overlap with the **Sim.DiffProc** package) Hermite series approximations to the likelihood, as well as non-parametric techniques for estimating the drift and diffusion coefficients of a diffusion model. Packages that cover more specific topics are also available: For example, the **fptdApprox** (Román-Román *et al.*, 2014) covers advanced techniques for numerically evaluating the first passage time density for diffusions with analytically tractable transition densities. There are also a number of packages with minor relation to diffusion processes such as the **fOptions** (Team *et al.*, 2015), **RQuantLib** (Eddelbuettel and Nguyen, 2015), and **NMOF** (Gilli *et al.*, 2011) packages, which touch on various financial applications of stochastic differential equations and well-known diffusion models.

With this literature in mind, we develop the **DiffusionRgqd** (Pienaar and Varughese, 2015a) package – a collection of tools for performing inference and analysis on a class of quadratic diffusion processes. in contrast to the **DiffusionRimp** package developed in the previous chapter, the routines developed here are centred around a computationally efficient numerical method for calculating accurate approximations to the transitional densities of polynomial diffusion processes. By using dynamic algorithm construction techniques, these routines construct solutions tailored to the model specification without requiring any mathematical input from the user over and above the model specification. As such we are able to optimize the computational efficiency of routines in the package whilst providing minimal constraints on the model specification within the applicable class of diffusion models. By separating the user from the underlying mathematical technicalities, the package provides access to a suitably general class of diffusions whilst demanding only basic programming skills and a graduate level understanding of likelihood based inference procedures. In this way, we develop a more accessible, self-contained package for performing inference and analysis on non-linear diffusions as compared to the routines developed earlier in this thesis.

2.2 The generalised quadratic diffusion (GQD) class

A common problem that arises when developing mathematical software is designing a mechanism for translating the abstract lexical structures of mathematics into usable syntax for the programming environment. Traversing this language barrier has become a science in itself, resulting in numerous schools of thought on what constitutes optimal interface mechanics. Indeed, in statistical fields such as generalised linear modelling, a very natural structural interface has evolved in the underlying mathematics that has been emulated in software design, with models being defined syntactically in much the same way as the written language. Although research papers on diffusion processes usually reiterate some mathematical grammar like the ubiquitous shorthand for diffusion processes where the process is expressed in differential form (i.e., a stochastic differential equation as opposed to an integral equation), a unified framework for specifying a diffusion model in the context of inference is less established. This is due in part to the diversity of methods and the classes of diffusions to which they apply. These classes are often defined by precluding assumptions about the model process such as time-homogeneity and/or reducibility. In the absence of such universally accepted 'language', we have adopted a simple design suitable for the methodology we have chosen whereby the analysis is constrained to scalar and bivariate diffusions that have, at most, second order polynomial terms in the drift and diffusion coefficients of the model process. We term these the generalised quadratic diffusions (GQDs) which are characterized by the SDEs

$$dX_t = [g_0(t) + g_1(t)X_t + g_2(t)X_t^2]dt + \sqrt{q_0(t) + q_1(t)X_t + q_2(t)X_t^2}dB_t \quad (2.2.1)$$

and

$$d\begin{bmatrix} X_t \\ Y_t \end{bmatrix} = \begin{bmatrix} \sum_{i+j\leq 2} a_{ij}(t)X_t^i Y_t^j \\ \sum_{i+j\leq 2} b_{ij}(t)X_t^i Y_t^j \end{bmatrix} dt + \begin{bmatrix} \sigma_{11}(X_t, Y_t, t) & \sigma_{12}(X_t, Y_t, t) \\ \sigma_{21}(X_t, Y_t, t) & \sigma_{22}(X_t, Y_t, t) \end{bmatrix} d\begin{bmatrix} B_t^{(1)} \\ B_t^{(2)} \end{bmatrix} \quad (2.2.2)$$

with

$$\boldsymbol{\sigma}(X_t, Y_t, t)\boldsymbol{\sigma}'(X_t, Y_t, t) = \begin{bmatrix} \sum_{i+j\leq 2} c_{ij}(t)X_t^i Y_t^j & \sum_{i+j\leq 2} d_{ij}(t)X_t^i Y_t^j \\ \sum_{i+j\leq 2} e_{ij}(t)X_t^i Y_t^j & \sum_{i+j\leq 2} f_{ij}(t)X_t^i Y_t^j \end{bmatrix}, \quad (2.2.3)$$

for the scalar and bivariate case respectively. For both scalar and bivariate GQDs, the indices of the coefficients are formulated to reflect powers of the diffusion process in the model, thus providing a grammatical link between the coefficients and variables contained in the model. Within this framework, a model can thus be specified simply by including the desired coefficients of the relevant GQD. For

example, a time-inhomogeneous stochastic volatility model with SDE

$$d\begin{bmatrix} X_t \\ Y_t \end{bmatrix} = \begin{bmatrix} \theta_1(\theta_2 - X_t) \\ \theta_4(\theta_5 + \theta_6 \sin(2\pi t) - Y_t) \end{bmatrix} dt + \begin{bmatrix} \theta_3\sqrt{Y_t} & 0 \\ 0 & \theta_7\sqrt{Y_t} \end{bmatrix} d\begin{bmatrix} B_t^{(1)} \\ B_t^{(2)} \end{bmatrix}, \quad (2.2.4)$$

can easily be be identified within the GQD framework by matching the coefficients of terms in the target model to the those in the general model:

$$\begin{aligned} a_{00}(t) &= \theta_1\theta_2 \\ a_{10}(t) &= -\theta_1 \\ b_{00}(t) &= \theta_4(\theta_5 + \theta_6 \sin(2\pi t)) \\ b_{01}(t) &= -\theta_4 \\ c_{01}(t) &= \theta_3^2 \\ f_{01}(t) &= \theta_7^2. \end{aligned} \quad (2.2.5)$$

In R, the lexical structure of equations 2.2.1 and 2.2.2 can then easily be replicated by defining functions with names that reflect those of the coefficients of the target model. For example, using the coefficients in Equation 2.2.5:

```
R> # Assign values to the parameter vector:
R> theta <- c(1,5,1,0.2,1,1,0.2)
R>
R> # Define the model:
R> a00 <- function(t){theta[1] * theta[2]}
R> a10 <- function(t){-theta[1]}
R> c01 <- function(t){theta[3] * theta[3]}
R> b00 <- function(t){theta[4] * (theta[5] + theta[6] * sin(2 * pi * t))}
R> b01 <- function(t){-theta[4]}
R> f01 <- function(t){theta[7] * theta[7]}
R>
R> # Now call some function from the DiffusionRgqd package:
R> BiGQD.density(Xs, Ys, Xt, Yt, s, t, delt)
```

By replicating this framework in the syntax of the **DiffusionRgqd** package, one can easily define and distinguish between various models in a single workspace by using the subscripts of the coefficients as visual cues for the model specification as opposed to having to identify models based on long expressions for the drift and diffusion terms. Indeed, given that we have allowed the models to have arbitrary time dependencies, such expressions would most likely run over multiple lines making them somewhat illegible. Interface considerations aside, placing constraints on the complexity of the state dependence in the drift and diffusion coefficients circumvents a number of mathematical subtleties that arise when attempting to approximate the transitional densities of models with higher order

non-linear drift and diffusion terms. The generalised quadratic framework thus enables us to set up a modelling 'sandbox' wherein robust and accurate numerical approximations of the transition density can be calculated whilst providing an intuitive interface for the software environment.

2.3 Approximating the transitional density of a GQD

2.3.1 Computationally efficient approximation of the transitional density

Re-iterating from Section 1.1, the probabilistic evolution of a k-dimensional diffusion process $\mathbf{X}_t = \{X_t^{(1)}, X_t^{(2)}, \ldots, X_t^{(k)}\}'$ with SDE:

$$\mathbf{dX_t} = \boldsymbol{\mu}(\mathbf{X}_t, t)dt + \boldsymbol{\sigma}(\mathbf{X}_t, t)\mathbf{dB}_t, \tag{2.3.1}$$

is given by the solution of the Kolmogorov forward equation:

$$\begin{aligned}\frac{\partial f(\mathbf{X}_t|\mathbf{X}_s)}{\partial t} = &-\sum_{i=1}^{k} \frac{\partial}{\partial X_t^{(i)}}\left[\mu_i(\mathbf{X}_t, t) f(\mathbf{X}_t|\mathbf{X}_s)\right] \\ &+ \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k} \frac{\partial^2}{\partial X_t^{(i)} \partial X_t^{(j)}}\left[\gamma_{ij}(\mathbf{X}_t, t) f(\mathbf{X}_t|\mathbf{X}_s)\right],\end{aligned} \tag{2.3.2}$$

with the initial condition $f(\mathbf{x}|\mathbf{X}_s) = \delta(\mathbf{x} - \mathbf{X}_s)$, where

$$\delta(\mathbf{y}) = \begin{cases}\infty & \text{if } \mathbf{y} = \mathbf{0}, \\ 0 & \text{otherwise.}\end{cases} \tag{2.3.3}$$

The intuition behind Equation 2.3.2 is that the transition density at time s starts from an infinite point mass at the initial condition $\mathbf{X}_s$ (since this point is occupied with certainty) and subsequently propagates mass over the state space as time increases. The behaviour of this probability flow is dictated by the drift and diffusion differential terms on the right hand side of Equation 2.3.2. When the drift and/or diffusion terms are time dependent, the probability current in the state space may increase, contract or oscillate with time. For example, Figure 2.3.1 illustrates the evolution of the transitional density of a time-inhomogeneous scalar diffusion model, whereby both the drift and diffusion terms are sinusoidal. The solution of Equation 2.3.2 thus gives the probabilistic evolution of Equation 2.3.1 and forms the starting point of probability based analysis of a given diffusion

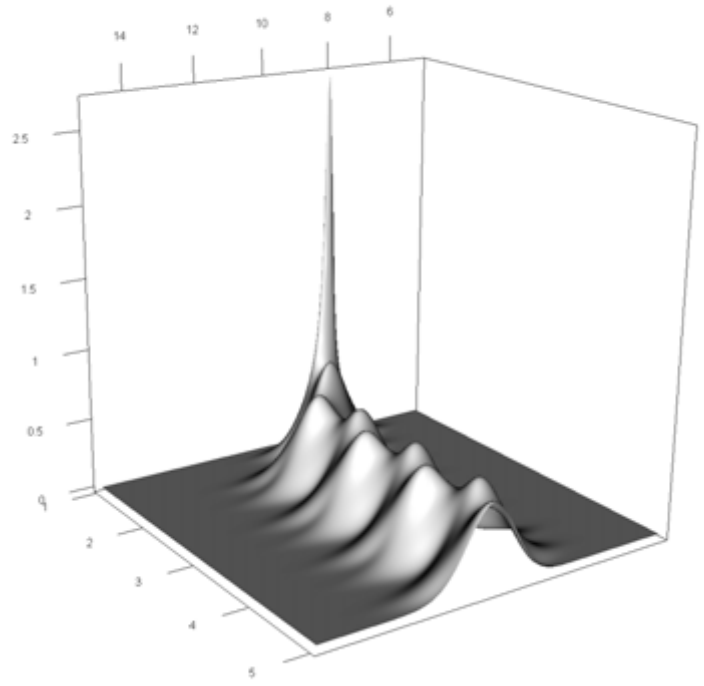


FIGURE 2.3.1: The approximate transitional density of a scalar diffusion process with drift $\mu(X_t, t) = 2(10 + \sin(2\pi(t - 0.5)) - X_t)$ and diffusion $\sigma(X_t, t) = \sqrt{0.25(1 + 0.75\sin(4\pi t))X_t}$ with initial value $X_1 = 8$. This surface was generated using the `GQD.density()` function in the **DiffusionRgqd** package. R code: Supplementary materials, Section 2.1.

process. Although partial differential equations (PDEs) with singular initial conditions such as Equation 2.3.2 present various difficulties from a computational perspective, the notion of a continuously evolving probability density function means that diffusion models can capture the dynamics of continuously evolving real-world phenomena in a very natural way. Since Equation 2.3.2 is not analytically tractable in general, one has to resort to numerical techniques in order to calculate the transitional density. Using the method of lines, we can calculate accurate approximations to the transitional density over arbitrarily large time horizons. The primary drawback of this strategy relates to its computational efficiency in practical applications where we need to evaluate quantities such as the likelihood function for a diffusion model of a time series. As such, 'brute-force' numerical strategies are not ideal from the perspective of developing software

applications. That said, excellent methods for calculating analytical approximations of the transition density do exist. For example, Aït-Sahalia (2008) derive accurate short-horizon approximations to Equation 2.3.2 based on a Hermite series that can accommodate non-linearities in the drift and diffusion of the model. Huang (2011) applies Wagner-Platen expansions (see Platen, 1999) to the conditional moments of a target diffusion which can then be plugged into a surrogate density in order to yield a short-horizon approximation to the transition density. The principal benefit of these strategies is that, since the approximation is formulated analytically, no numerical overhead is incurred beyond evaluating the elements of the approximation. Collectively, these methods produce desirable results for a wide variety of non-linear diffusions, however, the aim of the present research is to develop software that can handle inference at sample resolutions that may be too sparse for analytical short-horizon expansions whilst being computationally feasible in a non-parallel computing environment. Furthermore, as will be seen later, alternative applications of the transition density may require accurate approximations over very large transition horizons. For these purposes we adopt the cumulant truncation procedure for diffusion processes developed by Varughese (2013), wherein the transition density can be approximated accurately and efficiently over arbitrarily large transition horizons for a suitably general class of non-linear diffusion models. The scheme aims to encapsulate information about the trajectory of the transitional density in a finite system of ordinary differential equations that govern the evolution of the cumulants of the process as opposed to dealing with the Kolmogorov equation directly. Assuming that a sufficient amount of information is contained within these statistics, we may subsequently obtain accurate approximations of the transitional density by way of a surrogate density. This strategy has been applied with great success to population models by Marion *et al.* (2000) and Varughese and Fatti (2008), and scalar and bivariate stochastic epidemic models by Krishnarajah *et al.* (2005) and Krishnarajah *et al.* (2007), whereby the authors derive moment equations for various model types and make use of a moment closure approximation in order to accurately approximate the distribution of the process. Indeed, diffusion processes share various attributes with such processes, and many of the attractive features of the methodology readily carries over in its application to diffusion models.

In the sections that follow, we detail the cumulant truncation procedure as applied to generalised quadratic diffusion models. We detail the derivation of mathematical elements for this class of processes and outline the mechanics of calculating accurate approximations to the transitional density.

2.3.2 Deriving cumulant equations for GQDs

We begin by outlining the scheme for scalar diffusions and then expand to the bivariate case. For a scalar diffusion process with SDE

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dB_t, \tag{2.3.4}$$

let $M(X_t, t) = E[\exp(\alpha X_t)]$ denote the moment generating function (MGF) of X_t. Then it can be shown that $M(x, t)$ satisfies the partial differential equation (Varughese, 2013):

$$\frac{\partial}{\partial t}M(\alpha, t) = \alpha\mu\left(\frac{\partial}{\partial\alpha}, t\right)M(\alpha, t) + \frac{1}{2}\alpha^2\sigma^2\left(\frac{\partial}{\partial\alpha}, t\right)M(\alpha, t), \tag{2.3.5}$$

where $\mu(\frac{\partial}{\partial\alpha}, t)$ and $\sigma^2(\frac{\partial}{\partial\alpha}, t)$ are differential operators on $M(\alpha, t)$. That is, when $\mu(X_t, t)$ and $\sigma^2(X_t, t)$ contain integer powers of X_t, we may find a PDE for the MGF in terms of derivatives with respect to α. For example, let $\mu(x, t) = A + Bx + Cx^2$ and $\sigma^2(x, t) = D^2$ then:

$$\frac{\partial}{\partial t}M(\alpha, t) = \alpha\left[A + B\frac{\partial}{\partial\alpha} + C\frac{\partial^2}{\partial\alpha^2}\right]M(\alpha, t) + \frac{1}{2}\alpha^2D^2M(\alpha, t). \tag{2.3.6}$$

Now, let $u_j(t) = E(X_t^j)$ and

$$M(\alpha, t) = \sum_{j=0}^{\infty}\frac{\alpha^j u_j(t)}{j!}. \tag{2.3.7}$$

Then we may easily derive a system of ODEs for the non-central moments of a diffusion process by plugging Equation 2.3.7 into Equation 2.3.6 and equating the α coefficients on the LHS and RHS of the resulting equations. In our example problem, we would arrive at the system:

$$\frac{\partial}{\partial t}u_j(t) = j(A + Bu_j(t) + Cu_{j+1}(t)) + D^2\frac{j(j-1)}{2}u_{j-2}(t)\mathbb{1}_{j\geq 2}, \quad \forall j \geq 1; \tag{2.3.8}$$

where $\mathbb{1}_A = \text{Ind}(A)$. Note however that Equation 2.3.8 implies that an infinite dimensional system of non-central moments are required in order for the system to be determinate. The dimensionality of the system is caused by the inclusion of the quadratic term in the drift. That is, in order to evaluate the j-th non-central moment, we require knowledge of the trajectory of the $(j+1)$-th and subsequent non-central moments. Thus, when we look at a finite set, say the first d, non-central moments of a non-linear polynomial diffusion, we observe a substantial amount of *leakage* of information into the higher-order moments. In order to deal

with this leakage we may alternatively consider the behaviour of the cumulants of such diffusion processes. Let

$$K(\alpha, t) = \sum_{j=1}^{\infty} \frac{\alpha^j \kappa_j(t)}{j!} \tag{2.3.9}$$

be the cumulant generating function (CGF) where $\kappa_j(t)$ denotes the j-th cumulant of the process at time t. The cumulants $\kappa_j(t)$ are then defined in relation to the non-central moments by:

$$\sum_{j=1}^{\infty} \frac{\alpha^j \kappa_j(t)}{j!} = \log\left(1 + \sum_{j=1}^{\infty} \frac{\alpha^j u_j(t)}{j!}\right). \tag{2.3.10}$$

By making use of the relationship between the MGF and the CGF:

$$\frac{1}{M(\alpha, t)} \frac{\partial M(\alpha, t)}{\partial v} = \frac{\partial K(\alpha, t)}{\partial v} \tag{2.3.11}$$

and the recursive relation

$$\frac{1}{M}\left(\frac{\partial^{(r+1)} M}{\partial v^{(r+1)}}\right) = \left[\frac{\partial K}{\partial v}\right]\left[\frac{1}{M}\left(\frac{\partial^{(r)} M}{\partial v^{(r)}}\right)\right] + \frac{\partial}{\partial v}\left[\frac{1}{M}\left(\frac{\partial^{(r)} M}{\partial v^{(r)}}\right)\right] \tag{2.3.12}$$

for $r = 1, 2, \ldots$, we may derive a similar system of ODEs for the cumulants of a diffusion process. That is, by dividing both sides of Equation 2.3.5 by $M(\alpha, t)$ we may use Equation 2.3.12 in order to derive a PDE for the cumulant generating function. Thus, for scalar GQDs with drift and diffusion coefficients $\{g_i(t) : i = 0, 1, 2\}$ and $\{q_i(t) : i = 0, 1, 2\}$ respectively, we have:

$$\frac{\partial}{\partial t} K(\alpha, t) = \alpha \sum_{i=0}^{2} g_i(t) \frac{1}{M}\left(\frac{\partial^{(i)} M}{\partial u^{(i)}}\right) + \frac{1}{2}\alpha^2 \sum_{i=0}^{2} q_i(t) \frac{1}{M}\left(\frac{\partial^{(i)} M}{\partial u^{(i)}}\right), \tag{2.3.13}$$

where the terms $\frac{1}{M}(\frac{\partial^{(i)} M}{\partial u^{(i)}})$ may be expressed in terms of $K(\alpha, t)$ via equations 2.3.11 and 2.3.12.

Then by replacing $K(\alpha, t)$ by the d-th order truncated series

$$K^{(d)}(\alpha, t) = \sum_{j=1}^{d} \frac{\alpha^j \kappa_j(t)}{j!}, \tag{2.3.14}$$

plugging the relevant derivatives of Equation 2.3.14 into Equation 2.3.13 and equating coefficients on the LHS and RHS, one can derive a d-dimensional system of ODEs that describes the approximate evolution of the cumulants

$\{\kappa_j(t) : j = 1, ..., d; t > s\}$ over time. For scalar GQDs, we may derive a closed-form expression for the system of ODEs that govern the evolution of the cumulants. For a d-th order truncation of a scalar GQD, the ODEs that govern the evolution of the truncated cumulants are given by the system:

$$\begin{aligned}\frac{\partial}{\partial t}\kappa_j(t) = {} & g_0(t)\mathbb{1}_{j=1} \\ & + g_1(t)j\kappa_j(t) \\ & + g_2(t)\left(j\kappa_{j+1}(t) + \sum_{r=1}^{j} r\binom{j}{r}\kappa_r(t)\kappa_{j-r+1}(t)\right) \\ & + q_0(t)\mathbb{1}_{j=2} \\ & + q_1(t)\frac{j(j-1)}{2}\kappa_{j-1}(t)\mathbb{1}_{j\geq 2} \\ & + q_2(t)\left(\frac{j(j-1)}{2}\kappa_j(t) + \frac{j}{2}\sum_{r=1}^{j-1} r\binom{j-1}{r}\kappa_r(t)\kappa_{j-1-r+1}(t)\right)\mathbb{1}_{j\geq 2},\end{aligned} \tag{2.3.15}$$

with initial conditions $\kappa_1(s) = X_s$ and $\kappa_j(s) = 0\ \forall j > 1$. The initial conditions follow from the fact that, at time s, the state X_s is occupied with certainty, hence the first moment of the process reflects this position and the zero-valued higher order cumulants reflect the absolute certainty.

The advantage of this route is twofold: Firstly, we can manage the 'leakage' that occurs in the cumulants by assuming that cumulants above a certain order negligible and subsequently set them equal to zero – an assumption that is valid for diffusions of the quadratic class assumed in the present paper. Secondly, the non-central moments usually assume values that are orders of magnitude larger than the cumulants, which may lead to instabilities under numerical evaluation.

Consider now a bivariate diffusion process with SDE:

$$d\begin{bmatrix}X_t \\ Y_t\end{bmatrix} = \begin{bmatrix}\mu_1(X_t, Y_t, t) \\ \mu_2(X_t, Y_t, t)\end{bmatrix} dt + \begin{bmatrix}\sigma_{11}(X_t, Y_t, t) & \sigma_{12}(X_t, Y_t, t) \\ \sigma_{21}(X_t, Y_t, t) & \sigma_{22}(X_t, Y_t, t)\end{bmatrix} d\begin{bmatrix}B_t^{(1)} \\ B_t^{(2)}\end{bmatrix}. \tag{2.3.16}$$

By imposing the drift and diffusion structure of Equation 2.2.2, the corresponding PDE for the bivariate MGF, $M_2(X_t, Y_t, t) = E[\exp(\alpha X_t + \beta Y_t)]$, becomes:

$$\begin{aligned}\frac{\partial}{\partial t}M_2(\alpha,\beta,t) = &\left[\alpha \sum_{i+j\leq 2} a_{ij}(t)\frac{\partial^i}{\partial\alpha^i}\frac{\partial^j}{\partial\beta^j} + \beta \sum_{i+j\leq 2} b_{ij}(t)\frac{\partial^i}{\partial\alpha^i}\frac{\partial^j}{\partial\beta^j}\right]M_2(\alpha,\beta,t)\\ &+ \frac{1}{2}\left[\alpha^2 \sum_{i+j\leq 2} c_{ij}(t)\frac{\partial^i}{\partial\alpha^i}\frac{\partial^j}{\partial\beta^j} + \alpha\beta \sum_{i+j\leq 2} d_{ij}(t)\frac{\partial^i}{\partial\alpha^i}\frac{\partial^j}{\partial\beta^j}\right.\\ &\left.+ \alpha\beta \sum_{i+j\leq 2} e_{ij}(t)\frac{\partial^i}{\partial\alpha^i}\frac{\partial^j}{\partial\beta^j} + \beta^2 \sum_{i+j\leq 2} f_{ij}(t)\frac{\partial^i}{\partial\alpha^i}\frac{\partial^j}{\partial\beta^j}\right]M_2(\alpha,\beta,t).\end{aligned} \tag{2.3.17}$$

By applying the cumulant truncation procedure to the bivariate GQD we can derive in similar fashion a general expression for the system of ODEs that govern the evolution of the cumulants, truncated up to an arbitrary order d. That is by assuming that $\kappa_{ij}(t) = 0 \quad \forall \quad i + j > d$ the truncated CGF is given by:

$$K_2^{(d)}(\alpha,\beta,t) = \sum_{i+j\leq d} \frac{\alpha^i\beta^j}{i!j!}\kappa_{ij}(t). \tag{2.3.18}$$

Now define the retention operator, J_{ij}^{vw}, that operates on α and β:

$$J_{ij}^{vw}\alpha^r\beta^s = \mathbb{1}_{r+v=i,s+w=j}. \tag{2.3.19}$$

Furthermore, let

$$L_{xy} = \sum_{i=0}^{d}\sum_{j=0}^{d} \alpha^{i-x}\beta^{j-y} i^{\underline{x}} j^{\underline{y}}/(i!j!)\kappa_{ij}(t) \tag{2.3.20}$$

where $i^{\underline{x}} = i(i-1)\dots(i-x+1)$ and $j^{\underline{y}} = j(j-1)\dots(j-y+1)$, $\Theta = \{a, b, c, d, e, f\}'$, and define a set of multi-indexes $\lambda = \{(00), (10), (20), (01), (02), (11)\}'$. Let the subscript q denote the q-th entry of the vectors λ and Θ. Then the system of ODEs that govern the evolution of the cumulants of Equation 2.2.2 is given by:

$$\begin{aligned}\frac{\partial}{\partial t}\kappa_{ij}(t) = \sum_q J_{ij}^{\lambda_q}[&(\Theta_q)_{00}\mathbb{1}_{\lambda_q} + (\Theta_q)_{10}L_{10} + (\Theta_q)_{01}L_{01} + (\Theta_q)_{02}[L_{11} - L_{10}L_{01}]\\ &+ (\Theta_q)_{20}[L_{20} - L_{10}L_{10}] + (\Theta_q)_{02}[L_{02} - L_{01}L_{01}],\end{aligned} \tag{2.3.21}$$

with initial conditions $\kappa_{10}(s) = X_s$, $\kappa_{01}(s) = Y_s$ and $\kappa_{ij}(s) = 0$ otherwise. As in the scalar case, the initial conditions for the cumulant equations reflects the fact that the position of the process at the starting coordinate (X_s, Y_s) is known.

Consequently, all second order and higher cumulants have initial conditions equal to zero.

2.3.3 Surrogate densities

By applying the cumulant truncation procedure to SDEs of the generalised quadratic form we may accurately approximate the evolution of the cumulants over arbitrarily large time horizons. However, in order to approximate the transitional density at any given point in the support of the process, we need to carry these cumulants into a probability density function. If the first two or three cumulants contain sufficient information about the probabilistic evolution of the process, standard densities such as the Normal or Gamma distribution may be used. However, for diffusion processes, it often occurs that higher order cumulants are indeed informational. As such we employ a number of density approximations suitable for carrying higher order cumulants into the approximate transitional density. One such approach is to use the saddlepoint approximation. For scalar GQDs under truncation order d, the univariate saddlepoint approximation is given by (Daniels, 1954):

$$\tilde{f}^{(d)}_{SPT}(x_t|X_s) = \frac{1}{\sqrt{2\pi \frac{\partial^2 K^{(d)}}{\partial \alpha^2}(\alpha_0, t)}} \exp\Big(K^{(d)}(\alpha_0, t) - \alpha_0 x_t \Big), \tag{2.3.22}$$

where α_0 solves

$$\frac{\partial K^{(d)}}{\partial \alpha}(\alpha, t) = x_t. \tag{2.3.23}$$

As an alternative to the saddlepoint approximation, we may employ suitable members of the multimodal Pearson system developed by Cobb *et al.* (1983). The system consists of a set of kernel functions that depend on a predetermined number of non-central moments. These kernels serve to extend four principal classes of density, namely the Normal, Gamma, Inverse Gamma and Beta distributions. In addition, each distribution class is characterised by a system of equations which relate the non-central moments to the parameters of each respective kernel. Thus, given a finite set of non-central moments, we may evaluate any density nested within the system by calculating the kernel parameters and plugging them into the appropriate kernel expression. In the present context, we may thus, in similar fashion to the saddlepoint approximation, calculate the transitional density based on the trajectories of the non-central moments of a given diffusion. Formally, given an even number of d non-central moments $\{u_i(t) : i = 1 : d\}$, we re-iterate the expressions of Cobb *et al.* (1983) under the notation used here:

Let

$$\mathbf{A} = \begin{pmatrix} 1 & u_1(t) & \dots & u_{d/2}(t) \\ u_1(t) & u_2(t) & \dots & u_{d/2+1}(t) \\ \vdots & \vdots & \ddots & \vdots \\ u_{d/2}(t) & u_{d/2+1}(t) & \dots & u_d(t) \end{pmatrix}, \tag{2.3.24}$$

and $\boldsymbol{\beta} = \{\beta_1, \beta_2, \dots, \beta_{d/2+1}\}$.

Then the multimodal Normal class is given by:

$$\tilde{f}_N^{(d)}(x_t|X_s) \propto \exp\Big(- \sum_{i=1}^{d/2+1} \beta_i x_t^i / i \Big) \quad \forall \quad x_t \in (-\infty, \infty), \tag{2.3.25}$$

where $\boldsymbol{\beta} = \mathbf{A}\mathbf{v}$ and $\mathbf{v} = \{v_i = (i-1)u_{i-2}(t) : i = 1, ..., d/2+1\}'$.

The multimodal Gamma class is given by:

$$\tilde{f}_G^{(d)}(x_t|X_s) \propto x_t^{-\beta_1} \exp\Big(- \sum_{i=2}^{d/2+1} \beta_i x_t^{i-1} / (i-1) \Big) \quad \forall \quad x_t \in [0, \infty), \tag{2.3.26}$$

where $\boldsymbol{\beta} = \mathbf{A}\mathbf{v}$ and $\mathbf{v} = \{v_i = iu_{i-1}(t) : i = 1, ..., d/2+1\}'$.

The multimodal Inverse Gamma class is given by:

$$\tilde{f}_{IG}^{(d)}(x_t|X_s) \propto x_t^{-\beta_2} \exp\Big(-\beta_1/x_t - \sum_{i=3}^{d/2+1} \beta_i x_t^{i-2} / (i-2) \Big) \quad \forall \quad x_t \in [0, \infty), \tag{2.3.27}$$

where $\boldsymbol{\beta} = \mathbf{A}\mathbf{v}$ and $\mathbf{v} = \{v_i = (i+1)u_i(t) : i = 1, ..., d/2+1\}'$.

The multimodal Beta class is given by:

$$\tilde{f}_B^{(d)}(x_t|X_s) \propto x_t^{-\beta_1} (1-x)^{-\sum_{i=1}^{d/2+1} \beta_i} \exp\Big(- \sum_{i=1}^{d/2-1} \big(\sum_{j=i+1}^{d/2} \beta_j \big) x^i / i \Big) \quad \forall \quad x_t \in [0, 1], \tag{2.3.28}$$

where $\boldsymbol{\beta} = \mathbf{A}\mathbf{v}$ and $\mathbf{v} = \{v_i = iu_{i-1}(t) - (i+1)u_i(t) : i = 1, ..., d/2+1\}'$. By applying the cumulant truncation procedure we may recover the approximate trajectories of the non-central moments through the well-known relation:

$$\begin{aligned} u_1(t) &= \kappa_1(t), \\ u_i(t) &= \kappa_i(t) + \sum_{j=1}^{i-1} \binom{i-1}{j-1} \kappa_j(t) u_{i-j}(t) \quad \forall \quad i = 2, \dots, d. \end{aligned} \tag{2.3.29}$$

Subsequently, by plugging the $u_i(t)$ into any of the above densities we may approximate the transitional density of a scalar GQD.

For bivariate GQDs under an (even) d-th order truncation, the saddlepoint approximation (Renshaw, 2000) is given by:

$$\tilde{f}^{(d)}_{SPT}(x_t, y_t|X_s, Y_s) = \frac{\exp\big(K_2^{(d)}(\alpha_0, \beta_0) - \alpha_0 x_t - \beta_0 y_t\big)}{2\pi\sqrt{\frac{\partial^2 K_2^{(d)}}{\partial\alpha^2}\frac{\partial^2 K_2^{(d)}}{\partial\beta^2} - \big(\frac{\partial K_2^{(d)}}{\partial\alpha\partial\beta}\big)^2}}, \tag{2.3.30}$$

for

$$\begin{aligned} K_2^{(d)}(\alpha,\beta) = {} & \alpha\kappa_{10}(t) + \frac{\alpha^2}{2}\kappa_{20}(t) + \frac{\alpha^3}{6}\kappa_{30}(t) + \frac{\alpha^4}{24}\kappa_{40}(t) + \ldots + \frac{\alpha^d}{d!}\kappa_{d0}(t) \\ & + \beta\kappa_{01}(t) + \frac{\beta^2}{2}\kappa_{02}(t) + \frac{\beta^3}{6}\kappa_{03}(t) + \frac{\beta^4}{24}\kappa_{04}(t) + \ldots + \frac{\beta^d}{d!}\kappa_{0d}(t) \\ & + \alpha\beta\kappa_{11}(t) + \frac{\alpha^2\beta}{2}\kappa_{21}(t) + \frac{\alpha\beta^2}{2}\kappa_{12}(t) + \ldots + \frac{\alpha^{d/2}\beta^{d/2}}{((d/2)!)^2}\kappa_{(d/2)(d/2)}(t) + \ldots \\ & + \frac{\alpha^{d-1}\beta}{(d-1)!}\kappa_{(d-1)1}(t) + \frac{\alpha\beta^{d-1}}{(d-1)!}\kappa_{1(d-1)}(t), \end{aligned} \tag{2.3.31}$$

where α_0 and β_0 solves the system:

$$\begin{aligned} \frac{\partial K_2^{(d)}}{\partial\alpha}(\alpha,\beta) &= x_t, \\ \frac{\partial K_2^{(d)}}{\partial\beta}(\alpha,\beta) &= y_t. \end{aligned} \tag{2.3.32}$$

Since Equation 2.3.32 requires numerical evaluation, it is possible in certain circumstances that numerical instabilities arise when using the bivariate saddlepoint approximation. For this reason we include, as an alternative to the bivariate saddlepoint approximation, the bivariate Edgeworth expansion (Barndorff-Nielsen and Cox, 1979): Let $\tilde{f}_N(x_t, y_t|X_s, Y_s)$ denote the bivariate Normal distribution

$$\begin{aligned} &\tilde{f}_N(x_t, y_t|X_s, Y_s) = \\ &\frac{1}{2\pi\sqrt{\kappa_{20}\kappa_{02}(1-\rho_{11}^2)}}\exp\bigg(-\frac{(x_t-\kappa_{10})^2}{2\kappa_{20}(1-\rho_{11}^2)} + \frac{2\rho_{11}(x_t-\kappa_{10})(y_t-\kappa_{01})}{2\sqrt{\kappa_{20}\kappa_{02}}(1-\rho_{11}^2)} - \frac{(y-\kappa_{01})^2}{2\kappa_{02}(1-\rho_{11}^2)}\bigg), \end{aligned} \tag{2.3.33}$$

$H_i(u)$ the i-th scalar Hermite polynomial (see Andrews *et al.*, 1999)

$$H_i(u) = (-1)^i e^{\frac{u^2}{2}}\frac{\partial^i}{\partial u^i}e^{-\frac{u^2}{2}} \tag{2.3.34}$$

with $H_0(u) = 1$ and $\rho_{ij} = \frac{\kappa_{ij}}{\sqrt{\kappa^i_{20}\kappa^j_{02}}}$. Then the fourth-order bivariate Edgeworth expansion for the transitional density is given by the expression:

$$\begin{aligned}&\tilde{f}_{EDG}(x_t, y_t|X_s, Y_s) = \\ &\tilde{f}_N(x_t, y_t|X_s, Y_s) \times \left(1 + \frac{1}{6}A(\dot{x}_t, \dot{y}_t) + \frac{1}{24}B(\dot{x}_t, \dot{y}_t) + \frac{1}{72}C(\dot{x}_t, \dot{y}_t))\right)\end{aligned} \tag{2.3.35}$$

where $\dot{x}_t = (x_t - \kappa_{10})/\sqrt{\kappa_{20}}$, $\dot{y}_t = (y_t - \kappa_{01})/\sqrt{\kappa_{02}}$ and the terms A, B and C are given by:

$$\begin{aligned}A(u, v) =& H_3(u)\rho_{30} + 3H_2(u)H_1(v)\rho_{21} + 3H_1(u)H_2(v)\rho_{12} + H_3(v)\rho_{03}, \\ B(u, v) =& H_4(u)\rho_{40} + 4H_3(u)H_1(v)\rho_{31} + 6H_2(u)H_2(v)\rho_{22} \\ &+ 4H_1(u)H_3(v)\rho_{13} + H_4(v)\rho_{04},\end{aligned} \tag{2.3.36}$$

and

$$\begin{aligned}C(u, v) =& H_6(u)\rho^2_{30} + 6H_5(u)H_1(v)\rho_{21}\rho_{30} + 6H_4(u)H_2(v)\rho_{12}\rho_{30} \\ &+ 2H_3(u)H_3(v)\rho_{03}\rho_{30} + 9H_4(u)H_2(v)\rho^2_{21} + 18H_3(u)H_3(v)\rho_{21}\rho_{12} \\ &+ 6H_2(u)H_4(v)\rho_{21}\rho_{03} + 9H_2(u)H_4(v)\rho^2_{12} + 6H_1(u)H_5(v)\rho_{12}\rho_{03} \\ &+ H_1(u)H_6(v)\rho^2_{03}.\end{aligned} \tag{2.3.37}$$

We end off with some remarks on the cumulant truncation procedure as applied to GQDs: Due to the inter-dependence between the cumulants according to the systems of equations 2.3.15 and 2.3.21, one requires a sufficiently large truncation order (d) in order for the systems to accurately replicate the time-evolution of a given diffusion process' cumulants. That is unless it happens that the transitional density turns out to be Normal – in which case a second order truncation ($d = 2$) will describe the cumulants exactly – a sufficient number of higher order cumulants need to be included in the cumulant system in order for the cumulant trajectories to be accurate. Fortunately, under the GQD framework, truncation orders of $d = 4$ produce sufficiently accurate approximations for use in practical applications for both scalar and bivariate GQDs. Furthermore, we find that the saddlepoint approximations perform well as surrogate densities and prove to be quite reliable. Thus, as a default, we recommend using a 4-th order truncation in conjunction with the corresponding saddlepoint approximation in order to approximate the transitional density of a GQD. For scalar GQDs where the saddlepoint approximation breaks down – for example when the dynamics of the diffusion dictate that the diffusion has finite support – one may employ an appropriate member of the multimodal Pearson system of densities or Normal distribution as a surrogate density. Likewise, should the saddlepoint

approximation break down for whatever reason in the bivariate case, we provide the option of selecting either the bivariate Edgeworth expansion or bivariate Normal distribution. Although we fix the maximum truncation order for bivariate GQDs to $d = 4$, the **DiffusionRgqd** package supports cumulant and density truncations for even orders up to and including $d = 8$ for scalar GQDs.

With regard to the bivariate case, note that even under moderate truncations the number of terms on the RHS of Equation 2.3.21 can become daunting. For example, setting $d = 4$, Equation 2.3.21 translates into a 14-dimensional system with 220 non-zero *entries* when accounting for the presence of all of the coefficients of Equation 2.2.2. By *entry*, here we mean a grouped expression of terms for which any given coefficient of the model is a common factor (for example, row 4, column 6 of Table B.1.1 in Appendix B.1 constitutes an 'entry'). In comparison, a 6-th order truncation would have resulted in a 27-dimensional system with around 470 non-zero terms. Fortunately, since Equation 2.3.21 is essentially a mixture of relatively simple logical and algebraic expressions, it is easy to write a program in a computer algebra system or even R that expands Equation 2.3.21. For convenience, we have tabulated the appropriate terms of the cumulant equations under a fourth-order truncation for the bivariate form in Appendix B.1.

2.4 Building computationally optimized solutions in R with C++

Although the cumulant truncation procedure provides a computationally efficient means of approximating the transition density of a non-linear diffusion, the application thereof in the context of likelihood inference may be computationally demanding if not coded efficiently. In general, given N observations of a process, a single evaluation of the likelihood demands $N - 1$ approximations of the transitional density. This overhead is magnified when evaluations of the likelihood are made iteratively in MCMC procedures and/or when transition densities are approximated over relatively large transition horizons. For example, when evaluating the likelihood of 500 observations of a bivariate process for 100 000 iterations of an MCMC procedure, we are required to numerically solve (thus incurring further iteration) a multidimensional, non-linear system of ODEs $499 \times 100\,000$ times – a tedious task even under ideal computing conditions. Thus care needs to be taken when setting up code within interpretive languages such as R.

Given that inference procedures within the **DiffusionRgqd** package will be applied to datasets of varying complexity (i.e., size, time dependence, linearity etc.), we are prompted to maximise the computational efficiency of relevant routines. As a first

step in improving computational efficiency we circumvent the interpretive overhead of the R language by making use of the C++ language, which may be interfaced with the R environment through the **Rcpp** package (Eddelbuettel and François, 2011; Eddelbuettel, 2013). This is further facilitated by the **RcppArmadillo** package (Eddelbuettel and Sanderson, 2014) – a linear algebra library written for use with the C++ language in R. By combining the C++ language with the **RcppArmadillo** libraries we can efficiently manipulate large matrices, thus making it possible to vectorize all numerical elements of the cumulant truncation procedure within the C++ language.

A natural consequence of adopting the GQD framework is that various forms of mathematical redundancies surface that may be exploited for further computational gains. As mentioned in Section 2.3.2, depending on the coefficients that are included or excluded for a given GQD, components of the cumulant truncation procedure may be simplified in order to avoid unnecessary calculations. For example, when a model has redundant higher order cumulants, the dimensions of the cumulant system from equations 2.3.15 or 2.3.21 may be reduced. Where the analysis permits a second-order truncation, the saddlepoint approximation collapses to the Normal distribution further simplifying the calculation. Furthermore, when the data is of homogeneous resolution (equispaced observations) we circumvent the need to keep track of each transition horizon individually. Within the GQD framework, we may identify these redundancies prior to conducting the analysis and adapt a given routine accordingly. We emulate this in the GQD package source code by identifying redundant components from coefficients supplied by the user in real time and subsequently tailor an optimal solution from predefined blocks of code.

Consider for example applying the cumulant truncation procedure to the bivariate GQD. By checking which of the 36 coefficients are included in the model specification we can identify an optimum algorithmic solution. When the dimensions of the process are independent, one of four outcomes are possible: Either both dimensions are normally distributed, the X_t dimension is normally distributed and the Y_t dimension is non-normal or *vice versa*, or both are non-normal. Normality of either dimension can be established by checking whether the coefficients of non-linear terms in the drift or diffusion of the relevant dimension are absent. Furthermore, since the cross-cumulants of a bivariate density become redundant when no interaction terms are included, the dimensions of the cumulant system resulting from Equation 2.3.21 can be reduced to 4, 6, or 8 dimensions depending on which of the aforementioned outcomes apply. Finally, perhaps the most important reducible component for independent dimensions relates to the fact that we may evaluate the surrogate density analytically as a product of

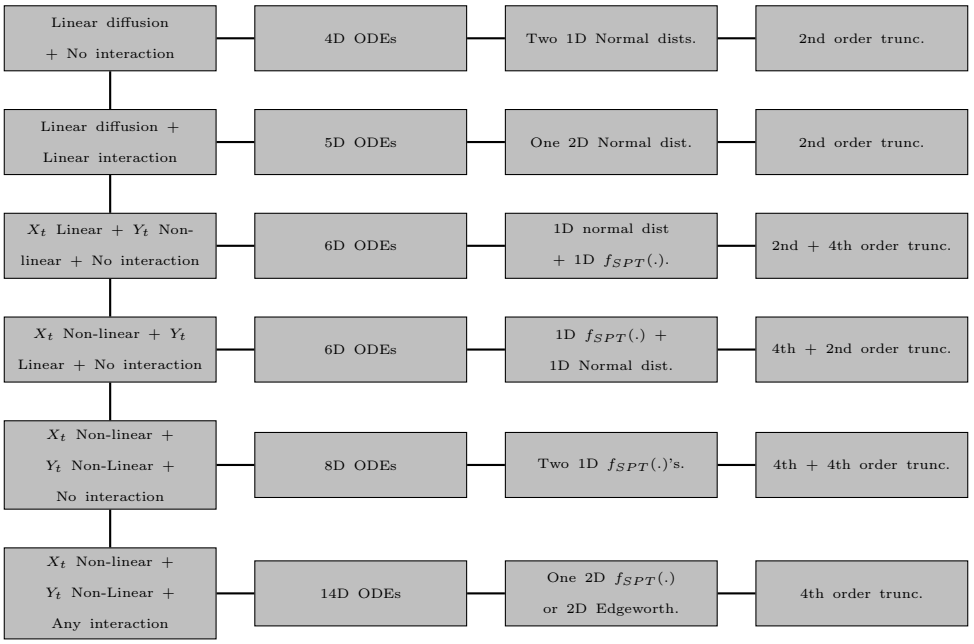


FIGURE 2.4.1: Default solution types for various bivariate GQD: Depending on the coefficients of a given model, the moment equations may become reducible due to redundant dimensions. As such, computational efficiency may be improved in certain cases by evaluating a smaller set of moment equations as well as a simpler density approximation. Although the dimensions of the cumulant equations (second column) are always determined by the diffusion, the package does allow one to override the type of density approximation used manually should the need arise.

scalar surrogate densities as opposed to using Equation 2.3.30 which requires the numerical evaluation of the roots of Equation 2.3.32.

Whenever interaction terms are included in the model, one of two outcomes are possible: Either the model has a bivariate Normal transition density or not, in which case we employ either the bivariate saddlepoint approximation or the bivariate Edgeworth expansion. The corresponding cumulant system is then 5-dimensional if bivariate Normal or 14-dimensional whenever non-linear terms are included. Thus, for a bivariate Normal density, we may evaluate the surrogate density analytically whilst the bivariate saddlepoint requires an additional layer of numerical overhead. Figure 2.4.1 summarises how various particular solutions of the cumulant truncation procedure for the bivariate GQD arise.

As an example of how model specification affects computational complexity and the structure of a 'coded' solution, consider evaluating the likelihood of an SDE:

$$\begin{aligned} dX_t &= \theta_1(\theta_2 + \theta_3 \sin(2\pi(t)) - X_t)dt + \theta_4\sqrt{X_t}dB_t^{(1)} \\ dY_t &= \theta_5 Y_t(\theta_6 - Y_t)dt + \theta_7 Y_t dB_t^{(2)}. \end{aligned} \tag{2.4.1}$$

By applying the cumulant truncation procedure, we first derive the system of ODEs that govern the evolution of the cumulants. Following Figure 2.4.1, we apply a 4-th order truncation on each dimension which results in the cumulant system:

$$\begin{aligned} \kappa'_{10}(t) &= \theta_1(\theta_2 + \theta_3 \sin(2\pi t)) - \theta_1\kappa_{10}(t) \\ \kappa'_{20}(t) &= -\theta_1 2\kappa_{20}(t) + \theta_4^2\kappa_{10}(t) \\ \kappa'_{30}(t) &= -\theta_1 3\kappa_{30}(t) + \theta_4^2 3\kappa_{20}(t) \\ \kappa'_{40}(t) &= -\theta_1 4\kappa_{40}(t) + \theta_4^2 6\kappa_{30}(t) \\ \kappa'_{01}(t) &= \theta_5\theta_6\kappa_{01} - \theta_5(\kappa_{01}\kappa_{01} + 1\kappa_{02}) \\ \kappa'_{02}(t) &= \theta_5\theta_6 2\kappa_{02} - \theta_5(4\kappa_{02}\kappa_{01} + 2\kappa_{03}) + \theta_7^2(\kappa_{02} + \kappa_{01}\kappa_{01}) \\ \kappa'_{03}(t) &= \theta_5\theta_6 3\kappa_{03} - \theta_5(6\kappa_{01}\kappa_{03} + 6\kappa_{02}\kappa_{02} + 3\kappa_{04}) + \theta_7^2(3\kappa_{03} + 6\kappa_{01}\kappa_{02}) \\ \kappa'_{04}(t) &= \theta_5\theta_6 4\kappa_{04} - \theta_5(8\kappa_{01}\kappa_{04} + 12\kappa_{02}\kappa_{03} + 12\kappa_{03}\kappa_{02}) \\ &\quad + \theta_7^2(6\kappa_{04} + 12\kappa_{01}\kappa_{03} + 12\kappa_{02}\kappa_{02}) \end{aligned} \tag{2.4.2}$$

with initial conditions $\kappa_{10}(s) = x_s$, $\kappa_{01}(s) = y_s$ and $\kappa_{\cdot}(s) = 0$ otherwise.

In order to solve Equation 2.4.2 we may employ single-step Runge-Kutta methods – a process whereby a system of ODEs is evaluated numerically at fixed points over a desired transition horizon $[s, t]$. Let

$$\frac{d}{dt}\boldsymbol{\kappa}(t) = h(\boldsymbol{\kappa}(t), t) \tag{2.4.3}$$

denote the system of cumulant equations for a given diffusion model (i.e., for Equation 2.4.1 $h(\boldsymbol{\kappa}(t), t)$ is given by the right hand side of Equation 2.4.2). Then, subject to the initial condition $\boldsymbol{\kappa}(s) = \boldsymbol{\kappa}_0(s)$, we can evaluate the cumulant trajectories at consecutive time points $t_0 = s, t_1, t_2, \ldots, t_{M+1} = t$ via updating equations of the form:

$$\boldsymbol{\kappa}_n(t_{i+1}) = \boldsymbol{\kappa}_0(t_i) + \sum_{j=0}^{n-1} c_j h(\boldsymbol{\kappa}_j(t_i + a_j\Delta), t_i + a_j\Delta)\Delta \tag{2.4.4}$$

for $i = 0, 1, \ldots M$, with

$$\boldsymbol{\kappa}_j(t_i + a_j\Delta) = \boldsymbol{\kappa}_0(t_i) + \sum_{k=0}^{j-1} b_{ij} h(\boldsymbol{\kappa}_k(t_i + a_k\Delta), t_i + a_k\Delta)\Delta \tag{2.4.5}$$

for $j = 1, 2, \ldots, n-1$, $a_\cdot$, $b_{\cdot\cdot}$ and $c_\cdot$ all real valued and integer n. The order of the approximation and its accuracy depend on both n (the number of stages used during a single update) and the values chosen for the coefficients $a_\cdot$, $b_{\cdot\cdot}$ and $c_\cdot$. For introductory material on Runge-Kutta methods see for example Butcher (2007) or Atkinson (2008). For purposes of the **DiffusionRgqd** package, we employ either a 4(5)-th order Runge-Kutta method using the coefficients of the so-called Runge-Kutta-Fehlberg method (Fehlberg, 1970), or the 8(10)-th order Runge-Kutta scheme of Feagin (2007) where the term in brackets indicates the embedded order used to evaluate the local truncation error estimate (see Appendix B.2 and B.3 for the relevant coefficient values). Note that in the case of the 4(5) method we use the 4-th order result in order to update the approximation and in the case of the 8(10) method we use the 10-th order result. Throughout the package we opt to use fixed step sizes, say $t_0 = s, t_1 = s + \Delta, ..., t_i = s + i\Delta, ..., t - \Delta, t$ with $\Delta = (t - s)/M$, as opposed to adaptive step sizes. In addition to providing more control over the quality of the approximate cumulant systems within a given application, we find that fixed step-size schemes are significantly less prone to breaking down in comparison to adaptive step sizes.

As the final step in evaluating the likelihood we need to evaluate the transitional density. Note that since both dimensions are non-linear and no interaction terms are present, we may approximate the transitional density at t by plugging the approximate solution to the cumulants into the product of two scalar saddlepoint approximations (see Equation 2.3.22):

$$f_{SPT}(x_t, y_t | x_s, y_s) = \frac{\exp\big(K_x^{(4)}(\alpha_0, t) - \alpha_0 x_t + K_y^{(4)}(\beta_0, t) - \beta_0 y_t\big)}{2\pi\sqrt{\frac{\partial^2}{\partial\alpha^2}K_x^{(4)}(\alpha_0, t)\frac{\partial^2}{\partial\beta^2}K_y^{(4)}(\beta_0, t)}}, \tag{2.4.6}$$

where

$$K_x^{(4)}(\alpha, t) = \kappa_{10}(t)\alpha + \frac{\kappa_{20}(t)\alpha^2}{2} + \frac{\kappa_{30}(t)\alpha^3}{6} + \frac{\kappa_{40}(t)\alpha^4}{24} \tag{2.4.7}$$

and

$$\alpha_0 = -\frac{\kappa_{30}(t)}{\kappa_{40}(t)} + \left(-\frac{q}{2} + \sqrt{C}\right)^{1/3} - \left(\frac{q}{2} + \sqrt{C}\right)^{1/3}, \tag{2.4.8}$$

with

$$
\begin{aligned}
p &= \frac{1}{3}\left(\frac{1}{2}\kappa_{40}(t)\kappa_{20}(t) - \frac{1}{4}\kappa_{30}(t)^2\right)/\left(\frac{1}{6}\kappa_{40}(t)\right)^2, \\
q &= \frac{1}{27}\left(\frac{27}{36}\kappa_{40}(t)^2(\kappa_{10}(t) - x_t) - \frac{3}{4}\kappa_{40}(t)\kappa_{30}(t)\kappa_{20}(t) + \frac{1}{4}\kappa_{30}(t)^3\right)/\left(\frac{1}{6}\kappa_{40}(t)\right)^3, \\
C &= \frac{q^2}{4} + \frac{p^3}{27}
\end{aligned}
\tag{2.4.9}
$$

and β_0 can be evaluated similarly for $K_y^{(4)}(\beta, t)$.

Finally, by combining these elements we can evaluate the log-likelihood of Equation 2.4.1 numerically for any number of transitions. Now, altering the model only slightly by for example including an interaction term, say $\theta_8 X_t Y_t$ in the drift of X_t, would result in a much more complex solution, which in turn would require similarly more complex code. As such, for any given model we would have to go through the same process as above in order to identify the most computationally efficient solution. This remains true for the scalar case as well. For example, depending on what density and truncation order that is used, various approximations to the likelihood may be constructed. As such we have developed the **DiffusionRgqd** package so as to identify the most efficient algorithm for a given model specification. From this, a solution is constructed in C++ code which can then be compiled by the `sourceCpp()` function from the **Rcpp** package, which may subsequently be called as a function in the R environment. Thus, by delegating computationally expensive parts of the cumulant truncation procedure to C++ we can significantly speed up evaluation of the likelihood whilst managing the algorithms that employ the likelihood equations using R. In this way, we are able to construct computationally efficient routines for performing inference and analysis on diffusion processes nested within the generalised quadratic framework.

2.5 The DiffusionRgqd package

2.5.1 Outline of the package

DiffusionRgqd consists of a set of functions that allow the user to perform inference and analysis on generalised quadratic diffusions. The main routines that appear in the package are (main functions that do **not** use C++ are indicated with an asterisk):

`BiGQD.density`*: Calculate the transition density of a bivariate GQD with time-inhomogeneous coefficients over a specified time interval.

`BiGQD.mcmc` : Use an MCMC algorithm to draw parameters of a bivariate GQD model with time-inhomogeneous coefficients.

`BiGQD.mle` : Calculate maximum likelihood estimates for the parameters of a bivariate GQD model with time-inhomogeneous coefficients.

`GQD.density`* : Calculate the transition density of a scalar GQD with time-inhomogeneous coefficients over a specified time interval.

`GQD.mcmc` : Use an MCMC algorithm to draw the parameters of a scalar GQD model with time-inhomogeneous coefficients.

`GQD.mle` : Calculate maximum likelihood estimates for the parameters of a GQD model with time-inhomogeneous coefficients.

`GQD.passage` : Approximate the first passage time density of a time-homogeneous GQD to a fixed threshold function (we defer discussion of first passage time problems to a later chapter in this thesis).

`GQD.TIpassage` : Approximate the first passage time density of a GQD with time-inhomogeneous coefficients to a fixed threshold function (we defer discussion of first passage time problems to a later chapter in this thesis).

In addition to the main routines, some supporting functions have been created to make the package more user friendly. These include:

`GQD.remove` : Removes the coefficients of an existing GQD model from the current workspace.

`GQD.dic` : Summarizes DIC values from a list of `GQD.mcmc` and `BiGQD.mcmc` objects.

`GQD.aic` : Summarizes AIC values from a list of `GQD.mle` and `BiGQD.mle` objects.

`GQD.plot` : Plot routines for various classes of objects in the **DiffusionRgqd** package.

2.5.2 `GQD.mcmc()` details

Before commencing with concrete examples of the package we detail some of the input parameters and how they relate to the theory developed in Section 2.3.2 by focusing on the `GQD.mcmc()` function. `GQD.mcmc()` is perhaps the most involved function in the package with respect to input, however most of the parametric input of `GQD.mcmc()` is shared by other routines in the package. The function call consists of (see *usage* in the package help files):

```
GQD.mcmc(X, time, mesh, theta, sds,...)
```

The input parameters for `GQD.mcmc()` are

`X`:

A vector containing discretely observed data points of a time series to be modelled.

`time`:

A vector containing the time points at which the observations in `X` were made. Although `time` values are restricted to be numeric e.g., $0, 0.1, 0.2, \ldots$, dates can easily be converted to numerical values prior to input using the standard `as.Date` functions in R. It is important to distinguish the unit of measurement used relative to the time signature e.g., monthly data using a yearly unit interval may result in a time signature `seq(0, 1, by = 1/12)` for each unit of time.

`mesh`:

`mesh` gives number of updates used in the evaluation of the chosen Runge-Kutta scheme (see equations 2.4.4 and 2.4.5) for all transition horizons. The number of updates imply the time discretization of each transition horizon. Each transition horizon `time[i + 1] - time[i]` is subdivided into `mesh` subintervals, i.e., `mesh + 1` mesh points, over which the cumulant equations are evaluated numerically. Note that the same number of points are used regardless of the individual magnitudes of `time[i + 1] - time[i]`. Thus if the data are not equispaced, `mesh` should be chosen so as to be large enough to ensure a sufficiently fine mesh on the maximal transition horizon `max(diff(time))`.

`theta`:

The parameter vector of the process. The values assigned to the vector `theta` are used as the starting values for parameter chains generated using a random walk Metropolis-Hastings (RWMH) algorithm. Note that the

estimation routines in **DiffusionRgqd** use C++. Thus, in order to ensure the syntactical compatibility with the underlying C++ code, `theta` is used as a reserved name for parameters in the coefficient functions. As such, any unrecognised variables will result in the execution of the algorithm being halted. In any event, where needed, routines will conduct a number of basic syntax checks in order to guide the user in the right direction.

`sds`:

Proposal distribution standard deviations under the RWMH-algorithm. That is, for the i-th parameter the proposal distribution is Normal(..., `sds[i] * sds[i]`).

`updates`:

The number of MCMC updates/iterations to perform (including burn-in).

`burns`:

The number of MCMC updates/iterations to burn/discard. When calculating parameter estimates, this value may be changed externally.

`Trunc`:

A vector indicating the truncation order to be used on the cumulant equations and the density approximation respectively. Possible values are `c(4, 4)`, `c(6, 4)`, `c(8, 4)`, `c(6, 6)`, `c(8, 6)` and `c(8, 8)`. This follows since the number of coordinates in the cumulant equations (Equation 2.3.15) preclude those in the corresponding density approximations.

`Dtype`:

The density approximation to be used. Possible values are `'Saddlepoint'`, `'Normal'`, `'Gamma'`, `'InvGamma'` and `'Beta'` corresponding to the saddlepoint approximation (Equation 2.3.22) and the respective classes of the Pearson system (equations 2.3.25, 2.3.26, 2.3.27 and 2.3.28).

`P`:

The number of mesh points used in the normalization of the Pearson system (see Appendix B.4).

`alpha`:

A 'spread' parameter controlling mesh spacing used in the normalization of the Pearson system (see Appendix B.4).

`lower, upper`:

Upper and lower bounds used in the normalization of the Pearson system (see Appendix B.4).

`plot.chain`:

If = `TRUE` (default), a trace plot of the MCMC chain will be made along with a trace of the acceptance rate.

`Tag`:

A text tag that can be assigned to a given model for easy identification when calling summary functions such as `GQD.dic()`.

2.6 Example applications

In the examples that follow we demonstrate how **DiffusionRgqd** package is used in practice. The package can be found on GitHub at https://github.com/eta21 and the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/package=DiffusionRgqd. Full scripts and replication materials for the examples shown here can be found in the indicated sections of the supplementary materials. Additionally, detailed examples can be found within the package vignettes. These can be found on the package CRAN page or by running the command: `browseVignettes("DiffusionRgqd")`.

2.6.1 Generate the transition density of a time-inhomogeneous GQD

As an introductory example of the **DiffusionRgqd** package we approximate the transitional density of a scalar diffusion over an arbitrarily chosen transition horizon. This can be achieved using the `GQD.density()` function. `GQD.density()` generates the transitional density of a GQD for a given initial value using the cumulant truncation procedure outlined in Section 2.3.2. The function serves as a good starting point for any analysis being conducted using the **DiffusionRgqd** package as it allows one to check whether a proposed model does not exhibit nonsensical dynamics with respect to the problem at hand. Perhaps more importantly, it can be used to check the validity of the density approximation and/or an appropriate truncation order for the cumulants.

Consider a Cox-Ingersoll-Ross (CIR) (Cox *et al.*, 1985) process with time-dependent coefficients:

$$dX_t = 2(10 + \sin(2\pi(t - 0.5)) - X_t)dt + \sqrt{0.25(1 + 0.75\sin(4\pi t))X_t}dB_t. \quad (2.6.1)$$

In order to analyse Equation 2.6.1 in R, we need to define the model within the workspace. Since **DiffusionRgqd** uses a functional input interface which relies on declaring the functional form of a given model's coefficients, we need to make sure that the workspace doesn't contain any object names that might clash with those of the model coefficients. We can do this by simply running the `GQD.remove()` command:

```
R> library("DiffusionRgqd")
R> GQD.remove()
```

```
[1] "Removed :  NA "
```

If any objects are recognised with names that may clash with names reserved for use with the `GQD.density()` function, they will subsequently be removed. In this case, we used a *vanilla* R session so the function will simply indicate that no clashes were detected and removed. The purpose of the `GQD.remove()` function is to act as a *model-eraser* in situations where multiple models are being defined in a single R session. The next step is to write Equation 2.6.1 in terms of its coefficients. We can define Equation 2.6.1 in the current R session by declaring the coefficient functions:

```
R> G0 <- function(t){2 * (10 + sin(2 * pi * (t - 0.5)))}
R> G1 <- function(t){-2}
R> Q1 <- function(t){0.25 * (1 + 0.75 * (sin(4 * pi * t)))}
```

The functions `G0`, `G1` and `Q1` together constitute the R-language counterpart of Equation 2.6.1 under the framework of Section 2.2. Note that for scalar GQDs we have capitalized the coefficient names given in Section 2.2 in order to avoid the possibility of calling `q()` accidentally, in which case the R console will prompt to quit.

In order to approximate the transitional density using the methodology of Section 2.3.2 we need to define the peripheral parameters of the problem. This consists of giving the initial condition of the SDE (the starting value of the diffusion process), the starting time of the diffusion process and the geometry of the transitional horizon, i.e., spatial values where the density is to be evaluated and the corresponding time mesh. These elements can be defined by assigning values to the arguments of `GQD.density()`: `Xs`, `Xt`, `s`, `t`, and `delt`. Respectively, these parameters represent the initial value, the values at which to evaluate the transition density, the starting time, the final time and the step size for the time mesh. That is, for a diffusion process starting at time `s` in state `Xs`, the transition density is approximated at times `s` + `delt`, `s` + 2`delt` ... up to and including `t` at all values of the vector `Xt`. In R:

```
R> states    <- seq(5, 15, 1/10)
R> initial   <- 8
R> Tmax      <- 5
R> Tstart    <- 1
R> increment <- 0.01
R>
R> res <- GQD.density(Xs = initial, Xt = states, s = Tstart, t = Tmax,
+   delt = increment)
```

`GQD.density()` creates a list containing a matrix of density values, spatial coordinates at which the density was evaluated, corresponding time coordinates for the time mesh and finally a matrix of trajectories for the cumulants and moments that were used in evaluating the density approximation. In this case, we have assigned the output to an object called `res` for further use in the workspace. Since the density approximation is evaluated over a space-time lattice it can best be visualised with a perspective plot. For example, using the **rgl** package (Adler *et al.*, 2016):

```
R> library("rgl")
R> open3d(windowRect = c(50, 50, 690, 690), zoom = 0.95)
R> persp3d(x = M$Xt,y = M$time,z = M$density, col = 3, box = F,
+   xlab = 'State (X_t)', ylab = 'Time(t)',zlab = 'Density f(X_t|X_s)')
```

we can recreate the surface in Figure 2.3.1. Alternatively, one can simply use `GQD.plot(res)` in order to visualise a given density approximation in a similar way.

Note that we have not directly specified what truncation order to be used in the calculation of the density approximation. As mentioned in Section 2.3.3, the default is to use a 4-th order truncation in conjunction with a 4-th order truncated saddlepoint approximation. That is, `GQD.density()` will set up the cumulant equations up to a fourth-order truncation, in this case:

$$\begin{aligned}
\kappa_1'(t) &= \quad 2 \times ((10 + \sin(2\pi(t - 0.5))) - \kappa_1(t)) \\
\kappa_2'(t) &= -2 \times 2\kappa_2(t) + 0.25(1 + 0.75\sin(4\pi t)) \times \kappa_1(t) \\
\kappa_3'(t) &= -2 \times 3\kappa_3(t) + 0.25(1 + 0.75\sin(4\pi t)) \times 3\kappa_2(t) \\
\kappa_4'(t) &= -2 \times 4\kappa_4(t) + 0.25(1 + 0.75\sin(4\pi t)) \times 6\kappa_3(t),
\end{aligned} \tag{2.6.2}$$

subject to the initial conditions $\{\kappa_1(s) = X_s, \kappa_i(s) = 0 : i = 2, 3, 4.\}$, which is subsequently solved numerically from s to t in discrete increments of size `delt`. At each iteration, the cumulants are plugged into Equation 2.3.22 and the density is evaluated at all points in the vector `Xt` corresponding to X_t. By collating the approximations evaluated at each increment, the density surface is formed

on the corresponding space-time lattice. In most cases, it suffices to use the default settings although, as will be shown in the following example, little effort is required to extend the analysis to higher order truncations and/or alternate density approximations. Although this example operates well within the limits of the capabilities of the methodology it serves to illustrate the simplicity of the interface of the **DiffusionRgqd** package – often producing desired results in less than 10 lines of code with minimal mathematical input.

2.6.2 Time-inhomogeneous Jacobi diffusion: A scalar diffusion with finite support

Although most practical applications of diffusion processes result in the use of diffusions with uni-modal transitional densities it is still possible to conceive of a diffusion within the GQD framework that exhibits atypical dynamics. For example, a particularly interesting phenomenon occurs when considering GQDs for which the diffusion term is of the form:

$$\sigma(X_t, t) = c\sqrt{X_t(1 - X_t)}. \tag{2.6.3}$$

Assuming that the process starts within the interval $[0, 1]$, whenever the diffusion approaches the bounds 0 or 1, the dynamics of the process is dominated by the behaviour of the drift function, $\mu(X_t, t)$. Provided that $\partial/\partial x \mu(x, t)|_{x=0} > 0$ and $\partial/\partial x \mu(x, t)|_{x=1} < 0 \quad \forall t$ are sufficiently large, the process will reflect from the boundaries and remain within the interval $[0, 1]$. A special case of this behaviour can be seen with the so-called Jacobi diffusion (Gouriéroux and Valéry, 2004), whereby $\mu(X_t, t)$ is linear in X_t. For purposes of the illustration we shall assume a time-inhomogeneous Jacobi diffusion:

$$dX_t = a(b(1 + 0.25\sin(\pi t)) - X_t)dt + c\sqrt{X_t(1 - X_t)}dB_t, \tag{2.6.4}$$

with $X_0, b \in [0, 1]$ and $a > 0$. The diffusion exhibits oscillating drift dynamics whereby the process is pulled towards a point fluctuating between $0.75b$ and $1.25b$. If and when the trajectory of the process hits 0 or 1 it is reflected toward the interior of the domain. By applying the cumulant truncation procedure to Equation 2.6.4, we approximate the evolution of the cumulants for various orders of truncation under the Beta-type density from the multimodal Pearson system (Equation 2.3.28). This may be achieved in R using the code:

```
R> GQD.remove()
R> a <- 0.5; b <- 0.6; cc <- 1;
R>
R> G0 <- function(t){a*(b*(1 + 0.25*sin(pi*t)))}
```

```
R> G1 <- function(t){-a}
R> Q1 <- function(t){cc}
R> Q2 <- function(t){-cc}
R>
R> Xs    <- 0.5
R> Xt    <- seq(0.001, 0.999, 0.001)
R> res1 <- GQD.density(Xs, Xt, 0, 2, 0.01, Dtype='Beta', Trunc = c(4, 4))
R> res2 <- GQD.density(Xs, Xt, 0, 2, 0.01, Dtype='Beta', Trunc = c(6, 6))
R> res3 <- GQD.density(Xs, Xt, 0, 2, 0.01, Dtype='Beta', Trunc = c(8, 8))
```

The objects `res1`, `res2` and `res3` now contain the transitional density approximations for truncation orders 4, 6 and 8. The additional parameter `Dtype = 'Beta'` sets the density approximation to Equation 2.3.28 while the variable `Trunc = c(6, 6)` sets the cumulant truncation (first item in `c(6, 6)`) to $d = 6$ (thus evaluating Equation 2.3.15 for all $i = 1, 2, \ldots, 6$) whilst setting $d = 6$ for Equation 2.3.28 (second item in `c(6, 6)`). Note that we separate the cumulant truncation order from that of the density in order to allow lower order density truncations under a given cumulant truncation order, for example `Trunc = c(6, 4)`.

In order to check the accuracy of the approximation, we may compare the approximate transition density to a simulated transition density of the Jacobi diffusion. This can be achieved by simulating a number of sample paths for Equation 2.6.4 and subsequently calculating the frequency distribution of the resulting trajectories. For a diffusion process with SDE

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dB_t, \tag{2.6.5}$$

we may simulate a trajectory at discrete time points by applying a Euler-Maruyama scheme to 2.6.5 and evaluate the resulting difference equation iteratively. That is, given an initial value X_{t_0}, we evaluate for all $i = 1, 2, \ldots$:

$$X_{t_i} = X_{t_{i-1}} + \mu(X_{t_{i-1}}, t_{i-1})\Delta + \sigma(X_{t_{i-1}}, t_{i-1})Z_\Delta \tag{2.6.6}$$

where $Z_\Delta \sim \mathrm{N}(0, \sigma^2 = \Delta)$ and $t_i - t_{i-1} = \Delta$. By plugging the drift and diffusion terms of Equation 2.6.4 into this updating equation we may simulate a number of trajectories for the restricted diffusion. However, since Equation 2.6.6 is not exact, the simulated paths may actually exit the $[0, 1]$ region. As such we employ the simple modification that at each iteration X_{t_i} is set to $\min(\max(0, X_{t_{i-1}} + \mu(X_{t_{i-1}}, t_{i-1})\Delta + \sigma(X_{t_{i-1}}, t_{i-1})Z_\Delta), 1)$. By simulating a large number of trajectories we can compare each respective density approximation by superimposing the approximation over a histogram of the simulated trajectories at any given time point. This can easily be achieved using the code:

```
R> delt <- 0.001
```

```
R> N    <- 100000
R> d    <- 0
R> X    <- rep(Xs,N)
R> when <- c(0, 0.25, 0.5, 1, 1.75)
R>
R> for(i in 2:2000)
+ {
+  X <- pmax(pmin(X + (G0(d) + G1(d)*X)*delt
+    + sqrt(Q1(d)*X + Q2(d)*X^2)*rnorm(length(X), sd = sqrt(delt)),
+    1), 0)
+
+  d <- d + delt
+  if(any(when == round(d, 3)))
+  {
+   index <- which(res1$time == round(d, 3))
+   hist(X, col = 'gray75', freq = F, breaks = 30,
+   main = paste0('Transitional Density at t  =  ', round(d, 3)),
+     ylim = c(0, 3), border = 'white')
+   lines(res1$density[, index] ~ res1$Xt, col = '#1B7837', lty = 'dotdash')
+   lines(res2$density[, index] ~ res2$Xt, col = '#D92120', lty = 'solid')
+   lines(res3$density[, index] ~ res3$Xt, col = '#5289C7', lty = 'dashed')
+  }
+ }
```

The resulting output is given in Figure 2.6.1. The figures suggest that the approximation remains accurate for truncation orders 6 and 8 whilst $d = 4$ produces less desirable results. Interestingly, as time progresses it seems that all orders of approximation produce a reasonable approximation of the transition density. This may be due to the long-run dynamics of the process being somewhat simpler than in early phases in the transition horizon. Figure 2.6.2 shows a perspective plot of the entire trajectory of the transitional density using the 8-th order truncation. As in the previous example, this can be done using the `GQD.plot()` command i.e., `GQD.plot(res2)`. As is typical for diffusion processes, we see that the transitional density starts out as an infinite point mass and shortly exhibits a normal-like distribution. As time progresses, however, the density transits into a 'U-shape' with oscillating peakedness in conjunction with the sinusoidal term of the drift function.

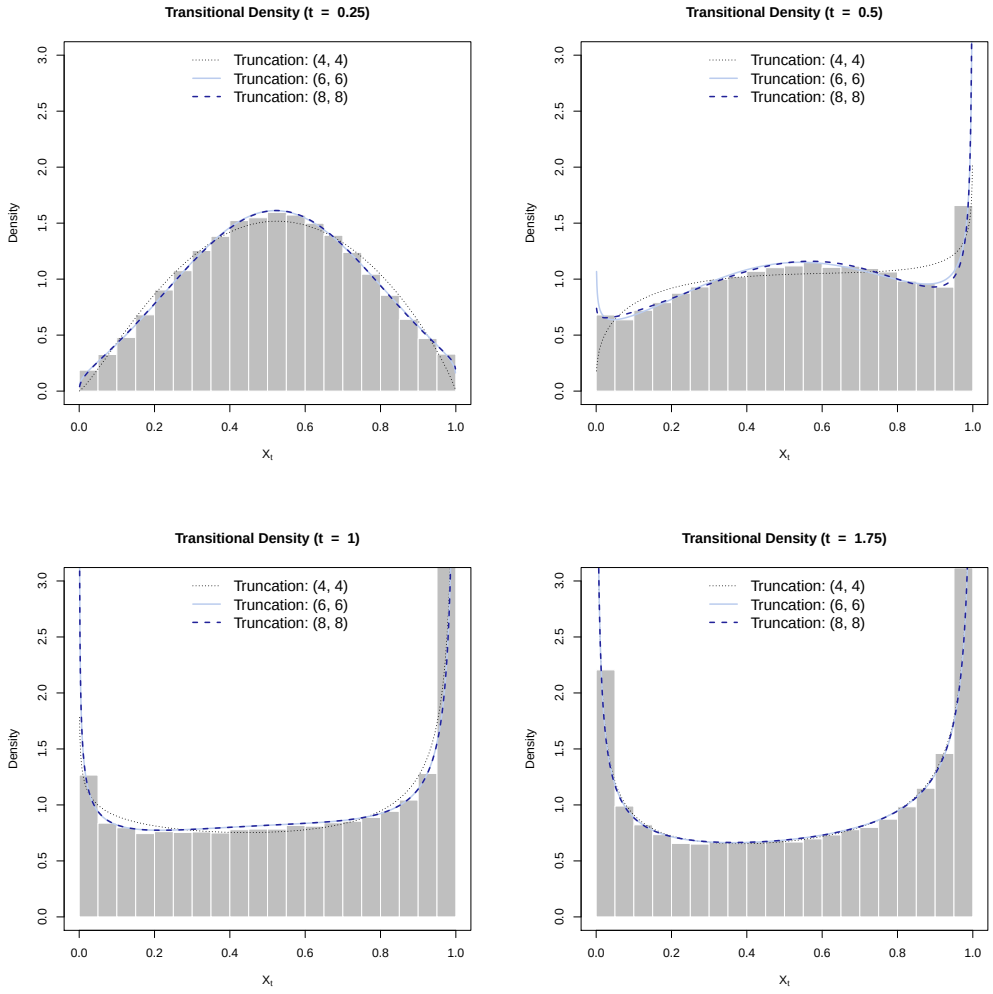


FIGURE 2.6.1: Histograms of simulated trajectories for Equation 2.6.4 and transitional density approximations for various truncation orders at the indicated times. Truncation orders 6 (light blue) and 8 (dark blue, dashed) prove accurate while a fourth-order truncation (black, dotted) fails to reproduce the desired shape for the transitional density at $t = 0.5$. R code: Supplementary materials, Section 2.2.

By altering the truncation order and the surrogate density used in the density approximation, we can experiment with various approximations in order to accurately approximate the transitional density over large transition horizons. Using the **DiffusionRgqd** package this can be achieved by simply tweaking the parameters of functions such as `GQD.density()` without having to deal with

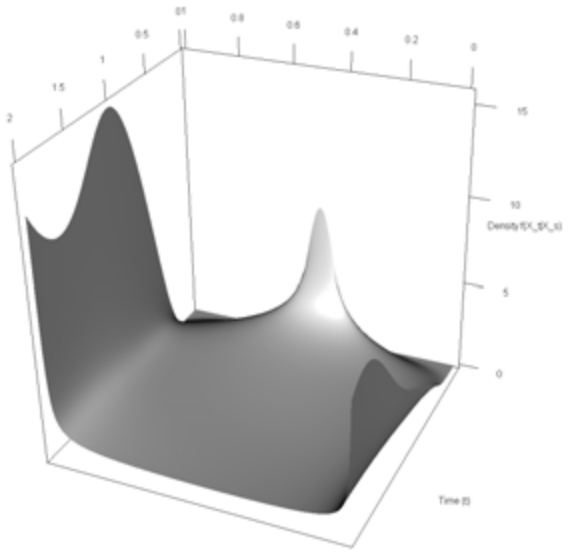


FIGURE 2.6.2: Evolution of the transition density of Equation 2.6.4 over time using the 8-th order cumulant truncation in conjunction with 8-th order truncated Pearson Beta density (see Equation 2.3.28). R code: Supplementary materials, Section 2.2.

the tedious mathematical elements involved in such a calculation. For example, compared to Equation 2.6.1, where we use a fourth-order approximation in conjunction with the saddlepoint approximation, the calculations involved in approximating the transition density of Equation 2.6.4 are somewhat more involved. For $d = 8$, the cumulant equations for the model are given by the system:

$$
\begin{aligned}
\kappa_1'(t) &= \quad a(b(1 + 0.25\sin(\pi t)) - \kappa_1(t))) \\
\kappa_2'(t) &= -2a\kappa_2(t) + c^2(\kappa_1(t) - (\kappa_2(t) + \kappa_1(t)^2)) \\
\kappa_3'(t) &= -3a\kappa_3(t) + c^2(3\kappa_2(t) - (3\kappa_3(t) + 6\kappa_1(t)\kappa_2(t))) \\
\kappa_4'(t) &= -4a\kappa_4(t) + c^2(6\kappa_3(t) - (6\kappa_4(t) + 12\kappa_1(t)\kappa_3(t) + 12\kappa_2(t)^2)) \\
\kappa_5'(t) &= -5a\kappa_5(t) + c^2(10\kappa_4(t) - (10\kappa_5(t) + 20\kappa_1(t)\kappa_4(t) + 60\kappa_2(t)\kappa_3(t))) \\
\kappa_6'(t) &= -6a\kappa_6(t) + c^2(15\kappa_5(t) - (15\kappa_6(t) + 30\kappa_1(t)\kappa_5(t) + 120\kappa_2(t)\kappa_4(t) + 90\kappa_3(t)^2)) \\
\kappa_7'(t) &= -7a\kappa_7(t) + c^2(21\kappa_6(t) - (21\kappa_7(t) + 42\kappa_1(t)\kappa_6(t) + 210\kappa_2(t)\kappa_5(t) + 420\kappa_3(t)\kappa_4(t))) \\
\kappa_8'(t) &= -8a\kappa_8(t) + c^2(28\kappa_7(t) - (28\kappa_8(t) + 56\kappa_1(t)\kappa_7(t) + 336\kappa_2(t)\kappa_6(t) + 840\kappa_3(t)\kappa_5(t) + 560\kappa_4(t)^2)).
\end{aligned}
\tag{2.6.7}
$$

Although the procedure used to solve Equation 2.6.7 is the same as for Equation 2.6.2, it is worth noting that Equation 2.6.7 contains numerous non-linear terms. Furthermore, for $d = 8$ under Equation 2.3.28, we are required to solve the system $\boldsymbol{\beta} = \mathbf{A}\mathbf{v}$ for a 5×5 matrix $\mathbf{A}$. In general, this can be achieved via standard numerical procedures (see for example `solve()` in R). However, in this context it is safer to calculate a general expression for the inverse of the matrix $\mathbf{A}$ in order to solve for the coefficients $\boldsymbol{\beta}$. This follows since it can be difficult to invert $\mathbf{A}$ numerically when the higher order non-central moments of the process are either very large or very small. As such, we calculate the inverse of $\mathbf{A}$ in terms of the non-central moments of the process in closed-form in order to evaluate the density approximation. Depending on the order of the truncation used, the number of operations required to do so may vary significantly. For reference, we provide the elements of $\mathbf{A}^{-1}$ in terms of the non-central moments for $d = 8$ in Appendix B.5.

2.6.3 Bivariate non-linear dynamics: The stochastic Lotka-Volterra equations

Although the **DiffusionRgqd** package allows one to perform comprehensive analysis on scalar quadratic diffusions, similar analysis can be conducted for interesting bivariate diffusions. A model that is often used to illustrate non-linear dynamics in the analysis of ODEs is that of the Lotka-Volterra model. The corresponding ODEs are typically given as:

$$\begin{aligned} \frac{\partial x_t}{\partial t} &= ax_t - bx_t y_t \\ \frac{\partial y_t}{\partial t} &= -cy_t + dx_t y_t \end{aligned} \tag{2.6.8}$$

for some positive a, b, c and d. The equations are often used to describe the dynamics of two interacting populations wherein the growth rate of each population is mutually influenced by the current level of the opposing population. As such the model has been used to explain oscillatory behaviour in predator-prey relationships (Hoppensteadt, 2006) where x_t denotes the prey population and y_t the predator population at time t. Continuing with the predator-prey metaphor, perhaps one deficiency of the model, one might argue, is the absence of random input and subsequent effects on population levels. Indeed, under the ODE formulation, the predicted population behaviour (given fixed parameters) are completely deterministic. Another deficiency might be the absence of growth inhibiting factors such as disease or over-grazing. For these purposes, we may

define an example of a stochastic counterpart to the Lotka-Volterra equations as:

$$\begin{aligned} dX_t &= (aX_t - bX_tY_t)dt + f\sqrt{X_t}dB_t^{(1)} \\ dY_t &= (-cY_t + dX_tY_t - eY_t^2)dt + g\sqrt{Y_t}dB_t^{(2)}. \end{aligned} \tag{2.6.9}$$

The intuition behind the example Equation 2.6.9 is that the drift terms replicate Equation 2.6.8 but with the addition of the $-eY_t^2$ term. The additional term represents the effect of overpopulation for the predator when, for example, harvesting of prey becomes ever more inefficient as the predator population grows. Furthermore, the population volatilities are assumed to increase with population size meaning that random fluctuations are less severe when populations are small and *vice versa*.

In order to analyse Equation 2.6.9, we may consider the evolution of its transitional density. This can be achieved in similar fashion to the previous examples through the `BiGQD.density()` function. For purposes of this example, consider the SDE:

$$\begin{aligned} dX_t &= (1.5X_t - 0.4X_tY_t)dt + \sqrt{0.05X_t}dB_t^{(1)} \\ dY_t &= (-1.5Y_t + 0.4dX_tY_t - 0.2Y_t^2)dt + \sqrt{0.1Y_t}dB_t^{(2)}, \end{aligned} \tag{2.6.10}$$

with initial values $X_0 = 5$ and $Y_0 = 5$. In R, we may generate the transitional density over the transition horizon $t \in [0, 10]$ using the code:

```
R> GQD.remove()
R> a10 <- function(t){1.5}
R> a11 <- function(t){-0.4}
R> c10 <- function(t){0.05}
R>
R> b01 <- function(t){-1.5}
R> b11 <- function(t){0.4}
R> b02 <- function(t){-0.2}
R> f01 <- function(t){0.1}
R>
R> res <- BiGQD.density(Xs = 5, Ys = 5, Xt = seq(3, 8, length = 50),
+   Yt = seq(2, 6, length = 50), s = 0, t = 10, delt = 0.01)
```

As in the scalar case, the model is coded by defining the coefficients of the diffusion as functions with names that reflect the powers of each terms' X_t and Y_t components. As with `GQD.density()`, `BiGQD.density()` approximates the transition density at times `s`, `s + delt` ... on a lattice of grid points given by the user (`Xt = seq(3, 8, length = 50)` and `Xt = seq(2, 6, length = 50)`) for a given pair of initial values (`Xs = 5,Ys = 5`). Since the transition density of a bivariate diffusion at a fixed time point is given by a surface in three dimensions, `BiGQD.density()` returns a three-dimensional array wherein

the transition density approximation is given as slices of this array corresponding to each time point `s`, `s+delt` etc. We can visualise the evolution of the transition density over time by making contour plots of the transition density at different time points. Incidentally, the mean trajectory of Equation 2.6.9 corresponds exactly to the trajectory of Equation 2.6.8 with the addition of the term $-0.2y_t^2$. Figure 2.6.3 illustrates the evolution of the transition density and compares the predicted expectation of Equation 2.6.10 under the cumulant truncation procedure to the simulated mean trajectory of the process. The transition density demonstrates that, in contrast to the deterministic nature of Equation 2.6.8, the inclusion of the Brownian motion terms in Equation 2.6.9 implies vastly different population dynamics: In the deterministic case the trajectories are predicted to move ever closer to an equilibrium point (attractor), whilst with the stochastic equations it is likely for either of the two populations to move far away from such an equilibrium point due to the presence of random fluctuations – even as the expected trajectory stabilizes over time. Thus, where the deterministic equations may predict both populations to survive with certainty, as under current parameter set, under the stochastic equations both extinction and explosion events are probable while on average, the population coordinates will tend to some equilibrium position as time increases.

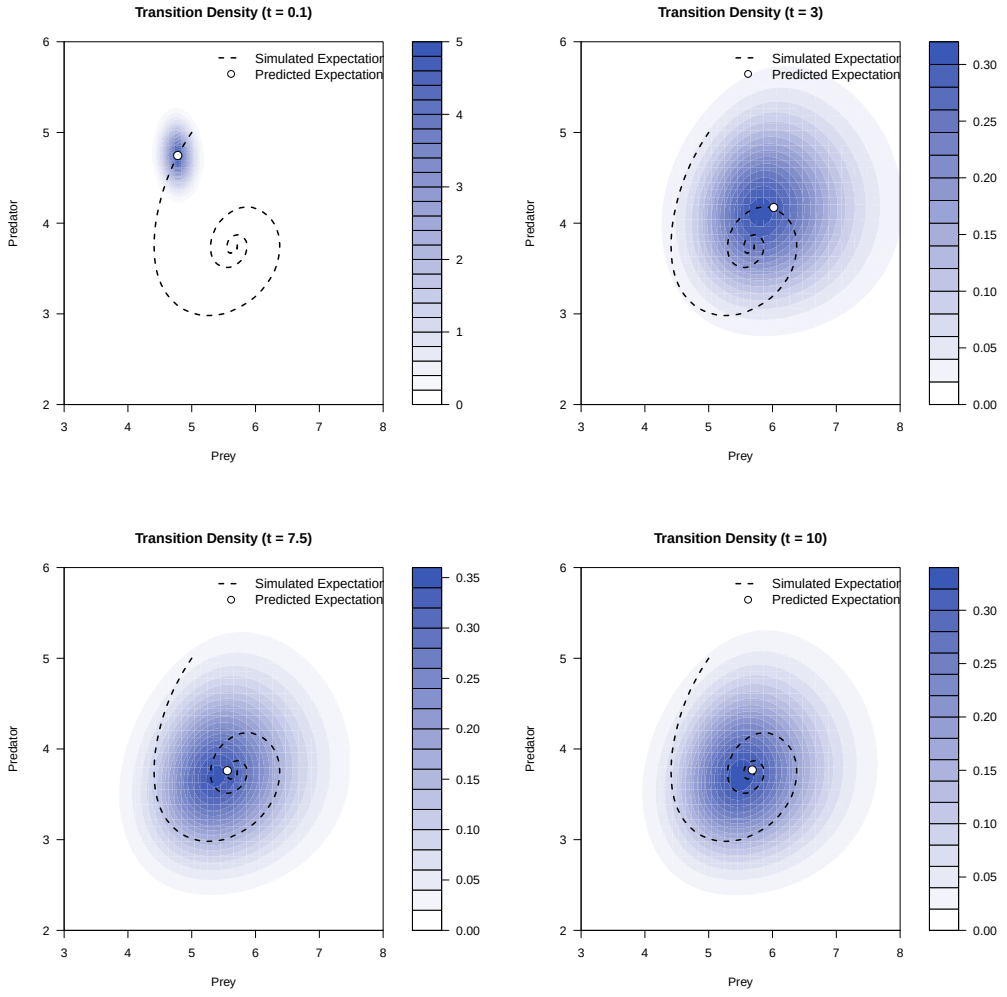


FIGURE 2.6.3: Coloured contour plots of the transition density at fixed time points $t \in \{0.1, 3, 7.5, 10\}$. Superimposed on the simulated mean trajectory, $(E(X_t), E(Y_t))$, of the process (dashed line) is the predicted expectation under the cumulant truncation procedure (white circle). R code: Supplementary materials, Section 2.3.

2.6.4 Maximum likelihood estimation: Stochastic volatility models

Diffusion processes are often used in financial applications to model the trajectories of asset prices or financial instruments. Although a great deal of literature is dedicated to the fitting and calibration of well-known diffusion models to financial data, less time is spent assessing how well the dynamics of such models represent that which is observed in a given real-world dataset. Thus, for the present example, we analyse a dataset that is often used to demonstrate the application of stochastic volatility models in finance, namely the Standard and Poor's 500 index (SPX)[1]. Although the term 'stochastic volatility' may be used to refer to any model with randomly varying higher order moments, in the context of diffusion models it usually refers to the process of treating the volatility terms of a scalar diffusion model as being driven by a diffusion process itself. Perhaps the most famous stochastic volatility model is the Heston model (Heston, 1993), a bivariate SDE of the form:

$$\begin{aligned} dS_t &= \theta_1 S_t dt + \theta_2 S_t \sqrt{V_t} dB_t^{(1)} \\ dV_t &= (\theta_3 - \theta_4 V_t)dt + \theta_5 \sqrt{V_t} dB_t^{(2)}, \end{aligned} \tag{2.6.11}$$

where S_t denotes the spot price process and V_t denotes the corresponding variance process. Thus, in the Heston model, the spot price is modelled by a geometric Brownian motion with the addition that the volatility coefficient for the spot price is itself driven by a diffusion process – in this case, a CIR process. Furthermore, it is assumed that the Brownian motions $dB_t^{(1)}$ and $dB_t^{(2)}$ are correlated i.e., $\text{corr}(B_t^{(1)}, B_t^{(2)}) = \theta_6$. For the SPX, although data for the spot process S_t is readily available, obtaining the trajectory of the volatility process is less trivial. In order to conduct the analysis, we shall assume a suitable proxy for the volatility of the index by making use if the Chicago Board Options Exchange (CBOE) Volatility Index (VIX), which is based on the implied volatility of options written on the SPX. We source data for the index by using the **Quandl** package (McTaggart and Daroczi, 2015) as follows:

```
R> quandldata1 <- Quandl("YAHOO/INDEX_GSPC", collapse = "weekly",
+ start_date = "1990-01-01", end_date = "2015-01-01",  type = "raw")
R> St <- rev(quandldata1[, names(quandldata1) == 'Close'])
R> time1 <- rev(quandldata1[, names(quandldata1) == 'Date'])
R>
R> quandldata2 <- Quandl("YAHOO/INDEX_VIX", collapse = "weekly",
+ start_date = "1990-01-01", end_date = "2015-01-01",  type = "raw")
R> Vt <- rev(quandldata2[, names(quandldata2) == 'Close'])
```

[1] The corresponding ticker symbols for Google finance and Yahoo finance are INX and ^GSPC respectively

Figure 2.6.4 plots the resulting weekly closing prices for the SPX and VIX for the period 1990-01-01 to 2015-01-01.

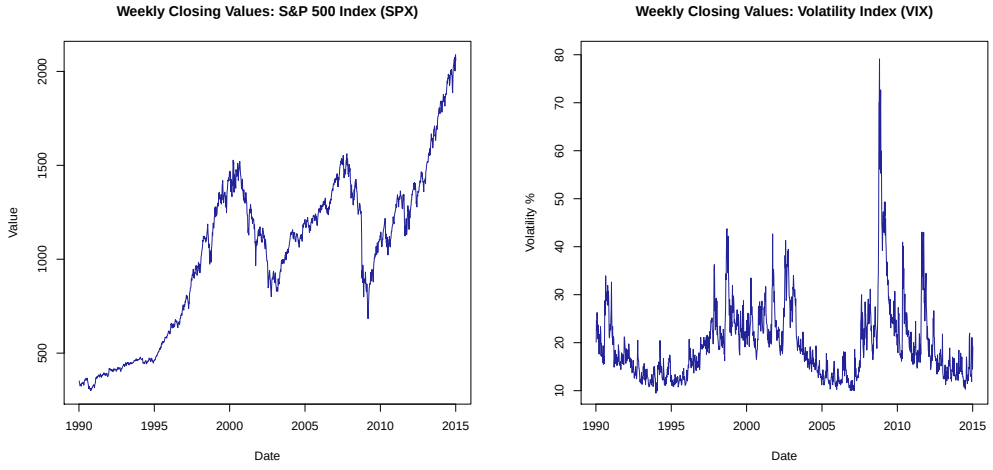


FIGURE 2.6.4: Weekly closing values for the Standard and Poor's 500 index (SPX) and Standard and Poor's 500 volatility index (VIX) for the period 1990-01-01 to 2015-01-01. Note that the volatility index is expressed in terms of volatility whilst models such as the Heston model are expressed in terms of the variance process (i.e., volatility squared). R code: Supplementary materials, Section 2.4.

Visual inspection of the time series suggests that the volatility of the index does indeed vary significantly over time as is evidenced by the VIX which was observed to be as low as 9.5% during December of 1993 and high as 79% during October of 2008.

In order to conduct the analysis, we consider the log-returns of the index. Let $R_t = \log(S_t)$, then by Itô's lemma the corresponding Heston model under the log-transform is given by:

$$\begin{aligned} dR_t &= (\theta_1 - \frac{1}{2}\theta_2^2 V_t)dt + \theta_2\sqrt{V_t}dB_t^{(1)} \\ dV_t &= (\theta_3 - \theta_4 V_t)dt + \theta_5\sqrt{V_t}dB_t^{(2)} \end{aligned} \tag{2.6.12}$$

with $\text{corr}(B_t^{(1)}, B_t^{(2)}) = \theta_6$. In order to incorporate the correlation coefficient of the Brownian motions into the SDE it is useful to write the SDE in terms of independent Brownian motions: Let $W_t^{(1)}$ and $W_t^{(2)}$ be independent Brownian motions then we can construct correlated Brownian motions $B_t^{(1)}$ and $B_t^{(2)}$ with

$\text{corr}(B_t^{(1)}, B_t^{(2)}) = \theta_6$ through the standard translation:

$$\begin{bmatrix} dB_t^{(1)} \\ dB_t^{(2)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \theta_6 & \sqrt{1-\theta_6^2} \end{bmatrix} \begin{bmatrix} dW_t^{(1)} \\ dW_t^{(2)} \end{bmatrix}. \tag{2.6.13}$$

Thus the appropriate diffusion tensor for the Heston model is given by:

$$\begin{bmatrix} \theta_2\sqrt{V_t} & 0 \\ 0 & \theta_5\sqrt{V_t} \end{bmatrix} \begin{bmatrix} 1 & \theta_6 \\ \theta_6 & 1 \end{bmatrix} \begin{bmatrix} \theta_2\sqrt{V_t} & 0 \\ 0 & \theta_5\sqrt{V_t} \end{bmatrix}' = \begin{bmatrix} \theta_2^2 V_t & \theta_2\theta_5\theta_6 V_t \\ \theta_2\theta_5\theta_6 V_t & \theta_5^2 V_t \end{bmatrix}. \tag{2.6.14}$$

Under the GQD framework, the Heston model can be defined in R using the code:

```
R> GQD.remove()
R> a00 <- function(t){theta[1]}
R> a01 <- function(t){-0.5 * theta[2] * theta[2]}
R> c01 <- function(t){theta[2] * theta[2]}
R> d01 <- function(t){theta[2] * theta[5] * theta[6]}
R> b00 <- function(t){theta[3]}
R> b01 <- function(t){-theta[4]}
R> e01 <- function(t){theta[2] * theta[5] * theta[6]}
R> f01 <- function(t){theta[5] * theta[5]}
```

Note the inclusion of the vector `theta` in the coefficients: In the **DiffusionRgqd** package the variable name `theta` is reserved for the parametrisation of coefficient functions. This is done so as to facilitate the modelling process in which case all components apart from the vector `theta` remain fixed.

The next step is to prepare the time series input for the analysis. In order to execute any of the estimation routines, we need to supply a column matrix with the observed trajectories as well as a numerical vector containing the time points at which the observations were made. For the SPX data we shall assume that we are working on an annual time scale (i.e., one unit of time is equivalent to one year) with a 365-day time convention:

```
R> X    <- cbind(log(St), (Vt / 100)^2)
R> time <- cumsum(c(0, diff(as.Date(time1)) * (1 / 365)))
```

Once the data is prepared we call the `BiGQD.mle()` routine in order to calculate maximum likelihood estimates of the parameter vector `theta`. As in the previous examples, the `Bi`-prefix to the GQD-function is used to distinguish between scalar and bivariate diffusions, whilst the suffix `.mle` refers to the purpose of the routine. In order to perform maximum likelihood estimation we supply `BiGQD.mle()` with the data matrix `X`, the time-vector `time` and some starting parameters:

```
R> theta.start <- c(0, 1, 1, 0.5, 1, 0)
```

```
R> model_1     <- BiGQD.mle(X, time, mesh = 10, theta = theta.start)
```

An important distinction to make between estimation routines in the package such as `BiGQD.mle()` and routines like `GQD.density()` used in the previous examples is the use of C++ code. As outlined in Section 2.4, where multiple evaluations of the likelihood are to be carried out using the cumulant truncation procedure, the estimation routines will set up and compile a C++ routine tailored to the current diffusion model. As such, when an estimation routine is called for the first time for a given set of model coefficients, there might be some initial delay before the estimation routine commences. Naturally, provided the model coefficients remain the same, a second run would not be subject to this delay. For the `BiGQD.mle()` routine, once the applicable C++ code is compiled, a summary of the model is printed and maximisation of the likelihood function commences using R's built in optimization routine `optim()`. After execution of the optimization routine, some additional information about the model is appended to the summary. The resulting output looks something like:

```
================================================================
                  GENERALIZED QUADRATIC DIFFUSION
================================================================
_____________________ Drift Coefficients ________________________
a00 : theta[1]
a01 : -0.5*theta[2]*theta[2]
...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
b00 : theta[3]
b01 : -theta[4]
___________________ Diffusion Coefficients ______________________
c01 : theta[2]*theta[2]
...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
d01 : theta[2]*theta[5]*theta[6]
...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
e01 : theta[2]*theta[5]*theta[6]
...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
f01 : theta[5]*theta[5]

___________________________ Model Info ___________________________
Time Homogeneous     : Yes
Data Resolution      : Homogeneous: dt=0.0192
# Removed Transits. : None
Density approx.      : 4th Ord. Truncation, Bivariate-Saddlepoint
Elapsed time         : 00:00:18
...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
dim(theta)           : 6
----------------------------------------------------------------
```

The textual output serves both as a visual buffer between models in the workspace as well as a visual check on whether the model that has been recognised by the

`BiGQD.mle()` function corresponds to what the user intended it to be. In the workspace itself, `BiGQD.mle()` returns a list object containing information about the optimization run as well as peripheral information about the model. In order to extract the resulting parameter estimates one may use the `GQD.estimates()` function, which will return a summary of the parameter estimates resulting from the estimation procedure:

```
R> GQD.estimates(model_1)
```

```
         Estimate Lower_95 Upper_95
theta[1]    0.083    0.031    0.135
theta[2]    0.769    0.740    0.799
theta[3]    0.167    0.128    0.206
theta[4]    3.820    2.814    4.826
theta[5]    0.431    0.414    0.447
theta[6]   -0.670   -0.700   -0.641
```

The Heston model thus provides insight into the dynamics of the SPX and its corresponding variance dynamics. Indeed, the resulting parameter estimates confirm well-documented features such as strong negative correlation in the noise of the SPX and VIX series as evidenced by the correlation parameter θ_6. By modifying the Heston model, we can easily perform a more detailed analysis by including other peripheral data such as risk-free rates and dividends in order to formulate the model in the market context (see for example Hurn *et al.*, 2015). However, we defer such analysis in favour of answering a more general question of whether the Heston model can be improved upon for the data at hand. For example, one might argue that the variance of both R_t and V_t should scale quadratically ($\theta_2^2 V_t^2$ and $\theta_5^2 V_t^2$) as opposed to linearly ($\theta_2^2 V_t$ and $\theta_5^2 V_t$) with respect to the state of the variance process, as in the case of the Heston model. That is, we postulate that R_t may be more sensitive to changes in volatility whilst the volatility process itself exhibits stronger state-dependence in its volatility. To test this, we can fit the model:

$$\begin{aligned} dR_t &= (\theta_1 - \frac{1}{2}\theta_2^2 V_t^2)dt + \theta_2\sqrt{V_t^2}dB_t^{(1)} \\ dV_t &= (\theta_3 - \theta_4 V_t)dt + \theta_5\sqrt{V_t^2}dB_t^{(2)}. \end{aligned} \tag{2.6.15}$$

Note that, apart from the terms θ_1, θ_3 and $-\theta_4 V_t$, there are no common terms between the previous model and the new model. Consequently, in order for `BiGQD.mle()` to ignore the previous model coefficients we need to remove the previous model coefficients from the workspace. This is achieved by making use of the `GQD.remove()` function:

```
R> GQD.remove()
```

```
[1] "Removed :  a00 a01 b00 b01 c01 d01 e01 f01"
```

```
R> a00 <- function(t){theta[1]}
R> a02 <- function(t){-0.5 * theta[2] * theta[2]}
R> c02 <- function(t){theta[2] * theta[2]}
R> d02 <- function(t){theta[2] * theta[5] * theta[6]}
R> b00 <- function(t){theta[3]}
R> b01 <- function(t){-theta[4]}
R> e02 <- function(t){theta[2] * theta[5] * theta[6]}
R> f02 <- function(t){theta[5] * theta[5]}
R>
R> theta.start <- c(0, 1, 1, 1, 1, 0)
R> model_2     <- BiGQD.mle(X, time, mesh = 10, theta = theta.start)
```

In order to compare model fit for the various models we compare AIC and BIC statistics for the two models. This can be achieved by compiling all of the model outputs `model_1`, `model_2` etc. into a list and passing the result as an argument to `GQD.aic()`. `GQD.aic()` will then produce a summary of AIC, BIC and likelihood statistics for all the listed models along with the number of parameters, the number of observations, and a convergence diagnostic from the `optim()` output.

```
R> GQD.aic(list(model_1, model_2))
```

```
        Convergence p min.likelihood             AIC             BIC    N
Model 1           0 6       -7852.21       -15692.42       -15661.38 1305
Model 2           0 6       -7965.99   [=] -15919.99   [=] -15888.95 1305
```

Based on the AIC and BIC statistics, the revised model does indeed improve upon the Heston model with respect to model fit. The result suggests that the volatility structure of the index and variance process is more sensitive to changes in volatility than is predicted by the Heston model. Indeed, the volatility structure of revised model need not be the only improvement that can be made. We may also postulate various forms of the drift interaction and compare AIC statistics accordingly. For these purposes, we consider three additional models:

$$\begin{aligned} dR_t &= -\frac{1}{2}\theta_1^2 V_t^2 dt + \theta_1 \sqrt{V_t^2} dB_t^{(1)} \\ dV_t &= (\theta_2 - \theta_3 V_t)dt + \theta_4 \sqrt{V_t^2} dB_t^{(2)}, \end{aligned} \tag{2.6.16}$$

which removes the constant term from the drift of Equation 2.6.15 resulting in a slightly simpler model that presents a drift free index on the original SPX scale

(S_t),

$$\begin{aligned} dR_t &= (\theta_1 + \theta_7 R_t - \frac{1}{2}\theta_2^2 V_t^2)dt + \theta_2\sqrt{V_t^2}dB_t^{(1)} \\ dV_t &= (\theta_3 - \theta_4 V_t)dt + \theta_5\sqrt{V_t^2}dB_t^{(2)}, \end{aligned} \tag{2.6.17}$$

which modifies the drift of Equation 2.6.15 to include an R_t-term, and finally we include drift feedback of the SPX into the VIX series using the model:

$$\begin{aligned} dR_t &= (\theta_1 - \frac{1}{2}\theta_2^2 V_t^2)dt + \theta_2\sqrt{V_t^2}dB_t^{(1)} \\ dV_t &= (\theta_3 - \theta_4 V_t + \theta_7 R_t)dt + \theta_5\sqrt{V_t^2}dB_t^{(2)}. \end{aligned} \tag{2.6.18}$$

Comparing the AIC and BIC values of Equation 2.6.12 and equations 2.6.15 through 2.6.18, we conclude that the revised model still fits the data best, whilst Equation 2.6.17 produces very similar AIC and BIC statistics:

```
R> GQD.aic(list(model_1, model_2, model_3, model_4, model_5))
```

```
        Convergence p min.likelihood            AIC            BIC    N
Model 1           0 6       -7852.21       -15692.42       -15661.37 1305
Model 2           0 6       -7965.99   [=] -15919.91   [=] -15888.95 1305
Model 3           0 5       -7948.55       -15887.10       -15861.23 1305
Model 4           0 7       -7965.96       -15917.92       -15881.71 1305
Model 5           0 7       -7955.93       -15897.87       -15861.65 1305
```

Looking at the parameter estimates of Equation 2.6.17, this can possibly be explained by the similarity of the fitted drift of equations 2.6.15 and 2.6.17 since the parameter estimate of θ_7 appears to be not significantly different from 0:

```
R> GQD.estimates(model_4)
```

```
         Estimate Lower_95 Upper_95
theta[1]    0.015   -0.392    0.423
theta[2]    4.255    4.095    4.415
theta[3]    0.087    0.059    0.115
theta[4]    2.699    1.615    3.783
theta[5]    1.744    1.681    1.807
theta[6]   -0.648   -0.678   -0.617
theta[7]    0.017   -0.043    0.077
```

2.6.5 Model selection for 2D diffusions via DIC

Although stochastic volatility models have been used with great success in financial contexts, the application of non-linear diffusion models extends to other fields of

science as well. Naturally, the context in which such models are applied presents its own challenges. Indeed, finance presents a somewhat ideal problem set in the sense that data can be sourced at very high resolution for regularly spaced observations over long periods of time. In fields such as ecology, for example, the data is often recorded irregularly and rarely exceed monthly observation frequencies (see for example Varughese (2011) and Varughese and Pienaar (2013)). That said, with the rapidly changing landscape of ecological monitoring, such datasets are likely to improve over time. Perhaps the principal difference between financial data and *naturally* occurring processes is the presence of dominating seasonal effects such as changes in temperature and/or long-term growth trends. For these purposes, we have developed the **DiffusionRgqd** package to accommodate time-inhomogeneous coefficients for generalised quadratic diffusion processes.

In order to demonstrate the ability of the package to identify and distinguish between temporal patterns for time-inhomogeneous models, we set up experimental data by simulating trajectories from a time-inhomogeneous bivariate GQD with stochastic volatility. For purposes of the experiment we shall simulate a target SDE

$$\begin{aligned} dX_t &= (1.0(7.5 - X_t) + 1.5Y_t)dt + 0.5\sqrt{X_t Y_t}dB_t^{(1)} \\ dY_t &= (1.5(5 - Y_t) + 3\sin(0.25\pi t))dt + 0.25\sqrt{Y_t}dB_t^{(2)} \end{aligned} \tag{2.6.19}$$

by performing Euler-Maruyama updates of the target diffusion and recording the trajectory at regularly spaced time points. For the current example, updates were made at a resolution of $\Delta = 1/2000$ after which every 500-th update was recorded for a total of 401 observations, thus resulting in equidistant transition horizons of $1/4$ time units. For convenience we have included a simulated trajectory of the target model in the package's datasets, which may be called into the workspace in standard fashion using the `data()` command:

```
R> data("SDEsim4")
R> attach(SDEsim4)
R> plot(Xt~time, type = 'l', col = 'blue', ylim = c(0, 30),
+   main = 'Simulated Data', xlab = 'Time (t)', ylab = 'State')
R> lines(Yt~time, col = 'red')
```

Figure 2.6.5 shows a plot of the resulting time series. For purposes of the experiment we shall fit three competing models of which one is the true data generating process. The competing models are chosen to have superficially similar dynamics so as to test the ability of the scheme to distinguish between various models. For brevity we shall assume that we know from the outset that the process exhibits sinusoidal dynamics with known periodicity. This is not unrealistic as in practice it is usually clear from the context what the periodicity of the process is relative to the time scale on which the data were observed. For the first model,

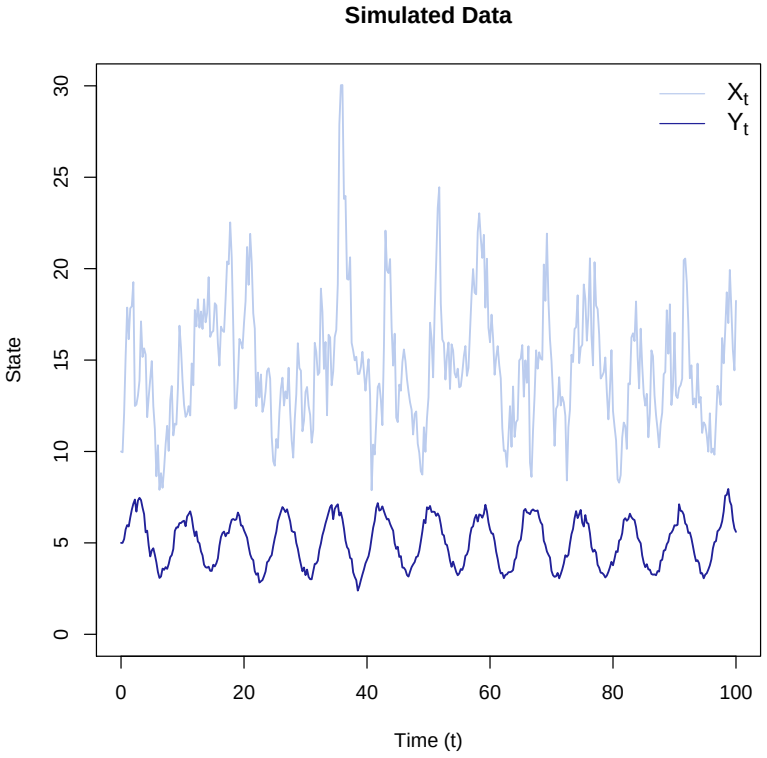


FIGURE 2.6.5: A simulated trajectory of Equation 2.6.19. The X_t trajectory (light blue) exhibits periodic increases in state and volatility in conjunction with Y_t (dark blue). The trajectory of Y_t exhibits dominating sinusoidal dynamics with little volatility. R code: Supplementary materials, Section 2.5.

we shall assume sinusoidal time-dependence for the drift of the y-dimension with interaction occurring only through the diffusion terms:

$$\begin{aligned} dX_t &= (\theta_1\theta_2 - \theta_1 X_t)dt + \theta_3\sqrt{X_t Y_t}dB_t^{(1)} \\ dY_t &= (\theta_4\theta_5 - \theta_4 Y_t + \theta_7 \sin(0.25\pi t))dt + \theta_6\sqrt{Y_t}dB_t^{(2)}. \end{aligned} \tag{2.6.20}$$

For the second model, we assume that the X_t and Y_t trajectories arise independently with the oscillatory dynamics resulting purely from time-inhomogeneity:

$$\begin{aligned} dX_t &= (\theta_1\theta_2 - \theta_1 X_t)dt + \sqrt{\theta_7(1+\sin(0.25\pi t)) + \theta_3^2 X_t}dB_t^{(1)} \\ dY_t &= (\theta_4\theta_5 - \theta_4 Y_t + \theta_8 \sin(0.25\pi t))dt + \theta_6\sqrt{Y_t}dB_t^{(2)}. \end{aligned} \tag{2.6.21}$$

Finally, for the third candidate, we assume the true model:

$$\begin{aligned} dX_t &= (\theta_1\theta_2 - \theta_1 X_t + \theta_7 Y_t)dt + \theta_3\sqrt{X_t Y_t}dB_t^{(1)} \\ dY_t &= (\theta_4\theta_5 - \theta_4 Y_t + \theta_8 \sin(0.25\pi t))dt + \theta_6\sqrt{Y_t}dB_t^{(2)}. \end{aligned} \tag{2.6.22}$$

In R, we can specify the first candidate model as per the usual GQD syntax:

```
R> GQD.remove()
R> a00 <- function(t){theta[1] * theta[2]}
R> a10 <- function(t){-theta[1]}
R> c11 <- function(t){theta[3] * theta[3]}
R> b00 <- function(t){theta[4] * theta[5]+theta[7] * sin(0.25 * pi * t)}
R> b01 <- function(t){-theta[4]}
R> f01 <- function(t){theta[6] * theta[6]}
```

After the model has been specified we may infer parameter estimates using the `BiGQD.mcmc()` routine. As outlined in Section 2.4, `BiGQD.mcmc()` sets up a computationally efficient likelihood evaluation routine in `C++` code which is subsequently called within a random walk Metropolis-Hastings (RWMH) algorithm in order to produce sample chain of the model parameters under a given model specification. Note that the methodology applies to more general Bayesian schemes as well. In the software we simply make use of symmetric proposals for ease of implementation and to avoid the complications associated with specifying different proposal distributions for different parameters.

The algorithm as implemented by `BiGQD.mcmc()` follows:

1. Given some starting parameter vectors $\boldsymbol{\theta} = (\theta_i)_{p\times 1}$ and $\boldsymbol{\sigma} = (\sigma_i)_{p\times 1}$, set $\boldsymbol{\theta}^{old} = \boldsymbol{\theta}$ and

2. propose an update for the parameter vector by setting

$$\theta_i^{new} = \theta_i^{old} + Z_{\sigma_i} \tag{2.6.23}$$

   for all $i = 1, 2, \ldots p$, where Z_{σ_i} are $\mathrm{N}(0, \sigma_i^2)$ random deviates.

3. Subsequently, evaluate the ratio:

$$R = \frac{\prod_{i=1}^{N-1} f(\mathbf{X}_{t_{i+1}}|\mathbf{X}_{t_i}, \boldsymbol{\theta}^{new})\pi(\boldsymbol{\theta}^{new})}{\prod_{i=1}^{N-1} f(\mathbf{X}_{t_{i+1}}|\mathbf{X}_{t_i}, \boldsymbol{\theta}^{old})\pi(\boldsymbol{\theta}^{old})} \tag{2.6.24}$$

   where $\pi(\boldsymbol{\theta})$ denotes a prior density on the parameter vector.

4. Then accept the proposed move with probability $\min(R, 1)$. That is, set

$$\theta^{old} = \begin{cases} \theta^{new} & \text{if } \min(R,1) > u \\ \theta^{old} & \text{otherwise,} \end{cases} \tag{2.6.25}$$

   where u is a $\mathrm{U}(0,1)$ random deviate.

5. Return to step 2.

In R, the parameters of the algorithm are passed as arguments to `BiGQD.mcmc()`. For example, the starting parameters corresponding to the vector $\boldsymbol{\theta}$ may be set using the `theta` argument, while the proposal standard deviations $\boldsymbol{\sigma}$ may be set using the `sds` argument. Other arguments pertaining the density approximation such as the time-mesh, step size, and density type may be specified as with `BiGQD.mle()` in the previous example. The algorithm is then repeated a specified number of times as per the argument `updates`.

Apart from the parametric nuances introduced by the Metropolis-Hastings algorithm, another point that warrants attention is the inclusion of prior densities. For all of the models above we shall assume prior densities on the parameters θ_1 and θ_4 with $\theta_1 \sim \mathrm{N}(1, 5^2)$ and $\theta_4 \sim \mathrm{N}(1, 5^2)$. These densities can be included in the R environment by defining a function named `priors()` (a function name that will be recognised by `BiGQD.mcmc()` in similar fashion to the model coefficients) taking the vector `theta` as an argument. The function body can then be written as a product of the desired prior densities using standard R density functions such as `dnorm()`, `dgamma()` etc. or any other user defined density function:

```
R> priors <- function(theta){dnorm(theta[1], 1, 5) * dnorm(theta[4], 1, 5)}
```

After defining a model and the prior densities to be used in the analysis we can make an initial run of the RWMH algorithm by providing a starting parameter vector, standard deviation vector for the proposal densities and the number of RWMH updates to perform:

```
R> mesh    <- 10
R> updates <- 150000
R> burns   <- 50000
R> X       <- cbind(Xt, Yt)
R> th      <- c(5, 5, 5, 5, 5, 5, 5)
R> par.sds <- c(0.22, 0.30, 0.02, 0.11, 0.04, 0.01, 0.21)
R> m1      <- BiGQD.mcmc(X, time, mesh, th, par.sds, updates, burns)
```

When the calculations are complete, `BiGQD.mcmc()` will return a list with the resulting MCMC chain, the history of acceptance rates and other information

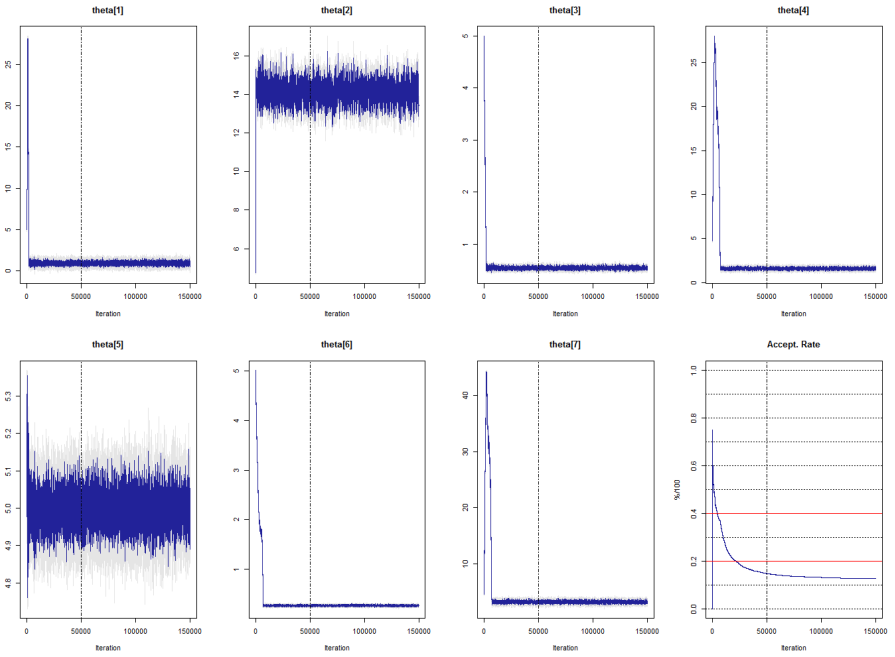


FIGURE 2.6.6: A trace plot of the parameter updates for model 2.6.20 generated by `BiGQD.mcmc()`. A trace plot is drawn for each parameter of the target model in conjunction with rejected proposals (light gray). The vertical dashed lines indicate the end of the burn-in period. In addition, a trace plot of the acceptance rate of the RWMH algorithm is drawn with guide lines ranging from 0% to 100% in increments of 10% (20% and 40% highlighted in red). R code: Supplementary materials, Section 2.5.

pertaining to the model such as likelihood and DIC statistics. By default, `BiGQD.mcmc()` will also make a trace plot of the resulting Markov chain as illustrated in Figure 2.6.6.

If the resulting trace plot is satisfactory, estimates for the model can be calculated by passing the model object to the `GQD.estimates()` function as in the previous example. `GQD.estimates()` will recognise the model as MCMC output and calculate parameter estimates by discarding the first `burns` iterations and thinning the chain by a specified amount (using the argument `thin`). Subsequently, the resulting parameter estimates, 90% credibility intervals and correlation matrix is printed to the console. For diagnostic purposes an ACF-plot for each element of the parameter chain is plotted:

```
R> GQD.estimates(m1, thin = 200)
```

```
          Estimate Lower_CI Upper_CI
```

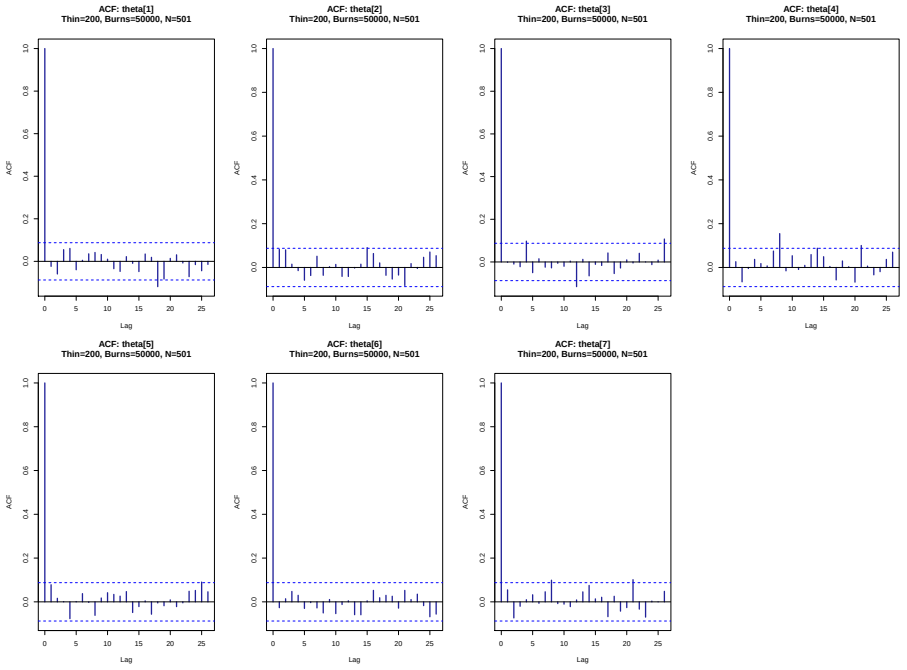


FIGURE 2.6.7: Autocorrelation function (ACF) plot for the thinned parameter chain of Equation 2.6.20. The ACF plot is produced when `GQD.estimates()` is called. R code: Supplementary materials, Section 2.5.

```
theta[1]    0.955    0.707    1.218
theta[2]    14.15   13.315   14.981
theta[3]    0.539    0.507    0.574
theta[4]    1.591    1.447     1.74
theta[5]    5.010    4.938    5.064
theta[6]    0.264     0.25    0.281
theta[7]    3.222     2.97    3.475
```

Should a more detailed analysis of the parameter chains be required, this can easily done using using packages such as **coda** (Plummer *et al.*, 2006) which provide a comprehensive set of MCMC analysis tools. For example, `mcmc.coda <- mcmc(m1$par.matrix)` creates an `mcmc` object that will be recognised by routines in the **coda** library.

In order to assess convergence of the RWMH algorithm it is standard practice to run multiple chains from different starting points. Although this is typically achieved by manually running the RWMH algorithm multiple times, the practice warrants at least some degree of automation. For these purposes we have included

a fail-over variable in the output of the `BiGQD.mcmc()` function. That is should `BiGQD.mcmc()` fail for any reason, whether it be unrealistic initial values or some numerical failure, a variable is returned in the output list indicating that failure has occurred. This variable may thus be monitored within a loop, making it easy to make repeated MCMC runs automatically. In addition to the fail-over variable we have included a tagging argument which can be used to mark an instance of a call to `BiGQD.mcmc()`. In the current example each model is assigned the tag '`Model A_run_()`' with the convention that `()` keeps track of which run the object pertains to. Using randomly chosen starting points one can thus perform a number of MCMC runs and store the output in a list object, for example `SaveOutput` in the code below:

```
R> M  <- 4
R> SaveOutput  <- list()
R> M.counter   <- 1
R> Tot.counter <- 1
R> kill.count  <- 1
R> while((M.counter <= M) & (kill.count < 20))
+ {
+   th <- runif(7, 1, 10)
+   m1 <- BiGQD.mcmc(X, time, mesh, th, par.sds, updates, burns,
+                    Tag = paste0('Model_A_run_', M.counter))
+   if(!m1$failed.chain)
+   {
+     SaveOutput[[Tot.counter]]  <-  m1
+     Tot.counter <- Tot.counter + 1
+     M.counter   <- M.counter + 1
+     kill.count  <- 1
+   }else
+   {
+     kill.count  <- kill.count + 1
+   }
+ }
```

By repeating this process we can conduct inference on the second and third model as well, and subsequently fill the `SaveOutput` list with consecutive MCMC runs for ach model. Once all the MCMC runs are concluded we may summarise the output via the `GQD.dic()` function. `GQD.dic()` collates DIC statistics and other relevant information from a list of `BiGQD.mcmc()` output objects in a concise form. For example, when the routine is applied to the `SaveOutput` variable. The minimum DIC model is then indicated by a `[=]` marker.

```
R> GQD.dic(SaveOutput)
```

```
              Elapsed_Time Time_Homogeneous       p         DIC      pD   N
Model_A_run_1     00:09:14               No    7.00      1640.18    7.10 401
Model_A_run_2     00:08:49               No    7.00      1640.04    7.04 401
```

```
Model_A_run_3     00:08:34              No    7.00      1640.04    7.04 401
Model_A_run_4     00:08:04              No    7.00      1640.11    7.07 401
Model_B_run_1     00:09:42              No    8.00      1649.66    7.86 401
Model_B_run_2     00:09:21              No    8.00      1649.65    7.83 401
Model_B_run_3     00:09:21              No    8.00      1650.00    8.03 401
Model_B_run_4     00:09:21              No    8.00      1650.10    8.07 401
Model_C_run_1     00:08:21              No    8.00  [=] 1618.95    7.78 401
Model_C_run_2     00:08:28              No    8.00      1619.48    8.05 401
Model_C_run_3     00:08:23              No    8.00      1619.64    8.12 401
Model_C_run_4     00:08:34              No    8.00      1619.67    8.14 401
```

From the DIC calculations, the algorithm does indeed narrow down the correct model. Finally, we can compare the estimated parameters of Equation 2.6.22 to the true model parameters by applying the `GQD.estimates()` function to the final MCMC run:

```
R> GQD.estimates(SaveOutput[[12]], thin = 200)
```

```
          Estimate Lower_CI Upper_CI
theta[1]     1.162    0.885    1.477
theta[2]     6.873    4.176    9.752
theta[3]      0.54    0.507    0.578
theta[4]     1.584    1.434     1.75
theta[5]     4.997     4.94    5.059
theta[6]     0.264    0.247    0.281
theta[7]     1.787    1.124    2.422
theta[8]     3.206    2.958    3.507
```

Given the relatively low sample resolution of the simulated dataset, the parameter estimates compare favourably with the true parameter set. Interestingly, the credibility intervals for the estimates of θ_2 and θ_7 are quite wide, whilst those for the diffusion parameters θ_3 and θ_6 are relatively narrow. This suggests that for a diffusion process with dynamics governed by Equation 2.6.19, the current observed trajectory is relatively short, making it difficult to accurately measure the drift dynamics. Indeed, this is an important practical consideration when conducting inference on diffusion models: Since one can only partially observe a single trajectory of the process over a finite horizon, the quality of inference that can be made on any given model is necessarily dictated by the regularity with which observations are made and the length of the observation horizon. Thus, for simple models such as Equation 2.6.20, there is often more certainty about what parameter values could have likely resulted in the observed trajectory than for more complicated models such as equations 2.6.21 and 2.6.22.

2.7 Chapter summary

By making use of a computationally efficient algorithm for calculating the transition density of a diffusion process we develop a class of scalar and bivariate quadratic diffusion processes termed the generalised quadratic diffusion processes. Using this framework we develop the **DiffusionRgqd** package: A collection of tools for performing inference and analysis on time-inhomogeneous quadratic diffusion processes. By identifying computational redundancies within the generalised quadratic class it is possible to define an algorithm whereby computational overhead of calculating the transitional density can be minimised. Using this architecture one can accurately and efficiently analyse diffusion models through their transitional density. This efficiency is further improved by constructing computationally intensive elements of the algorithm in C++ code which can subsequently be called within R. This allows us to perform iterative tasks such as evaluating the likelihood function of a non-linear diffusion model efficiently whilst retaining the ease of use afforded by the R language. As such, the package can be used to efficiently perform inference on discretely observed diffusion processes in a non-parallel computing environment. Furthermore, by framing the algorithm within a suitably general class of models, it is possible to separate the user from the mathematical elements of the scheme, thus making it possible to analyse non-linear, time-inhomogeneous diffusion models with minimal mathematical input over and above defining a model of interest.

Although the focus of this chapter has been primarily on the computational aspects of analysing diffusion processes in the software environment, the GQD-framework provides a flexible basis for analysing more general processes in a computationally efficient manner. In the chapters that follow, we show how the methodology can be generalised and applied to more complex problems in the analysis of non-linear diffusion processes.

Chapter 3

First Passage Time Problems

3.1 Introduction

A great deal of literature is dedicated to modelling real-world processes using various forms of stochastic differential equations. Typically, the analysis is framed in the context of time series analysis where a diffusion model is used to describe the dynamics of a process based on observations of the process trajectory over time. In some contexts, the objective of the analysis may be to model the time until an event of interest occurs. That is, instead of focusing on the dynamics of the trajectory of the process, we are interested in the so-called 'first passage time' to some predefined event on the trajectory of the process, for example the time it takes for interest rates to exceed or decline below a certain level, or the time to extinction of a given population process. Thus, where the underlying process can be modelled using a diffusion model, it can be extremely useful to analyse the first passage time of the model diffusion trajectory to such an event. For example, an idea that has been explored at length in the literature is that of modelling the trajectory of the 'membrane potential' of a neuron over time as a diffusion process, after which the firing time of the model neuron is treated as the first passage time of the potential to a given threshold under various threshold regimes. Thus, instead of focusing on the trajectory of the process itself, the analysis is concerned with the first passage time to the threshold potential under a given diffusion model. Subsequently, based on the distribution of the first passage time, the parameters of the underlying diffusion model can be estimated. Depending on the specification of the diffusion model and/or the threshold function, the characteristics of the first passage time distribution may change. Ricciardi and Sacerdote (1979) treat the membrane potential as an Ornstein-Uhlenbeck process with respect to a constant threshold and proceed to

calculate the moments of the first passage time density of interest. Over time various models and inference techniques for this problem have been explored by authors such as Inoue *et al.* (1995); Ditlevsen and Lansky (2005, 2006, 2007) and Zhang *et al.* (2009), mostly focusing on time-homogeneous Ornstein-Uhlenbeck processes as a model of the underlying membrane potential. Later, Iolov *et al.* (2014) added a sinusoidal drift component to the underlying diffusion model in order to account for the oscillatory behaviour observed in certain circumstances. First passage time problems have also been applied in fields such as economics, where for example Gutiérrez *et al.* (1999) focus on the application of a class of time-inhomogeneous log-Normal processes to various economic variables. Indeed, the analysis of first passage times plays an important role in the valuation of certain types of financial derivatives (Black and Cox, 1976). Despite this, the application of first passage time problems in this regard typically concern simple diffusion models such as geometric Brownian motion that are often used in finance. Where the financial process being modelled is more accurately modelled by a more complicated diffusion model, analysis of the first passage time distribution can lead to significantly more accurate valuations of derivative securities.

Perhaps the most common form of first passage time problem in the literature concerns the distribution of the time elapsed until a scalar diffusion crosses a single predefined threshold function as illustrated in Figure 3.1.1. That is, we wish to calculate the distribution of the first passage time variable:

$$T_{X_s \to \lambda_t} = \inf\{t > s : X_t \geq \lambda_t\}, \tag{3.1.1}$$

where X_t has dynamics

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dB_t \tag{3.1.2}$$

and λ_t denotes the threshold function.

Fortunately, for threshold events of the form described above, a *simple* expression exists which relates the transitional density of the diffusion process to that of the first passage time density: Given a diffusion process X_t with transitional density $f(X_t|X_s)$, the distribution

$$g_{X_s \to \lambda_t}(t) = \frac{\partial}{\partial t} P(T_{X_s \to \lambda_t} \leq t) \tag{3.1.3}$$

of the first passage time of a process starting in initial state X_s, to the threshold function λ_t, is given by the first-kind Volterra equation (Ricciardi *et al.*, 1984):

$$f(X_t = \lambda_t|X_s) = \int_s^t f(X_t = \lambda_t|X_u = \lambda_u) g_{X_s \to \lambda_t}(u)du. \tag{3.1.4}$$

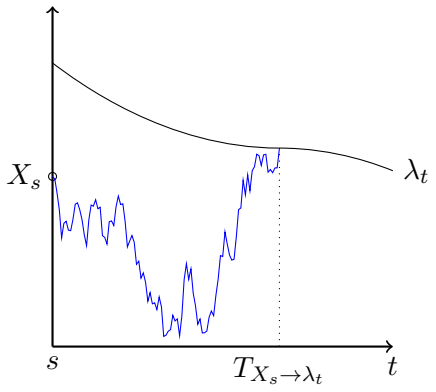


FIGURE 3.1.1: First passage time $T_{X_s \to \lambda_t}$ of a scalar diffusion (solid blue), starting in state X_s, to crossing a time-varying threshold function λ_t (solid black).

As in the case of the transitional density, the analysis of first passage time densities proves to be extremely difficult. Indeed, it is clear from Equation 3.1.4 that these difficulties stem from both the intractability of the transition density $f(X_t|X_s)$ and the encapsulating integral equation. Even when the transition density is available analytically, solutions to Equation 3.1.4 are not guaranteed to exist. As such, the analysis of first passage time problems has often been limited to problems with features that aid the calculation of analytical solutions. Consequently, when first passage time densities are used in practice, researchers often have to resort to using diffusion models with analytically tractable first passage time densities. In such cases, both the dynamics of the process and the shape of the threshold function are dictated *a priori* to conducting the analysis and may not necessarily be an accurate representation of the underlying process. In the sections that follow, we develop schemes for analysing first passage time problems of various forms for which the diffusion process may be non-linear and/or time-inhomogeneous.

3.2 Approximatng the first passage time density by numerical solution of the Volterra equation

In order to calculate the first passage time density of a scalar diffusion process crossing through a single predefined threshold level, we are required to find functions $f(.)$ and $g(.)$ that satisfy Equation 3.1.4. Unfortunately, establishing analytical expressions that do so is extremely difficult and can usually only be

achieved for very simple models and/or threshold functions. For example, in the case of an Ornstein-Uhlenbeck process:

$$dX_t = a(b - X_t)dt + \sigma dB_t \tag{3.2.1}$$

with initial value X_0, it can be shown that the first passage time to its long-run mean b, is given by the expression (Leblanc *et al.*, 2000):

$$g_{X_0 \to b}(t) = \\ \sigma^2 \frac{|b - X_0|}{\sqrt{2\pi}} \left(\frac{a}{\sigma^2 \sinh(at)} \right)^{3/2} \exp\left(\frac{a}{2\sigma^2} \left((X_0 - b)^2 + \sigma^2 t - (b - X_0)^2 \coth(at) \right) \right). \tag{3.2.2}$$

This is an example of a special circumstance where the first passage time density can be derived analytically. Indeed, this result is closely related to the first passage time density of a much simpler process, namely that of the first passage time of Brownian motion to a fixed threshold (Alili *et al.*, 2005). Despite the first passage time density being available in closed form, altering the parameters of the problem even slightly, for example by shifting the threshold level away from the long-run mean b renders the problem analytically intractable. It can perhaps be argued that this limitation stems from extrapolating the Brownian motion result to a more general first passage time problem under very specific conditions. For example, Equation 3.2.2 can be used in similar fashion to derive a result for a CIR process: Let $Y_t = X_t^2$ where $b = 0$. It then follows from Itô's lemma that the dynamics of Y_t are given by

$$dY_t = (\sigma^2 - 2aY_t)dt + 2\sigma\sqrt{Y_t}dB_t. \tag{3.2.3}$$

Since the transformation $Y_t = X_t^2$ is one-to-one on the positive real line (to which the resulting CIR process is confined) we can calculate the first passage time of Y_t to the zero bound in terms of an Ornstein-Uhlenbeck process by equivalence. That is, we analyse

$$T_{Y_s \to 0} = \inf\{t > s : Y_t = 0\} \tag{3.2.4}$$

by equivalence through the variable

$$T_{X_s^2 \to 0} = \inf\{t > s : X_t^2 = 0\}, \tag{3.2.5}$$

for which we can apply Equation 3.2.2 in order to evaluate the first passage time density. This result thus exploits the link between the CIR process, the Ornstein-Uhlenbeck process, and Brownian motion in order to derive a closed form expression for the first passage time density. Unfortunately, we can only exploit the link here for the zero bound of Equation 3.2.3 since for non-zero b, the

dynamics that result from applying Itô's lemma no longer corresponds to that of a CIR process. In order to solve first passage time problems under more general specifications we thus require an alternative strategy for calculating the first passage time density. For example, when the transitional density of a diffusion process can be evaluated directly, we may resort to solving Equation 3.1.4 numerically. That is, despite the fact that Equation 3.1.4 cannot be solved analytically – even when the transitional density contained in the equation is available in closed-form – we may invert the integral equation numerically in order to evaluate the first passage time density. This can be achieved in a number of ways. Perhaps the simplest strategy is to derive an updating equation from the Volterra equation directly. That is, by approximating Equation 3.1.4 using standard quadrature rules (see for example Linz (1969)), one may derive an iterative updating equation which approximates the first passage time density numerically at fixed points along the transition horizon. The procedure is as follows: Split the time domain into $N+1$ equispaced time nodes $t_0 = s, t_1 = s+\Delta, \ldots, t_N = s+N\Delta$ for some time step $\Delta = (t_N - t_0)/N$ and set $g_{X_s \to \lambda_t}(t_0) = 0$. Then for $i = 1, \ldots, N$ we may evaluate:

$$g_{X_s \to \lambda_t}(t_i) = \frac{f(X_{t_i} = \lambda_{t_i} | X_{t_0}) - \sum_{k=0}^{i-1} f(X_{t_i} = \lambda_{t_i} | X_{t_k} = \lambda_{t_k}) g_{X_s \to \lambda}(t_k) \Delta}{f(X_{t_i} = \lambda_{t_i} | X_{t_{i-1}} = \lambda_{t_{i-1}}) \Delta}. \tag{3.2.6}$$

Using this, we may approximate the first passage time density to a given threshold function by iteratively solving Equation 3.2.6 over a predefined transition horizon. Unfortunately, despite the fact that this relatively simple updating structure may provide reasonably accurate results, there are subtle yet significant limitations to this approach. Due to the singular integrand of Equation 3.1.4 at s (Ricciardi and Sacerdote, 1979), numerical solutions such as Equation 3.2.6 may be subject to systematic error. Although these singularities occur at an infinitesimal scale, the effects are not inconsequential on a finite scale (i.e., under discretization of the time domain). In order to deal with this, Buonocore *et al.* (1987) developed an alternative integral equation wherein the kernel of the integral equation (i.e., the integrand $f(X_t = \lambda_t | X_u = \lambda_u) g_{X_s \to \lambda_t}(u)$) can be modified in order to remove the singularities prior to the calculation of the first passage time density. In this case, a second-kind Volterra integral equation for the first passage time density under Buonocore *et al.* (1987) is given by:

$$g_{X_s \to \lambda_t}(t) = 2\psi(\lambda_t | X_s) - 2 \int_s^t g_{X_s \to \lambda_t}(v) \psi(\lambda_t | \lambda_v) dv \tag{3.2.7}$$

for $X_s < \lambda_s$, where

$$\psi(x_t | y_t) = \frac{\partial}{\partial t} F(x_t | y_t) + k(t) f(x_t | y_s) + r(t)[1 - F(x_t | y_s)] \tag{3.2.8}$$

and $k(t)$ and $r(t)$ are chosen in such a way so as to ensure 'regularity' of the kernel. Although the revised equation rather elegantly circumvents the theoretical complications that arise from Equation 3.1.4, it does introduce some minor practical complications. Specifically, the functions $k(t)$ and $r(t)$ are not defined explicitly and may depend on the specification of the first passage time problem. Furthermore, note that $\psi(x_t|y_s)$ depends on both the transition p.d.f. and c.d.f. of X_t. For purposes of the scheme that follows, it would be ideal to avoid calculations that involve the c.d.f. when solving Equation 3.2.7. Fortunately, the literature provides us with a strategy for avoiding said practical complications without impeding on the scope of the analysis: Giorno *et al.* (1989) derive results pertaining to the choice of $r(t)$ and $k(t)$ such that Equation 3.2.7 can be used to evaluate first passage time problems for a suitably general class of processes. These results are then further generalised by Jáimez *et al.* (1995) to a class of time-inhomogeneous processes. Coincidently, the results lead to a means by which the c.d.f. can be removed from the calculation of Equation 3.2.8. For purposes of the analysis that follows, we outline how these results are applied and make explicit the effect on relevant elements of Equation 3.2.7.

Note that Equation 3.2.8 can be related to the probability current

$$c(x_t|y_s) = [\mu(X_t,t) - \sigma(x_t,t)\sigma'(x_t,t)]f(x_t|y_s) - \frac{1}{2}\sigma^2(x_t,t)f'(x_t|y_s) \tag{3.2.9}$$

where $\sigma'(x_t,t) = \frac{\partial}{\partial u}\sigma(u,t)|_{u=x_t}$ and $f'(x_t|y_s) = \frac{\partial}{\partial u}f(u|y_s)|_{u=x_t}$. By observing that (Buonocore *et al.*, 1987):

$$\frac{\partial}{\partial t}f(x_t|y_s) = -\frac{\partial}{\partial x_t}c(x_t|y_s), \tag{3.2.10}$$

and thus

$$\frac{\partial}{\partial t}F(x_t|y_s) = -c(x_t|y_s), \tag{3.2.11}$$

one may simplify Equation 3.2.8 by removing the derivative of the cumulative density function:

$$\psi(x_t|y_s) = -c(x_t|y_s) + k(t)f(x_t|y_s) + r(t)[1 - F(x_t|y_s)]. \tag{3.2.12}$$

Then, following Corollary 1 in Jáimez *et al.* (1995), set

$$r(t) = 0 \tag{3.2.13}$$

and

$$k(t) = \frac{1}{2}\left[\mu(\lambda_t,t) - \frac{\partial}{\partial t}\lambda_t - \frac{1}{2}\sigma(\lambda_t,t)\sigma'(\lambda_t,t)\right]. \tag{3.2.14}$$

Subsequently, Equation 3.2.12 may be evaluated in terms of the transition probability density function and its first derivative alone:

$$\psi(x_t|x_s) = -\left[\frac{1}{2}\mu(x_t,t) - \frac{1}{2}\frac{\partial}{\partial t}\lambda_t - \frac{3}{4}\sigma(x_t,t)\sigma'(x_t,t)\right]f(x_t|x_s) + \frac{1}{2}\sigma^2(x_t,t)f'(x_t|x_s). \tag{3.2.15}$$

Using this, we circumvent the need to evaluate $F(x_t|y_s)$ whilst ensuring regularity of the kernel of Equation 3.2.7. Thus, in order to evaluate the first passage time density we may equivalently solve the revised (second-kind) Volterra equation. However, since Equation 3.2.7 is still not analytically tractable in general, Buonocore *et al.* (1987) gives the discrete counterpart to the revised second-kind Volterra equation in the form of the iterative updating equation:

$$\tilde{g}_{X_s\to\lambda_t}(t_i) = -2\psi(\lambda_{t_i}|X_{t_0}) + \sum_{k=0}^{i-1} 2\psi(\lambda_{t_i}|\lambda_{t_k})\tilde{g}_{X_s\to\lambda_t}(t_k)\Delta, \tag{3.2.16}$$

for $i = 1, \ldots, N$ where again $t_0 = s, t_1 = s + \Delta, \ldots, t_N = s + N\Delta$ subject to the initial condition $\tilde{g}_{X_s\to\lambda_t}(t_0) = 0$. Thus, by analytically removing the singular kernel of the integral equation, one can circumvent numerical instability of the original updating equation and instead approximate the first passage time density using Equation 3.2.16.

3.2.1 First passage times for scalar GQDs

Although the revised numerical scheme of Equation 3.2.16 makes it possible to evaluate the first passage time density of a time-inhomogeneous diffusion to a time dependent threshold, the equation still relies on the calculation of the transitional density. In order to fully exploit Equation 3.2.16, we may analyse first passage time densities for non-linear diffusions by numerically calculating the transitional density and plugging the relevant approximations into Equation 3.2.15. Specifically, by incorporating the cumulant truncation procedure in the evaluation of the transitional densities on which the updating equation relies, it is possible to evaluate first passage time densities for polynomial diffusion processes. Let $\tilde{f}^{(d)}(x_t|x_s)$ and $\tilde{f}'^{(d)}(x_t|x_s)$ denote the density approximation and its first derivative under the cumulant truncation procedure under a d-th order truncation. Then, the revised probability current may be approximated by:

$$\begin{aligned}\tilde{\psi}^{(d)}(x_t|x_s) = &-\left[\frac{1}{2}\mu(x_t,t) - \frac{1}{2}\frac{\partial}{\partial t}\lambda_t - \frac{3}{4}\sigma(x_t,t)\sigma'(x_t,t)\right]\tilde{f}^{(d)}(x_t|x_s) \\ &+ \frac{1}{2}\sigma^2(x_t,t)\tilde{f}'^{(d)}(x_t|x_s).\end{aligned} \tag{3.2.17}$$

In order to calculate the revised probability current, we require evaluation of the derivative of the density approximation $\tilde{f}^{(d)}(x_t|x_s)$. Depending on the density approximation being used, this may be achieved in various ways. For example, under the saddlepoint approximation (see Section 2.3.3) this can be achieved as follows: Since α_0 in Equation 2.3.22 depends on x_t through Equation 2.3.23, it follows that:

$$\frac{\partial}{\partial x_t}\alpha_0 = \left(\sum_{i=2}^{d} \frac{\alpha_0^{i-2}}{(i-2)!}\kappa_i(t)\right)^{-1} \tag{3.2.18}$$

for $d \geq 2$. Furthermore, let $\alpha_0' = \frac{\partial}{\partial x_t}\alpha_0$ then the first derivative of the saddlepoint approximation can be verified as:

$$\begin{aligned}&\tilde{f}_{SPT}^{\prime(d)}(x_t|x_s) = \tilde{f}_{SPT}^{(d)}(x_t|x_s)\times \\ &\left[-\frac{1}{2\frac{\partial^2 K^{(d)}}{\partial \alpha^2}(\alpha_0,t)}\left(\sum_{i=3}^{d}\frac{\alpha_0^{i-3}\alpha_0'}{(i-3)!}\kappa_i(t)\right) + \left(\sum_{i=3}^{d}\frac{\alpha_0^{i-3}\alpha_0'}{(i-3)!}\kappa_i(t) - \alpha_0' x_t - \alpha_0\right)\right]\end{aligned} \tag{3.2.19}$$

for $d \geq 3$ (for $d = 2$, the appropriate expression can be found by calculating the first derivative with respect to x_t of the Normal distribution). Subsequently, using Equation 3.2.19 in conjunction with equations 3.2.16 and 3.2.17 we can numerically evaluate the first passage time density of time-inhomogeneous non-linear diffusions to a time dependent threshold using the cumulant truncation procedure.

By imposing the generalised quadratic structure on the specification of the diffusion process, we can use various forms of surrogate density to do so with a great deal of accuracy. Indeed, in order to evaluate the first passage time density using Equation 3.2.16 we require the transitional density to be accurate over large time scales. Under these circumstances, the cumulant truncation procedure affords both the necessary accuracy and efficiency in order to make such a calculation feasible. Although closed-form approximations such as those developed in for example Aït-Sahalia (1999) or Yu (2007) are not subject to any numerical overhead, the transition horizons to which these approximations apply make them unfit for application in this context. Likewise, although direct numerical techniques such as the method of lines allow us to calculate transitional densities very accurately for highly non-linear model specifications over arbitrarily large transition horizons, the computational overhead associated with such a strategy means that applying such an approximation in this context would only be computationally feasible in the time-homogeneous case.

3.2.2 Exploiting redundancies for computational efficiency in time-homogeneous intractable problems

Although combining a numerical approximation of the transitional density with the second-kind Volterra integral equation allows one to analyse quite general first passage time problems, the iterative nature of the updating scheme implies that the strategy may incur significant computational overhead: When the transitional density is available analytically, the bulk of the computational overhead results from the iteration of the updating equations in order to calculate the first passage time density up to and including some finite time limit t_N. However, when one evaluates the transitional density numerically, the bulk of the computational overhead goes into calculating the $(N^2 - N)/2 + 2N$ transition probabilities contained in equations 3.2.6 or 3.2.16 ($(N^2-N)/2+N$ for $f(\lambda_{t_j}|\lambda_{t_i}) : i < j, i, j = 1, 2, \dots N$ and another N for the elements $f(\lambda_{t_i}|X_{t_0}) : i = 1, 2, \dots N$). In this case, the size of the transition horizon and the resolution of the time discretization may drastically affect the computational efficiency of the respective updating equations: Assuming that the transition density approximation is sufficiently accurate, the accuracy of the updating equation is governed by the step-size used in the discretization of the time domain. Consequently, as the step-size is decreased and the number of time nodes increases, the number of operations required to evaluate the first passage time density increases quadratically. Likewise, assuming that a sufficiently small time step is used, the number of calculations required to evaluate the first passage time density increases quadratically as the size of the transition horizon is increased. As a consequence, evaluating the first passage time density can become quite time consuming. Fortunately, when the underlying diffusion process and threshold function are time-homogeneous, a number of simplifications arise that may be exploited in order to improve computational efficiency of the algorithm. To see this, consider Equation 3.2.6: By expanding

the RHS Equation 3.2.6, we find the triangular system of equations:

$$\begin{aligned}
g_{X_s\to\lambda_t}(t_1) &= \frac{f(\lambda_{t_1}|X_{t_0})}{f(\lambda_{t_1}|\lambda_{t_0})\Delta} \\
g_{X_s\to\lambda_t}(t_2) &= \frac{f(\lambda_{t_2}|X_{t_0}) - f(\lambda_{t_2}|\lambda_{t_1})g_{X_s\to\lambda_t}(t_1)\Delta}{f(\lambda_{t_2}|\lambda_{t_1})\Delta} \\
&\vdots \\
g_{X_s\to\lambda_t}(t_j) &= \frac{f(\lambda_{t_j}|X_{t_0}) - \sum_{k=0}^{j-1} f(\lambda_{t_j}|\lambda_{t_k})g_{X_s\to\lambda_t}(t_k)\Delta}{f(\lambda_{t_j}|\lambda_{t_{j-1}})\Delta} \\
&\vdots \\
g_{X_s\to\lambda_t}(t_N) &= \frac{f(\lambda_{t_N}|X_{t_0}) - \sum_{k=0}^{N-1} f(\lambda_{t_N}|\lambda_{t_k})g_{X_s\to\lambda_t}(t_k)\Delta}{f(\lambda_{t_N}|\lambda_{t_{N-1}})\Delta}
\end{aligned} \tag{3.2.20}$$

subject to the initial condition $g(t_0) = 0$. When evaluating this system of equations using a numerical approximation of the transitional density, most of the computation time will be dedicated to calculating the transitional density in the kernel of the summation. However, in the time-homogeneous case we can calculate the sequence of probabilities:

$$f(\lambda_{t_j}|\lambda_{t_i}) : i < j \text{ for } i, j = 1, 2, \dots N, \tag{3.2.21}$$

more efficiently by exploiting the fact that the transition density depends only on the size of the transition horizon and not on the time at which it started. That is, since

$$f(X_t|X_s) = f(X_v|X_u) \tag{3.2.22}$$

for any $t - s = v - u$, we can calculate the appropriate sequence of probabilities in Equation 3.2.21 by simply recycling values from the sequence:

$$f(\lambda_{t_i}|\lambda_{t_0}) : i = 1, 2, \dots N. \tag{3.2.23}$$

Thus, instead of re-evaluating the elements $f(\lambda_{t_j}|\lambda_{t_i})$ for $i = 0, 1, \dots j-1$ at each iteration of the updating equation, the elements can simply be 'read' from the sequence in Equation 3.2.23 by matching the length of the transition horizon for each element to one contained in the sequence (i.e., match $t_j - t_i$ in Equation 3.2.21 with $t_i - t_0$ in Equation 3.2.23). Using this strategy, we can reduce the total number of transition probabilities required to evaluate either Equation 3.2.6 or Equation 3.2.16 (since this redundancy is inherited through the revised probability current in Equation 3.2.16) from $(N^2 - N)/2 + 2N$ to $2N$, thus significantly improving the computational efficiency of the updating equations. In order to illustrate the computational gains offered by this redundancy, consider a

time-homogeneous quadratic diffusion with dynamics given by the SDE

$$dX_t = aX_t(b - X_t)dt + \sigma X_t dB_t \tag{3.2.24}$$

and a fixed threshold function $\lambda_t = \lambda$ for $t > 0$. Table 3.2.1 compares the computation time in seconds for the first passage time density with and without recycled transition probabilities over an increasing transition horizon for a fixed step-size. For purposes of the experiment we use the parameter set $\{a, b, \sigma\} = \{0.1, 10, \sqrt{0.05}\}$ for $X_0 = 8$, $\lambda = 12$, and a step-size of $\Delta = 0.01$. Since the transition density of Equation 3.2.24 is not available in closed form, we use the cumulant truncation procedure in order to approximate the transitional density. Using Equation 3.2.17 under a $d = 4$-th order truncation in conjunction with Equation 3.2.16 we subsequently approximate the first passage time density over the applicable transition horizons. The results suggest that significant computational gains can be made by recycling transition density calculations whilst evaluating the updating equations. Figure 3.2.1 compares the resulting distributions for the final transition horizon, $[0, 25]$. For reference, we compare the resulting approximation to a frequency distribution calculated by simulating the first passage time problem repeatedly (see Appendix C.1 for details on simulating first passage time problems). Using the simulated first passage time density as a guide, we can see that the scheme indeed produces a valid approximation at a fraction of the computational time with the simulated first passage time density taking around 210 seconds to calculate using 100 000 trajectories simulated using a modified Euler–Maruyama scheme with a constant step size of 0.0005 time units.

Computation time in secs.

Horizon (t)	Standard	Recycled
5	1.03	0.03
10	3.99	0.02
15	9.16	0.04
20	16.87	0.04
25	26.07	0.06

TABLE 3.2.1: Compuation times for increasing transition horizons under Equation 3.2.16 calculated using the cumulant truncation procedure in standard fashion and by recycling redundant transition density calculations. R code: Supplementary materials, Section 3.1.

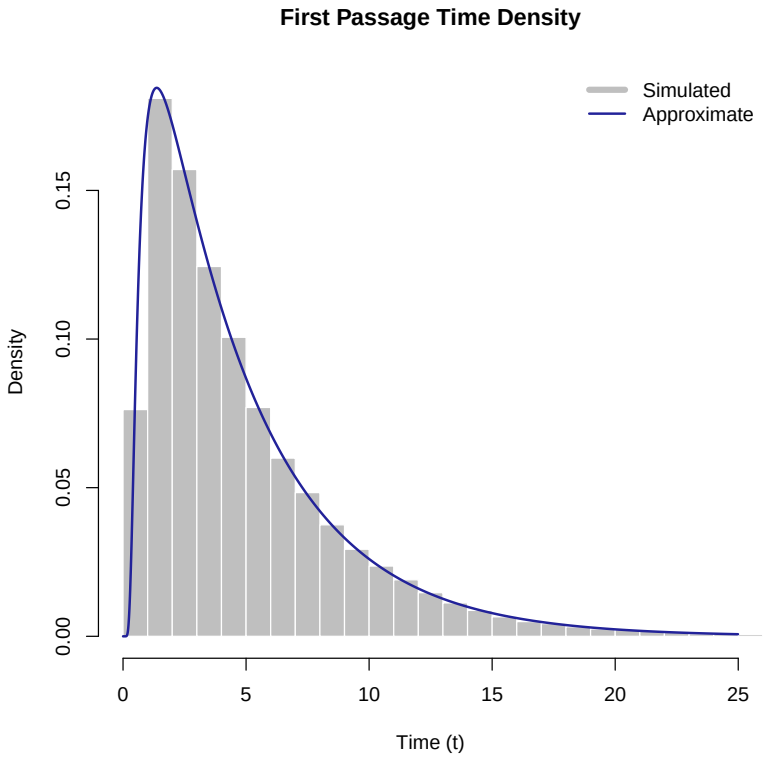


FIGURE 3.2.1: Simulated vs. approximate first passage time density of Equation 3.2.24 with $\{a, b, \sigma\} = \{0.1, 10, \sqrt{0.05}\}$ and initial value $X_0 = 8$ and $\lambda = 12$. R code: Supplementary materials, Section 3.1.

3.2.3 Software: DiffusionRgqd revisited

By combining the cumulant truncation procedure under the GQD framework with the revised Volterra equation, it is possible to calculate first passage time densities for non-linear, time-inhomogeneous diffusions to time-dependent thresholds. When the first passage time problem is time-homogeneous, we can further exploit computational redundancies in the updating equation in order to evaluate the first passage time density extremely efficiently. In order to achieve this, we exploit the architecture of the **DiffusionRgqd** package in order to build C++ routines for solving first passage time problems of varying degrees of complexity. As in the case of the `GQD.mcmc()` and `GQD.mle()` functions, this allows us to calculate first passage time densities with greater efficiency than would be achievable in R otherwise whilst still separating the user from the purely mathematical aspects of the algorithm. We demonstrate the application of the **DiffusionRgqd** package

at the hand of a number of time-inhomogeneous first passage time problems and proceed to show how the routines may be used to analyse a real-world dataset.

3.2.3.1 Time-inhomogeneous first passage time problems

Using the **DiffusionRgqd** package it is possible to calculate first passage time densities for time-inhomogeneous first passage problems via the `GQD.TIpassge()` function (for purposes of maximising computational efficiency we provide a separate routine, `GQD.passage()`, for time-homogeneous problems). The `GQD.TIpassage()` function uses the same interface as other functions in the package and operates in similar fashion to the `GQD.density()` function. However, since `GQD.TIpassage()` relies on calculating transitional density approximations for a large number of initial values in combination with the recursive updating algorithm of Equation 3.2.16, its internal workings have more in common with the `GQD.mle()` and `GQD.mcmc()` functions: Using various elements from the source code of these functions, `GQD.TIpassage()` constructs computationally optimised solutions in C++ which is subsequently executed in R. Thus, depending on the specification of the diffusion model as defined by the GQD-coefficients, the appropriate cumulant equations are derived and solved numerically after which elements such as Equation 3.2.17 and Equation 3.2.19 can be brought together in order to evaluate Equation 3.2.16.

As an introduction to the routine, we first compare the `GQD.TIpassage()` function to a routine from the **fptdApprox** (Román-Román *et al.*, 2014) package – an excellent R package for calculating numerical approximations to first passage time densities. Since **fptdApprox** can very effectively handle first passage time problems for diffusions with analytically tractable transitional densities, we use it to benchmark the `GQD.TIpassage()` function from the **DiffusionRgqd** package.

Consider a diffusion process with dynamics given by the SDE:

$$dX_t = 0.5(5 - X_t)dt + dB_t, \tag{3.2.25}$$

with $X_0 = 3$. For purposes of calculating a first passage time consider then also a time-dependent threshold function:

$$\lambda_t = 5 + 0.25\sin(2\pi t). \tag{3.2.26}$$

Under the **fptdApprox** package we may use the `Approx.fpt.density()` function in order to approximate the first passage time density. The interface requires that one define a diffusion process by configuring an object that consists of expressions giving the drift and diffusion coefficients as well as the exact transitional density

of the diffusion model. The resulting object is then used by the `Approx.fpt.density()` function to approximate the first passage time density. In R:

```
R> # Define the model object:
R> OU <- diffproc(c("alpha * x + beta", "sigma^2",
+   "dnorm((x-(y * exp(alpha * (t-s)) - beta * (1 - exp(alpha * (t-s)))
+   / alpha)) / (sigma * sqrt((exp(2 * alpha * (t-s)) - 1) / (2 * alpha))),
+   0, 1) / (sigma * sqrt((exp(2 * alpha * (t-s)) - 1) / (2 * alpha)))",
+   "pnorm(x,  y * exp(alpha * (t-s)) - beta * (1 - exp(alpha * (t-s))) /
+   alpha, sigma * sqrt((exp(2 * alpha * (t-s)) - 1) / (2 * alpha)))"))
R>
R> # Approximate the first passage time density:
R> res1 <- Approx.fpt.density(dp = OU,  t0 = 0, T = 10, id = 3,
+    S = "5+0.25 * sin(2 * pi * t)",
+    env = list(alpha = -0.5, beta = 0.5*5, sigma = 1)))
```

```
Computing...    Done.

The value of the cumulative integral of the approximation is
 0.992261204012805 < 1 - tol. If the value of the cumulative integral is
 not high and the final stopping instant is less than T, it may be
 appropriate:
   - Check if the value of the final stopping instant increases using k
     argument to summary the fptl class object, or
   - Approximate the density again with to.T = TRUE.
```

Here, the object `OU` defines the diffusion model through the `diffproc()` function by collating its drift and diffusion coefficients along with its transitional density and cumulative transitional density. Subsequently, `Approx.fpt.density()` evaluates the first passage time density for the process `dp = OU` from the initial value `di` to the threshold function `S` on the transition horizon `t0` to `T` for the parameter set listed in `env`. Note that the transitional density and corresponding cumulative transitional density are defined explicitly.

Using the **DiffusionRgqd** package, we begin by defining the model as per usual according to the GQD framework. That is, we defined the model in terms of the coefficients of the diffusion model under the GQD model and subsequently call the `GQD.TIpassage()` function in order to evaluate the first passage time density. Note that `GQD.TIpassage()` circumvents the need to specify the transitional density as it will use the model coefficients to recognise and construct the appropriate numerical approximation of the required transitional densities. In present form – for computational purposes – the `GQD.TIpassage()` function evaluates Equation 3.2.16 for constant thresholds only. Despite this, one can still evaluate first passage time problems for time-inhomogeneous threshold functions by making use of a simple transformation. For example, it can be shown that for

any threshold function that can be decomposed as

$$\lambda_t = \phi + h(t) \tag{3.2.27}$$

for continuous $h(t)$, the first passage time variable

$$T_{X_s \to \lambda_t} = \inf\{t > s : X_t \geq \lambda_t\}, \tag{3.2.28}$$

can be analysed equivalently as

$$T_{Y_s = X_s - h(s) \to \lambda_t - h(t)} = \inf\{t > s : Y_t = X_t - h(t) \geq \phi\}. \tag{3.2.29}$$

Thus, in order to apply the `GQD.TIpassage()` to a time dependent threshold problem, one need only specify the first passage time density under the equivalent static threshold transformation. That is, we first calculate the dynamics of Y_t under Itô's lemma and then apply the transformation to the initial value and threshold function and subsequently calculate the first passage time density under the transformed problem. For Equation 3.2.25, under the threshold function of Equation 3.2.26, we may equivalently analyse:

$$dY_t = 0.5(5 - \pi\cos(2\pi t) - 0.25\sin(2\pi t) - Y_t)dt + dB_t, \tag{3.2.30}$$

with $Y_0 = 3$ and $\lambda_t = 5$. Under the transformed problem, the first passage time density can be approximated using the **DiffusionRgqd** package by defining the transformed model and passing the revised peripheral parameters (initial value, threshold level, transition horizon, and step size to be used for Equation 3.2.16) to the `GQD.TIpassage()` function:

```
R> GQD.remove()
R> G0 <- function(t){0.5*5 - 0.5*pi*cos(2*pi*t) - 0.5*0.25*sin(2*pi*t)}
R> G1 <- function(t){-0.5}
R> Q0 <- function(t){1}
R> res2 <- GQD.TIpassage(Xs = 3, B = 5, s = 0, t = 10, delt = 0.005)
R>
R> plot(res1$y ~ res1$x, type = 'l', col = '#BBCCEE', lwd = 2)
R> lines(res2$density ~ res2$time, col = '#222299', lwd = 2, lty = 'dashed')
```

Plotting the respective calculations, we can compare the density approximations visually. Figure 3.2.2 illustrates the first passage time density approximations calculated using **DiffusionRgqd** and **fptdApprox** respectively. Indeed, the approximations are nearly identical and differences between the approximations are minute.

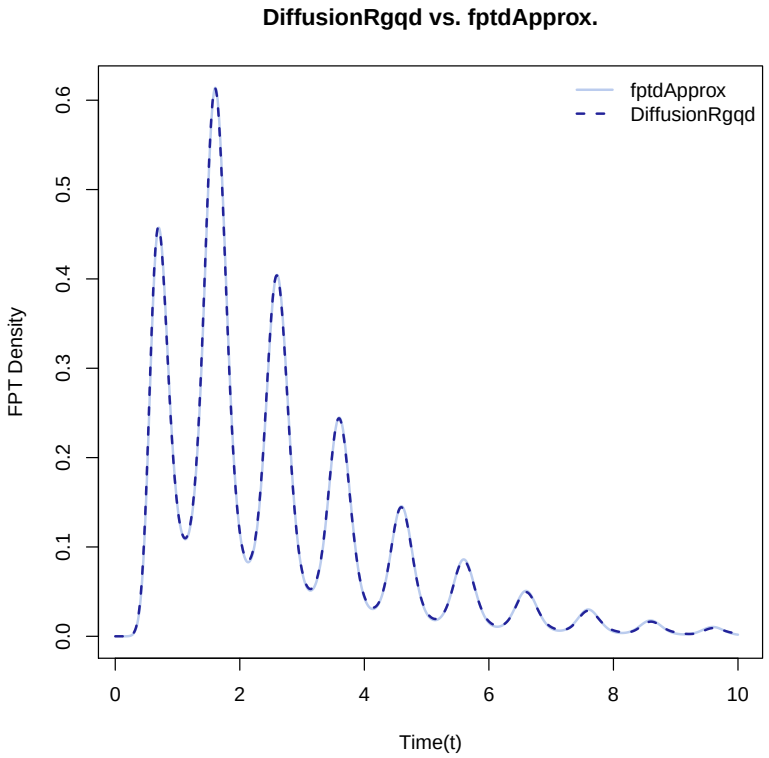


FIGURE 3.2.2: Approximate first passage times for Equation 3.2.25 passing through a sinusoidal threshold calculated using the **fptdApprox** package (light blue) and **DiffusionRgqd** package (dark blue, dashed). The approximate solutions produce nearly identical results. R code: Supplementary materials, Section 3.2.

Before continuing, it is worth noting that for this example it is possible to calculate the first passage time density more efficiently than is achieved by the `GQD.TIpassage()` here. Indeed, for this example the **fptdApprox** is somewhat more efficient in evaluating the first passage time density. However, the aim of the **DiffusionRgqd** package is tackle problems with intractable dynamics under the GQD framework where transition densities are rarely analytically available. Consequently, in order to maximise the range of first passage time problems which can be solved using the **DiffusionRgqd** package, we sacrifice some computational efficiency and adopt the premise that a numerical solution is always to be constructed under the GQD-framework.

Consider a diffusion with dynamics given by the SDE:

$$dX_t = \theta_1 X_t(10+0.2\sin(2\pi t)+0.3\sqrt{t}(1+\cos(3\pi t))-X_t)dt+\sqrt{0.1}X_t dB_t, \quad (3.2.31)$$

with $X_1 = 8$, θ_1 a fixed parameter, and a constant threshold $\lambda_t = 12$. For Equation 3.2.31, no analytical solution for the transition density exists. However, using the **DiffusionRgqd** package, we can evaluate the first passage time density by simply defining the model and making the appropriate call to `GQD.TIpassage()`. In R:

```
R> GQD.remove()
```

```
[1] "Removed :  G0 G1 Q0"
```

```
R> G1 <- function(t)
+ {
+   theta[1] * (10 + 0.2 * sin(2 * pi * t) + 0.3 * prod(sqrt(t),
+   1+cos(3 * pi * t)))
+ }
R> G2 <- function(t){-theta[1]}
R> Q2 <- function(t){0.1}
R> res3 = GQD.TIpassage(8, 12, 1, 4, 0.01, theta = c(0.5))
```

Note that we have parametrised the coefficients using the reserved variable `theta` in similar fashion to what would be done using `GQD.mcmc()`. This allows one to calculate the first passage time density for various parameter values without having to re-compile C++ code repeatedly (see Appendix C.2 for the corresponding C++ code). For example, we can evaluate the first passage time problem for Equation 3.2.31 for θ_1 running from 0.1 to 0.5 in discrete steps using a simple looping structure. This allows us to measure the effect that changing the parameter has on the first passage time distribution. For reference, we compare the resulting approximation to a simulated first passage time density for Equation 3.2.31 passing through $\lambda_t = 12$ for $\theta_1 = 0.5$. For brevity, the simulated first passage times can be accessed through the package datasets using the command `data("SDEsim6")`. The simulation consists of 500 000 first passage times simulated using a Euler–Maruyama scheme with a step-size of 0.0005 time units.

```
R> # Load simulated first passage times: `fpt.sim.times'
R> data("SDEsim6")
R> hist(fpt.sim.times, freq = F, col = 'gray85', border = 'white',
+    main = 'First Passage Time Density', ylab = 'Density', xlab = 'Time',
+    ylim = range(res3$density), xlim = range(res3$time), breaks = 100)
R>
R> library("colorspace")
R> colpal = function(n){rev(sequential_hcl(n, power = 1, l = c(20, 60)))}
```

```
R> th.seq = seq(0.1, 0.5, 0.05)
R> for(i in 2:length(th.seq))
+  {
+    res3 = GQD.TIpassage(8, 12, 1, 4, 0.01, theta = c(th.seq[i]))
+    lines(res3$density ~ res3$time, type = 'l', col = colpal(10)[i],
+    lty = 11 - i, lwd =1.5)
+  }
R> lines(res3$density ~ res3$time, type = 'l', col = colpal(10)[i], lwd = 2)
R> legend('topright', legend = th.seq, col = colpal(10), lty = 9:1,
R>   lwd = c(rep(1.5, 8), 2), title = expression(theta[1]), bty = 'n')
```

Figure 3.2.3 illustrates the effect of varying θ_1: As the value of the parameter decreases, the time taken to reach and exceed the threshold increases and the effect of the time dependent terms become less prominent. This makes sense since θ_1 in some sense dictates the 'speed' at which the process drifts toward the equilibrium line $10 + 0.2\sin(2\pi t) + 0.3\sqrt{t}(1 + \cos(3\pi t))$. Since this line starts out above the starting point $X_1 = 8$, θ_1 will dictate how intensely the drift of the process pulls it in the direction of the threshold. Finally, comparing the approximate first passage time density to that of the simulated first passage time, it can be seen that the approximation is indeed valid.

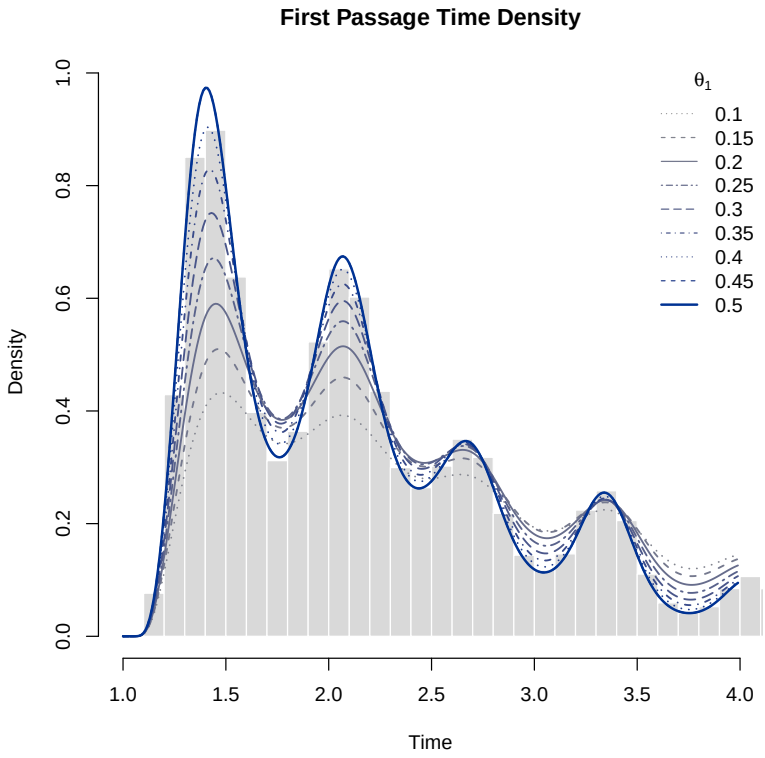


FIGURE 3.2.3: First passage time density of Equation 3.2.31 for $X_1 = 8$ through $\lambda_t = 12$ for various values of θ_1. For reference, we superimpose the approximate densities on a frequency distribution of 500 000 simulated first passage times at $\theta_1 = 0.5$ (indicated in gray). R code: Supplementary materials, Section 3.2.

3.2.3.2 Application to Amazon equity volatility

Although the methodology developed here makes it easy to analyse scalar first passage time problems of varying degrees of complexity as stand-alone problems, being able to accurately and efficiently approximate the first passage time density also affords the opportunity to enrich the analysis of diffusion models of time series. Indeed, since the interface of the `GQD.TIpassage()` routine operates in the same way as the `GQD.density()` and `GQD.mcmc()` functions discussed earlier, it can easily be applied in conjunction with diffusion models in the context of parametric inference.

As a brief example, we consider a time series of equity volatility for Amazon.com, one of the world's largest internet retailers. Using the **Quandl** library, we source data for Amazon equity volatility (VXAZN) for the period mid-2010 to the end

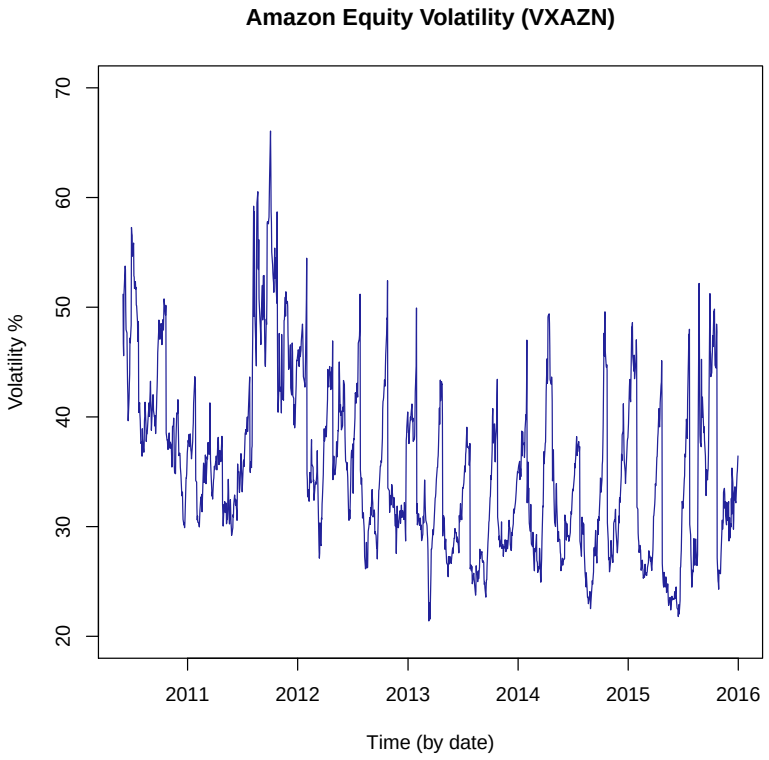


FIGURE 3.2.4: Daily equity volatility of Amazon shares for the period 2010-06-01 to 2016-01-01. R code: Supplementary materials, Section 3.3.

of 2015, sampled at daily intervals. For purposes of the analysis that follows we measure time in years and use exact dates for observations in order to construct transition horizons for consecutive observations. Throughout we will use the time of the initial observation as time zero. Figure 3.2.4 illustrates the trajectory of the time series.

We start out by modelling the volatility series using a simple diffusion model in the shape of a standard CIR process:

$$dX_t = \theta_1(\theta_2 - X_t)dt + \theta_3\sqrt{X_t}dB_t, \tag{3.2.32}$$

where θ_2 measures the long run mean of volatility, θ_1 controls the rate at which reversion to the level θ_2 occurs, and θ_3 denotes the volatility parameter of the process. In R, we can fit Equation 3.2.32 using the `GQD.mcmc()` routine:

```
R> quandldata <- Quandl("CBOE/VXAZN", collapse = "daily",
R>   start_date = "2010-06-01", end_date = "2016-01-01",  type = "raw")
```

```
R> Vt    <- rev(quandldata[, names(quandldata) == 'Close'])
R> time1 <- rev(quandldata[, names(quandldata) == 'Date'])
R> X     <- Vt / 100
R> time  <- cumsum(c(0, diff(as.Date(time1)) * (1 / 365)))
R>
R> # Define the standard CIR model:
R> GQD.remove()
R> G0 <- function(t){theta[1] * theta[2]}
R> G1 <- function(t){-theta[1]}
R> Q1 <- function(t){theta[3] * theta[3]}
R>
R> # Estimate parameters using the RWMH algorithm:
R> burns   <- 10000
R> updates <- 110000
R> theta   <- c(1, 0.5, 1)
R> sds     <- c(2.20, 0.02, 0.02)
R> mod1    <- GQD.mcmc(X, time, 10, theta, sds, updates, burns)
R> est1    <- GQD.estimates(mod1, thin = 100, corrmat = TRUE)
R> est1
```

```
$estimates
         Estimate Lower_CI Upper_CI
theta[1]   15.356   11.244   19.457
theta[2]    0.354    0.334    0.375
theta[3]    0.711    0.689    0.735

$corrmat
         theta[1] theta[2] theta[3]
theta[1]     1.00    -0.19     0.27
theta[2]    -0.19     1.00     0.03
theta[3]     0.27     0.03     1.00
```

Here, we estimate the parameters of Equation 3.2.32 using 100 000 iterations (not including burn-in) of the RWMH-algorithm with a burn-in period of 10 000 iterations. By storing the output of `GQD.mcmc()` as an object and passing it to the `GQD.estimates()` routine, we can calculate the desired parameter estimates. Using the parameter estimates, we can analyse the dynamics of the volatility series under Equation 3.2.32. In this context, since the series in question concerns the volatility of equity, it would be useful to analyse events of 'extreme' volatility. Using the final time series observation as an initial value, we can for example calculate the distribution of the time until volatility exceeds a given threshold level. That is, set $X_s = X_{5.586} = 0.3644$ and let λ denote the threshold volatility level, then we wish to calculate the distribution of the random variable $T_{X_{5.586} \to \lambda} = \inf\{t > s : X_t \geq \lambda\}$. For purposes of the analysis we set $\lambda = 0.50$, corresponding to a 50% threshold volatility. Under Equation 3.2.32, this can be achieved by setting up a first passage time problem and passing the relevant parameters of the problem to `GQD.TIpassage()`:

```
R> # Define the first passage time problem:
R> Xs     <- X[length(X)]    # Final observation (initial value for FPT-problem)
R> s      <- max(time)       # Final time
R> t      <- max(time) + 1.5 # Final time + 1.5 years
R> delt   <- 0.0025          # Step size
R> lambda <- 0.5             # Threshold volatility
R>
R> # Approximate the first passage time density @ par. estimate:
R> fpt1 <- GQD.TIpassage(Xs, lambda, s, t, delt, theta = est1$estimate[,1])
```

Here, we approximate the first passage time density under Equation 3.2.32 using the parameter estimates calculated earlier. As before, `GQD.TIpassage()` constructs the appropriate density approximations required to evaluate Equation 3.2.16 based on the model coefficients that are present in the current workspace and executes the updating algorithm on the designated transition horizon. Using this, we approximate the first passage time density on the transition horizon spanning 1.5 years following the final time-series observation using a step size of 0.0025 time units.

Using a similar arrangement, we can repeat the analysis for various diffusion models of the volatility series and compare the results. For example, a closer look at Figure 3.2.4 reveals that the volatility series appears to be somewhat cyclical. Specifically, the series exhibits increases and decreases that follow a quarterly cycle with respect to the annual time units. Without going into the details of the likely origin of this cyclical behaviour – we will return to this later in the thesis – we can test for seasonality empirically by fitting a time-inhomogeneous diffusion model to the volatility series. Consider a time-inhomogeneous counterpart to Equation 3.2.32:

$$dX_t = \theta_1(\theta_2 + \theta_3 \sin(8\pi t - (\theta_4 - 0.5)2\pi)) - X_t)dt + \theta_5\sqrt{X_t}dB_t. \qquad (3.2.33)$$

Equation 3.2.33 generalises Equation 3.2.32 by allowing the drift of the process to vary over time according to a quarterly cycle. In R we can easily fit Equation 3.2.33 using `GQD.mcmc()` and compare model fit via DIC values as before. However, since Equation 3.2.33 is parametrised in such a way that the drift function is periodic with respect to θ_4, we impose a Beta$(0.5, 0.5)$ prior on the parameter in order to avoid multiple maxima in the likelihood function. Thus:

```
R> GQD.remove()
R> G0 <- function(t)
+  {
+   theta[1]*(theta[2] + theta[3]*sin(8*pi*t + (theta[4] - 0.5)*2*pi))
+  }
R> G1 <- function(t){-theta[1]}
R> Q1 <- function(t){theta[5]*theta[5]}
```

```
R> priors <- function(theta){dbeta(theta[4], 0.5, 0.5)}
R>
R> theta <- c(1, 0.5, 1, 1, 0.5)
R> sds   <- c(2.20, 0.02, 0.02, 0.1, 0.01)
R> mod2  <- GQD.mcmc(X, time, 10, theta, sds, updates, burns)
R> est2  <- GQD.estimates(mod2 ,thin = 100, corrmat = TRUE)
R> est2
```

```
$estimates
         Estimate Lower_CI Upper_CI
theta[1]   18.836   14.801   23.264
theta[2]    0.353    0.338    0.369
theta[3]    0.074    0.052    0.100
theta[4]    0.425    0.380    0.474
theta[5]    0.708    0.685    0.733

$corrmat
         theta[1] theta[2] theta[3] theta[4] theta[5]
theta[1]     1.00    -0.11    -0.39    -0.34     0.22
theta[2]    -0.11     1.00     0.07    -0.05    -0.01
theta[3]    -0.39     0.07     1.00     0.14    -0.08
theta[4]    -0.34    -0.05     0.14     1.00    -0.11
theta[5]     0.22    -0.01    -0.08    -0.11     1.00
```

```
R> # Compare DIC values for each model:
R> GQD.dic(list(mod1,mod2))
```

```
        Elapsed_Time Time_Homogeneous         p           DIC        pD    N
Model 1     00:06:52               Yes     3.000      -6509.930     3.000 1409
Model 2     00:16:32                No     5.000  [=] -6535.920     5.080 1409
```

As expected, Equation 3.2.33 offers some improvement in fit over Equation 3.2.32.

Despite the fact that the models differ only in respect to a single sinusoidal term in the drift, this has rather striking implications for the behaviour of first passage time density. In R, we calculate the corresponding first passage time density to the 50% threshold, and compare the results:

```
R> # Approximate the first passage time density @ par. estimate:
R> fpt2  <- GQD.TIpassage(Xs, lambda, s, t, delt, theta = est2$estimate[,1])
```

Figure 3.2.5 compares the first passage time density for the volatility series from the final time series observation to a fixed threshold of 50% under the time-homogeneous and time-inhomogeneous CIR models respectively. Note that the first passage time density for the time-inhomogeneous model predicts that the likelihood of first exceeding volatility of 50% varies significantly over time. This appears to be much more consistent with the observed dynamics of the time series where extreme observations appear to occur periodically.

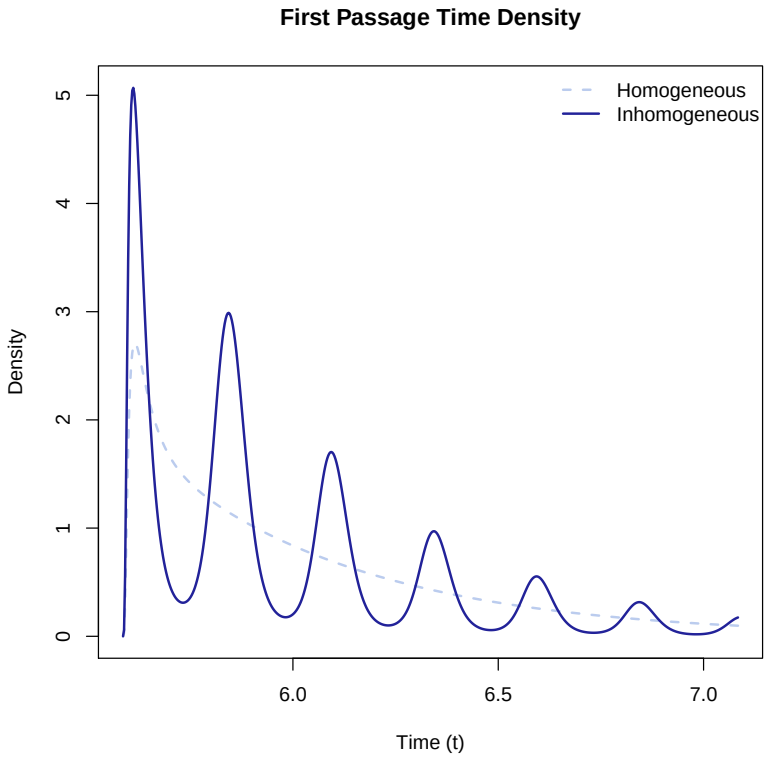


FIGURE 3.2.5: Approximate first passage time densities for equations 3.2.32 (light blue, dashed) and 3.2.33 (dark blue, solid) respectively, calculated using the final time series value $X_{5.586} = 0.3644$ to the threshold $\lambda = 0.5$ for 1.5 years following the final observation. In each case we use the parameter estimates calculated using the RWMH-algorithm. R code: Supplementary materials, Section 3.3.

Although the approximate first passage time densities calculated under the respective parameter vector estimates give a good indication as to the likelihood of exceeding the threshold volatility in the near future, the analysis can be further improved by accounting for uncertainty in the parameter estimates. Using the output from the RWMH-algorithm, we can repeatedly draw samples from the posterior distribution of the parameter vector and calculate the corresponding first passage time density at each iteration. Subsequently, we can compare the first passage time density calculated at the overall estimate to those calculated from the parameter samples in order to get a measure of the variation associated with the calculation.

Figure 3.2.6 compares the first passage time density calculated for 1000 samples

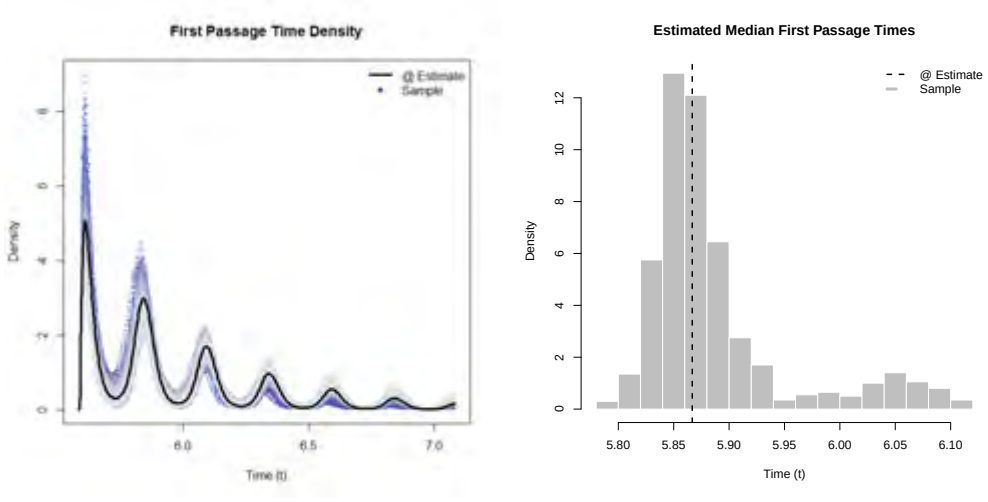


FIGURE 3.2.6: Left: Approximate first passage time densities calculated for 1000 samples of the MCMC output for Equation 3.2.33. Each density is colour-coded according to the rank of the median first passage time implied by the density (dark blue - light grey, ranked from shortest to longest). Right: Frequency distribution of the calculated median first passage times. The median first passage time calculated at the overall parameter vector estimate is indicated by the dashed black line. R code: Supplementary materials, Section 3.3.

drawn from the posterior distribution of the parameter vector of Equation 3.2.33. In addition, each sample density is coloured according to the rank of the median first passage time density as calculated from the resulting density (the approximate median first passage time density is returned as a list element by `GQD.TIpassage()`) and a frequency distribution for the median first passage times is drawn.

Comparing the first passage time density calculated at the mean parameter estimate (i.e., the parameter estimate calculated from the MCMC output) the parameter the to the first passage time densities calculated using individual samples from the posterior distribution, we note a reasonable amount of variation in the size of the peaks of the first passage time density. When the density is more peaked overall, the median first passage time is estimated to be shorter and *vice versa*. Interestingly, the estimated frequency distribution of the median first passage time reflects the multimodal nature of the first passage time density. Based on the frequency distribution of the estimated median first passage time density, we estimate the median first passage time to be around $t = 5.89$ (90% CI of $(5.83, 6.06)$), or approximately $t - s = 0.3$ years (90% CI of $(0.24, 0.47)$), from the final observation time.

3.3 Approximating the first passage time density by numerical solution of a PDE

Although combining numerical solutions of the transitional density with the revised Volterra equation makes it possible to calculate first passage time densities for non-linear diffusions very efficiently, an alternative (albeit significantly less efficient) technique exists for solving time-homogeneous first passage time problems. Let $\mathbf{X}_t$ denote a k-dimensional time-homogeneous diffusion process with dynamics given by the SDE:

$$d\mathbf{X}_t = \boldsymbol{\mu}(\mathbf{X}_t)dt + \boldsymbol{\sigma}(\mathbf{X}_t)d\mathbf{B}_t, \tag{3.3.1}$$

with $\boldsymbol{\mu}(\mathbf{X}_t) = (\mu_i(\mathbf{X}_t))_{k\times 1}$, $\boldsymbol{\sigma}(\mathbf{X}_t) = (\sigma_{ij}(\mathbf{X}_t))_{k\times k}$, and let $\boldsymbol{\Omega}$ denote a finite region of the state space enclosed by a perimeter $\boldsymbol{\lambda}$. Then, let

$$T_{\mathbf{X}_s\to\boldsymbol{\lambda}} = \inf\left\{t > s : \mathbf{X}_t \text{ reaches } \boldsymbol{\lambda}\right\} \tag{3.3.2}$$

denote the first passage time until $\mathbf{X}_t$ reaches the perimeter $\boldsymbol{\lambda}$ of the region $\boldsymbol{\Omega}$, and denote the cumulative first passage time density of $T_{\mathbf{X}_s\to\boldsymbol{\lambda}}$ by:

$$G_{\mathbf{X}_s\to\boldsymbol{\lambda}}(t) = P(T_{\mathbf{X}_s\to\boldsymbol{\lambda}} \leq t). \tag{3.3.3}$$

Based on elements of the work of Tuckwell and Wan (1984), it can be shown that the evolution of the cumulative first passage time density of a process $\mathbf{X}_t$, starting in $\mathbf{X}_s$ exiting the region $\boldsymbol{\Omega}$ is governed by the partial differential equation:

$$\frac{\partial G_{\mathbf{X}_s\to\boldsymbol{\lambda}}(t)}{\partial t} = \sum_{i=1}^{k}\mu_i(\mathbf{X}_s)\frac{\partial G_{\mathbf{X}_s\to\boldsymbol{\lambda}}(t)}{\partial X_s^{(i)}} + \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\gamma_{ij}(\mathbf{X}_s)\frac{\partial^2 G_{\mathbf{X}_s\to\boldsymbol{\lambda}}(t)}{\partial X_s^{(i)}\partial X_s^{(j)}}, \tag{3.3.4}$$

where $(\gamma_{ij}(\mathbf{X}_t,t))_{k\times k} = \boldsymbol{\sigma}(\mathbf{X}_t)\boldsymbol{\sigma}(\mathbf{X}_t)'$, subject to the boundary conditions:

$$G_{\mathbf{X}_s\to\boldsymbol{\lambda}}(u) = \begin{cases} 1 & \text{for } \mathbf{X}_s \notin \boldsymbol{\Omega} \quad \forall \quad u \in [s,t], \\ 0 & \text{for } \mathbf{X}_s \in \boldsymbol{\Omega} \text{ and } u = s. \end{cases} \tag{3.3.5}$$

Note that since the process is presumed to start within $\boldsymbol{\Omega}$ and the problem is defined up to and including the perimeter, the first element of the boundary conditions can equivalently be written as $G_{\mathbf{X}_s\to\boldsymbol{\lambda}}(u) = 1$ for $\mathbf{X}_s$ on $\boldsymbol{\lambda}$ $\quad\forall\quad$ $u \in [s,t]$. Note also that we have defined Equation 3.3.4 running forwards in time. Equivalently, one may consider the survival probability

$$S_{\mathbf{X}_s\to\boldsymbol{\lambda}}(t) = P(T_{\mathbf{X}_s\to\boldsymbol{\lambda}} > t), \tag{3.3.6}$$

for which we can derive the corresponding boundary value problem by plugging $G_{\mathbf{X}_s \to \boldsymbol{\lambda}}(u) = 1 - S_{\mathbf{X}_s \to \boldsymbol{\lambda}}(t)$ into equations 3.3.4 and 3.3.5.

As usual, Equation 3.3.4 is analytically intractable in general and one has to resort to solving the PDE numerically. For these purposes we may again apply the method of lines to Equation 3.3.4 in order to approximate the first passage time c.d.f. or the survival probability function. Reiterating from Section 1.2.1, this is achieved by constructing a discrete lattice on the domain of the boundary value problem. Subsequently, we can apply a finite difference scheme to approximate the spatial derivatives in the PDE in order to derive a system of ODEs which approximates the PDE at points on the lattice. By solving the resulting system of ODEs, we can approximate the c.d.f. or survival probability surface at fixed points on the lattice using standard numerical techniques. For purposes of this exposition, we will focus on the survival probability function. Although the techniques can be applied directly to the calculation of the c.d.f. of the first passage time variable, the behaviour of the survival probability function is easier to interpret in this context and simplifies the visualisation of solutions to the PDEs somewhat.

As an introductory example, consider the scalar case ($k = 1$). Let $\mathcal{L} = \{x_i : i = 0, 1, \ldots, N\}$ denote a discrete lattice on the domain $[\lambda^-, \lambda^+]$. Here, Ω can be thought of as the domain enclosed by the limits λ^- and λ^+. The lattice $\mathcal{L}$ can thus be defined in terms of discrete points that span the domain from λ^- to λ^+, for example $\mathcal{L} = \{x_0 = \lambda^-, x_1 = x_0 + \Delta, \ldots, x_{N-1} = x_0 + (N-1)\Delta, x_N = \lambda^+\}$ for some $\Delta = (\lambda^+ - \lambda^-)/N$. By applying a finite difference approximation to each of the spatial derivatives in Equation 3.3.4, we arrive at the system of ordinary differential equations under the equispaced lattice:

$$S_i'(t) = \mu(x_i)\left[\frac{S_{i+1}(t) - S_{i-1}(t)}{2\Delta}\right] + \frac{\sigma^2(x_i)}{2}\left[\frac{S_{i+1}(t) - 2S_i(t) + S_{i-1}(t)}{\Delta^2}\right] \quad (3.3.7)$$

for $i = 0, 1, \ldots, N$. The system in Equation 3.3.7 thus approximates the survival probability function at each x_i to form the survival probability surface for the first passage time problem. By solving Equation 3.3.7 numerically, we can approximate the evolution of the survival probability surface over time. In order to do so, we mimic the boundary conditions for the survival probability surface (derived from Equation 3.3.5) and set:

$$S_i(u) = \begin{cases} 0 & \text{for } i \in \{0, N\} \quad \forall \quad u \in [s, t], \\ 1 & \text{for } i = 1, 2, \ldots N-1 \text{ and } u = s. \end{cases} \quad (3.3.8)$$

Note that the approximation applies to varying initial values for the given first passage time problem. Using the survival probability surface, the first passage

time distribution for a given initial value can be derived. For example, for some initial value X_s, we can find the corresponding element on the lattice, say $x_{j^\star}$, and extract the target first passage time density by approximating the time derivative on the survival probability surface at $x_{j^\star}$:

$$g_{X_s \to \{\lambda^-, \lambda^+\}}(t_i) \approx -\left[\frac{S_{j^\star}(t_{i+1}) - S_{j^\star}(t_{i-1})}{t_{i+1} - t_{i-1}}\right] \tag{3.3.9}$$

where t_i are time nodes at which Equation 3.3.7 is evaluated numerically. When the initial value of interest does not fall on the lattice, the lattice can either be modified or the survival function for that initial value can be interpolated from neighbouring points on the lattice, form which the first passage time density can be subsequently calculated as in Equation 3.3.9.

3.3.1 A scalar non-linear diffusion with upper and lower bounds

Consider a non-linear diffusion with dynamics given by the SDE:

$$dX_t = aX_t(b - X_t^2)dt + \sigma dB_t \tag{3.3.10}$$

with parameters $\{a, b, \sigma\} = \{0.5, 1, 0.5\}$, moving in relation to lower and upper boundaries $\lambda^- = -0.5$ and $\lambda^+ = 2$ respectively. Using Equation 3.3.7, we can derive a system of ODEs which approximate the survival probability surface:

$$S_i'(t) = ax_i(b - x_i^2)\left[\frac{S_{i+1}(t) - S_{i-1}(t)}{2\Delta}\right] - \frac{\sigma^2}{2}\left[\frac{S_{i+1}(t) - 2S_i(t) + S_{i-1}(t)}{\Delta^2}\right], \tag{3.3.11}$$

for $\mathcal{L} = \{x_0 = -0.5, \ldots, x_N = 2\}$ and some $\Delta = (2 - (-0.5))/N$. Figure 3.3.1 illustrates the approximate survival probability surface calculated by solving Equation 3.3.11 numerically for $N = 50$ using the initial conditions given in Equation 3.3.8. Furthermore, we calculate the approximate first passage time density at $X_0 = 0.5$ using Equation 3.3.9 and compare the resulting approximation to a simulated first passage time density. As expected, the shape of the survival probability surface reflects a decline in the probability of not having reached the boundaries over time. Depending on the initial value of the process the probability of survival up to a given time may vary drastically, with survival rates dropping quickly for initial positions close to the perimeter. In comparison to the simulated first passage time density, the approximation calculated by taking the time-derivative of the survival probability surface proves quite accurate.

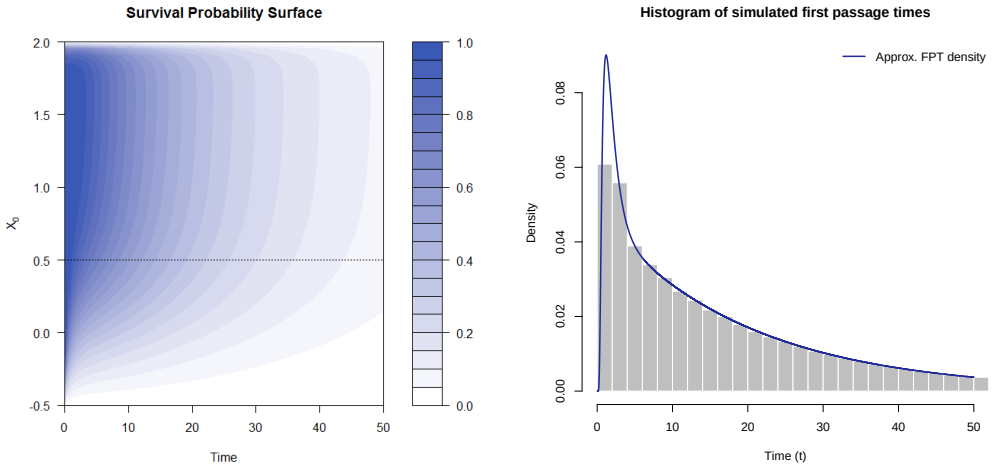


FIGURE 3.3.1: Left: Contour plot of the survival probability surface over time for Equation 3.3.10. The surface is calculated using the system of ODEs given by Equation 3.3.11. Right: The corresponding approximate first passage time density, calculated using Equation 3.3.9 at the desired initial value (dotted black line in left figure) superimposed on a simulated first passage time density calculated from 100 000 simulated first passage times. R code: Supplementary materials, Section 3.4.

Although this technique can handle quite general first passage time problems without having to deal with the technical pitfalls of the Volterra equation, it should be noted that this strategy can be significantly less efficient than calculating the first passage time density via a numerical solution of the second-kind Volterra equation. This follows since the quality of the approximation depends on both the mesh resolution and the step size used to solve the approximate survival surface. These, in turn, are heavily affected by the parameters of the problem. However, as will be shown in the sections that follow, this technique is much easier to modify for higher dimensional and non-standard first passage time problems.

3.3.2 Software: DiffusionRimp revisited

As in the case of the **DiffsuionRgqd** package, we are able to exploit some of the existing architecture of the **DiffusionRimp** package for calculating transition densities in order to write routines for calculating the survival probability surface of a first passage time problem. In Section 3.3 we showed how the method of lines may be applied to Equation 3.3.4 in order to approximate the first passage time density of a time-homogeneous non-linear diffusion exiting a region bounded by a

perimeter. As mentioned before, although the method is significantly less efficient than the Volterra equation, it is much easier to generalise to higher dimensional problems. As such, we can develop R-routines that can handle quite complicated first passage time problems. We demonstrate the workings of the routines at the hand of two highly non-linear first passage time problems and proceed to show how the methodology can be applied to problems with non-trivial perimeter functions.

Consider a bivariate non-linear diffusion with dynamics given by the SDE:

$$\begin{aligned} dX_t &= \big(a_x X_t(b_x - X_t^2) + Y_t\big)dt + \sigma_x dB_t^{(1)} \\ dY_t &= \big(a_y Y_t(b_y - Y_t^2) - X_t\big)dt + \sigma_y dB_t^{(2)} \end{aligned} \tag{3.3.12}$$

enclosed by the perimeter $\boldsymbol{\lambda}$ which circumscribes the quadrilateral region $[\lambda_x^-, \lambda_x^+] \times [\lambda_y^-, \lambda_y^+]$. In this case, the corresponding PDE for the survival probability surface is given by:

$$\begin{aligned} \frac{\partial S_{\mathbf{X}_s \to \boldsymbol{\lambda}}(t)}{\partial t} =& \big(a_x X_s(b_x - X_s^2) + Y_s\big)\frac{\partial S_{\mathbf{X}_s \to \boldsymbol{\lambda}}(t)}{\partial X_s} \\ &+ \big(a_y Y_s(b_y - Y_s^2) - X_s\big)\frac{\partial S_{\mathbf{X}_s \to \boldsymbol{\lambda}}(t)}{\partial Y_s} \\ &+ \frac{\sigma_x^2}{2}\frac{\partial^2 S_{\mathbf{X}_s \to \boldsymbol{\lambda}}(t)}{\partial X_s^2} + \frac{\sigma_x^2}{2}\frac{\partial^2 S_{\mathbf{X}_s \to \boldsymbol{\lambda}}(t)}{\partial Y_s^2}, \end{aligned} \tag{3.3.13}$$

subject to the boundary conditions:

$$S_{\mathbf{X}_s \to \boldsymbol{\lambda}}(u) = \begin{cases} 0 & \text{for } \mathbf{X}_s \notin (\lambda_x^-, \lambda_x^+) \times (\lambda_y^-, \lambda_y^+) \quad \forall \quad u \in [s, t], \\ 1 & \text{for } \mathbf{X}_s \in (\lambda_x^-, \lambda_x^+) \times (\lambda_y^-, \lambda_y^+) \text{ and } u = s. \end{cases} \tag{3.3.14}$$

The problem can thus be thought of as the first passage time until a particle whose trajectory over time is governed by Equation 3.3.12 escapes from a box in the two-dimensional real plane. In similar fashion to the bivariate transition density calculated in Section 1.2, we may analyse the evolution of the survival probability surface using the `BiMOL.passage()` function. By default, `BiMOL.passage()` assumes that the first passage time problems concerns a quadrilateral perimeter. By specifying the diffusion model in terms of its drift and diffusion coefficients the problem can be solved by passing the parameters that give perimeter positions on the x and y axes and some peripheral parameters for the method of lines. Let $\{a_x, b_x, \sigma_x, a_y, b_y, \sigma_y\} = \{0.5, 1, 1, 0.5, 1, 1\}$ and $\{\lambda_x^-, \lambda_x^+, \lambda_y^-, \lambda_y^+\} = \{-2, 2, -2, 2\}$ and choose a specific coordinate for which to calculate the first passage time density, say $(X_s, Y_s) = (0.5, 0.5)$. Then, in R:

```
R> # Define drift and diffusion terms:
```

```
R> mu1   <- function(X, Y){0.5*X*(1 - X^2) + Y}
R> mu2   <- function(X, Y){0.5*Y*(1 - Y^2) - X}
R> sig11 <- function(X, Y){1.0}
R> sig22 <- function(X, Y){1.0}
R>
R> # Peripheral parameters of the problem:
R> Xs   <- 0.5      # Starting X-coordinate
R> Ys   <- 0.5      # Starting Y-coordinate
R> t    <- 2        # Final horizon time
R> Xlim <- c(-2, 2) # Limits in X-dimension
R> Ylim <- c(-2, 2) # Limits in Y-dimension
R> N    <- 51       # How many nodes
R> delt <- 1/500    # Time step size
R>
R> res <- BiMOL.passage(Xs, Ys, t, limits = c(Xlim, Ylim), N, delt)
```

Figure 3.3.2 shows the resulting survival probability surface at various points along the transition horizon. Interestingly, due to the presence of the terms $+Y_t$ and $-X_t$ in the drift functions the trajectories of the process exhibit spin. This is reflected in the behaviour of the survival probability surface over time, resulting in a surface that is contorted in an anti-clockwise direction as time increases.

Although the default set-up is to construct a quadrilateral perimeter, it is possible to consider more complicated regions. For example, using the same parameter set as for Equation 3.3.12, we confine the process to the region circumscribed by a circle of radius 1.5 centred at the coordinate $(0.5, 0.5)$, thus creating an circular perimeter defined by the equation $(x - 0.5)^2 + (y - 0.5)^2 = 1.5^2$. In R, we can delineate the circular perimeter by passing an indicator function to `BiMOL.passage()` which determines whether a point in the xy-plane falls within the perimeter or not. For example, we may define:

```
R> # Define a region in the x-y plane:
R> region <- function(x,y){sqrt((x-0.5)^2+(y-0.5)^2)<1.5}
```

Here, the function `region(x,y)` returns `TRUE` if the coordinate (x, y) falls within the perimeter region and `FALSE` if not. By assigning `region(x,y)` to the `Phi` parameter of `BiMOL.passage()`, the first passage time problem can be modified to reflect the revised perimeter function. Note that here the lattice is still constructed on the quadrilateral region defined by the limits `c(Xlim, Ylim)`, so care needs to be taken in order to ensure that the lattice resolution remains sufficiently fine within the revised perimeter. In R:

```
R> # Define a region in the xy-plane and pass it to BiMOL.passage():
R> region <- function(x,y){sqrt((x - 0.5)^2 + (y - 0.5)^2) < 1.5}
R> res    <- BiMOL.passage(Xs, Ys, t, limits = c(Xlim, Ylim), N, delt, Phi =
    region)
```

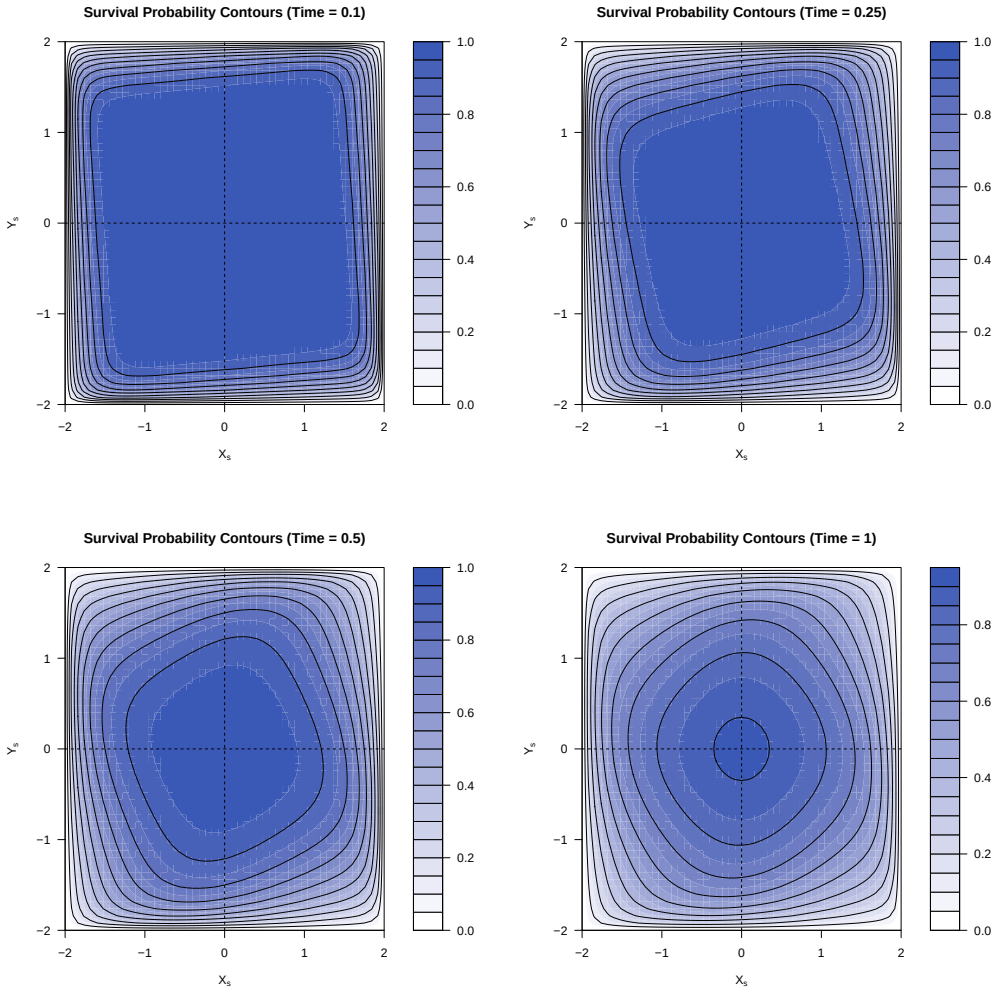


FIGURE 3.3.2: Contour plot of the survival probability surface of Equation 3.3.12 under the parameter set $\{a_x, b_x, \sigma_x, a_y, b_y, \sigma_y\} = \{0.5, 1, 1, 0.5, 1, 1\}$ for a quadrilateral perimeter $[-2, 2] \times [-2, 2]$ at times $t - s = \{0.1, 0.25, 0.5, 1\}$. R code: Supplementary materials, Section 3.5.

Figure 3.3.3 illustrates the evolution of the survival probability surface over time. Compared to Figure 3.3.2, the survival probability declines significantly quicker for the circular perimeter than for the quadrilateral perimeter. Indeed, the region enclosed under the circular perimeter regime is significantly smaller and thus it is to be expected that the first passage time to the perimeter should be shorter. Furthermore, the apparent contortion of the survival probability surface remains apparent, however, although the process is symmetric about the origin with respect to its probabilistic evolution and the perimeter itself is symmetric, the survival probability surface is not, reflecting the offset between the points of symmetry of the perimeter and the underlying process.

Based on the survival probability surfaces of each perimeter regime, we can extract approximate first passage time densities for the initial value $(X_s, Y_s) = (0.5, 0.5)$. By supplying `BiMOL.passage()` with the appropriate values, this is done automatically and the resulting calculation can be retrieved from the output of `BiMOL.passage()`. In order to verify the approximations, we compare the results to simulated first passage time densities under both perimeter regimes. Figure 3.3.4 compares the simulated and approximate first passage time densities. For reference, we draw a plot of the evolution of the 10%, 25%, and 50% contours of the survival probability surface around the initial value of the first passage time problem. Note that for the circular perimeter, despite the fact that the initial value falls on the furthest point from the perimeter, the median survival time is not maximised (the median survival time is maximised at the coordinate of the pinnacle of the 50% contour surface indicated in blue) for this initial condition. In both cases, the approximate and simulated first passage time densities match quite closely, illustrating that despite the complexity of the first passage time problems, an accurate approximation to the first passage time density can be calculated by numerical solution of the associated survival probability surface.

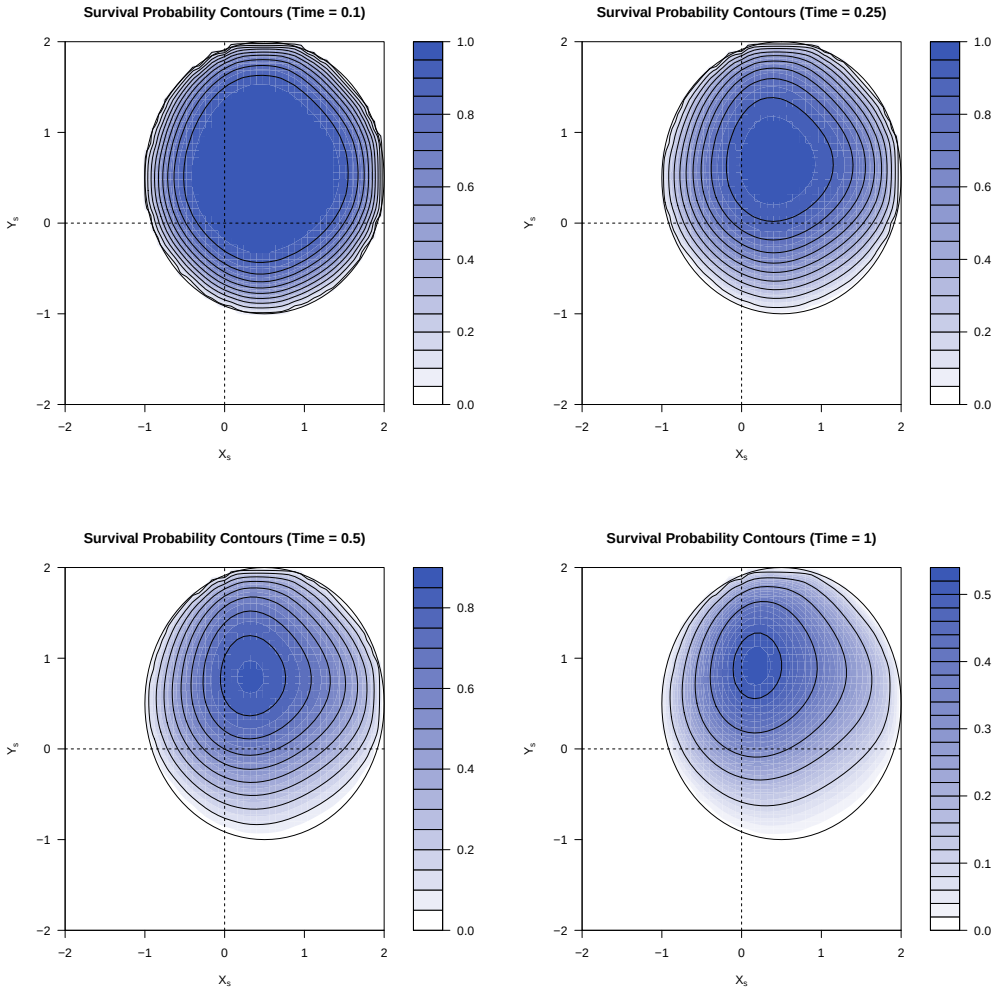


FIGURE 3.3.3: Contour plot of the survival probability surface of Equation 3.3.12 under the parameter set $\{a_x, b_x, \sigma_x, a_y, b_y, \sigma_y\} = \{0.5, 1, 1, 0.5, 1, 1\}$ for an circular perimeter $(x - 0.5)^2 + (y - 0.5)^2 = 1.5$ at times $t - s = \{0.1, 0.25, 0.5, 1\}$. R code: Supplementary materials, Section 3.6.

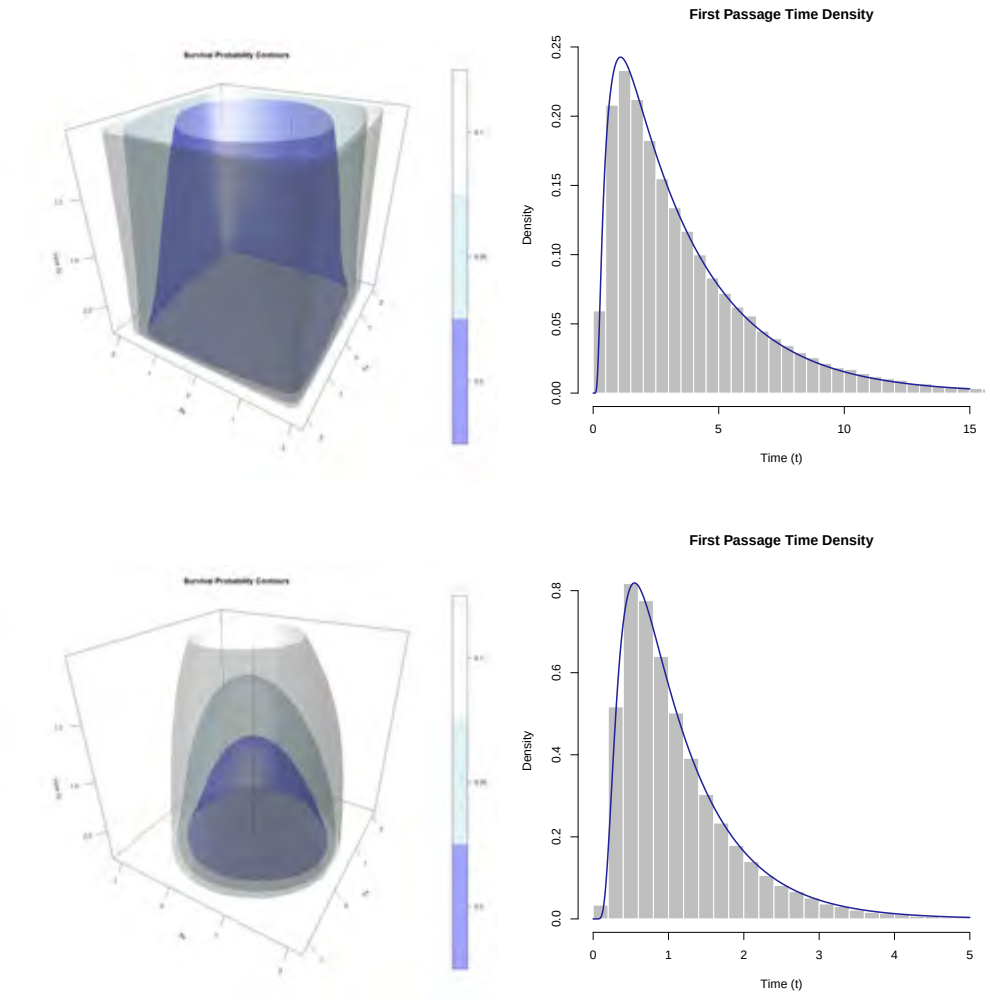


FIGURE 3.3.4: Evolution of the contours of the survival probability surface (left) for 10%, 25%, and 50% survival probabilities and simulated vs. approximate first passage time densities for the initial value $(X_s, Y_s) = (0.5, 0.5)$ (indicated by a vertical black line in each contour plot) under the quadrilateral (top) and circular (bottom) perimeter regimes. R code: Supplementary materials, Section 3.6.

3.4 Chapter summary

The analysis of first passage time problems for diffusion processes presents an important and challenging topic in the analysis of diffusion processes. Although first passage time problems in the context of diffusion models enjoy application in various fields of science, the analysis of first passage time problems are challenging – especially for problems where the underlying diffusion model is non-linear and/or time-inhomogeneous or where the geometry of the threshold function is not simple. For scalar diffusions moving in relation to a single threshold function, we can gain access to the first passage time density through the Volterra equation and various modifications thereof by way of the transitional density. Consequently, by solving the Volterra equation numerically, we can solve first passage time problems provided that the transitional density is analytically and/or numerically tractable. By combining techniques for analysing the transitional densities of non-linear, time-inhomogeneous diffusion models with these numerical solutions, we are able to analyse quite complicated time-inhomogeneous first passage time problems. In turn, by exploiting the existing architecture of the **DiffusionRgqd** package we develop routines for analysing first passage time densities of generalised quadratic diffusions. These routines offer significant improvements in efficiency over simulation and widen the scope of application to conducting inference and analysis on real-world datasets where repeated calculations of the first passage time density are required in order to conduct a rigorous analysis of the data.

Although the Volterra equation affords the opportunity to analyse suitably general scalar first passage time problems, the scope of the methodology as developed here is limited by our ability to accurately evaluate the transitional density as well as the stopping rule for which a suitable integral equation can be derived. In order to analyse first passage time problems with more general geometric properties, we develop a scheme for calculating the first passage time density by way of a partial differential equation. Although the methodology as developed here applies to time-homogeneous problems only, the scheme can be applied to highly non-linear problems and can easily be generalised to higher dimensions. Once again, by exploiting the existing architecture of the **DiffusionRimp** package, we develop routines for analysing such problems using the method of lines. Although the scheme incurs significantly more computational overhead than the approach via the Volterra equation, we can easily analyse first passage time problems for both scalar and bivariate diffusions moving in relation to perimeters with highly non-linear geometries.

Chapter 4

Non-Linear Multivariate Jump Diffusions with State-Dependent and/or Stochastic Intensity

4.1 Introduction

Real-world processes are often subject to numerous sources of random input, resulting in multifarious random behaviour that forms an integral part of the trajectory of the process. In modelling such phenomena, it is thus imperative that the model equations account for the various sources of randomness that govern the dynamics of the process. Although diffusion processes are used extensively in the modelling of continuous processes, it is usually assumed that Brownian motion suffices as driving mechanism for the stochastic evolution of the process. Where it does not suffice, it is possible to generalise the model process in order to make it more realistic for the application at hand. One such generalisation is to include randomly occurring 'jumps' in the trajectory of the model process. This modification has primarily been motivated in financial contexts where diffusion models are used to describe the dynamics of price/asset processes which are subject to seemingly spontaneous yet frequent jumps in observed time series. For example, it is often assumed that log-returns on a given stock price process are Normally distributed. This assumption can easily be accommodated in a stochastic differential equation by assuming that the dynamics of the stock price

process follows that of geometric Brownian motion:

$$dX_t = \mu X_t dt + \sigma X_t dB_t, \tag{4.1.1}$$

where X_t denotes the stock price at time t, from which it follows that $\log(X_t) - \log(X_s) \sim \mathrm{N}((\mu - \sigma^2/2)(t-s), \sigma^2(t-s))$ for $t > s$. However, the normality of stock-price returns have long been contested, and empirical evidence suggests that returns often exhibit features which are not well replicated by the Normal distribution. Perhaps the most well documented of these is the apparent lack of heavy tails in the model process. This is demonstrated by calculating descriptive statistics such as the skewness and kurtosis of the observed return series (Kou, 2007), which may subsequently be contrasted to the corresponding statistics under the Normal distribution. In the context of diffusion processes, this disparity is typically compensated for by formulating a stochastic volatility model wherein the diffusion coefficient of the returns process is itself treated as a stochastic process. That is, the revised process may, for example, assume the form:

$$\begin{aligned} d\log(X_t) &= \big(\mu - \frac{1}{2}\sigma_t^2\big)dt + \sigma_t dB_t^{(1)} \\ d\sigma_t^2 &= a(\sigma_t^2, t)dt + b(\sigma_t^2, t)dB_t^{(2)}, \end{aligned} \tag{4.1.2}$$

where $a(\sigma_t^2, t)$ and $b(\sigma_t^2, t)$ denote the drift and diffusion of the variance process respectively, and $B_t^{(1)}$ and $B_t^{(2)}$ are correlated Brownian motions. Among numerous other attractive properties, stochastic volatility models make it possible to more accurately capture the tail behaviour of the stock-price returns by explicitly allowing the variance of the log-returns to vary over time, resulting in a significantly improved approximation of the observed process. Indeed, Heston (1993) describe in great detail the effects of a stochastic volatility mechanism on the transitional density of log-returns under the Heston model. However, care needs to be taken when interpreting the stochastic volatility mechanism. For example, when $B_t^{(1)}$ and $B_t^{(2)}$ are uncorrelated the marginal distribution of the log-returns process conditional on a known initial value for the variance process (i.e., $X_t | X_s, \sigma_s^2$ for $t > s$) is very close to Normal, and even when strong correlation is present, in which case the marginal distribution of log-returns may be skew with slightly fatter tails than predicted under the Normal distribution, the resulting transitional density may not be sufficiently leptokurtic to account for extreme return events over short transition horizons. To illustrate the point, consider a rolling estimate of kurtosis for daily log-returns of the Standard and Poor's 500 index (S&P 500). Let X_{t_i} denote the value of the S&P 500 index at time t_i, then

define a backward-looking rolling estimate of kurtosis with bandwidth h by:

$$K(t_i, h) = \frac{1}{h \times \hat{s}^2(t_i, h)} \sum_{k=i-h+1}^{i} \Big(X_{t_k} - \hat{m}(t_i, h) \Big)^4 \tag{4.1.3}$$

for all $i \geq h$, where $\hat{s}(t_i, h) = \frac{1}{h} \sum_{k=i-h+1}^{i} \big(X_{t_k} - \hat{m}(t_i, h) \big)^2$ and $\hat{m}(t_i, h) = \frac{1}{h} \sum_{k=i-h+1}^{i} X_{t_k}$. Figure 4.1.1 depicts the rolling estimate of kurtosis along with a time-differenced estimate, calculated as $\{K(t_i, h) - K(t_{i-1}, h) : i = h, h+1, \ldots N\}$, using a bandwidth of $h = 250$ days for the time period 1990-01-01 to 2015-12-31. Under this bandwidth, the differenced series represents the change in estimated kurtosis caused by moving the rolling estimate one day forward for (approximately) the last year's worth of data. Additionally, we superimpose an overall estimate of kurtosis (calculated over the entire time period) along with that of the Normal distribution. Based on the overall estimate, the sample kurtosis clearly exceeds

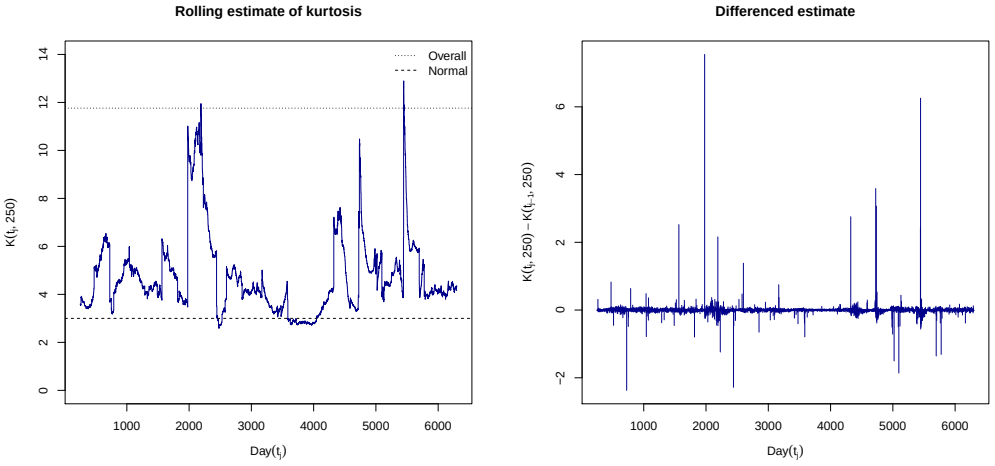


FIGURE 4.1.1: Rolling estimate of kurtosis calculated using a bandwidth of $h = 250$ days on daily log-returns of the S&P 500 index for the time period 1990-01-01 to 2015-12-31 (left) and the corresponding time-differenced rolling estimate (right). R code: Supplementary materials, Section 4.1.

that of the Normal distribution. However, the one year rolling estimate reveals that although the kurtosis of the log-returns series is typically higher than that of a Normal distribution, the size of the overall estimate can be attributed to the occurrence of a number of extreme return events. These events manifest as sudden spikes in the rolling estimate of kurtosis which can be clearly seen in the time-differenced estimate. In order to account for such extreme events Merton (1976) for example proposed the inclusion of jumps in the diffusion trajectory in

order to create a more accurate model of asset price returns than is predicted by the continuous paths of geometric Brownian motion, in which case the modified stochastic differential equation assumes the form

$$d\log(X_t) = (\mu - 0.5\sigma^2)dt + \sigma dB_t + \dot{z}_t dN_t, \tag{4.1.4}$$

where $\dot{z}_t$ denotes a normally distributed jump random variable and N_t is a Poisson process with constant intensity i.e., $N_t - N_s \sim \text{Poi}(\lambda(t-s))$. Under this formulation, extreme events are explicitly included in the stochastic differential equation as randomly occurring discontinuous jumps in the diffusion trajectory. Consequently, the disparity between observed tail behaviour of log-returns and that of Brownian motion is mitigated by the inclusion of a jump mechanism. Building on this, one may extend the model in order to formulate a stochastic volatility model with jumps, for example:

$$\begin{aligned} d\log(X_t) &= \big(\mu - \frac{1}{2}\sigma_t^2\big)dt + \sigma_t dB_t^{(1)} + \dot{z}_t^{(1)} dN_t \\ d\sigma_t^2 &= a(\sigma_t^2, t)dt + b(\sigma_t^2, t)dB_t^{(2)} + \dot{z}_t^{(2)} dN_t, \end{aligned} \tag{4.1.5}$$

where jumps affect both returns and volatility at the same time. Using this, the useful properties of the stochastic volatility specification are retained whilst directly accounting for extreme return events and jumps in volatility.

Although the addition of a jump mechanism serves to improve the flexibility of diffusion models and allows for the formulation of more realistic models of real-world processes, this flexibility comes at the cost of magnifying the already significant difficulties associated with the calculus of diffusion processes. Consequently, the space of analytically tractable jump diffusion models is even more sparse than that of the jump-free/pure diffusion processes. Furthermore, where analytical solutions to quantities such as the transitional density are available, they are often precluded by simplifying assumptions on the specification of both the diffusion part of the process as well as the jump mechanism of the model process. That said, a number of different jump mechanisms have been proposed in the literature: Ball and Torous (1985) propose log-normally distributed jumps under geometric Brownian motion as a model for stock price returns, whilst Ramezani and Zeng (1998) and Kou (2002) assume the same diffusion dynamics but modify the jump process to instead follow a double-exponential jump distribution. Although the choice of distribution is usually based on some *a priori* information of the process to be modelled, choosing a valid jump distribution can be a subtle process. For example, in the case of Ball and Torous (1985) it is actually meant that the log of the underlying process has normally distributed jumps (i.e., Brownian motion with drift and normally distributed jumps), implying that both the diffusion and jump dynamics

are based on the Normal distribution. In this case, Honore (1998) notes that depending on the quality of the data it can be difficult to distinguish which source of randomness (i.e., Brownian motion or jump process) is responsible for a random innovation (i.e., a stochastic change in the state of the process) in the underlying process, thus making it difficult to calculate reliable parameter estimates for such a model despite the relatively simple structure of the model.

Despite the limited set of analytically tractable jump diffusion models, numerous estimation techniques have been proposed for jump diffusion models with analytically intractable dynamics. Eraker (2001) apply Monte Carlo techniques, replacing missing sample paths with simulated trajectories (see also Eraker *et al.*, 2003; Eraker, 2004) in order to estimate the likelihood, thus circumventing the need for closed-form solutions to the likelihood function. Other notable approaches include the efficient method of moments (EMM) scheme of Gallant *et al.* (1997) which was later used by Craine *et al.* (2000) to perform inference on multivariate jump diffusions, and the empirical characteristic function estimation schemes of Jiang and Knight (2002) and Rockinger and Semenova (2005). Yu (2007) extended the popular Hermite series approximations for jump-free diffusions (Aït-Sahalia, 2002, 2008) in order to calculate closed-form likelihood approximations for multivariate jump diffusions whilst Zhang and Schmidt (2016) develop short horizon density approximations based on expansions of the characteristic function which can subsequently be used to approximate the likelihood function. Although most of the literature on the estimation of jump diffusion models are concerned with parametric inference, non-parametric techniques have also been developed by Johannes (1999); Bandi and Nguyen (2003) and Aït-Sahalia *et al.* (2012). In addition to traditional parametric and non-parametric methods, technical aspects regarding the nature of jump mechanisms in the context of inference have also been explored. For example, Aït-Sahalia (2004) shows how to separate the diffusion and jump dynamics for a given jump diffusion model, whilst Aït-Sahalia *et al.* (2009) and Aït-Sahalia and Jacod (2011) develop tests for the presence and frequency of jumps in partially observed processes respectively.

In order to choose an appropriate jump diffusion model of an observed process, we are required to validate the attributes of a given model quantitatively by contrasting it with a number of competing models. That is, by combining various forms of diffusion processes and jump mechanisms one may formulate an empirical basis for choosing a given model, as opposed to formulating a model on theoretical grounds alone. In this chapter, we endeavour to develop a method for approximating the transitional densities of non-linear, time-inhomogeneous jump diffusions with state-dependent, stochastic jump intensity. We subsequently apply the methodology to a host of test examples, exploring both the performance and limitations of the methodology. Following this, we compare the methodology to

existing methods in the literature, after which we apply the scheme by conducting inference on jump diffusion models of a real-world dataset. Finally, we outline the development of a software package based on the methodology and detail the workings and application of the software with theoretical and practical experiments.

4.2 Multivariate jump diffusion processes

Let $\mathbf{P}_t$ denote a multivariate, k-dimensional pure jump process with dynamics given in differential form by:

$$d\mathbf{P}_t = \mathbf{J}(\mathbf{P}_t, \dot{\mathbf{z}}_t, t)d\mathbf{N}_t, \tag{4.2.1}$$

where $\mathbf{J}(\mathbf{P}_t, \dot{\mathbf{z}}_t, t) = (\epsilon_{ij}(\mathbf{P}_t, \dot{\mathbf{z}}_t, t))_{k\times q}$ denotes the jump matrix, $\dot{\mathbf{z}}_t = (\dot{z}_t^{(ij)})_{k\times q}$ is a $k \times q$ matrix of random variables with independent columns, $\mathbf{N}_t = (N_t^{(j)})_{q\times 1}$ is a q-dimensional counting process with intensity vector $\boldsymbol{\lambda}(\mathbf{P}_t, \dot{\mathbf{r}}_t, t) = (\lambda_j(\mathbf{P}_t, \dot{\mathbf{r}}_t, t))_{q\times 1}$, and $\dot{\mathbf{r}}_t$ is a q-dimensional stochastic process on which the intensity vector may depend. Under this formulation, the jump matrix $\mathbf{J}(\mathbf{P}_t, \dot{\mathbf{z}}_t, t)$ relates discrete increments in the process $\mathbf{N}_t$ into non-discrete changes in state of the process $\mathbf{P}_t$. This is achieved by mapping the discrete increments to real-valued increments through realisations of the jump variables, $\dot{\mathbf{z}}_t$. In turn the mapping is governed by the jump matrix $\mathbf{J}(\mathbf{P}_t, \dot{\mathbf{z}}_t, t)$, which determines how the jump variables enter the process and how the mapping depends on the current state of the process. For example, if the first element of the process $\mathbf{N}_t$ increments at time τ, then every element of $\mathbf{P}_\tau$ changes in accordance to the outcome of the first column of the jump-matrix, which in turn is determined by the functional relation between the first column of jump variables in $\mathbf{z}_\tau$ and $\mathbf{P}_\tau$ through $\mathbf{J}(\mathbf{P}_\tau, \dot{\mathbf{z}}_\tau, \tau)$. To see this, it is useful to write the process $\mathbf{P}_t = \{P_t^{(1)}, P_t^{(2)}, \ldots, P_t^{(k)}\}'$ in terms of its individual components as:

$$P_t^{(i)} = \sum_{j=1}^{q} \epsilon_{ij}(\mathbf{P}_t, \dot{\mathbf{z}}_t^{(\cdot j)}, t) \quad \text{for} \quad i = 1, 2, \ldots, k, \tag{4.2.2}$$

where $\dot{\mathbf{z}}_t^{(\cdot j)} = \{\dot{z}_t^{(1j)}, \dot{z}_t^{(2j)}, \ldots, \dot{z}_t^{(kj)}\}'$ denotes the j-th column of the jump variable matrix with distribution function ϕ_j. $P_t^{(i)}$ thus consists of the sum of all jump realisations that have occurred up to and including time t for the i-th dimension, across all q counting processes $N_t^{(j)}$. Figure 4.2.1 illustrates how the jump process is constructed from a simulated trajectory for $k = 1$ and $q = 2$ i.e., a one-dimensional jump process impacted by two counting processes.

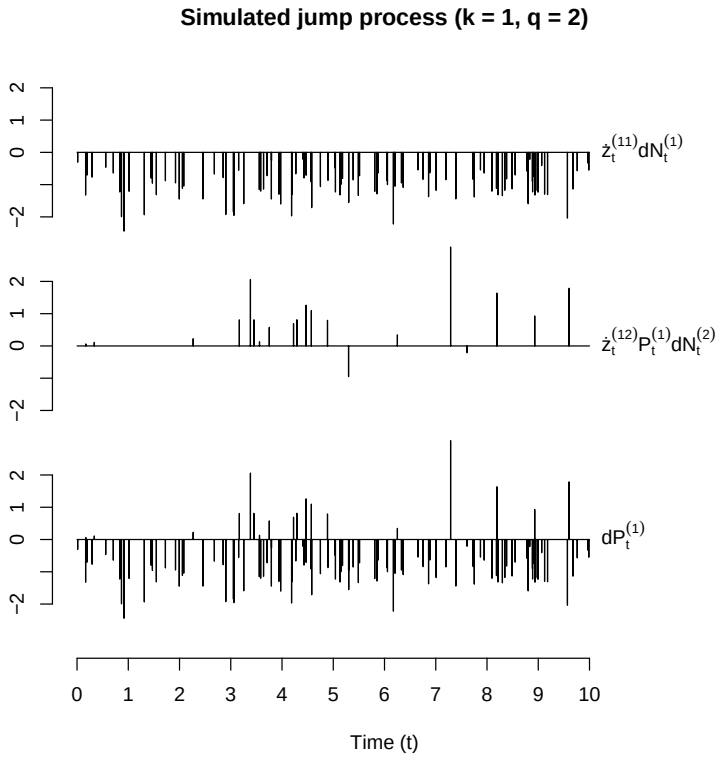


FIGURE 4.2.1: Construction of the jump process from the increments of individual jump constituents for $k = 1$ and $q = 2$. Here $\epsilon_{11}(P_t^{(1)}, \dot{z}_t^{(11)}, t) = z_t^{(11)}$ with $z_t^{(11)} \sim \mathrm{N}(-1, 0.5^2)$, $\epsilon_{12}(P_t^{(1)}, \dot{z}_t^{(21)}, t) = z_t^{(21)} P_t^{(1)}$ with $z_t^{(12)} \sim \mathrm{N}(0, 0.05^2)$ and $N_t^{(1)}$ and $N_t^{(2)}$ are subject to constant intensity functions, $\lambda_1 = 10$ and $\lambda_2 = 2$, respectively. $P_t^{(1)}$ is constructed by adding the increments of each jump process sequentially as they occur. Consequently the increments of $P_t^{(1)}$ simply reflect the combined increments of the individual jump components. R code: Supplementary materials, Section 4.2.

In addition to the jump constituents, the process is characterised by the rate at which jump realisations occur, i.e., the rate at which each increment in each $N_t^{(j)}$ occur. This is determined by the intensity vector, which is further allowed to depend on an external vector process $\dot{\mathbf{r}}_t$ as well as the current state of the process. For simplicity we will assume that the arrival rate of each of the Poisson components is also restricted to depend on only a single element of the vector $\dot{\mathbf{r}}_t$:

$$\lambda_j(\mathbf{P}_t, \dot{\mathbf{r}}_t, t) = \lambda_j(\mathbf{P}_t, \dot{r}_t^{(j)}, t) \tag{4.2.3}$$

where $\dot{r}_t^{(j)} \in \dot{\mathbf{r}}_t = \{\dot{r}_t^{(1)}, \dot{r}_t^{(2)}, \ldots, \dot{r}_t^{(k)}\}'$ and each $\dot{r}_t^{(j)}$ evolves independently according to a distribution function π_j. $\mathbf{P}_t$ thus represents a pure jump process that is characterised by the distributions of the jump-variables and the intensity processes which dictate the rate at which jump realisations occur. The purpose of this formulation is to encompass a class of jump processes with both state-dependent and stochastic intensity, whilst permitting multiple sources of randomness. For example, for $k = 1$ and $q = 2$ the trajectory of the process is subject to two jump sources with possibly distinct dynamics. This may be useful for cases where jumps observed in a real-world data have distinct sources with differing distributional characteristics. Likewise, allowing the intensity vector to be stochastic through the process $\dot{\mathbf{r}}_t$, it is possible to formulate a jump mechanism for which the frequency of jump realisations may pass stochastically through high and low intensity phases, which may be useful for modelling jump dynamics over long time periods in financial contexts or incorporating external drivers such as climate change in ecological models, for example.

Although pure jump processes are extremely useful modelling tools, a notable deficiency from the perspective of modelling continuously evolving processes is that the process remains dormant in a given state in between successive jump innovations. That is, for phenomena that exhibit jump behaviour but still evolve on small scales in between jumps, the present formulation does not suffice. In order to account for the stochastic evolution of the process between intermittent jumps, one may define a continuous mixture process consisting of a pure jump process and a diffusion process. The dynamics of the resulting k-dimensional jump diffusion $\mathbf{X}_t = \{X_t^{(1)}, X_t^{(2)}, \ldots, X_t^{(k)}\}'$ is then governed by the SDE:

$$d\mathbf{X}_t = \boldsymbol{\mu}(\mathbf{X}_t, t)dt + \boldsymbol{\sigma}(\mathbf{X}_t, t)d\mathbf{B}_t + d\mathbf{P}_t, \tag{4.2.4}$$

where, as before:

$$d\mathbf{P}_t = \mathbf{J}(\mathbf{X}_t, \dot{\mathbf{z}}_t, t)d\mathbf{N}_t \tag{4.2.5}$$

gives the jump mechanism with intensity vector $\boldsymbol{\lambda}(\mathbf{X}_t, \dot{\mathbf{r}}_t, t)$, $\boldsymbol{\mu}(\mathbf{X}_t, t) = (\mu_i(\mathbf{X}_t, t))_{k\times 1}$ gives the instantaneous drift vector, $\boldsymbol{\sigma}(\mathbf{X}_t, t) = (\sigma_{ij}(\mathbf{X}_t, t))_{k\times k}$ is the diffusion matrix of the process and $\mathbf{B}_t = (B_t^{(i)})_{k\times 1}$ is a vector of independent Brownian motions. Equation 4.2.4 thus constitutes a multivariate jump diffusion with state-dependent jumps and state-dependent and/or stochastic intensity. Consequently, the auxiliary variables contained $\dot{\mathbf{z}}_t$ have the effect of inducing discontinuous jumps in the otherwise continuous paths of the diffusion whenever the counting processes contained in $\mathbf{N}_t$ increment. Furthermore, the rate at which jumps occur may vary according to both the state of the jump diffusion and/or some external process $\dot{\mathbf{r}}_t$. The relationship between the various constituents of the jump diffusion can more easily be seen by writing equations 4.2.4 and 4.2.5 in

matrix form:

$$d\begin{bmatrix} X_t^{(1)} \\ \vdots \\ X_t^{(k)} \end{bmatrix} = \begin{bmatrix} \mu_1(\mathbf{X}_t, t) \\ \vdots \\ \mu_2(\mathbf{X}_t, t) \end{bmatrix} dt + \begin{bmatrix} \sigma_{11}(\mathbf{X}_t, t) & \dots & \sigma_{1k}(\mathbf{X}_t, t) \\ \vdots & \ddots & \vdots \\ \sigma_{k1}(\mathbf{X}_t, t) & \dots & \sigma_{kk}(\mathbf{X}_t, t) \end{bmatrix} d\begin{bmatrix} B_t^{(1)} \\ \vdots \\ B_t^{(k)} \end{bmatrix} + d\begin{bmatrix} P_t^{(1)} \\ \vdots \\ P_t^{(k)} \end{bmatrix} \tag{4.2.6}$$

where

$$d\begin{bmatrix} P_t^{(1)} \\ \vdots \\ P_t^{(k)} \end{bmatrix} = \begin{bmatrix} \epsilon_{11}(\mathbf{X}_t, \dot{z}_t^{(11)}, t) & \dots & \epsilon_{1q}(\mathbf{X}_t, \dot{z}_t^{(1q)}, t) \\ \vdots & \ddots & \vdots \\ \epsilon_{k1}(\mathbf{X}_t, \dot{z}_t^{(k1)}, t) & \dots & \epsilon_{kq}(\mathbf{X}_t, \dot{z}_t^{(kq)}, t) \end{bmatrix} d\begin{bmatrix} N_t^{(1)} \\ \vdots \\ N_t^{(q)} \end{bmatrix}. \tag{4.2.7}$$

Since Equation 4.2.4 is formulated in continuous time, the SDE can also be interpreted by relating the coefficients of the equation to its instantaneous moments: Let $(\gamma_{ij}(\mathbf{X}_t, t))_{k\times k} = \boldsymbol{\sigma}(\mathbf{X}_t, t)\boldsymbol{\sigma}'(\mathbf{X}_t, t)$ denote the covariance matrix of the diffusion. Given a jump SDE of the form of Equation 4.2.4, we have for $i, j = 1, 2, \dots k$:

$$\lim_{h\to 0} \frac{E[X_{t+h}^{(i)} - X_t^{(i)} | \mathbf{X}_t]}{h} = \mu_i(\mathbf{X}_t, t) + \sum_{m=1}^{q} E_{\dot{z}}[\epsilon_{im}(\mathbf{X}_t, \dot{\mathbf{z}}_t, t)] E_{\dot{r}}[\lambda_m(\mathbf{X}_t, \dot{\mathbf{r}}_t, t)] \tag{4.2.8}$$

and for $v + w \geq 2$

$$\lim_{h\to 0} \frac{E[(X_{t+h}^{(i)} - X_t^{(i)})^v (X_{t+h}^{(j)} - X_t^{(j)})^w | \mathbf{X}_t]}{h} =$$
$$\gamma_{ij}(\mathbf{X}_t, t)\mathbb{I}(v + w = 2) + \sum_{m=1}^{q} E_{\dot{z}}[\epsilon_{im}(\mathbf{X}_t, \dot{\mathbf{z}}_t, t)^v \epsilon_{jm}(\mathbf{X}_t, \dot{\mathbf{z}}_t, t)^w] E_{\dot{r}}[\lambda_m(\mathbf{X}_t, \dot{\mathbf{r}}_t, t)]. \tag{4.2.9}$$

Interestingly, equations 4.2.8 and 4.2.9 indicate that, on an infinitesimal scale, although the first two moments of a jump diffusion is dictated by a mixture of the diffusion and jump dynamics, the higher order instantaneous moments are completely determined by the jump mechanism of the process. Indeed, a quick comparison of these results to the instantaneous moments of a pure diffusion process gives some insight as to why this is.

Although the instantaneous dynamics of jump diffusion processes have both theoretical and practical applications, in the context of parametric inference we are often interested in the dynamical behaviour of the process over larger time horizons. As such, a pivotal quantity of interest in the analysis of jump diffusions is the evolution of the transitional density over time. Let $\{S \subseteq \mathbb{R}^k, \mathcal{X}, F\}$,

$\{\Psi_j, \mathcal{Z}_j, \phi_j\}_j$ and $\{\Omega_j, \mathcal{L}_j, \pi_j\}_j$ be probability spaces for $j = 1, 2, \ldots, q$ then the transitional density $f(\mathbf{X}_t|\mathbf{X}_s)$ of the process $\mathbf{X}_t$ at time t starting in $\mathbf{X}_s$ at time s is given by the Kolmogorov forward equation (Hanson, 2007):

$$\begin{aligned}\frac{\partial}{\partial t} f(\mathbf{X}_t|\mathbf{X}_s) = \\ -\sum_{i=1}^{k} \frac{\partial}{\partial X_t^{(i)}} \mu_i(\mathbf{X}_t, t) f(\mathbf{X}_t|\mathbf{X}_s) + \sum_{i=1}^{k}\sum_{j=1}^{k} \frac{\partial^2}{\partial X_t^{(i)} \partial X_t^{(j)}} \gamma_{ij}(\mathbf{X}_t, t) f(\mathbf{X}_t|\mathbf{X}_s) \\ + \sum_{j=1}^{q} \int_{\Psi_j}\int_{\Omega_j} \lambda_j(\boldsymbol{\nabla}(\mathbf{X}_t, \dot{\mathbf{z}}_t)^{(.j)}, \dot{r}_t^{(j)}, t) f(\boldsymbol{\nabla}(\mathbf{X}_t, \dot{\mathbf{z}}_t)^{(.j)}|\mathbf{X}_s)|\delta_j(\dot{\mathbf{z}}_t)| d\pi_j(\dot{r}_t^{(j)}) d\phi_j(\dot{\mathbf{z}}_t^{(.j)}) \\ - \sum_{j=1}^{q} \int_{\Psi_j}\int_{\Omega_j} \lambda_j(\mathbf{X}_t, \dot{r}_t^{(j)}, t) f(\mathbf{X}_t|\mathbf{X}_s) d\pi_j(\dot{r}_t^{(j)}) d\phi_j(\dot{\mathbf{z}}_t^{(.j)})\end{aligned} \tag{4.2.10}$$

where $(.j)$ again denotes the j-th column of a matrix and the elements $\boldsymbol{\nabla}(\mathbf{X}_t, \dot{\mathbf{z}}_t)^{(.j)} = \boldsymbol{\nu}_j(\mathbf{X}_t + \mathbf{J}(\mathbf{X}_t, \dot{\mathbf{z}}_t)^{(.j)}) = (\nu_{ij}(X_t^{(i)} + \epsilon_{ij}(\mathbf{X}_t, \dot{\mathbf{z}}_t)))_{k\times 1}$ and $|\delta_j(\dot{\mathbf{z}}_t)|$ have special meaning: The role of the function $\boldsymbol{\nu}_j(\mathbf{X}_t + \mathbf{J}(\mathbf{X}_t, \dot{\mathbf{z}}_t)^{(.j)})$ is to map jumps in such a way that the state of the process at time t, $\mathbf{X}_t$, is reached as the result of a jump occurrence at an instant just prior to t. $\boldsymbol{\nu}_j(\mathbf{X}_t + \mathbf{J}(\mathbf{X}_t, \dot{\mathbf{z}}_t)^{(.j)})$ thus has the action of reverting the state of the process to that which it was 'before' the jump occurrence. For example, if the jump matrix is independent of the process level and $\mathbf{J}(\mathbf{X}_t, \dot{\mathbf{z}}_t)^{(.j)} = \dot{\mathbf{z}}^{(.j)}$, then $\boldsymbol{\nu}_j(\mathbf{X}_t + \dot{\mathbf{z}}_t^{(.j)}) = \mathbf{X}_t - \dot{\mathbf{z}}_t^{(.j)}$. In turn, $|\delta_j(\dot{\mathbf{z}})|$ acts as the Jacobian resulting from the inversion. For example, following $\mathbf{J}(\mathbf{X}_t)^{(.j)} = \dot{\mathbf{z}}^{(.j)}$, $|\delta_j(\dot{\mathbf{z}})| = 1$. In order for Equation 4.2.10 to be well defined, we require boundary conditions on the equation. Similarly to jump-free diffusions, the initial conditions for Equation 4.2.10 are given by the multivariate Dirac delta function $f(\mathbf{x}_s|\mathbf{X}_s) = \delta(\mathbf{x}_s - \mathbf{X}_s)$ where

$$\delta(\mathbf{x}) = \begin{cases} \infty & \text{if } \mathbf{x} = \mathbf{0}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2.11}$$

However, due to the presence of the jump mechanism, we also require initial values for the jump components $\{\dot{r}_j(s) = \dot{r}_s, N_j(s) = 0 : j = 1, 2, \ldots, k\}$ and any additional time dependences – which we take to have known values at time s.

Although Equation 4.2.10 is extremely difficult to analyse, the transitional density forms the principal constituent to numerous techniques in the analysis of diffusion processes. In the sections that follow we develop a scheme for calculating approximate solutions to Equation 4.2.10 for time-inhomogeneous, non-linear jump diffusions, making it possible to analyse a wide spectrum of jump diffusion models.

4.3 Approximating the transitional density of a jump diffusion process

4.3.1 Deriving a partial difference differential equation for the moment generating function

For purposes of approximating the transitional density of a jump diffusion, it is useful to calculate the trajectories of the various moments of the jump diffusion. In order to derive the moment equations of a jump diffusion – a system of equations that govern the dynamics of the moments of the process – we make use of the moment generating function of the process in a similar fashion as for the cumulant truncation procedure under the methodology of Varughese (2013). This strategy has been applied with great success to population models by Marion *et al.* (2000) and Varughese and Fatti (2008), and scalar and bivariate stochastic epidemic models by Krishnarajah *et al.* (2005) and Krishnarajah *et al.* (2007), whereby the authors derive moment equations for various model types and make use of a moment closure approximation in order to accurately approximate the distribution of the process. Unfortunately the moment generating function is not available directly. As such, we derive a partial difference differential equation[1] (PDDE) which governs the evolution of the moment generating function over time, and can subsequently be used to identify systematic relationships between the moments of the process. For purposes of this section and the analysis that follows, we demonstrate how this can be achieved by focusing on bivariate jump diffusion processes. That is, for $k = 2$ the corresponding jump stochastic differential equation is given by:

$$d\begin{bmatrix} X_t \\ Y_t \end{bmatrix} = \begin{bmatrix} \mu_1(X_t, Y_t, t) \\ \mu_2(X_t, Y_t, t) \end{bmatrix} dt + \begin{bmatrix} \sigma_{11}(X_t, Y_t, t) & \sigma_{12}(X_t, Y_t, t) \\ \sigma_{21}(X_t, Y_t, t) & \sigma_{22}(X_t, Y_t, t) \end{bmatrix} d\begin{bmatrix} B_t^{(1)} \\ B_t^{(2)} \end{bmatrix} + d\begin{bmatrix} P_t^{(1)} \\ P_t^{(2)} \end{bmatrix} \tag{4.3.1}$$

where, for for example when $q = 2$, we have

$$d\begin{bmatrix} P_t^{(1)} \\ P_t^{(2)} \end{bmatrix} = \begin{bmatrix} \epsilon_{11}(X_t, Y_t, \dot{z}_t^{(11)}, t) & \epsilon_{12}(X_t, Y_t, \dot{z}_t^{(12)}, t) \\ \epsilon_{21}(X_t, Y_t, \dot{z}_t^{(21)}, t) & \epsilon_{22}(X_t, Y_t, \dot{z}_t^{(22)}, t) \end{bmatrix} d\begin{bmatrix} N_t^{(1)} \\ N_t^{(2)} \end{bmatrix}, \tag{4.3.2}$$

where $N_t^{(1)}$ and $N_t^{(2)}$ are counting processes with intensity parameters given by $\lambda_1(X_t, Y_t, \dot{r}_t^{(1)}, t)$ and $\lambda_2(X_t, Y_t, \dot{r}_t^{(2)}, t)$ respectively. The auxiliary random

[1] In Merton (1976), the term 'differential-difference' is used to describe a similar type of partial differential equation.

variables are then characterized by the distributions:

$$\begin{aligned} \dot{r}_t^{(1)} &\sim \pi_1, \\ \dot{r}_t^{(2)} &\sim \pi_2, \\ \{\dot{z}_t^{(11)}, \dot{z}_t^{(21)}\}' &\sim \phi_1, \\ \{\dot{z}_t^{(12)}, \dot{z}_t^{(22)}\}' &\sim \phi_2. \end{aligned} \tag{4.3.3}$$

As usual, $\mu_i(X_t, Y_t, t)$ and $\sigma_{ij}(X_t, Y_t, t)$ give the drift and diffusion coefficients which constitute the diffusion part of the process, whilst $\lambda_i(X_t, Y_t, \dot{r}_t^{(i)}, t)$, $\epsilon_{ij}(X_t, Y_t, \dot{z}_t^{(ij)}, t)$ and the distributions π_i and ϕ_{ij} characterise the jump mechanism of the process. Although the order (q) of the jump mechanism in general is allowed to vary – meaning that the jump mechanism may contain an arbitrary number of jump processes – we shall assume for ease of notation that $q = 2$ for the remainder of this section. Furthermore, although all of the components of the jump mechanism are allowed to be time dependent – whether parametrically or through the respective distributions of the random variables they contain – we will minimise any notational reference to time dependence throughout the calculations that follow and recall it where needed. As such we simplify the notation of the jump and intensity variables to $\dot{r}_t^{(i)} = \dot{r}_i$ and $\dot{z}_t^{(ij)} = \dot{z}_{ij}$ respectively and drop any time parameters contained in the coefficients of the process.

We start by analysing the moment generating function with reference to the Kolmogorov equation: Let

$$M(\alpha, \beta, t) = \iint_S e^{\alpha x + \beta y} f(x, y) dx dy \tag{4.3.4}$$

denote the moment generating function of the bivariate jump diffusion $\mathbf{X}_t = \{X_t, Y_t\}$. Then, by multiplying both sides of Equation 4.2.10 by $e^{\alpha x + \beta y}$ and integrating over x and y we arrive at the equation:

$$\frac{\partial}{\partial t} M(\alpha, \beta, t) = -\sum_{i=1}^{2} A_i(\alpha, \beta, t) + \sum_{i=1}^{2}\sum_{j=1}^{2} B_{ij}(\alpha, \beta, t) + \sum_{i=1}^{2} C_i(\alpha, \beta, t) \tag{4.3.5}$$

where

$$\begin{aligned} A_1(\alpha, \beta, t) &= \iint_S e^{\alpha x + \beta y} \frac{\partial}{\partial x} \big[\mu_1(x, y, t) f(x, y)\big] dy dx \\ A_2(\alpha, \beta, t) &= \iint_S e^{\alpha x + \beta y} \frac{\partial}{\partial y} \big[\mu_2(x, y, t) f(x, y)\big] dy dx, \end{aligned} \tag{4.3.6}$$

and

$$\begin{aligned}
B_{11}(\alpha,\beta,t) &= \iint_S e^{\alpha x+\beta y}\frac{\partial^2}{\partial x^2}\big[\gamma_{11}(x,y,t)f(x,y)\big]dydx \\
B_{12}(\alpha,\beta,t) &= \iint_S e^{\alpha x+\beta y}\frac{\partial^2}{\partial x\partial y}\big[\gamma_{12}(x,y,t)f(x,y)\big]dydx \\
B_{21}(\alpha,\beta,t) &= \iint_S e^{\alpha x+\beta y}\frac{\partial^2}{\partial x\partial y}\big[\gamma_{21}(x,y,t)f(x,y)\big]dydx \\
B_{22}(\alpha,\beta,t) &= \iint_S e^{\alpha x+\beta y}\frac{\partial^2}{\partial y^2}\big[\gamma_{22}(x,y,t)f(x,y)\big]dydx,
\end{aligned} \tag{4.3.7}$$

with

$$\begin{aligned}
&C_1(\alpha,\beta,t) = \\
&\int\cdots\int e^{\alpha x+\beta y}\lambda_1(\nabla_{\epsilon_{11}}(x),\nabla_{\epsilon_{21}}(y),\dot r_1,t)f(\nabla_{\epsilon_{11}}(x),\nabla_{\epsilon_{21}}(y))d\pi_1(\dot r_1)d\phi_1(\dot z_{11},\dot z_{21})dxdy \\
&-\int\cdots\int e^{\alpha x+\beta y}\lambda_1(x,y,\dot r_1,t)f(x,y)d\pi_1(\dot r_1)d\phi_1(\dot z_{11},\dot z_{21})dxdy
\end{aligned} \tag{4.3.8}$$

where $\nabla_{\epsilon_{11}}(x) = \nu_{11}(x+\epsilon_{11}(x,y,\dot z_{11},t))$, $\nabla_{\epsilon_{21}}(y) = \nu_{21}(y+\epsilon_{21}(x,y,\dot z_{21},t))$, and

$$\begin{aligned}
&C_2(\alpha,\beta,t) = \\
&\int\cdots\int e^{\alpha x+\beta y}\lambda_2(\nabla_{\epsilon_{12}}(x),\nabla_{\epsilon_{22}}(y),\dot r_2,t)f(\nabla_{\epsilon_{12}}(x),\nabla_{\epsilon_{22}}(y))d\pi_2(\dot r_2)d\phi_2(\dot z_{12},\dot z_{22})dxdy \\
&-\int\cdots\int e^{\alpha x+\beta y}\lambda_2(x,y,\dot r_2,t)f(x,y)d\pi_2(\dot r_2)d\phi_1(\dot z_{12},\dot z_{22})dxdy
\end{aligned} \tag{4.3.9}$$

with $\nabla_{\epsilon_{12}}(x) = \nu_{12}(x+\epsilon_{12}(x,y,\dot z_{12},t))$ and $\nabla_{\epsilon_{22}}(y) = \nu_{22}(y+\epsilon_{22}(x,y,\dot z_{22},t))$.

Subsequently, using integration by parts and some simplifying assumptions about the behaviour of the process (see Appendix D.1), one can rewrite the moment generating function as:

$$\begin{aligned}
\frac{\partial}{\partial t}M(\alpha,\beta,t) =& \alpha\mu_1\left(\frac{\partial}{\partial\alpha},\frac{\partial}{\partial\beta},t\right)M(\alpha,\beta,t) + \beta\mu_2\left(\frac{\partial}{\partial\alpha},\frac{\partial}{\partial\beta},t\right)M(\alpha,\beta,t) \\
&+ \alpha^2\gamma_{11}^2\left(\frac{\partial}{\partial\alpha},\frac{\partial}{\partial\beta},t\right)M(\alpha,\beta,t) + \alpha\beta\gamma_{12}^2\left(\frac{\partial}{\partial\alpha},\frac{\partial}{\partial\beta},t\right)M(\alpha,\beta,t) \\
&+ \alpha\beta\gamma_{21}^2\left(\frac{\partial}{\partial\alpha},\frac{\partial}{\partial\beta},t\right)M(\alpha,\beta,t) + \beta^2\gamma_{22}^2\left(\frac{\partial}{\partial\alpha},\frac{\partial}{\partial\beta},t\right)M(\alpha,\beta,t) \\
&+ C_1(\alpha,\beta,t) + C_2(\alpha,\beta,t).
\end{aligned} \tag{4.3.10}$$

Equation 4.3.10 thus expresses the moment generating function as a partial difference differential equation of which the differential terms are dictated by the functional form of the drift and diffusion elements, and for which the effect of the jump mechanism is dictated by the terms $C_1(\alpha,\beta,t)$ and $C_2(\alpha,\beta,t)$. From

these equations, it can be seen that the functional form of the $\epsilon_{ij}(x, y, \dot{z}_{ij}, t)$ and $\lambda_2(x, y, \dot{r}_2, t)$ terms ultimately determine what the expressions for $C_1(\alpha, \beta, t)$ and $C_2(\alpha, \beta, t)$ look like. When the intensity and jump coefficients are polynomial in x and y, it is possible to write sequence expressions for $C_1(\alpha, \beta, t)$ and $C_2(\alpha, \beta, t)$. For example, by writing $e^{\alpha x+\beta y}$ as a Taylor series and applying a change of variables

$$\begin{bmatrix} x^\star \\ y^\star \end{bmatrix} = \begin{cases} \begin{bmatrix} \nu_{11}(x + \epsilon_{11}(x, y, \dot{z}_{11}, t)) \\ \nu_{21}(y + \epsilon_{21}(x, y, \dot{z}_{21}, t)) \end{bmatrix} & \text{for } C_1(\alpha, \beta, t), \\ \\ \begin{bmatrix} \nu_{12}(x + \epsilon_{12}(x, y, \dot{z}_{12}, t)) \\ \nu_{22}(y + \epsilon_{22}(x, y, \dot{z}_{22}, t)) \end{bmatrix} & \text{for } C_2(\alpha, \beta, t), \end{cases} \tag{4.3.11}$$

it can be shown that:

$$C_1(\alpha, \beta, t) = E_{\dot{r}_1}\left[\lambda_1\left(\frac{\partial}{\partial\alpha}, \frac{\partial}{\partial\beta}, \dot{r}_1, t\right) \times \left(M_1^\star(\alpha, \beta, t) - M(\alpha, \beta, t)\right)\right] \tag{4.3.12}$$

and

$$C_2(\alpha, \beta, t) = E_{\dot{r}_2}\left[\lambda_2\left(\frac{\partial}{\partial\alpha}, \frac{\partial}{\partial\beta}, \dot{r}_2, t\right) \times \left(M_2^\star(\alpha, \beta, t) - M(\alpha, \beta, t)\right)\right], \tag{4.3.13}$$

where

$$M_k^\star(\alpha, \beta, t) = \\ \sum_{i=0}^{\infty}\sum_{j=0}^{i} \frac{\alpha^i\beta^{i-j}}{i!(i-j)!} E_{X,Y,\dot{z}}\left[\left(\sum_{r=0}^{i}\binom{i}{r} X_t^r \epsilon_{1k}(X_t, Y_t, \dot{z}_{1k})^{j-r}\right)\left(\sum_{s=0}^{i-j}\binom{i-j}{s} Y_t^s \epsilon_{2k}(X_t, Y_t, \dot{z}_{2k})^{i-j-s}\right)\right] \tag{4.3.14}$$

for $k = 1, 2$ (see Appendix D.2). Finally, by writing the moment generating function as a sequence

$$M(\alpha, \beta, t) = \sum_{i=0}^{\infty}\sum_{j=0}^{i} \frac{\alpha^i\beta^{i-j}}{i!(i-j)!} \iint_S x^i y^{i-j} f(x, y) dy dx \tag{4.3.15}$$

and plugging the appropriate expressions into Equation 4.3.10, it is possible to derive a system of equations that relate the moments of the jump diffusion process to each other. That is, by expanding the sequences $M(\alpha, \beta, t)$, $M_1^\star(\alpha, \beta, t)$, and $M_2^\star(\alpha, \beta, t)$ and subsequently applying the differential operators in Equation 4.3.10, we can equate the α and β coefficients of the resulting terms in order to derive a system of ordinary differential equations (ODEs) that govern the evolution of the moments of the jump diffusion. In the section that follows, we apply this technique to a class of non-linear processes for which we can accurately

approximate the transitional density.

4.3.2 Moment equations and the generalised quadratic jump diffusions

By deriving a PDDE for the moment generating function we can exploit the sequence properties of the moment generating function in order to derive a system of ODEs that govern the evolution of the moments of the jump diffusion process. Using this technique we can we can derive the moment equations of a wide range of multivariate jump diffusion processes. For purposes of this thesis, we extend the jump-free generalised quadratic diffusions to the spectrum of jump diffusion models by further developing the generalised quadratic framework to include suitably general jump mechanisms. Consequently we define the generalised quadratic jump diffusions, or 'jump-GQDs' for short, as follows: Let

$$\begin{aligned} dX_t &= \mu(X_t, t)dt + \sigma(X_t, t)dB_t + dP_t \\ dP_t &= J(X_t, \dot{z}_t, t)dN_t \end{aligned} \tag{4.3.16}$$

with intensity $\lambda(X_t, \dot{r}_t, t)$ denote a scalar jump diffusion process, then the scalar generalised quadratic jump diffusions are characterised by the coefficients:

$$\mu(X_t, t) = \sum_{i=0}^{2} g_i(t) X_t^i \tag{4.3.17}$$

and

$$\sigma^2(X_t, t) = \sum_{i=0}^{2} q_i(t) X_t^i, \tag{4.3.18}$$

where we assume that the intensity and jump coefficients are of the form:

$$\lambda(X_t, \dot{r}_t, t) = \sum_{i=0}^{2} \lambda_i(\dot{r}_t, t) X_t^i, \tag{4.3.19}$$

and

$$J(X_t, \dot{z}_t, t) = \sum_{i=0}^{1} h_i(\dot{z}_t, t) X_t^i \tag{4.3.20}$$

respectively, subject to the constraint:

$$\frac{\partial^2}{\partial x^2}\big(\lambda(x, \dot{r}_t, t) \times J(x, \dot{z}_t, t)\big) = K \tag{4.3.21}$$

for some constant K. For example:

$$\lambda(X_t, \dot{r}_t, t) = \lambda_0(t) + \lambda_1(t)X_t + \lambda_2(t)X_t^2 \tag{4.3.22}$$

with

$$J(X_t, \dot{z}_t, t) = \dot{z}_t, \tag{4.3.23}$$

or

$$\lambda(X_t, \dot{r}_t) = \lambda_0(t) + \lambda_1(t)X_t \tag{4.3.24}$$

with

$$J(X_t, \dot{z}_t) = \dot{z}_t \times X_t. \tag{4.3.25}$$

are both valid specifications for the coefficients of the jump mechanism. The second-order restrictions placed on the drift, diffusion and jump mechanism coefficients together constitute the scalar generalised quadratic jump diffusions.

By exploiting the sequence properties of the moment generating functions, we may derive a system of ODEs that governs the evolution of the moments of a jump diffusion process. Let $m_i(t)$ denote the i-th non-central moment of the process X_t then:

$$M(\alpha, t) = \sum_{i=0}^{\infty} \frac{\alpha^i}{i!} m_i(t). \tag{4.3.26}$$

Subsequently, by plugging $M(\alpha, t)$ into the scalar counterpart of Equation 4.3.10 under the drift, diffusion, and jump specification of the generalised quadratic framework, we may derive a general expression for the moment equations of the process. For example when $\lambda(X_t, \dot{\mathbf{r}}_t) = \dot{r}_t + \lambda_1(t)X_t + \lambda_2(t)X_t^2$ and $J(X_t, \dot{z}_t) = \dot{z}_t$:

$$\begin{aligned}
\frac{\partial}{\partial t} m_i(t) = {} & i\left(\sum_{k=0}^{2} g_k(t) m_{i+k-1}(t)\right) \\
& + \frac{i(i-1)}{2}\left(\sum_{k=0}^{2} q_k(t) m_{i+k-2}(t)\right)\mathbb{I}_{i\geq 2} \\
& + E_{\dot{r}}\left(\dot{r}_t\right)\left(\sum_{k=0}^{i} \binom{i}{k} m_k(t)\rho_{i-k}(t) - m_i(t)\right) \\
& + \sum_{l=1}^{2} \lambda_l(t)\left(\sum_{k=0}^{i} \binom{i}{k} m_{k+l-1}(t)\rho_{i-k+l}(t) - m_{i+l-1}(t)\right),
\end{aligned} \tag{4.3.27}$$

where ρ_i denotes the i-th non-central moment of the random variable $\dot{z}_t$. Alternatively, if $\lambda(X_t, \dot{r}_t) = \dot{r}_t + \lambda_1(t)X_t$ and $J(X_t, \dot{z}_t) = \dot{z}_t X_t$ we have:

$$\begin{aligned}\frac{\partial}{\partial t}m_i(t) &= i\bigg(\sum_{k=0}^{2} g_k(t)m_{i+k-1}(t)\bigg) \\ &\quad + \frac{i(i-1)}{2}\bigg(\sum_{k=0}^{2} q_k(t)m_{i+k-2}(t)\bigg)\mathbb{I}_{i\geq 2} \\ &\quad + E_{\dot{r}}\big(\dot{r}_t\big)\bigg(m_i(t)\sum_{k=0}^{i}\binom{i}{k}\rho_{i-k}(t) - m_i(t)\bigg) \\ &\quad + \lambda_1(t)\bigg(m_{i+1}(t)\sum_{k=0}^{i}\binom{i}{k}\rho_{i-k}(t) - m_{i+1}(t)\bigg),\end{aligned} \tag{4.3.28}$$

where ρ_i denotes the i-th non-central moment of the random variable $\dot{z}_t$. Using similar arguments, we can derive the moment equations of multivariate jump diffusion processes. For example, the dynamics of the corresponding bivariate generalised quadratic jump diffusion is given by the SDE:

$$\begin{aligned}d\begin{bmatrix}X_t \\ Y_t\end{bmatrix} &= \begin{bmatrix}\sum_{i+j\leq 2} a_{ij}(t)X_t^i Y_t^j \\ \sum_{i+j\leq 2} b_{ij}(t)X_t^i Y_t^j\end{bmatrix} dt + \begin{bmatrix}\sigma_{11}(X_t, Y_t, t) & \sigma_{12}(X_t, Y_t, t) \\ \sigma_{21}(X_t, Y_t, t) & \sigma_{22}(X_t, Y_t, t)\end{bmatrix} d\begin{bmatrix}B_t^{(1)} \\ B_t^{(2)}\end{bmatrix} \\ &\quad + \begin{bmatrix}\epsilon_{11}(X_t, Y_t, \dot{z}_{11}, t) & \epsilon_{12}(X_t, Y_t, \dot{z}_{12}, t) \\ \epsilon_{21}(X_t, Y_t, \dot{z}_{21}, t) & \epsilon_{22}(X_t, Y_t, \dot{z}_{22}, t)\end{bmatrix} d\begin{bmatrix}P_t^{(1)} \\ P_t^{(2)}\end{bmatrix}\end{aligned} \tag{4.3.29}$$

with

$$\boldsymbol{\sigma}(X_t, Y_t, t)\boldsymbol{\sigma}'(X_t, Y_t, t) = \begin{bmatrix}\sum_{i+j\leq 2} c_{ij}(t)X_t^i Y_t^j & \sum_{i+j\leq 2} d_{ij}(t)X_t^i Y_t^j \\ \sum_{i+j\leq 2} e_{ij}(t)X_t^i Y_t^j & \sum_{i+j\leq 2} f_{ij}(t)X_t^i Y_t^j\end{bmatrix}, \tag{4.3.30}$$

$$\epsilon_{ij}(X_t, Y_t, \dot{z}_{ij}, t) = \sum_{p+q\leq 1} g_{pq}^{ij}(t, \dot{z}_{ij})X_t^p Y_t^q, \tag{4.3.31}$$

for $i, j = 1, 2$, and

$$\lambda_i(X_t, Y_t, \dot{r}_i, t) = \sum_{p+q\leq 2} h_{pq}^{i}(t, \dot{r}_i)X_t^p Y_t^q \tag{4.3.32}$$

for $i = 1, 2$. In addition to the second-order restrictions on the drift and diffusion terms, we assume that:

$$\begin{aligned} \frac{\partial^2}{\partial x^2}\big(\lambda_i(x, y, \dot{r}_i, t) \times \epsilon_{ij}(x, y, \dot{z}_{ij}, t)\big) &= K_1, \\ \frac{\partial^2}{\partial y^2}\big(\lambda_i(x, y, \dot{r}_i, t) \times \epsilon_{ij}(x, y, \dot{z}_{ij}, t)\big) &= K_2, \end{aligned} \tag{4.3.33}$$

for $i, j = 1, 2$ and some constants K_1 and K_2. That is, the products of the intensity function and the corresponding rows of the jump matrix may not exceed second-order non-linearity in x and y. In simple terms, the class of bivariate quadratic jump diffusions extend the jump-free generalised quadratic diffusions by incorporating a jump process characterized by an arrival rate that may have state-dependent intensity and jump elements.

By applying the techniques outlined in Section 4.3.1 to Equation 4.3.29, it is possible to derive an expression for the moment generating function of the bivariate generalised quadratic template diffusion. From this we may derive a general expression for the system of ODEs that govern the evolution of the non-central moments of the bivariate jump-GQD. Let:

$$\begin{aligned} M^{(d)}(\alpha, \beta, t) &= \sum_{i=0}^{d} \sum_{j=0}^{i} \frac{\alpha^i \beta^{i-j}}{i!(i-j)!} \iint_S x^i y^{i-j} f(x, y) dy dx \\ &= \sum_{i=0}^{d} \sum_{j=0}^{i} \frac{\alpha^i \beta^{i-j}}{i!(i-j)!} E_{X,Y}(X_t^i Y_t^{i-j}) \end{aligned} \tag{4.3.34}$$

then, by plugging $M^{(d)}(\alpha, \beta, t)$ into Equation 4.3.10 for some truncation order d, we may derive the moment equations for the bivariate generalised quadratic jump diffusion. Let $m_{i,j} = E_{X,Y}(X_t^i Y_t^j)$ denote the ij-th non-central moment of the process, then the system of ODEs that govern the evolution of the non-central

moments of Equation 4.3.29 is given by:

$$\begin{aligned}\frac{\partial}{\partial t}m_{i,j}(t) &= i\Big(\sum_{r+s\leq 2} a_{rs}(t)m_{i+r-1,j+s}(t)\Big)\\ &+ j\Big(\sum_{r+s\leq 2} b_{rs}(t)m_{i+r,j+s-1}(t)\Big)\\ &+ \frac{i(i-1)}{2}\Big(\sum_{r+s\leq 2} c_{rs}(t)m_{i+r-2,j+s}(t)\Big)\mathbb{I}_{i\geq 2}\\ &+ ij\Big(\sum_{r+s\leq 2} d_{rs}(t)m_{i+r-1,j+s-1}(t)\Big)\mathbb{I}_{i,j\geq 1}\\ &+ ij\Big(\sum_{r+s\leq 2} e_{rs}(t)m_{i+r-1,j+s-1}(t)\Big)\mathbb{I}_{i,j\geq 1}\\ &+ \frac{j(j-1)}{2}\Big(\sum_{r+s\leq 2} f_{rs}(t)m_{i+r,j+s-2}(t)\Big)\mathbb{I}_{j\geq 2}\\ &+ E_{\dot{r}_1}\big(\lambda_1(\Gamma_x,\Gamma_y,t,\dot{r}_1)\big)\eta_1(i,j)\\ &+ E_{\dot{r}_2}\big(\lambda_2(\Gamma_x,\Gamma_y,t,\dot{r}_2)\big)\eta_2(i,j)\end{aligned} \tag{4.3.35}$$

for $i+j\leq d$. Here, the expressions for $\eta_1(i,j)$ and $\eta_2(i,j)$ are identified as the collection of terms in

$$M_1^{\star(d)}\big(\alpha,\beta,t\big) - M^{(d)}(\alpha,\beta,t) \tag{4.3.36}$$

and

$$M_2^{\star(d)}\big(\alpha,\beta,t\big) - M^{(d)}(\alpha,\beta,t) \tag{4.3.37}$$

with leading coefficients $\alpha^i\beta^j$, where

$$M_k^{\star(d)}\big(\alpha,\beta,t\big) = \sum_{i=0}^{d}\sum_{j=0}^{i}\frac{\alpha^i\beta^{i-j}}{i!(i-j)!}R_{i,i-j,k} \tag{4.3.38}$$

with

$$\begin{aligned}&R_{i,i-j,k} =\\ &E_{X,Y,\dot{z}}\Big(\Big[\sum_{r=0}^{i}\binom{i}{r}X_t^r\Big(\sum_{p+q\leq 2} g_{pq}^{k1}(\dot{z}_{k1},t)X_t^pY_t^q\Big)^{i-r}\Big]\Big[\sum_{s=0}^{i-j}\binom{i-j}{s}Y_t^s\Big(\sum_{p+q\leq 2} g_{pq}^{2k}(\dot{z}_{2k},t)X_t^pY_t^q\Big)^{i-j-s}\Big]\Big),\end{aligned} \tag{4.3.39}$$

and the operators Γ_x and Γ_y have the action

$$\Gamma_x m_{i,j}(t) = m_{i+1,j}(t) \tag{4.3.40}$$

and

$$\Gamma_y m_{i,j}(t) = m_{i,j+1}(t) \tag{4.3.41}$$

respectively. Equation 4.3.38 in turn is derived by plugging Equation 4.3.31 into Equation 4.3.14 under the assumptions of the generalised quadratic framework. Naturally, depending on the functional form of the jump matrix, the expressions for $\eta_1(i,j)$ and $\eta_2(i,j)$ may vary. For example, given a jump-GQD with purely additive jumps:

$$\mathbf{J}(X_t, Y_t, t) = \begin{bmatrix} \dot{z}_{11} & \dot{z}_{12} \\ \dot{z}_{21} & \dot{z}_{22} \end{bmatrix}, \tag{4.3.42}$$

the resulting expressions for $\eta_1(i,j)$ and $\eta_2(i,j)$ are relatively simple. Under a $d = 4$-th order truncation, the resulting expressions for each pair $i, j : i + j \leq 4$ are listed in Table 4.3.1. To see how these terms arise, consider the term $\eta_1(1,1)$. From Equation 4.3.38 we have

$$\begin{aligned} R_{1,1,1} &= E_{X,Y,\dot{z}}\left(\left[\sum_{r=0}^{1}\binom{1}{r}X_t^r \dot{z}_{11}^{1-r}\right]\left[\sum_{s=0}^{1}\binom{1}{s}Y_t^s \dot{z}_{21}^{1-s}\right]\right) \\ &= E_{X,Y,\dot{z}}\left(\dot{z}_{11}\dot{z}_{21}X_t^0 Y_t^0 + \dot{z}_{11}^1\dot{z}_{21}^0 X_t^0 Y_t^1 + \dot{z}_{11}^0\dot{z}_{21}^1 X_t^1 Y_t^0 + \dot{z}_{11}^0\dot{z}_{21}^0 X_t^1 Y_t^1\right) \\ &= m_{00}(t)\rho_{11}(t) + m_{01}(t)\rho_{10}(t) + m_{10}(t)\rho_{01}(t) + m_{11}(t)\rho_{00}(t), \end{aligned} \tag{4.3.43}$$

corresponding to the coefficients $\alpha^1\beta^1$ in the sequence $M_1^{\star(d)}(\alpha,\beta)$, where $\rho_{ij}(t)$ denotes the ij-th non-central moments of the variable pair $\{\dot{z}_{11}, \dot{z}_{21}\}'$. The corresponding term for the coefficients $\alpha^1\beta^1$ in $M^{(d)}(\alpha,\beta,t)$ on the other hand consists solely of $m_{11}(t)$. Subtracting the latter results in the sequence:

$$m_{00}(t)\rho_{11}(t) + m_{01}(t)\rho_{10}(t) + m_{10}(t)\rho_{01}(t), \tag{4.3.44}$$

corresponding to the entry in column one, row nine of Table 4.3.1. Subsequently, depending on the form of the intensity coefficients $\lambda_1(X_t, Y_t, \dot{r}_t, t)$ and $\lambda_2(X_t, Y_t, \dot{r}_t, t)$, the index operators can then be applied to the expressions in Table 4.3.1 in order to derive the appropriate system of moment equations from Equation 4.3.35.

ij	$\eta_1(i,j)$	$\eta_2(i,j)$
10	$1\rho_{10}m_{00}$	$1\omega_{10}m_{00}$
20	$2\rho_{10}m_{10}+1\rho_{20}m_{00}$	$2\omega_{10}m_{10}+1\omega_{20}m_{00}$
30	$3\rho_{10}m_{20}+3\rho_{20}m_{10}+1\rho_{30}m_{00}$	$3\omega_{10}m_{20}+3\omega_{20}m_{10}+1\omega_{30}m_{00}$
40	$4\rho_{10}m_{30}+6\rho_{20}m_{20}+4\rho_{30}m_{10}+1\rho_{40}m_{00}$	$4\omega_{10}m_{30}+6\omega_{20}m_{20}+4\omega_{30}m_{10}+1\omega_{40}m_{00}$
01	$1\rho_{01}m_{00}$	$1\omega_{01}m_{00}$
02	$2\rho_{01}m_{01}+1\rho_{02}m_{00}$	$2\omega_{01}m_{01}+1\omega_{02}m_{00}$
03	$3\rho_{01}m_{02}+3\rho_{02}m_{01}+1\rho_{03}m_{00}$	$3\omega_{01}m_{02}+3\omega_{02}m_{01}+1\omega_{03}m_{00}$
04	$4\rho_{01}m_{03}+6\rho_{02}m_{02}+4\rho_{03}m_{01}+1\rho_{04}m_{00}$	$4\omega_{01}m_{03}+6\omega_{02}m_{02}+4\omega_{03}m_{01}+1\omega_{04}m_{00}$
11	$\rho_{10}m_{01}+\rho_{01}m_{10}+\rho_{11}m_{00}$	$\omega_{10}m_{01}+\omega_{01}m_{10}+\omega_{11}m_{00}$
12	$\rho_{10}m_{02}+2\rho_{01}m_{11}+2\rho_{11}m_{01}+\rho_{02}m_{10}+\rho_{12}m_{00}$	$\omega_{10}m_{02}+2\omega_{01}m_{11}+2\omega_{11}m_{01}+\omega_{02}m_{10}+\omega_{12}m_{00}$
21	$2\rho_{10}m_{11}+\rho_{20}m_{01}+\rho_{01}m_{20}+2\rho_{11}m_{10}+\rho_{21}m_{00}$	$2\omega_{10}m_{11}+\omega_{20}m_{01}+\omega_{01}m_{20}+2\omega_{11}m_{10}+\omega_{21}m_{00}$
22	$2\rho_{10}m_{12}+\rho_{20}m_{02}+2\rho_{01}m_{21}+4\rho_{11}m_{11}+2\rho_{21}m_{01}+\rho_{02}m_{20}+2\rho_{12}m_{10}+\rho_{22}m_{00}$	$2\omega_{10}m_{12}+\omega_{20}m_{02}+2\omega_{01}m_{21}+4\omega_{11}m_{11}+2\omega_{21}m_{01}+\omega_{02}m_{20}+2\omega_{12}m_{10}+\rho_{22}m_{00}$
13	$\rho_{10}m_{03}+3\rho_{01}m_{12}+3\rho_{11}m_{02}+3\rho_{02}m_{11}+3\rho_{12}m_{01}+\rho_{03}m_{10}+\rho_{13}m_{00}$	$\omega_{10}m_{03}+3\omega_{01}m_{12}+3\omega_{11}m_{02}+3\omega_{02}m_{11}+3\omega_{12}m_{01}+\omega_{03}m_{10}+\omega_{13}m_{00}$
31	$3\rho_{10}m_{21}+3\rho_{20}m_{11}+\rho_{30}m_{01}+\rho_{01}m_{30}+3\rho_{11}m_{20}+3\rho_{21}m_{10}+\rho_{31}m_{00}$	$3\omega_{10}m_{21}+3\omega_{20}m_{11}+\omega_{30}m_{01}+\omega_{01}m_{30}+3\omega_{11}m_{20}+3\omega_{21}m_{10}+\omega_{31}m_{00}$

TABLE 4.3.1: Expressions for $\eta_1(i,j)$ and $\eta_2(i,j)$ for a purely additive jump matrix. Here ρ_{ij} and ω_{ij} respectively denote the ij-th non-central moments of the jump variable columns $\{\dot{z}_{11},\dot{z}_{21}\}'$ and $\{\dot{z}_{12},\dot{z}_{22}\}'$.

Finally, in order for the moment equations to be determinate, we require two things: Firstly, we require initial conditions for the non-central moments of the system in Equation 4.3.35. For a bivariate jump diffusion process starting in (X_s,Y_s) at time $s<t$, the appropriate initial conditions are given by:

$$m_{ij}(s)=E_{X,Y}(X_s^iY_s^j)=X_s^iY_s^j. \tag{4.3.45}$$

This follows since the point (X_s,Y_s) is occupied with certainty, and can be directly verified from Equation 4.2.10. Secondly, for models with quadratic drift, in which case the terms $a_{ij}(t)$, $b_{ij}(t)$ for $i+j=2$ are non-zero, the corresponding system of ODEs becomes indeterminate as the resulting system is of infinite dimension. That is, the trajectory of any non-central moment is dependent on successive higher order moments. Fortunately, within the quadratic framework, we can easily deal with this deficiency by formulating a truncated system whereby $(d+1)$-th order moments are replaced under the assumption of cumulant neglect. That is, we set all $(d+1)$-th order cumulants to zero and replace the resulting expression for the $(d+1)$-th order moments in terms of the first d-order moments by using a recursive relation for calculating moments in terms of cumulants (Harvey, 1972) (see Appendix D.3).

In conclusion, for any given polynomial jump diffusion process with jump random variables for which we can explicitly write out the non-central moments in terms of its parameters, we can derive a system of equations that govern the evolution of the moments of the process over time. Within the generalised quadratic framework, we can achieve this with a great degree of accuracy, making it possible to analyse time-inhomogeneous models with quadratic drift, stochastic volatility and state-dependent jump intensity. In the section that follows, we show how the moment equations may be used to analyse non-linear jump diffusion processes of varying degrees of complexity.

4.3.3 Example models

Before continuing the development of a scheme for accurately approximating the transitional density of a jump diffusion process, we demonstrate the use of the moment equations in the analysis of various non-linear models and make explicit the calculation of the moment equations and intermediate approximations of the transitional density.

4.3.3.1 Time-inhomogeneous CIR process with additive, normally distributed jumps and state-dependent intensity.

Consider a scalar jump diffusion process with dynamics given by the SDE:

$$\begin{aligned} dX_t &= \alpha_x(\beta_x + \nu_x \sin(2\pi t) - X_t)dt + \sigma_x\sqrt{X_t}dB_t + dP_t \\ dP_t &= \dot{z}_t dN_t \end{aligned} \tag{4.3.46}$$

with $\dot{z}_t \sim \mathrm{N}(\mu_z, \sigma_z^2)$ and $\lambda(X_t, t) = \kappa_0 + \kappa_1 X_t$. For purposes of the example, let $\{\alpha_x, \beta_x, \nu_x, \sigma_x, \kappa_0, \kappa_1, \mu_z, \sigma_z\} = \{1, 6, 2, 1, 1, 0.4, 1.5, 0.1\}$ and set the initial value for the process to $X_0 = 4$. Since Equation 4.3.46 is nested within the generalised quadratic jump diffusions we may easily derive the moment equations for the process using Equation 4.3.27. Under a 6-th order truncation, we can subsequently derive the moment equations for Equation 4.3.46 as:

$$\begin{aligned} m_1'(t) &= (\alpha_x\beta_x + \nu_x \sin(2\pi t)) - \alpha_x m_1(t) + j_1(t) + k_1(t) \\ m_2'(t) &= 2[(\alpha_x\beta_x + \nu_x \sin(2\pi t))m_1(t) - \alpha_x m_2(t)] + \sigma_x^2 m_1(t) + j_2(t) + k_2(t) \\ m_3'(t) &= 3[(\alpha_x\beta_x + \nu_x \sin(2\pi t))m_2(t) - \alpha_x m_3(t)] + \sigma_x^2 3m_2(t) + j_3(t) + k_3(t) \\ m_4'(t) &= 4[(\alpha_x\beta_x + \nu_x \sin(2\pi t))m_3(t) - \alpha_x m_4(t)] + \sigma_x^2 6m_3(t) + j_4(t) + k_4(t) \\ m_5'(t) &= 5[(\alpha_x\beta_x + \nu_x \sin(2\pi t))m_4(t) - \alpha_x m_5(t)] + \sigma_x^2 10m_4(t) + j_5(t) + k_5(t) \\ m_6'(t) &= 6[(\alpha_x\beta_x + \nu_x \sin(2\pi t))m_5(t) - \alpha_x m_6(t)] + \sigma_x^2 15m_5(t) + j_6(t) + k_6(t) \end{aligned} \tag{4.3.47}$$

with $m_i(0) = X_0^i$ for $i = 1, 2, \ldots, 6$ where

$$\begin{aligned}
j_1(t) &= \kappa_0\rho_1 \\
j_2(t) &= \kappa_0(2\rho_1 m_1(t) + \rho_2) \\
j_3(t) &= \kappa_0(3\rho_1 m_2(t) + 3\rho_2 m_1(t) + \rho_3) \\
j_4(t) &= \kappa_0(4\rho_1 m_3(t) + 6\rho_2 m_2(t) + 4\rho_3 m_1(t) + \rho_4) \\
j_5(t) &= \kappa_0(5\rho_1 m_4(t) + 10\rho_2 m_3(t) + 10\rho_3 m_2(t) + 5\rho_4 m_1(t) + \rho_5) \\
j_6(t) &= \kappa_0(6\rho_1 m_5(t) + 15\rho_2 m_4(t) + 20\rho_3 m_3(t) + 15\rho_4 m_2(t) + 6\rho_5 m_1(t) + \rho_6),
\end{aligned} \tag{4.3.48}$$

$$\begin{aligned}
k_1(t) &= \kappa_1(\rho_1 m_1(t)) \\
k_2(t) &= \kappa_1(2\rho_1 m_2(t) + \rho_2 m_1(t)) \\
k_3(t) &= \kappa_1(3\rho_1 m_3(t) + 3\rho_2 m_2(t) + \rho_3 m_1(t)) \\
k_4(t) &= \kappa_1(4\rho_1 m_4(t) + 6\rho_2 m_3(t) + 4\rho_3 m_2(t) + \rho_4 m_1(t)) \\
k_5(t) &= \kappa_1(5\rho_1 m_5(t) + 10\rho_2 m_4(t) + 10\rho_3 m_3(t) + 5\rho_4 m_2(t) + \rho_5 m_1(t)) \\
k_6(t) &= \kappa_1(6\rho_1 m_6(t) + 15\rho_2 m_5(t) + 20\rho_3 m_4(t) + 15\rho_4 m_3(t) + 6\rho_5 m_2(t) + \rho_6 m_1(t)),
\end{aligned} \tag{4.3.49}$$

and finally

$$\begin{aligned}
\rho_1 &= \mu_z \\
\rho_2 &= \mu_z^2 + \sigma_z^2 \\
\rho_3 &= \mu_z^3 + 3\mu_z^1\sigma_z^2 \\
\rho_4 &= \mu_z^4 + 6\mu_z^2\sigma_z^2 + 3\sigma_z^4 \\
\rho_5 &= \mu_z^5 + 10\mu_z^3\sigma_z^2 + 15\mu_z^1\sigma_z^4 \\
\rho_6 &= \mu_z^6 + 15\mu_z^4\sigma_z^2 + 45\mu_z^2\sigma_z^4 + 15\sigma_z^6.
\end{aligned} \tag{4.3.50}$$

The effect of the jump mechanism on the moments of the diffusion can immediately be seen through Equation 4.3.47 where the $j_i(t)$ and $k_i(t)$ dictate how the intensity and jump parameters feed into the moment structure of the process. For example, when $\mu_z = 0$ it follows that $\rho_1 = 0$ and the mean trajectory of the process remains unchanged from that of the jump-free process. Furthermore, if both $\mu_z = 0$ and $\sigma_z = 0$ or $\kappa = 0$, we recover the moment equations of the corresponding pure diffusion process.

As before, we can solve the moment equations numerically using standard Runge-Kutta methods in order to evaluate the moment trajectories of Equation 4.3.46 over time. In order to verify that the derived moment equations are indeed valid, we need to perform an independent check on the resulting moment trajectories. For these purposes we can simulate trajectories of Equation 4.3.46 over the desired transition horizon and calculate moment statistics with which we can compare the relevant quantities (see Appendix D.4 for details regarding the simulation of jump diffusion processes). Figure 4.3.1 compares the approximate moment

trajectories of Equation 4.3.46 for the first four non-central moments to simulated moment trajectories. The approximate moments are calculated using the (8)10-th order Runge-Kutta scheme of Feagin (2007), whilst the simulated moments are calculated from 10 000 simulated trajectories under a modified Euler-Murayama scheme outlined in Appendix D.4 using a step size of 0.001 time units. Indeed, the moment equations match the simulated moment trajectories very closely.

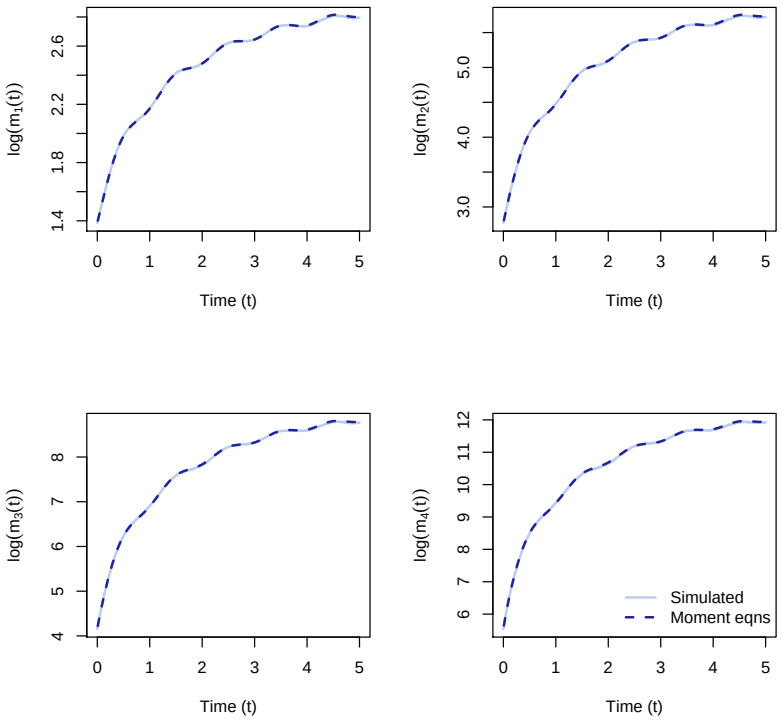


FIGURE 4.3.1: Log-scaled simulated moment trajectories (solid, light blue) and approximate moment equations (dashed, dark blue) for the first four non-central moments of Equation 4.3.46 over time. The moment equations match closely those of the simulated trajectories. R code: Supplementary materials, Section 4.3.

Citing the accuracy of the moment equations, it follows naturally that we can plug the moment equations into an appropriate surrogate density in order to approximate the transitional density. For example, here we apply the scalar

saddlepoint approximation:

$$f(X_t|X_s) \approx f^{(d)}_{SPT}(X_t|X_s) = \frac{1}{\sqrt{2\pi \frac{\partial^2}{\partial \alpha^2} K^{(d)}(\alpha_0, t)}} \exp\Big(K^{(d)}(\alpha_0, t) - \alpha_0 X_t \Big), \tag{4.3.51}$$

where

$$K^{(d)}(\alpha, t) = \sum_{i=1}^{d} \frac{\alpha^i \kappa_i(t)}{i!}, \tag{4.3.52}$$

α_0 solves

$$\frac{\partial}{\partial \alpha} K^{(d)}(\alpha, t) = X_t, \tag{4.3.53}$$

$\kappa_i(t)$ denotes the i-th cumulant of the process at time t, and d denotes the truncation order of the approximation. Figure 4.3.2 illustrates the transition density surface for $d = 4$ on the transition horizon $[0, 5]$. For reference, we compare the approximate transitional density to the frequency distribution of the simulated trajectories of Equation 4.3.46 at various time points. Interestingly, despite the significantly more complicated dynamics of the process as compared to a pure diffusion process, the density approximation remains accurate at the resolution studied here.

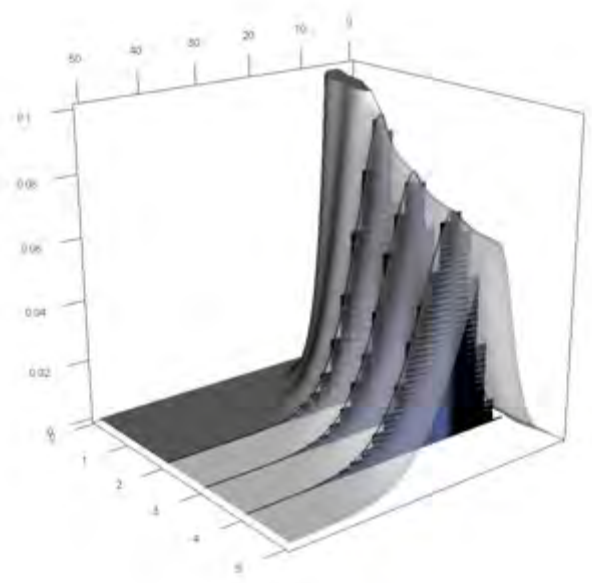


FIGURE 4.3.2: Approximate transition density (gray/lightgray)of Equation 4.3.46 and simulated transition density (light blue - dark blue) at times $t = 2$, $t = 3$ and $t = 4$. Transition density highlighted in black for each epoch of comparison. R code: Supplementary materials, Section 4.3.

4.3.3.2 Time-Inhomogeneous CIR process with additive, normally distributed jumps and stochastic intensity.

Under the methodology derived in sections 4.3.1 and 4.3.2 it is possible for the jump intensity of the process to evolve stochastically over time. For example, consider again a CIR process with additive jumps, but in this case let the intensity parameter be a two-state continuous time Markov chain:

$$\begin{aligned} dX_t &= \kappa_x(\mu_x - X_t)dt + \sigma_x(1 + 0.4\sin(\pi t))\sqrt{X_t}dB_t + dP_t \\ dP_t &= \dot{z}_t dN_t \end{aligned} \tag{4.3.54}$$

with $\dot{z}_t \sim \mathrm{N}(\mu_z, \sigma_z^2)$ and $\lambda(X_t, \dot{r}_t, t) = \dot{r}_t$ where the intensity parameter $\dot{r}_t$ has dynamics given by a continuous time Markov chain (CTMC):

$$\dot{r}_t = \begin{cases} \lambda_1 & \text{Low jump frequency,} \\ \lambda_2 & \text{High jump frequency,} \end{cases} \tag{4.3.55}$$

with transition rate matrix

$$R = \begin{pmatrix} -\beta_1 & \beta_1 \\ \beta_2 & -\beta_2 \end{pmatrix}. \tag{4.3.56}$$

Under the dynamics of Equation 4.3.54, the process exhibits linear drift and state-dependent volatility which varies periodically over time. In addition, the process is subject to randomly occurring jump events for which the jump intensity switches stochastically over time between levels λ_1 and λ_2. Since Equation 4.3.54 is nested within the scalar generalised quadratic framework, we may derive the moment equations of the process directly from Equation 4.3.27. In order to do so we need to evaluate the expectation of the intensity process over time. From the transition probability matrix of $\dot{r}_t$, the appropriate expression follows:

$$E(\dot{r}_t) = \begin{cases} \lambda_1 \frac{\beta_2 + \beta_1 e^{-(\beta_1+\beta_2)(t-s)}}{\beta_1+\beta_2} + \lambda_2 \frac{\beta_1(1-e^{-(\beta_1+\beta_2)(t-s)})}{\beta_1+\beta_2} & \text{if } \dot{r}_s = \lambda_1, \\ \lambda_2 \frac{\beta_1 + \beta_2 e^{-(\beta_1+\beta_2)(t-s)}}{\beta_1+\beta_2} + \lambda_1 \frac{\beta_2(1-e^{-(\beta_1+\beta_2)(t-s)})}{\beta_1+\beta_2} & \text{if } \dot{r}_s = \lambda_2. \end{cases} \tag{4.3.57}$$

Consequently, the moment equations for model 4.3.54 under a 6-th order truncation can be verified as:

$$\begin{aligned}
m_1'(t) &= \kappa_x \mu_x - \kappa_x m_1(t) + j_1(t) \\
m_2'(t) &= 2(\kappa_x \mu_x m_1(t) - \kappa_x m_2(t)) + \sigma_x^2 (1 + 0.4 \sin(\pi t))^2 m_1(t) + j_2(t) \\
m_3'(t) &= 3(\kappa_x \mu_x m_2(t) - \kappa_x m_3(t)) + \sigma_x^2 (1 + 0.4 \sin(\pi t))^2 3 m_2(t) + j_3(t) \\
m_4'(t) &= 4(\kappa_x \mu_x m_3(t) - \kappa_x m_4(t)) + \sigma_x^2 (1 + 0.4 \sin(\pi t))^2 6 m_3(t) + j_4(t) \\
m_5'(t) &= 5(\kappa_x \mu_x m_4(t) - \kappa_x m_5(t)) + \sigma_x^2 (1 + 0.4 \sin(\pi t))^2 10 m_4(t) + j_5(t) \\
m_6'(t) &= 6(\kappa_x \mu_x m_5(t) - \kappa_x m_6(t)) + \sigma_x^2 (1 + 0.4 \sin(\pi t))^2 15 m_5(t) + j_6(t)
\end{aligned} \tag{4.3.58}$$

with $m_i(s) = X_s^i$ for $i = 1, 2, \ldots, 6$ where

$$\begin{aligned}
j_1(t) &= E(\dot{r}_t)\rho_1 \\
j_2(t) &= E(\dot{r}_t)(2\rho_1 m_1(t) + \rho_2) \\
j_3(t) &= E(\dot{r}_t)(3\rho_1 m_2(t) + 3\rho_2 m_1(t) + \rho_3) \\
j_4(t) &= E(\dot{r}_t)(4\rho_1 m_3(t) + 6\rho_2 m_2(t) + 4\rho_3 m_1(t) + \rho_4) \\
j_5(t) &= E(\dot{r}_t)(5\rho_1 m_4(t) + 10\rho_2 m_3(t) + 10\rho_3 m_2(t) + 5\rho_4 m_1(t) + \rho_5) \\
j_6(t) &= E(\dot{r}_t)(6\rho_1 m_5(t) + 15\rho_2 m_4(t) + 20\rho_3 m_3(t) + 15\rho_4 m_2(t) + 6\rho_5 m_1(t) + \rho_6).
\end{aligned} \tag{4.3.59}$$

Here, the relations between the elements $\rho_1, \rho_2, \ldots, \rho_6$ and the parameters μ_z and σ_z follow straightforwardly from the non-central moments of the Normal distribution as in Equation 4.3.50.

For purposes of the experiment, we may again verify the moment equations by way of simulation. In order to do so, we modify the simulation scheme outlined in Appendix D.4 to reflect the stochastic intensity component of the process. Let τ be a scalar time index on the transition horizon $[s, t]$ and Δ a finite time step, then the revised iterative updating scheme for simulating a single trajectory of Equation 4.3.54 follows:

(1) Set $\tau = s$ and initialize the jump diffusion and intensity process X_τ and $\dot{r}_\tau$ respectively.

(2) (a) Set:

$$X_{\tau+\Delta} = X_\tau + \mu_i(X_\tau, \tau)\Delta + \sigma(X_\tau, \tau)Z, \tag{4.3.60}$$

by drawing $Z \sim \mathrm{N}(0, \Delta)$.

(b) If

$$1 - \exp(\lambda(X_\tau, \dot{r}_\tau, \tau)\Delta) > u \tag{4.3.61}$$

where $u \sim U(0, 1)$, draw $\dot{z}_\tau \sim \mathrm{N}(\mu_z, \sigma_z^2)$ and set

$$X_{\tau+\Delta} = X_{\tau+\Delta} + \dot{z}_\tau. \tag{4.3.62}$$

(4) Set

$$\dot{r}_{\tau+\Delta} = \begin{cases} \lambda_1 & \text{w.p. } \frac{\beta_2}{\beta_1+\beta_2} - \frac{\beta_2}{\beta_1+\beta_2} e^{-(\beta_1+\beta_2)\Delta} \text{ if } \dot{r}_\tau = \lambda_2, \\ \lambda_2 & \text{w.p. } \frac{\beta_1}{\beta_1+\beta_2} - \frac{\beta_1}{\beta_1+\beta_2} e^{-(\beta_1+\beta_2)\Delta} \text{ if } \dot{r}_\tau = \lambda_1, \\ \dot{r}_\tau & \text{otherwise.} \end{cases} \tag{4.3.63}$$

(4) Set $\tau = \tau + \Delta$ and if $\tau \leq t$ go to step (2).

Let $\boldsymbol{\theta} = \{\kappa_x, \mu_x, \sigma_x, \mu_z, \sigma_z\} = \{2, 5, 1, 1, 0.25\}$, $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2\} = \{1, 3\}$ and $\boldsymbol{\beta} = \{\beta_1, \beta_2\} = \{0.25, 1\}$ and fix the initial values of the process to $X_0 = 4$ and $\dot{r}_0 = \lambda_1$. Figure 4.3.3 compares log-scaled simulated moments of Equation 4.3.54 to those calculated from solving the moment equations numerically. The simulated moments are calculated using 10 000 trajectories and a step-size of 0.001 time units. As expected, the simulated moments and moment equations match closely with the mean trajectory of the process (corresponding to $m_1(t)$) being unaffected by the periodicity of the volatility coefficient.

Using the appropriate moment-cumulant relations we again employ the saddle-point approximation in order to approximate the transitional density. Figure 4.3.4 compares the resulting approximation to the frequency distribution calculated from the simulated trajectories at various points in time. As with the moment equations, the transition density approximation appears to accurately replicate the transition density at the indicated time epochs. The effect of the periodic volatility can clearly be seen in the oscillating shape of the transition density surface.

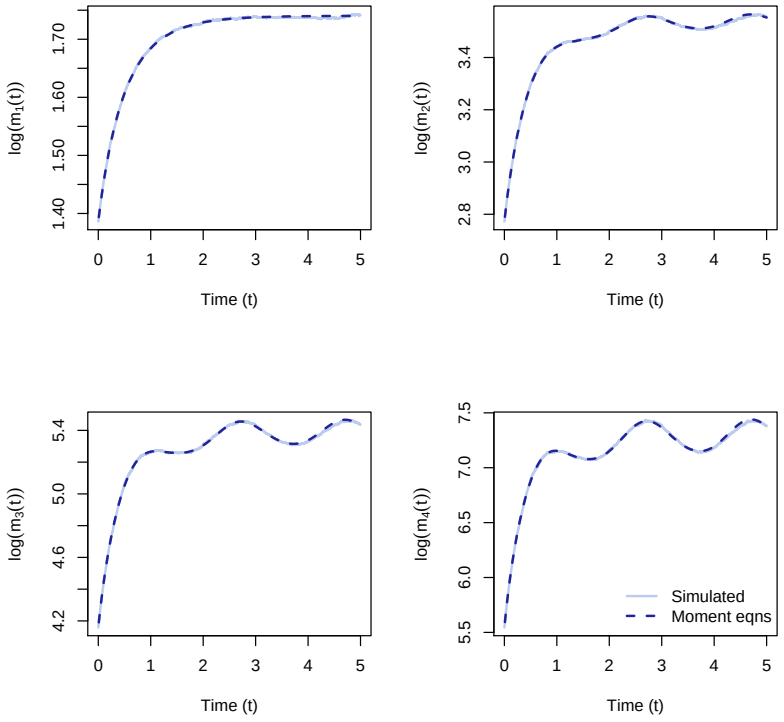


FIGURE 4.3.3: Log-scaled simulated moment trajectories (solid, light blue) and approximate moment equations (dashed, dark blue) for the first four non-central moments of Equation 4.3.54 over time. R code: Supplementary materials, Section 4.4.

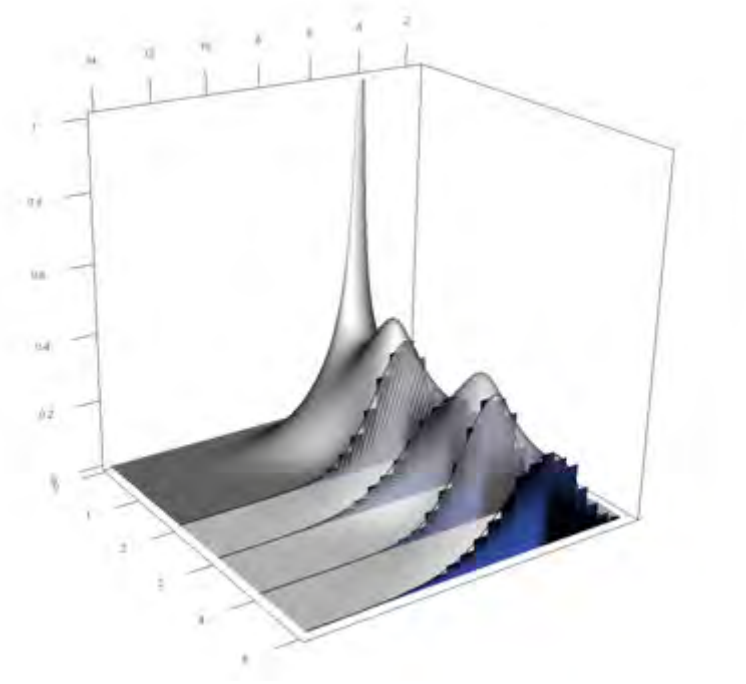


FIGURE 4.3.4: Approximate transition density (gray/lightgray) of Equation 4.3.54 and simulated transition density (light blue - dark blue) at times $t = 2$, $t = 3$, $t = 4$ and $t = 5$. The transition density surface is highlighted in black for each epoch of the comparison. R code: Supplementary materials, Section 4.4.

By repeating the calculation of the transition density approximation for a different set of initial conditions – where instead of starting in the low jump frequency state, we let the process start in the high jump frequency state – we can visualise the effect of the stochastic intensity. Figure 4.3.5 compares the approximate transition densities for the two initial states of the intensity process. Under the high intensity regime the transition density is significantly more skewed than under the low intensity regime. This follows intuitively since, although the jump distribution remains fixed regardless of the state of the intensity process, under the assumed parameter set jumps will typically assume positive values. Consequently, if jumps occur more frequently the process is likely to have propagated further from its initial state at a given time than under the low intensity regime. Noting that, despite there being a non-zero probability of the intensity process switching back to the low intensity state, on average the intensity is expected to be higher for the duration of the transition horizon than compared to starting from the

low intensity state. Indeed, this can be verified by comparing $h(t, \boldsymbol{\lambda}, \boldsymbol{\beta})$ for both initial states of the intensity process under the assumed parameter set.

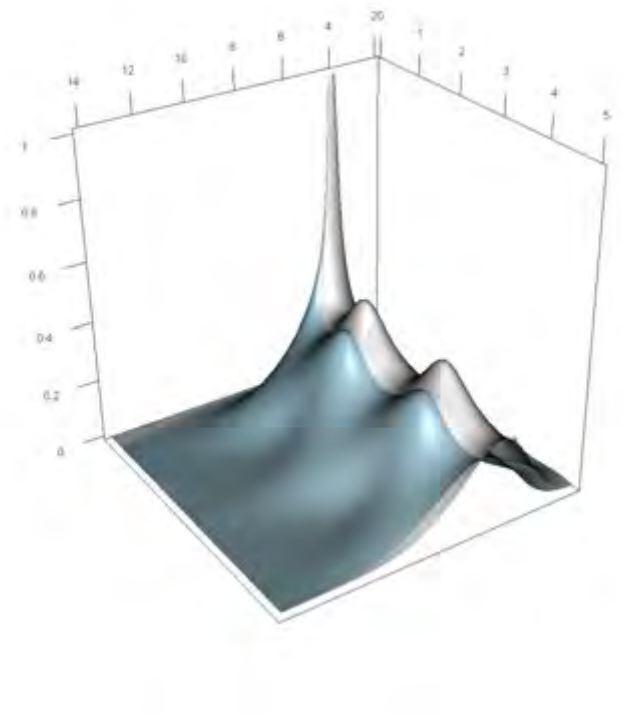


FIGURE 4.3.5: Approximate transition density of Equation 4.3.54 assuming different initial states for the intensity process. Under the high intensity regime (light blue) the transitional density is more skewed, reflecting the influence of a higher jump frequency, even though the distribution of the jumps are identical. R code: Supplementary materials, Section 4.4.

4.3.3.3 Coupled, non-linear bivariate process with time-inhomogeneous jump mechanism.

Although the methodology makes it possible to analyse quite complicated scalar jump diffusion models, it applies to equally complex multivariate diffusions. Indeed, although the algebra involved in calculating the moment equations of multivariate jump diffusions are more cumbersome, the procedure for analysing such processes does not differ much from the scalar case. Consider for example a

jump diffusion process with dynamics given by the SDE:

$$\begin{aligned} dX_t &= (\alpha_x(\beta_x - X_t) + \nu_x Y_t)dt + \sigma_x\sqrt{X_t Y_t}dB_t^{(1)} + dP_t^{(1)} \\ dY_t &= (\alpha_y(\beta_y - Y_t) + \nu_y X_t)dt + \sigma_y\sqrt{X_t Y_t}dB_t^{(2)} + dP_t^{(2)} \end{aligned} \tag{4.3.64}$$

with

$$\begin{aligned} dP_t^{(1)} &= \dot{z}_t^{(11)} dN_t^{(1)} \\ dP_t^{(2)} &= \dot{z}_t^{(21)} dN_t^{(1)} \end{aligned} \tag{4.3.65}$$

and $\lambda_1(X_t, Y_t, t) = \kappa(1 + \sin(3\pi t))$. Furthermore, let:

$$\begin{bmatrix} \dot{z}_t^{(11)} \\ \dot{z}_t^{(21)} \end{bmatrix} \sim \text{MVN}\left(\begin{bmatrix} \mu_{11}(1 + \sin(2\pi t)) \\ \mu_{21}(1 + \sin(2\pi t)) \end{bmatrix}, \begin{bmatrix} \sigma_{11}^2 & 0 \\ 0 & \sigma_{21}^2 \end{bmatrix} \right), \tag{4.3.66}$$

where $\text{MVN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the multivariate Normal distribution with location vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Note here that the jump mechanism is driven by a single jump process, i.e., $\lambda_2(X_t, Y_t, t)$ and the second column of the jump matrix are zero under Equation 4.3.29.

As before, by deriving the moment equations of Equation 4.3.64 from Equation 4.3.35 and solving the resulting system of ODEs numerically, we may evaluate the trajectories of the moments of a given jump diffusion model. For purposes of the experiment, we set $\{\alpha_x, \beta_x, \nu_x, \sigma_x, \alpha_y, \beta_y, \nu_y, \sigma_y, \kappa, \mu_{11}, \sigma_{11}, , \mu_{21}, \sigma_{21}\} = \{0.5, 5, -0.1, 0.15, 0.4, 6, -0.1, 0.1, 1, 0.75, 0.5, 0.75, 0.5\}$. Figure 4.3.6 compares a numerical solution of the moment equations (on log scale) to simulated trajectories for the corresponding moments of Equation 4.3.64 using 20 000 simulated trajectories calculated under a modified Euler-Murayama scheme with a constant step size of 0.001 time units.

Subsequently, by plugging these moment trajectories into a suitable surrogate density we may approximate the evolution of the transitional density over time. Figure 4.3.7 illustrates the evolution of the marginal transition densities of Equation 4.3.64 over time. For comparison we have superimposed histograms of simulated trajectories of Equation 4.3.64 at the indicated time points. The marginal transition densities are calculated by plugging the trajectories of $\{m_{10}(t), m_{20}(t), m_{30}(t), m_{40}(t)\}$ and $\{m_{01}(t), m_{02}(t), m_{03}(t), m_{04}(t)\}$ (for the X_t and Y_t dimension respectively) into a 4-th order univariate saddlepoint approximation. In order to evaluate the joint density, we may similarly employ the bivariate saddlepoint approximation or the bivariate Edgeworth series (see Appendix D.5) as a suitable surrogate density. For brevity, we defer further discussion of the joint density to Section 4.3.4. The transition density approximation under the moment equations appears to match the simulated marginals very closely. Note that the oscillatory behaviour of the

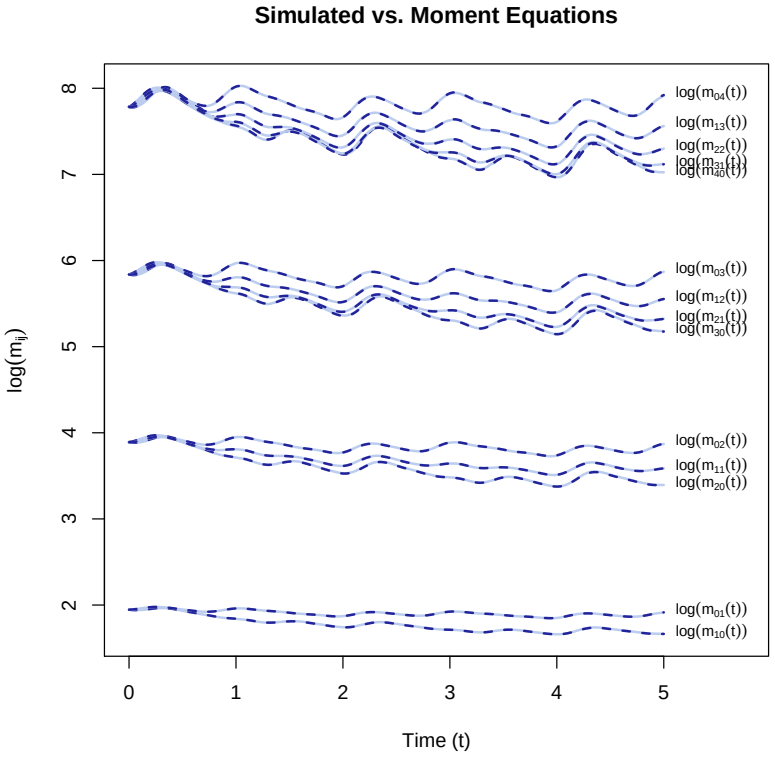


FIGURE 4.3.6: Log-scaled simulated moment trajectories (solid, light blue) and approximate moment equations (dashed, dark blue) for the fourth-order non-central moments of Equation 4.3.64 over time. R code: Supplementary materials, Section 4.5.

transition density is entirely due to temporal dependence of the distribution of the jump variables of the diffusion, illustrating that the presence of the jump mechanism significantly affects the trajectory of process.

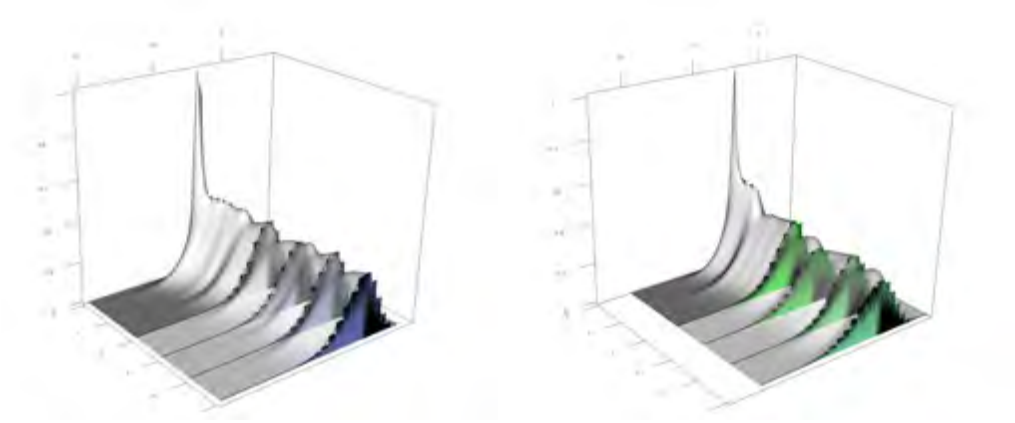


FIGURE 4.3.7: X_t (left) and Y_t (right) marginal transition densities over time. Histograms of simulated trajectories (light blue - dark blue for X_t and green-blue for Y_t) indicated at times $t = 1, 2, 3$ and 4 for comparison. R code: Supplementary materials, Section 4.5.

4.3.4 Short time scale dynamics and the mixture factorization

As demonstrated in the preceding examples, we may use the methodology of sections 4.3.1 and 4.3.2 to derive moment equations for quite complicated jump diffusion processes. By evaluating these equations we may subsequently approximate a solution to the Kolmogorov forward equation by plugging the resulting moment trajectories into an appropriate surrogate density. Assuming that a sufficient amount of information about the density function can be captured within a finite number of moments, we can *tune* the quality of the density approximation by varying the number of moments used in the calculation. For example, in the case of a scalar jump diffusion, using a Normal distribution we may approximate the transition density of a model process by using its first two moments. Using the Gamma density, we may use the first three moments in order to calculate a density approximation that accounts for skew. Using a saddlepoint approximation, we may include an arbitrary number of moments in order to account for the higher order properties in the probabilistic evolution of a model process. Unfortunately, in the context of jump diffusions, the process of constructing a valid density approximation via a surrogate density is not as straightforward as applying a high order density approximation: Due to the presence of the jump mechanism, the transition density of a jump diffusion processes is often multimodal on short transition horizons – a feature that is difficult to mimic using a typical surrogate density structure regardless of how many higher order moments are included in the density approximation. In principle, this follows from the discrepancy that

exists between the diffusion and jump dynamics of the process. For example, on sufficiently short time scales, the paths of a jump-free diffusion are approximately normally distributed and are unlikely to deviate very far from the point at which the process originated. However, by adding jumps to the trajectory of the process, should any jumps occur within the applicable transition horizon, the distribution of the process paths may differ vastly from that of the jump-free diffusion process. For example, consider the SDE:

$$dX_t = a(b - X_t)dt + cdB_t. \tag{4.3.67}$$

Given X_s, X_t for $t > s$ is Normally distributed with mean $X_s e^{-a(t-s)} + b(1 - e^{-a(t-s)})$ and variance $\frac{c^2}{2a}(1 - e^{-2a(t-s)})$. However, even for a simple jump mechanism, say:

$$dY_t = dX_t + \dot{z}_t dN_t \tag{4.3.68}$$

where $N_t - N_s \sim \text{Poi}(\lambda(t-s))$ and $\dot{z}_t \sim \text{N}(\mu_z, \sigma_z^2)$, the transition density of Y_t is not known. Qualitatively, it makes sense that the evolution of the transition density can be difficult to express analytically since at an infinitesimal level the dynamics of the process is governed by sources of randomness which differ vastly in nature. For example, where the diffusion part of the process is driven by small incremental changes in the Brownian motion, the jump mechanism leads to large instantaneous changes in the state of the process. In order to reflect this dichotomy within a surrogate density structure, we need to systematically account for the contrast between the behaviour of the diffusion and jump dynamics of the process. For these purposes, we decompose the transition density of a jump diffusion process as follows: Consider for the time being a scalar jump diffusion process with transition density $f_J(X_t|X_s)$, and let $f_D(X_t|X_s)$ denote the transition density of a jump-free process with identical diffusion dynamics to that of the jump process X_t. Then let:

$$f_J(X_t|X_s) = \alpha f_D(X_t|X_s) + (1 - \alpha) f_E(X_t|X_s) \tag{4.3.69}$$

for $\alpha \in [0, 1]$ where $f_E(X_t|X_s)$ denotes an *excess* distribution. Equation 4.3.69 thus decomposes the jump diffusion transition density in terms of its jump-free counterpart and an excess distribution that accounts for the effect of the jump mechanism. Using this, we can decompose the moment equations in similar fashion:

$$\begin{aligned} \int_S x^i f_J(x|X_s)dx &= \int_S x^i \big(\alpha f_D(x|X_s) + (1-\alpha) f_E(x|X_s)\big)dx \\ m_i^J(t) &= \alpha m_i^D(t) + (1-\alpha) m_i^E(t), \end{aligned} \tag{4.3.70}$$

where $\{m_i^J(t) : i = 0, 1, 2, \ldots\}$ corresponds to the non-central moments of the jump diffusion, $\{m_i^D(t) : i = 0, 1, 2, \ldots\}$ corresponds to the jump-free diffusion's moments and $\{m_i^E(t) : i = 0, 1, 2, \ldots\}$ and α are to be determined. This is similar to the technique used by Ball and Torous (1985) whereby the arrival process is assumed to follow a 'Bernoulli jump process', in which case the factorization would be exact since exactly one jump may occur or not. The distinction here is that, although the transitional density is factorized in terms of a jump-free distribution and one that accounts for the presence of a jump mechanism, we do not constrain the excess distribution to account for a single jump event alone but rather for the effect of any non-zero number of jumps.

As is, Equation 4.3.70 gives an under-determined system of equations that relate the moments of the jump diffusion process to its jump-free counterpart. In principle, α and $m_i^E(t)$ can be any numbers that satisfy Equation 4.3.70 subject to the constraint that $\alpha \in [0, 1]$. However, it can be justified that α should be chosen as the probability that no jumps occur up to and including time t, in which case the decomposition that directly relates the jump-free diffusion dynamics to that of the jump diffusion assumes the form:

$$f_J(X_t|X_s) = P(N_t - N_s = 0) f_D(X_t|X_s) + P(N_t - N_s > 0) f_E(X_t|X_s). \quad (4.3.71)$$

By decomposing the transition density of the jump diffusion in this way, we can formulate accurate mixture density approximations on both short and long time scales. For example, by solving the system 4.3.70 with $\alpha = P(N_t - N_s = 0)$ and calculating the moment-excess, $m_i^E(t)$ for each i, we can approximate $f_J(X_t|X_s)$ as a mixture of $f_D(X_t|X_s)$ with moments $m_i^D(t)$ and $f_E(X_t|X_s)$ with moments $m_i^E(t)$, where $f_D(.)$ and $f_E(.)$ may in turn be approximated by an appropriate surrogate density.

Although the factorization is theoretically quite simple, the mechanics of the factorization is not always trivial: In order to decompose the transition density in this way we are required to evaluate the probability $P(N_t - N_s = 0)$. When the intensity coefficient $\lambda(X_t, t)$ is independent of the state of the jump diffusion (i.e., $\lambda(X_t, t) = \lambda(t)$) the corresponding probability can easily be evaluated as:

$$P(N_t - N_s = 0) = e^{-\int_s^t \lambda(u) du}. \quad (4.3.72)$$

However, when the jump intensity depends directly on the state of the diffusion, we are required to evaluate the jump arrival probability as some function of the process X_t. For these purposes we evaluate the zero-jump probability as the

solution to the differential equation:

$$\frac{\partial}{\partial t}\log(P(N_t - N_s = 0)) = -\int_S \lambda(x,t) f_D(x|X_s)dx. \tag{4.3.73}$$

Note that we integrate over the jump-free diffusion's transitional density since the probability of observing the initial jump up to and including the instant that the first jump occurs evolves independently of the jump process (i.e., the process path is a pure diffusion process conditional on originating from an instant at which the process is a pure diffusion process or an instant after a jump has occurred). Thus, in order to evaluate $P(N_t - N_s = 0)$ we can directly incorporate the moment equations of the jump-free counterpart to the process of interest. For example, given an intensity function of the form:

$$\lambda(X_t, t) = \lambda_0(t) + \lambda_1(t)X_t + \lambda_2(t)X_t^2, \tag{4.3.74}$$

the corresponding probability is to be evaluated as:

$$\frac{\partial}{\partial t}\log(P(N_t - N_s = 0)) = -(\lambda_0(t) + \lambda_1(t)m_1^D(t) + \lambda_2(t)m_2^D(t)). \tag{4.3.75}$$

By factorizing the density in this way, we may contrast the diffusion and jump behaviour of the process in order to create an accurate approximation of the transitional density through a mixture of surrogate densities based on the transitional density of the jump-free distribution and an *excess* distribution that accounts for the behaviour of the jump mechanism.

To illustrate the mechanics of the factorization, consider the scalar jump diffusion:

$$\begin{aligned} dX_t &= 2(5 - X_t)dt + 0.25\sqrt{X_t}dB_t + dP_t \\ dP_t &= \dot{z}_t dN_t \end{aligned} \tag{4.3.76}$$

with $\dot{z}_t \sim \mathrm{N}(1, 0.5^2)$ and $X_0 = 4$. Consider now two alternative intensity processes whereby $\lambda(X_t, t) = 0.5(1+\sin(3\pi t))X_t + 0.1(1+\cos(3\pi t))X_t^2$ and $\lambda(X_t, t) = 0.2X_t$. Using the methodology of Section 4.3.2 we can derive the moment equations of Equation 4.3.76 for the two intensity specifications (see Appendix D.6). By augmenting the jump diffusion's moment equations with those of the jump-free diffusion moments (for which the expressions are nested within the jump diffusion moment equations) and Equation 4.3.73, we can evaluate the moment trajectories for $f_D(X_t|X_s)$ and $f_E(X_t|X_s)$ and calculate the zero-jump probability over arbitrary time horizons. Figure 4.3.8 depicts the evolution of the zero-jump probability over time for the two intensity regimes. From these systems we can calculate the corresponding excess moments and subsequently approximate the

density $f_J(X_t|X_s)$ as a mixture of $f_D(X_t|X_s)$ and $f_E(X_t|X_s)$, which in turn are approximated by suitable surrogate densities.

Figure 4.3.9 depicts the dichotomy of the transition density under the two intensity regimes by superimposing the approximate transition density over histograms calculated using numerous simulated trajectories (in this case, 50 000) on relatively short transition horizons ($[0, 0.02]$ and $[0, 0.1]$ respectively). Here, $f_D(X_t, Xs)$ and $f_E(X_t, Xs)$ are approximated by plugging the resulting moment trajectories into a fourth order truncated scalar saddlepoint approximation. For $\lambda(X_t, t) = 0.5(1 + \sin(3\pi t))X_t + 0.1(1 + \cos(3\pi t))X_t^2$ the effect of the presence of jumps manifests in a heavy right tail very early in the evolution of the transition density with a second mode around 1 unit away from the initial value of the diffusion, reflecting the mean of the jump random variables. For $\lambda(X_t, t) = 0.2X_t$ the effect is somewhat less pronounced. This is not unexpected since as more time has elapsed a greater region of the support could have been reached without any jump events occurring. Figure 4.3.10 depicts the evolution of the transition density for $\lambda(X_t, t) = 0.2X_t$ from inception up to including $t = 0.1$. At inception, the transition density evolves from a point mass and almost instantaneously exhibits dichotomous behaviour. This follows since there is a non-zero probability of a jump occurring immediately after inception. As time progresses the diffuse and jump dynamics 'mix' and the dichotomous behaviour becomes less prominent, leading to a more contiguous transition density.

Although the underlying mathematics of the moment equations is easily extended without much complication to higher dimensions, the process of factorizing the transition density is less trivial beyond the scalar case. For example, in the bivariate case, if the jump mechanism is driven by a single arrival process (i.e., $q = 1$ for Equation 4.2.1) then, following along the lines of Equation 4.3.71, we can factorize the transitional density in terms of the jump-free transitional density and an excess distribution just as for scalar jump diffusions. However, when the jump mechanism is driven by multiple arrival processes, a number of complications arise with respect to the mixture factorization. For example, in the bivariate case where the jump mechanism is driven by two Poisson processes ($q = 2$), one possible factorization is:

$$\begin{aligned} f_J(X_t, Y_t|X_s, Y_s) = P(N_t^{(1)} - N_s^{(1)} = 0, N_t^{(2)} - N_s^{(2)} = 0) f_D(X_t, Y_t|X_s, Y_s) \\ + P(N_t^{(1)} - N_s^{(1)} > 0, N_t^{(2)} - N_s^{(2)} > 0) f_E(X_t, Y_t|X_s, Y_s), \end{aligned} \tag{4.3.77}$$

where $f_J(X_t, Y_t|X_s, Y_s)$ denotes the jump diffusion transition density, $f_D(X_t, Y_t|X_s, Y_s)$ denotes the jump-free diffusion transition density and $f_E(X_t, Y_t|X_s, Y_s)$ again denotes the excess distribution. Although there are

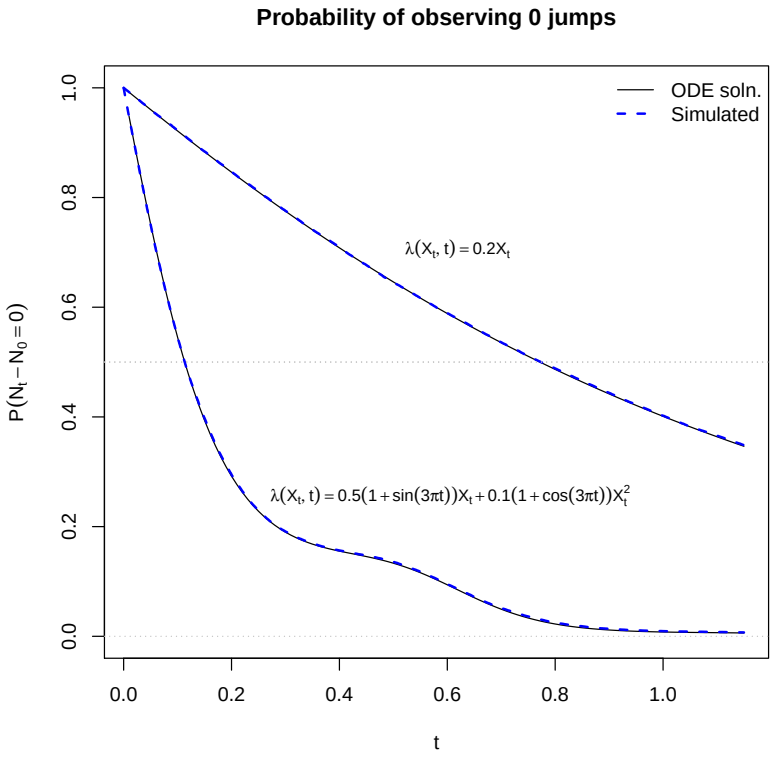


FIGURE 4.3.8: Evolution of the probability of zero jumps occuring up to and including time t. The probability as calculated using the moment equations and Equation 4.3.73 (solid black) match very closely the simulated zero-jump probabilities (dashed blue) for both intensity regimes. R code: Supplementary materials, Section 4.6.

certainly circumstances under which Equation 4.3.77 would produce a valid factorization, caution is required when applying Equation 4.3.77: For jump diffusions with multiple jump sources, the direction of the jump variables play a critical role in the dichotomy of the transitional density on short time scales. For example, for a bivariate diffusion process with a jump mechanism driven by two arrival processes (i.e., $q = 2$), if the jump variables corresponding to the arrival process $N_t^{(1)}$ predict movements in a different direction to the jump variables associated with $N_t^{(2)}$, then the transitional density can have up to four modes over short time scales. Consequently, the excess distribution under the factorization of Equation 4.3.77 will itself be multimodal. This can be seen by considering the arrival of the first one or two jumps within a short transition horizon. Suppose the process is at (X_s, Y_s) then arrivals due to $N_t^{(1)}$ and no arrivals due to $N_t^{(2)}$

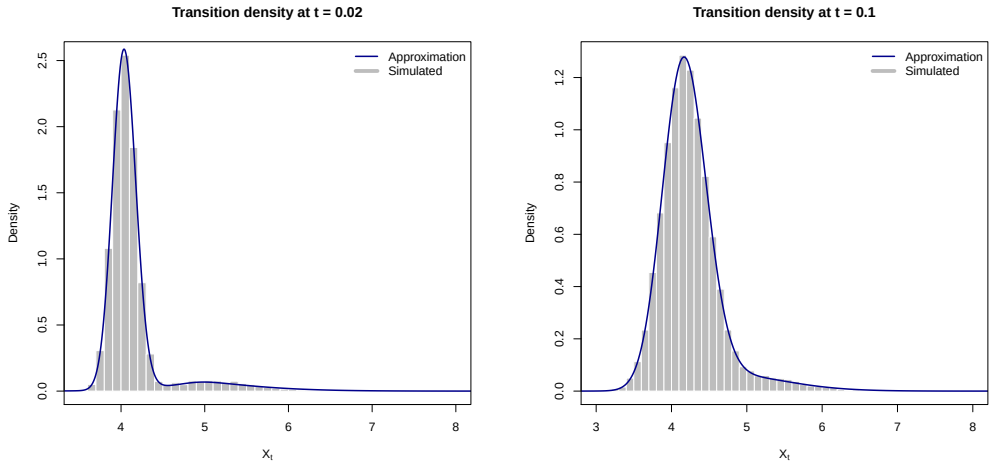


FIGURE 4.3.9: Simulated (histogram) and approximate (blue) transitional density of Equation 4.3.76 for $\lambda(X_t, t) = 0.5(1 + \sin(3\pi t))X_t + 0.1(1 + \cos(3\pi t))X_t^2$ (left) and $\lambda(X_t, t) = 0.2X_t$ (right) over short transition horizons. The approximation is calculated using the mixture factorization. R code: Supplementary materials, Section 4.6.

result in a distributional mode close to the coordinate $(X_s + \dot{z}_{11}, Y_s + \dot{z}_{12})$ (for additive jumps). On the contrary, if all arrivals are due to $N_t^{(2)}$ then another mode appears close to the coordinate $(X_s + \dot{z}_{21}, Y_s + \dot{z}_{22})$. For arrivals due to both $N_t^{(1)}$ and $N_t^{(2)}$ within a short time-lapse, a third mode appears close to the coordinate $(X_s + \dot{z}_{11} + \dot{z}_{21}, Y_s + \dot{z}_{12} + \dot{z}_{22})$. Were one to factorize the distribution with respect to the moments of each of these events, for example:

$$\begin{aligned} m_{ij}^J(t) = &P(N_t^{(1)} - N_s^{(1)} = 0, N_t^{(2)} - N_s^{(2)} = 0) m_{ij}^D(t) \\ &+ P(N_t^{(1)} - N_s^{(1)} > 0, N_t^{(2)} - N_s^{(2)} = 0) m_{ij}^{E_1}(t) \\ &+ P(N_t^{(1)} - N_s^{(1)} = 0, N_t^{(2)} - N_s^{(2)} > 0) m_{ij}^{E_2}(t) \\ &+ P(N_t^{(1)} - N_s^{(1)} > 0, N_t^{(2)} - N_s^{(2)} > 0) m_{ij}^{E_3}(t), \end{aligned} \tag{4.3.78}$$

where E_1, E_2 and E_3 denote three points of excess, one arrives at an underdetermined system. This follows since the elements $m_{ij}^{E_1}(t)$, $m_{ij}^{E_2}(t)$ and $m_{ij}^{E_3}(t)$ cannot be determined using the jump-free and jump diffusion moments alone. Fortunately, we may still use the single mixture factorization when the jump mechanism implies a uni-modal excess distribution, for example, when the rows of the jump matrix share a common mean (i.e., when more than one jump process is present, we require that on average jumps are made to the same location).

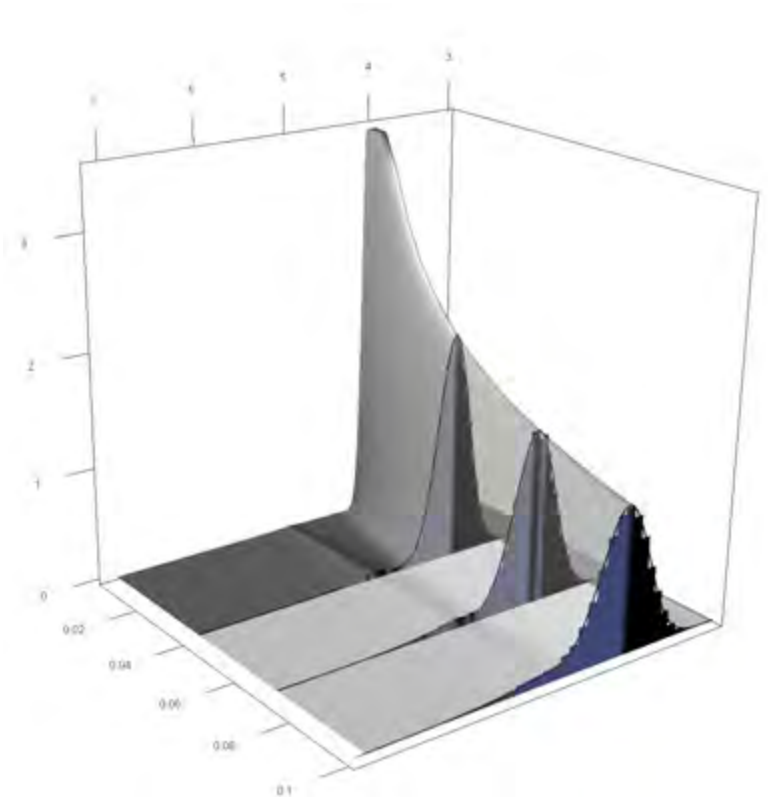


FIGURE 4.3.10: Approximate transition density (gray/lightgray) of Equation 4.3.76 over a short transition horizon with intensity $\lambda(X_t, t) = 0.2X_t$. Here, in order to exagerate the multimodality, we have set $\dot{z} \sim \mathrm{N}(1, 0.1^2)$. Simulated transition density (light blue - dark blue) is indicated for comparison. On short time-scales the transition density is clearly bimodal.

To illustrate the mixture factorization as applied to bivariate diffusions, consider a jump process with dynamics given by the SDE:

$$\begin{aligned} dX_t &= (0.5(5 - X_t) - 0.1Y_t)dt + 0.4\sqrt{X_t}dB_t^{(1)} + dP_t^{(1)} \\ dY_t &= (0.4(6 - Y_t) - 0.1X_t)dt + 0.3\sqrt{Y_t}dB_t^{(2)} + dP_t^{(2)} \end{aligned} \tag{4.3.79}$$

subject to the initial condition $(X_0, Y_0) = (7, 7)$, with

$$\begin{aligned} dP_t^{(1)} &= \dot{z}_{11}dN_t^{(1)} \\ dP_t^{(2)} &= \dot{z}_{21}dN_t^{(1)} \end{aligned} \tag{4.3.80}$$

and $N_t^{(1)} - N_s^{(1)} \sim \text{Poi}(1 \times (t-s))$. Furthermore, let:

$$\begin{bmatrix} \dot{z}_{12} \\ \dot{z}_{22} \end{bmatrix} \sim \text{MVN}\left(\begin{bmatrix} 0.75(1+\sin(2\pi t)) \\ 0.75(1+\sin(2\pi t)) \end{bmatrix}, \begin{bmatrix} 0.75^2(1+0.8\sin(2\pi t))^2 & 0 \\ 0 & 0.75^2(1+0.8\sin(2\pi t))^2 \end{bmatrix} \right), \tag{4.3.81}$$

where $\text{MVN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the multivariate Normal distribution with location vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

Using Equation 4.3.35 we can derive the moment equations for Equation 4.3.79 and subsequently solve for the moment trajectories of the process over time (see Appendix D.7 for the $d = 4$-th order moment equations). As in the case of Equation 4.3.76, we approximate the transition density by approximating the jump-free and excess distributions with a suitable surrogate density. For these purposes, we make use of the bivariate saddle point approximation. Figure 4.3.11 illustrates the evolution of the transition density of Equation 4.3.79 over time. For illustrative purposes, we have superimposed the coordinates of 200 simulated trajectories labelled according to which trajectories have undergone a jump innovation or not up to and including the indicated time. In the initial phases of the evolution of the transition density, it is clear that the dichotomy of the transition density is caused by the contrast between the diffuse and jump dynamics, with the jump dynamics resulting in a second mode some distance away from the initial values of the process. Using the simulated trajectories as a guide, one can clearly see that the second mode and its surrounding area is populated predominantly by trajectories that have been subjected to jumps, whilst the area around the primary mode of the transition density is populated mostly by paths which have not yet undergone jumps. As time progresses, the contrast becomes less apparent as the jump and diffuse characteristics 'mix' or 'aggregate', resulting in a more homogeneous distribution.

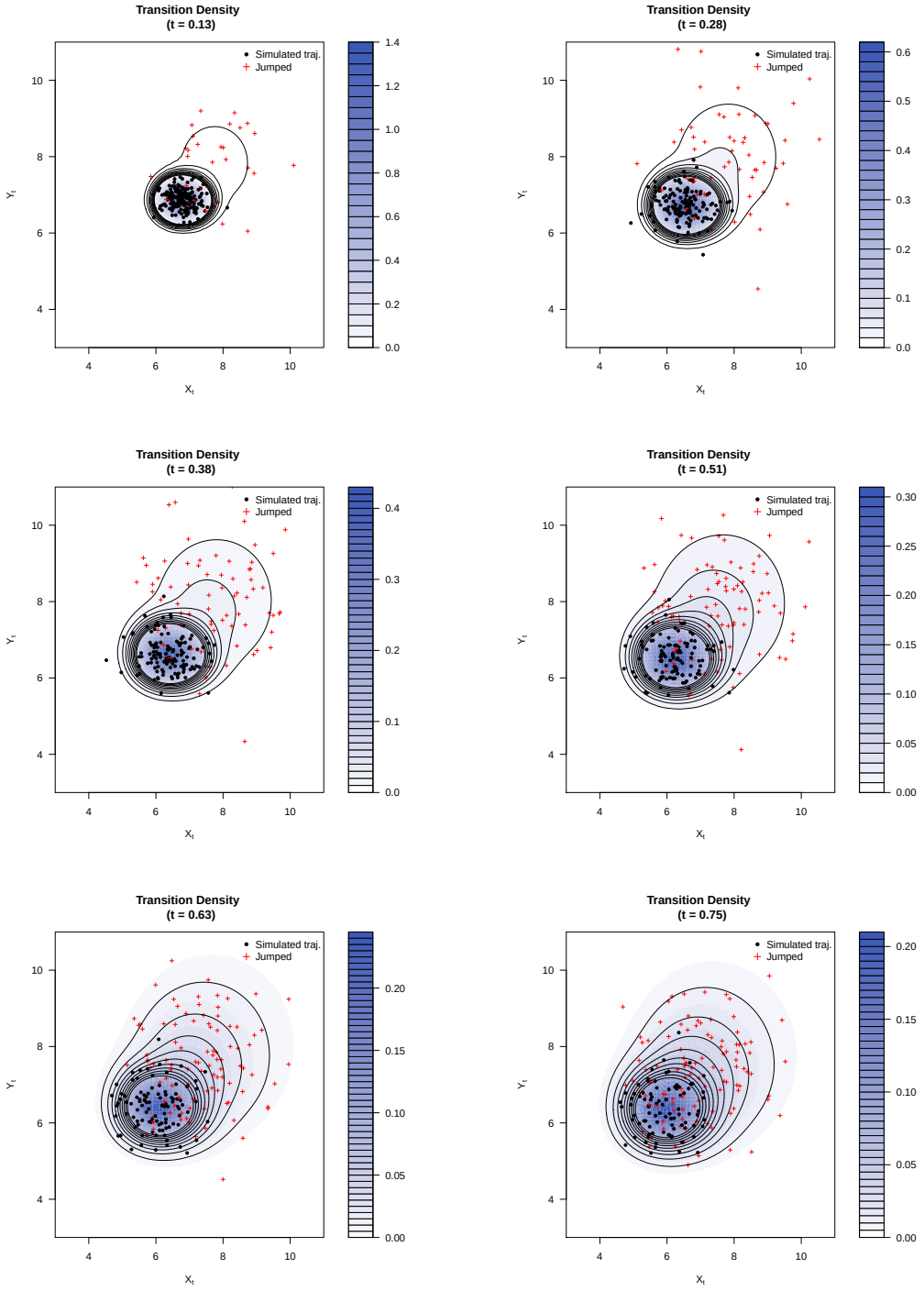


FIGURE 4.3.11: Evolution of the transition density under the factorization of Equation 4.3.71. Snapshots of the coordinates of simulated trajectories are superimposed and labelled according to which trajectories have undergone jump innovations (red cross) and those which have not (black dots). R code: Supplementary materials, Section 4.7.

We conclude this section by remarking that the notion of the transition density exhibiting 'dichotomous' behaviour is logically distinct from being multimodal: It is entirely possible for a jump diffusion to have a uni-modal transition density just as it is also entirely possible for a jump-free diffusion to have a multimodal transitional density. When referring to dichotomous behaviour we take it to mean that the behaviour of the density is explained by two distinct sources of randomness that interact to generate the observed dynamics.

4.4 Benchmarking and comparison to existing methods

As mentioned previously, only a few jump diffusion models are known to have analytically tractable transitional densities. As such, one typically has to resort to simulation techniques in order to verify the quality of a given approximation scheme. In order to assess the performance of the scheme derived in this thesis, we compare our methodology to existing techniques from the literature and where possible make reference to 'true' expressions for the transitional density.

4.4.1 Brownian motion with drift: Short time approximations of the transitional density

Consider a jump diffusion with dynamics given by the SDE:

$$\begin{aligned} dX_t &= \mu_x dt + \sigma_x dB_t + dP_t \\ dP_t &= \dot{z}_t dN_t \end{aligned} \tag{4.4.1}$$

where $\dot{z}_t \sim \mathrm{N}(\mu_z, \sigma_z^2)$, $N_t - N_s \sim \mathrm{Poi}(\lambda(t-s))$. Equation 4.4.1 constitutes a Brownian motion with drift and additive, Normal distributed jumps. In this case it is possible to derive an exact expression for the transitional density. That is, the true distribution of X_t at time t, given that it originated in state X_s at time $s < t$ is given by (Yu, 2007):

$$\begin{aligned} f_{true}(X_t|X_s) &= \frac{e^{-\lambda\tau}}{\sqrt{2\pi\sigma_x^2\tau}} \exp\left(- \frac{(X_t - X_s - \mu_x\tau)^2}{2\sigma_x^2\tau} \right) \\ &+ \sum_{i=1}^{\infty} \frac{e^{-\lambda\tau}\lambda^i\tau^j}{i!} \frac{1}{\sqrt{2\pi(\sigma_x^2\tau + i\sigma_z^2)}} \exp\left(- \frac{(X_t - X_s - \mu_x\tau - i\mu_z)^2}{2(\sigma_x^2\tau + i\sigma_z^2)} \right). \end{aligned} \tag{4.4.2}$$

where $\tau = t - s$. From the structure of Equation 4.4.2, it is easy to see that the true transitional density is constructed from a sequence of jump probabilities

and the distribution of the Brownian motion given that the associated number of jumps have occurred. Despite having an exact expression for the transition density we cannot evaluate the infinite sequence of terms in practice. Instead, for purposes the comparison over short transition horizons, it suffices to truncate the true expression and evaluate the transition density using only the first few terms of Equation 4.4.2. This follows since the contribution of each term to the true density decreases exponentially as the index of the terms in the summation increases. We thus denote the truncated true distribution by $f^{(m)}_{true}(X_t|X_s)$, where m denotes the number of terms (counting from zero) included in the sequence, i.e.:

$$\begin{aligned} f^{(m)}_{true}(X_t|X_s) &= \frac{e^{-\lambda\tau}}{\sqrt{2\pi\sigma_x^2\tau}} \exp\left(- \frac{(X_t - X_s - \mu_x\tau)^2}{2\sigma_x^2\tau} \right) \\ &+ \sum_{i=1}^{m} \frac{e^{-\lambda\tau}\lambda^j\tau^j}{j!} \frac{1}{\sqrt{2\pi(\sigma_x^2\tau + j\sigma_z^2)}} \exp\left(- \frac{(X_t - X_s - \mu_x\tau - j\mu_z)^2}{2(\sigma_x^2\tau + j\sigma_z^2)} \right). \end{aligned} \tag{4.4.3}$$

By choosing a sufficiently large value for m we may calculate a near exact approximation to the true transitional density. Subsequently, we can assess the quality of alternative approximation schemes by benchmarking them against Equation 4.4.3.

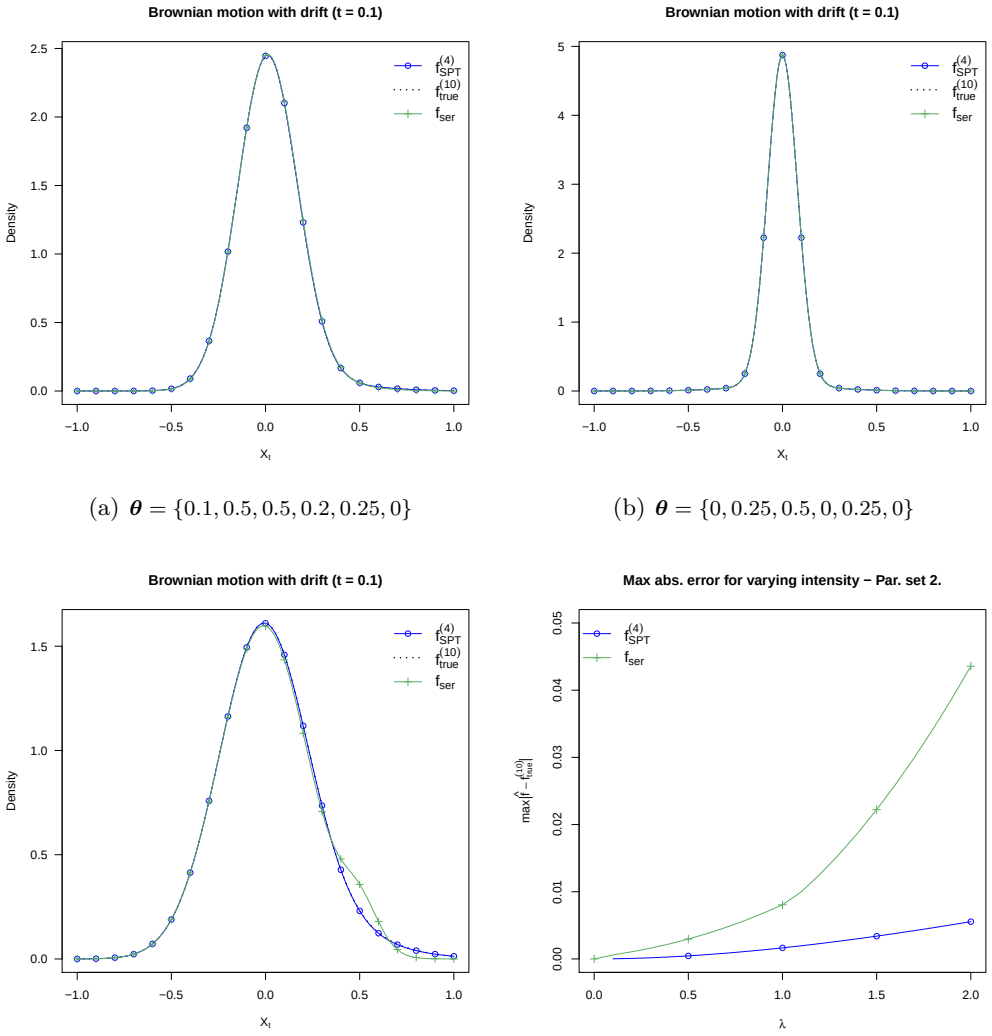
Yu (2007) develops an excellent scheme for deriving closed-form approximations to the transitional density under quite general assumptions about the coefficients of the jump diffusion. For example, using this scheme it is theoretically possible to calculate closed-form approximations for non-linear jump diffusions with state-dependent intensity. The scheme calculates small-time approximations to the transition density by using a series expansion in order to approximate a solution to the Kolmogorov forward and backward equations. The scheme operates in a similar fashion to the Hermite-series methodology developed by Aït-Sahalia (2002), although the order properties of the scheme differs somewhat from that of the Hermite-series methodology. Using this, it is possible to derive a closed-form short-time approximation to the transitional density of Equation 4.4.1. For example, applying the first order approximation to Equation 4.4.1 results in the expression (Yu, 2007):

$$\begin{aligned} &f_{ser}(X_t|X_s) = \\ &\frac{1}{\sqrt{2\pi\sigma_x^2\tau}} \exp\left(- \frac{(X_t - X_s)^2}{2\sigma_x^2\tau} + \frac{\mu_x(X_t - X_s)}{\sigma_x^2} \right) \left(1 - (\frac{\mu_x^2}{2\sigma_x^2} + \lambda)\tau \right) \\ &+ \frac{\lambda\tau}{\sqrt{2\pi\sigma_z^2}} \exp\left(- \frac{(X_t - X_s - \mu_z)^2}{2\sigma_z^2} \right). \end{aligned} \tag{4.4.4}$$

The author notes that for Equation 4.4.1 the mechanism of the series approximation is to incorporate information about the jump mechanism of the process by approximating the first few terms of the true transitional density through terms of a Taylor series expansion. Comparing Equation 4.4.4 to the exact transitional density, it is easy to see that for a fixed approximation order, the quality of a given approximation will be dictated for the most part by the parameter λ and the size of the transition horizon. For example, under Equation 4.4.4, if the transition horizon exceeds $(\mu_x^2/(2\sigma_x^2) + \lambda)^{-1}$ units of time, the approximation may assume negative values. However, since the methodology is developed under the premise that the transition horizon is relatively short (in which case it would be unlikely that this threshold is ever exceeded), the main concern with regard to the quality of the approximation revolves around the magnitude of λ.

In order to assess the performance of the scheme derived in this thesis, we approximate the transitional density of Equation 4.4.1 using the mixture factorization in conjunction with a fourth-order saddlepoint approximation, which we will denote $f_{SPT}^{(4)}(X_t|X_s)$, and compare the resulting approximation to that of Equation 4.4.4 for a number of experimental parameter sets around a common intensity parameter. The approximations can then be compared to the *true* transitional density by calculating $f_{true}^{(m)}(X_t|X_s)$ for large m. Figure 4.4.1 compares $f_{ser}(X_t|X_s)$ and $f_{SPT}^{(4)}(X_t|X_s)$ to $f_{true}^{(10)}(X_t|X_s)$ for three different parameter sets on a short transition horizon, $\tau = 0.1$, corresponding to an $\approx 4.8\%$ probability of observing at least one jump. Additionally, we calculate the maximum absolute deviation from the true density for each approximation scheme whilst increasing λ from 0 and keeping the remaining parameters fixed. Since $f_{SPT}^{(4)}(X_t|X_s)$ is exact and $f_{ser}(X_t|X_s)$ is nearly exact at $\lambda = 0$, this will give an indication of the rate of error propagation for each approximation as the jump intensity increases.

Although both $f_{ser}(X_t|X_s)$ and $f_{SPT}^{(4)}(X_t|X_s)$ produce good approximations when compared to the true transition density, $f_{ser}(X_t|X_s)$ seems to fair less well for the parameter set used in Figure 4.4.1(c) where the mass of the jump distribution is centred relatively far from the origin ($\mu_z = 0.5, \sigma_z = 0.1$). Based on the structure of the series approximation this may be attributed to the fact that the effect of the jump mechanism is accounted for by the term $\lambda\tau/\sqrt{2\pi\sigma_z^2}\exp(-0.5(X_t - X_s - \mu_z)^2/\sigma_z^2)$. Compared to $f_{true}^{(1)}(X_t|X_s)$ (the function which is approximated by $f_{ser}(X_t|X_s)$), for which the corresponding second term reads $\lambda\tau/\sqrt{2\pi(\sigma_z^2\tau + \sigma_z^2)}\exp(-0.5(X_t - X_s - \mu_x\tau - \mu_z)^2/(\sigma_z^2\tau + \sigma_z^2))$, the offset can likely be ascribed to the difference between the scale and location parameters of the approximation and those of the first order terms of the true density. Using the parameter set from Figure 4.4.1(b), but varying the intensity parameter from $\lambda = 0$ to $\lambda = 2$, it can be seen that $f_{SPT}^{(4)}(X_t|X_s)$ is less sensitive than $f_{ser}(X_t|X_s)$



(a) $\boldsymbol{\theta} = \{0.1, 0.5, 0.5, 0.2, 0.25, 0\}$

(b) $\boldsymbol{\theta} = \{0, 0.25, 0.5, 0, 0.25, 0\}$

(c) $\boldsymbol{\theta} = \{-0.1, 0.75, 0.5, 0.5, 0.1, 0\}$

(d) $\boldsymbol{\theta} = \{0, 0.25, \lambda, 0, 0.25, 0\}$

FIGURE 4.4.1: Transition density approximations for Brownian motion with Normal jumps under various parameter sets ($\boldsymbol{\theta} = \{\mu_x, \sigma_x, \lambda, \mu_z, \sigma_z, X_0\}$) at $\tau = 0.1$. Exact transition densities are indicated by dashed black lines. Panel (d) compares maximum absolute differences between each approximation and the true density for varying λ. R code: Supplementary materials, Section 4.8.

to increases in the jump intensity. Although this is to be expected since the expression $f_{ser}(X_t|X_s)$ is only first order in time.

A key point of interest in the analysis of jump diffusions is the specification of the jump mechanism. Unfortunately, arbitrarily altering the specification of the jump mechanism prevents one from deriving analytical expressions such as Equation 4.4.2 for the true transition density. However, a closer look at Equation 4.4.2 reveals a simple alternative to the series approximations or $f^{(4)}_{SPT}$ for approximating the transition density of a Brownian motion with jumps: Given that the drift and diffusion coefficients do not depend on the state of the process, the dynamics of the diffusion part of the process is not affected by jump occurrences apart from shifting the process by the magnitude of a jump. As such, there exists a degree of separation between the diffusion dynamics and the jump dynamics of the process which manifests in the structure of the transition density function. This can be seen from the terms that make up $f_{true}(X_t|X_s)$: Each term consists of a jump probability, giving the probability of a specified number of jumps occurring by a given time, and a density term which gives the distribution of the process given that said number of jumps have occurred. In the case of Equation 4.4.1 where $\dot{z} \sim \mathrm{N}(\mu_z, \sigma_z^2)$, the latter term is simply the sum of independent Normal random variates, which means that the series simplifies to a weighted sum of Normal distributions. When the jump distribution is not Normal, this simplification does not arise as naturally. However, assuming that the transition horizon is sufficiently short, we may exploit standard convolution techniques in order to calculate an approximate transitional density. Let $f_{BM}(B_t|B_s)$ denote the transition density of a Brownian motion with drift and let $\phi(\dot{z}_t)$ denote the jump variable density. Then, assuming that a jump either occurs or doesn't (in principle this idea can again be traced back to Ball and Torous (1985)), a first order approximation for the transitional density can be derived as:

$$\tilde{f}(X_t|X_s) = (1 - \lambda(X_s, s)\tau) f_{BM}(X_t|X_s) + \lambda(X_s, s)\tau \int_{-\infty}^{\infty} \phi(u) f_{BM}(X_t - u|X_s) du. \tag{4.4.5}$$

As before, this approximation can be thought of as being first order with respect to time in the sense that it only considers the effect of a single jump occurrence. Due to its simple structure, the convolution formula can be used for a variety of specifications for the jump mechanism. For example, consider a jump diffusion of the form of Equation 4.4.1 with Exponential jumps $\dot{z}_t \sim \mathrm{Exp}(\nu_z)$ and constant

jump intensity $\lambda(X_t, t) = \lambda$. For this specification the convolution formula yields:

$$\begin{aligned}
&\tilde{f}(X_t|X_s) = \\
&(1-\lambda\tau)\frac{1}{\sqrt{2\pi\sigma_x^2\tau}}\exp\left(-\frac{(X_t - X_s - \mu_x\tau)^2}{2\sigma_x^2\tau}\right) \\
&+ \lambda\tau\nu_z \exp\left(\frac{(X_t - X_s - \mu\tau - \nu_z\sigma_x^2\tau)^2}{2\sigma_x^2\tau} - \frac{(X_t - X_s - \mu_x\tau)^2}{2\sigma_x^2\tau}\right) \\
&\times \big(1 - \Phi\big(0, X_t - X_s - \mu_x\tau - \nu_z\sigma_x^2\tau, \sigma_x^2\tau\big)\big),
\end{aligned} \tag{4.4.6}$$

where $\Phi(x, \mu, \sigma^2)$ is the cumulative Normal distribution function with mean μ and variance σ^2 evaluated up to x. Similarly, for Laplace distributed jumps, $\dot{z}_t \sim \text{Laplace}(\mu_z, b_z) = (2b_z)^{-1}\exp(-|\dot{z}_t - \mu_z|b_z^{-1})$, we derive the convolution approximation:

$$\begin{aligned}
&\tilde{f}(X_t|X_s) = \\
&(1-\lambda\tau)\frac{1}{\sqrt{2\pi\sigma_x^2\tau}}\exp\left(-\frac{(X_t - X_s - \mu_x\tau)^2}{2\sigma_x^2\tau}\right) \\
&+ \frac{\lambda\tau}{2b_z}\exp\left(\frac{(X_t - X_s - \mu\tau + b_z^{-1}\sigma_x^2\tau)^2}{2\sigma_x^2\tau} - \frac{(X_t - X_s - \mu_x\tau)^2}{2\sigma_x^2\tau} - \frac{\mu_z}{b_z}\right) \\
&\times \Phi\big(\mu_z, X_t - X_s - \mu_x\tau + b_z^{-1}\sigma_x^2\tau, \sigma_x^2\tau\big) \\
&+ \frac{\lambda\tau}{2b_z}\exp\left(\frac{(X_t - X_s - \mu\tau - b_z^{-1}\sigma_x^2\tau)^2}{2\sigma_x^2\tau} - \frac{(X_t - X_s - \mu_x\tau)^2}{2\sigma_x^2\tau} + \frac{\mu_z}{b_z}\right) \\
&\times \big(1 - \Phi\big(\mu_z, X_t - X_s - \mu_x\tau - b_z^{-1}\sigma_x^2\tau, \sigma_x^2\tau\big)\big).
\end{aligned} \tag{4.4.7}$$

Despite modifying the jump distribution, it is still possible to calculate a series approximation to the transition density under the methodology of Yu (2007). In the case of Exponential jumps, the series approximation to the transitional would be

$$\begin{aligned}
&f_{ser}(X_t|X_s) = \\
&\frac{1}{\sqrt{2\pi\sigma_x^2\tau}}\exp\left(-\frac{(X_t - X_s)^2}{2\sigma_x^2\tau} + \frac{\mu_x(X_t - X_s)}{\sigma_x^2}\right)\left(1 - \big(\frac{\mu_x^2}{2\sigma_x^2} + \lambda\big)\tau\right) \\
&+ \lambda\tau\nu_z \exp\big(-\nu_z(X_t - X_s)\big)\mathbb{I}_{X_t > Xs},
\end{aligned} \tag{4.4.8}$$

whilst in the case of Laplace distributed jumps the series approximation evaluates to:

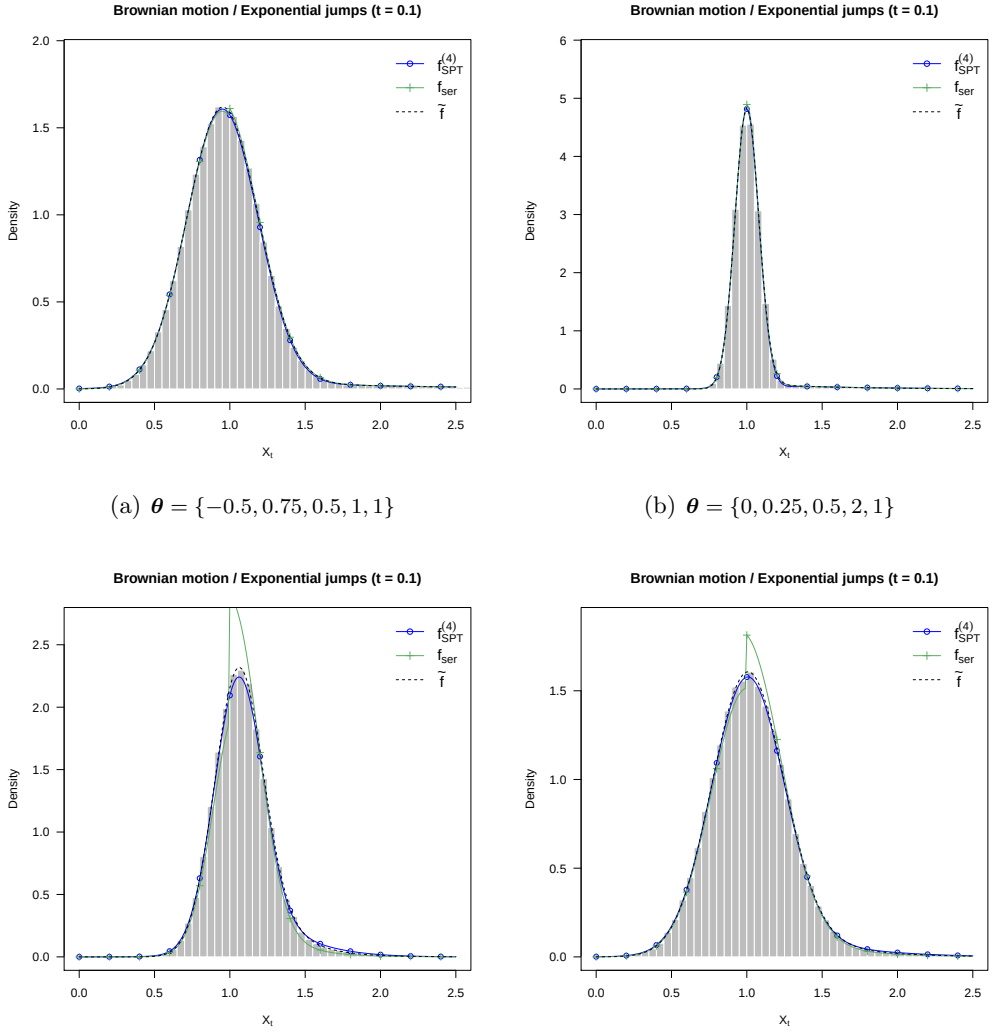
$$f_{ser}(X_t|X_s) = \frac{1}{\sqrt{2\pi\sigma_x^2\tau}}\exp\left(-\frac{(X_t-X_s)^2}{2\sigma_x^2\tau}+\frac{\mu_x(X_t-X_s)}{\sigma_x^2}\right)\left(1-(\frac{\mu_x^2}{2\sigma_x^2}+\lambda)\tau\right) + \frac{\lambda\tau}{2b_z}\exp\left(-\frac{|X_t-X_s-\mu_z|}{b_z}\right). \tag{4.4.9}$$

Figures 4.4.2 and 4.4.3 compare the convolution approximation, the series approximation, and the fourth order saddlepoint approximation to the transitional density under various parameter sets for Exponential and Laplace jump distributions respectively. Since we do not have an exact transition density to use as a benchmark for accuracy, we compare the three approximation schemes to simulated transitional densities. This is achieved by simulating numerous (in this case, 100 000) trajectories of the model process and subsequently evaluating the frequency distribution of the state of the process at the desired transition horizon.

For an Exponential jump distribution, all three approximations fair reasonably well although $f_{ser}(X_t|X_s)$ exhibits a small discontinuous spike at the initial value of the process – a consequence of the exponential density being defined for positive arguments only (i.e., for $X_t > X_s$ the second term in the approximation $f_{ser}(X_t|X_s)$ becomes zero). Depending on the parameter set, this discontinuity may present more or less prominently. Although $f^{(4)}_{SPT}(X_t|X_t)$ produces accurate approximations across all parameter sets, the approximation presents a slightly less peaked distribution than predicted by the simulated transitional density. Interestingly, the convolution approximation $\tilde{f}(X_t|X_t)$ produces accurate approximations across all parameter sets.

In the case of Laplace distributed jumps, $f^{(4)}_{SPT}(X_t|X_t)$ performs well under all experimental parameter sets although slightly underestimating the transitional density when the mass of the jump distribution is not centred at the origin. This is true for both $f_{ser}(X_t|X_t)$ and $\tilde{f}(X_t|X_t)$ albeit much more detrimental in the case of the parameter set used in Figure 4.4.2(c). Indeed it is difficult to identify the specific parameters in this set that cause $f_{ser}(X_t|X_t)$ and $\tilde{f}(X_t|X_t)$ to deviate so much, however it is most likely due to the combination of a relatively high jump probability ($\approx 9.5\%$) and the relatively low dispersion of the jump distribution.

In summary, in the case of jump SDEs with constant coefficients, it is possible to approximate the transitional density in a variety of ways. Compared to the series approximations and convolution techniques our methodology produces



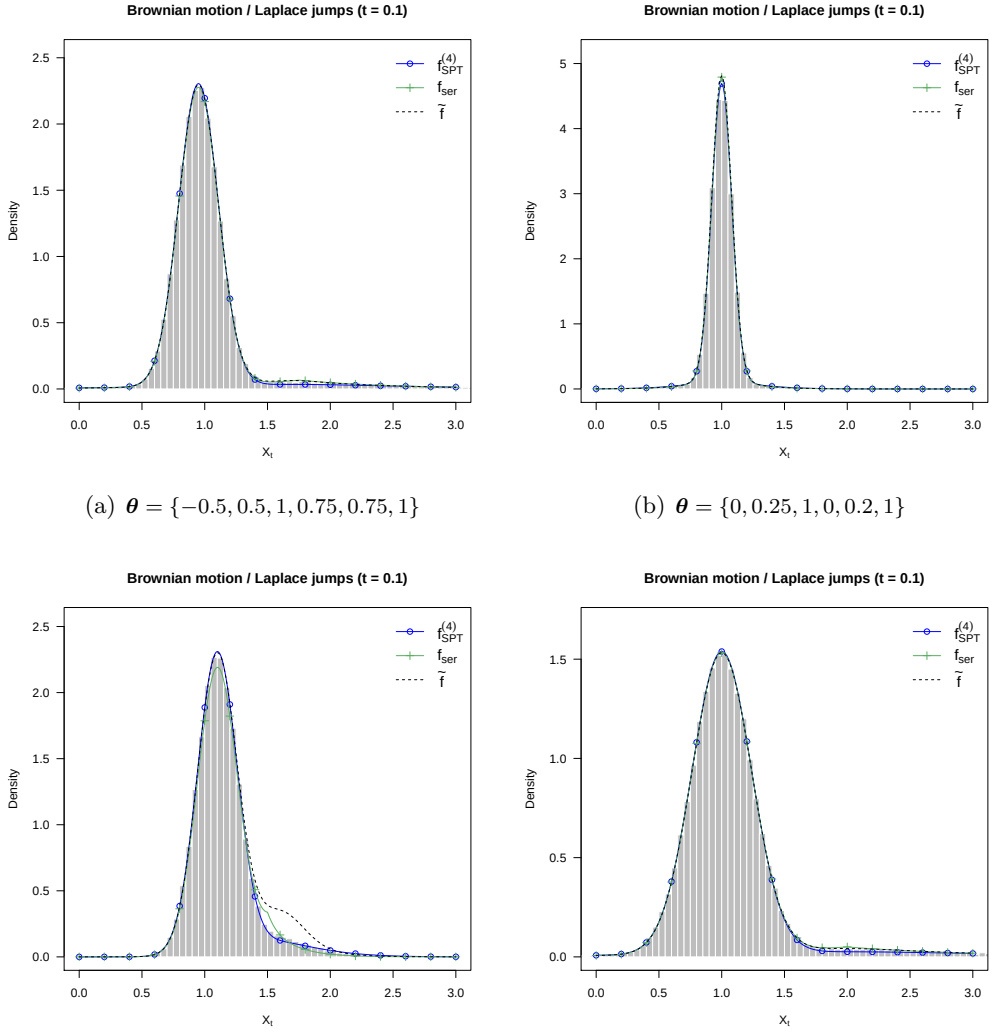
(a) $\boldsymbol{\theta} = \{-0.5, 0.75, 0.5, 1, 1\}$

(b) $\boldsymbol{\theta} = \{0, 0.25, 0.5, 2, 1\}$

(c) $\boldsymbol{\theta} = \{0.5, 0.5, 2, 5, 1\}$

(d) $\boldsymbol{\theta} = \{0, 0.75, 1, 3, 1\}$

FIGURE 4.4.2: Transition density approximations for Brownian motion with Exponential jumps under various parameter sets ($\boldsymbol{\theta} = \{\mu_x, \sigma_x, \lambda, \nu_z, X_0\}$) at $\tau = 0.1$. Simulated transition densities are indicated by gray histograms. R code: Supplementary materials, Section 4.8.



(a) $\boldsymbol{\theta} = \{-0.5, 0.5, 1, 0.75, 0.75, 1\}$

(b) $\boldsymbol{\theta} = \{0, 0.25, 1, 0, 0.2, 1\}$

(c) $\boldsymbol{\theta} = \{1, 0.5, 1, 0.5, 0.2, 1\}$

(d) $\boldsymbol{\theta} = \{0, 0.75, 1, 1, 1, 1\}$

FIGURE 4.4.3: Transition density approximations for Brownian motion with Laplace jumps under various parameter sets ($\boldsymbol{\theta} = \{\mu_x, \sigma_x, \lambda, \mu_z, b_z, X_0\}$) at $\tau = 0.1$. Simulated transition densities are indicated by gray histograms. R code: Supplementary materials, Section 4.8.

comparable approximations on short transition horizons for a variety of jump distributions whilst being less sensitive to increases in the size of the transition horizon and jump intensity. Although our methodology outperforms the series approximations, it can be argued that the latter scheme targets short transition horizons and although it may be subject to discontinuities when there is a miss-match between the support of the jump distribution and the diffusion part of the process, it would rarely make sense for practical applications to use models with such attributes (for example, jumps which may assume only positive values). With respect to the convolution technique, the scope of the approximation is limited to the domain of models with constant diffusion coefficients on short transition horizons, thus rendering the strategy obsolete when the analysis requires the evaluation of more complicated models.

4.4.2 State-dependent models: Characteristic equations

Although models with constant coefficients are useful for a number of applications, it is often more realistic to formulate models with state-dependent coefficients. In the case of Brownian motion, we could exploit the separation between the jump and diffuse dynamics in order to evaluate accurate approximations to the transitional density. When the diffusion model of interest has state-dependent coefficients, jumps directly affect the dynamics of the diffusion part of the process. This makes it extremely difficult to calculate true transitional densities or even valid approximations thereof. A technique that has often been used to circumvent this issue relies on calculating the true characteristic function from which one can subsequently approximate the transitional density numerically. Using this strategy, we can calculate near exact approximations to the transitional density for special cases of jump diffusion models with state-dependent coefficients over arbitrarily sized transition horizons, making it possible to assess the performance of the moment truncation scheme for models with more complicated dynamics than Brownian motion on varying transition horizons. We outline the strategy and compare our methodology at the hand of two jump diffusion models under different jump distributions.

A popular jump diffusion model used in financial applications is the so-called 'basic affine jump diffusion' (BAJD). The dynamics of the BAJD is governed by the SDE:

$$\begin{aligned} dX_t &= \alpha_x(\beta_x - X_t)dt + \sigma_x\sqrt{X_t}dB_t + dP_t \\ dP_t &= \dot{z}_t dN_t \end{aligned} \tag{4.4.10}$$

with $\dot{z}_t \sim \text{Exp}(\nu_z)$ such that $E_z(\dot{z}_t) = 1/\nu_z$, and $N_t - N_s \sim \text{Poi}(\lambda(t-s))$. Previously, Filipović *et al.* (2013) made use of the characteristic function in order

to evaluate the transitional density of the integrated BAJD (see Eckner (2009) for the corresponding characteristic function), whilst Jin *et al.* (2016) make use of the characteristic function in order to establish recurrence properties of the BAJD. Let $\tau = t - s$, then the true moment generating function of the BAJD is given by (Jin *et al.*, 2016):

$$\Psi(u) = E_X(\exp(uX_t)) = \left(1 - \frac{u\sigma_x^2}{2\alpha_x}(1 - e^{-\alpha_x \tau})\right)^{-\frac{2\alpha_x \beta_x}{\sigma_x^2}} \exp\left(\frac{u\lambda(1 - e^{-\alpha_x \tau})}{\alpha_x(\nu_z - u)} + \frac{uX_s e^{-\alpha_x \tau}}{1 - \frac{u\sigma_x^2}{2\alpha_x}(1 - e^{-\alpha_x \tau})}\right) \tag{4.4.11}$$

if $\alpha_x - 0.5\sigma_x^2 \nu_z = 0$, and

$$\Psi(u) = E_X(\exp(uX_t)) = \left(1 - \frac{u\sigma_x^2}{2\alpha_x}(1 - e^{-\alpha_x \tau})\right)^{-\frac{2\alpha_x \beta_x}{\sigma_x^2}} \left(\frac{\nu_z(1 - \frac{u\sigma_x^2}{2\alpha_x}) - u(1 - \frac{u\sigma_x^2 \nu_z}{2\alpha_x})e^{-\alpha_x \tau}}{\nu_z - u}\right)^{\frac{\lambda}{\alpha_x - 0.5\sigma_x^2 \nu_z}} \times \exp\left(\frac{uX_s e^{-\alpha_x \tau}}{1 - \frac{u\sigma_x^2}{2\alpha_x}(1 - e^{-\alpha_x \tau})}\right) \tag{4.4.12}$$

otherwise. Subsequently, we may solve for an approximate transitional density by numerically inverting the characteristic equation

$$\Psi(iv) = \int_{-\infty}^{\infty} e^{ivx} f(x|X_s) dx, \tag{4.4.13}$$

where $i^2 = -1$. This can be achieved by making use of the fast Fourier transform (FFT) (see for example Carr and Madan (1999) and the **fftw** (Krey *et al.*, 2011) R-package), through which one can evaluate the transitional density at discrete nodes on a finite strip of the state-space.

Figure 4.4.4 shows the transition density of the BAJD calculated by inverting Equation 4.4.12 via Equation 4.4.13 for various parameter sets. For comparison, we approximate the transitional density using the jump diffusion moments directly in conjunction with the fourth-order saddlepoint approximation (i.e., without factorization) as well as with the fourth-order saddlepoint approximation under the mixture factorization (as in $f^{(4)}_{SPT}(X_t|X_s)$ before). As evidenced by the close proximity of the density approximation to the solution calculated using the Fourier transform, apart from slight deviation at the modes of the distribution, the mixture factorization produces accurate approximations of the transitional density on both short and long transition horizons, whilst the single saddlepoint approximation under the jump diffusion moments fails to produce an accurate approximation for all but one of the experimental parameter sets. In the case

of Figure 4.4.4(d) it is in fact not a consequence of the particular parameter set resulting in the approximation being accurate regardless of whether the mixture factorization is applied or not, but rather the size of the transition horizon: On such a long horizon the mixture factorization becomes redundant since the probability having observed at least one jump is close to one. Consequently, the excess and jump diffusion moments will be nearly identical and the factorized and un-factorized density approximations will be equivalent.

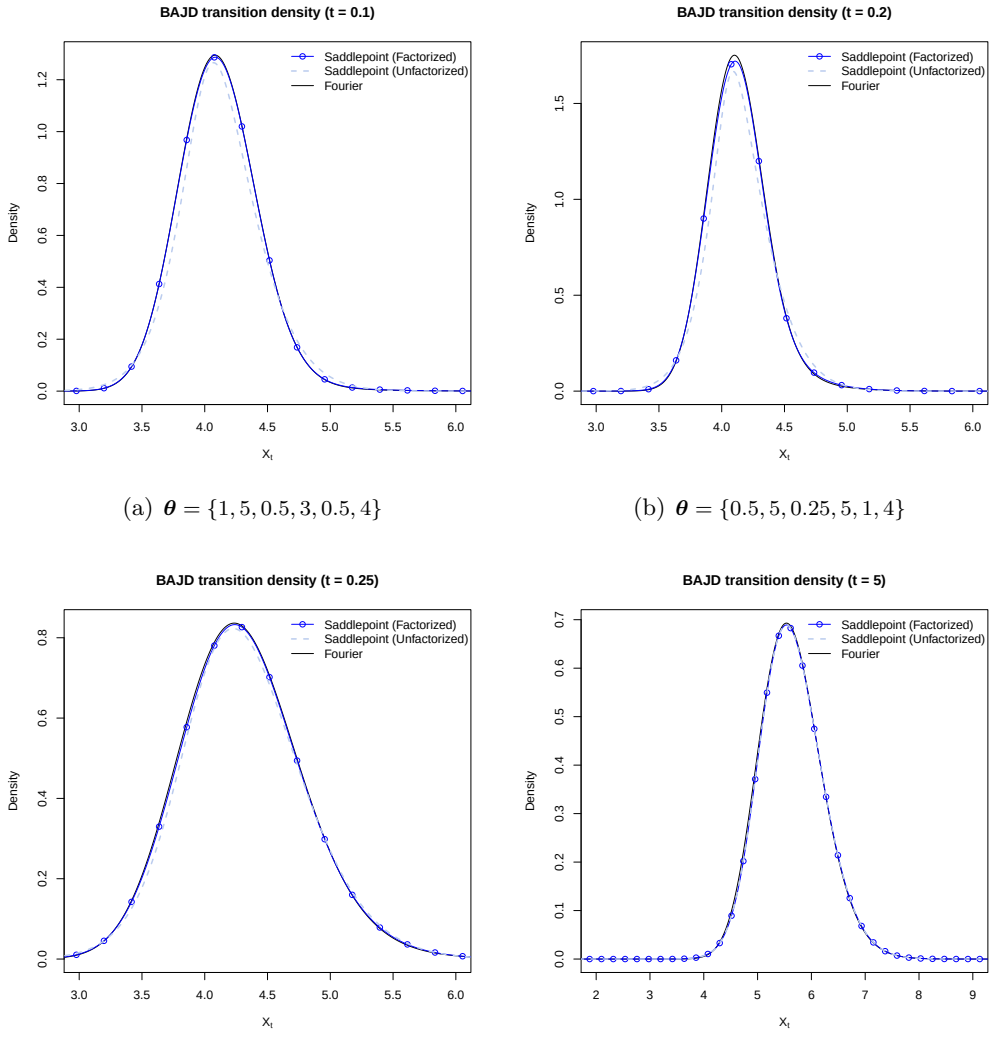
Similarly to the BAJD, it is possible to derive an analytical expression for the characteristic function of an Ornstein-Uhlenbeck type jump diffusion model with Laplace distributed jumps. That is, given a jump diffusion of the form:

$$\begin{aligned} dX_t &= -\alpha_x X_t dt + \sigma_x dB_t + dP_t \\ dP_t &= \dot{z}_t dN_t \end{aligned} \tag{4.4.14}$$

with $\dot{z}_t \sim \text{Laplace}(0, b_z)$, and $N_t - N_s \sim \text{Poi}(\lambda(t-s))$, the true MGF of the process is given by the expression Yu (2007):

$$\begin{aligned} &\Psi(u) = E_X(\exp(uX_t)) = \\ &\left(\frac{1-u^2 b_z^2 e^{-2\alpha_x \tau}}{1-u^2 b_z^2}\right)^{\frac{\lambda}{2\alpha_x}} \exp\left(uX_s e^{-\alpha_x \tau} - \frac{u^2\sigma_x^2}{4a}\left(1-e^{-2\alpha_x\tau}\right)\right). \end{aligned} \tag{4.4.15}$$

Figure 4.4.5 compares the density approximation for Equation 4.4.14 using the single fourth-order saddlepoint approximation and a fourth-order saddlepoint approximation under the mixture factorization to the numerical analogue calculated applying the FFT to Equation 4.4.13 under Equation 4.4.15. For Laplace distributed jumps, the factorized approximation produces reasonably accurate approximations for all parameter sets whilst slightly underestimating the peak of the distribution on longer transition horizons.



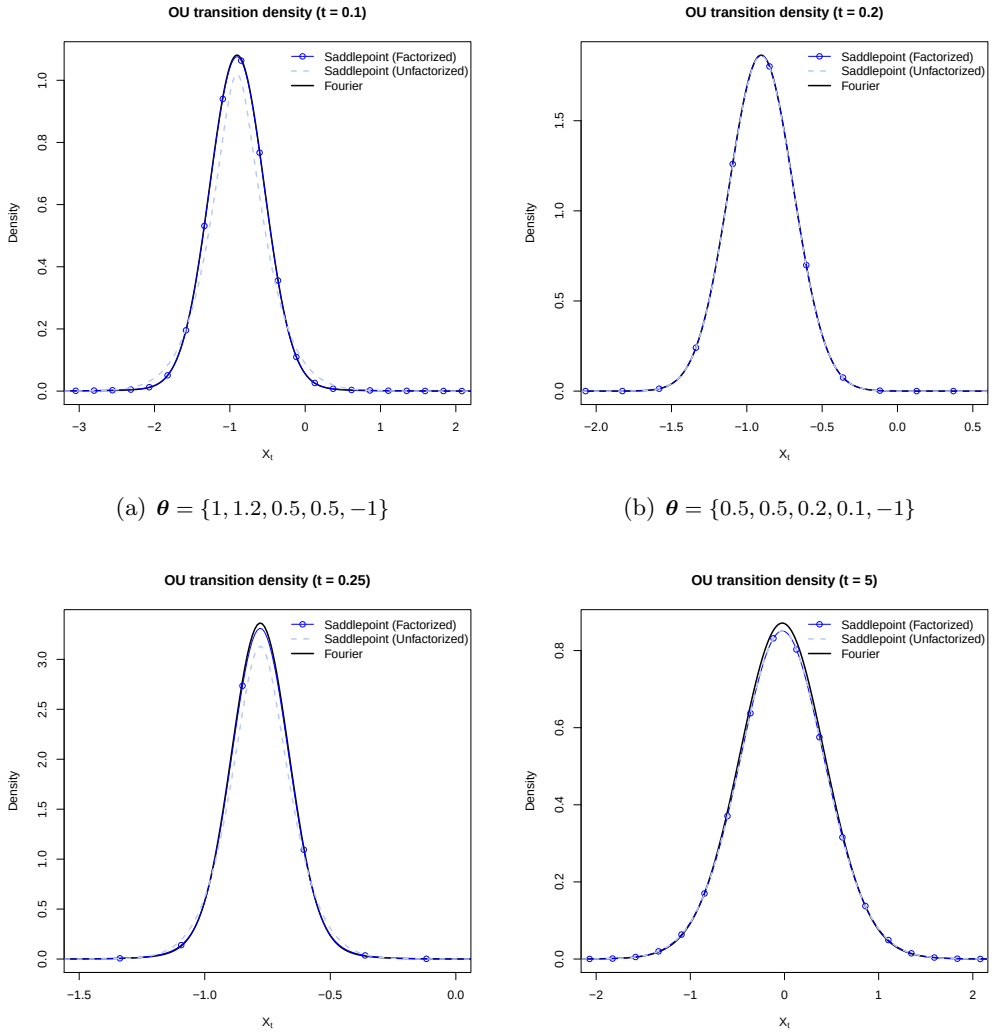
(a) $\boldsymbol{\theta} = \{1, 5, 0.5, 3, 0.5, 4\}$

(b) $\boldsymbol{\theta} = \{0.5, 5, 0.25, 5, 1, 4\}$

(c) $\boldsymbol{\theta} = \{1, 5, 0.5, 4, 1.5, 4\}$

(d) $\boldsymbol{\theta} = \{0.75, 5, 0.25, 6, 3, 4\}$

FIGURE 4.4.4: Transition density for Equation 4.4.10 evaluated under various parameter sets ($\boldsymbol{\theta} = \{\alpha_x, \beta_x, \sigma_x, \nu_z, \lambda, X_0\}$) and transition horizons. The benchmark transition density is calculated via the Fourier transform of Equation 4.4.13 in conjunction with Equation 4.4.12. For comparison, we calculate the approximate transitional density using the moment equations in conjunction with a saddlepoint approximation and Equation 4.3.71, respectively. R code: Supplementary materials, Section 4.9.



(a) $\boldsymbol{\theta} = \{1, 1.2, 0.5, 0.5, -1\}$

(b) $\boldsymbol{\theta} = \{0.5, 0.5, 0.2, 0.1, -1\}$

(c) $\boldsymbol{\theta} = \{1, 0.25, 1, 0.1, -1\}$

(d) $\boldsymbol{\theta} = \{0.75, 0.5, 1, 0.2, -1\}$

FIGURE 4.4.5: Transition density for Equation 4.4.14 evaluated under various parameter sets ($\boldsymbol{\theta} = \{\alpha_x, \sigma_x, \lambda, b_z, X_0\}$) and transition horizons. The benchmark transition density is calculated via the Fourier transform of Equation 4.4.13 in conjunction with Equation 4.4.15. For comparison, we calculate the approximate transitional density using the moment equations in conjunction with a saddlepoint approximation and Equation 4.3.71, respectively. R code: Supplementary materials, Section 4.9.

4.5 Likelihood inference

We now turn focus to the problem of performing inference on jump diffusion models. Consider a process observed discretely at time epochs $t_1, t_2, \ldots, t_n$ giving rise to the time series[2] $D_S = \{\mathbf{X}_{t_1}, \mathbf{X}_{t_2}, \ldots, \mathbf{X}_{t_n}\}$. Given some jump diffusion model parametrised by the vector $\boldsymbol{\theta}$, we can formulate the likelihood mathematically from the transitional density using the usual Markov arguments:

$$L(\boldsymbol{\theta}|D_S) \propto \prod_{i=1}^{n-1} f(\mathbf{X}_{t_{i+1}}|\mathbf{X}_{t_i}, \boldsymbol{\theta}). \tag{4.5.1}$$

The likelihood is thus constructed by measuring the probability that the process transits from one known state to another. Given the transitional density of the jump diffusion, we may construct the likelihood function accordingly. Although performing inference in this way is mathematically sound, it is important to understand how the dynamics of the process relate to the likelihood. Specifically, since the likelihood is constructed from observations on the jump diffusion trajectory with no direct account of the jump realisations an element of latency is incurred with respect to the jump mechanism. This follows since we only partially observe the dynamics of the underlying process in the sense that the jump part is only visible through its effect on the trajectories of the diffusion part of the process. Consequently, depending on the nature of the jump process and the resolution of the observed series, the quality of the inference that can be made may vary. However, this amounts to informational latency rather than a methodological issue. Indeed, despite the absence of direct observations for the jump realisations, the jumps are integrated into the trajectory of the process and thus the signature of the jump mechanism is embedded within the observed trajectory of the process. Thus, provided the sampling resolution is sufficiently high to observe such an effect, we may still infer the dynamics of the jump mechanism indirectly through the jump diffusion likelihood function.

Naturally, since the transitional density in general is not available, the likelihood function remains intractable. However, when a suitably accurate approximation to the transitional density is available we may replace the true transitional density with that of the approximation in order to calculate an approximate likelihood function. For example, applying the mixture factorization derived in Section 4.3.4

[2]Note that we switch to the subscript n for the final observation in order to avoid confusion with the Poisson component N_t.

we can write the likelihood as:

$$L(\boldsymbol{\theta}|D_S) \propto$$
$$\prod_{i=1}^{n-1}\Big(P(\mathbf{N}_{t_{i+1}}-\mathbf{N}_{t_i}=0)f_D(\mathbf{X}_{t_{i+1}}|\mathbf{X}_{t_i},\boldsymbol{\theta})+P(\mathbf{N}_{t_{i+1}}-\mathbf{N}_{t_i}>0)f_E(\mathbf{X}_{t_{i+1}}|\mathbf{X}_{t_i},\boldsymbol{\theta})\Big). \tag{4.5.2}$$

By replacing $f_D(.)$ and $f_E(.)$ in Equation 4.5.2 with appropriate approximations, we can use the resulting likelihood approximation in order to perform inference. Naturally, this hinges on the likelihood approximation being of sufficient quality to mimic the behaviour of the true likelihood over plausible parameter spaces.

4.5.1 Exact vs. approximate likelihoods

Despite the calculation of the likelihood function following naturally from the transitional density, conducting likelihood inference via an approximate transition density poses a much more strenuous test on the accuracy and robustness of the approximation scheme. This follows since we require the approximation to retain a reasonable degree of accuracy for a wide range of parameters. As an initial experiment we revisit the example of Brownian motion with jumps. In the case of Brownian motion with Normally distributed jumps we can calculate the 'true' likelihood using the truncated true transitional density and evaluate the likelihood function as:

$$L(\boldsymbol{\theta}|D_S) \underset{\sim}{\propto} \prod_{i=1}^{n-1} f^{(m)}_{true}(X_{t_{i+1}}|X_{t_i},\boldsymbol{\theta}=\{\mu_x,\sigma_x,\lambda,\mu_z,\sigma_z\}) \tag{4.5.3}$$

for m large. Similarly, by replacing the transitional density with the fourth order saddlepoint approximation under the mixture factorization we may approximate the likelihood function by:

$$L(\boldsymbol{\theta}|D_S) \underset{\sim}{\propto} \prod_{i=1}^{n-1} f^{(4)}_{SPT}(X_{t_{i+1}}|X_{t_i},\boldsymbol{\theta}=\{\mu_x,\sigma_x,\lambda,\mu_z,\sigma_z\}). \tag{4.5.4}$$

As with the transition density benchmarks, we may then use Equation 4.5.3 as a benchmark for calculating parameter estimates under an approximate likelihood function. By performing inference on a dataset for which we know the 'true' parameter values we can subsequently compare maximum likelihood estimates under the true and approximate likelihoods. In order to do so we may simulate a hypothetical dataset under a known parameter set by for example using the Euler–Maruyama scheme to simulate a discrete sample trajectory of the model

process at a suitably high resolution and subsequently draw observations at a desired sample resolution. For example, in the experiment that follows we simulate a sample trajectory of Equation 4.4.1 on the observation horizon [0,25] and draw observations at an equispaced resolution of $t_{i+1} - t_i = 1/10$ for $i = 1, 2, ..., 251$ using the parameters $\boldsymbol{\theta} = \{\mu_x, \sigma_x, \lambda, \mu_z, \sigma_z\} = \{1, 1, 1, 1, 1\}$. Subsequently we calculate maximum likelihood estimates for the parameter vector under the simulated dataset using equations 4.5.3 and 4.5.4. By repeating this experiment a number of times we can compare empirically the properties of the maximum likelihood estimates under the exact and approximate likelihoods. Figure 4.5.1 shows a sample of two hundred simulated trajectories for Equation 4.4.1 under the assumed parameter set.

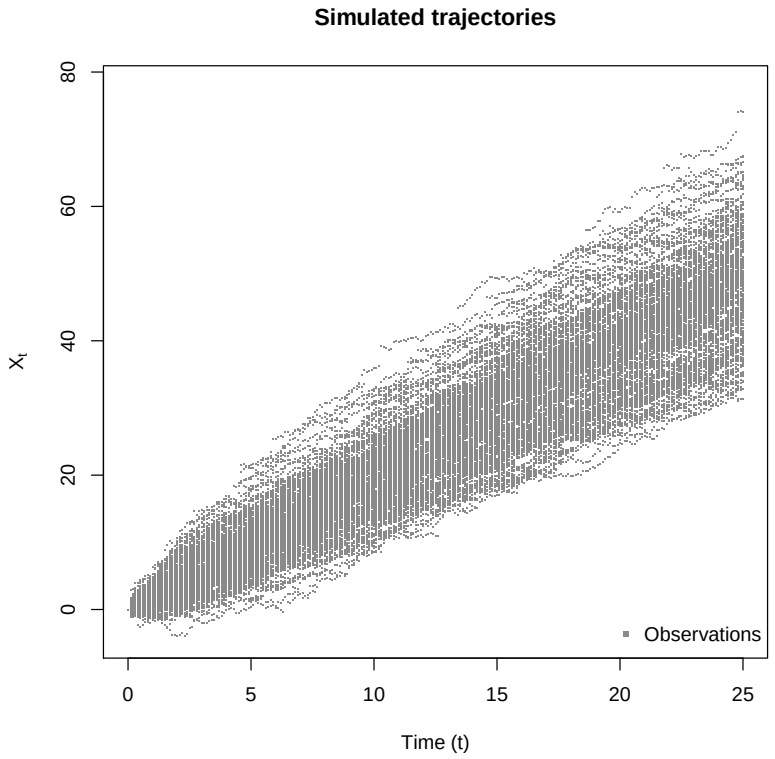


FIGURE 4.5.1: Two hundred simulated trajectories of Brownian motion with Normal distributed jumps. Observations are recorded at equispaced epochs 0.1 time units apart. R code: Supplementary materials, Section 4.10.

Figure 4.5.2 illustrates the offset of the maximum likelihood estimates from the true parameters for 2 500 simulation runs calculated under the true and

approximate likelihoods. The resulting figures suggest that there are no noticeable systematic differences between parameter estimates calculated under the true and approximate likelihoods. Note that we fixed the range of the x-axis on each figure in order to get an idea of the variability in maximum likelihood estimates calculated at the simulated data resolution. Indeed the asymptotic behaviour of the parameter estimates appears consistent with what is known for diffusion processes in general. For example, the variability in the estimates for the diffusion coefficient σ_x is significantly less than for the drift parameter μ_x. In both cases, the distribution of the offset from the true parameter for the diffusion parameters are more or less symmetric about zero. In contrast, the parameters of the jump mechanism show significantly more variability than the diffusion parameters, and in the case of λ a clear systematic bias is present. Indeed it makes sense that the intensity parameter should typically be underestimated given that the jump part of the process is observed indirectly and only a fraction of the dataset will contain any information about the jump mechanism.

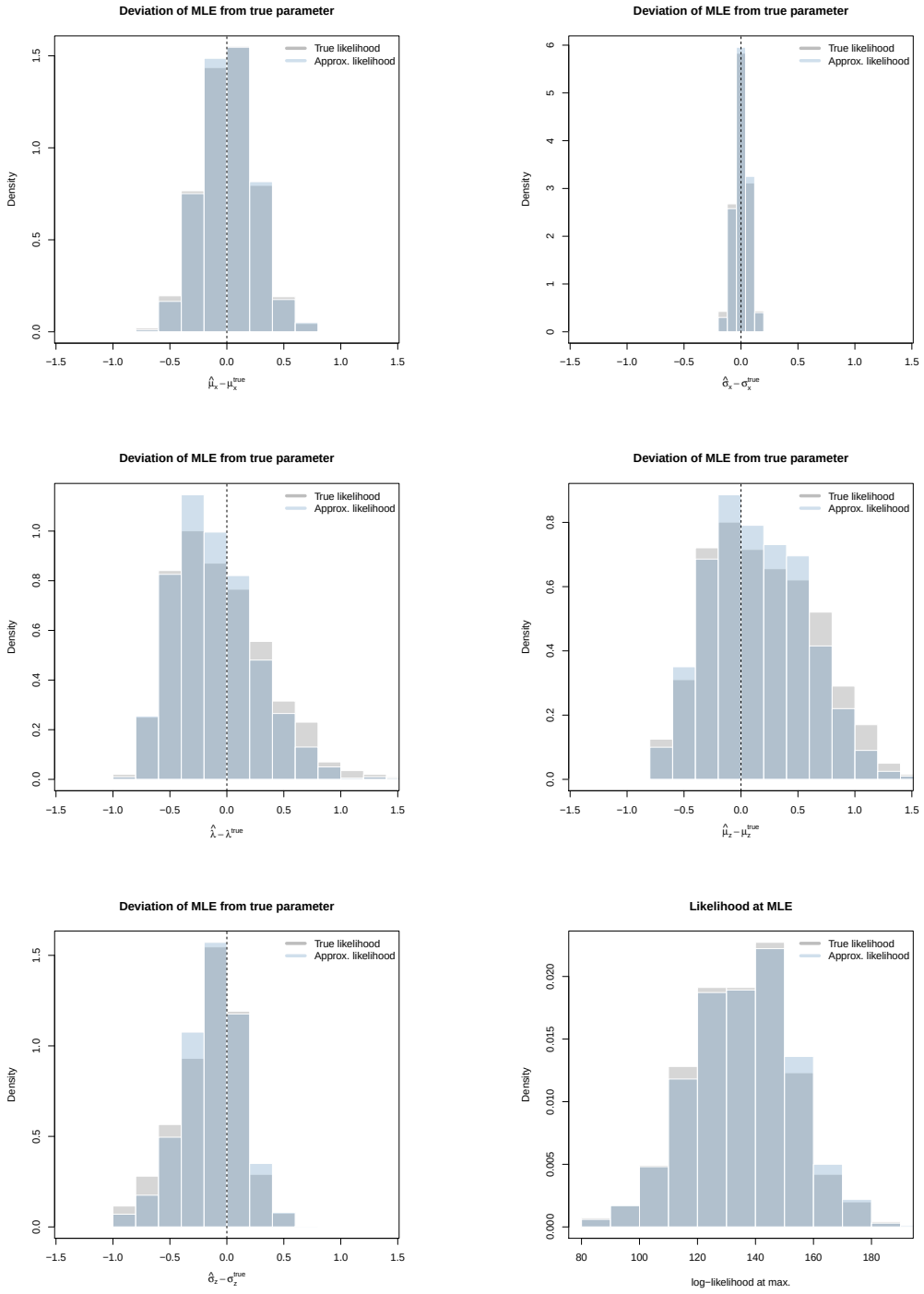


FIGURE 4.5.2: Frequency distributions of maximum likelihood estimate deviations from the true parameters under the true likelihood function (calculated using $f^{(10)}_{true}(X_t|X_s)$) and the approximate likelihood function (calculated using $f^{(4)}_{SPT}(X_t|X_s)$). R code: Supplementary materials, Section 4.10.

4.5.2 Likelihood inference for a non-linear process

Although results from the simulation experiment on Brownian motion are favourable, it can be argued that the model structure is too simple to make any broader conclusions as to the performance of the proposed methodology. Indeed, the purpose of developing the methodology in such a general way is to make inference tractable for non-linear jump diffusion processes with state-dependent jump elements. As such we repeat the simulation experiment of Section 4.5.1 for a more complicated jump diffusion process. For example, consider a jump diffusion process with dynamics given by the SDE:

$$\begin{aligned} dX_t &= \alpha_x(\beta_x - X_t)dt + \sigma_x\sqrt{X_t}dB_t + dP_t \\ dP_t &= \dot{z}_t dN_t \end{aligned} \tag{4.5.5}$$

with $\lambda(X_t, t) = \kappa X_t$ and $\dot{z} \sim \mathrm{N}(\mu_z, \sigma_z^2)$. Equation 4.5.5 is similar to the BAJD in the sense that the coefficients of the diffusion part of the process are state-dependent but with the important distinction that the jump intensity is dependent on the state of the process. Consequently, the probability of a number of jumps occurring varies in accordance to the state of the jump diffusion. Despite this complication, we can use the methodology of Section 4.3 in order to calculate an accurate transitional density approximation and subsequently construct an approximate likelihood function. Unfortunately, in this case we cannot benchmark the accuracy of the resulting parameter estimates since the true likelihood or suitable analogue thereof is not available. Note however that Equation 4.5.5 is constructed from elements which do have tractable likelihoods. For example, the diffusion part of the process consists of a CIR process for which the true likelihood follows (Cox *et al.*, 1985):

$$L(\boldsymbol{\theta}|D_S^\star) \propto \prod_{i=1}^{n-1} c_i \left(\frac{v_i}{u_i}\right)^{q/2} e^{-(u_i+v_i)} I_q(2\sqrt{u_i v_i}) \tag{4.5.6}$$

where $c_i = 2\alpha_x/(\sigma_x^2(1 - e^{-\alpha_x(t_{i+1}-t_i)}))$, $v_i = c_i X_{t_{i+1}}$, $u_i = c_i e^{\alpha_x(t_{i+1}-t_i)} X_{t_i}$, $q = 2\alpha_x\beta_x/\sigma_x^2 - 1$, $I_q(.)$ denotes the Bessel function of the first kind of order q, and $D_S^\star$ denotes observations of the process *ex*-jumps. Furthermore, let $D_J = \{\dot{z}_{\tau_i} : i = 1, 2, \ldots, n_J\}$ denote the sequence of jump realisations where $D_T = \{\tau_i : i = 1, 2, \ldots, n_J\}$ denotes the exact sequence of jump times (i.e. the exact times between or at the observation times of the jump diffusion at which jumps occurred). Then the likelihood of the jump observations can be calculated

straightforwardly as:

$$L(\{\mu_z, \sigma_z\}|D_J) = \prod_{i=1}^{n_J} \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\left(- \frac{(\dot{z}_{\tau_i} - \mu_z)^2}{2\sigma_z^2} \right). \tag{4.5.7}$$

In the case of the jump times, the situation is a bit more complicated since the intensity of the process depends on the jump diffusion. However, the appropriate expression for the likelihood can be derived as:

$$L(\lambda_z|D_T, \alpha_x, \beta_x, X_t) = \prod_{i=1}^{n_J-1} \kappa m_{X_t}(u|\tau_i) e^{-\kappa m_{X_t}(u|\tau_i)} \tag{4.5.8}$$

where

$$m_{X_t}(\tau_{i+1}|\tau_i) = \int_{\tau_i}^{\tau_{i+1}} \left[X_{\tau_i} e^{-\alpha_x(u-\tau_i)} + \beta_x(1 - e^{-\alpha_x(u-\tau_i)}) \right] du. \tag{4.5.9}$$

Naturally, in order to calculate likelihoods for the individual components of the model we require perfect data in the sense that we need to know the exact times at which jumps occurred as well as their exact magnitude, which is somewhat unrealistic, however by comparing the parameter estimates calculated under the approximate likelihood function for the jump diffusion observations data to that of the individual components calculated from 'perfect' data we can gain insight into the amount of informational loss that occurs.

Figure 4.5.3 shows two hundred simulated trajectories for Equation 4.5.5 under the parameter set $\boldsymbol{\theta} = \{\alpha_x, \beta_x, \kappa, \mu_z, \sigma_z\} = \{1, 5, 0.1, 0.2, 0.5, 0.5\}$ with initial value $X_0 = 5$. Based on these trajectories we can once again calculate maximum likelihood estimates for comparison to the true parameter set. Figure 4.5.4 shows frequency distributions for the deviations of MLEs from the true values calculated from 2 500 simulated trajectories observed at equispaced points 0.1 time units apart on the observation horizon $[t_1, t_{251}] = [0, 25]$. Superimposed we show deviations for direct MLEs calculated from the diffusion, jump and arrival time likelihoods respectively as constructed by recording the diffusion and jump components of the process directly during each simulation run. Interestingly, although there still remains some bias for the intensity parameter, the jump diffusion parameter estimates appear to be quite accurate as compared to the direct parameter estimates. Moreover, in comparison to the direct estimates for the jump mechanism parameters, apart from the expected intensity bias there appears to be little loss in accuracy from calculating the parameters via the jump diffusion trajectory under the approximate likelihood function.

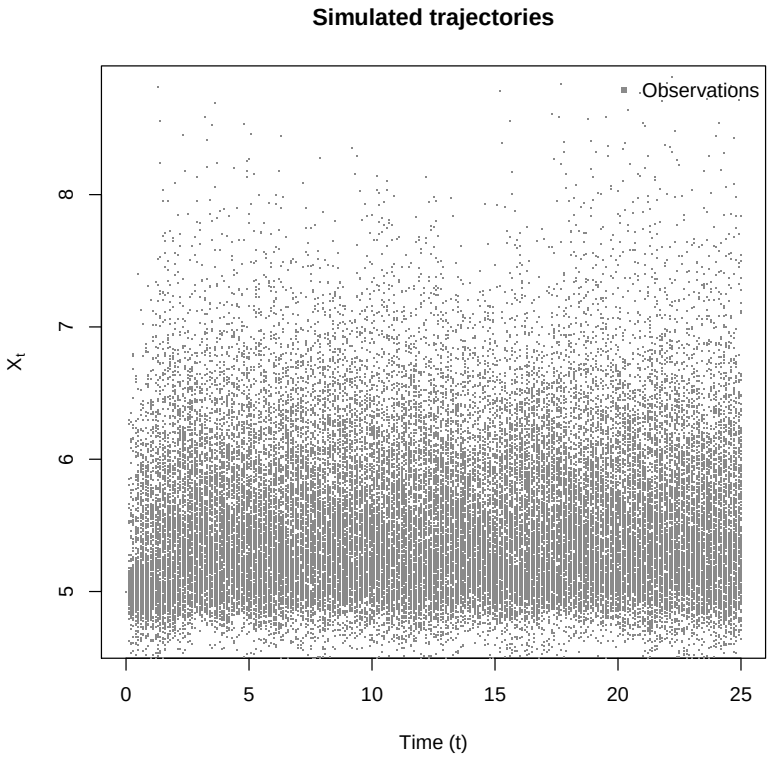


FIGURE 4.5.3: Two hundred simulated trajectories of a CIR process with Normal distributed jumps and state-dependent intensity. Observations are recorded at constant transition horizons of 0.1 time units. R code: Supplementary materials, Section 4.11.

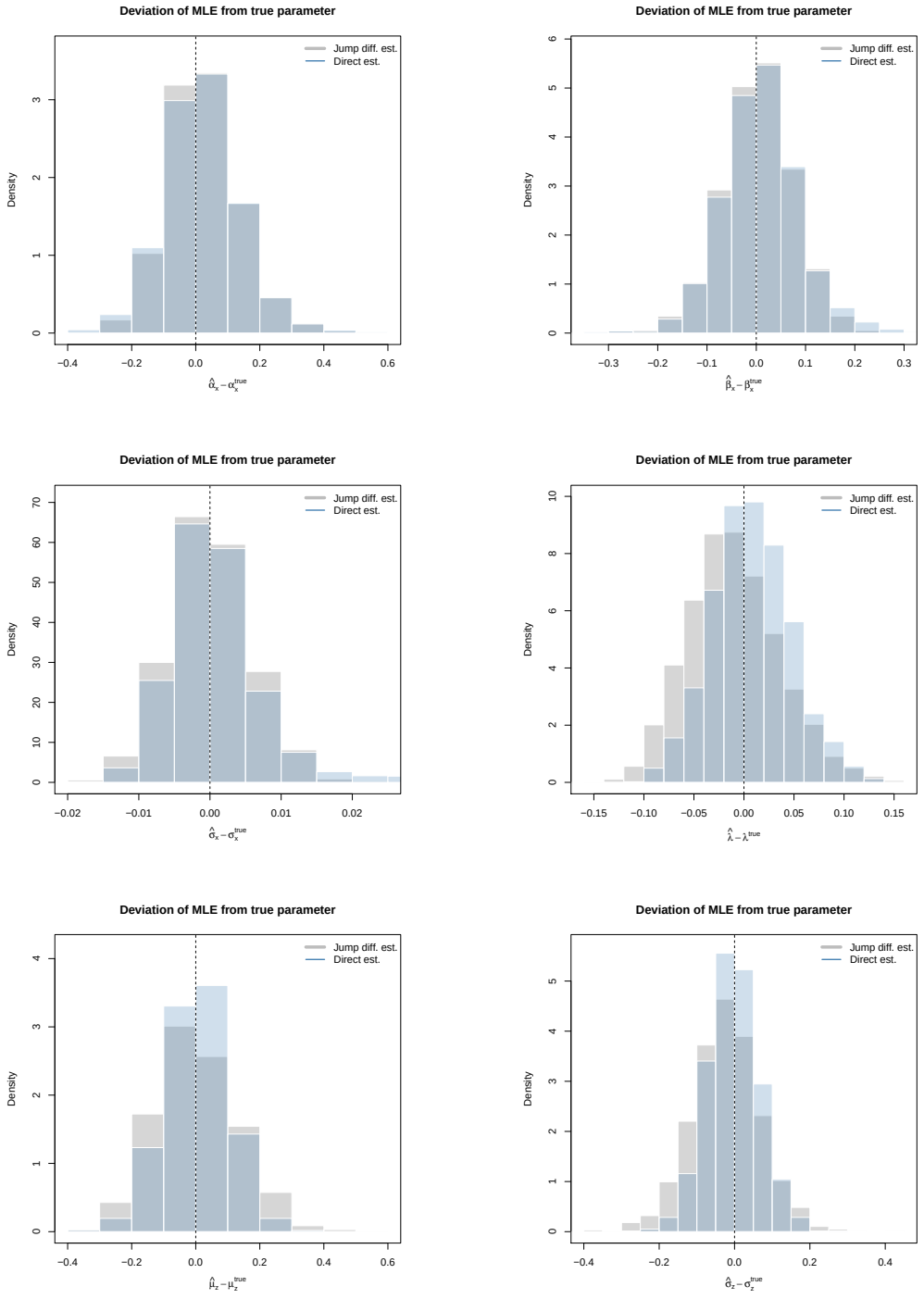


FIGURE 4.5.4: Frequency distributions of maximum likelihood estimate deviations from the true parameters calculated using the approximate likelihood function for 2 500 simulated trajectories for a fixed parameter set. Direct maximum likelihood estimate deviations for the diffusion and jump mechanism parameters are also indicated. R code: Supplementary materials, Section 4.11.

Although the results suggest that one can accurately extract parameters for the jump diffusion mechanism it is important to note the role that the characteristics of the data resolution play in the quality of the estimation results. For this example, we chose a moderate data resolution on a reasonable observation horizon. Naturally, it can be expected that the accuracy of the jump diffusion parameter estimates as calculated via the jump diffusion trajectory may deteriorate when the data is of inadequate resolution or when the jump signal is particularly weak. Indeed, for the present parameter set the jumps should be visible at the assumed data resolution. If the sample resolution were doubled, the jumps (or their effect on the process) would be significantly easier to detect and the accuracy would improve. Likewise, if the data resolution was sparse we would require a significantly longer dataset in order to retain the achieved accuracy. However, this again points to the informational efficiency of performing inference on a partially observed jump diffusion rather than the accuracy of the approximation. In this regard, when compared to estimates calculated from direct observations of the constituents of the jump diffusion, the estimates remain sufficiently accurate despite only being observed indirectly.

4.5.3 A bivariate, coupled process with Normally distributed jumps.

In the preceding simulation experiments, we show that using the approximate likelihood under the mixture factorization produces reliable parameter estimates for discretely observed jump diffusions. By repeatedly simulating trajectories we can get an idea of the amount of variation that can be expected in the maximum likelihood estimates of a given jump diffusion model's parameters. Naturally, this variation depends on a number of factors such as the specification of the model coefficients, the size of the parameters and most importantly the sampling resolution and the length of the observed series. However, in practice we only have access to a single realisation of the true data generating process and more often than not at a fixed sample resolution. Indeed, in the case of a single realisation of the data generating process it may be that the parameter estimates are relatively far from that of the true parameter set, not because of errors in the likelihood approximation but because that particular realisation may be one which is improbable (but not impossible) under the given parameter set. Consider for example the process:

$$\begin{aligned} dX_t &= \alpha_x(\beta_x + Y_t - X_t)dt + \sigma_x\sqrt{X_tY_t}dB_t^{(1)} + dP_t^{(1)} \\ dY_t &= \alpha_y(\beta_y - Y_t)dt + \sigma_y\sqrt{Y_t}dB_t^{(2)} + dP_t^{(2)} \end{aligned} \tag{4.5.10}$$

Parameter	True Value	Estimate	90% CI
μ_x	0.50	0.54	(0.39, 0.68)
β_x	2.00	1.92	(1.47, 2.33)
σ_x	0.10	0.11	(0.10, 0.11)
μ_y	1.00	1.05	(0.92, 1.18)
β_y	5.00	4.96	(4.88, 5.01)
σ_y	0.10	0.11	(0.10, 0.11)
λ	1.00	1.10	(0.82, 1.43)
$\mu_{z_{11}}$	0.50	0.46	(0.30, 0.61)
$\mu_{z_{21}}$	0.50	0.31	(0.21, 0.45)
$\sigma_{z_{11}}$	0.50	0.53	(0.41, 0.70)
$\sigma_{z_{21}}$	0.50	0.55	(0.47, 0.66)

TABLE 4.5.1: Parameter estimates and 90% credibility intervals for Equation 4.5.10 under the simulated dataset in Figure 4.5.5. R code: Supplementary materials, Section 4.12.

with

$$\begin{aligned} dP_t^{(1)} &= \dot{z}_t^{(11)} dN_t^{(1)} \\ dP_t^{(2)} &= \dot{z}_t^{(21)} dN_t^{(1)} \end{aligned} \tag{4.5.11}$$

where $N_t^{(1)} - N_s^{(1)} \sim \text{Poi}(\lambda(t-s))$, and

$$\{\dot{z}_t^{(11)}, \dot{z}_t^{(21)}\}' \sim \text{Bivariate Normal}(\{\mu_{z_{11}}, \mu_{z_{21}}\}', diag(\{\sigma_{z_{11}}^2, \sigma_{z_{21}}^2\}')). \tag{4.5.12}$$

For purposes of the simulation study we shall assume that the true values of the parameter vector are given by $\boldsymbol{\theta} = \{\alpha_x, \beta_x, \sigma_x, \alpha_y, \beta_y, \sigma_y, \lambda, \mu_{z_{11}}, \mu_{z_{21}}, \sigma_{z_{11}}, \sigma_{z_{21}}\} = \{0.5, 2, 0.1, 1, 5, 0.1, 1, 0.5, 0.5, 0.5, 0.5\}$. Figure 4.5.5 illustrates a simulated trajectory for Equation 4.5.10. For this particular simulation the parameter estimates for the jump distribution calculated directly from the sequence of jump realisations (ignoring the times at which they occurred) differ somewhat from the true values ($\{\hat{\mu}_{z_{11}}, \hat{\mu}_{z_{21}}, \hat{\sigma}_{z_{11}}^2, \hat{\sigma}_{z_{21}}^2\} = \{0.52, 0.32, 0.48, 0.56\}$). Naturally, despite the jump sequence only being observed indirectly through the jump diffusion trajectory, the characteristics of the jump sequence may still be preserved in the jump diffusion trajectory. Using the mixture factorization in conjunction with the bivariate saddlepoint approximation (see Appendix D.5) we can extract maximum likelihood estimates for the parameters of Equation 4.5.10. Table 4.5.1 gives parameter estimates and 90% credibility intervals for the simulated trajectory of Equation 4.5.10 calculated using the random walk Metropolis-Hastings algorithm.

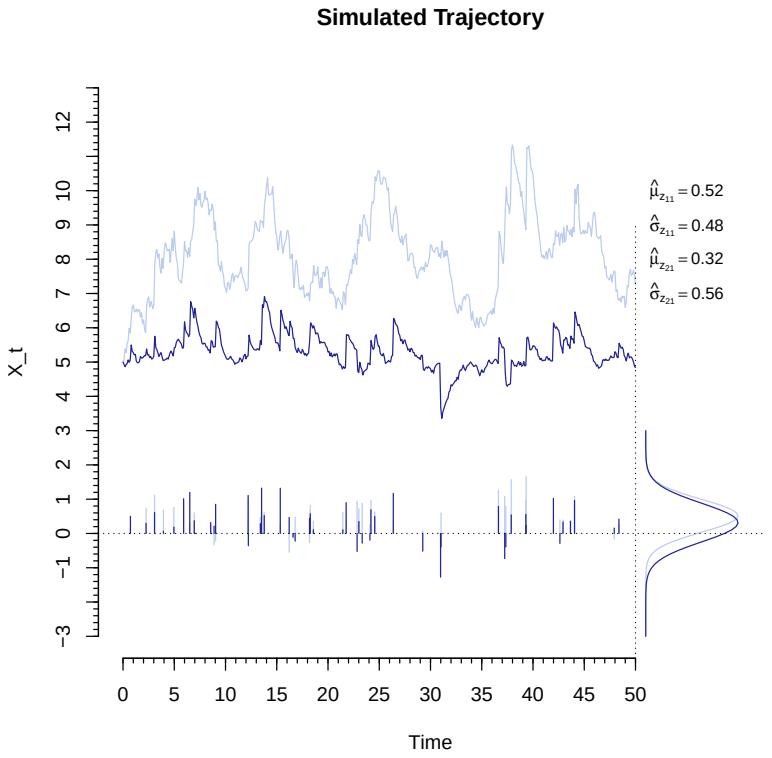


FIGURE 4.5.5: Simulated trajectory for Equation 4.5.10 sampled at a resolution of 0.1 units of time per transition on the observation horizon $[0, 50]$. Jump innovations for X_t (dark blue) and Y_t (light blue) are indicated around the time axis. Estimated mean and standard deviations calculated from the raw sequence of jump inovations also indicated. R code: Supplementary materials, Section 4.12.

The resulting parameter estimates match that of the true parameter set quite closely, with the notable exception of $\mu_{z_{21}}$. However, closer inspection reveals that the estimate calculated under the jump diffusion model is indeed valid since the value calculated from the jump realisations directly is quite similar. Indeed, for this experiment, the jump signal is quite strong since the dispersion of the jump distribution is large relative to the diffusion parameters. Consequently, the contrast between the jump and diffusion dynamics is sufficiently high to make quite accurate inference with regard to the dynamics of the jump mechanism. Although the estimated parameters are close to the true parameter set, the particular sequence of jump realisations generated in the simulation contains values which are relatively unlikely under the true parameter set. Despite this,

the parameters of the jump mechanism could still be extracted accurately, albeit preserving the attributes of the unlikely jump sequence.

4.5.4 Detecting jumps

One of the benefits of constructing the likelihood using the mixture factorization is that it affords the opportunity to 'decode' the sequence of jump arrival times from the discretely observed trajectory. Specifically, we can identify which transitions ($[t_i, t_{i+1}]$) likely contain jumps under a given model. In order to achieve this, we modify the Metropolis-Hastings algorithm by including a peripheral step which attempts to detect which transitions contain jumps. Formally:

1. Initialize a set of dummy indicator variables $\{\iota_1^{old} = 0, \iota_2^{old} = 0, \ldots, \iota_{n-1}^{old} = 0\}$. Given some starting parameter vectors $\boldsymbol{\theta} = (\theta_i)_{p\times 1}$ and $\boldsymbol{\sigma} = (\sigma_i)_{p\times 1}$, set $\boldsymbol{\theta}^{old} = \boldsymbol{\theta}$ and

2. propose an update for the parameter vector by setting

$$\theta_i^{new} = \theta_i^{old} + Z_{\sigma_i} \tag{4.5.13}$$

   for all $i = 1, 2, \ldots p$, where Z_{σ_i} are $\mathrm{N}(0, \sigma_i^2)$ random deviates.

3. Subsequently, evaluate the ratio:

$$R = \frac{\prod_{i=1}^{n-1} f(\mathbf{X}_{t_{i+1}}|\mathbf{X}_{t_i}, \boldsymbol{\theta}^{new})\pi(\boldsymbol{\theta}^{new})}{\prod_{i=1}^{n-1} f(\mathbf{X}_{t_{i+1}}|\mathbf{X}_{t_i}, \boldsymbol{\theta}^{old})\pi(\boldsymbol{\theta}^{old})} \tag{4.5.14}$$

   where $\pi(\boldsymbol{\theta})$ denotes a prior density on the parameter vector.

4. Then accept the proposed move with probability $\min(R, 1)$. That is set

$$\boldsymbol{\theta}^{old} = \begin{cases} \boldsymbol{\theta}^{new} & \text{if } \min(R,1) > u \\ \boldsymbol{\theta}^{old} & \text{otherwise,} \end{cases} \tag{4.5.15}$$

   where u is a $\mathrm{U}(0, 1)$ random deviate.

5. For each $j = 1, 2, \ldots, n-1$, draw $\iota_j^{new} \in \{0, 1\}$, calculate the ratio

$$R_j = \begin{cases} \exp(\ell_j^E - \ell_j^D) & \text{if } \iota_j^{old} = 0 \text{ and } \iota_j^{new} = 1 \\ \exp(\ell_j^D - \ell_j^E) & \text{if } \iota_j^{old} = 1 \text{ and } \iota_j^{new} = 0 \\ 1 & \text{otherwise} \end{cases} \tag{4.5.16}$$

where

$$\begin{aligned} \ell_j^D &= \log(P(\mathbf{N}_{t_{j+1}} - \mathbf{N}_{t_j} = 0) f_D(\mathbf{X}_{t_{j+1}} | \mathbf{X}_{t_j}, \boldsymbol{\theta}^{old})) \\ \ell_j^E &= \log(P(\mathbf{N}_{t_{j+1}} - \mathbf{N}_{t_j} > 0) f_E(\mathbf{X}_{t_{j+1}} | \mathbf{X}_{t_j}, \boldsymbol{\theta}^{old})) \end{aligned} \tag{4.5.17}$$

and set

$$\iota_j^{old} = \begin{cases} \iota_j^{new} & \text{if } \min(R_j, 1) > u \\ \iota_j^{old} & \text{otherwise,} \end{cases} \tag{4.5.18}$$

where u is a $\mathrm{U}(0,1)$ random deviate.

6. Return to step 2.

Note that the detection step (Step 5) does not affect the parameter updates (steps 2 – 4) in any way as the detection indicators do not feed into the likelihood calculation. The purpose of the detection step is to construct a binary chain of indicators that runs in parallel to the parameter chain. From the resulting binary chain, we can calculate the mean of the chain in order to get an indication of the relative probability of each transition containing a jump observation – i.e., detection probabilities. From this sequence we can formulate a basic rule for 'detecting' which transitions contain jumps. For example, for Equation 4.5.10 under the simulated data in Figure 4.5.5 while estimating the parameters if the process using the RWMH algorithm we can estimate the sequence of detection probabilities. From the resulting sequence, we can formulate a simple decision rule where a detection probability is deemed to be definitive if it is above a certain level.

Consider the simulation study of Section 4.5.3. During the estimation procedure, we apply the jump detection step at each iteration of the Metropolis-Hastings algorithm in order to calculate a binary chain of indicators. By calculating the mean of the indicators for each transition horizon, we estimate the probability that each transition contains a jump. Using, for example, an 80% detection threshold, we assign detection indicators to each transition. Comparing the detection results to the true sequence of jump arrivals, we can analyse the accuracy of the detection scheme. This can be done using a basic contingency table: First, we calculate the number of transitions which were correctly assessed to contain jumps and divide by the total number of actual jumps (row 1, column 1). From this, we can calculate the error rate, i.e. the number of missed detections (row 2, column 1). Naturally, it is possible to detect a jump where there is none, i.e. false positives (row 1, column 2). By symmetry, those transitions which are deemed not to contain jumps are indicated as jump-free (row 2, column 2). Depending on the parameters of the process and the resolution of the data, we can expect varying degrees of accuracy. Table 4.5.2 gives the resulting contingency table for

the simulation study under the 80% detection rule. The table indicates a 87% detection rate with 13% missed detections and a 0.7% false positive detection rate for this dataset.

	Actual	
Predicted	≥ 1 Jumps	No jumps
≥ 1 Jumps	0.87	0.007
No jumps	0.13	0.993

TABLE 4.5.2: Contingency table for jump detection rates under Equation 4.5.10. Rates are calculated by comparing jump indicators calculated by applying the 80% decision rule on the estimated detection probability sequence to the true jump sequence under the simulated data. The results indicate a 87% detection rate with 13% missed detections and a 0.7% false positive detection rate. R code: Supplementary materials, Section 4.12.

Naturally, it is to be expected that some jumps will go undetected since it can be difficult to distinguish for specific transitions whether the magnitude of the transition is due primarily to a jump realisation or the diffusion trajectory. Indeed, it is entirely possible for two jumps occurring in quick succession within a given transition horizon to cancel out. For example, repeating the experiment for a simulated dataset under Equation 4.5.5, we use the RWMH algorithm to estimate the parameters of the process in conjunction with a sequence of detection probabilities. Figure 4.5.6 shows the resulting sequence of detection probabilities for the simulated dataset under the parameter set $\boldsymbol{\theta} = \{\alpha_x, \beta_x, \kappa, \mu_z, \sigma_z\} = \{1, 5, 0.1, 0.2, 0.5, 0.5\}$ as well as the frequency distribution of the actual jump observations. Superimposed on the frequency distribution we indicate cumulative jump magnitudes for the transition horizons (the sum of all jumps within that transition horizon) for which jumps were left undetected (missed). As the figure suggests, those jumps which remain undetected appear to be of small magnitude, making them difficult to detect over the diffusion trajectory. Thus, despite having a detection rate of 84.6% (Table 4.5.3), jumps with small magnitudes will remain undetected. Indeed, by changing the jump distribution parameters to $\{\mu_z, \sigma_z\} = \{1, 0.25\}$ prior to running the simulation, we can achieve a 100% detection rate with zero false positives under the 80% decision rule. Alternatively, by changing the sample rate of the data from $t_{i+1} - t_i = 0.1$ to $t_{i+1} - t_i = 0.02$ on the same observation horizon, we can achieve an $\approx 0.98\%$ detection rate (a single missed detection) with zero false positives.

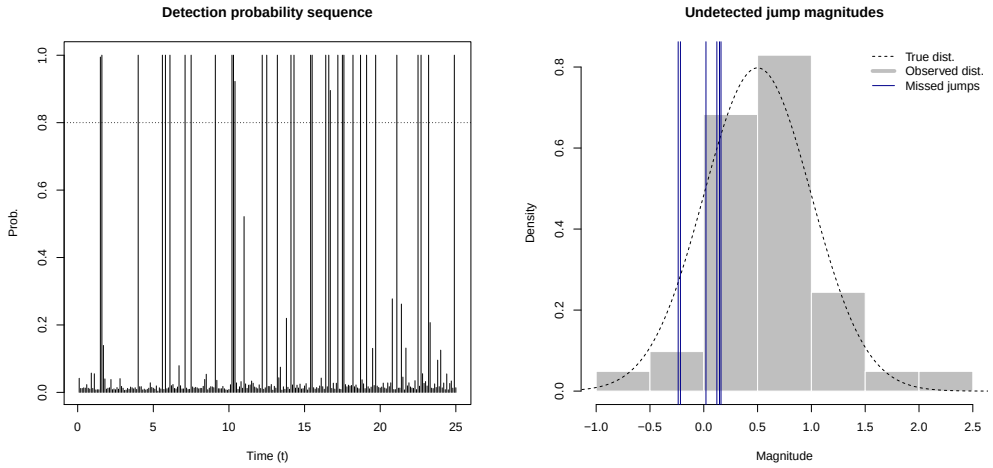


FIGURE 4.5.6: Estimated detection probabilities (left) with 80% decision threshold and undetected jump magnitudes relative to the true and observed jump distributions (right). Jumps which are left undetected have magnitudes quite close to the origin. R code: Supplementary materials, Section 4.13.

	Actual	
Predicted	≥ 1 Jumps	No jumps
≥ 1 Jumps	0.846	0.005
No jumps	0.154	0.995

TABLE 4.5.3: Contingency table for jump detection rates under a simulated dataset for Equation 4.5.5. The results indicate a 84.6% detection rate with 15.4% missed detections and a 0.5% false positive detection rate. R code: Supplementary materials, Section 4.13.

4.6 Application to Google stock price volatility

In a post sub-prime crisis world, investors have become increasingly aware of the importance of understanding the impact that large movements in equity values can have on portfolios and financial products. As such, techniques for analysing financial data have grown increasingly complex and often focus on better managing the risks and opportunities associated with extreme events, both at the high frequency and low-frequency trading spectrum. In conjunction with this, data markets have evolved similarly, with highly detailed data on thousands

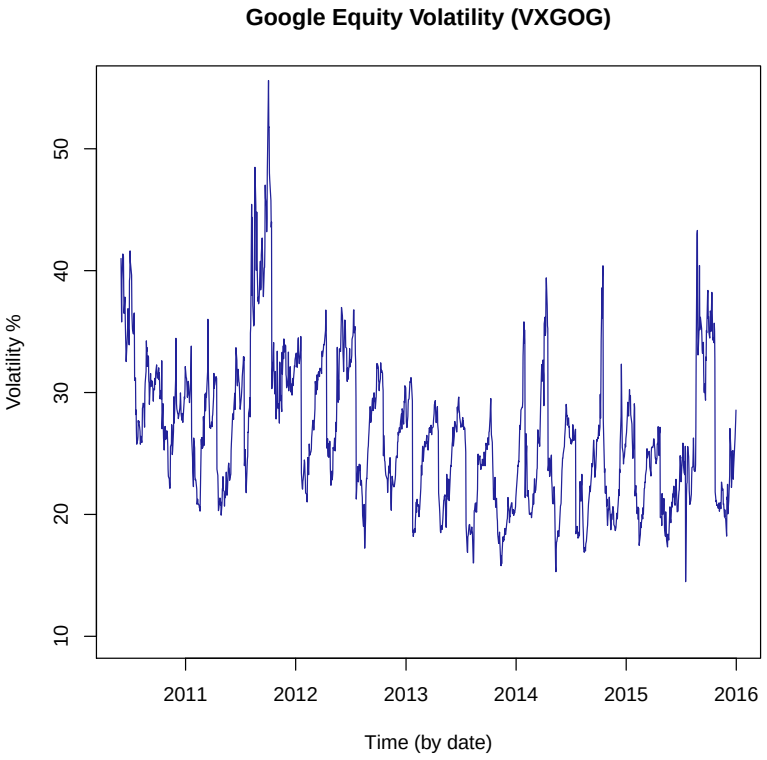


FIGURE 4.6.1: Daily equity volatility of Google shares for the period 2010-03-11 to 2016-01-01. R code: Supplementary materials, Section 4.14.

of economic variables and equities accessible at little to no cost. In keeping with this, the Chicago Board Options Exchange publishes volatility indexes for a number of large-cap stocks listed on major stock exchanges. By using the same principles that underlie established indices such as the S & P 500 volatility index, these equity volatility indices attempt to quantify the evolution of the volatility of individual stock price processes as opposed to that of an index of equities. Indeed, the dynamics of individual stock processes can be vastly different from that of an aggregated set of stocks. As such, equity volatility indices can be extremely useful in quantifying exposure in portfolios which have large investments in such equities and related processes. By using various jump diffusion models, we attempt to model the equity volatility of internet search giant Google. Figure 4.6.1 illustrates the trajectory of the Google equity volatility (VXGOG) from its inception in 2010 up to the end of 2015, sampled at daily intervals. For purposes of the analysis that follows, we measure time in years and use exact dates for observations in order to construct transition horizons for consecutive observations. In order to

model the volatility time series, we define a number of jump diffusion models nested within the SDE:

$$\begin{aligned} dX_t &= \mu_{\boldsymbol{\theta}}(X_t, t)dt + \sigma_{\boldsymbol{\theta}}(X_t, t)dB_t + dP_t \\ dP_t &= J(X_t, \dot{z}_t)dN_t \end{aligned} \tag{4.6.1}$$

with jump intensity $\lambda_{\boldsymbol{\theta}}(X_t, t)$, that aim to replicate the salient features and dynamics of the volatility series. Using the generalised quadratic framework of Section 4.3.2, we can formulate a template for Equation 4.6.1 that can be used to fit various forms of drift, diffusion, and jump specifications. For purposes of modelling the drift of the volatility series, we make use of linear mean reverting drift structures of the form:

$$\mu_{\boldsymbol{\theta}}(X_t, t) = \alpha(\beta + h(t, \boldsymbol{\theta}) - X_t). \tag{4.6.2}$$

Here, $\beta + h(t, \boldsymbol{\theta})$ represents a possibly fluctuating level to which the process reverts over time. Using this formulation we can recover standard mean reversion structures such as the CIR and Ornstein-Uhlenbeck models by setting $h(t, \boldsymbol{\theta}) = 0$ and test for the presence of volatility cycles by replacing $h(t, \boldsymbol{\theta})$ with a periodic function. For example in this case, we let

$$h(t, \nu_1, \nu_2) = \nu_1 \sin\big(8\pi(t + 0.25(\nu_2 - 0.5))\big) \tag{4.6.3}$$

for $\nu_1 > 0$ and $\nu_2 \in [0, 1]$, which specifies a quarterly volatility cycle. Due to the mean reverting structure, the model process will subsequently have long-run mean dynamics that mimic the behaviour of the term $\beta + h(t, \nu_1, \nu_2)$. For purposes of modelling the diffusion dynamics of the volatility series we assume various forms for the diffusion coefficient where

$$\sigma_{\boldsymbol{\theta}}(X_t, t) = \begin{cases} \sigma & \text{for constant volatility,} \\ \sigma\sqrt{X_t} & \text{for linear instantaneous variance,} \\ \sigma X_t & \text{for quadratic instantaneous variance.} \end{cases} \tag{4.6.4}$$

Similarly, in order to model the jump dynamics of the process, we alternate between combinations of constant and state-dependent coefficients for the jump mechanism where the intensity coefficient is defined as

$$\lambda_{\boldsymbol{\theta}}(X_t, t) = \begin{cases} \kappa & \text{for constant intensity,} \\ \kappa X_t & \text{for relative intensity,} \end{cases} \tag{4.6.5}$$

Mod.	$\mu(X_t,t)$	$\sigma(X_t,t)$	$\lambda(X_t,t)$	$J(X_t,\dot{z}_t)$	$\dot{z}_t \sim$	DIC	p_D
1	$\alpha(\beta - X_t)$	σ	$\cdot$	$\cdot$	$\cdot$	5776.10	2.77
2	$\alpha(\beta - X_t)$	$\sigma\sqrt{X_t}$	$\cdot$	$\cdot$	$\cdot$	5627.37	2.99
3	$\alpha(\beta - X_t)$	σX_t	$\cdot$	$\cdot$	$\cdot$	5482.72	3.22
4	$\alpha(\beta - X_t)$	$\sigma\sqrt{X_t}$	κ	$\dot{z}_t$	$\mathrm{N}(\mu_z,\sigma_z^2)$	4832.27	6.28
5	$\alpha(\beta - X_t)$	σX_t	κ	$\dot{z}_t$	$\mathrm{N}(\mu_z,\sigma_z^2)$	4798.80	5.69
6	$\alpha(\beta - X_t)$	σX_t	κX_t	$\dot{z}_t$	$\mathrm{N}(\mu_z,\sigma_z^2)$	4784.03	5.61
7	$\alpha(\beta - X_t)$	σX_t	κ	$\dot{z}_t X_t$	$\mathrm{N}(\mu_z,\sigma_z^2)$	4778.91	6.16
8	$\alpha(\beta + h(t,\nu_1,\nu_2) - X_t)$	σX_t	κ	$\dot{z}_t$	$\mathrm{N}(\mu_z,\sigma_z^2)$	4767.67	8.35
9	$\alpha(\beta + h(t,\nu_1,\nu_2) - X_t)$	σX_t	κX_t	$\dot{z}_t$	$\mathrm{N}(\mu_z,\sigma_z^2)$	4752.46	8.30
10	$\alpha(\beta + h(t,\nu_1,\nu_2) - X_t)$	σX_t	κ	$\dot{z}_t X_t$	$\mathrm{N}(\mu_z,\sigma_z^2)$	**4744.02**	7.54

TABLE 4.6.1: Various drift, diffusion, and jump mechanism specifications for the Google equity volatility series. Approximate DIC and fitted effective number of parameters (p_D) are also tabulated for each model (minimum DIC value indicated in bold). The results suggest that the observed series exhibits time-inhomogeneous linear drift with state-dependent volatility. In addition, there is evidence to suggest state-dependence within the jump mechanism of the process. R code: Supplementary materials, Section 4.14.

and the jump coefficient assumes the form

$$J(X_t,\dot{z}_t) = \begin{cases} \dot{z}_t & \text{for constant jump size,} \\ \dot{z}_t X_t & \text{for relative jump size,} \end{cases} \tag{4.6.6}$$

and it is assumed that jumps are normally distributed i.e.,

$$\phi(\dot{z}_t,\mu_z,\sigma_z^2) = \mathrm{N}(\mu_z,\sigma_z^2) = \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\left(- \frac{(\dot{z}_t - \mu_z)^2}{2\sigma_z^2} \right). \tag{4.6.7}$$

Table 4.6.1 gives various forms of drift, diffusion and jump mechanisms fitted to the VXGOG series. Using the methodology of sections 4.3 and 4.5 in conjunction with the random walk Metropolis-Hastings algorithm, we are able to efficiently calculate parameter estimates and deviance information criterion statistics for the various model specifications. For each case we place prior distributions on the parameters of the model as indicated in Table 4.6.2. For reference, we also include a set of jump-free models that fit within the forgoing model assumptions. Corresponding parameter estimates and 90% credibility intervals are given in tables 4.6.3 – 4.6.5.

Although the model space presented here is by no means exhaustive (the set of models considered here represent the best performing models among a number

Parameter	Prior distribution
α	Gamma$(0.001, 0.001)$
β	Normal$(25, 5^2)$
σ^2	Inv-Gamma$(0.001, 0.001)$
κ	Gamma$(0.001, 0.001)$
ν_1	Gamma$(0.001, 0.001)$
ν_2	Beta$(0.5, 0.5)$

TABLE 4.6.2: Prior distributions on the parameter space. Where the relevant terms are included in a given model, the corresponding prior distributions are applied. Although the prior distributions used here are mostly weakly informative, the prior distributions on β and ν_2 follow from basic inspection of the time series.

	Model 1		Model 2		Model 3		Model 4	
	Est.	90%CI	Est.	90%CI	Est.	90%CI	Est.	90%CI
α	15.07	(10.63, 19.26)	15.30	(11.30, 19.07)	13.36	(9.82, 17.28)	7.40	(5.10, 9.66)
β	26.40	(24.88, 27.83)	26.50	(25.01, 27.96)	27.04	(25.58, 28.79)	27.49	(25.49, 29.42)
σ	32.77	(31.80, 33.63)	6.10	(5.91, 6.30)	1.12	(1.09, 1.16)	3.38	(3.20, 3.54)
κ	·	·	·	·	·	·	23.97	(18.94, 30.72)
μ_z	·	·	·	·	·	·	-0.42	(-1.20, 0.33)
σ_z	·	·	·	·	·	·	4.90	(4.28, 5.67)

TABLE 4.6.3: Parameter estimates and 90% credibility intervals for models 1 to 4. Estimates for each model are calculated from 110 000 random walk Metropolis-Hastings updates with a burn-in period of 10 000 iterations. R code: Supplementary materials, Section 4.14.

	Model 5		Model 6		Model 7		Model 8	
	Est.	90%CI	Est.	90%CI	Est.	90%CI	Est.	90%CI
α	6.55	(4.24, 8.92)	6.30	(3.57, 9.31)	6.44	(4.03, 8.89)	10.69	(7.25, 13.75)
β	27.95	(25.76, 30.56)	27.97	(25.73, 30.88)	27.71	(25.39, 30.41)	26.97	(25.70, 28.43)
σ	0.67	(0.64, 0.70)	0.66	(0.64, 0.69)	0.65	(0.61, 0.68)	0.66	(0.63, 0.70)
κ	20.47	(14.66, 26.89)	0.85	(0.65, 1.09)	28.60	(21.37, 37.88)	21.52	(15.33, 27.96)
μ_z	-0.50	(-1.39, 0.37)	-0.53	(-1.43, 0.29)	-0.01	(-0.03, 0.01)	-0.35	(-1.26, 0.41)
σ_z	5.17	(4.39, 6.10)	4.93	(4.23, 5.81)	0.15	(0.13, 0.17)	5.03	(4.34, 5.77)
ν_1	·	·	·	·	·	·	6.70	(4.89, 8.82)
ν_2	·	·	·	·	·	·	0.56	(0.52, 0.62)

TABLE 4.6.4: Parameter estimates and 90% credibility intervals for models 5 to 8. Estimates for each model are calculated from 110 000 random walk Metropolis-Hastings updates with a burn-in period of 10 000 iterations. R code: Supplementary materials, Section 4.14.

	Model 9		Model 10	
	Est.	90%CI	Est.	90%CI
α	10.28	(6.84, 13.46)	10.71	(7.76, 14.09)
β	27.03	(25.71, 28.66)	26.72	(25.43, 28.03)
σ	0.66	(0.63, 0.69)	0.64	(0.61, 0.68)
κ	0.88	(0.63, 1.17)	28.09	(20.78, 35.40)
μ_z	-0.31	(-1.15, 0.43)	0.00	(-0.03, 0.02)
σ_z	4.89	(4.13, 5.81)	0.15	(0.13, 0.17)
ν_1	6.74	(4.77, 9.09)	6.53	(4.79, 8.70)
ν_2	0.56	(0.52, 0.61)	0.56	(0.51, 0.61)

TABLE 4.6.5: Parameter estimates and 90% credibility intervals for models 9 to 10. Estimates for each model are calculated from 110 000 random walk Metropolis-Hastings updates with a burn-in period of 10 000 iterations. R code: Supplementary materials, Section 4.14.

of additional specifications including quadratic drift models and iterations of the models analysed here with Laplace distributed jumps), the models serve as a basis for testing a number of hypotheses with regard to the volatility series. Using the approximate DIC values as a guide for comparing the various model specifications, we can identify which elements improve model fit and thus which elements most accurately replicate the observed dynamics. Within the jump-free model set, the fit is improved for models with diffusion coefficients which are more sensitive to changes in the state of the process. This suggests that the volatility of the volatility process increases in accordance with the level of the process. Perhaps the greatest improvement in model fit stems from including a jump mechanism in the diffusion model. Although this comes at the cost of three additional parameters, the jump diffusion models fair significantly better than their corresponding jump-free models. Despite the addition of the jump mechanism, model fit is still improved for models with greater state-dependence in volatility. With respect to the jump mechanism itself, there is evidence that jump magnitudes may indeed vary in accordance to the level of the process. Finally, by including a quarterly drift cycle we can further improve model fit.

To conclude the analysis we consider the jump detection probabilities for the volatility dataset. Although we cannot know what the true data generating process looks like, we attempt to decode a sequence of latent jumps under the model approximation and compare it to the observed dataset for reference. We may also compare the detection probabilities as calculated under the various model specifications. Despite the varying model specifications, the estimated detection probabilities are quite close throughout the model space with 80% of detections at the 80% decision rule being common for all of the jump models (i.e., 80% of the jumps were detected under all jump diffusion model specifications)

and 96% of detections being common for models 8-10. This suggests that the data resolution is sufficiently high in order to distinguish between transitions which are affected by jumps and transitions which are not. That said, given that the jump distribution is estimated to be centred close to the origin, we can expect that a number of jump events may have gone undetected. Based on the specification of Model 10, we apply the 80% decision rule to the estimated jump detection probability sequence (Figure 4.6.2) and superimpose the resulting detection times onto the observed time series (Figure 4.6.3). Whilst analysing the data, we had strong suspicions that the cyclical properties of the data were caused (in part) by the release of quarterly earnings reports. Indeed, when superimposing the dates of these reports for the sample period we find that all but one (Q3 earnings release for 2014 – with an estimated detection probability of 47% on the release date under Model 10) of the release dates correspond to jump detections made under the 80% rule. Although earnings report jumps account for only 37.5% of detected jumps under the assumed model and decision rule, the results nevertheless provide an interesting insight into the effect that information has on the volatility of the underlying stock price process. Indeed, further investigation may reveal other significant news and/or financial events that correlate with the estimated jump detection sequence.

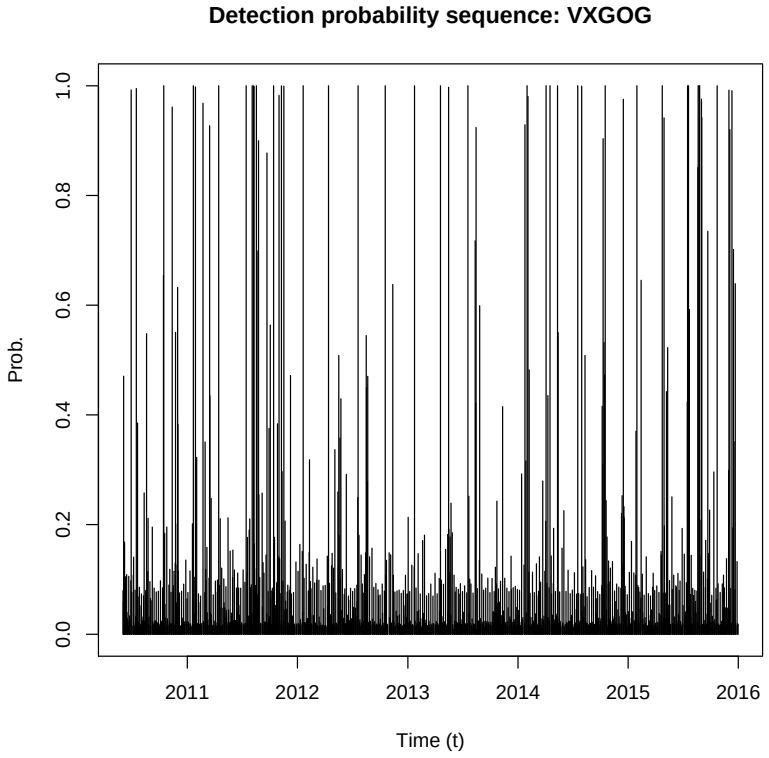


FIGURE 4.6.2: Estimated detection probability sequence under Model 10 for the VXGOG dataset. R code: Supplementary materials, Section 4.14.

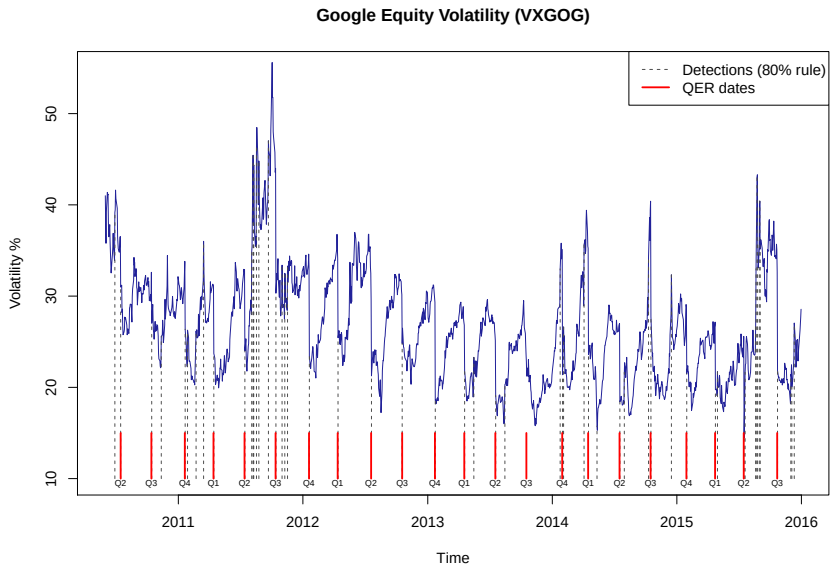


FIGURE 4.6.3: Jump detections calculated by applying an 80% decision rule to the estimated jump detection probability sequence. Quarterly earnings release (QER) dates superimposed in red. R code: Supplementary materials, Section 4.14.

4.7 Theoretical applications and extensions

As evidenced by the analysis of the preceding sections, one can use the moment equations in conjunction with the mixture factorization to analyse quite complex jump diffusion models of real-world phenomena. Because of the flexibility that the methodology affords, we may test various theories about observed processes, such as whether or not a process exhibits time-inhomogeneous and/or jump dynamics. Although the motivation behind the methodology has largely been to provide an efficient scheme that balances generality with ease of application, the scheme is still sufficiently general to allow one to propose new theories of how processes evolve. That is, by postulating models that have not been analysed before due to the absence of analytically tractable solutions, we may gain insight into new avenues of research or possible extensions for existing research. In this section, we propose a number of models that can be used to answer more fundamental questions about the dynamics of a given process.

4.7.1 Jump-reversion and parity to continuous reversion.

The Ornstein-Uhlenbeck and Cox-Ingersoll-Ross models have been a staple of financial engineering for quite some time. The primary strengths of these models are that they allow one to incorporate mean reverting dynamics and test for state-dependence in the volatility of a process whilst having the attractive mathematical property that the transitional density can be calculated in closed-form without much effort. Furthermore, the components of the model are easy to interpret, both heuristically and within formal mathematical contexts. For example, consider the CIR process:

$$dX_t = \kappa(\beta - X_t)dt + \sigma\sqrt{X_t}dB_t. \tag{4.7.1}$$

Under the dynamics of Equation 4.7.1, the process reverts to the level β over time. The force of the reversion or 'speed' is then dictated by the coefficient κ, which by way of scaling the drift in terms of the distance of the process from the level β acts against stochastic changes in the state of the process brought on by the diffusion term through the Brownian motion component. The premise of mean reversion in this context is that the process corrects itself over time as it moves away from the long-run mean, β. Moreover, since the reversion mechanism operates through the drift of the process, it does so continuously in time. A natural question to ask is: Why should the mean reverting mechanism depend entirely on how far the process has moved away from β. Furthermore, why should the reversion mechanism operate continuously? For example, in a real-world context such as finance, although a process may exhibit mean reversion, it is entirely plausible that some time my pass before a 'correction' ensues. For

these purposes, we propose a more general formulation of the mean reversion mechanism in the context of diffusion processes by making use of a special type of jump diffusion model. That is, we define a *jump-reversion* model:

$$dX_t = \dot{z}_t(\beta - X_t)dN_t + \sigma(X_t, t)dB_t, \tag{4.7.2}$$

with intensity $\lambda(X_t, \dot{r}_t, t)$ for $\dot{r}_t \sim \pi$ and jump distribution $\dot{z}_t \sim \phi$. Under Equation 4.7.2, the reversion mechanism operates through discontinuous jumps that arise at a rate given by the intensity coefficient. As such, the reversion mechanism can model corrections of varying sizes that operate independently of the Brownian motion component of the model. That is, in models such as the CIR process, the instantaneous reversion force is completely determined by how far innovations in the Brownian motion cause the process to deviate from β, whilst for Equation 4.7.2 reversion is determined both by innovations in the Brownian component and the jumps which are incurred. Depending on the nature of the jumps, the process may correct fractionally if jumps are of magnitude less than one, or over correct if jumps are of magnitude greater than one. It is even possible to formulate a type of renewal process by setting the jump distribution to that of a Dirac-delta function centred at one i.e., $\phi = \delta(\dot{z}_t - 1)$, thus implying that jump corrections will be exactly the magnitude that the process has deviated from β. As such, when corrections occur, the process will restart from β. Figure 4.7.1 illustrates a simulated trajectory for a time-homogeneous diffusion of the form of Equation 4.7.2 with corresponding jump realisations. From the simulated trajectory, the reversion mechanism can be seen to operate by correcting for deviations from the long-run mean with jumps in the direction of the long-run mean. At the outset, when the process is far away from the equilibrium line, $\beta = 5$, the jump corrections are large and decrease as the process trajectory moves closer to the equilibrium line. Furthermore, under the assumed parameter set the sign of the jumps are dictated by which side of the equilibrium line the process lies, assuming positive values when the process is below the equilibrium line and negative values when above.

Although the process appears superficially very similar to continuous mean reversion, the principal difference can be seen in the local characteristics of the transitional density, where the jump-reversion mechanism causes more rapid probability flow in the direction of the long-run mean. This can be visualised by investigating the transitional density on short time scales. Figure 4.7.2 compares the transitional density for a jump-reversion model with $\beta = 5.0$, $\sigma(X_t, t) = 0.25(2.0 + 0.8\sin(0.5\pi t))$, $\lambda(X_t, t) = 2.0$, and $\dot{z}_t \sim \mathrm{N}(1.0, 0.25)$, to its continuous reversion counterpart with $\mu(X_t, t) = 1 \times (\beta - X_t)$. Although the transitional densities differ characteristically over short transition horizons, it can

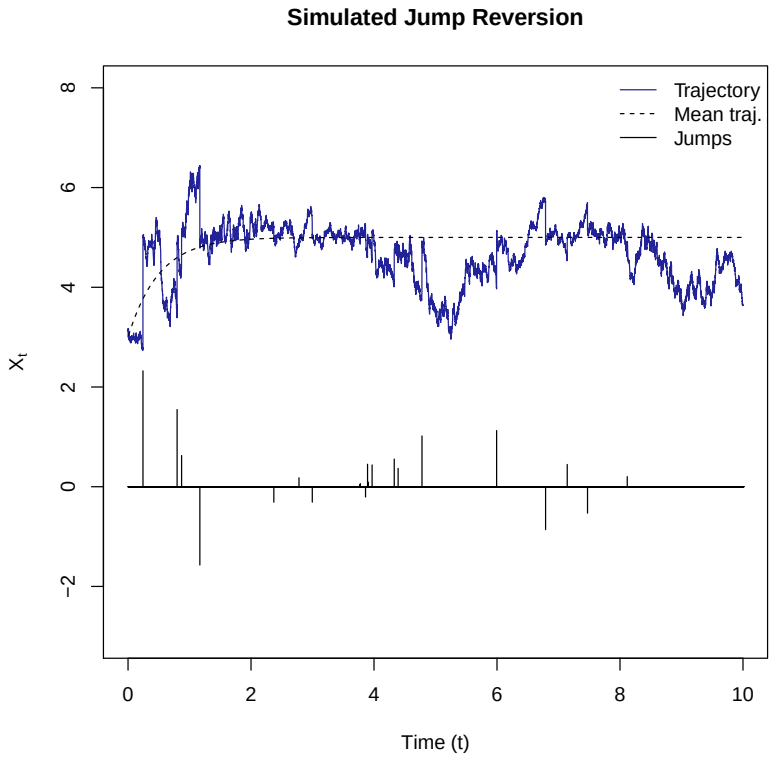


FIGURE 4.7.1: Simulated trajectory of a jump-reversion model with $\sigma(X_t, t) = \sigma\sqrt{X_t}$, $\lambda(X_t, t) = \lambda_0$, $\phi = \mathrm{N}(\mu_{\dot{z}}, \sigma_{\dot{z}}^2)$ and parameters $\{\beta, \sigma, \lambda_0, \mu_{\dot{z}}, \sigma_{\dot{z}}^2\} = \{5.0, 0.25, 2.0, 1.0, 0.25\}$. Mean trajectory (black, dashed) and jump corrections (black, solid) indicated. R code: Supplementary materials, Section 4.15.

be seen that the models behave similarly on long time scales with the transitional densities matching closely over sufficiently large transition horizons.

Citing the relationship between the local and long-run dynamics of jump and continuous reversion models, it can be argued that some parity should exist between the continuous and jump-reversion regimes. Indeed, although the reversion mechanism of Equation 4.7.2 behaves as a discontinuous process, it is still possible to mimic the continuous drift of the CIR and Ornstein-Uhlenbeck models using a jump-reversion model. This is achieved by assuming that very small jumps of nearly predictable size occur at a very high rate. For example, setting:

$$\lambda(X_t, t) = \frac{\kappa}{\delta} \qquad \dot{z}_t \sim \mathrm{U}(0, 2\delta), \tag{4.7.3}$$

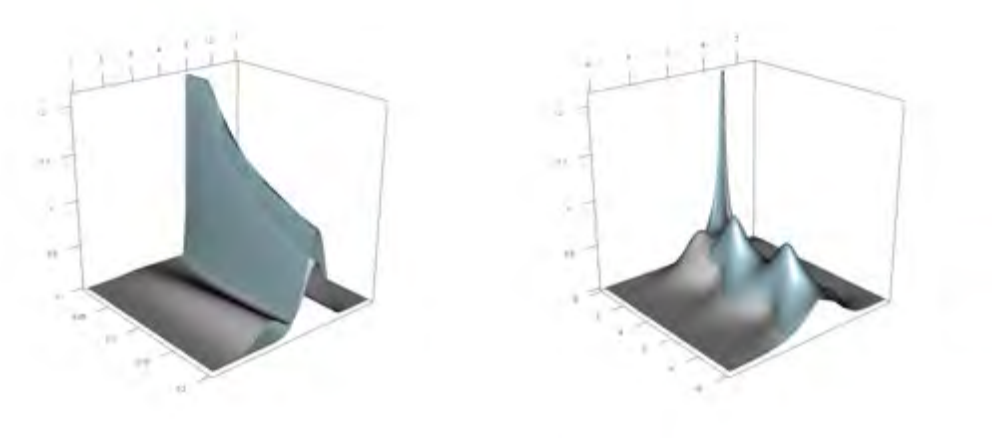


FIGURE 4.7.2: Approximate transition density of a jump recersion model (gray) compared to its continuous reversion counterpart (light blue) on short (left) and long (right) transition horizons. R code: Supplementary materials, Section 4.15.

and letting $\delta \to 0$ we can closely replicate continuous mean reversion. Figure 4.7.3 illustrates this parity by comparing the moments of a jump-reversion model under a high frequency, small jump regime to a continuous reversion counter part. That is, let:

$$dS_t = \kappa(\beta - S_t)dt + \sigma S_t dB_t, \tag{4.7.4}$$

and

$$dX_t = \dot{z}_t(\beta - X_t)dN_t + \sigma X_t dB_t, \tag{4.7.5}$$

with a jump mechanism characterised by Equation 4.7.3. Figure 4.7.3 compares the non-central moments of equations 4.7.4 and 4.7.5 for decreasing values of δ. As δ approaches zero, the moments of the jump-reversion model approach that of the continuous reversion model, illustrating the parity that exists between the two reversion regimes.

Indeed, one can formulate a mathematical argument for the origin of this parity. Consider the following: By deriving a PDDE for moment generating function for Equation 4.7.5

$$\frac{\partial}{\partial t}M(\alpha,t) = \frac{\sigma^2}{2}\frac{\partial^2}{\partial \alpha^2}M(\alpha,t) + \frac{\kappa}{\delta}\Big(M^{\star}(\alpha,t) - M(\alpha,t)\Big), \tag{4.7.6}$$

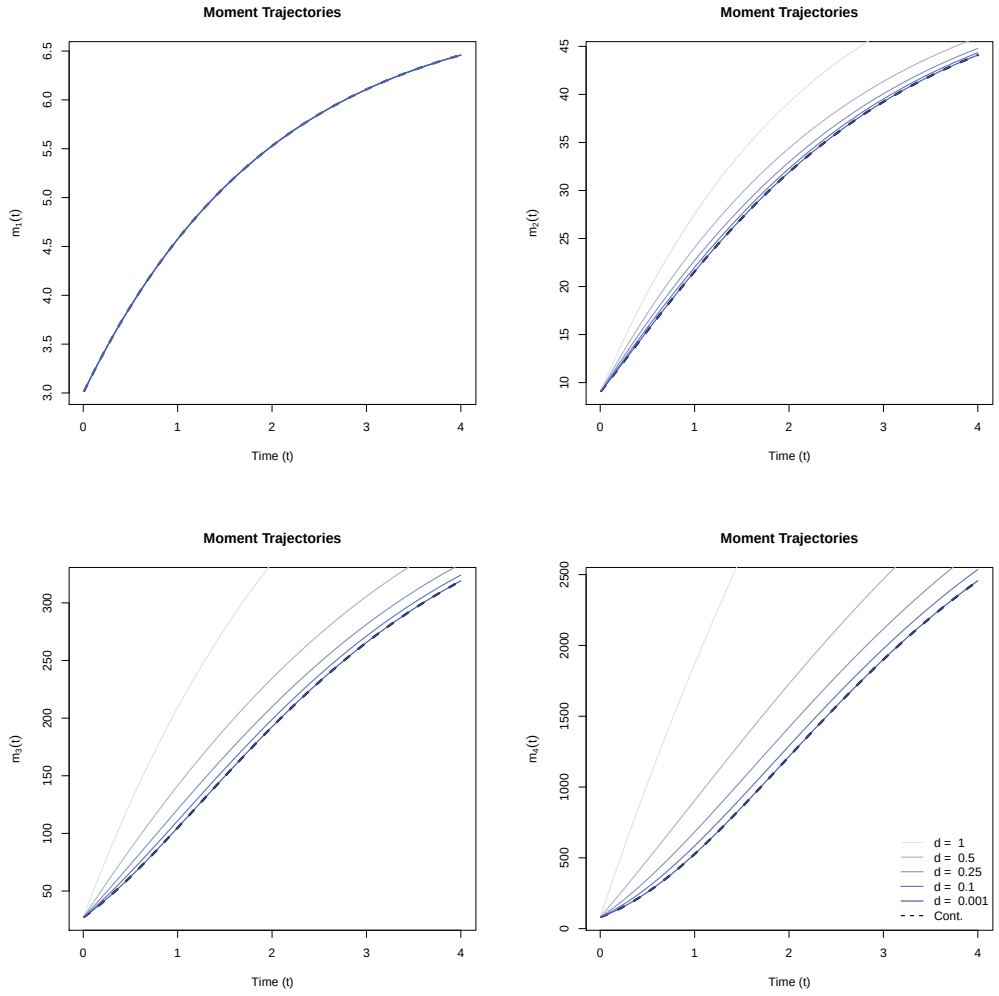


FIGURE 4.7.3: Non-central moments of Equation 4.7.4 (black, dashed) and Equation 4.7.5 for decreasing values of the parameter δ (blue, light - dark). R code: Supplementary materials, Section 4.15.

and replacing $M(\alpha, t)$ and $M^\star(\alpha, t)$ by their respective series representations where $M(\alpha, t) = E_X[\sum_{i=0}^{\infty} \alpha^i / i! X_t^i]$ and

$$\begin{aligned} M^\star(\alpha, t) &= E_{X,\dot{z}}\left[\sum_{i=0}^{\infty} \frac{\alpha^i}{i!}(X_t + \dot{z}_t(\beta - X_t))^i\right] \\ &= E_{X,\dot{z}}\left[\sum_{i=0}^{\infty} \frac{\alpha^i}{i!} \sum_{r=0}^{i} \binom{i}{r} (\beta - X_t)^r X_t^{i-r} \dot{z}_t^r\right], \end{aligned} \tag{4.7.7}$$

we can derive a system of ODEs for the moment equations:

$$\begin{aligned} m_i'(t) &= \sigma^2 \frac{i(i-1)}{2} m_i(t) \mathbb{I}_{i\geq 2} + \frac{\kappa}{\delta} E_{X,\dot{z}}\left[\sum_{r=0}^{i} \binom{i}{r} (\beta - X_t)^r X_t^{i-r} \dot{z}_t^r - X_t^i\right] \\ &= \sigma^2 \frac{i(i-1)}{2} m_i(t) \mathbb{I}_{i\geq 2} + \frac{\kappa}{\delta} E_X\left[\sum_{r=0}^{i} \binom{i}{r} (\beta - X_t)^r X_t^{i-r} \frac{(2\delta)^r}{r+1} - X_t^i\right] \\ &= \sigma^2 \frac{i(i-1)}{2} m_i(t) \mathbb{I}_{i\geq 2} + \kappa E_X\left[\sum_{r=0}^{i} \binom{i}{r} (\beta - X_t)^r X_t^{i-r} \frac{(2)^r \delta^{r-1}}{r+1} - X_t^i\right] \\ &= \sigma^2 \frac{i(i-1)}{2} m_i(t) \mathbb{I}_{i\geq 2} + \kappa E_X\left[\binom{i}{1} (\beta - X_t) X_t^{i-1}\right] \\ &\quad + \kappa E_X\left[\sum_{r=2}^{i} \binom{i}{r} (\beta - X_t)^r X_t^{i-r} \frac{(2)^r \delta^{r-1}}{r+1}\right]. \end{aligned} \tag{4.7.8}$$

Applying the limit as $\delta \to 0$, we derive the system:

$$m_i'(t) = \sigma^2 \frac{i(i-1)}{2} m_i(t) \mathbb{I}_{i\geq 2} + \kappa\beta i m_{i-1}(t) - \kappa i m_i(t), \tag{4.7.9}$$

subject to the initial conditions $m_i(s) = X_s^i$ for all $i = 1, 2, \ldots$, which matches exactly the moment equations of Equation 4.7.4 (see for example Equation 4.3.27). Since this holds for all moments of the process, the distributional parity is established by the uniqueness of the moment generating function.

4.7.2 Wright-Fisher diffusion with shocks

The Wright-Fisher process is often used in genetics to describe changes in gene frequencies in a population (see Bollback *et al.* (2008) for a concrete example). The model describes the process by which offspring inherit gene variants from their parents through a stochastic mechanism. The process is based on the idea

that gene variants are 'sampled' at random from the parent population, and thus the mechanism by which the frequency of a given gene type changes is formulated as a simple, discrete stochastic process. Following Durrett (2008) and Karlin and Taylor (1981), the process can be described in discrete terms as follows: Consider a diploid population of size N. Each individual is assumed to have one of two variants/allele of a given gene, say type A and type B. Let X_t denote the number of alleles of type A at time t and $N - X_t$ the number alleles of type B. Assuming that at each generation gene variants are sampled randomly, the evolution of the frequency of a given gene variant can be formulated as a stochastic process. For example, given a frequency X_t/N of type A at time t, the change in the frequency distribution over a single generation, can be described by the process:

$$P(X_{t+1} = n) = \binom{N}{n} \left(\frac{X_t}{N}\right)^n \left(1 - \frac{X_t}{N}\right)^{N-n} \tag{4.7.10}$$

for $n = 0, 1, \ldots, N$. Since the sampling structure depends only on the proportion of each allele in the total population at time t, the process describes a neutral genetic drift process. A natural extension of the model is to assume that at each update of the process the probability of choosing a given allele is either more or less likely. That is, by modifying the sampling probabilities one can extend the model to allow for situations where a given gene type selected more frequently (or *vice versa*). This results in a non-zero genetic drift effect, and can be incorporated in the sampling structure by increasing (decreasing) the relative probability with which allele types is selected (Karlin and Taylor, 1981). That is, for each individual over a single generation we have:

$$\frac{P(\text{Indiv.} = \text{type A})}{P(\text{Indiv.} = \text{type B})} = \left(1 + \frac{s}{N}\right) \tag{4.7.11}$$

where s denotes a 'selection' coefficient. That is, for $s > 0$, traits of type A are selected in greater proportion and less so for $s < 0$.

When N becomes large, analysis of such a model becomes difficult. As such, the model is often approximated by a diffusion process (Norman, 1975; Ethier and Norman, 1977). This is achieved by changing the unit of time measurement to generational units and subsequently letting $N \to \infty$. The diffusion approximation is then governed by the stochastic differential equation:

$$dX_t = sX_t(1 - X_t)dt + \sqrt{X_t(1 - X_t)}dB_t \tag{4.7.12}$$

where X_t describes the frequency of type A alleles over time where $t = 1 = N$ generations.

Although the Wright-Fisher diffusion is assumed to be time-homogeneous, one may easily incorporate time-inhomogeneous structures within the model. Although this makes for an interesting generalisation of the model, we may perhaps extend the model by making a more fundamental modification. For example, one of the deficiencies of the model is the assumption that the model process is defined as a closed system, and thus any random innovation in the frequency distribution is the result of the evolutionary mechanism alone. In order to address this, we may propose including external forces which act upon the frequency trajectory. One such mechanism would be to include randomly occurring frequency shocks, caused for example by changes in the population size or invasions where invading individuals have a different frequency distribution than the population in question. Consider, for example, a drift neutral process and assume that at any given transition between generations a frequency shock may occur and is distributed according to the distribution $\phi(t)$. Furthermore, assume that the rate at which these shocks occur is λ per N generations (i.e., shocks are expected to occur once every N/λ generations). The shocks then manifest as 'jumps' in the trajectory of the frequency process. For example, Figure 4.7.4 compares simulated trajectories for a standard drift neutral Wright-Fisher process and a drift neutral process subject to shocks (see Appendix D.8 for details on the simulation of Wright-Fisher process). From the simulated trajectories, it is clear that allowing for jumps in the frequency trajectory can have a significant effect on the frequency distribution over time.

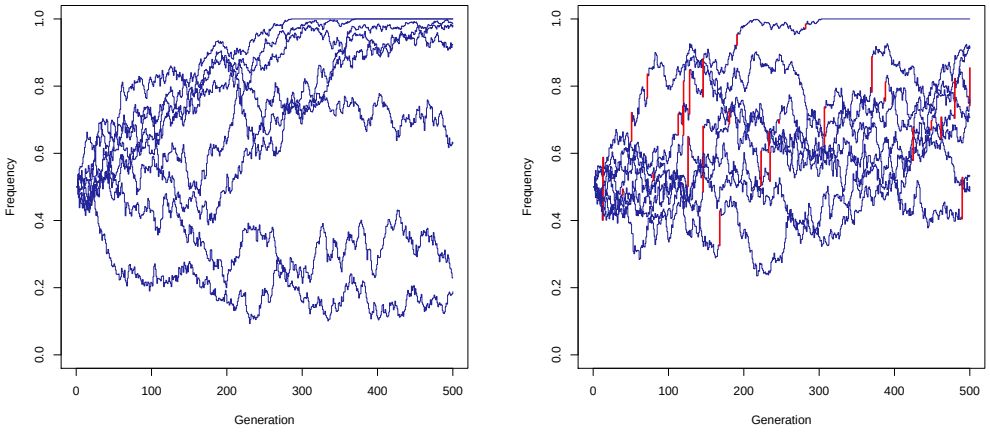


FIGURE 4.7.4: Simulated trajectories (discrete time) of a drift free Wright-Fisher process ($s = 0.0$) over five hundred generations for a population of $N = 1000$ with (right, indicated in red) and without (left) frequency shocks. R code: Supplementary materials, Section 4.16.

As in the case of the standard Wright-Fisher process, it is possible to formulate a diffusion approximation to the Wright-Fisher process with shocks. This is achieved by making use of a non-linear jump diffusion process. For example, consider a Wright-Fisher process subject to frequency shocks occurring at a constant rate. Furthermore, assume that shocks, when they occur may increase the frequency of type A alleles up to halfway to fixation. Then the process may be approximated by jump diffusion process of the form:

$$dX_t = sX_t(1 - X_t)dt + \sqrt{X_t(1 - X_t)}dB_t + \dot{z}_t(1 - X_t)dN_t \tag{4.7.13}$$

with intensity λ and $\dot{z}_t \sim \mathrm{U}(0, 0.5)$. Under this formulation, it is assumed that the size of frequency shocks diminish as the frequency increases. Depending on the application, the validity of this formulation can be debated, but this particular structure may be valid when for example the event that triggers a sudden change in frequency applies to alleles of type B only – of which there would be less as the frequency increases – or when shocks are purely the result of invasions, in which case the population increases and the effect of subsequent invasions diminish[3].

To illustrate the principle, consider a population of $N = 1000$ alleles of which the initial population consists of five hundred alleles of type A and five hundred alleles of type B respectively. For purposes of the experiment we shall also assume that the population is subject to a selection coefficient of $s = 0.01$ and that frequency shocks occur on average once every two hundred generations. In terms of Equation 4.7.13, this corresponds to an initial value of $X_0 = 0.5$, $s = 0.01$ and jump intensity of $\lambda = 5$. In order to verify that the jump diffusion approximation holds, we repeatedly simulate frequency trajectories for the Wright-Fisher process with shocks under the discrete model for a given number of generations and subsequently calculate the distribution of the simulated trajectories over time. Then, under the jump diffusion approximation, we can calculate the transitional density and compare it to the simulated frequency distribution. For example, Figure 4.7.5 compares the distribution of the simulated Wright-Fisher process with shocks for increasing numbers of generations to the transition density of Equation 4.7.13 under the corresponding parameters. Since the diffusion approximation measures time relative to the population size, the simulated distribution at say five hundred generations corresponds to the transitional density at $t = 0.5$. For reference, we also compare the corresponding Wright-Fisher diffusion without shocks under the same parameter set.

[3]In this case, care needs to be taken in interpreting the coefficients with respect to the discrete process as the time unit for the discrete model would change in accordance with changes in the population size

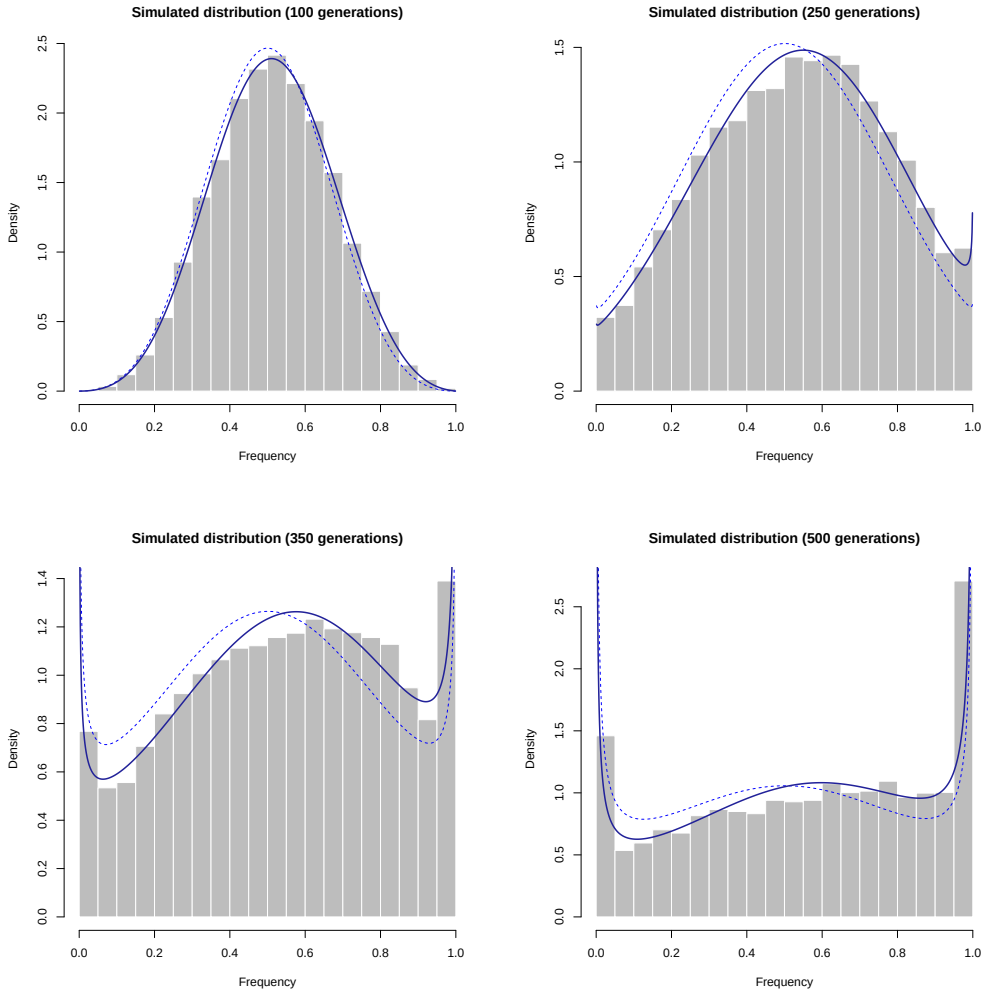


FIGURE 4.7.5: Frequency distribution of a simulated Wright-Fisher process with shocks for an increasing number of generations. The distribution is calculated using 10 000 simulated trajectories for a population of $N = 1000$. Superimposed is the approximate transition density for the jump diffusion approximation to the process (dark blue, solid). For comparison, we also include the shock-free process transition density (blue, dashed). R code: Supplementary materials, Section 4.16.

In order to calculate transitional density under the jump diffusion approximation, we derive the moment equations of Equation 4.7.13 up to an eighth-order truncation and evaluate the transitional density under the mixture factorization derived in Section 4.3.4. Since the process is restricted to lie on the interval

$[0, 1]$, we use the eighth-order Beta-type density from the multimodal Pearson system (Equation 2.3.28) to approximate both the jump-free distribution and the excess distribution under the mixture factorization. Indeed, the jump diffusion approximation accurately replicates the gene frequency distribution over the entire transition horizon. Interestingly, when compared to the jump-free diffusion approximation, the effect of the frequency shocks can clearly be seen in the skewness of the transitional density.

4.8 Software: The DiffusionRjgqd package

By extending the generalised quadratic class of diffusions to include jumps, it is possible to analyse complicated non-linear jump diffusion models with a high degree of accuracy. One of the benefits of developing the methodology on the basis of the generalised quadratic framework is that most of the computational elements of the jump-free GQDs under the cumulant truncation procedure are inherited by the jump-GQD class. That is, although the methodology focuses on the moment equations as opposed to the cumulant equations, and the mixture factorization on which the density approximations rely require that one evaluate both the jump diffusion and jump-free counterpart moments, the structure of the routines required to analyse such models are similar to those used in the analysis of jump-free GQDs. Thus, using routines from the **DiffusionRgqd** package as a template, we develop a new R package, **DiffusionRjgqd** (Pienaar and Varughese, 2015c) which provides routines for conducting inference and analysis on generalised quadratic jump diffusions.

Although the **DiffusionRjgqd** package shares most of its architecture with the **DiffusionRgqd** package, the workflow for routines contained in the package differs somewhat from that of sister-routines in the **DiffusionRgqd** package. In the **DiffusionRgqd** package, computationally optimal solutions are identified by identifying redundant sources of computation such as zero-valued cumulants and simplifications that arise in the surrogate density structure. In the **DiffusionRjgqd** package, the algorithm for calculating the transitional density is somewhat more involved, and thus fewer redundancies can be exploited in order to speed up the computation of the transitional density. In the sections that follow, we detail key components of the algorithm and show how solutions are constructed. Where needed, we will contrast these components to their counterparts in the **DiffusionRgqd** package.

4.8.1 Interface and template equations

For purposes of the **DiffusionRgqd** package we develop a functional input interface whereby a model is defined in the R workspace with reference to the template generalised quadratic diffusion by declaring functions that correspond to the coefficients of the template equations. Building on this idea, we use a similar interface for the generalised quadratic jump diffusions. However, due to the somewhat odd arrangement of the jump mechanism and how a change in the structure of the jump diffusion mechanism affects the mathematical constructs on which the algorithm is built, we have had to adopt slightly simplified variants of the the generalised quadratic jump diffusions. For example, in the scalar case the template jump diffusion assumes the form:

$$\begin{aligned} dX_t &= \mu(X_t, t)dt + \sigma(X_t, t)dB_t + dP_t \\ dP_t &= J(X_t, \dot{z}_t)dN_t \end{aligned} \tag{4.8.1}$$

where

$$\mu(X_t, t) = g_0(t) + g_1(t)X_t + g_2(t)X_t^2, \tag{4.8.2}$$

$$\sigma(X_t, t) = \sqrt{q_0(t) + q_1(t)X_t + q_2(t)X_t^2}, \tag{4.8.3}$$

N_t is a one-dimensional counting process with intensity

$$\lambda(X_t, t) = \lambda_0(t) + \lambda_1(t)X_t + \lambda_2(t)X_t^2, \tag{4.8.4}$$

$$J(X_t, \dot{z}_t) = \begin{cases} \dot{z}_t & \text{if jumps are 'additive',} \\ \dot{z}_t X_t & \text{if jumps are 'multiplicative',} \\ \dot{z}_t(\beta_1 + \beta_2 X_t) & \text{as a special case,} \end{cases} \tag{4.8.5}$$

subject to the constraint in Equation 4.3.21[4], and finally the jump variables may be distributed as:

$$\dot{z}_t \sim \begin{cases} \text{Normal}(\mu(t), \sigma(t)) \\ \text{Exponential}(\lambda(t)) \\ \text{Gamma}(\alpha(t), \beta(t)) \\ \text{Laplace}(a(t), b(t)) \\ \text{Uniform}(a(t), b(t)). \end{cases} \tag{4.8.6}$$

Using this template, we can preserve the freedom of specification afforded by the functional input interface, whilst not over encumbering the user with the peripherals of the jump mechanism. From this template, any jump diffusion model

[4]Routines in the package will override invalid specifications and default to a valid specification where needed.

nested within the equation may be specified simply by defining the drift, diffusion, and intensity coefficients along with the parameters of the jump distribution. The structure of the jump matrix is then defined by choosing from the list of available options. The reasoning behind this design is as follows: When modelling an observed process with a jump diffusion model, the key focus points of the jump mechanism are the rate at which discontinuous innovations occur and the distribution of the jump innovations. Although the mechanism with which the jumps enter the equation – which is dictated by the jump matrix – is of interest as well, it rarely makes sense in practice to consider specifications beyond jumps that enter either 'additively' or 'multiplicatively'. As such, we relegate the specification of the jump matrix to a list of pre-defined options. Consider for example a jump diffusion model with dynamics given by the jump SDE:

$$\begin{aligned} dX_t &= 0.1X_t\left(5 + \frac{\sin(4\pi t)}{t+1} - X_t\right)dt + \sqrt{0.1^2 X_t}dB_t + dP_t \\ dP_t &= \dot{z}_t dN_t \end{aligned} \tag{4.8.7}$$

with intensity $\lambda(X_t, t) = 0.5$ and $\dot{z}_t \sim \mathrm{U}(-0.1, 0.1)$. The model can then be specified and its transitional density evaluated in R using the code:

```
R> # Define the diffusion part:
R> G1 <- function(t){0.1*(5 + sin(2*pi*t)/(t + 1))}
R> G2 <- function(t){-0.1}
R> Q1 <- function(t){0.1^2}
R>
R> # Specify the jump mechanism:
R> Lam0 <- function(t){0.5}
R> Ja   <- function(t){-0.1}
R> Jb   <- function(t){0.1}
R>
R> # Generate the transition density
R> res <- JGQD.density(Xs, Xt, s, t, delt, Jdist = 'Uniform', Jtype= 'Add')
```

Clearly, some of the grammatical ease of the GQD framework has been lost with the inclusion of the jump mechanism. That is, although the coefficients of the intensity function preserve a grammatical link to the model equation whereby $\lambda_0(t) =$ `Lam0 <- function(t)...`, $\lambda_1(t) =$ `Lam1 <- function(t)...`, and so on, defining a grammatical link to the parameters of the respective jump distributions is a bit more cumbersome. As such, we define the various parameters and their corresponding function names using the prefix '`J`' followed by parameter specifications often used in conjunction with the corresponding jump distribution. Table 4.8.1 relates the various distributions, their parameters, and corresponding function names. Note that the special case in Equation 4.8.5 is not selected by way of the argument `Jtype`, but rather by passing (two) values to the argument `beta`, in which case the `Jtype` argument is ignored.

Distribution	Density	Interpretation	R-Syntax
Normal	$\frac{1}{\sqrt{2\pi\sigma(t)^2}} e^{-\frac{(\dot{z}_t-\mu(t))^2}{2\sigma(t)^2}}$	Location, Scale	`Jmu(t)`, `Jsig(t)`
Exponential	$\frac{1}{\lambda(t)} e^{-\frac{1}{\lambda(t)}\dot{z}_t}$	$E(\dot{z}_t) = \lambda(t)$	`Jlam(t)`
Gamma	$\frac{\beta(t)^{\alpha(t)}\dot{z}_t^{\alpha(t)-1}}{\Gamma(\alpha(t))} e^{-\beta(t)\dot{z}_t}$	Shape, Scale	`Jalpha(t)`, `Jbeta(t)`
Laplace	$\frac{1}{2b(t)} e^{-\frac{\lvert\dot{z}_t-a(t)\rvert}{b(t)}}$	Location, Scale	`Ja(t)`, `Jb(t)`
Uniform	$\frac{1}{b(t)-a(t)}$	Location, Scale	`Ja(t)`, `Jb(t)`

TABLE 4.8.1: Parametric specifications for the possible jump variable distributions and their corresponding R-names that are recognised by functions in the **DiffusionRjgqd** package.

Similarly, in the case of bivariate generalised quadratic jump diffusions, we adopt a simplified template from the more general equations of Section 4.3.2. At the time of this writing, the bivariate template diffusion assumes the form:

$$d\mathbf{X}_t = \boldsymbol{\mu}(\mathbf{X}_t, t)dt + \boldsymbol{\sigma}(\mathbf{X}_t, t)d\mathbf{B}_t + d\mathbf{P}_t \tag{4.8.8}$$

where $\mathbf{B}_t$ is a bivariate vector of independent Brownian motions,

$$\boldsymbol{\mu}(\mathbf{X}_t, t) = \begin{bmatrix} \sum_{i+j\leq 2} a_{ij}(t) X_t^i Y_t^j \\ \sum_{i+j\leq 2} b_{ij}(t) X_t^i Y_t^j \end{bmatrix}, \tag{4.8.9}$$

$$\boldsymbol{\sigma}(\mathbf{X}_t, t)\boldsymbol{\sigma}'(\mathbf{X}_t, t) = \begin{bmatrix} \sum_{i+j\leq 2} c_{ij}(t) X_t^i Y_t^j & \sum_{i+j\leq 2} d_{ij}(t) X_t^i Y_t^j \\ \sum_{i+j\leq 2} e_{ij}(t) X_t^i Y_t^j & \sum_{i+j\leq 2} f_{ij}(t) X_t^i Y_t^j \end{bmatrix}, \tag{4.8.10}$$

and

$$d\mathbf{P}_t = \mathbf{J}(\mathbf{X}_t, t, \dot{\mathbf{z}}_t)d\mathbf{N}_t \tag{4.8.11}$$

describes a Poisson process with intensity

$$\lambda(\mathbf{X}_t, t) = \sum_{i+j\leq 1} \lambda_{ij}(t) X_t^i Y_t^j \tag{4.8.12}$$

and the jump matrix is given by:

$$\mathbf{J}(\mathbf{X}_t, t, \dot{\mathbf{z}}_t) = \begin{cases} \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} & \text{if jumps are 'additive',} \\ \\ \begin{bmatrix} \dot{z}_1 X_t \\ \dot{z}_2 Y_t \end{bmatrix} & \text{if jumps are 'multiplicative'.} \end{cases}$$

Furthermore, it is assumed that the variables $\{z_1, z_2\}'$ are distributed

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} \sim \text{MVN}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{bmatrix} \right) \tag{4.8.13}$$

where $\text{MVN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the multivariate Normal distribution with location vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

4.8.2 Workflow, moment equations, and transition density approximations

Using the methodology of Section 4.3.1, it is possible to derive a system of equations that govern the evolution of the moments of any polynomial jump diffusion for which the jump distribution has finite moments. By imposing second order restrictions on the coefficients of a jump diffusion and its jump mechanism, we extend the generalised quadratic class of diffusions to the generalised quadratic jump diffusions. As in the pure diffusion case, the premise behind this formulation is to develop a class of models for which we can accurately and reliably analyse non-linear jump diffusions using a computationally efficient numerical approximation to the transitional density. Within this framework, although the complexity of the non-linear components is 'capped', the methodology maximises the freedom of specification with respect to time-inhomogeneity. As a result, the scheme can be used to analyse a vast array of jump diffusions.

In addition to providing a mathematical basis for the analysis of jump diffusions that makes it easy to define and interpret models within the computing environment, the GQD framework makes it possible to structure an algorithm by which the mathematics that underpin the methodology of sections 4.3.2 and 4.3.4 can be done by the package routines themselves, thus relieving the user from the mathematical burdens associated with the methodology. Based on the functional input interface, the workflow of the algorithm can be summarised broadly in three steps:

(1): Identify the model within the workspace: For any routine in the **Diffusion-Rjgqd** package, the interface requires that the routine identifies a model nested within the template SDE by identifying the corresponding function names in the current workspace. From this, the model input can be validated and any erroneous or missing inputs can be identified.

(2): Construct the moment equations: Based on the methodology of Section 4.3.2, it is possible to derive the appropriate system of ODEs that can be used in

order to evaluate the moments of the jump diffusion model over the desired transition horizon(s). This is achieved using a similar process to that of the **DiffusionRgqd** package, whereby the user defined coefficients are matched to terms of the moment equations (i.e., equations 4.3.27 and 4.3.35).

(3): Construct a transition density approximation: After the moment equations are constructed and solved numerically, the moments are carried into a suitable surrogate density. Depending on the scenario, this may require that the mixture factorization be applied to the transitional density or that alternative surrogate densities such as the multimodal Pearson system be used.

On face value, these steps mirror the structure used in the routines of the **DiffusionRgqd** package. However, within each step, a number of crucial differences arise. For example, the **DiffusionRgqd** package uses the cumulant equations as opposed to the moment equations. Although the moment equations would have sufficed for the **DiffusionRgqd** package, the cumulant equations present a number of computational advantages in the context of pure diffusion processes. For example, the cumulant equations tend to assume significantly smaller values than the moment equations. As such, it is easier to assess the accuracy of a numerical solution of the cumulant equations as compared to a numerical solution of the moment equations. However, due to the structure of the PDDE for the moment generating function (which cannot be written in terms of the standard MGF alone), it is extremely difficult to derive the corresponding PDDE for the cumulant generating function. As such when dealing with jump diffusions we sacrifice some computational efficiency for generality.

Another crucial difference with regard the inclusion of the jump mechanism is that we are required to solve significantly larger systems of ODEs. This follows since, in order to evaluate the transition density using the mixture factorization we need to contrast the dynamics of the jump diffusion with that of its jump-free counterpart. Thus, for a given truncation order, say d leading to a system of k ODEs ($k = d$ for scalar diffusions), we are required to solve two systems of moment equations: One for the jump diffusion and another for the jump-free diffusion. Furthermore, in order to evaluate the zero-jump probability, we need to further augment the system of moment equations by the corresponding ODE for the zero-jump probability. As such, the system of moment equations for a jump diffusion is actually consists of $2k + 1$ dimensions as opposed to k in the pure diffusion case. For example, for the simplified generalised quadratic jump diffusions used by the **DiffusionRjgqd** package, the corresponding moment equations are given in

augmented form by $\{m_1(t), m_2(t), \ldots, m_d(t), v_1(t), v_2(t), \ldots, v_d(t), p_0(t)\}$, where

$$\begin{aligned}\frac{\partial}{\partial t}m_i(t) = \quad & i\Big(\sum_{k=0}^{2} g_k(t)m_{i+k-1}(t)\Big) \\ & + \frac{i(i-1)}{2}\Big(\sum_{k=0}^{2} q_k(t)m_{i+k-2}(t)\Big)\mathbb{I}_{i\geq 2} \\ & + \sum_{l=0}^{2}\lambda_l(t)\Big(\sum_{k=0}^{i}\binom{i}{k}m_{k+l-1}(t)\rho_{i-k+l}(t) - m_{i+l-1}(t)\Big)\end{aligned} \tag{4.8.14}$$

if jumps enter 'additively' or

$$\begin{aligned}\frac{\partial}{\partial t}m_i(t) = \quad & i\Big(\sum_{k=0}^{2} g_k(t)m_{i+k-1}(t)\Big) \\ & + \frac{i(i-1)}{2}\Big(\sum_{k=0}^{2} q_k(t)m_{i+k-2}(t)\Big)\mathbb{I}_{i\geq 2} \\ & + \sum_{l=0}^{2}\lambda_l(t)\Big(m_{i+l}(t)\sum_{k=0}^{i}\binom{i}{k}\rho_{i-k}(t) - m_{i+l}(t)\Big)\end{aligned} \tag{4.8.15}$$

if jumps enter 'multiplicatively',

$$\frac{\partial}{\partial t}v_i(t) = i\Big(\sum_{k=0}^{2} g_k(t)v_{i+k-1}(t)\Big) + \frac{i(i-1)}{2}\Big(\sum_{k=0}^{2} q_k(t)v_{i+k-2}(t)\Big)\mathbb{I}_{i\geq 2} \tag{4.8.16}$$

for $i = 1, 2, \ldots, d$ where $\rho_i(t)$ denotes the i-th non-central moment of the random variable $\dot{z}_t$, and

$$\frac{\partial}{\partial t}\log(p_0(t)) = -\sum_{i=0}^{2}\lambda_i(t)v_i(t). \tag{4.8.17}$$

where $p_0(t)$ denotes the zero jump probability, $P(N_t - N_s = 0)$. In order to complete the specification of the moment equations, the moment structures are constructed from the specified jump density and matched accordingly to the parameter functions specified by the user. For example, under the Gamma density we set:

$$\rho_i(t) = \Gamma(\alpha_J(t) + i)/\Gamma(\alpha_J(t))\beta_J(t)^i. \tag{4.8.18}$$

In the bivariate case, under an $d = 4$-th order truncation the augmented moment equations is a 29 dimensional system of coupled ODEs, resulting from 14 equations for the jump diffusion moments, 14 equations for the diffusion moments and another for the zero-jump probability differential equation. For brevity, we will

not reproduce the equations here – the equations can be derived directly from Equation 4.3.35 and Table 4.3.1.

As demonstrated in Section 4.3.4, evaluating the transition density of a jump diffusion accurately by plugging its moment trajectories into a surrogate density can be difficult. Using the mixture factorization, it is possible to produce very accurate approximations over short transition horizons, despite the dichotomous behaviour of the process over short time lapses. Depending on the nature of the jump diffusion being analysed and the size of the transition horizon, various surrogate density structures may be used in order to accurately evaluate the transitional density. For example, when fitting a jump diffusion model to a discretely observed trajectory of a process, the mixture factorization plays a key role in the accuracy with which parameter estimates can be calculated. In other circumstances, where one is interested in the long-run dynamics of a process, the mixture factorization may be redundant for a large proportion of the transition horizon. This follows since the mixture factorization is calculated with reference to the probability of incurring at least one jump innovation over the transition horizon. Consequently, as time increases and the event of observing at least one jump approached certainty, contrasting the diffuse and jump dynamics becomes inconsequential to the transition density approximation. As such, we allow various specifications of the surrogate density structure. First, it is determined whether the mixture factorization is to be applied. If it is specified as not to be applied, the moments of the jump diffusion are plugged into a surrogate density, which for selected routines may be either a saddlepoint approximation or any one of the members of the Pearson system (see Section 2.3.3) with the default being the saddlepoint approximation. Alternatively, should the mixture factorization be applied, then one of a number of combinations can be specified. For example, the default behaviour is to use a saddlepoint approximation for both the jump-free transition density and the excess distribution. It is also possible to specify a combination of a Normal distribution and a saddlepoint approximation for the jump-free diffusion density and excess distribution respectively or, as in the case of the example in Section 4.7.2, apply the multimodal Beta distribution to both the jump-free and excess distribution. As a final comment with regard to the mixture factorization, in order to ensure numerical stability, we apply a heuristic cut-off based on the zero jump probability for applying the mixture factorization. That is, as time progresses and the zero-jump probability drops below a threshold of 1%, the mixture factorization is no longer applied and the transition density is approximated using the jump diffusion moments and a surrogate density alone. This ensures that the transition density approximation remains accurate over large transition horizons whilst preventing numerical instabilities caused by inverting Equation 4.3.70. Figure 4.8.1 illustrates how the mixture factorization reverts at the threshold zero jump probability for a hypothetical jump diffusion model.

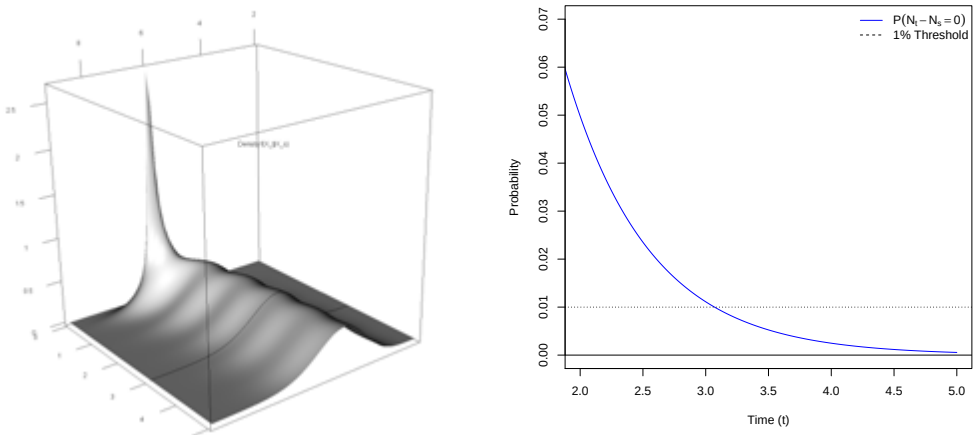


FIGURE 4.8.1: Transition density of a jump diffusion (from the interface example in Section 4.8.1) (left) and the evolution of the zero jump probability over time (right). At the point at which the threshold is crossed (indicated in black on the transition density surface) the transition density approximation reverts to using the jump diffusion moments and a surrogate density alone. R code: Supplementary materials, Section 4.17.

4.8.3 Using C++ to improve computational efficiency

Although combining the moment equations of the generalised quadratic jump diffusions with the mixture factorization results in a computationally efficient approximation of the transitional density, the iterative nature of likelihood inference procedures such as the random walk Metropolis-Hastings algorithm implies significant computational overhead when calculating the likelihood of a jump diffusion model using Equation 4.5.2. In the **DiffusionRgqd** package this was remedied to an extent by making use of the C++ language within R through the **Rcpp** (Eddelbuettel and François, 2011; Eddelbuettel, 2013) and **RcppArmadillo** (Eddelbuettel and Sanderson, 2014) packages. As such, we employ a similar strategy for the **DiffusionRjgqd** package where, whenever a routine iteratively evaluates the likelihood function, the routine constructs a computationally optimised (vectorized) C++ program that evaluates the likelihood of a given model which can then be called within an MCMC wrapper within the R environment. This is achieved by building an appropriate solution from pre-written blocks of C++ code and subsequently filling in the components that are unique to a given jump diffusion model using the workflow pattern to what is described in Section 4.8.2. For example, consider Equation 4.5.5, from the simulation study

in Section 4.5.2. By defining the model in terms of the template generalised quadratic jump diffusion using the code:

```
R> # Define the model:
R> G0 <- function(t){theta[1]*theta[2]}
R> G1 <- function(t){-theta[1]}
R> Q1 <- function(t){theta[3]*theta[3]}
R> Jmu  <- function(t){theta[5]}
R> Jsig <- function(t){theta[6]}
R> Lam1 <- function(t){theta[4]}

R> # Call an MCMC routine for the time series X~time:
R> model <- JGQD.mcmc(X, time, mesh, theta, sds, updates, burns)
```

As before, the `JGQD.mcmc()` routine will then construct a C++ routine for evaluating the likelihood that can be called within R. The function will subsequently set up and run the RWMH algorithm in order to calculate parameter estimates for the vector `theta`. As before, depending on the nature of the model being analysed, the complexity and structure of the C++ routine will vary. For example, the C++ routine for Equation 4.5.5 is given in Appendix D.9, whilst the routine for the bivariate model Equation 4.5.10 is given in Appendix D.10.

4.8.4 Outline of the package

DiffusionRjgqd consists of a set of functions that allow the user to perform inference and analysis on generalised quadratic jump diffusions. The main routines that appear in the package are (main functions that do not use C++ are indicated with an asterisk):

`BiJGQD.density`*: Calculate the transition density of a bivariate jump-GQD with time-inhomogeneous coefficients over a specified time interval.

`BiJGQD.mcmc` : Use an MCMC algorithm to draw parameters of a bivariate jump-GQD model with time-inhomogeneous coefficients.

`JGQD.density`* : Calculate the transition density of a scalar jump-GQD with time-inhomogeneous coefficients over a specified time interval.

`JGQD.mcmc` : Use an MCMC algorithm to draw the parameters of a scalar jump-GQD model with time-inhomogeneous coefficients.

In addition to the main routines, some supporting functions have been created to make the package more user friendly. These include:

`JGQD.remove` : Removes the coefficients of an existing jump-GQD model from the current workspace.

`JGQD.dic` : Summarizes DIC values from a list of `JGQD.mcmc` and `BiJGQD.mcmc` objects.

`JGQD.plot` : Plot routines for various classes of objects in the **DiffusionRjgqd** package.

4.8.5 Example applications

In the examples that follow we demonstrate how **DiffusionRjgqd** package is used in practice. The package can be found on GitHub at https://github.com/eta21 and the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/package=DiffusionRjgqd. In addition to the examples showcased here, detailed examples can be found within the package vignettes on the package CRAN page or by running the command: `browseVignettes("DiffusionRjgqd")` after loading the package in an R-session.

4.8.5.1 Generate the transition density of a jump diffusion with diffuse intensity

Using the **DiffusionRjgqd** package one can easily generate transitional densities for jump diffusion models with quite complicated dynamics. Consider for example a jump diffusion with diffuse intensity of the form:

$$\begin{aligned} dX_t &= \alpha_x X_t(\beta_x - X_t)dt + \sqrt{\sigma_x^2 X_t^2}dB_t + dP_t \\ dP_t &= \dot{z}_t dN_t \end{aligned} \tag{4.8.19}$$

and let $N_t - N_s \sim \text{Poi}(\int_s^t E(\dot{r}_u)du)$ where the intensity process is governed by the SDE:

$$d\dot{r}_t = a(b + \nu\sin(0.5\pi t + \rho) - \dot{r}_t)dt + \sigma\sqrt{\dot{r}_t}dB_t \tag{4.8.20}$$

with $\dot{z}_t \sim \text{N}(\mu_z, \sigma_z^2)$. Using the GQD framework the model can easily be defined within the R workspace in terms of the coefficients of the template equation. However, in order to evaluate the moment equations we are required to evaluate the expectation of the intensity process. Under Equation 4.8.20, the expectation

of the intensity process is given by the expression:

$$\begin{aligned} E(\dot{r}_t|\dot{r}_s) = \quad & \lambda_s e^{-a(t-s)} + b(1-e^{-a(t-s)}) \\ & + a\nu\left(\frac{(0.5\pi\sin(\rho) + a\cos(\rho))\sin(0.5\pi(t-s))}{(0.5\pi)^2 + a^2}\right) \\ & + a\nu\left(\frac{(a\sin(\rho) - 0.5\pi\cos(\rho))\cos(0.5\pi(t-s))}{(0.5\pi)^2 + a^2}\right) \\ & - a\nu\left(\frac{(a\sin(\rho) - 0.5\pi\cos(\rho))e^{-a(t-s)}}{(0.5\pi)^2 + a^2}\right). \end{aligned} \tag{4.8.21}$$

Subsequently, we can use the `JGQD.density()` function in order to evaluate the transitional density. Assuming the parameter set $\{\alpha_x, \beta_x, \sigma_x, a, b, \nu, \rho, \sigma, \mu_z, \sigma_z\}$ $= \{0.1, 8, \sqrt{0.025}, 1, 2, 2, 0.1, 0.1, 0.5, 0.25\}$ and initial values $\{X_0, \lambda_0\} = \{4, 1\}$, we have:

```
R> # Set some parameters for the intensity process:
R> alpha.x <- 0.1; beta.x <- 8; sigma.x <- sqrt(0.025);
R> a       <- 1  ; b      <- 2; nu      <- 2;
R> sigma   <- 0.1; rho    <- 0.1
R> mu.z    <- 0.5; sig.z  <- 0.25
R> X0      <- 4  ; l0     <- 1
R>
R> # Define the jump diffusion using the DiffusionRjgqd syntax:
R> G1   <- function(t){alpha.x*beta.x}
R> G2   <- function(t){-alpha.x}
R> Q2   <- function(t){sigma.x^2}
R> Jmu  <- function(t){mu.z}
R> Jsig <- function(t){sig.z}
R> Lam0 <- function(t)
+  {
+     (l0*exp(-a*t)+b*(1-exp(-a*t))
+     +a*nu*((((0.5*pi)^2*sin(rho)
+     +a*0.5*pi*cos(rho))*sin(0.5*pi*t))/(0.5*pi*((0.5*pi)^2+a^2))
+     +((a*sin(rho)-0.5*pi*cos(rho))*cos(0.5*pi*t))/((0.5*pi)^2+a^2)
+     -((a*sin(rho)-0.5*pi*cos(rho))*exp(-a*t))/((0.5*pi)^2+a^2)))
+  }
R>
R> # Calculate the approximate transition density:
R> res <- JGQD.density(Xs = X0, Xt = seq(4, 11, 0.1), s = 0, t = 8, delt = 0.01,
     factorize = TRUE)
```

As in the case of `GQD.density()`, the jump-free counterpart of the `JGQD.density()` function, a list containing various results is returned including the transition density, the moments, cumulants, excess moments, and the trajectory of the zero-jump probability. Subsequently, the transition density can be plotted by simply passing the return list to the `JGQD.plot()` function, in this case `JGQD.density(res)`. In

order to demonstrate the validity of the approximation we compare the transition density approximation to that calculated from simulated trajectory. Figure 4.8.3 compares the approximation at various points along the transition horizon from the present example. From this, the effect of the sinusoidal dynamics of the intensity process can be seen to manifest in the transition density of the jump diffusion – which apart from the intensity process is time-homogeneous.

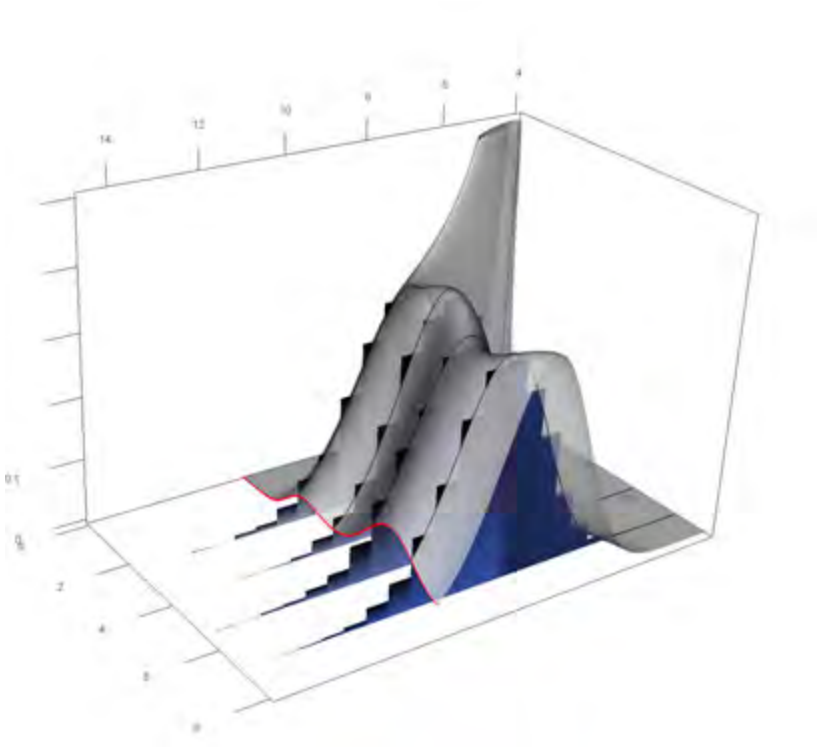


FIGURE 4.8.2: Approximate (gray) and simulated (blue histograms) transition density for Equation 4.8.19. The surface is cut away (red) in order to illustrate the effect of the sinusoidal drift dynamics of the intensity process. R code: Supplementary materials, Section 4.18.

4.8.5.2 Generate the transition density of a non-linear a Hawkes process

Although the focus of the **DiffusionRjgqd** package is on jump diffusion processes, it so happens that another class of processes are nested within the framework. When the diffusion terms of the template equations are set to zero, the resulting set of equations are that of a Hawkes process. Under this regime the stochastic

differential equation assumes the form:

$$dX_t = \mu(X_t, t)dt + \dot{z}_t dN_t \tag{4.8.22}$$

where N_t has intensity $\lambda(X_t, t)$ and $\dot{z}_t \sim \phi(t)$. In the case of the Hawkes processes, we still need to incorporate the mixture factorization in order to accurately approximate the transitional density. However, since the diffusion terms are zero, the diffuse part of the process collapses to an ordinary differential equation. Consequently, we decompose the transitional density as follows. Let $f_H(X_t|X_s)$ denote the transitional density of X_t, then:

$$f_H(X_t|X_s) = P(N_t - N_S = 0)\delta(X_t - m(t)) + P(N_t - N_s > 0)f_E(X_t|X_s) \tag{4.8.23}$$

where $m(t)$ is the solution to the ODE that results from setting the diffusion terms to zero (i.e., the ODE defined by the drift function), and $\delta(.)$ is the Dirac delta function. Consider the SDE:

$$dX_t = \alpha X_t(\beta - X_t)dt + \dot{z}_t dN_t \tag{4.8.24}$$

with $\lambda(X_t, t) = \lambda X_t$ and $z_t \sim N(\mu_z, \sigma_z^2)$. Note that, since the intensity coefficient depends on the state of the process, the SDE may be referred to as 'self-exciting', meaning that jumps occur more often when the process level increases. For purposes of the experiment we assume the parameter set $\{\alpha, \beta, \lambda, \mu_z, \sigma_z\} = \{0.1, 7, 1, 0, 0.5\}$ with initial value $X_0 = 4$. Within R, we can them use the GQD-syntax along with the `JGQD.density()` function in order to generate the transitional density:

```
R> # Remove any existing models:
R> JGQD.remove()
R>
R> # Define a Hawkes process using the DiffusionRjgqd syntax:
R> G1   <- function(t){0.1*7}
R> G2   <- function(t){-0.1}
R> Lam1 <- function(t){1}
R> Jsig <- function(t){0.5}
R>
R> # Calculate the approximate transition density:
R> Xs   <- 4
R> Xt   <- seq(0, 10, 0.01)
R> s    <- 0
R> t    <- 1
R> delt <- 0.004
R> res <- JGQD.density(Xs, Xt, s, t, delt, factor.type = 'Hawkes', factorize =
    TRUE)
```

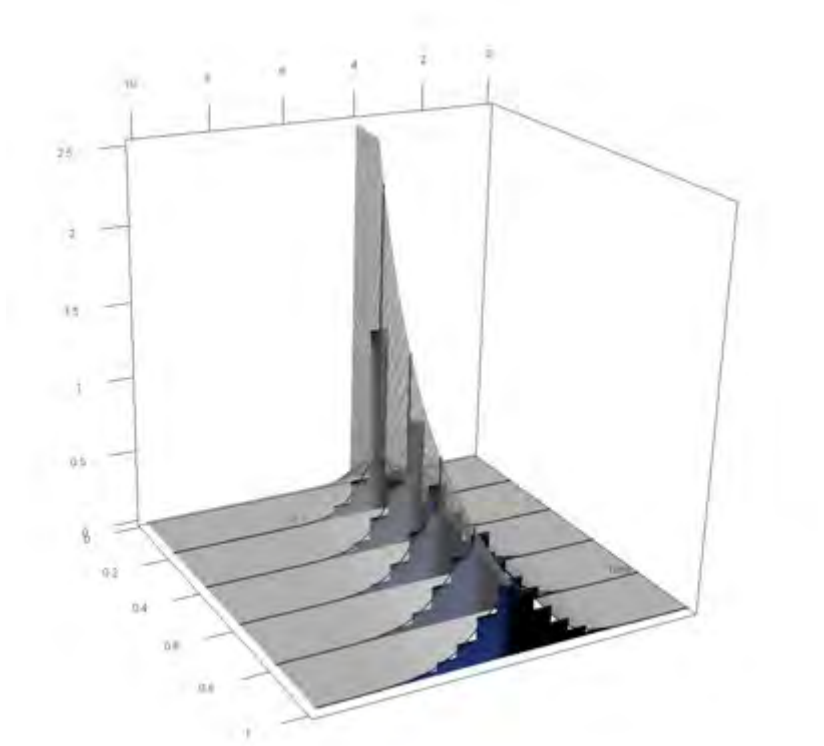


FIGURE 4.8.3: Approximate (gray) and simulated (blue histograms) transition density for Equation 4.8.22. The 'flat fin' like structure of the density follows from the zero diffusion terms. Within the approximation, the declining value of the density at the 'fin' follows from the fact that the Dirac-delta function is approximated using a finite difference. R code: Supplementary materials, Section 4.19.

4.8.5.3 Google equity volatility revisited

The **DiffusionRjgqd** package makes it very easy to fit and compare various jump diffusion models to real-world datasets. For example, in Section 4.6 we compared various models of Google equity volatility time series. In order to demonstrate how such an analysis may conducted in R using the **DiffusionRjgqd** package we fit a jump diffusion model to the Google equity volatility series. Since we know from the results of Section 4.6 what the best fitting model structure (among the models that were tested) should be, we focus only on a single model. Following

the specification of Model 10, we fit

$$\begin{aligned} dX_t &= \theta_1(\theta_2 + \theta_7 \sin(8\pi t + (\theta_8 - 0.5)2\pi) - X_t)dt + \theta_3 X_t dB_t + dP_t \\ dP_t &= \dot{z}_t X_t dN_t \end{aligned} \tag{4.8.25}$$

where $\lambda(X_t, t) = \theta_4$ and $\dot{z}_t \sim \mathrm{N}(\theta_5, \theta_6^2)$ to the VXGOG series. Furthermore, we place the same prior distributions on the parameter space of the model as in Section 4.6. Reiterating from Table 4.6.2, in terms of the θ-parametrisation:

Parameter	Prior distribution
θ_1	Gamma$(0.001, 0.001)$
θ_2	Normal$(25, 5^2)$
θ_3^2	Inv-Gamma$(0.001, 0.001)$
θ_4	Gamma$(0.001, 0.001)$
θ_7	Gamma$(0.001, 0.001)$
θ_8	Beta$(0.5, 0.5)$

TABLE 4.8.2: Prior distributions on the parameter space.

Using the **Quandl** (McTaggart and Daroczi, 2015) package, we first source the data:

```
R> # Source data for the Google VIX:
R> quandldata1 <- Quandl("CBOE/VXGOG", collapse="daily",
+  start_date="2010-03-11",end_date="2016-01-01", type="raw")
R> Vt <- rev(quandldata1[,names(quandldata1)=='Close'])
R> time1 <-rev(quandldata1[,names(quandldata1)=='Date'])
```

Subsequently, we fit the jump diffusion model using the GQD-syntax and define the prior function as a product of the prior distributions of each parameter:

```
R> JGQD.remove()
R> G0   <- function(t)
+  {
+     theta[1]*theta[2] + theta[1]*theta[7]*sin(8*pi*t + (theta[8]-0.5)*2*pi)
+  }
R> G1   <- function(t){-theta[1]}
R> Q2   <- function(t){theta[3]*theta[3]}
R> Lam0 <- function(t){theta[4]}
R> Jmu  <- function(t){theta[5]}
R> Jsig <- function(t){theta[6]}
R>
R> priors <- function(theta)
+  {
+     dgamma(theta[1], 0.001, 0.001)*
+     dnorm(theta[2], 25, 5)*
```

```
+     dgamma(1/theta[3]^2, 0.001, 0.001)*
+     dgamma(theta[4], 0.001, 0.001)*
+     dgamma(theta[7], 0.001, 0.001)*
+     dbeta(theta[8], 0.5, 0.5)
+ }
```

Subsequently, we can fit Equation 4.8.25 to the volatility series using the RWMH-algorithm. This is achieved by making use of the `JGQD.mcmc()` function. By specifying a starting parameter vector for the algorithm along with a proposal standard deviation vector, we can perform the desired number of updates (in this case, 110 000) using the code:

```
R> X       <- Vt
R> time    <- cumsum(c(0 , diff(as.Date(time1))*(1 / 365)))
R> updates <- 110000
R> burns   <- 10000
R> theta   <- c(50, 25, sqrt(9), 100, 0.1, 0.1, 50, 0.5)
R> sds     <- c(1.65, 0.79, 0.02, 4.29, 0.01, 0.01, 0.94, 0.03)/1.5
R> model_10<- JGQD.mcmc(X, time, 10, theta, sds, updates, burns, Jtype = 'Mult')
```

Note the use of the `Jtype` parameter: Here `Jtype='Mult'` sets the jump coefficient specification to multiplicative. When the routine has completed all updates, a trace-plot (Figure 4.8.4) is drawn depicting the parameter chains along with a trajectory of the acceptance rate. In addition, a plot is made of the estimated jump probability for each observed transition superimposed over the absolute value of the first differenced series. The latter figure can be used to assess the jump frequency as well as how sensitive the jump frequency is to changes in the level of the process. Finally, as in the case of the **DiffusionRgqd** package (see Chapter 2), parameter estimates can be calculated by passing the model to the `JGQD.estimates()` function, which will draw autocorrelation plots (Figure 4.8.5) for each parameter chain and produce a table of parameter estimates under a given thinning factor:

```
R> ests <- JGQD.estimates(model_10, thin = 200, burns = burns)
R> ests
```

```
R>          Estimate Lower_CI Upper_CI
R> theta[1]   10.225    6.915   13.607
R> theta[2]   26.949   25.536   28.650
R> theta[3]    0.643    0.608    0.676
R> theta[4]   27.931   20.269   36.525
R> theta[5]   -0.004   -0.027    0.019
R> theta[6]    0.150    0.128    0.174
R> theta[7]    6.845    4.776    9.439
R> theta[8]    0.563    0.515    0.614
```

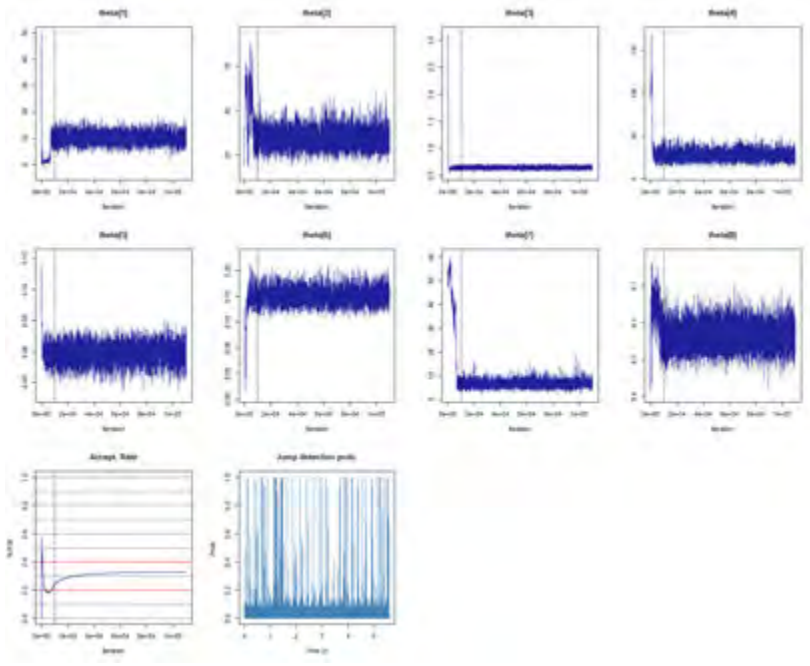


FIGURE 4.8.4: Trace-plots for the parameter chains of Equation 4.8.25 calculated using `JGQD.mcmc()` (iteration number indicated on x-axis). In addition, a traceplot of the acceptance rate is made as well as the estimated jump detection probabilities for each transition (observation times indicated on x-axis). R code: Supplementary materials, Section 4.20.

Finally, we can include a jump detection sequence by accessing the estimated jump detection probabilities. The appropriate sequence can be accessed from the `JGQD.mcmc()` return list under the variable `$decode.prob`, to which we can apply some decision rule to extract jump indicators. In R:

```
R> plot(X~time1, type='l', col = '#BBCCEE')
R> indicators <- which(model_10$decode.prob > 0.8)
R> t.jumps    <- time1[indicators + 1]
R> segments(t.jumps, X[indicators], t.jumps, X[indicators+1], col = '#222299')
```

Figure 4.8.6, illustrates the resulting graphical output.

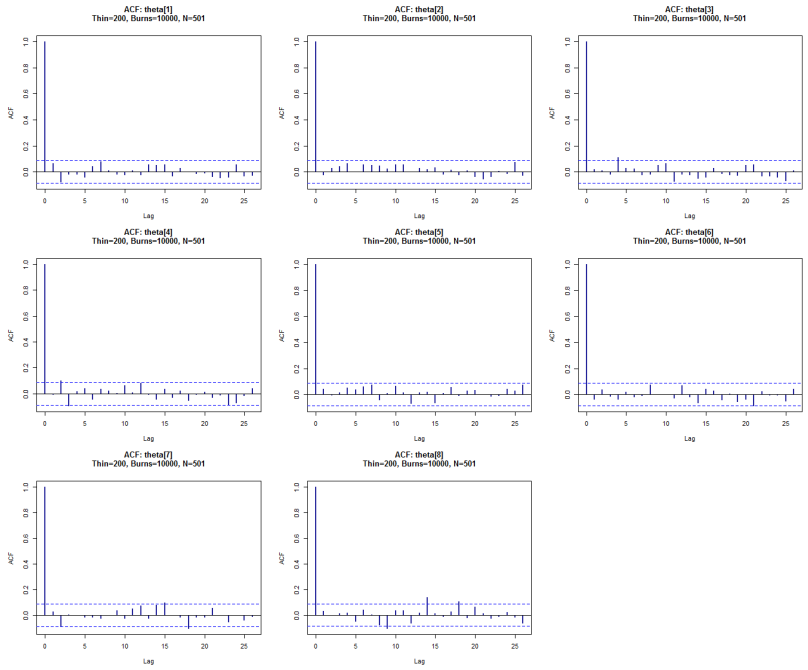


FIGURE 4.8.5: Autocorrelation plot for the thinned parameter chains calculated using `JGQD.mcmc()` (lag index indicated on x-axis). R code: Supplementary materials, Section 4.20.

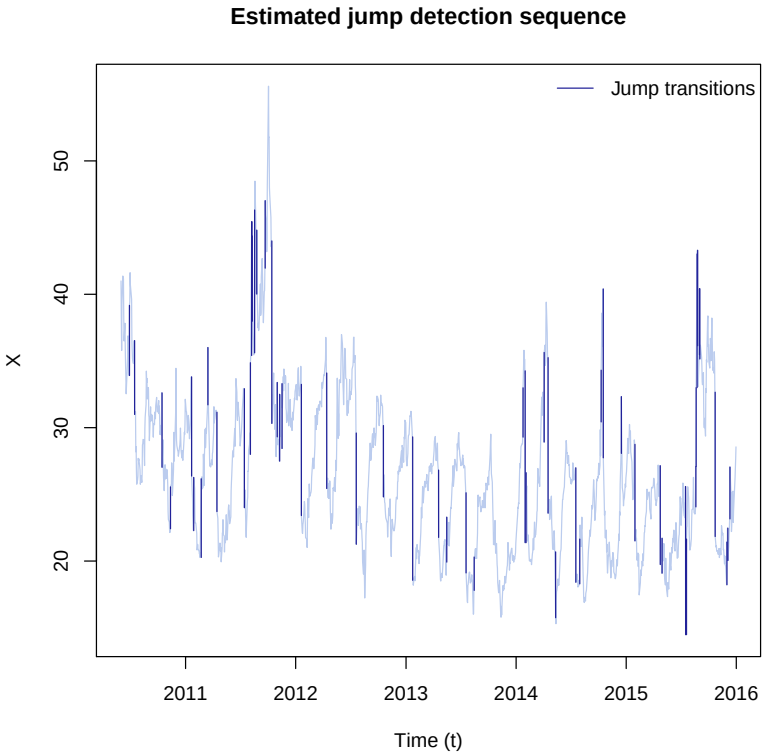


FIGURE 4.8.6: Estimated jump arrival times calculated by applying an 80% decision rule to the estimated jump detection probability sequence. R code: Supplementary materials, Section 4.20.

4.9 Chapter summary

Jump diffusions are an important class of continuous time Markov processes that find application in numerous fields of science. As in the case of pure diffusion processes, quantities such as the transitional densities of jump diffusion models are, for the most part, intractable. By deriving a partial difference differential equation for the moment generating function of a jump diffusion process, it is possible to derive the moment equations of the process in terms of elements of the jump mechanism. This can be achieved for a very general class of processes, incorporating non-linear models with state-dependent and/or stochastic jump intensity. Combining the moment equations of a model process with a simple factorization of the transitional density around that of the models' jump-free counterpart, we can accurately approximate the transitional density of such processes over both small and large time horizons. In comparing the methodology to that of analytical results and existing techniques for approximating the transitional density, we find that our scheme provides accurate results and may provide superior approximations over closed-form expansions studied here, especially over large transition horizons. Furthermore, we find that the approximation scheme can readily be applied in the context of performing inference on jump diffusion models for datasets where the jump mechanism cannot be directly observed. Indeed, the structure of the approximation can be used in order to estimate jump detection probabilities, thus giving a measure by which one can estimate which observed transitions are likely to contain jump events. Following this, we apply the methodology to a real-world dataset by modelling the equity volatility of Google shares using various forms of jump diffusion models. By comparing standard model fit statistics, we are able to identify a drift cycle as well as evidence of state dependence in the jump mechanism of the volatility series – attributes which are rarely explored in empirical finance due to the lack of such models with analytically tractable dynamics.

We proceed to show how the methodology can be applied to interesting new theoretical models, wherein we demonstrate a jump-reversion mechanism – a reversion mechanism by which the process tracks a long-run level in discrete, randomly sized steps as opposed to traditional mean reversion mechanisms that operate continuously in time. Indeed, we show that, by choosing a uniform jump distribution in conjunction with a special parametrisation of the intensity coefficient, a form of parity can be established between the jump-reversion and continuously reverting diffusions. We also demonstrate the application of non-linear jump diffusion models to the approximation of the Wright-Fischer process under frequency shocks.

Finally, building on the architecture of the **DiffusionRgqd** package, we develop the **DiffusionRjgqd** package – an R package for the analysis of generalised quadratic jump diffusions. We show how the package can be used to approximate the transitional densities of complex jump diffusion processes, and make explicit the process by which one can analyse real-world datasets by revisiting the Google equity volatility dadaset.

Chapter 5

Conclusion

5.1 Future research avenues

5.1.1 Introduction

In addition to the research showcased in the preceding chapters, a number of other problems relating to the analysis of non-linear diffusion processes have been considered. Although some of these problems have been considered in great detail, at the time of completing this thesis the research conducted in these areas may not have been sufficiently complete to warrant inclusion as stand-alone chapters. This follows both from technical matters that still need to be addressed as well as the timing of the research where some of the material is in its infancy. As such, we showcase two interesting and important topics that bear relation to the work already covered as works in progress and possible avenues of future research. We demonstrate shortly preliminary work on diffusion processes with Markov-switching parameters and the application of non-standard first passage time problems in ecology.

5.1.2 Markov-switching diffusion processes

One extension of the research conducted in this thesis is the analysis of Markov-switching diffusion models. Specifically, the analysis of diffusion models with Markov-switching parameters. In Section 4.3.3.2 we hinted at the possibility of formulating a jump diffusion model with Markov-switching jump intensity. In what follows, we proceed to show that one can approximate the moment

trajectories and transitional density of such a process and outline preliminary work for performing inference on Markov-switching diffusion models. Although the topic of Markov-switching diffusion models have been of some interest from the outset of the research conducted in this thesis, various technical issues that arise in the analysis of such models have made the development of a suitably general framework for conducting such analysis difficult. As a consequence, the methodology we have developed thus far consists of some incomplete and semi-complete elements which require more work before a full treatise on the topic can be assembled. As such, in the sections that follow we discus shortly the key focus points and limitations of this research as a work in progress.

5.1.2.1 Transition density approximations for Markov-switching diffusions

Consider a diffusion model with dynamics given by the SDE:

$$dX_t = \mu(X_t, t, \boldsymbol{\theta}_t)dt + \sigma(X_t, t, \boldsymbol{\theta}_t)dB_t, \tag{5.1.1}$$

where the drift and diffusion coefficients are parametrised by some parameter process $\boldsymbol{\theta}_t$. That is, the parameters of the diffusion process may vary stochastically over time. By specifying a diffusion model with coefficients that depend on some external process, it is possible to formulate more realistic models of real world phenomena. For example, by specifying a diffusion model stochastic volatility parameters, where the dynamics of the volatility process itself is governed by a diffusion model, we arrive at the so-called stochastic volatility models. In more general cases, we can formulate multi-factor diffusion models such as the Chen model (Chen, 1996) where drift components are also modelled using a diffusion process. Indeed, one of the benefits of formulating the dynamics of the parameters of the target process (i.e., the parameters of the original model of interest) in terms of diffusion models is that the resulting model can be written as a system of SDEs. Consequently, provided that we can find an appropriate proxy for the unobserved parameter process (as in the case of stochastic volatility models), we can readily apply existing techniques for analysing multivariate, non-linear diffusion models to such models. When a suitable proxy to the unobserved components cannot be found, it is still possible to analyse such a model by making appropriate modifications to the estimation scheme. Although such models have proven to be quite successful in modelling various processes and have been widely adopted in the literature, we can perhaps ask whether it is possible to formulate a diffusion model with coefficients that follow a process with a structure that is not that of a diffusion process? Indeed, it could be useful or perhaps even more accurate/realistic to formulate the dynamics of the parameters of a diffusion

model in terms of a completely different type of process. Borrowing from the field of hidden Markov models (HMMs), one such strategy may be to consider a diffusion model where the parameter process $\boldsymbol{\theta}_t$ alternates stochastically between sets of fixed values according to a continuous time Markov chain. That is, let $\boldsymbol{\theta}_t$ denote a parameter vector which may assume q possible states $\{\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, ..., \boldsymbol{\theta}^{(q)}\}$ according to a CTMC with transition rate matrix $\mathbf{R} = (\beta_{ij}(t))_{q\times q}$. Then the diffusion process is formulated in terms of a stochastic differential equation with Markov-switching parameters. Hidden Markov models provide a flexible class of models that have many attractive features from a modelling perspective, and it is easy to foresee the application of Markov-switching diffusions in fields such as economics and finance where the environment in which financial processes operate may be subject to distinct economic states.

On face value, this appears to be a relatively simple and natural generalisation of diffusion processes for which the analysis should follow equally as naturally: Since the analysis of simple CTMCs relies on transition probabilities which are typically available in closed form, it is tempting to conclude that the analysis of such models should be achievable with only minor modifications to existing techniques for analysing diffusion models. Unfortunately, it turns out that analysing such models is most certainly a non-trivial task, and the mechanics of such processes lead to subtleties which are not immediately obvious at the outset of the analysis. Despite the fact that one can analyse properties of the process without developing any new mathematical elements over and above what has already been developed in this thesis, it remains difficult to make practical use of the resulting quantities. For example, although it is possible to derive moment equations for Markov-switching diffusions, in which case it should, in theory, be a simple step toward approximating the transitional density by plugging the moments into a suitable surrogate density, this strategy fails quite spectacularly to replicate the behaviour of the transitional density of a Markov-switching diffusion. As in the case of jump diffusion processes, this appears to follow from the dichotomous nature of the process: The contrast between the discrete nature of the parameter process and the continuous dynamics of the underlying diffusion process result in dynamical behaviour which is difficult to characterise using standard techniques.

Following along the lines of Section 4.3.4, we may attempt to remedy the problem by making use of a mixture factorization of some sort. Here, instead of factorizing the transition density with respect to a jump mechanism, we may factorize the transitional density with respect to the states of the parameter chain. Noting that, should the parameter chain remain in its initial state for the entire duration of the transition horizon, the dynamics of the process will be that of the standard diffusion process, we may factorize the transition density of the Markov-switching diffusion around that of its non-switching counterpart. That is, let

$f_{MS}(X_t|X_s, \boldsymbol{\theta}_s = \boldsymbol{\theta}^{(i)})$ denote the transition density of the Markov-switching diffusion for a known initial parameter state (in position i), and $f_D(X_t|X_s, \boldsymbol{\theta}_s = \boldsymbol{\theta}^{(i)})$ denote the transition density of the non-switching diffusion under the parameter set corresponding to the initial state of the parameter chain, then we may factorize the density:

$$f_{MS}(X_t|X_s, \boldsymbol{\theta}^{(i)}) = \alpha f_D(X_t|X_s, \boldsymbol{\theta}^{(i)}) + (1-\alpha) f_E(X_t|X_s, \boldsymbol{\theta}^{(i)}), \tag{5.1.2}$$

where $f_E(X_t|X_s, \boldsymbol{\theta}^{(i)})$ once again denotes some excess distribution. As before, in order to complete the factorization, we need to replace α with an appropriate probability. Noting that the holding time for the CTMC starting in state σ_s can be related to the transition rate matrix, we may use $\alpha = P(\boldsymbol{\theta}_u = \boldsymbol{\theta}^{(i)} \quad \forall \quad u \in [s,t]|\boldsymbol{\theta}_s = \boldsymbol{\theta}^{(i)})$, which gives the probability that the parameter chain remains in the initial state throughout the duration of the transition horizon. In general, given the transition rate matrix $\mathbf{R} = (\beta_{ij}(t))_{q\times q}$, the corresponding probability can be calculated as:

$$P(\boldsymbol{\theta}_u = \boldsymbol{\theta}^{(i)} \quad \forall \quad u \in [s,t]|\boldsymbol{\theta}_s = \boldsymbol{\theta}^{(i)}) = \exp\left(-\int_s^t \sum_{j\neq i} \beta_{ij}(u)du \right). \tag{5.1.3}$$

Based on this factorization, we can approximate the transitional density in the same way as for jump diffusion models: First, we integrate and subsequently invert Equation 5.1.2 in order to approximate the moments of $f_E(X_t|X_s, \boldsymbol{\theta}^{(i)})$. Then we approximate $f_{MS}(X_t|X_s, \boldsymbol{\theta}^{(i)})$ as a mixture of $f_D(X_t|X_s, \boldsymbol{\theta}^{(i)})$ and $f_E(X_t|X_s, \boldsymbol{\theta}^{(i)})$ where $f_D(X_t|X_s, \boldsymbol{\theta}^{(i)})$ can be approximated using the cumulant truncation procedure (or equivalently by moment truncation) and $f_E(X_t|X_s, \boldsymbol{\theta}^{(i)})$ may be approximated by plugging the moments that result from the inversion into a suitable surrogate density.

To demonstrate the strategy, consider a diffusion process with dynamics given by the SDE:

$$dX_t = (\lambda_t - X_t)dt + \sigma_t\sqrt{X_t}dB_t, \tag{5.1.4}$$

where the pair $(\lambda_t, \sigma_t) \in \{(\lambda_1, \sigma_1), (\lambda_2, \sigma_2)\}$ switches states according to a two-dimensional CTMC with transition rate matrix $R = (\beta_{ij})_{2\times 2}$. Here, both the drift and volatility of the process may change stochastically over time. Moreover, the parameters of the process change concurrently, essentially causing the process to switch instantaneously between two entirely different parameter sets. Using Equation 5.1.2, we can approximate the transitional density of Equation 5.1.4. Let $m_i^{MS}(t)$ and $m_i^D(t)$ denote the i-th non-central moment of Equation 5.1.4 and its non-switching counterpart respectively, then we can approximate[1] the

[1]These expressions are exact for the first moment and approximate for the higher order moments.

moment equations for the process as:

$$\begin{aligned}\frac{\partial}{\partial t}m_i^D(t) &= i(\lambda_k m_{i-1}^D(t) - m_i^D(t)) + \frac{i(1-1)}{2}\sigma_k^2 m_{i-1}^D(t)\mathbb{I}(i \geq 2) \\ \frac{\partial}{\partial t}m_i^{MS}(t) &= i(\nu_{\lambda,k}(t)m_{i-1}^{MS}(t) - m_i^{MS}(t)) + \frac{i(1-1)}{2}\nu_{\sigma,k}(t)m_{i-1}^{MS}(t)\mathbb{I}(i \geq 2),\end{aligned} \tag{5.1.5}$$

where

$$\begin{aligned}\nu_{\lambda,k}(t) &= \lambda_1 p_{k1}(t) + \lambda_2 p_{k2}(t) \\ \nu_{\sigma,k}(t) &= \sigma_1^2 p_{k1}(t) + \sigma_2^2 p_{k2}(t)\end{aligned} \tag{5.1.6}$$

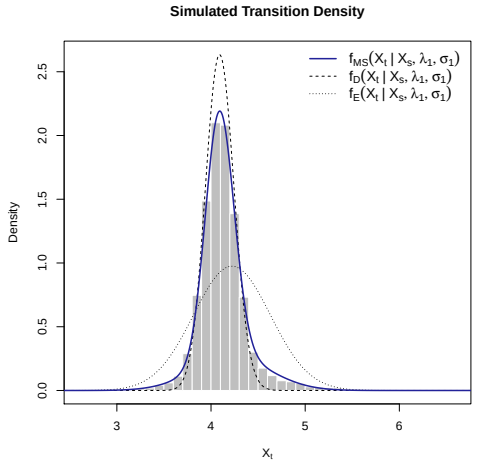
for $k = 1, 2$, and $\{p_{ij}(t) : i, j = 1, 2\}$ denote the transition probabilities of the CTMC. That is, $p_{ij}(t)$ denotes the probability that the parameter process will transit from state i to j by time t. From the transition rate matrix, these probabilities can be calculated by solving the system of ODEs:

$$\frac{\partial}{\partial t}p_{ij}(t) = \sum_{k=1}^{2}\beta_{ik}(t)p_{kj}(t), \tag{5.1.7}$$

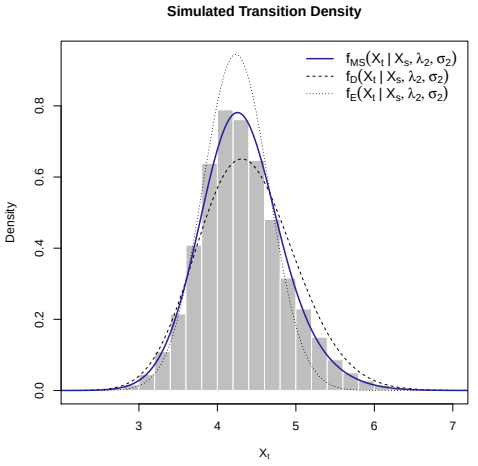
subject to known initial conditions, for example, $p_{ij}(s) = \mathbb{I}_{i=j}$. Using Equation 5.1.2 in conjunction with Equation 5.1.3, we can then approximate the transitional density by a mixture of surrogate densities that carry the moments into a transition density approximation. Figure 5.1.1 illustrates the approximation for both initial states of the parameter chain calculated under the parameter set $\{\lambda_1, \lambda_2, \sigma_1, \sigma_2, -\beta_{11} = \beta_{12}, -\beta_{22} = \beta_{21}, X_0\} = \{5, 8, 0.25, 1, 3, 6, 4\}$ at time $t = 0.1$. Assuming that the initial state of the parameter chain is not known exactly, but that we can assign an initial distribution to the parameter chain, we combine the results for the distinct initial conditions in order to formulate an approximation of the unconditional transition density. That is, let $\tilde{f}(X_t|X_s)$ denote the unconditional density and $\pi_s = (p_{11}(s), p_{22}(s))$ denote the initial distribution of the parameter chain, then we may calculate:

$$\tilde{f}(X_t|X_s) = p_{11}(s)f_{MS}(X_t|X_s, \lambda_1, \sigma_1) + p_{22}(s)f_{MS}(X_t|X_s, \lambda_2, \sigma_2). \tag{5.1.8}$$

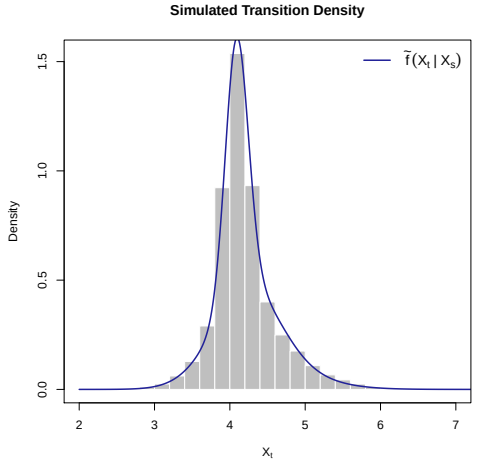
Based on this example, we can see that it is certainly possible to devise a strategy by which an approximation of the transitional density can be calculated. Using the techniques developed in this thesis, this can be achieved for a wide range of non-linear, time-inhomogeneous diffusion models. For example, using the **DiffusionRgqd** package it is possible to approximate the moments of Equation 5.1.4 under the assumed parameter set by defining a time-inhomogeneous diffusion where the Markov-switching elements have been replaced by their respective expectation trajectories:



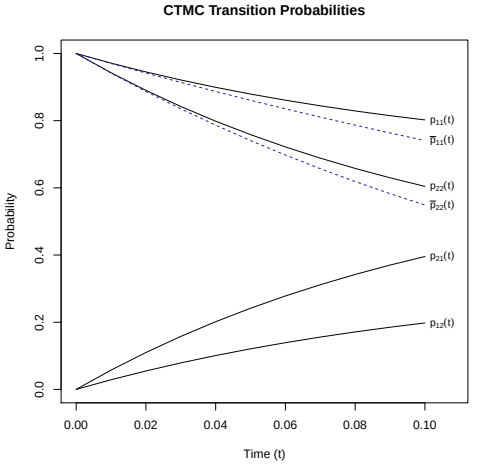
(a) Density for initial parameter state (λ_1, σ_1).



(b) Density for initial parameter state (λ_2, σ_2).



(c) Density for initial distribution of $\pi_0 = (0.6, 0.4)$ for the CTMC.



(d) Transition probabilities for the CTMC that governs the evolution of the parameter chain.

FIGURE 5.1.1: Frequency distributions at time $t = 0.1$ for Equation 5.1.4 under the parameter set $\{\lambda_1, \lambda_2, \sigma_1, \sigma_2, -\beta_{11} = \beta_{12}, -\beta_{22} = \beta_{21}, X_0\} = \{5, 8, 0.25, 1, 3, 6, 4\}$ calculated by simulating numerous trajectories of the process for various initial conditions. Superimposed are approximate density functions calculated using the mixture factorization. In addition, we show transition probabilities and the probabilities of remaining in a given initial state (denoted $\bar{p}_{kk}(t)$ for $k = 1, 2$) over the assumed transition horizon. R code: Supplementary materials, Section 5.1.

```
R>  lambda <- c(5, 8)     # Drift parameters
R>  sigmas <- c(0.25, 1)  # Diffusion parameters
R>  betas  <- c(3, 6)     # CTMC parameters beta12 and beta21
R>
R>  # CTMC transition probabilities and mean trajectories:
R>  p11 <- function(t){(betas[2] + betas[1]*exp(-sum(betas)*t))/sum(betas)}
R>  p12 <- function(t){betas[1]*(1 - exp(-sum(betas)*t))/sum(betas)}
R>
R>  m1 <- function(t){lambda[1]*p11(t)   + lambda[2]*p12(t)}
R>  m2 <- function(t){sigmas[1]^2*p11(t) + sigmas[2]^2*p12(t)}
R>
R>  # Time-inhomogeneous diffusion:
R>  GQD.remove()
R>  G0  <- function(t){m1(t)}
R>  G1  <- function(t){-1}
R>  Q1  <- function(t){m2(t)}
R>  fM1 <- GQD.density(Xs = 4, Xt = seq(2, 10, 0.01), s = 0, t =  0.1, delt =
    0.01, Trunc = c(6, 4))
```

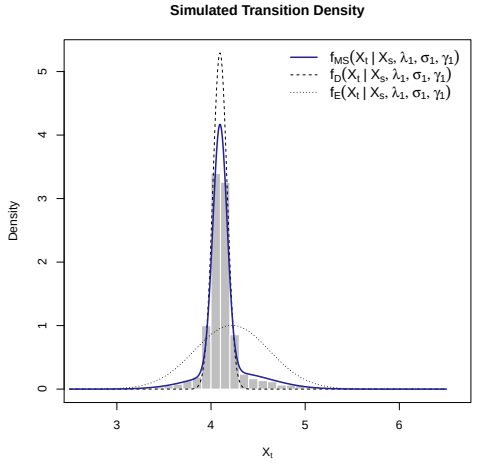
Here we approximate the moment equations of the Markov-switching diffusion by that of a standard time-inhomogeneous diffusion, however, since no account is made of the effect of changes in the parameter state on the density, the density approximation calculated by the `GQD.density()` function will necessarily be incorrect. Using the factorization strategy, we can remedy this by contrasting the dynamics of the non-switching counterpart to that of the Markov-switching diffusion. However, as noted in the introduction of this section, there are a number of subtleties that require careful consideration. For example, in order to employ the factorization we calculate the moments of the excess distribution by integrating and inverting Equation 5.1.2. Although computationally this follows quite naturally, it turns out that the moments that result from this inversion are often quite difficult to use in practice. Unfortunately, it seems, the behaviour of the excess distribution is such that it can be difficult to approximate using its moments in conjunction with a surrogate density: For example, in the calculation of Equation 5.1.2 under Equation 5.1.4, we were unable to employ the saddlepoint approximation in order to approximate the excess distribution. Instead, we had to use a 6-th order Normal type Pearson density (see Section 2.3.3) in order to get a valid approximation.

Based on these observations, we conclude that as a starting point, Markov-switching diffusion models can be analysed using techniques similar to those developed in this thesis, however, it remains to be seen if we can establish a technique which requires less manual *tuning* of the components of the approximation before a self-contained framework such as that of Chapter 4 can be developed and a reliable software package can be built. However, should this be feasible, the implications are quite interesting from a practical perspective. For

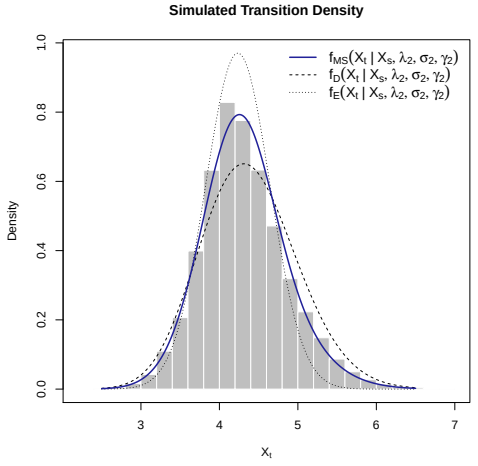
example, using the Markov-switching class of models, it is possible to define a Markov-switching diffusion in such a way that it not only switches parameter states, but also implicitly switches specification: Consider a slight modification of Equation 5.1.4:

$$dX_t = (\lambda_t - X_t)dt + \sqrt{\sigma_t^2 + \gamma_t^2 X_t} dB_t \tag{5.1.9}$$

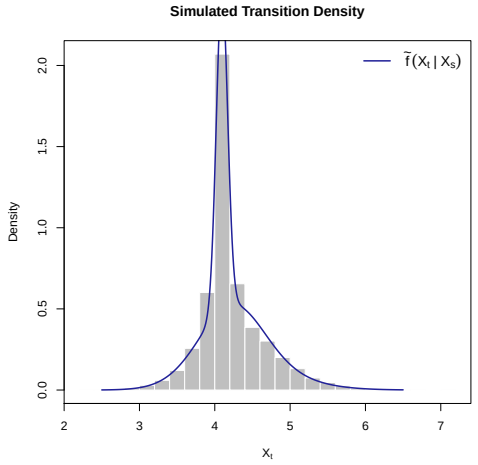
where the pair $(\lambda_t, \sigma_t, \gamma_t) \in \{(\lambda_1, \sigma_1, \gamma_1), (\lambda_2, \sigma_2, \gamma_2)\}$ switches states according to a two dimensional CTMC with transition rate matrix $\mathbf{R} = (\beta_{ij})_{2\times 2}$. Now, set $\{\lambda_1, \lambda_2, \sigma_1, \sigma_2, \gamma_1, \gamma_2, -\beta_{11} = \beta_{12}, -\beta_{22} = \beta_{21}, X_0\} = \{5, 8, 0.25, 0, 0, 1, 3, 6, 4\}$. Here, when the parameter vector switches from state 1 to state 2, the specification of the diffusion changes from an Ornstein-Uhlenbeck process to that of a CIR process with alternative parameters. Using the factorization strategy we can still replicate the transitional density with a reasonable degree of accuracy: Figure 5.1.2 illustrates the transition density approximation for both initial states of the parameter chain of Equation 5.1.9, the unconditional density and the transition probabilities of the CTMC component of the model as before.



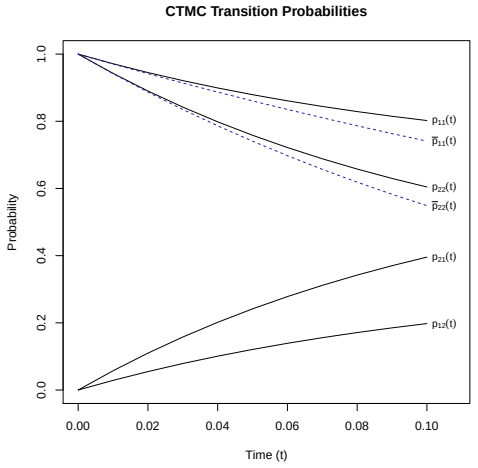
(a) Density for initial parameter state $(\lambda_1, \sigma_1, \gamma_1)$.



(b) Density for initial parameter state $(\lambda_2, \sigma_2, \gamma_2)$.



(c) Density for initial distribution of $\pi_0 = (0.5, 0.5)$ for the CTMC.



(d) Transition probabilities for the CTMC that governs the evolution of the parameter chain.

FIGURE 5.1.2: Frequency distributions at time $t = 0.1$ for Equation 5.1.9 under the parameter set $\{\lambda_1, \lambda_2, \sigma_1, \sigma_2, \gamma_1, \gamma_2, -\beta_{11} = \beta_{12}, -\beta_{22} = \beta_{21}, X_0\} = \{5, 8, 0.25, 0, 0, 1, 3, 6, 4\}$ calculated by simulating numerous trajectories of the process for various initial conditions. Superimposed are approximate density functions calculated using the mixture factorization. In addition, we show transition probabilities and the probabilities of remaining in a given initial state (denoted $\bar{p}_{kk}(t)$ for $k = 1, 2$) over the assumed transition horizon. R code: Supplementary materials, Section 5.2.

5.1.2.2 Likelihood inference for Markov-switching diffusions

Following from the analysis of Section 5.1.2.1, we probe the possibility of conducting inference on Markov-switching diffusion models. Since we have shown that it is possible to approximate the transition density of a Markov-switching diffusion process, it should conceivably be possible to calculate the likelihood function of a Markov-switching diffusion model. Given some discretely observed process for which we can observe both the state of the diffusion process as well as the states of the underlying parameter chain $D_M = \{(\mathbf{X}_{t_1}, \boldsymbol{\theta}_{t_1}), (\mathbf{X}_{t_2}, \boldsymbol{\theta}_{t_2}), \ldots, (\mathbf{X}_{t_n}, \boldsymbol{\theta}_{t_n})\}$, one possible formulation of the likelihood function is the expression:

$$L(\boldsymbol{\theta}|D_M) \propto f(\mathbf{X}_{t_2}|\mathbf{X}_{t_1}, \boldsymbol{\theta}_{t_1}) P(\boldsymbol{\theta}_{t_1}) \prod_{i=2}^{n-1} f(\mathbf{X}_{t_{i+1}}|\mathbf{X}_{t_i}, \boldsymbol{\theta}_{t_i}) P(\boldsymbol{\theta}_{t_i}|\boldsymbol{\theta}_{t_{i-1}}), \quad (5.1.10)$$

where $f(\mathbf{X}_{t_{i+1}}|\mathbf{X}_{t_i}, \boldsymbol{\theta}_{t_i})$ denotes the transitional density of the Markov-switching diffusion model with initial parameter state $\boldsymbol{\theta}_{t_i}$, $P(\boldsymbol{\theta}_{t_{i+1}}|\boldsymbol{\theta}_{t_i})$ denotes the probability of the parameter chain switching from state $\boldsymbol{\theta}_{t_i}$ to state $\boldsymbol{\theta}_{t_{i+1}}$ and $P(\boldsymbol{\theta}_{t_1})$ denotes the initial distribution of the parameter chain. Equation 5.1.10 is thus constructed from the transition density of the Markov-switching diffusion given that the state of the parameter chain is known at the start of each transition horizon and the sequence of transition densities associated with the observed states of the parameter chain. Using the methodology of Section 5.1.2.1, it is thus possible to approximate Equation 5.1.10 for non-linear, time-inhomogeneous Markov-switching diffusions: By approximating $f(\mathbf{X}_{t_{i+1}}|\mathbf{X}_{t_i}, \boldsymbol{\theta}_{t_i})$ on each transition horizon and calculating $P(\boldsymbol{\theta}_{t_{i+1}}|\boldsymbol{\theta}_{t_i})$ using the transition probabilities for the CTMC part of the process, we can calculate parameter estimates from a discretely observed dataset by approximating Equation 5.1.10. Consider for example a CIR process with Markov-switching volatility:

$$dX_t = \alpha(\beta - X_t)dt + \sigma_t\sqrt{X_t}dB_t, \quad (5.1.11)$$

where σ_t alternates between states σ_1 and σ_2 according to a CTMC with transition rate matrix $R = (\beta_{ij})_{2\times 2}$. For purposes of the illustration we shall assume that σ_1 and σ_2 denote low and high volatility states respectively (i.e., $\sigma_1 < \sigma_2$). Figure 5.1.3 illustrates a simulated trajectory for both Equation 5.1.11 and the volatility process under the parameter set $\{\alpha, \beta, \sigma_1, \sigma_2, -\beta_{11} = \beta_{12}, -\beta_{22} = \beta_{21}\} = \{1, 5, 0.25, 1, 0.25, 1\}$. Using a modified Euler-Maruyama scheme, we generate observations at equispaced intervals, $t_{i+1} - t_i = 0.1$ time units apart on the observation horizon $[0, 200]$. The effect of changes in the volatility of the process can clearly be seen in the trajectory of the diffusion process.

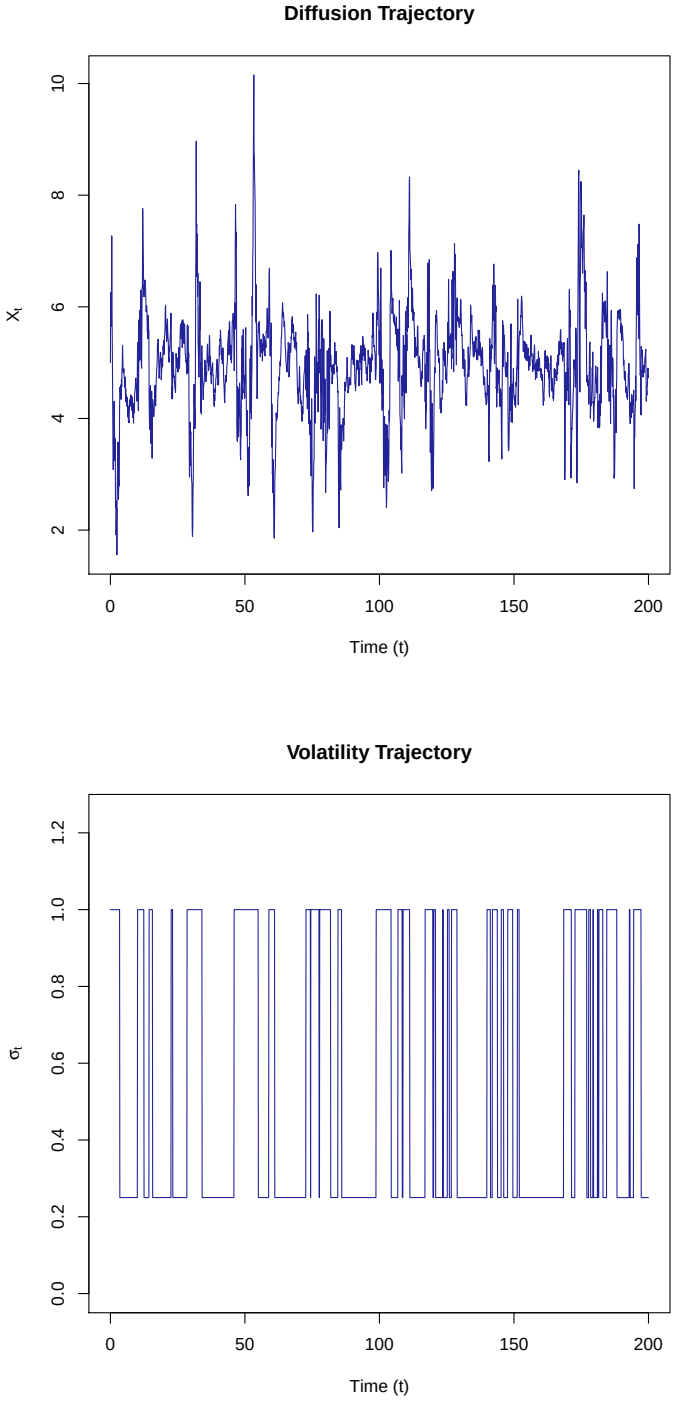


FIGURE 5.1.3: Simulated CIR process with Markov-switching volatility (top) and the underlying volatility trajectory (bottom) for the parameter set $\{\alpha, \beta, \sigma_1, \sigma_2, -\beta_{11} = \beta_{12}, -\beta_{22} = \beta_{21}\} = \{1, 5, 0.25, 1, 0.25, 1\}$. Observations are made at equispaced epochs 0.1 time units apart on the observation horizon $[0, 200]$. R code: Supplementary materials, Section 5.3.

Based on the mixture factorization, we approximate the likelihood function for Equation 5.1.11 under Equation 5.1.10 and calculate parameter estimates for the simulated trajectory in Figure 5.1.3. Table 5.1.1 gives the resulting parameter estimates and 90% credibility intervals calculated using the RWMH algorithm. Indeed, the parameter estimates match the true parameter set quite closely.

Parameter	True Value	Estimate	90% CI
α	1.00	1.079	(0.972, 1.219)
β	5.00	5.013	(4.947, 5.076)
σ_1	0.25	0.250	(0.242, 0.258)
σ_2	1.00	1.017	(0.967, 1.080)
$-\beta_{11} = \beta_{12}$	0.25	0.254	(0.200, 0.316)
$-\beta_{22} = \beta_{21}$	1.00	0.903	(0.701, 1.079)

TABLE 5.1.1: Parameter estimates and 90% credibility intervals for Equation 5.1.11 under the simulated dataset in Figure 5.1.3 calculated using the complete data likelihood using the true volatility trajectory. Estimates are calculated using 100 000 updates of the RWMH algorithm with a burn-in period of 10 000 iterations. R code: Supplementary materials, Section 5.3.

Although this experiment illustrates that it is possible to conduct likelihood inference on non-linear Markov-switching models when both the diffusion and parameter trajectory can be observed, it can easily be argued that this construction is somewhat unrealistic: In practice, the true states of the parameter chain are not observed. The problem thus becomes one of conducting inference on a Markov-switching diffusion model which depends on an unobserved/latent parameter chain. More precisely, the trajectory of the parameter chain is not known at the observation times of the diffusion process. Consequently, a 'complete-data' likelihood function such as Equation 5.1.10 will most likely not be fit for application in real world scenarios. Instead, we are required to construct a scheme by which the inference can be conducted based on the diffusion trajectory alone. Unsurprisingly, this can be quite difficult.

To get some perspective on the problem it is worth making a reference to jump diffusion models, as there are some similarities between Markov-switching and jump diffusion models. In the case of jump diffusion models, it is possible to perform inference accurately without directly observing any jump events. That is, by formulating a likelihood function that depends solely on the observations of the state of the jump diffusion, without directly observing jumps in the trajectory of the diffusion, we may still infer the structure of the jump mechanism with a high degree of accuracy. Since the jump events are embedded directly in the

trajectory of the process, the natural contrast between the diffusion dynamics and the jump mechanism is typically high enough to distinguish the jump dynamics indirectly with a high degree of accuracy – provided of course that the data is of sufficiently high resolution. In the case of Markov-switching diffusions, it appears to be significantly more difficult to identify the parameters of the latent components of the process as compared to jump diffusion models even though the behaviour of the transitional densities of the respective processes are superficially similar. Intuitively, it makes sense that this should be the case: Whereas jump events in a jump diffusion model manifest as instantaneous and direct changes in the state of the diffusion trajectory, instantaneous changes in the parameter process of a Markov-switching diffusion take time to manifest in the trajectory of the diffusion process. Moreover, since the trajectory of the process is only partially observed at discrete epochs to begin with, changes in the parameter process are further distorted by time lapses between observations. As such, we are tasked with distinguishing both the size and rate of change in the states of the parameter process indirectly from a diffusion path which is already only partially observed.

Despite the aforementioned technical considerations, we may nevertheless attempt to perform inference on a Markov-switching diffusion model for which the states of the parameter chain are unobserved. That is, we attempt to estimate the parameters of a hidden Markov diffusion model. Let $D_S = \{\mathbf{X}_{t_1}, \mathbf{X}_{t_2}, \ldots, \mathbf{X}_{t_n}\}$ denote a discretely observed trajectory of a multivariate diffusion process as before and denote the latent parameter process by $Q_S = \{\boldsymbol{\theta}_{t_1}, \boldsymbol{\theta}_{t_2}, \ldots, \boldsymbol{\theta}_{t_n}\}$. Following the notational framework of Zucchini and MacDonald (2009) as applied to a general hidden Markov Model, we construct the likelihood for a hidden Markov diffusion model as follows: Let $\boldsymbol{\pi}_s = \{\pi_1, \pi_2, \ldots, \pi_q\}'$ denote the stationary distribution of the parameter chain, and let $\mathbf{P}_{s,t} = (p_{ij}(t))_{q\times q}$ denote the transition probability matrix of the parameter process where

$$\frac{\partial}{\partial t} p_{ij}(t) = \sum_{k=1}^{q} \beta_{ik} p_{kj}(t) \tag{5.1.12}$$

subject to the initial conditions $p_{ij}(s) = \mathbb{I}_{i=j}$. Then, construct the diagonal matrix:

$$\mathbf{F}_{s,t} = \begin{bmatrix} f_{MS}(\mathbf{X}_t|\mathbf{X}_s, \boldsymbol{\theta}^{(1)}) & 0 & \cdots & 0 \\ 0 & f_{MS}(\mathbf{X}_t|\mathbf{X}_s, \boldsymbol{\theta}^{(2)}) & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & & f_{MS}(\mathbf{X}_t|\mathbf{X}_s, \boldsymbol{\theta}^{(q)}) \end{bmatrix}. \tag{5.1.13}$$

Then the likelihood of the hidden Markov diffusion model is given by:

$$L(\boldsymbol{\theta}|D_S) = \boldsymbol{\pi}'_{t_1}\mathbf{F}_{t_1,t_2}\mathbf{P}_{t_1,t_2}\mathbf{F}_{t_2,t_3}\ldots\mathbf{P}_{t_{n-2},t_{n-1}}\mathbf{F}_{t_{n-1},t_n}\mathbf{1} \tag{5.1.14}$$

where $\mathbf{1}$ denotes a column vector of ones. Note that strictly speaking, this is an approximate likelihood, as the transition density, though dependent on the initial state of the parameter chain, does not account for the terminal value of the parameter chain over each transition horizon. Also, the transitional density itself is constructed from approximate moments. Nevertheless, should the transition horizons be sufficiently short, the approximation should be sufficiently accurate for performing inference.

Using Equation 5.1.14, we can thus calculate parameter estimates for a Markov-switching diffusion model for which only the diffusion component of the process is observed. For example, under the simulated dataset for Equation 5.1.11 in Figure 5.1.3, we calculate parameter estimates from the diffusion trajectory alone under Equation 5.1.14 using the RWMH algorithm. Table 5.1.2 gives the resulting parameter estimates and 90% credibility intervals. When compared to the true parameter set, it is clear that the transition rate parameters are somewhat underestimated. However, considering that the transition rate parameters are estimated without any direct observations on the parameter chain, the results are satisfactory. Indeed, it can be expected that the estimates associated with the Markov-switching component will most certainly be affected by the sample resolution and the estimates generally improve as the observation horizon becomes longer – a property that once again bears a resemblance to the jump mechanism of a jump diffusion model.

Parameter	True Value	Estimate	90% CI
α	1.00	1.003	(0.859, 1.148)
β	5.00	5.020	(4.913, 5.061)
σ_1	0.25	0.252	(0.242, 0.258)
σ_2	1.00	1.026	(0.978, 1.091)
$-\beta_{11} = \beta_{12}$	0.25	0.200	(0.156, 0.259)
$-\beta_{22} = \beta_{21}$	1.00	0.743	(0.588, 0.928)

TABLE 5.1.2: Parameter estimates and 90% credibility intervals for Equation 5.1.11 under the simulated dataset in Figure 5.1.3 calculated using Equation 5.1.14. Estimates are calculated using 100 000 updates of the RWMH algorithm with a burn-in period of 10 000 iterations. R code: Supplementary materials, Section 5.4.

5.1.2.3 Summary

Based on the analysis we have showcased here, we conclude that there is at least some proof of concept for the development of a more general framework for conducting inference and analysis on Markov-switching diffusion processes. Although the various examples concerning the approximation of transition densities and the likelihood-based inference for Markov-switching diffusions with latent parameter chains indicate that it is certainly possible to analyse complicated models of this class, more work is needed in order to develop a robust and general methodology. Specifically, with regard to the mixture factorization, we need to find a means of approximating the excess distribution that is more robust and less susceptible to numerical instability. With respect to the methodology in general, a number of important issues need to be addressed regarding inference for Markov-switching diffusions: In the examples considered here, we focus on a time-homogeneous non-linear model. Although the methodology can readily be applied to time-inhomogeneous models (see Appendix E.1), this primarily applies to the diffusion component of the process. As such it remains to be seen if the methodology can be generalised to cases where the parameter process is also time-inhomogeneous. Indeed, such a generalisation would likely have a significant effect on how the likelihood is to be calculated. Perhaps a more obvious consideration is the application of the methodology to jump diffusion models. Indeed, we have shown in Section 4.3.3.2 that it is possible to accurately approximate the transition density of a jump diffusion process with Markov-switching intensity using the standard methodology. However, in the case where the Markov-switching component applies not only to the jump mechanism but to the diffusion component as well, we are left with the problem of devising a factorization which can simultaneously account for the diffusion, jump, and Markov-switching dynamics. Finally, perhaps a more interesting point of consideration is decoding the unobserved states of the parameter chain. Indeed, it is possible to extract a local decoding based on the construction of the likelihood in standard fashion (see Appendix E.1), however, since the likelihood is constructed from the transition density function as opposed to true probabilities, the numbers that result from such a calculation can appear dubious at times. As such, more work is required in order to validate such a calculation.

5.1.3 Non-standard first passage times: Trapping problems in ecology

Diffusion models offer a compact description of how processes evolve over time. By specifying coefficients of a diffusion model as functions which depend on

the current state of the process, we can formulate models with dynamics which may change in accordance with the position of the process in the state space. Although diffusion models are more often than not applied to processes which are not physical in nature, for example, the trajectory of the price of an asset or interest rate process, diffusion models can readily be applied to physical processes as well. One example of such an application, originating in the field of statistical ecology, is the modelling of animal movement from tracking data. Authors such as Brillinger (2003), Preisler (2004) and Horne *et al.* (2007) have successfully applied diffusion models in the modelling of animal movements based on direct observations of the movements of various species. Although the validity of representing the dynamics of animal movements using a diffusion model can be debated, in principle all that is required to model such a process is sufficiently high resolution tracking data which documents the position of the animal over time. Naturally, where elusive species are concerned and a given animal may be extremely difficult to find in the first place, the continuous tracking of the trajectory of animal movements for extended periods of time may be impractical if ever feasible. In such cases, researchers often have to resort to alternative tracking techniques such as tag and release programs or setting up camera traps in areas where the animal in question is likely to be observed. Although these techniques have been applied with great success to various species, the telemetry gathered from such studies are not quite suitable for modelling with diffusion processes using existing inference techniques. For example, although techniques such as the setting of camera traps have proved extremely successful in the sighting of elusive species such as Jaguar (Borchers *et al.*, 2014), it can be difficult to distill movement patterns from the telemetry gathered by such devices. This follows since the trapping array – an array of camera traps placed is such a way to form a grid of detection points – remains stationary for long periods of time, and thus only observes movement patterns indirectly when the animal happens to move to within an observable distance of the array sensors. Though various techniques have been developed specifically for analysing such data, this type of set-up creates the opportunity to formulate the problem in terms of a special kind of first passage time problem. Thus, in order to explore the possibility of modelling animal movements from camera-trap data using a diffusion model, we attempt to devise a scheme for conducting such analysis in terms of non-standard first passage time problems.

In order to model the trapping process as a first passage time problem, we need to create a mathematical analogue of the problem. For ease of exposition, while developing theoretical components of the methodology, we shall refer intermittently to the object of interest as an abstract particle (representing the animal) moving in relation to a detection array (representing the camera-trap array). Let $\mathbf{X}_t = \{X_t, Y_t\}'$ denote the coordinate of a particle moving in a

two-dimensional plane and let the trajectory of this particle be governed by a diffusion process with dynamics given by the bivariate SDE:

$$d\mathbf{X}_t = \boldsymbol{\mu}(\mathbf{X}_t, t)dt + \boldsymbol{\sigma}(\mathbf{X}_t, t)d\mathbf{B}_t. \tag{5.1.15}$$

Using Equation 5.1.15, we attempt to replicate the movements of the particle/animal over time. This is achieved by specifying coefficients of the SDE in terms of the current position of the process, thus relating how the trajectory of the particle/animal reacts to its position in the plane. Given an appropriate diffusion model, we can then formulate the trapping/detection problem as a special kind of first passage time problem: In general, we are concerned with the first passage time to an event $A(\mathbf{X}_t, t)$ on the trajectory of the particle. That is, given some initial position for the process, $\mathbf{X}_s$, we wish to analyse the distribution of the random variable:

$$T_{\mathbf{X}_s \to A(\mathbf{X}_t)} = \inf\{t > s : A(\mathbf{X}_t) \text{ occurs}\}. \tag{5.1.16}$$

In this context, the event of interest is that of the first time at which the particle moves to within a detectable distance of the detection array. In order to replicate the trapping problem, we define the first passage time variable in terms of the geometry of the trapping array. This is achieved by mapping the detection points of the trapping array onto the xy-plane: Let $\boldsymbol{\Phi} = \{\boldsymbol{\phi}_i = (x_i^\star, y_i^\star) : i = 1, \ldots, M\}$ denote a set of M coordinates for the elements of the detection array and assume that each detection point has a detection radius of size ϵ. Then the trapping array can be visualised as M loci circumscribed by rings of radius ϵ in the xy-plane as illustrated in Figure 5.1.4. Using this construction, we can define the first passage time variable in terms of the elements of the trapping array where, for some initial point $\mathbf{X}_s$ not within ϵ units of any detection locus, we have

$$T^{\epsilon}_{\mathbf{X}_s \to \boldsymbol{\Phi}} = \inf\left\{t > s : \sum_{i=1}^{M} \text{Ind}(\|\mathbf{X}_t - \boldsymbol{\phi}_i\| < \epsilon) = 1\right\}. \tag{5.1.17}$$

To use an alternative metaphor, the first passage time variable can thus be thought of as the time it takes for a billiards ball to enter any pocket on an edgeless table, where the pockets are placed at the coordinates in $\boldsymbol{\Phi}$ and the trajectory of the ball is governed by a bivariate diffusion process.

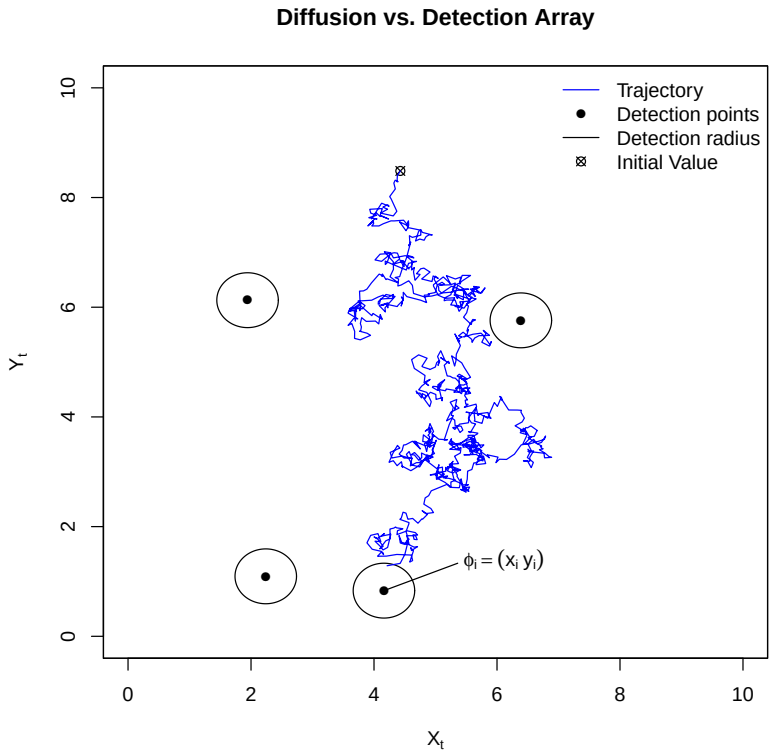


FIGURE 5.1.4: Path of a diffusion process (blue) moving in relation to an array of detection points (black dots) with fixed detection radius (black circles). At first contact with a trapping radius, the first passage time is recorded.

Before continuing the analysis, we note some important attributes of the first passage time problem considered here: Although we have considered bivariate problems earlier, the geometry of the problems in Section 3.3 concern the escape of a diffusion process from some pre-defined region. In that context, we could formulate the problem by tracing a smooth perimeter in the xy-plane and isolate a finite region of the state space within which calculations take place. In contrast, the first passage time event defined by Equation 5.1.17 implies that we are dealing with an infinite region with small holes through which the process 'escapes'. Consequently, the first passage time is affected not only by the placement of the detection points but also by magnitude of the detection radius.

In the sections that follow, we detail preliminary techniques for analysing trapping problems under the dynamics of a diffusion model. We showcase two strategies that we devised in an attempt to perform inference using camera trap telemetry data.

5.1.3.1 Scheme 1: A pair-wise detection scheme based on the Volterra equation

We proceed to develop an observational scheme that can be used to infer the dynamics of a diffusion process from first passage time data arising from a set of stationary detection points, spread throughout the state-space of the process of interest. In developing a scheme for performing inference on a diffusion model from camera-trap data, we first consider the simple case of a single detection point. Formally: Given some initial coordinate $\mathbf{X_s}$ we consider the case of the first passage time to a single node – ignoring all other nodes – on the trapping array, $\boldsymbol{\phi}_i \in \boldsymbol{\Phi}$ for some i. In this case, the first passage time variable follows:

$$T^{\epsilon}_{\mathbf{X}_s \to \boldsymbol{\phi}_i} = \inf\left\{t > s : \|\mathbf{X}_t - \boldsymbol{\phi}_i\| < \epsilon\right\}. \tag{5.1.18}$$

By transforming the process to polar coordinates centred at the detection point $\boldsymbol{\phi}_i$, i.e., $\mathbf{X}_t \leftrightarrow (r_t, \theta_t)$ where r_t and θ_t denote the distance and direction from $\boldsymbol{\phi}_i$ respectively, we can reformulate the first passage time problem in terms of a fixed barrier problem. That is, let $h(r_t, \theta_t)$ denote the transition density for the transformed process, we may evaluate the density of $T^{\epsilon}_{\mathbf{X}_s \to \boldsymbol{\phi}_i}$ via a slight modification of the Volterra equation (see Section 3.2):

$$\int h(\epsilon, \theta_t | r_s = \|\mathbf{X}_s - \boldsymbol{\phi}_i\|, \theta_s) d\theta_t = \int_s^t g^{\epsilon}_{\mathbf{X}_s \to \boldsymbol{\phi}_i}(u) \iint h(\epsilon, \theta_t | \epsilon, \theta_u) d\theta_u d\theta_t du, \tag{5.1.19}$$

where the integration with respect to θ accounts for the direction from which the detection radius is breached and θ_s on the LHS of Equation 5.1.19 can be expressed as $\theta_s = \arccos((X_s - x_i^{\star})/\sqrt{(X_s - x_i^{\star})^2 + (Y_s - y_i^{\star})^2})$. Using this transformation, we are essentially changing the problem to a marginal first passage time problem with respect to time it takes for the distance from the initial state of the process to decline below ϵ units. Unfortunately, Equation 5.1.19 is extremely difficult to work with. However, by invoking the assumption that the trapping radius is small relative to the area under consideration (in practice, this is not an unrealistic assumption, for example Jaguar typically have territories on the order of $\pm$ 50 km^2 whilst camera traps typically have a detection range of only a couple of metres), and letting ϵ tend to zero, one can see that the polar coordinate transformation effectively becomes redundant and we can write the equation in terms of the untransformed density as:

$$f(\mathbf{X}_t = \boldsymbol{\phi}_i | \mathbf{X}_s) = \int_s^t g_{\mathbf{X}_s \to \boldsymbol{\phi}_i}(u) f(\mathbf{X}_t = \boldsymbol{\phi}_i | \mathbf{X}_u = \boldsymbol{\phi}_i) du. \tag{5.1.20}$$

Using Equation 5.1.20, we can thus analyse the first passage time distribution with respect to a single detection point. For example, consider a particle with dynamics given by the SDE:

$$\begin{aligned} dX_t &= \alpha_x(\beta_x - X_t) + \sigma_x\sqrt{X_t}dB_t^{(1)} \\ dY_t &= \alpha_y(\beta_y - Y_t) + \sigma_y\sqrt{Y_t}dB_t^{(2)}, \end{aligned} \tag{5.1.21}$$

and set $\{\alpha_x, \beta_x, \sigma_x, \alpha_y, \beta_y, \sigma_y, \} = \{1.25, 5, 1.25, 1.25, 5, 1.25\}$, $\mathbf{X}_s = (X_0, Y_0) = (4, 4)$, and $\boldsymbol{\phi} = (5, 5)$. By solving Equation 5.1.20 numerically, we can analyse the distribution $g_{\mathbf{X}_s \to \boldsymbol{\phi}}(t)$ of the variable $T_{\mathbf{X}_s \to \boldsymbol{\phi}}$ at fixed points along the time horizon. This is achieved by applying a quadrature rule to the integral equation in order to derive an iterative updating scheme as in Section 3.2. As before, this calculation relies on the calculation of the transitional density of the underlying diffusion model. For these purposes, we make use of the cumulant truncation scheme in conjunction with the bivariate saddlepoint distribution (See Section 2.3). Thus, by combining a numerical approximation of the transitional density with a numerical approximation of Equation 5.1.20, we are able to approximate the first passage time density of Equation 5.1.21 transiting from $\mathbf{X}_s = (4, 4)$ to the detection point $\boldsymbol{\phi} = (5, 5)$. Figure 5.1.5 shows the resulting first passage time density. For reference, we have included a frequency distribution for simulated first passage times under the specification of Equation 5.1.21. In order to simulate first passage times for a single detection point, we use a modified discrete simulation scheme that allows us to simulate the first passage time of a bivariate diffusion to an exact coordinate (see Appendix E.2).

Using the Volterra equation, we can thus calculate the first passage time density for a bivariate diffusion process hitting a detection point placed in a two-dimensional plane. In order to apply this to the trapping inference problem, we need to formulate a likelihood function based on the first passage time distribution of the underlying diffusion model to various points on the detection array. In the case where the number of detection points is small, and the detection points are sparsely distributed, this can be achieved using independent first passage time densities for traversing between detection points. That is, for a detection array with M detection points, we calculate $M!/(M-2)!$ first passage time densities – one for each possible traversal of the state space between pairs of detection points. We refer to this as the pair-wise scheme. The premise of this scheme is that, when the detection array is sufficiently sparse, it suffices to treat time lapses between being observed at posts as independent first passage time events. In doing so, we formulate the likelihood by focusing on first passage time variables for which the Volterra equation can be used in order to calculate the density. For example, given a set of observed first passage times between detection points, it

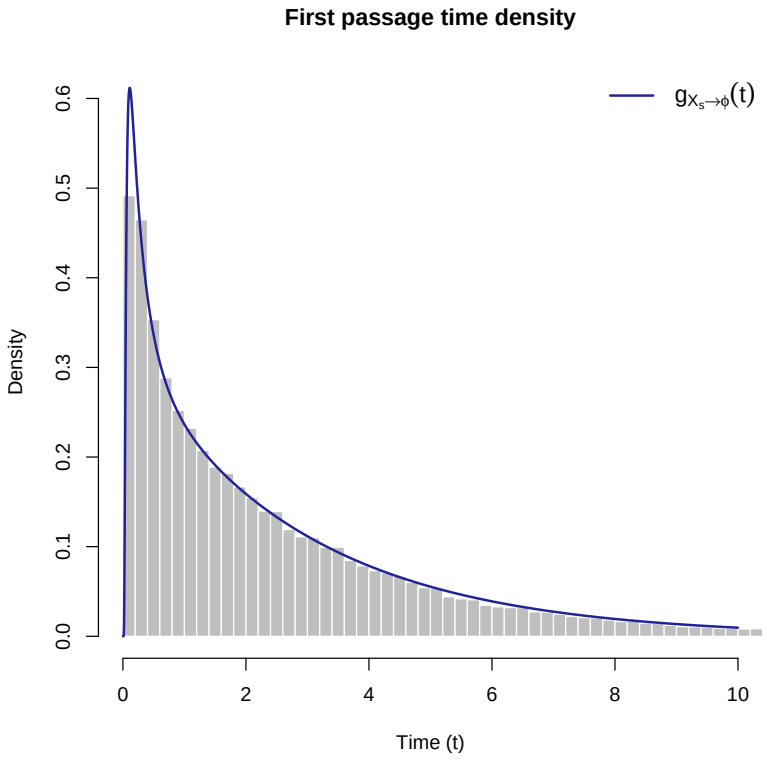


FIGURE 5.1.5: Simulated and approximate first passage time distribution for Equation 5.1.21 from a known initial value to the exact coordinate of a detection point. The approximate first passage time density, calculated by numerically solving Equation 5.1.20, matches the shape of the simulated distribution quite closely. R code: Supplementary materials, Section 5.5.

is possible to construct an approximate likelihood function:

$$L(\theta|T_s) \propto \prod_{i \neq j} \prod_{l=1}^{n^{i,j}} g_{\phi_i \to \phi_j}(\tau_l^{i,j}), \tag{5.1.22}$$

where $n^{i,j}$ denotes the number of first passage times, $\tau^{i,j}$, from detection point ϕ_i to detection point ϕ_j and T_S denotes the collection of all observed traversals. Note that $\tau_l^{i,j}$ is given by the time difference between the l-th traversal between points ϕ_i and ϕ_j that does not overlap with previous ϕ_i-ϕ_j traversals.

We conclude the groundwork construction of the pairwise scheme by noting some technical considerations: In light of the substantial difficulties that arise when

solving joint first passage time problems, the scheme is developed by only focusing on the marginal first passage times to detection. Under the assumptions that the detection array is relatively sparse, and that the effect of the detection radius is negligible, we can employ the Volterra equation in order to calculate a first passage time densities on which we can base a likelihood function. Thus, at the expense of arguably unrealistic assumptions, the problem is made tractable and can be solved with a reasonable degree of efficiency. However, by ignoring the probability of hitting another detection point before hitting a given detection point, some informational efficiency is lost since the order in which distinct traversals occur do not add to the likelihood. For a sufficiently large number of transitions, this information loss is assumed to negligible. However, in the context of applying the methodology to camera trap data, informational efficiency is key since it is unlikely that a large number of traversals can be recorded within the time frame of such a study. In the section that follows, we consider a more general approach and develop a framework for conducting the analysis which is more consistent with the problem at hand.

5.1.3.2 Scheme 2: An interior value problem approach

Although the scheme developed in Section 5.1.3.1 shows some promise for solving the trapping/detection problem, the method has a number of drawbacks that may render the strategy unfit for application in real-world scenarios. Specifically, although the scheme allows one to make computational efficiency gains through the Volterra equation, the informational efficiency of the scheme is compromised by the observation strategy with respect to calculating a usable likelihood function. Indeed, one of the primary considerations when setting up a camera trap array is the length of time for which the study can/should be conducted. Since such studies usually have to be discontinued at some point, it is important to maximise the informational efficiency of the techniques used to analyse the data. With an aim to improve on the informational efficiency of the pair-wise scheme of Section 5.1.3.1, we consider the more general case of calculating the joint first passage time density of a diffusion process moving to any point on the detection array. That is, we wish to calculate the density of the first passage time variable as defined originally in Equation 5.1.17.

For the pair-wise scheme, it is possible to formulate an integral equation for the first passage time density in terms of the transitional density of the process. Unfortunately, when considering multiple detection points, it may not be possible to construct an appropriate integral equation for the first passage time density. Indeed, such an integral equation will likely consist of a system of equations which simultaneously discounts for the probability of first passage to individual trapping

points in the detection array. Thus, in order to make the problem tractable, we consider an alternative approach to calculating the joint first passage time density: For these purposes, we revisit the strategy outlined in Section 3.3 whereby the first passage time density is calculated by way of a partial differential equation. That is, by framing the first passage time problem in terms of a special type of boundary value problem, we may calculate the appropriate first passage time density from a PDE for the survival probability function of a diffusion process moving in relation to a detection array. Formally, given a time-homogeneous diffusion process $\mathbf{X}_t = \{X_t^{(1)}, X_t^{(2)}\}$ starting in coordinate $\mathbf{X}_s$, the survival probability function:

$$S^{\epsilon}_{\mathbf{X}_s \to \mathbf{\Phi}}(t) = P(T^{\epsilon}_{\mathbf{X}_s \to \mathbf{\Phi}} > t) \tag{5.1.23}$$

is governed by the PDE:

$$\frac{\partial S^{\epsilon}_{\mathbf{X}_s \to \mathbf{\Phi}}(t)}{\partial t} = \sum_{i=1}^{2} \mu_i(\mathbf{X}_t, t) \frac{\partial S^{\epsilon}_{\mathbf{X}_s \to \mathbf{\Phi}}(t)}{\partial X_t^{(i)}} + \sum_{i=1}^{2}\sum_{j=1}^{2} \gamma_{ij}(\mathbf{X}_t, t) \frac{\partial^2 S^{\epsilon}_{\mathbf{X}_s \to \mathbf{\Phi}}(t)}{\partial X_t^{(i)} \partial X_t^{(j)}}, \tag{5.1.24}$$

subject to the boundary conditions:

$$S^{\epsilon}_{\mathbf{X}_s \to \mathbf{\Phi}}(u) = \begin{cases} 0 & \text{for } \|\mathbf{X}_u - \boldsymbol{\phi}_i\| < \epsilon \quad \forall \quad i = 1, \ldots, M \text{ and } u \in [s, t], \\ 1 & \text{for } u = s \text{ otherwise.} \end{cases} \tag{5.1.25}$$

Equations 5.1.24 and 5.1.25 together constitute an interior value problem on an open region on a two-dimensional plane. Here, by 'interior value problem', we mean that, as opposed to the traditional nomenclature where the PDE is framed in the context of a boundary value problem with well-defined boundary values, the boundary conditions of the problem are defined on the interior of the space on which the problem is defined. That is, as opposed to defining the problem on a finite section of the xy-plane where some or all of the values of the function that satisfies the PDE in question are known at the boundaries for the entire transition horizon, the region in which the problem is framed is not finite and values of the function that satisfies the PDE in question are known only at points in the interior of this region for the entire transition horizon. Naturally, this has important practical implications with regard to solving Equation 5.1.24: Since a closed-form solution to Equation 5.1.24 cannot be found in general, we resort to solving the interior value problem numerically. Since this typically involves the construction of a discrete analogue to the region on which the problem is framed, we need to consolidate the finite framework of a numerical solution with the non-finite geometry of the stated problem. In order to do this we first consider a mixed interior/boundary value problem for the detection array: Consider the case where the detection array is bounded by a perimeter that forms an enclosed region, say $\Omega \subseteq \mathbb{R}^2$. If we assume that the particle/animal may also be detected at

the perimeter, then we can reformulate the problem as a mixed interior/boundary value problem by modifying the interior/boundary conditions of Equation 5.1.24 accordingly. That is, the first passage time variable assumes the form:

$$T^{\epsilon}_{\mathbf{X}_s \to \mathbf{\Phi}} = \inf \left\{ t > s : \sum_i \text{Ind}(\|\mathbf{X}_t - \boldsymbol{\phi}_i\| < \epsilon) = 1 \text{ or } \mathbf{X}_t \notin \Omega \right\}, \tag{5.1.26}$$

and the appropriate boundary conditions are given by:

$$S^{\epsilon}_{\mathbf{X}_s \to \mathbf{\Phi}}(u) = \begin{cases} 0 & \text{for } \|\mathbf{X}_u - \boldsymbol{\phi}_i\| < \epsilon \text{ and } \mathbf{X}_u \notin \Omega \quad \forall \quad i = 1, \ldots, M \text{ and } u \in [s, t], \\ 1 & \text{for } u = s \text{ otherwise.} \end{cases} \tag{5.1.27}$$

Thus, in contrast to the Volterra equation, the structure of Equation 5.1.24 is invariant to the stopping rule defined by the first passage time variable, and the mechanics of calculating of the first passage time density for a trapping problem are quite similar to the standard bivariate problems considered in Section 3.3, regardless of how many detection points are present in the trapping array. The reason for this apparent invariance follows from the fact that the problem is framed directly in terms of the probability flow of the process on the applicable state-space. By imposing the geometry of the first passage time event on the equation by way of its boundary conditions, the evolution of the corresponding survival probability surface is then dictated by the PDE.

In order to solve Equation 5.1.24, we may employ standard numerical techniques for the analysis of partial differential equations. For our purposes, we resort again to using the method of lines. For example, consider a particle on a two-dimensional plane with dynamics governed by the SDE:

$$\begin{aligned} dX_t &= \mu_x(X_t, Y_t)dt + \sigma_x(X_t, Y_t)dB^{(1)}_t \\ dY_t &= \mu_y(X_t, Y_t)dt + \sigma_y(X_t, Y_t)dB^{(2)}_t. \end{aligned} \tag{5.1.28}$$

Now, let Ω denote a square region with limits $\{\lambda^x_L, \lambda^x_U, \lambda^y_L, \lambda^y_U\}$ and superimpose a lattice of equispaced nodes $\mathcal{L} = \{l_{ij} = (x_i, y_j) : i, j = 0, 1, \ldots N, x_0 = \lambda^x_L, x_N = \lambda^x_U, y_0 = \lambda^y_L, y_N = \lambda^y_U\}$ with $x_i - x_{i-1} = y_j - y_{j-1} = \Delta \quad \forall \quad i, j$ on Ω. Then we may approximate the evolution of the survival probability density at each node

on the lattice over time using the system of ODEs:

$$\begin{aligned}\frac{\partial S_{i,j}(t)}{\partial t} = \quad & \mu_x(x_i, y_j)\left(\frac{S_{i+1,j}(t) - S_{i-1,j}(t)}{2\Delta}\right) \\ & + \mu_y(x_i, y_j)\left(\frac{S_{i,j+1}(t) - S_{i,j-1}(t)}{2\Delta}\right) \\ & + \frac{\sigma_y^2(x_i, y_j)}{2}\left(\frac{S_{i+1,j}(t) - 2S_{i,j}(t) + S_{i-1,j}(t)}{\Delta^2}\right) \\ & + \frac{\sigma_y^2(x_i, y_j)}{2}\left(\frac{S_{i,j+1}(t) - 2S_{i,j}(t) + S_{i,j-1}(t)}{\Delta^2}\right)\end{aligned} \tag{5.1.29}$$

subject to:

$$S_{i,j}(u) = \begin{cases} 0 & \text{if } \|l_{ij} - \phi_k\| < \epsilon \quad \forall \quad u \in [s,t] \text{ and } k = 1, 2, \ldots, M, \\ 0 & \text{for } i, j \in \{0, N\} \quad \forall \quad u \in [s,t], \\ 1 & \text{for } u = s \text{ otherwise.} \end{cases} \tag{5.1.30}$$

Using this construction, we can approximate a solution to the survival probability surface $S^{\epsilon}_{\mathbf{X}_s \to \mathbf{\Phi}}(t)$ at each point on the lattice $\mathcal{L}$ by solving the system of ODEs in Equation 5.1.29 subject to the interior and boundary conditions in Equation 5.1.30. In order to achieve this, we can use standard numerical techniques such as a high order Runge-Kutta method with the modification that at each update of the numerical solution, the conditions in Equation 5.1.30 are enforced.

Note that here we have framed the problem on a finite region. In order to solve the trapping problem in its original form, we need to construct a similar solution for the case where there is no perimeter. The obvious solution to this problem is to 'stretch' the perimeter far enough that the diffusion process in all probability cannot reach it within the applicable time horizon. This can be achieved by simply modifying the limits of the lattice. However, this is a somewhat impractical solution as one would require an extremely large number of lattice nodes in order to calculate a valid approximation on such a large region. Since the computational time for the numerical solution increases quadratically in the number of lattice nodes, this proposition quickly becomes infeasible in practice. In order to get around this, we consider qualitative attributes of the unbounded trapping problem (i.e., when no perimeter is present): Although the problem is framed in an open region, most of the information on how the survival probability changes over time is concentrated near the detection points. Taking a cue from the behaviour of Equation 5.1.29 with respect to the boundary conditions, the nodes of the trapping array can be thought of as 'wells' into which the survival probability is being drained over time. Thus, if we were to view

a finite section of the survival probability surface which contains all detection points we would expect the survival probability to decrease over time at the edges of the field of view. Since the 'flow' of probability over time is completely dictated by the detection points, it stands to reason that we need not know how the probability changes outside our field of view in order to calculate the flow at the edge of our view. Thus, in order to calculate the evolution of the survival probability over time using a finite region, we need only find a way to describe the probability flow at the limits of the region we are considering. For example, assuming that the probability flow at the boundary is linear, we may modify the boundary conditions of the finite analogue to the trapping problem on a square region such that:

$$S_{i,j}(u) = \begin{cases} 0 & \text{if } \|l_{ij} - \phi_k\| < \epsilon \text{ for } k = 1, 2, \ldots, M \\ l_i^f(S_{i,j}(u)) & \text{for } i = N, j \notin \{0, N\} \\ l_i^b(S_{i,j}(u)) & \text{for } i = 0, j \notin \{0, N\} \\ l_j^f(S_{i,j}(u)) & \text{for } j = N, i \notin \{0, N\} \\ l_j^b(S_{i,j}(u)) & \text{for } j = 0, i \notin \{0, N\} \\ \frac{S_{i,j-1}(u)+S_{i-1,j}(u)}{2} & \text{for } i, j = N \\ \frac{S_{i,j+1}(u)+S_{i-1,j}(u)}{2} & \text{for } i = N, j = 0 \\ \frac{S_{i+1,j}(u)+S_{i,j-1}(u)}{2} & \text{for } i = 0, j = N \\ \frac{S_{i,j+1}(u)+S_{i+1,j}(u)}{2} & \text{for } i, j = 0, \end{cases} \tag{5.1.31}$$

for all $u \in [s, t]$ and $S_{i,j}(u) = 1$ for $u = s$ otherwise, where $l_k^b(.)$ and $l_k^f(.)$ extrapolate backwards and forwards (from the interior values) with respect to the index k respectively (noting of course that the corners of the region are calculated by averaging the flow in the x and y direction at neighbouring points). Here, the probability surface at the edges of the lattice are calculated by extrapolating values from the interior of the region on which the lattice is defined. For example, one extrapolation rule would be to simply set $l_i^f(S_{i,j}(u)) = S_{i-1,j}(u)$ and $l_i^b(S_{i,j}(u)) = S_{i+1,j}(u)$ i.e., the value of the survival probability surface at the edge of the lattice is extrapolated from the nearest interior point on the lattice. This rule thus assumes that the survival probability surface remains flat at the edge of the region on which the problem is being analysed. Alternatively, one may consider a linear extrapolation rule, whereby the survival probability surface is approximated by a linear equation at the edge of the region, for example, $l_i^f(S_{i,j}(u)) = S_{i-2,j}(u) + 2\Delta(S_{i-1,j}(u) - S_{i-2,j}(u))/\Delta = 2S_{i-1,j}(u) - S_{i-2,j}(u)$ and $l_i^b(S_{i,j}(u)) = 2S_{i+1,j}(u) - S_{i+2,j}(u)$. Using this scheme, we can approximate the evolution of the survival probability for a diffusion process moving in relation to a trapping array on a finite section of the region on which the problem is defined. For example, Figure 5.1.6 illustrates the time evolution of the survival

probability of a diffusion process with dynamics given by the SDE

$$\begin{aligned} dX_t &= -\alpha_x X_t dt + \sigma_x dB_t^{(1)} \\ dY_t &= -\alpha_y Y_t dt + \sigma_y dB_t^{(2)} \end{aligned} \tag{5.1.32}$$

for the parameter set $\{\alpha_x, \alpha_y, \sigma_x, \sigma_y\} = \{0.01, 0.01, \sqrt{0.2}, \sqrt{0.3}\}$ on a region Ω defined by the limits $\{-3.5, 3.5, -3.5, 3.5\}$ with $M = 5$ detection points $\mathbf{\Phi} = \{(2.55, 0.55), (-1.07, 0.07), (-0.07, 0.36), (0.50, -1.26), (-1.07, -1.45)\}$, all with a fixed detection radius of $\epsilon = 0.2$. Initially, the survival probability surface is almost flat exhibiting drops only close to the edges of the trapping radii. Over time the survival surface shapes conically around the detection points and deforms at the edges of the visible region compensating for the overall drop in probability. With respect to interpreting the results, it is important to note that each point on the survival probability surface corresponds to the probability of not being detected by the indicated time given that the process has started at that point.

As before, we can calculate the first passage time density function for a given initial value by calculating a numerical derivative of the survival probability surface:

$$g_{\mathbf{X}_s \to \mathbf{\Phi}}(t_k) \approx -\frac{S_{i^\star,j^\star}(t_{k+1}) - S_{i^\star,j^\star}(t_{k-1})}{t_{k+1} - t_{k-1}}, \tag{5.1.33}$$

where t_k denotes the time steps at which Equation 5.1.29 is evaluated numerically and $i^\star, j^\star$ indicates the position of the initial value on the lattice. Figure 5.1.7 compares the first passage time density for $\mathbf{X}_s = (1/6, 1/6)$ calculated from the survival probability surface using Equation 5.1.33 to a simulated first passage time density for the five-element trapping array.

Using this scheme we can derive a likelihood function based on an observational scheme which relies on multiple detection points. Here, instead of focusing on pair-wise traversals of the detection array, the sequence of detections can be used directly in conjunction with the joint first passage time density function. Using Figure 5.1.8 as a guide, the scheme follows: At the first time a detection is made, the time is recorded and the coordinate of the detection point is used as an initial value for a joint first passage time problem for the diffusion being detected at any other detection point in the array. That is, the initial detection point is removed from the array and the remaining points are used to formulate the first passage time density. At the subsequent detection, that detection point is removed and the previous detection point is replaced. Continuing in this way, we can construct a likelihood function based on the joint first passage time density. Thus, by calculating the first passage time density of the underlying diffusion model to any one of a multiple of detection points, we can improve the informational efficiency

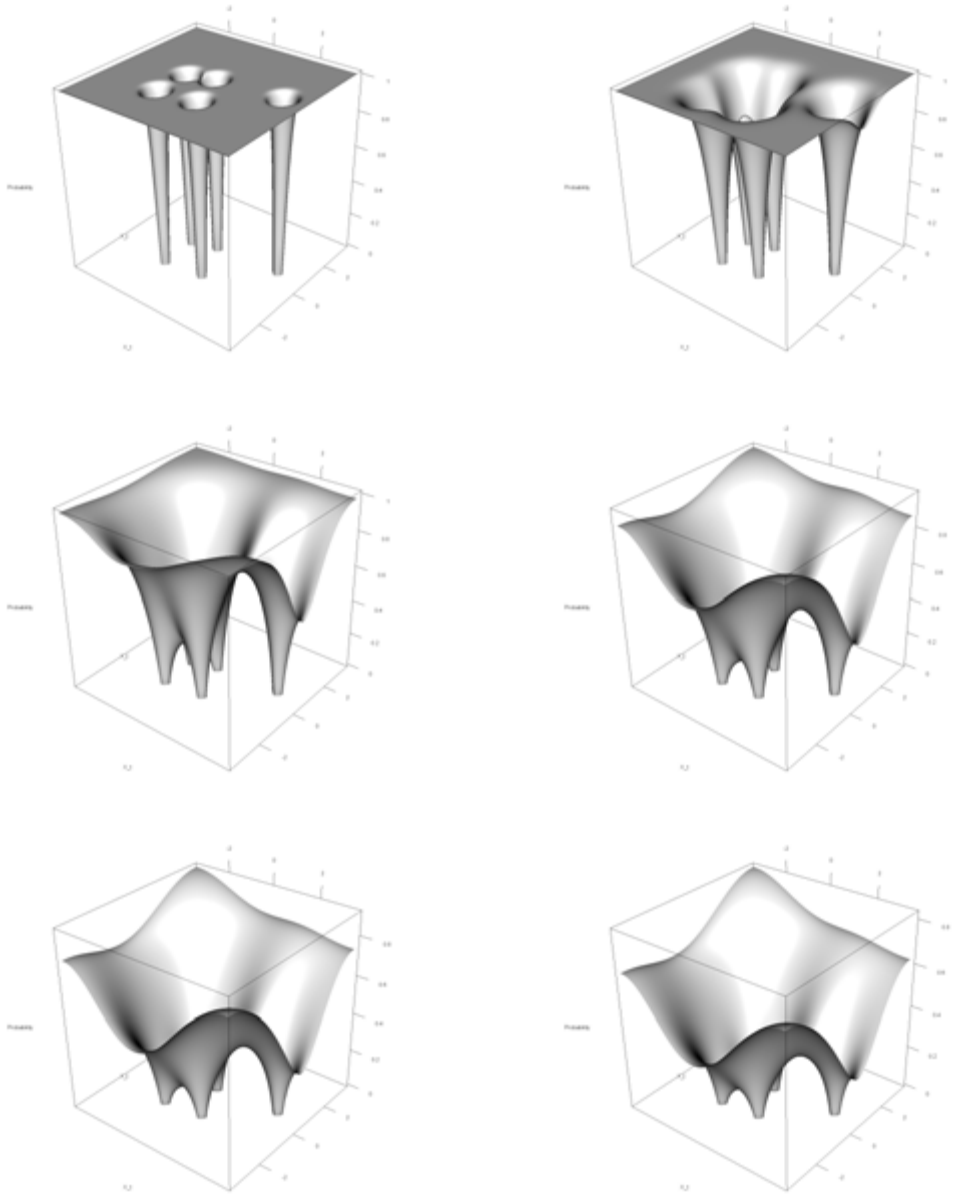


FIGURE 5.1.6: Time evolution of the survival probability density for Equation 5.1.32. Each point on the surface indicates the probability of survival given that the process has originated from the corresponding coordinate in the xy-plane. Surfaces are shown for times $t = 0.1, 1.0, 5.0, 10, 15$ and 20 respectively (from top left to right). R code: Supplementary materials, Section 5.6.

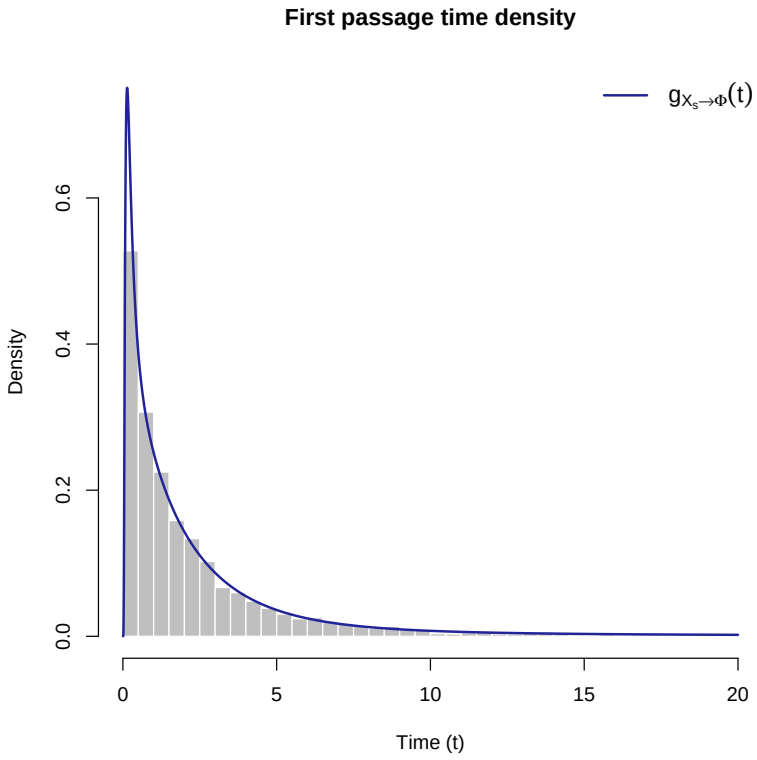


FIGURE 5.1.7: Simulated and approximate first passage time distribution for Equation 5.1.32 from a known initial value to a five-element detection array. R code: Supplementary materials, Section 5.6.

of the detection scheme. Here, the only informational loss that occurs is that of consecutive revisits to a given detection point.

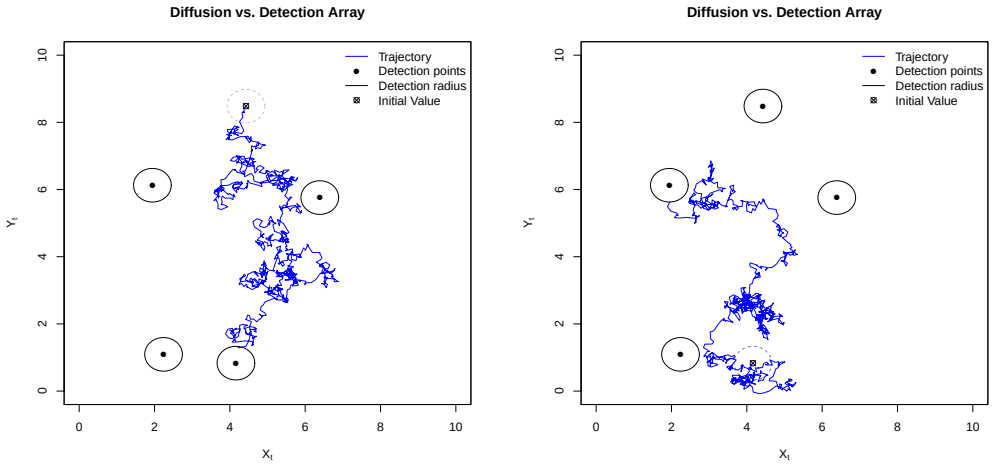


FIGURE 5.1.8: For consecutive trapping times observed from an array of detection points, we may formulate an approximate likelihood by treating consecutive coordinates at which detections are made as initial values of a first passage time problem. Thus, when a particle/animal is detected at a given trapping point that point is 'removed' from the detection array and the remaining traps are used to formulate the first passage time problem. The subsequent first passage time is then recorded as the first time at which any of the remaining detection points/cameras are triggered.

5.1.3.3 Summary

In order to model animal movements using a diffusion model in the context of camera-trap data, we formulate the problem as a non-standard first passage time problem. By making some simplifying assumptions with regard to the problem by focusing on a single detection point with a vanishing detection radius, it is possible to calculate the first passage time density in terms of the Volterra equation. Although this strategy has the particular advantage that it is computationally efficient, the scheme may not be suitable for constructing a likelihood function when the detection array has more than a few detection points or where the detection points lie close together. By defining the problem in terms of an interior value problem, it is possible to derive a framework which is more consistent with the problem. Using this strategy it is possible to calculate the first passage time density for a diffusion process moving in relation to multiple detection points. As such, it is possible to improve on the informational efficiency of a likelihood function that is based on the first passage time density by revising the observational scheme. However, this comes at the expense of significantly

more computational overhead. Consequently, in order to apply the methodology to a real-world dataset, it is likely that the methodology will have to be applied in a parallel computing environment. Thus, although we have demonstrated that we can solve first passage time problems for non-linear bivariate diffusions moving in relation to a detection array, a significant amount of work remains before the methodology can be applied to a real-world problem. Indeed, the sample properties of these problems need to be better understood and we need to establish how many first passage time observations are required in order to calculate satisfactory parameter estimates in this way.

5.2 Overview

In Chapter 1, we start by outlining a numerical technique for analysing the transitional density in terms of a large system of ordinary differential equations – the so-called method of lines. This strategy has various attractive properties: Firstly, the method allows for highly non-linear time-inhomogeneous model specifications and can readily be applied to multivariate problems. As such, we can calculate important quantities such as the transitional density for highly non-linear SDEs. Secondly, it is possible to apply the method to calculating the marginal transitional density of a multivariate diffusion. This is achieved by deriving a PDE for the marginal transition density which depends on a system of moment equations derived from the multivariate model. Unfortunately, this flexibility comes at the cost of computational efficiency. Particularly, as the dimensions of the model increase, the size of the approximating system of ODEs increases quadratically. Thus, applications that call for repeated evaluation of the transition density, such as calculating the likelihood, for example, become infeasible in practice. Fortunately, an alternative expression for the likelihood can be derived in the form of Girsanov's formula. Instead of focusing on the transitional density, this formula expresses the likelihood of a process in terms of stochastic integrals evaluated over the continuously observed trajectory of the process. Although Girsanov's formula provides a compact description of the likelihood, the formula can be difficult to apply to discretely observed trajectories since the stochastic integrals contained therein are more often than not intractable. This is particularly problematic when the data resolution is relatively low. In order to get around this, we make use of the data-imputation procedure originally developed by Roberts and Stramer (2001) and later extended to the multivariate case by Kalogeropoulos *et al.* (2011), The scheme relies on imputing missing parts of the discretely observed trajectory with Brownian bridges that start and end at consecutive observations. Thus by artificially enhancing the data resolution, we are able to calculate the likelihood using Girsanov's formula. By

way of a special factorization of the likelihood function, it is possible to perform inference on a discretely observed trajectory through an MCMC algorithm. Naturally, this strategy incurs some numerical overhead in that we are required to iteratively simulate numerous Brownian bridge trajectories. For these purposes, we develop a vectorized simulation scheme for generating strings of Brownian bridges. By embedding this scheme in the MCMC algorithm for the data-imputation procedure, we are able to improve the speed of the algorithm, which in turn makes it possible to analyse non-linear multivariate diffusion models with ease. Unfortunately, the scheme as treated here has the particular drawback that, for two distinct estimation runs under different model specifications, the likelihood calculations rely on different imputed paths. Since the likelihood calculation relies on the imputed paths in order to improve the resolution of the data such that we can calculate Girsanov's formula, it can be argued that comparing likelihood statistics such as the AIC calculated from separate runs is problematic. For these purposes, we combine the estimation output of the data-imputation scheme with the method of lines in order to calculate pseudo-AIC statistics of a given diffusion model.

Using the method of lines and the data-imputation scheme, we analyse the population-environment dynamics of *Emiliania huxleyi* – an abundant species of phytoplankton that can be found throughout the world's oceans – using a multivariate diffusion model. We start by fitting various time-inhomogeneous scalar diffusion models to water temperature observations that run concurrently with abundance observations for the species. Using the data-imputation scheme, we calculate parameter estimates for the various models and proceed to calculate pseudo-AIC values for each model using the method of lines. In order to validate the calculations, we calculate DIC values using the cumulant truncation procedure in conjunction with a saddlepoint approximation. Indeed, we find that the methods produce similar statistics throughout the model space. Subsequently, based on the pseudo-AIC statistics, we select a non-linear time-inhomogeneous diffusion model for the temperature time series. Following this, we develop various diffusion models of the population component of the time series. By including temperature terms in the drift of some of these models we are able to test various forms of temperature dependence for the population component. Based on pseudo-AIC values (which are again corroborated using the cumulant truncation procedure), we model the population-environment dynamics of the species using a non-linear multivariate diffusion model. Based on this model, we then proceed to develop a bloom potential function for the species. Based on the idea of conditional tail expectation, we calculate a risk measure for the occurrence of 'blooms' – periods in which the population undergoes rapid growth leading to extreme population numbers. Since the calculation relies on calculating the marginal transitional density, we are able to exploit a hybrid PDE which governs

the marginal transition density of the model in order to make repeated evaluation of the bloom potential function computationally feasible. Thus, using parameter samples drawn from the MCMC output, we are able to calculate the bloom potential function whilst accounting for uncertainty in the parameter estimates of the model.

Finally, we introduce the **DiffusionRimp** package: An R package which provides routines for employing methods such as the method of lines and the data-imputation procedure. We show how the package can be used to calculate transition densities for highly non-linear diffusion processes and illustrate how the package can be used to conduct inference by revisiting the *Emiliania huxleyi* dataset.

In chapter 2 we focus specifically on developing tools for the analysis of non-linear diffusion models in the software environment. Taking note of the various computational issues associated with methods such as the data-imputation procedure and the method of lines, we endeavour to develop a framework for analysing non-linear diffusion models which is both general enough to encompass a wide variety of models but also efficient enough to be implemented in a non-parallel computing environment. For these purposes we define the generalised quadratic diffusion class of diffusion models and develop the **DiffusionRgqd** package: An R package for performing inference and analysis on diffusion models of the generalised quadratic class. The premise behind this construction is that the class encompasses a wide range of non-linear diffusion models whilst being simple enough to analyse with a high degree of accuracy. Specifically, by applying the cumulant truncation procedure developed by Varughese (2013) to this class, we can derive explicit expressions for the cumulant equations of the generalised quadratic diffusions. Thus, within this 'sandbox' we can maximise the freedom with which one can specify a given model whilst still being able to analyse the model efficiently. By combining various forms of surrogate density with the cumulant equations, it is then possible to approximate the transitional density of such diffusions with great accuracy.

Another benefit of the generalised quadratic framework is that various computational redundancies arise that can be exploited in order to improve the computational efficiency of the cumulant truncation procedure. As such, depending on the model structure, it is possible to identify the most efficient mode of computation. This is mimicked in the software by stringing together predefined blocks of code in such a way that the most computationally efficient solution is constructed. In addition to structuring the code in this way, we make use of the **Rcpp** and **RcppArmadillo** packages in order to make further gains in computational efficiency by making use of the C++ language within R. This allows us

to construct efficient routines for calculating quantities such as the likelihood function.

For purposes of demonstrating the package and the underlying methodology, we analyse various diffusion models using routines from the package. We analyse the transition densities of interesting non-linear diffusion models, including a time-inhomogeneous Jacobi process and a stochastic counterpart to the non-linear Lotka-Volterra equations. Following the introductory examples, we then proceed to analyse various forms of stochastic volatility models of the S&P 500 Index and its corresponding volatility index. We start by fitting the so-called Heston model – a well-known stochastic volatility model often used in the analysis of financial time series – to the data using standard maximum likelihood techniques. This is achieved by making use of a C++ based routine from the package. Using this routine, it is possible to fit the model in a matter of seconds. Following this, we then show that by fitting competing stochastic volatility models to the time series, it is possible to improve on the fit of the Heston model for this dataset by formulating a competing stochastic volatility model for which the model has a diffusion tensor which is more sensitive to changes in the state of the process. This demonstrates the flexibility of the generalised quadratic framework in that it is possible to fit a wide range of models to a dataset and perform model selection in order to draw more accurate conclusions from the analysis. To demonstrate the scope of the methodology, we show how the package can be used to perform Bayesian inference at the hand of the random walk Metropolis-Hastings algorithm. Although by its nature, the Metropolis-Hastings algorithm is more computationally intensive than standard maximisation routines, the architecture of the package still makes it possible to analyse complex models with a reasonable degree of efficiency. Indeed, we show by way of a simulated dataset how the package may be used to analyse bivariate time-inhomogeneous non-linear models and compare model fit using DIC statistics.

Chapter 3 focuses on the analysis of first passage time problems for diffusion processes. Although the analysis of the probabilistic evolution of diffusion processes is challenging in its own right, the analysis of first passage time problems for such processes can often be significantly more difficult. Although there are various factors that contribute to this, the problem can perhaps be best summed up as follows: Typically, the analysis of diffusion models are carried out on short transition horizons. For example, in the context of inference we are usually concerned with finding short-horizon approximations for the transitional density in order to make the analysis tractable. In the case of first passage time problems we apply a stopping rule to the model in question and analyse the distribution of the time to given event. As such, the analysis of first passage time problems is usually framed on long time scales. Depending on the parameters of the process and the nature

of the stopping rule, the applicable transition horizon may be very long indeed. For purposes of the analysis, we start by considering a common stopping rule as applied to scalar diffusion processes. Specifically, we focus on the first passage time of a scalar diffusion to predefined threshold function. Fortunately, in this case, an explicit relationship between the distribution of the first passage time and the transitional density of the underlying process can be found by way of a simple integral equation in the form of the Volterra equation. Unfortunately, the Volterra equation – like the transitional density – is generally not tractable, even when the transitional density of the model being analysed is known. As such, we advocate calculating a numerical solution of the Volterra equation. Since the methodology outlined earlier in the thesis develops schemes for calculating the transitional density of non-linear diffusion models on arbitrarily large time scales, we embed the transitional density approximations in the numerical analysis of the Volterra equation in order to calculate accurate approximations of the first passage time density for non-linear models. This allows us to analyse first passage time problems that were previously only accessible via simulation.

Apart from the modelling freedom that this strategy affords, the Volterra equation lends itself well to the construction of computationally efficient routines for calculating the first passage time density numerically. Building on the architecture of the **DiffusionRgqd** package, it is possible to write **C++** routines for analysing first passage time problems for generalised quadratic diffusions under time-dependent threshold functions. As such, we extend the **DiffusionRgqd** library by including functions to conduct such an analysis. We proceed to illustrate the mechanics of these functions at the hand of a number of example first passage time problems. First, we compare our routines to that of an existing package for analysing first passage time problems for a diffusion model with an analytically tractable transition density. Then we show how the routine may be used to analyse a more complicated non-linear first passage time problem. Finally, as a demonstration of its use in the analysis of a real-world dataset, we fit a diffusion model to a time series of Amazon equity volatility. Specifically, we fit both a time-homogeneous and a time-inhomogeneous diffusion model to the Amazon equity volatility series and compare the first passage time density for the process to a threshold volatility level under each model. As demonstrated by the calculations, the inclusion of a time-inhomogeneous component to the model can drastically affect the distribution of the first passage time density and any calculations that follow from it. This demonstrates the importance of maximising the freedom of model and threshold specification in the analysis of first passage time problems – a goal which is reached to some extent by the combination of the cumulant truncation procedure with the Volterra equation.

Despite making it possible to analyse interesting first passage time problems,

the Volterra equation does have some limitations: Indeed, it is not difficult to conceive of more complicated stopping rules than we discuss here under the Volterra equation. Apart from a few other examples, the distribution of the first passage time variable which results from such a stopping rule cannot be evaluated at the hand of an expression such as the Volterra equation. Although the geometry of scalar first passage time problems is typically such that it is possible to derive an integral equation for the first passage time density, the methodology does not generalise well to higher dimensions and/or more complex geometries. Fortunately, whilst analysing time-homogeneous scalar first passage time problems, we found a strategy for analysing first passage time problems at the hand of a partial differential equation. Since the PDE in question is defined with reference to the geometry of the space in which the model is being applied, it scales well both with respect to the dimensions of the problem as well as with respect to the definition of the stopping rule. By making use of the method of lines, we are able to analyse the survival probability surface of a first passage time variable under time-homogeneous model specifications for both scalar and multivariate problems. Although the strategy is significantly more computationally intensive than the numerical techniques associated with the Volterra equation, we demonstrate the power of the methodology at the hand of a number of highly non-linear first passage time problems. As in the case of the **DiffusionRgqd** package and the Volterra equation, we are able to build on the architecture of the **DiffusionRimp** package, which already employs the method of lines, in order to analyse such first passage time problems. Using these routines we calculate the survival probability surface of bivariate first passage time problems for non-standard perimeter functions and demonstrate how the first passage time density can be extracted from such a calculation.

In Chapter 4 we introduce a more general class of models in the form of non-linear jump diffusion processes. The class is defined by adding a jump mechanism that acts on the trajectory a diffusion process, inducing randomly occurring 'jumps' in the state of the process. The process that results from such an amalgamation is that of a mixture between a pure diffusion process and a compound Poisson process. Naturally, combining two already complicated model structures leads to significant difficulties with respect to analysing the dynamics of such a process. Indeed, equations such as the Kolmogorov equation which governs the probabilistic evolution of the process is significantly more complex than for pure diffusion processes. This is especially true when the jump mechanism is allowed to depend on the state of the diffusion process. As a consequence, calculating quantities such as the transitional density of such a model is extremely difficult. Despite this, building on the analysis done earlier in this thesis, we are able to develop a methodology for analysing a very general class of jump diffusion models.

Specifically, we develop a means of analysing a non-linear multivariate jump diffusion models with state-dependent and/or stochastic intensity.

We start by deriving a partial difference differential equation for the moment generating function of a jump diffusion process. Thus, by way of a special type of partial differential equation, which relates how the moment generating function evolves over time, we implicitly derive expressions for how the moments of a jump diffusion process change over time. Using this, it is thus possible to exploit the sequence properties of the moment generating function in order to extract a system of equations which govern the evolution of the moments of a given model. We demonstrate how this can be achieved by applying the methodology to the generalised quadratic class subject to similar order restrictions on the state-dependence of the jump mechanism. Based on this framework, we are able to derive general expressions for the moment equations of jump-GQDs in terms of the parameters of the process, intensity process, and the moments of the corresponding jump distribution. Using these structures, we analyse a number of toy models in order to demonstrate the mechanics of the methodology and proceed to validate the calculation of the moments and transition density approximations by way of simulation.

During the development of the methodology, a particular nuance of the jump diffusion class was revealed with respect to the approximation of the transitional density. Although we are able to calculate the moments of a jump diffusion model with a high degree of accuracy and subsequently employ a surrogate density structure in order to approximate the transitional density, this strategy appears to be valid only on relatively long time scales. Although this seems perplexing, since one would expect it to be easier to approximate the transitional density on short time scales, the analysis reveals an important attribute of this class of models: Due to the dichotomous nature of the process, whereby the dynamics of the process is driven by processes with distinct *personalities*, the behaviour of the transitional density on short time scales is often times multi-modal and exhibit tail-behaviour which is difficult to replicate using standard density structures. Noting that the moment equations are accurate regardless of the time scale, the discrepancy between our original approximation strategy and the true density on short time-scales relates to how the moments of the process is carried into the density approximation. For these purposes, we derive a mixture factorization which contrasts the dynamics of the jump diffusion model to its pure diffusion counterpart. As such, we are able to accurately replicate the transitional density on short time scales by way of a two-step approximation based on the transition density of the pure diffusion counterpart of the process and an excess distribution.

In order to assess the performance of the scheme, we extensively compare our methodology to various exact and approximate results in the literature. We

find that the scheme performs favourably for various combinations of diffusion model and jump mechanism. Indeed, the scheme applies to much more general models than we can find appropriate benchmarks for. Following this, we apply the mixture factorization to the calculation of the likelihood function and benchmark maximum likelihood estimates calculated under the approximation to those calculated under a near-exact likelihood function for a simple jump diffusion process. We repeat the experiment for a more complex model for which we can analyse individual components in order to gain insight into the behaviour of the parameter estimates when the jump mechanism is not observed directly. The results suggest that the approximate likelihood function produces accurate parameter estimates bar the intrinsic bias associated with estimating the jump mechanism in the absence of any direct data on the jump mechanism trajectory. Following this, we show how the mixture factorization scheme can be used within an MCMC routine to estimate the probability that a given transition horizon contains a jump. As such, the method can be used to perform a local decoding of the jump mechanism in order to detect jumps in a given dataset.

Using the GQD-framework in conjunction with the mixture factorization, we fit various jump diffusion models to a real-world dataset in the form of an equity volatility index for Google shares. The volatility dynamics of stock price processes play an important role in the pricing of financial derivatives and the trade decisions of investors. Using these models, we demonstrate that traditional models of volatility such as the CIR model can be improved upon significantly by incorporating a jump mechanism and time-inhomogeneous coefficients in the model. Indeed, the methodology developed here affords the opportunity to select the best fitting model among a plethora of equations without much difficulty. Building on the analysis, we estimate jump detection probabilities for the time series and compare the results to the dates of quarterly earnings reports for the company. Interestingly, what the analysis reveals is that, over and above thy quarterly volatility cycle, jump events can be associated with the release of quarterly earnings reports. Although we conclude the analysis there, this poses some interesting research questions in this regard. For example, it would be interesting to compare a model with randomly occurring jumps to one where jumps occur in a structure pattern, as in the case of quarterly earnings reports.

One advantage of developing the methodology in such a general framework is that it allows us to analyse interesting new models. We demonstrate this at the hand of two novel models that build on existing ideas in the field of diffusion models. First, we analyse a special kind of jump mechanism specification whereby we can formulate a jump-reverting diffusion model. The premise is that we can formulate a mean-reverting diffusion model for which the reversion mechanism operates in distinct steps/jumps as opposed to traditional models which do so

continuously. We show that is possible to establish a form of parity between jump-reverting models and standard mean-reverting models by choosing a specific jump distribution and intensity parametrisation. As another example, we consider a generalisation of the diffusion approximation to the Wright-Fisher process whereby the gene frequency trajectory undergoes randomly occurring shocks/jumps. Using the discrete analogue of the process – i.e., simulating the actual Wright-Fisher process with shocks – as a benchmark, we demonstrate that the jump diffusion approximation is indeed valid.

Finally, we develop the **DiffusionRjgqd** package for analysing scalar and bivariate generalised quadratic jump diffusion models. We show how the mixture factorization is applied in the software environment and detail key points of the algorithm for constructing accurate transition density and likelihood approximations. We analyse a number of interesting non-linear models using routines from the package and demonstrate how the package can be used to analyse the transitional density of a Hawkes process – a class of continuous time processes which happen to be nested within the jump diffusion class. For purposes of demonstrating the application of the package to a real-world dataset, we revisit the Google equity volatility dataset and show how to calculate parameter estimates and jump detection probabilities under a jump diffusion model of the time series.

5.3 Conclusions

Using diffusion models it is possible to formulate compact and realistic models of real-world phenomena in terms of systems of stochastic differential equations. Unfortunately, such systems are for the most part intractable, and deriving statistical properties of such models is extremely difficult. This is especially true for diffusion models which contain non-linear elements. However, the value proposition of such models is that it is possible to analyse stochastic processes that exhibit non-linear dynamics using a small set of equations. In order to analyse such processes, we focus on key elements pertaining to the analysis of diffusion models such as the transitional density and/or likelihood function. In order to evaluate the transitional density, we apply various numerical techniques for solving partial differential equations to the Kolmogorov forward equation. Using the method of lines, it is possible to approximate the transition densities of non-linear time-inhomogeneous diffusion models with a high degree of accuracy over arbitrarily large transition horizons. Indeed, the method has the particular advantage that it allows a significant amount of freedom with respect to the model specification, making it possible to calculate transitional densities for models with extremely complicated drift and diffusion specifications. Despite the

generality of the method, it is not efficient enough for applications that call for the iterative evaluation of the transition density. Thus, although the method can, in theory, be used to calculate the likelihood function of a diffusion by way of its transitional density, the computational overhead associated with maximising the likelihood in such an application is such that the strategy is unfit for application in non-parallel computing environments. As such, we resort to an alternative strategies for calculating the likelihood. One such strategy is the data-imputation procedure. Although the data-imputation procedure as treated here relies on the reducibility of a given diffusion model, the strategy applies to a wide range of non-linear models and allows a great deal of freedom with respect to the specification of the drift structure. Indeed, provided that the time series being modelled is not too sparse, the algorithm can handle highly non-linear drift structures with ease. Where the imputation of Brownian bridges fail to replicate the dynamics of the model process (i.e., very low acceptance rates are observed at the imputation step), it is possible to construct more realistic bridge processes in order to improve the imputation, albeit at the cost of complicating the algorithm. Perhaps the biggest limitation of the data imputation scheme is the constraint of reducibility. This has important implications for the specification of the diffusion matrix and as a consequence useful classes of models may be excluded from the range of models to which the scheme can be applied. For example, stochastic volatility models such as the Heston model are not reducible. Despite this, it is still possible to modify the scheme in order to handle such models, once again at the expense of complicating the analysis. Overall the scheme thus presents a flexible strategy with a number of desirable features. By combining a vectorized scheme for simulating strings of Brownian bridges with the algorithm, it is possible to efficiently estimate the parameters of a non-linear diffusion model. Furthermore, provided that the data is not too sparse, it scales well to dimensions greater than two, requiring little modification of the constituents involved.

Despite the various attractive properties of the data imputation scheme, it is not without its limitations. Indeed, the scheme is simulation intensive and despite being computationally efficient compared to more brute force methods, it may suffer from significant computational overhead depending on the data at hand. In order to improve on the computational efficiency of the aforementioned methods in absolute terms and to provide perhaps a more self-contained strategy for performing inference and model selection, we focus on another strategy for calculating the transitional density: the cumulant truncation procedure. Using the cumulant truncation procedure, it is possible to analyse non-linear models of polynomial form at the hand of a moment closure approximation. By placing second-order restrictions on the coefficients of a polynomial diffusion process we derive the generalised quadratic class of diffusion models. Within this class, it is possible to derive accurate approximations to the transition density

for non-linear processes with highly non-linear time dependences over large transition horizons. Indeed, the class spans a wide range of non-linear models and provides a simple framework for the analysis of diffusion models in the software environment. In particular, the methodology allows one to significantly improve on the computational efficiency of the analysis compared to the method of lines and the data-imputation procedure. Based on the approximation scheme, it is possible to estimate model parameters in maximum likelihood and Bayesian frameworks and directly calculate useful quantities such as AIC, BIC and DIC statistics for a plethora of models, thus making it possible to more accurately model real-world datasets than is typically achieved under existing models with analytically tractable dynamics. It remains to be seen how well this strategy scales to higher dimensional models. Indeed, the dimension of the system of cumulant equations may increase quite quickly for high dimensional systems with many interactions, although the mechanics of evaluating such a system remains the same. However, the primary hurdle in this case would likely be the computation of a high-dimensional surrogate density. Perhaps more interestingly, it would be useful in future work to explore higher dimensional models under the premise that certain components of the model are not observed. For example, in the case of a stochastic volatility model where the volatility component is not observed one may readily adapt the methodology by calculating the joint moment equations and inferring the process dynamics from the marginal density (as opposed to the joint distribution under an observed proxy for the volatility process) as approximated using the corresponding moments of the (observed) dimension of interest. Indeed, the existing software architecture can readily be adapted to perform such an analysis. For these purposes, a rigorous analysis of the quality of inference that can be made under such a scheme is required to clarify the interpretation of results from such an analysis.

As with all stochastic processes, the analysis of non-linear diffusion processes is not restricted to evaluating the dynamics of the process itself, but also extends to the analysis of such processes under stopping rules. For scalar diffusion processes, it is possible to derive an integral equation in the form of the Volterra equation that governs the first passage time density of a diffusion process to a time-varying threshold function. Particularly, the Volterra equation relates the first passage time density to the transitional density of the diffusion process. By combining the cumulant truncation procedure under the generalised quadratic framework with a numerical procedure for evaluating the integral equation, we are able to efficiently calculate solutions to first passage time problems for non-linear, time-inhomogeneous diffusion processes under time-varying threshold regimes. Unfortunately, it is not always possible to derive an appropriate integral equation for the first passage time density. This follows since the structure of the equation is determined in some sense by the specification of the stopping rule. As such,

for more general stopping rules, or multivariate first passage time problems, the first passage time density cannot always be calculated in this way. For time-homogeneous first passage time problems, however, it is possible to derive a PDE which is invariant to the stopping rule in the sense that the stopping rule only effects the boundary conditions of the equation. Thus, applying the method of lines to this PDE, it is possible to analyse more general first passage time problems albeit under the restriction of time-homogeneity. Using these strategies it is possible to analyse a wide range of highly non-linear first passage time problems thus widening the scope for the application of diffusion models to real-world problems.

In addition to the analysis of pure diffusion processes, we consider various generalisations of diffusion models. Specifically, we develop methods for analysing an important class of mixture processes termed jump diffusion models. Jump diffusion processes are a mixture of diffusion processes and compound Poisson processes that extend the range of pure diffusion models by accounting for the possibility of randomly occurring jump events in the trajectory of the process. For these purposes, we develop a moment truncation scheme for the analysis of multivariate non-linear jump diffusion models. The scheme is particularly powerful in that it allows us to analyse models with state-dependent and/or stochastic jump intensity – a class of models for which very few strategies for analysing such processes exist – in a computationally efficient manner. Indeed, the scope of the methodology is such that it is possible to significantly improve on pure diffusion based models of real-world time series, decode jump events from a time-series, and even formulate interesting new theoretical models. Also, the strategy by which the transitional density is approximated at the hand of a mixture factorization shows promise for the application to further generalisations of diffusion processes such as non-linear Markov-switching diffusion models.

Appendix A

Data-imputation and the method of lines

A.1 Finite difference approximations

A key component setting up the method of lines approximation is the introduction of finite difference schemes to substitute the spatial derivatives in a given PDE. In the context of diffusion processes one may employ simple finite difference schemes in order to derive the desired ODE approximation. Consider for example the bivariate case, and for notational simplicity let $i_1 \equiv i$ and $i_2 \equiv j$. Furthermore, assume for simplicity that the lattice is constructed using equispaced nodes, say $x^{(1)}_{i+1} - x^{(1)}_i = x^{(2)}_{j+1} - x^{(2)}_j = \tau$ for all $i, j = 1, 2, \ldots$. If one applies a standard first order finite difference approximation to the first component of the first summation in Equation 1.2.1, the contribution to the system of ODEs is given by:

$$\Delta^1_1[\mu_1(\mathcal{L}, t) \cdot \mathbf{f}(t)]_{i,j} = \frac{1}{2\tau}\Big[\mu_1(x^{(1)}_{i+1}, x^{(2)}_j, t) f_{i+1,j}(t) - \mu_1(x^{(1)}_{i-1}, x^{(2)}_j, t) f_{i-1,j}(t)\Big]. \tag{A.1.1}$$

Similarly, a first order finite difference approximation of the second order derivative applied to the first term of the second summation in Equation 1.2.1 yields:

$$\Delta^1_{1,1}[\gamma_{1,1}(\mathcal{L}) \cdot \mathbf{f}(t)]_{i,j} = \\ \frac{1}{\tau^2}\Big[\gamma_{1,1}(x^{(1)}_{i+1}, x^{(2)}_j, t) f_{i+1,j}(t) - 2\gamma_{1,1}(x^{(1)}_i, x^{(2)}_j, t) f_{i,j}(t) + \gamma_{1,1}(x^{(1)}_{i-1}, x^{(2)}_j, t) f_{i-1,j}(t)\Big]. \tag{A.1.2}$$

Depending on the type of finite difference approximation used, a pattern or 'stencil' will form dictating how each $f_{i,j}(t)$ is related to surrounding $f_{(.,.)}(t)$ and $\{x^{(1)}_{(.)}, x^{(2)}_{(.)}\}$. Furthermore, depending on how the lattice is constructed, the

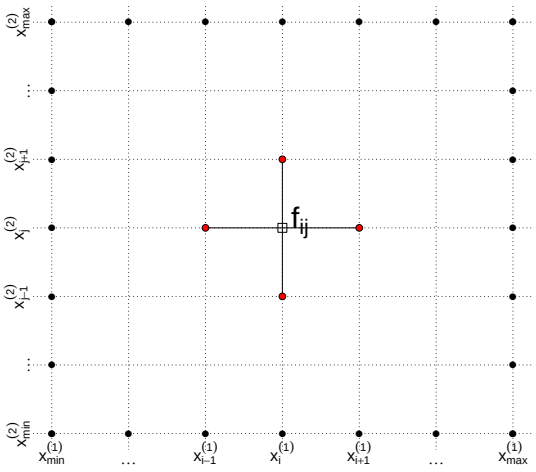


FIGURE A.1.1: A graphical representation of a stencil applied to a lattice as per the method of lines.

resulting expressions may then simplify. Figure A.1.1 illustrates the principle: $f_{i,j}(t)$ (indicated by a square) is evaluated at coordinates on the lattice given by vertices on the grid. The evolution over time of $f_{i,j}(t)$ is governed by a system of ODEs that result from a chosen local approximation to the spatial derivatives in Equation 1.1.5. For the scheme in equations A.1.1 and A.1.2 the resulting stencil is indicated by the solid lines and red dots.

Once the lattice is constructed and the system of ODEs have been derived we may proceed to solve the system numerically. In order for the system of ODEs to have a unique solution, a boundary condition must be specified. One may emulate the Dirac delta initial conditions as follows: Let $\{x^{(1)}_{i_0}, x^{(2)}_{j_0}\}$ be a vertex on $\mathcal{L}$ corresponding to the initial values/coordinates of the process. Then all $f_{i,j}(s)$ apart from $f_{i_0,j_0}(s)$ are set to 0 at time s and $f_{i_0,j_0}(s)$ is set to $1/\tau^2$. Assuming that the limits of $\mathcal{L}$ are chosen to be large far enough away from the initial value of the process, the boundaries of $\mathbf{f}(t)$ are set to 0 for all $t > s$. This is done so as to mimic the tails of the distribution where the density is close to 0 (indicated by the black dots in Figure A.1.1). The resulting system of ODEs can then be solved using standard numerical techniques.

A.2 Stability and discretization

A well-known issue with discretization based numerical methods for solving PDEs is that of stability. Although the method of lines is quite a robust algorithm, it is not exempt from stability issues. Fortunately, the way in which instabilities arise is such that it is easy to diagnose and 'tune out' by adjusting the parameters of the algorithm. For purposes of demonstrating the issue, consider a diffusion model with dynamics given by the SDE:

$$dX_t = -X_t^3 dt + dB_t. \tag{A.2.1}$$

Suppose we know that most of the mass of the density will lie in the interval $[-5, 5]$. Subsequently, we may then discretize the domain and apply the method of lines in order to derive a system of ODEs that approximate the transitional density. Once the system is derived we need only solve it numerically forward in time in order to approximate the evolution of the transitional density. Solving this system numerically involves a discrete (fractional) time step. As it happens, when the ratio of the time step to the spatial discretization is too large the system becomes unstable. Figure A.2.1 illustrates the density approximation under the method of lines calculated using a 51-dimensional system of ODEs calculated using different step sizes. For a step size of 0.005, the approximation remains stable over the entire transition horizon. Altering the step size, making it slightly larger at 0.006837607 the system becomes unstable at the end of the transition horizon. Extending the transition horizon, these instabilities will eventually propagate and the approximation will become unstable over the entire domain. Depending on the specification of the diffusion and the parameters of the problem, analysing the transition density using the method of lines requires some manual tuning. However, the instability issues are usually easily diagnosed in that the oscillations result in negative values for the density approximation. Subsequently, this can be dealt with by first decreasing the step size of the Runge-Kutta method and if needed modifying the range and resolution of the spatial discretization.

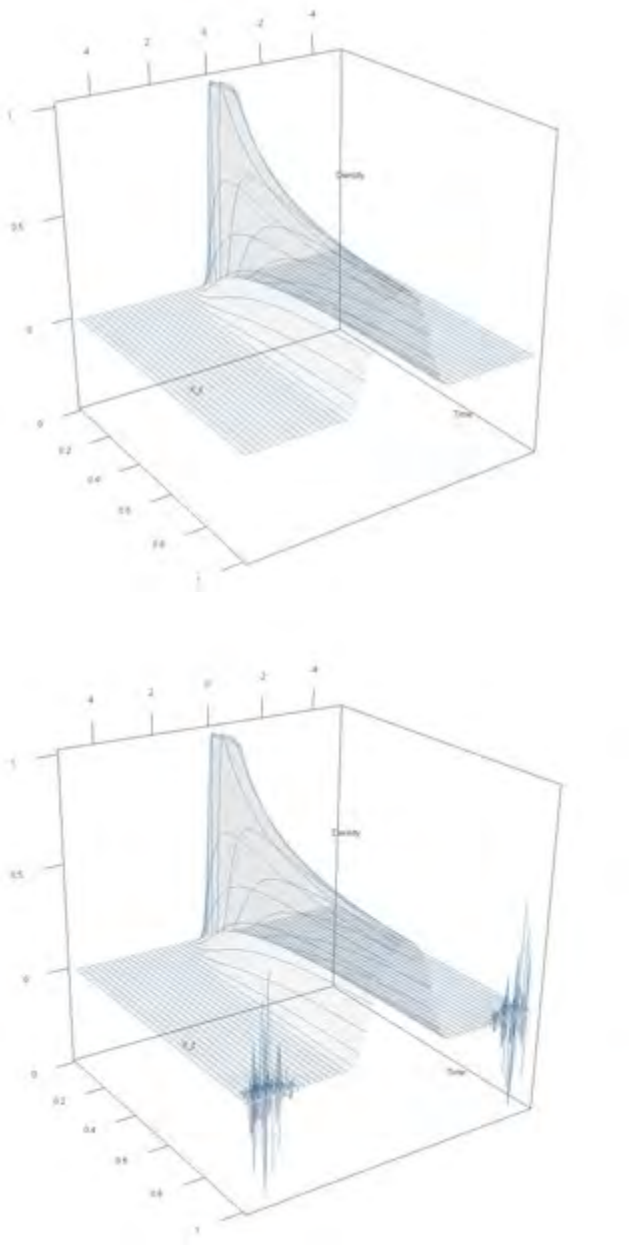


FIGURE A.2.1: Transition density of Equation A.2.1 over time. ODEs used in the approximation (blue lines) are calculated numerically using a fixed step size. For a step size of 0.005 time units, the approximation remains stable (top). For a slightly larger step size (0.006837607), numerical instabilities manifest as alternating spikes in the approximate transition density surface (bottom). R code: Supplementary materials, Section 1.12.

Appendix B

Generalised quadratic diffusions in the software environment

B.1 Cumulant equations for bivariate GQDs

By expanding Equation 2.3.21, one may derive a system of ODEs that approximates the evolution of the cumulants of a bivariate GQD over time. Tables B.1.1 to B.1.6 give terms to include on the RHS of the cumulant system for each coefficient of Equation 2.2.2. That is, for each dimension on the left of 2.3.21 (first column), include the terms in the column of every non-zero coefficient. Note that we have dropped the explicit dependence of κ_{ij} on t (i.e $\kappa_{ij}(t) = \kappa_{ij}$) for compactness. Also, we use the notation $\dot{\kappa}_{ij} = \frac{\partial}{\partial t}\kappa_{ij}$.

	a_{00}	a_{10}	a_{20}	a_{01}	a_{02}	a_{11}
$\dot{\kappa}_{10}$	1	$+1\kappa_{10}$	$+1\kappa_{10}\kappa_{10} + 1\kappa_{20}$	$+1\kappa_{01}$	$+1\kappa_{01}\kappa_{01} + 1\kappa_{02}$	$+1\kappa_{11} + 1\kappa_{10}\kappa_{01}$
$\dot{\kappa}_{20}$	.	$+2\kappa_{20}$	$+2\kappa_{10}\kappa_{20} + 2\kappa_{20}\kappa_{10} + 2\kappa_{30}$	$+2\kappa_{11}$	$+2\kappa_{01}\kappa_{11} + 2\kappa_{11}\kappa_{01} + 2\kappa_{12}$	$+2\kappa_{21} + 2\kappa_{10}\kappa_{11} + 2\kappa_{20}\kappa_{01}$
$\dot{\kappa}_{30}$	.	$+3\kappa_{30}$	$+3\kappa_{10}\kappa_{30} + 6\kappa_{20}\kappa_{20} + 3\kappa_{30}\kappa_{10} + 3\kappa_{40}$	$+3\kappa_{21}$	$+3\kappa_{01}\kappa_{21} + 6\kappa_{11}\kappa_{11} + 3\kappa_{21}\kappa_{01} + 3\kappa_{22}$	$+3\kappa_{31} + 3\kappa_{10}\kappa_{21} + 6\kappa_{20}\kappa_{11} + 3\kappa_{30}\kappa_{01}$
$\dot{\kappa}_{40}$	.	$+4\kappa_{40}$	$+4\kappa_{10}\kappa_{40} + 12\kappa_{20}\kappa_{30} + 12\kappa_{30}\kappa_{20} + 4\kappa_{40}\kappa_{10}$	$+4\kappa_{31}$	$+4\kappa_{01}\kappa_{31} + 12\kappa_{11}\kappa_{21} + 12\kappa_{21}\kappa_{11} + 4\kappa_{31}\kappa_{01}$	$+4\kappa_{10}\kappa_{31} + 12\kappa_{20}\kappa_{21} + 12\kappa_{30}\kappa_{11} + 4\kappa_{40}\kappa_{01}$
$\dot{\kappa}_{01}$	.	.	.	.	.	.
$\dot{\kappa}_{02}$	.	.	.	.	.	.
$\dot{\kappa}_{03}$	.	.	.	.	.	.
$\dot{\kappa}_{04}$	.	.	.	.	.	.
$\dot{\kappa}_{11}$	.	$+1\kappa_{11}$	$+1\kappa_{10}\kappa_{11} + 1\kappa_{11}\kappa_{10} + 1\kappa_{21}$	$+1\kappa_{02}$	$+1\kappa_{01}\kappa_{02} + 1\kappa_{02}\kappa_{01} + 1\kappa_{03}$	$+1\kappa_{12} + 1\kappa_{10}\kappa_{02} + 1\kappa_{11}\kappa_{01}$
$\dot{\kappa}_{12}$	.	$+1\kappa_{12}$	$+1\kappa_{10}\kappa_{12} + 2\kappa_{11}\kappa_{11} + 1\kappa_{12}\kappa_{10} + 1\kappa_{22}$	$+1\kappa_{03}$	$+1\kappa_{01}\kappa_{03} + 2\kappa_{02}\kappa_{02} + 1\kappa_{03}\kappa_{01} + 1\kappa_{04}$	$+1\kappa_{13} + 1\kappa_{10}\kappa_{03} + 2\kappa_{11}\kappa_{02} + 1\kappa_{12}\kappa_{01}$
$\dot{\kappa}_{21}$	.	$+2\kappa_{21}$	$+2\kappa_{10}\kappa_{21} + 2\kappa_{11}\kappa_{20} + 2\kappa_{20}\kappa_{11} + 2\kappa_{21}\kappa_{10} + 2\kappa_{31}$	$+2\kappa_{12}$	$+2\kappa_{01}\kappa_{12} + 2\kappa_{02}\kappa_{11} + 2\kappa_{11}\kappa_{02} + 2\kappa_{12}\kappa_{01} + 2\kappa_{13}$	$+2\kappa_{22} + 2\kappa_{10}\kappa_{12} + 2\kappa_{11}\kappa_{11} + 2\kappa_{20}\kappa_{02} + 2\kappa_{21}\kappa_{01}$
$\dot{\kappa}_{22}$	.	$+2\kappa_{22}$	$+2\kappa_{10}\kappa_{22} + 4\kappa_{11}\kappa_{21} + 2\kappa_{12}\kappa_{20} + 2\kappa_{20}\kappa_{12} + 4\kappa_{21}\kappa_{11} + 2\kappa_{22}\kappa_{10}$	$+2\kappa_{13}$	$+2\kappa_{01}\kappa_{13} + 4\kappa_{02}\kappa_{12} + 2\kappa_{03}\kappa_{11} + 2\kappa_{11}\kappa_{03} + 4\kappa_{12}\kappa_{02} + 2\kappa_{13}\kappa_{01}$	$+2\kappa_{10}\kappa_{13} + 4\kappa_{11}\kappa_{12} + 2\kappa_{12}\kappa_{11} + 2\kappa_{20}\kappa_{03} + 4\kappa_{21}\kappa_{02} + 2\kappa_{22}\kappa_{01}$
$\dot{\kappa}_{13}$	.	$+1\kappa_{13}$	$+1\kappa_{10}\kappa_{13} + 3\kappa_{11}\kappa_{12} + 3\kappa_{12}\kappa_{11} + 1\kappa_{13}\kappa_{10}$	$+1\kappa_{04}$	$+1\kappa_{01}\kappa_{04} + 3\kappa_{02}\kappa_{03} + 3\kappa_{03}\kappa_{02} + 1\kappa_{04}\kappa_{01}$	$+1\kappa_{10}\kappa_{04} + 3\kappa_{11}\kappa_{03} + 3\kappa_{12}\kappa_{02} + 1\kappa_{13}\kappa_{01}$
$\dot{\kappa}_{31}$	.	$+3\kappa_{31}$	$+3\kappa_{10}\kappa_{31} + 3\kappa_{11}\kappa_{30} + 6\kappa_{20}\kappa_{21} + 6\kappa_{21}\kappa_{20} + 3\kappa_{30}\kappa_{11} + 3\kappa_{31}\kappa_{10}$	$+3\kappa_{22}$	$+3\kappa_{01}\kappa_{22} + 3\kappa_{02}\kappa_{21} + 6\kappa_{11}\kappa_{12} + 6\kappa_{12}\kappa_{11} + 3\kappa_{21}\kappa_{02} + 3\kappa_{22}\kappa_{01}$	$+3\kappa_{10}\kappa_{22} + 3\kappa_{11}\kappa_{21} + 6\kappa_{20}\kappa_{12} + 6\kappa_{21}\kappa_{11} + 3\kappa_{30}\kappa_{02} + 3\kappa_{31}\kappa_{01}$

TABLE B.1.1: Cumulant equation terms for coefficients a_{00} to a_{11} of Equation 2.2.2.

	b_{00}	b_{10}	b_{20}	b_{01}	b_{02}	b_{11}
$\dot{\kappa}_{10}$	.	.	.	.	.	.
$\dot{\kappa}_{20}$	.	.	.	.	.	.
$\dot{\kappa}_{30}$	.	.	.	.	.	.
$\dot{\kappa}_{40}$	.	.	.	.	.	.
$\dot{\kappa}_{01}$	1	$+1\kappa_{10}$	$+1\kappa_{10}\kappa_{10} + 1\kappa_{20}$	$+1\kappa_{01}$	$+1\kappa_{01}\kappa_{01} + 1\kappa_{02}$	$+1\kappa_{11} + 1\kappa_{01}\kappa_{10}$
$\dot{\kappa}_{02}$	.	$+2\kappa_{11}$	$+2\kappa_{10}\kappa_{11} + 2\kappa_{11}\kappa_{10} + 2\kappa_{21}$	$+2\kappa_{02}$	$+2\kappa_{01}\kappa_{02} + 2\kappa_{02}\kappa_{01} + 2\kappa_{03}$	$+2\kappa_{12} + 2\kappa_{01}\kappa_{11} + 2\kappa_{02}\kappa_{10}$
$\dot{\kappa}_{03}$	.	$+3\kappa_{12}$	$+3\kappa_{10}\kappa_{12} + 6\kappa_{11}\kappa_{11} + 3\kappa_{12}\kappa_{10} + 3\kappa_{22}$	$+3\kappa_{03}$	$+3\kappa_{01}\kappa_{03} + 6\kappa_{02}\kappa_{02} + 3\kappa_{03}\kappa_{01} + 3\kappa_{04}$	$+3\kappa_{13} + 3\kappa_{01}\kappa_{12} + 6\kappa_{02}\kappa_{11} + 3\kappa_{03}\kappa_{10}$
$\dot{\kappa}_{04}$	.	$+4\kappa_{13}$	$+4\kappa_{10}\kappa_{13} + 12\kappa_{11}\kappa_{12} + 12\kappa_{12}\kappa_{11} + 4\kappa_{13}\kappa_{10}$	$+4\kappa_{04}$	$+4\kappa_{01}\kappa_{04} + 12\kappa_{02}\kappa_{03} + 12\kappa_{03}\kappa_{02} + 4\kappa_{04}\kappa_{01}$	$+4\kappa_{01}\kappa_{13} + 12\kappa_{02}\kappa_{12} + 12\kappa_{03}\kappa_{11} + 4\kappa_{04}\kappa_{10}$
$\dot{\kappa}_{11}$	.	$+1\kappa_{20}$	$+1\kappa_{10}\kappa_{20} + 1\kappa_{20}\kappa_{10} + 1\kappa_{30}$	$+1\kappa_{11}$	$+1\kappa_{01}\kappa_{11} + 1\kappa_{11}\kappa_{01} + 1\kappa_{12}$	$+1\kappa_{21} + 1\kappa_{01}\kappa_{20} + 1\kappa_{11}\kappa_{10}$
$\dot{\kappa}_{12}$	.	$+2\kappa_{21}$	$+2\kappa_{10}\kappa_{21} + 2\kappa_{11}\kappa_{20} + 2\kappa_{20}\kappa_{11} + 2\kappa_{21}\kappa_{10} + 2\kappa_{31}$	$+2\kappa_{12}$	$+2\kappa_{01}\kappa_{12} + 2\kappa_{02}\kappa_{11} + 2\kappa_{11}\kappa_{02} + 2\kappa_{12}\kappa_{01} + 2\kappa_{13}$	$+2\kappa_{22} + 2\kappa_{01}\kappa_{21} + 2\kappa_{02}\kappa_{20} + 2\kappa_{11}\kappa_{11} + 2\kappa_{12}\kappa_{10}$
$\dot{\kappa}_{21}$	.	$+1\kappa_{30}$	$+1\kappa_{10}\kappa_{30} + 2\kappa_{20}\kappa_{20} + 1\kappa_{30}\kappa_{10} + 1\kappa_{40}$	$+1\kappa_{21}$	$+1\kappa_{01}\kappa_{21} + 2\kappa_{11}\kappa_{11} + 1\kappa_{21}\kappa_{01} + 1\kappa_{22}$	$+1\kappa_{31} + 1\kappa_{01}\kappa_{30} + 2\kappa_{11}\kappa_{20} + 1\kappa_{21}\kappa_{10}$
$\dot{\kappa}_{22}$	.	$+2\kappa_{31}$	$+2\kappa_{10}\kappa_{31} + 2\kappa_{11}\kappa_{30} + 4\kappa_{20}\kappa_{21} + 4\kappa_{21}\kappa_{20} + 2\kappa_{30}\kappa_{11} + 2\kappa_{31}\kappa_{10}$	$+2\kappa_{22}$	$+2\kappa_{01}\kappa_{22} + 2\kappa_{02}\kappa_{21} + 4\kappa_{11}\kappa_{12} + 4\kappa_{12}\kappa_{11} + 2\kappa_{21}\kappa_{02} + 2\kappa_{22}\kappa_{01}$	$+2\kappa_{01}\kappa_{31} + 2\kappa_{02}\kappa_{30} + 4\kappa_{11}\kappa_{21} + 4\kappa_{12}\kappa_{20} + 2\kappa_{21}\kappa_{11} + 2\kappa_{22}\kappa_{10}$
$\dot{\kappa}_{13}$	.	$+3\kappa_{22}$	$+3\kappa_{10}\kappa_{22} + 6\kappa_{11}\kappa_{21} + 3\kappa_{12}\kappa_{20} + 3\kappa_{20}\kappa_{12} + 6\kappa_{21}\kappa_{11} + 3\kappa_{22}\kappa_{10}$	$+3\kappa_{13}$	$+3\kappa_{01}\kappa_{13} + 6\kappa_{02}\kappa_{12} + 3\kappa_{03}\kappa_{11} + 3\kappa_{11}\kappa_{03} + 6\kappa_{12}\kappa_{02} + 3\kappa_{13}\kappa_{01}$	$+3\kappa_{01}\kappa_{22} + 6\kappa_{02}\kappa_{21} + 3\kappa_{03}\kappa_{20} + 3\kappa_{11}\kappa_{12} + 6\kappa_{12}\kappa_{11} + 3\kappa_{13}\kappa_{10}$
$\dot{\kappa}_{31}$	.	$+1\kappa_{40}$	$+1\kappa_{10}\kappa_{40} + 3\kappa_{20}\kappa_{30} + 3\kappa_{30}\kappa_{20} + 1\kappa_{40}\kappa_{10}$	$+1\kappa_{31}$	$+1\kappa_{01}\kappa_{31} + 3\kappa_{11}\kappa_{21} + 3\kappa_{21}\kappa_{11} + 1\kappa_{31}\kappa_{01}$	$+1\kappa_{01}\kappa_{40} + 3\kappa_{11}\kappa_{30} + 3\kappa_{21}\kappa_{20} + 1\kappa_{31}\kappa_{10}$

TABLE B.1.2: Cumulant equation terms for coefficients b_{00} to b_{11} of Equation 2.2.2.

	c_{00}	c_{10}	c_{20}	c_{01}	c_{02}	c_{11}
$\dot{\kappa}_{10}$	.	.	.	.	.	.
$\dot{\kappa}_{20}$	1	$+1\kappa_{10}$	$+1\kappa_{20} + 1\kappa_{10}\kappa_{10}$	$+1\kappa_{01}$	$+1\kappa_{02} + 1\kappa_{01}\kappa_{01}$	$+1\kappa_{11} + 1\kappa_{01}\kappa_{10}$
$\dot{\kappa}_{30}$	.	$+3\kappa_{20}$	$+3\kappa_{30} + 3\kappa_{10}\kappa_{20} + 3\kappa_{20}\kappa_{10}$	$+3\kappa_{11}$	$+3\kappa_{12} + 3\kappa_{01}\kappa_{11} + 3\kappa_{11}\kappa_{01}$	$+3\kappa_{21} + 3\kappa_{01}\kappa_{20} + 3\kappa_{11}\kappa_{10}$
$\dot{\kappa}_{40}$	.	$+6\kappa_{30}$	$+6\kappa_{40} + 6\kappa_{10}\kappa_{30} + 12\kappa_{20}\kappa_{20} + 6\kappa_{30}\kappa_{10}$	$+6\kappa_{21}$	$+6\kappa_{22} + 6\kappa_{01}\kappa_{21} + 12\kappa_{11}\kappa_{11} + 6\kappa_{21}\kappa_{01}$	$+6\kappa_{31} + 6\kappa_{01}\kappa_{30} + 12\kappa_{11}\kappa_{20} + 6\kappa_{21}\kappa_{10}$
$\dot{\kappa}_{01}$	.	.	.	.	.	.
$\dot{\kappa}_{02}$	.	.	.	.	.	.
$\dot{\kappa}_{03}$	.	.	.	.	.	.
$\dot{\kappa}_{04}$	.	.	.	.	.	.
$\dot{\kappa}_{11}$	.	.	.	.	.	.
$\dot{\kappa}_{12}$	.	.	.	.	.	.
$\dot{\kappa}_{21}$	.	$+1\kappa_{11}$	$+1\kappa_{21} + 1\kappa_{10}\kappa_{11} + 1\kappa_{11}\kappa_{10}$	$+1\kappa_{02}$	$+1\kappa_{03} + 1\kappa_{01}\kappa_{02} + 1\kappa_{02}\kappa_{01}$	$+1\kappa_{12} + 1\kappa_{01}\kappa_{11} + 1\kappa_{02}\kappa_{10}$
$\dot{\kappa}_{22}$	.	$+1\kappa_{12}$	$+1\kappa_{22} + 1\kappa_{10}\kappa_{12} + 2\kappa_{11}\kappa_{11} + 1\kappa_{12}\kappa_{10}$	$+1\kappa_{03}$	$+1\kappa_{04} + 1\kappa_{01}\kappa_{03} + 2\kappa_{02}\kappa_{02} + 1\kappa_{03}\kappa_{01}$	$+1\kappa_{13} + 1\kappa_{01}\kappa_{12} + 2\kappa_{02}\kappa_{11} + 1\kappa_{03}\kappa_{10}$
$\dot{\kappa}_{13}$	.	.	.	.	.	.
$\dot{\kappa}_{31}$	.	$+3\kappa_{21}$	$+3\kappa_{31} + 3\kappa_{10}\kappa_{21} + 3\kappa_{11}\kappa_{20} + 3\kappa_{20}\kappa_{11} + 3\kappa_{21}\kappa_{10}$	$+3\kappa_{12}$	$+3\kappa_{13} + 3\kappa_{01}\kappa_{12} + 3\kappa_{02}\kappa_{11} + 3\kappa_{11}\kappa_{02} + 3\kappa_{12}\kappa_{01}$	$+3\kappa_{22} + 3\kappa_{01}\kappa_{21} + 3\kappa_{02}\kappa_{20} + 3\kappa_{11}\kappa_{11} + 3\kappa_{12}\kappa_{10}$

TABLE B.1.3: Cumulant equation terms for coefficients c_{00} to c_{11} of Equation 2.2.2.

	d_{00}	d_{10}	d_{20}	d_{01}	d_{02}	d_{11}
$\dot{\kappa}_{10}$	.	.	.	.	.	.
$\dot{\kappa}_{20}$	.	.	.	.	.	.
$\dot{\kappa}_{30}$	.	.	.	.	.	.
$\dot{\kappa}_{40}$	.	.	.	.	.	.
$\dot{\kappa}_{01}$	.	.	.	.	.	.
$\dot{\kappa}_{02}$	.	.	.	.	.	.
$\dot{\kappa}_{03}$	.	.	.	.	.	.
$\dot{\kappa}_{04}$	.	.	.	.	.	.
$\dot{\kappa}_{11}$	1	$+0.5\kappa_{10}$	$+0.5\kappa_{20} + 0.5\kappa_{10}\kappa_{10}$	$+0.5\kappa_{01}$	$+0.5\kappa_{02} + 0.5\kappa_{01}\kappa_{01}$	$+0.5\kappa_{11} + 0.5\kappa_{01}\kappa_{10}$
$\dot{\kappa}_{12}$	.	$+1\kappa_{11}$	$+1\kappa_{21} + 1\kappa_{10}\kappa_{11} + 1\kappa_{11}\kappa_{10}$	$+1\kappa_{02}$	$+1\kappa_{03} + 1\kappa_{01}\kappa_{02} + 1\kappa_{02}\kappa_{01}$	$+1\kappa_{12} + 1\kappa_{01}\kappa_{11} + 1\kappa_{02}\kappa_{10}$
$\dot{\kappa}_{21}$	.	$+1\kappa_{20}$	$+1\kappa_{30} + 1\kappa_{10}\kappa_{20} + 1\kappa_{20}\kappa_{10}$	$+1\kappa_{11}$	$+1\kappa_{12} + 1\kappa_{01}\kappa_{11} + 1\kappa_{11}\kappa_{01}$	$+1\kappa_{21} + 1\kappa_{01}\kappa_{20} + 1\kappa_{11}\kappa_{10}$
$\dot{\kappa}_{22}$	.	$+2\kappa_{21}$	$+2\kappa_{31} + 2\kappa_{10}\kappa_{21} + 2\kappa_{11}\kappa_{20} + 2\kappa_{20}\kappa_{11} + 2\kappa_{21}\kappa_{10}$	$+2\kappa_{12}$	$+2\kappa_{13} + 2\kappa_{01}\kappa_{12} + 2\kappa_{02}\kappa_{11} + 2\kappa_{11}\kappa_{02} + 2\kappa_{12}\kappa_{01}$	$+2\kappa_{22} + 2\kappa_{01}\kappa_{21} + 2\kappa_{02}\kappa_{20} + 2\kappa_{11}\kappa_{11} + 2\kappa_{12}\kappa_{10}$
$\dot{\kappa}_{13}$	.	$+1.5\kappa_{12}$	$+1.5\kappa_{22} + 1.5\kappa_{10}\kappa_{12} + 3\kappa_{11}\kappa_{11} + 1.5\kappa_{12}\kappa_{10}$	$+1.5\kappa_{03}$	$+1.5\kappa_{04} + 1.5\kappa_{01}\kappa_{03} + 3\kappa_{02}\kappa_{02} + 1.5\kappa_{03}\kappa_{01}$	$+1.5\kappa_{13} + 1.5\kappa_{01}\kappa_{12} + 3\kappa_{02}\kappa_{11} + 1.5\kappa_{03}\kappa_{10}$
$\dot{\kappa}_{31}$	.	$+1.5\kappa_{30}$	$+1.5\kappa_{40} + 1.5\kappa_{10}\kappa_{30} + 3\kappa_{20}\kappa_{20} + 1.5\kappa_{30}\kappa_{10}$	$+1.5\kappa_{21}$	$+1.5\kappa_{22} + 1.5\kappa_{01}\kappa_{21} + 3\kappa_{11}\kappa_{11} + 1.5\kappa_{21}\kappa_{01}$	$+1.5\kappa_{31} + 1.5\kappa_{01}\kappa_{30} + 3\kappa_{11}\kappa_{20} + 1.5\kappa_{21}\kappa_{10}$

TABLE B.1.4: Cumulant equation terms for coefficients d_{00} to d_{11} of Equation 2.2.2.

	e_{00}	e_{10}	e_{20}	e_{01}	e_{02}	e_{11}
$\dot{\kappa}_{10}$	.	.	.	.	.	.
$\dot{\kappa}_{20}$	.	.	.	.	.	.
$\dot{\kappa}_{30}$	.	.	.	.	.	.
$\dot{\kappa}_{40}$	.	.	.	.	.	.
$\dot{\kappa}_{01}$	.	.	.	.	.	.
$\dot{\kappa}_{02}$	.	.	.	.	.	.
$\dot{\kappa}_{03}$	.	.	.	.	.	.
$\dot{\kappa}_{04}$	.	.	.	.	.	.
$\dot{\kappa}_{11}$	1	$+0.5\kappa_{10}$	$+0.5\kappa_{20} + 0.5\kappa_{10}\kappa_{10}$	$+0.5\kappa_{01}$	$+0.5\kappa_{02} + 0.5\kappa_{01}\kappa_{01}$	$+0.5\kappa_{11} + 0.5\kappa_{01}\kappa_{10}$
$\dot{\kappa}_{12}$	.	$+1\kappa_{11}$	$+1\kappa_{21} + 1\kappa_{10}\kappa_{11} + 1\kappa_{11}\kappa_{10}$	$+1\kappa_{02}$	$+1\kappa_{03} + 1\kappa_{01}\kappa_{02} + 1\kappa_{02}\kappa_{01}$	$+1\kappa_{12} + 1\kappa_{01}\kappa_{11} + 1\kappa_{02}\kappa_{10}$
$\dot{\kappa}_{21}$	.	$+1\kappa_{20}$	$+1\kappa_{30} + 1\kappa_{10}\kappa_{20} + 1\kappa_{20}\kappa_{10}$	$+1\kappa_{11}$	$+1\kappa_{12} + 1\kappa_{01}\kappa_{11} + 1\kappa_{11}\kappa_{01}$	$+1\kappa_{21} + 1\kappa_{01}\kappa_{20} + 1\kappa_{11}\kappa_{10}$
$\dot{\kappa}_{22}$	.	$+2\kappa_{21}$	$+2\kappa_{31} + 2\kappa_{10}\kappa_{21} + 2\kappa_{11}\kappa_{20} + 2\kappa_{20}\kappa_{11} + 2\kappa_{21}\kappa_{10}$	$+2\kappa_{12}$	$+2\kappa_{13} + 2\kappa_{01}\kappa_{12} + 2\kappa_{02}\kappa_{11} + 2\kappa_{11}\kappa_{02} + 2\kappa_{12}\kappa_{01}$	$+2\kappa_{22} + 2\kappa_{01}\kappa_{21} + 2\kappa_{02}\kappa_{20} + 2\kappa_{11}\kappa_{11} + 2\kappa_{12}\kappa_{10}$
$\dot{\kappa}_{13}$	.	$+1.5\kappa_{12}$	$+1.5\kappa_{22} + 1.5\kappa_{10}\kappa_{12} + 3\kappa_{11}\kappa_{11} + 1.5\kappa_{12}\kappa_{10}$	$+1.5\kappa_{03}$	$+1.5\kappa_{04} + 1.5\kappa_{01}\kappa_{03} + 3\kappa_{02}\kappa_{02} + 1.5\kappa_{03}\kappa_{01}$	$+1.5\kappa_{13} + 1.5\kappa_{01}\kappa_{12} + 3\kappa_{02}\kappa_{11} + 1.5\kappa_{03}\kappa_{10}$
$\dot{\kappa}_{31}$	.	$+1.5\kappa_{30}$	$+1.5\kappa_{40} + 1.5\kappa_{10}\kappa_{30} + 3\kappa_{20}\kappa_{20} + 1.5\kappa_{30}\kappa_{10}$	$+1.5\kappa_{21}$	$+1.5\kappa_{22} + 1.5\kappa_{01}\kappa_{21} + 3\kappa_{11}\kappa_{11} + 1.5\kappa_{21}\kappa_{01}$	$+1.5\kappa_{31} + 1.5\kappa_{01}\kappa_{30} + 3\kappa_{11}\kappa_{20} + 1.5\kappa_{21}\kappa_{10}$

TABLE B.1.5: Cumulant equation terms for coefficients e_{00} to e_{11} of Equation 2.2.2.

	f_{00}	f_{10}	f_{20}	f_{01}	f_{02}	f_{11}
$\dot{\kappa}_{10}$	.	.	.	.	.	.
$\dot{\kappa}_{20}$	.	.	.	.	.	.
$\dot{\kappa}_{30}$	.	.	.	.	.	.
$\dot{\kappa}_{40}$	.	.	.	.	.	.
$\dot{\kappa}_{01}$	.	.	.	.	.	.
$\dot{\kappa}_{02}$	1	$+1\kappa_{10}$	$+1\kappa_{20} + 1\kappa_{10}\kappa_{10}$	$+1\kappa_{01}$	$+1\kappa_{02} + 1\kappa_{01}\kappa_{01}$	$+1\kappa_{11} + 1\kappa_{01}\kappa_{10}$
$\dot{\kappa}_{03}$	.	$+3\kappa_{11}$	$+3\kappa_{21} + 3\kappa_{10}\kappa_{11} + 3\kappa_{11}\kappa_{10}$	$+3\kappa_{02}$	$+3\kappa_{03} + 3\kappa_{01}\kappa_{02} + 3\kappa_{02}\kappa_{01}$	$+3\kappa_{12} + 3\kappa_{01}\kappa_{11} + 3\kappa_{02}\kappa_{10}$
$\dot{\kappa}_{04}$	.	$+6\kappa_{12}$	$+6\kappa_{22} + 6\kappa_{10}\kappa_{12} + 12\kappa_{11}\kappa_{11} + 6\kappa_{12}\kappa_{10}$	$+6\kappa_{03}$	$+6\kappa_{04} + 6\kappa_{01}\kappa_{03} + 12\kappa_{02}\kappa_{02} + 6\kappa_{03}\kappa_{01}$	$+6\kappa_{13} + 6\kappa_{01}\kappa_{12} + 12\kappa_{02}\kappa_{11} + 6\kappa_{03}\kappa_{10}$
$\dot{\kappa}_{11}$	.	.	.	.	.	.
$\dot{\kappa}_{12}$	.	$+1\kappa_{20}$	$+1\kappa_{30} + 1\kappa_{10}\kappa_{20} + 1\kappa_{20}\kappa_{10}$	$+1\kappa_{11}$	$+1\kappa_{12} + 1\kappa_{01}\kappa_{11} + 1\kappa_{11}\kappa_{01}$	$+1\kappa_{21} + 1\kappa_{01}\kappa_{20} + 1\kappa_{11}\kappa_{10}$
$\dot{\kappa}_{21}$	.	.	.	.	.	.
$\dot{\kappa}_{22}$	.	$+1\kappa_{30}$	$+1\kappa_{40} + 1\kappa_{10}\kappa_{30} + 2\kappa_{20}\kappa_{20} + 1\kappa_{30}\kappa_{10}$	$+1\kappa_{21}$	$+1\kappa_{22} + 1\kappa_{01}\kappa_{21} + 2\kappa_{11}\kappa_{11} + 1\kappa_{21}\kappa_{01}$	$+1\kappa_{31} + 1\kappa_{01}\kappa_{30} + 2\kappa_{11}\kappa_{20} + 1\kappa_{21}\kappa_{10}$
$\dot{\kappa}_{13}$	.	$+3\kappa_{21}$	$+3\kappa_{31} + 3\kappa_{10}\kappa_{21} + 3\kappa_{11}\kappa_{20} + 3\kappa_{20}\kappa_{11} + 3\kappa_{21}\kappa_{10}$	$+3\kappa_{12}$	$+3\kappa_{13} + 3\kappa_{01}\kappa_{12} + 3\kappa_{02}\kappa_{11} + 3\kappa_{11}\kappa_{02} + 3\kappa_{12}\kappa_{01}$	$+3\kappa_{22} + 3\kappa_{01}\kappa_{21} + 3\kappa_{02}\kappa_{20} + 3\kappa_{11}\kappa_{11} + 3\kappa_{12}\kappa_{10}$
$\dot{\kappa}_{31}$	.	.	.	.	.	.

TABLE B.1.6: Cumulant equation terms for coefficients f_{00} to f_{11} of Equation 2.2.2.

B.2 Butcher tableau for the Runge-Kutta-Fehlberg 4(5) method

Table B.2.1 gives the coefficients for the Runge-Kutta-Fehlberg 4(5) method in the so-called 'Butcher tableau' form. Terms listed under the a_i column relate at what point in time along the update horizon the RHS of the system of ODEs should be evaluated at a given stage of the update. The block of $b_{i,j}$ coefficients indicate how evaluations of the RHS of the system of ODEs are carried in during each stage of the algorithm. Finally, once all the stages of the algorithm are calculated the solution is approximated at the end of the update horizon. This is achieved by calculating a linear combination of the terms calculated at each stage of the algorithm. The coefficients of this combination are given by the c_j. Using different sequences of coefficients, a fourth and fifth order solution can be constructed. By comparing the approximation under calculated under each order, and estimate can be calculated of the error incurred during each update of the system.

	a_i	$b_{i,j}$					
i, j		0	1	2	3	4	5
0	.	.	.	.	.	.	.
1	$\frac{1}{4}$	$\frac{1}{4}$	.	.	.	.	.
2	$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$	.	.	.	.
3	$\frac{12}{13}$	$\frac{1932}{2197}$	$\frac{-7200}{2197}$	$\frac{7296}{2197}$	.	.	.
4	1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$\frac{-845}{4104}$	.	.
5	$\frac{1}{2}$	$\frac{-8}{27}$	2	$\frac{-3544}{2565}$	$\frac{1859}{4104}$	$\frac{-11}{40}$	.
	$c_j^{(4)}$	$\frac{25}{216}$	.	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$\frac{-1}{5}$	.
	$c_j^{(5)}$	$\frac{16}{135}$	.	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$\frac{-9}{50}$	$\frac{2}{55}$

TABLE B.2.1: Coefficients of the 4(5)-th order Runge-Kutta method of Fehlberg (1970) aranged in Butcher tableau form. Zero-valued coefficients indicated by '.'.

B.3 Butcher tableau for the Runge-Kutta 8(10) method

Table B.3.1 gives the coefficients for the tenth-order Runge-Kutta scheme of Feagin (2007). Due to the large number of coefficients of the scheme, we present the scheme using fractions which correspond to the coefficients given in the original text.

i,j	a_i	$b_{i,j}$															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	$\frac{1}{10}$	$\frac{1}{10}$	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
2	$\frac{26894}{49863}$	$\frac{-22366}{24439}$	$\frac{4354892}{2994011}$	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
3	$\frac{13447}{16621}$	$\frac{3599}{17794}$	.	$\frac{45265}{74599}$	.	.	.	.	.	.	.	.	.	.	.	.	.	.
4	$\frac{3943}{12759}$	$\frac{104303}{566788}$	.	$\frac{287919}{1454380}$	$\frac{-239}{3276}$	.	.	.	.	.	.	.	.	.	.	.	.	.
5	$\frac{11736}{119025}$	$\frac{98688}{1122721}$	.	.	$\frac{39760}{90867}$	$\frac{463369}{959925}$	.	.	.	.	.	.	.	.	.	.	.	.
6	$\frac{5}{6}$	$\frac{413}{4804}$	.	.	$\frac{8841399}{26720381}$	$\frac{9545}{19493}$	$\frac{-29100}{397619}$	.	.	.	.	.	.	.	.	.	.	.
7	$\frac{1877}{5302}$	$\frac{49347}{408061}$	.	.	.	$\frac{6301}{24223}$	$\frac{491}{15089}$	$\frac{-641}{10759}$	.	.	.	.	.	.	.	.	.	.
8	$\frac{11885}{13467}$	$\frac{2333159}{21047062}$	.	.	.	.	$\frac{-43633}{720300}$	$\frac{26300}{81737}$	$\frac{753602}{1476245}$	.	.	.	.	.	.	.	.	.
9	$\frac{3803}{5918}$	$\frac{122551}{1093674}$	.	.	.	.	$\frac{-2343}{16165}$	$\frac{-1433717}{4301972}$	$\frac{538369}{1078314}$	$\frac{693850}{1361813}$	.	.	.	.	.	.	.	.
10	$\frac{6028}{16867}$	$\frac{2435}{21364}$	.	.	.	.	$\frac{-352084}{4579577}$	$\frac{43462}{181449}$	$\frac{425599}{1069950}$	$\frac{3126757}{29070168 5}$	$\frac{-2317}{7069}$	.	.	.	.	.	.	.
11	$\frac{3599}{30637}$	$\frac{345249}{4324724}$	.	.	.	.	$\frac{-1130}{21717}$	$\frac{-15713}{272344}$	$\frac{805103}{4133356}$	$\frac{1101}{7573}$	$\frac{-224}{2861}$	$\frac{-33785}{295057}$	.	.	.	.	.	.
12	$\frac{5}{6}$	$\frac{45204}{45887}$	.	.	$\frac{8841399}{26720381}$	$\frac{9545}{19493}$	$\frac{-30693}{22258}$	$\frac{-166673}{19361}$	$\frac{30998}{5359}$	$\frac{729631}{221902}$	$\frac{-15442}{6471}$	$\frac{-57037}{17524}$	$\frac{-163255}{75461}$	.	.	.	.	.
13	$\frac{3943}{12759}$	$\frac{17557}{19615}$	.	$\frac{287919}{1454380}$	$\frac{-239}{3276}$	.	$\frac{-38973}{45784}$	$\frac{4268}{10715}$	$\frac{166141}{45651}$	$\frac{7255}{4686}$	$\frac{-35775 7}{16857 7}$	$\frac{-36343}{22951}$	$\frac{-66137}{38550}$	$\frac{-102766}{4211093}$	.	.	.	.
14	$\frac{26894}{49863}$	$\frac{-22366}{24439}$	$\frac{4354892}{2994011}$	.	.	$\frac{-3663994}{4713541}$	.	$\frac{-20510}{225163}$	.	.	.	.	.	$\frac{22483}{246823}$	$\frac{3263193}{4197931}$	$\frac{11726}{74603}$	.	.
15	$\frac{1}{10}$	$\frac{1}{10}$	.	$\frac{-11287}{71810}$	.	.	.	.	.	.	.	.	.	.	.	$\frac{-47193}{137686}$	$\frac{-27}{40}$	.
16	1	$\frac{6454098}{35504741}$	$\frac{27}{40}$	$\frac{72323}{211003}$	.	$\frac{286129}{1104271}$	$\frac{-225}{628}$	$\frac{-49715}{47531}$	$\frac{631217 47}{678489 28}$	$\frac{8563}{4812}$	$\frac{1}{10}$	$\frac{-905}{3203}$	$\frac{-76}{477}$	$\frac{-2760}{18967}$	$\frac{-114672}{442559}$	$\frac{-1}{30}$	$\frac{-1}{36}$	$\frac{1}{30}$
	$c_j^{(8)}$	$\frac{1}{30}$	$\frac{1}{36}$	$\frac{1}{30}$	.	$\frac{1}{20}$	.	$\frac{1}{25}$	.	$\frac{2648}{13993}$	$\frac{3095}{11156}$	$\frac{3095}{11156}$	$\frac{2648}{13993}$	$\frac{-1}{25}$	$\frac{-1}{20}$	$\frac{-1}{30}$	$\frac{-1}{36}$	$\frac{1}{30}$
	$c_j^{(10)}$	$\frac{1}{30}$	$\frac{1}{40}$	$\frac{1}{30}$	.	$\frac{1}{20}$	.	$\frac{1}{25}$	.	$\frac{2648}{13993}$	$\frac{3095}{11156}$	$\frac{3095}{11156}$	$\frac{2648}{13993}$	$\frac{-1}{25}$	$\frac{-1}{20}$	$\frac{-1}{30}$	$\frac{-1}{40}$	$\frac{1}{30}$

TABLE B.3.1: Coefficients of the 8(10)-th order Runge-Kutta method of Feagin (2007) aranged in Butcher tableau form. Note that some of the fractions used here may be approximate. The original text provides the coefficients in decimal form up to 60 significant digits. Zero-valued coefficients indicated by '.'.

B.4 Normalization of the Pearson-system densities

Although the Pearson system offers a considerable amount of flexibility with respect to carrying moments into a valid density approximation, this flexibility comes at the cost of having to compute normalizing constants. Unfortunately analytical expressions for integrals of the kernels in equations 2.3.25 - 2.3.28 over their respective support cannot in general be found for $d > 2$. As such we have to resort to numerical methods in order to normalize the Pearson densities. For example, for the Normal class we may employ the trapezoidal approximation:

$$\int_{-\infty}^{\infty} N(x|X_s)dx \approx \sum_{i=1}^{P-1} N\big(\tau_i|X_s\big)\rho_i\Delta \tag{B.4.1}$$

with the modifications $\tau_i = \frac{y_i}{1-(y_i)^2}e^{\alpha} + u_1$, $\rho_i = \frac{1+y_i^2}{(1-(y_i)^2)^2}e^{\alpha}$, where $P > 1$ is a positive integer with $y_i = -\mathcal{L} + 2i\Delta$ for $i = 0, 1, \ldots, P$.

Note that, in this particular instance, evaluation starts at (τ_1, ρ_1) and ends at (τ_{P-1}, ρ_{P-1}) since by definition $N\big(-\infty|X_s\big) = N\big(\infty|X_s\big) = 0$, thus negating the endpoints under the trapezoidal rule. The parameters $\mathcal{L}$ and Δ are in turn determined by the relations:

$$\mathcal{L} = -\frac{1}{2(u_1 - x^-)}\big(\sqrt{e^{2\alpha} + 4(u_1 - x^-)^2} - e^{\alpha}\big) \tag{B.4.2}$$

and

$$\Delta = \bigg(-\frac{1}{2(u_1 - x^+)}\big(\sqrt{e^{2\alpha} + 4(u_1 - x^+)^2} - e^{\alpha}\big) - \mathcal{L}\bigg)/P. \tag{B.4.3}$$

The parameters x^- and x^+ represent pre-defined lower and upper bounds on the integration range in the original coordinate system, i.e., the support of the diffusion. By evaluating the limits $\lim_{x^- \to -\infty} \tau$ and $\lim_{x^- \to \infty} \tau$, the behaviour of the infinite integral is preserved. However, in practice, these limits will typically be made finite in order to avoid numerical underflow at the extremes of a given distribution. The role of the parameters α and P is to control the mesh spacing within the limits $[x^-, x^+]$. For a given number of mesh points P, increasing α will lower the concentration of mesh points around u_1, whilst increasing P increases the mesh resolution. Finally, we note the important distinction that the resulting mesh under this scheme is in fact time dependent. This is due to the u_1 being included in the mesh translation. This, in conjunction with the exponential mesh spacing, ensures that the numerical integration accumulates

more information within 'dense' regions of the support and less information where there is little density, provided that it is unimodal. To see the effect of varying P and α parameters we follow up the example of Section 2.6.1 and use two sets of parameters for α and P:

```
R> M1 <- GQD.density(Xs = initial, Xt = states, s = Tstart, t = Tmax,
+   delt = increment, Dtype = 'Normal',  P  =  100, alpha = 1, lower  =  1,
+   upper  =  20)
R>
R> M2 <- GQD.density(Xs = initial, Xt = states, s = Tstart, t = Tmax,
+   delt = increment, Dtype = 'Normal',  P  =  200, alpha = 3, lower  =  1,
+   upper  =  20)
```

Subsequently, we can extract the mesh used for the normalization in each case and plot the mesh points as they vary over time. The resulting plots are given in Figure B.4.1.

```
R> plot(1, 1, type =  'n', xlim = c(1, 5), ylim = c(1, 20), xlab = 'Time (t)',
+   ylab  =  'Mesh')
R> for(i in 1:100)
+ {
+   lines(M1$mesh[i, ] ~ M1$time)
+ }
R> plot(1, 1, type =  'n', xlim = c(1, 5), ylim = c(1, 20), xlab = 'Time (t)',
+   ylab  =  'Mesh')
R> for(i in 1:200)
+ {
+   lines(M2$mesh[i, ] ~ M1$time)
+ }
```

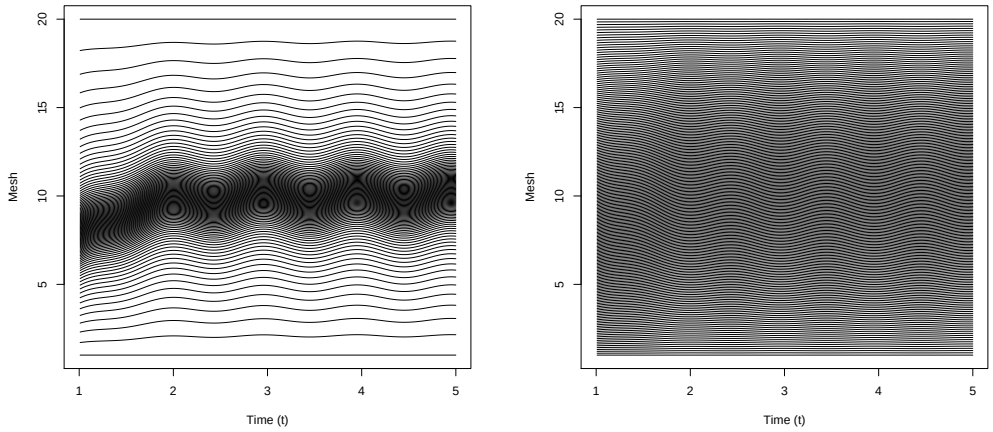


FIGURE B.4.1: Various mesh structures used for normalising the 4-th order Noramal type Pearson density. Using $\alpha = 1$ for $P = 100$ points over the interval $[1, 20]$ (left) results in a mesh that is concentrated arounf the mean trajectory of the process and sparse closer to the extrmes whilst setting $\alpha = 3$ for $P = 200$ (right) gives a more uniform mesh.

B.5 Inverting a matrix in order to evaluate the Pearson densities

In order to evaluate members of the Pearson system of densities, we are required to evaluate the system

$$\boldsymbol{\beta} = \mathbf{A}\mathbf{v} \tag{B.5.1}$$

for various forms of $\mathbf{v}$ as outlined in Section 2.3.3. This can be achieved numerically, however in order to ensure numerical stability of the algorithms developed here, we recommend solving these systems algebraically. Although this can be a tedious task, even for relatively small matrices, it can be easily achieved with the aid of a computer algebra system. Tables B.5.1 and B.5.2 give such expressions for elements of the inverse of $\mathbf{A}$ for $d = 8$ (i.e., a 5×5 matrix).

Element	Expression
$\lvert\mathbf{A}\rvert$	$-u_1(u_1(u_4(u_6u_8 - u_7u_7) - u_5(u_5u_8 - u_6u_7) + u_6(u_5u_7 - u_6u_6)) - u_3(u_2(u_6u_8 - u_7u_7) - u_5(u_3u_8 - u_4u_7) + u_6(u_3u_7 - u_4u_6)) + u_4(u_2(u_5u_8 - u_6u_7) - u_4(u_3u_8 - u_4u_7) + u_6(u_3u_6 - u_4u_5)) - u_5(u_2(u_5u_7 - u_6u_6) - u_4(u_3u_7 - u_4u_6) + u_5(u_3u_6 - u_4u_5)))+u_2(u_1(u_3(u_6u_8-u_7u_7)-u_5(u_4u_8-u_5u_7)+u_6(u_4u_7-u_5u_6))-u_2(u_2(u_6u_8-u_7u_7)-u_5(u_3u_8-u_4u_7)+u_6(u_3u_7-u_4u_6))+u_4(u_2(u_4u_8-u_5u_7)-u_3(u_3u_8-u_4u_7)+(u_3u_5 - u_4u_4)u_6) - u_5(u_2(u_4u_7 - u_5u_6) - u_3(u_3u_7 - u_4u_6) + u_5(u_3u_5 - u_4u_4))) - u_3(u_1(u_3(u_5u_8-u_6u_7)-u_4(u_4u_8-u_5u_7)+u_6(u_4u_6-u_5u_5))-u_2(u_2(u_5u_8-u_6u_7)-u_4(u_3u_8 - u_4u_7) + u_6(u_3u_6 - u_4u_5)) + u_3(u_2(u_4u_8 - u_5u_7) - u_3(u_3u_8 - u_4u_7) + (u_3u_5 - u_4u_4)u_6) - u_5(u_2(u_4u_6 - u_5u_5) - u_3(u_3u_6 - u_4u_5) + u_4(u_3u_5 - u_4u_4))) + u_2(u_4(u_6u_8 - u_7u_7) - u_5(u_5u_8 - u_6u_7) + u_6(u_5u_7 - u_6u_6)) - u_3(u_3(u_6u_8 - u_7u_7) - u_5(u_4u_8 - u_5u_7) + u_6(u_4u_7 - u_5u_6)) + u_4(u_3(u_5u_8 - u_6u_7) - u_4(u_4u_8 - u_5u_7) + u_6(u_4u_6 - u_5u_5)) + u_4(u_1(u_3(u_5u_7 - u_6u_6) - u_4(u_4u_7 - u_5u_6) + u_5(u_4u_6 - u_5u_5)) - u_2(u_2(u_5u_7 - u_6u_6) - u_4(u_3u_7 - u_4u_6) + u_5(u_3u_6 - u_4u_5)) + u_3(u_2(u_4u_7 - u_5u_6) - u_3(u_3u_7 - u_4u_6) + u_5(u_3u_5 - u_4u_4)) - u_4(u_2(u_4u_6 - u_5u_5) - u_3(u_3u_6 - u_4u_5) + u_4(u_3u_5 - u_4u_4))) - u_5(u_3(u_5u_7 - u_6u_6) - u_4(u_4u_7 - u_5u_6) + u_5(u_4u_6 - u_5u_5))$
$\mathbf{A}_{11}^{-1}\lvert\mathbf{A}\rvert$	$(u_2(u_4(u_6u_8 - u_7u_7) - u_5(u_5u_8 - u_6u_7) + u_6(u_5u_7 - u_6u_6)) - u_3(u_3(u_6u_8 - u_7u_7) - u_5(u_4u_8 - u_5u_7) + u_6(u_4u_7 - u_5u_6)) + u_4(u_3(u_5u_8 - u_6u_7) - u_4(u_4u_8 - u_5u_7) + u_6(u_4u_6 - u_5u_5)) - u_5(u_3(u_5u_7 - u_6u_6) - u_4(u_4u_7 - u_5u_6) + u_5(u_4u_6 - u_5u_5)))$
$\mathbf{A}_{12}^{-1}\lvert\mathbf{A}\rvert$	$(-u_1(u_4(u_6u_8 - u_7u_7) - u_5(u_5u_8 - u_6u_7) + u_6(u_5u_7 - u_6u_6)) + u_2(u_3(u_6u_8 - u_7u_7) - u_5(u_4u_8 - u_5u_7) + u_6(u_4u_7 - u_5u_6)) - u_3(u_3(u_5u_8 - u_6u_7) - u_4(u_4u_8 - u_5u_7) + u_6(u_4u_6 - u_5u_5)) + u_4(u_3(u_5u_7 - u_6u_6) - u_4(u_4u_7 - u_5u_6) + u_5(u_4u_6 - u_5u_5)))$
$\mathbf{A}_{13}^{-1}\lvert\mathbf{A}\rvert$	$(u_1(u_3(u_6u_8 - u_7u_7) - u_4(u_5u_8 - u_6u_7) + u_5(u_5u_7 - u_6u_6)) - u_2(u_2(u_6u_8 - u_7u_7) - u_4(u_4u_8 - u_5u_7) + u_5(u_4u_7 - u_5u_6)) + u_3(u_2(u_5u_8 - u_6u_7) - u_3(u_4u_8 - u_5u_7) + u_5(u_4u_6 - u_5u_5)) - u_4(u_2(u_5u_7 - u_6u_6) - u_3(u_4u_7 - u_5u_6) + u_4(u_4u_6 - u_5u_5)))$
$\mathbf{A}_{14}^{-1}\lvert\mathbf{A}\rvert$	$(-u_1(u_3(u_5u_8 - u_6u_7) - u_4(u_4u_8 - u_6u_6) + u_5(u_4u_7 - u_5u_6)) + u_2(u_2(u_5u_8 - u_6u_7) - u_4(u_3u_8 - u_5u_6) + u_5(u_3u_7 - u_5u_5)) - u_3(u_2(u_4u_8 - u_6u_6) - u_3(u_3u_8 - u_5u_6) + u_5(u_3u_6 - u_4u_5)) + u_4(u_2(u_4u_7 - u_5u_6) - u_3(u_3u_7 - u_5u_5) + u_4(u_3u_6 - u_4u_5)))$
$\mathbf{A}_{15}^{-1}\lvert\mathbf{A}\rvert$	$(u_1(u_3(u_5u_7 - u_6u_6) - u_4(u_4u_7 - u_5u_6) + u_5(u_4u_6 - u_5u_5)) - u_2(u_2(u_5u_7 - u_6u_6) - u_4(u_3u_7 - u_4u_6) + u_5(u_3u_6 - u_4u_5)) + u_3(u_2(u_4u_7 - u_5u_6) - u_3(u_3u_7 - u_4u_6) + u_5(u_3u_5 - u_4u_4)) - u_4(u_2(u_4u_6 - u_5u_5) - u_3(u_3u_6 - u_4u_5) + u_4(u_3u_5 - u_4u_4)))$
$\mathbf{A}_{21}^{-1}\lvert\mathbf{A}\rvert$	$(-u_1(u_4(u_6u_8 - u_7u_7) - u_5(u_5u_8 - u_6u_7) + u_6(u_5u_7 - u_6u_6)) + u_3(u_2(u_6u_8 - u_7u_7) - u_5(u_3u_8 - u_4u_7) + u_6(u_3u_7 - u_4u_6)) - u_4(u_2(u_5u_8 - u_6u_7) - u_4(u_3u_8 - u_4u_7) + u_6(u_3u_6 - u_4u_5)) + u_5(u_2(u_5u_7 - u_6u_6) - u_4(u_3u_7 - u_4u_6) + u_5(u_3u_6 - u_4u_5)))$
$\mathbf{A}_{22}^{-1}\lvert\mathbf{A}\rvert$	$(-u_2(u_2(u_6u_8 - u_7u_7) - u_5(u_3u_8 - u_4u_7) + u_6(u_3u_7 - u_4u_6)) + u_3(u_2(u_5u_8 - u_6u_7) - u_4(u_3u_8 - u_4u_7) + u_6(u_3u_6 - u_4u_5)) + u_4(u_6u_8 - u_7u_7) - u_5(u_5u_8 - u_6u_7) - u_4(u_2(u_5u_7 - u_6u_6) - u_4(u_3u_7 - u_4u_6) + u_5(u_3u_6 - u_4u_5)) + u_6(u_5u_7 - u_6u_6))$
$\mathbf{A}_{23}^{-1}\lvert\mathbf{A}\rvert$	$(u_2(u_1(u_6u_8 - u_7u_7) - u_4(u_3u_8 - u_4u_7) + u_5(u_3u_7 - u_4u_6)) - u_3(u_1(u_5u_8 - u_6u_7) - u_3(u_3u_8 - u_4u_7) + u_5(u_3u_6 - u_4u_5)) - u_3(u_6u_8 - u_7u_7) + u_4(u_5u_8 - u_6u_7) + u_4(u_1(u_5u_7 - u_6u_6) - u_3(u_3u_7 - u_4u_6) + u_4(u_3u_6 - u_4u_5)) - u_5(u_5u_7 - u_6u_6))$
$\mathbf{A}_{24}^{-1}\lvert\mathbf{A}\rvert$	$(-u_2(u_1(u_5u_8 - u_6u_7) - u_4(u_2u_8 - u_4u_6) + u_5(u_2u_7 - u_4u_5)) + u_3(u_1(u_4u_8 - u_6u_6) - u_3(u_2u_8 - u_4u_6) + u_5(u_2u_6 - u_4u_4)) + u_3(u_5u_8 - u_6u_7) - u_4(u_4u_8 - u_6u_6) - u_4(u_1(u_4u_7 - u_5u_6) - u_3(u_2u_7 - u_4u_5) + u_4(u_2u_6 - u_4u_4)) + u_5(u_4u_7 - u_5u_6))$
$\mathbf{A}_{25}^{-1}\lvert\mathbf{A}\rvert$	$(u_2(u_1(u_5u_7 - u_6u_6) - u_4(u_2u_7 - u_3u_6) + u_5(u_2u_6 - u_3u_5)) - u_3(u_1(u_4u_7 - u_5u_6) - u_3(u_2u_7 - u_3u_6) + u_5(u_2u_5 - u_3u_4)) - u_3(u_5u_7 - u_6u_6) + u_4(u_4u_7 - u_5u_6) + u_4(u_1(u_4u_6 - u_5u_5) - u_3(u_2u_6 - u_3u_5) + u_4(u_2u_5 - u_3u_4)) - u_5(u_4u_6 - u_5u_5))$

TABLE B.5.1: Elements of the matrix inverse required to evaluate any of the Pearson type densities for $d = 8$.

Element	Expression
$\mathbf{A}_{31}^{-1}\lvert\mathbf{A}\rvert$	$(u_1(u_3(u_6u_8-u_7u_7)-u_5(u_4u_8-u_5u_7)+u_6(u_4u_7-u_5u_6))-u_2(u_2(u_6u_8-u_7u_7)-u_5(u_3u_8-u_4u_7)+u_6(u_3u_7-u_4u_6))+u_4(u_2(u_4u_8-u_5u_7)-u_3(u_3u_8-u_4u_7)+(u_3u_5-u_4u_4)u_6)-u_5(u_2(u_4u_7-u_5u_6)-u_3(u_3u_7-u_4u_6)+u_5(u_3u_5-u_4u_4)))$
$\mathbf{A}_{32}^{-1}\lvert\mathbf{A}\rvert$	$(u_1(u_2(u_6u_8-u_7u_7)-u_5(u_3u_8-u_4u_7)+u_6(u_3u_7-u_4u_6))-u_3(u_2(u_4u_8-u_5u_7)-u_3(u_3u_8-u_4u_7)+(u_3u_5-u_4u_4)u_6)-u_3(u_6u_8-u_7u_7)+u_5(u_4u_8-u_5u_7)+u_4(u_2(u_4u_7-u_5u_6)-u_3(u_3u_7-u_4u_6)+u_5(u_3u_5-u_4u_4))-u_6(u_4u_7-u_5u_6))$
$\mathbf{A}_{33}^{-1}\lvert\mathbf{A}\rvert$	$(-u_1(u_1(u_6u_8-u_7u_7)-u_4(u_3u_8-u_4u_7)+u_5(u_3u_7-u_4u_6))+u_3(u_1(u_4u_8-u_5u_7)-u_2(u_3u_8-u_4u_7)+u_5(u_3u_5-u_4u_4))+u_2(u_6u_8-u_7u_7)-u_4(u_4u_8-u_5u_7)-u_4(u_1(u_4u_7-u_5u_6)-u_2(u_3u_7-u_4u_6)+u_4(u_3u_5-u_4u_4))+u_5(u_4u_7-u_5u_6))$
$\mathbf{A}_{34}^{-1}\lvert\mathbf{A}\rvert$	$(u_1(u_1(u_5u_8-u_6u_7)-u_4(u_2u_8-u_4u_6)+u_5(u_2u_7-u_4u_5))-u_3(u_1(u_3u_8-u_5u_6)-u_2(u_2u_8-u_4u_6)+u_5(u_2u_5-u_3u_4))-u_2(u_5u_8-u_6u_7)+u_4(u_3u_8-u_5u_6)+u_4(u_1(u_3u_7-u_5u_5)-u_2(u_2u_7-u_4u_5)+u_4(u_2u_5-u_3u_4))-u_5(u_3u_7-u_5u_5))$
$\mathbf{A}_{35}^{-1}\lvert\mathbf{A}\rvert$	$(-u_1(u_1(u_5u_7-u_6u_6)-u_4(u_2u_7-u_3u_6)+u_5(u_2u_6-u_3u_5))+u_3(u_1(u_3u_7-u_4u_6)-u_2(u_2u_7-u_3u_6)+(u_2u_4-u_3u_3)u_5)+u_2(u_5u_7-u_6u_6)-u_4(u_3u_7-u_4u_6)-u_4(u_1(u_3u_6-u_4u_5)-u_2(u_2u_6-u_3u_5)+u_4(u_2u_4-u_3u_3))+u_5(u_3u_6-u_4u_5))$
$\mathbf{A}_{41}^{-1}\lvert\mathbf{A}\rvert$	$(-u_1(u_3(u_5u_8-u_6u_7)-u_4(u_4u_8-u_5u_7)+u_6(u_4u_6-u_5u_5))+u_2(u_2(u_5u_8-u_6u_7)-u_4(u_3u_8-u_4u_7)+u_6(u_3u_6-u_4u_5))-u_3(u_2(u_4u_8-u_5u_7)-u_3(u_3u_8-u_4u_7)+(u_3u_5-u_4u_4)u_6)+u_5(u_2(u_4u_6-u_5u_5)-u_3(u_3u_6-u_4u_5)+u_4(u_3u_5-u_4u_4)))$
$\mathbf{A}_{42}^{-1}\lvert\mathbf{A}\rvert$	$(-u_1(u_2(u_5u_8-u_6u_7)-u_4(u_3u_8-u_4u_7)+u_6(u_3u_6-u_4u_5))+u_2(u_2(u_4u_8-u_5u_7)-u_3(u_3u_8-u_4u_7)+(u_3u_5-u_4u_4)u_6)+u_3(u_5u_8-u_6u_7)-u_4(u_4u_8-u_5u_7)-u_4(u_2(u_4u_6-u_5u_5)-u_3(u_3u_6-u_4u_5)+u_4(u_3u_5-u_4u_4))+u_6(u_4u_6-u_5u_5))$
$\mathbf{A}_{43}^{-1}\lvert\mathbf{A}\rvert$	$(u_1(u_1(u_5u_8-u_6u_7)-u_3(u_3u_8-u_4u_7)+u_5(u_3u_6-u_4u_5))-u_2(u_1(u_4u_8-u_5u_7)-u_2(u_3u_8-u_4u_7)+u_5(u_3u_5-u_4u_4))-u_2(u_5u_8-u_6u_7)+u_3(u_4u_8-u_5u_7)+u_4(u_1(u_4u_6-u_5u_5)-u_2(u_3u_6-u_4u_5)+u_3(u_3u_5-u_4u_4))-u_5(u_4u_6-u_5u_5))$
$\mathbf{A}_{44}^{-1}\lvert\mathbf{A}\rvert$	$(-u_1(u_1(u_4u_8-u_6u_6)-u_3(u_2u_8-u_4u_6)+u_5(u_2u_6-u_4u_4))+u_2(u_1(u_3u_8-u_5u_6)-u_2(u_2u_8-u_4u_6)+u_5(u_2u_5-u_3u_4))+u_2(u_4u_8-u_6u_6)-u_3(u_3u_8-u_5u_6)-u_4(u_1(u_3u_6-u_4u_5)-u_2(u_2u_6-u_4u_4)+u_3(u_2u_5-u_3u_4))+u_5(u_3u_6-u_4u_5))$
$\mathbf{A}_{45}^{-1}\lvert\mathbf{A}\rvert$	$(u_1(u_1(u_4u_7-u_5u_6)-u_3(u_2u_7-u_3u_6)+u_5(u_2u_5-u_3u_4))-u_2(u_1(u_3u_7-u_4u_6)-u_2(u_2u_7-u_3u_6)+(u_2u_4-u_3u_3)u_5)-u_2(u_4u_7-u_5u_6)+u_3(u_3u_7-u_4u_6)+u_4(u_1(u_3u_5-u_4u_4)-u_2(u_2u_5-u_3u_4)+u_3(u_2u_4-u_3u_3))-u_5(u_3u_5-u_4u_4))$
$\mathbf{A}_{51}^{-1}\lvert\mathbf{A}\rvert$	$(u_1(u_3(u_5u_7-u_6u_6)-u_4(u_4u_7-u_5u_6)+u_5(u_4u_6-u_5u_5))-u_2(u_2(u_5u_7-u_6u_6)-u_4(u_3u_7-u_4u_6)+u_5(u_3u_6-u_4u_5))+u_3(u_2(u_4u_7-u_5u_6)-u_3(u_3u_7-u_4u_6)+u_5(u_3u_5-u_4u_4))-u_4(u_2(u_4u_6-u_5u_5)-u_3(u_3u_6-u_4u_5)+u_4(u_3u_5-u_4u_4)))$
$\mathbf{A}_{52}^{-1}\lvert\mathbf{A}\rvert$	$(u_1(u_2(u_5u_7-u_6u_6)-u_4(u_3u_7-u_4u_6)+u_5(u_3u_6-u_4u_5))-u_2(u_2(u_4u_7-u_5u_6)-u_3(u_3u_7-u_4u_6)+u_5(u_3u_5-u_4u_4))-u_3(u_5u_7-u_6u_6)+u_4(u_4u_7-u_5u_6)+u_3(u_2(u_4u_6-u_5u_5)-u_3(u_3u_6-u_4u_5)+u_4(u_3u_5-u_4u_4))-u_5(u_4u_6-u_5u_5))$
$\mathbf{A}_{53}^{-1}\lvert\mathbf{A}\rvert$	$(-u_1(u_1(u_5u_7-u_6u_6)-u_3(u_3u_7-u_4u_6)+u_4(u_3u_6-u_4u_5))+u_2(u_1(u_4u_7-u_5u_6)-u_2(u_3u_7-u_4u_6)+u_4(u_3u_5-u_4u_4))+u_2(u_5u_7-u_6u_6)-u_3(u_4u_7-u_5u_6)-u_3(u_1(u_4u_6-u_5u_5)-u_2(u_3u_6-u_4u_5)+u_3(u_3u_5-u_4u_4))+u_4(u_4u_6-u_5u_5))$
$\mathbf{A}_{54}^{-1}\lvert\mathbf{A}\rvert$	$(u_1(u_1(u_4u_7-u_5u_6)-u_3(u_2u_7-u_4u_5)+u_4(u_2u_6-u_4u_4))-u_2(u_1(u_3u_7-u_5u_5)-u_2(u_2u_7-u_4u_5)+u_4(u_2u_5-u_3u_4))-u_2(u_4u_7-u_5u_6)+u_3(u_3u_7-u_5u_5)+u_3(u_1(u_3u_6-u_4u_5)-u_2(u_2u_6-u_4u_4)+u_3(u_2u_5-u_3u_4))-u_4(u_3u_6-u_4u_5))$
$\mathbf{A}_{55}^{-1}\lvert\mathbf{A}\rvert$	$(-u_1(u_1(u_4u_6-u_5u_5)-u_3(u_2u_6-u_3u_5)+u_4(u_2u_5-u_3u_4))+u_2(u_1(u_3u_6-u_4u_5)-u_2(u_2u_6-u_3u_5)+u_4(u_2u_4-u_3u_3))+u_2(u_4u_6-u_5u_5)-u_3(u_3u_6-u_4u_5)-u_3(u_1(u_3u_5-u_4u_4)-u_2(u_2u_5-u_3u_4)+u_3(u_2u_4-u_3u_3))+u_4(u_3u_5-u_4u_4))$

TABLE B.5.2: Elements of the matrix inverse required to evaluate any of the Pearson type densities for $d = 8$ (continued).

Appendix C

First Passage Time Problems

C.1 Simulating first passage times

By combining Equation 3.2.16 with the cumulant truncation procedure it is possible to analyse quite complicated first passage time problems for single-threshold regimes. Unfortunately, it remains difficult to verify the quality of the approximations independently without making reference to tractable first passage time problems. Indeed, for cases where the first passage time density can indeed be calculated analytically, it is easy to verify that Equation 3.2.16 produces astonishingly accurate approximations. Naturally, these results cannot be extrapolated beyond the scope of the specific examples and thus one has to resort to alternative methods for gauging the quality of the approximation. For these purposes, we may resort to simulation as a means of profiling first passage time problems for which the constituents do not permit the calculation of a closed-form solution or benchmark distribution.

As a first attempt at devising a scheme for simulating first passage times, one might be tempted to simply simulate the trajectory of the underlying diffusion model and subsequently monitor the trajectory of the process until the threshold function is crossed. That is, we may simulate the trajectory of the diffusion process using a Euler-Maruyama scheme by which the trajectory of the diffusion is approximated iteratively using

$$X_{t_i} = X_{t_{i-1}} + \mu(X_{t_{i-1}}, t_{i-1})(t_i - t_{i-1}) + \sigma(X_{t_{i-1}}, t_{i-1})Z \tag{C.1.1}$$

for $i = 0, 1, 2, \ldots$, where Z is a $\mathrm{N}(0, t_i - t_{i-1})$ random deviate. At each iteration, we can check the condition $X_{t_i} \geq \lambda_{t_i}$ and if the condition is met we record t_i as the

corresponding first passage time. By repeating this process a number of times we can calculate a frequency distribution of the first passage time variable. Although this strategy makes intuitive sense, Giraudo and Sacerdote (1999) point out that such a strategy would produce inherently biased first passage times. Indeed, this is a subtle problem and the source of the bias is rooted in the discrepancy between the discrete approximation used to simulate the trajectory of the process and the fractal nature of the true underlying diffusion process: In the context of simulating the diffusion trajectory, it suffices to choose a small enough step size for the simulation so that the transition density may be approximately Normal and the dynamics of the overall trajectory will mimic the behaviour of the diffusion process reasonably well. However, in the context of first passage time problems where the process is moving in proximity to a threshold function, the discrepancy between the discrete dynamics and the true dynamics propagates forward in time. Specifically, Giraudo and Sacerdote (1999) point out that regardless of how small the simulation step size is made, probabilistically speaking it is always possible to reach the threshold function within such an interval. Consequently, when the process is simulated and the trajectory is not 'observed' to cross the threshold function, one has omitted the possibility that it has already reached the threshold function. In other words, the probability flow at the threshold is incorrectly accounted for under discrete simulation and the resulting first passage times will be slightly biased in the sense that on average, the simulated trajectory will have survived too long. Consequently, first passage times simulated in this way will typically be slightly larger than expected under the true process. The mechanics of this phenomenon is very subtle, and can easily be missed when conducting simulation experiments heuristically. Indeed, later in this thesis, we encounter a first passage time problem which cannot be solved in this way despite being tractable numerically by way of a PDE for the survival probability surface.

In simple applications, it can often suffice for purposes of the analysis to use an extremely small time-step in conjunction with a large number of simulations in order to approximate the shape of the first passage time distribution. However, where the application calls for more precise analysis, it is possible to improve this strategy. Giraudo and Sacerdote (1999) develop a scheme which accounts for the probability of intermediate visits to the threshold function: By treating the trajectory of the process over each transition horizon as a Brownian motion with drift which transits to the threshold function and back, the authors approximate the probability of such an event as:

$$\exp\left\{-\frac{2(\lambda^2-\lambda X_{t_{i-1}}-\lambda X_{t_{i+1}}+X_{t_{i-1}}X_{t_i})}{2(t_i-t_{i-1})\sigma^2(X_{t_{i-1}})}\right\} \tag{C.1.2}$$

at each iteration for a constant threshold λ and where the diffusion process is

assumed to be time-homogeneous (hence $\sigma^2(X_{t_{i-1}})$). Using this strategy, the updating scheme can then be modified by terminating the simulation at each iteration with said probability and subsequently recording the first passage time.

To illustrate the phenomena, consider the first passage time problem for an Ornstein-Uhlenbeck process initially posed in Section 3.2. Figure C.1.1 compares simulated first passage time densities under the unmodified Euler-Maruyama scheme and the modified scheme which employs Equation C.1.2 (using identical seed values for the random number generators) to the true first passage time density under Equation 3.2.2. In each case, we simulate 100 000 first passage times in order to calculate the corresponding frequency distribution and repeat the experiment for constant step sizes $(t_i - t_{i-1})$ of 0.01 and 0.001 time units respectively. For purposes of the experiment, we assume the parameter set $\{a, b, \sigma\} = \{1, 10, 1\}$ and set the initial value of the diffusion to $X_0 = 8$ The results suggest that the discretization-bias is significantly reduced under the modified scheme for $t_i - t_{i-1} = 0.01$ although the overall shape of the distribution appears correct. For $t_i - t_{i-1} = 0.001$ the bias is less apparent with the primary discrepancy being at the mode of the first passage time density.

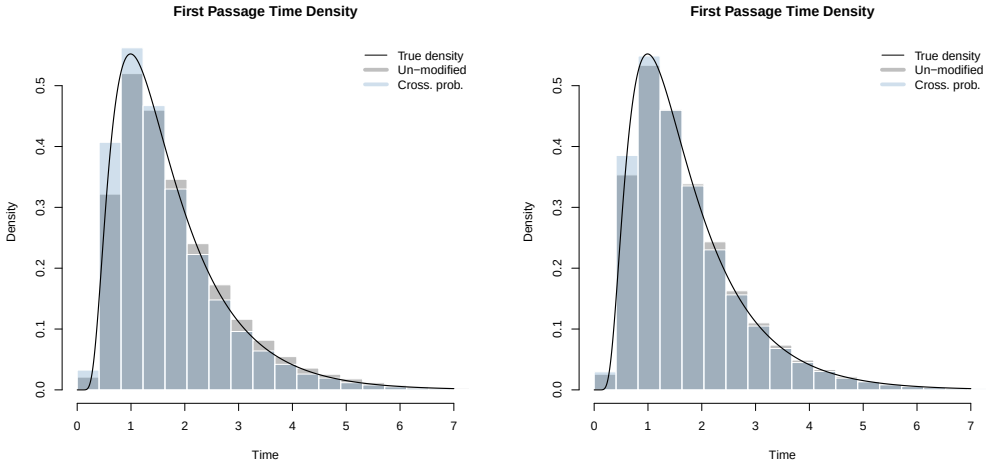


FIGURE C.1.1: Simulated first passage time densities under the unmodified Euler-Maruyama scheme (gray) and the modified scheme using Equation C.1.2 (light blue) for Equation 3.2.2 calculated using Equation C.1.1 with step sizes of $t_i - t_{i-1} = 0.01$ (left) and $t_i - t_{i-1} = 0.001$ (right) respectively. Superimposed we plot the true first passage time density. R code: Supplementary materials, Section 3.7.

Repeating the experiment for a more complicated diffusion, let:

$$dX_t = \alpha(\beta + 3\sin(2\pi t) - X_t)dt + \sigma\sqrt{(1 + 0.3\cos(2\pi t))^2 X_t}dB_t \qquad \text{(C.1.3)}$$

with $\{\alpha, \beta, \sigma\} = \{0.5, 10, 0.5\}$ and set $X_0 = 9$ and $\lambda_t = 11$. Again, using 100 000 simulated first passage times for constant step sizes $(t_i - t_{i-1})$ of 0.01 and 0.001 time units respectively, we calculate the frequency distribution of the simulated first passage times (Figure C.1.2). For comparison, we use the cumulant truncation procedure in conjunction with Equation 3.2.16 in order to approximate the first passage time distribution numerically. As in the case of the Ornstein-Uhlenbeck process, the discrepancy between the modified and unmodified scheme diminishes for the smaller simulation step size and the overall shape of the distribution remains consistent with that of the modified scheme. Furthermore, the first passage time density calculated using Equation 3.2.16 matches the simulated first passage time densities closely.

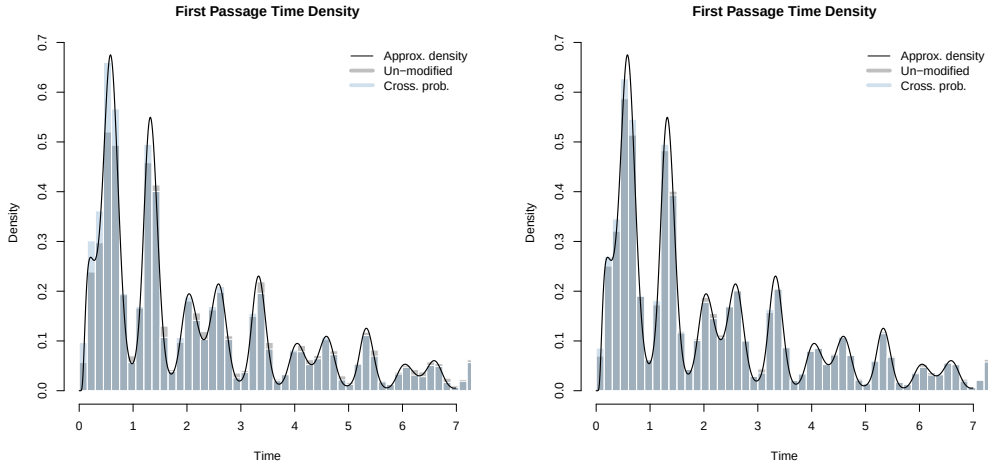


FIGURE C.1.2: Simulated first passage time densities under the unmodified Euler-Maruyama scheme (gray) and the modified scheme using Equation C.1.2 (light blue) for Equation C.1.3 calculated using Equation C.1.1 with step sizes of $t_i - t_{i-1} = 0.01$ (left) and $t_i - t_{i-1} = 0.001$ (right) respectively. Superimposed we plot the approximate first passage time density calculated using Equation 3.2.16. R code: Supplementary materials, Section 3.7.

Although the focus of this work is not on the simulation of first passage time problems, we note the subtleties that arise when doing so for completeness. Where needed, we will make reference to the technique used to simulated the first passage time density and indicate the parameters used in order to conduct the simulation

experiment. Throughout, we apply a suitably small step size and large sample size in order to mitigate the bias caused by discrete simulation.

C.2 C++ code for evaluating a first passage time density

The C++ code listed here is constructed during a call to the `GQD.TIpassage()` function in the non-linear first passage time problem analysed in Section 3.2.3.1. Here, the functions are constructed in order to evaluate Equation 3.2.17 for $d = 4$ as used in Equation 3.2.16. `solver()` evaluates the elements $\psi(\lambda_{t_i}|X_{t_0})$ for $i = 1, 2, \ldots N$ and $\psi(\lambda_{t_i}|\lambda_{t_k})$ for $k < i : k, i = 1, 2, \ldots N$ after which these elements are brought together and Equation 3.2.16 is evaluated using `ieq()`. Note that we use the tenth-order Runge-Kutta scheme of Feagin (2007) (see Appendix B.3 for the corresponding Butcher tableau) in order to solve the cumulant equations of the model (the cumulant equations are stated within the body of the `f()` function in the listed code).

```
#include <RcppArmadillo.h>
#include <math.h>
#include <Rcpp.h>
#define pi           3.14159265358979323846  /* pi */
using namespace arma;
using namespace Rcpp;
using namespace R;

// [[Rcpp::depends("RcppArmadillo")]]
// [[Rcpp::export]]
vec prod(vec a,vec b)
{
 return(a%b);
}

mat f(mat a,vec theta,vec t,int N2)
{
   mat atemp(N2,4);
  atemp.col(0)=(+(theta[1]*(10+0.2*sin(2*pi*t)+0.3*prod(sqrt(t),1+cos(3*pi*t))))%a.col(0)+(-theta
     [1])*(a.col(1)+a.col(0)%a.col(0)));
  atemp.col(1)=(+(theta[1]*(10+0.2*sin(2*pi*t)+0.3*prod(sqrt(t),1+cos(3*pi*t))))%(2*a.col(1))+(-
     theta[1])*(2*a.col(2)+4*a.col(0)%a.col(1))+(0.1)*(a.col(1)+a.col(0)%a.col(0)));
  atemp.col(2)=(+(theta[1]*(10+0.2*sin(2*pi*t)+0.3*prod(sqrt(t),1+cos(3*pi*t))))%(3*a.col(2))+(-
     theta[1])*(3*a.col(3)+6*a.col(0)%a.col(2)+6*a.col(1)%a.col(1))+(0.1)*(3*a.col(2)+6*a.col(0)%a.
     col(1)));
  atemp.col(3)=(+(theta[1]*(10+0.2*sin(2*pi*t)+0.3*prod(sqrt(t),1+cos(3*pi*t))))%(4*a.col(3))+(-
     theta[1])*(8*a.col(0)%a.col(3)+24*a.col(1)%a.col(2))+(0.1)*(6*a.col(3)+12*a.col(0)%a.col(2)+12
     *a.col(1)%a.col(1)));
  return atemp;
}

// [[Rcpp::export]]
vec saddle(mat xx,double Bt)
{
  vec p=(1.0/3.0) *(3*(xx.col(3)/6.0)%xx.col(1) - pow(xx.col(2)/2.0,2))/pow(xx.col(3)/6.0,2);
  vec q=(1.0/27.0)*(27*pow(xx.col(3)/6.0,2)%(xx.col(0)-Bt) - 9*(xx.col(3)/6.0)%(xx.col(2)/2.0)%xx.
     col(1) + 2*pow(xx.col(2)/2.0,3))/pow(xx.col(3)/6.0,3);
  vec chk=pow(q,2)/4.0 + pow(p,3)/27.0;
  vec th=-(xx.col(2)/2.0)/(3*(xx.col(3)/6.0))+pow(-q/2.0+sqrt(chk),(1.0/3.0))-pow(q/2.0+sqrt(chk)
     ,(1.0/3.0));

  vec K =xx.col(0)%th+(xx.col(1)%th%th)/2.0+(xx.col(2)%th%th%th)/6.0 +(xx.col(3)%th%th%th%th)/24.0;
  vec K1=xx.col(0)    +(xx.col(1)%th)       +(xx.col(2)%th%th)/2.0    +(xx.col(3)%th%th%th)/6.0;
  vec K2=xx.col(1)    +(xx.col(2)%th)       +(xx.col(3)%th%th)/2.0;
```

```
  vec val=exp(-0.5*log(2*3.141592653589793*K2)+(K-th%K1));
  return(val);
}
// [[Rcpp::export]]
vec saddle2(mat xx,double Bt)
{
  vec p=(1.0/3.0) *(3*(xx.col(3)/6.0)%xx.col(1) - pow(xx.col(2)/2.0,2))/pow(xx.col(3)/6.0,2);
  vec q=(1.0/27.0)*(27*pow(xx.col(3)/6.0,2)%(xx.col(0)-Bt) - 9*(xx.col(3)/6.0)%(xx.col(2)/2.0)%xx.
     col(1) + 2*pow(xx.col(2)/2.0,3))/pow(xx.col(3)/6.0,3);
  vec chk=pow(q,2)/4.0 + pow(p,3)/27.0;
  vec th=-(xx.col(2)/2.0)/(3*(xx.col(3)/6.0))+pow(-q/2.0+sqrt(chk),(1.0/3.0))-pow(q/2.0+sqrt(chk)
     ,(1.0/3.0));
  vec thdash=1.0/(xx.col(1)+th%xx.col(2)+0.5*th%th%xx.col(3));

  vec K =xx.col(0)%th+(xx.col(1)%th%th)/2.0+(xx.col(2)%th%th%th)/6.0 +(xx.col(3)%th%th%th%th)/24.0;
  vec K1=xx.col(0)    +(xx.col(1)%th)       +(xx.col(2)%th%th)/2.0    +(xx.col(3)%th%th%th)/6.0;
  vec K2=xx.col(1)    +(xx.col(2)%th)       +(xx.col(3)%th%th)/2.0;
  vec gg      = 1.0/sqrt(2*3.141592653589793*(K2));
  vec ggdash = -3.141592653589793*pow(2*3.141592653589793*(K2),-3.0/2.0)%(xx.col(2)%thdash+xx.col
     (3)%thdash%th);
  vec hh      = exp(K-th%K1);
  vec hhdash = exp(K-th%K1)%(-th%thdash%xx.col(1)-th%th%thdash%xx.col(2)-0.5*(th%th%th)%thdash%xx.
     col(3));

  vec val=(ggdash%hh+gg%hhdash);
  return(val);
}

// [[Rcpp::export]]
vec pcurr(mat xx,double Bt,vec theta,vec t)
{

  vec val =0.5*(+(theta[1] * (10 + 0.2 * sin(2 * pi * t) + 0.3 * prod(sqrt(t), 1 + cos(3 * pi * t))
     ))*Bt+(-theta[1])*Bt*Bt)%saddle(xx,Bt)-0.75*(+2*(0.1)*Bt)*saddle(xx,Bt)-0.5*(+(0.1)*Bt*Bt)*
     saddle2(xx,Bt);
  return(val);
}

// [[Rcpp::export]]
mat solver(vec Xs,double Bt,vec theta,int N,double delt,int N2,vec tt)
{
    mat fx0(N2,4);
    mat fx1(N2,4);
    mat fx2(N2,4);
    mat fx3(N2,4);
    mat fx4(N2,4);
    mat fx5(N2,4);
    mat fx6(N2,4);
    mat fx7(N2,4);
    mat fx8(N2,4);
    mat fx9(N2,4);
    mat fx10(N2,4);
    mat fx11(N2,4);
    mat fx12(N2,4);
    mat fx13(N2,4);
    mat fx14(N2,4);
    mat fx15(N2,4);
    mat fx16(N2,4);
    mat x0(N2,4);
    mat x1(N2,4);
    mat x2(N2,4);
    mat x3(N2,4);
    mat x4(N2,4);
    mat x5(N2,4);
    mat x6(N2,4);
    mat x7(N2,4);
    mat x8(N2,4);
    mat x9(N2,4);
    mat x10(N2,4);
    mat x11(N2,4);
    mat x12(N2,4);
    mat x13(N2,4);
    mat x14(N2,4);
    mat x15(N2,4);
    mat x16(N2,4);

    x0.fill(0);
    x0.col(0)=Xs;

    mat res(N2,N);
```

```
    res.fill(0);
    vec d=tt;
    for (int i = 1; i < N; i++)
{
  fx0=f(x0,theta,d,N2);
  x1=x0+delt*(0.1*fx0);
  fx1=f(x1,theta,d+0.1000000000000000000000000000000000000000000000000000000000000000000000*delt,N2);
  x2=x0+delt*(-0.915176561375291*fx0+1.45453440217827*fx1);
  fx2=f(x2,theta,d+0.539357840802981787532485197881302436857273449701009015505500*delt,N2);
  x3=x0+delt*(0.202259190301118*fx0+0.606777570903354*fx2);
  fx3=f(x3,theta,d+0.809036761204472681298727796821953655285910174551513523258250*delt,N2);
  x4=x0+delt*(0.184024714708644*fx0+0.197966831227192*fx2-0.0729547847313633*fx3);
  fx4=f(x4,theta,d+0.309036761204472681298727796821953655285910174551513523258250*delt,N2);
  x5=x0+delt*(0.0879007340206681*fx0+0.410459702520261*fx3+0.482713753678866*fx4);
  fx5=f(x5,theta,d+0.981074190219795268254879548310562080489056746118724882027805*delt,N2);
  x6=x0+delt*(0.085970050490246*fx0+0.330885963040722*fx3+0.48966295730945*fx4-0.0731856375070851*
     fx5);
  fx6=f(x6,theta,d+0.833333333333333333333333333333333333333333333333333333333333333333333333*delt,N2);
  x7=x0+delt*(0.120930449125334*fx0+0.260124675758296*fx4+0.0325402621549091*fx5-0.0595780211817361
     *fx6);
  fx7=f(x7,theta,d+0.354017365856802376329264185948796742115824053807373968324184*delt,N2);
  x8=x0+delt*(0.110854379580391*fx0-0.0605761488255006*fx5+0.321763705601778*fx6+0.510485725608063*
     fx7);
  fx8=f(x8,theta,d+0.882527661964732346425501486979669075182867844268052119663791*delt,N2);
  x9=x0+delt*(0.112054414752879*fx0-0.144942775902866*fx5-0.333269719096257*fx6+0.49926922955688*
     fx7+0.509504608929686*fx8);
  fx9=f(x9,theta,d+0.642615758240322548157075497020439535959501736363212695909875*delt,N2);
  x10=x0+delt*(0.113976783964186*fx0-0.0768813364203357*fx5+0.239527360324391*fx6+0.397774662368095
     *fx7+0.0107558956873607*fx8-0.327769124164019*fx9);
  fx10=f(x10,theta,d+0.357384241759677451842924502979560464040498263636787304090125*delt,N2);
  x11=x0+delt*(0.0798314528280196*fx0-0.0520329686800603*fx5-0.0576954146168549*fx6
     +0.194781915712104*fx7+0.145384923188325*fx8-0.0782942710351671*fx9-0.114503299361099*fx10);
  fx11=f(x11,theta,d+0.117472338035267653574498513020330924817132155731947880336209*delt,N2);
  x12=x0+delt*(0.985115610164857*fx0+0.330885963040722*fx3+0.48966295730945*fx4-1.37896486574844*
     fx5-0.861164195027636*fx6+5.78428813637537*fx7+3.28807761985104*fx8-2.38633905093136*fx9
     -3.25479342483644*fx10-2.16343541686423*fx11);
  fx12=f(x12,theta,d+0.833333333333333333333333333333333333333333333333333333333333333333333333*delt,N2);
  x13=x0+delt*(0.895080295771633*fx0+0.197966831227192*fx2-0.0729547847313633*fx3-0.851236239662008
     *fx5+0.398320112318533*fx6+3.63937263181036*fx7+1.5482287703983*fx8-2.12221714704054*fx9
     -1.58350398545326*fx10-1.71561608285936*fx11-0.0244036405750127*fx12);
  fx13=f(x13,theta,d+0.309036761204472681298727796821953655285910174551513523258250*delt,N2);
  x14=x0+delt*(-0.915176561375291*fx0+1.45453440217827*fx1+0*fx2+0*fx3-0.777333643644968*fx4+0*fx5
     -0.0910895662155176*fx6+0.0910895662155176*fx12+0.777333643644968*fx13);
  fx14=f(x14,theta,d+0.539357840802981787532485197881302436857273449701009015505500*delt,N2);
  x15=x0+delt*(0.1*fx0-0.157178665799771*fx2+0.157178665799771*fx14);
  fx15=f(x15,theta,d+0.1000000000000000000000000000000000000000000000000000000000000000000000*delt,N2);
  x16=x0+delt*(0.181781300700095*fx0+0.675*fx1+0.34275815984719*fx2+0*fx3+0.259111214548323*fx4
     -0.358278966717952*fx5-1.04594895940883*fx6+0.930327845415627*fx7+1.77950959431708*fx8+0.1*fx9
     -0.282547569539044*fx10-0.159327350119973*fx11-0.145515894647002*fx12-0.259111214548323*fx13
     -0.34275815984719*fx14-0.675*fx15);
  fx16=f(x16,theta,d+delt,N2);

  x0=x0+(0.033333333333333333333333333333333333333333333333333333333333333333333333*fx0
        +0.025000000000000000000000000000000000000000000000000000000000000000000000*fx1
        +0.033333333333333333333333333333333333333333333333333333333333333333333333*fx2
        +0.000000000000000000000000000000000000000000000000000000000000000000000000*fx3
        +0.050000000000000000000000000000000000000000000000000000000000000000000000*fx4
        +0.000000000000000000000000000000000000000000000000000000000000000000000000*fx5
        +0.040000000000000000000000000000000000000000000000000000000000000000000000*fx6
        +0.000000000000000000000000000000000000000000000000000000000000000000000000*fx7
        +0.189237478148923490158306404106012326238162346948625830327194*fx8
        +0.277429188517743176508360262560654340428504319718040836339472*fx9
        +0.277429188517743176508360262560654340428504319718040836339472*fx10
        +0.189237478148923490158306404106012326238162346948625830327194*fx11
        -0.040000000000000000000000000000000000000000000000000000000000000000000000*fx12
        -0.050000000000000000000000000000000000000000000000000000000000000000000000*fx13
        -0.033333333333333333333333333333333333333333333333333333333333333333333333*fx14
        -0.025000000000000000000000000000000000000000000000000000000000000000000000*fx15
        +0.033333333333333333333333333333333333333333333333333333333333333333333333*fx16)*delt;
    d=d+delt;
    res.col(i) = pcurr(x0,Bt,theta,d);
}
return(res);
}
// [[Rcpp::export]]
vec ieq(mat res1,vec res2,int N,double delt)
{
    vec v(N);
    v(0)=0;
    double smtemp=0;
```

```
    for  (int i = 1; i < N; i++)
   {
     smtemp=0;
      for (int j = 1; j < i; j++)
     {
       smtemp=smtemp+v(j)*res1(j,i-j);
     }
     v(i) = (2.0*res2(i)-2.0*smtemp*delt);
   }
   return(v);
}
```

Appendix D

Non-Linear Multivariate Jump Diffusions with State-Dependent and/or Stochastic Intensity

D.1 Simplification of the PDDE for the moment generating function

In order to simplify the expression for the PDDE of the moment generating function we apply standard multivariate integration by parts to the terms of the Kolmogorov equation. For example, let S_B denote the boundary of the process $\mathbf{X}_t = \{X_t, Y_t\}'$ (where S denotes the state space of the process – see Section 4.2), then:

$$\begin{aligned} A_1(x,y) &= \iint_S g(x,y)\frac{\partial}{\partial x}\big[\mu(x,y,t)f(x,y)\big]dxdy \\ &= \oint_{S_B} g(x,y)\mu(x,y,t)f(x,y)\mathbf{n}dB_S - \iint_S \frac{\partial}{\partial x}\big[g(x,y)\big]\mu(x,y,t)f(x,y)dxdy \\ &= \oint_{S_B} g(x,y)\mu(x,y,t)f(x,y)\mathbf{n}dB_S - \alpha\iint_S e^{\alpha x+\beta y}\mu(x,y,t)f(x,y)dxdy \\ &= \oint_{S_B} g(x,y)\mu(x,y,t)f(x,y)\mathbf{n}dB_S - \alpha\iint_S \sum_{i=0}^{\infty}\sum_{j=0}^{i}\frac{\alpha^i\beta^{i-j}}{i!(i-j)!}x^i y^{i-j}\mu(x,y,t)f(x,y)dxdy \end{aligned} \tag{D.1.1}$$

where $\mathbf{n}$ is the normal vector perpendicular to S_B. Now, for example, let $\mu_1(x,y,t) = \sum_{k+r\leq 2} c_{k,r}(t)x^k y^r$ and denote the first term of Equation D.1.1

by $R^1_{S_B}$, then:

$$\begin{aligned}
A_1(x,y) &= R^1_{S_B} - \alpha \sum_{k+r\leq 2} c_{k,r}(t) \iint_S \sum_{i=0}^{\infty} \sum_{j=0}^{i} \frac{\alpha^i \beta^{i-j}}{i!(i-j)} x^{i+k} y^{i-j+r} f(x,y) dxdy \\
&= R^1_{S_B} - \alpha \sum_{k+r\leq 2} c_{k,r}(t) \iint_S \sum_{m=k}^{\infty} \sum_{n=m-r}^{m-k} \frac{\alpha^{m-k} \beta^{m-n-r}}{(m-k)!(m-n-r)!} x^m y^{m-n} f(x,y) dxdy \\
&= R^1_{S_B} - \alpha \sum_{k+r\leq 2} c_{k,r}(t) \frac{\partial^{k+r}}{\partial \alpha^k \partial \beta^r} \iint_S \sum_{m=0}^{\infty} \sum_{n=0}^{m} \frac{\alpha^m \beta^{m-n}}{m!(m-n)!} x^m y^{m-n} f(x,y) dxdy \\
&= R^1_{S_B} - \alpha \mu_1 \left(\frac{\partial}{\partial \alpha}, \frac{\partial}{\partial \beta}, t \right) M(\alpha, \beta, t)
\end{aligned} \tag{D.1.2}$$

By applying this technique to the remaining $A_i(\alpha,\beta,t)$ and $B_{ij}(\alpha,\beta,t)$ we arrive at a revised PDDE for the moment generating function. Naturally, in order to arrive at the result of Equation 4.3.10, we require that the residual terms such as $R^i_{S_B}$ vanish. This is achieved by making some precluding assumptions on the nature of the underlying process. For purposes of the present paper, we impose the condition that the process either cannot reach S_B within the applicable time horizon, or that terms adjacent to the transitional density (e.g., $\mu(x,y,t)g(x,y)$) approach zero faster than $f(x,y)$ approaches infinity – in which case the kernel of the residual terms vanish.

D.2 Derivation of series expressions for $C_i(\alpha,\beta,t)$ i.t.o. $M^\star_i(\alpha,\beta,t)$

Let $g(x,y) = \exp(\alpha x + \beta y)$ then

$$\begin{aligned}
C_1(\alpha,\beta,t) &= \iiint\!\!\int g(x,y)\lambda_1(\nabla_{\epsilon_{11}}(x), \nabla_{\epsilon_{21}}(y), \dot{r}_1, t) f(\nabla_{\epsilon_{11}}(x), \nabla_{\epsilon_{21}}(y)) |\delta_1| d\pi_1(\dot{r}_1) d\phi_1(\dot{z}_{11}, \dot{z}_{21}) dxdy \\
&\quad - \iiint g(x,y)\lambda_1(x,y,\dot{r}_1,t) f(x,y) d\pi_1(\dot{r}_1) dxdy \\
&= E_{\dot{r}_1}\left[\iiint g(x,y)\lambda_1(\nabla_{\epsilon_{11}}(x), \nabla_{\epsilon_{21}}(y), \dot{r}_1, t) f(\nabla_{\epsilon_{11}}(x), \nabla_{\epsilon_{21}}(y)) d\phi_1(\dot{z}_{11}, \dot{z}_{21}) |\delta_1| dxdy \right] \\
&\quad - E_{\dot{r}_1}\left[\iint g(x,y)\lambda_1(x,y,\dot{r}_1,t) f(x,y) dxdy \right] \\
&= E_{\dot{r}_1}\Big[I_1(\alpha,\beta,t) - I_2(\alpha,\beta,t) \Big].
\end{aligned} \tag{D.2.1}$$

First, consider $I_2(\alpha, \beta, t)$. Using the Cauchy product:

$$\begin{aligned} g(x,y) &= e^{\alpha x} e^{\beta y} \\ &= \left(\sum_{i=1}^{\infty} \frac{\alpha^i x^i}{i!} \right) \times \left(\sum_{j=1}^{\infty} \frac{\beta^j y^j}{j!} \right) \\ &= \sum_{i=0}^{\infty} \sum_{j=0}^{i} \frac{\alpha^i \beta^{i-j}}{i!(i-j)!} x^i y^{i-j}, \end{aligned} \tag{D.2.2}$$

we arrive at

$$I_2(\alpha, \beta, t) = \iint \sum_{i=0}^{\infty} \sum_{j=0}^{i} \frac{\alpha^i \beta^{i-j}}{i!(i-j)!} x^i y^{i-j} \lambda_1(x, y, \dot{r}_1, t) f(x, y) dx dy. \tag{D.2.3}$$

Observing that if $\lambda_1(x, y, \dot{r}_1, t)$ is polynomial in x and y, then

$$\begin{aligned} I_2(\alpha, \beta, t) &= \iint \Big(\sum_{p+q \leq 2} h^1_{pq}(\dot{r}_1, t) x^p y^q \Big) \times \Big(\sum_{i=0}^{\infty} \sum_{j=0}^{i} \frac{\alpha^i \beta^{i-j}}{i!(i-j)!} x^i y^{i-j} \Big) f(x, y) dx dy \\ &= \iint \Big(\sum_{p+q \leq 2} \sum_{i=0}^{\infty} \sum_{j=0}^{i} \frac{\alpha^i \beta^{i-j}}{i!(i-j)!} h^1_{pq}(\dot{r}_1, t) x^{i+p} y^{i-j+q} \Big) f(x, y) dx dy \\ &= \iint \Big(\sum_{p+q \leq 2} \sum_{i=0}^{\infty} \sum_{j=0}^{i} h^1_{pq}(\dot{r}_1, t) \frac{\partial^p}{\partial \alpha^p} \frac{\partial^p}{\partial \beta^q} \frac{\alpha^{i+p} \beta^{i-j-q}}{(i+p)!(i-j+q)!} x^{i+p} y^{i-j+q} \Big) f(x, y) dx dy \\ &= \iint \Big(\sum_{p+q \leq 2} \sum_{i=0}^{\infty} \sum_{j=0}^{i} h^1_{pq}(\dot{r}_1, t) \frac{\partial^p}{\partial \alpha^p} \frac{\partial^q}{\partial \beta^q} \frac{\alpha^{i+p} \beta^{i-j-q}}{(i+p)!(i-j+q)!} x^{i+p} y^{i-j+q} \Big) f(x, y) dx dy \\ &= \lambda_1 \left(\frac{\partial}{\partial \alpha}, \frac{\partial}{\partial \beta}, \dot{r}_1, t \right) \int \sum_{i=0}^{\infty} \sum_{j=0}^{i} \frac{\alpha^i \beta^{i-j}}{i!(i-j)!} x^i y^{i-j} f(x, y) dx dy \\ &= \lambda_1 \left(\frac{\partial}{\partial \alpha}, \frac{\partial}{\partial \beta}, \dot{r}_1, t \right) M(\alpha, \beta, t). \end{aligned} \tag{D.2.4}$$

We can apply a similar strategy to $I_1(\alpha, \beta)$. However, in this case we need to account for the presence of the jump variables. Applying the transformation:

$$\begin{aligned} u &= \nu_{11}(x + \epsilon_{11}(x, y, \dot{z}_{11}, t)) \\ v &= \nu_{21}(y + \epsilon_{21}(x, y, \dot{z}_{21}, t)) \end{aligned} \tag{D.2.5}$$

we arrive at

$$E_{\dot{r}_1}\big[I_1(\alpha,\beta,t)\big] = \iiiint g(x,y)\lambda_1(u,v,\dot{r}_1,t)f(u,v)|\delta_1||\delta_1|^{-1}d\pi_1(\dot{r}_1)d\phi_1(\dot{z}_{11},\dot{z}_{21})dxdy, \tag{D.2.6}$$

but since $\nu_{11}(.)$ and $\nu_{21}(.)$ inverts the jump action, the result of the aforementioned transformation on x and y is just that of the jump mechanism:

$$\begin{aligned} x &= u + \epsilon_{11}(u,v,\dot{z}_{11},t) \\ y &= v + \epsilon_{21}(u,v,\dot{z}_{21},t). \end{aligned} \tag{D.2.7}$$

Consequently,

$$\begin{aligned} &E_{\dot{r}_1}\big[I_1(\alpha,\beta,t)\big] = \\ &\iiiint g(u+\epsilon_{11}(u,v,\dot{z}_{11},t), v+\epsilon_{21}(u,v,\dot{z}_{21},t))\lambda_1(u,v,\dot{r}_1,t)f(u,v)d\pi_1(\dot{r}_1)d\phi_1(\dot{z}_{11},\dot{z}_{21})dudv \end{aligned} \tag{D.2.8}$$

Once again, by writing $g(x,y)$ as an infinite series

$$\begin{aligned} &g(u+\epsilon_{11}(u,v,\dot{z}_{11},t), v+\epsilon_{21}(u,v,\dot{z}_{21},t)) \\ &= \sum_{i=0}^{\infty}\sum_{j=0}^{i}\frac{\alpha^i\beta^{i-j}}{i!(i-j)!}(u+\epsilon_{11}(u,v,\dot{z}_{11},t))^i(v+\epsilon_{21}(u,v,\dot{z}_{21},t))^{i-j}, \end{aligned} \tag{D.2.9}$$

and applying the binomial theorem to the terms in brackets, we have:

$$\begin{aligned} &g(u+\epsilon_{11}(u,v,\dot{z}_{11},t), v+\epsilon_{21}(u,v,\dot{z}_{21},t)) \\ &= \sum_{i=0}^{\infty}\sum_{j=0}^{i}\frac{\alpha^i\beta^{i-j}}{i!(i-j)!}\Bigg(\sum_{r=0}^{i}\binom{i}{r}u^r\epsilon_{11}(u,v,\dot{z}_{11},t)^{i-r}\Bigg)\Bigg(\sum_{s=0}^{i-j}\binom{i-j}{s}v^s\epsilon_{21}(u,v,\dot{z}_{21},t)^{i-j-s}\Bigg). \end{aligned} \tag{D.2.10}$$

Now, applying the same principle as before by assuming that $\lambda_1(x,y,\dot{r}_1,t)$ is polynomial in x and y, applying the same steps as in Equation D.2.4 yields:

$$\begin{aligned} &I_1(\alpha,\beta,t) = \\ &\lambda_1\Big(\frac{\partial}{\partial\alpha},\frac{\partial}{\partial\beta},\dot{r}_1,t\Big)\iint\Big(\sum_{i=0}^{\infty}\sum_{j=0}^{i}\frac{\alpha^i\beta^{i-j}}{i!(i-j)!}\Big(\sum_{r=0}^{i}\binom{i}{r}u^r\epsilon_{11}(u,v,\dot{z}_{11},t)^{i-r}\Big)\Big(\sum_{s=0}^{i-j}\binom{i-j}{s}v^s\epsilon_{21}(u,v,\dot{z}_{21},t)^{i-j-s}\Big)\Big)f(u,v)d\phi_1(\dot{z}_{11},\dot{z}_{21})du \\ &= \lambda_1\Big(\frac{\partial}{\partial\alpha},\frac{\partial}{\partial\beta},\dot{r}_1,t\Big)M_1^{\star}(\alpha,\beta,t) \end{aligned} \tag{D.2.11}$$

Subtracting $I_2(\alpha,\beta,t)$ from $I_1(\alpha,\beta,t)$ and taking expectation with respect to the variable $\dot{r}_1$, we have:

$$\begin{aligned} C_1(\alpha,\beta,t) &= E_{\dot{r}_1}\Big[I_1(\alpha,\beta,t) - I_2(\alpha,\beta,t)\Big] \\ &= E_{\dot{r}_1}\left[\lambda_1\left(\frac{\partial}{\partial\alpha},\frac{\partial}{\partial\beta},\dot{r}_1,t\right)\times\Big(M_1^{\star}(\alpha,\beta,t) - M(\alpha,\beta,t)\Big)\right] \end{aligned} \tag{D.2.12}$$

where $M_1^{\star}(\alpha,\beta,t)$ can be written more conveniently as:

$$M_1^{\star}(\alpha,\beta,t) = \sum_{i=0}^{\infty}\sum_{j=0}^{i}\frac{\alpha^i\beta^{i-j}}{i!(i-j)!}E_{X,Y,\dot{z}}\left[\Big(\sum_{r=0}^{i}\binom{i}{r}X_t^r\epsilon_{11}(X_t,Y_t,\dot{z}_{11},t)^{j-r}\Big)\Big(\sum_{s=0}^{i-j}\binom{i-j}{s}Y_t^s\epsilon_{21}(X_t,Y_t,\dot{z}_{21},t)^{i-j-s}\Big)\right]. \tag{D.2.13}$$

Repeating this process for $C_2(\alpha,\beta,t)$ yields a similar expression:

$$C_2(\alpha,\beta,t) = E_{\dot{r}_2}\left[\lambda_2\left(\frac{\partial}{\partial\alpha},\frac{\partial}{\partial\beta},\dot{r}_2,t\right)\times\Big(M_2^{\star}(\alpha,\beta,t) - M(\alpha,\beta,t)\Big)\right] \tag{D.2.14}$$

where

$$M_2^{\star}(\alpha,\beta,t) = \sum_{i=0}^{\infty}\sum_{j=0}^{i}\frac{\alpha^i\beta^{i-j}}{i!(i-j)!}E_{X,Y,\dot{z}}\left[\Big(\sum_{r=0}^{i}\binom{i}{r}X_t^r\epsilon_{21}(X_t,Y_t,\dot{z}_{21},t)^{j-r}\Big)\Big(\sum_{s=0}^{i-j}\binom{i-j}{s}Y_t^s\epsilon_{22}(X_t,Y_t,\dot{z}_{22},t)^{i-j-s}\Big)\right]. \tag{D.2.15}$$

By combining these two results we may derive an expression for $M_k^{\star}(\alpha,\beta,t)$ as in Equation 4.3.14.

D.3 Moment and cumulant relations

Harvey (1972) gives a recursive method for relating bivariate moments and cumulants. The strategy revolves around operators which can be used to carry out moment-cumulant arithmetic for arbitrary order moments by starting from first order moments and working up to higher order moment relations. Let $\kappa_{ij}(t)$ denote the ij-th cumulant of the process at time t, and $\{D_k(.), k = 1,2\}$ be operators with the properties:

$$\begin{aligned} &D_k(C) = 0 \text{ for some constant } C, \\ &D_k(\kappa_{ij} + \kappa_{nm}) = D_k(\kappa_{ij}) + D_k(\kappa_{nm}), \\ &D_k(\kappa_{ij}^n) = n\kappa_{ij}^{n-1}D_k(\kappa_{ij}). \end{aligned} \tag{D.3.1}$$

Then define the respective operations to be:

$$\begin{aligned} D_1(\kappa_{ij}) &= \kappa_{i+1j} \\ D_2(\kappa_{ij}) &= \kappa_{ij+1}. \end{aligned} \tag{D.3.2}$$

Using these operators the moments and cumulants of a bivariate process can be related by iterating through the equations:

$$\begin{aligned} m_{i+1j} &= D_1(m_{ij}) + \kappa_{10} m_{ij} \\ m_{ij+1} &= D_2(m_{ij}) + \kappa_{01} m_{ij}. \end{aligned} \tag{D.3.3}$$

subject to the initial conditions $m_{10} = \kappa_{10}$ and $m_{01} = \kappa_{01}$.

D.4 Simulating jump diffusion processes

In order to verify that the methodology developed here does indeed produce valid approximations, we need to generate benchmark values for the various elements of the approximation scheme. Since we typically focus on non-linear models with analytically intractable dynamics, the best way to achieve this is via simulation. That is, by evaluating Equation 4.2.4 numerically, we can generate a simulated trajectory for the model process on the desired transition horizon. By doing this repeatedly, we may use the simulated trajectories in order to calculate moment-statistics and frequency distributions at fixed points in time. Although a great deal of literature is dedicated to the development of simulation schemes for pure diffusion processes, the literature on simulations schemes dealing with jump diffusions specifically is relatively sparse. A handful of simulation schemes have been proposed by authors such as Platen (1982) (see also Xia and Giles (2012)) which advocate simulating Poisson jump realisations and subsequently augmenting a standard fixed step size scheme such as a Euler-Murayama or Milstein schemes. This is achieved by adding the jump process trajectory to the diffusion trajectory – accounting of course for the effect of the jump process being embedded in the jump diffusion trajectory. A more rigorous treatment can be found in Higham and Kloeden (2005), where the authors derive both implicit and explicit schemes that satisfy various stability criterion. In some cases, it is even possible to simulate exact trajectories of a jump diffusion (Broadie and Kaya, 2006). For our purposes, we adopt a strategy by which we modify the Euler-Murayama scheme to reflect the presence of the jump mechanism. This results in a scheme which is similar in structure to that of Platen (1982).

Let $\mathbf{X}_t$ denote a vector of process coordinates, τ a scalar time index on the transition horizon $[s, t]$ and $\Delta = 1/M$ the time step for some positive integer M.

Then, the iterative updating scheme for simulating a single trajectory of a jump diffusion process is given by:

(1) Initialize the vector $\mathbf{X}_\tau = \mathbf{X}_s$ and set $\tau = s$.

(2) (a) For $i = 1, 2, \ldots k$, set:

$$X^{(i)}_{\tau+\Delta} = X^{(i)}_\tau + \mu_i(\mathbf{X}_\tau, \tau)\Delta + \sum_{j=1}^{k} \sigma_{ij}(\mathbf{X}_\tau, \tau) Z^{(j)}, \tag{D.4.1}$$

where $Z^{(j)} \sim \mathrm{N}(0, \Delta)$ for $j = 1, 2, \ldots k$ and the same sequence of $Z^{(j)}$ are used for each i.

(b) For each $j = 1, 2, \ldots q$, draw $\dot{r}^{(j)}_\tau \sim \pi_j(\tau)$ and if

$$1 - \exp(\lambda_j(\mathbf{X}_\tau, \dot{r}^{(j)}_\tau, \tau)\Delta) > u \tag{D.4.2}$$

where $u \sim U(0, 1)$, draw $\dot{\mathbf{z}}^{(\cdot j)}_\tau \sim \phi_j(t)$ and set

$$\mathbf{X}_{\tau+\Delta} = \mathbf{X}_{\tau+\Delta} + \mathbf{J}(\mathbf{X}_\tau, \dot{\mathbf{z}}_\tau)^{(\cdot j)}. \tag{D.4.3}$$

(3) Set $\tau = \tau + \Delta$ and go to step (2).

By evaluating this algorithm repeatedly, it is possible to generate a large number of simulated trajectories which can them be used to calculate moment statistics, frequency distributions and other statistics for highly non-linear models. Subsequently, these statistics can then be used to verify elements such as the moment equations and/or transition density approximations on various time scales.

D.5 Surrogate Densities

Using the methodology of Section 4.3 we can derive the moment equations of a non-linear jump diffusion process. From the moment equations it is possible to approximate the transitional density by plugging the moment equations into as suitable surrogate density. Under the mixture factorization derived in Section 4.3.4, we can approximate the transitional density to a high degree of accuracy. In the scalar case a plethora of surrogate densities exist that can be used to carry the moments into a usable density approximation. In the bivariate case, we are limited to only a few candidates. Typically for these purposes we make

use of the bivariate saddlepoint approximation. Reiterating from Section 2.3.3, let

$$K^{(d)}(\alpha,\beta,t)=\sum_{i+j\leq d}\frac{\alpha^i\beta^j}{i!j!}\kappa_{ij}(t) \tag{D.5.1}$$

denote the d-th order truncated cumulant generating function, then the bivariate saddlepoint approximation (Renshaw, 2000) is given by the expression:

$$\tilde{f}^{(d)}_{SPT}(x_t,y_t|X_s,Y_s)=\frac{\exp\big(K^{(d)}(\alpha_0,\beta_0)-\alpha_0x_t-\beta_0y_t\big)}{2\pi\sqrt{\frac{\partial^2K^{(d)}}{\partial\alpha^2}\frac{\partial^2K^{(d)}}{\partial\beta^2}-\big(\frac{\partial K^{(d)}}{\partial\alpha\partial\beta}\big)^2}}, \tag{D.5.2}$$

for

$$\begin{aligned}K^{(d)}(\alpha,\beta)=&\alpha\kappa_{10}(t)+\frac{\alpha^2}{2}\kappa_{20}(t)+\frac{\alpha^3}{6}\kappa_{30}(t)+\frac{\alpha^4}{24}\kappa_{40}(t)+\ldots+\frac{\alpha^d}{d!}\kappa_{d0}(t)\\&+\beta\kappa_{01}(t)+\frac{\beta^2}{2}\kappa_{02}(t)+\frac{\beta^3}{6}\kappa_{03}(t)+\frac{\beta^4}{24}\kappa_{04}(t)+\ldots+\frac{\beta^d}{d!}\kappa_{0d}(t)\\&+\alpha\beta\kappa_{11}(t)+\frac{\alpha^2\beta}{2}\kappa_{21}(t)+\frac{\alpha\beta^2}{2}\kappa_{12}(t)+\ldots+\frac{\alpha^{d/2}\beta^{d/2}}{(d/2)!)^2}\kappa_{((d/2)(d/2)}(t)+\ldots\\&+\frac{\alpha^{d-1}\beta}{(d-1)!}\kappa_{(d-1)1}(t)+\frac{\alpha\beta^{d-1}}{(d-1)!}\kappa_{1(d-1)}(t),\end{aligned} \tag{D.5.3}$$

where α_0 and β_0 solves the system:

$$\begin{aligned}\frac{\partial K^{(d)}}{\partial\alpha}(\alpha,\beta)&=x_t,\\\frac{\partial K^{(d)}}{\partial\beta}(\alpha,\beta)&=y_t.\end{aligned} \tag{D.5.4}$$

Clearly, since the saddlepoint approximation requires the evaluation of the cumulants, we need to relate the respective moments to their cumulant counterparts. This can be achieved using the expressions in Appendix D.3. For purposes of the jump diffusion examples presented in this thesis, we typically use a fourth-order truncation in order to approximate a given bivariate density, whether it be the transitional density itself or the jump-free transitional density and excess distribution under the mixture factorization. Thus, we need to translate the fourth-order truncated moments to their cumulant counterparts. Using the equations outlined

in Appendix D.3, we make the relations explicit:

$$
\begin{aligned}
\kappa_{10} &= m_{10} \\
\kappa_{20} &= m_{20} - m_{10}^2 \\
\kappa_{30} &= m_{30} - 3m_{20}m_{10} + 2m_{10}^3 \\
\kappa_{40} &= m_{40} - 4m_{30}m_{10} - 3m_{20}^2 + 12m_{20}m_{10}^2 - 6m_{10}^4 \\
\kappa_{01} &= m_{01} \\
\kappa_{02} &= m_{02} - m_{01}^2 \\
\kappa_{03} &= m_{03} - 3m_{02}m_{01} + 2m_{01}^3 \\
\kappa_{04} &= m_{04} - 4m_{03}m_{01} - 3m_{02}^2 + 12m_{02}m_{01}^2 - 6m_{01}^4 \\
\kappa_{11} &= m_{11} - m_{10}m_{01} \\
\kappa_{21} &= m_{21} - 2m_{11}m_{10} - m_{20}m_{01} + 2m_{10}^2 m_{01} \\
\kappa_{12} &= m_{12} - 2m_{11}m_{01} - m_{02}m_{10} + 2m_{01}^2 m_{10} \\
\kappa_{22} &= m_{22} - 2m_{21}m_{01} - 2m_{12}m_{10} - m_{20}m_{02} - 2m_{11}^2 + 8m_{11}m_{01}m_{10} + 2m_{02}m_{10}^2 \\
&\quad + 2m_{20}m_{01}^2 - 6m_{10}^2 m_{01}^2 \\
\kappa_{31} &= m_{31} - 3m_{21}m_{10} - m_{30}m_{01} - 3m_{20}m_{11} + 6m_{11}m_{10}^2 + 6m_{20}m_{10}m_{01} - 6m_{10}^3 m_{01} \\
\kappa_{13} &= m_{13} - 3m_{12}m_{01} - m_{03}m_{10} - 3m_{02}m_{11} + 6m_{11}m_{01}^2 + 6m_{02}m_{01}m_{10} - 6m_{01}^3 m_{10}.
\end{aligned}
\tag{D.5.5}
$$

D.6 Moment equations of a CIR jump process with state-dependent jump intensity

Under the dynamics of Equation 4.3.76, the moment equations of the process under a sixth-order truncation can be verified as:

$$
\begin{aligned}
m_1'(t) &= (2 \times 5) - 2m_1(t) + j_1(t) \\
m_2'(t) &= 2[(2 \times 5)m_1(t) - 2m_2(t)] + \theta_4 m_1(t) + j_2(t) \\
m_3'(t) &= 3[(2 \times 5)m_2(t) - 2m_3(t)] + \theta_4 3m_2(t) + j_3(t) \\
m_4'(t) &= 4[(2 \times 5)m_3(t) - 2m_4(t)] + \theta_4 6m_3(t) + j_4(t) \\
m_5'(t) &= 5[(2 \times 5)m_4(t) - 2m_5(t)] + \theta_4 10m_4(t) + j_5(t) \\
m_6'(t) &= 6[(2 \times 5)m_5(t) - 2m_6(t)] + \theta_4 15m_5(t) + j_6(t)
\end{aligned}
\tag{D.6.1}
$$

with $m_i(0) = 4^i$ for $i = 1, 2, \ldots, 6$ where

$$\begin{aligned}
j_1(t) &= 0.2(\rho_1 m_1(t)) \\
j_2(t) &= 0.2(2\rho_1 m_2(t) + \rho_2 m_1(t)) \\
j_3(t) &= 0.2(3\rho_1 m_3(t) + 3\rho_2 m_2(t) + \rho_3 m_1(t)) \\
j_4(t) &= 0.2(4\rho_1 m_4(t) + 6\rho_2 m_3(t) + 4\rho_3 m_2(t) + \rho_4 m_1(t)) \\
j_5(t) &= 0.2(5\rho_1 m_5(t) + 10\rho_2 m_4(t) + 10\rho_3 m_3(t) + 5\rho_4 m_2(t) + \rho_5 m_1(t)) \\
j_6(t) &= 0.2(6\rho_1 m_6(t) + 15\rho_2 m_5(t) + 20\rho_3 m_4(t) + 15\rho_4 m_3(t) + 6\rho_5 m_2(t) + \rho_6 m_1(t)),
\end{aligned} \tag{D.6.2}$$

for $\lambda(X_t, t) = 0.2X_t$, and

$$\begin{aligned}
j_1(t) &= 0.5(1 + \sin(3\pi t))(\rho_1 m_1(t)) + 0.1(1 + \cos(3\pi t))(\rho_1 m_2(t)) \\
j_2(t) &= 0.5(1 + \sin(3\pi t))(2\rho_1 m_2(t) + \rho_2 m_1(t)) + 0.1(1 + \cos(3\pi t))(2\rho_1 m_3(t) + \rho_2 m_2(t)) \\
j_3(t) &= 0.5(1 + \sin(3\pi t))(3\rho_1 m_3(t) + 3\rho_2 m_2(t) + \rho_3 m_1(t)) \\
&\quad + 0.1(1 + \cos(3\pi t))(3\rho_1 m_4(t) + 3\rho_2 m_3(t) + \rho_3 m_2(t)) \\
j_4(t) &= 0.5(1 + \sin(3\pi t))(4\rho_1 m_4(t) + 6\rho_2 m_3(t) + 4\rho_3 m_2(t) + \rho_4 m_1(t)) \\
&\quad + 0.1(1 + \cos(3\pi t))(4\rho_1 m_5(t) + 6\rho_2 m_4(t) + 4\rho_3 m_3(t) + \rho_4 m_2(t)) \\
j_5(t) &= 0.5(1 + \sin(3\pi t))(5\rho_1 m_5(t) + 10\rho_2 m_4(t) + 10\rho_3 m_3(t) + 5\rho_4 m_2(t) + \rho_5 m_1(t)) \\
&\quad + 0.1(1 + \cos(3\pi t))(5\rho_1 m_6(t) + 10\rho_2 m_5(t) + 10\rho_3 m_4(t) + 5\rho_4 m_3(t) + \rho_5 m_2(t)) \\
j_6(t) &= 0.5(1 + \sin(3\pi t))(6\rho_1 m_6(t) + 15\rho_2 m_5(t) + 20\rho_3 m_4(t) + 15\rho_4 m_3(t) + 6\rho_5 m_2(t) + \rho_6 m_1(t)) \\
&\quad + 0.1(1 + \cos(3\pi t))(6\rho_1 \tilde{m}_7(t) + 15\rho_2 m_6(t) + 20\rho_3 m_5(t) + 15\rho_4 m_6(t) + 6\rho_5 m_3(t) + \rho_6 m_2(t)),
\end{aligned} \tag{D.6.3}$$

for $\lambda(X_t, t) = 0.5(1 + \sin(3\pi t))X_t + 0.1(1 + \cos(3\pi t))X_t^2$. Here ρ_i denotes the i-th non-central moment of the jump distribution. That is, for $\dot{z} \sim \mathrm{N}(\mu, \sigma^2)$:

$$\begin{aligned}
\rho_1 &= \mu \\
\rho_2 &= \mu^2 + \sigma^2 \\
\rho_3 &= \mu^3 + 3\mu^1\sigma^2 \\
\rho_4 &= \mu^4 + 6\mu^2\sigma^2 + 3\sigma^4 \\
\rho_5 &= \mu^5 + 10\mu^3\sigma^2 + 15\mu^1\sigma^4 \\
\rho_6 &= \mu^6 + 15\mu^4\sigma^2 + 45\mu^2\sigma^4 + 15\sigma^6.
\end{aligned} \tag{D.6.4}$$

D.7 Moment equations of a bivariate CIR jump process with time-inhomogeneous jump variables

Using equations 4.3.35 and the expressions in Table 4.3.1, we can derive the $d = 4$-th order truncated moment equations for Equation 4.3.79: Let $m'_{ij} \equiv \frac{\partial}{\partial t} m_{i,j}(t)$

and $\theta = \{0.5, 5, -0.1, 0.4, 0.4, 6, -0.1, 0.3, 1, 0.75, 0.75, 0.75, 0.75\}$, then:

$$
\begin{aligned}
m'_{10} =& \theta_1\theta_2(1m_{00}) - \theta_1(1m_{10}) + \theta_3(1m_{01}) + \theta_9(+1\rho_{10}m_{00}) \\
m'_{20} =& \theta_1\theta_2(2m_{10}) - \theta_1(2m_{20}) + \theta_3(2m_{11}) + \theta_4^2(1m_{10}) + \theta_9(+2\rho_{10}m_{10} + 1\rho_{20}) \\
m'_{30} =& \theta_1\theta_2(3m_{20}) - \theta_1(3m_{30}) + \theta_3(3m_{21}) + \theta_4^2(3m_{20}) + \theta_9(+3\rho_{10}m_{20} + 3\rho_{20}m_{10} + 1\rho_{30}) \\
m'_{40} =& \theta_1\theta_2(4m_{30}) - \theta_1(4m_{40}) + \theta_3(4m_{31}) + \theta_4^2(6m_{30}) + \theta_9(+4\rho_{10}m_{30} + 6\rho_{20}m_{20} + 4\rho_{30}m_{10} + 1\rho_{40}) \\
m'_{01} =& \theta_5\theta_6(1m00) + \theta_7(1m_{10}) - \theta_5(1m_{01}) + \theta_9(+1\rho_{01}m00) \\
m'_{02} =& \theta_5\theta_6(2m_{01}) + \theta_7(2m_{11}) - \theta_5(2m_{02}) + \theta_8^2(1m_{01}) + \theta_9(+2\rho_{01}m_{01} + 1\rho_{02}) \\
m'_{03} =& \theta_5\theta_6(3m_{02}) + \theta_7(3m_{12}) - \theta_5(3m_{03}) + \theta_8^2(3m_{02}) + \theta_9(+3\rho_{01}m_{02} + 3\rho_{02}m_{01} + 1\rho_{03}) \\
m'_{04} =& \theta_5\theta_6(4m_{03}) + \theta_7(4m_{13}) - \theta_5(4m_{04}) + \theta_8^2(6m_{03}) + \theta_9(+4\rho_{01}m_{03} + 6\rho_{02}m_{02} + 4\rho_{03}m_{01} + 1\rho_{04}) \\
m'_{11} =& \theta_1\theta_2(1m_{01}) - \theta_1(1m_{11}) + \theta_3(1m_{02}) + \theta_5\theta_6(1m_{10}) + \theta_7(1m_{20}) - \theta_5(1m_{11}) \\
&+ \theta_9(\rho_{10}m_{01} + \rho_{01}m_{10} + \rho_{11}) \\
m'_{12} =& \theta_1\theta_2(1m_{02}) - \theta_1(1m_{12}) + \theta_3(1m_{03}) + \theta_5\theta_6(2m_{11}) + \theta_7(2m_{21}) - \theta_5(2m_{12}) + \theta_8^2(1m_{11}) \\
&+ \theta_9(\rho_{10}m_{02} + 2\rho_{01}m_{11} + 2\rho_{11}m_{01} + \rho_{02}m_{10} + \rho_{12}) \\
m'_{21} =& \theta_1\theta_2(2m_{11}) - \theta_1(2m_{21}) + \theta_3(2m_{12}) + \theta_5\theta_6(1m_{20}) + \theta_7(1m_{30}) - \theta_5(1m_{21}) + \theta_4^2(1m_{11}) \\
&+ \theta_9(2\rho_{10}m_{11} + \rho_{20}m_{01} + \rho_{01}m_{20} + 2\rho_{11}m_{10} + \rho_{21}) \\
m'_{22} =& \theta_1\theta_2(2m_{12}) - \theta_1(2m_{22}) + \theta_3(2m_{13}) + \theta_5\theta_6(2m_{21}) + \theta_7(2m_{31}) - \theta_5(2m_{22}) + \theta_4^2(1m_{12}) + \theta_8^2(1m_{21}) \\
&+ \theta_9(2\rho_{10}m_{12} + \rho_{20}m_{02} + 2\rho_{01}m_{21} + 4\rho_{11}m_{11} + 2\rho_{21}m_{01} + \rho_{02}m_{20} + 2\rho_{12}m_{10} + \rho_{22}) \\
m'_{13} =& \theta_1\theta_2(1m_{03}) - \theta_1(1m_{13}) + \theta_3(1m_{04}) + \theta_5\theta_6(3m_{12}) + \theta_7(3m_{22}) - \theta_5(3m_{13}) + \theta_8^2(3m_{12}) \\
&+ \theta_9(\rho_{10}m_{03} + 3\rho_{01}m_{12} + 3\rho_{11}m_{02} + 3\rho_{02}m_{11} + 3\rho_{12}m_{01} + \rho_{03}m_{10} + \rho_{13}) \\
m'_{31} =& \theta_1\theta_2(3m_{21}) - \theta_1(3m_{31}) + \theta_3(3m_{22}) + \theta_5\theta_6(1m_{30}) + \theta_7(1m_{40}) - \theta_5(1m_{31}) + \theta_4^2(3m_{21}) \\
&+ \theta_9(3\rho_{10}m_{21} + 3\rho_{20}m_{11} + \rho_{30}m_{01} + \rho_{01}m_{30} + 3\rho_{11}m_{20} + 3\rho_{21}m_{10} + \rho_{31})
\end{aligned}
\tag{D.7.1}
$$

where

$$
\begin{aligned}
\rho_{10} &= \theta_{10}(1 + \sin(2\pi t)) \\
\rho_{20} &= \theta_{10}^2(1 + \sin(2\pi t))^2 + (\theta_{12}(1 + 0.8\sin(2\pi t)))^2 \\
\rho_{30} &= \theta_{10}^3(1 + \sin(2\pi t))^3 + 3\theta_{10}(1 + \sin(2\pi t))(\theta_{12}(1 + 0.8\sin(2\pi t)))^2 \\
\rho_{40} &= \theta_{10}^4(1 + \sin(2\pi t))^4 + 6\theta_{10}^2(1 + \sin(2\pi t))^2(\theta_{12}(1 + 0.8\sin(2\pi t)))^2 + 3(\theta_{12}(1 + 0.8\sin(2\pi t)))^4 \\
\rho_{01} &= \theta_{11}(1 + \sin(2\pi t)) \\
\rho_{02} &= \theta_{11}^2(1 + \sin(2\pi t))^2 + (\theta_{13}(1 + 0.8\sin(2\pi t)))^2 \\
\rho_{03} &= \theta_{11}^3(1 + \sin(2\pi t))^3 + 3\theta_{11}(1 + \sin(2\pi t))(\theta_{13}(1 + 0.8\sin(2\pi t)))^2 \\
\rho_{04} &= \theta_{11}^4(1 + \sin(2\pi t))^4 + 6\theta_{11}^2(1 + \sin(2\pi t))^2(\theta_{13}(1 + 0.8\sin(2\pi t)))^2 + 3(\theta_{13}(1 + 0.8\sin(2\pi t)))^4
\end{aligned}
\tag{D.7.2}
$$

and $\rho_{ij} = 0$ otherwise. Although we drop the time dependence in order to fit the above equations within the confines of the present margin it is important to note that each m_{ij} and ρ_{ij} is in fact a function of time.

D.8 Simulating Wright-Fisher processes

In order to validate the jump diffusion approximation to the Wright-Fisher process with shocks, we need to simulate the discrete model for numerous generations of a given population. Note that the simulation of the discrete model is distinct from the simulation of the diffusion approximation. That is, the diffusion process serves to approximate the discrete process. Indeed, the simulation of the diffusion approximation would follow along the lines of the Euler-Murayama scheme or perhaps some other numerical scheme (Jenkins and Spano (2015) develop an exact scheme for simulating various forms Wright-Fisher diffusion process – not including shocks of course). For purposes of the analysis in this thesis, we focus on simulating the discrete process to which we formulate an approximation under a jump diffusion process. Following the Wright-Fisher process with frequency shocks as described in Section 4.7.2, we outline the discrete algorithm corresponding to Equation 4.7.13:

Let X_t denote the number for alleles of type A for a population of size N. Let $\mathbf{a}(t) = \{a_i(t), i = 1, 2, \dots N\}$ denote a set of indicators corresponding to the type of each individual in the population at generation t. Furthermore, let f_t denote the frequency of allele A. Then:

1. Set $t = 0$ and initialize the population by setting X_t elements of $\mathbf{a}(t)$ to type A and the remaining elements to type B corresponding to the initial frequency.

2. Sample, with replacement from the set $\mathbf{a}(t)$ where alleles of type A are sampled with probability $(1 + s/N)/(2 + s/N)$ and alleles of type B are sampled with probability $1/(2 + s/N)$.

3. Simulate a random deviate $u \sim \mathrm{U}(0, 1)$ and if $\lambda/N \geq u$, then set

$$f_{t+1} = f_t + \dot{z}_t(1 - f_t) \tag{D.8.1}$$

   where $\dot{z}_t \sim \mathrm{U}(0, 0.5)$ and reset the population indicators in $\mathbf{a}(t)$ to match the revised frequency.

4. Set $t = t + 1$ and if t is less than the desired number of generations go to step 2, else stop.

Note that here X_t corresponds to the number of alleles of type A, whereas f_t corresponds to the frequency of allele A. Thus, the jump diffusion approximation corresponds to the dynamics of f_t.

D.9 A C++ routine for evaluating the likelihood of a jump diffusion model.

Here, the purpose of the `solver()` function is to evaluate the likelihood function of a scalar jump diffusion model using the mixture factorization. Specifically, the code pertains to Equation 4.5.5, and is constructed during a call to `JGQD.mcmc()` from the **DiffusionRjgqd** package (see R code: Supplementary materials, Section 4.11.).

```
#include <RcppArmadillo.h>
#include <math.h>
#define pi           3.14159265358979323846  /* pi */
using namespace arma;
using namespace Rcpp;
using namespace R;

// [[Rcpp::depends("RcppArmadillo")]]
// [[Rcpp::export]]
vec prod(vec a,vec b)
{
    return(a%b);
}
// [[Rcpp::export]]
mat f(mat a,vec theta,vec t,int N2)
{
  mat atemp(N2,10);
  double mu  =theta[5];
  double sig =theta[6];
  double mm1 = mu;
  double mm2 = pow(mu,2)+ pow(sig,2);
  double mm3 = pow(mu,3)+ 3* pow(mu,1)*pow(sig,2);
  double mm4 = pow(mu,4)+ 6* pow(mu,2)*pow(sig,2)+3*pow(sig,4);

  vec    coef1 =theta[1] * theta[2] + theta[1] * theta[7] * sin(8 * pi * t + (theta[8] - 0.5) * 2 *
      pi);
  double    coef2 =-theta[1];
  double    coef6 =theta[3] * theta[3];

  atemp.col(0)=(+( coef1)%(1*(1+0*a.col(0)))+( coef2)*(1*a.col(0))+(theta[4])*(+1*mm1*a.col(0)));
  atemp.col(1)=(+( coef1)%(2*a.col(0))+( coef2)*(2*a.col(1))+( coef6)*(1*a.col(1))+(theta[4])*(+2*
     mm1*a.col(1)+1*mm2*a.col(0)));
  atemp.col(2)=(+( coef1)%(3*a.col(1))+( coef2)*(3*a.col(2))+( coef6)*(3*a.col(2))+(theta[4])*(+3*
     mm1*a.col(2)+3*mm2*a.col(1)+1*mm3*a.col(0)));
  atemp.col(3)=(+( coef1)%(4*a.col(2))+( coef2)*(4*a.col(3))+( coef6)*(6*a.col(3))+(theta[4])*(+4*
     mm1*a.col(3)+6*mm2*a.col(2)+4*mm3*a.col(1)+1*mm4*a.col(0)));
  atemp.col(4)=(+( coef1)%(1*(1+0*a.col(4)))+( coef2)*(1*a.col(4)));
  atemp.col(5)=(+( coef1)%(2*a.col(4))+( coef2)*(2*a.col(5))+( coef6)*(1*a.col(5)));
  atemp.col(6)=(+( coef1)%(3*a.col(5))+( coef2)*(3*a.col(6))+( coef6)*(3*a.col(6)));
  atemp.col(7)=(+( coef1)%(4*a.col(6))+( coef2)*(4*a.col(7))+( coef6)*(6*a.col(7)));
  atemp.col(8)=(0*a.col(0));
  atemp.col(9)=(theta[4])*a.col(0);

 return atemp;
}

// [[Rcpp::export]]
List  solver(vec Xs,vec Xt,vec theta,int N,mat delt,int N2,vec tt,int P,double alpha,double lower,
     double upper,int tro,double eps=0.01)
{
  int steps=0;
  mat x0(N2,2*tro+2);
  mat y0(N2,2*tro+2);
  mat xa(N2,2*tro+2);
  mat xb(N2,2*tro+2);
  mat fx1(N2,2*tro+2);
  mat fx2(N2,2*tro+2);
  mat fx3(N2,2*tro+2);
  mat fx4(N2,2*tro+2);
  mat fx5(N2,2*tro+2);
  mat fx6(N2,2*tro+2);

  mat etemp(N2,1);
```

```
  x0.fill(0);
  for (int i = 1; i < tro+1; i++)
  {
    x0.col(i-1)=pow(Xs,i);
    x0.col(i-1+tro)=pow(Xs,i);
  }
  vec dd = tt;

  for (int i = 1; i < N+1; i++)
  {
    fx1 = delt%f(x0,theta,dd,N2);
    fx2 = delt%f(x0+0.25*fx1,theta,dd+0.25*delt.col(0),N2);
    fx3 = delt%f(x0+0.09375*fx1+0.28125*fx2,theta,dd+0.375*delt.col(0),N2);
    fx4 = delt%f(x0+0.879381*fx1-3.277196*fx2+ 3.320892*fx3,theta,dd+0.9230769*delt.col(0),N2);
    fx5 = delt%f(x0+2.032407*fx1-8*fx2+7.173489*fx3-0.2058967*fx4,theta,dd+delt.col(0),N2);
    fx6 = delt%f(x0-0.2962963*fx1+2*fx2-1.381676*fx3+0.4529727*fx4-0.275*fx5,theta,dd+0.5*delt.col
      (0),N2);
    xa=  x0+0.1157407*fx1+0.5489279*fx3+0.5353314*fx4-0.2*fx5;
    xb =  x0+0.1185185*fx1+0.5189864*fx3+0.5061315*fx4-0.18*fx5+0.03636364*fx6;
    x0=xa;
    dd=dd+delt.col(0);
  }

  y0.col(0)= x0.col(0+4);
  y0.col(1)= x0.col(1+4)-1*y0.col(0)%x0.col(0+4);
  y0.col(2)= x0.col(2+4)-1*y0.col(0)%x0.col(1+4)-2*y0.col(1)%x0.col(0+4);
  y0.col(3)= x0.col(3+4)-1*y0.col(0)%x0.col(2+4)-3*y0.col(1)%x0.col(1+4)-3*y0.col(2)%x0.col(0+4);

  y0.col(4)= ((x0.col(0)-x0.col(4)%exp(-x0.col(8)-x0.col(9)))/(1-exp(-x0.col(8)-x0.col(9))));
  y0.col(5)= ((x0.col(1)-x0.col(5)%exp(-x0.col(8)-x0.col(9)))/(1-exp(-x0.col(8)-x0.col(9))))-1*y0.
     col(4)%((x0.col(0)-x0.col(4)%exp(-x0.col(8)-x0.col(9)))/(1-exp(-x0.col(8)-x0.col(9))));
  y0.col(6)= ((x0.col(2)-x0.col(6)%exp(-x0.col(8)-x0.col(9)))/(1-exp(-x0.col(8)-x0.col(9))))-1*y0.
     col(4)%((x0.col(1)-x0.col(5)%exp(-x0.col(8)-x0.col(9)))/(1-exp(-x0.col(8)-x0.col(9))))-2*y0.
     col(5)%((x0.col(0)-x0.col(4)%exp(-x0.col(8)-x0.col(9)))/(1-exp(-x0.col(8)-x0.col(9))));
  y0.col(7)= ((x0.col(3)-x0.col(7)%exp(-x0.col(8)-x0.col(9)))/(1-exp(-x0.col(8)-x0.col(9))))-1*y0.
     col(4)%((x0.col(2)-x0.col(6)%exp(-x0.col(8)-x0.col(9)))/(1-exp(-x0.col(8)-x0.col(9))))-3*y0.
     col(5)%((x0.col(1)-x0.col(5)%exp(-x0.col(8)-x0.col(9)))/(1-exp(-x0.col(8)-x0.col(9))))-3*y0.
     col(6)%((x0.col(0)-x0.col(4)%exp(-x0.col(8)-x0.col(9)))/(1-exp(-x0.col(8)-x0.col(9))));

  vec p=(1.0/3.0) *(3*(y0.col(3)/6.0)%y0.col(1) - pow(y0.col(2)/2.0,2))/pow(y0.col(3)/6.0,2);
  vec q=(1.0/27.0)*(27*pow(y0.col(3)/6.0,2)%(y0.col(0)-Xt) - 9*(y0.col(3)/6.0)%(y0.col(2)/2.0)%y0.
     col(1) + 2*pow(y0.col(2)/2.0,3))/pow(y0.col(3)/6.0,3);
  vec chk=pow(q,2)/4.0 + pow(p,3)/27.0;
  vec th=-(y0.col(2)/2.0)/(3*(y0.col(3)/6.0))+pow(-q/2.0+sqrt(chk),(1.0/3.0))-pow(q/2.0+sqrt(chk)
     ,(1.0/3.0));

  vec K =y0.col(0)%th+(y0.col(1)%th%th)/2.0+(y0.col(2)%th%th%th)/6.0 +(y0.col(3)%th%th%th%th)/24.0;
  vec K1=y0.col(0)    +(y0.col(1)%th)        +(y0.col(2)%th%th)/2.0     +(y0.col(3)%th%th%th)/6.0;
  vec K2=y0.col(1)    +(y0.col(2)%th)        +(y0.col(3)%th%th)/2.0;
  vec val=exp(-0.5*log(2*3.141592653589793*K2)+(K-th%K1)-x0.col(8)-x0.col(9));
  List ret;
  ret["like2"]  = (-0.5*log(2*3.141592653589793*K2)+(K-th%K1));

  p=(1.0/3.0) *(3*(y0.col(7)/6.0)%y0.col(5) - pow(y0.col(6)/2.0,2))/pow(y0.col(7)/6.0,2);
  q=(1.0/27.0)*(27*pow(y0.col(7)/6.0,2)%(y0.col(4)-Xt) - 9*(y0.col(7)/6.0)%(y0.col(6)/2.0)%y0.col
     (5) + 2*pow(y0.col(6)/2.0,3))/pow(y0.col(7)/6.0,3);
  chk=pow(q,2)/4.0 + pow(p,3)/27.0;
  th=-(y0.col(6)/2.0)/(3*(y0.col(7)/6.0))+pow(-q/2.0+sqrt(chk),(1.0/3.0))-pow(q/2.0+sqrt(chk),(1.0/
     3.0));

  K =y0.col(4)%th+(y0.col(5)%th%th)/2.0+(y0.col(6)%th%th%th)/6.0 +(y0.col(7)%th%th%th%th)/24.0;
  K1=y0.col(4)    +(y0.col(5)%th)        +(y0.col(6)%th%th)/2.0     +(y0.col(7)%th%th%th)/6.0;
  K2=y0.col(5)    +(y0.col(6)%th)        +(y0.col(7)%th%th)/2.0;
  val= log(val+exp(-0.5*log(2*3.141592653589793*K2)+(K-th%K1))%(1-exp(-x0.col(8)-x0.col(9))));


  ret["like"]  = val;
  ret["like3"]  = log(exp(-0.5*log(2*3.141592653589793*K2)+(K-th%K1))%(1-exp(-x0.col(8)-x0.col(9)))
     );
  ret["exc"]  = (1-exp(-x0.col(8)-x0.col(9)));
  ret["steps"]  = steps;
  return(ret);
}
```

D.10 A C++ routine for evaluating the likelihood of a bivariate jump diffusion model.

Here, the purpose of the `solver()` function is to evaluate the likelihood function of a bivariate jump diffusion model using the mixture factorization. Specifically, the code pertains to Equation 4.5.10, and is constructed during a call to `BiJGQD.mcmc()` from the **DiffusionRjgqd** package (see R code: Supplementary materials, Section 4.12.).

```
#include <RcppArmadillo.h>
#include <math.h>
#include <Rcpp.h>
#define pi           3.14159265358979323846  /* pi */
using namespace arma;
using namespace Rcpp;
using namespace R;

// [[Rcpp::depends("RcppArmadillo")]]
// [[Rcpp::export]]
vec prod(vec a,vec b)
{
    return(a%b);
}
mat f(mat a,vec theta,vec t,int N2)
{
  mat atemp(N2,34);
  vec m00 = (1+0*a.col(0));
  vec m10 = a.col(0);
  vec m20 = a.col(1);
  vec m30 = a.col(2);
  vec m40 = a.col(3);
  vec m01 = a.col(4);
  vec m02 = a.col(5);
  vec m03 = a.col(6);
  vec m04 = a.col(7);
  vec m11 = a.col(8);
  vec m12 = a.col(9);
  vec m21 = a.col(10);
  vec m22 = a.col(11);
  vec m13 = a.col(12);
  vec m31 = a.col(13);

  vec mm00 = (1+0*a.col(0));
  vec mm10 = a.col(14);
  vec mm20 = a.col(15);
  vec mm30 = a.col(16);
  vec mm40 = a.col(17);
  vec mm01 = a.col(18);
  vec mm02 = a.col(19);
  vec mm03 = a.col(20);
  vec mm04 = a.col(21);
  vec mm11 = a.col(22);
  vec mm12 = a.col(23);
  vec mm21 = a.col(24);
  vec mm22 = a.col(25);
  vec mm13 = a.col(26);
  vec mm31 = a.col(27);

  double mu1=theta[8];
  double mu2=theta[9];
  double sig11=theta[10] * theta[10];
  double sig12=0;
  double sig22=theta[11] * theta[11];

  doublemv10=mu1;
  doublemv20=pow(mu1,2)+sig11;
  doublemv30=pow(mu1,3)+3*pow(mu1,1)*sig11;
  doublemv40=pow(mu1,4)+6*pow(mu1,2)*sig11+3*pow(sig11,2);
  doublemv01=mu2;
  doublemv02=pow(mu2,2)+sig22;
  doublemv03=pow(mu2,3)+3*pow(mu2,1)*sig22;
```

```
  doublemv04=pow(mu2,4)+6*pow(mu2,2)*sig22+3*pow(sig22,2);
  doublemv11=mu1*mu2+sig12;
  doublemv12=mu1*pow(mu2,2)+2*mu2*sig12+mu1*sig22;
  doublemv21=pow(mu1,2)*mu2+2*mu1*sig12+mu2*sig11;
  doublemv22=pow(mu1,2)*pow(mu2,2)+pow(mu2,2)*sig11+pow(mu1,2)*sig22+4*mu1*mu2*sig12+sig11*sig22+2*
      sig12*sig12;
  doublemv13=mu1*pow(mu2,3)+3*pow(mu2,2)*sig12+3*mu1*mu2*sig22+3*sig12*sig22;
  doublemv31=mu2*pow(mu1,3)+3*pow(mu1,2)*sig12+3*mu1*mu2*sig11+3*sig12*sig11;

  atemp.col(0)=+(theta[1] * theta[2])*(1*m00)+(-theta[1])*(1*m10)+(theta[1])*(1*m01)+(theta[7])*(+1
      *mv10*m00);
  atemp.col(1)=+(theta[1] * theta[2])*(2*m10)+(-theta[1])*(2*m20)+(theta[1])*(2*m11)+(theta[3] *
      theta[3])*(1*m11)+(theta[7])*(+2*mv10*m10+1*mv20);
  atemp.col(2)=+(theta[1] * theta[2])*(3*m20)+(-theta[1])*(3*m30)+(theta[1])*(3*m21)+(theta[3] *
      theta[3])*(3*m21)+(theta[7])*(+3*mv10*m20+3*mv20*m10+1*mv30);
  atemp.col(3)=+(theta[1] * theta[2])*(4*m30)+(-theta[1])*(4*m40)+(theta[1])*(4*m31)+(theta[3] *
      theta[3])*(6*m31)+(theta[7])*(+4*mv10*m30+6*mv20*m20+4*mv30*m10+1*mv40);
  atemp.col(4)=+(theta[4] * theta[5])*(1*m00)+(-theta[4])*(1*m01)+(theta[7])*(+1*mv01*m00);
  atemp.col(5)=+(theta[4] * theta[5])*(2*m01)+(-theta[4])*(2*m02)+(theta[6] * theta[6])*(1*m01)+(
      theta[7])*(+2*mv01*m01+1*mv02);
  atemp.col(6)=+(theta[4] * theta[5])*(3*m02)+(-theta[4])*(3*m03)+(theta[6] * theta[6])*(3*m02)+(
      theta[7])*(+3*mv01*m02+3*mv02*m01+1*mv03);
  atemp.col(7)=+(theta[4] * theta[5])*(4*m03)+(-theta[4])*(4*m04)+(theta[6] * theta[6])*(6*m03)+(
      theta[7])*(+4*mv01*m03+6*mv02*m02+4*mv03*m01+1*mv04);
  atemp.col(8)=+(theta[1] * theta[2])*(1*m01)+(-theta[1])*(1*m11)+(theta[1])*(1*m02)+(theta[4] *
      theta[5])*(1*m10)+(-theta[4])*(1*m11)+(theta[7])*(mv10*m01+mv01*m10+mv11);
  atemp.col(9)=+(theta[1] * theta[2])*(1*m02)+(-theta[1])*(1*m12)+(theta[1])*(1*m03)+(theta[4] *
      theta[5])*(2*m11)+(-theta[4])*(2*m12)+(theta[6] * theta[6])*(1*m11)+(theta[7])*(mv10*m02+2*
      mv01*m11+2*mv11*m01+mv02*m10+mv12);
  atemp.col(10)=+(theta[1] * theta[2])*(2*m11)+(-theta[1])*(2*m21)+(theta[1])*(2*m12)+(theta[4] *
      theta[5])*(1*m20)+(-theta[4])*(1*m21)+(theta[3] * theta[3])*(1*m12)+(theta[7])*(2*mv10*m11+
      mv20*m01+mv01*m20+2*mv11*m10+mv21);
  atemp.col(11)=+(theta[1] * theta[2])*(2*m12)+(-theta[1])*(2*m22)+(theta[1])*(2*m13)+(theta[4] *
      theta[5])*(2*m21)+(-theta[4])*(2*m22)+(theta[3] * theta[3])*(1*m13)+(theta[6] * theta[6])*(1*
      m21)+(theta[7])*(2*mv10*m12+mv20*m02+2*mv01*m21+4*mv11*m11+2*mv21*m01+mv02*m20+2*mv12*m10+mv22
      );
  atemp.col(12)=+(theta[1] * theta[2])*(1*m03)+(-theta[1])*(1*m13)+(theta[1])*(1*m04)+(theta[4] *
      theta[5])*(3*m12)+(-theta[4])*(3*m13)+(theta[6] * theta[6])*(3*m12)+(theta[7])*(mv10*m03+3*
      mv01*m12+3*mv11*m02+3*mv02*m11+3*mv12*m01+mv03*m10+mv13);
  atemp.col(13)=+(theta[1] * theta[2])*(3*m21)+(-theta[1])*(3*m31)+(theta[1])*(3*m22)+(theta[4] *
      theta[5])*(1*m30)+(-theta[4])*(1*m31)+(theta[3] * theta[3])*(3*m22)+(theta[7])*(3*mv10*m21+3*
      mv20*m11+mv30*m01+mv01*m30+3*mv11*m20+3*mv21*m10+mv31);
  atemp.col(14)=+(theta[1] * theta[2])*(1*mm00)+(-theta[1])*(1*mm10)+(theta[1])*(1*mm01);
  atemp.col(15)=+(theta[1] * theta[2])*(2*mm10)+(-theta[1])*(2*mm20)+(theta[1])*(2*mm11)+(theta[3]
      * theta[3])*(1*mm11);
  atemp.col(16)=+(theta[1] * theta[2])*(3*mm20)+(-theta[1])*(3*mm30)+(theta[1])*(3*mm21)+(theta[3]
      * theta[3])*(3*mm21);
  atemp.col(17)=+(theta[1] * theta[2])*(4*mm30)+(-theta[1])*(4*mm40)+(theta[1])*(4*mm31)+(theta[3]
      * theta[3])*(6*mm31);
  atemp.col(18)=+(theta[4] * theta[5])*(1*mm00)+(-theta[4])*(1*mm01);
  atemp.col(19)=+(theta[4] * theta[5])*(2*mm01)+(-theta[4])*(2*mm02)+(theta[6] * theta[6])*(1*mm01)
      ;
  atemp.col(20)=+(theta[4] * theta[5])*(3*mm02)+(-theta[4])*(3*mm03)+(theta[6] * theta[6])*(3*mm02)
      ;
  atemp.col(21)=+(theta[4] * theta[5])*(4*mm03)+(-theta[4])*(4*mm04)+(theta[6] * theta[6])*(6*mm03)
      ;
  atemp.col(22)=+(theta[1] * theta[2])*(1*mm01)+(-theta[1])*(1*mm11)+(theta[1])*(1*mm02)+(theta[4]
      * theta[5])*(1*mm10)+(-theta[4])*(1*mm11);
  atemp.col(23)=+(theta[1] * theta[2])*(1*mm02)+(-theta[1])*(1*mm12)+(theta[1])*(1*mm03)+(theta[4]
      * theta[5])*(2*mm11)+(-theta[4])*(2*mm12)+(theta[6] * theta[6])*(1*mm11);
  atemp.col(24)=+(theta[1] * theta[2])*(2*mm11)+(-theta[1])*(2*mm21)+(theta[1])*(2*mm12)+(theta[4]
      * theta[5])*(1*mm20)+(-theta[4])*(1*mm21)+(theta[3] * theta[3])*(1*mm12);
  atemp.col(25)=+(theta[1] * theta[2])*(2*mm12)+(-theta[1])*(2*mm22)+(theta[1])*(2*mm13)+(theta[4]
      * theta[5])*(2*mm21)+(-theta[4])*(2*mm22)+(theta[3] * theta[3])*(1*mm13)+(theta[6] * theta[6])
      *(1*mm21);
  atemp.col(26)=+(theta[1] * theta[2])*(1*mm03)+(-theta[1])*(1*mm13)+(theta[1])*(1*mm04)+(theta[4]
      * theta[5])*(3*mm12)+(-theta[4])*(3*mm13)+(theta[6] * theta[6])*(3*mm12);
  atemp.col(27)=+(theta[1] * theta[2])*(3*mm21)+(-theta[1])*(3*mm31)+(theta[1])*(3*mm22)+(theta[4]
      * theta[5])*(1*mm30)+(-theta[4])*(1*mm31)+(theta[3] * theta[3])*(3*mm22);
  atemp.col(28)=(theta[7])*m00;
  atemp.col(29)=(+0*a.col(0));
  atemp.col(30)=(+0*a.col(0));
  atemp.col(31)=(+0*a.col(0));
  atemp.col(32)=(+0*a.col(0));
  atemp.col(33)=(+0*a.col(0));
  return atemp;
}

// [[Rcpp::export]]
```

```
List  solver(vec Xs,vec Ys,vec Xt,vec Yt,vec theta,int N,double delt,int N2,vec tt,mat starts,int
     tro,int secmom,vec seq1,vec seq2)
{
  mat resss(N2,3);
  mat x0(N2,tro);
  mat xa(N2,tro);
  mat xe(N2,tro);
  mat fx1(N2,tro);
  mat fx2(N2,tro);
  mat fx3(N2,tro);
  mat fx4(N2,tro);
  mat fx5(N2,tro);
  mat fx6(N2,tro);
  double whch =0;
  x0.fill(0);
  for (int i = 1; i <= 14; i++)
  {
   x0.col(i-1)=pow(Xs,seq1[i-1])%pow(Ys,seq2[i-1]);
   x0.col(i-1+14)=pow(Xs,seq1[i-1])%pow(Ys,seq2[i-1]);
  }

  vec d=tt;
  for (int i = 1; i < N; i++)
 {
  fx1 = f(x0,theta,d,N2)*delt;
  fx2 = f(x0+0.25*fx1,theta,d+0.25*delt,N2)*delt;
  fx3 = f(x0+0.09375*fx1+0.28125*fx2,theta,d+0.375*delt,N2)*delt;
  fx4 = f(x0+0.879381*fx1-3.277196*fx2+ 3.320892*fx3,theta,d+0.9230769*delt,N2)*delt;
  fx5 = f(x0+2.032407*fx1-8*fx2+7.173489*fx3-0.2058967*fx4,theta,d+delt,N2)*delt;
  fx6 = f(x0-0.2962963*fx1+2*fx2-1.381676*fx3+0.4529727*fx4-0.275*fx5,theta,d+0.5*delt,N2)*delt;
  xa   =  x0+0.1185185*fx1+0.5189864*fx3+0.5061315*fx4-0.18*fx5+0.03636364*fx6;
  x0   =  x0+0.1157407*fx1+0.5489279*fx3+0.5353314*fx4-0.2*fx5;
  xe   =  abs(x0.col(1)-xa.col(1));
  if(xe.max()>whch)
  {
    whch = xe.max();
  }
  d=d+delt;
 }

  vec probs=exp(-x0.col(28)-x0.col(29)-x0.col(30)-x0.col(31)-x0.col(32)-x0.col(33));

  vec m00 = (1+0*x0.col(0));
  vec m10 = (x0.col(0) -probs%x0.col(14))/(1-probs);
  vec m20 = (x0.col(1) -probs%x0.col(15))/(1-probs);
  vec m30 = (x0.col(2) -probs%x0.col(16))/(1-probs);
  vec m40 = (x0.col(3) -probs%x0.col(17))/(1-probs);
  vec m01 = (x0.col(4) -probs%x0.col(18))/(1-probs);
  vec m02 = (x0.col(5) -probs%x0.col(19))/(1-probs);
  vec m03 = (x0.col(6) -probs%x0.col(20))/(1-probs);
  vec m04 = (x0.col(7) -probs%x0.col(21))/(1-probs);
  vec m11 = (x0.col(8) -probs%x0.col(22))/(1-probs);
  vec m12 = (x0.col(9) -probs%x0.col(23))/(1-probs);
  vec m21 = (x0.col(10)-probs%x0.col(24))/(1-probs);
  vec m22 = (x0.col(11)-probs%x0.col(25))/(1-probs);
  vec m13 = (x0.col(12)-probs%x0.col(26))/(1-probs);
  vec m31 = (x0.col(13)-probs%x0.col(27))/(1-probs);

  vec mm00 = (1+0*x0.col(0));
  vec mm10 = x0.col(14);
  vec mm20 = x0.col(15);
  vec mm30 = x0.col(16);
  vec mm40 = x0.col(17);
  vec mm01 = x0.col(18);
  vec mm02 = x0.col(19);
  vec mm03 = x0.col(20);
  vec mm04 = x0.col(21);
  vec mm11 = x0.col(22);
  vec mm12 = x0.col(23);
  vec mm21 = x0.col(24);
  vec mm22 = x0.col(25);
  vec mm13 = x0.col(26);
  vec mm31 = x0.col(27);


  vec k10 = m10;
  vec k20 = m20-pow(m10,2);
  vec k30 = m30-3*m20%m10+2*pow(m10,3);
  vec k40 = m40 -4*m30%m10-3*pow(m20,2)+12*m20%pow(m10,2)-6*pow(m10,4);
  vec k01 = m01;
```

```
vec k02 = m02-  pow(m01,2) ;
vec k03 = m03-3*m02%m01+2*pow(m01,3) ;
vec k04 = m04-4*m03%m01-3*pow(m02,2)+12*m02%pow(m01,2)-6*pow(m01,4);
vec k11 = m11-m10%m01;
vec k21 = m21-2*m11%m10-m20%m01+2*pow(m10,2)%m01;
vec k12 = m12-2*m11%m01-m02%m10+2*pow(m01,2)%m10;
vec k22 = m22-2*m21%m01-2*m12%m10-m20%m02-2*pow(m11,2)+8*m11%m01%m10+2*m02%pow(m10,2)+2*m20%pow(
   m01,2)-6*pow(m10,2)%pow(m01,2) ;
vec k31 = m31-3*m21%m10-m30%m01-3*m20%m11+6*m11%pow(m10,2)+6*m20%m10%m01-6*pow(m10,3)%m01 ;
vec k13 = m13-3*m12%m01-m03%m10-3*m02%m11+6*m11%pow(m01,2)+6*m02%m01%m10-6*pow(m01,3)%m10;

vec kk10 = mm10;
vec kk20 = mm20-pow(mm10,2);
vec kk30 = mm30-3*mm20%mm10+2*pow(mm10,3);
vec kk40 = mm40 -4*mm30%mm10-3*pow(mm20,2)+12*mm20%pow(mm10,2)-6*pow(mm10,4);
vec kk01 = mm01;
vec kk02 = mm02-  pow(mm01,2) ;
vec kk03 = mm03-3*mm02%mm01+2*pow(mm01,3) ;
vec kk04 = mm04-4*mm03%mm01-3*pow(mm02,2)+12*mm02%pow(mm01,2)-6*pow(mm01,4);
vec kk11 = mm11-mm10%mm01;
vec kk21 = mm21-2*mm11%mm10-mm20%mm01+2*pow(mm10,2)%mm01;
vec kk12 = mm12-2*mm11%mm01-mm02%mm10+2*pow(mm01,2)%mm10;
vec kk22 = mm22-2*mm21%mm01-2*mm12%mm10-mm20%mm02-2*pow(mm11,2)+8*mm11%mm01%mm10+2*mm02%pow(mm10
   ,2)+2*mm20%pow(mm01,2)-6*pow(mm10,2)%pow(mm01,2) ;
vec kk31 = mm31-3*mm21%mm10-mm30%mm01-3*mm20%mm11+6*mm11%pow(mm10,2)+6*mm20%mm10%mm01-6*pow(mm10
   ,3)%mm01 ;
vec kk13 = mm13-3*mm12%mm01-mm03%mm10-3*mm02%mm11+6*mm11%pow(mm01,2)+6*mm02%mm01%mm10-6*pow(mm01
   ,3)%mm10;


vec a(N2);
vec b(N2);
vec abser(N2);
abser=0.1+abser;
a.ones();
b.ones();
vec det=(k10%k01-k11%k11);
a=-(Xt-k10)%k20/det+(Yt-k01)%k11/det;
b=+(Xt-k10)%k11/det-(Yt-k01)%k02/det;
vec gg(N2);
vec hh(N2);
vec gg1(N2);
vec hh1(N2);
vec gg2(N2);
vec hh2(N2);
vec ar(N2);
vec br(N2);
vec anew(N2);
vec bnew(N2);
int ind=0;
while((max(abser)>0.001)&&(ind<1500))
{
gg=k10+k20%a+(1.0/2.0)*k30%a%a+(1.0/6.0)*k40%a%a%a +k11%b +(1.0/2.0)*k12%b%b+k21%a%b+(1.0/6.0)*b%
   b%b%k13+(1.0/2.0)*a%a%b%k31+(1.0/2.0)*a%b%b%k22-Xt;
hh=k01+k02%b+(1.0/2.0)*k03%b%b+(1.0/6.0)*k04%b%b%b +k11%a +k12%a%b+(1.0/2.0)*k21%a%a+(1.0/2.0)*a%
   b%b%k13+(1.0/6.0)*a%a%a%k31+(1.0/2.0)*a%a%b%k22-Yt;

gg1=k20+k30%a+(1.0/2.0)*k40%a%a+k21%b+a%b%k31+(1.0/2.0)*b%b%k22;
gg2=k11 +k12%b+k21%a+(1.0/2.0)*b%b%k13+(1.0/2.0)*a%a%k31+a%b%k22;

hh1=k11 +k12%b+k21%a+(1.0/2.0)*b%b%k13+(1.0/2.0)*a%a%k31+a%b%k22;
hh2=k02+k03%b+(1.0/2.0)*k04%b%b +k12%a+a%b%k13+(1.0/2.0)*a%a%k22;

anew  =a+(hh%gg2-gg%hh2)/(gg1%hh2-gg2%hh1);
bnew  =b-(hh%gg1-gg%hh1)/(gg1%hh2-gg2%hh1);
abser =(pow(anew-a,2)+pow(bnew-b,2));
a=anew;
b=bnew;
ind=ind+1;
}

vec dens2=-log(2*3.141592653589793)-0.5*log(gg1%hh2-gg2%hh1)+(k10%a+k01%b+(1.0/2.0)*k20%a%a+(1.0/
   2.0)*k02%b%b+(1.0/6.0)*k30%a%a%a+(1.0/6.0)*k03%b%b%b+(1.0/24.0)*k40%a%a%a%a+(1.0/24.0)*k04%b%b
   %b%b+k11%a%b+(1.0/2.0)*k12%a%b%b+(1.0/2.0)*k21%a%a%b+(1.0/6.0)*a%b%b%b%k13+(1.0/6.0)*a%a%a%b%
   k31+(1.0/4.0)*a%a%b%b%k22-a%Xt-b%Yt);

abser=0.1+0*abser;
a.ones();
b.ones();
det=(kk10%kk01-kk11%kk11);
```

```
  a=-(Xt-kk10)%kk20/det+(Yt-kk01)%kk11/det;
  b=+(Xt-kk10)%kk11/det-(Yt-kk01)%kk02/det;
  ind=0;
  while((max(abser)>0.001)&&(ind<1500))
  {
  gg=kk10+kk20%a+(1.0/2.0)*kk30%a%a+(1.0/6.0)*kk40%a%a%a +kk11%b +(1.0/2.0)*kk12%b%b+kk21%a%b+(1.0/
     6.0)*b%b%b%kk13+(1.0/2.0)*a%a%b%kk31+(1.0/2.0)*a%b%b%kk22-Xt;
  hh=kk01+kk02%b+(1.0/2.0)*kk03%b%b+(1.0/6.0)*kk04%b%b%b +kk11%a +kk12%a%b+(1.0/2.0)*kk21%a%a+(1.0/
     2.0)*a%b%b%kk13+(1.0/6.0)*a%a%a%kk31+(1.0/2.0)*a%a%b%kk22-Yt;

  gg1=kk20+kk30%a+(1.0/2.0)*kk40%a%a+kk21%b+a%b%kk31+(1.0/2.0)*b%b%kk22;
  gg2=kk11 +kk12%b+kk21%a+(1.0/2.0)*b%b%kk13+(1.0/2.0)*a%a%kk31+a%b%kk22;

  hh1=kk11 +kk12%b+kk21%a+(1.0/2.0)*b%b%kk13+(1.0/2.0)*a%a%kk31+a%b%kk22;
  hh2=kk02+kk03%b+(1.0/2.0)*kk04%b%b +kk12%a+a%b%kk13+(1.0/2.0)*a%a%kk22;

  anew  =a+(hh%gg2-gg%hh2)/(gg1%hh2-gg2%hh1);
  bnew  =b-(hh%gg1-gg%hh1)/(gg1%hh2-gg2%hh1);
  abser =(pow(anew-a,2)+pow(bnew-b,2));
  a=anew;
  b=bnew;
  ind=ind+1;
  }
  vec dens1=-log(2*3.141592653589793)-0.5*log(gg1%hh2-gg2%hh1)+(kk10%a+kk01%b+(1.0/2.0)*kk20%a%a
     +(1.0/2.0)*kk02%b%b+(1.0/6.0)*kk30%a%a%a+(1.0/6.0)*kk03%b%b%b+(1.0/24.0)*kk40%a%a%a%a+(1.0/
     24.0)*kk04%b%b%b%b+kk11%a%b+(1.0/2.0)*kk12%a%b%b+(1.0/2.0)*kk21%a%a%b+(1.0/6.0)*a%b%b%b%kk13
     +(1.0/6.0)*a%a%a%b%kk31+(1.0/4.0)*a%a%b%b%kk22-a%Xt-b%Yt);

  resss.col(1)=a;
  resss.col(2)=b;
  resss.col(0)=log(exp(dens1)%probs+exp(dens2)%(1-probs));
  List ret;
  ret["like"] = resss;
  ret["like2"] = dens1;
  ret["like3"] = dens2;
  ret["max"] = whch;
  ret["probs"] = probs;
  return(ret);
}
```

Appendix E

Conclusion

E.1 Estimating the parameters of a time-inhomogeneous Markov-switching diffusion

Despite the apparent sensitivity of the transition rate estimates to the sample resolution, Equation 5.1.14 remains surprisingly robust with respect to the inclusion of time-inhomogeneous elements. Indeed, despite the already complex nature of Markov-switching diffusions, the inclusion of time-dependent terms in a model of a real-world process play significant role in accurately replicating the dynamics of the underlying model – as evidenced by the various datasets considered in this thesis. As such it is important that the methodology under which the model is analysed be general enough to support such models. For example, consider a time-inhomogeneous version of Equation 5.1.11:

$$dX_t = \alpha(\beta + \nu \sin(\pi t) - X_t)dt + \sigma\sqrt{X_t}dB_t \tag{E.1.1}$$

where σ_t alternates between states σ_1 and σ_2 according to a CTMC with transition rate matrix $\mathbf{R} = (\beta_{ij})_{2\times 2}$ as before. Figure E.1.1 illustrates a simulated trajectory for both Equation 5.1.11 and the volatility process under the parameter set $\{\alpha, \beta, \sigma_1, \sigma_2, -\beta_{11} = \beta_{12}, -\beta_{22} = \beta_{21}, \nu\} = \{1, 5, 0.25, 1, 1, 2, 2\}$. Using a modified Euler-Maruyama scheme, we generate observations at equispaced intervals, $t_{i+1} - t_i = 0.05$ time units apart on the observation horizon $[0, 50]$.

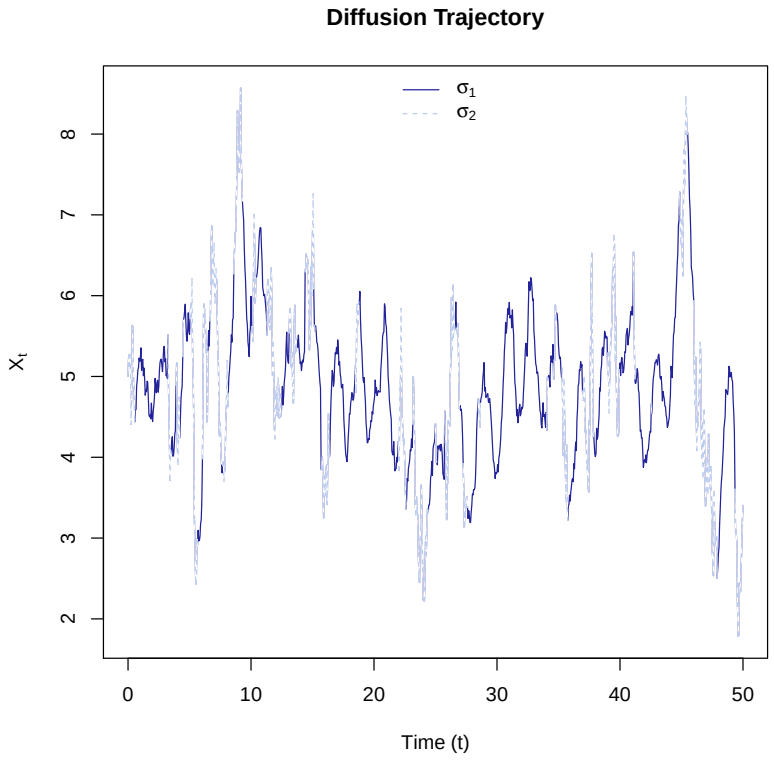


FIGURE E.1.1: Simulated time-inhomogeneous CIR process with Markov-switching volatility for the parameter set $\{\alpha, \beta, \sigma_1, \sigma_2, -\beta_{11} = \beta_{12}, -\beta_{22} = \beta_{21}, \nu\} = \{1, 5, 0.25, 1, 1, 2, 2\}$. Observations are made at equispaced epochs 0.05 time units apart on the observation horizon $[0, 50]$. R code: Supplementary materials, Section 5.7.

Based on the mixture factorization, we approximate the likelihood function for Equation E.1.1 under Equation 5.1.10 and calculate parameter estimates for the simulated trajectory in Figure E.1.1. Table E.1.1 gives the resulting parameter estimates and 90% credibility intervals calculated using the RWMH algorithm.

Parameter	True Value	Estimate	90% CI
α	1.00	1.145	(0.903, 1.319)
β	5.00	4.977	(4.854, 5.127)
σ_1	0.25	0.243	(0.231, 0.255)
σ_2	1.00	1.031	(0.964, 1.090)
$-\beta_{11} = \beta_{12}$	1.00	1.007	(0.726, 1.221)
$-\beta_{22} = \beta_{21}$	2.00	1.947	(1.458, 2.589)
ν	2.00	1.907	(1.608, 2.297)

TABLE E.1.1: Parameter estimates and 90% credibility intervals for Equation E.1.1 under the simulated dataset in Figure 5.1.3 calculated using the complete data likelihood using the true volatility trajectory. Estimates are calculated using 100 000 updates of the RWMH algorithm with a burn-in period of 10 000 iterations. R code: Supplementary materials, Section 5.7.

In addition to calculating the parameter estimates, we apply standard techniques for decoding likely states of the parameter chain. For purposes of the example illustrated here, we will focus on a local decoding since the calculation of the likelihood using Equation 5.1.14 readily facilitates the calculation of the so-called 'state probabilities' (see Chapter 5 of Zucchini and MacDonald (2009)). Figure E.1.2 illustrates the state probabilities for each state of the parameter chain, calculated using at the estimates shown in Table E.1.1. For reference we show the true trajectory of the state process. Note that the probabilities as calculated here are subject to scaling since the state-dependent probabilities of the process (Equation 5.1.13) are in fact densities and not probabilities.

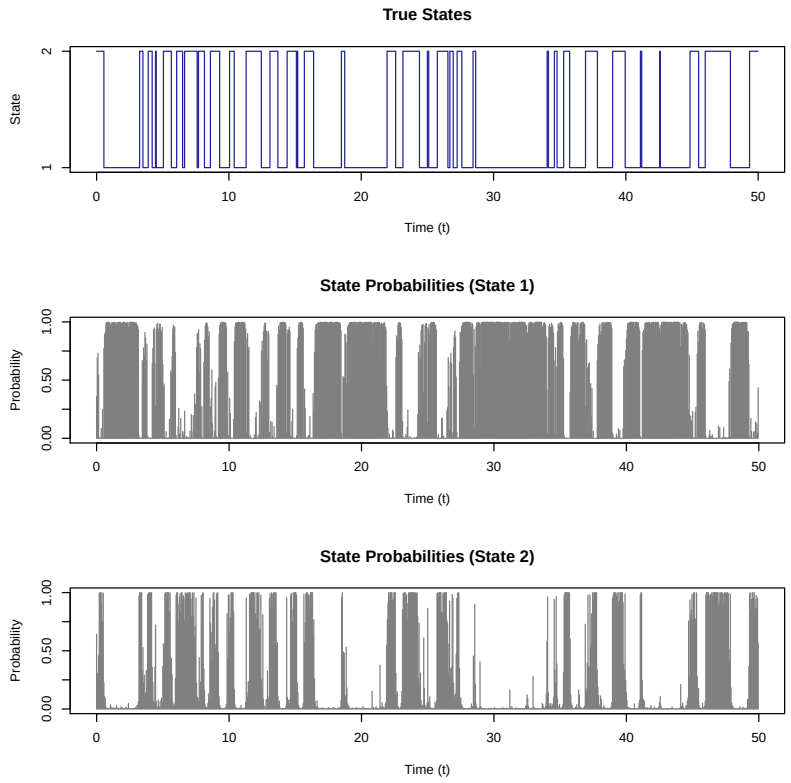


FIGURE E.1.2: True states of the parameter chain of the simulated dataset (top). State probabilities for low volatility (middle) and high volatility (bottom) respectively. The calculations are based on the parameter estimates shown in Table E.1.1. R code: Supplementary materials, Section 5.7.

E.2 A modified scheme for simulating first passage times to a detection point

Following the ideas in Giraudo and Sacerdote (1999), we use a modified Euler-Maruyama scheme in order to simulate the first passage time distribution of a bivariate diffusion to the exact coordinate of a detection point. In the case of the first passage time of a bivariate diffusion to an exact coordinate, it is clear that the standard technique of simulating the trajectory discretely and subsequently monitoring the trajectory until the first passage time event occurs will fail to produce a first passage time. This follows since the discrete updating scheme is highly unlikely to ever produce a state which matches the coordinate value

exactly. Consequently, we follow the convention that the process is likely to visit the exact coordinate within the transition horizon of the Euler step. As such, we modify the monitoring scheme by calculating the probability that the process moved to the coordinate and the updated state of the process at each Euler-Maruyama step.

Let $\mathbf{X}_t$ denote a bivariate diffusion process governed by the SDE in Equation 5.1.15 with initial state $\mathbf{X}_s$ at time $s < t$, $\boldsymbol{\phi}$ denote the coordinate of a detection point, and $\Delta > 0$ denote a finite step size, then the modified scheme follows:

1. Initialize: Set $d = s$ and $\mathbf{X}_d = \mathbf{X}_s$.
2. Update the trajectory of the process:
   (a) Using the Euler-Maruyama scheme, draw:
   $$\mathbf{X}_{d+\Delta} \sim \text{MVN}(\mathbf{X}_d + \boldsymbol{\mu}(\mathbf{X}_d, d)\Delta, \boldsymbol{\sigma}(\mathbf{X}_d, d)\boldsymbol{\sigma}'(\mathbf{X}_d, d)\Delta), \tag{E.2.1}$$
   where $\text{MVN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the multivariate Normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
   (b) Calculate :
   $$P = \frac{f(\mathbf{X}_{d+\Delta}|\boldsymbol{\phi}, d + \Delta/2) \times f(\boldsymbol{\phi}, d + \Delta/2|\mathbf{X}_d)}{f(\mathbf{X}_{d+\Delta}|\mathbf{X}_d)}, \tag{E.2.2}$$
   where $f(\mathbf{X}_t|\mathbf{X}_s)$ denotes the transitional density of the diffusion process – which may in-turn also be approximated using the multivariate Normal distribution. See Section 3 of Giraudo and Sacerdote (1999) for the intuition behind this step.
3. If $\mathbf{X}_{d+\Delta} = \boldsymbol{\phi}$ or $\min(P, 1) > u$ where $u \sim \text{U}(0,1)$, record the first passage time as $d + \Delta/2$. Otherwise, set $d = d + \Delta$ and go to step 2.

The algorithm can either be repeated until the desired number of first passage time events are simulated or vectorised in order to produce the desired number of simulated firs passage times in a single pass. Note that the algorithm of Giraudo and Sacerdote (1999) in the one-dimensional case relies on calculating the probability of a Brownian bridge crossing the threshold level within each time step. In this case, we employ the transitional density to calculate the probability of visiting the detection point halfway through each time step in order to conform to the geometry of the problem in two dimensions.

Bibliography

Adler D, Murdoch D, others (2016). **rgl***: 3D Visualization Using OpenGL*. R package version 0.95.1441, URL https://CRAN.R-project.org/package=rgl.

Aït-Sahalia Y (1999). "Transition Densities for Interest Rate and Other Nonlinear Diffusions." *The Journal of Finance*, **54**(4), 1361–1395.

Aït-Sahalia Y (2002). "Maximum Likelihood Estimation of Discretely Sampled Diffusions: A Closed-form Approximation Approach." *Econometrica*, **70**(1), 223–262.

Aït-Sahalia Y (2004). "Disentangling Diffusion From Jumps." *Journal of Financial Economics*, **74**(3), 487–528.

Aït-Sahalia Y (2008). "Closed-Form Likelihood Expansions for Multivariate Diffusions." *The Annals of Statistics*, **36**(2), 906–937.

Aït-Sahalia Y, Fan J, Peng H (2012). "Nonparametric Transition-Based Tests for Jump Diffusions." *Journal of the American Statistical Association*.

Aït-Sahalia Y, Jacod J (2011). "Testing Whether Jumps Have Finite or Infinite Activity." *The Annals of Statistics*, pp. 1689–1719.

Aït-Sahalia Y, Jacod J, *et al.* (2009). "Testing for Jumps in a Discretely Observed Process." *The Annals of Statistics*, **37**(1), 184–222.

Alili L, Patie P, Pedersen JL (2005). "Representations of the First Hitting Time Density of an Ornstein-Uhlenbeck Process 1." *Stochastic Models*, **21**(4), 967–980.

Andrews GE, Askey R, Roy R (1999). *Special Functions*, volume 71. Cambridge University Press.

Atkinson KE (2008). *An Introduction to Numerical Analysis*. John Wiley & Sons.

Ball CA, Torous WN (1985). "On Jumps in Common Stock Prices and Their Impact on Call Option Pricing." *The Journal of Finance*, **40**(1), 155–173.

Bandi FM, Nguyen TH (2003). "On the Functional Estimation of Jump-Diffusion Models." *Journal of Econometrics*, **116**(1), 293–328.

Barndorff-Nielsen O, Cox DR (1979). "Edgeworth and Saddle-Point Approximations With Statistical Applications." *Journal of the Royal Statistical Society B*, pp. 279–312.

Berglund N, Gentz B (2003). "Geometric Singular Perturbation Theory for Stochastic Differential Equations." *Journal of Differential Equations*, **191**(1), 1–54.

Beskos A, Papaspiliopoulos O, Roberts G (2009). "Monte Carlo Maximum Likelihood Estimation for Discretely Observed Diffusion Processes." *The Annals of Statistics*, pp. 223–245.

Black F, Cox JC (1976). "Valuing Corporate Securities: Some Effects of Bond Indenture Provisions." *The Journal of Finance*, **31**(2), 351–367.

Bollback JP, York TL, Nielsen R (2008). "Estimation of $2N_e s$ From Temporal Allele Frequency Data." *Genetics*, **179**(1), 497–502.

Borchers D, Distiller G, Foster R, Harmsen B, Milazzo L (2014). "Continuous-Time Spatially Explicit Capture–Recapture Models, with an Application to a Jaguar Camera-Trap Survey." *Methods in Ecology and Evolution*, **5**(7), 656–665.

Boukhetala K, Guidoum A, *et al.* (2011). "**Sim.DiffProc**: A Package for Simulation of Diffusion Processes in R." *Preprint Submitted to Journal of Statistical Software*.

Brillinger DR (2003). "Simulating Constrained Animal Motion Using Stochastic Differential Equations." *Lecture Notes-Monograph Series*, pp. 35–48.

Broadie M, Kaya Ö (2006). "Exact Simulation of Stochastic Volatility and Other Affine Jump Diffusion Processes." *Operations Research*, **54**(2), 217–231.

Buitenhuis ET (1999). "Photosynthesis and Calcification by Emiliania huxleyi (Prymnesiophyceae) as a Function of Inorganic Carbon Species." *Journal of Phycology*, **35**(5), 949.

Buonocore A, Nobile A, Ricciardi L (1987). "A New Integral Equation for the Evaluation of First-Passage-Time Probability Densities." *Advances in Applied Probability*, **19**, 784–800.

Butcher J (2007). "Runge-Kutta Methods." *Scholarpedia*, **2**(9), 3147. Revision #91735.

Carr P, Madan D (1999). "Option Valuation Using the Fast Fourier Transform." *Journal of Computational Finance*, **2**(4), 61–73.

Chen L (1996). *Stochastic Mean and Stochastic Volatility: A Three-Factor Model of the Term Structure of Interest Rates and Its Applications in Derivatives Pricing and Risk Management*. Blackwell Publishers.

Chib S, Shephard N, Pitt M (2004). *Likelihood Based Inference for Diffusion Driven Models*. University of Oslo.

Cobb L, Koppstein P, Chen NH (1983). "Estimation and Moment Recursion Relations for Multimodal Distributions of the Exponential Family." *Journal of the American Statistical Association*, **78**(381), 124–130.

Cox J, Ingersoll J, Ross S (1985). "A Theory of the Term Structure of Interest Rates." *Econometrica*, **53**, 385–407.

Craine R, Lochstoer LA, Syrtveit K (2000). "Estimation of a Stochastic-Volatility Jump-Diffusion Model." *Revista de Analisis Economico*, **15**, 61–87.

Daniels H (1954). "Saddlepoint Approximations in Statistics." *The Annals of Mathematical Statistics*, **25**, 631–650.

Dellaportas P, Friel N, Roberts GO (2006). "Bayesian Model Selection for Partially Observed Diffusion Models." *Biometrika*, **93**(4), 809–825.

Ditlevsen S, Lansky P (2005). "Estimation of the Input Parameters in the Ornstein-Uhlenbeck Neuronal Model." *Physical review E*, **71**(1), 011907.

Ditlevsen S, Lansky P (2006). "Estimation of the Input Parameters in the Feller Neuronal Model." *Physical Review E*, **73**(6), 061910.

Ditlevsen S, Lansky P (2007). "Parameters of Stochastic Diffusion Processes Estimated from Observations of First-Hitting Times: Application to the Leaky Integrate-and-Fire Neuronal Model." *Physical Review E*, **76**(4), 041906.

Durrett R (2008). *Probability Models for DNA Sequence Evolution*. Springer Science & Business Media.

Eckner A (2009). "Computational Techniques for Basic Affine Models of Portfolio Credit Risk." *Journal of Computational Finance*, **13**(1), 63.

Eddelbuettel D (2013). *Seamless R and C++ Integration With* **Rcpp**. Springer, New York. ISBN 978-1-4614-6867-7.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. URL http://www.jstatsoft.org/v40/i08/.

Eddelbuettel D, Nguyen K (2015). **RQuantLib***: R Interface to the 'QuantLib' Library*. R package version 0.4.2, URL https://CRAN.R-project.org/package=RQuantLib.

Eddelbuettel D, Sanderson C (2014). "**RcppArmadillo**: Accelerating R With High-Performance C++ Linear Algebra." *Computational Statistics and Data Analysis*, **71**, 1054–1063. URL http://dx.doi.org/10.1016/j.csda.2013.02.005.

Elerian O, Chib S, Shephard N (2001). "Likelihood Inference for Discretely Observed Nonlinear Diffusions." *Econometrica*, **69**(4), 959–993.

Engel A (2004). "Transparent Exopolymer Particles and Dissolved Organic Carbon Production by Emiliania huxleyi Exposed to Different CO2 Concentrations: A Mesocosm Experiment." *Aquatic Microbial Ecology*, **34**(1), 93.

Eraker B (2001). "MCMC Analysis of Diffusion Models With Application to Finance." *Journal of Business & Economic Statistics*, **19**(2), 177–191.

Eraker B (2004). "Do Stock Prices and Volatility Jump? Reconciling Evidence From Spot and Option Prices." *The Journal of Finance*, **59**(3), 1367–1404.

Eraker B, Johannes M, Polson N (2003). "The Impact of Jumps in Volatility and Returns." *The Journal of Finance*, **58**(3), 1269–1300.

Ethier SN, Norman MF (1977). "Error Estimate for the Diffusion Approximation of the Wright–Fisher Model." *Proceedings of the National Academy of Sciences*, **74**(11), 5096–5098.

Feagin T (2007). "A Tenth-Order Runge-Kutta Method With Error Estimate." In *Proceedings of the IAENG Conference on Scientific Computing*.

Fehlberg E (1970). "Classical Fourth and Lower Order Runge-Kutta Formulas With Stepsize Control and Their Application to Heat Transfer Problems." *Computing*, **6**(1), 61–71.

Filipović D, Mayerhofer E, Schneider P (2013). "Density Approximations for Multivariate Affine Jump-Diffusion Processes." *Journal of Econometrics*, **176**(2), 93–111.

Fokker AD (1914). "Die Mittlere Energie Rotierender Elektrischer Dipole im Strahlungsfeld." *Annalen der Physik*, **348**(5), 810–820.

Frank T, Beek P (2001). "Stationary Solutions of Linear Stochastic Delay Differential Equations: Applications to Biological Systems." *Physical Review E*, **64**(2), 021917.

Gallant AR, Hsieh D, Tauchen G (1997). "Estimation of Stochastic Volatility Models With Diagnostics." *Journal of Econometrics*, **81**(1), 159–192.

Gard T, Kannan D (1976). "On a Stochastic Differential Equation Modeling of Prey-Predator Evolution." *Journal of Applied Probability*, pp. 429–443.

Gilli M, Maringer D, Schumann E (2011). *Numerical Methods and Optimization in Finance*. Academic Press, Waltham, MA, USA. URL http://nmof.net.

Giorno V, Nobile A, Ricciardi L, Sato S (1989). "On the Evaluation of First-Passage-Time Probability Densities via Non-Singular Integral Equations." *Advances in Applied Probability*, pp. 20–36.

Giraudo MT, Sacerdote L (1999). "An Improved Technique for the Simulation of First Passage Times for Diffusion Processes." *Communications in Statistics-Simulation and Computation*, **28**(4), 1135–1163.

Gouriéroux C, Valéry P (2004). "Estimation of a Jacobi Process." *Preprint*, p. 116.

Guidoum A, Boukhetala K (2015). **Sim.DiffProc***: Simulation of Diffusion Processes.* **R** package version 3.1, URL http://CRAN.R-project.org/package=Sim.DiffProc.

Gutiérrez R, Román P, Torres F (1999). "Inference and First-Passage-Times for the Lognormal Diffusion Process With Exogenous Factors: Application to Modelling in Economics." *Applied Stochastic Models in Business and Industry*, **15**(4), 325–332.

Hamdi S, Schiesser WE, Griffiths GW (2007). "Method of Lines." *Scholarpedia*, **2**(7), 2859.

Hanson FB (2007). *Applied Stochastic Processes and Control for Jump-Diffusions: Modeling, Analysis, and Computation*, volume 13. SIAM.

Harvey JR (1972). "On Expressing Moments in Terms of Cumulants and Vice Versa." *The American Statistician*, **26**(4), 38–39.

Head RN, Crawford DW, Egge JK, Harris RP, Kristiansen S, Lesley DJ, Maranon E, Pond D, Purdie DA (1998). "The Hydrography and Biology of a Bloom of the Coccolithophorid Emiliania huxleyi in the Northern North Sea." *Journal of Sea Research*, **39**(3), 255–266.

Heston SL (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.

Higham DJ, Kloeden PE (2005). "Numerical Methods for Nonlinear Stochastic Differential Equations With Jumps." *Numerische Mathematik*, **101**(1), 101–119.

Honore P (1998). "Pitfalls in Estimating Jump-Diffusion Models." *Available at SSRN 61998*.

Hoppensteadt F (2006). "Predator-Prey Model." *Scholarpedia*, **1**(10), 1563. Revision #91666.

Horne JS, Garton EO, Krone SM, Lewis JS (2007). "Analyzing Animal Movements Using Brownian Bridges." *Ecology*, **88**(9), 2354–2363.

Huang X (2011). "Quasi-Maximum Likelihood Estimation of Discretely Observed Diffusions." *The Econometrics Journal*, **14**(2), 241–256.

Hurn AS, Lindsay KA, McClelland AJ (2015). "Estimating the Parameters of Stochastic Volatility Models Using Option Price Data." *Journal of Business & Economic Statistics*, **33**(4), 579–594.

Iacus SM (2009). *Simulation and Inference for Stochastic Differential Equations: With R Examples*. Springer Science & Business Media.

Iacus SM (2015). **sde***: Simulation and Inference for Stochastic Differential Equations*. R package version 2.0.14, URL https://CRAN.R-project.org/package=sde.

Inoue J, Sato S, Ricciardi LM (1995). "On the Parameter Estimation for Diffusion Models of Single Neuron's Activities." *Biological Cybernetics*, **73**(3), 209–221.

Iolov A, Ditlevsen S, Longtin A (2014). "Fokker–Planck and Fortet Equation-Based Parameter Estimation for a Leaky Integrate-and-Fire Model with Sinusoidal and Stochastic Forcing." *Journal of Mathematical Neuroscience*, **4**(1), 4.

Jáimez RG, Román PR, Ruiz FT (1995). "A Note on the Volterra Integral Equation for the First-Passage-Time Probability Density." *Journal of Applied Probability*, pp. 635–648.

Jenkins PA, Spano D (2015). "Exact Simulation of the Wright-Fisher Diffusion." *arXiv preprint arXiv:1506.06998*.

Jiang GJ, Knight JL (2002). "Estimation of Continuous-Time Processes via the Empirical Characteristic Function." *Journal of Business & Economic Statistics*, **20**(2).

Jin P, Rüdiger B, Trabelsi C (2016). "Positive Harris Recurrence and Exponential Ergodicity of the Basic Affine Jump-Diffusion." *Stochastic Analysis and Applications*, **34**(1), 75–95.

Johannes M (1999). "Jumps in Interest Rates: A Nonparametric Approach." *Journal of Finance*.

Kalogeropoulos K, Dellaportas P, Roberts GO (2011). "Likelihood Based Inference for Correlated Diffusions." *Canadian Journal of Statistics*, **39**(1), 52–72.

Karlin S, Taylor HE (1981). *A Second Course in Stochastic Processes*. Elsevier.

Kloeden PE, Platen E, Schurz H, Sørensen M (1996). "On Effects of Discretization on Estimators of Drift Parameters for Diffusion Processes." *Journal of Applied Probability*, pp. 1061–1076.

Kolmogoroff A (1931). "Über die analytischen Methoden in der Wahrscheinlichkeitsrechnung." *Mathematische Annalen*, **104**(1), 415–458.

Kou S (2007). "Jump-Diffusion Models for Asset Pricing in Financial Engineering." *Handbooks in Operations Research and Management Science*, **15**, 73–116.

Kou SG (2002). "A Jump-Diffusion Model for Option Pricing." *Management Science*, **48**(8), 1086–1101.

Krey S, Ligges U, Mersmann O (2011). **fftw***: Fast FFT and DCT Based on FFTW*. R package version 1.0-3, URL https://CRAN.R-project.org/package=fftw.

Krishnarajah I, Cook A, Marion G, Gibson G (2005). "Novel Moment Closure Approximations in Stochastic Epidemics." *Bulletin of Mathematical Biology*, **67**(4), 855–873.

Krishnarajah I, Marion G, Gibson G (2007). "Novel Bivariate Moment-Closure Approximations." *Mathematical Biosciences*, **208**(2), 621–643.

Leblanc B, Renault O, Scaillet O (2000). "A Correction Note on the First Passage Time of an Ornstein-Uhlenbeck Process to a Boundary." *Finance and Stochastics*, **4**(1), 109–111.

Linz P (1969). "Numerical Methods for Volterra Integral Equations of the First Kind." *The Computer Journal*, **12**(4), 393–397.

Majda AJ (1999). "Models for Stochastic Climate Prediction." *Proceedings of the National Academy of Sciences of the United States of America*, **96**(26), 14687.

Mao X, Marion G, Renshaw E (2002). "Environmental Brownian Noise Suppresses Explosions in Population Dynamics." *Stochastic Processes and Their Applications*, **97**(1), 95–110.

Marion G, Renshaw E, Gibson G (2000). "Stochastic Modelling of Environmental Variation for Biological Populations." *Theoretical Population Biology*, **57**(3), 197–217.

McTaggart R, Daroczi G (2015). **Quandl***: API Wrapper for Quandl.com*. R package version 2.6.0, URL http://CRAN.R-project.org/package=Quandl.

Merton RC (1976). "Option Pricing When Underlying Stock Returns are Discontinuous." *Journal of Financial Economics*, **3**(1), 125–144.

Norman MF (1975). "Approximation of Stochastic Processes by Gaussian Diffusions, and Applications to Wright-Fisher Genetic Models." *SIAM Journal on Applied Mathematics*, **29**(2), 225–242.

Paasche E (2001). "A Review of the Coccolithophorid Emiliania huxleyi (Prymnesiophyceae), With Particular Reference to Growth, Coccolith Formation, and Calcification-Photosynthesis Interactions." *Phycologia*, **40**(6), 503.

Pedersen AR (1995). "Consistency and Asymptotic Normality of an Approximate Maximum Likelihood Estimator for Discretely Observed Diffusion Processes." *Bernoulli*, pp. 257–279.

Pienaar EAD, Varughese MM (2015a). **DiffusionRgqd***: An R Package for Performing Inference and Analysis on Time-Inhomogeneous Quadratic Diffusion Processes*. R package version 0.1.2, URL https://CRAN.R-project.org/package=DiffusionRgqd.

Pienaar EAD, Varughese MM (2015b). **DiffusionRimp***: Inference and Analysis for Diffusion Processes via Data Imputation and Method of Lines*. R package version 0.1.0, URL https://CRAN.R-project.org/package=DiffusionRimp.

Pienaar EAD, Varughese MM (2015c). **DiffusionRjgqd***: An R Package for Performing Inference and Analysis on Time-Inhomogeneous Quadratic Jump Diffusions with State-Dependent Intensity*. R package version 0.1.1, URL https://CRAN.R-project.org/package=DiffusionRjgqd.

Planck M (1917). *Über einen Satz der Statistischen Dynamik und seine Erweiterung in der Quantentheorie*. Reimer.

Platen E (1982). "A Generalized Taylor Formula for Solutions of Stochastic Equations." *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 163–172.

Platen E (1999). "An Introduction to Numerical Methods for Stochastic Differential Equations." *Acta Numerica*, **8**, 197–246.

Plummer M, Best N, Cowles K, Vines K (2006). "**CODA**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL http://CRAN.R-project.org/doc/Rnews/.

Preisler HK (2004). "Modeling Animal Movements Using Stochastic Differential Equations." *Environmetrics*, **15**(7), 643.

Ramezani CA, Zeng Y (1998). "Maximum Likelihood Estimation of Asymmetric Jump-Diffusion Processes: Application to Security Prices." *Available at SSRN 606361*.

Renshaw E (2000). "Applying the Saddlepoint Approximation to Bivariate Stochastic Processes." *Mathematical Biosciences*, **168**(1), 57–75.

Ricciardi LM, Sacerdote L (1979). "The Ornstein-Uhlenbeck Process as a Model for Neuronal Activity." *Biological Cybernetics*, **35**(1), 1–9.

Ricciardi LM, Sacerdote L, Sato S (1984). "On an Integral Equation for First-Passage-Time Probability Densities." *Journal of Applied Probability*, pp. 302–314.

Roberts G, Stramer O (2001). "On Inference for Partially Observed Nonlinear Diffusion Models Using the Metropolis-Hastings Algorithm." *Biometrika*, **88**, 603–621.

Rockinger M, Semenova M (2005). "Estimation of Jump-Diffusion Processes via Empirical Characteristic Functions." *Available at SSRN 770785*.

Román-Román P, Serrano-Pérez J, Torres-Ruiz F (2014). "More General Problems on First-Passage Times for Diffusion Processes: A new Version of the **fptdApprox** R Package." *Applied Mathematics and Computation*, **244**, 432–446.

Smyth T (2011). "Temperature and Salinity Profiles at 5m Depth Resolution From Time Series Station L4 in the Western English Channel From 2002–2010." SBE19+ CTD. Plymouth Marine Laboratory, doi:10.1594/PANGAEA.757221.

Team RC, Wuertz D, Setz T, Chalabi Y (2015). **fOptions***: Rmetrics - Pricing and Evaluating Basic Options*. **R** package version 3022.85, URL https://CRAN.R-project.org/package=fOptions.

Tuckwell HC, Wan FY (1984). "First-Passage Time of Markov Process to Moving Barriers." *Journal of Applied Probability*, pp. 695–709.

Tyrrell T, Merico A (2004). "Emiliania huxleyi: Bloom Observations and the Conditions That Induce Them." In *Coccolithophores*, pp. 75–97. Springer.

Varughese M, Fatti L (2008). "Incorporating Environmental Stochasticity Within a Biological Population Model." *Theoretical Population Biology*, **74**(1), 115–129.

Varughese MM (2009). "On the Accuracy of a Diffusion Approximation to a Discrete State–Space Markovian Model of a Population." *Theoretical Population Biology*, **76**(4), 241–247.

Varughese MM (2011). "A Framework for Modelling Ecological Communities and Their Interactions With the Environment." *Ecological Complexity*, **8**(1), 105–112.

Varughese MM (2013). "Parameter Estimation for Multivariate Diffusion Systems." *Computational Statistics & Data Analysis*, **57**(1), 417–428.

Varughese MM, Pienaar EAD (2013). "Statistical Inference for a Multivariate Diffusion Model of an Ecological Time Series." *Ecosphere*, **4**(8), art104.

Widdicombe CE, Eloire D, Harbour D, Harris RP, Somerfield PJ (2010). "Long-term Phytoplankton Community Dynamics in the Western English Channel." *Journal of Plankton Research*, **32**(5), 643–655. Doi:10.1594/PANGAEA.758061.

Xia Y, Giles MB (2012). "Multilevel Path Simulation for Jump-Diffusion SDEs." In *Monte Carlo and Quasi-Monte Carlo Methods 2010*, pp. 695–708. Springer.

Yamada T, Watanabe S, *et al.* (1971). "On the Uniqueness of Solutions of Stochastic Differential Equations." *Journal of Mathematics of Kyoto University*, **11**(1), 155–167.

Yu J (2007). "Closed-form Likelihood Approximation and Estimation of Jump-Diffusions With an Application to the Realignment Risk of the Chinese Yuan." *Journal of Econometrics*, **141**(2), 1245–1280.

Zhang L, Schmidt WM (2016). "An Approximation of Amall-Time Probability Density Functions in a General Jump Diffusion Model." *Applied Mathematics and Computation*, **273**, 741–758.

Zhang X, You G, Chen T, Feng J (2009). "Maximum Likelihood Decoding of Neuronal Inputs From an Interspike Interval Distribution." *Neural Computation*, **21**(11), 3079–3105.

Zucchini W, MacDonald IL (2009). *Hidden Markov Models for Time Series: An Introduction Using R*, volume 150. CRC press.

Index

This thesis was typeset using a Latex template by Steven Gunn and Sunil Patel (Version 1.43 (17/5/14)).