Diffusion Processes and Applications to Financial Time Series

Jacobus Marthinus van der Berg - 13035836

WST895 Research Dissertation

Submitted in partial fulfillment of the degree MSc Mathematical Statistics

Supervisor: Dr. Etienne A.D. Pienaar

Co-supervisor: Dr. Schalk Human

Department of Statistics, University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

2021

February 24, 2022

Abstract

Diffusion processes are effective tools for modeling financial and economic phenomena. Diffusion models have been implemented with great success in financial markets where stochastic calculus based on such models allow researchers to probe the dynamics of processes ranging from stock prices, yields and interest rates to volatility studies and exchange rates. These processes, according to [17], allow for the investigation and quantification of the dynamics of various real world financial models. The dynamics of diffusion processes are governed by stochastic differential equations (SDEs), which dictate how these processes evolve over time. A key component in the analysis of such systems is the transitional density, which allows one to make predictions about the state of the process, or functions of the state of the process, when its parameters are known/fixed, or perhaps more importantly, when the parameters are not known a transition density allows one to estimate parameters and subsequently perform inference. Unfortunately, with the exception of certain processes, many of these models' transition density cannot be expressed by an explicit analytical expression. Therefore, efficient and consistent approximation techniques, to obtain an analytical expression for the transition density function, is of paramount interest and importance. The Hermite expansion method, of [3], outlines one of the most effective methods of obtaining an approximation to the transition density. The Saddlepoint, or Cumulant Truncation approximation method, provides a strong and robust alternative approximation method, [21] and [17]. In the present paper, we explore how these techniques can be used to analyse popular non-linear diffusion models from the world of finance. In particular, we focus on construction of the transition density approximations for the Ornstein-Uhlenbeck (OU) model, Cox-Ingersoll and Ross (CIR) model and the Heston model, and the application of these models to real-world datasets, such as the CBOE volatility/VIX index and the S&P 500 stock index. The Sapplepoint or Cumulant Truncated approximate transition density will be used to perform infernce on the mentioned datasets.

Declaration

I, *Jacobus Marthinus van der Berg,* declare that this dissertation, submitted in partial fulfillment of the degree *MSc Mathematical Statistics,* at the University of Pretoria, is my own work and has not been previously submitted at this or any other tertiary institution.

Jacobus Marthinus van der Berg

Acknowledgments

I would like to emphasise my immense gratitude towards Dr Pienaar, for his continued support, mentorship and guidance throughout this journey. Secondly, I would like to thank the UP Department of Statistics and Dr Human for the assistance and support towards the completion of this Dissertation. Lastly, the financial support, provided by the South African Reserve Bank, accompanied by the invaluable financial markets knowledge obtained, through the Bank, has been of great support.

Contents

List of Figures

1 Introduction

[17] defines a diffusion process as the stochastic generalisation to ordinary differential equations. Therefore, a general diffusion process, is defined by the following stochastic differential equation (SDE):

$$d\boldsymbol{X}_t = \mu(\boldsymbol{X}_t, t; \boldsymbol{\theta})dt + \sigma(\boldsymbol{X}_t, t; \boldsymbol{\theta})d\boldsymbol{W}_t, \tag{1}$$

s.t $t \in [s, T]$ represents the time domain over which the process continuously evolves. These models inherit the Markov property, and the stochastic nature is driven by Brownian motion $\boldsymbol{W}_t$, [1]. The state variable of interest, $\boldsymbol{X}_t$, is governed by the SDE, $d\boldsymbol{X}_t$; the time- and state dependent functions are given by $\mu(\boldsymbol{X}_t, t, \boldsymbol{\theta})$ and $\sigma(\boldsymbol{X_t}, t, \boldsymbol{\theta})$, respectively. $\boldsymbol{\theta}$ represents the parameter vector. *Itô*'s lemma is often applied to $\boldsymbol{X}_t$ to investigate the dynamics of the state variable, and to approximate the stochastic integrals in closed-form. Equation 1 can be viewed as the instantaneous change in the state of the process, $\boldsymbol{X}_t$. $d\boldsymbol{X_t}$ is governed by a time-dependent drift component $\mu(\boldsymbol{X}_t, t, \boldsymbol{\theta})$, which may depend on the state vector of the process and/or the time, as well as a stochastic component $\sigma(\boldsymbol{X}_t, t; \boldsymbol{\theta})d\boldsymbol{W}_t$ which may depend on the state of the process through the diffusion coefficient $\sigma(\boldsymbol{X}_t, t; \boldsymbol{\theta})$ and continuous time stochastic process, $\boldsymbol{W}_t$ which in this case we assume to be a vector of Brownian motions.

Contrary to the continuous nature of these models, data in finance and economics are mostly observed in a discrete time epochs. As such, despite the model process evolving continuously in time, we conduct the analysis of such models in the present context on discrete/finite time scales so as to interface discretely observed real-world processes with continuous time models. Indeed, a field of interest where diffusion models often find application is that of the modeling of (discretely observed) interest rate processes. The analysis of short rates is of major interest in the economic and fixed income environment. The dynamics of economic variables, such as price indices, exchange rates, stock indices and volatility indices; and fixed income variables, such as yields and short rates are often modeled with great success by diffusion models (in both the univariate and multivariate case). The Heston model, Ornstein Uhlenbeck (OU), Cox-Ingersoll-Ross (CIR), [7] and Black-Scholes model, to only name a few, is some of the most important models in the mentioned fields. In the present paper we focus on the analysis of the univariate and bivariate OU and CIR processes.

In the financial field of bond finance the yield curve is of fundamental importance. Given a yield curve is constructed by a sequence of interest rates, diffusion models can effectively be used as a modeling technique for the yield curve. One of the most popular and most used methods for fitting the yield curve is the Nelson-Siegel approach, as first introduced by [13]. In this methodology, optimization is used to estimate the parameters for the level, steepness and curvature of the yield curve. However, using diffusion processes to model yield curves have been an interesting field of research and provides new techniques to

produce promising results. The work of [5] indicates that diffusion models, can be used to estimate or fit a yield curve through the estimation of the sources of steepness, level and curvature of the yield curve, in which case the efficient method of moments (EMM) is utilized in to estimate the parameters and capture the mean drift and stochastic volatility in the given short rate diffusion. [14] modeled the yield curve by forecasting interest rates by the use of diffusion processes, more specifically CIR and Vascicek models. This methodology is based on a partitioning approach. Future interest rates are being forecasted, for each tenor of a given yield curve, based on partitioned financial market data. Diffusion processes provide valuable insights into the evolution of the trajectory of short-rates over time. In an extension, this evolution can often be attributed to an underlying process or processes, which can be modeled using a multivariate diffusion process. Modeling individual short-rates or tenors (with each tenor of the yield curve being an individual time-series of yields for a specific time to maturity, combining various tenors constitutes a yield curve), through univariate diffusion processes, may provide a decent fit to the yield curve, however an approach like this will not encapsulate underlying processes or the dependencies that the various maturities has on the other. Therefore, a multivariate diffusion process, modeling a certain tenor as the dependent process, and the most significant other tenor and/or other continuous rate may provide a more reliable estimate to that specific point on the yield curve, at a specific point in time. The latter will be showed in a multivariaty trajectory analysis of yield curve tenors.

The premise of the current paper is to model discretely observed interest rate time series using diffusion models, which replicate salient features of interest rate processes. In order to study how these processes evolve over time, the transition density function is analysed. However, an anlytical or closed-form solution to the true transition density rarely exists, and therefore this paper will focus on the approximation of this transition density function. Investigation into diffusion processes will include trajectory studies, Euler-Maruyama schemes and transition density analysis and approximation. The Hermite Approximation Method,[1, 4], will be compared to the Saddlepoint or Cumulant Truncation Approximation Method, [17, 21]. The theory and derivation of these approximate transition densities will be thoroughly explained. The Cumulant Truncation Approximation Method will provide the closed-form approximate density, which will be used for inference on financial data..

The paper is structured as follows; first a discussion on the fundamentals and dynamics of diffusion processes will be given, followed by the derivation and approximation of transition densities. The univariate case will be discussed, followed by the multivariate case, where examples will be provided at the end of each of the cases. The approximate densities will be applied for inferencial purposes on financial data. Finally , concluding remarks will be given, summarizing key findings.

2 Fundamentals of diffusion processes

For simplicity, the univariate instance will be explained prior to generalizing to the multivariate case.

2.1 Univariate difussion processes

[15], formally defines a diffusion process in Definition 1:

Definition 1. Consider a time dependent Markovian process,with state variable X_t, and transition density function $p(dy, t|x, s)$. dX_t is referred to as a diffusion process if, for all x with $\varepsilon > 0$:

1. $\int_{|x-y|>\epsilon} p(dy, t|x, s) = o(t-s)$ uniformly over $[s, t]$ for $s < t$, (i.e continuity),
2. there exists a real-valued function $\mu(x, t)$, (drift coefficient), s.t. $\int_{|x-y|\leq\epsilon} p(dy, t|x, s)(y-x) = \mu(x, t)(t-s) + o(t-s)$ uniformly over $[s, t]$ for $s < t$,
3. there exists a real-valued function $\sigma(x, t)$, (diffusion coefficient), s.t. $\int_{|x-y|\leq\epsilon} p(dy, t|x, s)(y-x)^2 = \sigma(x, t)(t-s) + o(t-s)$ uniformly over $[s, t]$ for $s < t$.

Lemma 2. *A real-valued function p(y) is o(y) if* $\lim_{y\to 0} \frac{p(y)}{y} = 0$

A transition density, is defined by [15]:

Definition 3. When considering a discrete state domain, the transition probability (density) of a shift, from $X_s = x$ to $X_t = y$ is defined as:

$\Pr(X_t = y|X_s = x) = p_{xy}(s, t) = p(y, t|x, s)$ in the time-inhomogeneous case (time-dependent), and

$\Pr(X_t = y|X_s = x) = \Pr(X_{t+j} = y|X_{s+j} = x) = p_{xy}(t-s)$ for all j, in the time homogeneous case (independent of shifts in time).

Where the probability measure function, $\Pr(.)$ is defined as:

Definition 4. $\Pr(.)$ is referred to as a probability measure function if a mapping of ψ is made into $[0, 1]$, s.t:

1. $\Pr(\Omega) = 1$, for Ω the non-empty set of all attainable elements,
2. $\Pr(A) \geq 0 \ \forall \ A \in \psi$,
3. if $A_1, A_2, A_3, ... \in \psi$. is a sequence of mutually disjoint subsets, then $\Pr\left[\bigcup_{i=1}^{\infty} A_i\right] = \sum_{i=1}^{\infty} \Pr(A_i)$.

Definition 5. Revisiting Equation 1, which will serve as the official notation in this papers.

$$dX_t = \mu(X_t, t; \boldsymbol{\theta})dt + \sigma(X_t, t; \boldsymbol{\theta})dW_t, \tag{2}$$

s.t $t \in [s, T]$, with $\boldsymbol{\theta}$ the parameter vector. Therefore, a general diffusion process, is defined by a stochastic differential equation (SDE), and inherits the Markov property, [17]. The diffusion process inherits both a deterministic and stochastic nature, given by the drift coefficient, $\mu(X_t, t; \boldsymbol{\theta})$, and diffusion coefficient $\mu(X_t, t; \boldsymbol{\theta})$, respectively. Randomness, or the stochastic nature is driven by a sequence of Brownian motions - W_t, s.t. W_t, $t \geq 0$ [1], (seen as the continuous counterpart of a Random Walk), which is defined as such:

Definition 6. [8] defines$W_{t\geq 0} \in \mathbb{R}$as a Brownian motion or Wiener Process if:

- for $t_0 < t_1 < t_2 < ... < t_{n-1} < t_n$ then W_{t_0}, $W_{t_1} - W_{t_0}, W_{t_2} - W_{t_1}$, ..., $W_{t_n} - W_{t_{n-1}}$ are independent,
- for $s, t \geq 0$, and A a subset of a $\sigma - algebra$ (please. see Definition 7), say ψ, then $W_{t+s} - W_s \sim N(0, t)$, i.e $\Pr(W_{t+s} - W_s \in A) = (2\pi t)^{-1/2} \int_A e^{-\frac{y^2}{2t}} dy$,
- $\Pr(t \to W_t \, is \, continuous) = 1$, and
- $W_{t=0} = W_0 = 0$.

Definition 7. A collection of subsets, say ψ, of Ω, is called a $\sigma - algebra/$ $-field$ if:

- for $\varnothing$ - the empty set - $\varnothing \in \Omega$,
- $A \in \psi$ then $A^C \in \psi$,
- for a sequence of sets $A_1, A_2, A_3, ... \in \psi$, then $\bigcup_{i=1}^{\infty} A_i \in \psi$.

In the present context, consider the time-space or time-domain $[s, T]$ s.t $t \in [s, T]$. An single occurrence (ς) of the sample path of the state space (Ω), is given by $X_t(\varsigma) \in \mathbb{R}$ s.t $\varsigma \in \Omega$. Since the sample path followed is assumed to be known the single occurrence can be denoted as $X_t(\varsigma)$.

A realisation of the process manifests as a solution to the SDE in Equation 2, via the integration of dX_t with respect to t, which through the trajectory of W_t, and the drift and diffusion coefficients gives rise to the trajectory of the process. . [17], explains that diffusion processes, such as in Equation 2, can be expressed through differential equations, where the deterministic part of the diffusion process is governed by the change in time, dt, and where the stochastic/random part of the diffusion process is governed by the change in a Wiener Process, dW_t. In consequence, the drift and diffusion coefficients can be defined as, [12]:

$$\mu(X_t, t; \boldsymbol{\theta}) = \lim_{\delta \to 0} \mathbb{E}\left[\frac{(X_{t+\delta} - X_t)|X_t}{\delta}\right],$$

and

$$\sigma^2(X_t, t; \boldsymbol{\theta}) = \lim_{\delta \to 0} \mathbb{E}\left[\frac{(X_{t+\delta} - X_t)^2|X_t}{\delta}\right],$$

where $\mathbb{E}(.)$ denotes the expected value.

Therefore, Equation 2 can be viewed as:

$$dX_t = \lim_{\delta \to 0} \mathbb{E}\left[\frac{(X_{t+\delta} - X_t)|X_t}{\delta}\right] dt + \sqrt{\lim_{\delta \to 0} \mathbb{E}\left[\frac{(X_{t+\delta} - X_t)^2|X_t}{\delta}\right]} dW_t.$$

Therefore the instantaneous variance of the process, is given by the diffusion coefficient. The movement in the process, are dictated by the drift coefficient, with the intensity of the stochastic movements determined by the diffusion coefficient. The randomness are reflected by the Brownian Motion. Solving Equation 2, over an appropriate transition horizon, yields the trajectory of the stochastic process X_t, provided the process started in the initial known state of X_s at time $s < t$. The solution can be obtained, by applying Ito's lemma (Lemma 9), through integration:

$$X_t = X_s + \int_s^t \mu(X_\tau, \tau; \boldsymbol{\theta}) d\tau + \int_s^t \sigma(X_\tau, \tau; \boldsymbol{\theta}) dW_\tau. \tag{3}$$

The drift and diffusion coefficients determine the existence of a solution to Definition 9. X_t in Equation 2 has a unit diffusion if the diffusion-coefficient is the unit diffusion, that is $\sigma(X_t, t; \boldsymbol{\theta}) = 1$. *Itô* Calculus, used in the trajectory derivation, is explained by the following results, combined from [20], [11] and [19].

Theorem 8. *Define p to be a continuous function, where $p(W_\tau)$ is the stochastic process, explained through Brownian motion (or the Wiener Process).*

If $\int_0^t \mathbb{E}[p(W_\tau)^2] d\tau < \infty$, for $\tau \in [0, t]$, then:

1. $\mathbb{E}\Big(\int_0^t p(W_\tau) dW_\tau\Big) = 0$,
2. $\mathbb{E}\Big(\Big|\int_0^t p(W_\tau) dW_\tau\Big|^2\Big) = \int_0^t \Big[\mathbb{E}|p(W_\tau)|^2\Big] dW_\tau$.

An *Itô* process can now be defined.

Definition 9. a stochastic process, X_t,which can be denoted in the following form, is known as an *Itô* process

$$X_t = X_0 + \int_0^t \Phi_\tau d\tau + \int_0^t \Theta_\tau dW_\tau, \tag{4}$$

where $\Phi$and Θ are driven by Brownian motion s.t

$$\int_0^t |\Phi_\tau|^2 d\tau < \infty,$$

and

$$\int_0^t \mathbb{E}[\Theta_\tau^2] d\tau < \infty.$$

When considering integrate function $g : \mathbb{R} \mapsto \mathbb{R}$, $G : \mathbb{R} \mapsto \mathbb{R}$ is defined as

$$G(t) = G(0) + \int_0^t g(s)d\tau, \tag{5}$$

differentiating over the time horizon yields

$$\tfrac{d}{dt}G(t) = \tfrac{d}{dt}G(0) + \tfrac{d}{dt}\textstyle\int_0^t g(s)d\tau s,$$

leads the following direct implication of the Fundamental Theorem of Calculus

$$\tfrac{d}{dt}G(t) = g(t), \text{ or}$$

$$G'(t) = g(t).$$

Which can be written in the standard SDE diffusion process form:

$$dX_t = \Phi_t dt + \Theta_t dW_t. \tag{6}$$

Itô's lemma naturally follows:

Theorem 10. *Itô's lemma: for a Brownian motion, $W_t \in \mathbb{R}$, where $t \in [0, T]$ with real-valued and twice differentiable function $g(x)$, it follows that:*

$$f(W_t) = f(0) + \frac{1}{2}\int_0^t f"(W_s)ds + \int_0^t f'(W_s)dW_s.$$

Lemma 11. *Itô's lemma in standard SDE notation: consider an Itô process - X_t ,with stochastic differential equation:*

$$dX_t = \mu_t dt + \sigma_t dW_t. \tag{7}$$

Define $f(t, X_t) : \mathbb{R} \to \mathbb{R}$,and $t \geqslant 0$, $Z_t = f(t, X_t)$, then

$$dZ_t = \frac{\partial}{\partial t}f(t, X_t)dt + \frac{\partial}{\partial X_t}f(t, X_t)dX_t + \frac{1}{2}\frac{\partial^2}{\partial X_t^2}f(t, X_t)(dX_t)^2. \tag{8}$$

Substituting Equation 7 into Equation 8:

$$dZ_t = \left(\frac{\partial}{\partial t}f(t, X_t) + \frac{\partial}{\partial t}f(t, X_t)\mu_t + \frac{1}{2}\frac{\partial^2}{\partial X_t^2}f(t, X_t)\sigma_t^2\right)dt + \frac{\partial}{\partial X_t}f(t, X_t)\sigma_t dW_t. \tag{9}$$

The following Examples will display the dynamics of two of the most acclaimed diffusion models in Finance and Economics, [1, 7], that is the univariate Ornstein-Uhlenbeck (OU) and Cox, Ingersoll and Ross (CIR) diffusion processes.

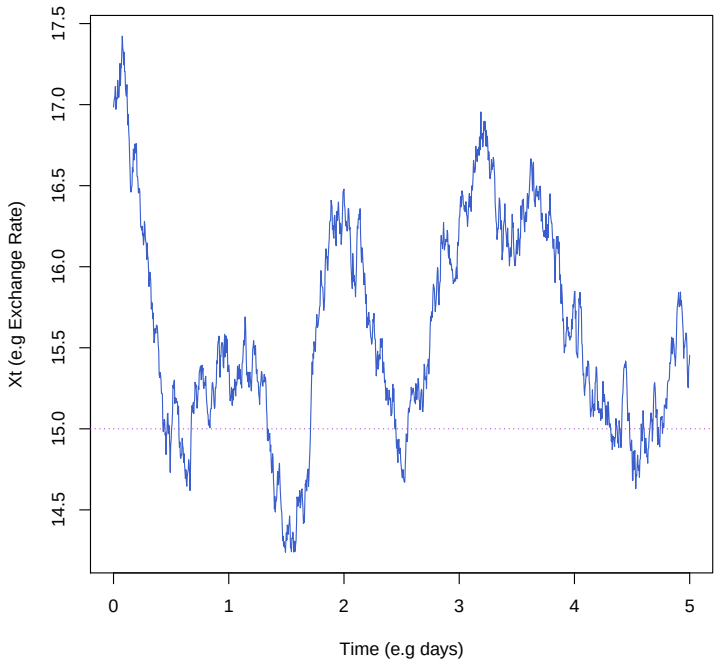


Figure 1: one simulated trajectory for a univariate OU process.

2.1.1 Examples of univariate diffusion processes

Example 12. Consider the univariate Ornstein Uhlenbeck diffusion process:

$$dX_t = \kappa(\alpha - X_t)dt + \sigma dW_t, \tag{10}$$

s.t. $t \in [s, T]$, and with W_t Brownian Motion, as in Definition 6. The parameter vector $\boldsymbol{\theta} = (\kappa, \alpha, \sigma)$, consists of κ, α, σ the deterministic parameters, where α refers to the mean reversion level (that is $\lim_{t\to\infty} X_t = \alpha$), κ the tempo of reversion and σ the volatility factor. In terms of Equation 2, the drift and diffusion coefficients are given by $\mu(X_t, t; \boldsymbol{\theta}) = \kappa(\alpha - X_t)$ and $\sigma(X_t, t; \boldsymbol{\theta}) = \sigma$, respectively. From the latter it can be seen that the diffusion coefficient is constant and independent of the value of the process. The OU model is a simple but effective model, regularly adopted in finance, especially in short rate modeling

Figure 2 illustrates the empirical distribution (histogram) of the process, constructed using simulated trajectories under the .Euler Maruyama scheme.
Performing the simulation study on $t \in [0, 5]$, $X_t \in [12, 19]$, $\boldsymbol{\theta} = (\kappa, \alpha, \sigma) = (0.85, 15, 0.75)$, with $X_0 = 16$, and $stepsize = 1/250$ (approximate number of annual trading days) . Figure 1 displays one simulated trajectory for a univariate OU process.

The effect and sensitivity of changing a single parameter value, whilst keeping all else equal, are given in Figure 3. As can be seen, changing α changes the mean revering level, where the speed a reversion in adjusted through changing κ, and the larger σ becomes, the more volatile the model becomes. Finally,

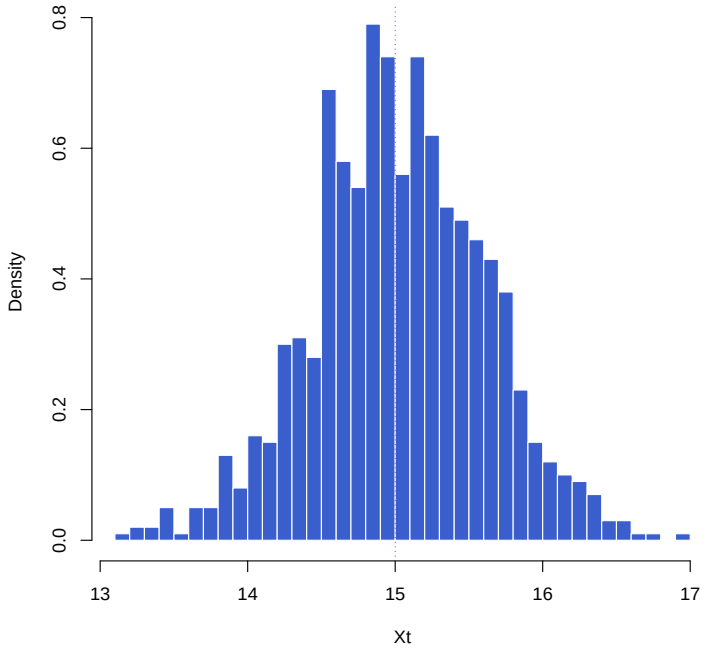


Figure 2: histogram (through the Euler Maruyama shceme) of the simulated process trajectories (univariate OU model).

the process can tend to a negative value if the mean reversion parameter - α - is negative. Please see Algorithms 1 and 4.

Often the volatility of a process (depicted through the diffusion coefficient) is dependent on the value of the process itself. Since diffusion coefficient is constant and independent of the value of the process in the OU model, the CIR process will be considered next.

Example 13. Univariate Cox, Ingersoll and Ross (CIR) diffusion process:
Define the well-known CIR model, as in , as in [7, 1], through the SDE :

$$dX_t = \kappa(\alpha - X_t)dt + \sigma\sqrt{X_t}dW_t, \tag{11}$$

s.t. $t \in [s, T]$, with W_t denoting a standard Brownian motion, as in Definition 6 The parameter vector $\boldsymbol{\theta} = (\kappa, \alpha, \sigma)$, consists of κ, α, σ the deterministic parameters, where α refers to the mean reversion level (that is $\lim_{t\to\infty} X_t = \alpha$), κ the tempo of reversion and σ the volatility factor . In terms of Equation 2, the drift and diffusion coefficients are given by $\mu(X_t, t; \boldsymbol{\theta}) = \kappa(\alpha - X_t)$ and $\sigma(X_t, t; \boldsymbol{\theta}) = \sigma\sqrt{X_t}$,respectively. The CIR model is an excellent model for capturing short rate dynamics, given the short rate is positive at all times. However, given that international Financial Markets have moved into negative interest rate environments in some domiciles (e.g European Fixed Income government bond yields), the drawback of the model becomes evident. Nonetheless, with most short rates still being non-negative, the CIR model is

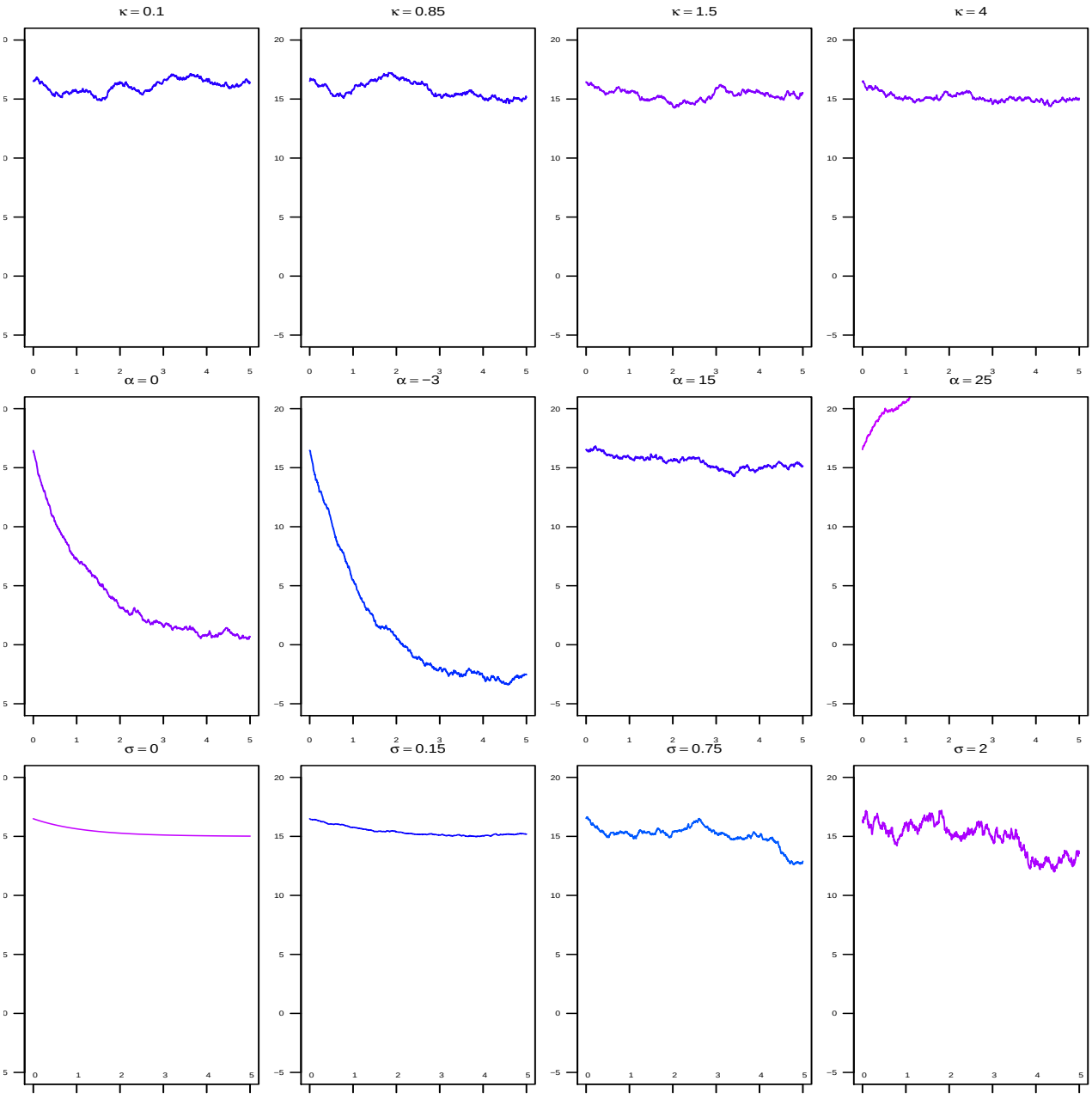


Figure 3: The effect and sensitivity of changing a single parameter value, whilst keeping all else equal (univariate OU model).

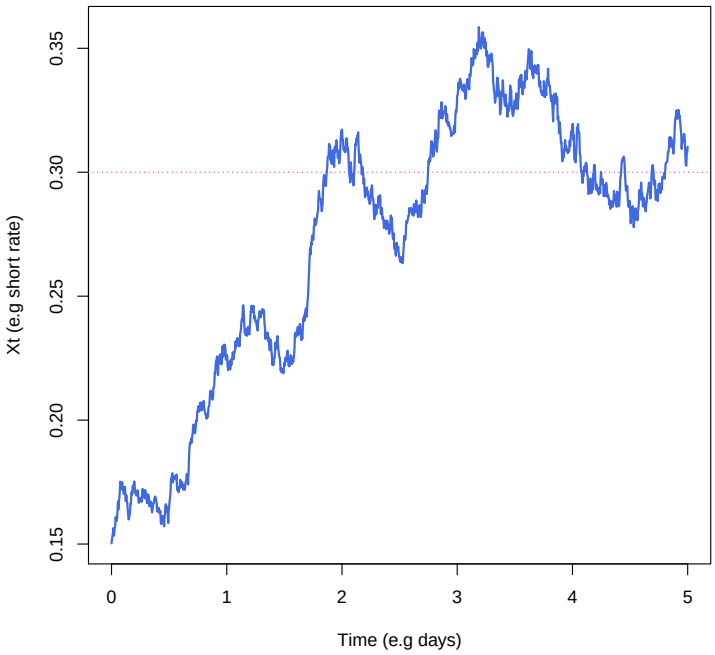


Figure 4: one simulated trajectory for a univariate CIR process.

still worth analyzing and implementing in financial and economic problems. According to [7], for $\kappa, \alpha > 0$, the defined process is equivalents to a continuous $Autoregressive(1)$ model. Since $\lim_{t\to\infty} X_t|\sigma^2 > 2\kappa\alpha = 0$ the parameters are constrained as $\sigma^2 \leq 2\kappa\alpha$ to ensure $\lim_{t\to\infty} X_t = \alpha$, (provided $\alpha > 0$), s.t. $t \in [s, T]$ for all $s \geq 0$. If $X_s > 0$ then $X_t > 0$ for all $t \geq 0$. Therefore X_t will then eventually settle in a steady state.The $\lim_{t\to\infty} X_t|\sigma^2 > 2\kappa\alpha = 0$ makes practical sense when considering an actively traded asset, it the assets volatility is in well in excess of $2\kappa\alpha$, this will eventually lead to a sell-off the asset, i.e the price and hence X_t reaching 0.

Performing the simulation study on $t \in [0, 5]$, $X_t \in [0, 1]$,$\boldsymbol{\theta} = (\kappa, \alpha, \sigma) = (0.9, 0.3, 0.075)$, with $X_0 = 0.15$, and $stepsize = 1/250$ (proximate number of annual trading days). Note that $2\kappa\alpha = 0.27 > \sigma^2 = 0.005625$. Figure 4 displays one simulated trajectory for the CIR process, the dotted reference line indicates the mean reverted level, i.e $\alpha = 0.3$. Figure 5 gives the first view of a distribution for X_t, that is due to a histogram (through the Euler Maruyama scheme which will be discussed later in the paper) of the simulated process trajectories.

The effect and sensitivity of changing a single parameter value, whilst keeping all else equal, are given in Figure 6. As can be seen, changing α changes the mean revering level, where the speed a reversion in adjusted throuh changing κ, and the larger σ becomes, the more volatile the model becomes. Lastly, no change in parameter leads to the X_t becoming negative, as expected through the model definition. Please see Algorithms 2 and 4.

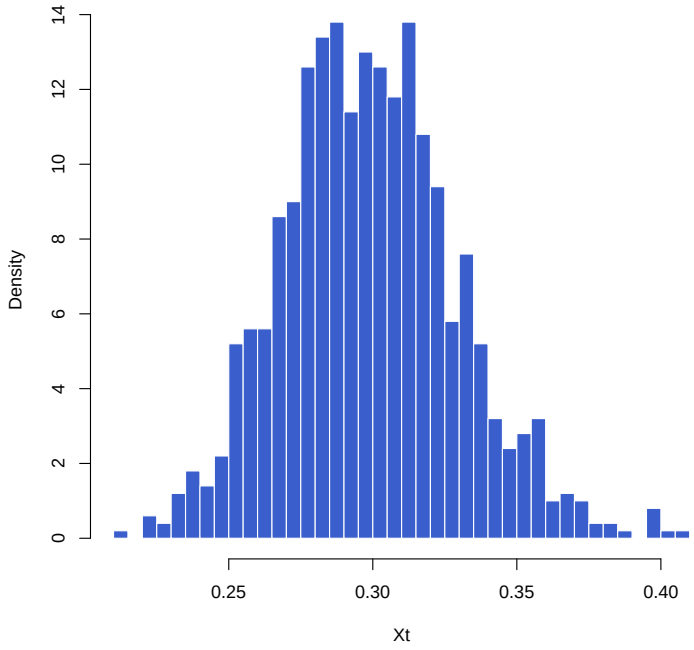


Figure 5: histogram (through the Euler Maruyama shceme) of the simulated process trajectories (univariate CIR model).

Lastly we will consider the famous univariate Black-Scholes model, which is synonymous with option pricing:

Example 14. Black and Scholes diffusion process:

Define the option pricing Black-Scholes (BS) model as:

$$dX_t = \frac{1}{2}\phi^2 X_t dt + \phi X_t dW_t, \tag{12}$$

s.t. $t \in [s, T]$, and with W_t Brownian Motion, as in Definition 6 The parameter ϕ is the key deterministic factor of the model. In terms of Equation 2, the drift and diffusion coefficients are given by $\mu(X_t, t; \boldsymbol{\theta}) = \frac{1}{2}\phi^2 X_t$ and $\sigma(X_t, t; \boldsymbol{\theta}) = \phi X_t$,respectively. Therefore both the drift and diffusion coefficients is dependent on the value of the process. The model is very sensitive to changes in ϕ, and tends to diverge quickly. Figure 7, shows the divergent nature of the process, through the simulation of 100 trajectories.

Performing the simulation study on $t \in [0, 1]$, $X_t \in [0, \infty)$,$\phi = 0.25$, with $X_0 = 1$, and $stepsize = 1/250$ (approximate number of annual trading days) .Please see Algorithms 1.

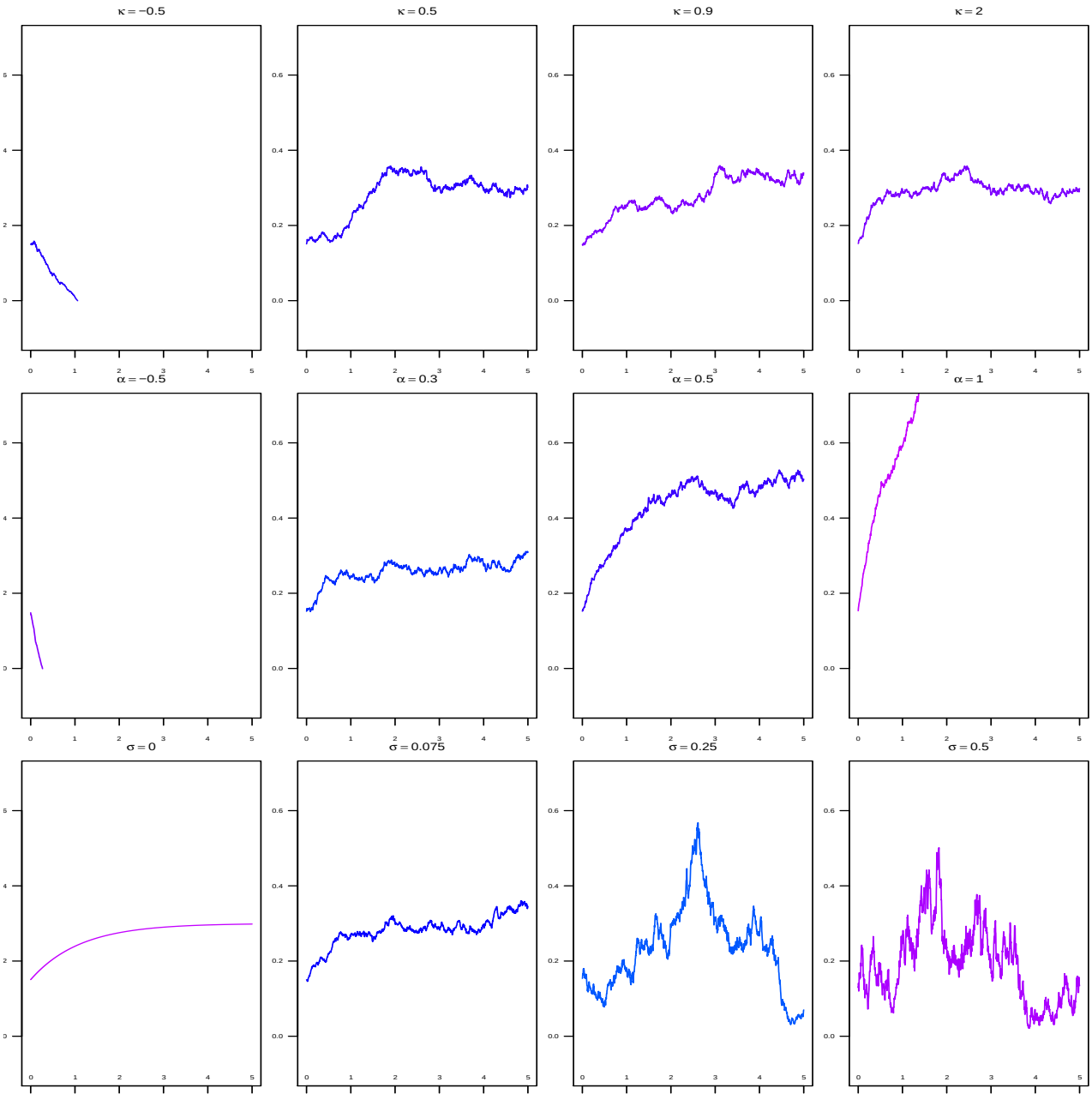


Figure 6: effect and sensitivity of changing a single parameter value, whilst keeping all else equal (univariate CIR model).

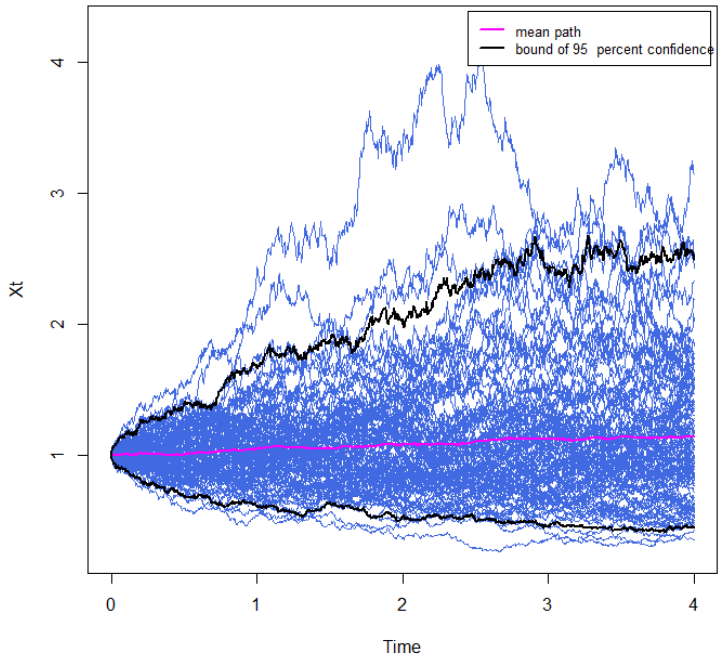


Figure 7: 100 simulated trajectory for a univariate BS process.

2.2 Multivariate diffusion processes

The dynamics of a general multivariate diffusion process , $\boldsymbol{X}_t$, can be defined by the stochastic differential equation:

$$d\boldsymbol{X}_t = \boldsymbol{\mu}(\boldsymbol{X}_t,t;\boldsymbol{\Theta})dt + \boldsymbol{\sigma}(\boldsymbol{X}_t,t;\boldsymbol{\Theta})d\boldsymbol{W}_t, \tag{13}$$

s.t. $\boldsymbol{X}_t : n \times 1$ is the state vector, $\boldsymbol{\mu}(\boldsymbol{X_t}, \boldsymbol{t}) : n \times 1$ the drift vector, and $\boldsymbol{\Sigma}(\boldsymbol{X_t}, \boldsymbol{t}) : n \times n$ the diffusion matrix of the matrix, [17]. $\boldsymbol{\Theta}$denotes the parameter space. The instantaneous covariance matrix of the process is defined as

$$\boldsymbol{\sigma}(\boldsymbol{X}_t,t)\boldsymbol{\sigma}(\boldsymbol{X}_t,t)^T = \boldsymbol{\gamma}(\boldsymbol{X}_t,t) : n \times n,$$

s.t $\boldsymbol{\sigma}^T = transpose(\boldsymbol{\sigma}), \boldsymbol{W}_t : n \times 1$ denotes the vector of Brownian Motions, as defined in 6. The dynamics of the process in Equation 13 is therefore dictated by stochastic elements, through the diffusion coefficient, as well as deterministic elements, via the drift coefficient. These coefficients can be expressed in terms of the instantaneous moments of the system:

$$\boldsymbol{\mu}_i(\boldsymbol{X_t},t) = \lim_{\delta \to 0} \frac{\mathbb{E}\left[\boldsymbol{X}^{(i)}_{t+h} - \boldsymbol{X}^{(i)}_t \mid \boldsymbol{X_t}\right]}{\delta} \tag{14}$$

and

$$\boldsymbol{\gamma}_{ij}(\boldsymbol{X_t},t) = \lim_{\delta \to 0} \frac{\mathbb{E}\Big[\big(\boldsymbol{X}^{(i)}_{t+\delta} - \boldsymbol{X}^{(i)}_{t}\big)\big(\boldsymbol{X}^{(j)}_{t+\delta} - \boldsymbol{X}^{(j)}_{t}\big)|\ \boldsymbol{X_t}\Big]}{\delta}, \forall\, i,j = 1,2,...,k. \tag{15}$$

The solution to Equation 13, is quite often focal point in the analysis of diffusion processes. The trajectory, at time t, of the general diffusion process, $\boldsymbol{X_t}$, at a known initial state of $\boldsymbol{X_s}$, s.t. $s < t$, is provided by the following equation, through the implementation of Ito calculus:

$$\boldsymbol{X}_t = \boldsymbol{X}_s + \int_s^t \mu(\boldsymbol{X}_v, v)dv + \int_s^t \sigma(\boldsymbol{X}_v, v)d\boldsymbol{W}_v \tag{16}$$

For the premise of this paper it is assumed that the initial value of the process, under consideration, and therefore the initial distribution, is always known. It is also assumed that the drift and diffusion coefficients of are Lipschitz continuous, and therefore satisfies the accompanied linear growth conditions. This implies continuity in the coefficients and that the coefficients inherit unbounded rates of change. These assumptions ensure the existence of a solution to Equation 13, which is a fundamental requirement for the methods to be considered in this paper. The distribution of the state vector, $\boldsymbol{X_t}$ is of absolute importance in the analysis of the dynamics of this stochastically driven process.

Definition 15. $(\Omega, \xi, \Pr)$ is is referred to as a probability space, if

1. A non-empty set, Ω, called the sample space, exists which includes all possible outcomes,
2. ξ is a $\sigma - algebra$, consisting of subsets of Ω, and
3. Pr is probability measure function on (Ω, ξ), s.t. (Ω, ξ) is a measurable space.

Let $(\Omega, \xi, \Pr)$ be a as a probability space, as per Definition 15, where $\Omega \subseteq \mathbb{R}^k$ is the sample space, ξ the $\sigma - algebra$ of subsets of Ω,as per Definition 7, and Pr denotes the probability measures of events in the $\sigma - algebra$ ξ. As per [4], the probability density function, of moving from states $\boldsymbol{X_s}$ to $\boldsymbol{X_t}$, denoted by $g(\boldsymbol{X_t}|\boldsymbol{X_s})$,is obtained by solving the Kolmogorov forward equation (or Fokker-Planck equation), [17]:

$$\begin{aligned}\frac{\partial}{\partial t} g(\boldsymbol{X_t}|\boldsymbol{X_s}) = &- \sum_{i=1}^{k} \frac{\partial}{\partial X^{(i)}_t} \left(\mu_i(\boldsymbol{X_t},t) g(\boldsymbol{X_t}|\boldsymbol{X_s})\right) \\ &+ \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{\partial^2}{\partial X^{(i)}_t \partial X^{(j)}_t} \left(\sigma_{ij}(\boldsymbol{X_t},t) g(\boldsymbol{X_t}|\boldsymbol{X_s})\right),\end{aligned} \tag{17}$$

where $X^{(r)}_t$denotes the r^{th} element of the process $\boldsymbol{X_t}$. The Dirac delta function, which specifies the initial conditions of the transition density, is defined by

$$g(\boldsymbol{x_s}|\boldsymbol{X_s}) = \Delta(\boldsymbol{x_s} - \boldsymbol{X_s}),$$

where $\Delta(.)$ is defined as

$$\Delta(\boldsymbol{x}) = \begin{cases} \boldsymbol{\infty} & \boldsymbol{if \;\; x = 0} \\ 0 & \boldsymbol{otherwise.} \end{cases}$$

The initial condition Dirac delta function implies that the dynamics of the general diffusion process, $\boldsymbol{X_t}$, are driven by drift vector $\boldsymbol{\mu}(\boldsymbol{X_t},t)$ and diffusion matrix $\boldsymbol{\sigma}(\boldsymbol{X_t},t)$. Consequently the shape of the transition density, $g(\boldsymbol{X_t}|\boldsymbol{X_s})$, are driven by the functional dynamics of the respective drift and diffusion functions. Since the transition density function yields important inferential traits, an analytical expression for this density function is paramount to this paper. However, to find a closed-form analytical solution to the true transition density is a difficult, and often impossible task.The main complication lies in expressing the dynamics of the process, probabalistically over finite transition horizons. When limiting the time-space to small time epochs, the transition density is multivariate Normal:

$$g(\boldsymbol{X_t}|\boldsymbol{X_s}) \backsim MV.NORM\Big(\boldsymbol{X_s} + \delta_{ts}\boldsymbol{\mu}(\boldsymbol{X_s}, \boldsymbol{s}), \delta_{ts}\boldsymbol{\sigma}(\boldsymbol{X_s}, \boldsymbol{s})\boldsymbol{\sigma}(\boldsymbol{X_s}, \boldsymbol{s})^T\Big),$$

with $\delta_{ts} = (t-s)$. Unfortunately, as the time horizon increase, the shape of the transition density diverges from the normal distribution.

To obtain an analytical expression for the transition density, allows for the identification of a closed-form likelihood function, which in tern can be maximized over a financial data set to find parameters of best fit. This allows for analysis, inference and forecasting/prediction on the particular dataset. Unfortunately, as indicated, closed-form theoretical analytical solutions to the true transition density rarely exist and warrants the investigation into consistent approximation techniques to obtain an expression for this approximate transition density function. Approximation methods, of particular interest in this paper, includes the Hermite Expansion" method, [3] and the Cumulant truncation method/Saddlepoint approximation method, [21] and [17]. The use of Monte Carlo and Euler simulations, where kernel density estimates can yield an approximate density, will also be briefly explored.

Example 16. Bivariate Cox Ingersoll and Ross (CIR) Diffusion Process applied to the South African Reserve Bank's Monetary Policy

Define the bivariate CIR model as, [16]:

$$\begin{aligned} dX_t &= (\alpha_1(\beta_1 - X_t) - \lambda_1 Y_t)dt + \sigma_1\sqrt{X_t}dW_t^{(1)} \\ dY_t &= (\alpha_2(\beta_2 - Y_t) - \lambda_2 X_t)dt + \sigma_2\sqrt{Y_t}dW_t^{(2)}, \end{aligned}$$

or equivalently:

$$d\boldsymbol{Z}_t = \begin{pmatrix} dX_t \\ dY_t \end{pmatrix} = \begin{pmatrix} (\alpha_1(\beta_1 - X_t) - \lambda_1 Y_t) \\ (\alpha_2(\beta_2 - Y_t) - \lambda_2 X_t) \end{pmatrix} dt + \begin{pmatrix} \sigma_1\sqrt{X_t} & 0 \\ 0 & \sigma_2\sqrt{Y_t} \end{pmatrix} \begin{pmatrix} dW_t^{(1)} \\ dW_t^{(2)} \end{pmatrix},$$

s.t. $t \in [s, T]$, with $s \geqslant 0$ and $X_t \in [X_s, X_T]$, $Y_t \in [Y_s, Y_T]$ and with $s \geqslant 0$, and $X_t, Y_t \geqslant 0$ for all t (due to $\sqrt{X_t}, \sqrt{Y_t}$ being in the Real space), and with $dW_t^{(i)} : i = 1, 2$ the Brownian Motions, as in Definition 6 The parameter space $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\sigma})$ consists of deterministic parameters $\{\alpha_{i=1,2}, \beta_{i=1,2}, \lambda_{ij=1,2}, \sigma_{i=1,2}\}$.Note $\sigma_{12} = \sigma_{21} = 0$ implying that the volatility sources $(dW_t^{(i)} : i = 1, 2)$ are independent . In terms of Equation 13, the drift and diffusion coefficients are given by

$$\boldsymbol{\mu}(\boldsymbol{Z_t}, t; \boldsymbol{\Theta}) = \begin{pmatrix} (\alpha_1(\beta_1 - X_t) - \lambda_1 Y_t) \\ (\alpha_2(\beta_2 - Y_t) - \lambda_2 X_t) \end{pmatrix}$$

and

$$\boldsymbol{\Sigma}(\boldsymbol{Z_t}, t; \boldsymbol{\Theta}) = \begin{pmatrix} \sigma_1\sqrt{X_t} & 0 \\ 0 & \sigma_2\sqrt{Y_t} \end{pmatrix}$$

respectively. Note that $\boldsymbol{Z}_t$ is defined as

$$\boldsymbol{Z}_t = \begin{pmatrix} X_t \\ Y_t \end{pmatrix}.$$

The South African Reserve Bank (SARB) are mandated to implement effective Monetary Policy to maintain price stability (to keep headline consumer price inflation between (CPI) 3% and 6%). The SARB performs this function, through monthly adjustments in the Repurchase (Repo) Rate. The adjustments are usually (but not constrained to) a binary move of of 25 basis points (0.25%). Economical theory dictates that an expansionary/hawkish monetary policy - reduction in the Repo Rate - will lead to expansionary consumer spending via credit cycles, resulting in future increased inflation. Similarly, an contractionary/dovish monetary policy - increase in the Repo Rate - will lead to slowed consumer spending, resulting in future decreased inflation. Therefore there is a strong relationship between the two variables. Through a simulation study, this relationship is displayed and in line with expectation. Performing the simulation study on $t \in [0, 24]$,$X_t \in [0, 5]$, $Y_t \in [0, 5]$; $\boldsymbol{\theta} = (\alpha_{i=1,2}, \beta_{i=1.2}, \lambda_{i=1.2}, \sigma_{i=12}) = (2, 2, 4.2, 6.75, 0.1, 0.1, 0.75, 0.5)$, with $X_0 = 3.8$ and $Y_0 = 6.5$, and $stepsize = 1/30$. Define

$$\boldsymbol{Z}_t = \begin{pmatrix} X_t = CPI \\ Y_t = Repo\ Rate \end{pmatrix}.$$

In Figure 8 it can be seen, that there is clear bias in movement, i.e, a hike in the Repo Rate leads to a

reduction in CPI in the coming months, and vice versa. The Repo Rate does not behave like a diffusion process, as it clearly changes discretely, due to the Central Bank's incremental 25 basis point (0.25%) changes. Smoothing the Repo rate would enable the series to behave as diffusion process and ennable further analysis. This is important to note, as in the market the data under review is not always in the ideal format, and transformation may be needed. Please see R code in Algorithm6.

3 Obtaining closed-form transition densities for diffusion processes

Having a closed-form true transition density for a diffusion process, simplifies the inference on diffusion processes significantly. However, very little of these models's true transition density can be obtained in closed-form. Therefore, to develop a suitable and accurate approximation technique to enable inference on a wider class of diffusion models, is of paramount interest.

3.1 Transition densities in the univariate state variable case

For this section, consider the general univariate diffusion process in Equation 2, i.e $dX_t = \mu(X_t, t; \boldsymbol{\theta})dt + \sigma(X_t, t; \boldsymbol{\theta})dW_t$.

3.2 Examples of diffusion processes with closed-form transition densities

Albeit rare, we are often privileged to obtain a true closed-form transition density for Equation 2. This true transition density,provided such exist, can be obtained through applying Ito calculus and solving the Kolmogorov Forward or Backward equation, [15]. Denote the true transition density by $p(x_t, t \mid x_s, s; \boldsymbol{\theta})$, $s \leq t$, the solution of following Kolmogorv equations:
Kolmogorov Forward Equation (KDE):

$$\frac{\partial}{\partial t}p(x_t, t \mid x_s, s; \boldsymbol{\theta}) = -\frac{\partial}{\partial x_t}[\mu(x_t, t; \boldsymbol{\theta})p(x_t, t \mid x_s, s; \boldsymbol{\theta})] + \frac{1}{2}\frac{\partial^2}{\partial x_t^2}[\sigma^2(x_t, t; \boldsymbol{\theta})p(x_t, t \mid x_s, s; \boldsymbol{\theta})], \tag{18}$$

and the Kolmogorov Backward Equation (KBE):

$$-\frac{\partial}{\partial s}p(x_t, t \mid x_s, s; \boldsymbol{\theta}) = -\mu(x_s, s; \boldsymbol{\theta})\frac{\partial}{\partial x_s}p(x_t, t \mid x_s, s; \boldsymbol{\theta}) + \frac{1}{2}\sigma^2(x_s, s; \boldsymbol{\theta})\frac{\partial^2}{\partial x_s^2}p(x_t, t \mid x_s, s; \boldsymbol{\theta}). \tag{19}$$

Example 17. Univariate Ornstein Uhlenbeck diffusion process true transition density.

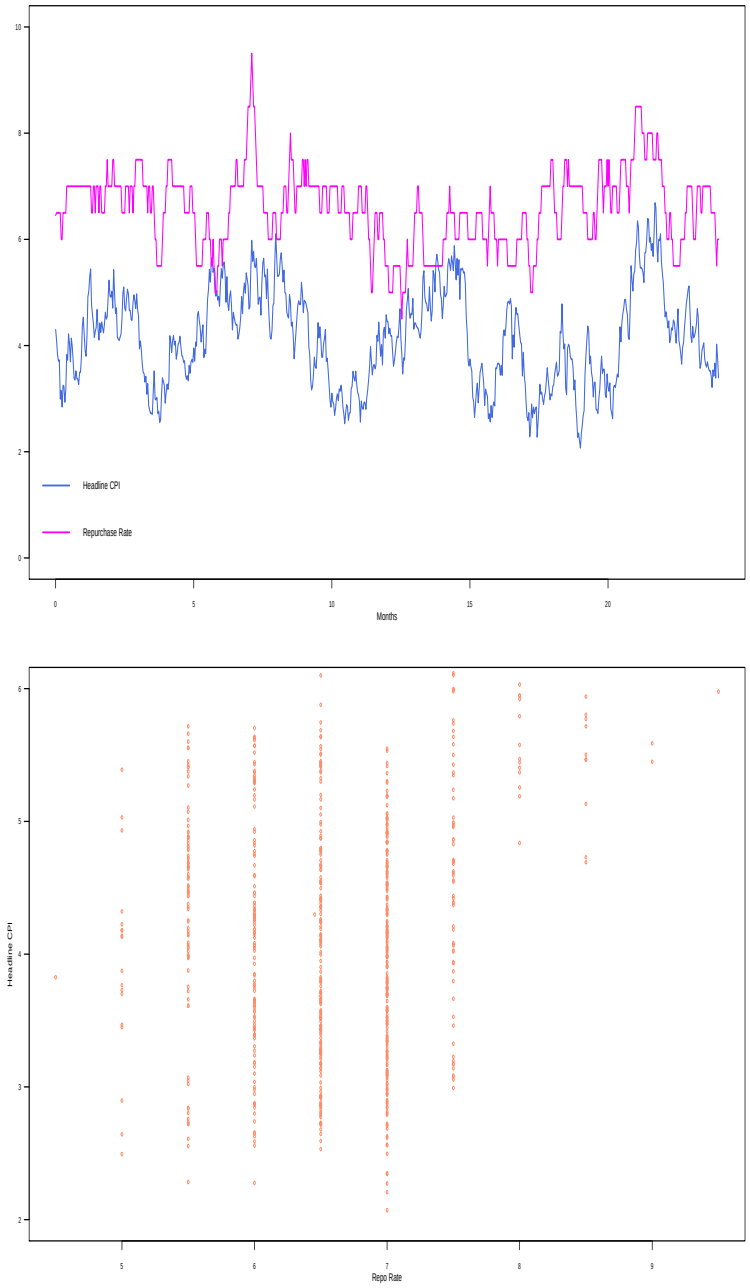


Figure 8: bivariate CIR model simulation study applied to the South African Reserve Bank's monetary policy implementation mechanism.

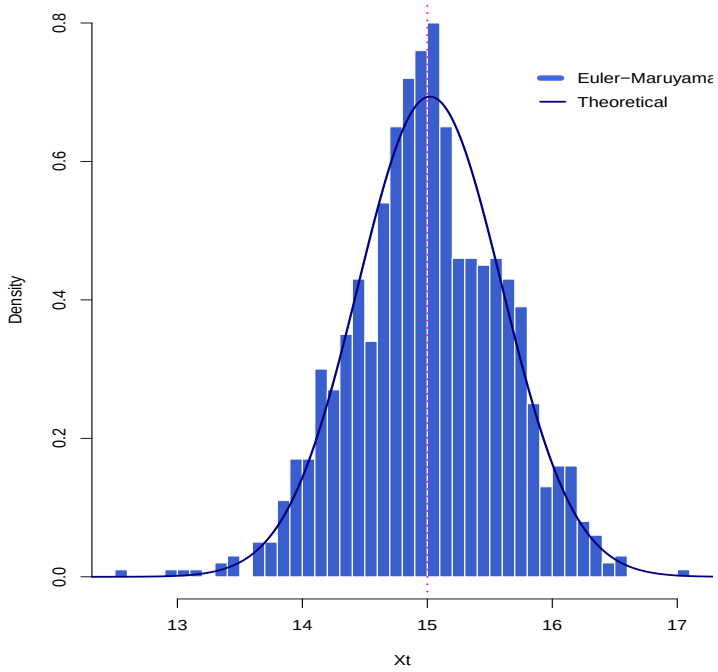


Figure 9: true transition density of the univariate OU process, overlay-ed on the Euler-Maruyama scheme.

Consider the defined OU model, and parameter values, as in Example 12, that is

$$dX_t = \kappa(\alpha - X_t)dt + \sigma dW_t, \tag{20}$$

s.t. $t \in [s, T]$, with $s \geqslant 0$ and $X_s \geqslant 0$.

Solving the KDE

$$\frac{\partial}{\partial t}p(x_t, t \mid x_s, s; \boldsymbol{\theta}) = -\frac{\partial}{\partial x_t}[\kappa(\alpha - x_t)p(x_t, t \mid x_s, s; \boldsymbol{\theta})] + \frac{1}{2}\frac{\partial^2}{\partial x_t^2}[\sigma^2 p(x_t, t \mid x_s, s; \boldsymbol{\theta})],$$

yields the true (and Gaussian) transition density for the univariate OU process, X_t, [1] :

$$p(x_t, t \mid x_s, s; \boldsymbol{\theta}) = \left(\frac{\pi\gamma^2}{\kappa}\right)^{-\frac{1}{2}} \exp\Big(-(\kappa/\gamma^2)((x_t - \alpha) - (x_s - \alpha)\exp(-\kappa\delta))^2\Big),$$

s.t.

$$\gamma = \Big(\sigma^2(1 - \exp(-2\kappa\delta))\Big)^{-1/2},$$

with δ increment in time. The true transition density is plotted in Algorithm 1 and displayed in Figure 9.

Example 18. Univariate CIR process true transition density.

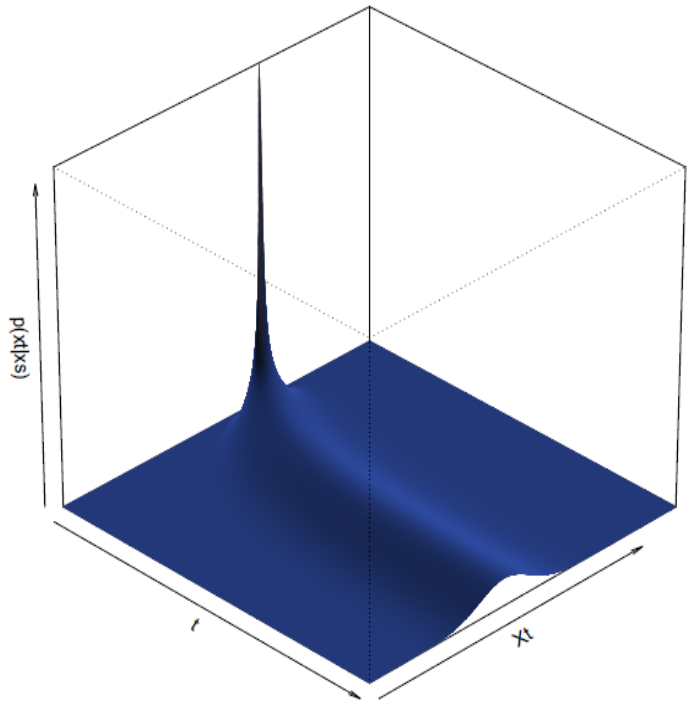


Figure 10: true transition density perspective plot of the univariate OU process,

Consider the CIR model, as in Example 13, [7, 1], with the SDE :

$$dX_t = \kappa(\alpha - X_t)dt + \sigma\sqrt{X_t}dW_t, \tag{21}$$

s.t. $t \in [s, T]$, with $s \geqslant 0$ and$X_t \geqslant 0$ for all t (due to $\sqrt{X_t}$ being in the Real space).

Solving the KDE

$$\frac{\partial}{\partial t}p(x_t, t \mid x_s, s; \boldsymbol{\theta}) = -\frac{\partial}{\partial x_t}[\kappa(\alpha - x_t)p(x_t, t \mid x_s, s; \boldsymbol{\theta})] + \frac{1}{2}\frac{\partial^2}{\partial x_t^2}[\sigma\sqrt{x_t}p(x_t, t \mid x_s, s; \boldsymbol{\theta})],$$

yields the true transition density for the univariate CIR process, X_t, [1] :

$$p(x_t, t \mid x_s, s; \boldsymbol{\theta}) = c\exp(-(u+v))\left(\frac{v}{u}\right)^{\frac{q}{2}} I_q(2(uv)^{\frac{1}{2}}), \tag{22}$$

where $s < t$, $c = \frac{2\kappa}{\sigma^2(1-\exp(-\kappa\delta))}$, $u = cx_s\exp(-\kappa\delta)$, $\delta = t - s$, $v = cx_t$ and $q = \frac{2\kappa\alpha}{\sigma^2} - 1$. Denote $I_q(2(uv)^{1/2})$ as a modified Bessel function of the 1^{st} kind and of the q^{th} order, with dynamics given by the following ODE:

$$t^2\frac{\partial^2}{\partial x_t^2}p(x_t, t \mid x_s, s; \boldsymbol{\theta})' + t\frac{\partial}{\partial x_t}p(x_t, t \mid x_s, s; \boldsymbol{\theta}) = (t^2 + q^2)x_t,$$

solved, as in [22], yields

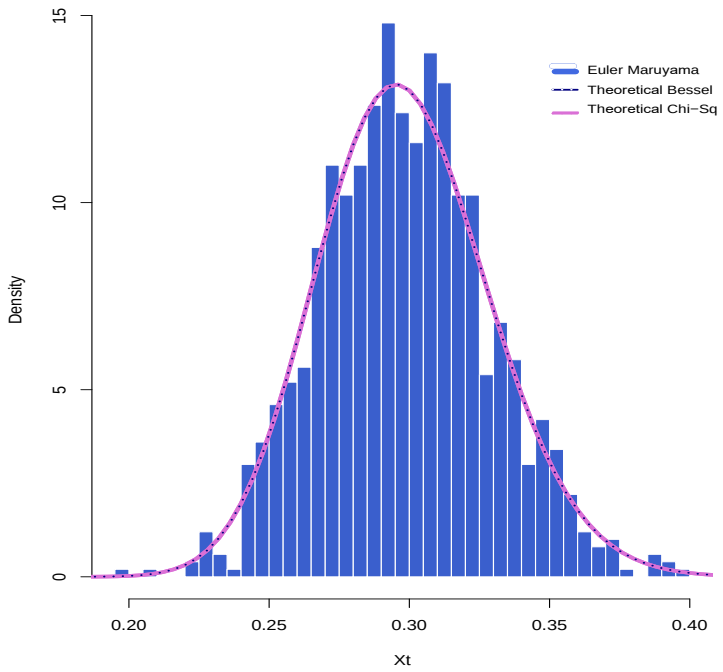


Figure 11: true transition density of the univariate CIR process, overlayed on the Euler-Maruyama scheme.

$$I_q(2(uv)^{1/2}) = (uv)^{q/2} \sum_{k=0}^{\infty} \left[\frac{1}{\Gamma(q+k+1)\Gamma(k+1)} (uv)^k \right],$$

with gamma function, $\Gamma(n) = (n-1)!$.

Alternatively [7], X_t is non-centrally Chi-squared distributed:

$$2cX_t \sim \chi^2(2(q+1), 2u),$$

The true transition density is plotted in Algorithm 2 and displayed in Figure 9.

3.3 Transition density approximation

The most important aspect of the given paper will now be discussed, that is the development a closed-form transition density approximation. That is, to obtain $\tilde{p}(x_t, t \mid x_s, s; \boldsymbol{\theta}) \approx p(x_t, t \mid x_s, s; \boldsymbol{\theta})$, with precision and generality. The methods to be discussed include the Euler-Maruyama (EM), [11], the Hermite series expansion, [4], and the Cumulant Truncation / Saddlepoint, [21, 17] transition density approximation methods.

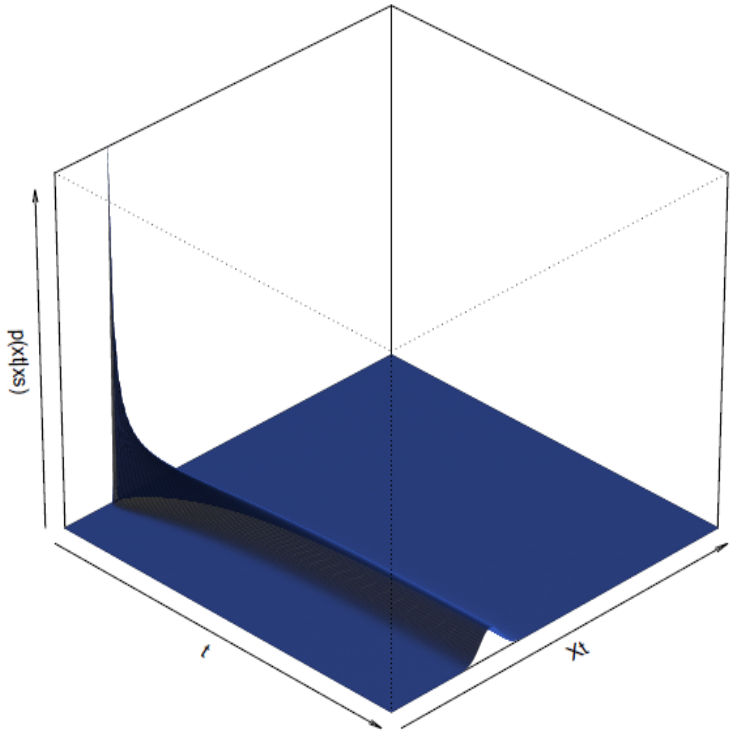


Figure 12: true transition density perspective plot of the univariate CIR process,

3.3.1 Euler-Maruyama

The EM transition density approximation methods, [11], finds numerical solutions for a process, as in Equation 2, by means of recursive simulation of the governing SDE. The method yields a distribution on which a kernel density can be fit to obtain a closed-form approximate transition density.

Consider the time-space $[s, T]$, define $\delta = \frac{t-s}{M}$ and $\varepsilon_i = i\delta$. Let $\tilde{X}_i$be a numerical approximation to $X(\varepsilon_i)$. Recursively the sequence of approximations is obtained

$$\tilde{X}_i = \tilde{X}_{i-1} + \mu(\tilde{X}_{i-1}, i-1; \boldsymbol{\theta})\delta + \sigma(\tilde{X}_{i-1}, i-1; \boldsymbol{\theta})(W_{\varepsilon_i} - W_{\varepsilon_{i-1}}), \tag{23}$$

for all $i = s+1, s+2, ..., s+M$, $\tilde{X}_s = X_s$ as initial condition.

As examples, the EM approximate distribution for the univariate OU and CIR process is displayed in Figures 9 and 11 and coded in Algorithms 2 and 1.

3.3.2 Hermite-series transition density approximation

[1], developed a approximation technique, based on Hermite series expansions, which provides a closed-from analytical expression for the process's transition density.

The Hermite transition density approximation, is developed in orders of approximation, i.e. $\tilde{p}_X^{(K)}(x_t, t|x_s, s; \boldsymbol{\theta})$, where $K \geq 0$, s.t $\tilde{p}_X^{(K)}(x_t, t|x_s, s; \boldsymbol{\theta})$ is the k^{th} order approximation of $p_X(x_t, t|x_s, s; \boldsymbol{\theta})$. Before an approx-

imation to the transition density for the process at hand can be obtained, i.e $\tilde{p}_X^{(K)}(x_t, t|x_s, s; \boldsymbol{\theta})$, a Hermite approximation for the transformed unit diffusion (where the diffusion coefficient is the unit diffusion, i.e. $\sigma(\gamma, t; \boldsymbol{\theta}) = 1$) process is obtained, namely $\tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta})$. The transformation from X_t to Y_t, to obtain the desired unit diffusion, is known as the Lamperti transform. [9] provides more information on the Lamperti Transform. The transformation, to obtain the required unit diffusion is given by

$$\lambda(X_t, t; \boldsymbol{\theta}) = \int^{X_t} \sigma^{-1}(\gamma, t; \boldsymbol{\theta}) d\gamma = Y_t, \tag{24}$$

where $\sigma^{-1}(\gamma, t; \boldsymbol{\theta})$ is the inverse of $\sigma(\gamma, t; \boldsymbol{\theta})$. Applying $Itô's$-lemma yields the desired unit diffusion:

$$dY_t = \mu_Y(Y_t, t; \boldsymbol{\theta})dt + 1dW_t, \tag{25}$$

s.t $\sigma(\gamma, t; \boldsymbol{\theta}) = 1$. By assuming $\sigma(X_t, t; \boldsymbol{\theta}) > 0$, implies the increasing and invertible nature of $\lambda(X_t, t; \boldsymbol{\theta})$ in Equation 24. The transformed drift coeffiecient, $\mu_Y(Y_t, t; \boldsymbol{\theta})$, in Equation 25 is determined as follows

$$\mu_Y(Y_t, t; \boldsymbol{\theta}) = \frac{\mu(\lambda^{-1}(Y_t, t; \boldsymbol{\theta}), t, \boldsymbol{\theta})}{\sigma(\lambda^{-1}(Y_t, t; \boldsymbol{\theta}), t, \boldsymbol{\theta})} - \frac{\partial \sigma(\lambda^{-1}(Y_t, t; \boldsymbol{\theta}), t, \boldsymbol{\theta})}{2\partial X_t}.$$

The transformation allows for the derivation of a closed-form approximtion, $\tilde{p}_Y(y_t, t|y_s, s; \boldsymbol{\theta})$, for $p_Y(y_t, t|y_s, s; \boldsymbol{\theta})$. As with the original process, the Hermite technique involves the approximation of the density increasing orders of K, that is$\tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta})$, s.t $K \geqslant 0$. . The Jacobian formula, as given in [1], is imployed to gain the required approximate density for X_t, $\tilde{p}_X(x_t, t|x_s, s; \boldsymbol{\theta})$ from given the approximate denisty for Y_t, $\tilde{p}_Y(y_t, t|y_s, s; \boldsymbol{\theta})$,was obtained. The Jacobian transformation follows as:

$$\begin{aligned}
\tilde{p}_X(x_t, t|x_s, s; \boldsymbol{\theta}) &= \frac{\partial \Pr[X_t \leq x_t | X_s = x_s]}{\partial x_t} \\
&= \frac{\partial \Pr[Y_t \leq \lambda(x_t, t; \boldsymbol{\theta}) | X_s = \lambda(x_s, s; \boldsymbol{\theta})]}{\partial x_t} \\
&= \frac{\partial \int^{\lambda(x_t, t; \boldsymbol{\theta})} \tilde{p}_Y(y_t, t|\lambda(x_s, s; \boldsymbol{\theta}), s; \boldsymbol{\theta}) dy_t}{\partial x_t} \\
&= \sigma^{-1}(\lambda(x_t, t; \boldsymbol{\theta}); \boldsymbol{\theta}) \tilde{p_Y}(\lambda(x_t, t; \boldsymbol{\theta}), t|\lambda(x_s, s; \boldsymbol{\theta}), s; \boldsymbol{\theta}).
\end{aligned} \tag{26}$$

The approximation procedure is started with the Hermite-series expansion for density function of Y_t around a Gaussian density. For increasing orders of K, closed-form analytical approximations, $\tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta})$, for$p_Y(y_t, t|y_s, s; \boldsymbol{\theta})$ is derived as follow,[1]:

$$\tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta}) = \frac{\exp\left[-\frac{1}{2}\left(\frac{y_t - y_s}{\delta^{1/2}}\right)^2\right]}{\delta^{1/2}\sqrt{2\pi}} \exp\left[\int_{y_s}^{y_t} \mu_Y(\gamma, t_\gamma; \boldsymbol{\theta}) d\gamma\right] \sum_{k=0}^{K} c_k(y_t, t|y_s, s; \boldsymbol{\theta}) \frac{\delta^k}{k!}, \tag{27}$$

where $\delta = t - s$ and $\frac{y_t - y_s}{\delta^{1/2}} \sim N(0,1)$ and $c_s(y_t, t|y_s, s; \boldsymbol{\theta}) = 1$ for all $n > s$, otherwise:

$$c_n(y_t, t|y_s, s; \boldsymbol{\theta}) = \frac{n \int_{y_s}^{y_t} (\gamma - y_s)^{n-1} \left[\left[\mu_{Y_t}^2(y_t, t; \boldsymbol{\theta}) + \frac{\partial \mu_{Y_t}(y_t, t; \boldsymbol{\theta})}{\partial y_t} \right] c_{n-1}(\gamma, t_\gamma | y_s, s; \boldsymbol{\theta}) + \frac{\partial^2 c_{n-1}(\gamma, t_\gamma | y_s, s; \boldsymbol{\theta})}{\partial \gamma^2} \right] d\gamma}{2(y_t - y_s)^n}. \tag{28}$$

The Kolmogorov forward and Kolmogorov backward equations, solve the sequence of equations, $\tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta})$, in Equation 27, for all $K \geq 0$:

$$\frac{\partial \tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta})}{\partial \delta} + \frac{\partial [\mu_{Y_t}(y_t, t; \boldsymbol{\theta}) \tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta})]}{\partial y_t} - \frac{1}{2} \frac{\partial^2 \tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta})}{\partial y_t^2} = o(\delta^K), \tag{29}$$

and

$$\frac{\partial \tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta})}{\partial \delta} - \mu_{Y_s}(y_s, s; \boldsymbol{\theta}) \frac{\partial \tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta})}{\partial y_s} - \frac{1}{2} \frac{\partial^2 \tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta})}{\partial y_s^2} = o(\Delta^K),$$

for all $K \geqslant 0$, s.t $t \in [s, T]$ and where a real-valued function $g(\delta)$ is $o(\delta)$ if $\lim_{\delta \to 0} \frac{g(\delta)}{\delta} = 0$.

Finally, the approximate density function for the variable of interest, X_t, can be obtained. Tha is $\tilde{p}_X^{(K)}(x_t, t|x_s, s; \boldsymbol{\theta})$, the K^{th} order approximation of $p_X(x_t, t|x_s, s; \boldsymbol{\theta})$. This is achieved by applying the Jacobian formula as in Equation 26 to obtain $\tilde{p}_X^{(K)}(x_t, t|x_s, s; \boldsymbol{\theta})$, from $\tilde{p}_Y^{(K)}(y_t, t|y_s, s; \boldsymbol{\theta})$:

$$\tilde{p}_X^{(K)}(x_t, t|x_s, s; \boldsymbol{\theta}) \equiv \sigma^{-1}(X_t, t; \theta) \tilde{p}_Y^{(K)}(\lambda(X_t, t; \boldsymbol{\theta}), t|\lambda(X_s, s; \boldsymbol{\theta}), s; \boldsymbol{\theta}), \tag{30}$$

for increasing orders of K. Approximating $\tilde{p}_X^{(1)}(x_t, t|x_s, s; \boldsymbol{\theta})$ and/or $\tilde{p}_X^{(2)}(x_t, t|x_s, s; \boldsymbol{\theta})$, has been proven effective in various financial settings, [1]. Higher orders of approximatio, will modelrately improve accuracy, but exponentiate computational complexity.

3.3.3 Cumulant truncated transition density approximation (saddlepoint approximation)

Cumulant or moment truncation is a transition density approximation technique where the system of ordinary differential moment equations, for the process under consideration, is solved and passed to a surrogate density, such as the saddlepoint density, [17].

Considering the general diffusion process, as in Equation 2, i.e

$$dX_t = \mu(X_t, t; \boldsymbol{\theta})dt + \sigma(X_t, t; \boldsymbol{\theta})dW_t,$$

s.t $t \in [s, T]$. The non-central moments of the diffusion process is obtained by solving a system of ordinary differential equations (ODEs) , [17]. The moments equations will be in the general form of:

$$m_i'(t) = f(m_{i-1}(t), m_i(t), \boldsymbol{\theta}), \tag{31}$$

s.t. $f(.)$ is a real valued function, $\boldsymbol{\theta}$ a function of parameters, and $i = 1, 2, ...$,s.t $\mathbb{E}[X_t^i | X_s] = m_i(t)$ the i^{th} non-central moment. To obtiain the system of ODEs, in Equation 31, the methodology as in [17] is imployed. Denote the moment generating function (MGF) of X_t by:

$$M(X_t, t) = \mathbb{E}[\exp(\nu X_t)].$$

It follows that $M(x, t)$ solves the partial differential equation (PDE):

$$\frac{\partial M(\nu, t)}{\partial t} = \nu\mu\Big(\frac{\partial}{\partial \nu}, t\Big)M(\nu, t) + \frac{1}{2}\nu^2\sigma^2\Big(\frac{\partial}{\partial \nu}, t\Big)M(\nu, t),$$

s.t $\mu\Big(\frac{\partial}{\partial \nu}, t\Big)$ and $\sigma^2\Big(\frac{\partial}{\partial \nu}, t\Big)$ are the differential operators on $M(\nu, t)$. The implication of this is that if integer powers of X_t are contained in $\mu(X_t, t; \boldsymbol{\theta})$ and $\sigma^2(X_t, t; \boldsymbol{\theta})$, a partial differential equation for the moment generating function in terms of derivatives w.r.t. ν (i.e. $\partial/\partial\nu$) can be obtained. For, example, by setting $\mu(x_t, t; \boldsymbol{\theta}) = B_0 + B_1 x_t + B_2 x_t^2$ and $\sigma^2(x_t, t; \boldsymbol{\theta}) = B_3^2$ it follows that:

$$\frac{\partial M(\nu, t)}{\partial t} = \nu\left[B_0 + B_1\frac{\partial}{\partial \nu} + B_2\frac{\partial^2}{\partial \nu^2}\right]M(\nu, t) + \frac{1}{2}\nu^2\sigma^2 B_3^2 M(\nu, t). \tag{32}$$

Define

$$m_i(t) = \mathbb{E}[X_t^i]$$

and

$$M(\nu, t) = \sum_{i=0}^{\infty}\frac{\nu^i m_i(t)}{i!}. \tag{33}$$

By substituting Equation 33 into Equation 32, we obtain a system ordinary differential equations for the non-central moments of the diffusion process, i.e $m_i'(t) = f(m_{i-1}(t), m_i(t), \boldsymbol{\theta})$. The number of moments are truncated at a specific order, often at the second or fourth moment. Higher order moments may improve accuracy, but the trade of against computational complexity, may deem it futile to do so.

The resulting system of ODEs can be solved by applying the Laplace transform ($\mathcal{L}\{.\}$):

$$\mathcal{L}\{m_j'(t)\} = \int_0^{\infty} e^{-vt} m_j(t) dt,$$

and solving the partial fractions. The resulting solution yields the desired moments for use in the transition density approximation:

$$m_i(t) = z(m_{i-1}(t), .., m_1(t), m_0(t), \boldsymbol{\theta}), \tag{34}$$

Given that the cumulants are used in the saddlepoint approximation, the cumulants must be found

from the moments. We simply calculate the cumulants by use of the following relation between the CGF and MGF:

$$K(\nu,t) = \ln\Big[M(\nu,t)\Big],$$

with $M(\nu,t)$, the moment generating function. The saddlepoint transition density function approximation is derived in closed-form, by substituting the calculated cumulants (k_i) into the surrogate saddlepoint density,[10]. Firstly, define the cumulant generating function as

$$K(\nu,t) = \ln\Big[\sum_{i=0}^{\infty} \frac{\nu^i m_i(t)}{i!}\Big], \tag{35}$$

With the first four cumulants given by

$$\begin{aligned}
k_1(t) &= m_1(t),\\
k_2(t) &= m_2(t) - (m_1(t))^2,\\
k_3(t) &= 2(m_1(t))^3 - 3(m_1(t))(m_2(t)) + m_3(t),\\
k_3(t) &= -6(m_1(t))^4 + 12(m_1(t))^2(m_2(t)) - 3(m_2(t))^2 - 4(m_1(t))(m_3(t)) + m_4(t).
\end{aligned}$$

Following calculate the first two partial derivatives partial derivatives, in terms of t:

$$K_m^{'}(\nu,t) = \frac{\partial}{\partial t}\ln\Big[\sum_{i=0}^{\infty} \frac{\nu^i m_i(t)}{i!}\Big] \tag{36}$$

and

$$K_m^{''}(\nu,t) = \frac{\partial^2}{\partial t^2}\ln\Big[\sum_{i=0}^{\infty} \frac{\nu^i m_i(t)}{i!}\Big], \tag{37}$$

with $k_m(\nu,t)$ denoting the m^{th} cumulant at time t. Next, Setting $X_t = K_m^{''}(\nu,t)$, t is determined as a function of X_t, that is:

$$t = \zeta(X_t). \tag{38}$$

Finally, substituing the resulting cumulant equations obtained in Equation 36 and 38, into the saddlepoint surrogate density, as in [10], the closed-form cumulant-truncated transition density function approximation, $p_X^{saddle}(X_t,t|X_s,s;\boldsymbol{\theta})$, is obtained as:

$$p_X^{saddle}(X_t,t|X_s,s;\boldsymbol{\theta})) = \exp(K_m(\nu,t) - \zeta(X_t)x_t)\Big(2\pi K_m^{''}(\nu,t)\Big)^{-1/2}, \tag{39}$$

for the m^{th} order of approximation.

3.3.4 Examples of fitting approximate transition densities to univariate diffusion processes.

Example 19. Hermite approximate transition density function derivation for a univariate CIR process

Consider the CIR model in Example 13 :

$$dX_t = \kappa(\alpha - X_t)dt + \sigma\sqrt{X_t}dW_t, \tag{40}$$

s.t.$t \in [0,5]$, $X_t \in [0,1]$,$\boldsymbol{\theta} = (\kappa, \alpha, \sigma) = (0.9, 0.3, 0.075)$, with $X_0 = 0.15$, and $stepsize = 1/250$. The Hermite approximate transition density function derivation for a univariate CIR process, as in [1] follows. Firstly, since dX_t does not have a unit diffusion, a transformation is to be made, in terms of a change of variable, from X_t to Y_t, to obtain a unit diffusion. The required transformation follows as

$$Y_t = \Omega(X_t, t; \boldsymbol{\theta}) = 2\sigma^{-1}\sqrt{X_t}. \tag{41}$$

To prove Y_t has unit diffusion, $Itô$'s lemma is applied

$$dY_t = \frac{\partial}{\partial t}\Omega(X_t, t; \boldsymbol{\theta})dt + \frac{\partial}{\partial X_t}\Omega(X_t, t; \boldsymbol{\theta})dX_t + \frac{1}{2}\frac{\partial^2}{\partial X_t^2}\Omega(X_t, t; \boldsymbol{\theta})(dX_t)^2, \tag{42}$$

with partial derivatives equal to

$$\begin{aligned} &\frac{\partial\Omega(X_t, t; \boldsymbol{\theta})}{\partial t}) = 0, \\ &\frac{\partial\Omega(X_t, t; \boldsymbol{\theta})}{\partial X_t} = \left(\sigma\sqrt{X_t}\right)^{-1}, \\ &\frac{\partial^2\Omega(X_t, t; \boldsymbol{\theta})}{\partial X_t^2} = -\frac{X_t^{-3/2}}{2\sigma}, \end{aligned} \tag{43}$$

yields the desired unit diffusion

$$dY_t = \Big[\frac{\kappa(\alpha - X_t) - \sigma^2}{\sigma\sqrt{X_t}}\Big]dt + 1dWt. \tag{44}$$

In Equation 44, Y_t clearly has the unit diffusion, as required.

The Hermite-series expansion for of order $K = 0$ for Y_t is given follows:

$$\tilde{p}_Y^{(0)}(y_t, t|y_s, s; \boldsymbol{\theta}) = \frac{\exp(-\frac{(y_t - y_s)^2}{2\delta} - \kappa\frac{y_t^2 - y_s^2}{4})}{\sqrt{2\pi\delta}}\left[y_s^{\frac{1}{2} - \frac{2\alpha\kappa}{\sigma^2}}\right]\left[y_t^{-\frac{1}{2} + \frac{2\alpha\kappa}{\sigma^2}}\right], \tag{45}$$

41 the Hermite-series transition density function approximation of order X_t, for $K = 0$, is given by means of the Jacobian transformation:

$$\tilde{p}_X^{(0)}(x_t, t|x_s, s; \boldsymbol{\theta}) = \sigma^{-1}(x_t, t; \boldsymbol{\theta})\tilde{p}_Y^{(0)}(\Omega(x_t, t; \boldsymbol{\theta}), t|\Omega(x_s, s; \boldsymbol{\theta}), s; \boldsymbol{\theta}),$$

$$\tilde{p}_X^{(0)}(x_t, t|x_s, s; \boldsymbol{\theta}) = \exp(-\frac{(\Omega(x_t, t; \boldsymbol{\theta}) - \Omega(x_s, s; \boldsymbol{\theta}))^2}{2\delta} - \kappa\frac{\Omega^2(x_t, t; \boldsymbol{\theta}) - \Omega^2(x_s, s; \boldsymbol{\theta})}{4})\frac{\Omega(x_t, t; \boldsymbol{\theta})^{-\frac{1}{2} + \frac{2\alpha\kappa}{\sigma^2}}\Omega(x_s, s; \boldsymbol{\theta})^{\frac{1}{2} - \frac{2\alpha\kappa}{\sigma^2}}}{\sigma\sqrt{2\pi\delta x_t}}, \tag{46}$$

The Hermite-series expansion of order $K = 1$, for Y_t is given by:

$$\tilde{p}_Y^{(1)}(y_t, t|y_s, s; \boldsymbol{\theta}) = \tilde{p}_Y^{(0)}(y_t, t|y_s, s; \boldsymbol{\theta})(1 + \delta c_1(y_t, t|y_s, s; \boldsymbol{\theta})), \tag{47}$$

where

$$c_1(y_t, t|y_s, s; \boldsymbol{\theta}) = -\frac{(48\alpha^2\kappa^2 - 48\alpha\kappa\sigma^2 + 9\sigma^4 + y_t\kappa^2\sigma^2(-24\alpha + y_t^2\sigma^2)y_s + y_t^2\kappa^2\sigma^4y_s^2 + y_t\kappa^2\sigma^4y_s^3)}{24y_ty_s\sigma^4}.$$

The Hermite-series transition density function approximation of order X_t, for $K = 1$, is given by means of the Jacobian transformation:

$$\tilde{p}_X^{(1)}(x_t, t|x_s, s; \boldsymbol{\theta}) = \sigma^{-1}(x_t, t; \boldsymbol{\theta})\tilde{p}_Y^{(1)}(\Omega(x_t, t; \boldsymbol{\theta}), t|\Omega(x_s, s; \boldsymbol{\theta}), s; \boldsymbol{\theta}),$$

$$\tilde{p}_X^{(1)}(x_t, t|x_s, s; \boldsymbol{\theta}) = \frac{\tilde{p}_Y^{(0)}(\Omega(x_t,t;\boldsymbol{\theta}),t|\Omega(x_s,s;\boldsymbol{\theta}),s;\boldsymbol{\theta})\{1+\delta c_1(\Omega(x_t,t;\boldsymbol{\theta}),t|\Omega(x_s,s;\boldsymbol{\theta}),s;\boldsymbol{\theta})\}}{\sigma\sqrt{x_t}}.$$

With the Hermite-series transition density function approximation, of order $K = 1$, for X_t, given by:

$$\begin{aligned}\tilde{p}_X^{(1)}(x_t, t|x_s, s; \boldsymbol{\theta}) = \frac{\exp(-\frac{(\Omega(x_t,t;\boldsymbol{\theta})-\Omega(x_s,s;\boldsymbol{\theta}))^2}{2\delta} - \kappa\frac{\Omega^2(x_t,t;\boldsymbol{\theta})-\Omega^2(x_s,s;\boldsymbol{\theta})}{4})\Omega(x_t,t;\boldsymbol{\theta})^{-\frac{1}{2}+\frac{2\alpha\kappa}{\sigma^2}}\Omega(x_s,s;\boldsymbol{\theta})^{\frac{1}{2}-\frac{2\alpha\kappa}{\sigma^2}}}{\sigma\sqrt{2\pi\delta x_t}} \\ \times \left[1 - \frac{\delta}{24\Omega(x_t,t;\boldsymbol{\theta})\Omega(x_s,s;\boldsymbol{\theta})\sigma^4}\left[48\alpha^2\kappa^2 - 48\alpha\kappa\sigma^2 + 9\sigma^4\right.\right. \\ \Omega(x_t,t;\boldsymbol{\theta})\kappa^2\sigma^2(-24\alpha + \Omega(x_t,t;\boldsymbol{\theta})^2\sigma^2)\Omega(x_s,s;\boldsymbol{\theta}) \\ \left.\left.\Omega(x_t,t;\boldsymbol{\theta})^2\kappa^2\sigma^4\Omega(x_s,s;\boldsymbol{\theta})^2 + \Omega(x_t,t;\boldsymbol{\theta})\kappa^2\sigma^4\Omega(x_s,s;\boldsymbol{\theta})^3)\right]\right].\end{aligned}$$

The Hermite-series expansion of order $K = 2$, for Y_t is given by:

$$\tilde{p}_Y^{(2)}(y_t, t|y_s, s; \boldsymbol{\theta}) = \tilde{p}_Y^{(0)}(y_t, t|y_s, s; \boldsymbol{\theta})(1 + \delta c_1(y_t, t|y_s, s; \boldsymbol{\theta}) + \frac{\delta^2}{2}c_2(y_t, t|y_s, s; \boldsymbol{\theta})),$$

s.t

$$c_2(y_t, t|y_s, s; \boldsymbol{\theta}) = \frac{\Lambda(y_t, t|y_s, s; \boldsymbol{\theta})}{576y_t^2y_s^2\sigma^8},$$

where

$$\begin{aligned}\Lambda(y_t, t|y_s, s; \boldsymbol{\theta}) =& 9(256(\kappa\alpha)^4 - 512(\kappa\alpha)^3\sigma^2 + 224(\kappa\alpha)\sigma^4 + 32(\kappa\alpha)\sigma^6 - 15\sigma^8) \\ &+6y_t\kappa^2\sigma^2(-24\alpha + y_t^2\sigma^2)(16\alpha^2\kappa^2 - 16\alpha\kappa\sigma^2 + 3\sigma^4)y_s \\ &+y_t^2\kappa^2\sigma^4(672\alpha^2\kappa^2 - 48\alpha\kappa(2 + y_t^2\kappa)\sigma^2 + (-6 + y_t^4\kappa^2)\sigma^4)y_s^2 \\ &+2y_t\kappa^2\sigma^4(48\alpha^2\kappa^2 - 24\alpha\kappa(2 + y_t^2\kappa)\sigma^2 + (9 + y_t^4\kappa^2)\sigma^4)y_s^3 \\ &+3y_t^2\kappa^4\sigma^6(-16\alpha + y_t^2\sigma^2)y_s^4 + 2y_t^3\kappa^4\sigma^8y_s^5 + y_t^2\kappa^4\sigma^8y_s^6.\end{aligned}$$

With the Hermite-series transition density function approximation, of order $K = 2$, for X_t, given by:

$$\tilde{p}_X^{(2)}(y_t, t|y_s, s; \boldsymbol{\theta}) \equiv \frac{\tilde{p}_Y^{(2)}(\Omega(x_t,t;\boldsymbol{\theta}),t|\Omega(x_s,s;\boldsymbol{\theta}),s;\boldsymbol{\theta})}{\sigma(x_t,t;\boldsymbol{\theta})},$$

$$, \quad \tilde{p}_Y^{(2)}(x_t,t|x_s,s;\boldsymbol{\theta}) \equiv \begin{array}{l} \frac{\tilde{p}_Y^{(0)}(\Omega(x_t,t;\boldsymbol{\theta}),t|\Omega(x_s,s;\boldsymbol{\theta}),s;\boldsymbol{\theta})(1+\delta c_1(\Omega(x_t,t;\boldsymbol{\theta}),t|\Omega(x_s,s;\boldsymbol{\theta}),s;\boldsymbol{\theta})}{\sigma(x_t,t;\boldsymbol{\theta})} \\ +\frac{\frac{\delta^2}{2}c_2(\Omega(x_t,t;\boldsymbol{\theta}),t|\Omega(x_s,s;\boldsymbol{\theta}),s;\boldsymbol{\theta})}{\sigma(x_t,t;\boldsymbol{\theta})}, \end{array} \tag{48}$$

with

$$c_2(x_t,t|x_s,s;\boldsymbol{\theta}) = \frac{\eta(x_t,t|x_s,s;\boldsymbol{\theta})}{576\Omega^2(x_t,t;\boldsymbol{\theta})\Omega^2(x_s,s;\boldsymbol{\theta})\sigma^8},$$

where

$$\begin{aligned}
\eta(x_t,t|x_s,s;\boldsymbol{\theta}) =& 9(256(\kappa\alpha)^4 - 512(\kappa\alpha)^3\sigma^2 + 224(\kappa\alpha)\sigma^4 + 32(\kappa\alpha)\sigma^6 - 15\sigma^8) \\
&+6\Omega(x_t,t;\boldsymbol{\theta})\kappa^2\sigma^2(-24\alpha + \Omega^2(x_t,t;\boldsymbol{\theta})\sigma^2)(16\alpha^2\kappa^2 - 16\alpha\kappa\sigma^2 + 3\sigma^4)\Omega(x_s,s;\boldsymbol{\theta}) \\
&+\Omega^2(x_t,t;\boldsymbol{\theta})\kappa^2\sigma^4(672\alpha^2\kappa^2 - 48\alpha\kappa(2 + \Omega^2(x_t,t;\boldsymbol{\theta})\kappa)\sigma^2 + (-6 + \Omega^4(x_t,t;\boldsymbol{\theta})\kappa^2)\sigma^4)\Omega^2(x_s,s;\boldsymbol{\theta}) \\
&+2\Omega(x_t,t;\boldsymbol{\theta})\kappa^2\sigma^4(48\alpha^2\kappa^2 - 24\alpha\kappa(2 + \Omega^2(x_t,t;\boldsymbol{\theta})\kappa)\sigma^2 + (9 + \Omega^4(x_t,t;\boldsymbol{\theta})\kappa^2)\sigma^4)\Omega^3(x_s,s;\boldsymbol{\theta}) \\
&+3\Omega^2(x_t,t;\boldsymbol{\theta})\kappa^4\sigma^6(-16\alpha + \Omega^2(x_t,t;\boldsymbol{\theta})\sigma^2)\Omega^4(x_s,s;\boldsymbol{\theta}) \\
&+2\Omega^3(x_t,t;\boldsymbol{\theta})\kappa^4\sigma^8\Omega^5(x_s,s;\boldsymbol{\theta}) + \Omega^2(x_t,t;\boldsymbol{\theta})\kappa^4\sigma^8\Omega^6(x_s,s;\boldsymbol{\theta}).
\end{aligned}$$

Figure13 shows the decreasing significance in the $c_k$coefficients as the order of approximationincrease. Algorithm 7 and Figure 16 contains the plotted Hermite Approximate CIR transition densities for $K = 1, 2$.

Example 20. Cumulant Truncated (saddlepoint) approximate transition density function derivation for a univariate CIR process

An alternative strategy for approximating the transitional density of the CIR process is by the so-called cumulant truncation procedure developed in [17]. The true transition density is plotted in Figure 16 for comparison.

Consider the CIR model in Example 13 :

$$dX_t = \kappa(\alpha - X_t)dt + \sigma\sqrt{X_t}dW_t, \tag{49}$$

s.t.$t \in [0,5]$, $X_t \in [0,1]$,$\boldsymbol{\theta} = (\kappa,\alpha,\sigma) = (0.9, 0.3, 0.075)$, with $X_0 = 0.15$, and $stepsize = 1/250$. The derived system of ODEs are given by:

$$\begin{aligned}
m_1^{'}(t) &= 1\kappa(\alpha - m_1(t)), \\
m_2^{'}(t) &= 2\kappa(\alpha m_1(t) - m_2(t)) + \sigma^2 m_1(t), \\
m_3^{'}(t) &= 3\kappa(\alpha m_2(t) - m_3(t)) + 3\sigma^2 m_2(t), \\
m_4^{'}(t) &= 4\kappa(\alpha m_3(t) - m_4(t)) + 6\sigma^2 m_3(t).
\end{aligned} \tag{50}$$

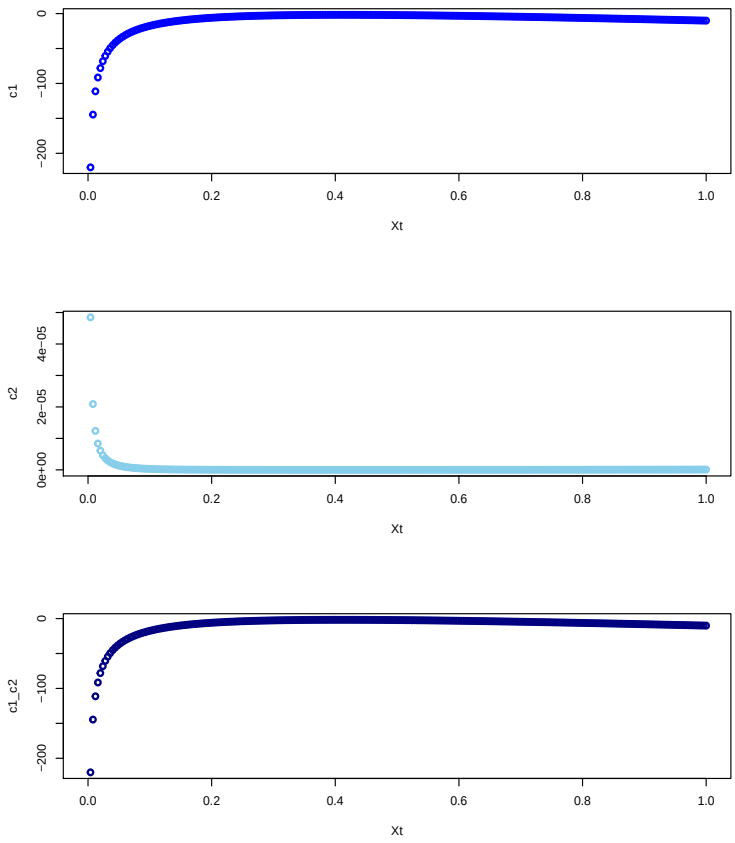


Figure 13: decreasing significance in the $c_k$coefficients as the order of approximation - k- increase.

The system of ODEs is solved by applying the Laplace transform and partial fractions, to obtain the non-central moments of the proces:

$$
\begin{aligned}
E[X_t|X_s] =& m_1(t) = X_s \exp(-\kappa t) + \alpha(1 - e^{-\kappa t}),\\
E[X_t^2|X_s] =& m_2(t) = X_s^2 \exp(-2\kappa t) + (\alpha + \frac{\sigma^2}{2\kappa})(\alpha + 2(X_s - \alpha)e^{-\kappa t} + (\alpha - 2X_s)e^{-2\kappa t}),\\
E[X_t^3|X_s] =& m_3(t) = X_s^3 \exp(-3\kappa t) + 3(\kappa\alpha + \sigma^2)(A + Be^{-\kappa t} + Ce^{-2\kappa t} + De^{-3\kappa t}),\\
E[X_t^4|X_s] =& m_4(t) = X_s^4 \exp(-4\kappa t) + (4\kappa\alpha + 6\sigma^2)\Big[E + Fe^{-\kappa t} + Ge^{-2\kappa t} + He^{-3\kappa t} + Ie^{-4\kappa t}\Big],
\end{aligned}
\tag{51}
$$

where

$$
\begin{aligned}
E =& \frac{3(\kappa\alpha + \sigma^2)(6\kappa^3 A)}{24\kappa^4},\\
I =& -\frac{1}{6\kappa^3}\Bigg(\Big(4\kappa^2 X_s^3 + 3(\kappa\alpha + \sigma^2)\big[11\kappa^2 A + 6\kappa^2 B + 3\kappa^2 C + 2\kappa^2 D\big] - \frac{13}{12}\frac{3(\kappa\alpha + \sigma^2)(6\kappa^3 A)}{\kappa}\Big)\\
&- 12\kappa^2\Big(5\kappa X_s^3 + 3(\kappa\alpha + \sigma^2)\big[6\kappa A + 5\kappa B + 4\kappa C + 3\kappa D\big] - \frac{3(\kappa\alpha + \sigma^2)(6\kappa^3 A)}{24\kappa^3}\Big)\\
&- 4\kappa\Big(\Big(5\kappa X_s^3 + 3(\kappa\alpha + \sigma^2)\big[6\kappa A + 5\kappa B + 4\kappa C + 3\kappa D\big] - \frac{3}{8}\frac{3(\kappa\alpha + \sigma^2)(6\kappa^3 A)}{\kappa}\Big)\\
&- 7\kappa\Big(5\kappa X_s^3 + 3(\kappa\alpha + \sigma^2)\big[6\kappa A + 5\kappa B + 4\kappa C + 3\kappa D\big] - \frac{3(\kappa\alpha + \sigma^2)(6\kappa^3 A)}{24\kappa^3}\Big)\Big)\Bigg),\\
H =& \frac{1}{2\kappa^2}\Bigg(\Big(5\kappa X_s^3 + 3(\kappa\alpha + \sigma^2)\big[6\kappa A + 5\kappa B + 4\kappa C + 3\kappa D\big] - \frac{3}{8}\frac{3(\kappa\alpha + \sigma^2)(6\kappa^3 A)}{\kappa}\Big)\\
&-7\kappa\Big(5\kappa X_s^3 + 3(\kappa\alpha + \sigma^2)\big[6\kappa A + 5\kappa B + 4\kappa C + 3\kappa D\big] - \frac{3(\kappa\alpha + \sigma^2)(6\kappa^3 A)}{24\kappa^3}\Big) - 6\kappa^2 I\Bigg)\\
F =& -(E + G + H + I),\\
A =& \frac{\alpha(\alpha + \frac{\sigma^2}{2\kappa})}{3\kappa},\\
C =& -4(\frac{1}{4\kappa^2}(\kappa(X_s^2 + (\alpha + \frac{\sigma^2}{2\kappa})(\alpha - 2X_s)) + 3\kappa\alpha(\alpha + \frac{\sigma^2}{2\kappa}) + 4\kappa(\alpha + \frac{\sigma^2}{2\kappa})(X_s - \alpha), -9\kappa^2 A)\\
&- \frac{1}{2\kappa}(X_s^2 + (\alpha + \frac{\sigma^2}{2\kappa})(\alpha - 2X_s) + \alpha(\alpha + \frac{\sigma^2}{2\kappa}) + 2(\alpha + \frac{\sigma^2}{2\kappa})(X_s - \alpha), -3\kappa A)),\\
B =& \frac{X_s^2 + (\alpha + \frac{\sigma^2}{2\kappa})(\alpha - 2X_s) + \alpha(\alpha + \frac{\sigma^2}{2\kappa}) + 2(\alpha + \frac{\sigma^2}{2\kappa})(X_s - \alpha) - 3\kappa X_s^2}{2\kappa},\\
&+\frac{-3\kappa X_s^2 + (\alpha + \frac{\sigma^2}{2\kappa})(\alpha - 2X_s) + \alpha(\alpha + \frac{\sigma^2}{2\kappa}) + 2(\alpha + \frac{\sigma^2}{2\kappa})(X_s - \alpha) - \kappa C}{2\kappa}\\
D =& -(A + B + C).
\end{aligned}
$$

As the cumulants are of main interest for the saddlepoint approximation, and given the following moment to cumulant conversion equations: Figure 14 depicts the empirical moments plotted againts the theoretical

moments, illustrationg the accuracy of the cumulant truncation procedure.

$$
\begin{aligned}
K_1(t) &= m_1(t), \\
K_2(t) &= m_2(t) - (m_1(t))^2, \\
K_3(t) &= 2(m_1(t))^3 - 3(m_1(t))(m_2(t)) + m_3(t), \\
K_3(t) &= -6(m_1(t))^4 + 12(m_1(t))^2(m_2(t)) - 3(m_2(t))^2 - 4(m_1(t))(m_3(t)) + m_4(t).
\end{aligned}
\tag{52}
$$

Therefore substituting the moments into the cumulant equations yields:

$$
\begin{aligned}
K_1(t) &= X_s \exp(-\kappa t) + \alpha(1 - e^{-\kappa t}), \\
K_2(t) &= X_s^2 \exp(-2\kappa t) + (\alpha + \frac{\sigma^2}{2\kappa})(\alpha + 2(X_s - \alpha)e^{-\kappa t} + (\alpha - 2X_s)e^{-2\kappa t}) \\
&\quad - (X_s^2 \exp(-2\kappa t) + (\alpha + \frac{\sigma^2}{2\kappa})(\alpha + 2(X_s - \alpha)e^{-\kappa t} + (\alpha - 2X_s)e^{-2\kappa t}))^2, \\
K_3(t) &= 2(X_s \exp(-\kappa t) + \alpha(1 - e^{-\kappa t}))^3 + m_3(t), \\
&\quad - 3(X_s \exp(-\kappa t) + \alpha(1 - e^{-\kappa t}))(X_s^2 \exp(-2\kappa t) + (\alpha + \frac{\sigma^2}{2\kappa})(\alpha + 2(X_s - \alpha)e^{-\kappa t} + (\alpha - 2X_s)e^{-2\kappa t})) \\
&\quad + X_s^3 \exp(-3\kappa t) + 3(\kappa\alpha + \sigma^2)(A + Be^{-\kappa t} + Ce^{-2\kappa t} + De^{-3\kappa t}) \\
K_3(t) &= -6(X_s \exp(-\kappa t) + \alpha(1 - e^{-\kappa t}))^4 \\
&\quad + 12(X_s \exp(-\kappa t) + \alpha(1 - e^{-\kappa t}))^2(X_s^2 \exp(-2\kappa t) + (\alpha + \frac{\sigma^2}{2\kappa})(\alpha + 2(X_s - \alpha)e^{-\kappa t} + (\alpha - 2X_s)e^{-2\kappa t})) \\
&\quad - 3(X_s^2 \exp(-2\kappa t) + (\alpha + \frac{\sigma^2}{2\kappa})(\alpha + 2(X_s - \alpha)e^{-\kappa t} + (\alpha - 2X_s)e^{-2\kappa t}))^2 \\
&\quad - 4(X_s \exp(-\kappa t) + \alpha(1 - e^{-\kappa t}))(X_s^3 \exp(-3\kappa t) + 3(\kappa\alpha + \sigma^2)(A + Be^{-\kappa t} + Ce^{-2\kappa t} + De^{-3\kappa t})) \\
&\quad + X_s^4 \exp(-4\kappa t) + (4\kappa\alpha + 6\sigma^2)\Big[E + Fe^{-\kappa t} + Ge^{-2\kappa t} + He^{-3\kappa t} + Ie^{-4\kappa t}\Big].
\end{aligned}
\tag{53}
$$

We can now differentiate and start plugging the expressions into the surrogate saddlepoint density approximate.

Figure 14, with the code contained in Algorithm 9, displays the theoretical cumulants of the CIR process, given in Equation 52, with the empirical cumulants. As the order of the cumulants increase, the empirical cumulants drift further apart from the theoretical cumulants.

$$
K_4^{'}(\nu, t) \approx \widetilde{K}_4(\nu, t) = tK_1(t) + \frac{1}{2!}t^2K_2(t) + \frac{1}{3!}t^3K_3(t) + \frac{1}{4!}t^4K_4(t), \tag{54}
$$

for $K_i(t)$ for $i = 1, 2, 3, 4$ as in Equation 52. A Taylor-series is applied to get $\widetilde{K}_4(\nu, t)$. Where the exact cumulant generating function of the CIR process is $K(\nu, t) = ln(M(\nu, t))$, provided $M(\nu, t)$ exists and $M(\nu, t) > 0$ for all values of t, where $M(\nu, t)$ is the exact moment generating function of the CIR process.

Consider the first and second order partial derivatives of Equation 54, in terms of t:

$$K_4'(X_t,t) \approx K_1(t) + tK_2(t) + \frac{1}{2}t^2K_3(t) + \frac{1}{6}t^3K_4(t), \tag{55}$$

$$K_4''(X_t,t) \approx K_2(t) + tK_3(t) + \frac{1}{2}t^2K_4(t). \tag{56}$$

Setting $X_t = K_1(t) + tK_2(t) + \frac{1}{2}t^2K_3(t) + \frac{1}{6}t^3K_4(t)$, we solve for t:

$$t = \frac{-K_2(t) + \sqrt{(K_2(t))^2 - 2K_3(t)(K_1(t) - X_t)}}{K_3(t)}. \tag{57}$$

Finally substituting all expressins into the saddlepoint approximation, yields the final cumulant truncated approximate transition density:

$$\begin{aligned} p_X^{saddle}(X_t,t|X_s,s;\boldsymbol{\theta}) =& \sqrt{(2\pi(K_2(t) + tK_3(t) + \frac{1}{2}t^2K_4(t)))^{-1}} \\ &\times \exp\left[tK_1(t) + \frac{1}{2!}t^2K_2(t) + \frac{1}{3!}t^3K_3(t) + \frac{1}{4!}t^4K_4(t)\right] \\ &- \left[\frac{-K_2(t) + \sqrt{(K_2(t))^2 - 2K_3(t)(K_1(t) - x_t)}}{K_3(t)}\right]X_t. \end{aligned} \tag{58}$$

Algorithm 7 and Figure 16 contains the code and plot for the Saddlepoint Approximate CIR transition density $p_X^{saddle}(X_t,t|X_s,s;\boldsymbol{\theta}))$.

3.3.5 Hermite-series transition density function approximation compared to the moment-truncated saddlepoint approximation

The work of [21] indicates that the Hermite-series transition density function approximation can only be applied to reducible (i.e $Y_t \to X_t$ is a one-to-one transformation) diffusion processes, although all univariate processes are reducible, not all multivariate diffusion processes are reducible. The Hermite-series transition density function approximation is difficult to implement and there is significant improvement needed in the accuracy from that provided by the Hermite-series transition density function approximation. Since a simpler, more general and accurate transition density function approximation is required, the saddlepoint approximation is ideal since it only requires the first few moment trajectories of the given diffusion process. The saddlepoint approximation also seems to be more robust to changes in the underlying parameters. Although neither the Hermite-series transition density function approximation, nor the moment-truncated saddlepoint approximation integrate to 1, this can be corrected for by normalizing constants.

Example 21. Univariate Ornstein Uhlenbeck diffusion processprocess' Hermite and Saddlepoint/Cumulant truncation approximate transition density comparison

Example 12 continued.

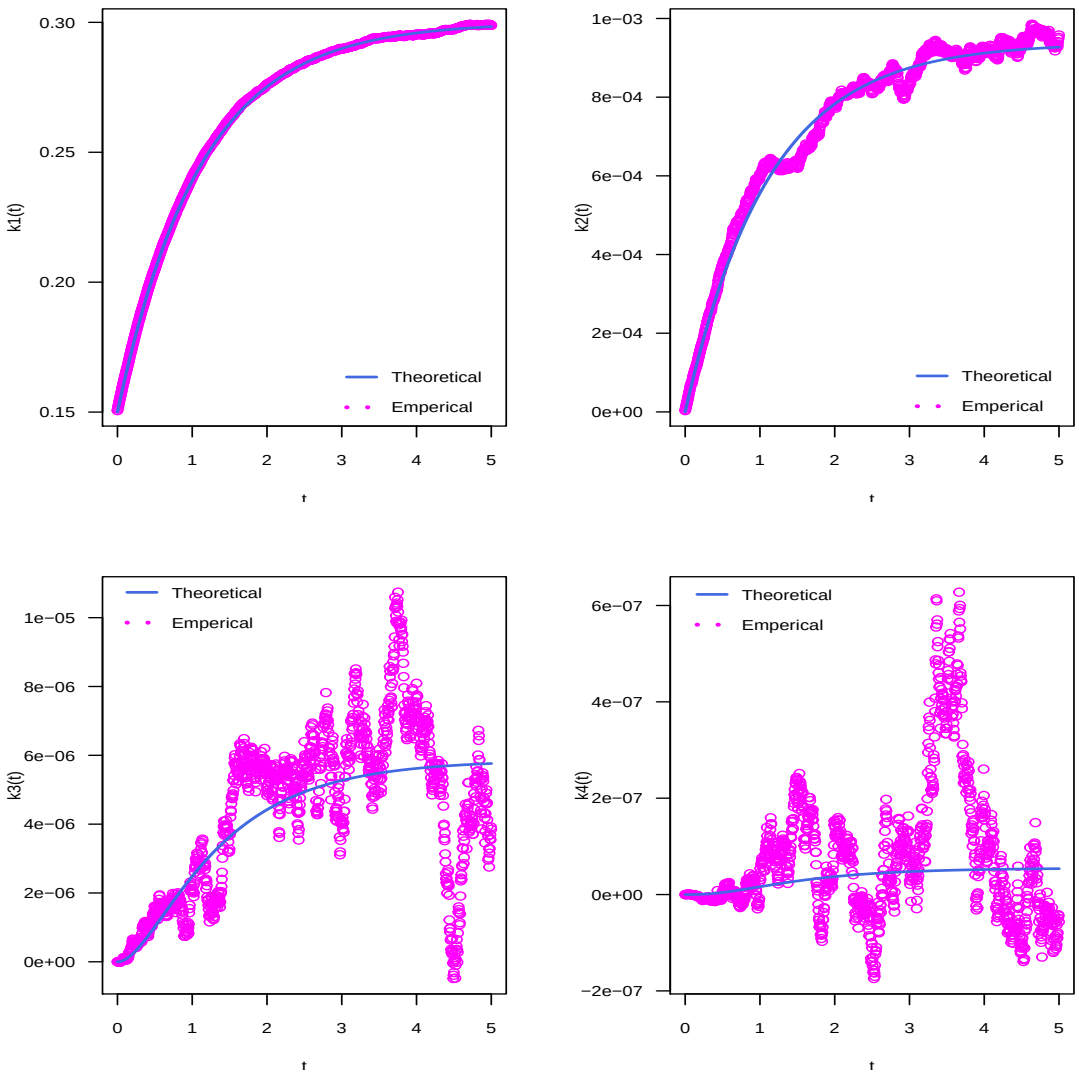


Figure 14: theoretical and empirical evolution of the cumulants of a univariate CIR Process.

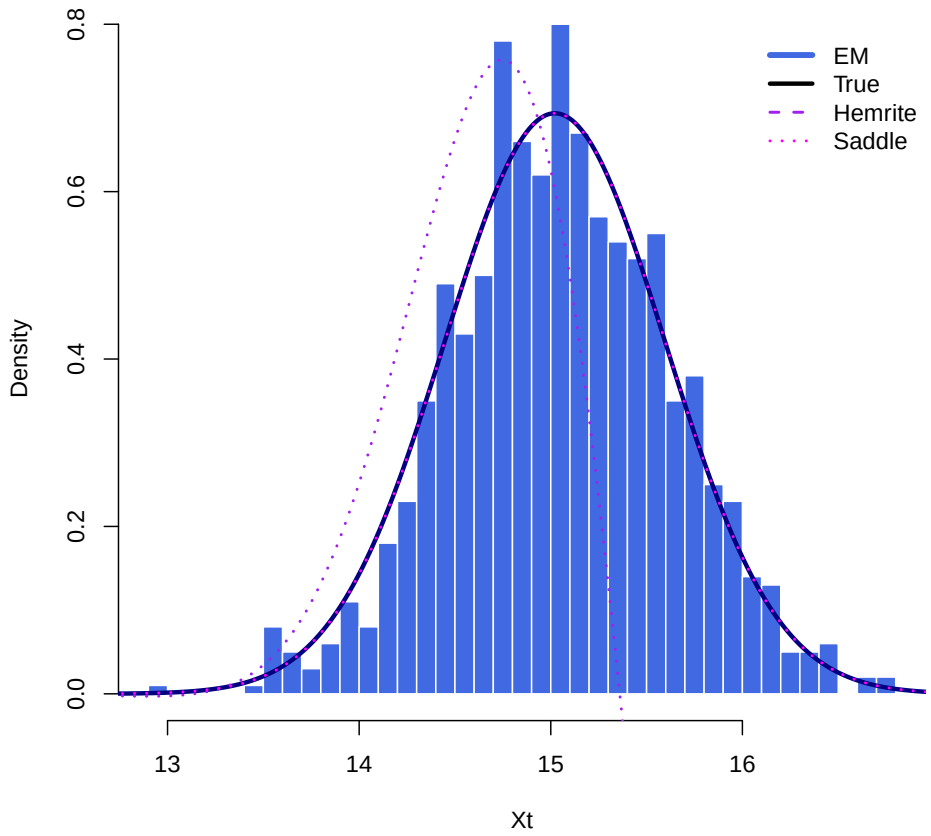


Figure 15: univariate OU Process's Theoretical density, EM distribution, Hermite transition approximation for $K = 1$, as well as the Saddlepoint/Cumulant truncation approximate transition density

$$dX_t = \kappa(\alpha - X_t)dt + \sigma dW_t, \tag{59}$$

s.t. $t \in [s, T]$, with $s \geqslant 0$ and $X_t \in [X_s, X_T]$, with $s \geqslant 0$, and $X_s \geqslant 0$ and with Wt Brownian Motion, as in Definition 6. Performing the simulation study on $t \in [0, 5]$, $X_t \in [12, 19]$,$\boldsymbol{\theta} = (\kappa, \alpha, \sigma) = (0.85, 15, 0.75)$, with $X_0 = 16$, and $stepsize = 1/250$ (aprroximate number of annual trading days) . Algorithm 7 and Figure 15 contains the plotted densities the Theoretical density, EM distribution, Hermite transition approximation for $K = 1$, as well as the Saddlepoint/Cumulant truncation approximate transition density. It can be seen that the Saddlemoint method is the best fit to the true density, with the Hermite approximate not perfroming well for $K = 1$, due to sensitivity to the size of increments. Algorithm 7.

The system of ODEs used for deriving the cumulant truncated approximate OU transition density is

given by:

$$\begin{aligned} m_1^{'}(t) &= \kappa(\alpha - m_1(t)), \\ m_2^{'}(t) &= 2\kappa(\alpha m_1(t) - m_2(t)) + \sigma^2, \\ m_3^{'}(t) &= 3\kappa(\alpha m_2(t) - m_3(t)) + 3\sigma^2 m_1(t), \\ m_4^{'}(t) &= 4\kappa(\alpha m_3(t) - m_4(t)) + 6\sigma^2 m_2(t). \end{aligned} \tag{60}$$

Example 22. Univariate CIR process' Hermite and Saddlepoint/Cumulant truncation approximate transition density comparison

Consider the CIR model, as in Example 13, [7, 1], with the SDE :

$$dX_t = \kappa(\alpha - X_t)dt + \sigma\sqrt{X_t}dW_t, \tag{61}$$

s.t. $t \in [s, T]$, , and with dW Brownian Motion. With for $t \in [0, 5]$, $X_t \in [0, 1]$, $\boldsymbol{\theta} = (\alpha, \beta, \sigma) = (0.9, 0.30.075)$, $X_0 = 0.15$ and $, stepsize = 1/250$ for fitting. Algorithm 7 and Figure 16 contains the code and plotted true and approximate densities respectively (that is, the true density, EM distribution, Hermite transition approximation for $K = 1, 2$, and the Saddlepoint/Cumulant truncation approximate transition density). It can be seen that the Cumulant truncated/ Saddlepoint approximation method is the best fit to the true density.

3.3.6 Inferenece on a diffusion process

Once an anlytical expression is obtained, maximum likelihood estimation (MLE) can be executed on the true, if available, approximate transition density. In this paper the sadlepoint approximation wil be subject to MLE, in order to show the efficiency of the method when a true densioty is not available. Assuming normality in the residual distribution, for an observed dataset with n observations the likelihood function is given by:

$$L(\boldsymbol{\theta}|\boldsymbol{X}) = \prod_{i=1}^{n}(p_X(X_i, i|X_s, s; \boldsymbol{\theta})), \tag{62}$$

for $\boldsymbol{X} = (X_s, ..., X_T)$. Define the log-likelihood function as:

$$\log(L(\boldsymbol{\theta}|\boldsymbol{X})) = \log\Big\{\prod_{i=1}^{n}(p_X(X_i, i|X_s, i-1; \boldsymbol{\theta}))\Big\} = \sum_{i=1}^{n}\log\Big\{(p_X(X_i, i|X_s, i-1; \boldsymbol{\theta}))\Big\}. \tag{63}$$

To find the maximum likelihood estimators, Equation 63 needs to be maximized, to obtain$\hat{\boldsymbol{\theta}}_{max}^{mle}$, to obtain the maximum likelihood estimators. I.e.

$$\hat{\boldsymbol{\theta}}_{max}^{mle} \leftarrow\!\shortmid \max_{\boldsymbol{\theta}}\Bigg(\log(L(\boldsymbol{\theta}|\boldsymbol{X}))\Bigg) = \max_{\boldsymbol{\theta}}\Bigg(\sum_{i=1}^{n}\log\Big\{p_X(X_i, i|X_s, i-1; \boldsymbol{\theta})\Big\}\Bigg),$$

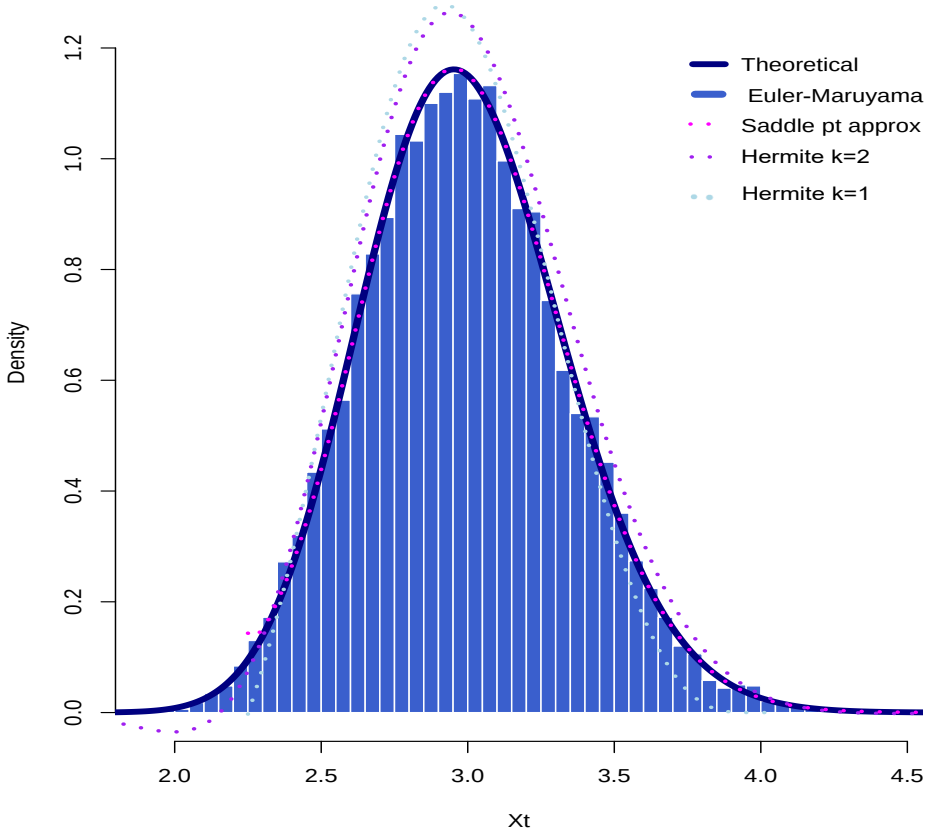


Figure 16: univariate CIR Process's Theoretical density, EM distribution, Hermite transition approximation for $K = 1, 2$, as well as the Saddlepoint/Cumulant truncation approximate transition density

is to be found which maximizes the log-likelihood function. Since diffusion processes have the Markov property the likelihood function can also be defined as,[21]:

$$L(\boldsymbol{\theta}|\boldsymbol{X}) = \prod_{i=1}^{n}(p_X(X_i, i|X_s, i-1; \boldsymbol{\theta})) = p_X(X_s, s; \boldsymbol{\theta}) \prod_{i=1}^{n}(p_X(X_i, i|X_{i-1}, i-1; \boldsymbol{\theta}).$$

For an approximate density $\tilde{p}_X(X_i, i|X_s, i-1; \boldsymbol{\theta})$ is used instead of $p_X(X_i, i|X_s, i-1; \boldsymbol{\theta})$. For example, when perfroming inference by use of the cumulant truncated transition density approximation, parameter estimates is obtained by:

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{max}^{mle} &\hookleftarrow \max_{\boldsymbol{\theta}}\left(\log(L(\boldsymbol{\theta}|\boldsymbol{X}))\right) \\
&= \max_{\boldsymbol{\theta}}\left(\sum_{i=1}^{n}\log\left\{p_X^{saddle}(X_t, i|X_s, i-1; \boldsymbol{\theta})\right\}\right) \\
&= \max_{\boldsymbol{\theta}}\left(\sum_{i=1}^{n}\log\left\{\exp(\widetilde{K}_X(t) - tx_t)\sqrt{(2\pi\widetilde{K}_X''(t))^{-1}}\right\}\right),
\end{aligned}$$

Maximum likelihood estimation will be conducted for a model of a financial time series on financial data in order to obatin the parameter of best fit for inferential purposes.

Example 23. Univariate CIR diffusion process inference on the Chicago Board Options Exchange Volatility Index (VIX Index)

The VIX Index is a volatility benchmark based on market estimates of th expected volatility of the S&P500 Index, calculated using the mid option bid/ask price quotes.Consider the VIX index's volatility values from 31 August 2020 to 31 August 2021, as can be seen in the timeplot in Figure 17. MLE on the Cumulant Truncated Approximate density for a univariate CIR model has been applied to the data, with the CIR difussion process as the underlying model:

$$dX_t = \kappa(\alpha - X_t)dt + \sigma\sqrt{X_t}dW_t. \tag{64}$$

Based on the Saddlepoint approximation derived in Example 12, MLE performed on 1 year's volatolity values. $\hat{\boldsymbol{\theta}}_{Saddle}^{mle}$ coverged to the maximum likelihood estimators $\hat{\boldsymbol{\theta}}_{Saddle}^{mle} = (\hat{\kappa}, \hat{\alpha}, \hat{\sigma}) = (22.27, 21.46, 32.65)$. The MLE procedure was initiated at $(50, 50, 50)$. The data was obtained from Bloomberg.

Figure 17 plots the fitted cumulant truncated/saddlepoint density, based on $\hat{\boldsymbol{\theta}}_{Saddle}^{mle} = (\hat{\kappa}, \hat{\alpha}, \hat{\sigma}) = (22.27, 21.46, 32.65)$. See Algorithm 12 for the R code used to plot the time series. Figure 18 displays the univariate CIR fitted saddlepoint density based on the MLE values. See Algorithm 12 for the R code used to plot the saddlepoint approximate density.

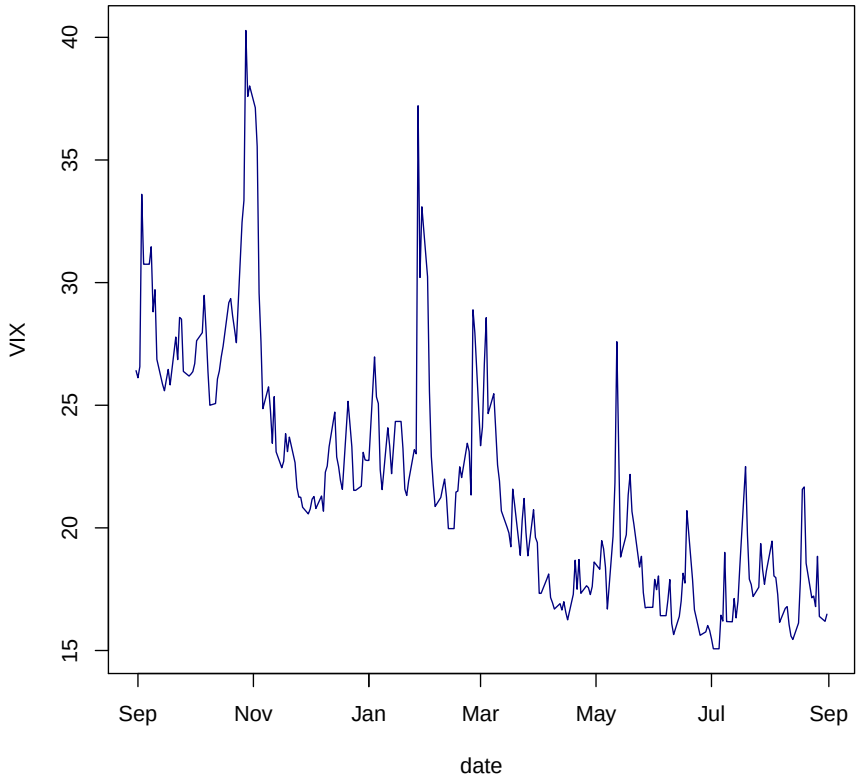


Figure 17: VIX index's volatility values from 31 August 2020 to 31 August 2021

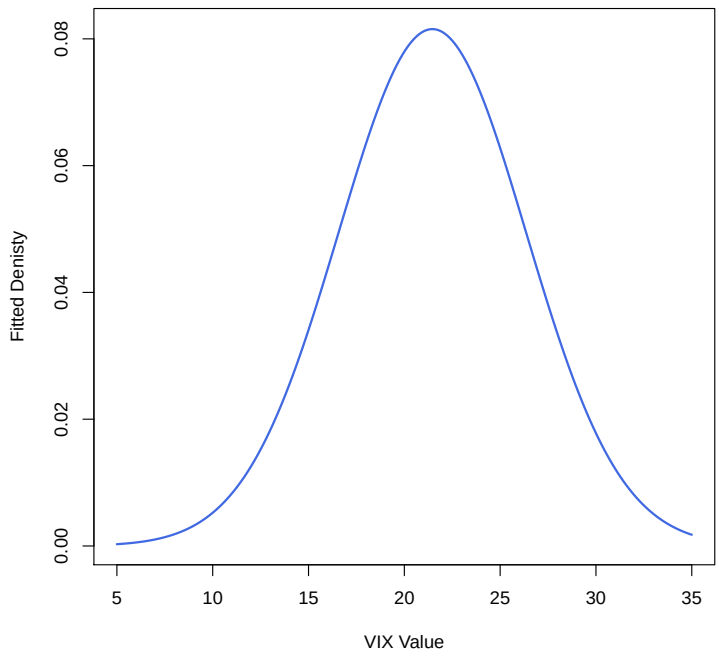


Figure 18: univariate CIR fitted saddlepoint density based on the MLE values.

4 Transition densities in the multivariate state variable case

To infer the implications of a process evolution over infintesmall time-epochs relies heavelly on the process'stransition density. However a true transition density rarerly exists in closed-form. This section is dedicated in exploring the methodologies for obtaining closed-form approximations for the true multivariate transition densities of om multivariate diffusion processes. The strenghts and limitations of the method will be explored and utilized to develop an efficient method of getting an approximation to a given multivariate process' transition density function. An obtained closed-form expression for a true transition density can be used for a variety of statistical procedures and inferences on data, i.e. financial and rates data as will be explored in this paper. With many financial models encapsolating various state variables, there is sufficient need to explore the multivariate case.

4.1 Multivariate Hermite Expansion method

Firstly the assumptions and model will be specified. Reducibility of a diffusion process and necessary and suficient conditions for redulcibility of a multivariate diffusion process will be discussed. In the current paper, the technique applied for getting explicit Hermite expansions, in the univariate diffusion case will be expanded to multivariate diffusions. The Hermite expansion method can be easily extended to the multivariate diffusion scenario, if the diffusion process is reducible, unfortately not alll of these processes are reducible in natrue. Therefore, another method will be introduced, where the cofficients (in closed-form) satisfying the Kolmogorov equations, are determined, [3].The method of closed-form log-likelihood expansions, as developed in [3], is based on the explicit calculation of the coefficients of a process's expansion based on the process unique structure. It should be noted that this method, provided that the multivariate diffusion process at hand is reducible, is an expansion to the univeriate Hermite expansion method. The Markov property, which diffusion processes inheret, allows for the construction of log-likelihood transition expansions over discretized time epochs to be reduced to the sum of the log-likelihod transition expansion function over consequtive observations. Quasi-likelohood inference are therefore made possible.

4.1.1 Assumptions and setup

The following time-homogeneous diffusion process are considered:

$$d\boldsymbol{X}_t = \boldsymbol{\mu}(\boldsymbol{X}_t, t, \boldsymbol{\theta})dt + \boldsymbol{\sigma}(X_t, t, \boldsymbol{\theta})d\boldsymbol{W}_t, \tag{65}$$

s.t $t \in [s, T]$, with $\boldsymbol{X}_t : m \times 1$ is the state vector of interest, $\mu(\boldsymbol{X}_t, t, \boldsymbol{\theta}) : m \times 1$ and $\sigma(\boldsymbol{X}_t, t, \boldsymbol{\theta}) : m \times m$ the diffusion vector and covariance matrix respectively. $\boldsymbol{W}_t : m \times 1$ is a vector of independent

Brownian Motions. This equation is equivalent to Equation 13, were $\boldsymbol{\theta}$ represents the parameter vector. Independence, without loss of generality, through the structuring of $\boldsymbol{\sigma}(\boldsymbol{X_t}, t, \boldsymbol{\theta})$, is assumed. The time variable, t , can be studied in both the time-inhomogeneous case, as well as the time homogeneous case, where t is considered as an additional state variable. The premise is to derive an approximate conditional density of $\boldsymbol{X}_{t+\delta} = \boldsymbol{x}$, given the initial condition $\boldsymbol{X}_t = \boldsymbol{x}_0$. The determination of an approximate transition density function, $\widetilde{P}_{\boldsymbol{X}}(\boldsymbol{x}|\boldsymbol{x}_0, \delta)$, for $P_{\boldsymbol{X}}(\boldsymbol{x}|\boldsymbol{x}_0, \delta)$, allows for inference to be conducted on the transition density profile and parameter structure, through maximum likelihood estimation, MLE.

Assume the parameterization of $\mu(\boldsymbol{X_t}, t)$, functionally dependent on $\boldsymbol{\theta}$, the parameter vector, where $\boldsymbol{X}_t$ is observed at dates $\{t = i\delta, i = 0, ..., n\}$ $s.t.$ $\delta > 0$. Considering Equation 65, inheriting the Markov Property, it can be implied that the log-likelihood function can be represented as:

$$L_n(\boldsymbol{\theta}, \delta) = \sum_{i=1}^{n} ln(P_X(\boldsymbol{X}_{i\delta}|\boldsymbol{X}_{(i-1)\delta}, \delta)). \tag{66}$$

It is of key importance to note the a closed-form function for $P_{\boldsymbol{X}}(\boldsymbol{x}|\boldsymbol{x}_0, \delta)$, and hence, $ln(P_X(\boldsymbol{x}|\boldsymbol{x}_0, \delta))$,does not necessarily exist in closed-form, which emphasizes the importance to develop an approximate closed-form function for the preceeding functions.

Now, let $\zeta_{\boldsymbol{X}} \subseteq \mathbb{R}^m$, represent the domain of $\boldsymbol{X}_t$. Define the Jacobian, for function $\eta(\boldsymbol{x})$, differentiable in $x.$ as

$$\delta\eta(\boldsymbol{x}) = \frac{\partial_{\eta_i}(\boldsymbol{x})}{\partial \boldsymbol{x}_j},$$
$$s.t.\ i = 1, ..., d\ and\ j = 1, ..., m.$$

Assume that $\zeta_{\boldsymbol{X}} \subseteq \mathbb{R}^m$ is the product of m intervals within open limits $(-\infty, +\infty)$.

The variance-covariance matrix, $\gamma(\boldsymbol{X_t}, t) : m \times m$, can be parameterized for use instead of $\sigma(\boldsymbol{X_t}, t)$, where:

$$\gamma(\boldsymbol{X_t}, t) = \sigma(\boldsymbol{X_t}, t)\sigma^T(\boldsymbol{X_t}, t), \tag{67}$$

where $\sigma^T(\boldsymbol{X_t}, t)$ denotes the transpose of $\sigma(\boldsymbol{X_t}, t)$. As such the transition density of the process depends on (μ, ν), and it can be shown that there exist exist a spectrum of solutions to Equation 67, for all σ. This can be viewed by the generator function, $\boldsymbol{A_X}$, of the process in which depends on ν. For function, $\boldsymbol{f}(\delta, \boldsymbol{x})$, differentiable on its domain, the function $\boldsymbol{A_X} \bullet f$ can be defined as:

$$\boldsymbol{A_X} \bullet f = \frac{\partial \boldsymbol{f}(\delta, \boldsymbol{x})}{\partial \boldsymbol{x}} + \sum_{i=1}^{m} \mu_i(x) \frac{\partial \boldsymbol{f}(\delta, \boldsymbol{x})}{\partial x_i} + \frac{1}{2} \sum_{i=1}^{m} \nu_{ij}(\boldsymbol{x}) \frac{\partial^{\boldsymbol{2}} \boldsymbol{f}(\delta, \boldsymbol{x})}{\partial x_i \partial y_i}. \tag{68}$$

The domain of $\boldsymbol{A_X}$ includes sufficiently differntiable functions, which will be of importance in the speci-

fication of the likelihood function. As result define $\Lambda_\upsilon(x)$ as:

$$\Lambda_\nu(x) = \frac{1}{2} ln(Det[\nu(x)]),$$

where the $\nu(x)$ matrix is assumed to be positive definite for all $x \in \zeta_{\boldsymbol{X}} \subseteq \mathbb{R}^m$.
Other assumptions to ensure that an unique solution to Equation 65 includes:

- $d\boldsymbol{X}_t$ is reducible,
- $\mid \gamma(\boldsymbol{x}) \mid = \mid \gamma(\boldsymbol{X_t}, t) \mid > 0$ for all $\boldsymbol{X} \in \zeta_{\boldsymbol{X}}$,
- $\gamma(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ are infinitely differentiable for all $\boldsymbol{x} \in \zeta_{\boldsymbol{X}}$, which implies the uniqueness of the solution and that the coefficients are Lipschitz ,
- linear growth is satisfied in the drift and diffusion functions, i.e there exists a K s.t $K \in \mathbb{N}$for all $\boldsymbol{x} \in \zeta_{\boldsymbol{X}}$and for all i, j s.t.

  $$\mid \mu_i(\boldsymbol{x}) \mid \leqslant K(1 + \parallel \boldsymbol{x} \parallel)$$

  and

  $$\mid \sigma_{ij}(x) \mid \leqslant K(1 + \parallel \boldsymbol{x} \parallel),$$

  where $\parallel x \parallel$, for all $\boldsymbol{x} \in \mathbb{R}^m$ denotes the euclidean norm. This assumption ensures the existence of a solution to the equation,
- the diffusion process $\boldsymbol{X}_t$ is fully defined by the drift and diffusion functions, $\boldsymbol{\mu}(\boldsymbol{X}_t, t)$ and $\boldsymbol{\sigma}(\boldsymbol{X}_t, t)$, within and at the boundaries of $\zeta_{\boldsymbol{X}}$.

.Where possible the diffusion, under consideration, will be transformed into a reducible diffusion.

Definition 24. Reducibility: a diffusion $\boldsymbol{X}$ is reducible if and only if a one to one transformation from diffusion $\boldsymbol{X}$ into Y, where $\boldsymbol{\sigma_Y}$ is the identity matrix. There exists an infinitely differentiable, invertible function $\boldsymbol{\Gamma(X)}$, for $\boldsymbol{X} \in \zeta_{\boldsymbol{X}}$, s.t. $\boldsymbol{Y_t \equiv \Gamma(X)}$satisfies the following SDE

$$d\boldsymbol{Y}_t = \mu_{\boldsymbol{Y}}(\boldsymbol{Y}_t, t)dt + d\boldsymbol{W}_t, \tag{69}$$

on domain $\zeta_{\boldsymbol{Y}} \subseteq \mathbb{R}^m$.

Itô's lemma implies that a for a reducible diffusion, the change of variable, γ (also known as the Lamperti Transform), satisfies:

$$\nabla\gamma(x) = \sigma^{-1}(x), \tag{70}$$

s.t $\sigma^{-1}(x)$ denotes the inverse of $\sigma(x)$.

All univariate diffusion processes are reducible, through the implementation of the following transformation

$$Y_t = \lambda(X_t) = \int^{X_t} \frac{1}{\sigma(u)} du. \tag{71}$$

.

Therefore for the univariate case, the Hermite expansion can be utilized to get a closed-form expansion for the density of Y, and therefore by transformation, for X can be obtained. As a result a closed-form expansion can be obtained for P_X,followed by P_Y. Unfortunately, not all multivariate diffusions are reducible. The reducibility of a multivariate diffusion depends on the specification of the diffusion matrix, $\boldsymbol{\sigma}$. We can express reducibility in the multivariate case, by means of the following proposition.

Proposition 25. *A diffusion, $\boldsymbol{X}$ is reducible, if and only if*

$$\sum_{l=1}^{m} \frac{\partial \sigma_{ik}(x)}{\partial x_l} \sigma_{lj}(x) = \sum_{l=1}^{m} \frac{\partial \sigma_{ij}(x)}{\partial x_l} \sigma_{lk}(x),$$

for all $\boldsymbol{x} \in \zeta_{\boldsymbol{x}} \subseteq \mathbb{R}^m$, where $i, j, k = 1, 2, ..., m$ s.t $k > j$. If $|\boldsymbol{\sigma}| > 0$ then the above expression can be expressed as

$$\frac{\partial}{\partial x_k} \sigma_{ij}^{-1}(x) = \frac{\partial}{\partial x_j} \sigma_{ik}^{-1}(x).$$

For example, in the bivariate case ($m = 2$), it follows that:

$$\frac{\partial}{\partial x_2} \sigma_{11}^{-1}(x) - \frac{\partial}{\partial x_1} \sigma_{12}^{-1}(x) = 0$$

and

$$\frac{\partial}{\partial x_2} \sigma_{21}^{-1}(x) - \frac{\partial}{\partial x_1} \sigma_{22}^{-1}(x) = 0,$$

therefore

$$\frac{\partial}{\partial x_2} \sigma_{11}^{-1}(x) - \frac{\partial}{\partial x_1} \sigma_{12}^{-1}(x) = \frac{\partial}{\partial x_2} \sigma_{21}^{-1}(x) - \frac{\partial}{\partial x_1} \sigma_{22}^{-1}(x).$$

In the multivariate case, when the diffusion is reducible, two techniques will be discussed for deriving closed-form expansions for the log-likelihood function. The first method focuses on the computation of the coefficients for the Hermite expansion for the transition density of the $\boldsymbol{Y}$, namely $P_{\boldsymbol{Y}}$. The coefficients are determined through a series expansion in the timeepochs between observations, namely $\triangle$. The second method, also using the Hermite series expansion, determines the coefficients, by solving the Kolmogorov

partial differential equations characterizing transition density function $P_{\boldsymbol{Y}}$. In both methods, the reversal of the change of variable in the Jacobian formula yields the transition density approximation for $P_{\boldsymbol{X}}$.

4.1.2 Closed-form likelihood expansion of reducible diffusions

For a reducible diffusion process, two methods for the expansion of the log-likelihood function can be constructed. Firstly, the coefficients of a Hermite expansion for $P_Y$can be computed. The coefficients are observed and computed by means of a series expansion in δ, the time between observations.The second method, is in the form of a Hermite series, and relies on obtaining the coefficients by solving the Kolmogorov partial differential equations (P.D.E), which specify the dynamics of P_Y.By reversing the change of variable, γ, and the Jacobian, both methods yield P_X, given the series for $P_Y$is obtained.

4.1.3 Multivariate Hermite expansions

Consider the multivariate counterpart to the univariate Hermite expansion as done in [2]. Let $\theta(x)$ denote the density function of an n-dimensional multivariate normal distribution, with mean zero and the covariance matrix being the identity matrix. For every vector $h = (h_1, ..., h_n) \in \mathbb{N}^n$. Denote $H_h(x)$ as the associated Hermite polynomials, such that

$$H_h(x) = \left[\frac{(-1)^{tr(h)}}{\theta(x)}\right]\left[\frac{\partial^{tr(h)}\theta(x)}{\partial x_1^{h_1}...\partial x_n^{h_n}}\right],$$

which can be explicitly computed to an arbitrary order of $tr(h)$. The polynomials are orthonormal, such that $\int_{\mathbb{R}^n} H_h(x)H_k(x)\theta(x)dx = h_1!...h_n!$ if $h = k$ and 0 otherwise. The Hermite series approximation for p_Y follows:

$$\tilde{p}_Y^{(J)}(y \mid y_0, \Delta) = \Delta^{\frac{-m}{2}}\theta\left(\frac{y - y_0}{\Delta^{\frac{1}{2}}}\right)\sum_{h\in\mathbb{N}^n, tr(h)\leq J}\eta_h(\Delta, y_0)H_h\left(\frac{y - y_0}{\Delta^{\frac{1}{2}}}\right), \tag{72}$$

where $\eta_h(\bigtriangleup, y_0)$, the Hermite coefficients, can be computed in the same manner as in the univariate case. That is, via the orthonormality of the Hermite polynomials, the conditional expectation yields the coefficients η_h:

$$\eta_h(\Delta, y_0) = \frac{1}{h_1!...h_n!}E\left[H_h\left(\Delta^{-\frac{1}{2}}(Y_{t+\Delta} - y_0) \mid Y_t = y_0\right)\right]. \tag{73}$$

This expression can be amended in computing an expansion in Δ, using a generator function. The conditional expectation can be evaluated , by using the deterministic Taylor expansion:

$$E_{Y_1}[f(Y_\Delta, Y_0, \Delta)|Y_0 = y_0] = \sum_{k=0}^{K}\frac{\Delta^k}{k!}A_Y^k.f(y, y_0, \delta)_{|y=y_0,\delta=0} + O(\Delta^{K+1}), \tag{74}$$

such that $A_Y$is an infinitesimal generator function of Ysuch that

$$A_Y.f = \frac{\partial f(y,\Delta)}{\partial \Delta} + \sum_{i=1}^{n} \mu_i(y) \frac{\partial f(y,\Delta)}{\partial y_i} + \frac{1}{2} \sum_{i,j=1}^{n} \upsilon_{kl}(y) \frac{\partial^2 f(y,\Delta)}{\partial y_i \partial y_j}, \tag{75}$$

where $A_Y$depends on ν rather than σ, where ν is positive definite for all $y \in \delta_Y$. Function f is differentiable on (y, δ) and iterable, by application of A_Y K times in A'_Ys domain. By the replacement of the unknown function $\eta_h$in Equation 72 by the expansion around Δ to order K the expansion of $\tilde{p}_Y^{(J)}$ can be obtained. The coefficients are obtained in increasing powers of Δ, which will be denoted by $\tilde{p}_Y^{(J,K)}$. A truncated series, in Δ, for $\tilde{p}_Y^{(J,K)}$ can be obtained by rewriting the terms of Equation 72:

$$\tilde{p}_Y^{(J,K)}(y|y_0,\Delta) = \Delta^{-\frac{m}{2}} \theta\left(\frac{y-y_0}{\Delta^{\frac{1}{2}}}\right) \sum_{k=0}^{K} c_Y^{(J,k)}(y|y_0) \frac{\Delta^k}{k!}. \tag{76}$$

Where the log-transition density function, for all J, similarly in the univariate case where the Hermite series is obtained as J tends to infinity, the following expression can be obtained:

$$l_Y^{(K)}(y|y_0,\Delta) = -\frac{m}{2} ln(2\pi\Delta) + \frac{C_Y^{(-1)}(y|y_0)}{\Delta} + \sum_{k=0}^{K} C_Y^{(k)}(y|y_0) \frac{\Delta^k}{k!}, \tag{77}$$

such that the coefficients $C_Y^{(k)}$, where $k = -1, 0, 1, 2, ..., K$, are combinations of the coefficients of Equation 72, through the identification of terms in $\triangle$ for the log of Equation 76. This approach is a natural extension of the Univariate Hermite Expansion method. However, the Hermite expansion approach a requires a reducible diffusion. Albeit all Univariate diffusions are reducible, not all multivariate diffusions are. This leads to the development of the following alternative method, namely the Connection to Kolmogorov Equations.

4.1.4 Connection to Kolmogorov Equations

As an alternative approach, a closed-form expansion for $l_Y(y|y_0,\Delta)$, can be obtained by using Equation 77 and solving the Kolmogorov equations. Therefore consider the forward and backward Kolmogorov equations:

$$\frac{\partial}{\partial \Delta} p_Y(y|y_0,\Delta) = -\sum_{i=1}^{m} \frac{\partial}{\partial y_i} \mu_{Y_i}(y) p_Y(y|y_0,\Delta) + \frac{1}{2} \sum_{i=1}^{m} \frac{\partial^2}{\partial y_i^2} p_Y(y|y_0,\Delta) \tag{78}$$

$$\frac{\partial}{\partial \Delta} p_Y(y|y_0,\Delta) = \sum_{i=1}^{m} \frac{\partial}{\partial y_{0i}} \mu_{Y_i}(y_0) p_Y(y|y_0,\Delta) + \frac{1}{2} \sum_{i=1}^{m} \frac{\partial^2}{\partial y_{0i}^2} p_Y(y|y_0,\Delta). \tag{79}$$

Using the forward equation, the equivalent form for l_Y is:

$$\frac{\partial}{\partial \Delta} l_Y(y|y_0,\Delta) = -\sum_{i=1}^{m} \frac{\partial}{\partial y_i} \mu_{Y_i}(y) - \sum_{i=1}^{m} \mu_{Y_i}(y) l_Y(y|y_0,\Delta) + \frac{1}{2} \sum_{i=1}^{m} \frac{\partial^2}{\partial y_i^2} l_Y(y|y_0,\Delta) + \frac{1}{2} \sum_{i=1}^{m} \left(\frac{\partial}{\partial y_i} l_Y(y|y_0,\Delta)\right)^2. \tag{80}$$

By substituting Equation 77 into Equation 80, the following set of equations is obtained:

$$\frac{\partial}{\partial\Delta}l_Y^{(K)}(y|y_0,\Delta) = -\frac{1}{\Delta^2}C_Y^{(-1)}(y|y_0) - \frac{m}{2\Delta} + \sum_{k=1}^{K-1} C_Y^{(k)}(y|y_0)\frac{\Delta^{k-1}}{(k-1)!}$$

$$\frac{\partial}{\partial y_i}l_Y^{(K)}(y|y_0,\Delta) = \frac{1}{\Delta}\frac{\partial}{\partial y_i}C_Y^{(-1)}(y|y_0) + \sum_{k=0}^{K} \frac{\partial}{\partial y_i}C_Y^{(-1)}(y|y_0)\frac{\Delta^k}{k!}$$

$$\frac{\partial^2}{\partial y_i^2}l_Y^{(K)}(y|y_0,\Delta) = \frac{1}{\Delta}\frac{\partial^2}{\partial y_i^2}C_Y^{(-1)}(y|y_0) + \sum_{k=0}^{K} \frac{\partial^2}{\partial y_i^2}C_Y^{(-1)}(y|y_0)\frac{\Delta^k}{k!}.$$

By Equating the coefficients of $\frac{1}{\Delta^2}$ on both sides of Equation 80 , the leading coefficient in the expansion $C_Y^{(-1)}$ solves the non-linear equation:

$$C_Y^{(-1)}(y|y_0) = -\frac{1}{2}\left(\frac{\partial}{\partial y_i}C_Y^{(-1)}(y|y_0)\right)^T\left(\frac{\partial}{\partial y_i}C_Y^{(-1)}(y|y_0)\right). \tag{81}$$

The approximate solution is strictly maximized at $y = y_0$,since transition density approximates the Normal Density as $\Delta \to 0$, that is:

$$C_Y^{(-1)}(y|y_0) = -\frac{1}{2} \parallel y - y_0 \parallel^2 = -\frac{1}{2}\sum_{i=1}^{m}(y_i - y_{0i})^2. \tag{82}$$

Considering the coefficients of $\frac{1}{\Delta}$ on both sides of Equation 80, it follows that:

$$\sum_{i=1}^{m}\frac{\partial}{\partial y_i}C_Y^{(0)}(y|y_0)(y_i - y_{0i}) = \sum_{i=1}^{m}\mu_{Y_i}(y)(y_i - y_{0i})$$

..

By integrating between y_0 and y, $C_Y^{(0)}(y|y_0)$ equates to

$$C_Y^{(0)}(y|y_0) = \sum_{i=1}^{m}(y_i - y_{0i})\int_0^1 \mu_{Y_i}(y)(y_0 + v(y - y_0))dv. \tag{83}$$

The higher order coefficients are obtained in a similar fashion.

Theorem 26. *The coefficients of $l_Y^{(K)}(y|y_0,\Delta)$ are given by Equation 82 and 83 and for all $k \geq 1$,*

$$C_Y^{(k)}(y|y_0) = k\int_0^1 G_Y^{(k)}(y)(y_0 + v(y - y_0) \mid y_0)v^{k-1}dv, \tag{84}$$

where $G_Y^{(k)}$ is obtained by

$$\begin{aligned} G_Y^{(1)}(y|y_0) = \quad & -\textstyle\sum_{i=1}^{m}\frac{\partial}{\partial y_i}\mu_{Y_i}(y) - \sum_{i=1}^{m}\mu_{Y_i}(y)\frac{\partial}{\partial y_i}C_Y^{(0)}(y|y_0) \\ & +\tfrac{1}{2}\textstyle\sum_{i=1}^{m}\left(\frac{\partial^2}{\partial y_i^2}C_Y^{(0)}(y|y_0) + \left(\frac{\partial}{\partial y_i}C_Y^{(0)}(y|y_0)\right)^2\right), \end{aligned} \tag{85}$$

and where $k \geq 2$

$$\begin{aligned} G_Y^{(k)}(y|y_0) = \quad & -\sum_{i=1}^m \mu_{Y_i}(y)\frac{\partial}{\partial y_i}C_Y^{(k-1)}(y|y_0) + \frac{1}{2}\sum_{i=1}^m \frac{\partial^2}{\partial y_i^2}C_Y^{(k-1)}(y|y_0) \\ & +\frac{1}{2}\sum_{i=1}^m \sum_{h=0}^{k-1}\begin{pmatrix} k-1 \\ h \end{pmatrix}\frac{\partial}{\partial y_i}C_Y^{(h)}(y|y_0)\frac{\partial}{\partial y_i}C_Y^{(k-1-h)}(y|y_0). \end{aligned} \tag{86}$$

The closed-form of $l_Y^{(K)}$ that solves the Kolmogorov equations for all orders of Δ, i.e Δ^k,$k = 1, 2, ..., K$, are then consequently obtained.

4.1.5 Change of variable

By use of the Jacobian formula, and given the obtained expression l_Y, the expression for l_X is obtained as such:

$$\begin{aligned} l_X(x|x_0, \Delta) = \quad & -\tfrac{1}{2}ln(Det[v(x)]) + l_Y(\Delta, \gamma(x)|\gamma(x_0)) \\ & = D_v(x) + l_Y(\Delta, \gamma(x)|\gamma(x_0)). \end{aligned} \tag{87}$$

$l_X^{(K)}$, for all orders K, in Δ is therefore defined as:

$$\begin{aligned} l_X^{(K)}(x|x_0, \Delta) &= -D_v(x) + l_Y^{(K)}(\Delta, \gamma(x)|\gamma(x_0)) \\ &= -\tfrac{m}{2}\ln(2\pi\Delta) - D_v(x) + \frac{C_Y^{(-1)}(\gamma(x)|\gamma(x_0))}{\Delta} + \sum_{k=0}^K C_Y^{(k)}(\gamma(x)|\gamma(x_0))\frac{\Delta^k}{k!}, \end{aligned} \tag{88}$$

where $l_Y^{(K)}$ is given as in Equation 77, and by utilizing the coefficients, $C_Y^{(k)}$, for all $k = -1, 0, 1, ..., K-1, K$. It therefore follows that the Kolmogorov equations is solved by $l_X^{(K)}$ for X, for all $k = -1, 0, 1, ..., K-1, K$.

4.1.6 Closed-form log-likelihood expansion of irreducible diffusions

For reducible diffusions, the Hermite method and solving of the Kolmogorov equations are equivalent. Unfortunately the upfront transformation $X \to Y$, followed by the computation of $l_X \to l_Y$, via the Jacobian formula, is no longer viable. However it it is possible to derive an expansion for l_X and to determine that the coefficients satisfy the Kolmogorov equations, for all $k = -1, 0, 1, ..., K-1, K$. The preceding can be accomplished by the following approach:

By considering the structure of the expansion around Δ, Equation 88 as per the reducible case, the postulation of the for an expansion of the log-likelihood is obtained by:

$$l_X^{(K)}(x|x_0, \Delta) = -\tfrac{m}{2}\ln(2\pi\Delta) - D_v(x) + \frac{C_X^{(-1)}(x|x_0)}{\Delta} + \sum_{k=0}^K C_X^{(k)}(x|x_0)\frac{\Delta^k}{k!} \ . \tag{89}$$

By using the Kolmogorov equations solutions for the coefficients can also be obtained. For the irreducible

case, for the process X, the equations can be expressed as follow:

$$\begin{aligned}
& & -\sum_{i=1}^{m}\frac{\partial}{\partial x_i}\mu_i(x)+\frac{1}{2}\sum_{i,j=1}^{m}\frac{\partial^2}{\partial x_i\partial x_j}\upsilon_{ij}(x) \\
\frac{\partial}{\partial\Delta}l_X^{(K)}(x|x_0,\Delta) &= & -\sum_{i=1}^{m}\mu_i(x)\frac{\partial}{\partial x_i}l_X^{(K)}(x|x_0,\Delta)+\sum_{i,j=1}^{m}\frac{\partial}{\partial x_i}\nu_i(x)\frac{\partial}{\partial x_j}l_X^{(K)}(x|x_0,\Delta) \\
& & +\frac{1}{2}\left(\sum_{i,j=1}^{m}\left(\nu_{ij}(x)\frac{\partial^2}{\partial x_i\partial x_j}l_X^{(K)}(x|x_0,\Delta)+\frac{\partial}{\partial x_i}l_X^{(K)}(x|x_0,\Delta)\nu_{ij}(x)\frac{\partial}{\partial x_j}l_X^{(K)}(x|x_0,\Delta)\right)\right)
\end{aligned} \tag{90}$$

$$\begin{aligned}
\frac{\partial}{\partial\Delta}l_X^{(K)}(x|x_0,\Delta) &= & \sum_{i=1}^{m}\mu_i(x_0)\frac{\partial}{\partial x_{0i}}l_X^{(K)}(x|x_0,\Delta) \\
& & +\frac{1}{2}\left(\sum_{i,j=1}^{m}\left(\nu_{ij}(x_0)\frac{\partial^2}{\partial x_{0i}\partial x_{0j}}l_X^{(K)}(x|x_0,\Delta)+\frac{\partial}{\partial x_{0i}}l_X^{(K)}(x|x_0,\Delta)\nu_{ij}(x_0)\frac{\partial}{\partial x)_j}l_X^{(K)}(x|x_0,\Delta)\right)\right),
\end{aligned} \tag{91}$$

In order to obtain a solution, the following method is applied: similarly to the reducible case, the substitution of Equation 89 into Equation 90 yields an equation for all orders $k=-1,0,1,...,K-1,K$ in Δ which is solved for the respective coefficients. Although the differential equation for l_X is nonlinear, through exponentiation it can be transformed into a linear equation, and hence the expansion $l_X^{(K)}(x|x_0,\Delta)$ will approximate l_X. Begin with with equation with order Δ^{-2}, which determines the leading order coefficient $C_X^{(-1)}(x|x_0)$. Where the leading coefficient, in the reducible case, is simply given by

$$C_X^{(-1)}(x|x_0)=-\frac{1}{2}||\gamma(x)-\gamma(x_0)||^2,$$

the irreducible case is more complicated. The equation determining the coefficient $C_X^{(-1)}(x|x_0)$ is derived by equating the terms of order Δ^{-2} in Equation 90:

$$C_X^{(-1)}(x|x_0)=-\frac{1}{2}\left(\frac{\partial}{\partial x}C_X^{(-1)}(x|x_0)\right)^T\nu(x)\left(\frac{\partial}{\partial x}C_X^{(-1)}(x|x_0)\right), \tag{92}$$

which yields a good geometric interpretation to the solution of the equation in $\mathbb{R}^m$.

4.1.7 Time and state expansion

The structure of $C_X^{(-1)}(x|x_0)$ implies that it would be near impossible to get an explicit characterization of the coefficients of the expansion in question, since Equation 92 won't generally yield an explicit solution. Therefore an explicit approximation in $(x-x_0)$ of $C_X^{(-1)}(x|x_0)$ will be derived:

Consider a quadratic approximation (around $(x-x_0)$) for the solution of Equation 92, which determines $C_X^{(-1)}(x)$. The non-singularity of $\nu(x)$ implies the constant and linear terms are zero. The 2^{nd}order expansion is written as

$$C_X^{(-1)}(x|x_0)=-\frac{1}{2}(x-x_0)^TV(x-x_0)+\epsilon(\parallel x-x_0\parallel^2).$$

Equation 92 implies the Equation

$$V = V\nu(x_0)V,$$

with solution

$$V = \nu^{-1}(x_0).$$

Consequently the leading term of $C_X^{(-1)}(x|x_0)$ around $(x - x_0)$ is

$$-\frac{1}{2}\Delta(x - x_0)^T\nu(x_0)(x - x_0),$$

such that the leading term of the log-likelihood expansion corresponds to a $N(x_0, \Delta\nu(x_0))$ distribution. Generally, for each $k = -1, 0, 1, ..., K$, a series around $(x - x_0)$, for each $C_X^{(k)}$ at an order j_k will be derived, i.e. $C_X^{(j_k,k)}$. Therefore, note that

$$X_\Delta - X_0 = U_p(\Delta^{\frac{1}{2}}),$$

such that

$$\left|C_X^{(k)}(X_\Delta \mid X_0)\Delta^k - C_X^{(j_k,k)}(X_\Delta \mid X_0)\Delta^k\right| = U_p\Big(\parallel X_\Delta - X_0 \parallel^{j_k} \Delta^k\Big) = U_p\Big(\Delta^{\frac{j_k}{2}+k}\Big).$$

Setting $\frac{j_k}{2} + k = K + 1$, will yield an approximation error due to the expansion around $(x - x_0)$ of the same order Δ^{K+1} for each term in Equation 89. As result the expansion take form:

$$\tilde{l}_X^{(K)}(x|x_0, \Delta) = -\frac{m}{2}\ln(2\pi\Delta) - D_\nu(x) + \frac{C_X^{(j_{-1},-1)}(x|x_0)}{\Delta} + \sum_{k=0}^{K} C_X^{(j_k,k)}(x|x_0)\frac{\Delta^k}{k!}. \tag{93}$$

Note that $D_\nu(x)$, which arises from the Jacobian transformation, in the reducible case, is independent of Δ, and can therefore be incorporated in $C_X^{(0)}$.

4.1.8 Determination of coefficients in the irreducible case

An explicit expansion of $C_X^{(j_k,k)}$, around $(x - x_0)$ will now be derived. Define a vector $\boldsymbol{i} \equiv (i_1, i_2, ..., i_m)$ of integers, and $I_k = \Big\{\boldsymbol{i} \equiv (i_1, i_2, ..., i_m) \in \mathbb{N}^m : 0 \leq tr(i) \leq j_k\Big\}$, such that

$$C_X^{(j_k,k)}(x|x_0) = \sum_{i \in I_k} \beta_i^{(k)}(x_0)(x_1 - x_{01})^{i_1}(x_2 - x_{02})^{i_2}...(x_m - x_{0m})^{i_m}. \tag{94}$$

The coefficients are then calculated recursively. From $C_X^{(j_{-1},-1)}$the following term, $C_X^{(j_0,0)}$ is derived. From $C_X^{(j_0,0)}$, $C_X^{(j_1,1)}$ is calculated explicitly etc. In order to state the final result, the following functions

are defined:

$$\begin{aligned} G_X^{(0)}(x|x_0) &= \tfrac{m}{2} - \textstyle\sum_{i=1}^m \mu_i(x)\frac{\partial}{\partial x_i}C_X^{(-1)}(x|x_0) + \sum_{i,j=1}^m \frac{\partial}{\partial x_i}\nu_{ij}(x)\frac{\partial}{\partial x_j}C_X^{(-1)}(x|x_0) \\ &+ \tfrac{1}{2}\textstyle\sum_{i,j=1}^m \nu_{ij}(x)\frac{\partial^2}{\partial x_i\partial x_j}C_X^{(-1)}(x|x_0) - \sum_{i,j=1}^m \nu_{ij}(x)\frac{\partial}{\partial x_i}C_X^{(-1)}(x|x_0)\frac{\partial}{\partial x_j}D_\nu(x), \end{aligned}$$

$$\begin{aligned} G_X^{(1)}(x|x_0) = \textstyle\sum_{i=1}^m \frac{\partial}{\partial x_i}\mu_i(x) + \frac{1}{2}\sum_{i,j=1}^m \frac{\partial^2}{\partial x_i\partial x_j}\nu_{ij}(x) - \sum_{i=1}^m \mu_i(x)\Big(\frac{\partial}{\partial x_i}C_X^{(0)}(x|x_0) - \frac{\partial}{\partial x_i}D_\nu(x)\Big) = \\ + \textstyle\sum_{i,j=1}^m \frac{\partial}{\partial x_i}\nu_{ij}(x)\Big(\Big(\frac{\partial}{\partial x_j}C_X^{(0)}(x|x_0) - \frac{\partial}{\partial x_j}D_\nu(x)\Big)\Big) \\ + \tfrac{1}{2}\textstyle\sum_{i,j=1}^m \nu_{ij}(x)\times \\ \Big\{\frac{\partial^2}{\partial x_i\partial x_j}C_X^{(0)}(x|x_0) - \frac{\partial^2}{\partial x_i\partial x_j}D_\nu(x) + \Big(\frac{\partial}{\partial x_i}C_X^{(0)}(x|x_0) - \frac{\partial}{\partial x_i}D_\nu(x)\Big)\Big(\frac{\partial}{\partial x_j}C_X^{(0)}(x|x_0) - \frac{\partial}{\partial x_j}D_\nu(x)\Big)\Big\}, \end{aligned}$$

and generally for $k \geq 2$:

$$\begin{aligned} G_X^{(k)}(x|x_0) - \textstyle\sum_{i=1}^m \mu_i(x)\frac{\partial}{\partial x_i}C_X^{(k-1)}(x|x_0) = \\ + \textstyle\sum_{i,j=1}^m \frac{\partial}{\partial x_i}\nu_{ij}(x)\frac{\partial}{\partial x_j}C_X^{(k-1)}(x|x_0) \\ + \tfrac{1}{2}\textstyle\sum_{i,j=1}^m \nu_{ij}(x)\frac{\partial^2}{\partial x_i\partial x_j}C_X^{(k-1)}(x|x_0) \\ + \tfrac{1}{2}\textstyle\sum_{i,j=1}^m \nu_{ij}(x)\times \\ \left\{\Big(\frac{\partial}{\partial x_i}C_X^{(0)}(x|x_0) - 2\frac{\partial}{\partial x_i}D_\nu(x)\Big)\frac{\partial}{\partial x_j}C_X^{(k-1)}(x|x_0) + \sum_{h=0}^{k-2}\begin{pmatrix} k-1 \\ h \end{pmatrix}\frac{\partial}{\partial x_i}C_X^{(h)}(x|x_0)\frac{\partial}{\partial x_j}C_X^{(k-1-h)}(x|x_0)\right\}. \end{aligned} \tag{95}$$

In order to determine the coefficients $C_X^{(j_k,k)}$, i.e. $\beta_i^{(k)}$ *for* $i \in I_k$, consider the following theorem.

Theorem 27. *The coefficient,* $C_X^{(k)}(x|x_0)$*, for* $k = -1, 0, ..., K$*, in equation*

$$l_X^{(K)}(x|x_0,\Delta) = -\tfrac{m}{2}\ln(2\pi\Delta) - D_\upsilon(x) + \tfrac{C_X^{(-1)}(x|x_0)}{\Delta} + \textstyle\sum_{k=0}^K C_X^{(k)}(x|x_0)\frac{\Delta^k}{k!},$$

solves

$$h_X^{(k-1)}(x|x_0) = 0,$$

s.t

$$h_X^{(-2)}(x|x_0) = -2C_X^{(-1)}(x|x_0) - \sum_{i,j=1}^m \nu_{ij}(x)\frac{\partial}{\partial x_i}C_X^{(-1)}(x|x_0)\frac{\partial}{\partial x_j}C_X^{(-1)}(x|x_0)$$

and

$$h_X^{(-1)}(x|x_0) = -\sum_{i,j=1}^m \nu_{ij}(x)\frac{\partial}{\partial x_i}C_X^{(-1)}(x|x_0)\frac{\partial}{\partial x_j}C_X^{(0)}(x|x_0) - G_X^{(0)}(x|x_0)$$

and for $k \geq 1$

$$h_X^{(k-1)}(x|x_0) = C_X^{(k)}(x|x_0) - \frac{1}{k}\sum_{i,j=1}^m \nu_{ij}(x)\frac{\partial}{\partial x_i}C_X^{(-1)}(x|x_0)\frac{\partial}{\partial x_j}C_X^{(k)}(x|x_0) - G_X^{(k)}(x|x_0),$$

where the coefficients $\beta_i^{(k)}$ *for* $i \in I_k$*, explicitly solves a system of linear equations.*

Applying the above theorem, the coefficients are determined recursively. That is $h_X^{(j+1)}(x|x_0) = 0$ yields $C_X^{(j)}$ which allows for $G_X^{(j+1)}$to be determined, for $j = 1, 2, ..., k$. That is $\beta_i^{(k)}$ for $i \in I_k$, is determined. Each of the equations are solved explicitly, by form of the expansion $C_X^{(j_k,k)}$ of $C_X^{(k)}$ around $(x - x_0)$ at order j_k, where $\beta_i^{(k)}(x_0)$ for $i \in I_k$ are determined by setting $h_X{}^{(j_k,k-1)}$ of $h_X^{(k-1)}$ equal to zero. A closed-form solution is obtained by solving a system of linear equations: for $tr[i] = 0$, $\beta_i^{(k)}$ is determined, then for $tr[i] = 1$, $\beta_i^{(k)}$ is determined, continued until $\beta_i^{(k)}$ is determined for $tr[i] = j_k$. Note that the polynomial has no linear or constant terms, i.e. $\beta_i^{(-1)} = 0$ for $tr[i] = 0$. For $tr[i] = 2$, with $j_{-1} \geq 2$:

$$\sum_{tr[i]=2; i \in I_{-1}} \beta_i^{(-1)}(x_0) \prod_{j=1}^{m} (x_j - x_{0j})^j = -\frac{1}{2}(x - x_0)^T \nu^{-1}(x_0)(x - x_0).$$

For $j_{-1} \geq 3$ only the terms $\beta_i^{(-1)}$ for $tr[i] = 3, 4, ..., j_{-1}$. Hence the solution $\beta_i^{(k)}$only depends on the dynamics of the diffusion matrix, $\nu(x)$.

Finally, in order to obtain an expansion for p_X,instead of l_X, the exponential of $\tilde{l}_X^{(K)}$can be determined, or the exponential in Δ can be expanded to obtain the coefficients c_X,for the expansion of the density p_X, from the coefficients C_X, for the expansion of the log-density l_X. To ensure the density approximations for l_X and p_X integrate to one, division by the integral over $\zeta_{\boldsymbol{X}}$ should be applied.

4.1.9 Application of the irreducible approach to reducible diffusions

Theorem 27 is more general than Theorem 26, in the sense that reducibility is not required. However, explicit coefficients are only available in the series expansion of x around x_0. In order to view the relationship between the two approaches, the following proposition will be considered.

Proposition 28. *Suppose a given diffusion process, X, is reducible; with the log-likelihood calculated by applying Theorem 26, denoted by $l_X^{(K)}$. Further, suppose the log-likelihood expansion, denoted by $\tilde{l}_X^{(K)}$, without the transformation of X to the unit diffusion of Y, i.e. by the direct application of Theorem 27. Every coefficient, $C_X^{(j_k,k)}(x \mid x_0)$ from the log-likelihood expansion $\tilde{l}_X^{(K)}$, is an expansion around $(x - x_0)$, at order j_k of the coefficient $C_X^{(k)}(x \mid x_0) = C_Y^{(k)}(\gamma(x) \mid \gamma(x_0))$, from the log-likelihood $l_X^{(K)}$.*

Therefore, by applying the irreducible approach to a reducible diffusion, the expression for $C_X^{(k)}(x \mid x_0)$ is replaced by its series around $(x - x_0)$. However, this is not needed if the diffusion is reducible and the transformation $\gamma : X \mapsto Y$ is explicit. When the diffusion is reducible, but the transformation $\gamma : X \mapsto Y$ is not explicit, Proposition 28 is applied. Finally, when considering a reducible diffusion, the double series in Δ and around $(x - x_0)$, is equivalent to the expansion produced by the Hermite series since its coefficients are determined as a series in Δ by the computation of the conditional expectation. For each order of Δ, the coefficients solve the Kolmogorov Equations. Hence the methods are equivalent.

4.1.10 Approximate maximum likelihood estimation (MLE) and convergence to the true log-likelihood

Let (μ, σ) be parameterized by parameter vector, $\boldsymbol{\theta}$. Suppose that (μ, σ), with its derivatives are continuously differentiable over $\boldsymbol{\theta}$. The differentiability of the coefficients also applies to the log-likelihood l_X. Define the parameter space as $\boldsymbol{\Theta} \subseteq \mathbb{R}^r$, and true parameter value $\boldsymbol{\theta}_0$. Assume, for fixed $n \in \mathbb{N}$ and Δ. A unique maximum likelihood estimator (MLE), $\hat{\boldsymbol{\theta}}_{n,\Delta} \in \boldsymbol{\Theta}.\hat{\boldsymbol{\theta}}_{n,\Delta}$, exists for $\boldsymbol{\theta} \mapsto l_n(\boldsymbol{\theta}, \Delta)$. Define approximate MLE, $\hat{\boldsymbol{\theta}}^{(K)}_{n,\Delta}$, which is obtained by maximizing $l^{(K)}_n(\boldsymbol{\theta}, \Delta)$ (or $\tilde{l}^{(K)}_n(\boldsymbol{\theta}, \Delta)$ in the irreducible case), with expansion $l^{(K)}_X$ (or $\tilde{l}^{(K)}_X$ in the irreducible case), instead of l_X (true log-likelihood transition density function). Therefore, the following theorem is stated:

Theorem 29. *For any* $n \in \mathbb{N}$

$$\limsup_{\boldsymbol{\theta}\in\boldsymbol{\Theta},\Delta\to 0} \mid \tilde{l}^{(K)}_n(\boldsymbol{\theta}, \Delta) - l_n(\boldsymbol{\theta}, \Delta) \mid =_{in\ probability} 0. \tag{96}$$

The same holds for $l^{(K)}_n$ *in the reducible case. The approximate Maximum Likelihood Estimate sequence exists, i.e* $\hat{\boldsymbol{\theta}}^{(K)}_{n,\Delta}$ *and satisfies*

$$\lim_{\theta\in\Theta,\Delta\to 0}\left(\hat{\boldsymbol{\theta}}^{(K)}_{n,\Delta} - \hat{\boldsymbol{\theta}}_{n,\Delta}\right) =_{in\ probability} 0. \tag{97}$$

Further,

$$\lim_{n\to\infty} \hat{\boldsymbol{\theta}}^{(K)}_{n,\Delta} =_{in\ probability} \hat{\boldsymbol{\theta}}_{n,\Delta}.$$

There exist a sequence of matrices, $\mid S_{n,\Delta} : r \times r \mid > 0$, *s.t.*

$$S^{-1}_{n,\Delta}\left(\hat{\boldsymbol{\theta}}_{n,\Delta} - \boldsymbol{\theta}_0\right) = O_p(1). \tag{98}$$

There then exists a sequence Δ_n, *where* $\lim_{n\to\infty} \Delta_n = 0$, *s.t.*

$$S^{-1}_{n,\Delta_n}\left(\hat{\boldsymbol{\theta}}^{(K)}_{n,\Delta_n} - \hat{\boldsymbol{\theta}}_{n,\Delta_n}\right) = O_p(1). \tag{99}$$

Theorem 29 indicates that the approximation error is small, when sufficiently close to $\Delta = 0$ (reducible case) or $x = x_0$ (irreducible case), due to the Taylor expansion of the log-likelihood around $\Delta = 0$ or $x = x_0$. Lastly, $\hat{\boldsymbol{\theta}}^{(K)}_{n,\Delta_n}$ and $\hat{\boldsymbol{\theta}}_{n,\Delta_n}$ has the same asymptotic distribution.

4.2 Cumulant truncation transition density approximation method

The Cumulant (or Moment) Truncation transition density approximation method, based on the works of [17, 21], is a consistent alternative for deriving a a robust closed-form density approximation to a general multivariate diffusion process. The procedures relies in the evaluating moment trajectories of the model process in consideration, which in turn is used in a saddlepoint density (a surrogate density) function, to get a final approximation. The moment equation of the process is derived by the evaluation of the moment generating function (MGF)

Theorem 30. *Partial differential equation (PDE) for the MGF of a multivariate diffusion process.*

Denote the MGF by

$$M(\theta, t) = \sum_{i=0}^{\infty} \frac{\theta^i E[X_t^i]}{i!},$$

it then follows that the MGF of a multivariate diffusion process is governed by the PDE:

$$\frac{\partial}{\partial t} M(\theta, t) = \theta\mu\left(\frac{\partial}{\partial \theta}, t\right) M(\theta, t) + \theta^2\mu^2\left(\frac{\partial}{\partial \theta}, t\right) M(\theta, t).$$

Consider the multivariate diffusion process in Equation 13. Consider $\boldsymbol{\theta_m} = (\theta_1, \theta_2, ..., \theta_m)$, the parameter vector of the process, with MGF

$$M(\boldsymbol{\theta}) = E\left[e^{\theta_1 X_1 + \theta_2 X_2 + ... + \theta_m X_m}\right].$$

and cumulant generating function (CGF)

$$\begin{aligned} K(\boldsymbol{\theta}) &= ln\left(M(\boldsymbol{\theta})\right) \\ &= ln\left(\mathbb{E}(e^{\theta_1 X_1 + \theta_2 X_2 + ... + \theta_m X_m})\right). \end{aligned}$$

The moment truncation approximation method can therefore also be referred to as the cumulant truncation method. The transformation from moment to cumulant generating function is given by

$$M(\theta) = 1 + \sum_{n=1}^{\infty} \frac{\mu_n' \theta^n}{n!} = e^{\sum_{n=1}^{\infty} \frac{\kappa_n' \theta^n}{n!}} = e^{K(\theta)},$$

where μ_n' and κ_n'denotes the n^{th} central moment and cumulant respectively.

Assuming the existence of the MGF on its domain, upon evaluation, using the saddlepoint approximation, as surrogate density, an approximate closed-form density can be obtained as:

$$f(\boldsymbol{x}) = (2\pi)^{-\frac{m}{2}} \left|\nabla^2 K(\boldsymbol{\theta})\right|^{-\frac{1}{2}} e^{(K(\boldsymbol{\theta}) - \boldsymbol{\theta}^T \boldsymbol{x})}, \tag{100}$$

where $\nabla^2 K(\boldsymbol{\theta}) : m \times m$ denotes the Hessian Matrix for $K(\boldsymbol{\theta})$ and $\nabla K(\boldsymbol{\theta}) = \boldsymbol{x}$.

For the instances the CGF is unknown, it may be approximated by

$$K(\boldsymbol{\theta}) = E[e^{\boldsymbol{\tau X}}] \approx \sum_{i=1}^{n \leq m} \frac{\tau^n}{n!} k_n,$$

where $(k_1, k_2, ..., k_n)$ denotes the first $n \leq m$ cumulants of the process.[17]

Due to the fact that a diffusion process inherits the Markov property, the likelihood of the diffusion process, at N discrete time epochs, is given by:

$$L(\boldsymbol{\theta}) = \boldsymbol{f}(x_{t_1}) \prod_{i=2}^{N} \boldsymbol{f}(\boldsymbol{x_{t_i}} \mid \boldsymbol{x_{t_{i-1}}}), \tag{101}$$

where $f(.)$ is the saddlepoint approximation to the true density.

Example 31. Bivariate Ornstein Uhlenbeck Process applied to the South African Reserve Bank's Monetary Policy

Define the bivariate OU model as:

$$dX_t = (\alpha_1(\beta_1 - X_t) - \lambda_1 Y_t)dt + \sigma_1 dW_t^{(1)}$$
$$dY_t = (\alpha_2(\beta_2 - Y_t) - \lambda_2 X_t)dt + \sigma_2 dW_t^{(2)},$$

or equivalently:

$$d\boldsymbol{Z}_t = \begin{pmatrix} dX_t \\ dY_t \end{pmatrix} = \begin{pmatrix} (\alpha_1(\beta_1 - X_t) - \lambda_1 Y_t) \\ (\alpha_2(\beta_2 - Y_t) - \lambda_2 X_t) \end{pmatrix} dt + \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} dW_t^{(1)} \\ dW_t^{(2)} \end{pmatrix},$$

s.t. $t \in [s, T]$, with $s \geqslant 0$ and $X_t \in [X_s, X_T]$, $Y_t \in [Y_s, Y_T]$, and with $dW_t^{(i)} : i = 1, 2$ the Brownian Motions, as in Definition 6 The parameter space $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\sigma})$ consists of deterministic parameters $\{\alpha_{i=1,2}, \beta_{i=1.2}, \lambda_{ij}, \sigma_{i=12}\}$.Note $\sigma_{12} = \sigma_{21} = 0$ implying that the volatility sources $(dW_t^{(i)} : i = 1, 2)$ are independent . In terms of Equation 13, the drift and diffusion coefficients are given by

$$\boldsymbol{\mu}(\boldsymbol{Z_t}, t; \boldsymbol{\Theta}) = \begin{pmatrix} (\alpha_1(\beta_1 - X_t) - \lambda_1 Y_t) \\ (\alpha_2(\beta_2 - Y_t) - \lambda_2 X_t) \end{pmatrix}$$

and

$$\boldsymbol{\Sigma}(\boldsymbol{Z_t}, t; \boldsymbol{\Theta}) = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix},$$

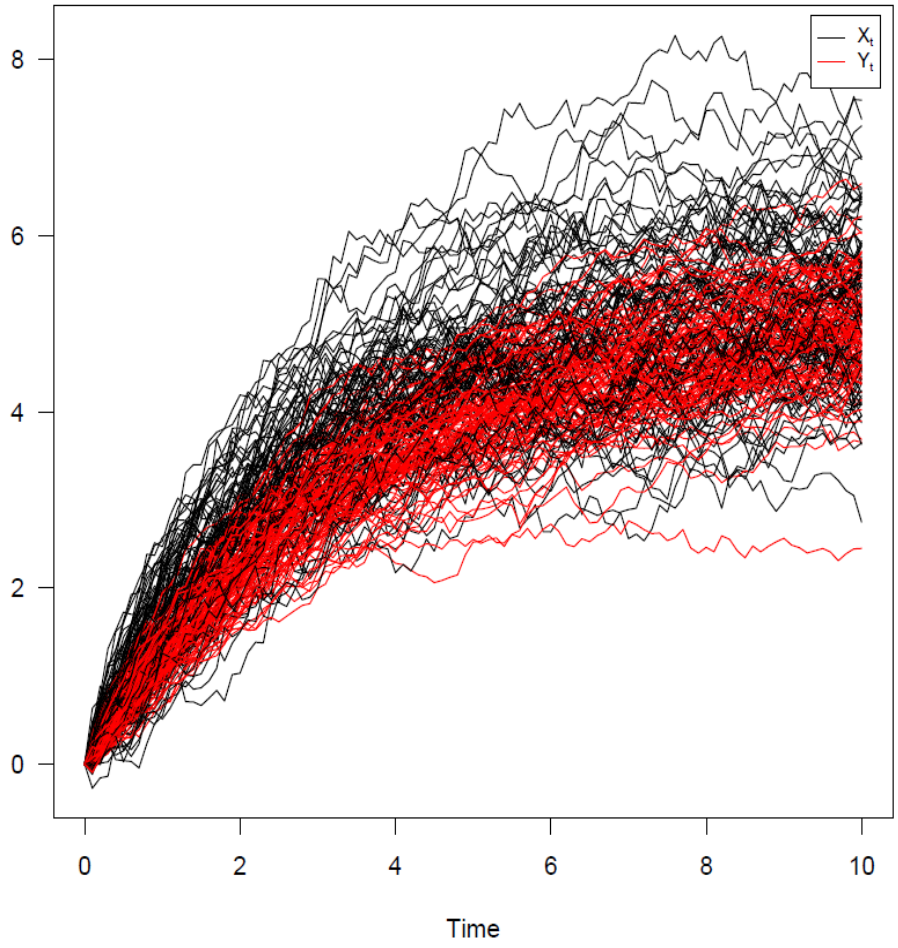


Figure 19: 100 simulated trajectories from the bivariate OU Process

respectively. Note that $\boldsymbol{Z}_t$ is defined as

$$\boldsymbol{Z}_t = \begin{pmatrix} X_t \\ Y_t \end{pmatrix}.$$

Performing the simulation study on $t \in [0, 100]$,$X_t \in [0, 10]$, $Y_t \in [0, 10]$; $\boldsymbol{\theta} = (\alpha_{i=1,2}, \beta_{i=1.2}, \lambda_{i=1.2}, \sigma_{i=12}) = (4.5, 6, 0.2, 0.15, 0.09, 0.25, 0.5, 0.25)$, with $X_0 = 3.8$ and $Y_0 = 6.5$, and $stepsize = 1/30$. Note

$$\boldsymbol{Z}_t = \begin{pmatrix} X_t = CPI \\ Y_t = Repo\ Rate \end{pmatrix}.$$

Figure19 illustrates displays 100 simulated trajectories from the bivariate OU Process. Please see Algorithm 11, as perfromed via R package Sim.DiffProc, [6].

Based on the simulations an Euler-Maruyama distributions is fitted to the process, which can be viewed in the contour plot, perspective plot, and marginal kernel density estimates in Figures 20, 21, 22. Fitting a bivariate Hermite Approximation, $k = 1$, at $t = 10$ yields the Hermite approximate densities,

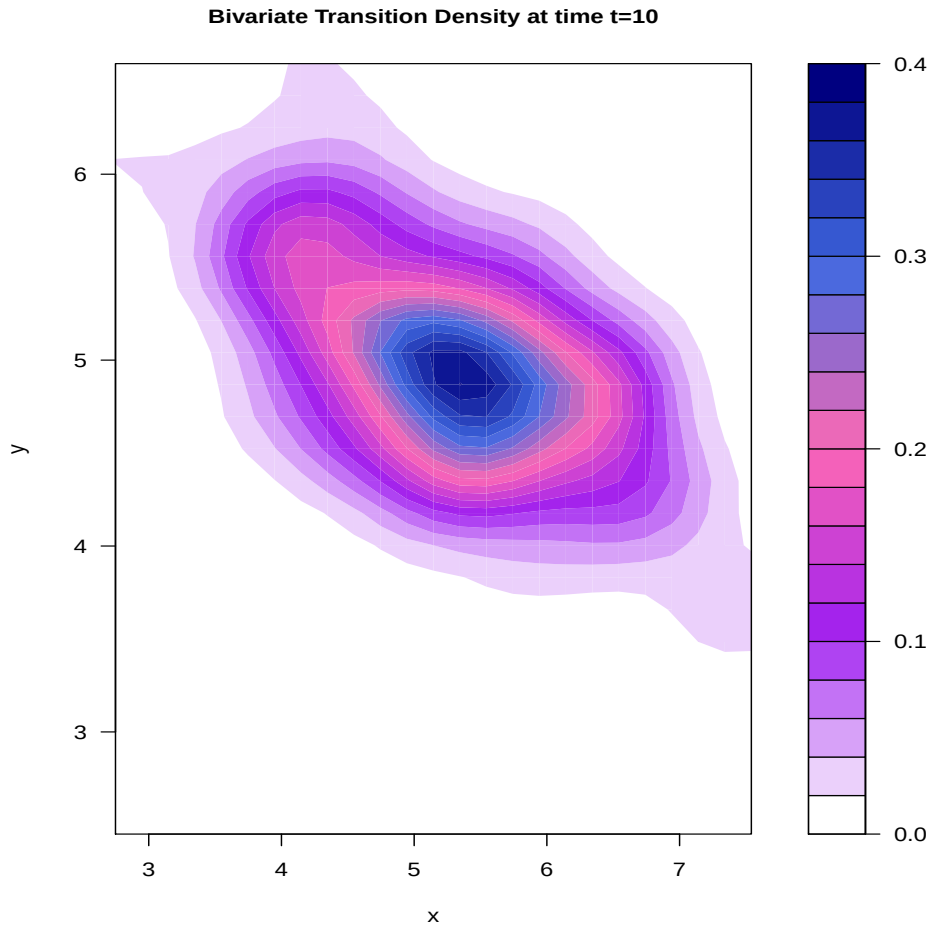


Figure 20: Euler approximate bivariate OU Process viewed at $t = 10$ contour plot

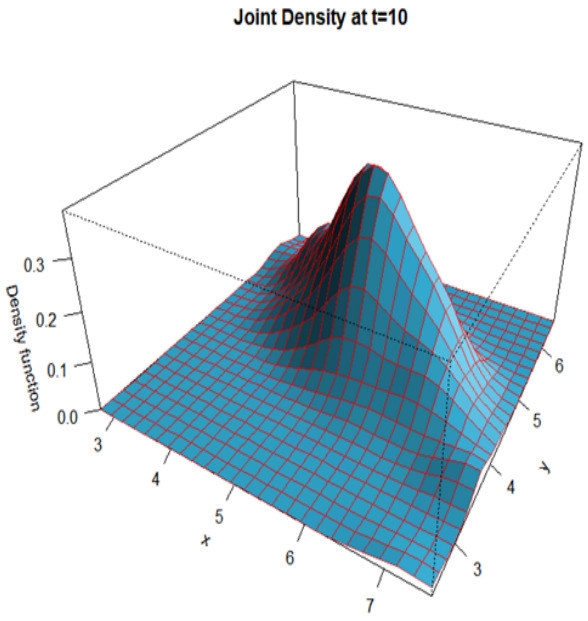


Figure 21: Euler approximate bivariate OU Process viewed at $t = 10$ perspective plot

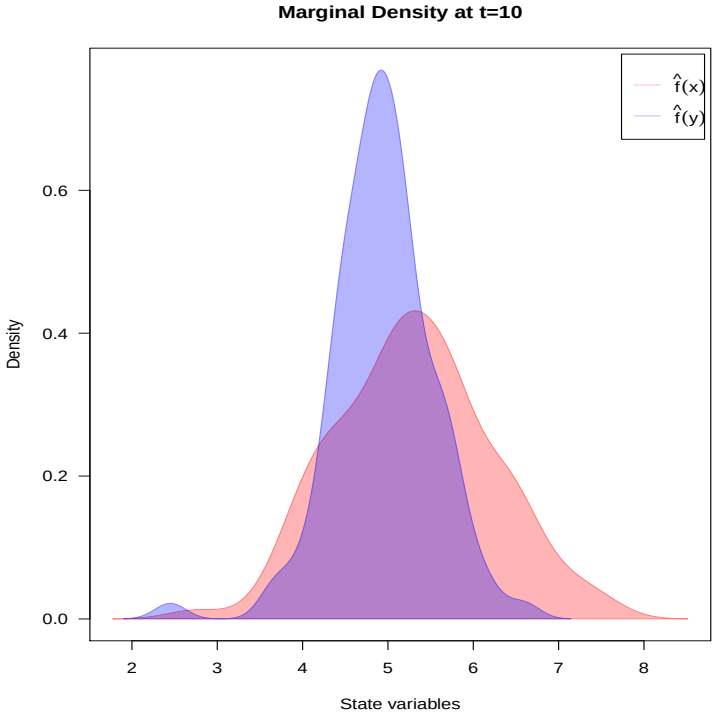


Figure 22: Euler approximate bivariate OU Process viewed at $t = 10$ kernel-approximated/fitted marginal densities

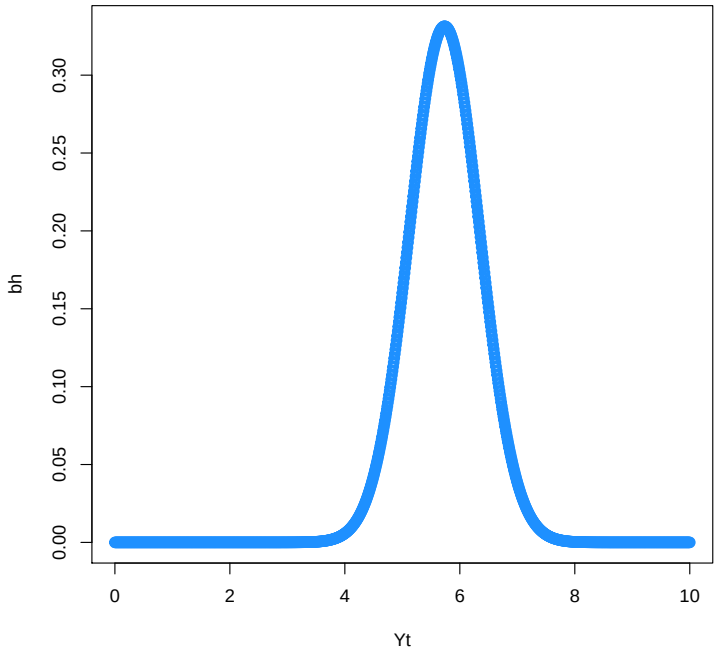


Figure 23: bivariate Hermite Approximation, $k = 1$, at $t = 10$

as shown in Figure 23 (X_t) and Figure24 (Y_t).

In order to obtain a Cumulant truncated approximate density, truncated at 2, the moments and cumulants are calculated, as shown in Figure25, and plugged into the saddlepoint surrogate density.

The cumulant truncated (at 2 moments) approximated transition density for the bivairate OU model, plotted as a contour plot at $t = 10$, through the surrogate saddelpoint density, which can be seen in Figure 26.

4.2.1 Inference on a bivariate diffusion process

Example 32. Bivariate Cox Ingersoll and Ross (CIR) Diffusion Process applied to the VIX index and USDZAR Exhange rate

Consider the Chicago Board Volatilty Index/ VIX index as well as the daily USDZAR exhange rate values, drawn from Bloomberg.

In an attemt to explain the data and relationship, define the bivariate CIR model as, [16]:

$$\boldsymbol{Z}_t = \begin{pmatrix} X_t = VIX \\ Y_t = USDZAR \end{pmatrix},$$

where

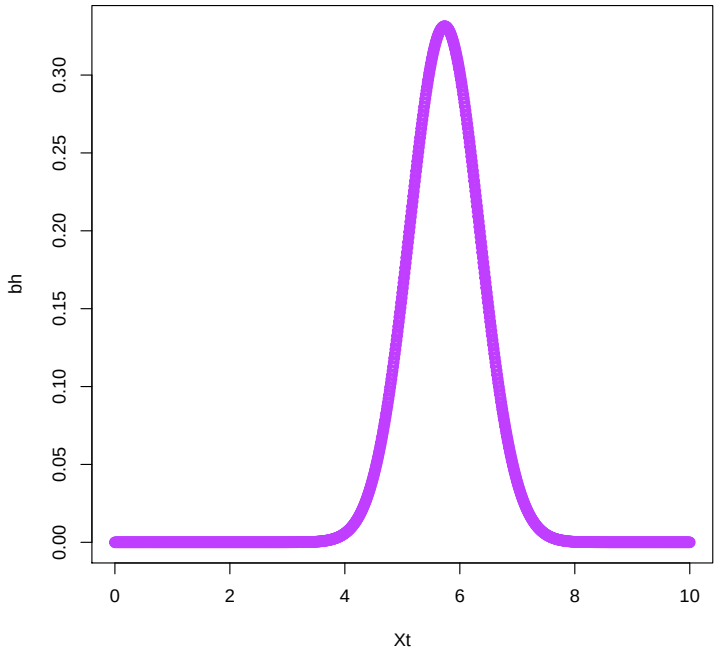


Figure 24: bivariate Hermite Approximation, $k = 1$, at $t = 10$

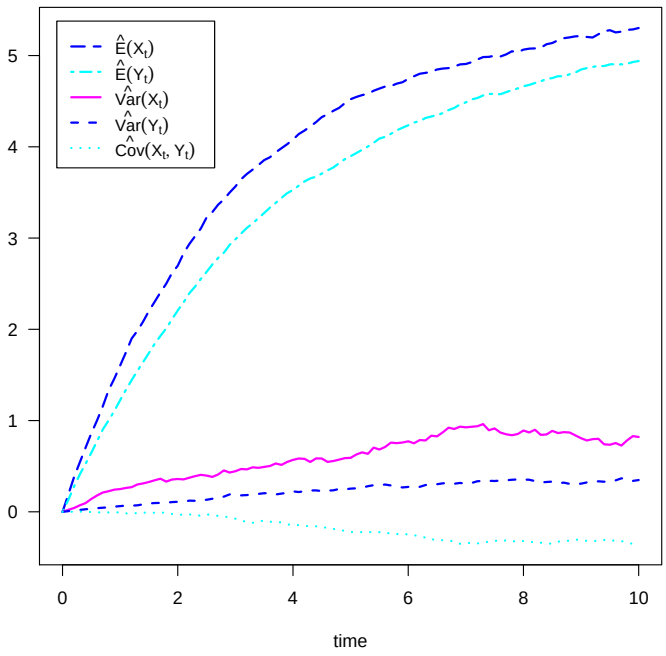


Figure 25: bivariate OU moment/cumulant equations as part of the cumulant truncation approximation method (truncated at 2)

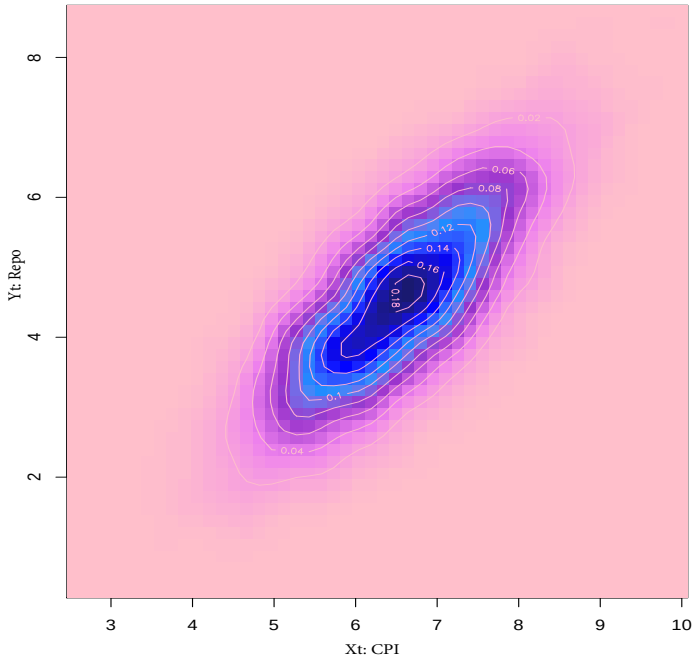


Figure 26: contour plot showing the cumulant truncated (at 2 moments) approxiamted transition density for the bivairate OU model.

$$dX_t = (\alpha_1(\beta_1 - X_t) - \lambda_1 Y_t)dt + \sigma_1\sqrt{X_t}dW_t^{(1)}$$
$$dY_t = (\alpha_2(\beta_2 - Y_t) - \lambda_2 X_t)dt + \sigma_2\sqrt{Y_t}dW_t^{(2)},$$

or equivalently:

$$d\boldsymbol{Z}_t = \begin{pmatrix} dX_t \\ dY_t \end{pmatrix} = \begin{pmatrix} (\alpha_1(\beta_1 - X_t) - \lambda_1 Y_t) \\ (\alpha_2(\beta_2 - Y_t) - \lambda_2 X_t) \end{pmatrix} dt + \begin{pmatrix} \sigma_1\sqrt{X_t} & 0 \\ 0 & \sigma_2\sqrt{Y_t} \end{pmatrix} \begin{pmatrix} dW_t^{(1)} \\ dW_t^{(2)} \end{pmatrix},$$

s.t. $t \in [s, T]$, with $s \geqslant 0$ and $X_t \in [X_s, X_T]$, $Y_t \in [Y_s, Y_T]$ and with $s \geqslant 0$, and $X_t, Y_t \geqslant 0$ for all t (due to $\sqrt{X_t}, \sqrt{Y_t}$ being in the Real space), and with $W : i = 1, 2$ the Brownian Motions, as in Definition 6 The parameter space $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\sigma})$ consists of deterministic parameters $\{\alpha_{i=1,2}, \beta_{i=1.2}, \lambda_{ij}, \sigma_{i=12}\}$.Note $\sigma_{12} = \sigma_{21} = 0$ implying that the volatility sources $(dW_t^{(i)} : i = 1, 2)$ are independent . In terms of Equation 13, the drift and diffusion coefficients are given by

$$\boldsymbol{\mu}(\boldsymbol{Z_t}, t; \boldsymbol{\Theta}) = \begin{pmatrix} (\alpha_1(\beta_1 - X_t) - \lambda_1 Y_t) \\ (\alpha_2(\beta_2 - Y_t) - \lambda_2 X_t) \end{pmatrix}$$

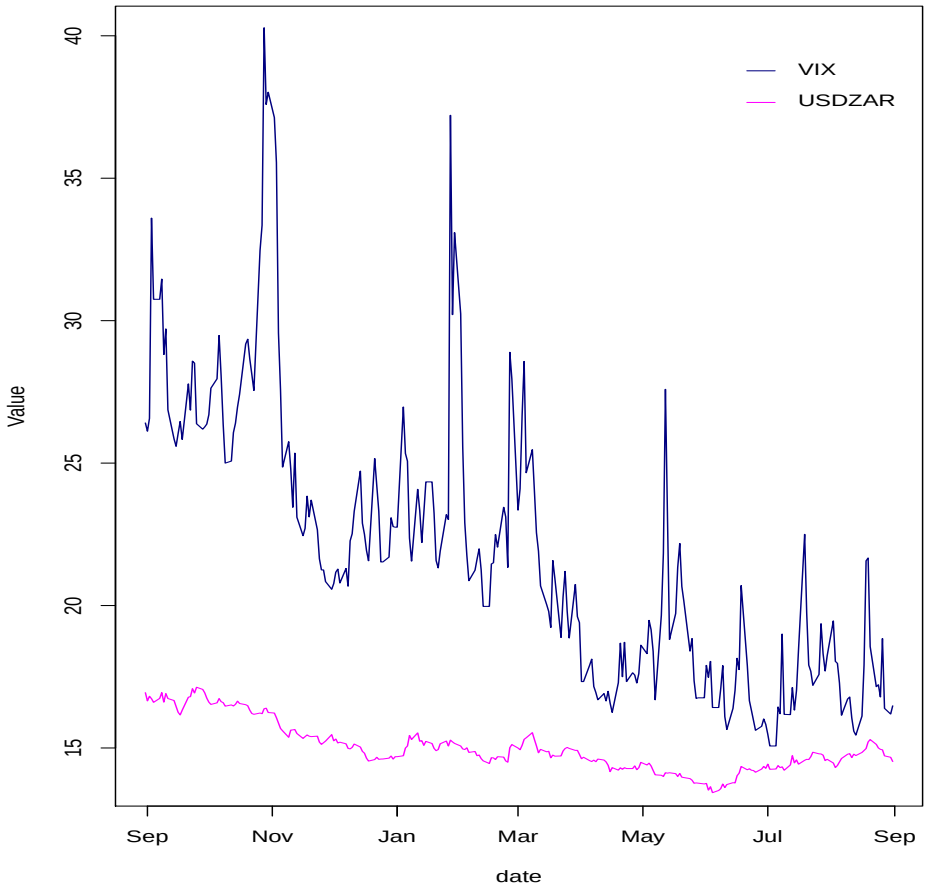


Figure 27: theoretical and empirical evolution of the cumulants of a univariate CIR Process.

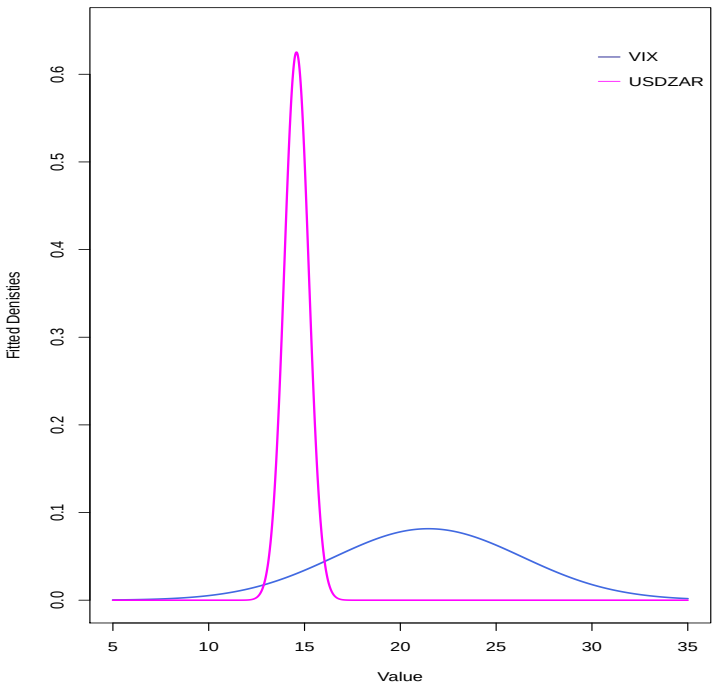


Figure 28: marginal denisties fitted, by use of the MLE estimates, based on a bivariate CIR process.

and

$$\boldsymbol{\Sigma}(\boldsymbol{Z_t},t;\boldsymbol{\Theta}) = \begin{pmatrix} \sigma_1\sqrt{X_t} & 0 \\ 0 & \sigma_2\sqrt{Y_t} \end{pmatrix},$$

respectively. Based on the Saddlepoint approximation or Cumulant Truncation Approximation technique MLE performed on 1 year's volatolity and Rand/Dollar values. $\hat{\boldsymbol{\theta}}^{mle}_{Saddle}$ coverged to the maximum likelihood estimators $\hat{\boldsymbol{\theta}}^{mle}_{Saddle} = (\hat{\kappa_1}, \hat{\alpha_2}, \hat{\sigma_3}, \hat{\kappa_4}, \hat{\alpha_5}, \hat{\sigma_6}) = (22.27, 21.46, 32.65, 5.89, 14.58, 2.19)$. The MLE procedure was initiated at $(50, 50, 50, 16, 15, 5)$. Based on the MLE values the Marginal densities has been plotted, as can be seen in Figure 28.

4.2.2 Inference on bivariate financial data

Example 33. Bivariate Heston model, eith inference on the Chicago Board Options Exchange Volatility Index (VIX Index) and and teh S&P 500 Index.

The VIX Index is a volatility benchmark based on market estimates of th expected volatility of the S&P500 Index (consisting of 500 leading US entities). Bothe indices are obtained from Bloomberg, and plotted as seen in Figure29.

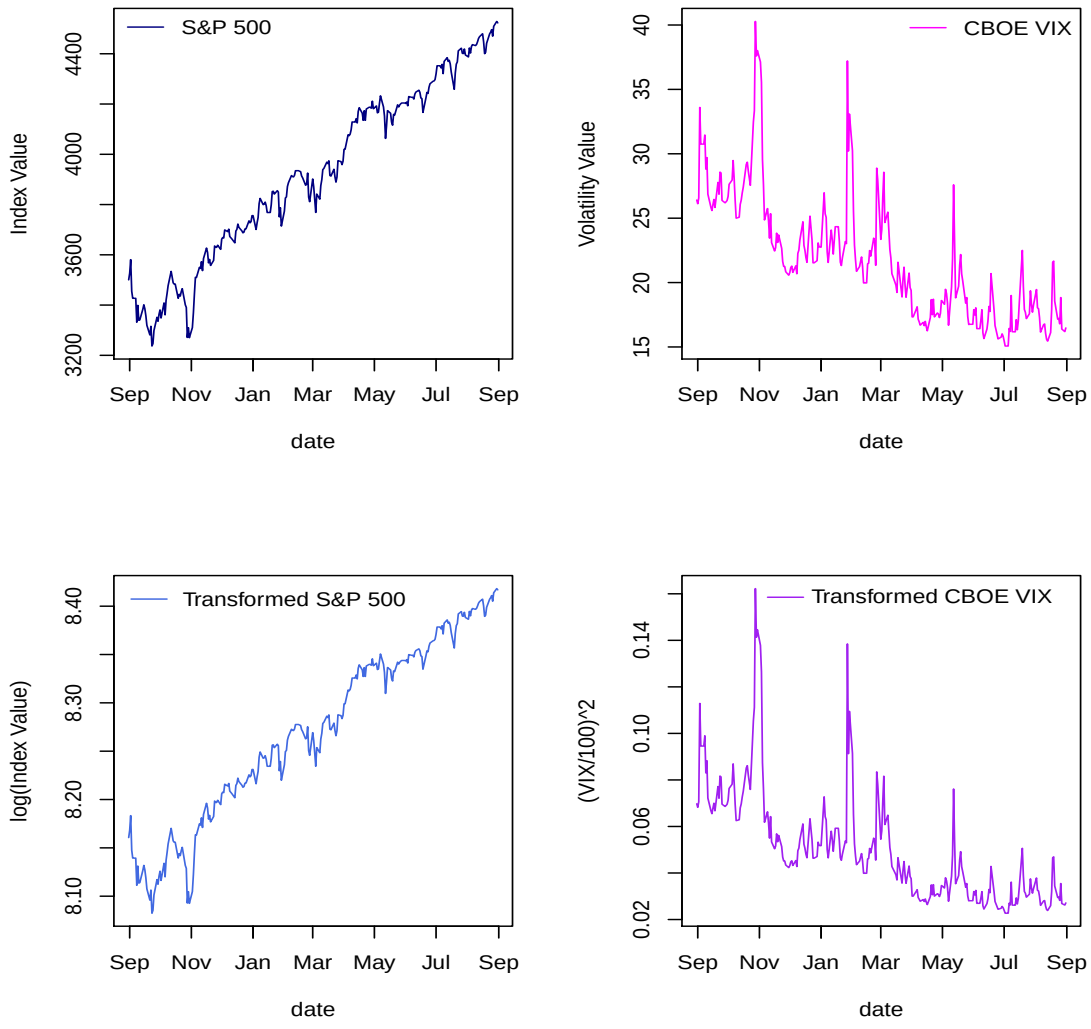


Figure 29: observed and transformed S&P500 and CBOE VIX time-series.

Therefore, as defined in [18] the bivariate Heston model as

$$d\boldsymbol{Z}_t = \begin{pmatrix} dX_t \equiv dS\&P\ 500 \\ dY_t \equiv dCBOE\ VIX \end{pmatrix},$$

where X_t denotes the S&P 500 spot price, and $Y_t$denotes the CBOE Vix volatility value.

$$dX_t = \alpha_1 X_t dt + \alpha_2 X_t \sqrt{Y_t} d\omega_t^{(1)}$$
$$dY_t = (\alpha_3 - \alpha_4 Y_t)dt + \alpha_6 d\omega_t^{(2)},$$

A geometric Brownian motion is used to model the S&P 500 value, where its volatility component is driven by a CIR process. Define the correlation between the Brownian Motion for the two process as:

$$corr(\omega_t^{(1)}, \omega_t^{(2)}) = \alpha_6.$$

Since we are interested in the log of the asset price, i.e $R_t = \log(X_t)$, by means of Ito's lemma we find the Heston Model under log-transform (also seen in Figure 29):

$$dX_t = (\alpha_1 - 0.5\alpha_2^2 Y_t)dt + \alpha_2 \sqrt{Y_t} d\omega_t^{(1)}$$
$$dY_t = (\alpha_3 - \alpha_4 Y_t)dt + \alpha_6 d\omega_t^{(2)},$$

and in order to incorporate the correlated brownian motions, $\omega_t^{(1)}, \omega_t^{(2)}$, we need to write the process in terms of independent Brownian motions, i.e $W_t^{(1)}, W_t^{(2)}$, defined as

$$\begin{bmatrix} d\omega_t^{(1)} \\ d\omega_t^{(2)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \alpha_6 & \sqrt{1-\alpha_6^2} \end{bmatrix} \begin{bmatrix} dW_t^{(1)} \\ dW_t^{(2)} \end{bmatrix},$$

with diffusion tensor calculated as

$$\begin{bmatrix} \alpha_2\sqrt{Y_t} & 0 \\ 0 & \alpha_5\sqrt{Y_t} \end{bmatrix} \begin{bmatrix} 1 & \alpha_6 \\ \alpha_6 & 1 \end{bmatrix} \begin{bmatrix} \alpha_2\sqrt{Y_t} & 0 \\ 0 & \alpha_5\sqrt{Y_t} \end{bmatrix}^T = \begin{bmatrix} \alpha_2^2 Y_t & \alpha_2\alpha_5\alpha_6 Y_t \\ \alpha_2\alpha_5\alpha_6 Y_t & \alpha_5^2 Y_t \end{bmatrix}.$$

Performing maximum likelihood estimation, by use of R package DiffusionRgqd [17] , yields fitted paarameters

$$\tilde{\boldsymbol{\theta}} = (\tilde{\alpha}_i)_{i=1,2,3,4,5,6} = (0.143, 0.673, 0.567, 8.154, 0.725, -0.754).$$

initiated at $(8, 1, 0.05, 0.5, 1, 0)$ with $AIC = -3689.995$. $\tilde{\alpha}_6 = -0.752$ implies the strong negative corre-

lation between the VIX and S&P 500. This makes sense over the period the data was gathered, at the economic correction phaseamidst the Covid-19 pandemic; as volatilities subsided returns strenghtened. Please see code in Algorithm 14.

5 Fixed income market simulation study

As the order of the difssion model increase, the mathematical and numerical computations becomes exponentially larger and more complicated, and difficult to display in 2d or 3d space. Nonetheless, through the simulation studies, such as simulation of trajectories in this paper can be efficiently en successfully applied. The parameter values can be estimated through a variety of techniques, e.g maximum likelihood estimation through linear regression, and a correlation analysis.. An excellent use case is the simulation or construction of the yield curve in bond (fixed income) markets. A yield curve is constructed out of numerous tenors, e.g $\{1day, 10days, 1month, 3months, 1year, 2years, 5years, 10years, 30years\}$with each tenor representing the market yield, however each tenor is dependent on the other. Therefore I attempt to replicate a yield curve, including jumps, by studying the dynamics of each tenor to get an idea of the parameters to use to define the diffusion model. Each tenor's yield (as a forward rate in this case, however spot rates could also be used), is modeled as a dependent factor, being dependent on the other tenors as underlying processes. The model is composed of simulating the yield-trajectories forward for each tenor and the combining them to form a predilection for a specified day. The yield curve for a specified date is compared to the fitted curve, and visually it seems to be a good fit. The accuracy in jump detection should be noted. Through more accurate and robust estimation of parameters and enhanced simulations the model can provide an even more accurate fit. Accuracy of fit can be seen in Figure 30. A multivariate CIR diffusion process, with jumps, are simulated. See Algorithm 30, for the R code used in generating the plots in Figure 30.

The model is represented as

$$dX_t^i = (\alpha_i(\beta_i - X_t^i) - \sum_{\forall i \neq j} \lambda_j Y_t^j)dt + \sigma_i \sqrt{X_t^i} dW_t^i + dP_t^i, \tag{102}$$

$$\forall\, i\, tenors,$$

$$dt \rightarrow change\, in\, time,$$

$$W_t^i \rightarrow Brownian\, Motion,$$

$$P_t^i \rightarrow Poisson\, Process\, (Jump),$$

$$\alpha_i \rightarrow Reversion\, speed,$$

$$\beta_i \rightarrow Reverting\, mean,$$

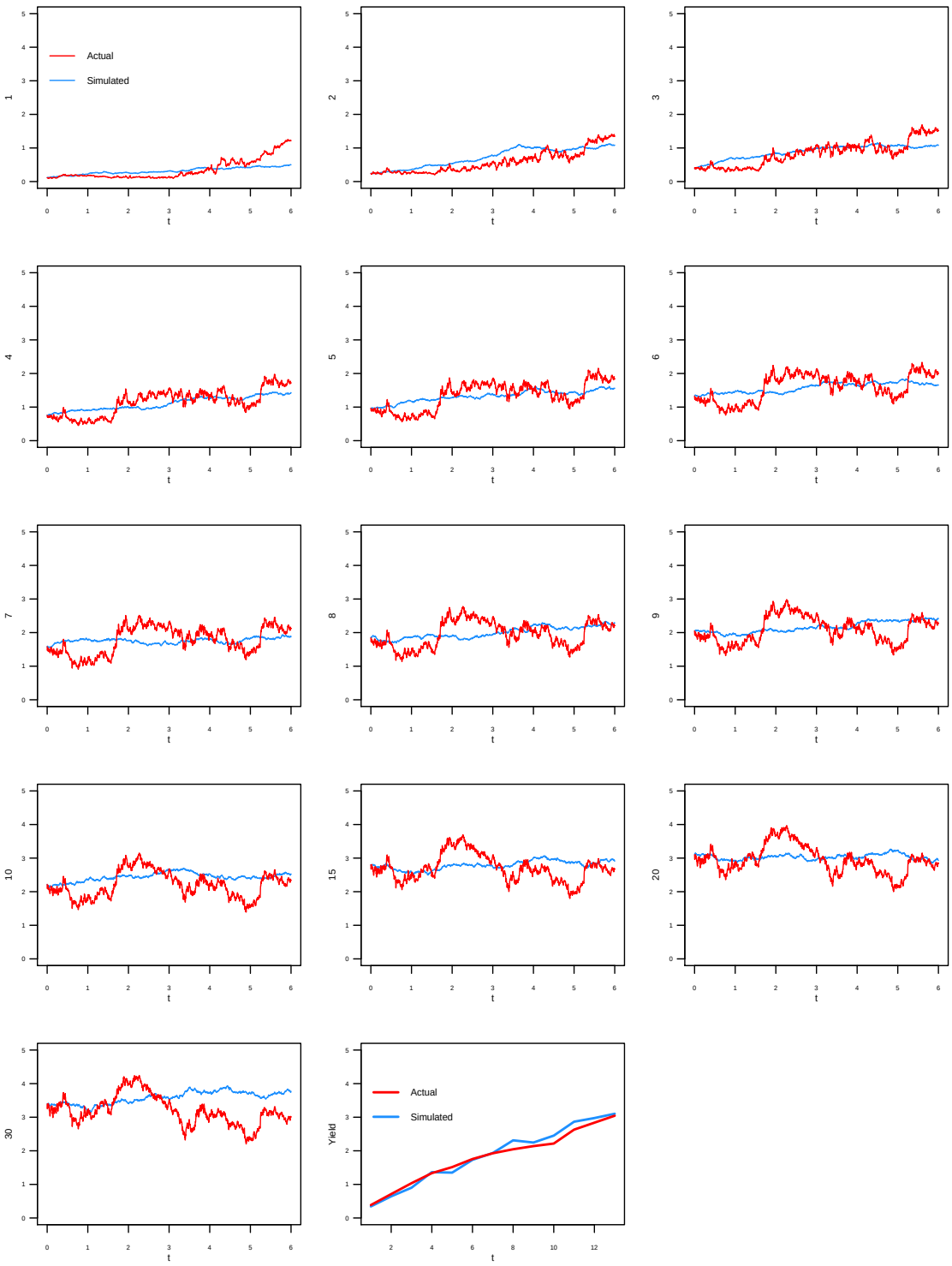
$$\lambda_i \rightarrow Dependence\ factor,$$

and

$$\sigma_i \rightarrow Volatility\ factor.$$

Each tenor's yield (as a forward rate in this case, however spot rates could also be used), is modeled as a dependent factor, being dependent on the other tenors as underlying processes. The correct parameter values can be estimated through a variety of techniques, e.g maximum likelihood, estimation through linear or non-parametric regression, and even a correlation analysis. The model is composed of simulating the yield-trajectories forward for each tenor and the combining them to form a predilection for a specified day. The fitted/simulated, with the actual yields, are plotted, then combined to construct a full yield curve. See Figure 30.

Figure 30: Multivariate CIR Fitted Forward Rate Yield Curve



The yield curve for a specified date is compared to the fitted curve, and visually it seems to be a good fit. The accuracy in jump detection should be noted.

6 Conclusion

In this paper, it was shown that understanding the dynamics of diffusion models allows for the explanation of various financial and economic phenomena. Building on that, the inferential strength attainable from obtaining a closed-form transition density was illustrated. As discussed, a true transition density in closed-form seldom exists, therefore the need to develop an effective closed-form approximation technique, proved fundamental. Firstly, simulating a distribution through the Euler-Maruyama scheme, provided a good visual view of the underlying density, however the method lacks in analytical and inferential power. The Hermite approximation technique,[1], proved effective and accurate under certain conditions, however lacked accuracy when the time domain was enlarged. This method also required substantial additional complexity as the order of approximation was increased. Lastly, the Cumulant Truncation approximation technique,[17], has proved to be the most accurate and robust under various conditions. By means of the Saddlepoint approximation, with cumulants, as inputs, valuable inference on financial time series was conducted.

References

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

[2] Yacine Aït-Sahalia. Maximum likelihood estimation of discretely sampled siffusions: a closed-form approximation approach. *Econometrica*, 70(1):223–262, 2002.

[3] YacineAit-Sahalia Closed-formlikelihoodexpansions for multivariate diffusions. *The Annals of Statistics* 36(1):906–937,2008.

[4] YacineAit-Sahalia Closed-formlikelihoodexpansions for multivariate diffusions. *The Annals of Statistics*, 36(2):906–937, 2008.

[5] Torben G Andersen and Jesper Lund. Stochastic volatility and mean drift in the short rate diffusion: sources of steepness, level and curvature in the yield curve. Technical report, Citeseer, 1997.

[6] Kamal Boukhetala and Arsalane Guidoum. Sim. diffproc: A package for simulation of diffusion processes in r. 2011.

[7] John C Cox, Jonathan E Ingersoll Jr, and Stephen A Ross. A theory of the term structure of interest rates. *Econometrica: Journal of the Econometric Society*, pages 385–407, 1985.

[8] Rick Durrett. *Probability: Theory and Examples*. Cambridge University Press, 2010.

[9] Patrick Flandrin, Pierre Borgnat, and Pierre-Olivier Amblard. From stationarity to self-similarity, and back: variations on the lamperti transformation. *Processes with Long-Range Correlations*, pages 88–117, 2003.

[10] Constantino Goutis and George Casella. Explaining the saddlepoint approximation. *The American Statistician*, 53(3):216–224, 1999.

[11] Desmond J Higham. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM Review*, 43(3):525–546, 2001.

[12] George J Jiang and John L Knight. A nonparametric approach to the estimation of diffusion processes, with an application to a short-term interest rate model. *Econometric Theory*, 13(5):615–645, 1997.

[13] Charles R Nelson and Andrew F Siegel. Parsimonious modeling of yield curves. *Journal of business*, pages 473–489, 1987.

[14] Giuseppe Orlando, Rosa Maria Mininni, and Michele Bufalo. Forecasting interest rates trough vasicek and cir models: a partitioning approach. *arXiv preprint arXiv:1901.02246*, 2019.

[15] Grigorios A Pavliotis. *Stochastic Processes and Applications*. Springer, 2016.

[16] Etienne A.D. Pienaar. Bivariate jump diffusions, February 2016.

[17] Etienne AD Pienaar. *Non-linear Diffusion Processes and Applications*. PhD thesis, University of Cape Town, 2016.

[18] Etienne AD Pienaar and Melvin M Varughese. Diffusionrgqd: an r package for performing inference and analysis on time-inhomogeneous quadratic diffusion processes, 2016.

[19] Steven E Shreve. *Stochastic Calculus for Finance II: Continuous-time Models*, volume 11. Springer Science & Business Media, 2004.

[20] Johan Swart. Arbitrage theory- the partial differential equations approach. Unpublished manuscript, University of Pretoria, 2016.

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

[22] George Neville Watson. *A Treatise on the Theory of Bessel Functions*. Cambridge University Press, 1995.

B Algorithms

Algorithm 1 univariate OU process simulation and density study R Code.

```
rm(list=ls(all=TRUE))
set.seed(2021)

s              = 0
t              = 5
Xs             = 17
kappa          = 0.75
alpha          = 15
sigma          = 1.1
delta_t        = 1/250
startingstate  = 12
endstate       = 19
simulations    = 1000
timespace      = seq(s,t,delta_t)
statespace     = seq(startingstate,endstate,delta_t)

uni_OU_trajectory = function(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate)
{

  timeseq        = (seq(s,t,delta_t))
  uni_OU_dm     = matrix(0,nrow = length(timeseq), ncol = 1)
  Z1             = rnorm(1,mean = 0, sd = sqrt(delta_t))
  Xt             = Xs + kappa*(alpha-Xs)*delta_t + sigma*Z1
  uni_OU_dm[1] = Xt

  for(i in 2:length(timeseq))
  {
    dWt            = rnorm(1,mean = 0, sd = sqrt(delta_t))
    Xtplus1        = Xt  + kappa*(alpha-Xt)*delta_t + sigma*dWt
    Xt             = Xtplus1
    uni_OU_dm[i] = Xtplus1
  }

  X = uni_OU_dm


  plot(X~seq(s,t,delta_t),type ='l', col = "royalblue3",xlab="Time (e.g days)",ylab = "Xt
```

```
       (e.g Exchange Rate)")
  abline(h=15, col="purple", lty = 3, lwd = 1)
}

trajectory_plot = uni_OU_trajectory(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)

OU_perpective = function(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate)
{

  timespace  = seq(s,t,delta_t)
  statespace = seq(startingstate,endstate,delta_t)
   uni_OU_dm = matrix(0,length(timespace),length(statespace))

  for (t in s:length(timespace))
  {
    for (state in startingstate:length(statespace))
    {

      gamma      = ((sigma^2)*(1 - exp(-2*kappa*(timespace[t]-s))))^(1/2)
      dens_point = ((pi*gamma^2)/kappa)^(-1/2)*exp(-(statespace[state]-alpha-(Xs-
          alpha)*exp(-kappa*(timespace[t]-s)))^2*(kappa/gamma^2))
      uni_OU_dm[t,state] = dens_point
    }

  }


  persp(timespace,statespace,uni_OU_dm, col = "royalblue3",xlab="Times", ylab="States",
      zlab="Surface", border = NA, shade = 0.9 , theta = 45, phi = 35, r = 35)
}
perspective_plot = OU_perpective(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate)

OU_EM = function(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate,simulations)
{
  mufunc = function(Xt,t)  {     return(kappa*(alpha - Xt))  }

  sigfunc = function(Xt,t)  {    return(sigma)  }

  histfunc = function(Xs,s,t,delta_t,simulations)
  {

    Xt = rep(Xs,simulations)
```

```
77      timespace = seq(s,t,delta_t)
78
79      for(i in 2:length(timespace))
80      {
81        dWt = sqrt(delta_t)*rnorm(simulations)
82        Xt = Xt + mufunc(Xt, timespace[i])*delta_t + sigfunc(Xt,timespace[i])*dWt
83        hist(Xt, freq = FALSE, col = 'royalblue3', border = 'white', breaks = 50, main = NA
              ) # ylim = c(0,2)
84      }
85
86      return(list(Xt=Xt,time = t))
87
88    }
89
90    plot = histfunc(Xs,s,t,delta_t,simulations)
91  }
92
93  EM_plot = OU_EM(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate,simulations)
94
95  OU_theoretical1 = function(s,t,Xs,Xt,kappa,alpha,sigma)
96  {
97    gamma       = ((sigma^2)*(1 - exp(-2*kappa*(t-s))))^(1/2)
98    dens_point = ((pi*gamma^2)/kappa)^(-1/2)*exp(-(Xt-alpha-(Xs-alpha)*exp(-kappa*(t-s)))
          ^2*(kappa/gamma^2))
99
100   return(dens_point)
101 }
102
103 Xt = statespace
104 plot_theoretical1 = OU_theoretical1(s,t,Xs,Xt,kappa,alpha,sigma)
105
106 lines(plot_theoretical1~Xt,col = "black",lwd = 3)
107
108 OU_hermite = function(s,t,Xs,Xt,kappa,alpha,sigma,K)
109 {
110
111   invsigxt = 1/(sigma)
112   gamxt = ((Xt)/sigma) # = Yt
113   gamxs = ((Xs)/sigma) # = Ys
114   part1 = 1/sqrt(2*pi*(t-s))
115   part2 = exp( - (((gamxt - gamxs)^2)/(2*(t-s))) - (((gamxt^2)*kappa)/2) + (((gamxs^2)*
          kappa)/2) + ((gamxt*alpha*kappa)/sigma) - ((gamxs*alpha*kappa)/sigma))
116   p = part1*part2
```

```
  c1 = -(1/(6*sigma^2)) * (kappa*( 3*alpha^2*kappa - 3*(gamxt+gamxs)*alpha*kappa*sigma  +
       (-3 + gamxt^2*kappa + gamxt*gamxs*kappa +gamxs^2*kappa)*sigma^2))
  hermitedens = invsigxt*p
  if (K>0)  {    hermitedens = invsigxt*p*(1+(t-s)*c1)  }

  return(hermitedens)
   }

K = 1
Xt = statespace
plot_hermite = OU_hermite(s,t,Xs,Xt,kappa,alpha,sigma,K)
lines(plot_hermite~Xt,lty = 3,col = "darkorchid1", lwd = 3)

library(expm)
Xs = initial
y0 =c(1, Xs, Xs^2, Xs^3, Xs^4)

a_x = 0.25
b_x = 0.07
s_x = 0.022^(0.5)
A  = rbind(c(0,0,0,0,0),
           c(a_x*b_x, -a_x, 0, 0, 0),
           c(0, 2*a_x*b_x+s_x^2, -2*a_x, 0, 0),
           c(0, 0, 3*a_x*b_x+3*s_x^2, -3*a_x, 0),
           c(0, 0, 0, 4*a_x*b_x+6*s_x^2, -4*a_x))

yt =expm(A*(Tmax-Tstart))%*%y0

res_package$moments[,dim(res_package$moments)[2]]
yt

xt = states
u = yt[1:4+1]
mm = u*0

mm[1] = u[1]
mm[2] = u[2] - 1*mm[1]*u[1]
mm[3] = u[3] - 1*mm[1]*u[2] - 2*mm[2]*u[1]
mm[4] = u[4] - 1*mm[1]*u[3] - 3*mm[2]*u[2] - 3*mm[3]*u[1]

p = 1/3 * (3*(mm[4]/6)*mm[2] - ((mm[3]/2)^2))/((mm[4]/6)^2)
q = 1/27 *(27*((mm[4]/6)^2)*(mm[1]-xt) - 9*(mm[4]/6)*(mm[3]/2)*mm[2] + 2*((mm[3]/2)^3))
    /((mm[4]/6)^3)
```

```
chk = (q^2)/4 + (p^3)/27
th = -(mm[3]/2)/(3*(mm[4]/6))+(-q/2 + sqrt(chk))^(1/3) - (q/2 + sqrt(chk))^(1/3)

k = (mm[1]*th) + (mm[2]*th^2)/2 + (mm[3]*th^3)/6 + (mm[4]*th^4)/24
k1 = mm[1] + (mm[2]*th) + (mm[3]*th^2)/2 + (mm[4]*th^3)/6
k2 = mm[2] + (mm[3]*th) + (mm[4]*th^2)/2
k3 = mm[3] + (mm[4]*th)
k4 = mm[4]
dens = 1/sqrt(2*pi*(k2))*exp(k-th*k1)
dens

lines(dens~xt, type ='l',col = "green" ,xlab="Xt",ylab = "Density", lty = 1, lwd = 2)
lines(res_package$density[,dim(res_package$density)[2]]~states, lty = 2, lwd = 2, col = '
    red')

labels = c("Theoretical", "Hermite approx", "Euler-Maruyama", "Saddle pt approx")
legend("topright", inset = 0.0005, title = NA,labels,lty = c(1,3,2,3), lwd = c(3,3,6,3) ,
    col=c("black", "gray47","royalblue","green", "red"), bty = 'n')
```

Algorithm 2 univariate CIR process simulation and density study R Code.

```
rm(list=ls(all=TRUE))
set.seed(2021)

s              = 0
t              = 5 # years
Xs             = 0.15
kappa          = 0.9
alpha          = 0.3
sigma          = 0.075
delta_t        = 1/250    #step length #trade days in year
startingstate  = 0
endstate       = 1
simulations    = 1000
timespace      = seq(s,t,delta_t)
statespace     = seq(startingstate,endstate,delta_t)


CIR_trajectory = function(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate)
{

  timeseq        = (seq(s,t,delta_t))
  datamatrix     = matrix(0,nrow = length(timeseq), ncol = 1)
```

```
  Z1             = rnorm(1,mean = 0, sd = sqrt(delta_t))
  Xt             = Xs + kappa*(alpha-Xs)*delta_t + sigma*sqrt(Xs)*Z1
  datamatrix[1] = Xt

  for(i in 2:length(timeseq))
  {
    dWt            = rnorm(1,mean = 0, sd = sqrt(delta_t))
    Xtplus1        = Xt  + kappa*(alpha-Xt)*delta_t + sigma*sqrt(Xt)*dWt
    Xt             = Xtplus1
    datamatrix[i] = Xtplus1
  }

  X = datamatrix


  plot(X~seq(s,t,delta_t),type ='l', col = "royalblue",xlab="Time",ylab = "Xt")
  abline(h=0.3, col="magenta", lty = 3, lwd = 1)
}

trajectory_plot = CIR_trajectory(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate)
```

Algorithm 3 univariate BS process simulation and density study R Code.

```
library(Sim.DiffProc)

set.seed(2021)

phi <- 0.25
x0 <- 1
f <- expression(0.5 * phi^2 * x)
g <- expression(phi * x)

general_model <- snssde1d(drift = f, diffusion = g, x0 = x0, M = 100, type = "ito", col =
     blue,Dt=1/250)
general_model

plot(general_model, col = "royalblue", lwd = 1, ylab = "Xt")
lines(time(general_model),apply(general_model$X,1,mean),lwd=2,col = "magenta1")
lines(time(general_model),apply(general_model$X,1,bconfint,level=0.95)[1,],col = "black",
    lwd=2)
lines(time(general_model),apply(general_model$X,1,bconfint,level=0.95)[2,],col = "black",
    lwd=2)
legend("topright",c("mean path",paste("bound of", 95," percent confidence")),inset = .01,
```

```
    col=c("magenta1", "royalblue"),lwd=2,cex=0.8)

moments <- MEM.sde(drift = f, diffusion = g, type = "ito",)
moments

start <- c(m = x0, S = 0)
mem.general_model <- MEM.sde(drift = f, diffusion = g, type = "ito", solve = TRUE, parms
    = c(phi = 0.75), init = start, time = seq(0, 1, by = 0.001))
summary(mem.general_model, at = 1)

plot(mem.general_model$sol.ode, ylab = "m(t)", select = "m", xlab = "Time", main = "",
    col = "blue", lty = 1, las = 1, lwd = 2)
legend("topleft", expression(m[general_model](t)), inset = 0.01,  col = "blue", lty = 1,
    lwd = 2, cex = 1.4)
plot(mem.general_model$sol.ode, ylab = "S(t)", select = "S", xlab = "Time", main = "",
    col = "violet", lty = 1, las = 1, lwd = 2)
legend("topleft", expression(S[general_model](t)), inset = 0.01,  col = "violet", lty =
    1, lwd = 2, cex = 1.4)
```

Algorithm 4 univariate OU process parameter sensitivity analysis R code.

```
rm(list=ls(all=TRUE))
set.seed(2021)

s              = 0
t              = 5
Xs             = 16.5
kappa          = 0.85
alpha          = 15
sigma          = 0.75
delta_t        = 1/250
startingstate  = 12
endstate       = 19
simulations    = 1000
timespace      = seq(s,t,delta_t)
statespace     = seq(startingstate,endstate,delta_t)

uni_CIR_sim_traj = function(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate)
{

  timeseq        = (seq(s,t,delta_t))
  uni_CIR_dm     = matrix(0,nrow = length(timeseq), ncol = 1)
  Z1             = rnorm(1,mean = 0, sd = sqrt(delta_t))
```

```
  Xt             = Xs + kappa*(alpha-Xs)*delta_t + sigma*Z1
  uni_CIR_dm[1] = Xt

  for(i in 2:length(timeseq))
  {
    dWt            = rnorm(1,mean = 0, sd = sqrt(delta_t))
    Xtplus1        = Xt  + kappa*(alpha-Xt)*delta_t + sigma*dWt
    Xt             = Xtplus1
    uni_CIR_dm[i] = Xtplus1
  }

  X = uni_CIR_dm

  plot(X~seq(s,t,delta_t),type ='l', col = rainbow(1, start = runif(1,0.55,0.8), end =
      runif(1,0.55,0.7)) ,xlab="t",ylab = "Xt", ylim = c(5,20))
}


par(mfrow=c(3,4),ps=9,cex.lab=1,cex.axis=0.75,mar=c(1, 1, 2, 1), mgp=c(1.5, 0.8, 0), las
    =1)

alpha          = 15
sigma          = 0.75

kappa = 0.2
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(kappa == .(kappa)))
kappa = 0.85
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(kappa == .(kappa)))
kappa = 1.5
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(kappa == .(kappa)))
kappa = 2
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(kappa == .(kappa)))


alpha = 12
```

```
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(alpha == .(alpha)))
alpha = 5
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(alpha == .(alpha)))
alpha = 15
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(alpha == .(alpha)))
alpha = 220
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(alpha == .(alpha)))


kappa          = 0.85
alpha          = 15


sigma = 0.05
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(sigma == .(sigma)))
sigma = 0.45
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(sigma == .(sigma)))
sigma = 0.75
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(sigma == .(sigma)))
sigma = 1.5
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(sigma == .(sigma)))
```

Algorithm 5 univariate CIR process parameter sensitivity analysis R code.

```
rm(list=ls(all=TRUE))

s              = 0
```

```
t              = 5 # years
Xs             = 0.15
kappa          = 0.9
alpha          = 0.3
sigma          = 0.075
delta_t        = 1/250    #step length #trade days in year
startingstate  = 0
endstate       = 1
simulations    = 100
timespace      = seq(s,t,delta_t)
statespace     = seq(startingstate,endstate,delta_t)


uni_CIR_sim_traj = function(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate)
{

  timeseq        = (seq(s,t,delta_t))
  uni_CIR_dm     = matrix(0,nrow = length(timeseq), ncol = 1)
  Z1             = rnorm(1,mean = 0, sd = sqrt(delta_t))
  Xt             = Xs + kappa*(alpha-Xs)*delta_t + sigma*sqrt(Xs)*Z1
  uni_CIR_dm[1] = Xt

  for(i in 2:length(timeseq))
  {
    dWt            = rnorm(1,mean = 0, sd = sqrt(delta_t))
    Xtplus1        = Xt  + kappa*(alpha-Xt)*delta_t + sigma*sqrt(Xt)*dWt
    Xt             = Xtplus1
    uni_CIR_dm[i] = Xtplus1
  }

  X = uni_CIR_dm

  plot(X~seq(s,t,delta_t),type ='l', col = rainbow(1, start = runif(1,0.55,0.8), end =
      runif(1,0.55,0.7)) ,xlab="t",ylab = "Xt", ylim = c(-0.1,0.7))
}


par(mfrow=c(3,4),ps=9,cex.lab=1,cex.axis=0.75,mar=c(1, 1, 2, 1), mgp=c(1.5, 0.8, 0), las
    =1)

kappa          = 0.9
alpha          = 0.3
sigma          = 0.075
```

```
45
46 kappa = -0.5
47 trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
       endstate)
48 title(main=bquote(kappa == .(kappa)))
49 kappa = 0.5
50 trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
       endstate)
51 title(main=bquote(kappa == .(kappa)))
52 kappa = 0.9
53 trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
       endstate)
54 title(main=bquote(kappa == .(kappa)))
55 kappa = 2
56 trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
       endstate)
57 title(main=bquote(kappa == .(kappa)))
58
59
60 kappa          = 0.9
61 alpha          = 0.3
62 sigma          = 0.075
63
64 alpha = -0.5
65 trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
       endstate)
66 title(main=bquote(alpha == .(alpha)))
67 alpha = 0.3
68 trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
       endstate)
69 title(main=bquote(alpha == .(alpha)))
70 alpha = 0.5
71 trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
       endstate)
72 title(main=bquote(alpha == .(alpha)))
73 alpha = 1
74 trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
       endstate)
75 title(main=bquote(alpha == .(alpha)))
76
77
78 kappa          = 0.9
79 alpha          = 0.3
```

```
sigma          = 0.075

sigma = 0
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(sigma == .(sigma)))
sigma = 0.075
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(sigma == .(sigma)))
sigma = 0.25
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(sigma == .(sigma)))
sigma = 0.5
trajectory_plot = uni_CIR_sim_traj(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,
    endstate)
title(main=bquote(sigma == .(sigma)))
```

Algorithm 6 bivariate CIR model simulation study applied to the SARB's monetary policy implementation mechanism

```
#Bivariate CIR Diffusion Process Analysis
#General:
#1. dXt = mu1(Xt,Yt,t)*dt + sigma11(Xt,t)*dWt1
#2. dYt = mu2(Xt,Yt,t)*dt + sigma22(Xy,t)*dWt2
#Biv CIR:
#1. dXt = (alpha1*(beta1-Xt)-lambda1*Yt)*dt + sigma1*sqrt(Xt)*dWt1
#2. dYt = (alpha2*(beta2-Yt)-lambda1*Xt)*dt + sigma2*sqrt(Yt)*dWt2

rm(list=ls(all=TRUE))

#Seed
seed = round(runif(1, min=0, max=10000))
set.seed(seed)

#Parameters
s              = 0
t              = 24
Xs             = 3.8
Ys             = 6.5
delta_t        = 1/30    #step length
startingstate = 0
```

```
endstate       = 5
numbsims       = 10000
timespace      = seq(s,t,delta_t)
statespace     = seq(startingstate,endstate,delta_t)


#Simulating the trajectory

CIR_trajectory = function(s,t,Xs,alpha1,beta1,sigma1,lambda1,Ys,alpha2,beta2,sigma2,
    lambda2,delta_t,startingstate,endstate)
{

  timeseq          = (seq(s,t,delta_t))
  datamatrix       = matrix(0,nrow = length(timeseq), ncol = 2)
  Z11              = rnorm(1,mean = 0, sd = sqrt(delta_t))
  Z21              = rnorm(1,mean = 0, sd = sqrt(delta_t))
  Xt               = Xs + (alpha1*(beta1-Xs)-lambda1*Ys)*delta_t + sigma1*sqrt(Xs)*Z11
  Yt               = Ys + (alpha2*(beta2-Xs)-lambda2*Xs)*delta_t + sigma2*sqrt(Ys)*Z21
  datamatrix[1,1] = Xt
  datamatrix[1,2] = Yt

  for(i in 2:length(timeseq))
  {
    dWt1             = rnorm(1,mean = 0, sd = sqrt(delta_t))
    dWt2             = rnorm(1,mean = 0, sd = sqrt(delta_t))
    Xtplus1          = Xt + (alpha1*(beta1-Xt)-lambda1*Yt)*delta_t + sigma1*sqrt(Xt)*dWt1
    Ytplus1          = round((Yt + (alpha2*(beta2-Yt)-lambda2*Xt)*delta_t + sigma2*sqrt(Yt
        )*dWt2)/0.5)*0.5
    Xt               = Xtplus1
    Yt               = Ytplus1
    datamatrix[i,1] = Xtplus1
    datamatrix[i,2] = Ytplus1
  }

  X = datamatrix

  par(mfrow=c(2,1),ps=9,cex.lab=1,cex.axis=0.75,mar=c(3, 3, 2, 1), mgp=c(1.5, 0.8, 0),
      las=1)
  plot(X[,1]~seq(s,t,delta_t),type ='l', col = "royalblue" ,xlab="Months",ylab=NA,ylim=c
      (0,10),lwd=2)
  lines(X[,2]~seq(s,t,delta_t),type ='l', col = "magenta" ,xlab="Months",ylab = NA,ylim=c
      (4,10),lwd=2)
  labels = c("Headline CPI", "Repurchase Rate")
```

```
  legend("bottomleft", title = NA,labels,lty = c(1,1), lwd = c(2,2) , col=c("royalblue","
      magenta"), bty = "n")

  plot(X[,1]~X[,2],type ='p', cex=0.5,col = "salmon1" ,xlab="Repo Rate",ylab = "Headline
      CPI", ylim = c(2,6))

}


beta1   = 4.2
sigma1  = 0.75
alpha1  = 2
lambda1 = 0.1
beta2   = 6.75
sigma2  = 0.5
alpha2  = 2
lambda2 = 0.1

trajectory_plot = CIR_trajectory(s,t,Xs,alpha1,beta1,sigma1,lambda1,Ys,alpha2,beta2,
    sigma2,lambda2,delta_t,startingstate,endstate)
```

Algorithm 7 Hermite and Saddlepoint approximate densities for the univariate CIR model

```
rm(list=ls(all=TRUE))
set.seed(2021)

s              = 0
t              = 5 # years
Xs             = 0.15
kappa          = 0.9
alpha          = 0.3
sigma          = 0.075
delta_t        = 1/250   #step length #trade days in year
startingstate  = 0
endstate       = 1
simulations    = 1000
timespace      = seq(s,t,delta_t)
statespace     = seq(startingstate,endstate,delta_t)

CIR_perpective = function(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate)
{
  #Creating the grid

```

```
  timespace  = seq(s,t,delta_t)
  statespace = seq(startingstate,endstate,delta_t)

  datamatrix = matrix(0,length(timespace),length(statespace))

  #Populating the matrix of densities

  for (t in s:length(timespace))
  {
    for (state in startingstate:length(statespace))
    {
      c                  = (2*kappa)/((sigma^2)*(1-exp(-kappa*(timespace[t]-s))))
      u                  = c*Xs*exp(-kappa*(timespace[t]-s))
      v                  = c*statespace[state]
      q                  = 2*kappa*alpha/(sigma^2) - 1
      besselparameter    = 2*(u*v)^(0.5)
      logbessel          = log(besselI(besselparameter,q,expon.scaled = TRUE))+
          besselparameter
      logfXt_t           = log(c) - (u+v) + (q/2)*log(v/u) + logbessel

      datamatrix[t,state] = exp(logfXt_t)
    }

  }

  #PLotting the perspective plot
  persp(timespace,statespace,datamatrix, col = "royalblue",xlab="t", ylab="Xt",zlab="p(xt
      |xs)", border = NA, shade = 0.9 , theta = 45, phi = 35, r = 35)
}

perspective_plot = CIR_perpective(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate
    )

CIR_EM = function(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate,simulations)
{
  mufunc = function(Xt,t)  {    return(kappa*(alpha - Xt))  }

  sigfunc = function(Xt,t)  {    return(sigma*sqrt(Xt))  }

  histfunc = function(Xs,s,t,delta_t,simulations)
  {

    Xt = rep(Xs,simulations)
```

```
    timespace = seq(s,t,delta_t)

    for(i in 2:length(timespace))
    {
      dWt = sqrt(delta_t)*rnorm(simulations)
      Xt = Xt + mufunc(Xt, timespace[i])*delta_t + sigfunc(Xt,timespace[i])*dWt
      hist(Xt, freq = FALSE, col = 'royalblue3', border = 'white', breaks = 50,  main =
          NA)
    }

    return(list(Xt=Xt,time = t))

  }

  plot = histfunc(Xs,s,t,delta_t,simulations)
}

EM_plot = CIR_EM(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate,simulations)


CIR_theoretical1 = function(s,t,Xs,Xt,kappa,alpha,sigma)
{
  c               = (2*kappa)/((sigma^2)*(1-exp(-kappa*(t-s))))
  u               = c*Xs*exp(-kappa*(t-s))
  v               = c*Xt
  q               = 2*kappa*alpha/(sigma^2) - 1
  besselparameter = 2*(u*v)^(0.5)
  besselfunction  = besselI(besselparameter,q,expon.scaled = TRUE)
  logbessel       = log(besselI(besselparameter,q,expon.scaled = TRUE))+besselparameter
  logfXt          = log(c) - (u+v) + (q/2)*log(v/u) + logbessel
  return(exp(logfXt))
}

Xt = statespace
plot_theoretical1 = CIR_theoretical1(s,t,Xs,Xt,kappa,alpha,sigma)

lines(plot_theoretical1~Xt,col = "navy",lwd = 2,lty=3)

theodensity = function(Xs,Xt,s,t)
{
a = ((sigma^2)/kappa)*(exp(-kappa*(t-s))-exp(-2*kappa*(t-s)))
b = (1-exp(-kappa*(t-s)))
mean = Xs*exp(-kappa*(t-s)) + alpha*b
```

```
variance = Xs*a + alpha*((sigma^2)/(2*kappa))*(b^2)
theta = variance/mean
kappa = (mean^2)/variance
gammadensity = dgamma(Xt,scale=theta,shape=kappa)
 return(list(density = gammadensity, Xt = Xt))
}

Xt = statespace
plot2 = theodensity(Xs,Xt,s,t)

lines(plot2$density~plot2$Xt,col = "orchid",lwd = 4,lty=1)
labels = c("Euler-Maruyama", "Th Bessel", "Th Chi-Sq")
legend("topright", inset = 0.03, title = NA,labels,lty = c(2,3,1), lwd = c(6,2,2) ,col=c(
    "royalblue","white","orchid"), bty = 'n')

CIR_hermite = function(s,t,Xs,Xt,kappa,alpha,sigma,K)
{

  invsigxt = 1/(sigma*sqrt(Xt))
  gamxt = ((2*sqrt(Xt))/sigma) # = Yt
  gamxs = ((2*sqrt(Xs))/sigma) # = Ys
  p1 = 1/sqrt(2*pi*(t-s))
  p2 = exp(-((gamxt-gamxs)^2)/(2*(t-s))-(kappa*(gamxt^2)/4)+(kappa*(gamxs^2)/4))*(gamxt
      ^(-0.5+2*kappa*alpha/sigma^2))*(gamxs^(0.5-2*kappa*alpha/sigma^2))
  p = p1*p2
  c1 = -1/(24*gamxt*gamxs*sigma^4)*(48*(kappa*alpha)^2-48*kappa*alpha*(sigma^2)+9*(sigma
      ^4)+gamxt*(kappa^2)*(sigma^2)*gamxs*(-24*alpha+(gamxt^2)*(sigma^2))+(gamxt^2)*(
      kappa^2)*(sigma^4)*(gamxs^2)+gamxt*(kappa^2)*(sigma^4)*(gamxs^3))
  hermitedens = invsigxt*p
  if (K>0)   {    hermitedens = invsigxt*p*(1+(t-s)*c1)   }

  return(hermitedens)
}

K = 1
Xt = statespace
plot_hermite = CIR_hermite(s,t,Xs,Xt,kappa,alpha,sigma,K)

lines(plot_hermite~Xt,lty = 3,col = "gray47", lwd = 3)

del = Xs^2 + (alpha + ((sigma^2)/(2*kappa)))*(alpha-2*Xs)+alpha*(alpha + ((sigma^2)/(2*
    kappa)))+2*(alpha + ((sigma^2)/(2*kappa)))*(Xs-alpha)
gamma = kappa*(Xs^2 + (alpha + ((sigma^2)/(2*kappa)))*(alpha-2*Xs))+3*kappa*alpha*(alpha
```

```
    + ((sigma^2)/(2*kappa)))+4*kappa*(alpha + ((sigma^2)/(2*kappa)))*(Xs-alpha)
kappa = 2*(kappa^2)*alpha*(alpha + ((sigma^2)/(2*kappa)))
A = kappa/(6*kappa^3)
C = -4*((1/(4*kappa^2))*(gamma-9*A*kappa^2)-(1/(2*kappa))*(del-3*kappa*A))
B = (1/(2*kappa))*(del-3*kappa*A-kappa*C)
D = -A-B-C

gamma_star = Xs^3 + 3*(kappa*alpha + sigma^2)*(A + B + C + D)
lambda_star = 3*kappa*Xs^3 + 3*(kappa*alpha + sigma^2)*(6*kappa*A + 5*kappa*B + 4*kappa*C
     + 3*kappa*D)
omega_star = 2*(kappa^2)*Xs^3 + 3*(kappa*alpha + sigma^2)*(11*(kappa^2)*A + 6*(kappa^2)*B
     + 3*(kappa^2)*C + 2*(kappa^2)*D)
nu_star = 3*(kappa*alpha + sigma^2)*(6*A*kappa^3)

E = nu_star/(24*kappa^4)
I = (-1/(6*kappa^3))*(((omega_star-(13*nu_star/(12*kappa)))-12*(kappa^2)*(gamma_star-(nu_
    star/(24*kappa^3))))-4*kappa*((lambda_star-(3*nu_star/(8*kappa^2)))-7*kappa*((gamma_
    star-(nu_star/(24*kappa^3))))))
H = (1/(2*kappa^2))*(((lambda_star-(3*nu_star/(8*kappa^2)))-7*kappa*((gamma_star-(nu_star
    /(24*kappa^3))))) - 6*(kappa^2)*I)
G = (-1/kappa)*((gamma_star-(nu_star/(24*kappa^3)))+ 2*kappa*H + 3*kappa*I)
FF = -E - G - H - I

theomoment1 = Xs*exp(-kappa*(t-s)) + alpha*(1 - exp(-kappa*(t-s)))
theomoment2 = (Xs^2)*exp(-2*kappa*(t-s)) + (alpha + (sigma^2)/(2*kappa))*(alpha + 2*(Xs-
    alpha)*exp(-kappa*(t-s)) + (alpha - 2*Xs)*exp(-2*kappa*(t-s)))
theomoment3 = (Xs^3)*exp(-3*kappa*(t-s)) + (3*kappa*alpha+3*sigma^2)*(A + B*exp(-kappa*(t
    -s)) + C*exp(-2*kappa*(t-s)) + D*exp(-3*kappa*(t-s)))
theomoment4 = (Xs^4)*exp(-4*kappa*(t-s)) + (4*kappa*alpha + 6*sigma^2)*(E + FF*exp(-1*
    kappa*(t-s)) + G*exp(-2*kappa*(t-s)) + H*exp(-3*kappa*(t-s)) + I*exp(-4*kappa*(t-s)))


theocumulant1 = theomoment1
theocumulant2 = theomoment2-(theomoment1)^2
theocumulant3 = theomoment3 - 3*theomoment1*theomoment2 + 2*theomoment1^3
theocumulant4 = -6*(theomoment1^4) + 12*(theomoment1^2)*(theomoment2) - 3*(theomoment2^2)
     - 4*theomoment1*theomoment3 + theomoment4

X = statespace

s = (1/theocumulant3)*(sqrt(theocumulant2^2 - 2*theocumulant3*(theocumulant1-X)) -
    theocumulant2)
Ksapprox = theocumulant1*s + theocumulant2*((1/2)*s^2) + theocumulant3*((1/6)*s^3) +
```

```
    theocumulant4*((1/24)*s^4)
Ks2approx = theocumulant2 + theocumulant3*s + 0.5*theocumulant4*s^2

saddle_pt_approx = exp(Ksapprox-s*X)*sqrt(1/(2*pi*Ks2approx))
print(saddle_pt_approx)

lines(saddle_pt_approx~statespace,lty = 3,col = "firebrick1", lwd = 3)
labels = c("Theoretical", "Hermite approx", "Euler-Maruyama", "Saddle pt approx")
legend("topright", inset = 0.0005, title = NA,labels,lty = c(1,3,2,3), lwd = c(3,3,6,3) ,
    col=c("black", "gray47","dodgerblue3","firebrick1"), bty = 'n')
```

Algorithm 8 univariate CIR Hermite orders of approximation

```
rm(list=ls(all=TRUE))
library(RColorBrewer)
set.seed(2021)

s              = 0
t              = 5 # years
Xs             = 0.15
alpha          = 0.9
beta           = 0.3
sigma          = 0.075
delta_t        = 1/250    #step length #trade days in year
startingstate  = 0
endstate       = 1
numbsims       = 1000
timespace      = seq(s,t,delta_t)
statespace     = seq(startingstate,endstate,delta_t)

par(mfrow=c(3), ps=10,cex.lab=1.5,cex.axis=1,mar=c(3.5,3.5,3.5,2.5), mgp=c(2.5, 1, 0),
    las=1)

#Hermite Approximation:

CIR_main = function(Xt,K)
{

  Xt = statespace

  #Theoretical density (Sahalia 1999)

  CIR_theoretical1 = function(s,t,Xs,Xt,alpha,beta,sigma)
```

```
  {
    c                = (2*alpha)/((sigma^2)*(1-exp(-alpha*(t-s))))
    u                = c*Xs*exp(-alpha*(t-s))
    v                = c*Xt
    q                = 2*alpha*beta/(sigma^2) - 1
    besselparameter  = 2*(u*v)^(0.5)
    besselfunction   = besselI(besselparameter,q,expon.scaled = TRUE)
    logbessel        = log(besselI(besselparameter,q,expon.scaled = TRUE))+besselparameter
    logfXt           = log(c) - (u+v) + (q/2)*log(v/u) + logbessel
    return(exp(logfXt))
  }


  plot_theoretical1 = CIR_theoretical1(s,t,Xs,Xt,alpha,beta,sigma)

  plot(plot_theoretical1~Xt,col = "royalblue", type = "l" , lwd = 2, ylab = "Density",
      xlab="t")


CIR_hermite = function(s,t,Xs,Xt,alpha,beta,sigma,K)
{

  Xt           = statespace

  invsigxt     = 1/(sigma*sqrt(Xt))
  gamxt        = ((2*sqrt(Xt))/sigma) # = Yt
  gamxs        = ((2*sqrt(Xs))/sigma) # = Ys
  p1           = 1/sqrt(2*pi*(t-s))
  p2           = exp(-((gamxt-gamxs)^2)/(2*(t-s))-(alpha*(gamxt^2)/4)+(alpha*(gamxs^2)/4))
      *(gamxt^(-0.5+2*alpha*beta/sigma^2))*(gamxs^(0.5-2*alpha*beta/sigma^2))
  p            = p1*p2
  c1           = -1/(24*gamxt*gamxs*sigma^4)*(48*(alpha*beta)^2-48*alpha*beta*(sigma^2)
      +9*(sigma^4)+gamxt*(alpha^2)*(sigma^2)*gamxs*(-24*beta+(gamxt^2)*(sigma^2))+(gamxt
      ^2)*(alpha^2)*(sigma^4)*(gamxs^2)+gamxt*(alpha^2)*(sigma^4)*(gamxs^3))
  c2           = (1/(576*gamxt^2*gamxs^2))*(9*(256*alpha^4*beta^4-512*alpha^3*beta^3*sigma
      ^2+224*alpha*beta*sigma^6-15*sigma^8)+6*gamxt*alpha^2*sigma^2*(-24*beta+gamxt^2*
      sigma^2)*(16*beta^2*alpha^2-16*beta*alpha*sigma^2+3*sigma^4)*gamxs+gamxt^2*alpha^2*
      sigma^4*(672*beta^2*alpha^2-48*beta*alpha*(2+gamxt^2*alpha)*sigma^2+(-6+gamxt^4*
      alpha^2)*sigma^4)*gamxs^2+2*gamxt*alpha^2*sigma^4*(48*beta^2*alpha^2-24*beta*alpha
      *(2+gamxt^2*alpha)*sigma^2+(9+gamxt^4*alpha^2)*sigma^4)*gamxs^3+3*gamxt^2*alpha^4*
      sigma^6*(-16*beta+gamxt^2*sigma^2)*gamxs^4+2*gamxt^3*alpha^4*sigma^8*gamxs^5+gamxt
      ^2*alpha^4*sigma^8*gamxs^6)

```

```
62
63    if (K==0)  {    hermitedens = invsigxt*p  }
64
65    if (K==1)  {    hermitedens = invsigxt*p*(1+(t-s)*c1)  }
66
67    if (K==2)  {    hermitedens = invsigxt*p*(1+(t-s)*c1 + (((t-s)^2)/2)*c2)  }
68
69
70    hermite_plot = lines(hermitedens~Xt, lty = 3, col = "magenta", lwd = 5)
71
72    return(hermite_plot)
73
74  }
75
76  call_hermite  = CIR_hermite(s,t,Xs,Xt,alpha,beta,sigma,K)
77  return(call_hermite)
78
79  }
80
81
82  call_main = CIR_main(Xt,0)
83  labels = c("Theoretical", "Hermite: K=0")
84  legend("top", inset = -0.095, title = NA,labels,lty = c(1,3), lwd = c(2,3) , col=c("black
      ","azure4"), bty = 'n')
85
86  call_main = CIR_main(Xt,1)
87  labels = c("Theoretical", "Hermite: K=1")
88  legend("top", inset = -0.095, title = NA,labels,lty = c(1,3), lwd = c(2,3) , col=c("black
      ","azure4"), bty = 'n')
89
90  call_main = CIR_main(Xt,2)
91  labels = c("Theoretical", "Hermite: K=2")
92  legend("top", inset = -0.095, title = NA,labels,lty = c(1,3), lwd = c(2,3) , col=c("black
      ","azure4"), bty = 'n')
93
94  CIR_hermite_diff = function(s,t,Xs,Xt,alpha,beta,sigma,K1,K2)
95  {
96
97    Xt          = statespace
98
99    invsigxt    = 1/(sigma*sqrt(Xt))
100   gamxt       = ((2*sqrt(Xt))/sigma) # = Yt
101   gamxs       = ((2*sqrt(Xs))/sigma) # = Ys
```

```
  p1           = 1/sqrt(2*pi*(t-s))
  p2           = exp(-((gamxt-gamxs)^2)/(2*(t-s))-(alpha*(gamxt^2)/4)+(alpha*(gamxs^2)/4))
      *(gamxt^(-0.5+2*alpha*beta/sigma^2))*(gamxs^(0.5-2*alpha*beta/sigma^2))
  p            = p1*p2
  c1           = -1/(24*gamxt*gamxs*sigma^4)*(48*(alpha*beta)^2-48*alpha*beta*(sigma^2)
      +9*(sigma^4)+gamxt*(alpha^2)*(sigma^2)*gamxs*(-24*beta+(gamxt^2)*(sigma^2))+(gamxt
      ^2)*(alpha^2)*(sigma^4)*(gamxs^2)+gamxt*(alpha^2)*(sigma^4)*(gamxs^3))
  c2           = (1/(576*gamxt^2*gamxs^2))*(9*(256*alpha^4*beta^4-512*alpha^3*beta^3*sigma
      ^2+224*alpha*beta*sigma^6-15*sigma^8)+6*gamxt*alpha^2*sigma^2*(-24*beta+gamxt^2*
      sigma^2)*(16*beta^2*alpha^2-16*beta*alpha*sigma^2+3*sigma^4)*gamxs+gamxt^2*alpha^2*
      sigma^4*(672*beta^2*alpha^2-48*beta*alpha*(2+gamxt^2*alpha)*sigma^2+(-6+gamxt^4*
      alpha^2)*sigma^4)*gamxs^2+2*gamxt*alpha^2*sigma^4*(48*beta^2*alpha^2-24*beta*alpha
      *(2+gamxt^2*alpha)*sigma^2+(9+gamxt^4*alpha^2)*sigma^4)*gamxs^3+3*gamxt^2*alpha^4*
      sigma^6*(-16*beta+gamxt^2*sigma^2)*gamxs^4+2*gamxt^3*alpha^4*sigma^8*gamxs^5+gamxt
      ^2*alpha^4*sigma^8*gamxs^6)


  if (K1==0)  {    hermitedens1 = invsigxt*p  }

  if (K1==1)  {    hermitedens1 = invsigxt*p*(1+(t-s)*c1)  }

  if (K1==2)  {    hermitedens1 = invsigxt*p*(1+(t-s)*c1 + (((t-s)^2)/2)*c2)  }


  if (K2==0)  {    hermitedens2 = invsigxt*p  }

  if (K2==1)  {    hermitedens2 = invsigxt*p*(1+(t-s)*c1)  }

  if (K2==2)  {    hermitedens2 = invsigxt*p*(1+(t-s)*c1 + (((t-s)^2)/2)*c2)  }



  plot(hermitedens1~Xt, type = "l" , col = "azure4", lwd =2,ylim=c
      (0.5000000002,0.5000000004), xlim = c(2.4941,2.49418),ylab="Density",axes = F)
  axis(1, xaxp=c(2.4941, 2.49418, 1), las=2)
  #axis(2, yaxp=c(0.5000000002, 0.5000000004, 1),outer = F, las=2)
  title(main="Density in [0.5000000002, 0.5000000004]")
  box()
  lines(hermitedens2~Xt, lty = 1 , col = "cornflowerblue", lwd = 2, ylim=c
      (0.5000000002,0.5000000004), xlim = c(2.4941,2.49418),ylab=NA)

}

```

```
133
134 CIR_hermite_coeff = function(s,t,Xs,Xt,alpha,beta,sigma,coeff)
135 {
136
137   Xt           = statespace
138
139   invsigxt     = 1/(sigma*sqrt(Xt))
140   gamxt        = ((2*sqrt(Xt))/sigma) # = Yt
141   gamxs        = ((2*sqrt(Xs))/sigma) # = Ys
142   p1           = 1/sqrt(2*pi*(t-s))
143   p2           = exp(-((gamxt-gamxs)^2)/(2*(t-s))-(alpha*(gamxt^2)/4)+(alpha*(gamxs^2)/4))
        *(gamxt^(-0.5+2*alpha*beta/sigma^2))*(gamxs^(0.5-2*alpha*beta/sigma^2))
144   p            = p1*p2
145   c1           = -1/(24*gamxt*gamxs*sigma^4)*(48*(alpha*beta)^2-48*alpha*beta*(sigma^2)
        +9*(sigma^4)+gamxt*(alpha^2)*(sigma^2)*gamxs*(-24*beta+(gamxt^2)*(sigma^2))+(gamxt
        ^2)*(alpha^2)*(sigma^4)*(gamxs^2)+gamxt*(alpha^2)*(sigma^4)*(gamxs^3))
146   c2           = (1/(576*gamxt^2*gamxs^2))*(9*(256*alpha^4*beta^4-512*alpha^3*beta^3*sigma
        ^2+224*alpha*beta*sigma^6-15*sigma^8)+6*gamxt*alpha^2*sigma^2*(-24*beta+gamxt^2*
        sigma^2)*(16*beta^2*alpha^2-16*beta*alpha*sigma^2+3*sigma^4)*gamxs+gamxt^2*alpha^2*
        sigma^4*(672*beta^2*alpha^2-48*beta*alpha*(2+gamxt^2*alpha)*sigma^2+(-6+gamxt^4*
        alpha^2)*sigma^4)*gamxs^2+2*gamxt*alpha^2*sigma^4*(48*beta^2*alpha^2-24*beta*alpha
        *(2+gamxt^2*alpha)*sigma^2+(9+gamxt^4*alpha^2)*sigma^4)*gamxs^3+3*gamxt^2*alpha^4*
        sigma^6*(-16*beta+gamxt^2*sigma^2)*gamxs^4+2*gamxt^3*alpha^4*sigma^8*gamxs^5+gamxt
        ^2*alpha^4*sigma^8*gamxs^6)
147   sum.c1.c2        = c1+c2
148
149
150   if (coeff == 1)
151   {
152   plot(c1~Xt, type = "p" , col = "blue", lwd = 2)
153   }
154
155   if (coeff == 2)
156   {
157     plot(c2~Xt, type = "p" , col = "skyblue", lwd = 2)
158   }
159
160   if (coeff == 7)
161   {
162     plot(sum.c1.c2~Xt, type = "p" , col = "navy", lwd = 2)
163   }
164
165 }
```

```
par(mfrow=c(3,1))
CIR_hermite_coeff(s,t,Xs,Xt,alpha,beta,sigma,1)
CIR_hermite_coeff(s,t,Xs,Xt,alpha,beta,sigma,2)
CIR_hermite_coeff(s,t,Xs,Xt,alpha,beta,sigma,7)
```

Algorithm 9 univariate CIR diffusion Process Cumulants

```
rm(list=ls(all=TRUE))

set.seed(2021)


s              = 0
t              = 5 # years
Xs             = 0.15
alpha          = 0.9
beta           = 0.3
sigma          = 0.075
delta_t        = 1/250    #step length #trade days in year
startingstate  = 0
endstate       = 1
numbsims       = 1000
timespace      = seq(s,t,delta_t)
statespace     = seq(startingstate,endstate,delta_t)

par(mfrow=c(2,2),ps=10,cex.lab=1,cex.axis=1,mar=c(3.5,3.5,3.5,2.5), mgp=c(2.8, 1, 0), las
    =1)


CIR_cumulant1 = function(s,t,Xs,alpha,beta,sigma,delta_t,startingstate,endstate,numbsims)
{
  mufunc = function(Xt,t)
  {
    return(alpha*(beta - Xt))
  }

  sigfunc = function(Xt,t)
  {
    return(sigma*sqrt(Xt))
  }

  cumulantfunc = function(Xs,s,t,delta_t,numbsims)
  {
```

```
36
37      Xt = rep(Xs,numbsims)
38      timespace = seq(s,t,delta_t)
39
40      cumulant1mat = matrix(Xs,nrow=length(timespace),ncol=1)
41      for(i in 1:length(timespace))
42      {
43        dWt = sqrt(delta_t)*rnorm(numbsims)
44        Xt = Xt + mufunc(Xt, timespace[i])*delta_t + sigfunc(Xt,timespace[i])*dWt
45        cumulant1mat[i] = mean(Xt)
46      }
47
48      plot(cumulant1mat~timespace,xlab='t',ylab='k1(t)',type = 'p',lwd = 1 ,col = 'magenta'
            )
49
50
51     m_t1 = Xs*exp(-alpha*timespace) + beta*(1 - exp(-alpha*timespace))
52
53
54      K_t1 =m_t1
55      lines(K_t1~timespace,col="royalblue",lwd = 2)
56
57    }
58
59    c = cumulantfunc(Xs,s,t,delta_t,numbsims)
60  }
61
62
63  CIR_cumulant2 = function(s,t,Xs,alpha,beta,sigma,delta_t,startingstate,endstate,numbsims)
64  {
65    mufunc = function(Xt,t)
66    {
67      return(alpha*(beta - Xt))
68    }
69
70    sigfunc = function(Xt,t)
71    {
72      return(sigma*sqrt(Xt))
73    }
74
75    cumulantfunc = function(Xs,s,t,delta_t,numbsims)
76    {
77
```

```
78      Xt = rep(Xs,numbsims)
79      timespace = seq(s,t,delta_t)
80
81      cumulant2mat =  matrix(Xs,nrow=length(timespace),ncol=1)
82      for(i in 1:length(timespace))
83      {
84        dWt = sqrt(delta_t)*rnorm(numbsims)
85        Xt = Xt + mufunc(Xt, timespace[i])*delta_t + sigfunc(Xt,timespace[i])*dWt
86        cumulant2mat[i] = mean(Xt^2) - (mean(Xt))^2
87      }
88
89      plot(cumulant2mat~timespace,xlab='t',ylab='k2(t)',type = 'p',lwd = 1 ,col = 'magenta'
            )
90
91
92     m_t1 = Xs*exp(-alpha*timespace) + beta*(1 - exp(-alpha*timespace))
93     m_t2 = (Xs^2)*exp(-2*alpha*timespace) + (beta + (sigma^2)/(2*alpha))*(beta + 2*(Xs-
           beta)*exp(-alpha*timespace) + (beta - 2*Xs)*exp(-2*alpha*timespace))
94
95
96      K_t2 =m_t2-(m_t1)^2
97      lines(K_t2~timespace,col="royalblue",lwd = 2)
98
99    }
100
101   c = cumulantfunc(Xs,s,t,delta_t,numbsims)
102 }
103
104
105 CIR_cumulant3 = function(s,t,Xs,alpha,beta,sigma,delta_t,startingstate,endstate,numbsims)
106 {
107   mufunc = function(Xt,t)
108   {
109     return(alpha*(beta - Xt))
110   }
111
112   sigfunc = function(Xt,t)
113   {
114     return(sigma*sqrt(Xt))
115   }
116
117   cumulantfunc = function(Xs,s,t,delta_t,numbsims)
118   {
```

```
119
120     Xt = rep(Xs,numbsims)
121     timespace = seq(s,t,delta_t)
122
123     cumulant3mat =  matrix(Xs,nrow=length(timespace),ncol=1)
124     for(i in 1:length(timespace))
125     {
126       dWt = sqrt(delta_t)*rnorm(numbsims)
127       Xt = Xt + mufunc(Xt, timespace[i])*delta_t + sigfunc(Xt,timespace[i])*dWt
128       #K_t3 =m_t3 - 3*m_t1*m_t2 + 2*m_t1^3
129        cumulant3mat[i] = mean(Xt^3) - 3*mean(Xt)*(mean(Xt^2)) + 2*mean(Xt)^3
130     }
131
132     plot(cumulant3mat~timespace,xlab='t',ylab='k3(t)',type = 'p',lwd = 1 ,col = 'magenta'
            )
133
134     #theoretical Moment
135     del = Xs^2 + (beta + ((sigma^2)/(2*alpha)))*(beta-2*Xs)+beta*(beta + ((sigma^2)/(2*
            alpha)))+2*(beta + ((sigma^2)/(2*alpha)))*(Xs-beta)
136     gamma = alpha*(Xs^2 + (beta + ((sigma^2)/(2*alpha)))*(beta-2*Xs))+3*alpha*beta*(beta
            + ((sigma^2)/(2*alpha)))+4*alpha*(beta + ((sigma^2)/(2*alpha)))*(Xs-beta)
137     kappa = 2*(alpha^2)*beta*(beta + ((sigma^2)/(2*alpha)))
138     A = kappa/(6*alpha^3)
139     C = -4*((1/(4*alpha^2))*(gamma-9*A*alpha^2)-(1/(2*alpha))*(del-3*alpha*A))
140     B = (1/(2*alpha))*(del-3*alpha*A-alpha*C)
141     D = -A-B-C
142
143    m_t1 = Xs*exp(-alpha*timespace) + beta*(1 - exp(-alpha*timespace))
144    m_t2 = (Xs^2)*exp(-2*alpha*timespace) + (beta + (sigma^2)/(2*alpha))*(beta + 2*(Xs-
           beta)*exp(-alpha*timespace) + (beta - 2*Xs)*exp(-2*alpha*timespace))
145    m_t3 = (Xs^3)*exp(-3*alpha*timespace) + (3*alpha*beta+3*sigma^2)*(A + B*exp(-alpha*
           timespace) + C*exp(-2*alpha*timespace) + D*exp(-3*alpha*timespace))
146
147     #theoretical Cumulant
148     K_t1 =m_t1
149     K_t2 =m_t2-(m_t1)^2
150     K_t3 =m_t3 - 3*m_t1*m_t2 + 2*m_t1^3
151     lines(K_t3~timespace,col="royalblue",lwd = 2)
152
153   }
154
155   c = cumulantfunc(Xs,s,t,delta_t,numbsims)
156 }
```

```
157
158
159 CIR_cumulant4 = function(s,t,Xs,alpha,beta,sigma,delta_t,startingstate,endstate,numbsims)
160 {
161   mufunc = function(Xt,t)
162   {
163     return(alpha*(beta - Xt))
164   }
165
166   sigfunc = function(Xt,t)
167   {
168     return(sigma*sqrt(Xt))
169   }
170
171   cumulantfunc = function(Xs,s,t,delta_t,numbsims)
172   {
173
174     Xt = rep(Xs,numbsims)
175     timespace = seq(s,t,delta_t)
176
177     cumulant4mat =  matrix(Xs,nrow=length(timespace),ncol=1)
178     for(i in 1:length(timespace))
179     {
180       dWt = sqrt(delta_t)*rnorm(numbsims)
181       Xt = Xt + mufunc(Xt, timespace[i])*delta_t + sigfunc(Xt,timespace[i])*dWt
182       #K_t3 =m_t3 - 3*m_t1*m_t2 + 2*m_t1^3
183       cumulant4mat[i] = -6*(mean(Xt)^4) + 12*(mean(Xt)^2)*(mean(Xt^2)) - 3*(mean(Xt^2)^2)
              - 4*mean(Xt)*mean(Xt^3) + mean(Xt^4)
184     }
185
186     plot(cumulant4mat~timespace,xlab='t',ylab='k4(t)',type = 'p',lwd = 1 ,col = 'magenta'
          )
187
188     #theoretical Moment
189     del = Xs^2 + (beta + ((sigma^2)/(2*alpha)))*(beta-2*Xs)+beta*(beta + ((sigma^2)/(2*
          alpha)))+2*(beta + ((sigma^2)/(2*alpha)))*(Xs-beta)
190     gamma = alpha*(Xs^2 + (beta + ((sigma^2)/(2*alpha)))*(beta-2*Xs))+3*alpha*beta*(beta
          + ((sigma^2)/(2*alpha)))+4*alpha*(beta + ((sigma^2)/(2*alpha)))*(Xs-beta)
191     kappa = 2*(alpha^2)*beta*(beta + ((sigma^2)/(2*alpha)))
192     A = kappa/(6*alpha^3)
193     C = -4*((1/(4*alpha^2))*(gamma-9*A*alpha^2)-(1/(2*alpha))*(del-3*alpha*A))
194     B = (1/(2*alpha))*(del-3*alpha*A-alpha*C)
195     D = -A-B-C
```

```
196     gamma_star = Xs^3 + 3*(alpha*beta + sigma^2)*(A + B + C + D)
197     lambda_star = 3*alpha*Xs^3 + 3*(alpha*beta + sigma^2)*(6*alpha*A + 5*alpha*B + 4*
            alpha*C + 3*alpha*D)
198     omega_star = 2*(alpha^2)*Xs^3 + 3*(alpha*beta + sigma^2)*(11*(alpha^2)*A + 6*(alpha
            ^2)*B + 3*(alpha^2)*C + 2*(alpha^2)*D)
199     nu_star = 3*(alpha*beta + sigma^2)*(6*A*alpha^3)
200
201     E = nu_star/(24*alpha^4)
202     I = (-1/(6*alpha^3))*(((omega_star-(13*nu_star/(12*alpha)))-12*(alpha^2)*(gamma_star
            -(nu_star/(24*alpha^3))))-4*alpha*((lambda_star-(3*nu_star/(8*alpha^2)))-7*alpha
            *((gamma_star-(nu_star/(24*alpha^3))))))
203     H = (1/(2*alpha^2))*(((lambda_star-(3*nu_star/(8*alpha^2)))-7*alpha*((gamma_star-(nu_
            star/(24*alpha^3))))) - 6*(alpha^2)*I)
204     G = (-1/alpha)*((gamma_star-(nu_star/(24*alpha^3)))+ 2*alpha*H + 3*alpha*I)
205     FF = -E - G - H - I
206
207    m_t1 = Xs*exp(-alpha*timespace) + beta*(1 - exp(-alpha*timespace))
208    m_t2 = (Xs^2)*exp(-2*alpha*timespace) + (beta + (sigma^2)/(2*alpha))*(beta + 2*(Xs-
           beta)*exp(-alpha*timespace) + (beta - 2*Xs)*exp(-2*alpha*timespace))
209    m_t3 = (Xs^3)*exp(-3*alpha*timespace) + (3*alpha*beta+3*sigma^2)*(A + B*exp(-alpha*
           timespace) + C*exp(-2*alpha*timespace) + D*exp(-3*alpha*timespace))
210    m_t4 = (Xs^4)*exp(-4*alpha*timespace) + (4*alpha*beta + 6*sigma^2)*(E + FF*exp(-1*
           alpha*timespace) + G*exp(-2*alpha*timespace) + H*exp(-3*alpha*timespace) + I*exp
           (-4*alpha*timespace))
211
212     #theoretical Cumulant
213     K_t1 =m_t1
214     K_t2 =m_t2-(m_t1)^2
215     K_t3 =m_t3 - 3*m_t1*m_t2 + 2*m_t1^3
216     K_t4 = -6*(m_t1^4) + 12*(m_t1^2)*(m_t2) - 3*(m_t2^2) - 4*m_t1*m_t3 +m_t4
217     lines(K_t4~timespace,col="royalblue",lwd = 2)
218
219   }
220
221   c = cumulantfunc(Xs,s,t,delta_t,numbsims)
222 }
223
224 #Plots
225
226
227 C1_plot = CIR_cumulant1(s,t,Xs,alpha,beta,sigma,delta_t,startingstate,endstate,numbsims)
228 labels = c("Theoretical", "Emperical")
229 legend("bottomright", title = NA,labels,lty = c(1,3), lwd = c(2,3) ,col=c("royalblue","
```

```
    magenta"), bty = 'n', inset = -0.025)

C2_plot = CIR_cumulant2(s,t,Xs,alpha,beta,sigma,delta_t,startingstate,endstate,numbsims)
labels = c("Theoretical", "Emperical")
legend("bottomright", title = NA,labels,lty = c(1,3), lwd = c(2,3) ,col=c("royalblue","
    magenta"), bty = 'n', inset = -0.025)

C3_plot = CIR_cumulant3(s,t,Xs,alpha,beta,sigma,delta_t,startingstate,endstate,numbsims)
labels = c("Theoretical", "Emperical")
legend("bottomright", title = NA,labels,lty = c(1,3), lwd = c(2,3) ,col=c("royalblue","
    magenta"), bty = 'n', inset = -0.025)

C4_plot = CIR_cumulant4(s,t,Xs,alpha,beta,sigma,delta_t,startingstate,endstate,numbsims)
labels = c("Theoretical", "Emperical")
legend("bottomright", title = NA,labels,lty = c(1,3), lwd = c(2,3) ,col=c("royalblue","
    magenta"), bty = 'n',inset=-0.025)
```

Algorithm 10 OU Hermite and Saddlepoint approximate densities

```
#Univariate OU model:
#General model: dXt = mu(Xt,t)dt + sigma(Xt,t)dWt
#dXt = kappa*(alpha-Xt)*dt + sigma*dWt

rm(list=ls(all=TRUE))

library(RColorBrewer)
# col = brewer.pal(3, "GnBu")

set.seed(2021)

s             = 0
t             = 5
Xs            = 16.5
kappa         = 0.85
alpha         = 15
sigma         = 0.75
delta_t       = 1/250    #step length
startingstate = 12
endstate      = 19
numbsims      = 500
timespace     = seq(s,t,delta_t)
statespace    = seq(startingstate,endstate,delta_t)

```

```
25
26 OU_perpective = function(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate)
27 {
28   #Creating the grid
29
30   timespace  = seq(s,t,delta_t)
31   statespace = seq(startingstate,endstate,delta_t)
32
33   datamatrix = matrix(0,length(timespace),length(statespace))
34
35   #Populating the matrix of densities
36
37   for (t in s:length(timespace))
38   {
39     for (state in startingstate:length(statespace))
40     {
41
42       gamma      = ((sigma^2)*(1 - exp(-2*kappa*(timespace[t]-s))))^(1/2)
43       dens_point = ((pi*gamma^2)/kappa)^(-1/2)*exp(-(statespace[state]-alpha-(Xs-alpha)*
             exp(-kappa*(timespace[t]-s)))^2*(kappa/gamma^2))
44       datamatrix[t,state] = dens_point
45     }
46
47   }
48
49   #PLotting the perspective plot
50   persp(timespace,statespace,datamatrix, col = "dodgerblue3",xlab="Times", ylab="States",
         zlab="Surface", border = NA, shade = 0.9 , theta = 45, phi = 35, r = 35)
51 }
52
53  OU_perpective(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate)
54
55
56 OU_EM = function(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate,numbsims)
57 {
58   mufunc = function(Xt,t)
59   {
60     return(kappa*(alpha - Xt))
61   }
62
63   sigfunc = function(Xt,t)
64   {
65     return(sigma)
```

```
  }

  histfunc = function(Xs,s,t,delta_t,numbsims)
  {

    Xt = rep(Xs,numbsims)
    timespace = seq(s,t,delta_t)

    for(i in 2:length(timespace))
    {
      dWt = sqrt(delta_t)*rnorm(numbsims)
      Xt = Xt + mufunc(Xt, timespace[i])*delta_t + sigfunc(Xt,timespace[i])*dWt
      hist(Xt, freq = FALSE, col = 'royalblue', border = 'white', breaks = 50, main = NA)
           # ylim = c(0,2)
    }

    return(list(Xt=Xt,time = t))

  }

  plot = histfunc(Xs,s,t,delta_t,numbsims)
}

EM_plot = OU_EM(s,t,Xs,kappa,alpha,sigma,delta_t,startingstate,endstate,numbsims)

OU_theoretical1 = function(s,t,Xs,Xt,kappa,alpha,sigma)
{
  gamma      = ((sigma^2)*(1 - exp(-2*kappa*(t-s))))^(1/2)
  dens_point = ((pi*gamma^2)/kappa)^(-1/2)*exp(-(Xt-alpha-(Xs-alpha)*exp(-kappa*(t-s)))
      ^2*(kappa/gamma^2))

  return(dens_point)
}

Xt = statespace
plot_theoretical1 = OU_theoretical1(s,t,Xs,Xt,kappa,alpha,sigma)

lines(plot_theoretical1~Xt,col = "navy",lwd = 3)

OU_hermite = function(s,t,Xs,Xt,kappa,alpha,sigma,K)
{
  #-----
  invsigxt = 1/(sigma)
```

```
  gamxt = ((Xt)/sigma) # = Yt
  gamxs = ((Xs)/sigma) # = Ys
  part1 = 1/sqrt(2*pi*(t-s))
  part2 = exp( - (((gamxt - gamxs)^2)/(2*(t-s))) - (((gamxt^2)*kappa)/2) + (((gamxs^2)*
      kappa)/2) + ((gamxt*alpha*kappa)/sigma) - ((gamxs*alpha*kappa)/sigma))
  p = part1*part2
  c1 = -(1/(6*sigma^2)) * (kappa*( 3*alpha^2*kappa - 3*(gamxt+gamxs)*alpha*kappa*sigma  +
       (-3 + gamxt^2*kappa + gamxt*gamxs*kappa +gamxs^2*kappa)*sigma^2))
  hermitedens = invsigxt*p
  if (K>0)
  {
    hermitedens = invsigxt*p*(1+(t-s)*c1)
  }

  return(hermitedens)
  #----

}

K = 1
Xt = statespace
plot_hermite = OU_hermite(s,t,Xs,Xt,kappa,alpha,sigma,K)

lines(plot_hermite~Xt,lty = 3,col = "purple", lwd = 3)

#Parameters
s              = 0
t              = 5
Xs             = 16.5
kappa          = 0.85
alpha          = 15
sigma          = 0.75
delta_t        = 1/250    #step length
startingstate  = 12
endstate       = 19
numbsims       = 500
timespace      = seq(s,t,delta_t)
statespace     = seq(startingstate,endstate,delta_t)


states <- statespace
initial <- Xs
Tmax <- 5
```

```
Tstart <- 0
increment <- 1/250


library(expm)
Xs = initial
y0 =c(1, Xs, Xs^2, Xs^3, Xs^4)

a_x = kappa#kappa tempo
b_x = alpha #alpha mean
s_x = sigma
A  = rbind(c(0,0,0,0,0),
           c(a_x*b_x, -a_x, 0, 0, 0),
           c(+s_x^2, 2*a_x*b_x, -2*a_x, 0, 0),
           c(0, +3*s_x^2, 3*a_x*b_x, -3*a_x, 0),
           c(0, 0, +6*s_x^2+6*s_x^2, 4*a_x*b_x, -4*a_x))

yt =expm(A*(Tmax-Tstart))%*%y0

res_package$moments[,dim(res_package$moments)[2]]
yt

xt = states
u = yt[1:4+1]
mm = u*0

mm[1] = u[1]
mm[2] = u[2] - 1*mm[1]*u[1]
mm[3] = u[3] - 1*mm[1]*u[2] - 2*mm[2]*u[1]
mm[4] = u[4] - 1*mm[1]*u[3] - 3*mm[2]*u[2] - 3*mm[3]*u[1]

p = 1/3 * (3*(mm[4]/6)*mm[2] - ((mm[3]/2)^2))/((mm[4]/6)^2)
q = 1/27 *(27*((mm[4]/6)^2)*(mm[1]-xt) - 9*(mm[4]/6)*(mm[3]/2)*mm[2] + 2*((mm[3]/2)^3))
    /((mm[4]/6)^3)
chk = (q^2)/4 + (p^3)/27
th = -(mm[3]/2)/(3*(mm[4]/6))+(-q/2 + sqrt(chk))^(1/3) - (q/2 + sqrt(chk))^(1/3)

k = (mm[1]*th) + (mm[2]*th^2)/2 + (mm[3]*th^3)/6 + (mm[4]*th^4)/24
k1 = mm[1] + (mm[2]*th) + (mm[3]*th^2)/2 + (mm[4]*th^3)/6
k2 = mm[2] + (mm[3]*th) + (mm[4]*th^2)/2
k3 = mm[3] + (mm[4]*th)
k4 = mm[4]
dens = 1/sqrt(2*pi*(k2))*exp(k-th*k1)
```

```
dens

res_package$cumulants[,dim(res_package$cumulants)[2]]
mm[1]
mm[2]

dens = dnorm(states,mm[1],sqrt(mm[2]))

lines(dens~states, type ='l',col = "magenta" ,xlab="Xt",ylab = "Density", lty = 1, lwd =
    2)
#lines(res_package$density[,dim(res_package$density)[2]]~states, lty = 2, lwd = 5, col =
    'red')

labels = c("Theoretical","Euler-M", "Hermite", "Cumulant T")
legend("topright", inset = 0.0005, title = NA,labels,lty = c(1,3,2,3), lwd = c(3,3,6,3) ,
    col=c("navy", "royalblue","purple","magenta"), bty = 'n')
```

Algorithm 11 bivariate OU CPI-Repo analysis and densities.

```
#Multivariate CIR Jump process

rm(list=ls(all=TRUE))

library(readxl)
Forward0 <- read_excel("C:/Users/P523119/Dropbox/Thinus/SARB/FMD/RESMAN/Team/Byran/
    YieldCurvePCA/Forward.xlsx")
BondTermCorrelations <- read_excel("C:/Users/P523119/Dropbox/Thinus/SARB/FMD/RESMAN/Team/
    Byran/YieldCurvePCA/BondTermCorrelations.xlsx")
Forward <- Forward0[1000:2500,]

#n = nrow(Forward)

#Seed:
#set.seed(7)

#Parameters
s               = 0
t               = 6
delta_t         = 0.004    #step length
startingstate   = 0
endstate        = 5
numbsims        = 10
timespace       = seq(s,t,delta_t)
```

```
statespace     = seq(startingstate,endstate,delta_t)


alpha1 = 0.05
alpha2 = 0.1
alpha3 = 0.12
alpha4 = 0.17
alpha5 = 0.14
alpha6 = 0.15
alpha7 = 0.15
alpha8 = 0.15
alpha9 = 0.05
alpha10 = 0.05
alpha11 = 0.02
alpha12 = -0.02
alpha13 = -0.05

beta1  = 2.07745
beta2  = 2.30683
beta3  = 2.44751
beta4  = 2.57561
beta5  = 2.65372
beta6  = 2.73313
beta7  = 2.78896
beta8  = 2.81654
beta9  = 2.838
beta10 = 2.85736
beta11 = 2.98396
beta12 = 3.04719
beta13 = 3.11052


sigma1 = 0.1
sigma2 = 0.1
sigma3 = 0.1
sigma4 = 0.1
sigma5 = 0.1
sigma6 = 0.1
sigma7 = 0.1
sigma8 = 0.1
sigma9 = 0.1
sigma10 = 0.1
sigma11 = 0.1
```

```
sigma12 = 0.1
sigma13 = 0.1

lambda1 = 0.01
lambda2 = 0.01
lambda3 = 0.01
lambda4 = 0.01
lambda5 = 0.01
lambda6 = 0.01
lambda7 = 0.01
lambda8 = 0.01
lambda9 = 0.01
lambda10 = 0.01
lambda11 = 0.01
lambda12 = 0.01
lambda13 = 0.01


gamma = as.matrix(BondTermCorrelations,nrow=13,ncol=13) #adjust acordingly if required
gamma = gamma*0.1

#Simulating the trajectory


  timeseq           = (seq(s,t,delta_t))
  datamatrix        = matrix(0,nrow = length(timeseq), ncol = 13)
  Y = datamatrix
  Z = Y

for( k in 1:numbsims)
{
  j11               = rnorm(1,mean = 0, sd = sqrt(delta_t))
  j21               = rnorm(1,mean = 0, sd = sqrt(delta_t))
  j31               = rnorm(1,mean = 0, sd = sqrt(delta_t))
  j41               = rnorm(1,mean = 0, sd = sqrt(delta_t))
  j51               = rnorm(1,mean = 0, sd = sqrt(delta_t))
  j61               = rnorm(1,mean = 0, sd = sqrt(delta_t))
  j71               = rnorm(1,mean = 0, sd = sqrt(delta_t))
  j81               = rnorm(1,mean = 0, sd = sqrt(delta_t))
  j91               = rnorm(1,mean = 0, sd = sqrt(delta_t))
  j101              = rnorm(1,mean = 0, sd = sqrt(delta_t))
  j111              = rnorm(1,mean = 0, sd = sqrt(delta_t))
  j121              = rnorm(1,mean = 0, sd = sqrt(delta_t))
```

```
j131              = rnorm(1,mean = 0, sd = sqrt(delta_t))
poi11             = rpois(1,lambda1*(timespace[1]-0))
poi21             = rpois(1,lambda2*(timespace[1]-0))
poi31             = rpois(1,lambda3*(timespace[1]-0))
poi41             = rpois(1,lambda4*(timespace[1]-0))
poi51             = rpois(1,lambda5*(timespace[1]-0))
poi61             = rpois(1,lambda6*(timespace[1]-0))
poi71             = rpois(1,lambda7*(timespace[1]-0))
poi81             = rpois(1,lambda8*(timespace[1]-0))
poi91             = rpois(1,lambda9*(timespace[1]-0))
poi101            = rpois(1,lambda10*(timespace[1]-0))
poi111            = rpois(1,lambda11*(timespace[1]-0))
poi121            = rpois(1,lambda12*(timespace[1]-0))
poi131            = rpois(1,lambda13*(timespace[1]-0))
z11               = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*sin
    (2*pi*timeseq)))
z21               = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*sin
    (2*pi*timeseq)))
z31               = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*sin
    (2*pi*timeseq)))
z41               = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*sin
    (2*pi*timeseq)))
z51               = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*sin
    (2*pi*timeseq)))
z61               = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*sin
    (2*pi*timeseq)))
z71               = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*sin
    (2*pi*timeseq)))
z81               = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*sin
    (2*pi*timeseq)))
z91               = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*sin
    (2*pi*timeseq)))
z101              = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*sin
    (2*pi*timeseq)))
z111              = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*sin
    (2*pi*timeseq)))
z121              = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*sin
    (2*pi*timeseq)))
z131              = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*sin
    (2*pi*timeseq)))

Xsvec = matrix(0,nrow = 13, ncol = 1)
Xsvec[1] = Forward$`1`[1]
```

```
Xsvec[2] = Forward$`2`[1]
Xsvec[3] = Forward$`3`[1]
Xsvec[4] = Forward$`4`[1]
Xsvec[5] = Forward$`5`[1]
Xsvec[6] = Forward$`6`[1]
Xsvec[7] = Forward$`7`[1]
Xsvec[8] = Forward$`8`[1]
Xsvec[9] = Forward$`9`[1]
Xsvec[10] = Forward$`10`[1]
Xsvec[11] = Forward$`15`[1]
Xsvec[12] = Forward$`20`[1]
Xsvec[13] = Forward$`30`[1]


Xt1               = Xsvec[1] + alpha1*((beta1-Xsvec[1])-(gamma[1,]%*%Xsvec-Xsvec[1]))*
    delta_t + sigma1*sqrt(Xsvec[1])*j11 + z11*poi11
Xt2               = Xsvec[2] + alpha2*((beta2-Xsvec[2])-(gamma[2,]%*%Xsvec-Xsvec[2]))*
    delta_t + sigma2*sqrt(Xsvec[2])*j21 + z21*poi21
Xt3               = Xsvec[3] + alpha3*((beta3-Xsvec[3])-(gamma[3,]%*%Xsvec-Xsvec[3]))*
    delta_t + sigma3*sqrt(Xsvec[3])*j31 + z31*poi31
Xt4               = Xsvec[4] + alpha4*((beta4-Xsvec[4])-(gamma[4,]%*%Xsvec-Xsvec[4]))*
    delta_t + sigma4*sqrt(Xsvec[4])*j41 + z41*poi41
Xt5               = Xsvec[5] + alpha5*((beta5-Xsvec[5])-(gamma[5,]%*%Xsvec-Xsvec[5]))*
    delta_t + sigma5*sqrt(Xsvec[5])*j51 + z51*poi51
Xt6               = Xsvec[6] + alpha6*((beta6-Xsvec[6])-(gamma[6,]%*%Xsvec-Xsvec[6]))*
    delta_t + sigma6*sqrt(Xsvec[6])*j61 + z61*poi61
Xt7               = Xsvec[7] + alpha7*((beta7-Xsvec[7])-(gamma[7,]%*%Xsvec-Xsvec[7]))*
    delta_t + sigma7*sqrt(Xsvec[7])*j71 + z71*poi71
Xt8               = Xsvec[8] + alpha8*((beta8-Xsvec[8])-(gamma[8,]%*%Xsvec-Xsvec[8]))*
    delta_t + sigma8*sqrt(Xsvec[8])*j81 + z81*poi81
Xt9               = Xsvec[9] + alpha9*((beta9-Xsvec[9])-(gamma[9,]%*%Xsvec-Xsvec[9]))*
    delta_t + sigma9*sqrt(Xsvec[9])*j91 + z91*poi91
Xt10              = Xsvec[10] + alpha10*((beta10-Xsvec[10])-(gamma[10,]%*%Xsvec-Xsvec
    [10]))*delta_t + sigma10*sqrt(Xsvec[10])*j101 + z101*poi101
Xt11              = Xsvec[11] + alpha11*((beta11-Xsvec[11])-(gamma[11,]%*%Xsvec-Xsvec
    [11]))*delta_t + sigma11*sqrt(Xsvec[11])*j111 + z111*poi111
Xt12              = Xsvec[12] + alpha12*((beta12-Xsvec[12])-(gamma[12,]%*%Xsvec-Xsvec
    [12]))*delta_t + sigma12*sqrt(Xsvec[12])*j121 + z121*poi121
Xt13              = Xsvec[13] + alpha13*((beta13-Xsvec[13])-(gamma[13,]%*%Xsvec-Xsvec
    [13]))*delta_t + sigma13*sqrt(Xsvec[13])*j131 + z131*poi131

datamatrix[1,1]  = Xt1
datamatrix[1,2]  = Xt2
```

```
169   datamatrix[1,3]  = Xt3
170   datamatrix[1,4]  = Xt4
171   datamatrix[1,5]  = Xt5
172   datamatrix[1,6]  = Xt6
173   datamatrix[1,7]  = Xt7
174   datamatrix[1,8]  = Xt8
175   datamatrix[1,9]  = Xt9
176   datamatrix[1,10]  = Xt10
177   datamatrix[1,11]  = Xt11
178   datamatrix[1,12]  = Xt12
179   datamatrix[1,13]  = Xt13
180
181   for(i in 2:length(timeseq))
182   {
183     dWt1               = rnorm(1,mean = 0, sd = sqrt(delta_t))
184     dWt2               = rnorm(1,mean = 0, sd = sqrt(delta_t))
185     dWt3               = rnorm(1,mean = 0, sd = sqrt(delta_t))
186     dWt4               = rnorm(1,mean = 0, sd = sqrt(delta_t))
187     dWt5               = rnorm(1,mean = 0, sd = sqrt(delta_t))
188     dWt6               = rnorm(1,mean = 0, sd = sqrt(delta_t))
189     dWt7               = rnorm(1,mean = 0, sd = sqrt(delta_t))
190     dWt8               = rnorm(1,mean = 0, sd = sqrt(delta_t))
191     dWt9               = rnorm(1,mean = 0, sd = sqrt(delta_t))
192     dWt10              = rnorm(1,mean = 0, sd = sqrt(delta_t))
193     dWt11              = rnorm(1,mean = 0, sd = sqrt(delta_t))
194     dWt12              = rnorm(1,mean = 0, sd = sqrt(delta_t))
195     dWt13              = rnorm(1,mean = 0, sd = sqrt(delta_t))
196
197     z1                 = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*
          sin(2*pi*timeseq)))
198     z2                 = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*
          sin(2*pi*timeseq)))
199     z3                 = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*
          sin(2*pi*timeseq)))
200     z4                 = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*
          sin(2*pi*timeseq)))
201     z5                 = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*
          sin(2*pi*timeseq)))
202     z6                 = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*
          sin(2*pi*timeseq)))
203     z7                 = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*
          sin(2*pi*timeseq)))
204     z8                 = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*
```

```
        sin(2*pi*timeseq)))
205     z9                  = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*
        sin(2*pi*timeseq)))
206     z10                 = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*
        sin(2*pi*timeseq)))
207     z11                 = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*
        sin(2*pi*timeseq)))
208     z12                 = rnorm(1,mean = 0.15*(1 + sin(2*pi*timeseq)), sd = 0.15*(1 + 0.2*
        sin(2*pi*timeseq)))
209     z13                 = rnorm(1,mean = 0.25*(1 + sin(2*pi*timeseq)), sd = 0.25*(1 + 0.5*
        sin(2*pi*timeseq)))
210
211
212     poi1                = rpois(1,lambda1*(timespace[i]-timespace[i-1]))
213     poi2                = rpois(1,lambda2*(timespace[i]-timespace[i-1]))
214     poi3                = rpois(1,lambda3*(timespace[i]-timespace[i-1]))
215     poi4                = rpois(1,lambda4*(timespace[i]-timespace[i-1]))
216     poi5                = rpois(1,lambda5*(timespace[i]-timespace[i-1]))
217     poi6                = rpois(1,lambda6*(timespace[i]-timespace[i-1]))
218     poi7                = rpois(1,lambda7*(timespace[i]-timespace[i-1]))
219     poi8                = rpois(1,lambda8*(timespace[i]-timespace[i-1]))
220     poi9                = rpois(1,lambda9*(timespace[i]-timespace[i-1]))
221     poi10               = rpois(1,lambda10*(timespace[i]-timespace[i-1]))
222     poi11               = rpois(1,lambda11*(timespace[i]-timespace[i-1]))
223     poi12               = rpois(1,lambda12*(timespace[i]-timespace[i-1]))
224     poi13               = rpois(1,lambda13*(timespace[i]-timespace[i-1]))
225
226     c = as.matrix(datamatrix[i-1,],nrow=13,ncol=1)
227
228     Xt1plus1             = Xt1 + alpha1*((beta1-Xt1)-(gamma[1,]%*%c-Xt1))*delta_t + sigma1*
        sqrt(Xt1)*dWt1 + z1*poi1
229     Xt2plus1             = Xt2 + alpha2*((beta2-Xt2)-(gamma[2,]%*%c-Xt2))*delta_t + sigma2*
        sqrt(Xt2)*dWt2 + z2*poi2
230     Xt3plus1             = Xt3 + alpha3*((beta3-Xt3)-(gamma[3,]%*%c-Xt3))*delta_t + sigma3*
        sqrt(Xt3)*dWt3 + z3*poi3
231     Xt4plus1             = Xt4 + alpha4*((beta4-Xt4)-(gamma[4,]%*%c-Xt4))*delta_t + sigma4*
        sqrt(Xt4)*dWt4 + z4*poi4
232     Xt5plus1             = Xt5 + alpha5*((beta5-Xt5)-(gamma[5,]%*%c-Xt5))*delta_t + sigma5*
        sqrt(Xt5)*dWt5 + z5*poi5
233     Xt6plus1             = Xt6 + alpha6*((beta6-Xt6)-(gamma[6,]%*%c-Xt6))*delta_t + sigma6*
        sqrt(Xt6)*dWt6 + z6*poi6
234     Xt7plus1             = Xt7 + alpha7*((beta7-Xt7)-(gamma[7,]%*%c-Xt7))*delta_t + sigma7*
        sqrt(Xt7)*dWt7 + z7*poi7
```

```
    Xt8plus1            = Xt8 + alpha8*((beta8-Xt8)-(gamma[8,]%*%c-Xt8))*delta_t + sigma8*
        sqrt(Xt8)*dWt8 + z8*poi8
    Xt9plus1            = Xt9 + alpha9*((beta9-Xt9)-(gamma[9,]%*%c-Xt9))*delta_t + sigma9*
        sqrt(Xt9)*dWt9 + z9*poi9
    Xt10plus1           = Xt10 + alpha10*((beta10-Xt10)-(gamma[10,]%*%c-Xt10))*delta_t +
        sigma10*sqrt(Xt10)*dWt10 + z10*poi10
    Xt11plus1           = Xt11 + alpha11*((beta11-Xt11)-(gamma[11,]%*%c-Xt11))*delta_t +
        sigma11*sqrt(Xt11)*dWt11 + z11*poi11
    Xt12plus1           = Xt12 + alpha12*((beta12-Xt12)-(gamma[12,]%*%c-Xt12))*delta_t +
        sigma12*sqrt(Xt12)*dWt12 + z12*poi12
    Xt13plus1           = Xt13 + alpha13*((beta13-Xt13)-(gamma[13,]%*%c-Xt13))*delta_t +
        sigma13*sqrt(Xt13)*dWt13 + z13*poi13


    Xt1                 = Xt1plus1
    Xt2                 = Xt2plus1
    Xt3                 = Xt3plus1
    Xt4                 = Xt4plus1
    Xt5                 = Xt5plus1
    Xt6                 = Xt6plus1
    Xt7                 = Xt7plus1
    Xt8                 = Xt8plus1
    Xt9                 = Xt9plus1
    Xt10                = Xt10plus1
    Xt11                = Xt11plus1
    Xt12                = Xt12plus1
    Xt13                = Xt13plus1


    datamatrix[i,1]    = Xt1plus1
    datamatrix[i,2]    = Xt2plus1
    datamatrix[i,3]    = Xt3plus1
    datamatrix[i,4]    = Xt4plus1
    datamatrix[i,5]    = Xt5plus1
    datamatrix[i,6]    = Xt6plus1
    datamatrix[i,7]    = Xt7plus1
    datamatrix[i,8]    = Xt8plus1
    datamatrix[i,9]    = Xt9plus1
    datamatrix[i,10]   = Xt10plus1
    datamatrix[i,11]   = Xt11plus1
    datamatrix[i,12]   = Xt12plus1
    datamatrix[i,13]   = Xt13plus1

```

```
272
273 }
274
275 X = datamatrix
276 Y = Y + X
277 Z = (1/k)*Y
278
279 par(mfrow=c(5,3),ps=9,cex.lab=1,cex.axis=0.75,mar=c(3, 3, 2, 1), mgp=c(1.5, 0.8, 0),
        las=1)
280
281 # nd = nrow(datamatrix)
282 # nf = nrow(Forward)
283 # ns = length(seq(s,t,delta_t))
284 # nd
285 # nf
286 # ns
287 #
288
289 plot(X[,1]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "1",ylim=c
        (0,5))
290 lines(y=Forward$`1`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "1")
291 labels = c("Actual", "Simulated")
292 legend("topleft", title = NA,labels,lty = c(1,1), lwd = c(1,1) ,col=c("red","dodgerblue
        "), bty = 'n')
293
294 plot(X[,2]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "2",ylim=c
        (0,5))
295 lines(y=Forward$`2`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "2")
296
297
298 plot(X[,3]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "3",ylim=c
        (0,5))
299 lines(y=Forward$`3`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "3")
300
301
302 plot(X[,4]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "4",ylim=c
        (0,5))
303 lines(y=Forward$`4`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "4")
304
305
306 plot(X[,5]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "5",ylim=c
        (0,5))
307 lines(y=Forward$`5`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "5")
```

```
308
309
310 plot(X[,6]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "6",ylim=c
        (0,5))
311 lines(y=Forward$`6`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "6")
312
313
314 plot(X[,7]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "7",ylim=c
        (0,5))
315 lines(y=Forward$`7`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "7")
316
317
318 plot(X[,8]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "8",ylim=c
        (0,5))
319 lines(y=Forward$`8`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "8")
320
321
322 plot(X[,9]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "9",ylim=c
        (0,5))
323 lines(y=Forward$`9`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "9")
324
325
326 plot(X[,10]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "10",ylim=c
        (0,5))
327 lines(y=Forward$`10`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "10")
328
329
330 plot(X[,11]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "15",ylim=c
        (0,5))
331 lines(y=Forward$`15`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "15")
332
333
334 plot(X[,12]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "20",ylim=c
        (0,5))
335 lines(y=Forward$`20`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "20")
336
337
338 plot(X[,13]~seq(s,t,delta_t),type ='l', col = "dodgerblue" ,xlab="t",ylab = "30",ylim=c
        (0,5))
339 lines(y=Forward$`30`,x=seq(s,t,delta_t),type ='l', col = "red" ,xlab="t",ylab = "30")
340
341 plot(Z[1000,]~seq(1,13,1),type ='l', col = "dodgerblue" ,xlab="t",ylab = "Yield",ylim=c
        (0,5),lwd=2)
```

```
  lines(y=Forward[1000,],x=seq(1,13,1),type ='l', col = "red" ,xlab="t",ylab = "Term",lwd
      =2)
  labels = c("Actual", "Simulated")
  legend("topleft", title = NA,labels,lty = c(1,1), lwd = c(2,2) ,col=c("red","dodgerblue
      "), bty = 'n')
}
```

Algorithm 12 inference on VIX data based on the univariate CIR model

```
#Univariate UNI_CIR model:
#General model: dXt = mu(Xt,t)dt + sigma(Xt,t)dWt
#dXt = kappa*(alpha-Xt)*dt + sigma*sqrt(Xt)dWt

rm(list=ls(all=TRUE))
library(RColorBrewer)
set.seed(2021)


library(readxl)
VixData <- read_excel("C:/Users/P523119/Dropbox/Thinus/MastersAppliedDataAnalytics_MSc/
    Dissertation/FinalDissertation/VixData.xlsx")

dt = 1/262


X= VixData$`VIX Index`

plot(VixData$Dates,VixData$`VIX Index`, lty = 1, lwd = 1, col = "navy", type="l", ylab =
    "VIX", xlab = "date")

likelihood = function(theta)
{
  N   = length(X)
  Xs  = X[-N]
  Xt  = X[-1]
  k1  = Xs*exp(-theta[1]*dt)+theta[2]*(1-exp(-theta[1]*dt))
  k2  = theta[3]^2/(2*theta[1])*(1-exp(-2*theta[1]*dt))
  ldens = dnorm(Xt,k1,sqrt(k2), log = TRUE)
  return(-sum(ldens))

}

res = nlm(likelihood, c(50,50,50))
```

```
res


#Parameters
s              = 0
t              = 5
Xs             = 26
kappa          = 22.27057
alpha          = 21.46218
sigma          = 32.65406

delta_t        = 1/250   #step length
startingstate = 5, for the R code generating the plots.
endstate       = 35
numbsims       = 1000
timespace      = seq(s,t,delta_t)
statespace     = seq(startingstate,endstate,delta_t)


#Theoretical density 1: PLotted from the density given in Sahalia-paper

CIR_theoretical1 = function(s,t,Xs,Xt,kappa,alpha,sigma)
{
  gamma       = ((sigma^2)*(1 - exp(-2*kappa*(t-s))))^(1/2)
  dens_point = ((pi*gamma^2)/kappa)^(-1/2)*exp(-(Xt-alpha-(Xs-alpha)*exp(-kappa*(t-s)))
      ^2*(kappa/gamma^2))

  return(dens_point)
}

Xt = statespace
plot_theoretical1 = CIR_theoretical1(s,t,Xs,Xt,kappa,alpha,sigma)

plot(plot_theoretical1~Xt,col = "royalblue",lwd =2,ylab = 'Fitted Denisty', xlab = "VIX
    Value", type='l') #"p(xt|xs)"
‘
```

Algorithm 13 bivariate CIR inference on VIX and USDZAR values.

```
#Univariate UNI_CIR model:
#General model: dXt = mu(Xt,t)dt + sigma(Xt,t)dWt
#dXt = kappa*(alpha-Xt)*dt + sigma*sqrt(Xt)dWt

```

```
rm(list=ls(all=TRUE))
library(RColorBrewer)
set.seed(2021)


library(readxl)
VixData <- read_excel("C:/Users/P523119/Dropbox/Thinus/MastersAppliedDataAnalytics_MSc/Dissertation/FinalDissertation/VixData.xlsx")
USDZARData <- read_excel("C:/Users/P523119/Dropbox/Thinus/MastersAppliedDataAnalytics_MSc/Dissertation/FinalDissertation/USDZARData.xlsx")
dt = 1/262


X= VixData$`VIX Index`
Y = USDZARData$`USDZAR Curncy`

plot(VixData$Dates,VixData$`VIX Index`, lty = 1, lwd = 1, col = "navy", type="l", ylab = "Value", xlab = "date", ylim = c(14,40))
lines(VixData$Dates,USDZARData$`USDZAR Curncy`, lty = 1, lwd = 1, col = "magenta")
labels = c("VIX", "USDZAR")
legend("topright", inset = 0.0005, title = NA,labels,lty = c(1,1), lwd = c(1,1) ,col=c("navy","magenta"), bty = 'n')


likelihood = function(theta)
{
  N   = length(X)

  Xs = X[-N]
  Xt = X[-1]
  k1 = Xs*exp(-theta[1]*dt)+theta[2]*(1-exp(-theta[1]*dt))
  k2 = theta[3]^2/(2*theta[1])*(1-exp(-2*theta[1]*dt))
  ldens1 = dnorm(Xt,k1,sqrt(k2), log = TRUE)

  Ys = Y[-N]
  Yt = Y[-1]
  k12 = Ys*exp(-theta[4]*dt)+theta[5]*(1-exp(-theta[4]*dt))
  k22 = theta[6]^2/(2*theta[4])*(1-exp(-2*theta[4]*dt))
  ldens2 = dnorm(Yt,k12,sqrt(k22), log = TRUE)

  ldens_bi = ldens1+ldens2

  return(-sum(ldens_bi))
```

```
44
45  }
46
47  res = nlm(likelihood, c(50,50,50,16,15,5))
48  res
49
50
51  #Parameters
52  s              = 0
53  t              = 5
54  Xs             = 26
55  kappa          = 22.27057
56  alpha          = 21.46218
57  sigma          = 32.65406
58
59  delta_t        = 1/250    #step length
60  startingstate  = 5
61  endstate       = 35
62  numbsims       = 1000
63  timespace      = seq(s,t,delta_t)
64  statespace     = seq(startingstate,endstate,delta_t)
65
66
67  #Theoretical density 1: PLotted from the density given in Sahalia-paper
68
69  OU_theoretical1 = function(s,t,Xs,Xt,kappa,alpha,sigma)
70  {
71    gamma       = ((sigma^2)*(1 - exp(-2*kappa*(t-s))))^(1/2)
72    dens_point = ((pi*gamma^2)/kappa)^(-1/2)*exp(-(Xt-alpha-(Xs-alpha)*exp(-kappa*(t-s)))
          ^2*(kappa/gamma^2))
73
74    return(dens_point)
75  }
76
77  Xt = statespace
78  plot_theoretical1 = OU_theoretical1(s,t,Xs,Xt,kappa,alpha,sigma)
79
80  plot(plot_theoretical1~Xt,col = "royalblue",lwd =2,ylab = 'Fitted Denisty', xlab = "VIX
      Value", type='l') #"p(xt|xs)"
81  '
```

Algorithm 14 bivariate Heston modelfitted to S&P 500 and CBOE VIX data.

```
rm(list = ls())
library(DiffusionRgqd)

library(readxl)
S_P<- read_excel("C:/Users/P523119/Dropbox/Thinus/MastersAppliedDataAnalytics_MSc/
    Dissertation/FinalDissertation/spxvix.xlsx")


X = S_P$SPX
Y = S_P$VIX

par(mfrow=c(2,2))

plot(S_P$Dates,S_P$SPX, lty = 1, lwd = 1, col = "navy", type="l", ylab = "Index Value",
    xlab = "date")
labels = c("S&P 500")
legend("topright", inset = 0.0005, title = NA,labels,lty = c(1), lwd = c(1) ,col=c("navy"
    ), bty = 'n')
plot(S_P$Dates,S_P$VIX, lty = 1, lwd = 1, col = "magenta",ylab = "Volatility Value", xlab
     = "date", type='l')
labels2 = c("CBOE VIX")
legend("topright", inset = 0.0005, title = NA,labels2,lty = c(1), lwd = c(1) ,col=c("
    magenta"), bty = 'n',)


plot(S_P$Dates,log(S_P$SPX), lty = 1, lwd = 1, col = "royalblue", type="l", ylab = "log(
    Index Value)", xlab = "date")
labels = c("Transformed S&P 500")
legend("topright", inset = 0.0005, title = NA,labels,lty = c(1), lwd = c(1) ,col=c("
    royalblue"), bty = 'n')
plot(S_P$Dates,(S_P$VIX/100)^2, lty = 1, lwd = 1, col = "purple",ylab = "(VIX/100)^2",
    xlab = "date", type='l')
labels2 = c("Transformed CBOE VIX")
legend("topright", inset = 0.0005, title = NA,labels2,lty = c(1), lwd = c(1) ,col=c("
    purple"), bty = 'n',)


Z = cbind(log(X),(Y/100)^2)
Z

time_diff = diff(S_P$Dates)
time = cumsum(c(0,time_diff/365))

```

```
35
36 GQD.remove()
37 #X
38 a00 <- function(t){theta[1]}
39 a01 <- function(t){-0.5*theta[2]*theta[2]}
40 c01 <- function(t){theta[2]*theta[2]}
41 d01 <- function(t){theta[2]*theta[5]*theta[6]}
42 #Y
43 b00 <- function(t){theta[3]}
44 b01 <- function(t){-theta[4]}
45 e01 <- function(t){theta[2]*theta[5]*theta[6]}
46 f01 <- function(t){theta[5]*theta[5]}
47
48
49 theta.start <- c(8, 1, 0.05, 0.5, 1, 0)
50
51 model_h <- BiGQD.mle(Z, time, mesh = 100, theta = theta.start)
52
53
54 GQD.estimates(model_h)
55 GQD.aic(list(model_h))
56
57 theta <- c(0.143,0.673,0.566,8.138,0.726,-0.754)
```