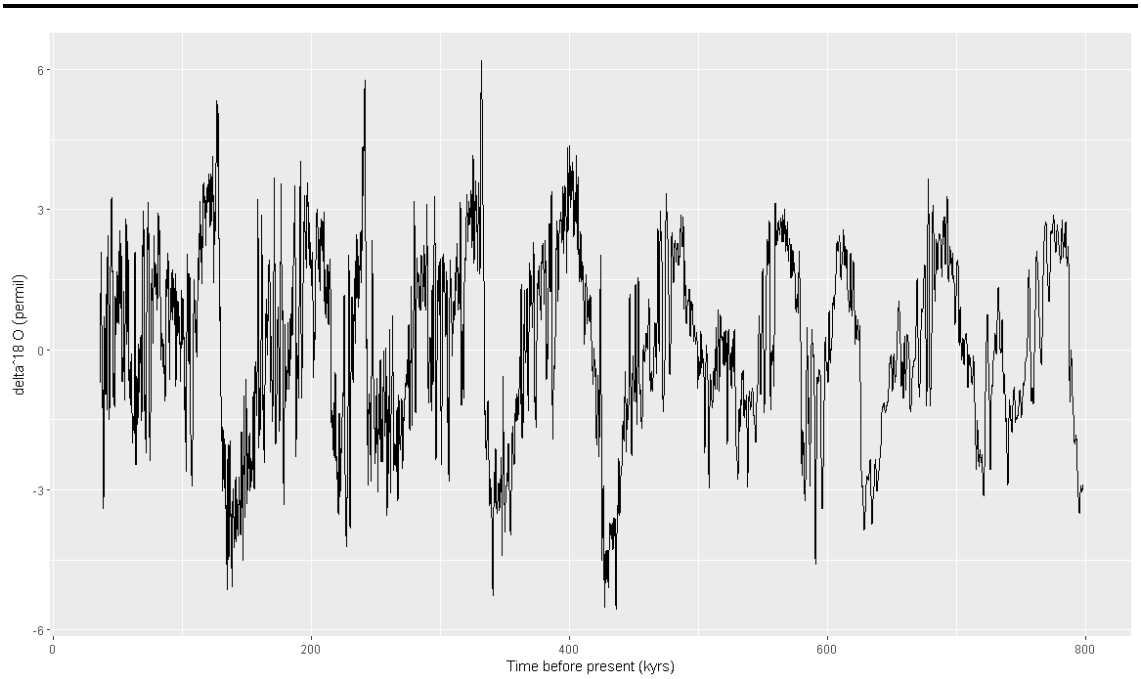
Bachelor project in mathematics (statistics)

Modelling of isotopic composition process by stochastic differential equations to predict climate variability



Gabriel Christian Boeskov (dnt911)
Supervisor: Susanne Ditlevsen
Submitted: 07. June 2024
Number of characters: 60.000

Department of Mathematical Sciences
Department of Mathematical Sciences
Pages: 34

Abstract

Stochastic Differential Equation (SDE) models have been increasingly utilized for analyzing the isotopic composition of ice core data. The most used models follow the generic form characterized by drift and diffusion components. This study is inspired by the synthetic ice core data record spanning 800 kyrs back as presented by Stephen Barker et al. (2011) [**1**], where I confront the challenge posed by the jump mechanisms inherent in the data. While the more traditional models, that solely incorporate drift and diffusion terms can offer insights as i will show, they may also fall short of modelling the abrupt jumps in the data. This research aims to tackle this issue by adding a jump term to the model consisting of a Poisson point process. Additionally since the models will be having a non-linear drift term, the transition densities will be analytically intractable. I will therefore use the Euler-Maruyama method for approximating the transition densities and thus I can estimate the parameters for the model by minimizing the negative log-likelihood function, that I will be deriving for both models.

Contents

1 Introduction

Ice core data provide a good insight into the earth's past climate variability. Oxygen isotopes are widely used to infer paleotemperatures. It is shown that minerals at low temperatures are enriched in $\delta^{18}O$, while minerals formed at higher temperatures show less enrichment (James F. Kasting et al. (2006)) [**2**]. Thus modelling the ice core data I gain insight into earth's past climate variability, the ability to predict future climate trends and give indications on the harm of anthropogenic activity. By measuring the isotopic composition at different length intervals of the ice cores, a time series can be obtained. This makes stochastic differential equations (SDEs) an effective tool for modelling these data. SDE's has been increasingly used in multiple fields to model scientific phenomena due to its ability to capture the deterministic and random components of their dynamics. The generic SDE follows the form

$$dX_t = \mu(X_t, t)dt + \Sigma(X_t, t)dW_t \tag{1}$$

where W_t represents a Wiener process, μ is the drift function, and Σ is the diffusion matrix. The drift function $\mu(X_t, t)$ captures the deterministic part of the system's evolution, while the diffusion term $\Sigma(X_t, t)dW_t$ accounts for the random part. In this paper I will be creating such a model and one with a jump term, which has the form

$$dX_t = \mu(X_t, t)dt + \Sigma(X_t, t)dW_t + \alpha(X_t, t)dN_t \tag{2}$$

Where α is the jump size's and N_t is a Poisson point process and the rest is defined as in the generic form.

To determine the optimal parameters for both models, I will employ Maximum Likelihood Estimation (MLE) theory. In the context of Stochastic Differential Equations (SDEs), finding the MLE parameters requires the transition densities of the data. However, for the model I will be using, these transition densities become analytically intractable. Thus, I will use the Euler-Maruyama approximation to estimate the transition densities.

Additionally, I will also use the Euler-Maruyama scheme to simulate data using the MLE-derived parameters. For improved accuracy of the Euler-Maruyama scheme, I will implement a subsampling technique during data simulation.

After obtaining the MLE parameters and generating data via the Euler-Maruyama method, I will utilize a bootstrap technique to assess the robustness of the optimization algorithm used to find the MLE parameters. This process will also allow me to derive confidence intervals for the bootstrapped distributions of the parameters based on the simulated data. This will also allow me to estimate how far the models are from bifurcation points, which i will be deriving.

In the following sections I will detail my theory, methodology, present the results of my modelling efforts, and discuss the broader implications of my findings.

2 Theory

I will start by considering the one dimensional SDE on the Itô form as

$$dX_t = \mu(X_t, \theta)dt + \Sigma(X_t, \theta)dW_t \tag{3}$$

formally defined as

$$X_{t+dt} = X_t + \int_t^{t+dt} \mu(X_u, \theta)du + \int_t^{t+dt} \Sigma(X_u, \theta)dW_u \tag{4}$$

Where X_t is the state variable at time t, $\mu(\cdot,\cdot) : \Theta \times \mathbb{R} \to \mathbb{R}$ is the drift function, $\Sigma(\cdot,\cdot) : \Theta \times \mathbb{R} \to \mathbb{R}$ is the diffusion matrix, here only a single value or a 1 by 1 matrix since were only considering a one dimensional model and W_t is a wiener process. I will also consider the above model but with a poisson point process term aswell. This model is defined as

$$dX_t = \mu(X_t, \theta)dt + \Sigma(X_t, \theta)dW_t + \alpha(X_t, t)dN_t \tag{5}$$

or formally

$$X_{t+dt} = X_t + \int_t^{t+dt} \mu(X_u, \theta)du + \int_t^{t+dt} \Sigma(X_u, \theta)dW_u + \int_t^{t+dt} \alpha(X_u, \theta)dN_u \tag{6}$$

With the same definitions as above and $\alpha(\cdot,\cdot) : \mathbb{R} \times \Theta \to \mathbb{R}$ is the jump function that determines the magnitude of the jumps. N_t is defined as a Lévy process where the increments are Poisson distributed with rate parameter λdt.

The Wiener process has the following properties[8]:

1) $W_0 = 0 \quad a.s.$
2) $W_{t+u} - W_t, u \geq 0$ is independent of $W_s, s < t$ i.e. independent increments
3) $W_{t+u} - W_t \sim \mathcal{N}(0, u), u \geq 0$
4) W_t a.s. continuous in t

And the Lévy process is known as the Poisson point process with same properties as the Wiener process except for property 3). In case of the Poisson point process the distribution of $N_{t+u} - N_t$ follow the Poisson distribution with rate parameter $\lambda((t+u) - t) = \lambda u$ as also stated above.

2.1 Models and Euler-Maruyama scheme

For this study I will firstly be using the additive double well potential model, and secondly be adding the jump term. This model follows the form

$$dX_t = -\frac{dV(X_t)}{dx}dt + \sigma dW_t \tag{7}$$

Where $\mu(X_u, \theta) = -\frac{dV(X_t)}{dx}$ and $\Sigma(X_u, \theta) = \sigma$ and $V(X_t)$ is the potential function defined as

$$V(X_t) = -\beta_1 \frac{X_t^4}{4} + \beta_2 \frac{X_t^2}{2} + \beta_3 X_t \tag{8}$$

resulting in the model

$$dX_t = (\beta_1 X_t^3 - \beta_2 X_t - \beta_3)dt + \sigma dW_t \tag{9}$$

and for the jump diffusion model we get with $\alpha(X_t, t) = e^{\frac{x^* - X_t}{\gamma}}$

$$dX_t = (\beta_1 X_t^3 - \beta_2 X_t - \beta_3)dt + \sigma dW_t + e^{\frac{x^* - X_t}{\gamma}} dN_t \tag{10}$$

Giving us the parameter vectors $\Theta = (\beta_1, \beta_2, \beta_3, \sigma)$ and $\Theta = (\beta_1, \beta_2, \beta_3, \sigma, \lambda, \gamma, x^*)$ for the model with only diffusion and the jump-diffusion model respectively.

If i consider the model in (3) then the Euler-Maruyama scheme is given by the following algorithm[3]

$$X_{t_{n+1}} = X_{t_n} + \mu(X_{t_n}, \theta)dt + \Sigma(X_{t_n}, \theta)dW_t \tag{11}$$

$$\Rightarrow X_{t_{n+1}} = X_{t_n} + (\beta_1 X_{t_n}^3 - \beta_2 X_{t_n} - \beta_3)dt + \sigma dW_t \tag{12}$$

and the Euler-Maruyama scheme for the jump diffusion model in (5) is given by

$$X_{t_{n+1}} = X_{t_n} + \mu(X_{t_n}, \theta)dt + \Sigma(X_{t_n}, \theta)dW_t + \alpha(X_{t_n}, t)dN_t \tag{13}$$

$$\Rightarrow X_{t_{n+1}} = X_{t_n} + (\beta_1 X_{t_n}^3 - \beta_2 X_{t_n} - \beta_3)dt + \sigma dW_t + e^{\frac{x^* - X_{t_n}}{\gamma}} dN_t \tag{14}$$

Which exists since the Euler-Maruyama scheme is well defined because the SDE is not hypoelliptic i.e. the matrix $\Sigma\Sigma^T = \sigma^2$ is not singular.

2.2 Parameter Estimation

For the parameter estimation in this paper, I will be approximating the transition densities for the models by the Euler-Maruyama scheme. The reason why we are approximating the transition densities is because the drift in both models is non-linear, leaving the exact transition density analytically intractable[3]. Thus making the likelihood terms intractable as well. Firstly because all SDE's posses the Markov property, we can write the likelihood of the data given some parameters as follow.

$$p(X_{t_1}, X_{t_2}, ..., X_{t_N}|\theta) = \prod_{i=1}^{N-1} p(X_{t_{i+1}}|X_{t_i}; \theta) \tag{15}$$

where each $p(X_{t_{n+1}}|X_{t_n}; \theta)$ is the transition density from time t_n to t_{n+1}. We will be using the maximum likelihood method, since we have a finite dataset $X_{t_1}, X_{t_2}, ..., X_{t_N}$, in order to find the best set of parameters. I do this by minimizing the negative log-likelihood function on the form

$$\ell(\theta) = -\log(p(X_{t_1}, X_{t_2}, ..., X_{t_N}|\theta)) \tag{16}$$

$$= -\log(\prod_{i=1}^{N-1} p(X_{t_{i+1}}|X_{t_i}; \theta)) \tag{17}$$

$$= -\sum_{i=1}^{N-1} \log(p(X_{t_{i+1}}|X_{t_i}; \theta)) \tag{18}$$

Where we have neglected the term $\log(p(X_{t_1}|\theta))$ because its influence vanishes asymptotically.
I now numerically optimize for[4] in order to obtain the Maximum likelihood estimators[5]

$$\hat{\theta}_{MLE} =_\theta \ell(\theta) \tag{19}$$

Next I replace the transition densities with the approximations from the Euler-Maruyama approximation in (12) and (14). For (12) I see that since the Euler-Maruyama approximation between two points define a Gaussian process i.e. some deterministic drift and the Wiener process which by definition produce the Gaussian process. Thus giving the approximation

$$p(X_{t_{n+1}}|X_{t_n}; \theta) \approx \mathcal{N}\left(X_{t_{n+1}}; X_{t_n} + \mu(X_{t_n}, t_n; \theta)dt, \Sigma(X_{t_n}, t_n; \theta)\Sigma(X_{t_n}, t_n; \theta)dt\right) \tag{20}$$

Resulting in the negative log-likelihood becomes

$$\ell(\theta) = -\log\left(\prod_{n=1}^{N-1} \frac{1}{\sqrt{2\pi\Sigma(X_{t_n}, t_n;\theta)\Sigma(X_{t_n}, t_n;\theta)dt}} \exp\left(-\frac{1}{2}\left(\frac{X_{t_{n+1}} - X_{t_n} - \mu(X_{t_n},\, t_n;\theta)dt}{\sqrt{\Sigma(X_{t_n}, t_n;\theta)\Sigma(X_{t_n}, t_n;\theta)dt}}\right)^2\right)\right) \tag{21}$$

$$= \frac{1}{2}\sum_{n=1}^{N-1} \log(2\pi\Sigma(X_{t_n}, t_n;\theta)\Sigma(X_{t_n}, t_n;\theta)dt) + \sum_{n=1}^{N-1} \frac{(X_{t_{n+1}} - X_{t_n} - \mu(X_{t_n},\, t_n;\theta)dt)^2}{2\Sigma(X_{t_n}, t_n;\theta)\Sigma(X_{t_n}, t_n;\theta)dt} \tag{22}$$

giving us

$$\ell(\theta) = \frac{1}{2}\sum_{n=1}^{N-1} \log(2\pi\sigma^2 dt) + \sum_{n=1}^{N-1} \frac{(X_{t_{n+1}} - X_{t_n} - (\beta_1 X_t^3 - \beta_2 X_t - \beta_3)dt)^2}{2\sigma^2 dt} \tag{23}$$

For the jump diffusion (10) model the Euler-Maruyama scheme (14) induce a Gaussian process with a Poisson point process for the jumps. This means that the transition density will be the transition density for the Gaussian process with mean $X_{t_n} + \mu(X_{t_n}, \theta)dt + \alpha(X_{t_n}, t)i$ where i is the number of jumps, resulting in the transition density

$$p(X_{t_{n+1}}|X_{t_n};\theta) \approx \mathcal{N}\left(X_{t_{n+1}}; X_{t_n} + \mu(X_{t_n},\, t_n;\theta)dt + \alpha(X_{t_n}, t)i,\, \Sigma(X_{t_n}, t_n;\theta)\Sigma(X_{t_n}, t_n;\theta)dt\right) \cdot \mathcal{P}(\lambda dt) \tag{24}$$

Where $\mathcal{P}(\lambda dt)$ is the Poisson distribution with mean λdt. This approximation gives us the negative log likelihood function

$$\begin{aligned}\ell(\theta) = -\log\Bigg(&\prod_{n=1}^{N-1}\sum_{i=0}^{\infty} \frac{1}{\sqrt{2\pi\Sigma(X_{t_n}, t_n;\theta)\Sigma(X_{t_n}, t_n;\theta)dt}} \\ &\exp\left(-\frac{1}{2}\left(\frac{X_{t_{n+1}} - X_{t_n} - \mu(X_{t_n},\, t_n;\theta)dt - \alpha(X_{t_n}, t)i}{\sqrt{\Sigma(X_{t_n}, t_n;\theta)\Sigma(X_{t_n}, t_n;\theta)dt}}\right)^2\right) \frac{(\lambda dt)^i}{i!}\exp\left(-\lambda dt\right)\Bigg)\end{aligned} \tag{25}$$

$$\begin{aligned}&= \frac{1}{2}\sum_{n=1}^{N-1}\sum_{i=0}^{\infty} \log(2\pi\Sigma(X_{t_n}, t_n;\theta)\Sigma(X_{t_n}, t_n;\theta)dt) + \\ &\frac{(X_{t_{n+1}} - X_{t_n} - \mu(X_{t_n},\, t_n;\theta)dt - \alpha(X_{t_n}, t)i)^2}{2\Sigma(X_{t_n}, t_n;\theta)\Sigma(X_{t_n}, t_n;\theta)dt} - \log\left(\frac{(\lambda dt)^i}{i!}\exp\left(-\lambda dt\right)\right)\end{aligned} \tag{26}$$

Giving us

$$\begin{aligned}&= \frac{1}{2}\sum_{n=1}^{N-1}\sum_{i=0}^{\infty} \log(2\pi\sigma^2 dt) + \frac{\left(X_{t_{n+1}} - X_{t_n} - (\beta_1 X_t^3 - \beta_2 X_t - \beta_3)dt - \exp\left(\frac{x^* - x}{\gamma}\right)i\right)^2}{2\sigma^2 dt} \\ &\quad - \log\left(\frac{(\lambda dt)^i}{i!}\exp\left(-\lambda dt\right)\right)\end{aligned} \tag{27}$$

Additionally when estimating the parameters I will truncate the sum in which I calculate the probability that i jumps appear, since I have for the Poisson distribution with rate parameter $\lambda \cdot dt$

and values $\lambda = 59.99457$ and $dt = 0.05$ that

$$\mathcal{P}\left(N \in \{0, ..., 10\}\right) = \sum_{i=0}^{10} \frac{(\lambda dt)^i}{i!} \exp(-\lambda dt) \tag{28}$$

$$\Rightarrow \mathcal{P}\left(N \notin \{0, ..., 10\}\right) = 1 - \sum_{i=0}^{10} \frac{(\lambda dt)^i}{i!} \exp(-\lambda dt) \tag{29}$$

$$= 1 - \sum_{i=0}^{10} \frac{(59.99457 \cdot 0.05)^i}{i!} \exp(-59.99457 \cdot 0.05) \tag{30}$$

$$= 0.0002921171 \tag{31}$$

the probability that more then 10 jumps appear is sufficiently small to justify the truncation. This is also further justified in [9] additionally in [10] they use the truncation of 15. Therefore since the calculation of the negative log-likelihood function runs in $O(n * I)$ where I is the number that is truncated to, the accuracy I gain from iterating to 15 is not deemed worthy as I would use a lot of computational power.

2.3 Lipschitz condition

In this section I will be studying the Lipschitz condition and some convergence results for the Euler-Maruyama scheme and our model. Firstly I will look at the Lipschitz condition for the drift, diffusion and Jump functions. A real-valued function is called Lipschitz continuous[11] if it for all real x_1 and x_2 holds that

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2| \tag{32}$$

We will show that this is the case for the diffusion function but not for the drift and jump functions. For the diffusion function $f(x) = \sigma$ we have

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2| \tag{33}$$

$$\Rightarrow |\sigma - \sigma| = 0 \leq K|x_1 - x_2| \quad \forall k \in \mathbb{R} \tag{34}$$

For the drift function $f(x) = -\beta_1 x^3 + \beta_2 x + \beta_3$ we have

$$|f(x_1) - f(x_2)| = |(\beta_1 x_1^3 - \beta_2 x_1 - \beta_3) - (\beta_1 x_2^3 - \beta_2 x_2 - \beta_3)| \tag{35}$$

$$= |\beta_1(x_1^3 - x_2^3) + \beta_2(x_2 - x_1) + (\beta_3 - \beta_3)| \tag{36}$$

$$= |\beta_1(x_1 - x_2)(x_2^2 + x_1^2 + x_1 x_2) + \beta_2(x_2 - x_1)| \tag{37}$$

$$= |x_1 - x_2||\beta_1(x_2^2 + x_1^2 + x_1 x_2) + \beta_2| \leq K|x_1 - x_2| \tag{38}$$

$$\Rightarrow |\beta_1(x_2^2 + x_1^2 + x_1 x_2) + \beta_2| \leq K \tag{39}$$

Now since $(x_2^2 + x_1^2 + x_1 x_2)$ grows without bound as x and y increases, no such constant K can satisfy this inequality for all $x, y \in \mathbb{R}$, Thus the drift function is not globally Lipschitz. However the drift function is locally Lipschitz, so for any bounded subset of $\mathbb{R}$, $|f(x_1) - f(x_2)|$ will be finite and therefore there exist such constant K such that the Lipschitz condition holds for that subset.

Note that for the jump function $f(x) = e^{\frac{x^* - x}{\gamma}}$ we have that since e^x is not uniformly continuous neither is $e^{\frac{x^* - x}{\gamma}}$ and since it is not uniformly continuous it is also not globally Lipschitz. It is still locally Lipschitz for the same reason that the drift function is.

3 Ice core data

The data i will be using for this project is a synthetic record of the ice core data gathered from Greenland spanning 800 thousands of years (kyrs) back. The data has been collected and synthetically generated to ensure equidistant observations throughout the entire period. The age distance between each observation is 0.05 kyrs or 50 yrs meaning that we will set $dt = 0.05$. The data is presented below

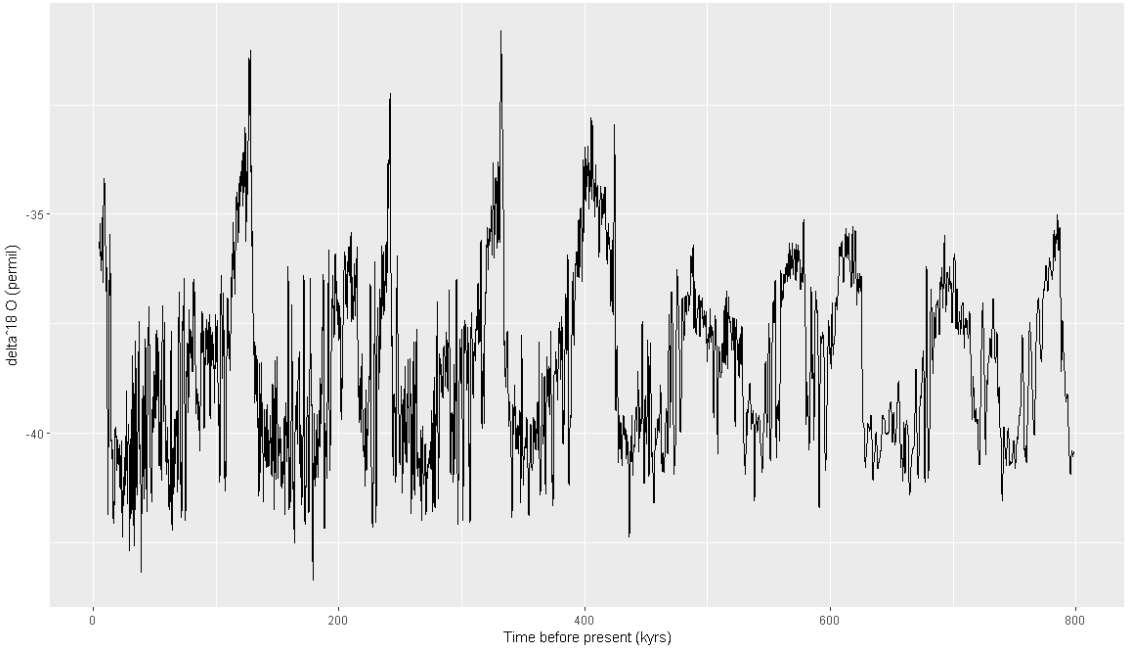


Figure 1: Ice core data against time before processing the data

Now note that the x-axis is measured in kyrs going back in time, so it is reversed. To correctly interpret the data, it should be viewed from right to left to ensure the chronological order is accurate.

The data is then processed by calculating the rolling mean for the previous 625 observations of a certain observation and subtracting that value from said observation. This operation is done to all observations except the first 625 since the rolling mean cant be calculated. This results in the following data

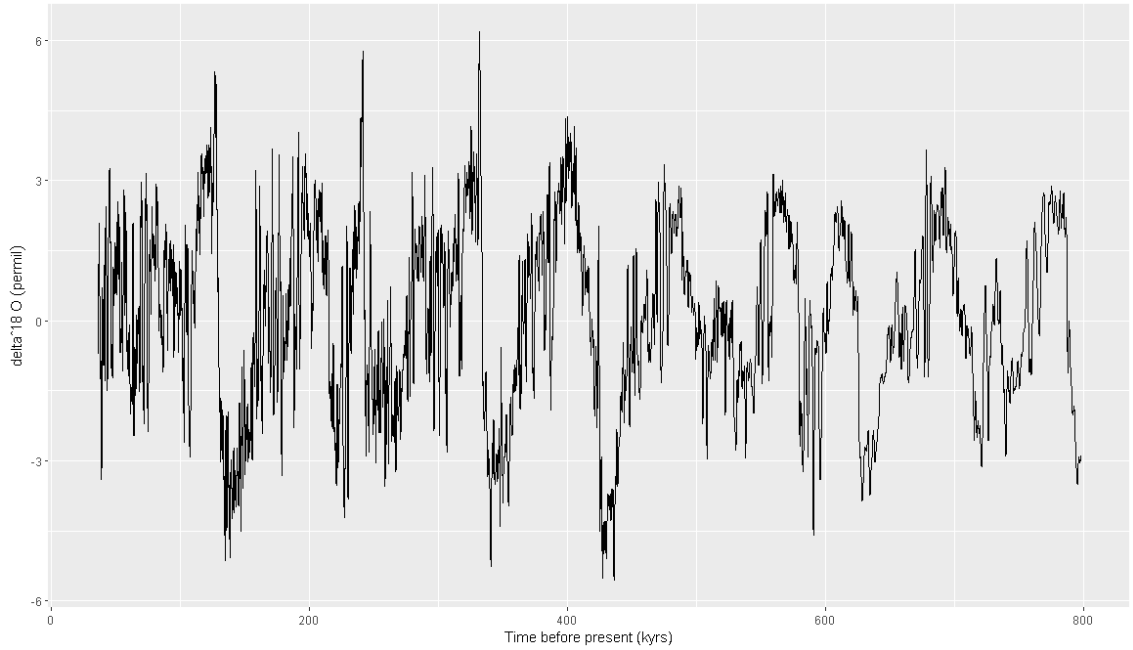


Figure 2: Ice core data against time after processing the data

Which will be the data that I will model after.

4 Methodology

For the model I decided to use the double-well potential model. I found this model well fitting because the data excerpts some bi-stability property, which the double-well potential model is designed to capture. Bistability describes a system with two stable states separated by some unstable state, in this models case it would be the wiener process and/or the jump process creating the transition between the states. The double well model without the jump term would be able to describe the model as we will see later but when observing the data I noticed that the transitions from a high value to a lower value is a slower and "smoother" transition and the transition from a low value to a higher value is a much faster and direct transition. This led me to model the data using the double well potential model with a jump term to better capture these asymmetric transition behaviors. The double-well potential function $V(x)$ (red) is shown below together with the negative derivative (blue) that we use as drift, with the values $\beta_1 = 1, \beta_2 = 2$ and $\beta_3 = \frac{1}{2}$

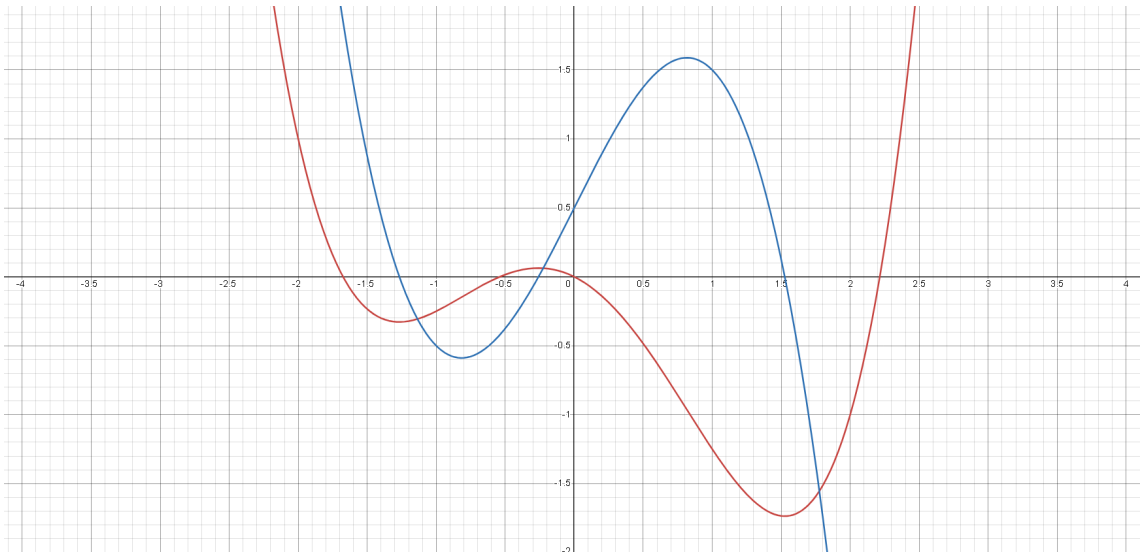


Figure 3: Potential function (red) and negative derivative (blue)

For the jump term i have used the function $\alpha(X_t, t) = e^{\frac{x^* - x}{\gamma}}$. The reason for this choice is that i want the jump to only be able to jump from one state and not the other one, as shown below

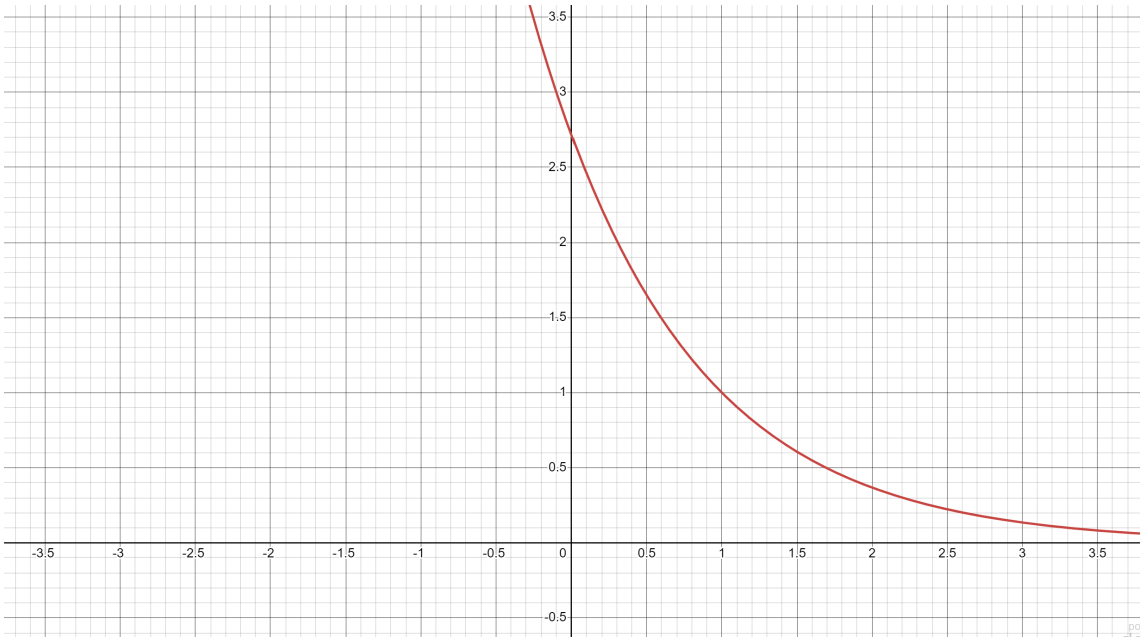


Figure 4: Jump function

The jump function is plotted for values $x^* = 1$ and $\gamma = 1$. The point of the function, relates to the dynamics within the double-well potential model. Specifically, we observe that the two stable states within the model should exhibit asymmetrical behavior in terms of the magnitude of their jumps. Because the observed data experiences asymmetric jumps, namely only jumps going upwards. Thus observing figure 3 we want only one of the wells to have large jump-term values and the other well to have smaller jump-term values. This asymmetry is therefore crucial as it closely mimics the one-sided jumps observed in the empirical data.

By performing the estimation and further study the parameters, I will gain insights into how the system transitions between the two states at a greater detail. I will discover how relevant the jump

transition's will be when comparing the model without and without the jump term. Additionally with the parameters estimated I can examine the models ability to have asymmetric stability in the two stable states.

This behavior can be particularly useful for understanding systems where such asymmetrical transitions are key characteristics. By fine-tuning the double-well potential model to replicate the one-sided jumps seen in the data, we can more accurately simulate and predict the behavior of the real-world system under study. Thus the analysis of the jump function under the given parameters not only enhances our understanding of the model's dynamics but also provides a robust framework for approximating the complex behaviors in the observed data. This approach may allow for a more precise and predictive modelling of systems characterized by asymmetric state transitions.

5 Parameter estimation

Estimating the parameters in SDE's is challenging because calculating the likelihood of the observations require knowledge of the transition densities between two discrete time points, which for complex SDE's is unknown. In this study we overcome this by using the Euler-Maruyama scheme to approximate the transition densities and enabling us to evaluate the likelihood of the observations. The Euler-Maruyama scheme induces a transition density resulting in the negative log-likelihood functions in (23) and (27). For the diffusion model and likelihood in (23) we initialize the optimizing algorithm with starting values $(\beta_1, \beta_2, \beta_3, \sigma) = (1, 1, 1, 1)$ using Nelder-Mead[6]. The resulting parameters for the model is

$\hat{\beta_1}$	$\hat{\beta_2}$	$\hat{\beta_3}$	$\hat{\sigma}$
-0.00069272	0.03581118	-0.00447839	0.55070983

Table 1: Maximum likelihood estimates for the parameters of the diffusion model in (9).

Rounded to the nearest 8'th decimal. The optimisation algorithm found convergence for these values.

Secondly, we perform the parameter estimation for the jump-diffusion model. The Nelder-Mead method with initial parameters $(\beta_1, \beta_2, \beta_3, \sigma, \lambda, \gamma, x^*) = (1, 1, 1, 1, 1, 1, 1)$ succeeds in converging to a set of parameters. Though it does not take a great deal of analytical ability to observe that the parameters are not very well suited for the model (shown in table below)

$\hat{\beta_1}$	$\hat{\beta_2}$	$\hat{\beta_3}$	$\hat{\sigma}$	$\hat{\lambda}$	$\hat{\gamma}$	$\hat{x^*}$
-9.324325	-72.613334	-39.554411	182.066875	40.048020	211.540513	-148.627924

Table 2: Maximum likelihood estimates with poor initial values for model in (10)

Simulating data with these values using the Euler-Maruyama scheme results in an extremely unstable process, diverging within only a few data points. In an attempt to combat the optimizers inability to handle bad initial parameters, I will firstly be using the resulting MLE parameters from the simpler model in table 1 as the initial parameters. These parameters are $(\beta_1, \beta_2, \beta_3, \sigma, \lambda, \gamma, x^*) = (-0.00069272, 0.03581118, -0.00447839, 0.55070983, 1, 2, 1.8)$ where the last three parameters were chosen with an educated guess. Secondly, I will be utilizing the L-BFGS-B method to identify improved initial values. Although my theoretical understanding of this technique is limited, the rationale behind its use is that L-BFGS-B allows us to constrain the parameter space by specifying upper and lower bounds for each parameter. The bounds for the

parameters were -5 for lower bound and 5 for upper bound, with lower bound for σ and λ being 0.0001, since the parameters are defined above 0. This helps in finding optimal parameters within a suitable range. Our previous experience has shown that unconstrained algorithms will converge to parameter values that are excessively extreme for the data I am modelling. By using L-BFGS-B, I can ensure that the parameters remain within a reasonable and effective range. The result of using the L-BFGS-B is a converging set of parameters but with convergence message "CONVERGENCE: REL REDUCTION OF F <= FACTR * EPSMCH". Resulting in convergence based on the lack of significant change in the negative log-likelihood function, which may not necessarily indicate that it has found the optimal solution. This type of convergence can sometimes imply that the algorithm is stuck in a local minimum or a flat region of the objective function. Therefore after finding these parameters, we will use them as initial parameters for the Nelder-Mead method, since it is not using upper or lower bounds for the optimizing and thus a less restrictive algorithm in an attempt to escape the local minimum or flat region. The initial parameters are thus $(\beta_1, \beta_2, \beta_3, \sigma, \lambda, \gamma, x^*) = (0.2534909, 0.8729793, 1.3926512, 1.2725224, 5.0000000, 0.8243845, -5.0000000)$. Here we observe that the L-BFGS-B method could not find improvement in λ and x^* from the bounds or the parameters. Lastly we run the Nelder-Mead with the above parameters and find convergence with parameters

$\hat{\beta_1}$	$\hat{\beta_2}$	$\hat{\beta_3}$	$\hat{\sigma}$	$\hat{\lambda}$	$\hat{\gamma}$	$\hat{x^*}$
−0.0007019743	0.03509965	−0.001341275	0.5506856	59.99457	6.018592	−60.56531

Table 3: Maximum likelihood estimates with poor initial values for model in (10)

Throughout this analysis I have truncated the sum calculating the probability of n jumps to only calculate the probability of 0 to 10 jumps, Thus neglecting the probabilities that 11 or more jumps appear. The reasoning behind is that since with the MLE parameter $\hat{\lambda} = 59.99457$ the probability that more then 10 jumps appear is 0.0002921, as explained in the theory section.

Here we observe for the first four parameters, a strong resemblance to the parameters in the diffusion model, which consist only of those four parameters. This makes sense because the parameter $\hat{\lambda}$ being 59.99457 results in the mean of the Poisson distribution, determining the amount of jumps, being $\hat{\lambda} \cdot dt = 59.99457 \cdot 0.05 = 2.9997285$. Inducing on average almost three jump between two data point, which is an excessive amount of jumps when relating the model to the observed data and what the aim of the jump term was. Additionally combined with the large mean of the jump distribution, the $\hat{\gamma}$ and $\hat{x^*}$ gave us jump sizes $\exp\left(\frac{x^*-x}{\gamma}\right) \Rightarrow \exp\left(\frac{-60.56531-x}{6.018592}\right)$, as shown in the plot below

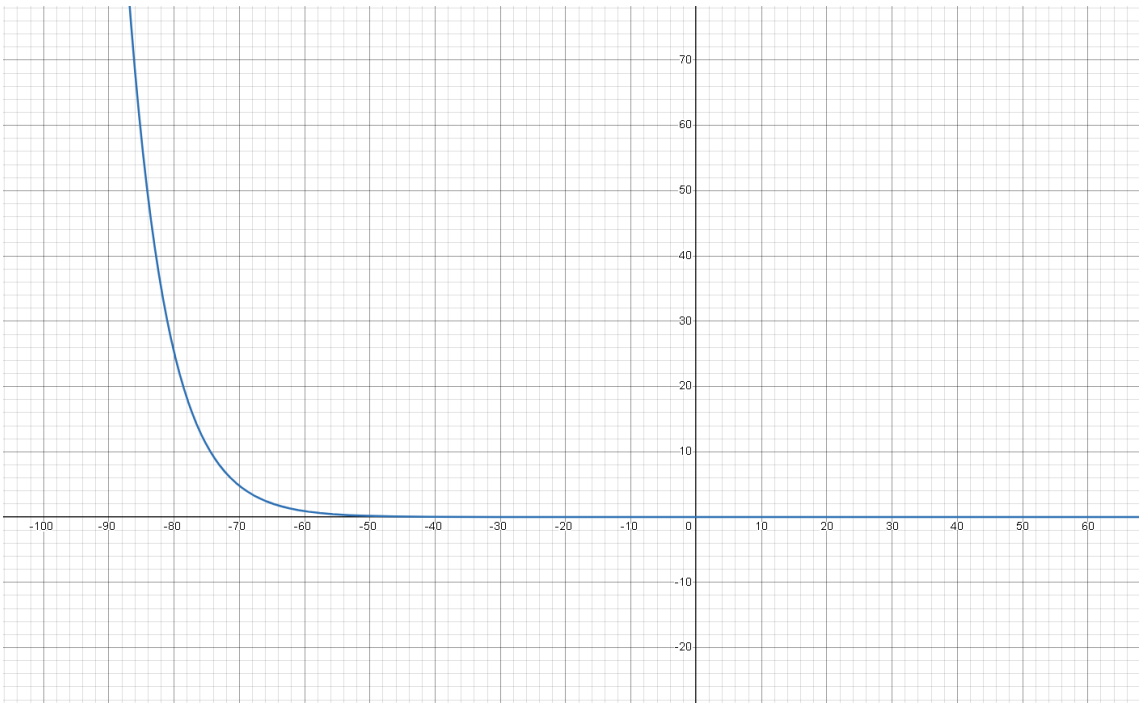


Figure 5: Jump function

Given that the observed data in Figure 2 rarely escapes the interval $[-6, 6]$, the jump function with the MLE parameters should be able to capture jumps within the interval $[\exp\left(\frac{-60.56531-6}{6.018592}\right), \exp\left(\frac{-60.56531-(-6)}{6.018592}\right)] = [0.0000157, 0.0001155]$. This results in very small jumps. It becomes apparent that the model fits the jump diffusion model by minimizing the jumps, resulting in many very small jumps. This approach might only help in modelling the variance, which was initially thought to be captured by the Wiener process, and is not well-suited for capturing the sharp jumps observed in the real data. Consequently, the model may not effectively capture the sharp jumps, which was the primary objective of incorporating the jump term. However, it might still offer a better representation of the data dynamics compared to the simpler diffusion model. Although the jump diffusion model essentially reduces to the diffusion model, as the extremely small jump term almost becomes negligible. The performance of both models will be analyzed in the following sections to determine which is best for modelling the data.

6 Bootstrap study

To further investigate the parameters, I perform a bootstrap study on them. I will do this by using the already found MLE parameters to simulate new data using the Euler-Maruyama scheme. Thereafter i will be performing the same maximum likelihood analysis as i did on the observed data, but on the simulated data this time. The initial values for the optimizing algorithm for the maximum likelihood analysis will also be the MLE parameters. This is to ensure convergence as we have already seen convergence for the maximum likelihood estimates. In the bootstrap study we will be doing 100 iterations in total, resulting in the distribution of each parameter, this will help me assessing the stability of the optimizing algorithm and the variance of the MLE parameters on Euler-Maruyama simulated data. To mitigate the discretization bias inherent in the Euler-Maruyama scheme, I will implement a subsampling technique on the simulated data, reducing the step size by a factor 10 from $dt = 0.05$ to $dt = \frac{0.05}{10} = 0.005$. This adjustment enhances the accuracy of the simulated path towards the actual solution paths. Subsampling ensures a finer granularity,

thus refining the approximation and bringing it closer to the true solution. The bootstrapped confidence interval for the MLE parameters is shown below for the diffusion model in (9).

Parameters:	$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_3$	$\hat{\sigma}$
Lower Bound:	−0.0067144834	0.000467015	−0.05756727	0.5455941
MLE Value:	−0.00069272	0.03581118	−0.00447839	0.55070983
Upper Bound:	0.0006922403	0.069858018	0.03716470	0.5550662

Table 4: Bootstrapped confidence interval for MLE parameters in diffusion model (9)

In the table above, we observe very tight confidence intervals, indicating the high precision of the parameter estimates. This suggests that the optimization algorithm used for maximum likelihood estimation (MLE) is robust and performs consistently well. Moreover, the data simulated using the estimated MLE parameters will inherit the properties of the observed data, further validating the accuracy of our model.

Additionally, the bootstrap analysis employed in this study provides a comprehensive assessment of the parameter estimation process. By repeatedly simulating new data and re-estimating the parameters, we ensure that the derived confidence intervals are reflective of the true parameter uncertainty. This method also helps in identifying any potential biases or inconsistencies in the MLE and optimizing approach.

Overall, the combination of the EM scheme for data simulation and the bootstrap analysis for evaluating the MLE parameter estimates, demonstrate the effectiveness of our approach. The results confirm that the simulated data retains the essential features of the observed data, thereby supporting the validity of our model and the stability of the optimization algorithm.

Parameters:	$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_3$	$\hat{\sigma}$	$\hat{\lambda}$	$\hat{\gamma}$	$\hat{x}^*$
Lower Bound:	−0.0055190	−0.004186	−0.034953	0.545356	59.9862	1.67860	−70.907
MLE Value:	−0.0007020	0.035099	−0.001341	0.550687	59.9946	6.01859	−60.565
Upper Bound:	0.0007887	0.061601	0.043760	0.556707	60.0202	7.13976	−47.401

Table 5: Bootstrapped confidence interval for MLE parameters in jump-diffusion model in (10).

In the bootstrap study for the jump diffusion model above, I observe very tight bounds around $\hat{\sigma}$ and $\hat{\gamma}$, again indicating that these parameters are accurately estimated and the optimization algorithm performs consistently with those parameters. For all the parameters included in the drift and diffusion term's, the confidence intervals are very similar to the ones we found in the simpler diffusion model, as were the MLE parameters. Since we have already concluded that the jump diffusion model basically reduced to the diffusion model with the MLE parameters, the found confidence intervals only confirms this more. Observing the parameters $\hat{\gamma}$ and $\hat{x}^*$, the confidence intervals are relatively large when comparing the parameters to the other parameters. This observation asserts the instability that I experienced with the optimizing algorithm and the fact that the model has trouble fitting to the data. When taking the tendency for the jump diffusion model to reduce itself to the simpler diffusion model, together with the fact that the parameters for the jump term and Poisson point process. It is hard to justify using the model with the jump term, since the jump term has almost no relevance and only increases the complexity of the model. To further confirm this theory I will study the resulting models and how well they are able to model the data in the following sections.

7 Bifurcation and stability of fixed points

Bifurcation in the context of stochastic differential equations, is when a drastic qualitative change in the behavior of a system occurs as a result of a parameter varying. The bifurcation refers to the critical parameter value in which the system undergoes the sudden change in the stability of its equilibrium points (fixed points). The double well potential model is an often used example when studying bifurcation as it dynamical behaviour and multiple equilibrium points creates some interesting bifurcation aspects. In order to understand how stable our system is we need to understand how close our control parameter (β_3) in the double-well potential drift function is to the bifurcation points. If the control parameter is far from the bifurcation point the model's behavior around the equilibrium points (the wells) is more predictable and stable. We see that for the function $f(x) = -0.00069272x^3 + 0.03581118x + 0.00447839$ which is our drift function for the diffusion model with MLE parameters. We see that for this function the fixed points are found when solving $f(x) = 0 \Rightarrow -0.00069272x^3 + 0.03581118x + 0.00447839 = 0$ which yields

$$\Rightarrow -0.00069272x^3 + 0.03581118x + 0.00447839 = 0 \tag{40}$$

$$\Rightarrow x \approx \begin{cases} -7.12666 \\ -0.125094 \\ 7.25176 \end{cases} \tag{41}$$

As also seen in the plot below

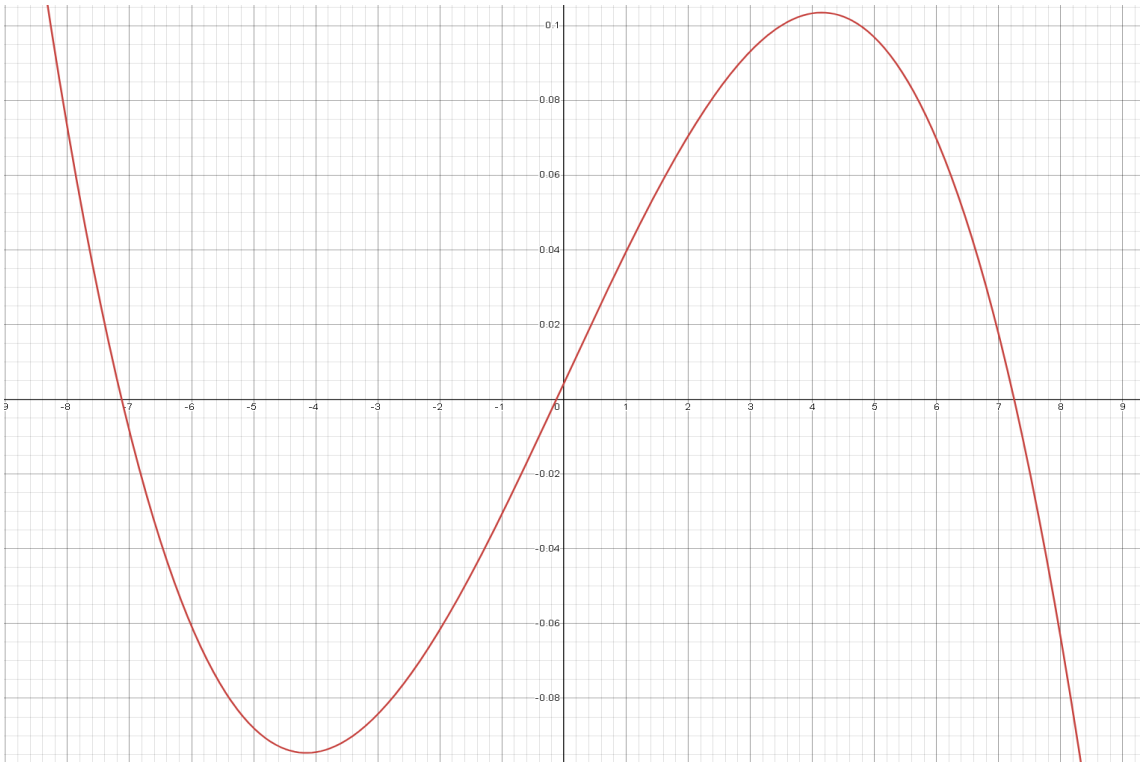


Figure 6: Drift function with MLE parameters to visualise wells and roots

To determine the stability and eventually which type of bifurcation would happen, we differentiate

$f(x)$ and evaluate the derivative at the solutions to $f(x) = 0$, this gives us

$$\frac{d}{dx}f(x) = \frac{d}{dx} - 0.00069272x^3 + 0.03581118x + 0.00447839 \tag{42}$$

$$= -0.00207816x^2 + 0.03581118 \tag{43}$$

$$\Rightarrow \begin{cases} \frac{d}{dx}f(-7.12666) & = -0.069737 \\ \frac{d}{dx}f(-0.125094) & = 0.035779 \\ \frac{d}{dx}f(7.25176) & = -0.073475 \end{cases} \tag{44}$$

Since if $\frac{d}{dx}f(X) < 0$ the fixed point X is stable and if $\frac{d}{dx}f(X) > 0$ the point is unstable resulting in stable fixed points $X \approx -7.12666$ and $X \approx 7.25176$ and unstable fixed point $X \approx -0.125094$, creating the stability plot below

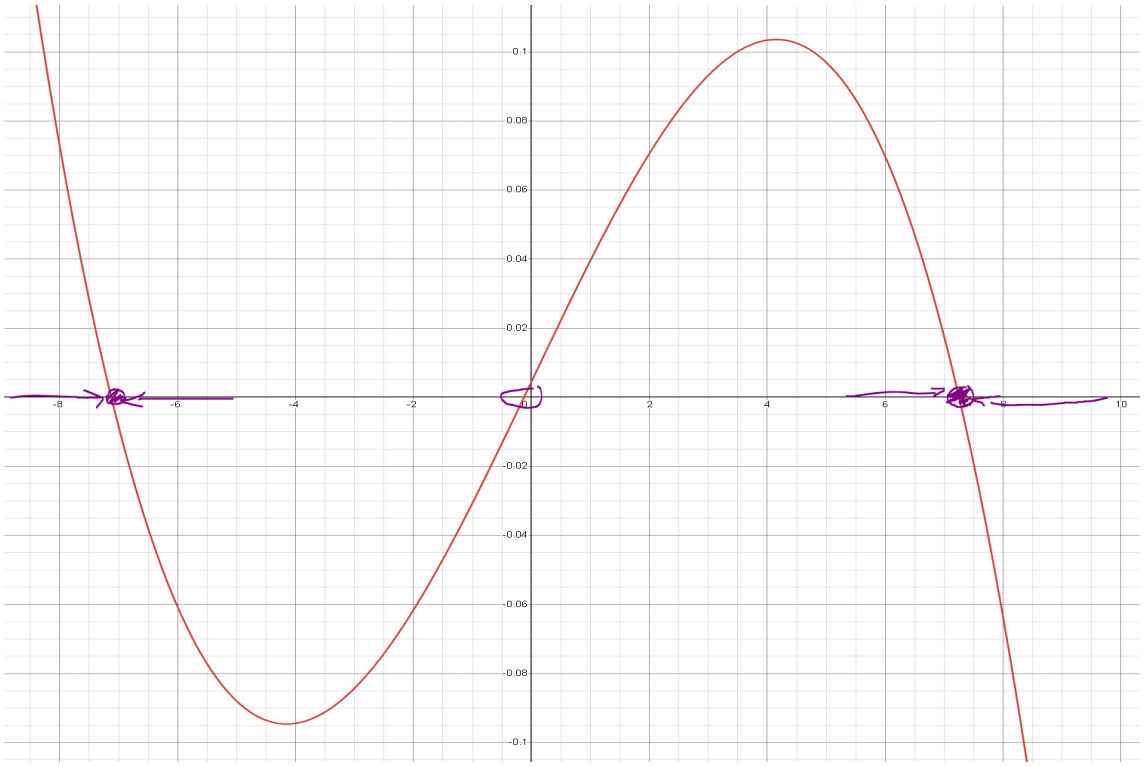


Figure 7: Drift function with MLE parameters and stability of the fixed points

Here since the control parameter either pushes the drift function up or down, two neighbour fixed points are going to collide when bifurcation happens and since the only neighbour pair of fixed points are an unstable and a stable one, a saddle-point bifurcation is going to happen. In order to find which values for β_3 will cause bifurcation we need to solving $\triangle = 0$ where $\triangle$ is the discriminant for the cubic equation that is our drift. The discriminant for a cubic equation on the form $ax^3 + b^2 + cx + d$ is given by $a^4(r_1 - r_2)^2(r_1 - r_3)^2(r_2 - r_3)^2$. The cubic equation is trivially zero at the exact moment when two roots are equal to each other, which is the same moment where two fixed points collide and bifurcation occurs as also explained above, which is why we solve for $\triangle = 0$. For ease of calculation we will be using the discriminant of the depressed cubic equation on the form $t^3 + pt + q$, which is $-4p^3 - 27q^2$, since our drift function is a depressed cubic function.

We therefore rewrite the drift function with MLE values to the form with unassigned β_3

$$-0.00069272x^3 + 0.03581118x + \beta_3 = 0 \tag{45}$$

$$x^3 + \frac{0.03581118}{-0.00069272}x + \frac{\beta_3}{-0.00069272} = 0 \tag{46}$$

$$x^3 - 51.6965x - 1443.5847\beta_3 = 0 \tag{47}$$

Yielding us discriminant

$$-4p^3 - 27q^2 = 0 \tag{48}$$

$$\Rightarrow -4(-51.6965)^3 - 27(-1443.5847\beta_3)^2 = 0 \tag{49}$$

$$\Rightarrow 4(51.6965)^3 - 27(1443.5847^2)\beta_3^2 = 0 \tag{50}$$

$$\Rightarrow 27(1443.5847^2)\beta_3^2 = 4(51.6965)^3 = 0 \tag{51}$$

$$\Rightarrow \beta_3 = \left(\frac{4(51.6965)^3}{27(1443.5847)^2}\right)^{\frac{1}{2}} \tag{52}$$

$$\Rightarrow \beta_3 = \pm 0.0991055 \tag{53}$$

Thus the bifurcation points for the control parameter β_3 occur approximately at $\beta_3 \approx \pm 0.099$. These values represent the points where the nature of the fixed points changes, indicating a transition in the double-well potential function's stability and dynamics. The resulting drift functions with control parameter β_3 as the bifurcation values are shown below

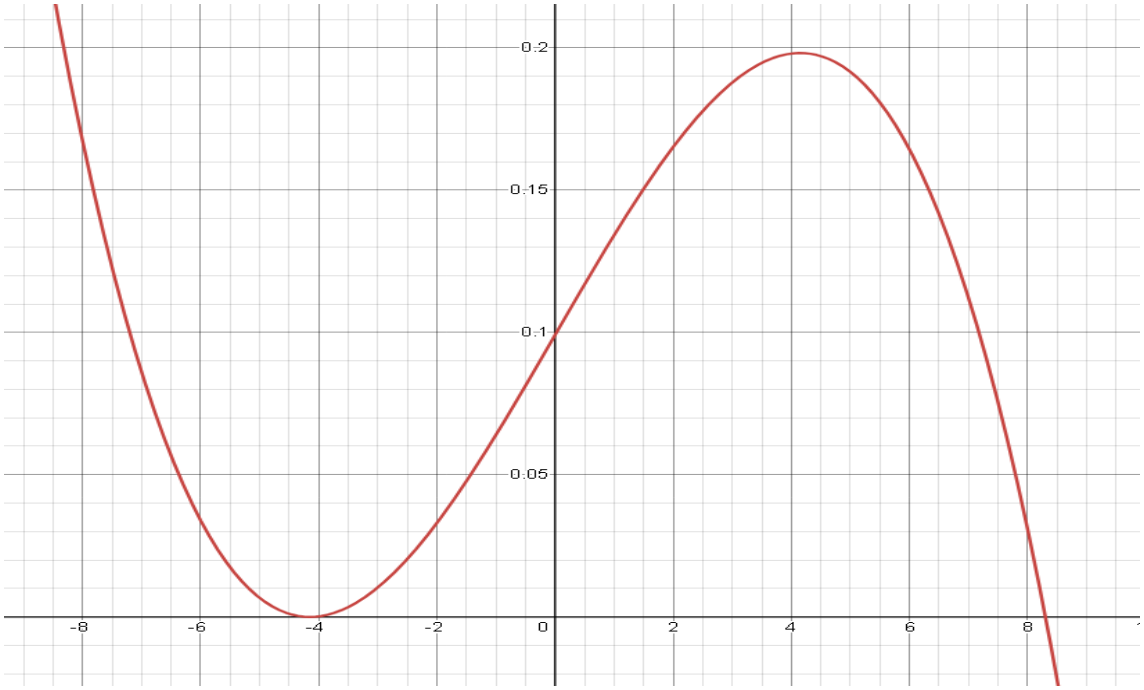


Figure 8: Drift function with MLE parameters β_1 and β_2 and $\beta_3 = 0.0991055$ to show bifurcation state

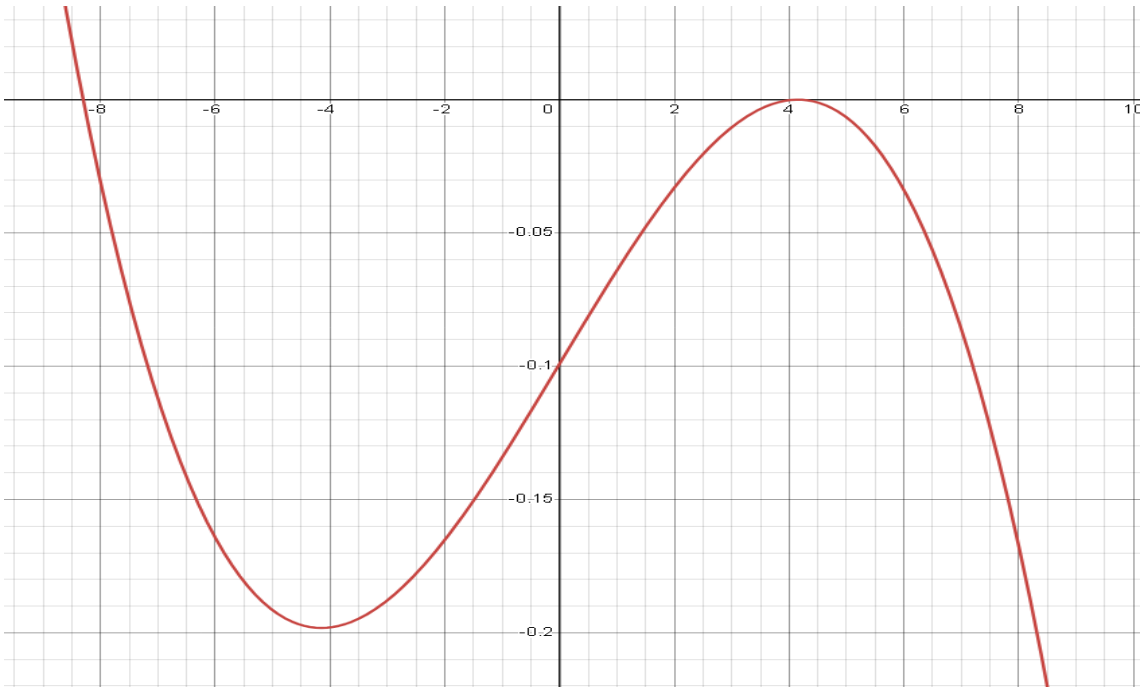


Figure 9: Drift function with MLE parameters β_1 and β_2 and $\beta_3 = -0.0991055$ to show bifurcation state

In the parameter estimation and bootstrap study we found MLE parameter for β_3 in the diffusion model to be $\hat{\beta_3} = -0.00447839$ and confidence interval to be $[-0.05756727, 0.03716470]$. Since the bifurcation points are outside the confidence interval and somewhat far from the MLE value $\hat{\beta_3}$, we can infer that the system is stable with respect to bifurcation. Since the bifurcation in the double well drift function makes the system highly sensitive to perturbations near the critical point, it is important for the stability of our model, that the MLE parameter $\hat{\beta_3}$ is not close to the bifurcation points. This leads to more numerical stable and converging simulation results.

The Bifurcation analysis above was only done for the simple diffusion model in (9). The analysis for the jump-diffusion in (10) is practically the same. In the jump-diffusion model, we would get fixed points $X \approx \{-7.05191, -0.0382152, 7.09013\}$ with bifurcation points $X_{\text{bifurcation}} = \pm 0.0955280$ which is approximately the same as above. Thus the same conclusion can be drawn for the jump-diffusion model as we did for the simpler diffusion model. This is again because we saw that for the MLE values, the jump-diffusion model practically reduced down to become the diffusion model.

8 Result

After performing the parameter estimation for both models I ended up with

$$dX_t = (-0.00069272X_t^3 + 0.03581118X_t + 0.00447839)dt + 0.55070983dW_t \tag{54}$$

For the diffusion model and for the jump-diffusion model I got

$$dX_t = (-0.0007019743X_t^3 + 0.035099X_t + 0.001341275)dt + 0.550687dW_t + e^{\frac{-60.565-X_t}{6.01859}}dN_t \tag{55}$$

With rate parameter $59.9946 * 0.05 = 2.99973$ for the Poisson point process.

Next I simulate data from both these models using the Euler-Maruyama scheme from (12) and (14). When simulating data i use a subsampling technique with a factor 10, meaning that i decrease

the step size from 0.05 to 0.005 and only plot every 10'th data point to attempt to combat the discretizing bias in the Euler-Maruyama scheme as explained earlier. When simulating data with the MLE parameters for both models we get the paths shown below

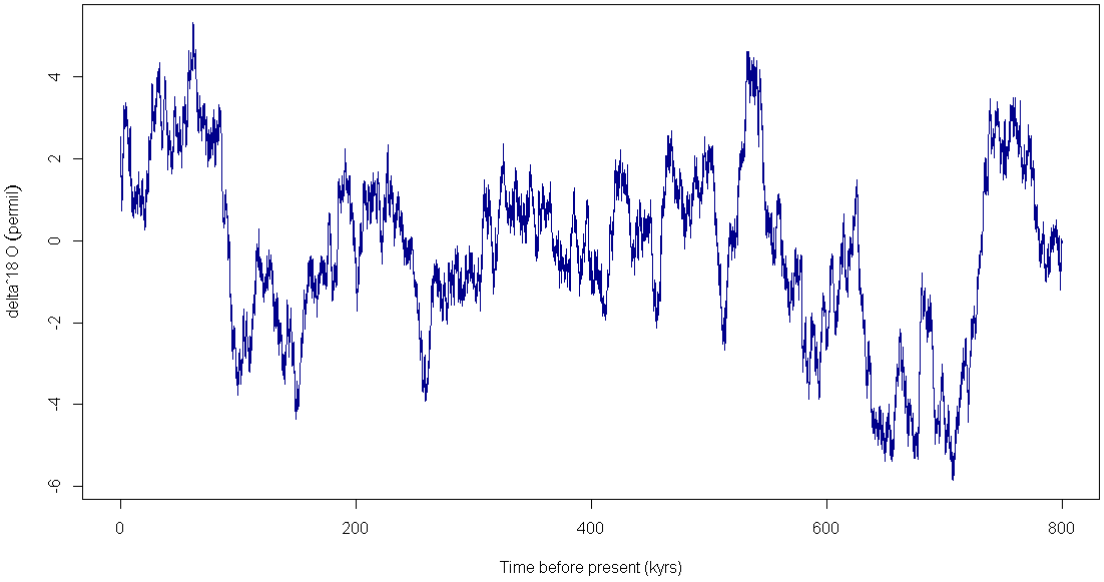


Figure 10: Data path simulated with diffusion model (9)

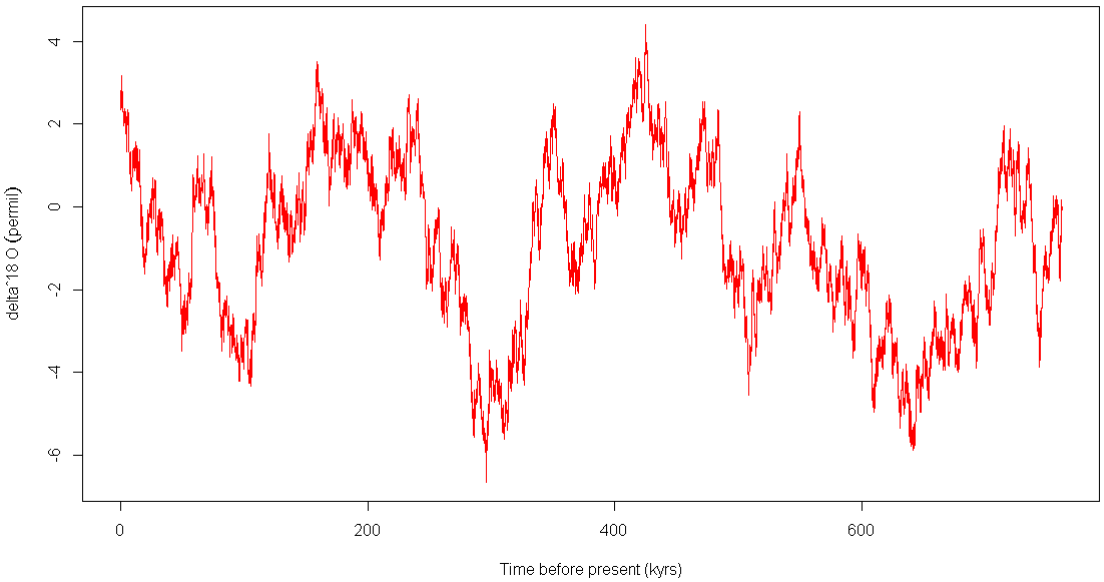


Figure 11: Data path simulated with jump-diffusion model (10)

Since the jumps from the jump-diffusion model are almost non distinguishable from the variance caused by the wiener process the two realisations of the Euler-Maruyama scheme will produce very similar paths. Even though when looking at the figure 6 and 7 they emulate the observed data well, they do not seem to capture the one sided jumps very well. This is not surprising when

observing the realised path by the diffusion model as it should not be able to transition with such sharp jumps as the real data does. Since the parameter estimation in the jump diffusion model resulted in extremely small jumps and very large confidence intervals for the $\hat{\gamma}$ and $\hat{x}^*$ parameters, perhaps the extra jump term was not a very good method for modelling the sharper transitions.

Below we will be comparing the densities for the simulated data from the Euler-Maruyama scheme with both models. I will be comparing the densities in order to validate the accuracy and reliability of the models.

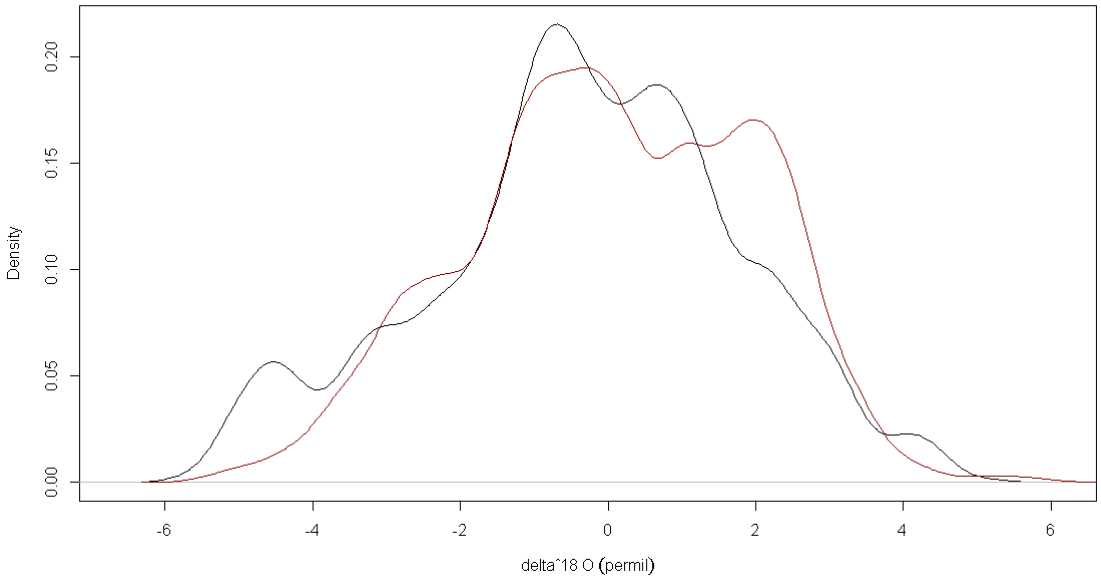


Figure 12: Densities of the EM simulated data from the diffusion model (black) and the observed data (red)

In figure 8 it is evident that the diffusion model has captured the underlying processes and dynamics of the data effectively. Since the process I am simulating data with is stochastic, the simulated density (black) will change for each time I simulate new data. Therefore in order to again reduce the variance in the simulation I used the sub sampling technique, which leads to a more robust and reliable comparison between the simulated and observed data. Further decreasing the step size dt, would increase accuracy of this comparison. Additionally I created a QQ-plot of the simulated data versus the observed data, to get a better visualisation of the accuracy of the simulated data distribution of the observed data distribution.

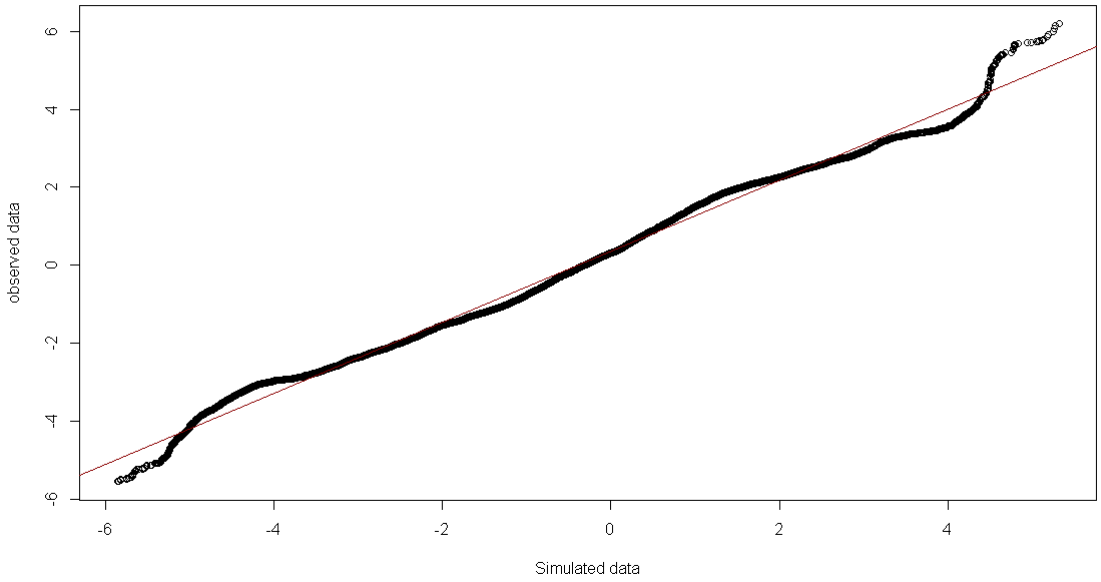


Figure 13: Densities of the EM simulated data from the diffusion model (black) and the observed data (red)

The QQ-plot above in figure (9) confirms that the diffusion model is a well fitting model for the ice core data. since the quantiles are very equal for the simulated data and the observed data.

Below we perform the same distribution comparison and QQ plot analysis

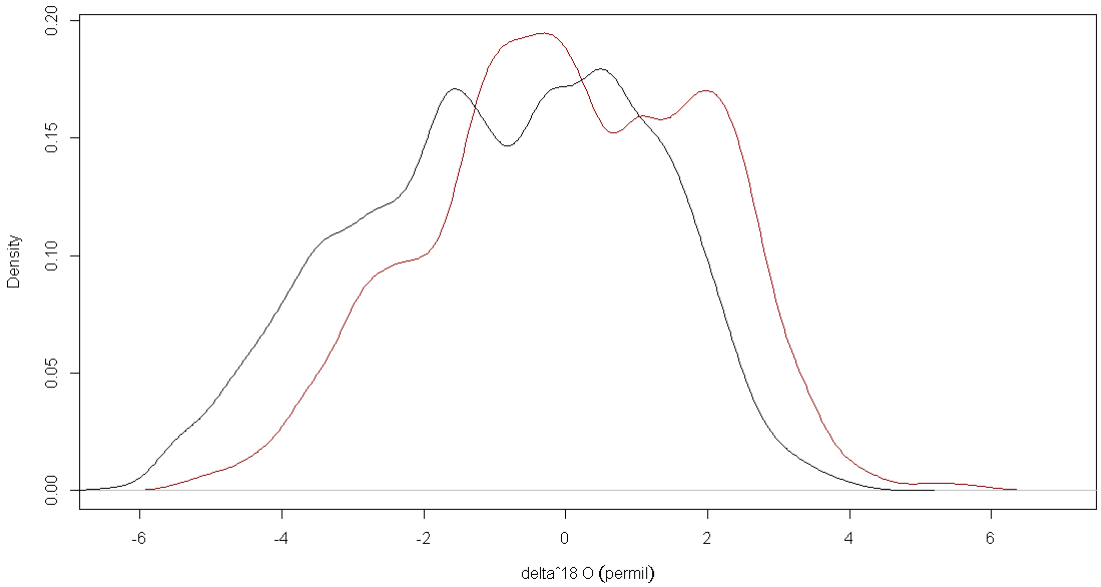


Figure 14: Densities of the EM simulated data from the jump-diffusion model (black) and the observed data (red)

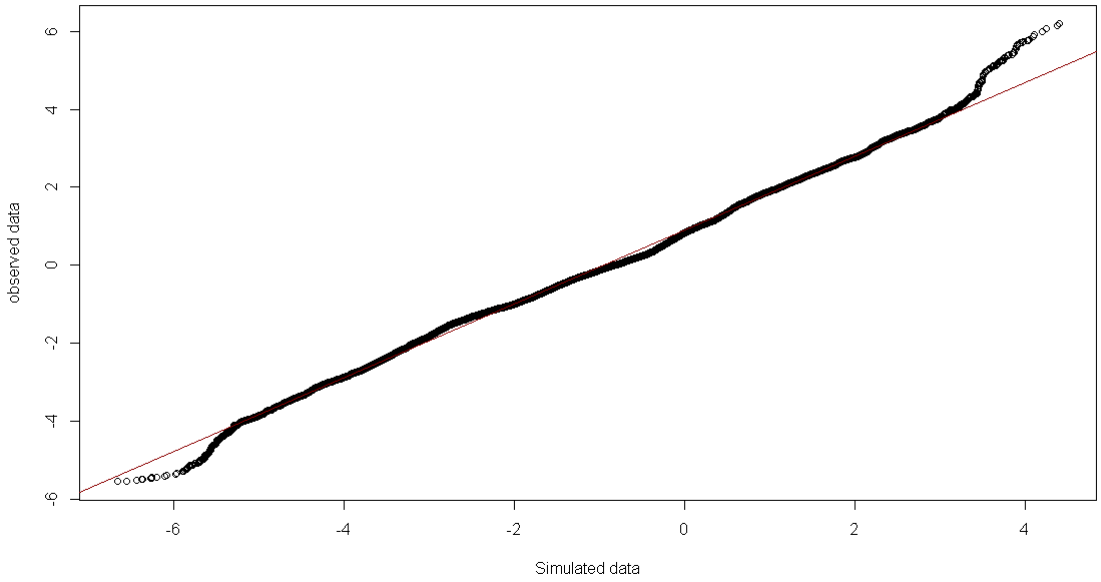


Figure 15: QQplot of the simulated data with the jump diffusion model vs. the observed data

Here I conclude the same as in the distribution analysis for the diffusion model, which again confirms that the jump diffusion model simply reduced to the diffusion model.

When adding a jump term to the model we essentially give the model another term to try and capture the characteristics of the data. In this case the parameter estimation found minimum in the negative log-likelihood function with parameters that undermined the jumps. This indicates that letting the Poisson process contribute to modelling the data's movement instead of only the wiener process did not increase the overall performance of the model whilst increasing the complexity of it. Therefore not providing the intended ability to capture the jumps in the data and in conclusion failing to work as a better model for predicting the climate variability that was intended.

The alternative method I used for finding convergence in the optimizing algorithm for the negative log-likelihood function, is also an indicator towards the models trouble with finding a good set of parameters for the model. When attempting to minimize the negative log-likelihood function using both "Nelder-Mead" and "L-BFGS-B", I get different results based on which initial parameters i use. I attempted with many different initial parameters, which resulted in either non optimizer-converging parameter values, errors or as seen in the parameter estimation section, values that are too extreme and are not numerically stable. Thus leading to the conclusion that the negative log-likelihood function is very unstable under numerical optimization and not suited for inaccurate initial parameters.

9 Discussion

In this study the main objective was to fit a double-well potential model to the ice core data and a double-well potential model with a jump term. I found for the model without the jump term a well fitting set of MLE parameters. For the model with jump term I found the model had been reduced to the simpler model without a jump, since the jumps were extremely small. Additionally, I experienced difficulties in achieving convergence for the jump diffusion model when optimizing the negative log-likelihood. This suggests that the model is inherently more unstable and challenging

to fit to the data.

The results from this paper points to a conclusion that the additional jump term that we added to the double-well potential model, is not a well fitting solution to model the jumps we observe in the data. It is apparent from the MLE parameters that neglecting the jump term gives the best results. This may be a problem stemming from the fact that we let the jumps range from 0 and ∞, since it may be hard for the model to distinguish which movement should be modelled by the wiener process and what movement should be modelled by the Poisson point process. Even though the additional jump term did not provide the intended ability to model the sharp state transitions we saw in the data, the simpler model without the jump term still provides us with a robust model for predicting the climate variability. As we see in the results section, the simulated data captures the underlying distribution of the data well. Moreover the double-well potential model works well for modelling systems that exhibit bistability. In the observed data, we see a bistability between the two predominant climate states, the glacial (cold) and interglacial (warm) periods, representing the large and low data values respectively.

As the primary problem for the jump-diffusion model was distinguishing which movement should be modelled by the wiener process or the Poisson point process, as a consequence of letting the jumps be of sizes $(0, \infty)$. Since the jumps were able to be close to 0, the small movement in the data also got modelled by the Poisson point process and not only the Wiener process. For further studies of the observed data, using models where the jump terms are constant and not close to 0 could be an alternative model in order to not have the Poisson point process and the Wiener process confused. A way to accomplish this, would be to use the optimizing algorithm "L-BFGS-B" with bounds on the parameters, and choose some bound for the constant jump that makes sure the model does not confuse the Poisson point process with the Wiener process. An example of such bounds would be, such that the Jump term is in the interval $[1, \infty)$ so the jumps are far away from zero and thus distinguishable from the draws from the Wiener process. Another idea to find a better model would be to increase the number of wells in the function, this might enhance the modelling ability and with a jump function decrease the probability of divergence in the data simulation. Lastly a Stochastic Hybrid Systems that on the form $dX_t = f_i(X_t)dt + \sigma dW_t$, that lets you incorporate random switching between different deterministic dynamics in order gain more control of the different states of the model.

The jump-diffusion model is somewhat unstable when having the double-well potential function as the drift and the chosen jump function. The drift- and jump-function being locally Lipschitz continuous means that within any bounded region, there exists a Lipschitz constant that can guarantee convergence. Therefore, as long as the solution stays within a bounded region, the Euler-Maruyama scheme can provide a good approximation of the true solution. Though since the drift function is not globally Lipschitz continuous, there is no guarantee that the Euler-Maruyama scheme will converge for all initial conditions and for arbitrarily large time intervals. This lack of global Lipschitz continuity can, when large values of the state variable X_t occur, cause the EM solution to grow rapidly, leading to numerical instability and divergence. Additionally it can lead to regions where the drift function's growth is not controlled, thus the numerical solution becomes unbounded and potentially blow-up the numerical scheme. This stems from the fact that without the Lipschitz condition, we cannot prove the weak and strong order convergence for the Euler-Maruyama scheme[7][12][13]. Without convergence guarantees, the numerical solution might indicate spurious transitions or fail to capture actual transitions between states. This would lead to incorrect interpretations and predictions about the paleoclimate's behavior. In order to mitigate this as mentioned above, using functions that are globally Lipschitz or at least a globally Lipschitz jump function might prove to create a more robust and numerically stable EM-simulation and a

model where the MLE parameters will not end up neglecting the jump term.

Lastly we had bifurcation points that were far away from the MLE control parameter β_3. This helps with ensuring the numerical stability and convergence of the models. This will be an important property to have when studying the jump-diffusion model with other jump functions. Because if we choose a jump function that results in a model, with MLE parameters for the drift function that are close to the bifurcation values, the resulting model will have an increased sensitivity to perturbations in addition to the loss of stability, meaning that stable points can become unstable and vice versa. This can lead to sudden transitions between different states of the system, often referred to as "tipping points". The Euler-Maruyama scheme will also become numerically more unstable and more often experience divergence.

10 Conclusion

In conclusion the double-well potential model is a well-fitting model to use when modelling paleo-climate data. It managed to capture the bistability of the data well as we saw in the bifurcation where the parameters showed stable wells and showed great predictive ability using the Euler-Maruyama scheme to simulate data. Also the models parameters had tight confidence intervals showing the stability and significance of the optimizing algorithm and the parameters respectively. In my attempt to increase the accuracy and predictive ability of the double-well potential model I deployed a jump term and analyzed the expanded jump-diffusion model. I succesfully found MLE parameters using the Euler-Maruyama induced approximation of the transition density, which led to a neglection of the intended large jump and resulted in small relatively frequent jumps. Thus helping modelling the small movement that was intended for the Wiener process and not the large jump that the process was intended for. I dove into the reason for this together with the instability of the optimizing algorithm and concluded that because of the global Lipschitz condition not holding for both the drift function and the jump function. The double-well potential model is together with the jump term, not a very stable model, since the unstable states of the drift function is easier transitioned to by the jumps and thus divergence is often seen when simulating data with the Euler-Maruyama scheme. Lastly I discussed other models that would perhaps better capture the sharper transitions of the data. Here i recommended globally Lipschitz functions for the jump function if not both the jump- and drift-functions for further studying of this subject.

References

[1] Stephen Barker et al. ,800,000 Years of Abrupt Climate Variability. Science334,347-351(2011).DOI:10.1126/science.1203580

[2] James F. Kasting, M. Tazewell Howard, Klaus Wallmann, Ján Veizer, Graham Shields, Jasmine Jaffrés, Paleoclimates, ocean depth, and the oxygen isotopic composition of seawater, Earth and Planetary Science Letters,2006, https://doi.org/10.1016/j.epsl.2006.09.029.

[3] Kloeden PE, Platen E, Schurz H. The numerical solution of nonlinear stochastic dynamical systems: a brief introduction. Int J Bifur Chaos 1991;1:277±86.

[4] Rossi, Richard J. (2018). Mathematical Statistics: An Introduction to Likelihood Based Inference. New York: John Wiley & Sons. p. 227. ISBN 978-1-118-77104-4.

[5] Picchini, U. & Ditlevsen, S. (2011). Practicle estimation of high dimensional stochastic differential mixed-effects models. Comput. Statist. Data. Anal. 55, 1426–1444.

[6] Nelder, John A.; R. Mead (1965). "A simplex method for function minimization". Computer Journal. 7 (4): 308–313. doi:10.1093/comjnl/7.4.308

[7] Kloeden, P. E., & Platen, E. (1992). Numerical solution of stochastic differential equations. Springer.

[8] Durrett, Rick (2019). "Brownian Motion". Probability: Theory and Examples (5th ed.). Cambridge University Press. ISBN 9781108591034.

[9] BALL, C., A. & TOROUS W. N. (1983) A Simplified Jump Process for Common Stock Returns. Journal of Financial and Quantitative Analysis, Vol 18, pp 53 - 65.

[10] JORION, P. (1988) On Jump Processes in the Foreign Exchange and Stock Markets. The Review of Financial Studies Vol. 1, (4.), pp 427 - 445.

[11] Searcóid, Mícheál Ó (2006), "Lipschitz Functions", Metric Spaces, Springer undergraduate mathematics series, Berlin, New York: Springer-Verlag, ISBN 978-1-84628-369-7

[12] Vaes, Urbain. Lecture notes (In link)
chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://urbain.vaes.uk/static/teaching/lectures/build/lectures-w6.pdf

[13] Hutzenthaler Martin, Jentzen Arnulf and Kloeden Peter E. 2011Strong and weak divergence in finite time of Euler's method for stochastic differential equations with non-globally Lipschitz continuous coefficientsProc. R. Soc. A.4671563–1576 http://doi.org/10.1098/rspa.2010.0348

[14] Pratt, Orson (1866). New and Easy Method of Solution of the Cubic and Biquadratic Equations: Embracing Several New Formulas, Greatly Simplifying this Department of Mathematical Science. Longmans, Green, Reader, and Dyer. p. 13. ISBN 9781974130924. ...if two roots are imaginary, the product is positive...

11 Appendix

```
rm(list = ls())

library(ggplot2)
library(dplyr)
library(tidyr)
library(zoo)
library(purrr)
library(knitr)
library(kableExtra)
library(tinytex)
library(tidyverse)
library(stats4)
set.seed(123)

# Data load ind
skip = 5

#data files
df <- read.csv("data/Syn_Greenland.csv",sep=",",skip = skip)
df_in <- read.csv("data/China_cave.csv",sep=",",skip = skip)

# Renaming column names for ease of reference
df <- rename(df, age = EDC3.Age..kyr., d18O =  GLT_syn. 18O ....)
df_in <- rename(df_in, age = Age..ka.BP., d18O =  18O .carb....VPDB.)

# Data processing for duplicated data
preprocess_data <- function(data) {

  return_df <- data %>%

    mutate(rolling_mean = rollmean(d18O, k = 625, fill = NA, align = "right
        ")) %>%

    mutate(d18O = d18O - rolling_mean) %>%

    select(-rolling_mean) %>%

    arrange(desc(age))

  return(data.frame(return_df))
}
# Data processing
df_pro <- drop_na(preprocess_data(df))

# Plotting the data after and before data processing
ggplot(data = df, aes(x = age, y = d18O)) +
  geom_line() +
  xlab("Time␣before␣present␣(kyrs)") +
  ylab(expression(delta^{18} * O ~ (permil)))
```

```r
ggplot(data = df_pro, aes(x = age, y = d18O)) +
  geom_line() +
  xlab("Time before present (kyrs)") +
  ylab(expression(delta 18 ~ O ~ (permil)))


# Define the SDE parameters and functions
F_drift <- function(X, beta1,beta2,beta4){
  return(beta4*X^3 - beta2*X - beta1)
}

# visualize drift function
plot(seq(-2,2,0.01),F_drift(seq(-2,2,0.01), 1,  1, 0.5), type='l')

F_jump <- function(x, x_val, alpha) {
  jump <- 1 + exp((x_val-x)/alpha)
  return(jump)
}

# Parameters
dt <- 0.05
T_start <- df_pro[length(df_pro[,1]), 1]
T_end <- df_pro[1, 1]
x0 <- 0
num_steps <- (T_end - T_start) / 0.05
ylim <- c(-6, 6)

# y-axis limits for plotting
T_start <- df_pro[length(df_pro[,1]),1]
T_end <- df_pro[1,1]
x0=0

# Euler-Maruyama negative log likelihood for jump-diffusion model
EM_nll_jump <- function(theta, X, dt) {
  N <- length(X) - 1
  beta1<-theta[1]
  beta2<-theta[2]
  beta4<-theta[3]
  sigma<-theta[4]
  lambda <- theta[5]
  alpha <- theta[6]
  x_val <- theta[7]
  # Initialize negative log likelihood
  nll <- 0

  for (n in 1:N) {
    # Drift function
    F <- F_drift(X[n], beta1, beta2, beta4)
    alpha_1 <- F_jump(X[n],x_val,alpha)

    # Diffusion matrix
    SigmaSigma <- sigma^2
```

```
    # Update negative log likelihood

    for (i in 1:10){
      nll <- nll + 0.5 * log(2 * pi * SigmaSigma * dt) +
        0.5 * (X[n + 1] - X[n] - F * dt - alpha_1 * i) *
        1/(SigmaSigma * dt) * (X[n + 1] - X[n] - F * dt - alpha_1 * i) -
        log((lambda * dt)^(i)/factorial(i)*exp(-lambda * dt))
    }
  }
  return(nll)
}

# optimizing algorithm for finding MLE parameters
#                            beta1,        beta2,       beta3,      sigma,
    lambda,gamma,  x
initial_beta_jump_bfsg <- c(  0  .00447839, 0.03581118,   0  .00069272,
    0.55070983, 1, 2, 1.8)

result_EM_1_jump_bfsg <- optim(par = initial_beta_jump_bfsg,
                           fn = EM_nll_jump,
                           X = df_pro[,2],
                           dt = 0.05,
                           control=list(maxit=1000),
                           method='L-BFGS-B',
                           lower=c(-5,-5,-5,0.0001,0.0001,-5,-5),
                           upper=c(5,5,5,5,5,5,5))


initial_beta_jump <- result_EM_1_jump_bfsg$par
(result_EM_1_jump <- optim(par = initial_beta_jump_bfsg,
                           fn = EM_nll_jump,
                           X = df_pro[,2],
                           dt = 0.05,
                           control=list(maxit=1000)))

result_jump <- result_EM_1_jump$par

# defining T, dt X_0 and iterations for bootstrap analysis
T <- length(df_pro[,2])*0.05
dt <- 0.05
x0 <- 0
iterations <- 100

# 1D EM-simulation of data
simulate_EM_jump <- function(T, dt, b1, b2, b4, sigma, lambda, alpha, x_val
    , x0) {
  t <- seq(0, T, by = dt)
  n <- length(t) - 1
  dW <- rnorm(n) * sqrt(dt)  # Correct Wiener increment
  dN <- rpois(n, lambda * dt)  # Poisson increment
  X <- numeric(n + 1)
  X[1] <- x0
```

```
  jump <- 0
  for (i in 2:(n + 1)) {
    drift_term <- F_drift(X[i-1], b1, b2, b4)
    jump_term <- F_jump(X[i-1], x_val, alpha)
    jump <- jump + jump_term
    X[i] <- X[i-1] + drift_term * dt + sigma * dW[i-1] + dN[i-1] * jump_
        term
  }
  return(data.frame(t = T - t, X = X))  # Adjust time vector for correct
      plotting
}

# plot of simulated data with MLE parameters and subsampling
a <- simulate_EM_jump(T, dt/10, result_jump[1], result_jump[2],result_jump
    [3],
                  result_jump[4], result_jump[5], result_jump[6], result_
                      jump[7],x0)

plot(a[seq(1, nrow(a), 10), ]$t, a[seq(1, nrow(a), 10), ]$X,type='l', col='
    red',
      xlab = "Time␣before␣present␣(kyrs)",ylab = expression(delta 18 ~ O ~
          (permil)))

#plot of observed data
plot(df_pro[,1], df_pro[,2], type='l')

# Euler-Maruyama negative log likelihood for diffusion model
EM_nll <- function(theta, X, dt) {
  N <- length(X) - 1
  beta1<-theta[1]
  beta2<-theta[2]
  beta4<-theta[3]
  sigma<-theta[4]

  # Initialize negative log likelihood
  nll <- 0

  for (n in 1:N) {
    # Drift function
    F <- F_drift(X[n], beta1, beta2, beta4)

    # Diffusion matrix
    SigmaSigma <- sigma^2

    # Update negative log likelihood
    nll <- nll + 0.5 * log(SigmaSigma * dt) +
      0.5 * (X[n + 1] - X[n] - F * dt) *
      1/(SigmaSigma * dt) * (X[n + 1] - X[n] - F * dt)
  }
  return(nll)
}

# optimizing for MLE parameters in diffusion model
```

```
initial_beta <- c(1,1,1,1)
(result_EM_1 <- optim(par = initial_beta, fn = EM_nll,
                      X = df_pro[,2], dt = 0.05))
result <- result_EM_1$par


# simulating data with the MLE parameters
simulate_EM <- function(T, dt, b1, b2, b4, sigma, x0) {
  t <- seq(0, T, by = dt)
  n<-length(t)-1
  dW <- rnorm(n)
  X <- numeric(n + 1)
  X[1] <- x0
  for (i in 2:(n + 1)) {
    drift_term <- F_drift(X[i-1], b1, b2, b4)
    X[i] <- X[i-1] + drift_term * dt + sigma * dW[i-1] * sqrt(dt)
  }
  return(data.frame(t = T-t, X = X))
}
# Subsampling
step <- simulate_EM(800, dt/10, result[1], result[2],
                                result[3],result[4], x0)

# taking every 10'th value since we produces 10 times as many data points
    per subsampling
result_eul_11d <- step[seq(1, nrow(step), 10), ]

# plot of simulated data versus observed data
plot(df_pro[,1],df_pro[,2], type='l', col='black')
lines(result_eul_11d, type='l', col='darkblue')

# plot of simulated data formatted nicely
plot(result_eul_11d, type='l', col='darkblue', xlab = "Time␣before␣present␣
    (kyrs)",ylab = expression(delta 18 ~ O ~ (permil)))

# Density plots of both models versus observed data
plot(density(df_pro[,2]), col='darkred', main="", xlab=expression(delta 18
     ~ O ~ (permil)))
lines(density(a[seq(1, nrow(a), 10), ]$X), main="")

plot(density(result_eul_11d[1:length(df_pro[,2]),2]),main="", xlab=
    expression(delta 18 ~ O ~ (permil)))
lines(density(df_pro[,2]), col='darkred')


# qqplot of simulated data versus observed data for both models
qqplot(a[seq(1, nrow(a), 10), ]$X,df_pro[,2], xlab = "Simulated␣data", ylab
     = "observed␣data")

abline(lm(sort(df_pro[,2]) ~ sort(a[seq(0, nrow(a), 10), ]$X)), col = "
    darkred")

qqplot(result_eul_11d[1:length(df_pro[,2]),2],df_pro[,2],
```

```
       xlab = "Simulated data", ylab = "observed data")

abline(lm(sort(df_pro[,2]) ~ sort(result_eul_11d[1:length(df_pro[,2]),2])),
     col = "darkred")


# Bootstrap analysis of the MLE parameters
boot_params <- function(iter, init_params, mult){
  boot_res <- matrix(0, iter, length(init_params)+1)
  boot_res[1,] <- c(init_params,0)

  for (i in 2:iter){
    data <- simulate_EM(800, dt/mult, boot_res[1,1], boot_res[1,2],
                        boot_res[1,3],boot_res[1,4], x0)

    optim_data <- data[seq(1, nrow(data), 10), ]

    res <- optim(par = boot_res[1,1:4], fn = EM_nll,
                 X = optim_data$X, dt = 0.05)

    boot_res[i,] <- c(res$par, res$convergence)
  }
  return(boot_res)
}

Boot <- boot_params(100, result, 10)

boot_params_jump <- function(iter, init_params, mult){
  boot_res <- matrix(0, iter, length(init_params)+1)
  boot_res[1,] <- c(init_params,0)

  for (i in 2:iter){
    data <- simulate_EM_jump(800, dt/mult, boot_res[1,1], boot_res[1,2],
                        boot_res[1,3],boot_res[1,4],
                        boot_res[1,5],boot_res[1,6],
                        boot_res[1,7], x0)

    optim_data <- data[seq(1, nrow(data), 10), ]

    res <- optim(par = boot_res[1,1:7], fn = EM_nll_jump,
                 X = optim_data$X, dt = 0.05)

    boot_res[i,] <- c(res$par, res$convergence)
  }
  return(boot_res)
}
Boot_jump <- boot_params_jump(100, result_jump, 10)
Boot_jump

quantile(Boot_jump[,2], c(0.025, 0.975))
quant_jump <- matrix(0,2,7)
for (i in 1:7){
  quant_jump[,i] <- quantile(Boot_jump[,i], c(0.025, 0.975))
```

```
}

quantile(Boot[,2], c(0.025, 0.975))
quant <- matrix(0,2,4)
for (i in 1:4){
  quant[,i] <- quantile(Boot[,i], c(0.025, 0.975))
}

par(mfrow=c(2,4))

plot(density(Boot_jump[,1]), main='beta1')
plot(density(Boot_jump[,2]), main='beta2')
plot(density(Boot_jump[,3]), main='beta3')
plot(density(Boot_jump[,4]), main='sigma')
plot(density(Boot_jump[,5]), main='lambda')
plot(density(Boot_jump[,6]), main='gamma')
plot(density(Boot_jump[,7]), main='x*')

# transition Densities calculated for each datapoint but not visualized in
    qq plot since i could not figure out how to # compare to convolution of
     Poisson- and normal-distribution
trans_dens_jump <- function(X,dt,theta){
  n <- length(X)-1
  beta1<-theta[1]
  beta2<-theta[2]
  beta4<-theta[3]
  sigma<-theta[4]
  lambda <- theta[5]
  alpha <- theta[6]
  x_val <- theta[7]


  jump <- F_jump(X, x_val, alpha)
  drift <- F_drift(X, beta1,beta2,beta4)

  mean_val <- X + unlist(lapply(drift, function(df) df * dt))

  jump_mean <- unlist(lapply(jump, function(df) df*c(0:10)))

  var_val <- sigma^2*dt

  density_vals <- matrix(0,nrow = length(X), ncol = 11)
  for (i in 0:10){
    density_vals[,i+1] <- pnorm(q = X, mean = mean_val + jump_mean[seq(1+i,
        length(jump_mean), 11)],
                          sd = sqrt(var_val)) * ppois(q = i, lambda * dt)

  }
  return(rowSums(density_vals))
}

# calculating transition densities
dens <- trans_dens_jump(df_pro[,2], dt, result_jump)
```

```r
# Plotting density of transition densities
plot(density(dens))


# calculating the accuracy lost by truncating the likelihood of more then
    10 jumps in the MLE for the jump diffusion model with lambda = 59.99457
Pois_trunc <- 1-sum(dpois(0:10,59.99457*0.05))
print(Pois_trunc)
```