Modeling of Stochastic Differential Equations in Combination with Autoencoders

Jannis Klingler

Masterarbeit

Zur Erlangung des akademischen Grades
Master of Science

vorgelegt am 12. Oktober 2022

Albert-Ludwigs-Universität Freiburg

Institut für Mathematik und Physik
Betreuung: Prof. Dr. Harald Binder

Vorgelegt von:	Jannis Klingler
	Sundgauallee 42
	79110 Freiburg im Breisgau
	jannis-klingler@web.de
Matrikelnummer:	4331982
Studiengang:	M. Sc. Mathematik
Bearbeitungszeitraum:	12.04.2022 bis 12.10.2022

Erklärung zur Masterarbeit

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig verfasst wurde und dass keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden.
Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen anderer Autoren entsprechen, wurden gekennzeichnet.
Diese Erklärung bezieht sich auch auf in der Arbeit enthaltene Grafiken und bildliche Darstellungen.
Darüber hinaus versichere ich, dass diese Arbeit nicht und auch nicht auszugsweise bereits für eine andere Prüfung angefertigt wurde.

Datum, Ort	Unterschrift

Contents

Introduction

Medical time series data, as well as clinical cohort studies, often exhibit certain stochasticity over time. Therefore, it is useful to model such data in a way that takes this stochasticity into account. However, the study of data including individual development patterns is complicated by the often highly irregular time structure of the data. An example is the *SMArtCARE* dataset containing observations of the motor skills of patients suffering from spinal muscular atrophy. The given data shows that patients' motor skills do not further improve after diagnosis. Moreover, the measured values are present only at specific points in time with individual frequency for each patient. Since the development of a patient's condition is uncertain, stochasticity must be taken into account. For a general analysis, it is then possible to group patients based on age and severity of their symptoms. Additionally, on an individual level it is possible to determine certain baseline variables that characterize the development of a patient's condition.

This thesis is motivated by the analysis of data sets such as SMArtCARE. The aim of this analysis is to learn underlying individual development patterns of the data. Using the additional baseline variables, the group structure of the individuals can be learned and the individual development of the time series can be characterized. Since medical data sets are often extensive, machine learning tools like an autoencoder can be used to reduce the dimension of the given data and to obtain a lower-dimensional representation of the initially high-dimensional trajectories. This new data can be examined with less effort.

The work of Chen [CRBD18] and Yildiz [YHL19] showed that in the analysis of data exhibiting latent development patterns it is useful to assume that there exists an underlying dynamical system of the data which can be modeled by an ordinary differential equation. Hackenberg [HHP$^+$22] demonstrated that dynamic modeling can be useful even in an extreme situation in which only baseline characterizations and one other observation point are available. This thesis builds in parts on Hackenberg's work and attempts to incorporate stochasticity modeling in the analysis of sparse data. We assume that latent evolution can be described in terms of a stochastic differential equation and focus particularly on Ornstein-Uhlenbeck processes.

In the first chapter of this thesis we will discuss the theoretical background which is necessary to formulate the applied methods. We will consider neural networks and examine under which conditions they can be used as universal function approximators and how to iteravely update them until we reach a certain accuracy. Furthermore, we will derive the stochastic integral for continuous processes in order to discuss stochastic differential equations. Finally, we will define the Ornstein-Uhlenbeck process and consider its properties. In the second chapter the focus will be on the methods that are later used for the analysis of the given data. In the setting of this thesis, we introduce three methods for this analysis. The knowledge gained from the application of each of these methods is incorporated in the subsequent methods. The Euler-Maruyama method aims to approximate the latent underlying process using the Euler-Maruyama approximation for stochastic differential equations. The covariance loss method splits the learning of drift and diffusion parameters of the process into different loss functions. The third and main method of this thesis is the Fokker-Planck method. In this method we exploit properties of the Ornstein-Uhlenbeck process to obtain the parameters of the underlying Ornstein-Uhlenbeck process using the baseline variables. The goal is to learn the empirical distribution of the data. After having developed these methods, they will be tested on a suitable simulation design in the third chapter. For this purpose we will generate a data set which is kept as simple as possible but still satisfies the requirements mentioned before. This data set is inspired by the data of the SMArtCARE database.

1 Background

This chapter focuses on the mathematical background for the methods that will be developed in chapter 2. We first discuss the machine learning concepts that we will use. For this purpose, we introduce neural networks and define autoencoders. We show under what conditions arbitrary functions can be approximated using neural networks and go into more detail about the training of these. Second, we give a short introduction of the stochastic integral for continuous processes in order to be able to define stochastic differential equations. The goal is to define the Ornstein-Uhlenbeck process which is relevant for the methods that will be developed in chapter 2. We show that an Ornstein-Uhlenbeck process exists and that it is unique. Further, we show that it is a time-homogeneous normally distributed Markov process. In order to compute the distribution of the Ornstein-Uhlenbeck process we choose an alternative way using the Fokker-Planck equation compared to the conventional computation of its distribution.

1.1 Machine Learning

In this section we introduce neural networks which we will use mainly to reduce the dimension of the data. Our approach belongs to unsupervised learning, a subfield of machine learning. Furthermore, we see that the realizations of neural networks are universal approximators under certain conditions and are thus a particularly suitable tool for our work. We also consider neural network training and have a closer look at two optimization algorithms.

1.1.1 Neural Networks

Given a data set, the aim of machine learning could be to find a lower-dimensional representation of the data which preserves the structure of the data as best as possible. In order to achieve this, we are looking for a function that maps the original data into a lower dimensional space. We realize this with a neural network based approach.

Definition 1.1 (Neural network)**.** [1] Let $D, d, L \in \mathbb{N}$ and $N_1, \ldots, N_{L-1} \in \mathbb{N}$. A *neural network* with input dimension D, output dimension d and L layers is a sequence of matrix-vector tuples

$$\Phi = ((A_1, b_1), (A_2, b_2), \ldots, (A_L, b_L)),$$

where for all $l = 1, \ldots, L$ the matrices $A_l \in \mathbb{R}^{N_l \times N_{l-1}}$ are called *weight matrices* and the vectors $b_l \in \mathbb{R}^{N_l}$ are called the *bias vectors* of the neural network Φ. The numbers $N_0 := D$, $N_1, \ldots, N_{L-1} \in \mathbb{N}$ and $N_L := d$ are the dimensions of the layers. These matrices and vectors contain the *weights* and *biases*. They are also called *parameters* of the network and we notate them by θ. By Θ we notate the parameter space which contains the parameters θ.
The *realization* of a neural network Φ with *activation functions* $\rho_1, \ldots, \rho_L : \mathbb{R} \to \mathbb{R}$ is given by the function

$$R(\Phi) : \mathbb{R}^D \to \mathbb{R}^d, \qquad R(\Phi)(x) := x_L,$$

where the output x_L results from

$$\begin{aligned} &x_0 := x \\ &x_l = \rho_l.(A_l x_{l-1} + b_l) \text{ for } l = 1, \ldots, L. \end{aligned} \tag{1}$$

Here $\rho_l.$ denotes the component-wise application of the activation function ρ_l to the vectors $A_l x_{l-1} + b_l$.
We call $N(\Phi) := D + \sum_{j=1}^{L} N_l$ the *number of neurons*, $L(\Phi) := L$ the *number of layers* or *depth* and

$$M(\Phi) := \sum_{l=1}^{L} M_l := \sum_{l=1}^{L} ||A_l||_0 + ||b_l||_0$$

the *number of weights*. Here $||\cdot||_0$ denotes the number of non-zero entries of a matrix or vector.
A vector $S = (N_0, \ldots, N_L) \in \mathbb{N}^{L+1}$ is called *architecture* of a neural network Φ. Given such a vector S, we denote by $\mathcal{NN}(S)$ the set of all neural networks with architecture S.
By a *multilayer perceptron (MLP)* we refer to the realization of a neural network.

A schematic representation of the realization of a neural network can be found in figure 1 and some examples for activation functions can be found in figure 2. We will see later that the choice of activation function plays an important role in simple MLP's for approximating arbitrary continuous functions.

[1] This definition is based on Definition 2.8 from [Pet22].

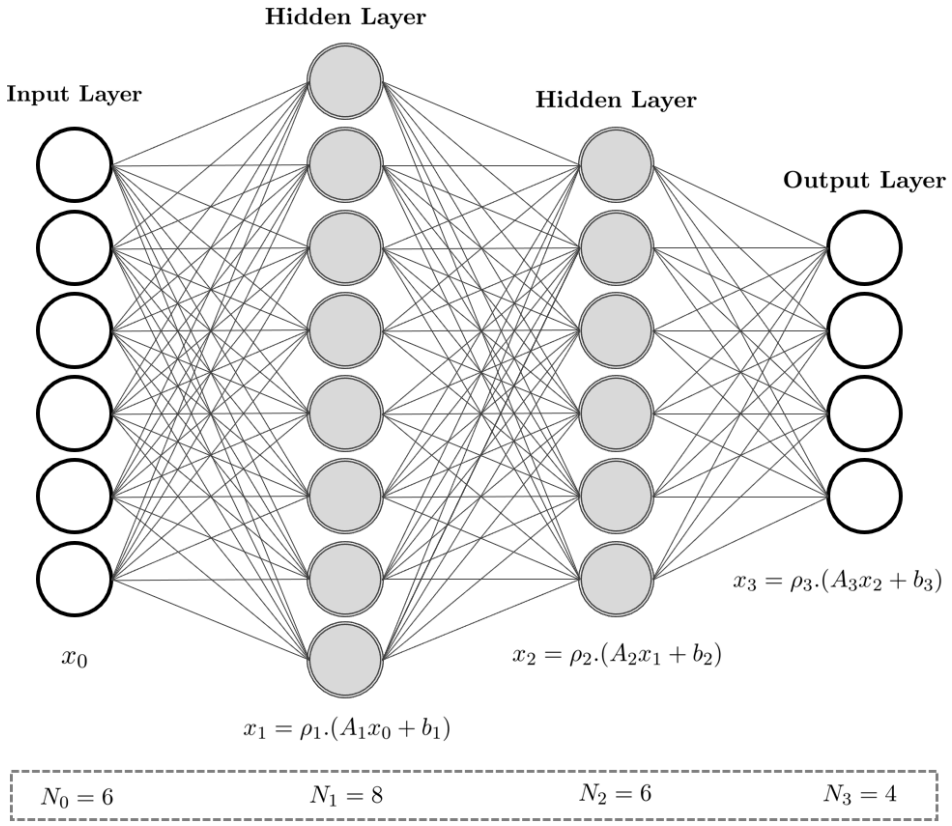


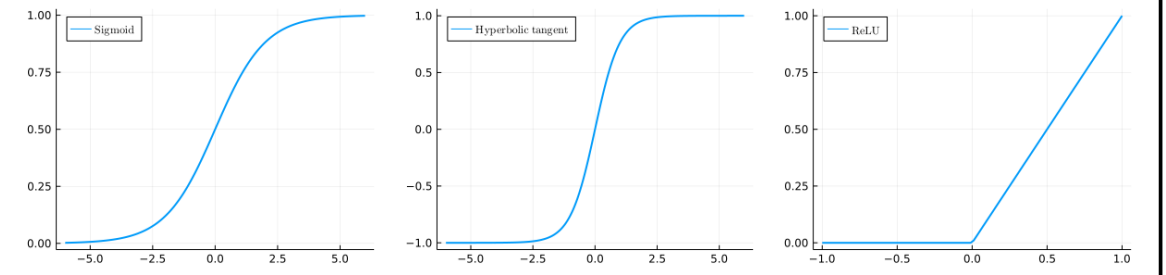
Figure 1: Schematic representation of the realization of the neural network $\Phi = ((A_1, b_1), (A_2, b_2), (A_3, b_3))$ with activation functions ρ_1, ρ_2, ρ_3 and architecture $S = (6, 8, 6, 4)$.

Remark 1.2. The set $\mathcal{NN}(S)$ is a finite-dimensional normed vector space with dimension

$$\begin{aligned}
\dim(\mathcal{NN}(S)) &= \dim\left(\left(\mathbb{R}^{N_1 \times N_0} \oplus \mathbb{R}^{N_1}\right) \oplus \ldots \oplus \left(\mathbb{R}^{N_L \times N_{L-1}} \oplus \mathbb{R}^{N_L}\right)\right) \\
&= \left(\left(\dim\left(\mathbb{R}^{N_1 \times N_0}\right) + \dim\left(\mathbb{R}^{N_1}\right)\right) + \ldots + \left(\dim\left(\mathbb{R}^{N_L \times N_{L-1}}\right) + \dim\left(\mathbb{R}^{N_L}\right)\right)\right) \\
&= (N_1 \cdot N_0 + N_1) + \ldots + (N_L \cdot N_{L-1} + N_L) \\
&= \sum_{l=1}^{L} N_l \cdot N_{l-1} + N_l
\end{aligned}$$

and norm

$$||\Phi|| := \max_l ||A_l||_\infty + \max_l ||b_l||_\infty.$$



Sigmoid

$\tilde{S} : \mathbb{R} \to \mathbb{R};$

$$x \mapsto \frac{1}{1+e^{-x}}$$

$\tilde{S}'(x) = \tilde{S}(x)(1-\tilde{S}(x))$

Hyperbolic tangent

$\tanh : \mathbb{R} \to \mathbb{R};$

$$x \mapsto \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$\tanh'(x) = 1 - \tanh^2(x)$

Rectified linear unit(ReLU)

$\mathrm{ReLU} : \mathbb{R} \to \mathbb{R};$

$$x \mapsto \max\{x, 0\}$$

$\mathrm{ReLU}'(x) = \mathbb{1}_{\mathbb{R}_+}(x)$

Figure 2: Examples of commonly used activation functions.

Now, we can define an autoencoder as follows.

Definition 1.3 (Autoencoder)**.** Let $L_1, L_2 \in \mathbb{N}$ and $D, d \in \mathbb{N}$ with $d \ll D$. Then an *autoencoder* $\mathrm{AE}(\Phi_1, \Phi_2)$ is defined by the two neural networks $\Phi_1 = ((A_1^1, b_1^1), \dots, (A_{L_1}^1, b_{L_1}^1))$ with input dimension D and output dimension d and $\Phi_2 = ((A_1^2, b_1^2), \dots, (A_{L_2}^2, b_{L_2}^2))$ with input dimension d and output dimension D. The neural network Φ_1 is also called *encoder* network and the neural network Φ_2 is also called *decoder* network. The realization $R(\mathrm{AE}(\Phi_1, \Phi_2)) : \mathbb{R}^D \to \mathbb{R}^D$ of an autoencoder is defined by

$$R(\mathrm{AE}(\Phi_1, \Phi_2))(x) := (R(\Phi_2) \circ R(\Phi_1))(x)$$

for every $x \in \mathbb{R}^D$. The realization of the encoder network is then called *encoding* and the realization of the decoder network is called *decoding*. The space in which the encoded data resides is called *encoded data space* or *latent space*.

1.1.2 Universal Approximation Theorem

One of the most famous results in neural network theory is that under minor conditions to the activation function on a compactum, arbitrary continuous functions can be approximated by a MLP with two layers. This result was first shown by Hornik [HSW89] and Cybenko [Cyb89]. Neural networks thus realize approximately arbitrary functions that map the original data to their lower-dimensional representation. This subsection is based on chapter 2.1 in [Pet22].
To obtain a suitable approximation term we need to define a topology on the space of functions of interest. Let $K \subset \mathbb{R}^D$ be a compact set and

$$C(K) = \{f : K \to \mathbb{R} \mid f \text{ is continuous}\}$$

the set of continuous functions on K which we equip with the uniform norm

$$||f||_\infty = \sup_{x \in K} |f(x)|.$$

If K is a compact space, we know from Riesz's theorem that the topological dual space of $C(K)$ is the space

$$\mathscr{M} = \{\mu \mid \mu \text{ is a signed Borel measure on } K\}.$$

Now, we can define the concept of universality.

Definition 1.4 (Universality)**.** Let $D, L \in \mathbb{N}$, $\rho : \mathbb{R} \to \mathbb{R}$ be a continuous activation function and K be a compact set. Denote by $\mathrm{MLP}(\rho, D, d, L)$ the set of all MLP's with D-dimensional input, L layers, output dimension d and activation functions ρ with last activation function as the identity, i.e. the set of realizations of neural networks with architecture $S = (D, N_1, \ldots, N_{L-1}, d)$ for $N_1, \ldots, N_{L-1} \in \mathbb{N}$ and activation functions $\rho_1, \ldots, \rho_{L-1}, Id$ with $\rho = \rho_1 = \ldots = \rho_{L-1}$.
We call the set of MLPs with one-dimensional output $\mathrm{MLP}(\rho, D, 1, L)$ universal if $\mathrm{MLP}(\rho, D, 1, L)$ is dense in $C(K)$.

Definition 1.5 (Discriminatory)**.** Let $D \in \mathbb{N}$ and $K \subset \mathbb{R}^D$ be a compact set. A continuous function $f : \mathbb{R} \to \mathbb{R}$ is called *discriminatory* if for all $\mu \in \mathscr{M}$ such that

$$\int_K f(Ax - b)\mathrm{d}\mu(x) = 0 \qquad \text{for all } A \in \mathbb{R}^{1 \times D} \text{ and } b \in \mathbb{R}$$

holds follows $\mu \equiv 0$.

The classical universal approximation theorem is now given by:

Theorem 1.6 (Universal approximation theorem)**.** Let $D \in \mathbb{N}$, $K \subset \mathbb{R}^D$ be a compact set and $\rho : \mathbb{R} \to \mathbb{R}$ be a discriminatory activation function. Then $\mathrm{MLP}(\rho, D, 1, 2)$ is universal.

Proof. The proof can be found in theorem 2.4 [Pet22]. □

It was proven by Leshno [LLPS92] that this theorem can be extended further.

Theorem 1.7 (Extension of the universal approximation theorem)**.** Let $D \in \mathbb{N}$, $K \subset \mathbb{R}^D$ be a compact set and $\rho : \mathbb{R} \to \mathbb{R}$ be a continuous acitvation function. Then the set of multilayer perceptrons $\mathrm{MLP}(\rho, D, 1, 2)$ with bias $b_2 = 0$ is universal if and only if the activation function ρ is not polynomial.

Proof. The proof can be found in [LLPS92]. □

Moreover, we note that this theorem can be extended to both MLPs with arbitrary depth $L \geq 2$ and arbitrary output dimension $d \in \mathbb{N}$.
The universal approximation theorem states that every arbitrary continuous function can be approximated by a sequence of neural networks. But it makes no statement about the choice of the networks with which we can approximate the function that maps the input data into the output data. For this reason, we consider in the following section how to find the parameters of the neural network in such a way that we receive a suitable approximation. This process is called *training*.

1.1.3 Training of Neural Networks

In this section we restrict our attention to neural network training.
Let a data set $\mathcal{X} = \{x_i\}_{i=1,\dots,N} \subset \mathbb{R}^{D \times N}$ be given. We now want to approximate a function that maps this data set into a lower-dimensional representation while respecting a task. In the supervised setting for example such a task could be the classification of labeled data. In the unsupervised setting with an autoencoder such a task could be to encode the data into the latent space and then decode the encoded data such that the realization of the autoencoder is as close as possible to the original data. We then want to approximate this function with a neural network by adjusting the parameters of the network with an iterative algorithm. We define an objective function which we want to minimize with this algorithm.

Definition 1.8 (Loss function)**.** Let Φ be a neural network with output dimension $d \in \mathbb{N}$ and parameters θ. We define the output data set $\hat{\mathcal{X}} = \{\hat{x}_i\}_{i=1,\dots,N} \subset \mathbb{R}^d$ by $\hat{x}_i := R(\Phi)(x_i)$ for all $i = 1, \dots, N$. Then, we want to assign a loss to a given realization of the neural network $R(\Phi)(x)$. A *loss function* is a function

$$L : \Theta \times \mathbb{R}^d \to [0, \infty),$$

such that for all $\theta \in \Theta$ the function $L(\theta, \cdot) : \mathbb{R}^d \to [0, \infty)$ is Borel-measurable.

Given parameters θ and realization $\hat{x}_i$ we can now compute the loss $L(\theta, \hat{x}_i)$.

Definition 1.9 (Objective loss function)**.** We define the *objective loss function* which calculates the empirical risk of the realization with parameters θ by

$$\begin{aligned} &\mathcal{L} : \Theta \to [0, \infty) \\ &\theta \mapsto \mathcal{L}(\theta) := \mathbb{E}[L(\theta, x)] \approx \frac{1}{N} \sum_{i=1}^{N} L(\theta, \hat{x}_i). \end{aligned}$$

Note that the expected value is calculated using the probability measure P_θ with respect to the parameter θ on the data space. The goal is now to solve the nonlinear and typically nonconvex optimization problem

$$\arg\min_{\theta \in \Theta} \mathcal{L}(\theta).$$

If the objective loss function is differentiable, we can minimize it by calculating the gradients

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta L(\theta, \hat{x}_i) = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta L(\theta, R(\Phi)(x_i)).$$

We require that the loss function which depends on the realization of the neural network is differentiable. The gradients of a data point $x = x_i$ can then be calculated applying the chain rule [2]

$$\nabla_\theta L(\theta, R(\Phi)(x)) = \nabla_\theta x_{L-1} \nabla_{x_{L-1}} L(\theta, x_L) = \nabla_\theta x_{L-2} \nabla_{x_{L-2}} x_{L-1} \nabla_{x_{L-1}} L(\theta, x_L) = \cdots .$$

Thus, we can calculate the gradient of the loss with respect to all parameters of the network,

[2] The notation of $x_l = \rho_l.(A_l x_{l-1} + b_l)$ for $l = 1, \dots, L-1$ here is notation from equation 1 for a corresponding neural network.

i.e. the weights and biases, by differentiating iteratively backwards. This method is called *backpropagation* and was developed by Rumelhart et. al [RHW86]. Here we must additionally require that the activation functions of the neural network are at least piecewise differentiable. We now want to optimize the parameters and thereby train our model by calculating the gradients using backpropagation. In particular, we want to focus on two optimization algorithms. First, we introduce the *gradient descent* algorithm as an optimization algorithm which is represented by algorithm 1. The idea of this algorithm is to take steps in the direction of the negative gradient in the parameter space. The negative gradient defines a descent direction of the objective loss function. Therefore, we minimize the objective loss function. The stepwidth depends on the learning rate of the algorithm.

Algorithm 1: Gradient descent

Require: dataset $\mathcal{X}$, learning rate $\eta \in (0, 1]$, θ_0, epochs
$k \leftarrow 0$
while $k <$ epochs **do**
  $\nabla \leftarrow \text{backprop}(\theta_k, \mathcal{L}, \mathcal{X})$
  $\theta_{k+1} \leftarrow \theta_k - \eta\nabla$
  $k \leftarrow k + 1$
end

Curry [Cur44] proved that this algorithm for nonlinear optimization problems converges to a local minimum. For convex objective loss functions, a local minimum also corresponds to a global minimum. Due to the highly nonconvex structure of the realization of the neural network the algorithm does not necessarily converge to a global minimum.
The second algorithm we want to introduce is the *stochastic gradient descent* algorithm from page 72 [KW$^+$19] which is represented by algorithm 2. This algorithm uses randomly sampled subsets $\mathcal{D}_1, \ldots, \mathcal{D}_{n_{\text{batches}}}$ of the dataset $\mathcal{X}$, so called *mini batches*. On each mini batch we update the parameters using gradient descent. The stochasticity introduced by this algorithm mitigates the consequences of the convexity problem mentioned above. Compared to the gradient descent algorithm the convergence to better local minima is possible and faster convergence of the algorithm per epoch is given.
Another optimizer we will use later for our methods is Kingma's Adam optimizer [KB14]. This is fundamentally based on the momentum method.

Algorithm 2: Stochastic gradient descent

Require: dataset $\mathcal{X}$, learning rate $\eta \in (0, 1]$, θ_0, epochs, n_{batches}

$k, e \leftarrow 0$

while $e <$ epochs **do**

  sample mini batches $\mathcal{D}_1, \ldots, \mathcal{D}_{n_{\text{batches}}} \subset \mathcal{X}$

  for n_{batch} in $1 : n_{\text{batches}}$ **do**

    $\nabla \leftarrow \text{backprop}(\theta_k, \mathcal{L}, \mathcal{D}_{n_{\text{batch}}})$

    $\theta_{k+1} \leftarrow \theta_k - \eta\nabla$

    $k \leftarrow k + 1$

  end

  $e \leftarrow e + 1$

end

1.2 Stochastic Differential Equations

In this chapter we introduce stochastic differential equations. To be able to introduce them, we first have to deal with stochastic processes in order to be able to define the stochastic integral.

1.2.1 Stochastic Processes and the Stochastic Integral

This section is based on [Sch21a]. In the following sections let $(\Omega, \mathcal{F}, P)$ be a probability space and (E, d) a complete, separable metric space with Borel-σ-algebra $\mathcal{B}(E)$. The objective of this section is to construct the stochastic integral

$$H \cdot X = \left(\int_0^t H_s \mathrm{d}X_s \right)_{t \in \mathbb{R}_{\geq 0}}$$

for a semimartingale X and an appropriate process H. With this knowledge we can then define stochastic differential equations of the form

$$\begin{aligned} \mathrm{d}X_t &= b(t, X_t)\mathrm{d}t + \boldsymbol{\sigma}(t, X_t)\mathrm{d}W_t \\ X_0 &= \xi. \end{aligned}$$

We will later define the Ornstein-Uhlenbeck process as the solution of a specific stochastic differential equation. This process is important for our methods. We start by defining a stochastic process.

Definition 1.10 (Stochastic process)**.** A *stochastic process* is a family $X = (X_t)_{t \in \mathbb{R}_{\geq 0}}$ such that $X_t : \Omega \to E$ is a random variable, i.e. X_t is $\mathcal{F}/\mathcal{B}(E)$-measurable, for all $t \in \mathbb{R}_{\geq 0}$. For a fixed $\omega \in \Omega$ we call the map $t \mapsto X_t(\omega)$ a *path* of X. We call a stochastic process continuous if all paths are continuous.

Definition 1.11 (Filtration)**.** An increasing sequence $\mathbb{F} = (\mathcal{F}_t)_{t \in \mathbb{R}_{\geq 0}}$ of sub-σ-algebras of $\mathcal{F}$, that is

$$\mathcal{F}_s \subset \mathcal{F}_t \subset \mathcal{F} \text{ for all } 0 \leq s \leq t,$$

is called a *filtration*. Further, we call a filtration *right-continuous* if $\mathcal{F}_t = \bigcap_{u \geq t} \mathcal{F}_u$ holds for all $t \geq 0$.
We call a stochastic process X $\mathbb{F}$-*adapted* if X_t is $\mathcal{F}_t$-measurable for all $t \geq 0$. For simplicity we write adapted instead of $\mathbb{F}$-adapted.

Definition 1.12 (Stochastic basis)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space and let $\mathbb{F}$ be a right-continuous filtration $\mathbb{F}$. Then we call $(\Omega, \mathcal{F}, P, \mathbb{F})$ a *filtered probability space* or *stochastic basis*.
A stochastic basis satisfies the *usual conditions* if it is complete, i.e. if every subset of a P-null set is measurable and $\mathcal{F}_0$ contains all P-null sets.

In the following we work on a given stochastic basis. For the next two definitions we refer to Definition 16.1 and 16.19 [Pfa].

Definition 1.13 (Markov process)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space with filtration $\mathbb{F}$. An adapted stochastic process X is called *Markov process* if $\mathcal{F}_s$ and X_t are independent given X_s for $s \leq t$ that means if for all $A \in \mathcal{B}(E)$

$$P(X_t \in A|\mathcal{F}_s) = P(X_t \in A|X_s)$$

holds.

Definition 1.14 (Time-homogeneous Markov process)**.** A Markov process X is called *time-homogeneous* if for all $A \in \mathcal{B}(E)$ and $s \leq t$

$$P(X_t \in A|\mathcal{F}_s) = P(X_t \in A|X_s)$$

only depends on $t - s$, i.e. if we have

$$P(X_t \in A|X_s) = P(X_{t-s} \in A|X_0)$$

for all $A \in \mathcal{B}(E)$.

We now define a Brownian motion which is an important example for a stochastic process.

Definition 1.15 (Brownian motion)**.** A *Brownian motion* W is an adapted, continuous stochastic process such that

(i) $W_0 = 0$,

(ii) $\mathbb{E}[W_t] = 0$ and $\mathbb{V}[W_t] < \infty$ for all $t \in \mathbb{R}_{\geq 0}$,

(iii) $W_t - W_s$ is independent of $\mathcal{F}_s$ for all $0 \leq s \leq t$ and

(iv) W has *normal increments*, that is $W_t - W_s \sim \mathcal{N}(0, t - s)$ for all $0 \leq s \leq t$.

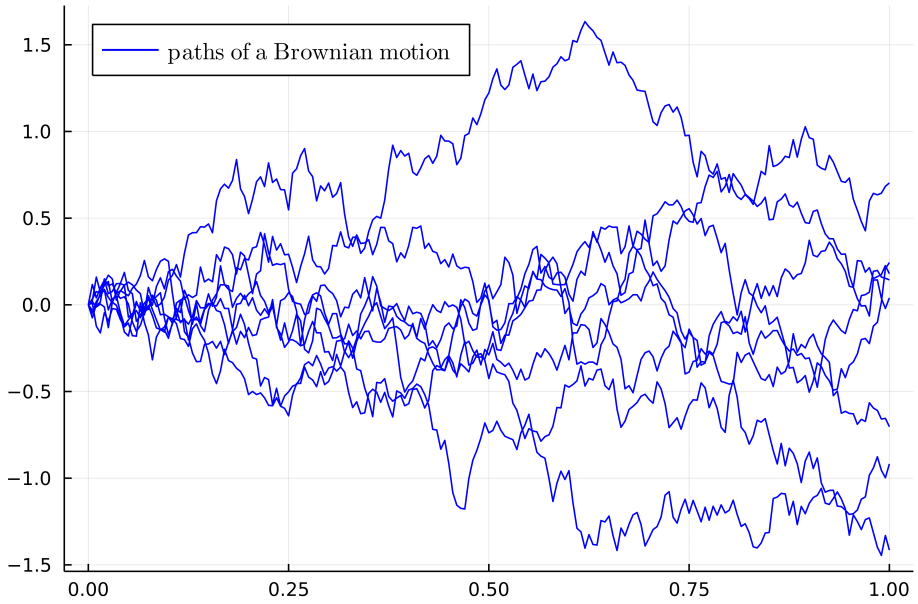


Figure 3: Representation of eight paths of a Brownian motion.

Some paths of a Brownian motion are represented in figure 3.

Definition 1.16 (Martingale)**.** A *(sub-/super-) martingale* M is an adapted, continuous stochastic process such that

(i) $\mathbb{E}[M_t] < \infty$ for all $t \geq 0$ and

(ii) $\mathbb{E}[M_t|\mathcal{F}_s] = M_s$ for all $0 \leq s \leq t$.

For submartingales the second condition is replaced by $\mathbb{E}[M_t|\mathcal{F}_s] \geq M_s$ and for supermartingales by $\mathbb{E}[M_t|\mathcal{F}_s] \leq M_s$.

We note that a Brownian motion W is a martingale because

$$\mathbb{E}[W_t|\mathcal{F}_s] = \mathbb{E}[W_t - W_s + W_s|\mathcal{F}_s] = \mathbb{E}[W_t - W_s|\mathcal{F}_s] + \mathbb{E}[W_s|\mathcal{F}_s] = \mathbb{E}[W_t - W_s] + W_s = W_s$$

holds for all $0 \leq s \leq t$.

Definition 1.17 (Locally bounded variation)**.** We call a stochastic process X *of locally bounded variation* if

$$\mathrm{Var}(X)_t := \sup_{0\leq t_0\leq\ldots\leq t_n\leq t} \sum_{i=1}^{n} \left|X_{t_i}(\omega) - X_{t_{i-1}}(\omega)\right| < \infty$$

holds for all $t \geq 0$ and $\omega \in \Omega$.

The Brownian motion is not of locally bounded variation. For the prove of this we refer to Theorem 29 of [Pro04]. Now, we can define the class of processes with respect to which we can integrate.

Definition 1.18 (Semimartingale)**.** A stochastic process X is called a *semimartingale* if it has a decomposition

$$X = X_0 + M + A,$$

where M is a local martingale such that $M_0 = 0$ and A is a stochastic process of locally bounded variation.

In this definition it is required that M is a local martingale, a concept we have not introduced yet. A stochastic process satisfies a property "locally" if there exists a localizing sequence, i.e. a sequence of stopping times which increases to infinity such that the process stopped at any of these stopping times satisfies the desired property. See also Definition 4.1 [Sch21a] or Definition 1.6.2 [Tap18]. For the concept of a stopping time we refer to Definition 2.1 [Sch21a]. We will not discuss local martingales further in this thesis. Important for us to know is that every martingale is a local martingale.
We note that the decomposition of continuous semimartingales is unique. Let X be a continuous semimartingale with decompositions $X = X_0 + M + A = X_0 + N + B$ for local martingales M and N with $M_0 = 0 = N_0$ and processes A and B with locally bounded variation. Then the process $M - N = B - A$ is a continuous local martingale with locally bounded variation. Further Theorem 5.21 [Sch21a] implies that $M - N = 0 = B - A$. Thus the uniqueness is shown. In particular, the Brownian motion has a unique semimartingale decomposition since it is a continuous martingale.
We can now define the stochastic integral with simple integrands and then expand it.

Definition 1.19 (Stochastic integral with simple integrands)**.** We call a stochastic process H *simple* if it has the form

$$H = Y\mathbb{1}_{[0]} \qquad \text{or} \qquad H = Y\mathbb{1}_{(u,v]}$$

for a bounded and $\mathcal{F}_u$-measurable random variable Y and $0 \leq u \leq v$.
Let X be a semimartingale. Then we can define the stochastic integral with respect to a simple stochastic process H as the stochastic process $H \cdot X$ given by

$$(H \cdot X)_t := \int_0^t H_s \mathrm{d}X_s := \begin{cases} 0 & H = Y\mathbb{1}_{[0]} \\ Y\left(X_{\min\{v,t\}} - X_{\min\{t,u\}}\right) & H = Y\mathbb{1}_{(u,v]} \end{cases}.$$

The idea is to continuously extend the integral to suitable large spaces by taking limits of simple integrands.

Theorem 1.20 (Stochastic integral)**.** Let X be a semimartingale. The map $H \mapsto H \cdot X$ defined on the space of continuous, simple integrands has an extension to the space of locally bounded, continuous integrands such that

(i) the stochastic process $H \cdot X$ is adapted and continuous and

(ii) the map $H \mapsto H \cdot X$ is linear

Proof. For the proof, see [JS03] theorem 4.31. □

Definition 1.21 (Quadratic covariation)**.** Let X and Y be two continuous semimartingales. The *quadratic covariation* of X and Y is the stochastic process $[X, Y]$ defined by

$$[X, Y]_t := X_0 Y_0 + XY - (X \cdot Y)_t - (Y \cdot X)_t.$$

Lemma 1.22. Let X and Y be two continuous semimartingales. For $t > 0$ a partition $\xi_n = \{\tau_{n,k_1}, \ldots, \tau_{n,k_n}\}$ of $[0, t]$ is given such that $\max_k |\tau_{n,k} - \tau_{n,k-1}| \xrightarrow{n\to\infty} 0$. Then

$$Z_n := \sum_{k=1}^{k_n} (X_{t_{n,k}} - X_{t_{n,k-1}})(Y_{t_{n,k}} - Y_{t_{n,k-1}}) \xrightarrow{n\to\infty}_p [X, Y]_t$$

holds where $\to_p$ denotes the convergence in probability.

Proof. For the statement and the proof we refer to Lemma 19.50 [Pfa]. □

The following theorem is one of the most important results of stochastic analysis.

Theorem 1.23 (Itô's lemma for continuous semimartingales)**.** Let $X = (X^1, \dots, X^d)$ be a d-dimensional continuous semimartingale with component-wise semimartingale decomposition $X^i = X_0^i + M^i + A^i$ and let f be a function in $C^2(\mathbb{R}^d, \mathbb{R})$. The stochastic process $f(X)$ is then a semimartingale such that for all $t \geq 0$

$$f(X)_t = f(X_0) + \sum_{i=1}^{d} \int_0^t \partial_i f(X)_s \mathrm{d}X_s^i + \frac{1}{2} \sum_{i,j=1}^{d} \int_0^t \partial_{ij} f(X)_s \mathrm{d}[M^i, M^j]_s.$$

Proof. We refer to the proof of Theorem 4.57 in [JS03]. □

1.2.2 Stochastic Differential Equations

In this section we define stochastic differential equations and their solutions. We also mention under what conditions unique solutions exist. In order to determine a numerical solution of a stochastic differential equation we introduce the Euler-Maruyama method. This section is based on Section 5.2 of [KS12].

Definition 1.24 (Stochastic differential equation)**.** Let $b : \mathbb{R}_{\geq 0} \times \mathbb{R}^d \to \mathbb{R}^d$ and $\boldsymbol{\sigma} : \mathbb{R}_{\geq 0} \times \mathbb{R}^d \to \mathbb{R}^{d \times r}$ be Borel-measurable functions and let W be an r-dimensional Brownian motion. We call b the drift vector and $\boldsymbol{\sigma}$ the dispersion matrix. An equation of the form

$$\mathrm{d}X_t = b(t, X_t)\mathrm{d}t + \boldsymbol{\sigma}(t, X_t)\mathrm{d}W_t \tag{2}$$

is called *stochastic differential equation*. A stochastic differential equation can also be written component-wise as

$$\mathrm{d}X_t^i = b^i(t, X_t)\mathrm{d}t + \sum_{j=1}^{r} \boldsymbol{\sigma}_{ij}(t, X_t)\mathrm{d}W_t^j$$

for $1 \leq i \leq d$. In this context X is a continuous d-dimensional stochastic process. The drift vector b and the dispersion matrix $\boldsymbol{\sigma}$ are the coefficients of this equation and we call the matrix $\boldsymbol{\sigma}\boldsymbol{\sigma}^{\mathrm{T}}$ the *diffusion matrix*.
If there is also a d-dimensional random variable ξ given such that in addition to the stochastic differential equation 2 also $X_0 = \xi$ needs to be held we call this problem a *stochastic initial value problem*. Further, we call ξ the *initial value*.

We distinguish strong and weak solutions and first want to define strong solutions of stochastic differential equations.
In order to define strong solutions we fix a probability space $(\Omega, \mathcal{F}, P)$ on which an r-dimensional Brownian motion exists. Furthermore, we assume that a d-dimensional random variable ξ exists which is independent of W. We consider

$$\mathcal{G}_t = \sigma(\xi, W_s; 0 \leq s \leq t) \text{ with } \mathcal{G}_\infty = \sigma\left(\bigcup_{t\geq 0} \mathcal{G}_t\right)$$

for all $t \geq 0$ as well as the collection of null sets

$$N_0 = \{N \subset \Omega \mid \exists G \in \mathcal{G}_\infty \text{ with } N \subset G \text{ and } P(G) = 0\}.$$

Now, we can create the filtration $\mathbb{F} = (\mathcal{F}_t)_{t\in\mathbb{R}_{\geq 0}}$ by

$$\mathcal{F}_t = \sigma(\mathcal{G}_t \cup N_0)$$

for all $t \geq 0$ and

$$\mathcal{F}_\infty = \sigma\left(\bigcup_{t\geq 0} \mathcal{F}_t\right).$$

Following the proof of Theorem 2.7.7 of [KS12] we observe that the filtration $\mathbb{F}$ satisfies the usual conditions.
For the following definition see also Definition 2.1 [KS12].

Definition 1.25 (Strong solution)**.** A continuous stochastic process X on the probability space $(\Omega, \mathcal{F}, \mathbb{F})$ with respect to the Brownian motion W and initial condition ξ is called a *strong solution* of the stochastic differential equation 2 if it satisfies the following properties:

(i) X is adapted to the above defined filtration $\mathbb{F}$,

(ii) $P(X_0 = \xi) = 1$,

(iii) for all $t \geq 0$ we have

$$P\left(\int_0^t |b_i(s, X_s)| + \boldsymbol{\sigma}_{ij}^2(s, X_s)\mathrm{d}s < \infty\right) = 1$$

(iv) and for all $t \geq 0$ we have

$$X_t = X_0 + \int_0^t b(s, X_s)\mathrm{d}s + \int_0^t \boldsymbol{\sigma}(s, X_s)\mathrm{d}W_s.$$

Definition 1.26 (Weak solution)**.** A *weak solution* of the stochastic differential equation 2 is a tuple $((X, W), (\Omega, \mathcal{F}, P, \mathbb{F}))$ where

(i) $(\Omega, \mathcal{F}, P, \mathbb{F})$ is a stochastic basis,

(ii) X is an adapted continuous stochastic process and

(iii) W is an r-dimensional Brownian motion

such that the properties (iii) and (iv) from definition 1.25 are satisfied.

Therefore, the difference between weak and strong solutions becomes clear. For strong solutions of the stochastic differential equation the Brownian motion is given. In contrast, for weak solutions we do not only look for a stochastic process that solves the stochastic differential equation but also for a Brownian motion. Thus, it is clear that every strong solution is also a weak solution.

Theorem 1.27 (Existence and uniqueness of strong solutions)**.** Suppose that the random variable ξ is square integrable, i.e $\mathbb{E}[||\xi||^2] < \infty$. Further assume that the coefficients b and $\boldsymbol{\sigma}$ are globally Lipschitz continuous with Lipschitz constants $K_b > 0$ and $K_{\boldsymbol{\sigma}} > 0$ and satisfy the *linear growth conidition*, i.e. that for $K = K_b + K_{\boldsymbol{\sigma}} > 0$ and for all $t \geq 0$ and $x \in \mathbb{R}^d$ we have

$$||b(t, x)||^2 + ||\boldsymbol{\sigma}(t, x)||^2 \leq K^2(1 + ||x||^2). \tag{3}$$

Then, there exists a unique[3] strong solution of the stochastic differential equation 2.

Proof. For the statements and proofs see Theorem 2.5 and Theorem 2.9 [KS12]. □

We will see that the stochastic differential equations considered in this thesis will satisfy the conditions of Theorem 1.27.

[3]By uniqueness we mean uniqueness up to indistinguishability as defined in Definition 1.4 [Sch21a]

We now want to introduce the Euler-Maruyama method based on the method in section 6.1.1 of [Gla04]. This method builds on the explicit Euler method for determining an approximate solution of an ordinary initial value problem, see also Algorithm 21.1 [Bar16]. The Euler-Maruyama method is a numerical method for approximating the solution of a stochastic initial value problem.

Definition 1.28 (Euler-Maruyama method)**.** Let W be an r-dimensional Brownian motion and a stochastic initial value problem

$$\begin{aligned} \mathrm{d}X_t &= b(t, X_t)\mathrm{d}t + \boldsymbol{\sigma}(t, X_t)\mathrm{d}W_t \\ X_0 &= x_0 \end{aligned}$$

for a given vector $x_0 \in \mathbb{R}^d$. To determine the numerical approximation of the solution on an interval $[0, T]$ with $T > 0$ we choose discrete and deterministic time points

$$0 = \tau_0 < \tau_1 < \ldots < \tau_n = T$$

for $n \in \mathbb{N}$. We define the individual step width h_k of the time points by $h_k = \tau_{k+1} - \tau_k$ for $k = 0, \ldots, n-1$. The stochastic differential $\mathrm{d}W_t$ is now replaced by the increments

$$\Delta W_k := W_{\tau_{k+1}} - W_{\tau_k}$$

for $k = 0, \ldots, n-1$. From property (iv) of the definition of the Brownian motion 1.15 we know that

$$\Delta W_k \sim \mathcal{N}(0, h_k)$$

holds for $k = 0, \ldots, n-1$. The *Euler-Maruyama method* calculates an approximation $\hat{X}$ of X by

$$\begin{aligned} \hat{X}_{k+1} &= \hat{X}_k + b(\tau_k, \hat{X}_k) \cdot h_k + \boldsymbol{\sigma}(\tau_k, \hat{X}_k) \cdot \Delta W_k \\ \hat{X}_0 &= x_0 \end{aligned}$$

for $k = 0, \ldots, n-1$. Then $\hat{X}_k$ is an approximation of X_{τ_k}.

1.2.3 Ornstein-Uhlenbeck Process

In this section we take a closer look at the Ornstein-Uhlenbeck process as a solution of a particular stochastic differential equation. This process will play an important role in the application of our methods in Chapter 3. We first define the Ornstein-Uhlenbeck process in the one-dimensional case and compute its explicit representation. Then we want to extend this concept to any number of dimensions.
Let in the following $(\Omega, \mathcal{F}, P)$ be a probability space with filtration $\mathbb{F}$ that satisfies the usual conditions.

Definition 1.29 (Ornstein-Uhlenbeck process)**.** Let $x_0, \mu \in \mathbb{R}$ and constants $\vartheta, \sigma > 0$ be given. An one-dimensional stochastic process X is called *Ornstein-Uhlenbeck process* with initial value x_0, equilibrium level μ, stiffness ϑ and diffusion σ if it solves the stochastic inital value problem

$$\begin{aligned} \mathrm{d}X_t &= \vartheta \cdot (\mu - X_t)\mathrm{d}t + \sigma \mathrm{d}W_t \\ X_0 &= x_0. \end{aligned} \tag{4}$$

The initial value x_0 of an Ornstein-Uhlenbeck process describes the value of the process at the starting time. The equilibrium level μ describes the value which the paths of the process approach over time and the stiffness ϑ is the "attraction" of the equilibrium level on the process. If ϑ is larger, the process will approach μ more quickly. The diffusion σ describes the stochastic influence of the Brownian motion W. If $\sigma = 0$, the process will simply converge exponentially to μ but if the diffusion is larger this convergence will be increasingly randomly perturbed. Figure 4 represents different paths of the Ornstein-Uhlenbeck process with initial value $x_0 = 0$, equilibrium level $\mu = 3$, stiffness $\vartheta = 5$ and diffusion $\sigma = 1$.

Theorem 1.30 (Existence and uniqueness of an Ornstein-Uhlenbeck process)**.** Let $x_0, \mu \in \mathbb{R}$ and constants $\vartheta, \sigma > 0$ be given. Then there exists a unique stochastic process X that solves the stochastic initial value problem 4.

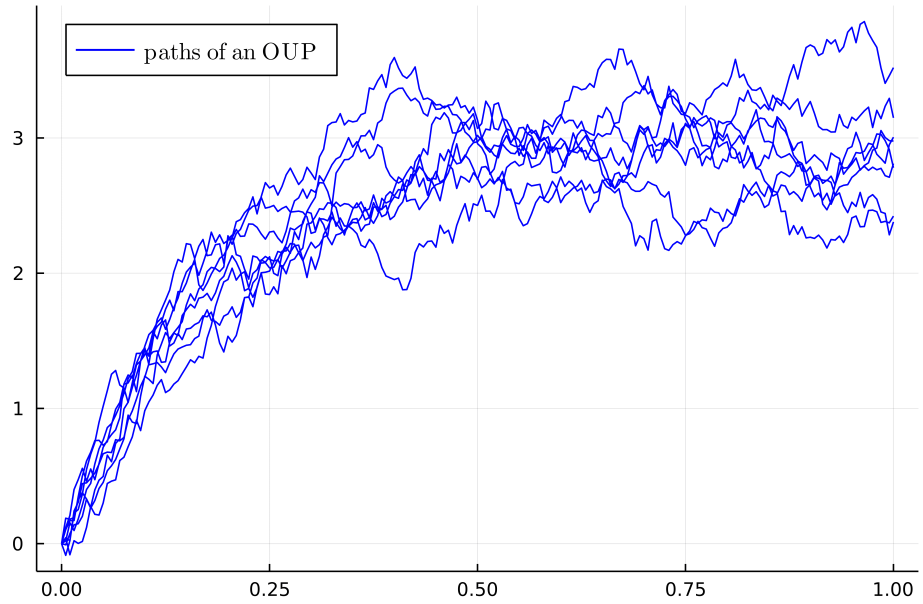


Figure 4: Representation of eight paths of an Ornstein-Uhlenbeck process.

Proof of Theorem 1.30. In order to show that a unique solution of this stochastic differential equation exists we must show that the additional requirements of Theorem 1.27 are satisfied. First, we note that the second moment of x_0 exists if the initial value x_0 is a constant in $\mathbb{R}$. Considering the drift coefficient $b(t,x) = \vartheta(\mu - x)$ and the diffusion coefficient $\boldsymbol{\sigma}(t,x) = \sigma$ we see that for all $t, t' \geq 0$ and $x, x' \in \mathbb{R}$ we have

$$\begin{aligned} \big|\big|b(t,x) - b(t',x')\big|\big| &= \big|\vartheta(\mu - x) - \vartheta(\mu - x')\big| \\ &= \vartheta \cdot \big|x - x'\big| \\ &\leq \vartheta \cdot (|t - t| + \big|x - x'\big|) \\ &= \vartheta \cdot \big|\big|(t,x) - (t',x')\big|\big| \end{aligned}$$

and

$$\big|\big|\boldsymbol{\sigma}(t,x) - \boldsymbol{\sigma}(t',x')\big|\big| = |\sigma - \sigma| = 0 < \sigma \cdot \big|\big|(t,x) - (t',x')\big|\big|.$$

Thus b and $\boldsymbol{\sigma}$ are globally Lipschitz continuous functions with Lipschitz constants ϑ and σ.

Further, we note that for the summed up Lipschitz constants $K := \vartheta + \sigma$ and for all $t \geq 0$ and $x \in \mathbb{R}$

$$\begin{aligned} ||b(t,x)||^2 + ||\boldsymbol{\sigma}(t,x)||^2 &= ||\vartheta(\mu - x)||^2 + \sigma^2 \\ &= \vartheta^2|(\mu+1) - (1+x)|^2 + \sigma^2 \\ &\leq \vartheta^2|1+x|^2 + \sigma^2|1+x|^2 \\ &\leq (\vartheta+\sigma)^2|1+x|^2 \\ &= K^2||1+x||^2 \end{aligned}$$

holds which means b and $\boldsymbol{\sigma}$ satisfy the linear growth condition 3. As mentioned in Theorem 1.27 these conditions are sufficient for the existence and uniqueness of a strong solution of the stochastic initial value problem. □

In the following theorem we want to show that the solution of the initial value problem 4 has an explicit representation.

Theorem 1.31 (Explicit representation of an Ornstein-Uhlenbeck process)**.** Let $x_0, \mu \in \mathbb{R}$ and $\vartheta, \sigma > 0$ be constants. The solution of the stochastic initial value problem 4 has the following explicit representation

$$X_t = x_0 e^{-\vartheta t} + \mu(1 - e^{-\vartheta t}) + \sigma \int_0^t e^{-\vartheta(t-s)} \mathrm{d}W_s. \tag{5}$$

Proof. This can be shown by Itô's lemma 1.23. Let us consider the function $f \in C^2(\mathbb{R}\times\mathbb{R}_{\geq 0}, \mathbb{R})$ given by $f(X_t, t) = X_t e^{\vartheta t}$. Then the following holds for the partial derivatives

$$\partial_x f(X_t,t) = e^{\vartheta t} \qquad \partial_t f(X_t,t) = \vartheta X_t e^{\vartheta t} \qquad \partial_{xx} f(X_t,t) = 0.$$

Using Itô's lemma and the fact that $[\mathrm{id}, \mathrm{id}]_t = 0$, we now get

$$\begin{aligned} X_t e^{\vartheta t} = f(X_t,t) &= f(X_0,0) + \int_0^t \partial_x f(X_s,s)\mathrm{d}X_s + \int_0^t \partial_s f(X_s,s)\mathrm{d}s \\ &= X_0 + \int_0^t e^{\vartheta s}\mathrm{d}X_s + \int_0^t \vartheta X_s e^{\vartheta s}\mathrm{d}s \\ &= X_0 + \int_0^t e^{\vartheta s}(\vartheta(\mu - X_s)\mathrm{d}s + \sigma \mathrm{d}W_s) + \int_0^t \vartheta X_s e^{\vartheta s}\mathrm{d}s \end{aligned}$$

$$= X_0 + \vartheta\mu \int_0^t e^{\vartheta s} \mathrm{d}s + \sigma \int_0^t e^{\vartheta s} \mathrm{d}W_s$$
$$= X_0 + \mu(e^{\vartheta t} - 1) + \sigma \int_0^t e^{\vartheta s} \mathrm{d}W_s.$$

If we multiply this equation by $e^{-\vartheta t}$ we get

$$X_t = X_0 e^{-\vartheta t} + \mu(1 - e^{-\vartheta t}) + \sigma \int_0^t e^{-\vartheta(t-s)} \mathrm{d}W_s,$$

which gives the above solution with $X_0 \equiv x_0$. □

We now want to extend this to $d \in \mathbb{N}$ dimensions.

Definition 1.32 (Multidimensional Ornstein-Uhlenbeck process)**.** Let X be a d-dimensional stochastic process and a vector $\mu \in \mathbb{R}^d$, an invertible matrix $\Sigma \in \mathbb{R}^{d\times d}$ and the diagonal matrix $\boldsymbol{\sigma} \in \mathbb{R}^{d\times d}$ are given. The stochastic process X is called d-dimensional Ornstein-Uhlenbeck process with initial value $x_0 \in \mathbb{R}^d$, equilibrium level μ, stiffness matrix Σ and dispersion matrix $\boldsymbol{\sigma}$ if it solves the following stochastic initial value problem

$$\begin{aligned} \mathrm{d}X_t &= \Sigma \cdot (\mu - X_t)\mathrm{d}t + \boldsymbol{\sigma}\mathrm{d}W_t \\ X_0 &= x_0. \end{aligned} \tag{6}$$

We assume that the dispersion matrix is a diagonal matrix. We further note that the Brownian motion is d-dimensional instead of an r-dimensional process compared to definition 1.24.

Theorem 1.33 (Existence and uniqueness of the solution of a multidimensional Ornstein-Uhlenbeck process)**.** Let vectors $x_0, \mu \in \mathbb{R}^d$, an invertible matrix $\Sigma \in \mathbb{R}^{d\times d}$ and a diagonal matrix $\boldsymbol{\sigma} \in \mathbb{R}^{d\times d}$ be given. Then a unique strong solution of the stochastic differential equation 6 exists.

Proof. The proof can be done analogously to the proof of Theorem 1.30 in the one-dimensional case. The Lipschitz constants are given by $||\Sigma||$ and $||\boldsymbol{\sigma}||$, where $||\cdot||$ is a suitable matrix norm on $\mathbb{R}^{d\times d}$. □

Theorem 1.34. Let X be the d-dimensional Ornstein-Uhlenbeck process solving the stochastic initial value problem 6. Then X is a Markov process.

Proof. From Theorem 1.33 we know that the Ornstein-Uhlenbeck process is a unique strong solution of the stochastic initial value problem 6. Then with Theorem 21.15 [Pfa] follows that X is a Markov process. □

In the following our goal is to find a suitable multidimensional explicit representation of the d-dimnesional Ornstein-Uhlenbeck process. If the stiffness matrix Σ is a diagonal matrix with nonvanishing entries this explicit multidimensional representation should correspond to the component-wise one-dimensional explicit representation 5. For this purpose let us now show the following theorem.

Theorem 1.35 (Explicit representation of a multidimensional Ornstein-Uhlenbeck process)**.** Let X be a d-dimensional Ornstein-Uhlenbeck process with initial value $x_0 \in \mathbb{R}^d$, equilibrium level $\mu \in \mathbb{R}^d$, invertible stiffness matrix $\Sigma \in \mathbb{R}^{d\times d}$ and diagonal dispersion matrix $\boldsymbol{\sigma} \in \mathbb{R}^{d\times d}$. Then the solution X of the stochastic initial value problem 6 is explicitly given by

$$X_t = e^{-\Sigma t}x_0 + \mu - e^{-\Sigma t}\mu + \int_0^t e^{-\Sigma(t-s)}\boldsymbol{\sigma}\mathrm{d}W_s. \tag{7}$$

For a matrix Σ the *matrix exponential* $exp : \mathbb{R}^{d\times d} \to \mathbb{R}^{d\times d}$ is given by the series representation

$$exp(\Sigma) = \sum_{n=0}^{\infty} \frac{\Sigma^n}{n!},$$

where Σ^0 corresponds to the unit matrix E_d. In order to prove Theorem 1.35 we have to look at a few more rules of calculation. If $A, B \in \mathbb{R}^{d\times d}$ are matrices that commute, i.e. $AB = BA$, we have the following rule of calculation

$$exp(A + B) = exp(A) \cdot exp(B),$$

because

$$\begin{aligned} exp(A+B) &= \sum_{n=0}^{\infty} \frac{(A+B)^n}{n!} \\ &\underset{\text{thm.}}{\overset{\text{binom.}}{=}} \sum_{n=0}^{\infty}\sum_{k=0}^{n} \frac{\binom{n}{k}}{n!} A^k B^{n-k} \\ &= \sum_{n=0}^{\infty}\sum_{k=0}^{n} \frac{A^k}{k!}\frac{B^{n-k}}{(n-k)!} \end{aligned}$$

$$\begin{aligned} &\overset{\text{Cauchy}}{\underset{\text{prod.}}{=}} \left(\sum_{n=0}^{\infty} \frac{A^n}{n!}\right) \cdot \left(\sum_{n=0}^{\infty} \frac{B^n}{n!}\right) \\ &= \quad exp(A) \cdot exp(B) \end{aligned}$$

holds. Since the matrix A obviously commutates with $-A$

$$e^A e^{-A} = E_d = e^{-A} e^A$$

holds, where E_d is the d-dimensional unit matrix. The entries $exp(A)_{ij}$ of the matrix $exp(A)$ for $i, j = 1, \ldots, d$ are represented by

$$exp(A)_{ij} = \left(\sum_{n=0}^{\infty} \frac{A^n}{n!}\right)_{ij} = \sum_{n=0}^{\infty} \frac{A_{ij}^n}{n!} = \delta_{ij} + \sum_{n=1}^{\infty} \frac{A_{ij}^n}{n!}. \tag{8}$$

Here δ_{ij} stands for the Kronecker delta and by A_{ij}^n we refer to the ij-th entry of the matrix A^n.

Proof of Theorem 1.35. The stochastic initial value problem 6 for all $i = 1, \ldots, d$ can be written as

$$\begin{aligned} \mathrm{d}X_t^i &= (\Sigma \cdot (\mu - X_t))_i \mathrm{d}t + \boldsymbol{\sigma}_{ii} \mathrm{d}W_t^i = \sum_{j=1}^{d} \Sigma_{ij}(\mu_j - X_t^j)\mathrm{d}t + \boldsymbol{\sigma}_{ii}\mathrm{d}W_t^i \\ X_0^i &= x_0^i, \end{aligned} \tag{9}$$

where $X = (X^1, \ldots, X^d)$. The proof of this theorem is analogous to the proof in the one-dimensional case where we can apply Itô's lemma 1.23 component-wise and reassemble the equation at the end. For this purpose for $i = 1, \ldots, d$ we consider the functions $f^i : \mathbb{R}^d \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ defined by

$$f^i(X_t, t) = (e^{\Sigma t} X_t)_i = \sum_{j=1}^{d} e_{ij}^{\Sigma t} X_t^j.$$

The partial derivatives in the direction X^k are calculated for $k = 1, \ldots, d$ by

$$\partial_{X^k} f^i(X_t, t) = \partial_{X^k} \sum_{j=1}^{d} e_{ij}^{\Sigma t} X_t^j = \sum_{j=1}^{d} e_{ij}^{\Sigma t} \underbrace{\partial_{X^k} X_t^j}_{=\delta_{kj}} = e_{ik}^{\Sigma t}. \tag{10}$$

We see that

$$\partial_{X^l X^k} f^i(X_t, t) = 0$$

holds for all $l = 1, \ldots, d$. Further, we calculate

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t} e^{\Sigma t} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\sum_{n=0}^{\infty} \frac{\Sigma^n t^n}{n!}\right) \\ &= \sum_{n=1}^{\infty} \frac{\Sigma^n t^{n-1}}{(n-1)!} \\ &= e^{\Sigma t}\Sigma,\end{aligned}$$

therefore,

$$\partial_t e^{\Sigma t} X_t = e^{\Sigma t}\Sigma X_t$$

is valid. Thus, the partial derivative in direction t is

$$\partial_t f^i(X_t, t) = (e^{\Sigma t}\Sigma X_t)_i. \tag{11}$$

Further, the equation

$$\begin{aligned}\sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \sum_{j=1}^{d} \Sigma_{kj} X_s^j \mathrm{d}s &= \int_0^t \sum_{j=1}^{d} \left(\sum_{k=1}^{d} e_{ik}^{\Sigma s}\Sigma_{kj}\right) X_s^j \mathrm{d}s \\ &= \int_0^t \sum_{j=1}^{d} (e^{\Sigma s}\Sigma)_{ij} X_s^j \mathrm{d}s \\ &= \int_0^t (e^{\Sigma s}\Sigma X_s)_i \mathrm{d}s.\end{aligned} \tag{12}$$

holds. We calculate

$$\begin{aligned}\int_0^t e_{ik}^{\Sigma s}\mathrm{d}s &\overset{8}{=} \int_0^t \sum_{n=0}^{\infty} \frac{\Sigma_{ik}^n s^n}{n!}\mathrm{d}s \\ &\overset{\text{Fub.}}{=} \sum_{n=0}^{\infty} \frac{\Sigma_{ik}^n}{n!} \int_0^t s^n \mathrm{d}s \\ &= \sum_{n=0}^{\infty} \frac{\Sigma_{ik}^n t^{n+1}}{(n+1)!}\end{aligned}$$

$$
\begin{aligned}
&= \sum_{n=0}^{\infty} (\Sigma^{n+1}\Sigma^{-1})_{ik} \frac{t^{n+1}}{(n+1)!} \\
&= \sum_{n=0}^{\infty} \sum_{m=1}^{d} \Sigma_{im}^{n+1} \Sigma_{mk}^{-1} \frac{t^{n+1}}{(n+1)!} \\
&= \sum_{m=1}^{d} \left(\sum_{n=0}^{\infty} \Sigma_{im}^{n+1} \frac{t^{n+1}}{(n+1)!} \right) \Sigma_{mk}^{-1} \\
&= \sum_{m=1}^{d} \left(\sum_{n=0}^{\infty} \frac{\Sigma_{im}^{n} t^{n}}{n!} - \delta_{im} \right) \Sigma_{mk}^{-1} \\
&\overset{8}{=} \sum_{m=1}^{d} \left(e_{im}^{\Sigma t} - \delta_{im} \right) \Sigma_{mk}^{-1} \\
&= (e^{\Sigma t}\Sigma^{-1})_{ik} - \Sigma_{ik}^{-1},
\end{aligned}
$$

from this follows

$$
\begin{aligned}
\int_0^t (e^{\Sigma s}\Sigma\mu)_i \mathrm{d}s &= \int_0^t \sum_{k=1}^{d} e_{ik}^{\Sigma s} (\Sigma\mu)_k \mathrm{d}s \\
&= \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \mathrm{d}s (\Sigma\mu)_k \\
&= \sum_{k=1}^{d} \left((e^{\Sigma t}\Sigma^{-1})_{ik} - \Sigma_{ik}^{-1} \right) (\Sigma\mu)_k \\
&= (e^{\Sigma t}\Sigma^{-1}\Sigma\mu)_i - (\Sigma^{-1}\Sigma\mu)_i \\
&= (e^{\Sigma t}\mu)_i - \mu_i.
\end{aligned}
\tag{13}
$$

With the help of these preliminary considerations and by applying Itô's lemma 1.23 for all $i = 1, \ldots, d$ we get

$$
\begin{aligned}
(e^{\Sigma t} X_t)_i &= f^i(X_t, t) \\
&\overset{1.23}{=} f^i(X_0, 0) + \sum_{k=1}^{d} \int_0^t \partial_{X^k} f^i(X_s, s) \mathrm{d}X_s^k + \int_0^t \partial_t f^i(X_s, s) \mathrm{d}s \\
&\underset{11}{\overset{10}{=}} X_0^i + \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \mathrm{d}X_s^k + \int_0^t (e^{\Sigma s}\Sigma X_s)_i \mathrm{d}s
\end{aligned}
$$

$$
\begin{aligned}
&\overset{9}{=} X_0^i + \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \left(\sum_{j=1}^{d} \Sigma_{kj}(\mu_j - X_s^j)\mathrm{d}s + \boldsymbol{\sigma}_{kk}\mathrm{d}W_s^k \right) + \int_0^t (e^{\Sigma s}\Sigma X_s)_i \mathrm{d}s \\
&= X_0^i + \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \sum_{j=1}^{d} \Sigma_{kj}\mu_j \mathrm{d}s - \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \sum_{j=1}^{d} \Sigma_{kj} X_s^j \mathrm{d}s \\
&\quad + \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \boldsymbol{\sigma}_{kk}\mathrm{d}W_s^k + \int_0^t (e^{\Sigma s}\Sigma X_s)_i \mathrm{d}s \\
&\overset{12}{=} X_0^i + \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \sum_{j=1}^{d} \Sigma_{kj}\mu_j \mathrm{d}s + \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \boldsymbol{\sigma}_{kk}\mathrm{d}W_s^k \\
&= X_0^i + \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} (\Sigma\mu)_k \mathrm{d}s + \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \boldsymbol{\sigma}_{kk}\mathrm{d}W_s^k \\
&= X_0^i + \int_0^t (e^{\Sigma s}\Sigma\mu)_i \mathrm{d}s + \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \boldsymbol{\sigma}_{kk}\mathrm{d}W_s^k \\
&\overset{13}{=} X_0^i + (e^{\Sigma t}\mu)_i - \mu_i + \sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \boldsymbol{\sigma}_{kk}\mathrm{d}W_s^k.
\end{aligned}
$$

In the second equation we used that the second partial derivatives always vanish. This is valid because all partial derivatives not directed to t vanish according to the initial remark and for the second partial derivatives in direction t the integrator vanishes. If we now put the above equations together we get

$$
e^{\Sigma t} X_t = X_0 + e^{\Sigma t}\mu - \mu + \left(\sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \boldsymbol{\sigma}_{kk}\mathrm{d}W_s^k \right)_{i=1,\ldots,d}.
$$

If we multiply this equation from the left by $e^{-\Sigma t}$ we obtain

$$
\begin{aligned}
X_t &= e^{-\Sigma t} X_0 + \mu - e^{-\Sigma t}\mu + e^{-\Sigma t} \left(\sum_{k=1}^{d} \int_0^t e_{ik}^{\Sigma s} \boldsymbol{\sigma}_{kk}\mathrm{d}W_s^k \right)_{i=1,\ldots,d} \\
&= e^{-\Sigma t} X_0 + \mu - e^{-\Sigma t}\mu + \left(\sum_{j=1}^{d} e_{ij}^{-\Sigma t} \sum_{k=1}^{d} \int_0^t e_{jk}^{\Sigma s} \boldsymbol{\sigma}_{kk}\mathrm{d}W_s^k \right)_{i=1,\ldots,d}
\end{aligned}
$$

$$
\begin{aligned}
&= e^{-\Sigma t}X_0 + \mu - e^{-\Sigma t}\mu + \left(\sum_{k=1}^{d}\int_0^t \sum_{j=1}^{d} e_{ij}^{-\Sigma t} e_{jk}^{\Sigma s}\boldsymbol{\sigma}_{kk}\mathrm{d}W_s^k\right)_{i=1,\dots,d} \\
&= e^{-\Sigma t}X_0 + \mu - e^{-\Sigma t}\mu + \left(\sum_{k=1}^{d}\int_0^t e_{ik}^{-\Sigma(t-s)}\boldsymbol{\sigma}_{kk}\mathrm{d}W_s^k\right)_{i=1,\dots,d} \\
&= e^{-\Sigma t}X_0 + \mu - e^{-\Sigma t}\mu + \int_0^t e^{-\Sigma(t-s)}\boldsymbol{\sigma}\mathrm{d}W_s.
\end{aligned}
$$

By setting $X_0 \equiv x_0$ we are done with the proof. □

We note that in the case of independent components of a d-dimensional Ornstein-Uhlenbeck process, the explicit representation 7 of the process agrees component-wise with the one-dimensional explicit representation 5. Due to this representation of the Ornstein-Uhlenbeck process we obtain the following theorem.

Theorem 1.36. Let X be a d-dimensional Ornstein-Uhlenbeck process as described in Theorem 1.35. The process X_t is then multivariate normally distributed with mean vector

$$M_{x_0}(t) = e^{-\Sigma t}x_0 + (E_d - e^{-\Sigma t})\mu$$

and covariance matrix

$$\omega(t) = \int_0^t e^{-\Sigma(t-s)}\boldsymbol{\sigma}^2 e^{-\Sigma^{\mathrm{T}}(t-s)}\mathrm{d}s.$$

Proof. By Theorem 1.35 we note that X_t is normally distributed because the integrand of the integral corresponding to the Brownian motion W is deterministic. By the explicit representation of X_t we note that

$$
\begin{aligned}
\mathbb{E}[X_t] &= \mathbb{E}\left[e^{-\Sigma t}x_0 + \mu - e^{-\Sigma t}\mu + \int_0^t e^{-\Sigma(t-s)}\boldsymbol{\sigma}\mathrm{d}W_s\right] \\
&= e^{-\Sigma t}x_0 + \mu - e^{-\Sigma t}\mu + \underbrace{\mathbb{E}\left[\int_0^t e^{-\Sigma(t-s)}\boldsymbol{\sigma}\mathrm{d}W_s\right]}_{=0} \\
&= e^{-\Sigma t}x_0 + \mu - e^{-\Sigma t}\mu
\end{aligned}
$$

holds and by

$$X_t - \mathbb{E}[X_t] = \int_0^t e^{-\Sigma(t-s)} \boldsymbol{\sigma} \mathrm{d}W_s$$

and the Itô-isometry (Lemma 3.1.5 [Oks02]) follows

$$\mathbb{V}[X_t] = \mathbb{E}[(X_t - \mathbb{E}[X_t])^2] = \mathbb{E}\left[\left(\int_0^t e^{-\Sigma(t-s)} \boldsymbol{\sigma} \mathrm{d}W_s\right)^2\right] = \int_0^t e^{-\Sigma(t-s)} \boldsymbol{\sigma}^2 e^{-\Sigma^{\mathrm{T}}(t-s)} \mathrm{d}s.$$

□

In the following section we will look at an alternative way to determine the distribution of an Ornstein-Uhlenbeck process. For this purpose we consider the Fokker-Planck equation.

1.3 The Fokker-Planck Equation

In this section we want to introduce the Fokker-Planck equation for stochastic differential equations. This equation describes the time evolution of the probability density function of the stochastic differential equations solution. We apply this equation to an Ornstein-Uhlenbeck process and be able to calculate its probability density function explicitly.

Definition 1.37 (Fokker-Planck equation for stochastic differential equations)**.** Let a stochastic differential equation

$$\mathrm{d}X_t = b(t, X_t)\mathrm{d}t + \boldsymbol{\sigma}(t, X_t)\mathrm{d}W_t$$

for a d-dimensional drift vector, a $d \times r$-dimensional dispersion matrix $\boldsymbol{\sigma}$ and an r-dimensional Brownian motion W be given. Then the probability density function P_t for X_t satisfies the *Fokker-Planck equation*

$$\frac{\partial}{\partial t}P_t(x) = -\frac{\partial}{\partial x}b(t,x)P_t(x) + \frac{\partial^2}{\partial x^2}D(t,x)P_t(x)$$

for the *diffusion tensor* $D = \frac{1}{2}\boldsymbol{\sigma}\boldsymbol{\sigma}^{\mathrm{T}}$.

We now want to apply the Fokker-Planck equation to the Ornstein-Uhlenbeck process from Definition 1.32 and determine its probability density function. In this section we rely on section 6.5 of [Ris96] and [VP19].

Corollary 1.38 (Fokker-Planck equation of an Ornstein-Uhlenbeck process)**.** Let X be a d-dimensional Ornstein-Uhlenbeck process solving the following stochastic inital value problem

$$\begin{aligned}\mathrm{d}X_t &= \Sigma \cdot (\mu - X_t)\mathrm{d}t + \boldsymbol{\sigma}\mathrm{d}W_t \\ X_{t_0} &= x_0.\end{aligned}$$

where $x_0 \in \mathbb{R}^d$ is the initial value at time $t_0 \geq 0$, $\Sigma \in \mathbb{R}^{d\times d}$ is an invertible matrix, $\mu \in \mathbb{R}^d$ is a vector and $\boldsymbol{\sigma} \in \mathbb{R}^{d\times d}$ is a diagonal matrix. We assume that Σ is a diagonal matrix with nonvanishing diagonal entries. Therefore, the eigenvalues $\lambda_1 \ldots, \lambda_d$ of Σ can be read directly from the diagonal. The probability density function of the Ornstein-Uhlenbeck process can be described by the Fokker-Planck equation

$$\frac{\partial}{\partial t}P_t(x) = -\left(\frac{\partial}{\partial x}\Sigma(\mu - x)P_t(x)\right) + \frac{\partial^2}{\partial x^2}\frac{\boldsymbol{\sigma}^2 P_t(x)}{2}$$

with initial condition[4]

$$P_{t_0}(x|x_0) = \mathbb{1}_{\{x=x_0\}}(x).$$

In the following theorem we note that an Ornstein-Uhlenbeck process is normally distributed.

Theorem 1.39 (Distribution of an Ornstein-Uhlenbeck process)**.** Let X be a d-dimensional Ornstein-Uhlenbeck process satisfying the stochastic initial value problem described in corollary 1.38. Then for the stochastic process X every X_t for $t \geq t_0$ has a d-dimensional normal distribution with mean vector

$$M_{x_0}(t - t_0) = e^{-\Sigma(t-t_0)}x_0 + (E_d - e^{-\Sigma(t-t_0)})\mu \tag{14}$$

and covariance matrix

$$\omega(t - t_0) = \int_{t_0}^{t} e^{-\Sigma(t-s)}\boldsymbol{\sigma}^2 e^{-\Sigma^{\mathrm{T}}(t-s)}\mathrm{d}s. \tag{15}$$

Proof. This theorem can be shown by calculating the characteristic function of the Ornstein-Uhlenbeck process by taking the Fourier transformation of the associated Fokker-Planck equation and applying the method of characteristics. A detailed prove can be found in Theorem 1 [VP19]. □

To compute the vector M and the matrix ω more precisely we need to find a complete biorthogonal set[5] of the matrix Σ, i.e. we need to find column vectors $u_{\lambda_1}, \ldots, u_{\lambda_d}$ and row vectors $v_{\lambda_1}, \ldots, v_{\lambda_d}$ such that

$$\Sigma u_{\lambda_i} = \lambda_i u_{\lambda_i} \qquad \text{and} \qquad v_{\lambda_i}\Sigma = \lambda_i v_{\lambda_i}$$

holds for all $i = 1, \ldots, d$ and the orthonormality and completeness relation

$$\left(\sum_{k=1}^{d} v^i_{\lambda_k} u^j_{\lambda_k}\right)_{i,j=1,\ldots,d} = E_d \qquad \text{and} \qquad \left(u^i_{\lambda_j} v^i_{\lambda_k}\right)_{j,k=1,\ldots,d} = E_d$$

holds for all $i = 1, \ldots, d$. Such a set exists if all eigenvectors $\lambda_1, \ldots, \lambda_d$ are different. With loss of generality we can assume that they are different. Otherwise, we can add the individual

[4] Eq. 6.109 [Ris96] respectively eq. 7 [VP19].

[5] This ansatz originates on the part from [Ris96] around equations 6.119 to 6.123 where the notation is adapted to our setting and we calculate the individual terms directly.

values in each case with an ε and take the limit $\varepsilon \to 0$ at the end. With our assumption that Σ is a diagonal matrix we notice that this set just consists of the standard normal basis of $\mathbb{R}^d$, i.e. $u_{\lambda_i} = e_i$ and $v_{\lambda_i} = e_i^{\mathrm{T}}$ for all $i = 1, \ldots, d$. The spectral decomposition of the matrix Σ then reads

$$\Sigma = \left(\sum_{k=1}^{d} \lambda_k u_{\lambda_k}^i v_{\lambda_k}^j \right)_{i,j=1,\ldots,d} = \left(\sum_{k=1}^{d} \lambda_k \delta_{ik} \delta_{kj} \right)_{i,j=1,\ldots,d} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{pmatrix}$$

what we expect. We now have

$$e^{-\Sigma t} = \left(\sum_{k=1}^{d} e^{-\lambda_k t} u_{\lambda_k}^i v_{\lambda_k}^j \right)_{i,j=1,\ldots,d} = \left(\sum_{k=1}^{d} e^{-\lambda_k t} \delta_{ik} \delta_{kj} \right)_{i,j=1,\ldots,d} = \begin{pmatrix} e^{-\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{-\lambda_2 t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{-\lambda_d t} \end{pmatrix}.$$

By inserting this expression in 14 and 15 we get the mean vector and can perform the integration and obtain

$$\begin{aligned} M_{x_0}(t - t_0) &= e^{-\Sigma(t-t_0)} x_0 + (E_d - e^{-\Sigma(t-t_0)})\mu \\ &= \begin{pmatrix} e^{-\lambda_1(t-t_0)} & 0 & \cdots & 0 \\ 0 & e^{-\lambda_2(t-t_0)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{-\lambda_d(t-t_0)} \end{pmatrix} x_0 \\ &\quad + \begin{pmatrix} 1 - e^{-\lambda_1(t-t_0)} & 0 & \cdots & 0 \\ 0 & 1 - e^{-\lambda_2(t-t_0)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 - e^{-\lambda_d(t-t_0)} \end{pmatrix} \mu \\ &= \begin{pmatrix} e^{-\lambda_1(t-t_0)}((x_0)_1 - \mu_1) + \mu_1 \\ \vdots \\ e^{-\lambda_d(t-t_0)}((x_0)_d - \mu_d) + \mu_d \end{pmatrix} \end{aligned}$$

and[6]

$$\begin{aligned}\omega(t-t_0)_{i,j} &= 2\sum_{k,l=1}^{d}\frac{1-e^{-(\lambda_k+\lambda_l)(t-t_0)}}{\lambda_k+\lambda_l}v_{\lambda_k}\underbrace{\frac{\boldsymbol{\sigma}^2}{2}}_{\text{diag.mat.}}v_{\lambda_l}^{\mathrm{T}}\underbrace{u_{\lambda_k}^i}_{=\delta_{ik}}\underbrace{u_{\lambda_l}^j}_{=\delta_{jl}}\\ &= \sum_{k,l=1}^{d}\frac{1-e^{-(\lambda_k+\lambda_l)(t-t_0)}}{\lambda_k+\lambda_l}\delta_{kl}\boldsymbol{\sigma}_{kl}^2\delta_{ik}\delta_{jl}\\ &= \frac{1-e^{-2\lambda_i(t-t_0)}}{2\lambda_i}\boldsymbol{\sigma}_{ii}^2\delta_{ij}\end{aligned}$$

which can be read as

$$\omega(t-t_0) = \begin{pmatrix}\frac{1-e^{-2\lambda_1(t-t_0)}}{2\lambda_1}\boldsymbol{\sigma}_{11}^2 & 0 & \cdots & 0\\ 0 & \frac{1-e^{-2\lambda_2(t-t_0)}}{2\lambda_2}\boldsymbol{\sigma}_{22}^2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \frac{1-e^{-2\lambda_d(t-t_0)}}{2\lambda_d}\boldsymbol{\sigma}_{dd}^2\end{pmatrix}.$$

Thus, we obtain that the probability density function P_{t,t_0} or transition probability $P_{t,t_0}(x|x_0)$ of x with given initial value x_0 at time t_0 can be represented by

$$P_{t,t_0}(x|x_0) = \frac{1}{\sqrt{(2\pi)^d det(\omega(t-t_0))}}\exp\left(-\frac{1}{2}(x-M_{x_0}(t-t_0))^{\mathrm{T}}\omega^{-1}(t-t_0)(x-M_{x_0}(t-t_0))\right). \tag{16}$$

See also Equation 6.124 [Ris96].

We recall from Theorem 1.34 that the Ornstein-Uhlenbeck process is a Markov process. We now have calculated the probability density function of the process with the initial value x_0 at time t_0. Thus, the probability density function for the time difference $t-t_0$ is given for all $t \geq t_0$. This means that this probability density function depends only on the time difference $t-t_0$ but not on the time points themselves. Consequently, the Ornstein-Uhlenbeck process is a time-homogeneous Markov process described as in Definition 1.14.

Finally, we summarize the previous results in the following corollary.

[6] In this calculation δ is the Kronecker delta.

Corollary 1.40 (Distribution of an Ornstein-Uhlenbeck process)**.** Let X be a d-dimensional Ornstein-Uhlenbeck process with initial time $t_0 \geq 0$ solving the following stochastic inital value problem

$$\begin{aligned} \mathrm{d}X_t &= \Sigma \cdot (\mu - X_t)\mathrm{d}t + \boldsymbol{\sigma}\mathrm{d}W_t \\ X_{t_0} &= x_0 \end{aligned}$$

where $\Sigma \in \mathbb{R}^{d\times d}$ is an invertible matrix, $\mu, x_0 \in \mathbb{R}^d$ are vectors and $\boldsymbol{\sigma} \in \mathbb{R}^{d\times d}$ is a diagonal matrix. We assume that Σ is a diagonal matrix with nonvanishing diagonal entries and eigenvalues $\lambda_1, \ldots, \lambda_d$.
Then we obtain that X is an time-homogeneous Markov process and X_t is for all $t \geq t_0$ normally distributed with mean vector

$$M_{x_0}(\delta_t) = e^{-\Sigma\delta_t}x_0 + (E_d - e^{-\Sigma\delta_t})\mu = \begin{pmatrix} e^{-\lambda_1\delta_t}((x_0)_1 - \mu_1) + \mu_1 \\ \vdots \\ e^{-\lambda_d\delta_t}((x_0)_d - \mu_d) + \mu_d \end{pmatrix}$$

and covariance matrix

$$\omega(\delta_t) = \begin{pmatrix} \frac{1-e^{-2\lambda_1\delta_t}}{2\lambda_1}\boldsymbol{\sigma}_{11}^2 & 0 & \cdots & 0 \\ 0 & \frac{1-e^{-2\lambda_2\delta_t}}{2\lambda_2}\boldsymbol{\sigma}_{22}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1-e^{-2\lambda_d\delta_t}}{2\lambda_d}\boldsymbol{\sigma}_{dd}^2 \end{pmatrix},$$

where $\delta_t := t - t_0$. Thus, the probability density function P_t or transition probability $P_{\delta_t}(x|x_0)$ of x at time t with given initial value x_0 at time t_0 can be represented by

$$P_{\delta_t}(x|x_0) = \frac{1}{\sqrt{(2\pi)^d det(\omega(\delta_t))}} \exp\left(-\frac{1}{2}(x - M_{x_0}(\delta_t))^{\mathrm{T}}\omega^{-1}(\delta_t)(x - M_{x_0}(\delta_t))\right).$$

The normal distribution of the Ornstein-Uhlenbeck process that we determined with this corollary is illustrated in Figure 5.

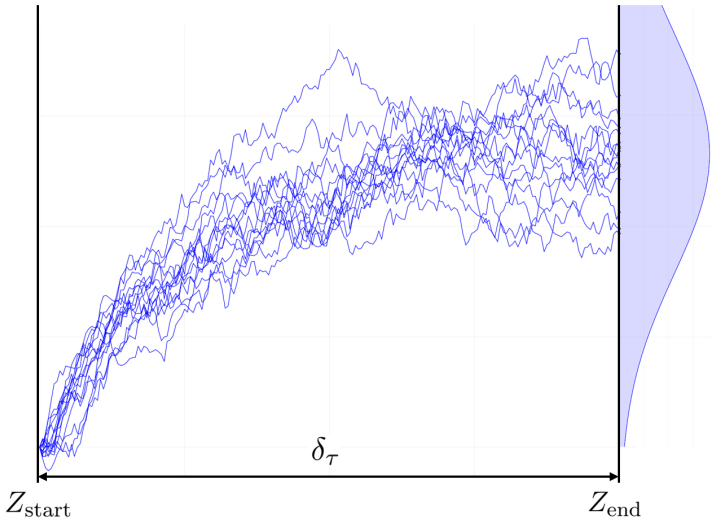


Figure 5: Schematic representation of the normal distribution of an Ornstein-Uhlenbeck process.

2 Methods

Clinical cohort studies that investigate diseases are intended to record and to evaluate the health status of the participants. In the course of such studies, measurements of the individual participants are typically taken at different points in time. These measurements exhibit certain latent patterns of development over time, depending on the disease state. We assume that this development can be described in terms of an Ornstein-Uhlenbeck process. Once this process is parameterized, predictions about the future course of the patient's disease can be determined. However, for each patient the development of the condition over time may depend on patient-specific characteristics. Since these characteristics influence the condition it is recommended that baseline variables are collected during the study in addition to the values measured over time. The development of the condition of patients whose variables have similarities can be similar. As a consequence, patients can be divided into different groups. A naive example is the division of study participants into those who are healthy and those who suffer from the disease being studied. The goal in analyzing such data is, on the one hand, to parameterize the Ornstein-Uhlenbeck process of each individual using the baseline variables and, on the other hand, to learn implicitly the group structure. This setting is motivated by the SMArtCARE dataset which contains data on patients suffering from spinal muscular atrophy. Patients can be classified into different groups by their age and severity of their symptoms. Further, the patients' measurements consisting of data on motor skills show certain individual latent developmental patterns based on the associated group. Additionally, these time series data present a highly temporal irregularity.

In this chapter we first specify the general setting for which we will develop the methods. Three different methods will be introduced to analyze the data described in the setting. The knowledge gained from the application of each of these methods is incorporated in the subsequent methods. The first method approximates the latent process based on the Euler-Maruyama method 1.28. The second method focuses on learning the drift and diffusion parameters of the process using various loss functions. The last method exploits the properties of an Ornstein-Uhlenbeck process mentioned in 1.3 in order to model its probability density function.

2.1 Setting

In this section we introduce the general setting to which our developed methods will be applied.

Let $N \in \mathbb{N}$ be the number of patients and some timepoints $0 < \tau_1^i \leq \ldots \leq \tau_{n_i}^i < \infty$ are given for $n_i \in \mathbb{N}_{\geq 2}$ and $i = 1, \ldots, N$. We refer to the individual first time τ_1^i as *study entry*. Now let $\mathcal{X} = (\mathcal{B}, \mathcal{M})$ be the data set where $\mathcal{B} = \{B^1, \ldots, B^N\}$ denotes the *set of baseline variables* with individual *baseline variables* $B^i \in \mathbb{R}^{n_\mathcal{B}}$ for every individual $i \in \{1, \ldots, N\}$ and $\mathcal{M} = \{M^1, \ldots, M^N\}$ denotes the *set of the series of measurements* with individual series of measurements $M^i \in \mathbb{R}^{D \times n_i}$ for every individual $i \in \{1, \ldots, N\}$. Here $n_\mathcal{B} \in \mathbb{N}$ describes the number of *baseline variables* per patient and $D \in \mathbb{N}$ the number of variables measured over time. By $M_j^i \in \mathbb{R}^D$ we denote the measured values of the i-th patient at time τ_j^i by the j-th column of M^i for all $j \in \{1, \ldots, n_i\}$ and $i \in \{1, \ldots, N\}$.
Further, let $\Phi_\mathcal{M} \in \mathcal{NN}(S_\mathcal{M})$ be a neural network with architecture $S_\mathcal{M} := (D, D, d)$ and realization

$$R(\Phi_\mathcal{M})(M_j^i) = Z_j^i \in \mathbb{R}^d$$

for all $j = 1, \ldots, n_i$ and $i \in \{1, \ldots, N\}$ where $d \in \mathbb{N}$ with $d \ll D$.
By $\mathcal{Z} = \{Z^1, \ldots, Z^N\}$ we denote the *set of encoded states* with individual *encoded states* $Z^i \in \mathbb{R}^{d \times n_i}$ for every individual $i \in \{1, \ldots, N\}$. Analogously to the definition of M_j^i we define by $Z_j^i \in \mathbb{R}^d$ the latent measurements of the i-th patient at time τ_j^i for all $j \in \{1, \ldots, n_i\}$ and $i \in \{1, \ldots, N\}$.
We now assume that the *latent state* $Z^i = (Z_t^i)_{t \geq 0}$ of the i-th patient for $i \in \{1, \ldots, N\}$ can be described in terms of a d-dimensional Ornstein-Uhlenbeck process with unobserved initial state $Z_0^i \in \mathbb{R}^d$. Furthermore, we assume that the process at times τ_j^i just corresponds to the encoded states Z_j^i, i.e. it is $Z_{\tau_j^i}^i = Z_j^i$ for $j \in \{1, \ldots, n_i\}$. Thus, the process for all $i = 1, \ldots, N$ solves the stochastic initial value problem

$$\begin{aligned} \mathrm{d}Z_t^i &= \Sigma^i \cdot (\mu^i - Z_t^i)\mathrm{d}t + \boldsymbol{\sigma}^i \mathrm{d}W_t \\ Z_0^i &= Z_0^i, \end{aligned} \tag{17}$$

where $\mu^i \in \mathbb{R}^d$ is a vector, $\Sigma^i \in \mathbb{R}^{d \times d}$ is an invertible diagonal matrix, $\boldsymbol{\sigma}^i \in \mathbb{R}^{d \times d}$ is a diagonal matrix and W is a d-dimensional Brownian motion. Applying the Euler-Maruyama method

1.28 we obtain that the process Z^i can be approximated by the difference equation

$$Z^i_{j+1} = Z^i_j + \Sigma^i \mu^i - \Sigma^i Z^i_j + \varepsilon^i$$

for all $j = 1, \ldots, n_i - 1$ with initial condition $Z^i_{\tau^i_1} = Z^i_1$. The time points were chosen equidistant with step width 1. Here $\varepsilon^i = (\varepsilon^i_1, \ldots, \varepsilon^i_d)$ are indipendent random variables with $\varepsilon^i_l \sim \mathcal{N}(0, (\boldsymbol{\sigma}^i_{ll})^2)$ for all $l = 1, \ldots, d$. Let $\sigma^i = (\boldsymbol{\sigma}^i_{11}, \ldots, \boldsymbol{\sigma}^i_{dd})$ then

$$\varphi^i := (\mu^i, \Sigma^i, \sigma^i) \in \mathbb{R}^d \times \mathbb{R}^{d \times d} \times \mathbb{R}^d_+$$

describes the parameter of the difference equation and stochastic differential equation, respectively. We assume that these parameters can be modeled by the realization the neural network $\Phi_{\mathcal{B}} \in \mathcal{NN}(S_{\mathcal{B}})$ with architecture $S_{\mathcal{B}} := (n_{\mathcal{B}}, n_{\mathcal{B}}, 3d, 3d)$ such that

$$R(\Phi_{\mathcal{B}})(B^i) = \varphi^i$$

holds for all $i \in \{1, \ldots, N\}$.

We can summarize the assumptions of our setting as follows. For each patient an individual time series of measured values is given. These measurements were taken at individual specific time points. Additionally, the baseline variables are given for each individual. The series of measured values has a lower dimensional representation for each patient which can be described in terms of an Ornstein-Uhlenbeck process. This process can be parameterized using the baseline variables.
We will use the methods developed in the following sections to implement this setting. These methods have emerged in the course of this thesis and have constantly evolved. They each differ fundamentally in their approach. Each method has its strengths and weaknesses. The main goal of the methods is to determine the parameters of the Ornstein-Uhlenbeck process in the latent space based on the baseline variables. We will see later in the results that one of the major difficulties is to determine the stochasticity that is included in the data.

2.2 Method: Euler-Maruyama

A first naive approach is to encode the data and approximate the solution of the Ornstein-Uhlenbeck process using the Euler-Maruyama method in the latent space. For this purpose the encoder network is used to obtain the initial value from the series of measurements. Further, the baseline network is used to obtain the parameters of the process from the baseline variables. With these values an approximation of the solution can then be determined.

Let $\mathcal{X}$ be the data set, $0 < \tau_1^i \leq \ldots \leq \tau_{n_i}^i < \infty$ be the individual time points for $n_i \in \mathbb{N}$ and $i = 1, \ldots, N$ and $\mathcal{Z}$ be the latent encoded state as described in section 2.1. We note that the latent state Z^i of an individual i can again be described as an Ornstein-Uhlenbeck process with unobserved initial value Z_0^i. Consequently, it solves the stochastic initial value problem

$$\begin{aligned} \mathrm{d}Z^i &= \Sigma^i(\mu^i - Z^i)\mathrm{d}t + \boldsymbol{\sigma}^i \mathrm{d}W \\ Z_0^i &= Z_0^i \end{aligned}$$

where $\mu^i \in \mathbb{R}^d$ is a vector, $\Sigma^i \in \mathbb{R}^{d\times d}$ is an invertible diagonal matrix, $\boldsymbol{\sigma}^i \in \mathbb{R}^{d\times d}$ is a diagonal matrix and W is a d-dimensional Brownian motion. Since the initial value Z_0^i is unobserved and the data is known only from the individual study entry τ_1^i, the process can be modeled from that time point. The first measured value is given by $Z_{\tau_1}^i$ and Z_1^i, respectively. To obtain a numerical approximate solution we apply the Euler-Maruyama method mentioned in Definition 1.28. For this, let $\delta_j^i := \tau_{j+1}^i - \tau_j^i$ for $j = 1, \ldots, n_i - 1$. Thus, as an approximation of the solution the values $\hat{Z}_2^i, \ldots, \hat{Z}_{n_i}^i$ at the time points $\tau_2^i, \ldots, \tau_{n_i}^i$ with initial value Z_1^i at τ_1^i are given by

$$\begin{aligned} \hat{Z}_{j+1}^i &= \hat{Z}_j^i + \Sigma^i(\mu^i - \hat{Z}_j^i)\delta_j^i + \boldsymbol{\sigma}^i \varepsilon^i \\ \hat{Z}_{\tau_1^i}^i &= Z_1^i \end{aligned}$$

for $j = 1, \ldots, n_i - 1$. The term ε^i is a d-dimensional vector whose entries are sampled from $\mathcal{N}(0, \delta_j^i)$. In order to reconstruct the data series using this approximate solution we introduce the decoder network $\Phi_{\hat{\mathcal{M}}}$ with architecture $S_{\hat{\mathcal{M}}} = (d, D, D)$, such that $\hat{\mathcal{M}}$ is the reconstructed set of series of measurements. Thus, with the neural network $\Phi_{\mathcal{M}}$ we have an autoencoder $AE(\Phi_{\mathcal{M}}, \Phi_{\hat{\mathcal{M}}})$ as described in Definition 1.3. Let $\theta_{\mathcal{M}}$ and $\theta_{\hat{\mathcal{M}}}$ be the parameters belonging to these networks and $\theta_{\mathcal{B}}$ be the parameters belonging to the baseline network $\Phi_{\mathcal{B}}$. We define the parameters θ_{EM} of this model by $\theta_{\mathrm{EM}} = (\theta_{\mathcal{M}}, \theta_{\hat{\mathcal{M}}}, \theta_{\mathcal{B}})$. Then we define the loss function

L_{EM} by the summed column-wise squared error loss

$$\begin{aligned}
L_{\text{EM}}(\theta_{\text{EM}}, M^i) &= \sum_{j=1}^{n_i} \left|\left| M_j^i - \hat{M}_j^i \right|\right|_2^2 \\
&= \sum_{j=2}^{n_i} \left|\left| M_j^i - R(\Phi_{\hat{\mathcal{M}}})(\hat{Z}_j^i) \right|\right|_2^2 \\
&= \sum_{j=2}^{n_i} \left|\left| M_j^i - R(\Phi_{\hat{\mathcal{M}}})(\text{EM}(Z_1^i, \varphi^i)) \right|\right|_2^2 \\
&= \sum_{j=2}^{n_i} \left|\left| M_j^i - R(\Phi_{\hat{\mathcal{M}}})(\text{EM}(R(\Phi_{\mathcal{M}})(M_1^i), R(\Phi_{\mathcal{B}})(B^i))) \right|\right|_2^2
\end{aligned}$$

for every series of measurements $M^i \in \mathcal{M}$ whose j-th column is denoted by $M_j^i \in \mathbb{R}^D$ with the corresponding baseline vector $B^i \in \mathcal{B}$. By $EM(Z_1^i, \varphi^i)$ we denote the application of the above mentioned Euler-Maruyama method using the initial encoded state Z_1^i and the parameters φ^i with corresponding time points $\tau_1^i, \ldots, \tau_{n_i}^i$. We train these networks using the gradient descent algorithm 1 from section 1.1.3. A schematic representation of the model can be found in figure 6.

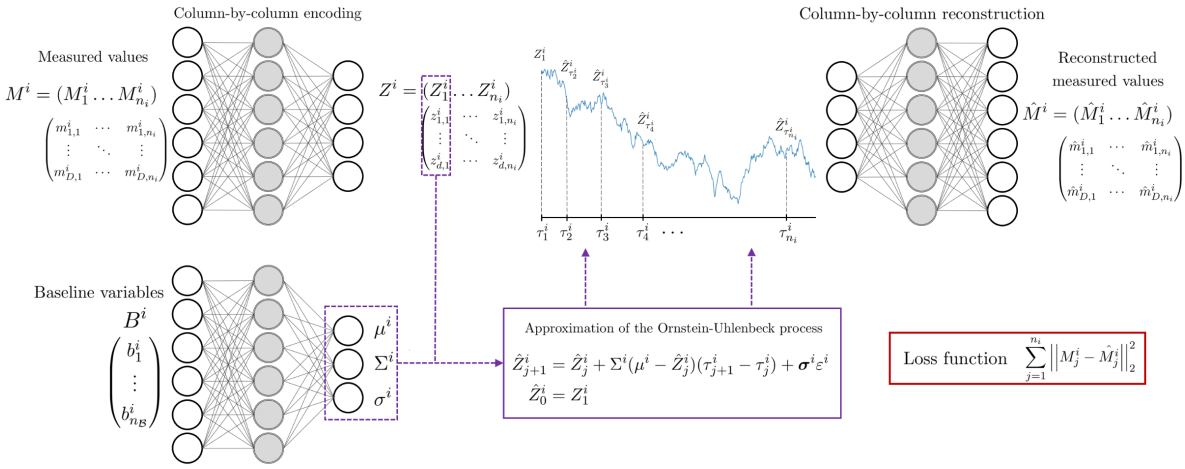


Figure 6: Schematic Representation of the Euler-Maruyama method.

2.3 Method: Covariance Loss

In this method we split the learning of the drift and diffusion parameters of the Ornstein-Uhlenbeck process by finding them by using different terms in the loss function. In addition to the reconstruction loss already presented in the first method, we introduce two new loss functions. In order to learn the drift term we introduce the latent reconstruction loss. The idea of this loss function is that the approximation of the solution determined within the latent space is as close as possible to the states encoded by the encoder network. To learn the diffusion term, we further introduce the covariance loss. The idea hereby is to learn the stochasticity of the data by the covariance of the encoded states.

Let the data set $\mathcal{X}$ and the encoded data $\mathcal{Z}$ be given as described in section 2.1. Again the assumption applies that the latent state Z^i of an individual i can be described in terms of the stochastic initial value problem corresponding to an Ornstein-Uhlenbeck process

$$\begin{aligned} \mathrm{d}Z_t^i &= \Sigma^i \cdot (\mu^i - Z_t^i)\mathrm{d}t + \boldsymbol{\sigma}^i \mathrm{d}W_t \\ Z_0^i &= Z_0^i, \end{aligned}$$

where $\mu^i \in \mathbb{R}^d$ is a vector, $\Sigma^i \in \mathbb{R}^{d\times d}$ is an invertible diagonal matrix, $\boldsymbol{\sigma}^i \in \mathbb{R}^{d\times d}$ is a diagonal matrix and W is a d-dimensional Brownian motion. For simplicity, we will omit the i of the i-th patient in this section and write Z instead of Z^i. The same applies to the parameters and time points. We recall that this initial value problem without the posterior summand is an ordinary differential equation. The posterior summand represents the stochastic influence of the Brownian motion on the solution. Considering the initial value problem without the last summand we get the ordinary initial value problem

$$\begin{aligned} \mathrm{d}Z_t &= \Sigma \cdot (\mu - Z_t)\mathrm{d}t \\ Z_0 &= Z_0. \end{aligned}$$

In Corollary 1.40 we saw that an Ornstein-Uhlenbeck process is normally distributed. Let us now consider the expected value M_{Z_0} of this process that is for all $t \geq 0$ given by

$$M_{Z_0}(t) = e^{-\Sigma t} Z_0 + (E_d - e^{-\Sigma t})\mu.$$

Then

$$\begin{aligned}\frac{\mathrm{d}M_{Z_0}(t)}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(e^{-\Sigma t}Z_0 + (E_d - e^{-\Sigma t})\mu\right) \\ &= -\Sigma e^{-\Sigma t}Z_0 + \Sigma e^{-\Sigma t}\mu \\ &= \Sigma\mu - \Sigma e^{-\Sigma t}Z_0 - \Sigma\mu + \Sigma e^{-\Sigma t}\mu \\ &= \Sigma \cdot (\mu - M_{Z_0}(t))\end{aligned}$$

holds with $M_{Z_0}(0) = Z_0$. This means that the Ornstein-Uhlenbeck in expectation is a solution of the ordinary initial value problem. Since the initial value Z_0 is unobserved and the data is known only from the individual study entry τ_1, the solution can be modeled from that time point. The first encoded value is given by Z_{τ_1} and Z_1, respectively. By using the solution M with given initial encoded value Z_1 we can determine the remaining latent states $\hat{Z}_j$ at times τ_j by

$$\hat{Z}_j = M_{Z_1}(\tau_j - \tau_1)$$

for all $j = 2, \ldots, n_i$. To obtain the reconstructed set of series of measurements $\hat{\mathcal{M}}$ using the latent data series determined in this way we use the decoder network $\Phi_{\hat{\mathcal{M}}}$. Let $\theta_{\mathcal{M}}$ and $\theta_{\hat{\mathcal{M}}}$ be the parameters belonging to the encoder and decoder networks. We define the parameters of the autoencoder consisting of the encoder and decoder network θ_{AE} by $\theta_{\mathrm{AE}} = (\theta_{\mathcal{M}}, \theta_{\hat{\mathcal{M}}})$. In order to train the autoencoder network we introduce the *reconstruction loss*. This loss is based on the loss function described in method 1 but depends on the parameters of the autoencoder. The *reconstruction loss* L_{rec} is defined by the summed up column-wise squared error loss

$$\begin{aligned}L_{\mathrm{rec}}(\theta_{\mathrm{AE}}, M^i) &= \sum_{j=2}^{n_i} \left|\left| M_j^i - \hat{M}_j^i \right|\right|_2^2 \\ &= \sum_{j=2}^{n_i} \left|\left| M_j^i - R(\Phi_{\hat{\mathcal{M}}})(\hat{Z}_j^i) \right|\right|_2^2 \\ &= \sum_{j=2}^{n_i} \left|\left| M_j^i - R(\Phi_{\hat{\mathcal{M}}})(M_{Z_1^i}(\tau_j^i - \tau_1^i)) \right|\right|_2^2\end{aligned}$$

for every series of measurements $M^i \in \mathcal{M}$ whose j-th column is denoted by $M_j^i \in \mathbb{R}^D$ with corresponding baseline vector $B^i \in \mathcal{B}$. To learn the parameters $\theta_{\mathcal{B}}$ of the baseline network $\Phi_{\mathcal{B}}$ we introduce two more loss functions.

In order to learn the parameters of this ordinary differential equation and the drift parameters of the stochastic differential equation, respectively, we introduce the *latent reconstruction loss*. This loss function ensures that the latent data series $\hat{Z}$ determined in the latent space matches the encoded states as closely as possible. We have indicated this condition as $Z_{\tau_j} = Z_j$. The *latent reconstruction loss* L_{latrec} is defined by

$$L_{\text{latrec}}(\theta_{\mathcal{B}}, Z^i) = \sum_{j=2}^{n_i} \left|\left| Z_j - \hat{Z}_{\tau_j} \right|\right|_2^2$$

for all $Z^i \in \mathcal{Z}$ whose j-th column is denoted by $Z_j^i \in \mathbb{R}^d$ with the corresponding baseline vector $B^i \in \mathcal{B}$. In comparison to the reconstruction loss and the loss function from method 1 this loss function uses the entire encoded data series.
Further, to learn the diffusion parameter of the stochastic differential equation we introduce the *covariance loss*. Considering the quadratic covariation process $[Z^k, Z^l]$ of the latent process Z defined in Definition 1.21 we note that with Lemma 1.22 for all $k, l = 1, \ldots, d$ and $t > 0$

$$\sum_{j=2}^{n_i} (Z_{\tau_j}^k - Z_{\tau_{j-1}}^k)(Z_{\tau_j}^l - Z_{\tau_{j-1}}^l) \xrightarrow{n_i \to \infty}_p [Z^k, Z^l]_t$$

holds. Further, according to the properties of the covariation and Z we know that

$$\begin{aligned}
[Z^k, Z^l]_t &= \left[\int_{\tau_1}^t (\Sigma(\mu - Z_s))_k \, ds + \int_{\tau_1}^t \boldsymbol{\sigma}_{kk} \mathrm{d}W_s^k, \int_{\tau_1}^t (\Sigma(\mu - Z_s))_l \, ds + \int_{\tau_1}^t \boldsymbol{\sigma}_{ll} \mathrm{d}W_s^l \right] \\
&= \left[\int_{\tau_1}^t \boldsymbol{\sigma}_{kk} \mathrm{d}W_s^k, \int_{\tau_1}^t \boldsymbol{\sigma}_{ll} \mathrm{d}W_s^l \right] \\
&= \int_{\tau_1}^t \sigma_k \boldsymbol{\sigma}_{ll} \mathrm{d} \underbrace{[W^k, W^l]_s}_{=s} \\
&= \boldsymbol{\sigma}_{kk} \boldsymbol{\sigma}_{ll} (t - \tau_1)
\end{aligned}$$

holds. In the second equality we used that the quadratic covariation of two continuous processes vanishes if one of them is a semimartingale and the other one is a process of bounded variation, see also Theorem 7.4 [Sch21a]. In the third equation we used the chain rule that can also be seen in Corollary 19.37 [Pfa] and the fact that for a Brownian motion $[W]_t = t$ holds. This property can be shown by Lemma 2.19 (iii) [Sch21b] using that $W^2 - t$ is a local martingale and the process $X_t = t$ is of locally bounded variation.

We are now able to define the third loss function. The *covariance loss* L_{cov} is defined by

$$L_{\text{cov}}(\theta_{\mathcal{B}}, B^i) = \sum_{k,l=1}^{d} \left(\sum_{j=2}^{n_i} \left(Z^k_{\tau_j} - Z^k_{\tau_{j-1}} \right) \left(Z^l_{\tau_j} - Z^l_{\tau_{j-1}} \right) - \sigma_k \sigma_l (\tau_{n_i} - \tau_1) \right).$$

We effectively just learn the diffusion matrix $\boldsymbol{\sigma}\boldsymbol{\sigma}^{\mathrm{T}}$, but $\boldsymbol{\sigma}$ can not be determined better. Finally, we define the *general covariance loss* L_{CL} belonging to the parameters of the model $\theta_{\text{CL}} = (\theta_{\text{AE}}, \theta_{\mathcal{B}})$ by

$$L_{\text{CL}}(\theta_{\text{CL}}, (M^i, B^i)) = L_{\text{rec}}(\theta_{\text{AE}}, M^i) + \alpha_{\text{latrec}} L_{\text{latrec}}(\theta_{\mathcal{B}}, Z^i) + \alpha_{\text{cov}} L_{\text{cov}}(\theta_{\mathcal{B}}, B^i),$$

for every $M^i \in \mathcal{M}$ with corresponding $B^i \in \mathcal{B}$ and corresponding $Z^i \in \mathcal{Z}$. Here, the positive constants α_{latrec} and α_{cov} are hyperparameters that are determined during the evaluation of the model. We train the networks using the gradient descent algorithm 1 from section 1.1.3. During the training it turned out that first another reconstruction loss has to be used to train the autoencoder network. This ensures that the structure of the data is preserved. Second, the networks are then trained using the *general covariance loss*. This will be discussed in more detail in section 3.2.3.

2.4 Method: Fokker-Planck Equation

In this method we want to take advantage of the fact that the solution process of a stochastic differential equation can also be described in terms of a probability density function which satifies the *Fokker-Planck equation*. As we saw in section 1.3 the probability density function of an Ornstein-Uhlenbeck process is normally distributed and can explicitly be represented. This offers us two advantages. First, we are able to learn the parameters of the normal distribution of the process from the empirical distribution of the data. Second, the results of the model can then be visually represented as a density at any point in time.

Ansatz

Let's start by connecting our setting with the knowledge from section 1.3.
First, let the data set $\mathcal{X}$ be given as in subsection 2.1 with time points $0 < \tau_1^i \leq \ldots \leq \tau_{n_i}^i < \infty$ for $n_i \in \mathbb{N}$ and $i = 1, \ldots, N$. We further assume that the latent state of the i-th patient can be described by the d-dimensional Ornstein-Uhlenbeck process Z^i. Here, at all time points, the values of the process should again match those of the encoded data Z^i realized by the neural network $\Phi_{\mathcal{M}}$. The parameters describing this Ornstein-Uhlenbeck process are $\varphi^i = (\mu^i, \Sigma^i, \sigma^i)$. During the training we approximate those parameters by the realized parameters of the baseline network $\Phi_{\mathcal{B}}$. For simplicity, we will name these realized parameters equally. Further, we omit the i of the i-th patient in this section and write $\varphi = (\mu, \Sigma, \sigma)$ instead of $\varphi^i = (\mu^i, \Sigma^i, \sigma^i)$. The same applies to the latent representation and the time points. The d-dimensional Ornstein-Uhlenbeck process Z solves for all $t \geq 0$ the following stochastic initial value problem

$$\begin{aligned} \mathrm{d}Z_t &= \Sigma \cdot (\mu - Z_t)\mathrm{d}t + \boldsymbol{\sigma}\mathrm{d}W_t \\ Z_0 &= Z_0 \end{aligned}$$

where $\Sigma \in \mathbb{R}^{d\times d}$ is an invertible diagonal matrix, $Z_0 \in \mathbb{R}^d$ is the unobserved initial state, $\mu \in \mathbb{R}^d$ is a vector and $\boldsymbol{\sigma} \in \mathbb{R}^{d\times d}$ is a diagonal matrix with diagonal entries belonging to the vector $\sigma \in \mathbb{R}_+^d$. In section 1.3 we saw that the probability density function P_t of Z_t for all $t \geq 0$ satisfies the Fokker-Planck equation

$$\frac{\partial}{\partial t}P_t = -\left(\frac{\partial}{\partial z}\Sigma(\mu - z)P_t\right) + \frac{\partial^2}{\partial z^2}\frac{\boldsymbol{\sigma}^2 P_t}{2} \tag{18}$$

with initial condition

$$P_0(z|Z_0) = \mathbb{1}_{\{z=Z_0\}}(z).$$

Further, in Corollary 1.40 we noticed that the Ornstein-Uhlenbeck process is a normally distributed time-homogeneous Markov process. This means the distribution of the Ornstein-Uhlenbeck process only depends on the initial value and the time difference between the initial time and the time at which we want to determine the distribution. The neural network $\Phi_{\mathcal{B}}$ with which we obtain the approximating parameters φ to our process can then be trained using the empirical distribution of the data. We will discuss the training in more detail later. Let us consider the process Z with the assumption that the encoded states Z_j correspond to the values Z_{τ_j} at times τ_j for all $j = 1, \ldots, n_i$. Then the process with initial time $\tau_j \in \{\tau_1, \ldots, \tau_{n_i}\}$ is also a solution of the stochastic initial value problem

$$\begin{aligned} \mathrm{d}Z_t &= \Sigma \cdot (\mu - Z_t)\mathrm{d}t + \boldsymbol{\sigma}\mathrm{d}W_t \\ Z_{\tau_j} &= Z_j \end{aligned}$$

for all $j = 1, \ldots, n_i$. Moreover, from corollary 1.40 we notice that for $\tau \geq \tau_j$ and $\delta_\tau := \tau - \tau_j$ a solution

$$P_{\delta_\tau}(z|Z_j) = \frac{1}{\sqrt{(2\pi)^d det(\omega(\delta_\tau))}} \exp\left(-\frac{1}{2}(z - M_{Z_j}(\delta_\tau))^{\mathrm{T}}\omega^{-1}(\delta_\tau)(z - M_{Z_j}(\delta_\tau))\right) \tag{19}$$

of the corresponding Fokker-Planck equation exists. Here, the matrix $M_{Z_j}(\delta_\tau)$ is given by

$$M_{Z_j}(\delta_\tau) = \begin{pmatrix} e^{-\lambda_1\delta_\tau}((Z_j)_1 - \mu_1) + \mu_1 \\ \vdots \\ e^{-\lambda_d\delta_\tau}((Z_j)_d - \mu_d) + \mu_d \end{pmatrix} \tag{20}$$

where $\lambda_1, \ldots, \lambda_d$ are the eigenvalues of the matrix Σ. Further, the matrix $\omega(\delta_\tau)$ is given by

$$\omega(\delta_\tau) = \begin{pmatrix} \frac{1-e^{-2\lambda_1\delta_\tau}}{2\lambda_1}\boldsymbol{\sigma}_{11}^2 & 0 & \cdots & 0 \\ 0 & \frac{1-e^{-2\lambda_2\delta_\tau}}{2\lambda_2}\boldsymbol{\sigma}_{22}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1-e^{-2\lambda_d\delta_\tau}}{2\lambda_d}\boldsymbol{\sigma}_{dd}^2 \end{pmatrix}. \tag{21}$$

That means, once we have trained our model, we are able to extrapolate the process at each of the observed time points τ_j using the associated probability density function. Let us now consider the process of training in more detail.

Training

In this subsection we go into more detail about neural network training. Compared to the other methods from sections 2.2 and 2.3, in this method the individual networks are trained in two steps.
In the first step we want to learn the latent representation of the data. The goal is to keep the structure of the data during encoding as good as possible. If stochasticity is present in the data it should still be present in the latent representation so that it can then be determined appropriately in the second step. We achieve this by first training our autoencoder on the data.
In the second step of the training we want to train our baseline network. This gives us the individual parameters to the distribution which approximate the real distribution, i.e. empirical distribution, of the data. We can determine the empirical distribution of the individual data using the latent representation learned in the first step.

Let the autoencoder $AE(\Phi_{\mathcal{M}}, \Phi_{\hat{\mathcal{M}}})$ be given with encoder network $\Phi_{\mathcal{M}}$ described above and decoder network $\Phi_{\hat{\mathcal{M}}}$ as in definition 1.3 from section 1.1.1. The neural network $\Phi_{\mathcal{M}}$ has the architecture $S_{\mathcal{M}} = (D, D, d)$ as described in section 2.1 and the network $\Phi_{\hat{\mathcal{M}}}$ has the architecture $S_{\hat{\mathcal{M}}} = (d, D, D)$. Let $\theta_{\mathcal{M}}$ and $\theta_{\hat{\mathcal{M}}}$ be the parameters belonging to these networks so $\theta_{\mathrm{AE}} = (\theta_{\mathcal{M}}, \theta_{\hat{\mathcal{M}}})$ be the parameters belonging to the autoencoder. Then we define the loss function $L_{\mathrm{enc-dec}}$ of the autoencoder by the summed column-wise squared error loss

$$L_{\mathrm{enc-dec}}(\theta_{\mathrm{AE}}, M^i) = \sum_{j=1}^{n_i} \left|\left| M_j^i - \hat{M}_j^i \right|\right|_2^2 = \sum_{j=1}^{n_i} \left|\left| M_j^i - R(AE(\Phi_{\mathcal{M}}, \Phi_{\hat{\mathcal{M}}}))(M_j^i) \right|\right|_2^2$$

for every $M^i \in \mathcal{M}$ whose j-th column is denoted by $M_j^i \in \mathbb{R}^D$. We train these networks using the gradient descent algorithm 1 from section 1.1.3. A schematic representation of the autoencoder training can be taken from figure 7.

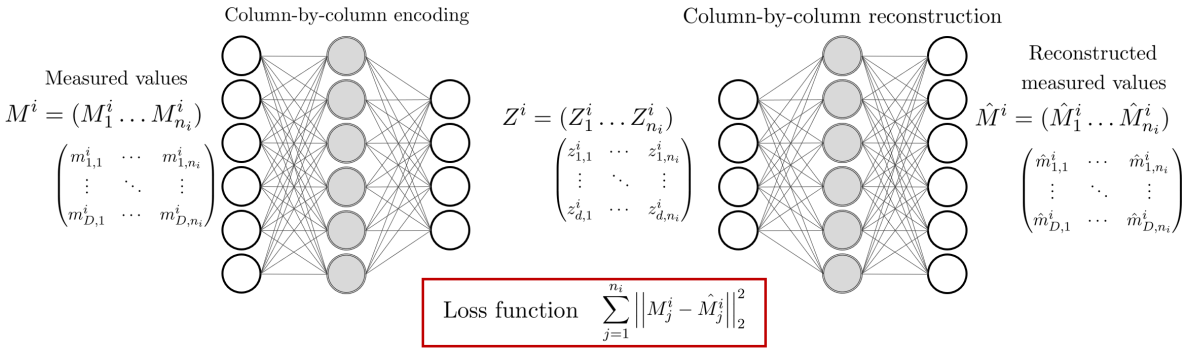


Figure 7: Schematic representation of the autoencoder training corresponding to the Fokker-Planck method.

Once we have trained our autoencoder and obtained the latent representation, we can train the baseline network $\Phi_{\mathcal{B}}$ and therefore determine approximately the parameters of the real underlying Ornstein-Uhlenbeck process belonging to this latent representation. Let $\theta_{\mathcal{B}}$ be the parameters of the neural network $\Phi_{\mathcal{B}}$. We now exploit that the Ornstein-Uhlenbeck process is a normally distributed time-homogeneous Markov process.
During the training we want to adjust the individual parameters φ^i obtained from the baseline network such that the normal distribution of the given Ornstein-Uhlenbeck process corresponds to the true individual distribution of the data, that is the empirical distribution of the individual latent representation of the data Z^i. Assuming that the time evolution of the latent data corresponds to an Ornstein-Uhlenbeck process whose parameters are unknown it is sufficient to learn the parameters of the associated normal distribution. We do this by fitting the expected value vector $M_{Z_j}(\delta_\tau)$ from equation 20 and the covariance matrix $\omega(\delta_\tau)$ from equation 21 to the empirical distribution of the latent data. These distribution parameters depend on the time difference δ_τ and the initial value Z_j. Note that we want to do this for all individuals $i = 1, \dots, N$. In order to determine the empirical distribution of the data we must first divide the data set $\mathcal{M}$ into different temporal intervals δ_τ and then make a subdivision into different initial values.
If we divide for an individual i the associated time points $\tau^i_1, \dots, \tau^i_{n_i}$ into all combination of temporal differences $\{\tau^i_l - \tau^i_k\}_{1 \le k < l \le n_i}$ we get $\frac{n_i(n_i-1)}{2}$ many combinations. A schematic representation of division into temporal differences of the latent representation Z^i of an individual i can be taken from figure 8. For an individual with a high number of observation points we get many possibilities, however, for example for an individual with $n_i = 2$ observation

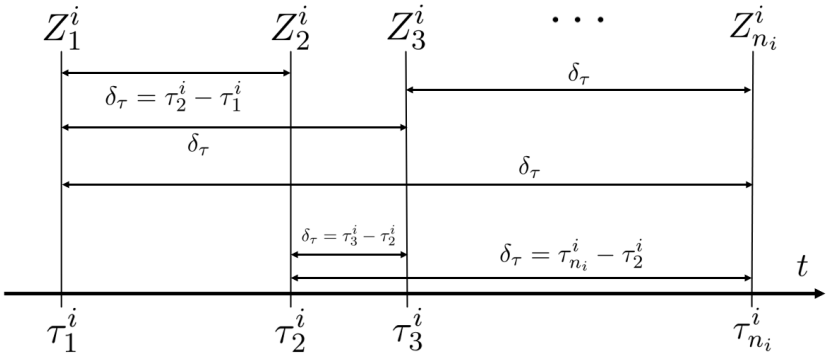


Figure 8: Schematic representation of the division into temporal differences δ_τ of the latent representation Z^i of an individual i.

points we only get one possibility. For this reason, we need to combine parts of the data from individuals that are similar. Specifically, this means from individuals whose empirical latent representations have similar distribution parameters. In the context of medical data it has already been mentioned that individuals can be divided into different groups. This is something we do not want to impose on our model but we want our model itself to be able to make such a classification. We achieve this by choosing a suitable mini batch $\mathcal{D} \subset \mathcal{M}$ for an individual $i \in \{1, \ldots, N\}$ consisting of the data of similar individuals to i and i itself. For this purpose, we define

$$\gamma^k := \left|\left|\varphi^i - \varphi^k\right|\right|_2 = \left|\left|R(\Phi_{\mathcal{B}})(B^i) - R(\Phi_{\mathcal{B}})(B^k)\right|\right|_2$$

for all $k = 1, \ldots, N$. Let $\pi : \{1, \ldots, N\} \to \{1, \ldots, N\}$ be the permutation of $\{1, \ldots, N\}$ such that

$$\gamma^{\pi(1)} \leq \ldots \leq \gamma^{\pi(N)}.$$

We then obtain the mini batch $\mathcal{D}$ by

$$\mathcal{D} := \{M^{\pi(1)}, \ldots, M^{\pi(n_{\text{batch}})}\}$$

for the size $1 \leq n_{\text{batch}} \leq N$ of the mini batch $\mathcal{D}$. Thus, $\mathcal{D}$ contains the n_{batch} series of measurements based on the distance of the parameters φ given by the baseline network. We note that $\gamma^i = 0$ such that $\pi(1) = i$ and therefore, $M^i \in \mathcal{D}$.
One of our assumptions was that individuals could be assigned to different groups. By gener-

ating the mini batches as described above we get the advantage that without any additional information about these groups the model is implicitly able to learn a group structure and assign the individuals to these groups.
Given such a mini batch $\mathcal{D}$ we now want to sort this according to the temporal differences belonging to the mini batch. For this purpose let $\Delta_\tau^{\mathcal{D}}$ be the set of all temporal differences given by

$$\Delta_\tau^{\mathcal{D}} := \left\{ \tau_l^{\pi(i)} - \tau_k^{\pi(i)} \mid 1 \leq k < l \leq n_{\pi(i)},\ i = 1, \dots, n_{\text{batch}} \right\}.$$

This set contains $\frac{1}{2} \sum_{i=1}^{n_{\text{batch}}} n_{\pi(i)}(n_{\pi(i)} - 1)$ elements. Let further be $\delta_{\text{mesh}} \in \mathbb{N}$. Then we decompose the interval $\left(\min\{\Delta_\tau^{\mathcal{D}}\}, \max\{\Delta_\tau^{\mathcal{D}}\}\right]$ into δ_{mesh} intervals $I_1, \dots, I_{\delta_{\text{mesh}}}$ given by

$$I_j = \left(\min\{\Delta_\tau^{\mathcal{D}}\} + (j-1) \frac{\max\{\Delta_\tau^{\mathcal{D}}\} - \min\{\Delta_\tau^{\mathcal{D}}\}}{\delta_{\text{mesh}}}, \min\{\Delta_\tau^{\mathcal{D}}\} + j \frac{\max\{\Delta_\tau^{\mathcal{D}}\} - \min\{\Delta_\tau^{\mathcal{D}}\}}{\delta_{\text{mesh}}} \right]$$

for $j = 1, \dots, \delta_{\text{mesh}}$. With these intervals we can partition the set $\Delta_\tau^{\mathcal{D}}$. We define the set of indices $\mathcal{D}^{\text{ind},j}$ that partition the set $\Delta_\tau^{\mathcal{D}}$ in the indices from the mini batch which differences belong to the interval I_j by

$$\mathcal{D}^{\text{ind},j} := \left\{ (\pi(i), k, l) \mid \tau_l^{\pi(i)} - \tau_k^{\pi(i)} \in \Delta_\tau^{\mathcal{D}} \cap I_j \right\}$$

for $j = 1, \dots, \delta_{\text{mesh}}$. The midpoint $\delta_\tau^{\mathcal{D},j}$ of the interval I_j is for all $j = 1, \dots, \delta_{\text{mesh}}$ defined by

$$\begin{aligned} \delta_\tau^{\mathcal{D},j} &:= \frac{1}{2} \left(\min\{\Delta_\tau^{\mathcal{D}}\} + (j-1) \frac{\max\{\Delta_\tau^{\mathcal{D}}\} - \min\{\Delta_\tau^{\mathcal{D}}\}}{\delta_{\text{mesh}}} + \min\{\Delta_\tau^{\mathcal{D}}\} + j \frac{\max\{\Delta_\tau^{\mathcal{D}}\} - \min\{\Delta_\tau^{\mathcal{D}}\}}{\delta_{\text{mesh}}} \right) \\ &= \min\{\Delta_\tau^{\mathcal{D}}\} + \left(j - \frac{1}{2} \right) \frac{\max\{\Delta_\tau^{\mathcal{D}}\} - \min\{\Delta_\tau^{\mathcal{D}}\}}{\delta_{\text{mesh}}}. \end{aligned}$$

Next, given such a set of indices $\mathcal{D}^{\text{ind},j}$ we want to sort the corresponding measurements dimensionally. That means we define for all dimensions $z_{\text{dim}} = 1, \dots, d$ and all $j = 1, \dots, \delta_{\text{mesh}}$ the sets $Z^{z_{\text{dim}},\mathcal{D},j}$ by

$$Z^{z_{\text{dim}},\mathcal{D},j} := \left\{ \left(Z^{\pi(i)}_{z_{\text{dim}},k}, Z^{\pi(i)}_{z_{\text{dim}},l} \right) \mid (\pi(i), k, l) \in \mathcal{D}^{\text{ind},j} \right\},$$

where $Z^{\pi(i)}_{z_{\text{dim}},k}$ and $Z^{\pi(i)}_{z_{\text{dim}},l}$ are the entries of the matrix $Z^{\pi(i)} \in \mathcal{Z}$ which we get from the component-wise realization of the neural network $\Phi_{\mathcal{M}}$ from the matrix $M^{\pi(i)} \in \mathcal{M}$. We now want to partition the set $Z^{z_{\text{dim}},\mathcal{D},j}$ into subsets sorted by the value of the first entry $Z^{\pi(i)}_{z_{\text{dim}},k}$

of the tuples. For this purpose we denote the number of elements of $Z^{z_{\text{dim}},\mathcal{D},j}$ by $n_{z_{\text{start}}} \in \mathbb{N}$ which depends on z_{dim}, $\mathcal{D}$ and j. With the partitions of the set $Z^{z_{\text{dim}},\mathcal{D},j}$ we want to end up roughly with $\sqrt[3]{n_{z_{\text{start}}}}$ many subsets each with $\sqrt{n_{z_{\text{start}}}}$ many elements. This does not add up exactly since $\sqrt[3]{n_{z_{\text{start}}}}$ and $\sqrt{n_{z_{\text{start}}}}$ are not necessarily natural numbers. We obtain the subsets $Z^{z_{\text{dim}},\mathcal{D},j,1}, \ldots, Z^{z_{\text{dim}},\mathcal{D},j,z_{\text{mesh}}}$ for $z_{\text{mesh}} \in \mathbb{N}$ by sorting the set $Z^{z_{\text{dim}},\mathcal{D},j}$ according to the value of the first tuple entry such that in each of the subsets these values are approximately equal. According to this construction, the individual subsets contain all tuples with similar initial values, i.e. first tuple entries in each case.
Now, we have sorted our latent data on the one hand according to temporal differences and on the other hand according to the initial values as described at the beginning. So let us consider the average initial value z_{start}, i.e. the average of the first tuple entries and the final values, i.e. the second tuple entries from one of the subsets $Z^{z_{\text{dim}},\mathcal{D},j,k}$ for $z_{\text{dim}} \in \{1, \ldots, d\}$, $j \in \{1, \ldots, \delta_{\text{mesh}}\}$ and $k \in \{1, \ldots, z_{\text{mesh}}\}$. The idea is that for an Ornstein-Uhlenbeck process with a given initial value, the final values come from a normal distribution. This is illustrated in figure 9.

We now want to estimate the parameters of this normal distribution appropriately using

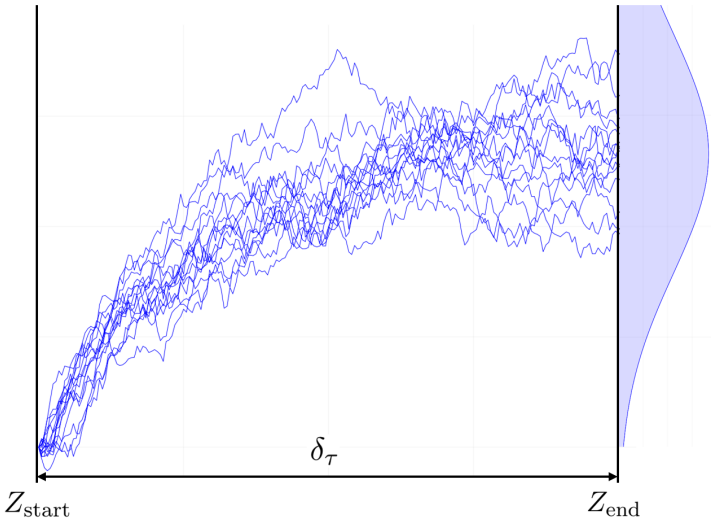


Figure 9: Schematic representation of the normal distribution of an Ornstein-Uhlenbeck process.

the data, i.e. the final values. The arithmetic mean of the empirical data is recommended as an estimator for the expected value. According to example 2.62 [vH21], this estimator is

a uniformly mininum-variance unbiased (UMVU) estimator. The unbiased sample variance of the empirical data is a recommended estimator for the variance. This estimator is UMVU according to Example 2.62 [vH21]. For the $n_{\text{end}} \in \mathbb{N}$ final values $z_{\text{ends}} = z_1, \dots, z_{n_{\text{end}}}$ according to a subset $Z^{z_{\text{dim}},\mathcal{D},j,k}$ the arithmetic mean is defined by

$$\overline{z_{\text{ends}}} := \frac{1}{n_{\text{end}}} \sum_{l=1}^{n_{\text{end}}} z_l$$

and the unbiased sample variance is defined by

$$S^2(z_{\text{ends}}) := \frac{1}{n_{\text{end}} - 1} \sum_{l=1}^{n_{\text{end}}} (z_l - \overline{z_{\text{ends}}})^2.$$

We are now able to define the loss function.
The Fokker-Planck loss L_{FP} on a mini batch $\mathcal{D}$ corresponding to the neural network $\Phi_{\mathcal{B}}$ with parameters $\theta_{\mathcal{B}}$ is defined by

$$L_{\text{FP}}(\theta_{\mathcal{B}}, B^i) = \sum_{j=1}^{\delta_{\text{mesh}}} \sum_{z_{\text{dim}}=1}^{d} \sum_{k=1}^{z_{\text{mesh}}} \left(M_{z_{\text{start}}}(\delta_\tau^{\mathcal{D},j}, \varphi^i) - \overline{z_{\text{ends}}} \right)^2 + \left(\omega(\delta_\tau^{\mathcal{D},j}, \varphi^i) - S^2(z_{\text{ends}}) \right)^2$$

for $i \in \{1, \dots, N\}$. A schematic representation of the baseline network training can be found in figure 10.
For $n_{\text{batches}} \in \mathbb{N}$ sampled individuals $i_1, \dots, i_{n_{\text{batches}}} \in \{1, \dots, N\}$ and corresponding mini batches $\mathcal{D}_1, \dots, \mathcal{D}_{n_{\text{batches}}}$ which are generated as described above we define the objective loss function $\mathcal{L}_{\text{FP}}$ by

$$\mathcal{L}_{\text{FP}}(\theta_{\mathcal{B}}) := \frac{1}{n_{\text{batches}}} \sum_{j=1}^{n_{\text{batches}}} L_{\text{FP}}(\theta_{\mathcal{B}}, B^{i_j}).$$

By sampling the individuals with which we generate our mini batches we get a kind of stochastic gradient descent. The training of this method can then be described by algorithm 3.

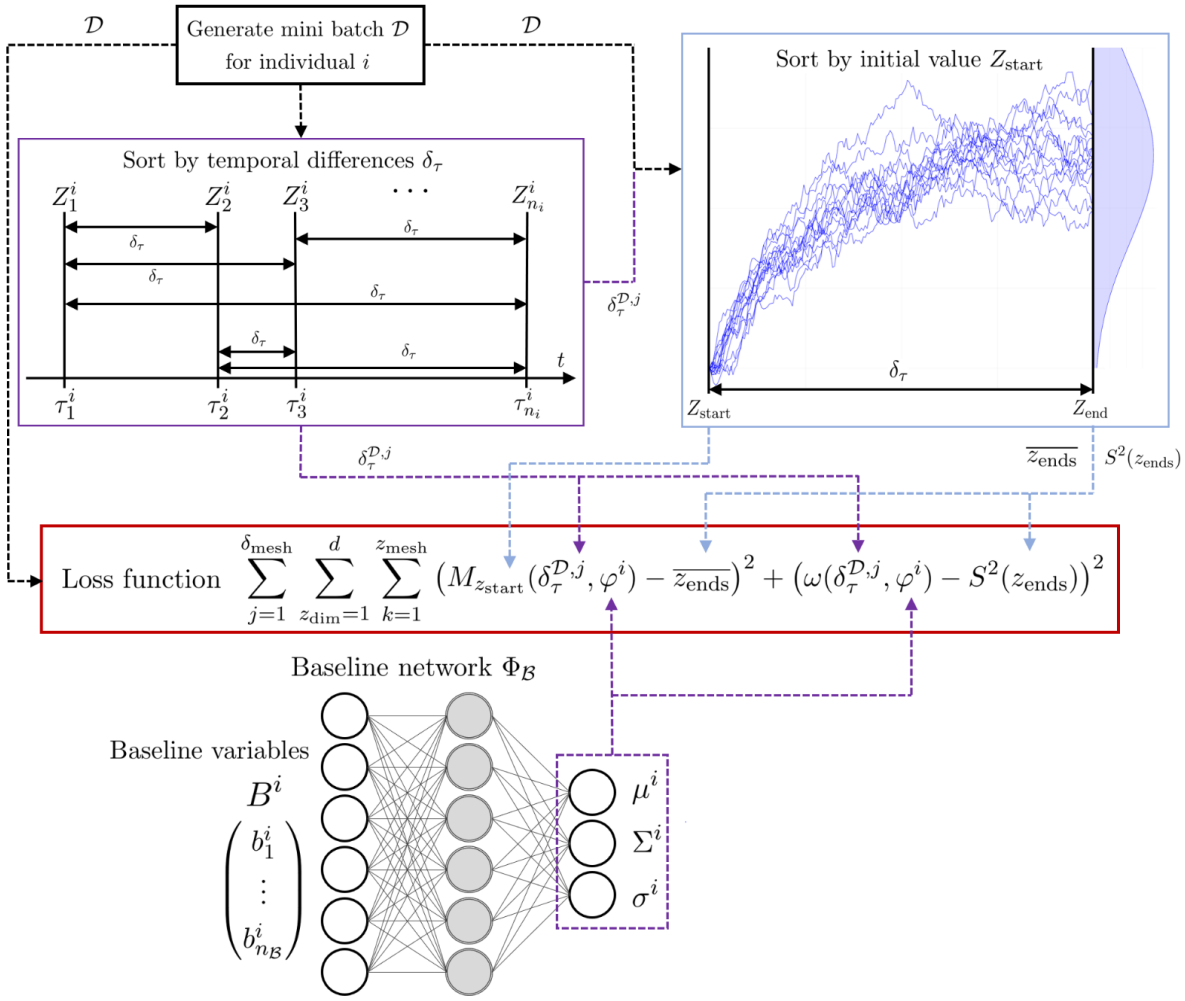


Figure 10: Schematic representation of the baseline network training on a mini batch corresponding to the Fokker-Planck method.

Algorithm 3: Fokker-Planck

Require: dataset $\mathcal{X}$, learning rate η, $\theta_{\mathcal{B}0}$, epochs, n_{batches}, δ_{mesh}
$k, e \leftarrow 0$
while $e <$ epochs **do**
  sample indivuals $i_1, \ldots, i_{n_{\text{batches}}} \in \{1, \ldots, N\}$
  generate mini batches $\mathcal{D}_1, \ldots, \mathcal{D}_{n_{\text{batches}}}$ corresponding to the individuals
  divide mini batches by temporal differences δ_τ
  subdivide divided mini batches by initial values
  for n_{batch} in $1 : n_{\text{batches}}$ **do**
    obtain δ_τ, z_{start}, $\overline{z_{\text{ends}}}$, $S^2(z_{\text{ends}})$ from the mini batch $\mathcal{D}_{n_{\text{batch}}}$
    $\nabla \leftarrow \text{backprop}(\theta_{\mathcal{B}k}, L_{\text{FP}}, \mathcal{D}_{n_{\text{batch}}})$
    $\theta_{\mathcal{B}k+1} \leftarrow \text{update}(\theta_{\mathcal{B}k}, \nabla, \eta)$
    $k \leftarrow k + 1$
  end
  $e \leftarrow e + 1$
end

3 Simulation Study

In order to see how the performance of the methods developed in chapter 2 is we test them within a simulation study on a generated suitable data set. This data set primarily fits the setting described in section 2.1. Since this thesis is motivated by the application to medical data collected in the course of a study on a disease, such as the SMArtCARE data, the simulated data set is inspired by such real data. Participants in such studies have different characteristics that are important for the health status measured within the study. This means that they can be assigned to different groups. A simple classification is the distinction between a group consisting of participants who suffer from the disease and a group consisting of those who are healthy. In addition, the measured values of the participants show a temporal development over the individual measurement times. This development is individual for each participant. Within the above mentioned groups, however, the general temporal development of the measured values should be similar. Considering the SMArtCARE data set Considering the SMArtCARE data set it turns out that the assumption of an underlying Ornstein-Uhlenbeck process is suitable for such a development. Overall, we consider a simulated data set that is inspired by real data, but still designed to be simple enough such that weaknesses in the methods can be quickly identified.

3.1 Simulation Design

As a kind of extension of Hackenberg's work [HHP$^+$22] on stochastic differential equations, the simulation design explained below was developed building on that one presented in her thesis.

The following general conditions apply to our generated simulation design. We have a total of $N = 1000$ individuals. Each individual i has $2 \leq n_i \leq 10$ observations over time. The study entry τ_1^i of every individual is sampled uniformly between 0 and $\frac{1}{4}$, such that τ_1^i is sampled from $U([0, \frac{1}{4}])$. The individual observation times are then given by $0 \leq \tau_1^i \leq \ldots \leq \tau_{n_i}^i \leq 1$. This means each individual has an individual study entry and an individual number of observation points at individual time points. Each measurement over time contains ten features, i.e. $D = 10$. The values of the measurement series for each individual are given by different paths of different Ornstein-Uhlenbeck processes. In order to assign the individuals to different groups, the parameters of the Ornstein-Uhlenbeck processes differ in each group. We distinguish between two groups. The first group is meant to simulate those individuals, which are "healthy" and the second group those who are "sick". The individuals $i \in \{1, \ldots, N\}$

are randomly assigned to one of the groups, such that every group contains $\frac{N}{2}$ individuals. Each group is based on two different Ornstein-Uhlenbeck processes, i.e. two different tuples of parameters corresponding to these processes. The values of the first five features of the measurement series are described by paths of the first Ornstein-Uhlenbeck process and the values of the remaining five features are described by paths of the other one. The initial values and parameters of these processes can be taken from table 1. The paths of these Ornstein-Uhlenbeck processes are approximated by the Euler-Maruyama method from Definition 1.28 at the given individual time points.

The process of simulating the set of series of measurements $\mathcal{M}$ is described by Algorithm 4. An example of the data from individuals of both groups can be found in figures 11 and 12. The idea to generate the data with these parameters is that the measured values of healthy individuals evolve positively over time. The data of the sick individuals, on the other hand, partly shows a certain destructive development of the values over time. In addition, these are more stochastically influenced.

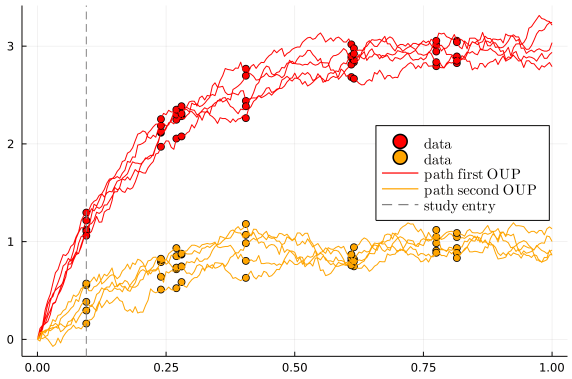


Figure 11: Data series of an individual from group 1 (healthy individuals).

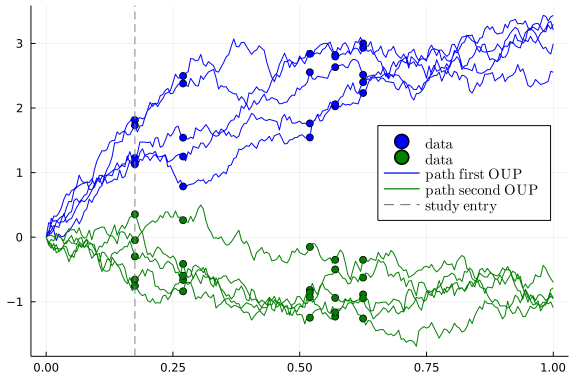


Figure 12: Data series of an individual from group 2 (sick individuals).

Further, we have a look at the baseline variables. These contain the specific characteristics of each individual. Each baseline vector contains 50 features, i.e. $n_{\mathcal{B}} = 50$. There are two variants to simulate these variables. The first variant is to distinguish between individuals of the first and the second group. The idea is to fill the baseline vectors, up to a certain value,

[8]The notation from this table is adapted to the notation of the one-dimensional Ornstein-Uhlenbeck process from Definition 1.29.

Algorithm 4: Simulation of series of measurements

Require: $N \in 2\mathbb{N}$, $D \in 2\mathbb{N}$, components $x_0, \vartheta, \mu, \sigma$ from all OUP's
initialize random permutation $\pi : \{1, \dots, N\} \to \{1, \dots, N\}$
group1 $= \left\{\pi(1, \dots, N)_k \mid k = 1, \dots, \frac{N}{2}\right\}$
group2 $= \left\{\pi(1, \dots, N)_k \mid k = \frac{N}{2}, \dots, N\right\}$
for $i = 1, \dots, N$ **do**
  sample τ_1^i from $U\left(\left[0, \frac{1}{4}\right]\right)$
  sample n_i from $U([2:10])$
  sample $n_i - 1$ values from $U([\tau_1^i, 1])$
  sort these values such that $\tau_2^i \leq \ldots \leq \tau_{n_i}^i$
  $M^i = \text{zeros}(D, n_i)$
  for $j = 1, \dots, D$ **do**
    if $j \leq \frac{D}{2}$ **then**
      if $i \in$ group1 **then**
        $x_0, \vartheta, \mu, \sigma \leftarrow$ components first OUP group1
      else
        $x_0, \vartheta, \mu, \sigma \leftarrow$ components first OUP group2
      end
      $M^i[j, :] = \text{Eul-Mar}(x_0, \vartheta, \mu, \sigma, \tau_1^i \dots, \tau_{n_i}^i)$
    else
      if $i \in$ group1 **then**
        $x_0, \vartheta, \mu, \sigma \leftarrow$ components second OUP group1
      else
        $x_0, \vartheta, \mu, \sigma \leftarrow$ components second OUP group2
      end
      $M^i[j, :] = \text{Eul-Mar}(x_0, \vartheta, \mu, \sigma, \tau_1^i \dots, \tau_{n_i}^i)$
    end
  end
end
Result: $\mathcal{M} = \{M^1, \dots, M^N\}$

		Initial value	Parameters		
		x_0	ϑ	μ	σ
Group 1 "healthy" individuals	first OUP	0	5	3	0.5
	second OUP	0	5	1	0.5
Group 2 "sick" individuals	first OUP	0	3	3	1
	second OUP	0	3	-1	1

Table 1: The initial values and parameters corresponding to the Ornstein-Uhlenbeck processes (OUP) of the different groups.[8]

with the noisy information about the associated group of the individual. The remaining entries are then filled with noise values that cannot be clearly delineated between the groups. For any individual from the first group (group of "healthy" individuals), the first $\mathcal{B}_{\text{info}}$ entries (where $\mathcal{B}_{\text{info}} \in \mathbb{N}$ with $\mathcal{B}_{\text{info}} \leq n_{\mathcal{B}}$) of the vector correspond to realizations of an $\mathcal{N}(1, \sigma_{\text{info}})$ distributed random variable, where $\sigma_{\text{info}} > 0$. The remaining entries are then realizations of an $\mathcal{N}(0, \sigma_{\text{noise}})$ distributed random variable with $\sigma_{\text{noise}} > 0$. For an individual from the second group, the first $\mathcal{B}_{\text{info}}$ entries of the vector correspond to realizations of an $\mathcal{N}(-1, \sigma_{\text{info}})$ distributed random variable and the remaining $\mathcal{B}_{\text{info}}$ entries correspond to realizations of an $\mathcal{N}(0, \sigma_{\text{noise}})$ distributed random variable.

In the second variant, the baseline variables contain noisy information about the parameters of the Ornstein-Uhlenbeck processes. When filling the baseline vectors, the same procedure is used as in the previously described variant. The number $\mathcal{B}_{\text{info}}$ is a multiple of the number of parameters of the processes. Since the data from each group is generated using two different Ornstein-Uhlenbeck processes, a total of six parameters are given, i.e. $\mathcal{B}_{\text{info}} \in 6\mathbb{N}$ with $\mathcal{B}_{\text{info}} \leq n_{\mathcal{B}}$.

The process of simulating the baseline variables using the first variant can be described by algorithm 5. The simulation method using the second variant can be described by algorithm 6.

Algorithm 5: Simulation of baseline variables with group membership

Require: $n_{\mathcal{B}} \in 2\mathbb{N}$, $\mathcal{B}_{\text{info}} \in \mathbb{N}_{\leq n_{\mathcal{B}}}$, $\sigma_{\text{info}}, \sigma_{\text{noise}} > 0$
for $i = 1, \ldots, N$ **do**
  $B^i = \text{zeros}(n_{\mathcal{B}})$
  for $j = 1, \ldots, \mathcal{B}_{\text{info}}$ **do**
    if $i \in$ group1 **then**
      sample B^i_j from $\mathcal{N}(1, \sigma_{\text{info}})$
    else
      sample B^i_j from $\mathcal{N}(-1, \sigma_{\text{info}})$
    end
  end
  for $j = (\mathcal{B}_{\text{info}} + 1), \ldots, n_{\mathcal{B}}$ **do**
    sample B^i_j from $\mathcal{N}(0, \sigma_{\text{noise}})$
  end
end
Result: Baseline variables $\mathcal{B} = \{B^1, \ldots, B^N\}$

Algorithm 6: Simulation of baseline variables with true OUP parameters

Require: $n_{\mathcal{B}} \in 2\mathbb{N}$, $\mathcal{B}_{\text{info}} \in 6\mathbb{N}$, $\mathcal{B}_{\text{info}} \leq n_{\mathcal{B}}$, $\sigma_{\text{info}}, \sigma_{\text{noise}} > 0$
for $i = 1, \ldots, N$ **do**
  get 6 params from OUP's of the group in which i is containted
  $\varphi \leftarrow \text{OUPparams}_{\text{group}_i}$
  $B^i = \text{zeros}(n_{\mathcal{B}})$
  for $j = 0, \ldots, (\mathcal{B}_{\text{info}} - 1)$ **do**
    sample B^i_j from $\mathcal{N}(\varphi[(j \bmod 6) + 1], \sigma_{\text{info}})$
  end
  for $j = (\mathcal{B}_{\text{info}} + 1), \ldots, n_{\mathcal{B}}$ **do**
    sample B^i_j from $\mathcal{N}(0, \sigma_{\text{noise}})$
  end
end
Result: Baseline variables $\mathcal{B} = \{B^1, \ldots, B^N\}$

3.2 Results

In this section we will have a look at the results of our methods from section 2 using the simulation design described in section 3.1. We will encounter various problems. The aim of the methods is to suitably determine the temporal evolution underlying the data. This means to learn the parameters of the Ornstein-Uhlenbeck process. We will see that the drift parameters can be learned well by using the developed methods. The learning of the stochasticity, i.e. the diffusion parameters, on the other hand, turns out to be much more difficult. We will first look at some implementation details and then examine the results of each method.

3.2.1 Implementation Details

The developed methods were implemented in the Julia programming language of version 1.7.0 with the additional packages DataFrames.jl (v1.3.2), Flux.jl (v0.12.9), JLD2.jl (v0. 4.22), LaTeXStrings.jl (v1.3.0), StatsBase.jl (v0.33.21), Plots.jl (v1.35.0), Distributions.jl (v0.25.75), Distributed.jl and Random.jl. The exact architecture of the networks can be seen in the following table. As an optimizer during the training the ADAM optimizer [KB14] was used. The programming is loosely based on the work of Maren Hackenberg [HHP$^+$22].

	Encoder	**Outputsize**	**Parameters**	**Activation Function**
1.	Input-Layer	10	0	-
2.	Dense-Layer	10	110	tanh
3.	Dense-Layer	2	22	-
	Decoder	**Outputsize**	**Parameters**	**Activation Function**
1.	Input-Layer	2	0	-
2.	Dense-Layer	10	22	tanh
3.	Dense-Layer	10	110	-
	Baseline Network	**Outputsize**	**Parameters**	**Activation Function**
1.	Input-Layer	50	0	-
2.	Dense-Layer	50	2550	tanh
3.	Dense-Layer	50	2550	tanh
4.	Dense-Layer	2+2+2	306	$3\tanh + 1$
5.	Diagonal-Layer	6	0	-

3.2.2 Method: Euler-Maruyama

We first look at some results using the Euler-Maruyama method from Section 2.2. The idea is to approximate the latent process using the Euler-Maruyama method. Images of individual data series with their associated reconstructions, encoded and latent states can be found in Figures 13 and 14.

We note here that the reconstructions initially look promising, but do not exhibit any stochasticity, i.e. σ was learned to be equal to zero. This can be explained by the fact that the model is only trained by the reconstruction loss of the column-wise reconstruction of the data. Adding stochasticity to the latent state by the Euler-Maruyama method increases the reconstruction loss. This is what the model tries to avoid and it learns $\sigma = 0$ to keep the error as minimal as possible. Another problem is that the reconstructed latent data series is constant and thus the parameters that are supposed to indicate the drift of the latent process have not been learned correctly. The assumption that the reconstructed latent data series is equal to the encoded data series is not satisfied. Moreover, the latent reconstruction depends only on the state of the first encoded data point. To determine the parameters of the process it would be better to use the whole encoded data series.

Considering the latent space more precisely we see that the encoded states already do not have any stochasticity included. This means that already the encoder smoothes the original data in such a way that these do not exhibit stochasticity anymore. The structure of the data, which was generated by paths of the Ornstein-Uhlenbeck processes, does not remain in the encoded data. Accordingly, both drift and diffusion parameters of the processes of the latent data are not learned. On the other hand, it should be mentioned that the model is able to distinguish between individuals of the two groups.

In summary, this method has two major difficulties. First, the structure of the data is not preserved by encoding, which means that neither drift nor diffusion can be learned properly. Second, the model avoids adding stochasticity to minimize the loss.

The idea is now to independently train the autoencoder network and the baseline network by using different loss functions. This preserves the structure of the data during encoding and the parameters of the latent process can then be learned appropriately. A loss term that specifically considers the stochasticity based on the whole encoded data series is added to learn it in a suitable way such that σ is unequal to zero.

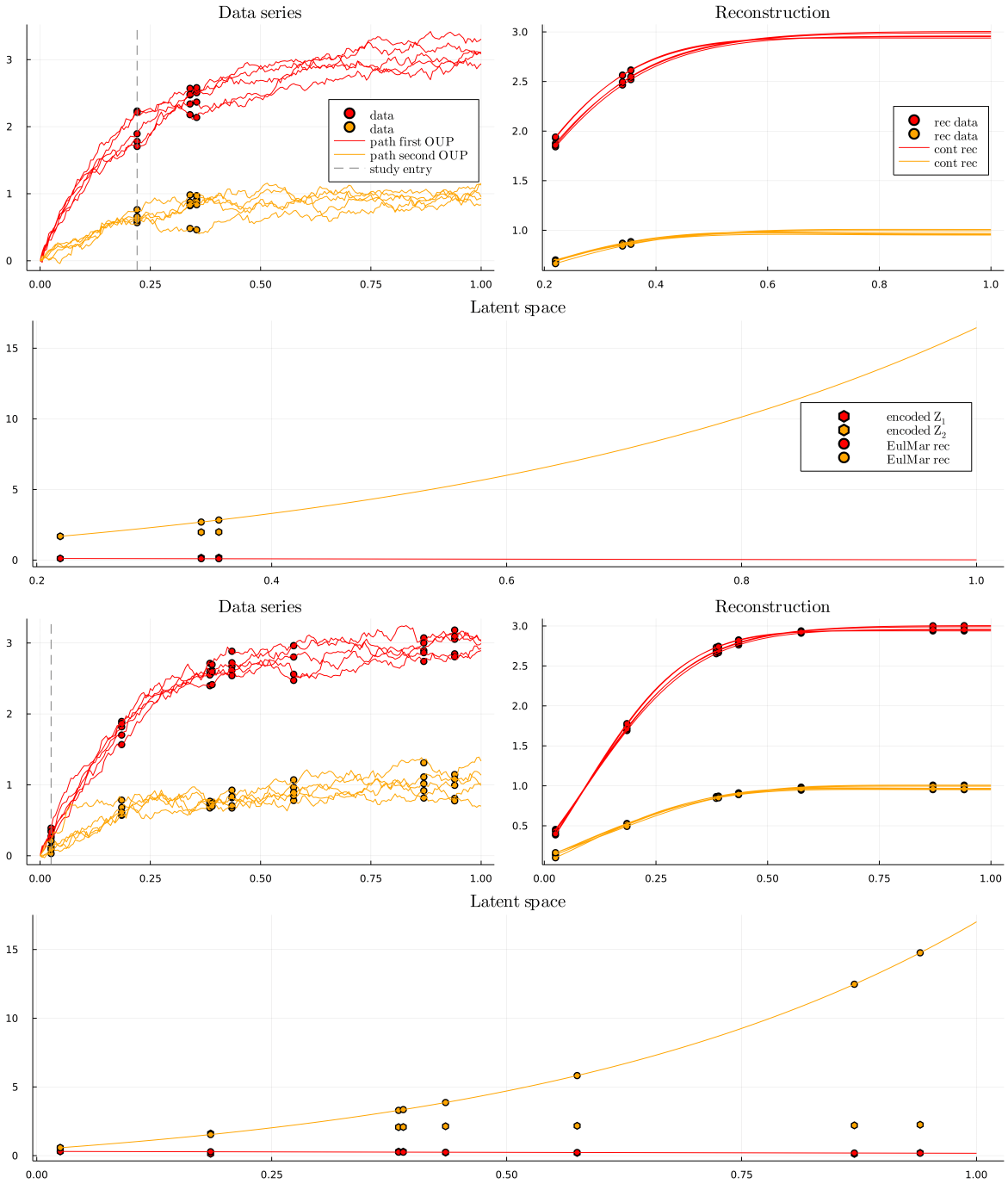


Figure 13: Data series with corresponding reconstruction, encoded and latent states of two individuals assigned to group one (healthy group) using the Euler-Maruyama method.

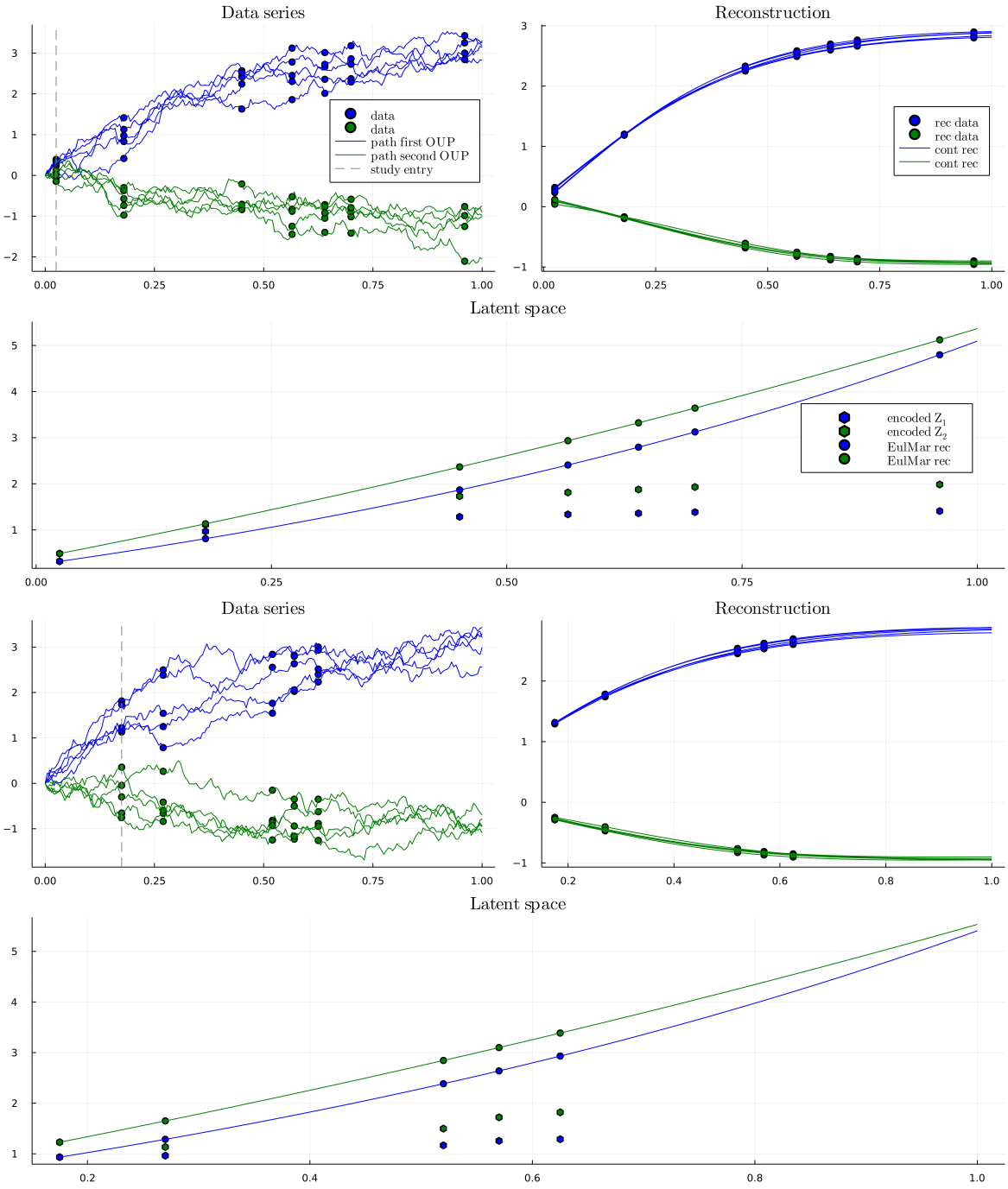


Figure 14: Data series with corresponding reconstruction, encoded and latent states of two individuals assigned to group two (sick group) using the Euler-Maruyama method.

3.2.3 Method: Covariance Loss

This section focuses on the results obtained using the covariance loss method from section 2.3. One problem that arose using this method was that the stochasticity was again learned to be equal to zero. During the application of this method, it turned out that for our setting the training of the model in two steps is suitable. That means the training of the autoencoder based on the data is first performed using a simple reconstruction loss, and afterwards the baseline network is trained by using the *general covariance loss*. Therefore, the parameters of the latent process can be learned. This *simple reconstruction loss* is defined by

$$\begin{aligned} L_{\text{srec}}(\theta_{\text{AE}}, M^i) &= \sum_{j=1}^{n_i} \left|\left| M_j^i - \hat{M}_j^i \right|\right|_2^2 \\ &= \sum_{j=1}^{n_i} \left|\left| M_j^i - R(\Phi_{\text{AE}})(M_j^i) \right|\right|_2^2 \end{aligned}$$

for all $M^i \in \mathcal{M}$. We preserve the structure of the data by using this loss first and then study the latent space. For the weights of the *general covariance loss* in the second step it turned out that in our setting $\alpha_{\text{latrec}} = 10$ and $\alpha_{\text{cov}} = 100$ is a good choice. That means the *general covariance loss* is represented by

$$L_{\text{CL}}(\theta_{\text{CL}}, (M^i, B^i)) = L_{\text{rec}}(\theta_{\text{AE}}, M^i) + 10 L_{\text{latrec}}(\theta_{\mathcal{B}}, Z^i) + 100 L_{\text{cov}}(\theta_{\mathcal{B}}, B^i).$$

Moreover, during the application of the method, it has been found that the gradient descent algorithm 1 is suitable for the training in the first step and the stochastic gradient descent algorithm 2 is suitable for the training in the second step. Some images of individual data series with their associated reconstructions, encoded and latent states can be found in figures 15 and 16. To represent the learned stochasticity, the noise was added to each of the reconstructions sized by the encoded stochasticity σ. This only serves to represent the size of σ and is not intended to reflect the reconstruction of the data. The important part of this representation of the reconstructed data is the learned drift. We see that some of the problems that we had observed using the Euler-Maruyama method are at least no longer present to the same extent. On the one hand, if we consider the encoded data, we see that there is stochasticity and that the data is showing a trend. On the other hand, the latent reconstructed data corresponds to the encoded data. Furthermore, it can be seen that σ was not learned equal to zero. Although the model has learned stochasticity equal to zero in one of the components of group one, this

is not evident in the others. In addition, the model is able to distinguish between individuals of both groups.

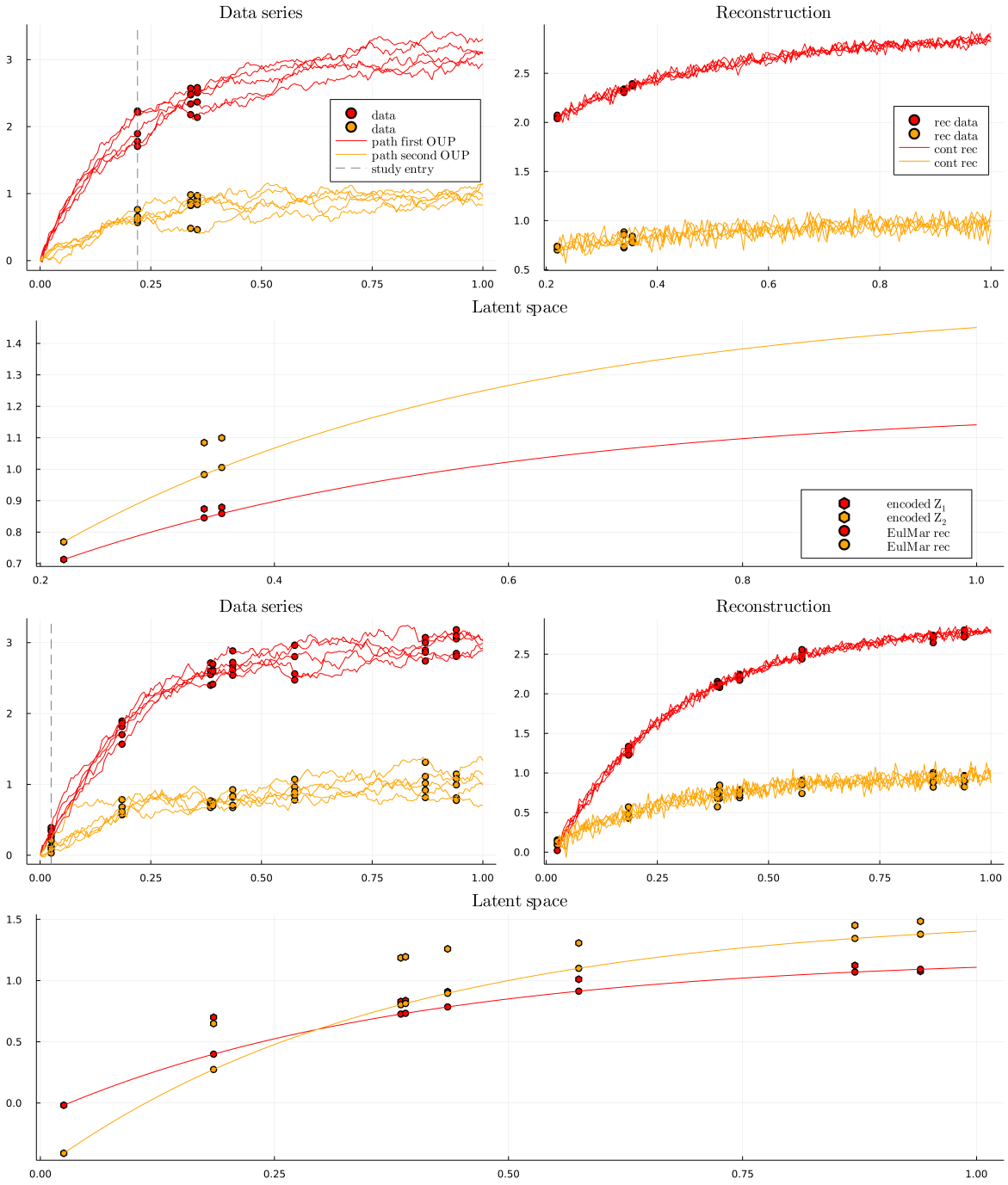


Figure 15: Data series with corresponding reconstruction and, encoded and latent states of two individuals assigned to group one (healthy group) using the covariance loss method.

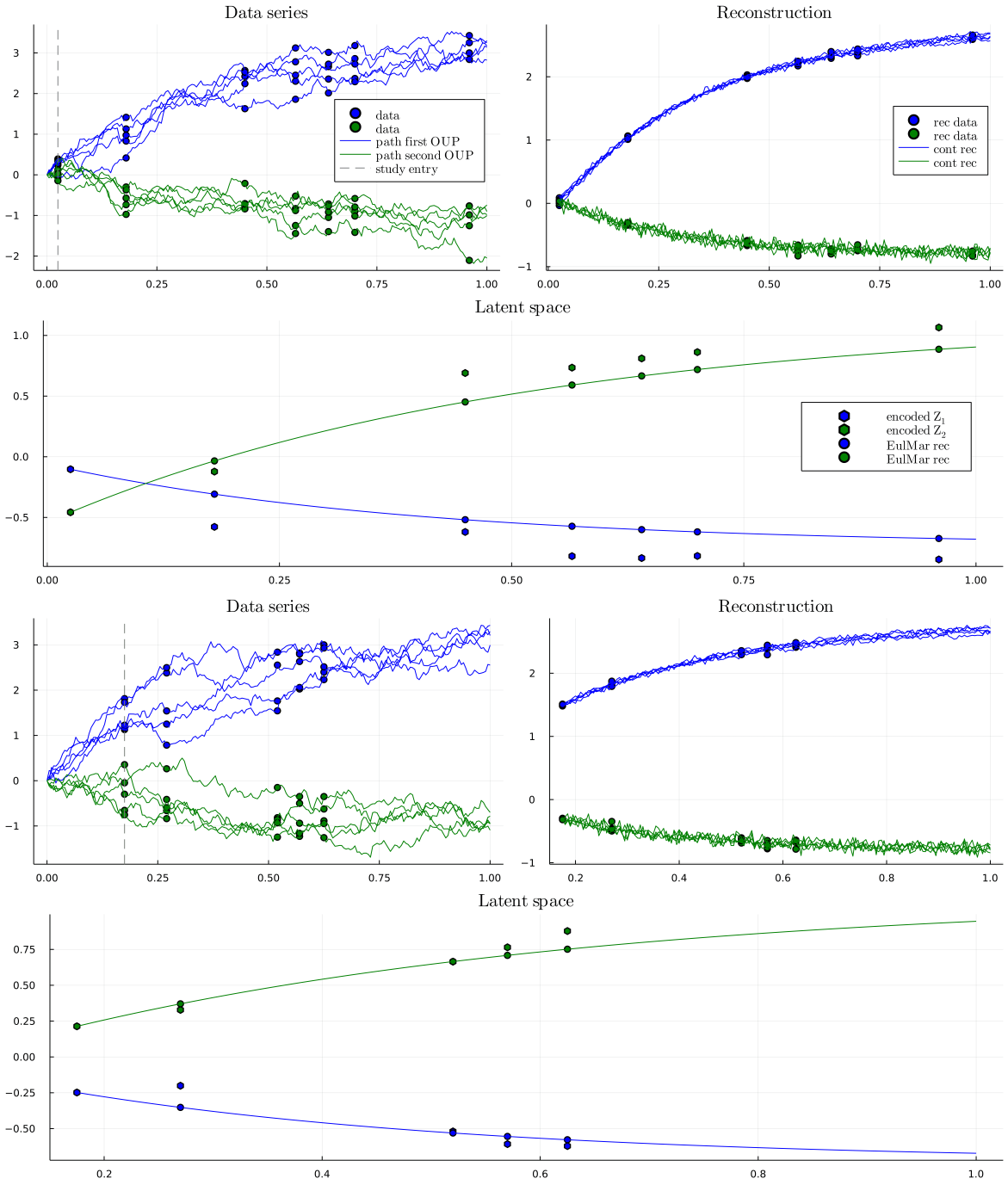


Figure 16: Data series with corresponding reconstruction, encoded and latent states of two individuals assigned to group two (sick group) using the covariance loss method.

3.2.4 Method: Fokker-Planck

In this section we consider the Fokker-Planck method developed in section 2.4. The idea of this method is to estimate the parameters of the Ornstein-Uhlenbeck process associated with the process underlying the encoded data. In applying this method, the data is first centered. This means that from all measured values and all baseline variables, the mean value of each is subtracted from the data, thus centering the data around zero. Since results from the covariance loss method 3.2.3 have already shown that the reconstruction loss in the first step is suitable to preserve the structure of the data during encoding, we focus on the latent space. We want to take advantage of the fact that the learned Ornstein-Uhlenbeck process can be represented by a probability density function. The probability density function in the latent space with the learned parameters of some individuals is represented in figures 17 and 18. We want to represent this function by the expected value and the 95% confidence interval at all time points given the study entry as the initial value. Since the latent process is normally distributed one gets an impression of the learned distribution. An advantage of this method is that this representation allows continuous extrapolation from each of the individual time points with given encoded value as initial value. Considering the figures, it can be observed that the expected value matches the data fairly closely in each dimension. The stochasticity of the data is also well established. We see that only the stochasticity of the second component of group two was learned a little too low. Also, we see that especially individuals with few observation points benefit from similar individuals with many observations. Further, we see that the group assignment of the individuals was also learned. This is demonstrated in Figure 19 in more detail. The baseline variables of all individuals were realized and displayed with the baseline network.

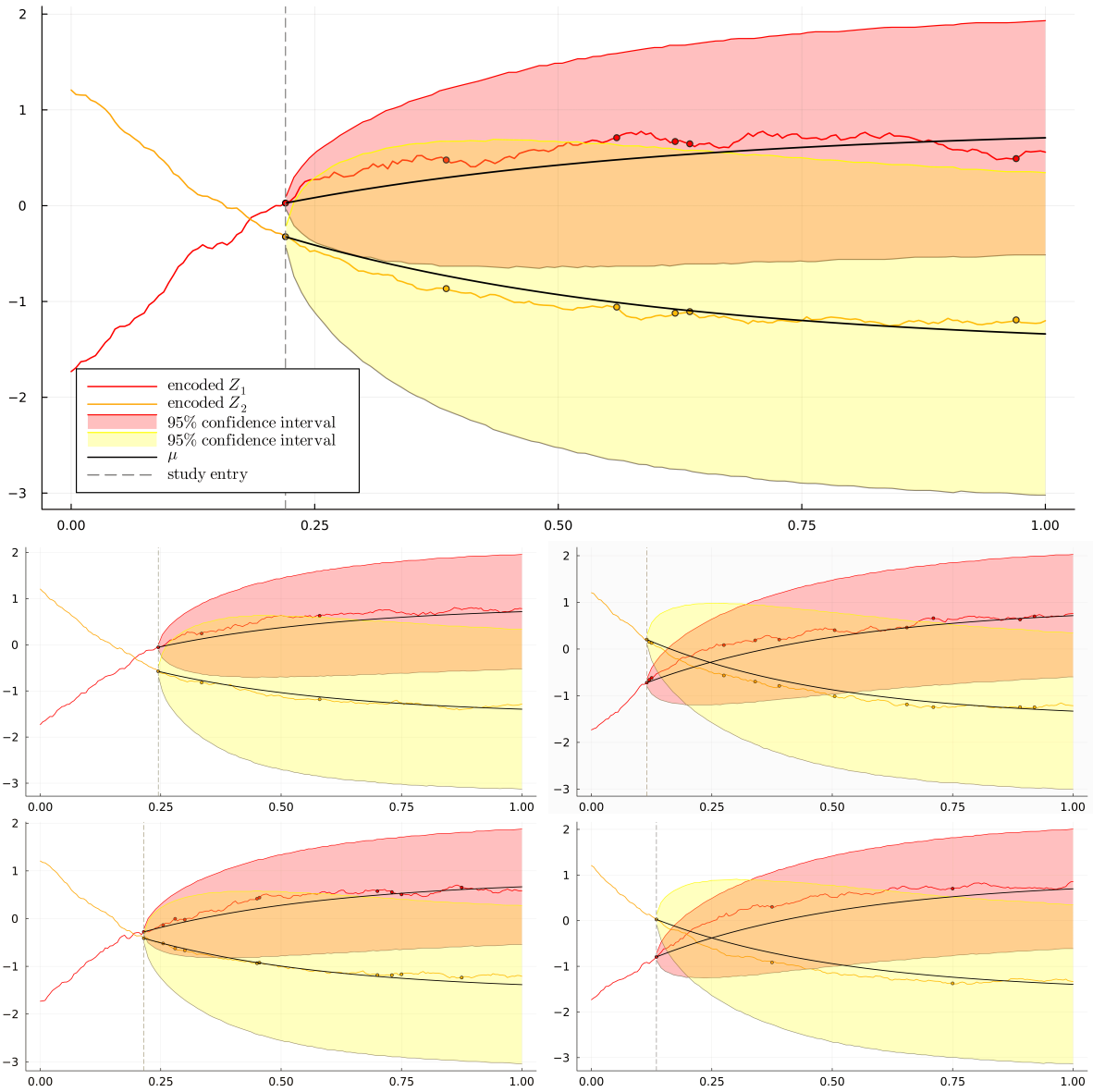


Figure 17: Latent probability density functions represented by the expected value and the 95% confidence interval at each time point given the study entry as initial value of some individuals assigned to group one (healthy group) using the Fokker-Planck method.

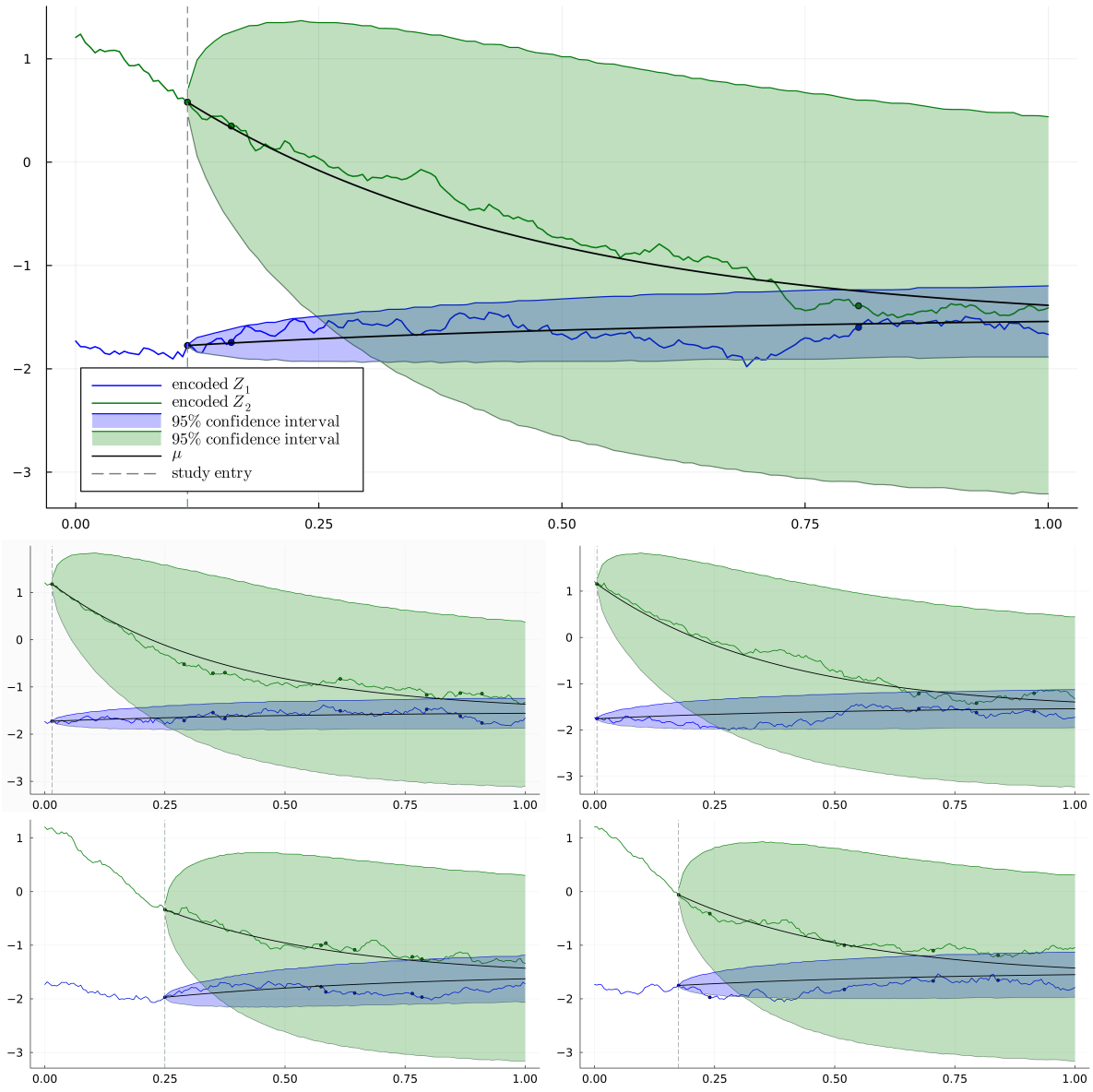


Figure 18: Latent probability density functions represented by the expected value and the 95% confidence interval at each time point given the study entry as initial value of some individuals assigned to group two (sick group) using the Fokker-Planck method.

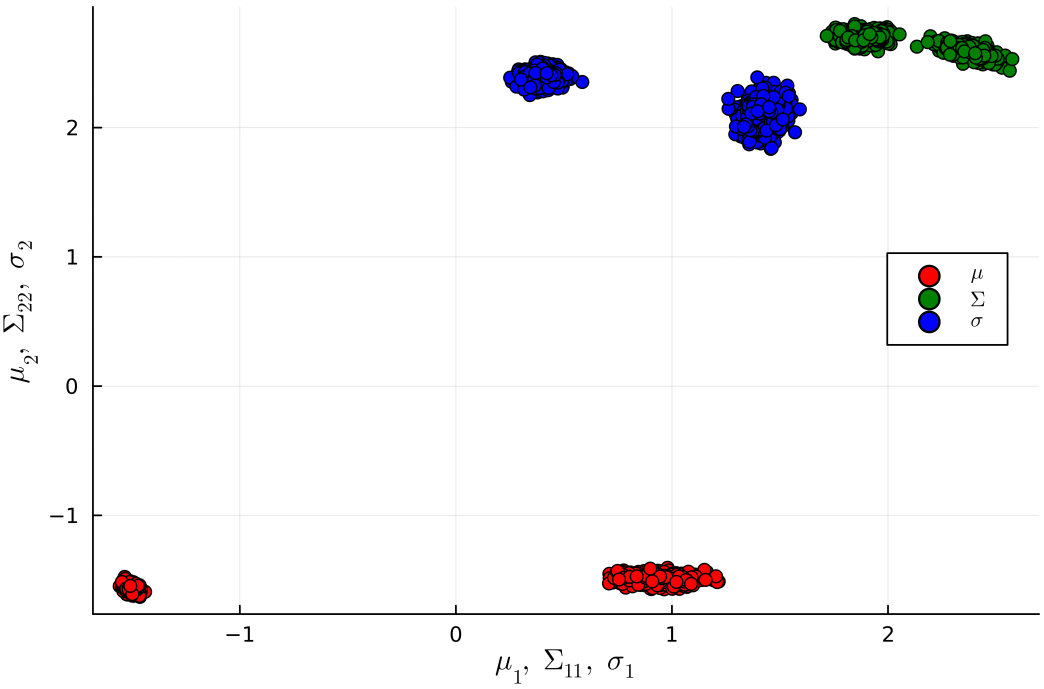


Figure 19: Representation of the learned parameters realized by the baseline network of all individual baseline variables using the Fokker-Planck method. The model learns the group structure of the data implicitly.

Discussion

In this thesis temporal data motivated by clinical data on a disease was analyzed. These data exhibit some stochasticity and have an individual number of observation points at individual times. It was also assumed that the data has a latent evolution over time. This evolution can be described by a stochastic differential equation. In addition, individual baseline variables are given to parameterize this differential equation. The data is subject to a certain group structure, which is not known and has to be learned additionally. Within this thesis a simulation design was generated motivated by the data of the SMArtCARE database. The solution of the stochastic differential equation is an Ornstein-Uhlenbeck process. Three different methods were introduced and tested on the simulation design. These methods were developed in chronological order and each of the subsequent methods benefited from the experience gained from the mistakes of the previous ones. The Euler-Maruyama method approximates the underlying process by using the Euler-Maruyama method in the latent space. The covariance loss method is designed to preserve the structure of the data and to learn the drift and diffusion parameters of the latent process through various loss functions. In the Fokker-Planck method, the empirical distribution given by the data is learned by subdividing the data. A major difficulty was to learn the diffusion term that reflects the stochasticity of the data. Also, it had to be taken into account that the individuals are assigned to different groups, which should be learned implicitly using the baseline variables. The goal was to assign each individual to one of the implicitly learned groups on the basis of the baseline variables, in order to then be able to predict the individual development of the data. This should be applicable even if there are only very few observations of some individuals. The methods applied to the simulation design offer promising results. The Fokker-Planck method was also applied to the SMArtCARE data. The results are demonstrated in figure 20.

The Fokker-Planck method nevertheless offers potential for improvement in future research. The parameters of the process can be estimated from the empirical distribution of the data within the sampled mini batches by a maximum likelihood estimator.

Furthermore, only the situation was considered in which the stochastic differential equation underlying the data is an Ornstein-Uhlenbeck process. The methods for analyzing such data could also be generalized to a bigger class of stochastic differential equations. For this purpose, a suitable method for calibration has to be chosen for a given process class.

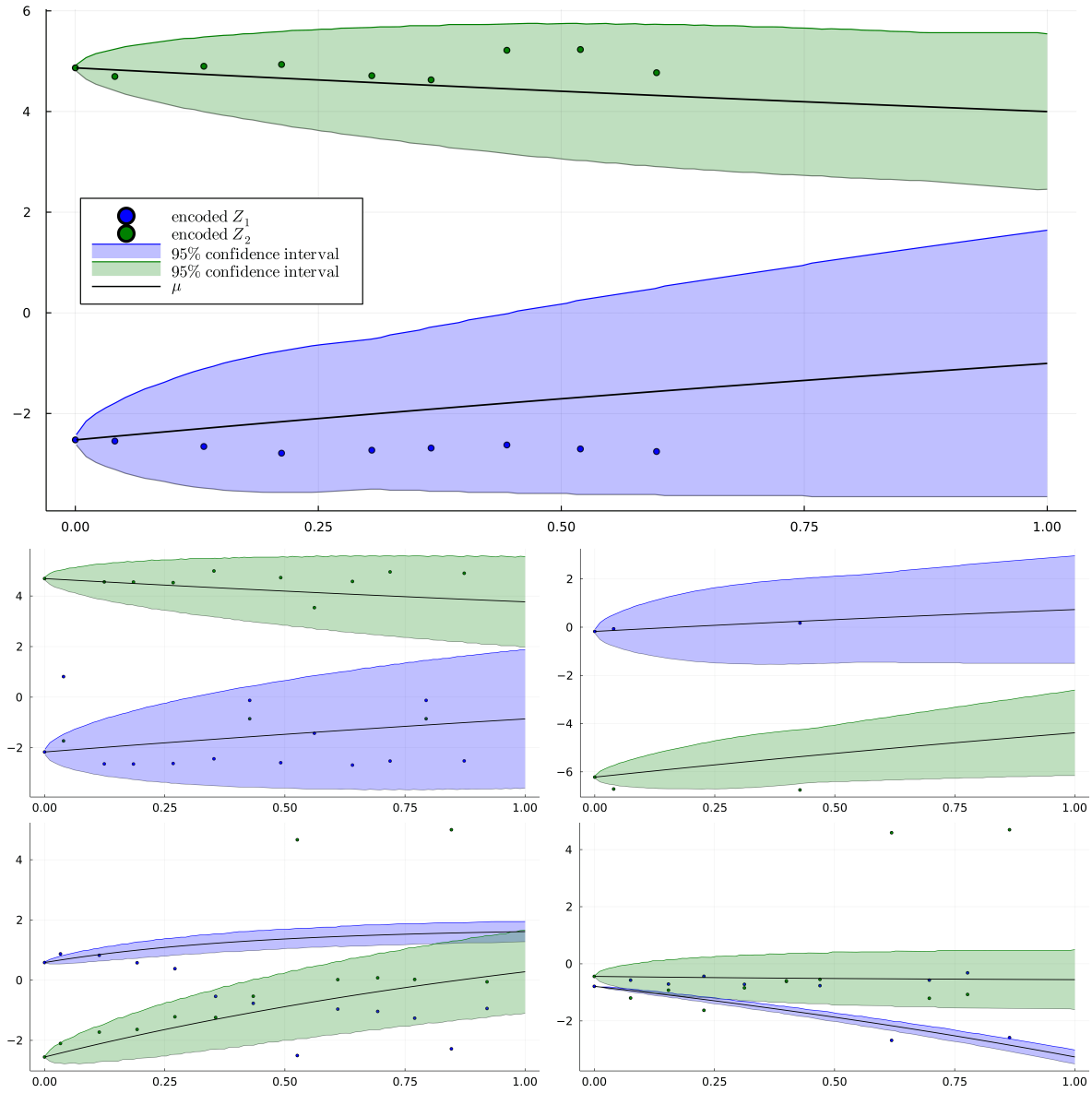


Figure 20: Application of the Fokker-Planck method to the SMArtCARE data.

References

[Bar16] Sören Bartels. *Numerik 3x9*. Springer, 2016.

[CRBD18] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[Cur44] Haskell B Curry. The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics*, 2(3):258–261, 1944.

[Cyb89] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[Gla04] Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer, 2004.

[HHP+22] Maren Hackenberg, Philipp Harms, Michelle Pfaffenlehner, Astrid Pechmann, Janbernd Kirschner, Thorsten Schmidt, and Harald Binder. Deep dynamic modeling with just two time points: Can we still allow for individual trajectories? *Biometrical Journal*, 2022.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[JS03] Jean Jacod and Albert Shiryaev. *Limit theorems for stochastic processes*, volume 288. Springer Science & Business Media, 2003.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KS12] Ioannis Karatzas and Steven Shreve. *Brownian motion and stochastic calculus*, volume 113. Springer Science & Business Media, 2012.

[KW+19] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[LLPS92] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1992.

[Oks02] Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2002.

[Pet22] Philipp Christian Petersen. Neural network theory. *University of Vienna*, 2022.

[Pfa] Professor Dr. Peter Pfallelhuber. Skriptum Stochastikzyklus. `https://www.stochastik.uni-freiburg.de/professoren/pfaffelhuber/inhalte/2020zyklus`.

[Pro04] P Protter. Stochastic integration and stochastic differential equations. *Berlin: Springer-Verlag, 1991*, 2004.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[Ris96] Professor Dr. Hannes Risken. *The Fokker-Planck equation: methods of solution and applications*. Springer Series in Synergetics №18. Springer, 2 edition, 1996.

[Sch21a] Professor Dr. Thorsten Schmidt. Skript zur Vorlesung stochastische Prozesse - WS 2020/21, 2021.

[Sch21b] Professor Dr. Thorsten Schmidt. Skriptum Stochastic Finance, 2021.

[Tap18] Professor Dr. Stefan Tappe. Skriptum Stochastische Prozesse, 2018.

[vH21] Dr. Ernst August Frhr. v. Hammerstein. Skriptum Mathematische Statistik, 2021.

[VP19] P. Vatiwutipong and N. Phewchean. Alternative way to derive the distribution of the multivariate ornstein–uhlenbeck process. *Advances in Difference Equations*, 7, 2019.

[YHL19] Cagatay Yildiz, Markus Heinonen, and Harri Lahdesmaki. Ode2vae: Deep generative second order odes with bayesian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Zusammenfassung in deutscher Sprache

Medizinische Zeitreihendaten wie die Daten zu klinischen Kohortenstudien enthalten oftmals gewisse Stochastizität. Eine Berücksichtigung dieser Stochastizität kann bei der Analyse solcher Daten also von großem Nutzen sein. Das Untersuchen von Daten, die individuelle latente Entwicklungsmuster aufweisen, wird zudem durch die oft sehr unregelmäßige Zeitstruktur der Daten erschwert. Ein Beispiel für solche Daten ist der Datensatz SMArtCARE, der Beobachtungen über die motorischen Fähigkeiten von StudienteilnehmerInnen beinhaltet, die an spinaler Muskelatrophie leiden. Die Daten zeigen, dass PatientenInnen, bei denen diese Krankheit diagnostiziert wird, ihre motorischen Fähigkeiten nicht mehr weiterentwickeln und sogar zurück entwickeln. Weiterhin stellt sich heraus, dass die Daten individuelle latente Entwicklungsmuster aufweisen, die durch gewisse Stochastizität beeinflusst wird. Außerdem sind innerhalb des Datensatzes die Messwerte der Teilnehmenden nur zu individuellen Zeitpunkten mit individueller Frequenz vorhanden. Dabei lassen sich die einzelnen PatientenInnen basierend auf ihrem Alter und der Schwere der Krankheitssymptome in unterschiedliche Gruppen einteilen.

Diese Arbeit ist motiviert durch die Analyse von Daten wie jenen aus SMArtCARE. Das Ziel dieser Analyse ist sowohl das Ermitteln der den Daten zugrundeliegenden individuellen Entwicklungsmuster unter Berücksichtigung der enthaltenen Stochastizität als auch das implizite Erlernen der Gruppenstruktur. Um diese Gruppenstruktur zu erlernen, werden die zusätzlich gegebenen Baseline-Variablen verwendet, mit denen auch die individuellen Entwicklungen charakterisiert werden sollen. Da medizinische Datensätze oftmals sehr groß sind, bieten sich Methoden zur Dimensionsreduktion der Daten des maschinellen Lernens wie der Autoencoder an. Dadurch wird eine niedrigerdimensionale latente Darstellung der Daten gewonnen, welche mit geringerem Aufwand analysiert werden kann. Es hat sich zudem gezeigt, dass es bei der Analyse von Daten, die latente Entwicklungsmuster aufweisen, hilfreich ist anzunehmen, dass den Daten ein dynamisches System zugrundeliegt, das durch eine gewöhnliche Differentialgleichung beschrieben werden kann. Dieses dynamische System soll in dieser Arbeit aufgrund der in den Daten enthaltenen Stochastizität durch eine stochastische Differentialgleichung beschrieben werden. Der Fokus dieser Arbeit liegt dabei speziell auf Ornstein-Uhlenbeck-Prozessen.

In dieser Arbeit werden drei Methoden zur Analyse der oben beschriebenen Daten entwickelt und getestet. Diese Methoden wurden chronologisch entwickelt und nachfolgende Methoden bauen auf dem Wissen der Vorigen auf. Das Ziel der Euler-Maruyama-Methode ist,

den latenten zugrundeliegenden Prozess mittels des Euler-Maruyama-Verfahrens zu approximieren. Die Kovarianz-Verlust-Methode verteilt das Erlernen der Drift- und Diffusionsparameter des Prozesses auf unterschiedliche Verlustfunktionen auf. Die Fokker-Planck-Methode nutzt Eigenschaften des den Daten zugrundeliegenden Ornstein-Uhlenbeck-Prozesses aus, um die Parameter des Prozesses anhand der empirischen Verteilung der Daten zu erlernen. Die Methoden werden auf einem generierten Simulationsdatensatz getestet. Dieser ist inspiriert durch SMArtCARE und ist möglichst einfach gehalten, um Stärken und Schwächen der Methoden schnell ausmachen zu können.
Die Ergebnisse der Methoden führen bei der Anwendung auf diesen simulierten Datensatz zu vielversprechenden Resultaten. Die Methoden bieten gute Rekonstruktionen und auch die Entwicklungsmuster der Daten wurden entsprechend erlernt. Auch die zugrundeliegende Gruppenstruktur wurde bei der Anwendung aller Methoden erlernt. Es hat sich jedoch herausgestellt, dass vor allem das Ermitteln der Stochastizität der Daten die größte Schwierigkeit ist. Die Methoden zielen daher zunehmend darauf ab diese zu lernen und sind auch in der Lage dazu.
Zusätzlich wurden die Methoden auf den Datensatz SMArtCARE angewandt.
Insgesamt bieten die Methoden auch weiterhin Bedarf zur Weiterentwicklung. So würde es sich bei der Fokker-Planck Methode anbieten die Parameter des Ornstein-Uhlenbeck-Prozesses innerhalb der Einteilung der Daten durch einen Maximum-Likelihood-Schätzer zu ermitteln. Außerdem wurde die Annahme des dynamischen Systems auf Ornstein-Uhlenbeck-Prozesse beschränkt. Hier würden sich Methoden anbieten, die auch auf größere Prozessklassen angewandt werden können.