ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

PD-NJ-ODE for Predictions in Convex Spaces

Master's Thesis

William Andersson

July 7, 2024

Advisors: Florian Ofenheimer-Krach, Prof. Dr. Josef Teichmann, Prof. Dr. Dennis Komm
Department of Computer Science, ETH Zürich

Abstract

In this thesis we provide extensions to the PD-NJ-ODE model of Krach et al. (2022) to explore ways to improve performance when predicting stochastic processes that take values in a convex set Q. We are particularly interested in finding ways to guarantee that the model's output lies in the same space. To this end, we provide three techniques that build on the PD-NJ-ODE framework and prove that they converge to the optimal prediction. We also perform experiments on various synthetic datasets to show their validity, finding them to be competitive. Our implementation can be found online.

Contents

1. Introduction

Given a (stochastic) process taking values in some closed, convex set Q known a priori, is there a way to effectively train a prediction model so that the output is always guaranteed to lie in Q? In this thesis we present techniques to do so, while also proving they converge to the optimal prediction. Our experiments confirm the theoretical results.

Perhaps the most relevant example of how this could be applied is to transformers (Vaswani et al., 2017), which are often a core component of increasingly popular large language and generative AI models. The decoder part of a transformer produces output probabilities for each word in the vocabulary, which is then used to predict the next word. This is akin to a stochastic process eventually revealing the final sentence, where at each timestep a new word is revealed, and the best prediction of the next word is given by the conditional expectation of the words given so far. Since a probability density is confined to the simplex, we can apply this technique to learn these probabilities effectively.

A more general example is that of finite state-space models, where a process takes a state in each timestep according to some probability density on the set of all states. A simple but widely known and broadly applicable family of modes is the Markov chain (with a finite number of states). At each step the process jumps to a new state, where the probability of jumping to a new state depends only on the current state. As such, the behaviour of the process is described through probability distributions over the states. Since a probability distribution forms a simplex, techniques such as the ones we present could allow us to accurately and efficiently predict a Markov chain process.

Other examples are simply of natural processes that are confined, for example Brownian motion constrained to a ball or sphere. This can model the motion of a particle inside a tank of fluid. We can also analyse cases where the Brownian motion has drift and diffusion, which could for example model a particle following a fluid current or a particle being affected by gravity.

In this thesis we suggest three techniques to solve this problem and prove that by integrating them into the PD-NJ-ODE framework, the respective models converge to the optimal solution—the conditional expectation. They are: optimally projecting to the output space; interpreting the process as a weight process on vertices (in the case Q is a polytope); and introducing an additional loss term.

This thesis is structured as follows. In Section 2, we introduce the concepts of forecasting, neural networks, and the PD-NJ-ODE model at a high level. In Section 3 we present some more technical concepts and results, including the PD-NJ-ODE framework's defining equations. Following that, we introduce the three techniques: *optimal projection* in Section 4, the *vertex approach* in Section 5, and the *new loss term* in Section 6. In Section 7 we provide some example processes satisfying the assumptions necessary for convergence. The implementation details, dataset configurations, and experiment results are in Section 8. Finally, we conclude the thesis in Section 9.

2. Background

We briefly introduce the concept of forecasting, then of feedforward, residual, and recurrent neural networks. At the end we introduce the PD-NJ-ODE framework.

2.1 Forecasting

In this thesis we work with a model used for forecasting, namely the task of predicting future values of a time series. There are two forms: offline forecasting, where a model is developed but static when used for predictioning; and online forecasting, where the model receives observations over time and bases its predictions on the information it has observed so far.

Machine learning models, especially neural network models like the one we will consider, have historically severely underperformed classic statistical techniques in both accuracy and efficiency. In Makridakis et al. (2020) the authors study the results of the fourth Makridakis competition (also known as the M4 competition) in which researchers, firms, and individuals are invited to develop models to accurately forecast (offline) a dataset consisting of a large number (100 000) of time series from various domains and with varying frequencies. They found that all pure machine learning methods ranked amongst the worst of all prediction strategies. The best were all purely statistical models or methods that combine statistical with machine learning techniques.

The field has advanced considerably since then. In the fifth edition of the competition (Makridakis et al., 2022), which attracted far more participants than all previous competitions, pure machine learning methods were shown for the first time to be significantly more accurate than all submitted statistical techniques (even those combined with machine learning). They also demonstrated more flexibility and the ability to learn across time series. Most of the best performing models are based on neural networks, with some of those utilising a variation of the recurrent neural network architecture. This is similar to the model we will analyse.

2.2 Feedforward Neural Networks

A neural network (sometimes called an artificial neural network) is a computational model that has enabled much of the recent growth in the capability of machine learning. Neural networks are inspired by the way the neurons in a brain connect, process, and forward signals between each other to produce some response to an input (e.g. sensory).

At its most basic form, a (feedforward) neural network consists of consecutive layers, where the first one receives the input and is named the input layer; the last layer is correspondingly referred to as the output layer. All layers between are hidden layers. Each layer consists of a certain number of neurons. Every neuron in each layer connects to every layer in the following layer; the exception being the output layer, where these connections terminate. Associated with each connection between a pair of neurons is a weight, namely a coefficient that determines how important the link between this pair of neurons is. The sum of outputs of the previous layer's neurons multiplied by the connections' corresponding weights forms the input to a neuron of the following layer. An activation function is applied to the output of a neuron before it is passed to the next layer.

As an example, the formula corresponding to a 3-layer neural network with input x, weights $W^{(1)}, W^{(2)}, W^{(3)}$, and activation function σ can be written as

$$x \mapsto W^{(3)}\sigma(W^{(2)}\sigma(W^{(1)}x)).$$

At a further level of abstraction, let n be the number of hidden layers. We denote by h^i, $i \in \{0, 1, \dots, n\}$, the output of the i-th hidden later, i.e., $\sigma(W^{(i)}h^{i-1})$. We can therefore also describe a neural network by writing the output layer as

$$h^n = \sigma(W^{(n)}h^{n-1}), \text{ where } h^0 := x.$$

Given a neural network specification, the goal is to "learn" (via mathematical optimisation) the weights that allow the network to make accurate predictions given some unseen input. The key revelation supporting the effectiveness of neural networks is the universal approximation theorem (Hornik, 1991), which states that even a two-layer neural network can be a universal function approximator if the activation function is non-linear. With this in hand, the final problem is how to optimise the weights. A breakthrough came with the introduction of backpropagation (Linnainmaa, 1976) which allows techniques such as stochastic gradient descent (SGD) (Robbins and Monro, 1951) to effectively optimise weights. One thing to note is that weight optimisation is generally a non-convex task and SGD (as well as other similar algorithms) often only find local optima, even if a global optimum exists.

2.3 Advanced Neural Network Architectures and the PD-NJ-ODE Framework

In order to introduce the PD-NJ-ODE framework, we must first define two new variants of neural networks, since the framework's architecture uses both. After introducing these, we discuss the research predating and inspiring the PD-NJ-ODE framework, before finally presenting the framework itself.

2.3.1 From Residual Neural Networks to the Neural ODE

A residual neural network (He et al., 2015) is a variant of the standard neural network described before. In addition to the output of the previous layer (after weights and activation function are applied), the layer itself is passed forward. This is known as a residual connection.

We can therefore express a residual neural network by the recursive equation

$$h^n = h^{n-1} + \sigma(W^{(n-1)}h^{n-1}), \text{ where } h^0 := x.$$

Residual neural networks gained significant recognition after achieving state of the art results in image recognition (He et al., 2015). We interpret them in a more general context.

Let f be the transformation defined by the neural network, parameterised by the weights θ_t. Using $t \in \{0, 1, \dots, n\}$ as an index to highlight the connection to time, we can write the update as

$$h^{t+1} = h^t + f(h^t; \theta_t).$$

This update can be interpreted as an Euler discretisation of a continuous transformation f (Chen et al., 2019). By denoting the number of hidden layers as $T = n$ and referring to h^t as the hidden state at time $t \in \{0, 1, \dots, T\}$, we can now frame residual neural networks as updating a hidden state over time.

Going further, Chen et al. (2019) propose that if we add more hidden layers and reduce the size of the time steps, we arrive at the ODE $\frac{\partial h(t)}{\partial t} = f(h(t), t, \theta)$. The output layer h^T is therefore a solution to the above ODE at time T, with initial value $h^0 = x$. In this way, a black-box ODE solver can evaluate the dynamics described by f (which is a neural network) to determine the solution (the output). The authors call this a neural ODE (Chen et al., 2019).

2.3.2 Recurrent Neural Networks

We now describe a different neural network architecture, the recurrent neural network (Rumelhart et al., 1986). In contrast to a feedforward neural network and a residual neural network, where each input corresponds to one output, an RNN takes a sequence of inputs and generates a sequence of outputs (perhaps of different or varying lengths). This property makes RNNs particularly suited to time series data, where each time step of data is input to the network and the output is continually generated.

Let the input be a sequence (X_t) for $t \in [0, T]$. An RNN that generates the output stream (Y_t) can be expressed in general form as

$$h_{t+1} = f(h_t, X_{t+1}), \quad Y_{t+1} = g(h_{t+1}), \text{ where } h_0 = X_{t_0}.$$

The sequence (h_t) is the hidden state,[1] taking values in the latent space $\mathbb{R}^{d_h}$. Both $f : \mathbb{R}^{d_X} \to \mathbb{R}^{d_H}$ and $g : \mathbb{R}^{d_H} \to \mathbb{R}^{d_Y}$ are arbitrarily deep neural networks; the former, known as the RNN cell, describes the evolution of the hidden state, while the latter is the readout function mapping the latent space to the output space. If one wishes to predict X, the input and output spaces would be the same.

2.3.3 The PD-NJ-ODE Framework and Its Evolution

The introduction of the neural ODE quickly spawned research into it they could be applied to various problems, and how it could be incorporated into other models. Researchers combined ODE models with RNNs to predict irregularly sampled time series while also modelling their behaviour between observation times. There are two such papers that are relevant to us, which we go over before we introduce the (PD)-NJ-ODE model.

The first of which, by Rubanova et al. (2019), introduces the ODE-RNN as a part of the proposed latent ODE model. In the ODE-RNN, a neural ODE model predicts the process's dynamics between observation times, and an RNN cell updates the hidden state at an observation. This procedure develops a latent ODE (hence the name) as each observation of the path is fed into the model. The latent ODE acts as an encoder of the input process, which is then followed by a final neural ODE decoder. In this way, the dynamics and hidden state can be modelled between irregular sample times. In contrast, an RNN model requires regularly sampled data and therefore cannot model these continuous dynamics. The ODE-RNN is similar to the architecture of the (PD-)NJ-ODE model (Krach et al., 2022), but tackles the offline forecasting problem instead.

The second such paper, released at the same time, introduces the GRU-ODE-Bayes method (Brouwer et al., 2019). The authors use what can be described as a specialised

1. Not to be confused with a hidden layer. Note the subscript t and that t lies in an interval, not a range of indices.

variant of an ODE-RNN to further improve the modelling of the time series, especially at observation times. They add additional logic at observation times to update the hidden state effectively at the "jumps" produced when the model is given the next sample. The PD-NJ-ODE and GRU-ODE-Bayes frameworks are the same for online forecasting but differ in their objective functions (Krach et al., 2022). Additionally, no convergence guarantees are given for GRU-ODE-Bayes.

We are finally ready to introduce the neural jump ODE (NJ-ODE) framework (Herrera et al., 2021), which we build on in this thesis. In Section 3.3 we formalise the description. As the name suggests, it is also based on combining a neural ODE with another neural network. At each observation the jump neural network updates the hidden state accordingly. Between observations a neural ODE model updates the hidden state according to a discretisation scheme that allows for continuous-in-time ODE evaluation. After each update an output (or readout) neural network maps the hidden state to the output space. All three neural networks are optimised during training. The key is the objective function, which considers both the distance between the observation and the prediction before the jump.

With this model, particularly the objective function, the authors are able to prove that the NJ-ODE model trained under the framework will converge to the L^2-optimal prediction of the input process. Amongst all processes that are measurable with respect to the information σ-algebra, the conditional expectation of the input process is closest (in an L^2 sense) to the input process. The NJ-ODE model therefore predicts the conditional expectation. In Krach et al. (2022) the path-dependent neural jump ODE (PD-NJ-ODE) model was introduced to extend the NJ-ODE model to non-Markovian paths. In Andersson et al. (2024) the PD-NJ-ODE framework was made more general by removing or adjusting the assumptions on the process to be learned.

This thesis extends the PD-NJ-ODE model to explore the convergence guarantees we can attain if the input process is constrained to a convex set and we use various techniques to ensure the output of the model is also in this set. The extensions from Andersson et al. (2024) also apply here.

3. Technical Considerations

We introduce some necessary technical terms in the following, beginning with Proposition 1—the core result that enables all results in this thesis. In addition, we give a more technical introduction of the PD-NJ-ODE model, so that the ensuing sections are understandable.

3.1 Why Q Should Be Convex

An arbitrary process X might take values in a non-convex set, so it is not clear why we should enforce that Q must be convex if the goal is to predict X. We already know is that the NJ-ODE framework can perfectly recover the conditional expectation; what's important to note is that conditional expectation might lie outside of the values X takes, if they are in a non-convex set. Trivial examples are Bernoulli random variables with non-trivial probability densities (there is non-zero mass on 0 and 1). The expectation of such a random variable lies strictly between 0 and 1.

Given that our process takes values in a compact set Q, we still need to show that its conditional expectation also lies in Q. This is a standard result, but for completeness we give a proof here.

Proposition 1 *If X is a random variable taking values in a closed convex subset Q of $\mathbb{R}^d$, then $\mathbb{E}[X \mid \mathcal{A}] \in Q$ almost surely for any sub-σ-algebra $\mathcal{A}$.*

Proof Since $\mathbb{R}^d$ is separable, Q is constructible and can be represented by the intersection of countably many half spaces $H_c^u = \{x \mid \langle x, u\rangle \leq c\}$ for some family $\mathcal{H}$ of pairs $(u, c) \in \mathbb{R}^d \times \mathbb{R}$. For each such pair (u, c), $\langle X, u\rangle \leq c$ almost surely and $\langle \mathbb{E}[X \mid \mathcal{A}], u\rangle = \mathbb{E}[\langle X, u\rangle \mid \mathcal{A}]$ almost surely, thus $\langle \mathbb{E}[X \mid \mathcal{A}], u\rangle \leq c$ almost surely.

Since $\mathcal{H}$ is countable, the above holds for every $(u, c) \in \mathcal{H}$, and so $\mathbb{E}[X \mid \mathcal{A}] \in Q$ almost surely. ■

3.2 Some Facts About Convex Sets

Here we present some useful properties about certain convex sets, to be referenced later in this thesis.

One such fact is that the following.

Proposition 2 *Optimal projection onto a closed, convex, and bounded set is continuous and bounded.*

This holds since optimal projection onto a convex set is known to be continuous,[2] and clearly a projection onto a bounded set cannot be unbounded. Proposition 2 will be key to one of the strategies we present later. It will allow us to incorporate optimal projection into a neural network, while keeping certain properties.

One family of closed, convex and bounded sets are (convex, bounded) polytopes. These are generalisations of standard 3-dimensional polyhedrons. Unbounded polytopes exist but are less relevant in this thesis. We always refer to convex, bounded polytopes when using the term "polytope". A point in such a polytope can be expressed as a convex combination of its finite number of extreme points, referred to as vertices. Let $Q \subset \mathbb{R}^d$ be a d-dimensional polytope defined by the extreme vertices $v_1, v_2, \ldots, v_n \in \mathbb{R}^d$. Thus, for every $q \in Q$ there exist weights $w_1, w_2, \ldots, w_n$ such that

$$q = w_1 v_1 + w_2 v_2 + \cdots + w_n v_n, \text{ where } w_i \geq 0 \text{ for all } 1 \leq i \leq n \text{ and } \sum_{i=1}^{n} w_i = 1.$$

This means that a polytope can be defined as the convex hull of its extreme points, i.e.,

$$Q = \text{conv}(v_1, v_2, \ldots, v_n).$$

Such a definition is called the vertex representation of a polytope and means that a polytope is uniquely defined through its extreme points.

2. See for example `https://math.stackexchange.com/a/3273008`.

A related[3] geometric object, which is a generalisation of the triangle, is the simplex. A simplex is the simplest polytope of a given dimension, hence its name. However, we use the term to refer to the probability simplex, also known as the standard simplex. A $(d-1)$ probability simplex, denoted by Δ^{d-1}, is defined by the d standard unit vectors in $\mathbb{R}^d$. It can be written as

$$\Delta^{d-1} = \{x \in \mathbb{R}^d \mid x_1 + x_2 + \cdots + x_d = 1, x_i \geq 0 \text{ for } 1 \leq i \leq d\}.$$

Note the equality, which means this is not a convex combination (although simplexes are convex). A 1-segment is a line segment in $\mathbb{R}^1$, a 2-simplex is a triangle in $\mathbb{R}^2$, and a 3-simplex is a tetrahedron in $\mathbb{R}^3$.

3.3 Details of the PD-NJ-ODE Framework

We have already explained the purpose and importance of the PD-NJ-ODE framework (Krach et al., 2022); here we introduce the notation and technical details required for us to later build upon. Some less relevant aspects are left informal.

3.3.1 Stochastic Process, Random Observations, Information σ-algebra

To begin with, we define $X := (X_t)_{t\in[0,T]} \in Q \subset \mathbb{R}^{d_X}$, where Q is closed, convex, and bounded, as the stochastic process the framework aims to predict; it is càdlàg and adapted to the filtered probability space $(\Omega, \mathcal{F}, \mathbb{F} := \{\mathcal{F}_t\}_{0\leq t\leq T}, \mathbb{P})$. Its associated running maximum process is $X_t^* := \sup_{0\leq s\leq t\leq T} |X_s|_1$. The process X may jump (have a discontinuity), so we define $X_{t-} := \lim_{\epsilon\downarrow 0} X_{t-\epsilon}$ as the left limit of X at time t.

We define the other random variables on a separate probability space, namely $(\tilde{\Omega}, \tilde{\mathcal{F}}, \tilde{\mathbb{F}} := \{\tilde{\mathcal{F}}_t\}_{0\leq t\leq T}, \tilde{\mathbb{P}})$. They are:

- $n : \tilde{\Omega} \to \mathbb{N}_{\geq 0}$, an $\tilde{\mathcal{F}}$-measurable random variable denoting the number of observations.
- $K := \sup\left\{k \in \mathbb{N} \,|\, \tilde{\mathbb{P}}(n \geq k) > 0\right\} \in \mathbb{N} \cup \{\infty\}$ is the maximal value of n.
- $t_i : \tilde{\Omega} \to [0,T] \cup \{\infty\}$ for $0 \leq i \leq K$ are the *sorted*[4] stopping times[5], describing the random observation times, with $t_i(\tilde{\omega}) = \infty$ if and only if $n(\tilde{\omega}) < i$.
- $\tau : [0,T] \times \tilde{\Omega} \to [0,T],\ (t, \tilde{\omega}) \mapsto \tau(t, \tilde{\omega}) := \max\{t_i(\tilde{\omega}) \,|\, 0 \leq i \leq n(\tilde{\omega}),\ t_i(\tilde{\omega}) \leq t\}$ is the time of the last observation before a certain time t.
- $M = (M_k)_{0\leq k\leq K}$ is the observation mask, which is a sequence of random variables on $(\tilde{\Omega}, \tilde{\mathcal{F}}, \tilde{\mathbb{P}})$ taking values in $\{0,1\}^{d_X}$ such that M_k is $\tilde{\mathcal{F}}_{t_k}$-measurable. The j-th coordinate of the k-th element of the sequence M, i.e., $M_{k,j}$, signals whether $X_{t_k,j}$, denoting the j-th coordinate of the stochastic process at observation time t_k is observed. In particular, $M_{k,j} = 1$ means that it is observed, while $M_{k,j} = 0$ means that it is not. By abuse of notation, we also write $M_{t_k} := M_k$.

3. In fact every polytope can be decomposed into a union of simplexes; this is a highly useful fact in geometry, combinatorics, and topology.
4. For all $\tilde{\omega} \in \tilde{\Omega}$, $0 = t_0 < t_1(\tilde{\omega}) < \cdots < t_{n(\tilde{\omega})}(\tilde{\omega}) \leq T$.
5. In particular, t_i is a random variable s.t. $\{t_i \leq t\} \in \tilde{\mathcal{F}}_t$ for all $1 \leq i \leq n$ and $t \in [0,T]$.

Next is the information σ-algebra, which is defined on the "combination" of the two existing probability spaces. We write it as $(\Omega \times \tilde{\Omega}, \mathcal{F} \otimes \tilde{\mathcal{F}}, \mathbb{F} \otimes \tilde{\mathbb{F}}, \mathbb{P} \times \tilde{\mathbb{P}})$ where $\mathbb{F} \otimes \tilde{\mathbb{F}}$ consists of the tensor-product σ-algebras $(\mathcal{F} \otimes \tilde{\mathcal{F}})_t := \mathcal{F}_t \otimes \tilde{\mathcal{F}}_t$ for $t \in [0,T]$. On this probability space, we define the filtration of the currently available information $\mathbb{A} := (\mathcal{A}_t)_{t\in[0,T]}$ by

$$\mathcal{A}_t := \boldsymbol{\sigma}\left(X_{t_i,j}, t_i \mid t_i \leq t,\, j \in \{l \leq l \leq d + X \mid M_{t_i,l} = 1\}\right),$$

where t_i are the random observation times and $\boldsymbol{\sigma}(\cdot)$ denotes the generated σ-algebra. By the definition of τ we have $\mathcal{A}_t = \mathcal{A}_{\tau(t)}$ for all $t \in [0,T]$.

Two concepts using the information σ-algebra are the set $\mathbb{D}$ and the definition of indistinguishability of two processes. Both will be used to refer to processes that are similar (in some sense) to X. We define the former like so.

Definition 3 *$\mathbb{D}$ is the set of all càdlàg $\mathbb{R}^{d_X}$-valued $\mathbb{A}$-adapted processes on the probability space $(\Omega \times \tilde{\Omega}, \mathcal{F} \otimes \tilde{\mathcal{F}}, \mathbb{F} \otimes \tilde{\mathbb{F}}, \mathbb{P} \times \tilde{\mathbb{P}})$.*

We paraphrase take this definition of indistinguishability from Andersson et al. (2024).

Definition 4 *Let $c_0(k) := (\mathbb{P}(n \geq k))^{-1}$. A distance between càdlàg $\mathbb{A}$-adapted processes $Z, \xi : [0,T] \times (\Omega \times \tilde{\Omega}) \to \mathbb{R}^r$ is defined through the pseudo metrics*

$$d_k(Z,\xi) = c_0(k)\, \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\mathbb{1}_{\{n\geq k\}}\, |Z_{t_k-} - \xi_{t_k-}|_2\right], \tag{1}$$

for $1 \leq k \leq K$ and two processes are called indistinguishable, if $k(Z,\xi) = 0$ for all $1 \leq k \leq K$.

3.3.2 Further Notation and Assumptions on the Stochastic Process X

We denote the conditional expectation process of X by $\hat{X} = (\hat{X}_t)_{0\leq t\leq T}$, defined by $\hat{X}_t := \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}[X_t \mid \mathcal{A}_t]$. Additionally, we define $\tilde{X}^{\leq t}$ to be the interpolation of the observations of X made until time t. More specifically, it is the rectilinear interpolation (also known as forward-fill), except jumps are replaced by linear interpolations between the previous observation time and the jump point. By this definition, $\tilde{X}^{\leq\tau(t)}$ is $\mathcal{A}_{\tau(t)}$-measurable for all t, and for any $r \geq t$ and all $s \leq \tau(t)$ we have $\tilde{X}_s^{\leq\tau(t)} = \tilde{X}_s^{\leq\tau(r)}$. In Krach et al. (2022) the authors show that, via the Doob–Dynkin Lemma (Taraldsen, 2018, Lemma 2), there exist measurable functions F_j such that $\hat{X}_{t,j} = F_j(t, \tau(t), \tilde{X}^{\leq\tau(t)})$.

Next, we outline the assumptions to be used in later proofs. Not all of the Assumptions in Krach et al. (2022) are relevant for us.

Assumption 1 *For every $1 \leq k, l \leq K$, M_k is independent of t_l and of n, $\tilde{\mathbb{P}}(M_{k,j} = 1) > 0$ and $M_{0,j} = 1$ for all $1 \leq j \leq d_X$ (every coordinate can be observed at any observation time and X is completely observed at 0) and $|M_k|_1 > 0$ for every $1 \leq k \leq K$ $\tilde{\mathbb{P}}$-almost surely (at every observation time at least one coordinate is observed).*

Assumption 2 *The probability that any two observation times are closer than $\epsilon > 0$ converges to 0 when ϵ does, i.e., if $\delta(\tilde{\omega}) := \min_{0\leq i\leq n(\tilde{\omega})} |t_{i+1}(\tilde{\omega}) - t_i(\tilde{\omega})|$ then $\lim_{\epsilon\to 0} \tilde{\mathbb{P}}(\delta < \epsilon) = 0$.*

Assumption 3 *Almost surely, with respect to the measure* $(\mathbb{P} \times \tilde{\mathbb{P}})$*, X is not observed at a jump.*

Assumption 4 *F_j are continuous and differentiable in their first coordinate t such that their partial derivatives with respect to t, denoted by f_j, are again continuous and there exists a $B > 0$ and $p \in \mathbb{N}_{\geq 1}$ such that for every $t \in [0, T]$ the functions f_j, F_j are polynomially bounded in X^*, i.e.,*

$$|F_j(\tau(t), \tau(t), \tilde{X}^{\leq \tau(t)})| + |f_j(t, \tau(t), \tilde{X}^{\leq \tau(t)})| \leq B(X_t^* + 1)^p.$$

Assumption 5 *X^* is L^{2p}-integrable, i.e., $\mathbb{E}[(X_T^*)^{2p}] < \infty$.*

Assumption 6 *The random number of observations n is integrable, i.e., $\mathbb{E}_{\tilde{\mathbb{P}}}[n] < \infty$.*

With Assumption 4 we can rewrite $\hat{X}$ by the fundamental theorem of calculus as

$$\hat{X}_{t,j} = F_j(\tau(t), \tau(t), \tilde{X}^{\leq \tau(t)}) + \int_{\tau(t)}^{t} f_j(s, \tau(t), \tilde{X}^{\leq \tau(t)})\, ds,$$

implying it is càdlàg and therefore $\hat{X} \in \mathbb{D}$ (Definition 3). We remark that jumps of $\hat{X}$ occur only at new observation times, i.e., at t_i for $1 \leq i \leq n$.

3.3.3 Truncated Signature

The signature of a path is an (infinite) tuple of features, which allows us to approximate any continuous function of the path by a linear combination of the signature terms. It is defined in Andersson et al. (2024) as follows.

Definition 5 *Let J be a closed interval in $\mathbb{R}$ and $\mathbf{X} : J \to \mathbb{R}^d$ be a continuous path with finite variation. The signature of $\mathbf{X}$ is defined as the collection of iterated integrals*

$$S(\mathbf{X}) = \left(1, \mathbf{X}_J^1, \mathbf{X}_J^2, \dots\right),$$

where, for each $m \geq 1$,

$$\mathbf{X}_J^m = \int_{\substack{u_1 < \cdots < u_m \\ u_1, \dots, u_m \in J}} d\mathbf{X}_{u_1} \otimes \cdots \otimes d\mathbf{X}_{u_m} \in (\mathbb{R}^d)^{\otimes m}.$$

The truncated signature of $\mathbf{X}$ of order $m \in \mathbb{N}$ is defined as

$$\pi_m(\mathbf{X}) = \left(1, \mathbf{X}_J^1, \mathbf{X}_J^2, \dots, \mathbf{X}_J^m\right),$$

i.e., the first $m + 1$ terms (levels) of the signature of $\mathbf{X}$. For a path of dimension d, the dimension of the truncated signature of order m is

$$\begin{cases} m + 1, & \text{if } d = 1, \\ \frac{d^{m+1} - 1}{d - 1}, & \text{if } d > 1. \end{cases}$$

In essence, the truncated signature of a path is a transformation that condenses the information of a path into a vector of fixed size. Instead of the discrete observations of X themselves, the PD-NJ-ODE model uses the truncated signature of $\tilde{X} - X_0$ as input to the neural networks. To be specific, for a truncation order m and time t, the framework takes the truncated signature $\pi_m(\tilde{X}^{\leq \tau(t)} - X_0)$. All available information at time t, denoted by $\mathcal{A}_t$, is included in the combination of X_0 and the above truncated signature. As such, we can use a fixed-length vector as input to the model, regardless of the number of observations. There are other technical properties that make it even more useful; see Section 3.3 in Krach et al. (2022).

3.3.4 Bounded Output Neural Networks

That the outputs of the neural networks we use are bounded is useful for convergence proofs, where we often need to provide upper bounds. For this reason, we introduce a special class of neural networks. These Definitions follow Definition 3.12 and Definition 3.13 in Krach et al. (2022).

Definition 6 *For any dimension $d \in \mathbb{N}$ we define the bounded output activation function with trainable parameter $\gamma > 0$ as the Lipschitz continuous function*

$$\Gamma_\gamma : \mathbb{R}^d \to \mathbb{R}^d, x \mapsto x \cdot \min\left(1, \frac{\gamma}{|x|_2}\right).$$

Then we define the class of bounded output neural networks *as*

$$\mathcal{N} := \{f_{(\theta,\gamma)} := \Gamma_\gamma \circ \bar{f}_{\bar{\theta}} \,|\, \gamma > 0, \bar{f}_{\bar{\theta}} \in \bar{\mathcal{N}}\},$$

where $\bar{\mathcal{N}}$ can be any set of neural networks. We use the notations $\bar{f}_{\bar{\theta}} \in \bar{\mathcal{N}}$ and $f_\theta \in \mathcal{N}$ for $\theta = (\bar{\theta}, \gamma)$ to highlight the trainable weights $\bar{\theta}$ (and γ) of the respective (bounded output) neural networks.

We also define another set $\tilde{N}$ of feedforward neural networks as follows (Krach et al., 2022, Section 3.3).

Definition 7 *$\tilde{\mathcal{N}}$ is a set of feedforward neural networks with Lipschitz continuous activation functions such that for any $d, D \in \mathbb{N}$ and any compact subset $\mathcal{X} \subset \mathbb{R}^d$ we have that $\tilde{\mathcal{N}}$ is dense in the space of continuous functions $C(\mathcal{X}, \mathbb{R}^D)$ (with respect to the supremum-norm). In particular, $\tilde{\mathcal{N}}$ must satisfy the standard universal approximation theorem, which is the case e.g. for the set of 1-hidden-layer neural networks with continuous, bounded and non-constant activation function (Hornik, 1991, Theorem 2). Moreover, we assume that* $\mathrm{id} \in \tilde{\mathcal{N}}$.

3.3.5 Model Architecture and the Objective Function

We have now presented all relevant background necessary to understand the PD-NJ-ODE architecture, which is defined here. It will be adapted slightly in some of the later sections.

Definition 8 *The Path-Dependent Neural Jump ODE (PD-NJ-ODE) model is given by*

$$H_0 = \rho_{\theta_2}(0, 0, \pi_m(0), X_0),$$

$$
\begin{aligned}
dH_t &= f_{\theta_1}\left(H_{t-}, t, \tau(t), \pi_m(\tilde{X}^{\leq \tau(t)} - X_0), X_0\right) dt \\
&\quad + \left(\rho_{\theta_2}\left(H_{t-}, t, \pi_m(\tilde{X}^{\leq \tau(t)} - X_0), X_0\right) - H_{t-}\right) du_t, \\
Y_t &= \tilde{g}_{\tilde{\theta}_3}(H_t).
\end{aligned}
$$

The functions $f_{\theta_1}, \rho_{\theta_2} \in \mathcal{N}$ are bounded output feedforward neural networks and $\tilde{g}_{\tilde{\theta}_3} \in \tilde{\mathcal{N}}$ is a feedforward neural network. They have trainable parameters $\theta = (\theta_1, \theta_2, \tilde{\theta}_3) \in \Theta$, where $\theta_i = (\tilde{\theta}_i, \gamma_i)$ for $i \in \{1, 2\}$ and Θ is the set of all possible weights for the PD-NJ-ODE model; $m \in \mathbb{N}$ is the signature truncation level and u is the jump process counting the observations.

In words, f models the ODE dynamics, ρ models the jumps when new observations are made, and $\tilde{g}$ is the readout map from the latent space to the target space.

The objective function for the (PD)-NJ-ODE framework is an important part of proving that the model converges to optimal solution, since it determines the loss function with which the model is trained. Let $\mathbb{D}$ be as in Definition 3. The objective function Ψ and associated theoretical loss function Φ are given as

$$
\begin{aligned}
&\Psi : \mathbb{D} \to \mathbb{R}, \\
&\quad Z \mapsto \Psi(Z) := \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\frac{1}{n}\sum_{i=1}^{n}\left(|M_i \odot (X_{t_i} - Z_{t_i})|_2 + |M_i \odot (Z_{t_i} - Z_{t_i-})|_2\right)^2\right], \qquad (2) \\
&\Phi : \Theta \to \mathbb{R}, \theta \mapsto \Phi(\theta) := \Psi(Y^\theta(\tilde{X}^{\leq \tau(\cdot)})),
\end{aligned}
$$

where $\odot$ is the element-wise multiplication (Hadamard product).

Let us assume that we observe N independent realisations of the path X together with independent realisations of the observation mask M at times $(t_1^{(j)}, \ldots, t_{n^{(j)}}^{(j)})$, $1 \leq j \leq N$, which are themselves independent realisations of the random vector $(n, t_1, \ldots, t_n)$. In particular, let us assume that $X^{(j)} \sim X$, $M^{(j)} \sim M$ and $(n^{(j)}, t_1^{(j)}, \ldots, t_{n^{(j)}}^{(j)}) \sim (n, t_1, \ldots, t_n)$ are i.i.d. random processes (respectively variables) for $1 \leq j \leq N$ and that our training data is one realisation of them. We write $Y^{\theta,j} := Y^\theta(\tilde{X}^{\leq \tau(\cdot),(j)})$. Then the Monte Carlo approximation of the theoretical loss function

$$
\hat{\Phi}_N(\theta) := \frac{1}{N}\sum_{j=1}^{N}\frac{1}{n^{(j)}}\sum_{i=1}^{n^{(j)}}\left(\left|M_i^{(j)} \odot \left(X_{t_i^{(j)}}^{(j)} - Y_{t_i^{(j)}}^{\theta,j}\right)\right|_2 + \left|M_i^{(j)} \odot \left(Y_{t_i^{(j)}}^{\theta,j} - Y_{t_i^{(j)}-}^{\theta,j}\right)\right|_2\right)^2,
$$

converges $(\mathbb{P} \times \tilde{\mathbb{P}})$-a.s. to $\Phi(\theta)$ as $N \to \infty$ by the law of large numbers (Krach et al., 2022, Theorem 4.4). This convergence theorem is a core theorem of Krach et al. (2022) (and of Herrera et al. (2021) too), so much of the effort in this thesis is spent on ensuring the theorem holds when we change the context.

4. Optimal Projection

The first strategy we present is one that directly solves the problem we face. To do so, we simply add a final projection "layer" to the model, so that the model's output is projected optimally onto Q. To implement this, we re-interpret the readout mapping from the original PD-NJ-ODE model in that we simply compose it with a projection function.

4.1 Setting

Let us first clarify that we desire optimality with respect to the Euclidean norm, i.e., the optimal projection function $P : \mathbb{R}^{d_X} \to Q$ is such that

$$P(x) := \min_{q \in Q} \|x - q\|_2.$$

Now let us specify the new readout function. Let $\tilde{g}$ be the readout feedforward neural network according to Definition 8. Let P be the optimal projection function as above. Then the final readout function for this strategy is

$$g_\theta : \mathbb{R}^{d_H} \to Q, \quad g_\theta(H_t) := P \circ \tilde{g}(H_t) = P(\tilde{g}_\theta(H_t)) = Y_t,$$

where Y_t is the prediction from the PD-NJ-ODE model at time t according to the hidden state H_t. The trainable weights are given by θ.

We take care to separate the readout and the projection because the readout network $\tilde{g}$ can still be trained, while the projection function P is fixed.

Remark 9 *The boundedness assumption (Assumption 4) is not needed here if Q is bounded, as can be seen in the proof in Section 5.2.*

4.2 Proving the PD-NJ-ODE Results

First, we need to show that changing the readout function in this way retains continuity and boundedness of the output. These points are necessary for Theorem 4.1 and Theorem 4.4 (Krach et al., 2022), the two parts of the convergence proof. Proposition 2 proves this holds, as the new readout function is a composition of an optimal projection function P and a continuous, bounded neural network $\tilde{g}$.

The first part of the convergence proof is Theorem 4.1 in Krach et al. (2022), the relevant part of which is paraphrased as follows.

Theorem 10 *The PD-NJ-ODE model with optimal projection approximates $\hat{X}$ arbitrarily well.*

Proof We assume the context of the original proof and explain only the few changes necessary. We write g instead of $\tilde{g}$ for the (new) readout function to avoid confusion. To begin with, we note that it is not generally possible for $g_{\theta_3^*}$ to be the identity function if we need to enforce the output is optimally projected to Q. This contradicts an assumption in the original proof. Instead, we assume that the trainable layers (i.e., all layers except for the projection) form the identity function. This means that $\tilde{g}_\theta = \text{id}$ and therefore $g(H_t) = P(H_t)$. Thus, in the context of the proof, we have $Y_t^{\theta_m^*} = P(H_t)$.

We have three inequalities to adjust. For the first, we just use the fact that P gives an optimal projection, and that $\hat{X}_t \in Q$ to get an initial inequality. Thus

$$\left| Y_t^{\theta_m^*} - \hat{X}_t \right|_1 = \left| P(H_t) - \hat{X}_t \right|_1 \leq \left| H_t - \hat{X}_t \right|_1$$

for all $t \in [0, T]$. This can be applied to the two inequalities in Krach et al. (2022) directly to gain a bound between the output and $\hat{X}$. For the final one, we once again use that $|P(H_t) - \hat{X}_t|_2 \leq |H_t - \hat{X}_t|_2$ and the value of c_m follows.

The proof of Theorem 4.1 (Krach et al., 2022) can then be finished as in the original paper. ■

The second of the two convergence proofs (Krach et al., 2022, Theorem 4.4) is simpler to validate. In fact, we just need to verify continuity and boundedness (with respect to weights) of the three neural networks constituting the PD-NJ-ODE model. Continuity is required for Lemma 4.5 (Krach et al., 2022) and boundedness for the beginning of the proof of Theorem 4.4 itself. According to Proposition 2 this holds for the new readout function, so Theorem 4.4's proof can otherwise be kept as is.

4.3 Discussion

With this, we now have a conceptually simple way to enforce that the PD-NJ-ODE model's output lies in Q. The convergence proofs still hold, so the model should learn to predict $\hat{X}$. However, we have not leveraged that we know X lies in a closed, convex space; the next two approaches do so.

Additionally, we acknowledge the major limitation of this approach, which is that, in general, it is hard to calculate optimal projections; in many cases no closed-form equation exists. Approximations or iterative methods to project are possible in some of these cases but are not optimal. One must also find a differentiable projection mapping to train the model. These properties may make it impractical for some applications.

5. Vertex Approach

For this approach we use the structure of the convex set Q directly. However, we require it to be a polytope defined by the vertices $v_1, v_2, \dots, v_V \in \mathbb{R}^{d_X}$ such that $Q = \text{conv}(v_1, v_2, \dots, v_V)$. Please refer to Section 3.2 for more details about polytopes. We interpret X to be implied by a weight process on these vertices.

Our model takes $X_t \in Q \subset \mathbb{R}^{d_X}$ as input. To ensure that its output always lies in Q, we "project" using the softmax activation function after the last layer, giving an output in Δ^{V-1} (see Section 3.2 about simplexes). The final output will be these weights used in a convex combination of $v_1, v_2, \dots, v_V$ to give the final prediction $Y \in Q$. In this sense, H_t from the model approximates an unnormalised weight process associated with X_t. After normalising via softmax, this then models the weight process that we assume implies X_t.

5.1 Setting

Let us formally specify the assumptions we make on the process X_t.

Assumption 7 *The process X_t, which takes values in $Q = \text{conv}(v_1, v_2, \dots, v_V)$, is defined through a weight process W_t taking values in Δ^{V-1} such that*

$$X_t = (W_t)_1 v_1 + (W_t)_2 v_2 + \cdots + (W_t)_V v_V.$$

Writing $\hat{W}_t := \mathbb{E}[W_t \mid \mathcal{A}_t]$, by linearity of expectation it holds that $\hat{X}_t = \sum_{i=1}^{V} (\hat{W}_t)_i v_i$. As such, we can equivalently verify the assumptions against W. Note that $\mathcal{A}_t$ is generated via observations of X, not of W.

Now let us introduce the softmax function. Given V vertices defining $Q \subset \mathbb{R}^{d_X}$ and an input $X \in \mathbb{R}^V$, the standard softmax activation function σ is defined for each coordinate $i = 1, \dots, V$ as

$$\sigma(X)_i = \frac{e^{X_i}}{\sum_{j=1}^{d_X} e^{X_j}}.$$

We also need a right inverse of it.

Lemma 11 *Let σ be the softmax activation function. A right inverse of σ is given by $\sigma^{-1} : \text{relint}(\Delta^{V-1}) \to \mathbb{R}^V$, $y_i \mapsto \ln(y_i) + C$ for $1 \leq i \leq V$ and any $C \in \mathbb{R}$.*

Proof Let σ^{-1} be as stated, and C some constant. Then, because $y \in \text{relint}(\Delta^{V-1})$ implies the sum of its coordinates is 1, we have for $1 \leq i \leq V$

$$\sigma(\sigma^{-1}(y))_i = \frac{e^{\ln(y_i)+C}}{\sum_{j=1}^{V} e^{\ln(y_i)+C}} = \frac{e^C}{e^C} \frac{y_i}{\sum_{j=1}^{V} y_j} = y_i.$$

■

We are now ready to discuss the modifications to the PD-NJ-ODE model, in particular to the readout function. We let

$$\tilde{g} : \Delta^{V-1} \to Q, \ \tilde{g}(\alpha_1, \alpha_2, \dots, \alpha_V) = \sum_{i=1}^{V} \alpha_i v_i$$

be the map from the weight space to the process space, by applying the weights to form a convex combination of the vertices $v_1, v_2, \dots, v_V$. We have $\tilde{g}(W_t) = X_t$ and $\tilde{g}(\hat{W}_t) = \sum_{i=1}^{V} (\hat{W}_t)_i v_i = \hat{X}_t \in Q$. The readout map of the PD-NJ-ODE model for this technique will therefore be $g := \tilde{g} \circ \sigma$.

Note that with this readout function the output can never be equal to one of the vertices (or any point where one vertex has weight 0), as softmax produces values strictly between 0 and 1.

5.2 Proving the PD-NJ-ODE Results

With the formulation of X as being defined via a weight process, we can adjust the assumptions slightly to be more specific. Additionally, we no longer need boundedness assumptions; instead, we just use a property of the model itself and that Q is bounded. This will become clear in the proof of Theorem 15.

Let us define $\xi_t^\alpha := \sigma^{-1}(\hat{W}_t + \alpha)$, $\alpha > 0$, with vector–scalar addition the usual way. This prevents issues when $\hat{W}$ contains a 0 coordinate. As in Section 2.3 of Krach et al. (2022), we can use the Doob–Dynkin Lemma (Taraldsen, 2018, Lemma 2) to show that there exist measurable functions $F_j^\xi : [0,T] \times [0,T] \times BV^c([0,T]) \to \mathbb{R}$ such that $\xi_{t,j}^\alpha = F_j^\xi(t, \tau(t), \tilde{X}^{\leq \tau(t)})$, where $\xi_t^\alpha = (\xi_{t,1}^\alpha, \dots, \xi_{t,V}^\alpha)$. Here $BV^c([0,T])$ refers to the set of $\mathbb{R}^{d_X}$-valued paths with bounded variation (see Definition 3.1 in Krach et al. (2022)) on $[0,T]$.

The following Assumption replaces Assumption 4 in Krach et al. (2022). All other assumptions are retained, although Assumption 5 is trivially true.

Assumption 8 *The partial derivative of F_j^{ξ} with respect to t, denoted by f_j^{ξ}, is continuous for all $1 \leq j \leq V$.*

Remark 12 *Since $(\xi_t^{\alpha})_j = \ln(\hat{W}_j + \alpha) + C$ for some constant C, the derivative in question is*

$$\frac{\partial}{\partial t}\sigma^{-1}(\hat{W} + \alpha)_j = \frac{1}{\hat{W}_j + \alpha}\frac{\partial}{\partial t}\hat{W}_j.$$

Thus, the assumption is satisfied if $\hat{W}$ is continuously differentiable with respect to t.

That $\hat{X}$ is the unique minimiser (up to indistinguishability, see Definition 4) of Ψ follows directly from the assumptions (Krach et al., 2022). Proving that the model can approximate $\hat{X}$ arbitrarily well is more challenging. There are two core results we need to construct the proof, which we express here as Propositions.

Proposition 13 *There exists a constant $C > 0$ independent of α such that for all $\alpha > 0$*

$$\left|Y_t - \hat{X}_t\right|_1 \leq C\alpha + C\sum_{j=1}^{V} \left|(H_t)_j - (\xi_t^{\alpha})_j\right|.$$

Proof We use that softmax is Lipschitz-continuous with constant L_σ (Gao and Pavel, 2018). Additionally, we note that the v_i are fixed, bounded, and finite in number. Then

$$\begin{aligned}
\left|Y_t - \hat{X}_t\right|_1 &= \left|\tilde{g}(\sigma(H_t)) - \tilde{g}(\hat{W}_t)\right|_1 \\
&= \left|\sum_{j=1}^{V} v_j(\sigma(H_t)_j - (\hat{W}_t)_j)\right|_1 \\
&\leq \sum_{j=1}^{V} |v_j|_1 \left|\sigma(H_t)_j + \alpha - (\hat{W}_t)_j - \alpha\right| \\
&\leq \sum_{j=1}^{V} |v_j|_1 \left(\alpha + \left|\sigma(H_t)_j - (\hat{W}_t)_j - \alpha\right|\right) \\
&= \sum_{j=1}^{V} |v_j|_1 \left(\alpha + \left|\sigma(H_t)_j - \sigma(\sigma^{-1}(\hat{W}_t + \alpha))_j\right|\right) \\
&\leq \sum_{j=1}^{V} |v_j|_1 \left(\alpha + L_\sigma \left|(H_t)_j - (\xi_t^{\alpha})_j\right|\right) \\
&\leq |v^{\max}|_1 V\alpha + |v^{\max}|_1 L_\sigma \sum_{j=1}^{V} \left|(H_t)_j - (\xi_t^{\alpha})_j\right|
\end{aligned}$$

where we define $v^{\max} := \operatorname{argmax}_{v \in \{v_1, \ldots, v_V\}} |v|_1$. ■

If we can now show that the latent variable H_t of the PD-NJ-ODE model can approximate ξ_t^α well, we can simplify the overall approximation proof. In the following we define

$$\mathrm{UAT}(\varepsilon) := \{X_T^* < 1/\varepsilon\} \cap \{n < 1/\varepsilon\} \cap \{\delta > \varepsilon\}$$

to be the event that means we can apply the universal approximation theorem.

Proposition 14 *Let $\varepsilon > 0$ be a constant. If $\mathrm{UAT}(\varepsilon)$ holds, then H_t from the PD-NJ-ODE model can approximate ξ_t^ε arbitrarily well. In particular, there exist $m \in \mathbb{N}$ and neural network weights $\theta^{*,m}$ such that $|H_t - \xi_t^\varepsilon|_1 \leq (T+1)\varepsilon$, i.e., H_t approximates ξ_t^α up to error $(T+1)\varepsilon$ for $\alpha = \varepsilon$.*

Proof This proof follows the latter half of the proof of Theorem 4.1 in Krach et al. (2022) closely. Some details are omitted for notational clarity and brevity; the reader may refer to the proof of Theorem 4.1 in Krach et al. (2022).

For ε as above, let $N_\varepsilon := \lceil 2(T+1)\varepsilon^{-2} \rceil$ (implying that $\lim_{\varepsilon \to 0} N_\varepsilon = \infty$) and $\mathcal{P}_\varepsilon$ be the closure of the set A_{N_ε} of Remark 3.11 in Krach et al. (2022), which is compact. For any $1 \leq j \leq V$, the function f_j^ξ is continuous by Assumption 8. As in Krach et al. (2022), there exists an $m_0 = m_0(\varepsilon) \in \mathbb{N}$ and a continuous function $\hat{f}_j^\xi$ such that

$$\sup_{(t,\tau,X) \in [0,T]^2 \times \mathcal{P}_\varepsilon} \left| f_j^\xi(t,\tau,X) - \hat{f}_j^\xi(t,\tau,\pi_{m_0}(X - X_0), X_0) \right| \leq \varepsilon/2.$$

Since the variation of functions in $\mathcal{P}_\varepsilon$ is bounded, the closure of $\pi_{m_0}(\mathcal{P}_\varepsilon)$, denoted by Π_ε, is compact. The universal approximation theorem for neural networks (Hornik, 1991) implies that there exists an $m_1 = m_1(\varepsilon) \in \mathbb{N}$ and neural network weights $\tilde{\theta}_1^{*,m_1}$ such that, for $1 \leq j \leq V$, $\hat{f}_j^\xi$ is approximated up to $\varepsilon/2$ by the j-th coordinate of the neural network (under the aforementioned optimal weights) on the compact set $[0,T]^2 \times \Pi_\varepsilon$.

Combining this result with the first approximation statement using the triangle inequality, we attain

$$\sup_{(t,\tau,X) \in [0,T]^2 \times \mathcal{P}_\varepsilon} \left| f_j^\xi(t,\tau,X) - \tilde{f}^\xi_{\tilde{\theta}_1^{*,m_1},j}(t,\tau,\pi_{m_0}(X - X_0), X_0) \right| \leq \varepsilon.$$

Applying similar logic using the universal approximation theorem gives us

$$\sup_{(t,X) \in [0,T] \times \mathcal{P}_\varepsilon} \left| F_j^\xi(t,t,X) - \tilde{\rho}_{\tilde{\theta}_2^{*,m_2},j}(t,\pi_{m_0}(X - X_0), X_0) \right| \leq \varepsilon$$

for $1 \leq j \leq V$, width $m_2 = m_2(\varepsilon) \in \mathbb{N}$, and weights $\tilde{\theta}_2^{*,m_2}$.

Thus for $m := \max\{m_0, m_1, m_2\}$ the PD-NJ-ODE model can approximate F_j^ξ and f_j^ξ up to ε when $\mathrm{UAT}(\varepsilon)$ holds (which implies the path $\tilde{X}^{\leq \tau(t)} - X_0$ is an element of A_{N_ε}).

For the final part of this proof, we apply these results to show that $|H_t - \xi_t^\varepsilon|$ is bounded by an arbitrarily small constant. Applying the approximation bounds for f_j^ξ and F_j^ξ we have

$$|(H_t)_j - (\xi_t^\varepsilon)_j| \leq \left| (H_{\tau(t)})_j - (\xi_{\tau(t)}^\varepsilon)_j \right|$$

$$
\begin{aligned}
&+ \int_{\tau(t)}^{t} \left| f^{\xi}_{\theta_1^{*,m_1},j}\left(H_{s-}, s, \tau(t), \pi_m(\tilde{X}^{\leq \tau(t)} - X_0), X_0\right) - f^{\xi}_j(s, \tau(t), \tilde{X}^{\leq \tau(t)}) \right|_1 ds \\
&= \left| F^{\xi}_j(t, t, X) - \tilde{\rho}_{\tilde{\theta}_2^{*,m_2},j}(t, \pi_{m_0}(X - X_0), X_0) \right| \\
&+ \int_{\tau(t)}^{t} \left| f^{\xi}_{\theta_1^{*,m_1},j}\left(H_{s-}, s, \tau(t), \pi_m(\tilde{X}^{\leq \tau(t)} - X_0), X_0\right) - f^{\xi}_j(s, \tau(t), \tilde{X}^{\leq \tau(t)}) \right|_1 ds \\
&\leq \varepsilon + \int_{\tau(t)}^{t} \varepsilon \, ds \leq (T+1)\varepsilon
\end{aligned}
$$

which is an arbitrarily small upper bound as T is a constant. ■

Finally, we combine these results to prove that the model can approximate $\hat{X}$ arbitrarily well.

Theorem 15 *The PD-NJ-ODE model approximates $\hat{X}$ arbitrarily well using the vertex approach.*

Proof We need to bound the distance between $Y_t^{\theta_m^*}$ and $\hat{X}$. As in Krach et al. (2022), we distinguish two cases. If we can apply the universal approximation theorem, Proposition 14 gives us an upper bound which we can translate into a bound for the distance from $\hat{X}$ using Proposition 13. To begin, let us fix an $\varepsilon > 0$. Proposition 13 states that there exists a positive C such that

$$
\left| Y_t^{\theta_m^*} - \hat{X}_t \right|_1 \leq C\varepsilon + C \sum_{j=1}^{V} \left| (H_t)_j - (\xi_t^{\varepsilon})_j \right|,
$$

which we can then combine with Proposition 14 to bound the summands, giving us the final upper bound

$$
\left| Y_t^{\theta_m^*} - \hat{X}_t \right|_1 \leq C\varepsilon + CV(T+1)\varepsilon.
$$

Otherwise, if the universal approximation theorems cannot be applied, we know that the output of the neural network is always in Q, so the general upper bound of $|Y_t - \hat{X}_t|_1$ is at most the largest distance between any two points in Q. Taking a naive bound, we use $B_Q := 2 \max_{q \in Q} |q|_1$. This property also lets us Assumption 4 on X, which requires that F and f are bounded.

With that in mind, for all $t \in [0, T]$ it holds that

$$
\begin{aligned}
\left| Y_t^{\theta_m^*} - \hat{X}_t \right|_1 &= \left| Y_t^{\theta_m^*} - \hat{X}_t \right|_1 \mathbb{1}_{\mathrm{UAT}(\varepsilon)} + \left| Y_t^{\theta_m^*} - \hat{X}_t \right|_1 \mathbb{1}_{\mathrm{UAT}(\varepsilon)^c} \\
&\leq (C\varepsilon + CV(T+1)\varepsilon)\mathbb{1}_{\mathrm{UAT}(\varepsilon)} + B_Q \mathbb{1}_{\mathrm{UAT}(\varepsilon)^c}.
\end{aligned}
$$

Moreover, by equivalence of the 1- and 2-norm, there exists a constant $c > 0$ such that for all $t \in [0, T]$

$$
\begin{aligned}
\left| Y_t^{\theta_m^*} - \hat{X}_t \right|_2 &\leq c(C\varepsilon + CV(T+1)\varepsilon) + cB_Q \mathbb{1}_{\mathrm{UAT}(\varepsilon)^c} \\
&=: c_m.
\end{aligned}
$$

Until now, we have fixed an $\varepsilon > 0$ and demonstrated that there exists some $m \in \mathbb{N}$ such that the neural network approximation bounds hold with $(T+1)\varepsilon$-error. However, what we actually need to show is that this error converges to 0 while increasing the truncation level and network size m. To that end, we begin by defining $\varepsilon_m \geq 0$ to be the smallest number such that the above bounds hold with error ε_m when the model has truncation level m and weights in Θ_m.

Since increasing m cannot worsen the approximations (the new weights can be set to 0), we have $\varepsilon_{m_1} \geq \varepsilon_{m_2}$ for any $m_1 \leq m_2$. More generally, $(\varepsilon_m)_{m\geq 0}$ is a decreasing sequence, so our previous derivations prove that $\lim_{m\to\infty} \varepsilon_m = 0$. Simultaneously this also implies that the approximation $\xi_t^{\varepsilon_m} = \sigma^{-1}(\hat{W}_t + \varepsilon_m)$ converges to $\sigma^{-1}(\hat{W}_t)$ as $m \to \infty$, although $\sigma^{-1}(\hat{W}_t)$ is not necessarily defined if a coordinate of $\hat{W}_t 0$ is 0 (cf. the beginning of this section).

In the following we denote by $\theta_m^* \in \Theta_m$ the optimal weights in Θ_m to approximate F_j^ξ, f_j^ξ.

Following equation (17) from Krach et al. (2022), we have

$$\begin{aligned}
\min_{Z\in\mathbb{D}} \Psi(Z) &\leq \Phi(\theta_m^{\min}) \leq \Phi(\theta_m^*) \\
&= \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\left|M_{t_i}\odot(X_{t_i} - Y_{t_i}^{\theta_m^*})\right|_2 + \left|M_{t_i}\odot(Y_{t_i}^{\theta_m^*} - Y_{t_{i-}}^{\theta_m^*})\right|_2\right)^2\right] \qquad (3)\\
&\leq \Psi(\hat{X}) + c\left(\mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[c_m^2\right] + \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[c_m^2\right]^{1/2}\right).
\end{aligned}$$

We also know that

$$\mathbb{1}_{\mathrm{UAT}(\varepsilon_m)^c} \leq \mathbb{1}_{\{X_T^* \geq 1/\varepsilon_m\}} + \mathbb{1}_{\{n\geq 1/\varepsilon_m\}} + \mathbb{1}_{\{\delta\leq\varepsilon_m\}} \xrightarrow[m\to\infty]{\mathbb{P}\times\tilde{\mathbb{P}}-a.s.} 0.$$

Now we need to show that $\mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}[c_m^2] \xrightarrow{m\to\infty} 0$. For a large enough c, independent of ε_m and m, we have

$$\mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[c_m^2\right] \leq c\varepsilon_m^2 + c\mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[B_Q^2\,\mathbb{1}_{\mathrm{UAT}(\varepsilon_m)^c}\right] \xrightarrow{m\to\infty} 0$$

by dominated convergence (Krach et al., 2022).

The (short) remainder of this proof is identical to the one in Krach et al. (2022), so we do not repeat it here. ■

Having now proven Theorem 4.1, we still need to prove the second half of the convergence proof (Krach et al., 2022, Theorem 4.4). For Lemma 4.5 (Krach et al., 2022), which is used in the proof of Theorem 4.4, we need that the readout neural network is continuous in weight θ. For the beginning of the proof of Theorem 4.4 we need that it is also bounded. The following Proposition demonstrates this.

Proposition 16 *The readout function $g = \tilde{g} \circ \sigma$ is continuous and bounded.*

Proof The softmax activation function σ is clearly bounded and continuous; the mapping $\tilde{g}$ is simply a linear transformation (from weights to point), so g is continuous and bounded.

■

Thus, the rest of the proof of Theorem 4.4 can remain unchanged, and the convergence proof is complete.

5.3 Discussion

The vertex approach provides a robust adaptation of the PD-NJ-ODE model that directly takes advantage of the fact that we know X lies in Q, here a polytope. Additionally, its architecture closely matches the assumptions on how X is generated. It also requires no projection function, in contrast to the optimal projection technique.

Although it is limited to polytopes, it could approximate a general bounded convex set via an inscribed polytope which can be refined to an arbitrary extent by adding vertices.

6. Adding a Loss Term

Like the optimal projection approach, this method does not alter PD-NJ-ODE's framework much. In fact, only the objective function is changed; the model remains as is. Here we add an additional term to the objective function to penalise predictions that lie outside of Q. Although the model will learn $\hat{X}$ (which lies in Q) eventually, this new term directly encourages the model to predict inside Q.

Let us first introduce the term, then prove that the PD-NJ-ODE results still hold.

6.1 Setting

We create a new objective function Ψ' by adding a loss term to the original objective function Ψ, see (2). The new objective function is defined as

$$\Psi' : \mathbb{D} \to \mathbb{R},$$

$$\begin{aligned} Z \mapsto \Psi'(Z) &:= \lambda \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\int_0^T l(Z_t)\ dt\right] + \Psi(Z) \\ &= \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\lambda \int_0^T l(Z_t)\ dt + \frac{1}{n}\sum_{i=1}^{n}\left(|M_i \odot (X_{t_i} - Z_{t_i})|_2 + |M_i \odot (Z_{t_i} - Z_{t_i-})|_2\right)^2\right], \end{aligned} \tag{4}$$

Here $\lambda \geq 0$ regularises the additional loss terms given by l; we recover the original loss function Ψ if $\lambda = 0$.

The new term l should penalise predictions if they are outside of Q. The following formalises the assumptions on l.

Assumption 9 *The penalising function $l : \mathbb{R}^{d_X} \to \mathbb{R}$ is*

1. *continuous;*
2. *non-negative;*
3. *equal to 0 if its argument is in Q, i.e., $l(Z) = 0$ if $Z \in Q$; and*
4. *for all t, $l(Y_t^{\theta_m^*})$ is almost-surely upper bounded by the random variable $C(c_m + 1)^2$ for an arbitrary $C > 0$.*

All of these assumptions are natural except for the final one, which is needed for the following proof.

Remark 17 *With $\hat{X}_t = \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}[X_t \mid \mathcal{A}_t]$, $l(\hat{X}_t) = 0$ for all t. To show this, we can apply Proposition 1 to $\hat{X}_t$ to conclude that $\hat{X}_t \in Q$ for all t, which then implies $l(\hat{X}_t) = 0$ by Assumption 9.*

Remark 18 *That l is upper bounded by $C(c_m + 1)^2$ implies it is integrable, since c_m is square-integrable (Krach et al., 2022).*

A standard example of a penalising function satisfying this Assumption is the squared Euclidean distance from Q $l(Z_t) = \min_{q\in Q} \|Z_t - q\|_2^2$. The first three properties are clearly true; for the final one we can use a result from Krach et al. (2022). To be specific, they provide the bound $|Y_t^{\theta^*_m} - \hat{X}_t|_2 \leq c_m$. Squaring this, we can find the relation

$$\min_{q\in Q} \|Y_t - q\|_2^2 \leq \left|Y_t^{\theta^*_m} - \hat{X}_t\right|_2^2 \leq c_m^2,$$

which demonstrates the assumption is satisfied for the squared Euclidean distance from Q.

Via norm equivalence, this also means that all norms (and their squares) are also valid candidates for l. Multiplying them by a non-negative constant factor is also clearly fine.

6.2 Proving the PD-NJ-ODE Results

That the new loss function Ψ' is simply an addition of $\lambda\mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}[\int_0^T l(Z_t)\ dt]$ to the existing objective function makes it simple to verify that the convergence guarantees of the PD-NJ-ODE framework still hold.

For the previous techniques, to verify Theorem 4.1 (Krach et al., 2022) we only had to show that the modified PD-NJ-ODE model can approximate $\hat{X}$ arbitrarily well. This is still necessary, but since we change the objective function, we must also verify the first part of Theorem 4.1. We do so in the following Theorem.

Theorem 19 *$\hat{X}$ is the unique minimiser of Ψ' up to indistinguishability, as in Definition 4.*

Proof This follows the beginning of the proof of Theorem 4.1 (Krach et al., 2022). Using that l is non-negative and equal to 0 for values in Q, we can follow the derivation in the paper and write

$$\begin{aligned}
\Psi'(\hat{X}) &= \lambda\mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\int_0^T l(\hat{X}_t)\ dt\right] + \Psi(\hat{X}) \\
&= 0 + \Psi(\hat{X}) \\
&\leq \min_{Z\in\mathbb{D}} \Psi(Z) \\
&\leq \min_{Z\in\mathbb{D}} \lambda\mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\int_0^T l(Z_t)\ dt\right] + \Psi(Z) \\
&= \min_{Z\in\mathbb{D}} \Psi'(Z).
\end{aligned}$$

This shows that $\hat{X}$ is a minimiser of Ψ'. To show uniqueness, use that $\Psi' \geq \Psi$ and $\Psi'(\hat{X}) = \Psi(\hat{X})$. With these we can follow the derivation exactly as in the original paper to finish the proof. ■

Now we prove the second part of Theorem 4.1.

Theorem 20 *The PD-NJ-ODE framework with an additional penalising term approximates $\hat{X}$ arbitrarily well.*

Proof Following the steps in the last part of the proof of Theorem 4.1 (Krach et al., 2022), we have

$$\begin{aligned}
\min_{Z \in \mathbb{D}} \Psi'(Z) &\leq \Phi'(\theta_m^{\min}) \leq \Phi'(\theta_m^*) \\
&= \lambda \mathbb{E}_{\mathbb{P} \times \tilde{\mathbb{P}}} \left[\int_0^T l(Y_t^{\theta_m^*}) \, dt \right] + \Phi(\theta_m^*) \\
&\leq \lambda \mathbb{E}_{\mathbb{P} \times \tilde{\mathbb{P}}} \left[\int_0^T l(Y_t^{\theta_m^*}) \, dt \right] + \Psi(\hat{X}) + c \left(\mathbb{E}_{\mathbb{P} \times \tilde{\mathbb{P}}} \left[c_m^2 \right] + \mathbb{E}_{\mathbb{P} \times \tilde{\mathbb{P}}} \left[c_m^2 \right]^{1/2} \right)
\end{aligned}$$

where we use the bound for $\Phi(\theta_m^*)$ and c_m as in their paper. The variable $c > 0$ is a suitable constant. Assumption 9 implies that $\lambda T C \mathbb{E}_{\mathbb{P} \times \tilde{\mathbb{P}}}[(c_m + 1)^2]$ is an upper bound for the first term, while also assuring us that it is integrable (see Remark 18). Also using that $\Psi'(\hat{X}) = \Psi(\hat{X})$ since $l(\hat{X}) = 0$, we find that

$$\min_{Z \in \mathbb{D}} \Psi'(Z) \leq \Psi'(\hat{X}) + \lambda T C \mathbb{E}_{\mathbb{P} \times \tilde{\mathbb{P}}} \left[(c_m + 1)^2 \right] + c \left(\mathbb{E}_{\mathbb{P} \times \tilde{\mathbb{P}}} \left[c_m^2 \right] + \mathbb{E}_{\mathbb{P} \times \tilde{\mathbb{P}}} \left[c_m^2 \right]^{1/2} \right).$$

The authors show that $\mathbb{E}_{\mathbb{P} \times \tilde{\mathbb{P}}}[c_m^2] \xrightarrow{m \to \infty} 0$, so by noting that

$$\Phi'(\theta_m^{\min}) - \Psi'(\hat{X}) = \Phi'(\theta_m^{\min}) - \Psi(\hat{X}) \geq \Phi(\theta_m^{\min}) - \Psi(\hat{X})$$

we can follow the rest of the proof (without modification) to show that $\hat{X}$ can be approximated arbitrarily well by the PD-NJ-ODE framework even with the additional loss term. ■

To fully prove convergence, we still need to show that the Monte Carlo approximation of the new loss function converges to the new theoretical loss function; this is Theorem 4.4 in Krach et al. (2022). Our new term is only relevant at the end of the proof of Theorem 4.4, where the authors use arguments from Theorem 4.1. Since we have just shown that Theorem 4.1 still holds, there is nothing more to be done. The proof of Theorem 4.4 is unchanged.

6.3 Discussion

The benefit of this technique of adding an additional loss term is that it can extend to most variants or extensions of the PD-NJ-ODE framework. For example, it can be used with the optimal projection approach by being applied to the predictions before projection. Additionally, the restrictions on the penalising function l are lax, so it can be selected as a

hyperparameter depending on the dataset. In a sense it is like a regularising term used in traditional loss functions for deep learning.

One hurdle is that the penalising function must measure the distance of a point to Q in some sense. Since the easiest way to do this is to find the optimal projection and use the distance to that, this technique can suffer from the same drawbacks as the optimal projection approach. A second minor hurdle is the need to discretise the integral in the additional term, although this is straightforward to do if time steps are provided.

Finally, it is important to note, however, that this technique cannot truly enforce that the output will always be in Q. There is no form of projection, this new loss term can only encourage the prediction to fall into Q. It is a soft constraint, not a hard constraint.

7. Processes Satisfying the Assumptions

There is little literature concerning the stochastic processes confined to a compact set, especially those with a closed form. Here we provide a few examples and show that they satisfy the assumptions required by the NJ-ODE framework. We will use these for experiments.

7.1 Reflected Brownian Motion

A reflected Brownian motion (RBM), as the name suggests, describes a process whereby the particle moves according to some Brownian motion until it "collides" with a boundary, after which it is reflected in the opposite direction.

We will consider the case where the underlying Brownian motion has drift and diffusion. Because of this, the natural intuition of an RBM describing the motion of a particle on a surface of water surrounded by some wall is not entirely correct. With drift the particle will tend toward either the lower or upper boundary, depending on whether the drift is negative or positive, respectively.

Three broad, physical examples come out of this. A particle subject to gravity and contained between an upper and lower boundary will tend toward the lower boundary and can therefore be represented as an RBM with negative drift. Similarly, one can imagine the behaviour of a helium balloon in a room with floor and ceiling as an RBM with positive drift. A reflected Brownian motion with drift 0 could represent a particle moving on a surface with two boundaries, as long as the particle receives no impulse except if perpendicular to both boundaries.

As for more sophisticated examples, we first note that an RBM can easily represent a positivity constraint. As such, we could model a stock's price via an RBM with positive drift: stock prices should in general tend upwards, and they cannot be negative. In a similar vein, the price of a commodity or service with a price cap could be modelled by an RBM (Veestraeten, 2004). Finally, population growth in many places could be modelled as an RBM with positive drift (Veestraeten, 2004).

Although the intuition is clear, it is not trivial to mathematically define what it means to "collide" or to "reflect". Let us formulate this now, according to Veestraeten (2004).

Definition 21 *A reflected Brownian motion (RBM) is a Brownian motion X_t with drift $\mu \in \mathbb{R}$ and diffusion $\sigma \in \mathbb{R}_{>0}$ satisfying*

$$dX_t = \mu dt + \sigma dW_t$$

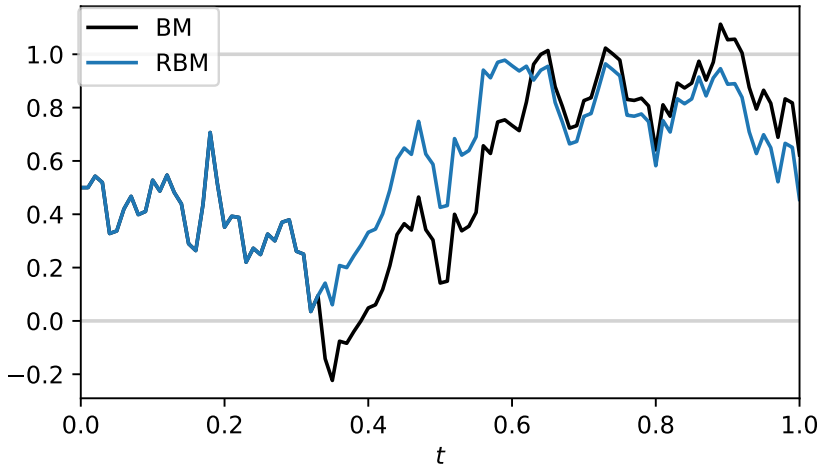

Figure 1: One path of a Brownian motion (black) and the reflected Brownian motion it drives (blue). The boundaries are at 0 and 1, and we use $\mu = 0$, $\sigma = 1$.

for $t \geq t_0$, such that it is instantaneously reflected at a lower and an upper boundary c and d. In particular, the transition or conditional probability density function $p(X, t; X_0, t_0) := \mathbb{P}[X_t \mid X_{t_0}, t_0]$ satisfies the Fokker-Planck forward equation with $g = \frac{1}{2}\sigma^2$:

$$g\frac{\partial^2 p(X,t;X_0,t_0)}{\partial X^2} - \mu\frac{\partial p(X,t;X_0,t_0)}{\partial X} = \frac{\partial p(X,t;X_0,t_0)}{\partial t}$$

for $c < X, X_0 < d$ and $t > t_0$.

Additionally, it satisfies two boundary conditions

$$\lim_{X \downarrow c} \; g\frac{\partial p(X,t;X_0,t_0)}{\partial X} - \mu p(X,t;X_0,t_0) = 0 \text{ and}$$
$$\lim_{X \uparrow d} \; -\left(g\frac{\partial p(X,t;X_0,t_0)}{\partial X} - \mu p(X,t;X_0,t_0)\right) = 0,$$

as well as the initial condition

$$\lim_{t \downarrow t_0} \; p(X,t;X_0,t_0) = \delta(X - X_0)\delta(t - t_0).$$

Here $\delta(\cdot)$ is the Dirac delta function.

The boundary conditions, one for each, ensure that the probability density function we find will place no mass outside of the boundaries. The initial condition ensures that the probability of the process being at the start position X_0 at the start time t_0 tends to 1 as t goes to t_0.

One-sided reflected Brownian motions (where $c = -\infty$ or $d = \infty$) can easily be extracted from the formulation by taking the appropriate limit.

So, to finally formalise what it means to "collide" or "reflect", we say that to collide is to near a boundary in the limit, while to reflect is to have zero probability of touching a boundary.

7.1.1 Verifying Assumptions

With a lower bound c and upper bound d, we can take $Q = [c, d] \subset \mathbb{R}$. Assumption 5 is immediate, since a RBM is bounded. For the same reason, boundedness assumptions on F and f in Assumption 4 can be removed for approaches that guarantee their predictions lie in Q; we can use the direct bound of $B_Q := 2 \max_{q \in Q} |q|_1$ instead of $B(X_t^* + 1)^p$. See the proof in Section 5.2 for this.

It therefore remains to verify that the F_j are continuous and differentiable in t such that their partial derivatives with respect to t, denoted by f_j, are again continuous. This then shows that Assumption 4 is fully satisfied.

To begin, we must first find an analytical expression for $p(X, t; X_0, t_0)$, the conditional density. Such an expression was first found by Svensson (1991, Appendix A.3), but it is relatively difficult to work with.[6] Instead, we use the equations given by Veestraeten (2004), which were derived using different techniques. Although still laborious, it is easier to verify the assumptions using these.

According to Equation 11 in Veestraeten (2004), the conditional probability density function for a reflected Brownian motion with lower and upper boundaries c and d respectively, provided with drift μ and diffusion $\sigma > 0$, is given by

$$\begin{aligned} p(X, t; X_0, t_0) = \alpha &\sum_{n=-\infty}^{\infty} \exp(T_{1,1}) \exp(-T_{1,2}) \\ &+ \alpha \sum_{n=-\infty}^{\infty} \exp(-T_{2,1}) \exp(-T_{2,2}) \\ &- \beta \sum_{n=0}^{\infty} \exp(T_{3,1}) \Phi_{\mathcal{N}}(-T_{3,2}) \\ &+ \beta \sum_{n=0}^{\infty} \exp(T_{4,1}) \Phi_{\mathcal{N}}(T_{4,2}), \end{aligned} \tag{5}$$

where $\Phi_{\mathcal{N}}$ is the CDF of the standard normal distribution. We use its symmetry to write $\Phi_{\mathcal{N}}(-T_{3,2})$ instead of $1 - \Phi_{\mathcal{N}}(T_{3,2})$ in the third sum.

The terms are given by

$$\begin{aligned} \alpha &= \frac{1}{\sigma\sqrt{2\pi(t - t_0)}}, \quad \beta = \frac{2\mu}{\sigma^2}, \\ T_{1,1} &= \frac{2\mu n(c - d)}{\sigma^2}, \\ T_{1,2} &= \frac{(X + 2n(d - c) - X_0 - \mu(t - t_0))^2}{2\sigma^2(t - t_0)}, \\ T_{2,1} &= \frac{2\mu(nd - (n + 1)c + X_0)}{\sigma^2}, \\ T_{2,2} &= \frac{(2nd - 2(n + 1)c + X_0 + X - \mu(t - t_0))^2}{2\sigma^2(t - t_0)}, \end{aligned}$$

6. The equation is also slightly incorrect, see the corrected version in de Jong (1994).

$$
\begin{aligned}
T_{3,1} &= \frac{2\mu(nd - (n+1)c + X)}{\sigma^2}, \\
T_{3,2} &= \frac{\mu(t - t_0) + 2nd - 2(n+1)c + X_0 + X}{\sigma\sqrt{t - t_0}}, \\
T_{4,1} &= \frac{2\mu(nc - (n+1)d + X)}{\sigma^2}, \text{and} \\
T_{4,2} &= \frac{\mu(t - t_0) - 2(n+1)d + 2nc + X_0 + X}{\sigma\sqrt{t - t_0}}.
\end{aligned}
$$

We can now derive the conditional expectation.

Proposition 22 *Let X_t be a reflected Brownian motion. The conditional expectation is*

$$
\begin{aligned}
\mathbb{E}[X_t \mid X_0, t_0] = \alpha &\sum_{n=-\infty}^{\infty} \exp(T_{1,1}) \int_c^d X \exp(-T_{1,2})\ dX \\
&+ \alpha \sum_{n=-\infty}^{\infty} \exp(-T_{2,1}) \int_c^d X \exp(-T_{2,2})\ dX \\
&- \beta \sum_{n=0}^{\infty} \int_c^d X \exp(T_{3,1}) \Phi_{\mathcal{N}}(-T_{3,2})\ dX \\
&+ \beta \sum_{n=0}^{\infty} \int_c^d X \exp(T_{4,1}) \Phi_{\mathcal{N}}(T_{4,2})\ dX.
\end{aligned}
$$

The integrals can all be expressed analytically, as shown in the ensuing proof.

Proof We simply integrate $Xp(X, t; X_0, t_0)$ with respect to X, which has range (c, d). For ease of notation, we compute the indefinite integrals (omitting the constant); they should be evaluated for $X = c$ and $X = d$.

Many terms are expressed using the error function

$$
\operatorname{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2}\ dt,
$$

which has the following relation to the CDF of the standard normal distribution:

$$
\Phi_{\mathcal{N}}(x) = \frac{1}{2}\left(\operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right).
$$

For completeness, its derivative is $\frac{d}{dx}\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\exp(-x^2)$.

The following terms are constant with respect to X: α, β, $T_{1,1}$ and $T_{2,1}$. Let us now integrate the others. The first is given by

$$
\begin{aligned}
&\int X \exp(-T_{1,2})\ dX = \\
&-\frac{\sqrt{\pi}}{2}(2n(d-c) - X_0 - \mu(t - t_0))\sqrt{2\sigma^2(t - t_0)}\operatorname{erf}\left(\frac{X + 2n(d-c) - X_0 - \mu(t - t_0)}{\sqrt{2\sigma^2(t - t_0)}}\right)
\end{aligned}
$$

$$- \sigma^2(t-t_0)\exp(-T_{1,2});$$

the second is almost identical as

$$\begin{aligned}\int X\exp(-T_{2,2})\ dX = \\ -\frac{\sqrt{\pi}}{2}(2nd-2(n+1)c+X_0-\mu(t-t_0))\sqrt{2\sigma^2(t-t_0)} \\ \cdot\operatorname{erf}\left(\frac{2nd-2(n+1)c+X_0+X-\mu(t-t_0)}{\sqrt{2\sigma^2(t-t_0)}}\right) \\ -\sigma^2(t-t_0)\exp(-T_{2,2}).\end{aligned}$$

The final two sums are more involved. Let us begin with the third. Here

$$\int X\exp(T_{3,1})\Phi_{\mathcal{N}}(-T_{3,2})\ dX = \int X\exp\left(A_3+\frac{2\mu}{\sigma^2}X\right)\Phi_{\mathcal{N}}\left(-B_3-\frac{1}{\sigma\sqrt{t-t_0}}X\right)\ dX,$$

where we define

$$\begin{aligned}A_3 &= \frac{2\mu(nd-(n+1)c)}{\sigma^2} \text{ and} \\ B_3 &= \frac{\mu(t-t_0)+2nd-2(n+1)c+X_0}{\sigma^2\sqrt{t-t_0}}.\end{aligned}$$

Partial integration gives us the following equations:

$u_3 = \Phi_{\mathcal{N}}\left(-B_3-\frac{1}{\sigma\sqrt{t-t_0}}X\right)$ $\quad u_3' = \frac{1}{\sqrt{2\pi}}\frac{1}{\sigma\sqrt{t-t_0}}\exp\left(-\frac{1}{2}\left(\frac{1}{\sigma\sqrt{t-t_0}}X-B_3\right)^2\right)$

$v_3 = \frac{\sigma^2}{2\mu}\exp\left(\frac{2\mu}{\sigma^2}X\right)\left(X-\frac{\sigma^2}{2\mu}\right)$ $\quad v_3' = X\exp\left(\frac{2\mu}{\sigma^2}X\right)$

Continuing with the third sum, we have

$$\begin{aligned}&\int X\exp\left(A_3+\frac{2\mu}{\sigma^2}X\right)\Phi_{\mathcal{N}}\left(-B_3-\frac{1}{\sigma\sqrt{t-t_0}}X\right)\ dX \\ &= \exp(A_3)\left(u_3v_3-\int u_3'v_3\ dX\right) \\ &= \exp(A_3)\left(u_3v_3-\frac{1}{\sigma\sqrt{t-t_0}}\frac{1}{\sqrt{2\pi}}\frac{\sigma^2}{2\mu}\int I_3\ dX\right),\end{aligned}$$

with

$$I_3 = \exp\left(-\frac{1}{2}\left(\frac{1}{\sigma\sqrt{t-t_0}}X-B_3\right)^2\right)\exp\left(\frac{2\mu}{\sigma^2X}\right)\left(X-\frac{\sigma^2}{2\mu}\right)\ dX.$$

The integral can be evaluated as

$$\begin{aligned}\int I_3\ dX = \\ \frac{1}{\left(\sigma\sqrt{t-t_0}\right)^3}\left(-\frac{1}{\sigma\sqrt{t-t_0}}\exp\left(X\left(\frac{2\mu}{\sigma^2}-\frac{1}{2}\frac{1}{\left(\sigma\sqrt{t-t_0}\right)^2}\right)+\frac{1}{\sigma\sqrt{t-t_0}}B_3X-\frac{1}{2}B_3^2\right)\right.\end{aligned}$$

$$-\frac{\sigma^2}{2\mu}\sqrt{\frac{\pi}{2}}\left(-\frac{1}{\left(\sigma\sqrt{t-t_0}\right)^2}+\frac{1}{\sigma\sqrt{t-t_0}}\frac{2\mu}{\sigma^2}B_3+\left(\frac{2\mu}{\sigma^2}\right)^2\right)$$
$$\cdot\exp\left(\left(\sigma\sqrt{t-t_0}\right)^2\cdot\frac{2\mu}{\sigma^2}\left(\frac{1}{\sigma\sqrt{t-t_0}}B_3+\frac{1}{2}\frac{2\mu}{\sigma^2}\right)\right)$$
$$\cdot\operatorname{erf}\left(\frac{1}{\sqrt{2}}\left(\frac{2\mu}{\sigma^2}\sigma\sqrt{t-t_0}-\frac{1}{\sigma\sqrt{t-t_0}}X+B_3\right)\right)\Bigg).$$

We can therefore write

$$\int X\exp(T_{3,1})\Phi_{\mathcal{N}}(-T_{3,2})\;dX=\exp(A_3)\left(u_3v_3-\frac{1}{\sigma\sqrt{t-t_0}}\frac{1}{\sqrt{2\pi}}\frac{\sigma^2}{2\mu}\int I_3\;dX\right).$$

The fourth and final sum uses a similar approach. Here we have

$$\int X\exp(T_{4,1})\Phi_{\mathcal{N}}(T_{4,2})\;dX=\int X\exp\left(A_4+\frac{2\mu}{\sigma^2}X\right)\Phi_{\mathcal{N}}\left(B_4+\frac{1}{\sigma\sqrt{t-t_0}}X\right)\;dX,$$

where we let

$$A_4=\frac{2\mu(nc-(n+1)d)}{\sigma^2}\text{ and}$$
$$B_4=\frac{\mu(t-t_0)-2(n+1)d+2nc+X_0}{\sigma\sqrt{t-t_0}}.$$

Performing partial integration, we have

$u_4=\Phi_{\mathcal{N}}\left(\frac{1}{\sigma\sqrt{t-t_0}}X+B_4\right)$ $\quad u_4'=\frac{1}{\sqrt{2\pi}}\frac{1}{\sigma\sqrt{t-t_0}}\exp\left(-\frac{1}{2}\left(\frac{1}{\sigma\sqrt{t-t_0}}X+B_4\right)^2\right)$

$v_4=v_3$ $\quad v_4'=v_3'$

Continuing now with the fourth sum, we get

$$\int X\exp\left(A_4+\frac{2\mu}{\sigma^2}X\right)\Phi_{\mathcal{N}}\left(B_4+\frac{1}{\sigma\sqrt{t-t_0}}X\right)\;dX$$
$$=\exp(A_4)\left(u_4v_4-\int u_4'v_4\;dX\right)$$
$$=\exp(A_4)\left(u_4v_4-\frac{1}{\sqrt{2\pi}}\frac{1}{\sigma\sqrt{t-t_0}}\frac{\sigma^2}{2\mu}\int I_4\;dX\right)$$

where

$$I_4=\exp\left(-\frac{1}{2}\left(\frac{1}{\sigma\sqrt{t-t_0}}X+B_4\right)^2\right)\exp\left(\frac{2\mu}{\sigma^2}X\right)\left(X-\frac{\sigma^2}{2\mu}\right).$$

This integrand is the same as I_3 except for some signs and the B term; it can therefore be evaluated to a similar expression, namely

$$\int I_4\;dX=$$
$$\frac{1}{\left(\sigma\sqrt{t-t_0}\right)^3}\left(-\frac{1}{\sigma\sqrt{t-t_0}}\exp\left(X\left(\frac{2\mu}{\sigma^2}+\frac{1}{2}\frac{1}{\left(\sigma\sqrt{t-t_0}\right)^2}\right)-\frac{1}{\sigma\sqrt{t-t_0}}B_4X-\frac{1}{2}B_4^2\right)\right.$$

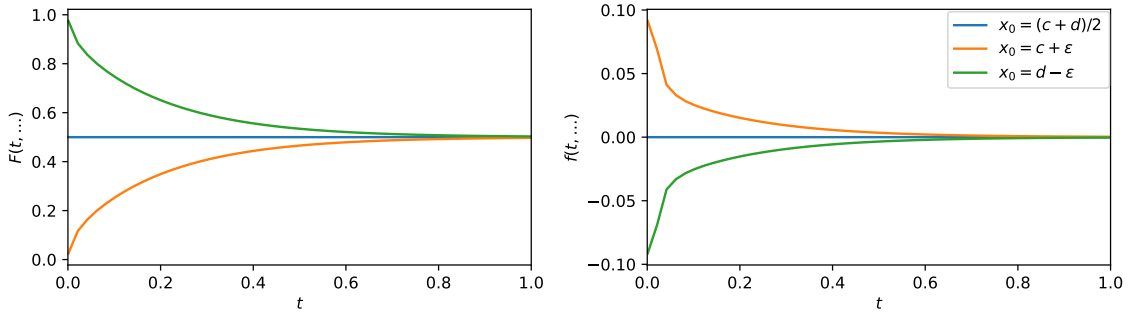

Figure 2: Numerical approximations of F and f of an RBM plotted against time, for various x_0. We use $\mu = 0$, $\sigma = 1$, $c = 0$, $d = 1$ and truncate the sums to 10 terms for the first two sums, 5 terms for the other two.

$$
\begin{aligned}
&-\frac{\sigma^2}{2\mu}\sqrt{\frac{\pi}{2}}\left(-\frac{1}{\left(\sigma\sqrt{t-t_0}\right)^2}-\frac{1}{\sigma\sqrt{t-t_0}}\frac{2\mu}{\sigma^2}B_4+\left(\frac{2\mu}{\sigma^2}\right)^2\right)\\
&\cdot\exp\left(\left(\sigma\sqrt{t-t_0}\right)^2\cdot\frac{2\mu}{\sigma^2}\left(-\frac{1}{\sigma\sqrt{t-t_0}}B_4+\frac{1}{2}\frac{2\mu}{\sigma^2}\right)\right)\\
&\cdot\operatorname{erf}\left(\frac{1}{\sqrt{2}}\left(-\frac{2\mu}{\sigma^2}\sigma\sqrt{t-t_0}+\frac{1}{\sigma\sqrt{t-t_0}}X+B_4\right)\right)\Bigg).
\end{aligned}
$$

Finally, we have

$$
\int X\exp(T_{4,1})\Phi_{\mathcal{N}}(T_{4,2})\,dX=\exp(A_4)\left(u_4v_4-\frac{1}{\sqrt{2\pi}}\frac{1}{\sigma\sqrt{t-t_0}}\frac{\sigma^2}{2\mu}\int I_4\,dX\right).
$$

It is straightforward to combine these results to attain the final conditional expectation. Deriving the final expression shows the validity of the integrals. ■

We now need to verify that this is continuously differentiable with respect to t. By inspection, we see that most terms of it are individually continuous; all terms contain a factor or combination of the functions exp, erf, $\Phi_{\mathcal{N}}$. All are continuous and continuously differentiable on $\mathbb{R}$. Furthermore, the derivative with respect to t will have a very similar structure.

It then remains to determine the behaviour of the function as $t \to t_0$, where issues with division by 0 appear. Analysing it is out of scope for this thesis, though we may turn to numerical approximations as in Figure 2.

Figure 2 provides empirical evidence that both functions are continuous for all t. Thus, the conditional expectation is continuously differentiable in t. Without proof, we remark that this is expected because erf and $\Phi_{\mathcal{N}}$ are bounded, and that exp converges faster than linear terms. The terms with a numerator in $(t-t_0)$ also play a role in ensuring the function remains bounded as $t \to t_0$.

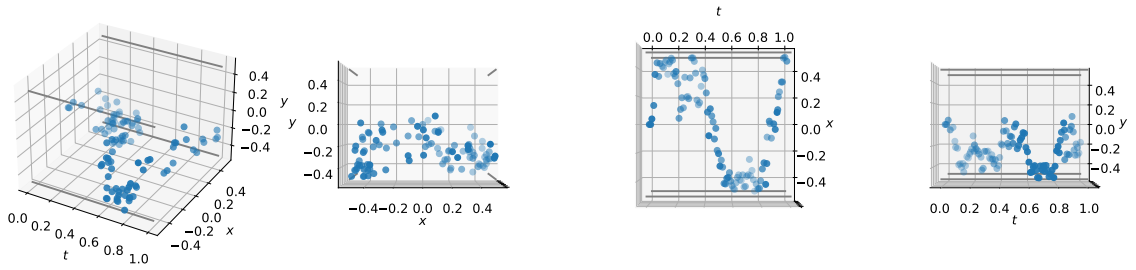

Figure 3: One path of a rectangle generated via two RBMs, viewed from different angles. The rectangle has width and height 1. The RBMs driving it both have $\mu = 0$ and $\sigma = 1$.

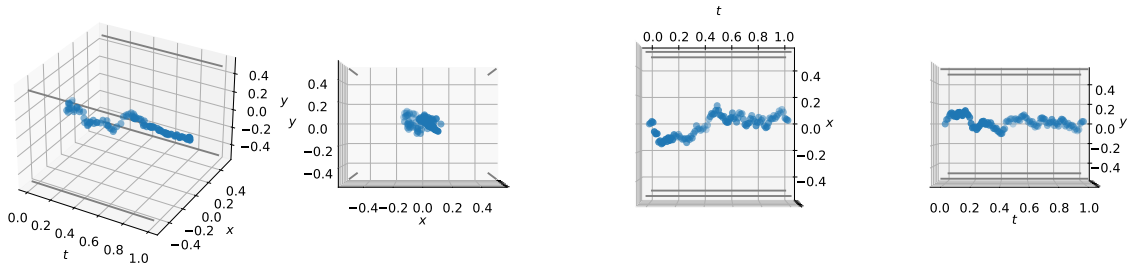

Figure 4: One path of a rectangle generated via Brownian motion weight process on four vertices, viewed from different angles. The rectangle has width and height 1.

7.2 Hyperrectangles Using Reflected Brownian Motion

A limitation about reflected Brownian motion is that the literature is often restricted to the one-dimensional case. We therefore use a simple method to create a multi-dimensional RBM.

To do this we can use multiple independent RBMs to form a multi-dimensional boundary. For $c_1 < d_1$ and $c_2 < d_2$, the process $X = (X_1, X_2)^T$ where X_1 is an RBM constrained to (c_1, d_1) and X_2 an independent RBM in (c_2, d_2) forms a process that is constrained to a two-dimensional rectangle. This process can be extended to any dimension. See Figure 3 for an example.

7.3 Weight-space Processes on Vertices

We generate a weight process by normalising a Brownian motion using softmax. In particular, we denote the unnormalised weight process as $\tilde{W}$ which is simply a Brownian motion without drift or diffusion. To create a weight process, we let $W = \sigma(\tilde{W})$ so that W is normalised. Thus, like in Section 5, we have $X_t = \tilde{g}(W_t) = \tilde{g}(\sigma(\tilde{W}_t))$. An example of a weight-space process on a rectangle is given in Figure 4; cf. Figure 3 for a process generated on the same rectangle, but by RBMs.

The trouble with inverting a convex combination, which we have already encountered, becomes apparent once again as we try to find $\hat{X}_t = \mathbb{E}[X_t \mid \mathcal{A}_t]$. Recall that with this example, $\mathcal{A}_t$ is formed by previous observations of X. However, there is no formula to find

the weights defining a given point in Q. As such, we cannot access what we know about W or $\tilde{W}$ to find $\hat{X}$.

We circumvent this issue by computing a conditional expectation that should at least emulate $\hat{X}$, by additionally conditioning on $\tilde{W}_s$ which is not $\mathcal{A}_t$-measurable for any $s < t$ by the above logic. That is to say, this technique is only an approximation of the target conditional expectation and does not technically represent what the model can learn or see. Nevertheless, the empirical results seem plausible.

The approximation works by finding $\mathbb{E}[W_t \mid \tilde{W}_s]$ for $0 \leq s < t$. We can then approximate $\hat{X}_t$ by using these weights in a convex combination, i.e., $\mathbb{E}[X_t \mid \mathcal{A}_s] \approx \tilde{g}(\mathbb{E}[W_t \mid \tilde{W}_s])$. The conditional expectation is given for each coordinate $1 \leq i \leq V$ as

$$\begin{aligned}
\mathbb{E}\Big[W_t \,\Big|\, \tilde{W}_s\Big]_i &= \mathbb{E}\Big[\sigma(\tilde{W}_t)_i \,\Big|\, \tilde{W}_s\Big] \\
&= \mathbb{E}\left[\frac{\exp\Big((\tilde{W}_t)_i - (\tilde{W}_s)_i + (\tilde{W}_s)_i\Big)}{\sum_{j=1}^{V} \exp\Big((\tilde{W}_t)_j - (\tilde{W}_s)_j + (\tilde{W}_s)_j\Big)} \,\middle|\, \tilde{W}_s\right] \\
&= \mathbb{E}\left[\frac{\exp((N_{t-s})_i)\exp((\tilde{W}_s)_i)}{\sum_{j=1}^{V} \exp((N_{t-s})_j)\exp((\tilde{W}_s)_j)} \,\middle|\, \tilde{W}_s\right] \\
&= \mathbb{E}\left[\frac{\exp((N_{t-s})_i)\exp(w_i)}{\sum_{j=1}^{V} \exp((N_{t-s})_j)\exp(w_j)}\right]\Bigg|_{w=\tilde{W}_s}.
\end{aligned}$$

Here N_{t-s} is a random variable such that $N_{t-s} \sim \mathcal{N}(0, (t-s)I_V)$. Despite the lack of analytical solution, we can use a Monte Carlo method to estimate the expectation.

To satisfy the assumptions, we need that F is continuously differentiable in t. By noting that $N_{t-s} = N_1\sqrt{t-s}$ it is clear that $\mathbb{E}[W_t \mid \tilde{W}_s]$ is continuously differentiable. Thus $\tilde{g}(\mathbb{E}[W_t \mid \tilde{W}_s]$ is continuously differentiable, indicating that F is likely also continuously differentiable.

7.4 Brownian Motion Inside a Ball

In a similar vein, we can use stochastic processes for the radius and angle(s) defining the movement of a particle inside a ball. For this process we first demonstrate the results for the two-dimensional case. We let $Q := B_R^2(0)$ be a closed 2-ball centered at the origin and with radius R with respect to the 2-norm.

The process $X \in Q$ is defined through two independent 1-dimensional Brownian motions B_r and B_θ; we interpret these as the two processes defining the radius and angle of the process. Finally, we define the process X as

$$X_t := R\,|\tanh(B_{r,t})| \begin{bmatrix}\cos(\alpha_\theta B_{\theta,t})\\ \sin(\alpha_\theta B_{\theta,t})\end{bmatrix}, \text{ or more compactly } X_t := Rr_t \begin{bmatrix}\cos(\alpha_\theta \tilde{\theta}_t)\\ \sin(\alpha_\theta \tilde{\theta}_t)\end{bmatrix}$$

for $r_t := |\tanh(B_{r,t})|$ and $\tilde{\theta}_t := B_{\theta,t}$.

See Figure 5 for what this process looks like.

We apply $|\tanh(\cdot)|$ to the Brownian motion since it is symmetric around 0 and is bounded between 0 and 1, which we then scale by a factor R. Similarly, we provide a scaling factor

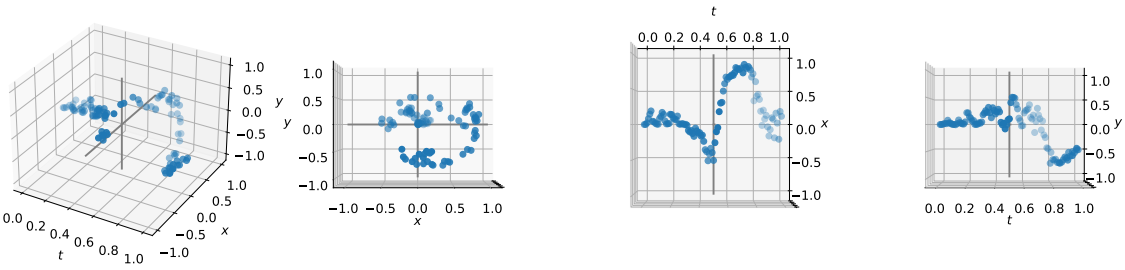

Figure 5: One path of a 2-ball generated via two Brownian motions, viewed from different angles. The process has $R = 1$ and $\alpha_\theta = \pi$.

α_θ for the angle since the Brownian motion itself oscillates around 0. A value of π is appropriate. Both scaling factors affect the motion and can be adjusted until the resulting process behaves as desired.

Before moving on to the conditional expectation, let us first discuss what is and is not $\mathcal{A}_t$-measurable. Both r_t and the angle $\theta_t := \alpha_\theta \tilde{\theta}_t \pmod{2\pi}$ are measurable via X_t. They can be found as follows:

$$\begin{aligned}
\|X_t\|_2 &= (X_{t,x}^2 + X_{t,y}^2)^{1/2} & X_{t,x} &= Rr_t \cos(\theta_t) \\
&= ((Rr_t)^2 \cos^2(\theta_t) + (Rr_t)\sin^2(\theta_t))^{1/2} & &\implies \theta_t = \arccos(X_{t,x}/Rr_t) \\
&= Rr_t(\cos^2(\theta_t) + \sin^2(\theta_t))^{1/2} \\
&\implies r_t = \|X_t\|_2/R
\end{aligned}$$

Furthermore, we can derive $|B_{r,t}|$ using the inverse hyperbolic tangent since $|\tanh(B_{r,t})|$ equals $\tanh(|B_{r,t}|)$, thus $\operatorname{artanh}(r_t) = |B_{r,t}|$. The value of $B_{r,t}$ itself is not $\mathcal{A}_t$-measurable. Given the angle θ_t, we can find $\tilde{\theta}_t \pmod{2\pi}$, but $\tilde{\theta}_t = B_{\theta,t}$ is not $\mathcal{A}_t$-measurable because $B_{\theta,t}$ does not necessarily lie in $[0, 2\pi]$. On the other hand, both $\cos(\tilde{\theta}_t)$ and $\sin(\tilde{\theta}_t)$ are $\mathcal{A}_t$-measurable.

Let us now compute the conditional expectation. Since B_r and B_θ are independent, we can define the conditional expectations $\mathbb{E}[r_t \mid \mathcal{A}_s]$ and $\mathbb{E}[\cos(\theta_t) \mid \mathcal{A}_s]$, $\mathbb{E}[\sin(\theta_t) \mid \mathcal{A}_s]$ separately. (We omit scaling factors for clarity.)

We begin with $\mathbb{E}[r_t \mid \mathcal{A}_s]$ for $s < t$. As we will see, only $B_{r,s} = \operatorname{artanh}(r_s) \in \mathcal{A}_s$ is necessary. We have

$$\begin{aligned}
\mathbb{E}[r_t \,|\, \mathcal{A}_s] &= \mathbb{E}[|\tanh(B_{r,t})| \,|\, \mathcal{A}_s] \\
&= \mathbb{E}[|\tanh(B_{r,t} - B_{r,s} + B_{r,s})| \,|\, \mathcal{A}_s] \\
&= \mathbb{E}[|\tanh(N_{t-s} + B_{r,s})| \,|\, \mathcal{A}_s] \\
&= \mathbb{E}[|\tanh(N_{t-s} + |B_{r,s}|)| \,|\, \mathcal{A}_s]
\end{aligned}$$

where $N_{t-s} \sim \mathcal{N}(0, t-s)$ is a normally distributed random variable with variance $t - s$. In the final step we draw attention to the fact that $|\tanh(N_{t-s} + B_{r,s})|$ and $|\tanh(N_{t-s} - B_{r,s})|$ are equal in law because tanh is symmetric around 0 and so is the density of N_{t-s}. We can therefore use $|B_{r,s}|$, which is $\mathcal{A}_s$-measurable.

For $\mathbb{E}[\cos(\theta_t) \mid \mathcal{A}_s]$ we can also reduce the information σ-algebra; we only need $\cos(B_{\theta,s}) = \cos(\tilde{\theta}_s)$ and $\sin(B_{\theta,s}) = \sin(\tilde{\theta}_s)$, both of which are $\mathcal{A}_s$-measurable. Proceeding, we have

$$\begin{aligned}
&\mathbb{E}[\cos(\theta_t) \,|\, \mathcal{A}_s] \\
&= \mathbb{E}\Big[\cos(\tilde{\theta}_t \bmod 2\pi) \,\Big|\, \mathcal{A}_s)\Big] \\
&= \mathbb{E}\Big[\cos(\tilde{\theta}_t) \,\Big|\, \mathcal{A}_s\Big] \\
&= \mathbb{E}[\cos(B_{\theta,t}) \,|\, \mathcal{A}_s] \\
&= \mathbb{E}[\cos(B_{\theta,t} - B_{\theta,s} + B_{\theta,s}) \,|\, \mathcal{A}_s] \\
&= \mathbb{E}[\cos(N_{t-s} + B_{\theta,s}) \,|\, \mathcal{A}_s] \\
&= \mathbb{E}[\cos(N_{t-s})\cos(B_{\theta,s}) - \sin(N_{t-s})\sin(B_{\theta,s}) \,|\, \mathcal{A}_s] \\
&= \mathbb{E}[\cos(N_{t-s})]\cos(B_{\theta,s}) - \mathbb{E}[\sin(N_{t-s})]\sin(B_{\theta,s}).
\end{aligned}$$

Again, we take N_{t-s} to be a random variable following the normal distribution $\mathcal{N}(0, t-s)$.

Contrary to $\mathbb{E}[r_t \mid \mathcal{A}_s]$, we can find an analytical expression for this term. We simply need to find $\mathbb{E}[\cos(N_{t-s})]$ and $\mathbb{E}[\sin(N_{t-s})]$. We use the exponential form of cosine $\cos(x) = (\exp(ix) + \exp(-ix))/2$. We also need the characteristic function (with parameter u) for a random variable N following the normal distribution with mean μ and standard deviation σ, which is $\mathbb{E}[\exp(iuN)] = \exp(iu\mu - \frac{1}{2}\sigma^2 u^2)$. The first expected value is then

$$\begin{aligned}
\mathbb{E}[\cos(N_{t-s})] &= \frac{1}{2}\mathbb{E}\big[e^{iN_{t-s}} + e^{iN_{t-s}}\big] \\
&= \frac{1}{2}(e^{-\frac{1}{2}(t-s)} + e^{-\frac{1}{2}(t-s)}) \\
&= (1/\sqrt{e})^{t-s}.
\end{aligned}$$

For the sine term, we note that sin is an antisymmetric function and the probability density function φ of $\mathcal{N}(0, t-s)$ is symmetric. Thus, using the law of the unconscious statistician, we have

$$\mathbb{E}[\sin(N_{t-s})] = \int_{\mathbb{R}} \sin(x)\varphi(x)\, dx = 0.$$

Combining these results, we can write

$$\mathbb{E}[\cos(\theta_t) \,|\, \mathcal{A}_s] = (1/\sqrt{e})^{t-s}\cos(B_{\theta,s}).$$

Unsurprisingly, finding the conditional expectation $\mathbb{E}[\sin(\theta_t) \mid \mathcal{A}_s]$ is similar. We present the derivation here but omit some steps that are already presented in the previous derivation.

$$\begin{aligned}
\mathbb{E}[\sin(\theta_t) \,|\, \mathcal{A}_s] &= \mathbb{E}[\sin(B_{\theta,t}) \,|\, \mathcal{A}_s] \\
&= \mathbb{E}[\sin(N_{t-s} + B_{\theta,s}) \,|\, \mathcal{A}_s] \\
&= \mathbb{E}[\sin(N_{t-s})\cos(B_{\theta,s}) + \cos(N_{t-s})\sin(B_{\theta,s}) \,|\, \mathcal{A}_s] \\
&= \mathbb{E}[\sin(N_{t-s})]\cos(B_{\theta,s}) + \mathbb{E}[\cos(N_{t-s}))]\sin(B_{\theta,s})
\end{aligned}$$

$$= (1/\sqrt{e})^{t-s} \sin(B_{\theta,s}).$$

Combining everything, we can write the final conditional expectation for $s < t$, including the scaling factors R and α_θ, as

$$\mathbb{E}[X_t \,|\, \mathcal{A}_s] = R\mathbb{E}[|\tanh(N_{t-s} + b)|] \begin{bmatrix} (1/\sqrt{e})^{\alpha_\theta(t-s)} x \\ (1/\sqrt{e})^{\alpha_\theta(t-s)} y \end{bmatrix}$$

evaluated with $b = |B_{r,s}| = \operatorname{artanh}(r_s)$, $x = \cos(\alpha_\theta B_{\theta,s}) = \cos(\theta_s)$, $y = \sin(\alpha_\theta B_{\theta,s}) = \sin(\theta_s)$. As before, N_{t-s} is a random variable following the $\mathcal{N}(0, t-s)$ distribution.

Once again, we must use a Monte Carlo method to approximate this conditional expectation.

That the expression is continuously differentiable in t is clear by inspection, so the assumptions on X are satisfied.

8. Experiments

To verify the theory built so far, we now perform experiments to see if the new techniques can learn processes taking values in convex spaces better, compared to the plain PD-NJ-ODE benchmark. In each of the following subsections we will introduce the relevant dataset and models tested; present the outcomes of the experiments; and finally discuss the results. Parameters for the datasets and models are provided for transparency, as is the implementation which can be found at `https://github.com/willandersson1/PD-NJODE-CONVEX`.

In general, we analyse the convergence behaviour of the various strategies and compare them to the baseline PD-NJ-ODE model. In addition to the standard optimal projection and vertex approaches, we also consider models where the optimal projection is only applied at inference time (i.e., not during training). For all of these we also analyse the number of predictions that lie in Q, especially in the final set of experiments where we focus on the additional loss term. The datasets are based on the processes proven to satisfy the assumptions in Section 7.

There are certain choices we have made that hold for each dataset and each model. Instead of restating them each time in the following, we explain them once here. They are identical to the ones used when testing the original PD-NJ-ODE model (Krach et al., 2022), primarily because we would like to compare the models using the PD-NJ-ODE as a benchmark, but also because we found these hyperparameters to work reasonably well with the new strategies too.

The default dataset values are in Table 1. A trend the reader will notice in the specific datasets is that the sets are centered around the origin. This is because the weights of the PD-NJ-ODE model are initialised around 0, so the first prediction lies around there. Note that although we try to keep the same setting as in Krach et al. (2022), computational limitations prevent us from using a large number of paths.

Likewise, the default model parameters can be found in Table 1. In addition to the parameters displayed in Table 1, all neural networks in the PD-NJ-ODE architecture (see Section 3.3) consist of three hidden layers of 50 neurons each, using the tanh activation function. This is a relatively large network. An RNN is used for the jumps. Neither the

Parameter	Value
Steps	100
Maturity (T)	1
Observation probability	10%
Train:test split	80:20

Parameter	Value
Epochs	200
Batch size	200
Learning rate	0.001
Dropout rate	0.1
Bias	✓

Table 1: Default dataset (left) and model (right) parameters.

encoder nor the readout networks have residual connections. The latent variable process has size 10, which is small.

Some datasets use Monte Carlo approximations for computing the conditional expectation. To be specific, weight-space processes on vertices (Section 7.3), and Brownian motions inside a ball (Section 7.4) require Monte Carlo approximations for the conditional expectation $\hat{X}$. The number of samples we use for the approximation depends on the dataset and chosen by inspecting plots of the path until they appear smooth.

A final comment: in some of the upcoming convergence plots, the models do not converge to the optimal loss—even though we have proved that they should. The optimal loss uses the conditional expectation we provide it, which is sometimes not truly the conditional expectation of the process X we predict. In most cases this is due to the aforementioned numerical approximations. In one case however, the conditional expectation we use for weight-space processes on vertices is an approximation at the conceptual level from the outset, as it uses more information than available to our model (see Section 7.3). Hence, the corresponding optimal loss is comparable to an oracle, which should be understood as an unreachable lower bound.

We now proceed to the experiments themselves.

8.1 Reflected Brownian Motion

Despite having the conditional probability density (see Section 7.1.1), it is not trivial to generate sample paths of an RBM. One would need to use a method to draw samples from a probability distribution, such as Markov chain Monte Carlo methods. This is inefficient and excessive for our case. Instead, we use the method outlined in Algorithm 1.

Algorithm 1 Sample a path of an RBM with drift μ, diffusion σ, and n time steps.

$(X_1, X_2, \dots, X_n) \leftarrow$ Brownian motion with drift μ and diffusion σ
for $i = 1, 2, \dots, n$ **do**
  if $X_i <$ `lower_bound` **then**
    $(X_i, X_{i+1}, \dots, X_n) \leftarrow (X_i, X_{i+1}, \dots, X_n) + 2(\texttt{lower\_bound} - X_i)$
  else if $X_i >$ `upper_bound` **then**
    $(X_i, X_{i+1}, \dots, X_n) \leftarrow (X_i, X_{i+1}, \dots, X_n) - 2(X_i - \texttt{upper\_bound})$
  end if
end for
return $(X_1, X_2, \dots, X_n)$

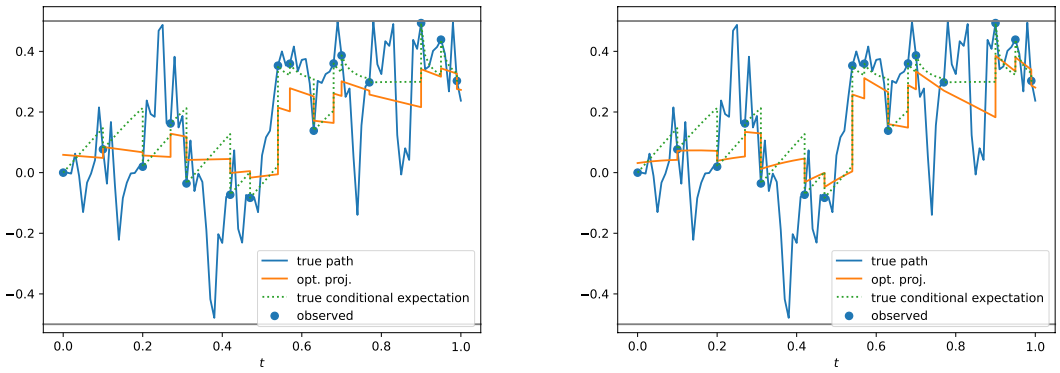

Figure 6: An optimal projection model predicting an RBM process at epoch 20 (left) and epoch 200 (right).

There is however a technical issue with this algorithm: it assumes that the shift value will itself not be outside the bounds. This could happen with tight bounds or a large variance but is not an issue for our experiments. Indeed, it can also be mitigated by repeating the logic inside the loop until the result does lie in Q.

For the probability density function itself, we let the truncation level of the sums be a hyperparameter which we set to 3 for all of our experiments. The function is well approximated even with a small number of terms, as can be seen in Figure 2. We also make some changes compared to the equation as presented in Equation 5. In particular, we first sum the exponents before evaluating the exponential, instead of multiplying exponentials to avoid overflows. For the same reason, we compute the Φ terms in the last two sums first and set the entire sum to 0 if the associated Φ term is equal to 0. Finally, we check and validate for floating point issues in multiple places.

Our implementation of the conditional expectation uses numerical integration with the conditional probability density to compute $\int_c^d Xp(X; X_0, t_0)\, dX$. We do not use the analytical conditional expectation found in Proposition 22 due to the complexity in implementing it, and because numerical quadrature is quick and precise enough for our purposes.

The dataset information is in Table 2. We test two RBMs: a standard one, and one that is more erratic. The convergence results of the benchmark PD-NJ-ODE model and the optimal projection strategy on the two datasets is shown in Figure 7. Projection is done by clamping the point between the lower and upper bounds. Note we do not test the vertex approach, since the dataset is trivial for it.

The results are as we expect due to the simplicity of the dataset: all strategies can easily learn the correct conditional expectation. Both optimal projection models converge slightly faster and more steadily than the benchmark PD-NJ-ODE model. We also see that the model with projection at inference finds better predictions at the end of training than the others. Additionally, none of the strategies made predictions outside of the interval.

Name	Paths	Interval	μ	σ
RBMStandard	200	$[-0.5, 0.5]$	0	1
RBMMoreBounces	200	$[-0.5, 0.5]$	1.5	1.5

Table 2: RBM dataset configurations.

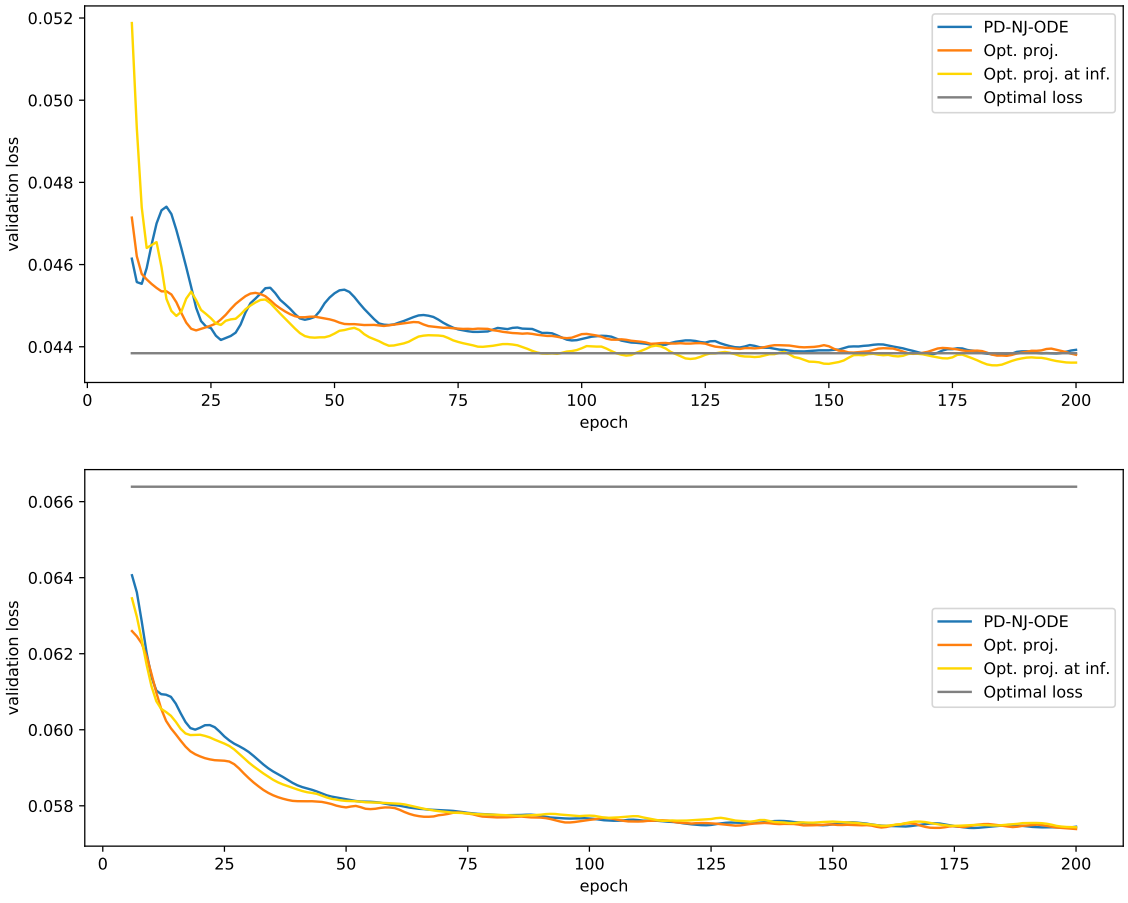

Figure 7: Convergence plots for RBMStandard (top) and RBMMoreBounces (bottom) datasets, for all strategies.

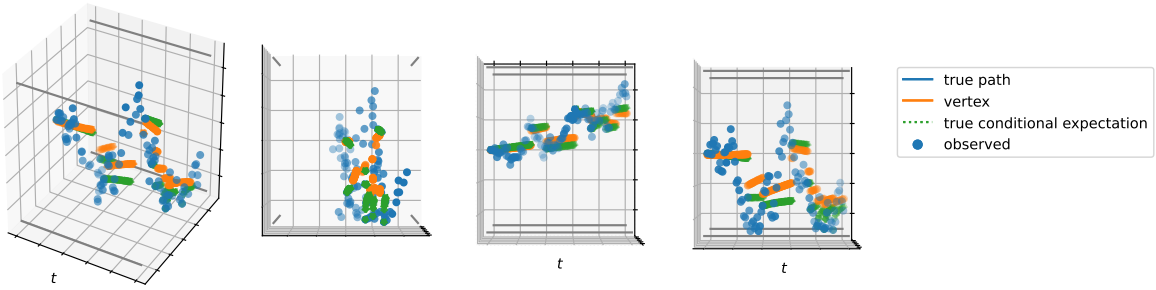

Figure 8: A vertex approach model predicting a rectangle process produced by two independent RBMs, from multiple views.

Name	Paths	Dimensions	μ_x	σ_x	μ_y	σ_y
RectangleStandard	100	1×1	0	1	0	1
RectangleWider	100	1×1	0.6	2	-0.1	1
RectangleBMWeights	100	1×1	—	—	—	—

Table 3: Rectangle dataset configurations.

8.2 Rectangles

As explained earlier in Section 7.2, we can simulate a 2D RBM by taking two independent RBMs. To implement this, we simply instantiate two RBMs and compute the paths, conditional expectation, etc. coordinate-wise. This is how the first two datasets (RectangleStandard and RectangleWider) are generated. The latter is a path that should be more difficult to learn due to the drift and diffusion.

In contrast, although it has the same dimensions, the RectangleBMWeights dataset is created by applying a weight process to the rectangle's vertices. This follows the process outlined in Section 7.3.

Configuration details for the three datasets can be found in Table 3. For the optimal projection method, we project each coordinate separately, i.e., clamp each dimension to the respective lower and upper bound given by the sides of the rectangle. Figure 9 displays the convergence characteristics of the three models on these datasets.

We hypothesise that the vertex approach should perform best for RectangleBMWeights dataset, since generating a probability distribution over vertices is an inherent part of the algorithm as softmax is part of the architecture. In this sense the model's architecture is similar to the way the dataset is generated.

The results support this hypothesis somewhat, as we see the vertex approach converging better on the RectangleBMWeights compared to the other datasets, especially RectangleWider where converges slowly and to a worse minima. All other strategies (PD-NJ-ODE, optimal projection, and optimal projection only at inference) perform equally well on all datasets, both in convergence speed and final convergence value. Generally, projecting only at inference instead of during training appears to allow slightly faster convergence. The PD-NJ-ODE and optimal projection techniques make almost no predictions outside of the rectangle for the simpler dataset. In contrast there are slightly more errant predictions in

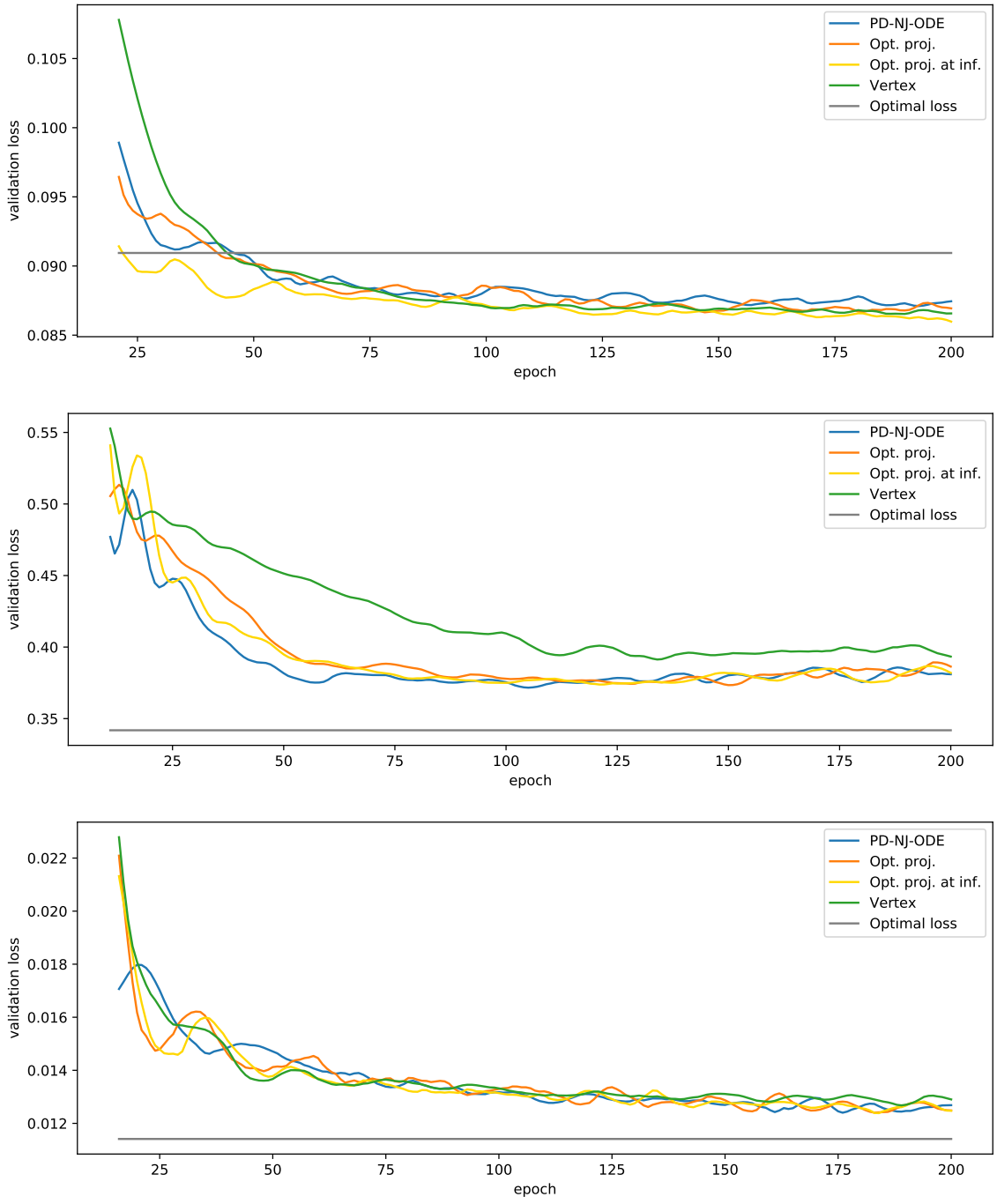

Figure 9: Convergence plots for the RectangleStandard (top), RectangleWider (middle), and RectangleBMWeights (bottom) datasets, for all strategies.

the more complicated dataset, likely due to the drift. In this case the projection is helpful in keeping the process inside the set.

8.3 Simplexes

To provide more examples of weight-space processes on vertices (as in Section 7.3), we create datasets for the 1-simplex Δ^1 and the 2-simplex Δ^2. These (probability) simplexes are interesting primarily because the results here can be applied to learning processes on finite state spaces, where the output is a probability distribution over the states. Secondly, we hypothesise that the vertex approach will work well here due to its architecture, like we predicted for the RectangleBMWeights dataset in Section 8.2.

Another advantage to consider is that there exist efficient algorithms for computing the projection of a point x onto the simplex. We use the algorithm by Chen and Ye (2011), which is highly efficient; their empirical results demonstrate a roughly linear relationship between dimension and computation time, although the algorithm as presented has quadratic time complexity in the worst case.

In Algorithm 2 we document it roughly as implemented in our program, though ours differs in structure to its original presentation in Chen and Ye (2011). It is most similar to their provided code.[7]

Algorithm 2 Projection of a point $y \in \mathbb{R}^n$ onto the simplex Δ^n.

$s \leftarrow$ `sort_descending(`y`)`
`sum` $\leftarrow 0$
`max` $\leftarrow 0$
`skipped` $\leftarrow$ `false`
for $i = 1, 2, \ldots, n-1$ **do**
  `sum` $\leftarrow$ `sum` $+ s_i$
  `max` $\leftarrow \frac{\texttt{sum}-1}{i}$
  if `max` $\geq s_{i+1}$ **then**
    `skipped` $\leftarrow$ `true`
  end if
end for
if `skipped` $==$ `false` **then**
  `max` $\leftarrow \frac{\texttt{sum}+s_n-1}{n}$
end if
return $(y - \texttt{max})_+$

The datasets themselves are created by first sampling a multidimensional Brownian motion, applying the Softmax function to normalise them into weights, then using the weights in a convex combination to generate a point in $\mathbb{R}^{d_X}$ (which is either Δ^1 or Δ^2). The parameters for the datasets are in Table 4. Experiment results are displayed in Figure 11.

The plots in Figure 11 strongly support our hypothesis, as the vertex approach converges quickly to the optimal prediction. The standard optimal projection model does however converge slightly faster. Interestingly, optimal projection only at inference barely improves

7. Found at `https://mathworks.com/matlabcentral/fileexchange/30332`.

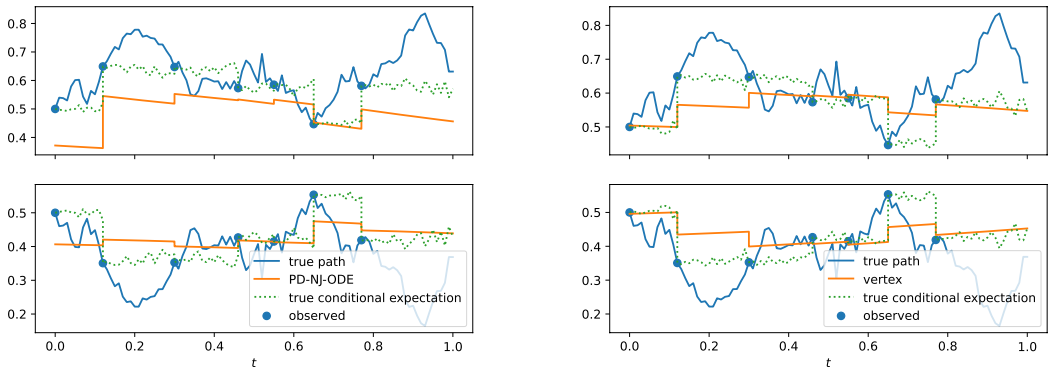

Figure 10: PD-NJ-ODE (left) and the vertex approach (right) predicting a weight process on a 1-simplex, at epoch 20. Note the superior performance of the vertex approach.

Name	Paths	Vertices
1Simplex	500	$(1,0)$, $(0,1)$
2Simplex	500	$(1,0,0)$, $(0,1,0)$, $(0,0,1)$

Table 4: Simplex dataset configurations.

on the PD-NJ-ODE model's performance (which is significantly worse than the others). Thus, projecting during training was beneficial. We see that none of the PD-NJ-ODE or optimal projection (before projection) models were able to make predictions inside the set, likely because the simplex is a lower dimensional subset of $\mathbb{R}^{d_X}$. As a result, so projecting onto the set would be particularly helpful for these datasets.

8.4 Brownian Motion in a 2-ball

As a less trivial example of a Brownian motion inside a convex set, we perform experiments on Brownian motion processes inside a 2-ball. A higher-dimensional ball could also easily be used.

The datasets are sampled as in Section 7.4, with the specific details in Table 5. We test the PD-NJ-ODE and two optimal projection models, the convergence results of which are in Figure 13. Optimal projection of a point onto the ball is performed by the function $x \mapsto Rx/\|x\|_2$ when $\|x\|_2 > R$ (i.e., the point is outside of the ball). We do not test the vertex approach, since the only way to do so would be to approximate the ball by a large number of vertices. Although interesting, it would not be a meaningful comparison.

Figure 13 allows us to compare the two strategies. The convergence behaviour is almost identical for the PD-NJ-ODE benchmark and the model projecting only at inference for both datasets. This indicates that the optimal projection was not necessary. Indeed, all strategies always place their predictions inside of the ball. Meanwhile, the standard optimal projection model converges slightly faster and to the same optimal prediction as the others in the Ball2DStandard dataset, but slower and to a worse prediction in the other dataset.

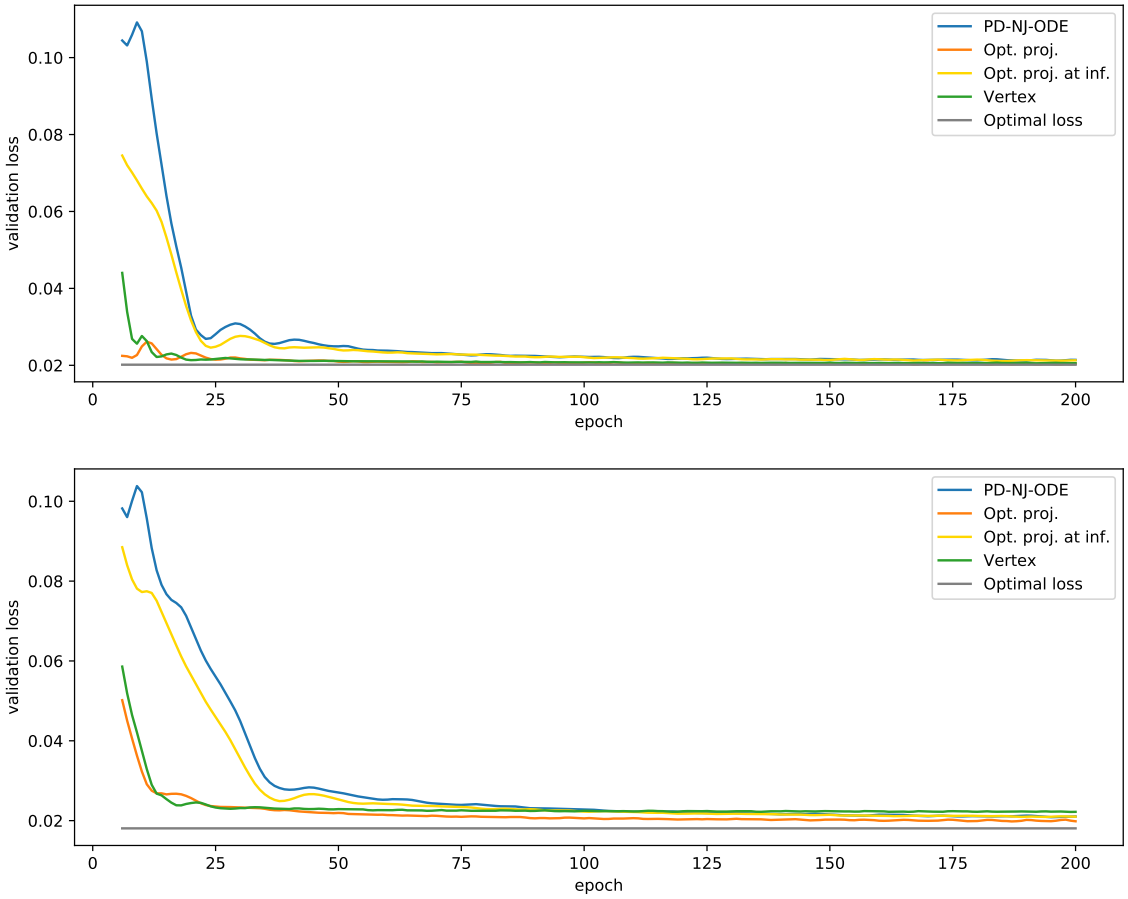

Figure 11: Convergence plots for 1Simplex (top) and 2Simplex (bottom) datasets, for all strategies.

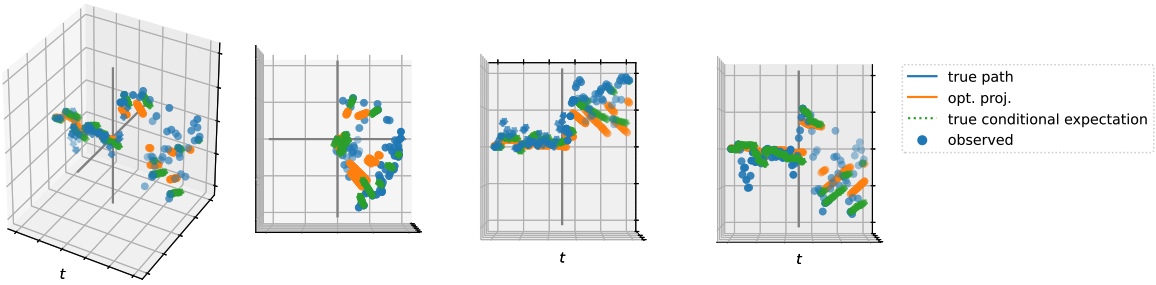

Figure 12: An optimal projection model predicting a 2-ball process, from multiple views.

Name	Paths	R	α_θ
Ball2DStandard	500	1	π
Ball2DLarge	500	10	π

Table 5: Ball dataset configurations.

We attribute this to learning difficulties in the case where a prediction lies outside of the ball, which can sometimes keep the optimiser in a poor local minima.

8.5 Penalising Functions

We now discuss the last technique: adding a new term to the loss function to penalise predictions outside of Q. Since it was given in the form of an integral (Equation 4), we approximate it using the discrete grid given by the existing time steps. We also look at the number of predictions made inside Q during the training process. Note that we discuss only the PD-NJ-ODE model and the optimal projection technique, as the vertex approach is not applicable. For the optimal projection strategies, we naturally discuss values before the projection is applied.

All models on all datasets except the simplex ones were able to place all of their predictions inside Q. Due to the complicated requirements for a point to be in a simplex, it is unsurprising that all strategies managed to place only a small number of their predictions inside it.

In any case, this demonstrates the need for a new dataset. We use a 2-ball with radius $R = 0.0005$ so that X can take values nearer to the boundary, and so the predictions are more likely to be outside of it. All other parameters, including those of the models, remain as before. Figure 14 demonstrates that this is a harder dataset to learn; the the other Ball datasets in Figure 13 provide a good comparison.

Here we see that the standard optimal projection technique is stuck in a bad local minima. This is due to the issue of vanishing gradients: the initial predictions are outside of the set and then projected onto the same point on the boundary, so the optimiser is stuck in a poor local minima. The additional penalty term does not help this, as it also just encourages the predicted points to be near the boundary. For this reason, we do not explore the effect of the penalty term on the standard optimal projection model any further. We also do not experiment with the value of λ when the projection is only applied during

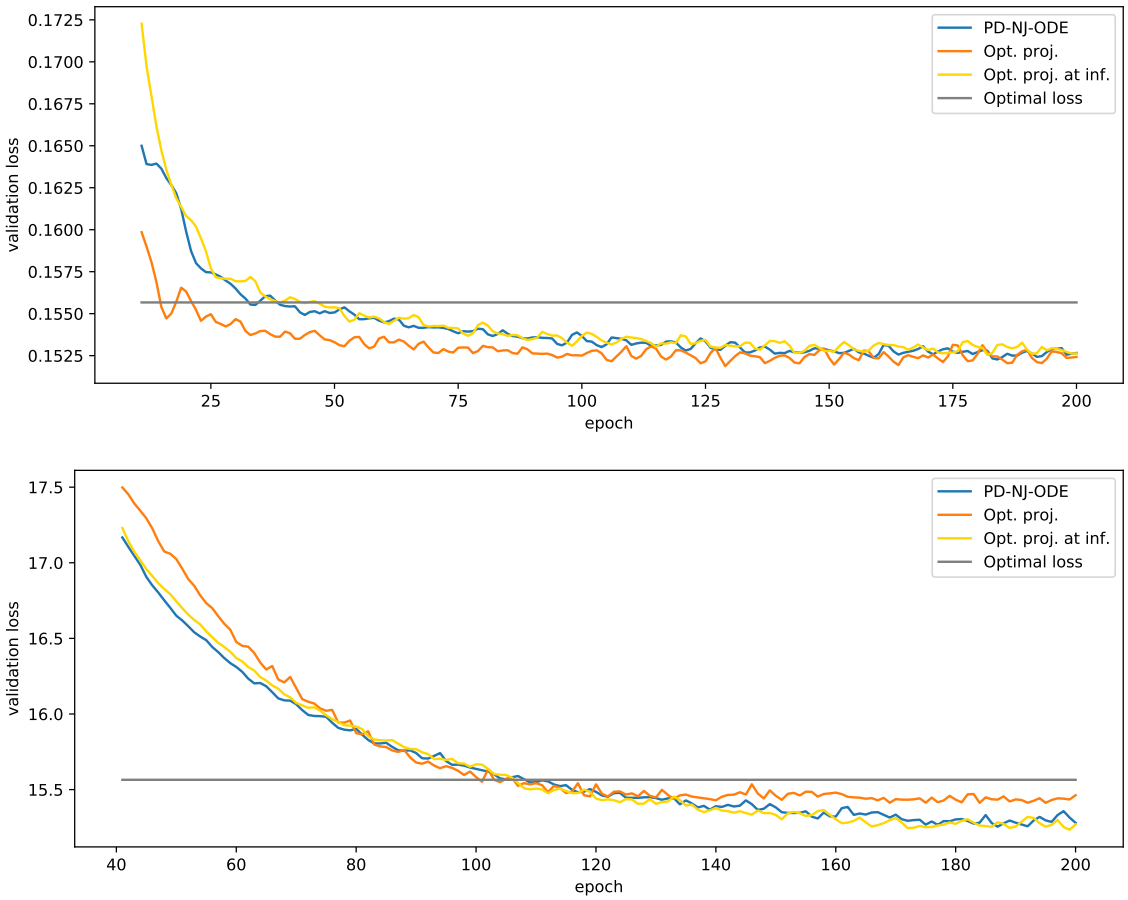

Figure 13: Convergence plots for Ball2DStandard (top) and Ball2DLarge (bottom) datasets, for all strategies.

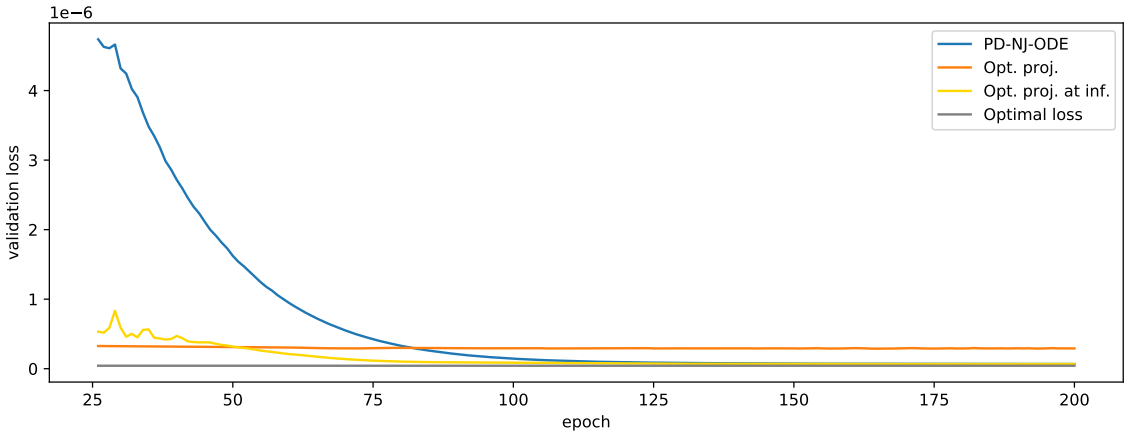

Figure 14: Convergence plot for the PD-NJ-ODE, standard optimal projection, and optimal projection at inference models on a 2-ball with $R = 0.0005$.

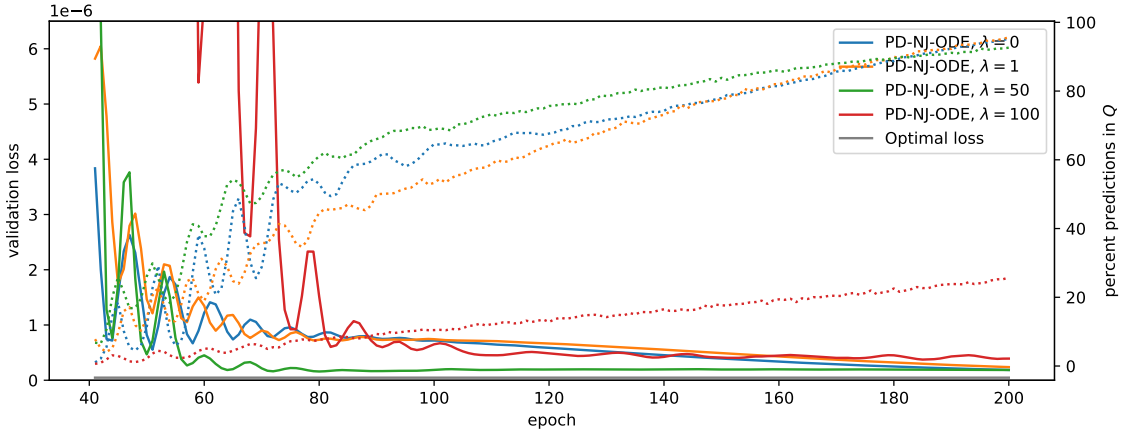

Figure 15: Convergence plot for the PD-NJ-ODE model on a 2-ball with $R = 0.0005$, for various values of λ.

inference, since we expect the effect to be similar, but more pronounced, compared to that of the PD-NJ-ODE model.

We analyse the PD-NJ-ODE model with $\lambda \in \{0, 1, 50, 100\}$. For the distance metric l we compare the predicted point to its projection onto the ball (see Section 8.4) and use the distance (measured by the squared 2-norm) between the two.

Figure 15 displays the convergence behaviour, with the training loss on the left y-axis; the percent of predictions inside the ball during training is on the right y-axis. Note this is not the same realisation of the dataset as in Figure 14. We see that λ can have a large impact on the model's performance, depending on its value.

For $\lambda = 1$ (orange) we see slightly worse loss convergence compared to the baseline $\lambda = 0$ (blue) model. The number of points predicted inside Q is also lower compared to the baseline, but in fact overtakes all other models near the end of training. In contrast, the model with $\lambda = 50$ (green) learns to predict values in Q much faster than all others. It also converges closer to the optimal loss. For very large λ we expect the model performance to suffer and perhaps degenerate, which is what we see with $\lambda = 100$ (red). The model converges slowly and to an optimum far from the optimal and struggles to predict values in Q.

From these results we see that the additional term can improve the performance of the PD-NJ-ODE model, but it is a delicate hyperparameter to choose. Experiments for $\lambda = 45$ and $\lambda = 55$ both exhibited worse behaviour than for $\lambda = 0$. Additionally, although the $\lambda = 50$ is the best of those tested, the relatively small value of $\lambda = 1$ already affects the model significantly. Finally, we see that for very large values of λ the model struggles to learn. This is due to the vanishing gradients problem, as the model is highly discouraged from predicting any values outside of Q; as such it is easily stuck in an undesirable local minima. This is the same issue that sometimes plagues the optimal projection strategy, which is logical as taking $\lambda \to \infty$ effectively leads to optimal projection (from the training point of view).

Overall, we see that the penalising term can be effective for the PD-NJ-ODE model, if it is chosen carefully, but the optimal projection technique is too susceptible to the vanishing gradients issue to generate meaningful results.

8.6 Discussion of Results

Our experiments show that all of the new techniques can be useful, some more than others depending on the dataset. The optimal projection technique performs the same or better than the PD-NJ-ODE benchmark on most datasets. Notable is the RectangleWider dataset where it outperforms both other strategies. Also interesting is the 2Simplex dataset, where it is significantly better than the PD-NJ-ODE model. However, it suffers from the vanishing gradients issue; depending on the dataset, it can fail to learn anything. This is clearly indicated on the very small 2-ball dataset.

On the other hand, the vertex approach displays good results on all datasets. It is particularly useful for the RectangleBMWeights, 1Simplex, and 2Simplex datasets where it is clearly better than the other two models. This agrees with our hypothesis, as these datasets are generated via weight processes on vertices and this is exactly what the vertex approach model aims to recover.

Finally, the additional loss term can benefit the PD-NJ-ODE model. If the value of λ is chosen carefully, it can increase how quickly the model can learn to predict accurately (in terms of number of predictions inside Q), as well as how well it can predict $\hat{X}$.

Overall, the optimal projection strategy when applied during inference is the most versatile and reliable. With a carefully chosen value of λ for the additional loss term, we expect it can perform even better. Of the other strategies, we find the standard optimal projection model to perform at least as well and sometimes significantly better than the vertex approach. It does however sometimes suffer from the vanishing gradients problem.

We must acknowledge that we report only the validation loss and not the evaluation metric as in Krach et al. (2022). The evaluation metric assesses the model's performance relative to the conditional expectation over the entire time interval, not just at observation times. In this sense it is more complete measure of performance than the validation loss. The reason we do not use it is that the conditional expectations we use are all approximations either at the numerical level (e.g. Monte Carlo or numerical integration) or at the conceptual level (e.g. weight-space processes on vertices, Section 7.3). On the other hand, the paths are generated exactly as specified by the theory. Due to this discrepancy it is not appropriate to use the evaluation metric.

9. Conclusion

In this thesis we have devised and tested methods for learning to predict stochastic processes that take values in a closed, convex set (known a priori). By integrating them into the PD-NJ-ODE framework, we have shown that they still converge to the optimal prediction, which is the conditional expectation. To investigate the relative effectiveness of these models, we used the standard PD-NJ-ODE model as a benchmark and tested all models on a variety of synthetic datasets.

The experiments demonstrate all approaches are effective. The two new model variants perform well, and significantly better than the PD-NJ-ODE in some cases. The results also

show that the additional loss term can help improve the performance of the PD-NJ-ODE model.

Other contributions to the literature include a derivation and brief inspection of the conditional expectation of a reflected Brownian motion, building on the work by Veestraeten (2004), as well as a Python implementation of the simplex approximation algorithm by Chen and Ye (2011). We also restructure and modernise the relevant parts of the (PD-)NJ-ODE codebase (developed in Herrera et al. (2021); Krach et al. (2022); Andersson et al. (2024)).

Future work includes exploring the idea of using the optimal projection or vertex approach in learning discrete state space processes where the NJ-ODE model could also be used as a generative model. It would also be interesting to create more complex datasets and polytopes, especially for the vertex approach, where one could explore applying it to non-polytopes. Finally, one could explore the case where Q is not known a priori, including the task of learning Q via the observations.

References

William Andersson, Jakob Heiss, Florian Krach, and Josef Teichmann. Extending path-dependent NJ-ODEs to noisy observations and a dependent observation framework. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=0T2OTVCCC1.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series, 2019. URL https://arxiv.org/abs/1905.12374.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019. URL https://arxiv.org/abs/1806.07366.

Yunmei Chen and Xiaojing Ye. Projection onto a simplex. 2011. doi: 10.48550/ARXIV. 1101.6081. URL https://arxiv.org/abs/1101.6081.

F. de Jong. A univariate analysis of ems exchange rates using a target zone model. *Journal of Applied Econometrics*, 9(1):31–45, 1994. ISSN 08837252, 10991255. URL http://www.jstor.org/stable/2285235.

Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning, 2018. URL https://arxiv.org/abs/1704.00805.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Calypso Herrera, Florian Krach, and Josef Teichmann. Neural jump ordinary differential equations: Consistent continuous-time prediction and filtering. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=JFKR3WqwyXR.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Florian Krach, Marc Nübel, and Josef Teichmann. Optimal estimation of generic dynamics by path-dependent neural jump odes. *CoRR*, abs/2206.14284, 2022. doi: 10.48550/arXiv.2206.14284. URL https://doi.org/10.48550/arXiv.2206.14284.

Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT*, 16(2): 146–160, jun 1976. ISSN 0006-3835. doi: 10.1007/BF01931367. URL https://doi.org/10.1007/BF01931367.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.04.014. URL https://www.sciencedirect.com/science/article/pii/S0169207019301128. M4 Competition.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.11.013. URL https://www.sciencedirect.com/science/article/pii/S0169207021001874. Special Issue: M5 competition.

Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. doi: 10.1214/aoms/1177729586. URL https://doi.org/10.1214/aoms/1177729586.

Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent odes for irregularly-sampled time series, 2019. URL https://arxiv.org/abs/1907.03907.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1986. URL https://api.semanticscholar.org/CorpusID:62245742.

Lars E.O. Svensson. The term structure of interest rate differentials in a target zone: Theory and swedish data. *Journal of Monetary Economics*, 28(1):87–116, 1991. ISSN 0304-3932. doi: https://doi.org/10.1016/0304-3932(91)90026-K. URL https://www.sciencedirect.com/science/article/pii/030439329190026K.

Gunnar Taraldsen. Optimal learning from the doob-dynkin lemma, 2018. URL https://arxiv.org/abs/1801.00974.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Dirk Veestraeten. The conditional probability density function for a reflected brownian motion. *Computational Economics*, 24(2):185–207, September 2004. ISSN 0927-7099. doi: 10.1023/b:csem.0000049491.13935.af. URL http://dx.doi.org/10.1023/B:CSEM.0000049491.13935.af.