universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Solving stochastic differential equations and Kolmogorov equations by means of deep learning and Multilevel Monte Carlo simulation“

verfasst von / submitted by

Julius Konstantin Berner, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2018 / Vienna 2018

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 821
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Master's programme in Mathematics
Betreut von / Supervisor:	Assoz. Prof. Dr. Philipp Grohs

Abstract

This thesis links mathematical learning theory with stochastic calculus in order to employ deep neural networks for efficiently solving a class of high-dimensional partial differential equations on a given domain. The connection between certain linear parabolic partial differential equations, so-called Kolmogorov equations, and stochastic differential equations via the Feynman-Kac formula is exploited and reformulated into an equivalent minimization problem. By way of sampling a temporal discretization of the stochastic differential equation this leads to an empirical Learning Problem for a neural network and a new way of Multilevel Learning is proposed for better performance. The latter approach is motivated by Multilevel Monte Carlo simulations and the given examples reveal encouraging success. The proposed algorithm is based on mathematical theory of numerical stochastic analysis, statistical learning and neural networks, which is presented and proven in a broad context. This general framework allows for further applications and extensions and seeks to be an important step in overcoming the curse of dimensionality, when solving high-dimensional problems involving stochastic or partial differential equations.

Zusammenfassung

Die vorliegende Arbeit vereinigt die mathematische Theorie des maschinellen Lernens mit stochastischer Analysis, um unter Verwendung von tiefen neuronalen Netzwerken eine Klasse an hochdimensionalen partiellen Differentialgleichungen auf einem gegebenen Gebiet zu lösen. Mittels der Feynman-Kac Formel wird eine Verbindung zwischen bestimmten linearen, parabolischen partiellen Differentialgleichungen, sogenannten Kolmogorov Gleichungen, und stochastischen Differentialgleichungen hergestellt und in ein äquivalentes Minimierungsproblem überführt. Nach einer temporalen Diskretisierung der stochastischen Differentialgleichung können Stichproben für ein empirisches Lernproblem simuliert werden, wobei mehrere Lernprobleme auf Basis verschiedener Diskretisierungsstufen kombiniert werden. Diese Idee entstammt sogenannten Multilevel Monte Carlo Simulationen und die numerischen Beispiele sind sehr erfolgsversprechend. Der Algorithmus stützt sich auf Resultate aus numerischer stochastischer Analysis, statistischer Lerntheorie und Theorie neuronaler Netzwerke, welche in großem Umfang präsentiert und bewiesen werden. Damit schafft die Arbeit den Rahmen für weitere Anwendungen und Forschungsarbeiten und stellt einen wichtigen Teil zur Überwindung des Fluchs der Dimensionalität bei der Lösung hochdimensionaler Probleme dar.

Contents

Chapter 1

Introduction

In many applications, especially in financial engineering, one is interested in the expected value of an output $\mathbb{E}\big[\varphi(S_T^x)\big]$ which depends on the terminal state of a solution to a stochastic differential equation of the generic form

$$S_t^x = x + \int_0^t \mu(S_s^x)\,ds + \int_0^t \sigma(S_s^x)\,dB_s\,, \quad 0 \le t \le T\,. \tag{1.1}$$

Using a temporal discretization of the stochastic differential equation and simple Monte Carlo methods this problem can be numerically solved and is well-studied in the literature (cf. Glasserman [32], Graham [34], Kloeden [53] and Hutzenthaler [44]). Furthermore by performing Multilevel Monte Carlo simulations, that is combining results with different discretization precisions, one can significantly reduce the computational cost (cf. Giles [29] & [30], Heinrich [40] and Cliffe [20]). Based on this method one can develop efficient algorithms to evaluate solutions of Kolmogorov equations, a class of linear parabolic partial differential equations,

$$\begin{cases} \frac{\partial f}{\partial t}(t,x) = \frac{1}{2}\,\mathrm{Trace}_{\mathbb{R}^d}\big(\sigma(x)[\sigma(x)]^*(\mathrm{Hess}_x\, f)(t,x)\big) + \big\langle \mu(x), (\nabla_x f)(t,x)\big\rangle_{\mathbb{R}^d} \\ f(0,x) = \varphi(x)\,. \end{cases} \tag{1.2}$$

at any given space-time locations exploiting the Feynman-Kac formula

$$f(T,x) = \mathbb{E}\big[\varphi(S_T^x)\big]. \tag{1.3}$$

Due to the dimension independent convergence rate of the Monte Carlo method this approach is particularly attractive for high-dimensional problems, where there are only a limited number of practical other algorithms.
In this thesis we want to go one step further and aim to approximate not only the solution at a given space-time location, but the entire solution function

$$[a,b]^d \ni x \mapsto f(T,x) \tag{1.4}$$

at a given time T. For this purpose we merge the theory above with classical learning theory (cf. Cucker [21], Poggio [67], Hastie [37] and James [46]) and show that the solution function above is the unique minimizer of the least squares error

$$\min_{F \in \mathscr{C}([a,b]^d,\mathbb{R})} \mathbb{E}\big[|F(X) - Y|^2\big] \tag{1.5}$$

with respect to suitable data $(X,Y) = \big(X, \varphi\big(S_T^X\big)\big)$ where X is uniformly distributed in $[a,b]^d$. Taking realizations $\big((x_i,y_i)\big)_{i=1}^m$ of independent samples drawn from the distribution of the data

by a temporal discretization of the stochastic differential equation, such as the Euler-Maruyama scheme, we arrive at the empirical Learning problem

$$\min_{F\in\mathscr{H}} \frac{1}{m}\sum_{i=1}^{m} |F(x_i)-y_i|^2 \tag{1.6}$$

over a hypothesis space $\mathscr{H}$. This allows us to tackle the problem with machine learning algorithms and we will make use of deep neural networks due to their great success in dealing with a large class of high-dimensional problems (cf. Hinton [41], Krizhevsky [55], LeCun [56] and Silver [77]). Motivated by the Multilevel Monte Carlo simulations we furthermore propose to combine empirical Learning Problems with different discretization precisions in order to decrease the computational cost of calculating the realizations. Despite the incomplete theoretical understanding of neural networks we hope that following this approach we can overcome the „curse of dimensionality“. This expression was coined by Bellman [7] and describes the problem that the complexity of algorithms for solving partial differential equations are usually exponentially scaling with the dimension. Instead of directly pursuing our goal of solving Kolmogorov equations we will present the learning theory in more generality to give the reader a complete and expandable framework, which is also applicable to other problems.
In this thesis a good understanding of measure-theoretic probability theory, stochastic analysis and functional analysis will be assumed. References are for instance Ash [4], Klenke [52], Athreya [5], Pollard [68], Le Gall [27], Kloeden [54], Schilling [75], Billingsley [10], Aliprantis [2], Cannarsa [17], Bobwroski [13] and Protter [69]. To begin with let us fix some notation, mostly straightforward generalizations for vector-valued functions. Note that we could extend most of the content of this thesis even to Banach space-valued functions (cf., for example, Da Prato [23]).

Definition 1.0.1 (Notation)**.** *We will follow the convention that $\mathbb{N}$ represents the positive integers and $\mathbb{N}_0=\mathbb{N}\cup\{0\}$. Let $d,n\in\mathbb{N}$, let $\mathcal{D}\subseteq\mathbb{R}^d$, let $(\Omega,\mathcal{G},\mathbb{P})$ be a probability space and let $\mathscr{H}$ be a metrizable space, then we denote*

(i) by $\mathcal{B}(\mathscr{H})$ the Borel σ-algebra of $\mathscr{H}$ (cf. Aliprantis [2, Section 4.4]),

(ii) by $\|\cdot\|_{\mathbb{R}^d}$, $\langle\cdot,\cdot\rangle_{\mathbb{R}^d}$ and $\|\cdot\|_{\mathbb{R}^{d\times n}}$ the euclidean norm and inner product on $\mathbb{R}^d$ and the Frobenius (also known as Hilbert-Schmidt) norm on $\mathbb{R}^{d\times n}$ respectively

and we say that

(i) a function $X=(X_i)_{i=1}^d\colon\Omega\to\mathcal{D}$ is a random vector if it is $\mathcal{G}/\mathcal{B}(\mathcal{D})$-measurable and

(ii) a function $F\colon\mathcal{D}\to\mathbb{R}^n$ is Borel measurable if it is $\mathcal{B}(\mathcal{D})/\mathcal{B}(\mathbb{R}^n)$-measurable.

For a random vector $X\colon\Omega\to\mathcal{D}$ we denote by $\mathbb{P}_X$ the law (also known as image or push forward measure) of X under $\mathbb{P}$ on the measurable space $(\mathcal{D},\mathcal{B}(\mathcal{D}))$ (cf. Klenke [52, Definition 1.98]). For $p\in[1,\infty)$ we define $L^p(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})$ to be the Banach space of random vectors $Y\colon\Omega\to\mathbb{R}^n$ with finite norm

$$\|Y\|_{L^p(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}=\left(\int_\Omega \|Y\|_{\mathbb{R}^n}^p\,d\mathbb{P}\right)^{\frac{1}{p}}=\left(\mathbb{E}\left[\|Y\|_{\mathbb{R}^n}^p\right]\right)^{\frac{1}{p}}<\infty\,, \tag{1.7}$$

where random vectors which agree $\mathbb{P}$-a.s. are identified as usual (cf. Da Prato [23, Section 1.1]). In the case $p=2$ the norm is induced by the inner product

$$\langle Y,Z\rangle_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}=\int_\Omega\langle Y,Z\rangle_{\mathbb{R}^n}\,d\mathbb{P}=\mathbb{E}\left[\langle Y,Z\rangle_{\mathbb{R}^n}\right] \tag{1.8}$$

for $Y, Z \in L^2(\mathbb{P}; \|\cdot\|_{\mathbb{R}^n})$. The expectation of a random vector $Y \in L^1(\mathbb{P}; \|\cdot\|_{\mathbb{R}^n})$ is given component-wise, i.e.

$$\mathbb{E}[Y] = \big(\mathbb{E}[Y_i]\big)_{i=1}^n \in \mathbb{R}^n\,, \tag{1.9}$$

the bias of Y with respect to (w.r.t.) $\tilde{y} \in \mathbb{R}^n$ is defined by

$$\operatorname{Bias}_{\tilde{y}}(Y) = \mathbb{E}[Y] - \tilde{y} \in \mathbb{R}^n \tag{1.10}$$

and the variance is given by the trace of the covariance matrix

$$\mathbb{V}[Y] = \operatorname{Trace}_{\mathbb{R}^n}\big(\operatorname{Cov}[Y]\big) = \mathbb{E}\big[\|Y - \mathbb{E}[Y]\|_{\mathbb{R}^n}^2\big] = \|Y - \mathbb{E}[Y]\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}^2 \in [0, \infty]\,. \tag{1.11}$$

Analogously we define the conditional expectation of Y given a σ-algebra $\mathcal{H} \subseteq \mathcal{G}$ component-wise, i.e.

$$\mathbb{E}[Y|\mathcal{H}] = \big(\mathbb{E}[Y_i|\mathcal{H}]\big)_{i=1}^n\,, \tag{1.12}$$

and for $Y \in L^2(\mathbb{P}; \|\cdot\|_{\mathbb{R}^n})$ we define the conditional variance of Y given $\mathcal{H}$ by

$$\mathbb{V}[Y|\mathcal{H}] = \mathbb{E}\big[\|Y - \mathbb{E}[Y|\mathcal{H}]\|_{\mathbb{R}^n}^2\big|\mathcal{H}\big]\,. \tag{1.13}$$

Let J be an arbitrary index set, let $Z^j \colon \Omega \to \mathcal{D}$, $j \in J$, be a family of mappings, then we denote by

$$\sigma\big(Z^j, j \in J\big) \tag{1.14}$$

the smallest σ-algebra $\mathcal{H}$ on Ω such that for every $j \in J$ the mapping Z^j is $\mathcal{H}/\mathcal{B}(\mathcal{D})$-measurable (cf., for instance, Klenke [52, Definition 1.79]). Furthermore we follow the convention of writing upper case letters for random vectors $X \colon \Omega \to \mathcal{D}$ and the corresponding lower case letters for their realizations

$$x = X(\omega) \in \mathcal{D} \tag{1.15}$$

for a given outcome $\omega \in \Omega$. Finally for $t, s \in \mathbb{R}$ let us define the abbreviation

$$t \wedge s = \min\{t, s\} \in \mathbb{R}\,. \tag{1.16}$$

Chapter 2

The Mathematical Learning Problem

2.1 Formulation

The definitions and notions in this chapter are based on Cucker [21], Poggio [67] and the first chapters in Hastie [37] and James [46]. Our goal is to learn a relationship among data $Z = (X, Y)$ which occurs as a random vector on some probability space $(\Omega, \mathcal{G}, \mathbb{P})$. We call X the input variables (also known as predictors, features or independent variables) and Y the output variables (also response, criterion or dependent variables) and we assume that the relationship between X and Y can be expressed by a function F via

$$Y \approx F(X)\,. \tag{2.1}$$

We seek to understand the effect of the predictors on the response variables in terms of statistical inference and estimate the output value for a given input. More precisely, observing $X(\omega)$ (for some outcome $\omega \in \Omega$) we want to predict the value of $Y(\omega)$ by $F(X(\omega))$. This can model many situations in areas of science, finance and industry.

Definition 2.1.1 (The (Mathematical) Learning Problem)**.** *Let $d, n \in \mathbb{N}$, $\mathcal{D} \subseteq \mathbb{R}^d$, let $(\Omega, \mathcal{G}, \mathbb{P})$ be a probability space and let $X\colon \Omega \to \mathcal{D}$ and $Y\colon \Omega \to \mathbb{R}^n$ be random vectors. For a Borel measurable function $F\colon\ \mathcal{D} \to \mathbb{R}^n$ define the least squares error of F w.r.t. the data $Z = (X, Y)$ by*

$$\mathcal{E}_{(X,Y)}(F) = \int_\Omega \|F(X)-Y\|_{\mathbb{R}^n}^2 \, d\mathbb{P} = \mathbb{E}\big[\|F(X)-Y\|_{\mathbb{R}^n}^2\big] = \|F(X)-Y\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}^2 \in [0,\infty]\,. \tag{2.2}$$

The (Mathematical) Learning Problem asks for a function F which minimizes $\mathcal{E}_{(X,Y)}(F)$.

Note that the integrand $\|F(X)-Y\|_{\mathbb{R}^n}^2$ is $\mathcal{G}/\mathcal{B}(\mathbb{R})$-measurable due to the facts that compositions of measurable functions are measurable, continuous functions are Borel measurable and the collection of measurable real-valued functions is closed under most point-wise operations (cf. Aliprantis [2, Section 4.5 & 4.6]). If it is clear from the context, we will use the abbreviation

$$\mathcal{E}(F) = \mathcal{E}_{(X,Y)}(F)\,. \tag{2.3}$$

One could also consider other loss functions to penalize the errors in prediction, but we will stick to the most common and convenient squared loss. In this case we get the following solution to the Mathematical Learning Problem subject to a natural assumption in order to obtain finiteness of the least squares error.

2.2 Solution

Recall the notation in Definition 1.0.1 and in particular denote by $\mathbb{P}_X$ the law of X under $\mathbb{P}$ on the measurable space $(\mathcal{D}, \mathcal{B}(\mathcal{D}))$.

Theorem 2.2.1 (Solution to the Learning Problem)**.** *Assume that Y has finite variance and let $\hat{F} \in L^2(\mathbb{P}_X; \|\cdot\|_{\mathbb{R}^n})$ be the $\mathbb{P}_X$-unique function defined by*

$$\mathcal{D} \ni x \mapsto \hat{F}(x) = \mathbb{E}[Y|X = x] \in \mathbb{R}^n \,. \tag{2.4}$$

Then for every $F \in L^2(\mathbb{P}_X; \|\cdot\|_{\mathbb{R}^n})$ it holds that

$$\mathcal{E}_{(X,Y)}(F) = \int_{\mathcal{D}} \big\|F(x) - \hat{F}(x)\big\|_{\mathbb{R}^n}^2 \, d\mathbb{P}_X(x) + \mathcal{E}_{(X,Y)}\big(\hat{F}\big) \in [0, \infty) \,. \tag{2.5}$$

The function $\hat{F}$ is called the regression function w.r.t. the data $Z = (X, Y)$ and solves the Mathematical Learning Problem.

Proof. First let us define

$$L^2(\mathbb{P}|_{\sigma(X)}; \|\cdot\|_{\mathbb{R}^n}) = \big\{V \in L^2(\mathbb{P}; \|\cdot\|_{\mathbb{R}^n}) : V \text{ is } \sigma(X)/\mathcal{B}(\mathbb{R}^n)\text{-measurable}\big\} \,. \tag{2.6}$$

and note that the assumption implies that $Y \in L^2(\mathbb{P}; \|\cdot\|_{\mathbb{R}^n})$. We will show that for all $V \in L^2(\mathbb{P}|_{\sigma(X)}; \|\cdot\|_{\mathbb{R}^n})$ it holds that

$$\mathbb{E}\big[\|V - Y\|_{\mathbb{R}^n}^2\big] = \mathbb{E}\big[\|V - \mathbb{E}[Y|X]\|_{\mathbb{R}^n}^2\big] + \mathbb{E}\big[\|Y - E[Y|X]\|_{\mathbb{R}^n}^2\big] \,. \tag{2.7}$$

This is often referred to as the least squares property of the conditional expectation and states that $E[Y|X]$ is the orthogonal projection of Y onto the linear subspace $L^2(\mathbb{P}|_{\sigma(X)}; \|\cdot\|_{\mathbb{R}^n}) \subseteq L^2(\mathbb{P}; \|\cdot\|_{\mathbb{R}^n})$. First we calculate

$$\begin{aligned} &\mathbb{E}\big[\|V - Y\|_{\mathbb{R}^n}^2\big] = \mathbb{E}\big[\|(V - E[Y|X]) + (E[Y|X] - Y)\|_{\mathbb{R}^n}^2\big] \\ &= \mathbb{E}\big[\|V - \mathbb{E}[Y|X]\|_{\mathbb{R}^n}^2\big] + \mathbb{E}\big[\|E[Y|X] - Y\|_{\mathbb{R}^n}^2\big] + 2\mathbb{E}\big[\langle V - E[Y|X], E[Y|X] - Y\rangle_{\mathbb{R}^n}\big] \end{aligned} \tag{2.8}$$

and then apply the tower property of the conditional expectation (cf., for instance, Klenke [52, Theorem 8.14]) to show that the last term equals zero

$$\begin{aligned} &\mathbb{E}\big[\langle V - E[Y|X], E[Y|X] - Y\rangle_{\mathbb{R}^n}\big] \\ &= \mathbb{E}\Big[\mathbb{E}\big[\langle V - E[Y|X], E[Y|X] - Y\rangle_{\mathbb{R}^n}\big|\sigma(X)\big]\Big] \\ &= \mathbb{E}\big[\langle V - \mathbb{E}[Y|X], \mathbb{E}[Y|X] - E[Y|X]\rangle_{\mathbb{R}^n}\big] = 0 \,. \end{aligned} \tag{2.9}$$

This proves the least squares property of the conditional expectation (2.7) and for every $F \in L^2(\mathbb{P}_X; \|\cdot\|_{\mathbb{R}^n})$ we can apply the latter to the random vector $V = F(X) \in L^2(\mathbb{P}|_{\sigma(X)}; \|\cdot\|_{\mathbb{R}^n})$ which yields

$$\mathcal{E}_{(X,Y)}(F) = \mathbb{E}\big[\|F(X) - Y\|_{\mathbb{R}^n}^2\big] = \mathbb{E}\big[\|F(X) - \mathbb{E}[Y|X]\|_{\mathbb{R}^n}^2\big] + \mathbb{E}\big[\|Y - E[Y|X]\|_{\mathbb{R}^n}^2\big] \,. \tag{2.10}$$

The function $\hat{F}$ is the $\mathbb{P}_X$-unique function such that for each $A \in \mathcal{B}(\mathcal{D})$ it holds that

$$\int_\Omega \mathbb{1}_{\{X \in A\}} \mathbb{E}[Y|X] \, d\mathbb{P} = \int_\Omega \mathbb{1}_{\{X \in A\}} Y \, d\mathbb{P} = \int_A \hat{F}(x) \, d\mathbb{P}_X(x) \tag{2.11}$$

and thus satisfies $\hat{F}(X) = \mathbb{E}[Y|X]$ (cf. Ash [4, Theorem 5.3.3 and Section 5.4]). We conclude that for every $F \in L^2(\mathbb{P}_X; \|\cdot\|_{\mathbb{R}^n})$ it holds that

$$\begin{aligned}\mathcal{E}_{(X,Y)}(F) &= \mathbb{E}\Big[\big\|F(X) - \hat{F}(X)\big\|_{\mathbb{R}^n}^2\Big] + \mathbb{E}\Big[\big\|Y - \hat{F}(X)\big\|_{\mathbb{R}^n}^2\Big] \\ &= \int_{\mathcal{D}} \big\|F(x) - \hat{F}(x)\big\|_{\mathbb{R}^n}^2 \, d\mathbb{P}_X(x) + \mathcal{E}_{(X,Y)}\big(\hat{F}\big)\,.\end{aligned} \tag{2.12}$$

□

Under a natural assumption on the output variable Y Theorem 2.2.1 implies that $\mathcal{E}_{(X,Y)}\big(\hat{F}\big)$ constitutes a lower bound on the least squares error, which only depends on the distribution of X and Y. Moreover, every Borel measurable function attaining that lower bound must coincide $\mathbb{P}_X$-a.s. with the regression function $\hat{F}$. The number $\mathcal{E}_{(X,Y)}\big(\hat{F}\big)$ can therefore be interpreted as a condition number associated to our Learning Problem with data $Z = (X, Y)$ and is also referred to as the irreducible error (see later in Proposition 2.3.7). Due to the fact that

$$\mathbb{E}\big[\mathbb{V}[Y|X]\big] = \mathbb{E}\Big[\mathbb{E}\big[\|Y - \mathbb{E}[Y|X]\|_{\mathbb{R}^n}^2 \big| X\big]\Big] = \mathcal{E}_{(X,Y)}\big(\hat{F}\big) \tag{2.13}$$

it can also be called the expected conditional variance of Y given X. Using the tower property one can further investigate properties of the random vector $\hat{F}(X) - Y$, namely

$$\begin{aligned}&\mathbb{E}\big[\hat{F}(X) - Y\big] = 0 \\ &\mathbb{V}\big[\hat{F}(X) - Y\big] = \mathcal{E}_{(X,Y)}\big(\hat{F}\big)\,.\end{aligned} \tag{2.14}$$

In order to speak of the regression function w.r.t. the data (X, Y) let us in the rest of the chapter assume that Y has finite variance.

2.3 Sampling

Usually one lacks knowledge about the distributions of X and Y or cannot explicitly calculate the conditional expectation in order to obtain the regression function $\hat{F}$. Consequently we aim to find a suitable approximation of $\hat{F}$ (i.e. „learn“ $\hat{F}$) from random samples of our data $Z = (X, Y)$.

Definition 2.3.1 (Empirical Error)**.** *Let*

$$\mathbf{z} = \big((x_i, y_i)\big)_{i=1}^m \tag{2.15}$$

be $m \in \mathbb{N}$ realizations of samples independently drawn from the distribution of (X, Y). For a function $F\colon \mathcal{D} \to \mathbb{R}^n$ we define the empirical error of F w.r.t. the realizations $\mathbf{z}$ by

$$\mathcal{E}_{\mathbf{z}}(F) = \mathcal{E}_{((x_i,y_i))_{i=1}^m}(F) = \frac{1}{m}\sum_{i=1}^m \|F(x_i) - y_i\|_{\mathbb{R}^n}^2 \in [0, \infty)\,. \tag{2.16}$$

For the sake of mathematical accuracy we reformulate Definition 2.3.1 in a more precise and general way.

Definition 2.3.2 (Empirical Error as Random Variable)**.** *Let $m \in \mathbb{N}$, let $X_i \colon \Omega \to \mathcal{D}$, $i \in \mathbb{N}_0$, and $Y_i \colon \Omega \to \mathbb{R}^n$, $i \in \mathbb{N}_0$, be random vectors such that it holds that $(X_i, Y_i) \colon \Omega \to \mathcal{D} \times \mathbb{R}^n$, $i \in \mathbb{N}_0$, are independent and identically distributed (i.i.d) random vectors with $(X_0, Y_0) = (X, Y)$. In particular we get m independent samples*

$$\mathbf{Z} = \big((X_i, Y_i)\big)_{i=1}^m \tag{2.17}$$

drawn from the distribution of our data (X, Y). We define the empirical error of a function $F \colon \mathcal{D} \to \mathbb{R}^n$ w.r.t. to the samples $\mathbf{Z}$ by

$$\begin{aligned} \mathcal{E}_{\mathbf{Z}}(F) = \mathcal{E}_{((X_i,Y_i))_{i=1}^m}(F) &= \frac{1}{m} \sum_{i=1}^m \|F(X_i) - Y_i\|_{\mathbb{R}^n}^2 \\ &= \frac{1}{m} \big\| \big[F(X_1) - Y_1 \,\big|\, F(X_2) - Y_2 \,\big|\, \dots \,\big|\, F(X_m) - Y_m \big] \big\|_{\mathbb{R}^{n \times m}}^2 . \end{aligned} \tag{2.18}$$

For a Borel measurable function $F \colon \mathcal{D} \to \mathbb{R}^n$ the empirical error $\mathcal{E}_{\mathbf{Z}}$ of F w.r.t. $\mathbf{Z}$ can therefore be interpreted as a random variable

$$\Omega \ni \omega \mapsto \mathcal{E}_{\mathbf{Z}(\omega)}(F) \in [0, \infty) . \tag{2.19}$$

For a given outcome $\omega \in \Omega$ we obtain m realizations of samples independently drawn from the distribution of (X, Y) by

$$\mathbf{Z}(\omega) = \big((X_i(\omega), Y_i(\omega))\big)_{i=1}^m = \big((x_i, y_i)\big)_{i=1}^m = \mathbf{z} \tag{2.20}$$

and the notion of the empirical error $\mathcal{E}_{\mathbf{Z}(\omega)}(F) = \mathcal{E}_{\mathbf{z}}(F)$ reduces to Definition 2.3.1. Note that we can always find a collection of independent samples drawn from the distribution of (X, Y) by enlarging our probability space to the product space $\big(\Omega^{\mathbb{N}_0}, \mathcal{F}^{\otimes \mathbb{N}_0}, \mathbb{P}^{\otimes \mathbb{N}_0}\big)$ (cf. Ash [4, chapter 2.7]). For every $i \in \mathbb{N}_0$ let $\pi_i \colon \Omega^{\mathbb{N}_0} \to \Omega$ be the projection onto the i-th coordinate given by

$$\pi_i\big((\omega_j)_{j \in \mathbb{N}_0}\big) = \omega_i \tag{2.21}$$

for every $(\omega_j)_{j \in \mathbb{N}_0} \in \Omega^{\mathbb{N}_0}$. Then it holds that

$$\big((X_i, Y_i)\big)_{i \in \mathbb{N}_0} = \big((X \circ \pi_i, Y \circ \pi_i)\big)_{i \in \mathbb{N}_0} \tag{2.22}$$

indeed constitutes such a collection of samples and we can redefine our data $Z = (X \circ \pi_0, Y \circ \pi_0)$ on $\big(\Omega^{\mathbb{N}_0}, \mathcal{F}^{\otimes \mathbb{N}_0}, \mathbb{P}^{\otimes \mathbb{N}_0}\big)$. Nevertheless we will keep writing $(\Omega, \mathcal{G}, \mathbb{P})$ for our probability space and assume that it is chosen in an appropriate way.

Remark 2.3.3 (Empirical Error as Monte Carlo Approximation)**.** *For fixed $F \in L^2(\mathbb{P}_X; \|\cdot\|_{\mathbb{R}^n})$ we obtain that $\mathcal{E}_{\mathbf{Z}}(F)$ is a Monte Carlo approximation of $\mathcal{E}(F)$ with order m (cf. Graham [34], Kloeden [53], Keller [48], Müller-Gronbach [60] and Glasserman [32] for literature on Monte Carlo methods). Due to the strong law of large numbers (cf. Klenke [52, Theorem 5.17] and Athreya [5, Example 8.2.2]) the empirical error $\mathcal{E}_{\mathbf{Z}}(F)$ $\mathbb{P}$-a.s. converges to*

$$\mathbb{E}\big[\mathcal{E}_{\mathbf{Z}}(F)\big] = \frac{1}{m} \sum_{i=1}^m \mathbb{E}\big[\|F(X_i) - Y_i\|_{\mathbb{R}^n}^2\big] = \mathcal{E}(F) \tag{2.23}$$

as the number of samples m tends to infinity. Notice that we seek to make the defect $\mathcal{E}_{\mathbf{Z}}(F) - \mathcal{E}(F)$ as small as possible for every suitable F in order to assure that by minimizing the empirical error we also minimize the least squares error. If the variance of $\|F(X) - Y\|_{\mathbb{R}^n}^2$ is finite, i.e.

$$\sigma^2 = \mathbb{V}\big[\|F(X) - Y\|_{\mathbb{R}^n}^2\big] < \infty , \tag{2.24}$$

one can make additional statements about the rate of convergence. Due to the independence of the samples it follows that the mean squared error of the defect equals

$$\mathbb{E}\left[|\mathcal{E}_{\mathbf{Z}}(F) - \mathcal{E}(F)|^2\right] = \mathbb{V}\left[\mathcal{E}_{\mathbf{Z}}(F)\right] = \frac{1}{m^2}\sum_{i=1}^{m}\mathbb{V}\left[\|F(X_i) - Y_i\|_{\mathbb{R}^n}^2\right] = \frac{\sigma^2}{m} \tag{2.25}$$

and by Chebyshev's inequality (cf., for instance, Billingsley [10, Section 5]) it holds for every $\varepsilon > 0$ *that*

$$\mathbb{P}\left[|\mathcal{E}_{\mathbf{Z}}(F) - \mathcal{E}(F)| \geq \varepsilon\right] \leq \frac{\sigma^2}{m\varepsilon^2}\,. \tag{2.26}$$

If further there exists $M \in (0,\infty)$ *such that it holds* $\mathbb{P}$*-a.s. that* $\|F(X) - Y\|_{\mathbb{R}^n} \leq M$ *(which implies* $\sigma^2 \leq M^4$*), we could employ Hoeffding's inequality (cf. Boucheron [15, Theorem 2.8])*

$$\mathbb{P}\left[|\mathcal{E}_{\mathbf{Z}}(F) - \mathcal{E}(F)| \geq \varepsilon\right] \leq 2\exp\left(-\frac{2m\varepsilon^2}{M^4}\right) \tag{2.27}$$

or alternatively Bernstein's inequality (cf. Boucheron [15, Corollary 2.11])

$$\mathbb{P}\left[|\mathcal{E}_{\mathbf{Z}}(F) - \mathcal{E}(F)| \geq \varepsilon\right] \leq 2\exp\left(-\frac{m\varepsilon^2}{2\left(\sigma^2 + \frac{1}{3}M^2\varepsilon\right)}\right)\,. \tag{2.28}$$

Whereas for sufficient small m *Chebyshev's inequality can yield better bounds, for larger values of* m *Bernstein's inequality is superior to Hoeffding's if* σ^2 *is significantly smaller than* M^4 *(cf. Cucker [21]).*

Since any function F vanishing on the realization points, i.e.

$$F(x_i) - y_i = 0 \tag{2.29}$$

for every $i \in \{1, 2, \ldots, m\}$, has zero empirical error, one needs to restrict the class of allowed functions first in order to have a well-posed empirical Learning Problem. Informally „Learning processes do not take place in a vacuum. Some structure needs to be present at the beginning of the process“ (Cucker [21]). We define $\mathscr{B}(\mathcal{D}, \mathbb{R}^n)$ to be the Banach space of bounded Borel measurable functions $F\colon \mathcal{D} \to \mathbb{R}^n$ equipped with the norm

$$\|F\|_\infty = \sup_{x\in\mathcal{D}} \|F(x)\|_{\mathbb{R}^n} \tag{2.30}$$

(cf. Bobrowski [13, Definition 2.2.15 & Exercise 2.2.17]).

Definition 2.3.4 (Hypothesis space, (Empirical) Target Function)**.** *Let* $\mathscr{H}$ *be a compact (topological) subspace of* $\mathscr{B}(\mathcal{D}, \mathbb{R}^n)$ *and let*

$$\mathbf{z} = \big((x_i, y_i)\big)_{i=1}^m \tag{2.31}$$

be m *realizations of samples independently drawn from the distribution of* (X, Y)*. We call* $\mathscr{H}$ *the hypothesis space and seek to find minimizers of the empirical error in that space. Accordingly we define the empirical target function* $\hat{F}_{\mathbf{z}}$ *in the hypothesis space* $\mathscr{H}$ *w.r.t. the realizations* $\mathbf{z}$ *to be a function minimizing the empirical error* $\mathcal{E}_{\mathbf{z}}(F)$ *over* $F \in \mathscr{H}$*, i.e. a minimizer of*

$$\min_{F\in\mathscr{H}} \frac{1}{m}\sum_{i=1}^{m} \|F(x_i) - y_i\|_{\mathbb{R}^n}^2\,. \tag{2.32}$$

As the regression function $\hat{F}$ *is in general not an element of our hypothesis space* $\mathscr{H}$*, we define the target function* $\hat{F}_{\mathscr{H}}$ *to be a function minimizing the least squares error* $\mathcal{E}(F)$ *over* $F \in \mathscr{H}$*, i.e. a minimizer of*

$$\min_{F\in\mathscr{H}} \mathbb{E}\left[\|F(X) - Y\|_{\mathbb{R}^n}^2\right]. \tag{2.33}$$

Let us show that the minima in the definition are well-defined. First note that $F \in \mathscr{H} \subseteq \mathscr{B}(\mathcal{D}, \mathbb{R}^n)$ implies $F \in L^2(\mathbb{P}_X; \|\cdot\|_{\mathbb{R}^n})$. Together with the assumption that Y has finite variance the least squares error and the empirical error represent continuous functions from $\big(\mathscr{H}, \|\cdot\|_\infty\big)$ to $\big([0,\infty), |\cdot|\big)$. Indeed the square root of each error can be written as a norm and the corresponding inverse triangle inequality yields the claim. Since the space $\mathscr{H}$ is compact, the minima will be attained (cf., for instance, Aliprantis [2, Corollary 2.35]). However for uniqueness one would need to consider a *convex* compact subspace of $\mathscr{B}(\mathcal{D}, \mathbb{R}^n)$. Although the minimizer is not necessarily unique, we define the empirical target function $\hat{F}_{\mathbf{z}}$ by abuse of notation to be *one* (not further specified) minimizer of the empirical error in the hypothesis space $\mathscr{H}$ (and analogously for the target function $\hat{F}_{\mathscr{H}}$). The optimization algorithm used for minimizing the empirical error over $\mathscr{H}$ depends on the given hypothesis space and we will see two versions in Example 2.3.10 and Section 4.4. In accordance with Definition 2.3.2 we want to stress the dependency of the empirical target function on different realizations of the samples and state the following more elaborated definition.

Definition 2.3.5 (Empiricial Target Function as Random Function)**.** *Let*

$$\mathbf{Z} = \big((X_i, Y_i)\big)_{i=1}^m \tag{2.34}$$

be $m \in \mathbb{N}$ independent samples drawn from the distribution of (X, Y). For every outcome $\omega \in \Omega$ we define the empirical target function $\hat{F}_{\mathbf{Z}(\omega)}$ in the hypothesis space $\mathscr{H}$ w.r.t. the samples $\mathbf{Z}$ to be a function minimizing the empirical error $\mathcal{E}_{\mathbf{Z}(\omega)}(F)$ over $F \in \mathscr{H}$, i.e. a minimizer of

$$\min_{F \in \mathscr{H}} \frac{1}{m} \sum_{i=1}^m \|F(X_i(\omega)) - Y_i(\omega)\|_{\mathbb{R}^n}^2 . \tag{2.35}$$

Therefore the empricial target function w.r.t. the samples $\mathbf{Z}$ can be interpreted as a random function

$$\Omega \ni \omega \mapsto \hat{F}_{\mathbf{Z}(\omega)} \in \mathscr{H} \tag{2.36}$$

and the next lemma deals with its measurability.

Lemma 2.3.6 (Measurability of the Empirical Target Function)**.** *For every $\omega \in \Omega$ one can choose the empirical target function $\hat{F}_{\mathbf{Z}(\omega)}$ in a way, such that it holds that*

(i) the mapping

$$\Omega \ni \omega \mapsto \hat{F}_{\mathbf{Z}(\omega)} \in \mathscr{H} \tag{2.37}$$

is $\mathcal{G}/\mathcal{B}(\mathscr{H})$-measurable and

(ii) the mapping

$$\Omega \times \mathcal{D} \ni (\omega, x) \mapsto \hat{F}_{\mathbf{Z}(\omega)}(x) \in \mathbb{R}^n \tag{2.38}$$

is $(\mathcal{G} \otimes \mathcal{B}(\mathcal{D}))/\mathcal{B}(\mathbb{R}^n)$-measurable.

Proof. First observe that $\mathscr{H}$ is a separable metric space induced by the uniform norm. Definition 2.3.2 indicates that for every $F \in \mathscr{H}$ the function

$$\Omega \ni \omega \mapsto \mathcal{E}_{\mathbf{Z}(\omega)}(F) \in [0,\infty) \tag{2.39}$$

is $\mathcal{G}/\mathcal{B}([0,\infty))$-measurable and the discussion after Definition 2.3.4 shows that for every $\omega \in \Omega$ the function

$$\mathscr{H} \ni F \mapsto \mathcal{E}_{\mathbf{Z}(\omega)}(F) \in [0,\infty) \tag{2.40}$$

is continuous, thus $\mathcal{B}(\mathscr{H})/\mathcal{B}([0,\infty))$-measurable. This implies that the function

$$\Omega \times \mathscr{H} \ni (\omega, F) \mapsto \mathcal{E}_{\mathbf{Z}(\omega)}(F) \in [0,\infty) \tag{2.41}$$

is in the class of Carathéodory functions and $(\mathcal{G} \otimes \mathcal{B}(\mathscr{H}))/\mathcal{B}([0,\infty))$-measurable (cf. Aliprantis [2, Lemma 4.51]). The Measurable Maximum Theorem in Aliprantis [2, Theorem 18.19 with $(S,\Sigma) = (\Omega,\mathcal{G})$, $X = \mathscr{H}$, $f(s,x) = \mathcal{E}_{\mathbf{Z}(s)}(x)$ and $\varphi(s) = \mathscr{H}$ for every $s \in S$] assures that the set-valued function of minimizers of

$$\min_{F \in \mathscr{H}} \mathcal{E}_{\mathbf{Z}(\omega)}(F) \tag{2.42}$$

admits a measurable selector. That is to say, there exists a $\mathcal{G}/\mathcal{B}(\mathscr{H})$-measurable mapping $\hat{F}_{\mathbf{Z}} \colon \Omega \to \mathscr{H}$ such that for every $\omega \in \Omega$ the function $\hat{F}_{\mathbf{Z}(\omega)}$ is a minimizer of (2.42). This establishes item (i). For the proof of the second item observe that the evaluation functional

$$\mathcal{D} \times \mathscr{H} \ni (x, F) \mapsto \operatorname{eval}_x(F) = F(x) \in \mathbb{R}^n \tag{2.43}$$

is also a Carathéodory function and therefore $(\mathcal{B}(\mathcal{D}) \otimes \mathcal{B}(\mathscr{H}))/\mathcal{B}(\mathbb{R}^n)$-measurable. This yields the claim as compositions of measurable functions are again measurable. □

For the subsequent analysis we will assume that the empirical target function is chosen in the sense of Lemma 2.3.6. This and Theorem 2.2.1 allow us to view the least squares error of the empirical target function $\hat{F}_{\mathbf{Z}(\omega)}$ as a random variable

$$\Omega \ni \omega \mapsto \mathcal{E}\big(\hat{F}_{\mathbf{Z}(\omega)}\big) = \int_{\mathcal{D}} \big\|\hat{F}_{\mathbf{Z}(\omega)}(x) - \hat{F}(x)\big\|_{\mathbb{R}^n}^2 \, d\mathbb{P}_X(x) + \mathcal{E}\big(\hat{F}\big) \in [0,\infty) \tag{2.44}$$

depending on the different realizations of our samples. With the following proposition it is possible to analyze the (expected) generalization error we make by only using samples of our data and by the assumption that the dependency between the input and output variables can be expressed by functions in some hypothesis space.

Proposition 2.3.7 (Least Squares Error Decompositions)**.** *The (expected) least squares error of our empirical target function $\mathcal{E}\big(\hat{F}_{\mathbf{Z}}\big)$ can be decomposed into three nonnegative quantities, namely*

(i) a sample error $S(\mathbf{Z},\mathscr{H})$ or variance term,

(ii) a model error (also known as approximation error) $M(\mathscr{H})$ or squared bias term and

(iii) an irreducible error by

$$\mathcal{E}\big(\hat{F}_{\mathbf{Z}}\big) = \underbrace{\Big(\mathcal{E}\big(\hat{F}_{\mathbf{Z}}\big) - \mathcal{E}\big(\hat{F}_{\mathscr{H}}\big)\Big)}_{S(\mathbf{Z},\mathscr{H})} + \underbrace{\Big(\mathcal{E}\big(\hat{F}_{\mathscr{H}}\big) - \mathcal{E}\big(\hat{F}\big)\Big)}_{M(\mathscr{H})} + \underbrace{\mathcal{E}\big(\hat{F}\big)}_{\text{irred. error}} \tag{2.45}$$

or

$$\mathbb{E}\big[\mathcal{E}\big(\hat{F}_{\mathbf{Z}}\big)\big] = \int_{\mathcal{D}} \mathbb{V}\big[\hat{F}_{\mathbf{Z}}(x)\big] + \big\|\operatorname{Bias}_{\hat{F}(x)}\big(\hat{F}_{\mathbf{Z}}(x)\big)\big\|_{\mathbb{R}^n}^2 \, d\mathbb{P}_X(x) + \mathcal{E}\big(\hat{F}\big) \tag{2.46}$$

respectively.

Proof. Note that in view of equation (2.44) the sample error is again a random variable and there is nothing to proof for the first decomposition. For the second claim let us denote by $\mathbb{P}_{\mathbf{Z}}$ the law of the random vector

$$\mathbf{Z} = \big((X_i, Y_i)\big)_{i=1}^m \colon \Omega \to (\mathcal{D} \times \mathbb{R}^n)^m \tag{2.47}$$

on the measurable space

$$\big((\mathcal{D} \times \mathbb{R}^n)^m, \mathcal{B}((\mathcal{D} \times \mathbb{R}^n)^m)\big)\,. \tag{2.48}$$

An application of Theorem 2.2.1 and Tonelli's Theorem (cf. Athreya [5, Theorem 5.2.1]) yield

$$\begin{aligned} \mathbb{E}\big[\mathcal{E}\big(\hat{F}_{\mathbf{Z}}\big)\big] &= \int_{(\mathcal{D}\times\mathbb{R}^n)^m} \int_{\mathcal{D}} \big\|\hat{F}_\zeta(x) - \hat{F}(x)\big\|_{\mathbb{R}^n}^2 \, d\mathbb{P}_X(x)\, d\mathbb{P}_{\mathbf{Z}}(\zeta) + \mathcal{E}\big(\hat{F}\big) \\ &= \int_{\mathcal{D}} \mathbb{E}\left[\big\|\hat{F}_{\mathbf{Z}}(x) - \hat{F}(x)\big\|_{\mathbb{R}^n}^2\right] d\mathbb{P}_X(x) + \mathcal{E}\big(\hat{F}\big)\,. \end{aligned} \tag{2.49}$$

Finally we employ a special version of the least squares property (2.7) to rewrite the expectation

$$\begin{aligned} \mathbb{E}\left[\big\|\hat{F}_{\mathbf{Z}}(x) - \hat{F}(x)\big\|_{\mathbb{R}^n}^2\right] &= E\left[\big\|\hat{F}(x) - \mathbb{E}\big[\hat{F}_{\mathbf{Z}}(x)\big]\big\|_{\mathbb{R}^n}^2\right] + \mathbb{E}\left[\big\|\hat{F}_{\mathbf{Z}}(x) - \mathbb{E}\big[\hat{F}_{\mathbf{Z}}(x)\big]\big\|_{\mathbb{R}^n}^2\right] \\ &= \big\|\hat{F}(x) - \mathbb{E}\big[\hat{F}_{\mathbf{Z}}(x)\big]\big\|_{\mathbb{R}^n}^2 + \mathbb{E}\left[\big\|\hat{F}_{\mathbf{Z}}(x) - \mathbb{E}\big[\hat{F}_{\mathbf{Z}}(x)\big]\big\|_{\mathbb{R}^n}^2\right] \end{aligned} \tag{2.50}$$

for every $x \in \mathcal{D}$ and combining (2.49) with (2.50) proves the second decomposition. □

Given a fixed hypothesis space, we can only control the sample error (by increasing m) as solely the latter depends on the samples $\mathbf{Z}$. On the other hand if we fix the number of samples m and enlarge the hypothesis space $\mathscr{H}$, the model error will certainly decrease, but usually the sample error will at some point increase due to so-called overfitting. Since a too complex model for the empirical target function will be able to depict random fluctuations caused by the variance of the samples, it does not generalize well to the whole data (see Example 2.3.10 below). This trade-off between minimizing $S(\mathbf{Z}, \mathscr{H})$ and $M(\mathscr{H})$ suggests that there is an optimal complexity of our hypothesis space. This is closely tied to the so-called bias-variance trade-off (with bias related to the model error and variance to the sample error). According to Bishop [11]: „A model which is too simple, or too inflexible, will have a large bias, while one which has too much flexibility in relation to the particular data set will have a large variance. Bias and variance are complementary quantities, and the best generalization is obtained when we have the best compromise between the conflicting requirements of small bias and small variance.“
One can make the relations between the number of samples m and the capacity of our hypothesis space $\mathscr{H}$ more precise by considering the covering number of $\mathscr{H}$ and using the concentration inequalities from Remark 2.3.3 to bound the sample error $S(\mathbf{Z}, \mathscr{H})$. For every $\delta \in (0, \infty)$ the δ-covering number of $\mathscr{H}$, denoted by $N(\mathscr{H}, \delta)$, is defined as the minimal $k \in \mathbb{N}$ such that there exist k balls in $\mathscr{H}$ with radius δ covering $\mathscr{H}$.

Theorem 2.3.8 (Bound on the Sample Error)**.** *Assume that there exists a real number $M \in (0, \infty)$ such that for every $F \in \mathscr{H}$ it holds $\mathbb{P}$-a.s. that*

$$\|F(X) - Y\|_{\mathbb{R}^n} \leq M \tag{2.51}$$

and let

$$\sigma^2 = \sup_{F \in \mathscr{H}} \mathbb{V}\big[\|F(X) - Y\|_{\mathbb{R}^n}^2\big] \in \big[0, M^4\big]\,. \tag{2.52}$$

Then it holds for every $\varepsilon > 0$ that

$$\mathbb{P}\left[\mathcal{E}\big(\hat{F}_{\mathbf{Z}}\big) - \mathcal{E}\big(\hat{F}_{\mathscr{H}}\big) \leq \varepsilon\right] \geq 1 - 2N\big(\mathscr{H}, \tfrac{\varepsilon}{16M}\big) \exp\left(-\frac{m\varepsilon^2}{8\big(4\sigma^2 + \frac{1}{3}M^2\varepsilon\big)}\right)\,. \tag{2.53}$$

Proof. Cucker [21, Theorem C] □

Furthermore for various hypothesis spaces it is also possible to estimate the model error (cf. Cucker [21] and Poggio [67]). Before we go on to our first example we prove a technical lemma which allows us to compute the regression function $\hat{F}(x) = \mathbb{E}[Y|X = x]$ in a certain setting.

Lemma 2.3.9 (Computation of the Regression Function)**.** *Let (C, Σ) be a measurable space, let $W\colon \Omega \to C$ be a $\mathcal{G}/\Sigma$-measurable random variable independent of X and for every $x \in \mathcal{D}$ let $\Phi^x\colon C \to \mathbb{R}^n$ be a $\Sigma/\mathcal{B}(\mathbb{R}^n)$-measurable function. Assume that it holds that*

(i) $Y = \Phi^X(W)$ $\mathbb{P}$-a.s. and

(ii) for every $u \in C$ the mapping $\mathcal{D} \ni x \mapsto \Phi^x(u) \in \mathbb{R}^n$ is continuous.

Then it holds that

$$\hat{F}(x) = \mathbb{E}[Y|X = x] = \mathbb{E}\big[\Phi^X(W)\big|X = x\big] = \mathbb{E}\big[\Phi^x(W)\big] \tag{2.54}$$

for $\mathbb{P}_X$-a.s. $x \in \mathcal{D}$.

Proof. First we note that the second item establishes that the mapping

$$C \times \mathcal{D} \ni (u, x) \mapsto \Phi^x(u) \in \mathbb{R}^n \tag{2.55}$$

is a Carathéodory function and $(\Sigma \otimes \mathcal{B}(\mathcal{D}))/\mathcal{B}(\mathbb{R}^n)$-measurable. Thus Fubini's theorem and the independence of X and W assure that for all $A \in \mathcal{B}(\mathcal{D})$ it holds that

$$\begin{aligned}
&\int_\Omega \mathbb{1}_{\{X\in A\}} Y \, d\mathbb{P} = \int_\Omega \mathbb{1}_{\{X\in A\}} \Phi^X(W) \, d\mathbb{P} = \int_{\mathcal{D}\times C} \mathbb{1}_{\{x\in A\}} \Phi^x(u) \, d\mathbb{P}_{(X,W)}(x,u) \\
&= \int_C \int_{\mathcal{D}} \mathbb{1}_{\{x\in A\}} \Phi^x(u) \, d\mathbb{P}_X(x) d\mathbb{P}_W(u) = \int_A \int_C \Phi^x(u) \, d\mathbb{P}_W(u) d\mathbb{P}_X(x) \\
&= \int_A \int_\Omega \Phi^x(W) \, d\mathbb{P} d\mathbb{P}_X(x) = \int_A \mathbb{E}\big[\Phi^x(W)\big] d\mathbb{P}_X(x)
\end{aligned} \tag{2.56}$$

and the defining property of the conditional expectation (2.11) yields the claim (cf. Pollard [68, Chapter 4] and Aliprantis [2, Change of Variables Theorem 13.46]). □

Now we will illustrate the concepts behind the Learning Problem and the bias-variance trade-off with an example.

Example 2.3.10. *Let $p \in \mathbb{N}$, $a, \varrho, c_0, \ldots, c_p \in \mathbb{R}$, $b \in [a, \infty)$, let $\varphi\colon \mathbb{R} \to \mathbb{R}$ be a polynomial of the form $\varphi(x) = \sum_{j=0}^p c_j x^j$, let $X\colon \Omega \to [a, b]$ be a (continuously) uniformly distributed random variable, let $W\colon \Omega \to \mathbb{R}$ be a standard normal random variable independent of X and define the random variable $Y\colon \Omega \to \mathbb{R}$ by*

$$Y = \varphi\big(Xe^{\varrho W}\big)\,. \tag{2.57}$$

It is well known that the Laplace transform of the standard normal random variable W satisfies

$$\mathbb{E}\big[e^{zW}\big] = e^{\frac{1}{2}z^2} \tag{2.58}$$

for all $z \in \mathbb{C}$ (cf., for instance, Le Gall [27, Section 1.1]). The assumptions of Theorem 2.2.1 are satisfied and it follows that the Learning Problem of minimizing

$$\mathcal{E}_{(X,Y)}(F) = \mathbb{E}\big[|F(X) - Y|^2\big] \tag{2.59}$$

is solved by the regression function

$$\hat{F}(x) = \mathbb{E}[Y \big| X = x] = \mathbb{E}\big[\varphi\big(xe^{\varrho W}\big)\big] = \sum_{j=0}^{p} c_j x^j \mathbb{E}\big[e^{j\varrho W}\big] = \sum_{j=0}^{p} c_j x^j e^{\frac{1}{2}(j\varrho)^2} \tag{2.60}$$

for $\mathbb{P}_X$-a.s. $x \in [a,b]$, where in the second equality we used Lemma 2.3.9 (with $\Phi^x(u) = \varphi(xe^{\varrho u})$ for every $x \in [a,b]$ and $u \in \mathbb{R}$). Denote by λ the Lebesgue measure on $\big([a,b], \mathcal{B}([a,b])\big)$ and observe that

$$\mathbb{P}_X = \tfrac{1}{b-a}\lambda \tag{2.61}$$

and that the regression function $\hat{F}$ is $\mathbb{P}_X$-unique. Therefore we can state that the version of the regression function $\hat{F}$ defined above is the is unique continuous function solving the Learning Problem. Next suppose we want to learn this relationship between X and Y from m realizations

$$\mathbf{z} = \big((x_i, y_i)\big)_{i=1}^{m} \tag{2.62}$$

of independent samples drawn from the distribution of the data $Z = (X, Y)$ (see Figure 2.1).

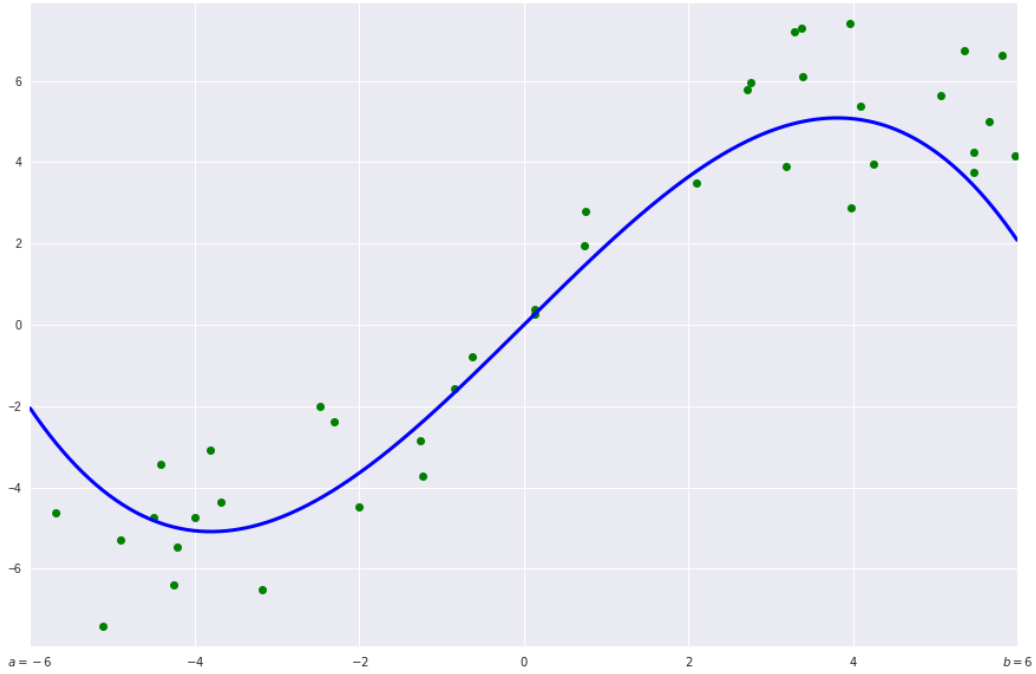

Figure 2.1: The plot shows the regression function (blue) and $m = 40$ realizations (green) of independent samples drawn from the distribution of the data (X, Y) with $\varrho = 0.5, a = -6, b = 6, p = 3, c_0 = 0, c_1 = 1.77, c_2 = 0, c_3 = -0.015$.

Let us fix the space $\mathscr{P}_s([a,b], \mathbb{R})$ of polynomials on $[a,b]$ with a maximal degree of $s \in \mathbb{N}$ and coefficients bounded by $R \in (0, \infty)$ as a valid hypothesis space

$$\mathscr{H} = \mathscr{P}_s([a,b], \mathbb{R})\,. \tag{2.63}$$

Minimizing the empirical error over $\mathscr{P}_s([a,b], \mathbb{R})$ is equivalent to a linear least squares problem. Every function $F \in \mathscr{P}_s([a,b], \mathbb{R})$ can be written as $F(x) = \sum_{j=0}^{s} \theta_j x^j$ and by defining

$$A = (x_i^j)_{1 \le i \le m, 0 \le j \le s} \in \mathbb{R}^{m,s+1}, \quad y = (y_i)_{1 \le i \le m} \in \mathbb{R}^m, \quad \theta = (\theta_j)_{0 \le j \le s} \in [-R, R]^{s+1} \tag{2.64}$$

it holds that

$$\min_{F\in\mathscr{P}_s([a,b],\mathbb{R})} \mathcal{E}_{\mathbf{z}}(F) = \min_{\theta\in[-R,R]^{s+1}} \|A\theta - y\|^2_{\mathbb{R}^m} \,. \tag{2.65}$$

From linear algebra (cf. Roman [71, Theorem 17.3]) it is known that minimizing coefficients θ are given by applying the Moore-Penrose pseudoinverse $A^\dagger$ of A to y (if R is chosen big enough), i.e. the empirical target function is defined by

$$\hat{F}_{\mathbf{z}}(x) = \sum_{j=0}^{s} (A^\dagger y)_j x^j \tag{2.66}$$

for every $x \in [a,b]$. The next three figures illustrate the bias-variance trade-off.

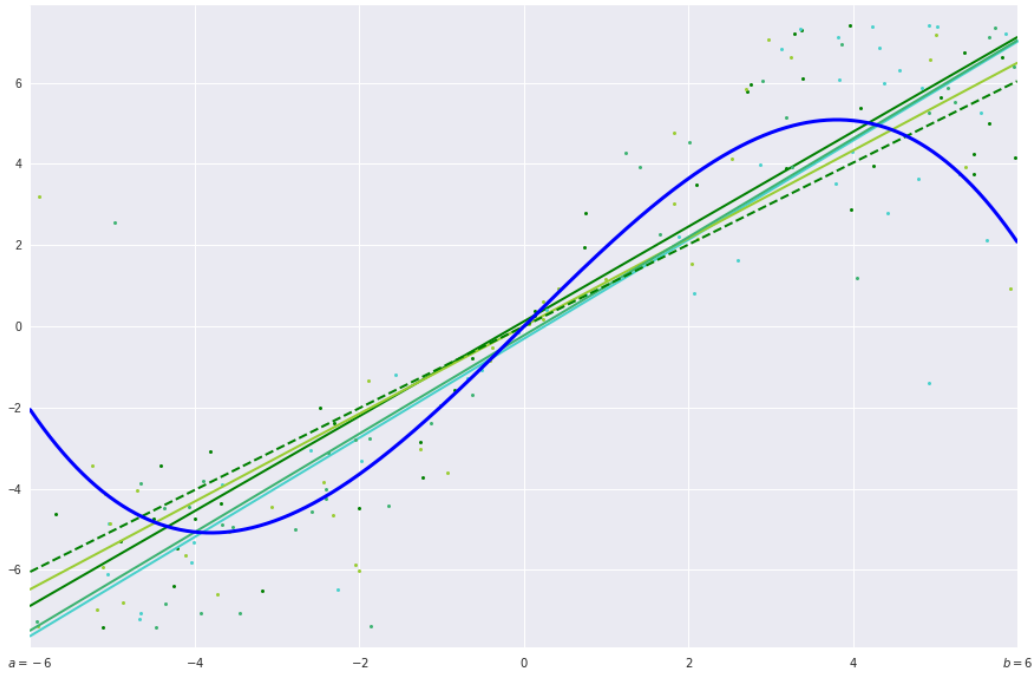

Figure 2.2: The plot shows the previous regression function (blue) and four empirical target functions in $\mathscr{P}_1([a,b],\mathbb{R})$ w.r.t. four independent trials with 40 samples each (green colors). One can observe that the functions do not deviate much from each other (small variance), but cannot approximate the regression function (large bias). The dashed line corresponds to the target function, which can be computed by orthogonal projection.

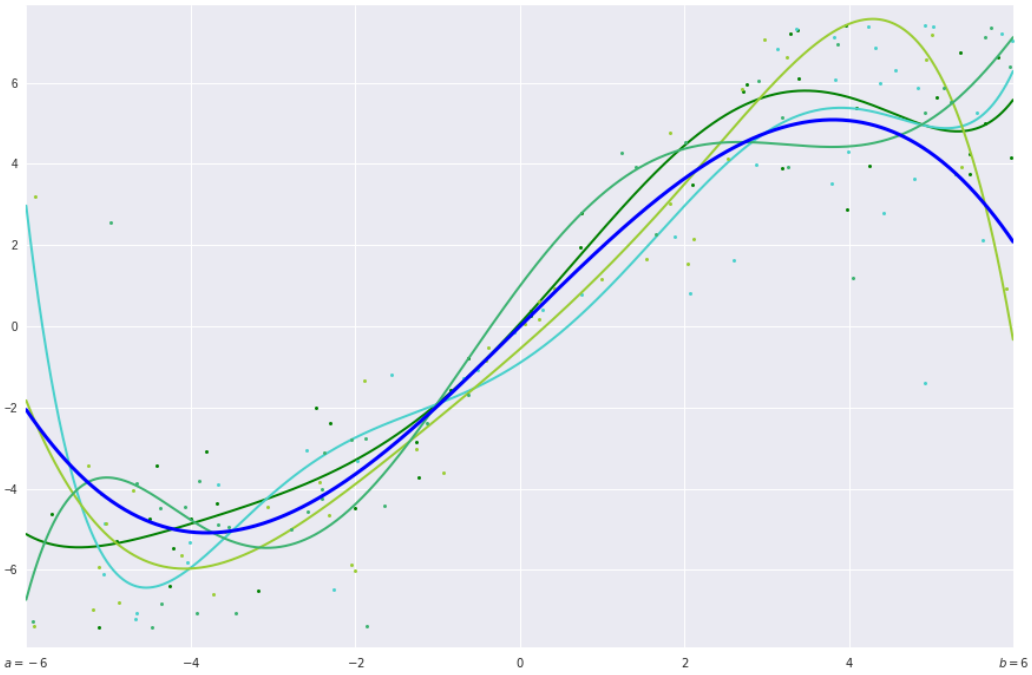

Figure 2.3: The plot shows the previous regression function (blue) and four empirical target functions in $\mathscr{P}_6([a,b],\mathbb{R})$ w.r.t. four independent trials with 40 samples each (green colors). One can observe that the functions approximate the regression function better (smaller bias), but deviate more from each other (larger variance). The target function equals the regression function in this case.

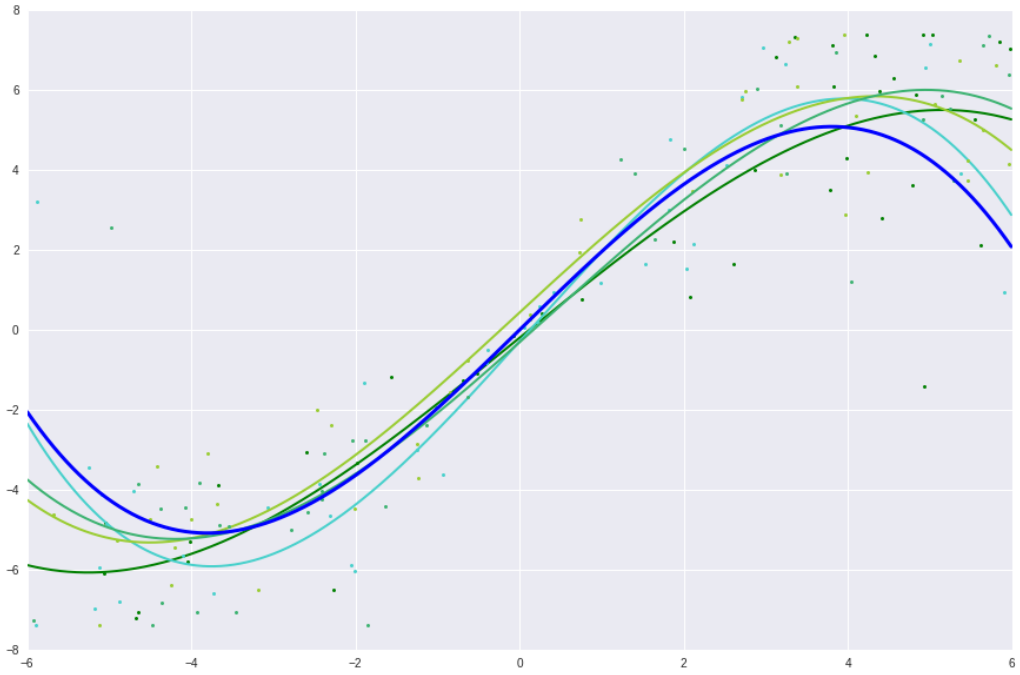

Figure 2.4: The plot shows the previous regression function (blue) and four empirical target functions in $\mathscr{P}_3([a,b],\mathbb{R})$ w.r.t. four independent trials with 40 samples each (green colors). This corresponds to the optimal complexity of the hypothesis space in order to balance the bias-variance trade-off.

If the size of the hypothesis space is chosen to small, i.e. in our case the degree s, we tend to make a large model error. The (empirical) target function may be a too simplistic model and underfits the data in the sense that it cannot capture the relevant relations between X and Y. This can be identified with a large (squared) bias

$$\frac{1}{b-a}\int_{[a,b]} \left|\operatorname{Bias}_{\hat{F}(x)}\left(\hat{F}_{\mathbf{z}}(x)\right)\right|^2 dx = \frac{1}{b-a}\int_{[a,b]} \left|\mathbb{E}\left[\hat{F}_{\mathbf{z}}(x) - \hat{F}(x)\right]\right|^2 dx \tag{2.67}$$

but usually a lower variance

$$\frac{1}{b-a}\int_{[a,b]} \mathbb{V}\left[\hat{F}_{\mathbf{z}}(x)\right] dx = \frac{1}{b-a}\int_{[a,b]} \mathbb{E}\left[\left|\hat{F}_{\mathbf{z}}(x) - \mathbb{E}\left[\hat{F}_{\mathbf{z}}(x)\right]\right|^2\right] dx \tag{2.68}$$

over different samples (see Figure 2.2). On the other side choosing a large hypothesis space certainly reduces the model error as the target function can better represent the regression function. Employing a more complex model for the empirical target function will capture the samples more accurately and thus lead to a lower bias but due to the fluctuations in the samples also to a higher variance (see Figure 2.3). This so-called overfitting of the samples results also in a larger sample error. Of course in this case the choice of $s = p$ would be the optimal complexity for our hypothesis space (see Figure 2.4) in order to minimize both errors. However a priori one usually does not know the regression function and it is a difficult task to choose a model that both accurately captures the regularities in the samples, but also generalizes well to the whole data.

2.4 Multilevel Learning

Assume that our output data is approximated by a simulation process with adjustable precision. That means we do not have access to $Z = (X, Y)$ directly but can only compute realizations of approximated data $\mathcal{Z}^N = \left(X, \mathcal{Y}^N\right)$, $N \in \mathbb{N}$, where increasing N is denoting a more precise, but also more computationally expensive simulation. For the analysis let us state some assumptions on the asymptotic behavior of the approximation quality and the computational cost for the simulations.

(A) Let $\alpha, \gamma \in (0, \infty)$. We assume that $Y\colon \Omega \to \mathbb{R}^n$ and $\mathcal{Y}^N\colon \Omega \to \mathbb{R}^n$, $N \in \mathbb{N}$, have finite variance and that there exists a strictly increasing sequence $N_l \in \mathbb{N}$, $l \in \mathbb{N}_0$, of precision levels such that the quality of the approximation $\left\|Y - \mathcal{Y}^{N_l}\right\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}$ and the expected computational cost $\mathbb{E}\left[\mathcal{C}^{N_l}\right]$ of calculating a single realization of the output variable $\mathcal{Y}^{N_l}$ follow the asymptotic behavior

$$\left\|Y - \mathcal{Y}^{N_l}\right\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})} = \mathcal{O}\left(N_l^{-\alpha}\right), \tag{2.69}$$

and

$$\mathbb{E}\left[\mathcal{C}^{N_l}\right] = \mathcal{O}\left(N_l^{\gamma}\right) \tag{2.70}$$

for l tending to infinity.

We keep the notation $\hat{F}$ for the regression function w.r.t. the data (X, Y) and for every $N \in \mathbb{N}$ we define $\hat{\mathcal{F}}^N$ to be the regression function w.r.t. the data $\left(X, \mathcal{Y}^N\right)$. Let us revisit the empirical Learning Problem according to Section 2.3 using the approximated data $\mathcal{Z}^{N_l} = (X, \mathcal{Y}^{N_l})$. Let $l \in \mathbb{N}_0$ and assume we have simulated m realizations of samples independently drawn from

the distribution of the approximated data $\mathcal{Z}^{N_l} = \big(X, \mathcal{Y}^{N_l}\big)$, which we will also denote by $\mathbf{z} = \big((x_i, y_i)\big)_{i=1}^m$. Then for a given hypothesis space $\mathscr{H}$, compact subset of $\mathscr{B}(\mathcal{D}, \mathbb{R}^n)$, we aim to learn $\hat{\mathcal{F}}^{N_l}$ and define the empirical target function $\hat{\mathcal{F}}_{\mathbf{z}}^{N_l}$ to be a function minimizing the empirical error $\mathcal{E}_{\mathbf{z}}(F)$ over $F \in \mathscr{H}$, i.e. an optimizer of

$$\min_{F \in \mathscr{H}} \frac{1}{m} \sum_{i=1}^{m} \|F(x_i) - y_i\|_{\mathbb{R}^n}^2 . \tag{2.71}$$

In the same fashion we define the target function $\hat{\mathcal{F}}_{\mathscr{H}}^{N_l}$ w.r.t. to the data $\mathcal{Z}^{N_l}$ to be a function minimizing the least squares error $\mathcal{E}_{(X,\mathcal{Y}^{N_l})}(F)$ over $F \in \mathscr{H}$. By solving the minimizing problem (2.71) and therefore obtaining an empirical target function $\hat{\mathcal{F}}_{\mathbf{z}}^{N_l}$ we make three kinds of errors, which have to be controlled in order to state that

$$\hat{\mathcal{F}}_{\mathbf{z}}^{N_l}(x) \approx \hat{\mathcal{F}}_{\mathscr{H}}^{N_l}(x) \approx \hat{\mathcal{F}}^{N_l}(x) \approx \hat{F}(x) \tag{2.72}$$

for $x \in \mathcal{D}$:

(i) Error by using approximated data $\mathcal{Z}^{N_l}$ instead of Z - can be controlled by choosing l large enough due to assumption (A).

(ii) Model error $M(\mathscr{H})$ by seeking a minimizer in a hypothesis space - can be controlled by selection of a sufficiently large hypothesis space

(iii) Sample error $S(\mathbf{z}, \mathscr{H})$ by using realizations of samples instead of the actual data distribution - can be controlled by choosing a less complex hypothesis space and sufficiently large number of samples m (see Theorem 2.3.8).

Let us again stress the bias-variance trade-off we face for choosing the hypothesis space in items (ii) and (iii). We will try to cope with this problem in Chapter 4 by choosing appropriate deep neural networks as hypothesis space. In order to decrease the computational cost for computing large number m of samples at a high precision (large l), which is necessary due to items (i) and (ii), we will introduce a new way of Multilevel Learning in this chapter. Observe that assumption (A), Jensen's inequality (cf., for instance, Athreya [5, Theorem 12.2.4]) and Theorem 2.2.1 assure that it holds that

$$\begin{aligned} &\mathcal{E}_{(X,Y)}\big(\hat{\mathcal{F}}^{N_l}\big) - \mathcal{E}_{(X,Y)}\big(\hat{F}\big) = \int_{\mathcal{D}} \big\|\hat{\mathcal{F}}^{N_l}(x) - \hat{F}(x)\big\|_{\mathbb{R}^n}^2 \, d\mathbb{P}_X(x) \\ &= \mathbb{E}\Big[\big\|\mathbb{E}\big[\mathcal{Y}^{N_l} - Y\big|X\big]\big\|_{\mathbb{R}^n}^2\Big] \leq \big\|Y - \mathcal{Y}^{N_l}\big\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}^2 = \mathcal{O}\big(N_l^{-2a}\big) \end{aligned} \tag{2.73}$$

for l tending to infinity. Therefore in mathematical terms we obtain that

$$\mathcal{E}\big(\hat{\mathcal{F}}_{\mathbf{z}}^{N_l}\big) = \underbrace{\Big(\mathcal{E}\big(\hat{\mathcal{F}}_{\mathbf{z}}^{N_l}\big) - \mathcal{E}\big(\hat{\mathcal{F}}_{\mathscr{H}}^{N_l}\big)\Big)}_{S(\mathbf{z},\mathscr{H})} + \underbrace{\Big(\mathcal{E}\big(\hat{\mathcal{F}}_{\mathscr{H}}^{N_l}\big) - \mathcal{E}\big(\hat{\mathcal{F}}^{N_l}\big)\Big)}_{M(\mathscr{H})} + \underbrace{\Big(\mathcal{E}\big(\hat{\mathcal{F}}^{N_l}\big) - \mathcal{E}\big(\hat{F}\big)\Big)}_{\mathcal{O}(N_l^{-2a})} + \mathcal{E}\big(\hat{F}\big) , \tag{2.74}$$

where the least squares error is taken w.r.t. to the data (X, Y) (see also Figure 2.5). In the spirit of Remark 2.3.3 let us compute the mean squared error of the defect. Throughout the rest of this section we will stick to the following notation for the samples of $\big(X, \mathcal{Y}^N\big)$ on different levels of precision N. Let $X_{i,l} \colon \Omega \to \mathcal{D}$, $i, l \in \mathbb{N}_0$, and $\mathcal{Y}_{i,l}^N \colon \Omega \to \mathbb{R}^n$, $i, l \in \mathbb{N}_0, N \in \mathbb{N}$, be random vectors such that it holds that the σ-algebras

$$\sigma\big(\big(X_{i,l}, \mathcal{Y}_{i,l}^N\big), i \in \mathbb{N}_0, N \in \mathbb{N}\big) , \quad l \in \mathbb{N}_0 , \tag{2.75}$$

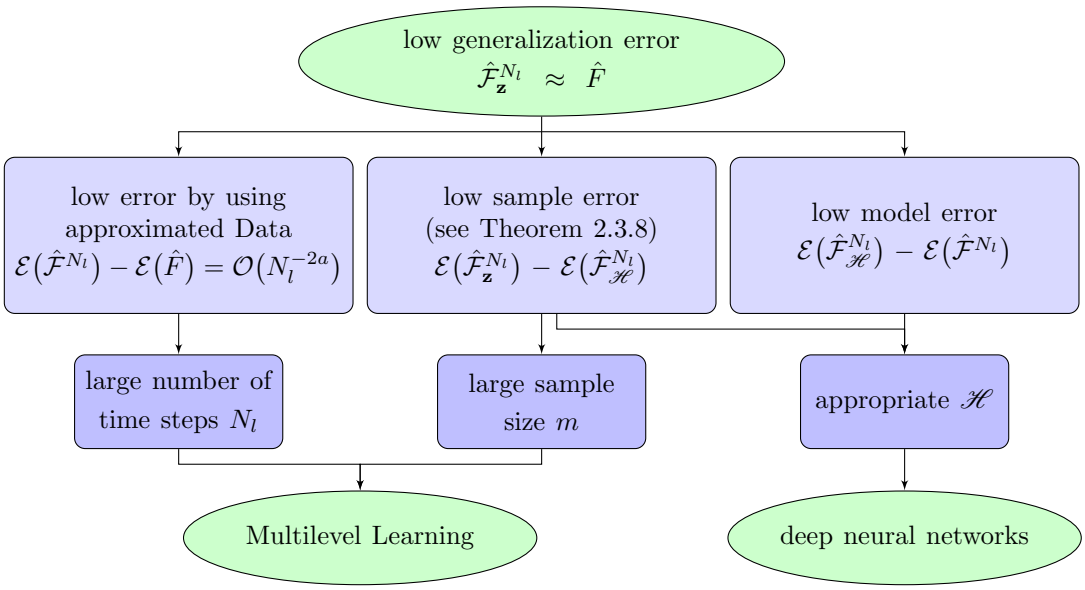

Figure 2.5: Schematic representation of equation (2.74) suggesting possible steps to reduce the generalization error.

are independent and for every $N \in \mathbb{N}$ it holds that $\left(X_{i,l}, \mathcal{Y}_{i,l}^N\right) \colon \Omega \to \mathcal{D} \times \mathbb{R}^n$, $i, l \in \mathbb{N}_0$, are i.i.d. random vectors with $\left(X_{0,0}, \mathcal{Y}_{0,0}^N\right) = \left(X, \mathcal{Y}^N\right)$. In the beginning we will not make use of the second index, thus we fix $l = 0$ and omit writing the index, i.e. $\left(X_i, \mathcal{Y}_i^N\right) = \left(X_{i,0}, \mathcal{Y}_{i,0}^N\right)$.

Proposition 2.4.1 (Mean Squared Error of the Defect for Approximated Data)**.** *Let $F \in \mathscr{H}$ and assume that $Y \in L^4(\mathbb{P}; \|\cdot\|_{\mathbb{R}^n})$ and $\sup_{l\in\mathbb{N}} \|\mathcal{Y}^{N_l}\|_{L^4(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})} < \infty$. Then there exists a real number $C \in (0, \infty)$ such that for every $l, m \in \mathbb{N}$ it holds that*

$$\mathbb{E}\left[\left|\mathcal{E}_{((X_i,\mathcal{Y}_i^{N_l}))_{i=1}^m}(F) - \mathcal{E}_{(X,Y)}(F)\right|^2\right] \leq C\left(\left\|Y - \mathcal{Y}^{N_l}\right\|^2_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})} + \frac{1}{m}\right) \tag{2.76}$$

Proof. First observe that for vectors $u, v, w \in \mathbb{R}^n$ it holds that

$$\left|\|u-v\|^2_{\mathbb{R}^n} - \|u-w\|^2_{\mathbb{R}^n}\right| = \left|\langle 2u - w - v, w - v\rangle_{\mathbb{R}^n}\right| \leq \|2u - w - v\|_{\mathbb{R}^n}\|w - v\|_{\mathbb{R}^n}\,. \tag{2.77}$$

Then by the same decomposition as in equation (2.50) it holds for every $l \in \mathbb{N}$ that

$$\begin{aligned}
&\mathbb{E}\left[\left|\mathcal{E}_{((X_i,\mathcal{Y}_i^{N_l}))_{i=1}^m}(F) - \mathcal{E}_{(X,Y)}(F)\right|^2\right] \\
&= \left(\mathbb{E}\left[\mathcal{E}_{((X_i,\mathcal{Y}_i^{N_l}))_{i=1}^m}(F)\right] - \mathcal{E}_{(X,Y)}(F)\right)^2 + \mathbb{V}\left[\mathcal{E}_{((X_i,\mathcal{Y}_i^{N_l}))_{i=1}^m}(F)\right] \\
&= \left(\mathbb{E}\left[\left\|F(X) - \mathcal{Y}^{N_l}\right\|^2_{\mathbb{R}^n} - \|F(X) - Y\|^2_{\mathbb{R}^n}\right]\right)^2 + \frac{1}{m}\mathbb{V}\left[\left\|F(X) - \mathcal{Y}^{N_l}\right\|^2_{\mathbb{R}^n}\right] \\
&\leq \left(\mathbb{E}\left[\left\|2F(X) - Y - \mathcal{Y}^{N_l}\right\|_{\mathbb{R}^n}\left\|Y - \mathcal{Y}^{N_l}\right\|_{\mathbb{R}^n}\right]\right)^2 + \frac{1}{m}\mathbb{E}\left[\left\|F(X) - \mathcal{Y}^{N_l}\right\|^4_{\mathbb{R}^n}\right] \\
&\leq \underbrace{\mathbb{E}\left[\left\|2F(X) - Y - \mathcal{Y}^{N_l}\right\|^2_{\mathbb{R}^n}\right]}_{\mathbf{A}_l}\mathbb{E}\left[\left\|Y - \mathcal{Y}^{N_l}\right\|^2_{\mathbb{R}^n}\right] + \frac{1}{m}\underbrace{\mathbb{E}\left[\left\|F(X) - \mathcal{Y}^{N_l}\right\|^4_{\mathbb{R}^n}\right]}_{\mathbf{B}_l}
\end{aligned} \tag{2.78}$$

and due to the assumptions we can bound both factors $\mathbf{A}_l$ and $\mathbf{B}_l$ uniformly over $l \in \mathbb{N}$, i.e.

$$\begin{aligned}
\sup_{l\in\mathbb{N}} \mathbf{A}_l &\leq \sup_{l\in\mathbb{N}} \left(2\|F\|_\infty + \|Y\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})} + \left\|\mathcal{Y}^{N_l}\right\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}\right)^2 < \infty \\
\sup_{l\in\mathbb{N}} \mathbf{B}_l &\leq \sup_{l\in\mathbb{N}} \left(\|F\|_\infty + \left\|\mathcal{Y}^{N_l}\right\|_{L^4(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}\right)^4 < \infty\,.
\end{aligned} \tag{2.79}$$

Defining $C = \max\{\sup_{l\in\mathbb{N}} \mathbf{A}_l, \sup_{l\in\mathbb{N}} \mathbf{B}_l\}$ proves the proposition. □

Using the approximated data we observe that the term $\frac{1}{m}$ due to the Monte Carlo approximation (see Remark 2.3.3) remains, but also the new term $\left\|Y - \mathcal{Y}^{N_l}\right\|^2_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}$ is introduced, which by assumption (A) has an asymptotic behavior of $\mathcal{O}\big(N_l^{-2\alpha}\big)$ as l tends to infinity. Again this stresses the need to take a sufficiently large number of samples m, as well as a sufficiently large value for l, i.e. a sufficiently precise simulation, in order to keep the defect small. More precisely for a mean squared error of the defect of size ε^2, one needs to choose $m, l \in \mathbb{N}$ such that

$$N_l = \mathcal{O}\big(\varepsilon^{-\frac{1}{\alpha}}\big) \tag{2.80}$$

and

$$m = \mathcal{O}(\varepsilon^{-2}) \tag{2.81}$$

and thus gets an expected overall computational cost for the simulations of

$$\mathbb{E}[\mathcal{C}] = m\mathbb{E}\big[\mathcal{C}^{N_l}\big] = \mathcal{O}\big(\varepsilon^{-2}\big)\mathcal{O}\big(N_l^{\gamma}\big) = \mathcal{O}\big(\varepsilon^{-2}\big)\mathcal{O}\big(\varepsilon^{-\frac{\gamma}{\alpha}}\big) = \mathcal{O}\big(\varepsilon^{-\frac{\gamma}{\alpha}-2}\big) \tag{2.82}$$

for ε tending to zero. We try to improve on the overall computational cost by taking advantage of the idea of Multilevel Monte Carlo simulations (cf. Giles [29] & [30], Cliffe [20] and Heinrich [40]). For this Multilevel Learning approach we use data based on different levels of precision.

Proposition 2.4.2 (Multilevel Learning)**.** *Let $L \in \mathbb{N}$ and define the Multilevel data $\big(\mathfrak{Z}^l\big)_{l=0}^L$ with*

(i) 0-th level data $\mathfrak{Z}^0 = \mathcal{Z}^{N_0} = \big(X, \mathcal{Y}^{N_0}\big)$

(ii) l-th level data $\mathfrak{Z}^l = \big(X, \mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}}\big)$ for $l \in \{1, 2, \dots, L\}$

and for every $l \in \{0, 1, \dots, L\}$ denote by $\hat{\mathfrak{F}}^l$ the regression function w.r.t. to the data $\mathfrak{Z}^l$. Then for $\mathbb{P}_X$-a.s. $x \in \mathcal{D}$ it holds that

$$\sum_{l=0}^{L} \hat{\mathfrak{F}}^l(x) = \hat{\mathcal{F}}^{N_L}(x)\,. \tag{2.83}$$

Proof. For the convenience of the reader we define $\mathcal{Y}^{N_{-1}} = 0$ and thus it holds that

$$\big(\mathfrak{Z}^l\big)_{l=0}^L = \Big(\big(X, \mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}}\big)\Big)_{l=0}^L\,. \tag{2.84}$$

By assumption (A) for every $l \in \{0, 1, \dots, L\}$ the output data $\mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}}$ has finite variance and according to Theorem 2.2.1 it holds that

$$\hat{\mathfrak{F}}^l(x) = \mathbb{E}\big[\mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}}\big|X = x\big] \tag{2.85}$$

for $\mathbb{P}_X$-a.s. $x \in \mathcal{D}$. Therefore the linearity of the conditional expectation establishes that

$$\begin{aligned}\sum_{l=0}^{L} \hat{\mathfrak{F}}^l(x) &= \mathbb{E}\left[\sum_{l=0}^{L} \mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}}\middle| X = x\right] \\ &= \mathbb{E}\big[\mathcal{Y}^{N_L} - \mathcal{Y}^{N_{-1}}\big|X = x\big] = \mathbb{E}\big[\mathcal{Y}^{N_L}\big|X = x\big] = \hat{\mathcal{F}}^{N_L}(x)\end{aligned} \tag{2.86}$$

for $\mathbb{P}_X$-a.s. $x \in \mathcal{D}$. □

That shows that solving the (standard) Learning Problem w.r.t. to the data

$$\mathcal{Z}^{N_L} = \big(X, \mathcal{Y}^{N_L}\big) \tag{2.87}$$

is equivalent to solving the $L+1$ Learning Problems with Multilevel data

$$\big(\mathfrak{Z}^l\big)_{l=0}^{L} = \Big(\big(X, \mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}}\big)\Big)_{l=0}^{L} \tag{2.88}$$

and adding up the regression functions. The advantage of the latter approach is that we expect the differences $\mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}}$ to be small as both terms approximate the same output data only on a different level of precision. This would lead to a smaller variance in the output variables and we expect to need less samples to obtain an accurate estimate via the empirical target function. Only in the lowest level $l = 0$ we do not have this reduction, but in exchange the simulations of the output data at lower levels have decreased computational cost. Thus we hope that following this Multilevel Learning method we get the same precision of the empirical Learning Problem at a significantly lower computational cost. But we cannot perform a rigorous analysis due to the minimization procedure and the severe dependency of the (empirical) target function on the chosen hypothesis space. Thus for motivational purposes we will only consider the extreme case of $\mathcal{D} = \{x\}$ for $x \in \mathbb{R}^d$. Then for every random vector $V \in L^2(\mathbb{P}; \|\cdot\|_{\mathbb{R}^n})$, every $m \in \mathbb{N}$ and i.i.d. samples $\mathbf{Z} = \big((X_i, V_i)\big)_{i=1}^{m}$ drawn from the distribution of $(X, V) = (x, V)$ the empirical target function $\hat{F}_{\mathbf{Z}}$ w.r.t. to the samples $\mathbf{Z}$ satisfies that

$$\hat{F}_{\mathbf{Z}}(x) = \frac{1}{m} \sum_{i=1}^{m} V_i \tag{2.89}$$

and thus it holds that

$$\mathbb{E}\big[\hat{F}_{\mathbf{Z}}(x)\big] = \mathbb{E}[V] \tag{2.90}$$

and

$$\mathbb{V}\big[\hat{F}_{\mathbf{Z}}(x)\big] = \frac{1}{m} \mathbb{V}[V]\,. \tag{2.91}$$

Under these assumptions we can make it plausible that it is asymptotically advantageous to use the Multilevel Learning method.

Proposition 2.4.3 (Computational Cost of Multilevel Learning)**.** *Let $x \in \mathbb{R}^d$, assume that $\mathcal{D} = \{x\}$, assume that assumption (A) holds with $\alpha \geq \frac{\gamma}{2}$ and that $\sup_{l \in \mathbb{N}} \big\|\mathcal{Y}^{N_l}\big\|_{L^2(\mathbb{P}; \|\cdot\|_{\mathbb{R}^n})} < \infty$. Then it holds that there exist $L \in \mathbb{N}$ and $m_0, m_1, \ldots, m_L \in \mathbb{N}$ such that by defining the Multilevel samples*

$$\begin{aligned} \mathbf{Z}_0 &= \big((X_{0,i}, \mathcal{Y}_{0,i}^{N_0})\big)_{i=1}^{m_0}\,, \\ \mathbf{Z}_l &= \big((X_{l,i}, \mathcal{Y}_{l,i}^{N_l} - \mathcal{Y}_{l,i}^{N_{l-1}})\big)_{i=1}^{m_l} \end{aligned} \tag{2.92}$$

for $l \in \{1, 2, \ldots, L\}$ and the corresponding empirical target functions $\hat{\mathfrak{F}}_{\mathbf{Z}_l}$ we obtain a mean squared error

$$\mathbb{E}\left[\left\|\sum_{l=0}^{L} \hat{\mathfrak{F}}_{\mathbf{Z}_l}(X) - \hat{F}(X)\right\|_{\mathbb{R}^n}^2\right] = \mathcal{O}\big(\varepsilon^2\big) \tag{2.93}$$

with an expected overall computational cost for the simulations of

$$\mathbb{E}[\mathcal{C}] = \begin{cases} \mathcal{O}\big(\varepsilon^{-2}\big), & \text{if } \alpha > \frac{\gamma}{2} \\ \mathcal{O}\big(\varepsilon^{-2}(\log \varepsilon)^2\big), & \text{if } \alpha = \frac{\gamma}{2} \end{cases} \tag{2.94}$$

for ε tending towards zero.

Proof. Define $\mathcal{Y}^{N_{-1}} = 0$ for a simpler notation and recall the decomposition in equation (2.50). This, the independence of the Multilevel samples $(\mathbf{Z}_l)_{l=0}^L$ and equations (2.90) & (2.91) assure that it holds that

$$\begin{aligned}
&\mathbb{E}\left[\left\|\sum_{l=0}^{L}\hat{\mathfrak{F}}_{\mathbf{Z}_l}(X) - \hat{F}(X)\right\|_{\mathbb{R}^n}^2\right] = \sum_{l=0}^{L}\mathbb{V}\left[\hat{\mathfrak{F}}_{\mathbf{Z}_l}(x)\right] + \left\|\hat{F}(x) - \sum_{l=0}^{L}\mathbb{E}\left[\hat{\mathfrak{F}}_{\mathbf{Z}_l}(x)\right]\right\|_{\mathbb{R}^n}^2 \\
&= \sum_{l=0}^{L}\frac{1}{m_l}\mathbb{V}\left[\mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}}\right] + \left\|\mathbb{E}\left[Y\right] - \sum_{l=0}^{L}\mathbb{E}\left[\mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}}\right]\right\|_{\mathbb{R}^n}^2 \\
&\leq \sum_{l=0}^{L}\frac{1}{m_l}\mathbb{E}\left[\left\|\mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}}\right\|_{\mathbb{R}^n}^2\right] + \mathbb{E}\left[\left\|Y - \mathcal{Y}^{N_L}\right\|_{\mathbb{R}^n}^2\right] \\
&\leq \sum_{l=0}^{L}\frac{1}{m_l}\left(\left\|Y - \mathcal{Y}^{N_l}\right\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})} + \left\|Y - \mathcal{Y}^{N_{l-1}}\right\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}\right)^2 + \left\|Y - \mathcal{Y}^{N_L}\right\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}^2.
\end{aligned} \tag{2.95}$$

The claim follows by using a theorem on the complexity of Multilevel Monte Carlo simulations in Cliffe [20, Theorem 1 with $\beta = 2\alpha$] or Giles [29, Theorem 3.1]. Note that in the latter references it is assumed for simplicity that there exists $s \in \mathbb{N}\setminus\{1\}$ such that for all $l = 1, 2, \ldots, L$ it holds that $N_l = sN_{l-1}$. □

For a discussion on optimal or sensible values for the numbers $L, m_0, m_1, \ldots, m_L, s$ we refer the reader to Giles [29]. Note that the computational cost of the standard approach in this special case coincides with the estimate in equation (2.82). Indeed by choosing $m, l \in \mathbb{N}$ such that $m = \mathcal{O}\left(\varepsilon^{-2}\right)$ and $N_l = \mathcal{O}\left(\varepsilon^{-\frac{1}{\alpha}}\right)$, defining the samples

$$\mathbf{Z} = \left(\left(X_i, \mathcal{Y}_i^{N_l}\right)\right)_{i=1}^m \tag{2.96}$$

and the corresponding empirical target function $\hat{\mathcal{F}}_{\mathbf{Z}}^{N_l}$ we obtain a mean squared error

$$\begin{aligned}
\mathbb{E}\left[\left\|\hat{\mathcal{F}}_{\mathbf{Z}}^{N_l}(X) - \hat{F}(X)\right\|_{\mathbb{R}^n}^2\right] &= \mathbb{V}\left[\hat{\mathcal{F}}_{\mathbf{Z}}^{N_l}(x)\right] + \left\|\hat{F}(x) - \mathbb{E}\left[\hat{\mathcal{F}}_{\mathbf{Z}}^{N_l}(x)\right]\right\|_{\mathbb{R}^n}^2 \\
&= \frac{1}{m}\mathbb{V}\left[\mathcal{Y}^{N_l}\right] + \left\|\mathbb{E}\left[Y - \mathcal{Y}^{N_l}\right]\right\|_{\mathbb{R}^n}^2 \\
&\leq \frac{1}{m}\sup_{j\in\mathbb{N}}\left\|\mathcal{Y}^{N_j}\right\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}^2 + \left\|Y - \mathcal{Y}^{N_l}\right\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^n})}^2 = \mathcal{O}\left(\varepsilon^2\right)
\end{aligned} \tag{2.97}$$

with an expected overall computational cost for the simulations of

$$\mathbb{E}[\mathcal{C}] = \mathcal{O}\left(\varepsilon^{-\frac{\gamma}{\alpha}-2}\right). \tag{2.98}$$

Therefore under the (harsh) assumptions of the previous Theorem 2.4.3 the Multilevel Learning approach can yield significant computational savings and we hope that this will transfer also to more general settings. In Section 3.5 we revisit the Multilevel Learning approach in the context of Kolmogorov equations and Chapter 5 presents promising numerical results.

Remark 2.4.4. *Note that we did not include the computational cost of minimizing the empirical error, i.e. finding the empirical target function, in our analysis. In other words we assumed the cost of the $L+1$ minimization problems in the Multilevel setting to be comparable to the cost of a single minimization problem for the standard setting or neglectable in comparison to the cost of the simulations.*

Chapter 3

Stochastic Interpretation of Kolmogorov Equations

In this chapter we will investigate the link between a class of linear parabolic partial differential equations, so-called Kolmogorov equations, and solution processes to stochastic differential equations with the help of the Feynman-Kac formula. This probabilistic interpretation allows us to reformulate the problem of calculating the solution to a Kolmogorov equation into a Learning Problem as defined in Chapter 2. Thereupon we will temporally discretize the solution process to the stochastic differential equation by the Euler-Maruyama scheme in order to obtain samples for the empirical Learning Problem, analyze the occurring errors and propose a suitable Multilevel Learning Problem.

3.1 Setting

Throughout this chapter let $T \in (0, \infty)$, $d \in \mathbb{N}$, let $(\Omega, \mathcal{G}, \mathbb{P})$ be an appropriate probability space equipped with a complete filtration $(\mathcal{G}_t)_{t\in[0,T]}$, let

$$B = \big((B_t^1, B_t^2, \dots, B_t^d)\big)_{t\in[0,T]} \colon [0,T] \times \Omega \to \mathbb{R}^d \tag{3.1}$$

be a standard $(\Omega, \mathcal{G}, \mathbb{P}, (\mathcal{G}_t)_{t\in[0,T]})$-Brownian motion (cf., for instance, Schilling [75]), let

$$\begin{aligned} \mu &= (\mu_i)_{i=1}^d \colon \mathbb{R}^d \to \mathbb{R}^d \\ \sigma &= (\sigma_{ij})_{i,j=1}^d \colon \mathbb{R}^d \to \mathbb{R}^{d\times d} \end{aligned} \tag{3.2}$$

be (globally) Lipschitz continuous functions, i.e. there exists $K \in (0, \infty)$ such that for all $x, y \in \mathbb{R}^d$ it holds that

$$\begin{aligned} \|\mu(x) - \mu(y)\|_{\mathbb{R}^d} &\leq K \|x - y\|_{\mathbb{R}^d} \\ \|\sigma(x) - \sigma(y)\|_{\mathbb{R}^{d\times d}} &\leq K \|x - y\|_{\mathbb{R}^d} \end{aligned} \tag{3.3}$$

and let $\varphi \colon \mathbb{R}^d \to \mathbb{R}$ be a twice continuously differentiable function with an at most polynomially growing gradient, i.e. there exists $c \in [1, \infty)$ such that for all $x \in \mathbb{R}^d$ it holds that

$$\|\nabla\varphi(x)\|_{\mathbb{R}^d} \leq c\big(1 + \|x\|_{\mathbb{R}^d}\big)^c. \tag{3.4}$$

Observe that the Lipschitz condition (3.3) on σ and μ implies that there exists a real number $\bar{K} \in (0, \infty)$ such that for every $x \in \mathbb{R}^d$ the linear growth condition

$$\begin{aligned} \|\mu(x)\|_{\mathbb{R}^d} &\leq \bar{K}\big(1 + \|x\|_{\mathbb{R}^d}\big) \\ \|\sigma(x)\|_{\mathbb{R}^{d\times d}} &\leq \bar{K}\big(1 + \|x\|_{\mathbb{R}^d}\big) \end{aligned} \tag{3.5}$$

holds (cf. Schilling [75, Section 18.3]). Recall the definition of stopping times and stochastic integrals (cf., for example, Le Gall [27, Definition 2.18 and Chapter 5]) and note that we compute multi-dimensional integrals component-wise, that is for suitable integrands

$$\begin{aligned} G &= \left((G_t^i)_{i=1}^d\right)_{t\in[0,T]} \colon [0,T]\times\Omega \to \mathbb{R}^d\,, \\ H &= \left((H_t^{ij})_{i,j=1}^d\right)_{t\in[0,T]} \colon [0,T]\times\Omega \to \mathbb{R}^{d\times d} \end{aligned} \tag{3.6}$$

and $0 \leq \tau_1 \leq \tau_2 \leq T$ we define

$$\int_{\tau_1}^{\tau_2} G_s\,ds = \left(\int_{\tau_1}^{\tau_2} G_s^i\,ds\right)_{i=1}^d \tag{3.7}$$

and

$$\int_{\tau_1}^{\tau_2} H_s\,dB_s = \left(\sum_{j=1}^d \int_{\tau_1}^{\tau_2} H_s^{ij}\,dB_s^j\right)_{i=1}^d . \tag{3.8}$$

Let us proof bounds on these stochastic and deterministic integrals, which will be used in subsequent proofs.

Lemma 3.1.1 (Bounds on (Stochastic) Integrals)**.** *Let $p \in [2,\infty)$, $\tau_1 \in [0,T]$, $\tau_2 \in [\tau_1,T]$, let $\mathcal{T}\colon \Omega \to [0,\infty]$ be a stopping time and let $G\colon [0,T]\times\Omega \to \mathbb{R}^d$ and $H\colon [0,T]\times\Omega \to \mathbb{R}^{d\times d}$ be $(\mathcal{G}_t)_{t\in[0,T]}$-adapted stochastic processes with right-continuous sample paths, which satisfy $\mathbb{P}$-a.s. that*

$$\sup_{s\in[0,T]} \|G_s\|_{\mathbb{R}^d} < \infty \tag{3.9}$$

and

$$\sup_{s\in[0,T]} \|H_s\|_{\mathbb{R}^{d\times d}} < \infty\,. \tag{3.10}$$

Then it holds that

$$\mathbb{E}\left[\max_{\tau_1\leq t\leq\tau_2} \left\|\int_{\tau_1}^{t\wedge\mathcal{T}} G_s\,ds\right\|_{\mathbb{R}^d}^p\right] \leq (\tau_2-\tau_1)^{p-1} \int_{\tau_1}^{\tau_2} \mathbb{E}\left[\|G_{s\wedge\mathcal{T}}\|_{\mathbb{R}^d}^p\right] ds \tag{3.11}$$

and

$$\mathbb{E}\left[\max_{\tau_1\leq t\leq\tau_2} \left\|\int_{\tau_1}^{t\wedge\mathcal{T}} H_s\,dB_s\right\|_{\mathbb{R}^d}^p\right] \leq p^p(\tau_2-\tau_1)^{\frac{p}{2}-1} \int_{\tau_1}^{\tau_2} \mathbb{E}\left[\|H_{s\wedge\mathcal{T}}\|_{\mathbb{R}^{d\times d}}^p\right] ds\,. \tag{3.12}$$

Proof. For the first claim the triangle inequality, Hölder's inequality (cf., for instance, Klenke [52, Theorem 7.16]) and Tonelli's theorem yield

$$\begin{aligned} &\mathbb{E}\left[\max_{\tau_1\leq t\leq\tau_2} \left\|\int_{\tau_1}^{t\wedge\mathcal{T}} G_s\,ds\right\|_{\mathbb{R}^d}^p\right] = \mathbb{E}\left[\max_{\tau_1\leq t\leq\tau_2} \left\|\int_{\tau_1}^{t} G_s\mathbb{1}_{[0,\mathcal{T}]}\,ds\right\|_{\mathbb{R}^d}^p\right] \\ &\leq \mathbb{E}\left[\max_{\tau_1\leq t\leq\tau_2} \left(\int_{\tau_1}^{t} \|G_s\mathbb{1}_{[0,\mathcal{T}]}\|_{\mathbb{R}^d}\,ds\right)^p\right] \leq \mathbb{E}\left[\left(\int_{\tau_1}^{\tau_2} \|G_{s\wedge\mathcal{T}}\|_{\mathbb{R}^d}\,ds\right)^p\right] \\ &\leq (\tau_2-\tau_1)^{p-1} \int_{\tau_1}^{\tau_2} \mathbb{E}\left[\|G_{s\wedge\mathcal{T}}\|_{\mathbb{R}^d}^p\right] ds\,, \end{aligned} \tag{3.13}$$

for the second claim we apply the Burkholder-Davis-Gundy inequality (cf., for example, Da Prato [23, Section 4.6] and Le Gall [27, Theorem 5.16]) and Hölder's inequality to obtain

$$\begin{aligned}
&\mathbb{E}\left[\max_{\tau_1\le t\le\tau_2}\left\|\int_{\tau_1}^{t\wedge\mathcal{T}} H_s\,dB_s\right\|_{\mathbb{R}^d}^p\right] = \mathbb{E}\left[\max_{\tau_1\le t\le\tau_2}\left\|\int_{\tau_1}^{t} H_s\mathbb{1}_{[0,\mathcal{T}]}(s)\,dB_s\right\|_{\mathbb{R}^d}^p\right] \\
&\le p^p\left(\int_{\tau_1}^{\tau_2}\mathbb{E}\big[\|H_s\mathbb{1}_{[0,\mathcal{T}]}(s)\|_{\mathbb{R}^{d\times d}}^p\big]^{\frac{2}{p}}\,ds\right)^{\frac{p}{2}} \le p^p\left(\int_{\tau_1}^{\tau_2}\mathbb{E}\big[\|H_{s\wedge\mathcal{T}}\|_{\mathbb{R}^{d\times d}}^p\big]^{\frac{2}{p}}\,ds\right)^{\frac{p}{2}} \\
&\le p^p(\tau_2-\tau_1)^{\frac{p}{2}-1}\int_{\tau_1}^{\tau_2}\mathbb{E}\big[\|H_{s\wedge\mathcal{T}}\|_{\mathbb{R}^{d\times d}}^p\big]\,ds
\end{aligned} \tag{3.14}$$

and this proves the lemma. □

Let us define the notion of a solution process to a stochastic differential equation and exhibit its properties.

Definition 3.1.2 (Solution Process to a Stochastic Differential Equation)**.** *Let $X\colon\Omega\to\mathbb{R}^d$ be a $\mathcal{G}_0$-measurable random vector. We will say that $S^X\colon[0,T]\times\Omega\to\mathbb{R}^d$ is a solution process to the stochastic differential equation (SDE)*

$$S_t^X = X + \int_0^t \mu\big(S_s^X\big)\,ds + \int_0^t \sigma\big(S_s^X\big)\,dB_s \tag{3.15}$$

if $S^X = \big(S_t^X\big)_{t\in[0,T]}\colon[0,T]\times\Omega\to\mathbb{R}^d$ is an $(\mathcal{G}_t)_{t\in[0,T]}$-adapted stochastic process with continuous sample paths which satisfies that for every $t\in[0,T]$ equation (3.15) *$\mathbb{P}$-a.s. holds.*

Note that in our setting an up to indistinguishability unique solution process to the SDE (3.15) exists (cf., for instance, Le Gall [27, Theorem 8.3 & 8.5] and Schilling [75, Section 18.3]) and fulfills the following bound on the moments (cf. also Arnold [3, Section 7.1], Friedman [26, Chapter 5, Theorem 2.3] and Kloeden [54, Chapter 4] for slightly different versions).

Lemma 3.1.3 (Bound on the Moments of the Solution Process)**.** *Let $p\in[2,\infty)$, $\tau_1\in[0,T]$, let $X\colon\Omega\to\mathbb{R}^d$ be a $\mathcal{G}_0$-measurable random vector with*

$$\mathbb{E}\big[\|X\|_{\mathbb{R}^d}^p\big] < \infty \tag{3.16}$$

and let $S^X\colon[0,T]\times\Omega\to\mathbb{R}^d$ be a solution process to the SDE (3.15)*. Then there exists a real number $C\in(0,\infty)$ depending only on p, $\bar{K}$ and T such that for all $\tau_2\in[\tau_1,T]$ it holds that*

$$\mathbb{E}\left[\max_{\tau_1\le t\le\tau_2}\big\|S_t^X\big\|_{\mathbb{R}^d}^p\right] \le Ce^{C(\tau_2-\tau_1)}\Big(1+\mathbb{E}\big[\|X\|_{\mathbb{R}^d}^p\big]\Big). \tag{3.17}$$

Proof. For the convenience of the reader we will just write S instead of S^X for the solution process to the SDE (3.15). For every $m\in\mathbb{N}$ let

$$\mathcal{T}_m = \inf\big\{0\le t\le T\colon \|S_t\|_{\mathbb{R}^d}\ge m\big\} \tag{3.18}$$

be a stopping time with the convention that $\inf\varnothing=\infty$ (cf. Le Gall [27, Proposition 3.9] and Protter [69, Section I.1]), which assures that the upcoming expectations are well-defined. Several times in the proof we will apply the following elementary inequality, which can be

verified by Hölder's inequality. Let $N \in \mathbb{N}$, $\vartheta \in [1, \infty)$ and $a_1, a_2, \ldots, a_N \in \mathbb{R}$, then it holds that

$$\left(\sum_{i=1}^{N} |a_i| \right)^{\vartheta} \leq N^{\vartheta - 1} \sum_{i=1}^{N} |a_i|^{\vartheta} \leq N^{\vartheta} \sum_{i=1}^{N} |a_i|^{\vartheta} . \tag{3.19}$$

Observe that the linear growth condition (3.5) and the previous inequality imply that for every $x \in \mathbb{R}^d$ it holds that

$$\begin{aligned} \|\mu(x)\|_{\mathbb{R}^d}^p &\leq (2\bar{K})^p \left(1 + \|x\|_{\mathbb{R}^d}^p\right) \\ \|\sigma(x)\|_{\mathbb{R}^{d\times d}}^p &\leq \left(2\bar{K}\right)^p \left(1 + \|x\|_{\mathbb{R}^d}^p\right) . \end{aligned} \tag{3.20}$$

Let $\tau_1, \tau_2 \in [0, T]$ and $m \in \mathbb{N}$, then equations (3.15) & (3.19) establish that

$$\begin{aligned} \mathbb{E}\left[\max_{\tau_1 \leq t \leq \tau_2} \|S_{t\wedge\mathcal{T}_m}\|_{\mathbb{R}^d}^p \right] \leq 3^{p-1} \Bigg(\mathbb{E}\big[\|X\|_{\mathbb{R}^d}^p\big] &+ \underbrace{\mathbb{E}\left[\max_{\tau_1 \leq t \leq \tau_2} \left\| \int_{\tau_1}^{t\wedge\mathcal{T}_m} \mu(S_s)\, ds \right\|_{\mathbb{R}^d}^p \right]}_{\mathbf{A}} \\ &+ \underbrace{\mathbb{E}\left[\max_{\tau_1 \leq t \leq \tau_2} \left\| \int_{\tau_1}^{t\wedge\mathcal{T}_m} \sigma(S_s)\, dB_s \right\|_{\mathbb{R}^d}^p \right]}_{\mathbf{B}} \Bigg) , \end{aligned} \tag{3.21}$$

applying Lemma 3.1.1 and the growth estimate (3.20) to first integral in (3.21) yields

$$\begin{aligned} \mathbf{A} &\leq (\tau_2 - \tau_1)^{p-1} \int_{\tau_1}^{\tau_2} \mathbb{E}\big[\|\mu(S_{s\wedge\mathcal{T}_m})\|_{\mathbb{R}^d}^p\big]\, ds \\ &\leq (2\bar{K})^p\, T^{p-1} \left(T + \int_{\tau_1}^{\tau_2} \mathbb{E}\big[\|S_{s\wedge\mathcal{T}_m}\|_{\mathbb{R}^d}^p\big]\, ds \right) \end{aligned} \tag{3.22}$$

and for the stochastic integral we obtain analogously

$$\begin{aligned} \mathbf{B} &\leq p^p (\tau_2 - \tau_1)^{\frac{p}{2}-1} \int_{\tau_1}^{\tau_2} \mathbb{E}\big[\|\sigma(S_{s\wedge\mathcal{T}_m})\|_{\mathbb{R}^{d\times d}}^p\big]\, ds \\ &\leq (2\bar{K}p)^p\, T^{\frac{p}{2}-1} \left(T + \int_{\tau_1}^{\tau_2} \mathbb{E}\big[\|S_{s\wedge\mathcal{T}_m}\|_{\mathbb{R}^d}^p\big]\, ds \right) . \end{aligned} \tag{3.23}$$

In view of (3.21), (3.22) and (3.23) there exists a real number $C \in (0, \infty)$ depending only on p, $\bar{K}$ and T such that for every $m \in \mathbb{N}$, $\tau_2 \in [\tau_1, T]$ it holds that

$$\begin{aligned} \mathbb{E}\left[\max_{\tau_1 \leq t \leq \tau_2} \|S_{t\wedge\mathcal{T}_m}\|_{\mathbb{R}^d}^p \right] &\leq C \left(1 + \mathbb{E}\big[\|X\|_{\mathbb{R}^d}^p\big] + \int_{\tau_1}^{\tau_2} \mathbb{E}\big[\|S_{s\wedge\mathcal{T}_m}\|_{\mathbb{R}^d}^p\big]\, ds \right) \\ &\leq C \left(1 + \mathbb{E}\big[\|X\|_{\mathbb{R}^d}^p\big] + \int_{\tau_1}^{\tau_2} \mathbb{E}\left[\max_{\tau_1 \leq t \leq s} \|S_{t\wedge\mathcal{T}_m}\|_{\mathbb{R}^d}^p \right] ds \right) . \end{aligned} \tag{3.24}$$

Now we can apply Grönwall's lemma (cf. Klenke [52, Lemma 26.9] and Kloeden [54, Lemma 4.5.1]) to show that for all $m \in \mathbb{N}$, $\tau_2 \in [\tau_1, T]$ it holds that

$$\mathbb{E}\left[\max_{\tau_1 \leq t \leq \tau_2} \|S_{t\wedge\mathcal{T}_m}\|_{\mathbb{R}^d}^p \right] \leq C e^{C(\tau_2 - \tau_1)} \Big(1 + \mathbb{E}\big[\|X\|_{\mathbb{R}^d}^p\big] \Big) . \tag{3.25}$$

Note that the continuous sample paths of S imply that for every $\omega \in \Omega$ it holds that

$$\lim_{m\to\infty} \mathcal{T}_m(\omega) = \infty \tag{3.26}$$

and thus the monotone convergence theorem (cf., for instance, Athreya [5, Theorem 2.3.4]) proves the claim when letting m tend to infinity. □

Next we define the special class of parabolic linear partial differential equations we are dealing with and state sufficient conditions on the smoothness and growth of the solutions for the subsequent theory.

Definition 3.1.4 (Kolmogorov Equation)**.** *We call a function* $f = (f(t,x))_{(t,x)\in[0,T]\times\mathbb{R}^d} \in \mathscr{C}^{1,2}([0,T]\times\mathbb{R}^d,\mathbb{R})$ *a solution to the Kolmogorov equation if it satisfies for every* $t\in[0,T]$, $x=(x_1,x_2,\dots,x_d)\in\mathbb{R}^d$ *that*

$$\begin{cases} \frac{\partial f}{\partial t}(t,x) = \frac{1}{2}\operatorname{Trace}_{\mathbb{R}^d}\big(\sigma(x)[\sigma(x)]^*(\operatorname{Hess}_x f)(t,x)\big) + \big\langle \mu(x), (\nabla_x f)(t,x)\big\rangle_{\mathbb{R}^d} \\ \qquad\qquad = \frac{1}{2}\sum_{i,j,l=1}^{d} \sigma_{il}(x)\sigma_{jl}(x)\frac{\partial^2 f}{\partial x_i \partial x_j}(t,x) + \sum_{i=1}^{d}\mu_i(x)\frac{\partial f}{\partial x_i}(t,x) \\ f(0,x) = \varphi(x)\,. \end{cases} \tag{3.27}$$

We say that the solution to the Kolmogorov equation is at most polynomially growing, if there exists $c\in[1,\infty)$ *such that for every* $x\in\mathbb{R}^d$ *it holds that*

$$\max_{0\le t\le T}|f(t,x)| \le c\left(1+\|x\|_{\mathbb{R}^d}^c\right). \tag{3.28}$$

Remark 3.1.5 (Kolmogorov Backward Equation)**.** *The Kolmogorov equation* (3.27) *is also referred to as Kolmogorov partial differential equation (PDE) or Kolmogorov backward equation in the literature (cf. Hairer [36]). The latter name origins from the fact that one can rewrite the Kolmogorov equation backwards in time. Let* f *be a solution to the Kolmogorov equation* (3.27) *and let* $g\in\mathscr{C}^{1,2}([0,T]\times\mathbb{R}^d,\mathbb{R})$ *be the function defined by*

$$g(t,x) = f(T-t,x) \tag{3.29}$$

for all $t\in[0,T]$, $x\in\mathbb{R}^d$. *Then by the chain rule it holds for every* $t\in[0,T]$, $x\in\mathbb{R}^d$ *that*

$$\begin{cases} \frac{\partial g}{\partial t}(t,x) = -\frac{1}{2}\operatorname{Trace}_{\mathbb{R}^d}\big(\sigma(x)[\sigma(x)]^*(\operatorname{Hess}_x g)(t,x)\big) - \big\langle \mu(x), (\nabla_x g)(t,x)\big\rangle_{\mathbb{R}^d} \\ g(T,x) = \varphi(x)\,. \end{cases} \tag{3.30}$$

Note that the backward formulation (3.30) *has a terminal condition opposed to the initial condition in* (3.27). *For time-dependent coefficient functions* μ *and* σ *the backward formulation would be the appropriate one, but in our time-homogeneous setting the two formulations are equivalent.*

Our goal is to approximately calculate the function $\mathbb{R}^d \ni x \mapsto f(T,x)\in\mathbb{R}$ on some subset of $\mathbb{R}^d$. To fix ideas we consider real numbers $a,b\in\mathbb{R}$ with $a\le b$ and we suppose that our goal is to approximately calculate the function

$$[a,b]^d \ni x \mapsto f(T,x)\in\mathbb{R}\,. \tag{3.31}$$

3.2 Feynman-Kac Formula

The next theorem provides the important connection between the solution of the Kolmogorov equation and the solution process to an associated stochastic differential equation.

Theorem 3.2.1 (Feynman-Kac Formula)**.** *Assume there exists an at most polynomially growing solution $f \in \mathscr{C}^{1,2}([0,T] \times \mathbb{R}^d, \mathbb{R})$ to the Kolmogorov equation (3.27) and for every $x \in \mathbb{R}^d$ let $S^x \colon [0,T] \times \Omega \to \mathbb{R}^d$ be a solution process to the SDE*

$$S_t^x = x + \int_0^t \mu(S_s^x)\, ds + \int_0^t \sigma(S_s^x)\, dB_s\,. \tag{3.32}$$

Then for every $x \in \mathbb{R}^d$ it holds that

$$f(T,x) = \mathbb{E}\big[f(0, S_T^x)\big] = \mathbb{E}\big[\varphi(S_T^x)\big]. \tag{3.33}$$

Proof. The proof will be based on the backward formulation of the Kolmogorov equation in Remark 3.1.5 and we define $g \in \mathscr{C}^{1,2}([0,T] \times \mathbb{R}^d, \mathbb{R})$ by

$$g(t,x) = f(T-t,x) \tag{3.34}$$

for all $t \in [0,T]$, $x \in \mathbb{R}^d$. The assumption that f and thus g is polynomially bounded implies that there exists $c \in [1,\infty)$ such that for every $x \in \mathbb{R}^d$ it holds that

$$\max_{0 \le t \le T} |g(t,x)| \le c\left(1 + \|x\|_{\mathbb{R}^d}^c\right). \tag{3.35}$$

Now we fix $x \in \mathbb{R}^d$ and for our convenience just write S instead of S^x for the solution process to the SDE (3.32). For every $m \in \mathbb{N}$ let

$$\mathcal{T}_m = \inf\big\{0 \le t \le T \colon \|S_t\|_{\mathbb{R}^d} \ge m\big\} \tag{3.36}$$

be a stopping time. Itô's formula (cf., for instance, Le Gall [27, Section 5.2]) on the process $g(t \wedge \mathcal{T}_m, S_{t \wedge \mathcal{T}_m})$ together with equation (3.30) and the assumption that f solves the Kolmogorov equation yields

$$\begin{aligned} g(T \wedge \mathcal{T}_m, S_{T \wedge \mathcal{T}_m}) &= g(0,S_0) + \int_0^{T \wedge \mathcal{T}_m} \sum_{i,j=0}^d \tfrac{\partial g}{\partial x_i}(s,S_s)\sigma_{ij}(S_s)\, dB_s^j + \int_0^{T \wedge \mathcal{T}_m} \tfrac{\partial g}{\partial t}(s,S_s)\, ds \\ &\quad + \int_0^{T \wedge \mathcal{T}_m} \sum_{i=1}^d \mu_i(S_s)\tfrac{\partial g}{\partial x_i}(s,S_s) + \tfrac{1}{2}\sum_{i,j,l=1}^d \sigma_{il}(S_s)\sigma_{jl}(S_s)\tfrac{\partial^2 g}{\partial x_i \partial x_j}(s,S_s)\, ds \\ &= g(0,S_0) + \sum_{i,j=0}^d \int_0^{T \wedge \mathcal{T}_m} \tfrac{\partial g}{\partial x_i}(s,S_s)\sigma_{ij}(S_s)\, dB_s^j\,. \end{aligned} \tag{3.37}$$

Due to the definition of the stopping time, the smoothness of g and the Lipschitz condition (3.3) on σ the second summand is a martingale and has vanishing expectation

$$\mathbb{E}\left[\sum_{i,j=0}^d \int_0^{T \wedge \mathcal{T}_m} \tfrac{\partial g}{\partial x_i}(s,S_s)\sigma_{ij}(S_s)\, dB_s^j\right] = 0 \tag{3.38}$$

(cf. Le Gall [27, Proposition 4.7 and Section 5.1.2]). Thus by taking expectations in (3.37) for every $m \in \mathbb{N}$ it holds that

$$\mathbb{E}\big[g(\mathcal{T}_m, S_{\mathcal{T}_m})\mathbb{1}_{\{\mathcal{T}_m \le T\}}\big] + \mathbb{E}\big[g(T,S_T)\mathbb{1}_{\{\mathcal{T}_m > T\}}\big] = g(0,S_0) = f(T,x)\,. \tag{3.39}$$

Lemma 3.1.3 assures that for every $p \in [2, \infty)$ there exists a real number $C \in (0, \infty)$ such that it holds that

$$\mathbb{E}\left[\max_{0\leq t\leq T} \|S_t\|_{\mathbb{R}^d}^p\right] \leq Ce^{CT}\left(1 + \|x\|_{\mathbb{R}^d}^p\right). \tag{3.40}$$

With (3.35) we can bound the first term of equation (3.39) in absolute value by

$$\mathbb{E}\big[|g(\mathcal{T}_m, S_{\mathcal{T}_m})|\mathbb{1}_{\{\mathcal{T}_m\leq T\}}\big] \leq c\,(1 + m^c)\mathbb{P}[\mathcal{T}_m \leq T] \tag{3.41}$$

and by Chebyshev's inequality it holds that

$$\begin{aligned}\mathbb{P}[\mathcal{T}_m \leq T] = \mathbb{P}\left[\max_{0\leq t\leq T} \|S_t\|_{\mathbb{R}^d} \geq m\right] &\leq m^{-2c}\,\mathbb{E}\left[\max_{0\leq t\leq T} \|S_t\|_{\mathbb{R}^d}^{2c}\right] \\ &\leq m^{-2c}\,Ce^{CT}\big(1 + \|x\|_{\mathbb{R}^d}^{2c}\big).\end{aligned} \tag{3.42}$$

We conclude that the first term in (3.39) converges to zero as m tends to infinity. For the second term we observe that for every $m \in \mathbb{N}$, $\omega \in \Omega$ it holds that

$$|g(T, S_T)|\mathbb{1}_{\{\mathcal{T}_m>T\}} \leq c\left(1 + \|S_T\|_{\mathbb{R}^d}^c\right) \tag{3.43}$$

and due to (3.39) & (3.40) the dominated convergence theorem (cf. Athreya [5, Theorem 2.3.11]) assures that

$$\mathbb{E}\big[f(0, S_T)\big] = \mathbb{E}\Big[\lim_{m\to\infty} g(T, S_T)\mathbb{1}_{\{\mathcal{T}_m>T\}}\Big] = \lim_{m\to\infty} \mathbb{E}\big[g(T, S_T)\mathbb{1}_{\{\mathcal{T}_m>T\}}\big] = f(T, x). \tag{3.44}$$

The assumption that for every $x \in \mathbb{R}^d$ it holds that $\varphi(x) = f(0, x)$ concludes the proof. □

Remark 3.2.2 (Other Versions of the Feynman-Kac Formula)**.** *There are various other and more general formulations of the Feynman-Kac formula (cf., for instance, Øksendal [63, Chapter 8], Schilling [75, Theorem 8.6] and Durrett [24, Section 8.3]). Under further assumptions on φ, σ and μ one does not need to assume a priori that there exists a solution to the Kolmogorov equation but the function $f\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ given by*

$$f(t, x) = \mathbb{E}\big[\varphi(S_t^x)\big] \tag{3.45}$$

for all $t \in [0, T]$, $x \in \mathbb{R}^d$ automatically represents a (unique) solution to the Kolmogorov equation (3.27) *(cf. Friedman [26, Theorem 5.6.1], Kloeden [54, Theorem 4.8.6] and Hairer [36, Corollary 4.17 and Remark 4.1]). Also note that this result is directly related to the fact that the solution of the SDE* (3.32) *is a Markov process w.r.t. the filtration $(\mathcal{G}_t)_{t\in[0,T]}$ with Feller semigroup (Q_t) defined for every $\varphi \in \mathscr{B}(\mathbb{R}^d, \mathbb{R})$ by*

$$Q_t\varphi(x) = \mathbb{E}\big[\varphi(S_t^x)\big] \tag{3.46}$$

and generator L defined (at least) for every $f \in \mathscr{C}_c^2(\mathbb{R}^d, \mathbb{R})$ by

$$Lf(x) = \tfrac{1}{2}\,\mathrm{Trace}_{\mathbb{R}^d}\big(\sigma(x)[\sigma(x)]^*(\mathrm{Hess}_x f)(t, x)\big) + \big\langle \mu(x), (\nabla_x f)(t, x)\big\rangle_{\mathbb{R}^d} \tag{3.47}$$

(cf., for instance, Le Gall [27, Theorem 8.6 and Theorem 8.7]).

3.3 Connection to the Learning Problem

Recall the Mathematical Learning Problem from Chapter 2 and the definition of the Kolmogorov equation (3.27). Using the Feynman-Kac representation (3.33) of a solution to the Kolmogorov equation $f(T,x)$ at time T we can formulate an equivalent Learning Problem.

Proposition 3.3.1 (Learning Problem for the Kolmogorov PDE)**.** *Let $f \in \mathscr{C}^{1,2}([0,T]\times\mathbb{R}^d,\mathbb{R})$ be an at most polynomially growing solution to the Kolmogorov equation, let $X\colon\Omega\to[a,b]^d$ be a (continuously) uniformly distributed $\mathcal{G}_0$-measurable random vector, which is independent of $\sigma(B_t, t\in[0,T])$, and let $S^X\colon[0,T]\times\Omega\to\mathbb{R}^d$ be a solution process to the SDE*

$$S_t^X = X + \int_0^t \mu\big(S_s^X\big)\,ds + \int_0^t \sigma\big(S_s^X\big)\,dB_s\,. \tag{3.48}$$

Define the data of the Learning Problem for the Kolmogorov equation by

$$Z = (X,Y) = \big(X, \varphi\big(S_T^X\big)\big) \tag{3.49}$$

and denote by $\hat{F}$ the regression function w.r.t. to (X,Y). Then it holds that

(i) $\hat{F}(x) = \mathbb{E}\big[\varphi(S_T^x)\big] = f(T,x)$ for a.e. $x\in[a,b]^d$ and

(ii) the function $[a,b]^d \ni x \mapsto f(T,x)\in\mathbb{R}$ is the unique minimizer of

$$\min_{F\in\mathscr{C}([a,b]^d,\mathbb{R})} \mathbb{E}\Big[\big|F(X) - \varphi\big(S_T^X\big)\big|^2\Big]\,. \tag{3.50}$$

Proof. First note that the uniformly distributed random vector X has bounded moments (cf. Zwillinger [80, Section 7.3.2]) and Lemma 3.1.3 therefore assures that for every $p\in[2,\infty)$ it holds that

$$\mathbb{E}\Big[\big\|S_T^X\big\|_{\mathbb{R}^d}^p\Big] < \infty\,. \tag{3.51}$$

The assumption on f assures that there exists a real number $c\in[1,\infty)$ such that for every $x\in\mathbb{R}^d$ it holds that

$$|\varphi(x)| = |f(0,x)| \le c\left(1 + \|x\|_{\mathbb{R}^d}^c\right). \tag{3.52}$$

Together with inequality (3.19) and (3.51) this shows that for every $p\in[2,\infty)$ it holds that

$$\mathbb{E}\Big[\big|\varphi\big(S_T^X\big)\big|^p\Big] \le \mathbb{E}\Big[c^p\Big(1+\big\|S_T^X\big\|_{\mathbb{R}^d}^c\Big)^p\Big] \le \mathbb{E}\Big[2^p c^p\Big(1+\big\|S_T^X\big\|_{\mathbb{R}^d}^{cp}\Big)\Big] < \infty\,. \tag{3.53}$$

This establishes that the random vector $Y = \varphi\big(S_T^X\big)$ has finite variance and according to Theorem 2.2.1 the corresponding Learning Problem has a $\mathbb{P}_X$-unique regression function $\hat{F}$ minimizing the least squares error. Note that $\mathbb{P}_X$ is just a multiple of the Lebesgue measure on $[a,b]^d$ and we claim that for a.e. $x\in[a,b]^d$ it holds that

$$\hat{F}(x) = \mathbb{E}\big[Y\big|X=x\big] = \mathbb{E}\big[\varphi\big(S_T^X\big)\big|X=x\big] = \mathbb{E}\big[\varphi(S_T^x)\big]\,. \tag{3.54}$$

The last equality can be made precise by the following argument. Define $\mathcal{B}\big(\mathscr{C}([0,T],\mathbb{R}^d)\big)$ to be the Borel σ-algebra on the Banach space $\big(\mathscr{C}([0,T],\mathbb{R}^d),\|\cdot\|_\infty\big)$ and Σ to the be its completion by all $\mathbb{P}_B$-null sets. Adopt from Le Gall [27, Theorem 8.5] that for every $x\in\mathbb{R}^d$ there exists a $\Sigma/\mathcal{B}(\mathbb{R}^d)$-measurable mapping $\Psi^x\colon\mathscr{C}([0,T],\mathbb{R}^d)\to\mathbb{R}^d$ such that it holds that

(i) $S_T^x = \Psi^x(B)$ $\mathbb{P}$-a.s.,

(ii) $S_T^X = \Psi^X(B)$ $\mathbb{P}$-a.s. and

(iii) for every $u \in \mathscr{C}([0,T],\mathbb{R}^d)$ the mapping $[a,b]^d \ni x \mapsto \Psi^x(u) \in \mathbb{R}^d$ is continuous.

Consequently Lemma 2.3.9 yields the claim (with $\Phi^x(u) = \varphi(\Psi^x(u))$ for every $x \in [a,b]^d$ and $u \in \mathscr{C}([0,T],\mathbb{R}^d)$). Using the Feynman-Kac formula it follows that for a.e. $x \in [a,b]$ it holds that

$$f(T,x) = \mathbb{E}\big[\varphi(S_T^x)\big] = \hat{F}(x) \tag{3.55}$$

and due to the assumption that $[a,b]^d \ni x \mapsto f(T,x) \in \mathbb{R}$ is continuous it must be the unique *continuous* function $F\colon [a,b]^d \to \mathbb{R}$ minimizing the least squares error

$$\mathcal{E}_{(X,Y)}(F) = \mathbb{E}\Big[\big|F(X) - \varphi\big(S_T^X\big)\big|^2\Big] = \frac{1}{(b-a)^d}\int_{[a,b]^d} \big|F(x) - \hat{F}(x)\big|^2\,dx + \mathcal{E}_{(X,Y)}\big(\hat{F}\big)\,. \tag{3.56}$$

This proves Proposition 3.3.1. □

That means that the continuous version of the regression function w.r.t. to the data $Z = \big(X, \varphi\big(S_T^X\big)\big)$ equals the solution of the Kolmogorov equation on $[a,b]^d$ at time T. We will demonstrate this connection with an easy example.

Example 3.3.2. *Let $d = 1$, $p \in \mathbb{N}$, $a, \bar{\sigma}, c_0, \dots, c_p, T \in \mathbb{R}$, $b \in [a,\infty)$, define the polynomial $\varphi\colon \mathbb{R} \to \mathbb{R}$ by $\varphi(x) = \sum_{j=0}^p c_j x^j$ for every $x \in \mathbb{R}$ and suppose we want to calculate a solution to the Kolmogorov equation*

$$\begin{cases} \frac{\partial f}{\partial t}(t,x) = \frac{1}{2}\bar{\sigma}^2\big(x^2\frac{\partial^2 f}{\partial x^2}(t,x) + x\frac{\partial f}{\partial x}(t,x)\big) \\ f(0,x) = \varphi(x) \end{cases} \tag{3.57}$$

at time T for $x \in [a,b]$. In our previous notation the functions $\mu\colon \mathbb{R} \to \mathbb{R}$ and $\sigma\colon \mathbb{R} \to \mathbb{R}$ are thus defined by

$$\mu(x) = \tfrac{1}{2}\bar{\sigma}^2 x \tag{3.58}$$

and

$$\sigma(x) = \bar{\sigma}x \tag{3.59}$$

for all $x \in \mathbb{R}$. Note that the functions φ, σ and μ are sufficiently smooth in order to employ a version of the Feynman-Kac formula cited in Remark 3.2.2 and to conclude that this Kolmogorov equation has a unique solution given by the Feynman-Kac representation in (3.45)*. Let us investigate the corresponding Learning Problem. Given a standard $(\Omega, \mathcal{G}, \mathbb{P}, (\mathcal{G}_t)_{t\in[0,T]})$-Brownian motion $B\colon [0,T]\times\Omega \to \mathbb{R}$ and a (continuously) uniformly distributed $\mathcal{G}_0$-measurable random variable $X\colon \Omega \to [a,b]$, which is independent of $\sigma(B_t, t \in [0,T])$, the associated SDE reads*

$$S_t^X = X + \int_0^t \tfrac{1}{2}\bar{\sigma}^2 S_s^X\,ds + \int_0^t \bar{\sigma}S_s^X\,dB_s\,. \tag{3.60}$$

By Itô's formula the stochastic process

$$\big(S_t^X\big)_{t\in[0,T]} = \big(Xe^{\bar{\sigma}B_t}\big)_{t\in[0,T]}\colon [0,T]\times\Omega \to \mathbb{R} \tag{3.61}$$

is the up to indistinguishability unique solution process to the SDE (3.60)*. According to Proposition 3.3.1 let us define*

$$Y = \varphi\big(S_T^X\big) = \varphi\big(Xe^{\bar{\sigma}B_T}\big) \tag{3.62}$$

and note that B_T is a centered normally distributed random variable with variance T. We observe that the data $Z = (X, Y)$ corresponds exactly to our Example 2.3.10 with $\varrho = \bar{\sigma}\sqrt{T}$ and $W = \frac{1}{\sqrt{T}} B_T$. Therefore equation (2.60) establishes that the unique continuous version of the regression function is given by

$$\hat{F}(x) = \sum_{j=0}^{p} c_j x^j e^{\frac{1}{2}(j\bar{\sigma})^2 T} \tag{3.63}$$

for all $x \in [a, b]$ and one can check that $\hat{F}(x)$ indeed equals the solution of the Kolmogorov equation $f(T, x)$ at time T.

In a general situation we can neither calculate the regression function nor a solution to the stochastic differential equation explicitly. While we can approximately solve the first problem by finding the empirical target function and balancing the bias-variance trade-off as treated in Sections 2.3 and 2.4, we will cope with the second problem in the next section.

3.4 Approximation by the Euler-Maruyama Scheme

Since in most cases we do not know an explicit solution to the stochastic differential equation and even cannot compute realizations of the random vectors S_T^X or S_T^x, we will approximate the solution of the stochastic differential equation by the Euler-Maruyama scheme (cf., for example, Kloeden [54] and Maruyama [58]). To motivate the definition let $X \colon \Omega \to \mathbb{R}^d$ be a $\mathcal{G}_0$-measurable random vector and let $S^X \colon [0, T] \times \Omega \to \mathbb{R}^d$ be a solution process to the SDE

$$S_t^X = X + \int_0^t \mu\big(S_s^X\big)\, ds + \int_0^t \sigma\big(S_s^X\big)\, dB_s\,. \tag{3.64}$$

Let $N \in \mathbb{N}$ and partition the time interval $[0, T]$ into N subintervals of equal length, i.e. for every $n \in \{0, 1, \ldots, N\}$ we set $t_n = \frac{nT}{N}$. Note that (3.64) implies that for every $n \in \{0, 1, \ldots, N-1\}$ it holds $\mathbb{P}$-a.s. that

$$S_{t_{n+1}}^X = S_{t_n}^X + \int_{t_n}^{t_{n+1}} \mu\big(S_s^X\big)\, ds + \int_{t_n}^{t_{n+1}} \sigma\big(S_s^X\big)\, dB_s\,. \tag{3.65}$$

This suggests that for sufficiently small step size $\frac{T}{N} = t_{n+1} - t_n$ it holds that

$$S_{t_{n+1}}^X \approx S_{t_n}^X + \mu\big(S_{t_n}^X\big)\, \tfrac{T}{N} + \sigma\big(S_{t_n}^X\big)\, \big(B_{t_{n+1}} - B_{t_n}\big)\,. \tag{3.66}$$

and we take (3.66) as a definition for the Euler-Maruyama scheme.

Definition 3.4.1 (Euler-Maruyama Scheme)**.** *Let $N \in \mathbb{N}$, let $X \colon \Omega \to \mathbb{R}^d$ be a $\mathcal{G}_0$-measurable random vector and let*

$$\mathcal{S}^{N,X} = \big(\mathcal{S}_n^{N,X}\big)_{n \in \{0,1,\ldots,N\}} \colon \{0, 1, \ldots, N\} \times \Omega \to \mathbb{R}^d \tag{3.67}$$

be the stochastic process which satisfies for every $n \in \{0, 1, \ldots, N-1\}$ that $\mathcal{S}_0^{N,X} = X$ and

$$\mathcal{S}_{n+1}^{N,X} = \mathcal{S}_n^{N,X} + \mu\big(\mathcal{S}_n^{N,X}\big)\, \tfrac{T}{N} + \sigma\big(\mathcal{S}_n^{N,X}\big)\, \Big(B_{\frac{(n+1)T}{N}} - B_{\frac{nT}{N}}\Big)\,. \tag{3.68}$$

Then we call $\mathcal{S}^{N,X}$ the Euler-Maruyama approximation for the SDE (3.64) with step size $\frac{T}{N}$.

Observe that (3.66) and (3.68) suggest that for every $n \in \{0, 1, \dots, N\}$ it holds that

$$S^X_{\frac{nT}{N}} = S^X_{t_n} \approx \mathcal{S}^X_n \,. \tag{3.69}$$

We want to analyze the quality of the approximation (3.69) more precisely, therefore we state a technical lemma on the moments (cf. also Kloeden [54, proof of Theorem 10.6.3]) and then a theorem on the strong convergence rate for the Euler-Maruyama scheme.

Lemma 3.4.2 (Bound on the Moments of the Euler-Maruyama Scheme)**.** *Let $p \in [2, \infty)$, let $X \colon \Omega \to \mathbb{R}^d$ be a $\mathcal{G}_0$-measurable random vector, which satisfies that*

$$\mathbb{E}\big[\|X\|^p_{\mathbb{R}^d}\big] < \infty \,, \tag{3.70}$$

let $S^X \colon [0,T] \times \Omega \to \mathbb{R}^d$ be a solution process to the SDE

$$S^X_t = X + \int_0^t \mu\big(S^X_s\big)\, ds + \int_0^t \sigma\big(S^X_s\big)\, dB_s \tag{3.71}$$

and for every $N \in \mathbb{N}$ let $\mathcal{S}^{N,X} \colon \{0, 1, \dots, N\} \times \Omega \to \mathbb{R}^d$ be the Euler-Maruyama approximation for the SDE (3.71) with step size $\frac{T}{N}$. Then there exists a real number $C \in (0, \infty)$ depending only on p, $\bar{K}$ and T such that it holds that

$$\sup_{M \in \mathbb{N}} \max_{n \in \{0,1,\dots,M\}} \mathbb{E}\Big[\big\|\mathcal{S}^{M,X}_n\big\|^p_{\mathbb{R}^d}\Big] \leq C\Big(1 + \mathbb{E}\big[\|X\|^p_{\mathbb{R}^d}\big]\Big) \,. \tag{3.72}$$

Proof. The proof proceeds in an analogous spirit to the proof of Lemma 3.1.3 and for our convenience we just write $\mathcal{S}^N$ instead of $\mathcal{S}^{N,X}$ for the Euler-Maruyama approximation with step size $\frac{T}{N}$. We define for every $N \in \mathbb{N}$ the constant extension

$$\bar{\mathcal{S}}^N = \big(\bar{\mathcal{S}}^N_t\big)_{t \in [0,T]} \colon [0,T] \times \Omega \to \mathbb{R}^d \tag{3.73}$$

of the Euler-Maruyama approximation to $[0,T]$ which for every $n \in \{0, 1, \dots, N-1\}$ and $t \in \big[\frac{nT}{N}, \frac{(n+1)T}{N}\big)$ is given by

$$\bar{\mathcal{S}}^N_t = \mathcal{S}^N_n \tag{3.74}$$

and

$$\bar{\mathcal{S}}^N_T = \mathcal{S}^N_N \,. \tag{3.75}$$

Then for every $N \in \mathbb{N}$, $n \in \{0, 1, \dots, N\}$ it holds that

$$\bar{\mathcal{S}}^N_{\frac{nT}{N}} = \mathcal{S}^N_n = X + \int_0^{\frac{nT}{N}} \mu\big(\bar{\mathcal{S}}^N_s\big)\, ds + \int_0^{\frac{nT}{N}} \sigma\big(\bar{\mathcal{S}}^N_s\big)\, dB_s \,. \tag{3.76}$$

For every $m \in \mathbb{N}$ let

$$\mathcal{T}_m = \inf\big\{0 \leq t \leq T \colon \|B_t\|_{\mathbb{R}^d} \geq m\big\} \tag{3.77}$$

be a stopping time, which assures that the upcoming expectations are well-defined. Let $m, N \in \mathbb{N}$, $\tau \in [0,T]$, then equation (3.76) and inequality (3.19) establish that

$$\begin{aligned} \mathbb{E}\Big[\big\|\bar{\mathcal{S}}^N_{\tau \wedge \mathcal{T}_m}\big\|^p_{\mathbb{R}^d}\Big] \leq 3^{p-1}\Bigg(\mathbb{E}\big[\|X\|^p_{\mathbb{R}^d}\big] &+ \underbrace{\mathbb{E}\left[\max_{0 \leq t \leq \tau}\left\|\int_0^{t \wedge \mathcal{T}_m} \mu\big(\bar{\mathcal{S}}^N_s\big)\, ds\right\|^p_{\mathbb{R}^d}\right]}_{\mathbf{A}} \\ &+ \underbrace{\mathbb{E}\left[\max_{0 \leq t \leq \tau}\left\|\int_0^{t \wedge \mathcal{T}_m} \sigma\big(\bar{\mathcal{S}}^N_s\big)\, dB_s\right\|^p_{\mathbb{R}^d}\right]}_{\mathbf{B}}\Bigg) \,, \end{aligned} \tag{3.78}$$

applying Lemma 3.1.1 and the growth estimate (3.20) to first integral in (3.78) yields

$$\begin{aligned}\mathbf{A} &\leq \tau^{p-1} \int_0^\tau \mathbb{E}\Big[\big\|\mu\big(\bar{\mathcal{S}}^N_{s\wedge\mathcal{T}_m}\big)\big\|^p_{\mathbb{R}^d}\Big]\, ds \\ &\leq (2\bar{K})^p\, T^{p-1} \left(T + \int_0^\tau \mathbb{E}\Big[\big\|\bar{\mathcal{S}}^N_{s\wedge\mathcal{T}_m}\big\|^p_{\mathbb{R}^d}\Big]\, ds\right)\end{aligned} \tag{3.79}$$

and for the stochastic integral we obtain analogously

$$\begin{aligned}\mathbf{B} &\leq p^p \tau^{\frac{p}{2}-1} \int_0^\tau \mathbb{E}\Big[\big\|\sigma\big(\bar{\mathcal{S}}^N_{s\wedge\mathcal{T}_m}\big)\big\|^p_{\mathbb{R}^{d\times d}}\Big]\, ds \\ &\leq (2\bar{K}p)^p\, T^{\frac{p}{2}-1} \left(T + \int_0^\tau \mathbb{E}\Big[\big\|\bar{\mathcal{S}}^N_{s\wedge\mathcal{T}_m}\big\|^p_{\mathbb{R}^d}\Big]\, ds\right).\end{aligned} \tag{3.80}$$

In view of (3.78), (3.79) and (3.80) there exists a real number $\tilde{C} \in (0,\infty)$ depending only on p, $\bar{K}$ and T such that for every $m, N \in \mathbb{N}$, $\tau \in [0,T]$ it holds that

$$\mathbb{E}\left[\big\|\bar{\mathcal{S}}^N_{\tau\wedge\mathcal{T}_m}\big\|^p_{\mathbb{R}^d}\right] \leq \tilde{C}\left(1 + \mathbb{E}\big[\|X\|^p_{\mathbb{R}^d}\big] + \int_0^\tau \mathbb{E}\Big[\big\|\bar{\mathcal{S}}^N_{s\wedge\mathcal{T}_m}\big\|^p_{\mathbb{R}^d}\Big]\, ds\right). \tag{3.81}$$

Now we can apply Grönwall's lemma and the monotone convergence theorem to show that for all $N \in \mathbb{N}$, $\tau \in [0,T]$ it holds that

$$\mathbb{E}\Big[\big\|\bar{\mathcal{S}}^N_\tau\big\|^p_{\mathbb{R}^d}\Big] \leq \tilde{C} e^{\tilde{C}\tau}\Big(1 + \mathbb{E}\big[\|X\|^p_{\mathbb{R}^d}\big]\Big). \tag{3.82}$$

The definition of the constant extension of the Euler-Maruyama scheme therefore establishes that it holds that

$$\sup_{M\in\mathbb{N}} \max_{n\in\{0,1,\ldots,M\}} \mathbb{E}\Big[\big\|\mathcal{S}^M_n\big\|^p_{\mathbb{R}^d}\Big] = \sup_{M\in\mathbb{N}} \sup_{\tau\in[0,T]} \mathbb{E}\Big[\big\|\bar{\mathcal{S}}^M_\tau\big\|^p_{\mathbb{R}^d}\Big] \leq \tilde{C} e^{\tilde{C}T}\Big(1 + \mathbb{E}\big[\|X\|^p_{\mathbb{R}^d}\big]\Big) \tag{3.83}$$

and thus proves the lemma. □

Let us continue with a theorem on the strong convergence rate for the Euler-Maruyama scheme (cf. also Kloeden [54, Theorem 10.2.2], Milstein [59], Hofmann [42] and Müller-Gronbach [61]).

Theorem 3.4.3 (Strong Convergence Rate for the Euler-Maruyama Scheme)**.** *Let $p \in [2,\infty)$, let $X\colon \Omega \to \mathbb{R}^d$ be a $\mathcal{G}_0$-measurable random vector, which satisfies that*

$$\mathbb{E}\big[\|X\|^p_{\mathbb{R}^d}\big] < \infty, \tag{3.84}$$

let $S^X\colon [0,T]\times\Omega \to \mathbb{R}^d$ be a solution process to the SDE

$$S^X_t = X + \int_0^t \mu\big(S^X_s\big)\, ds + \int_0^t \sigma\big(S^X_s\big)\, dB_s \tag{3.85}$$

and for every $N \in \mathbb{N}$ let

$$\mathcal{S}^{N,X}\colon \{0,1,\ldots,N\}\times\Omega \to \mathbb{R}^d \tag{3.86}$$

be the Euler-Maruyama approximations for the SDE (3.85) with step size $\frac{T}{N}$. Then there exists a real number $C \in (0,\infty)$ depending only on p, K, $\bar{K}$ and T such that for every $N \in \mathbb{N}$ it holds that

$$\max_{n\in\{0,1,\ldots,N\}} \Big\|S^X_{\frac{nT}{N}} - \mathcal{S}^{N,X}_n\Big\|_{L^p(\mathbb{P};\|\cdot\|_{\mathbb{R}^d})} \leq \frac{C}{\sqrt{N}}\Big(1 + \|X\|_{L^p(\mathbb{P};\|\cdot\|_{\mathbb{R}^d})}\Big) \tag{3.87}$$

Proof. Again we just write S instead of S^X for the solution process to the SDE (3.85), $\mathcal{S}^N$ instead of $\mathcal{S}^{N,X}$ for the Euler-Maruyama approximation with step size $\frac{T}{N}$ and for every $N \in \mathbb{N}$ we define the constant extension

$$\bar{\mathcal{S}}^N = \big(\bar{\mathcal{S}}^N_t\big)_{t\in[0,T]} \colon [0,T] \times \Omega \to \mathbb{R}^d \tag{3.88}$$

of the Euler-Maruyama approximation to $[0,T]$ according to the proof of Lemma 3.4.2. Then it holds for every $n \in \{0,1,\dots,N\}$ that

$$\mathcal{S}^N_n = X + \int_0^{\frac{nT}{N}} \mu(\bar{\mathcal{S}}^N_s)\, ds + \int_0^{\frac{nT}{N}} \sigma(\bar{\mathcal{S}}^N_s)\, dB_s\,. \tag{3.89}$$

For every $N \in \mathbb{N}$, $n \in \{0,1,\dots,N-1\}$, $\frac{nT}{N} \le \tau < \frac{(n+1)T}{N}$, equations (3.85) & (3.89) and inequality (3.19) yield

$$\begin{aligned}
&\mathbb{E}\Big[\big\|S_\tau - \bar{\mathcal{S}}^N_\tau\big\|^p_{\mathbb{R}^d}\Big] = \mathbb{E}\left[\left\|\Big(S_\tau - S_{\frac{nT}{N}}\Big) + \Big(S_{\frac{nT}{N}} - \mathcal{S}^N_n\Big)\right\|^p_{\mathbb{R}^d}\right] \\
&\le 4^{p-1}\Bigg(\underbrace{\mathbb{E}\left[\left\|\int_{\frac{nT}{N}}^\tau \mu\big(S_s\big)\, ds\right\|^p_{\mathbb{R}^d}\right]}_{\mathbf{A}} + \underbrace{\mathbb{E}\left[\left\|\int_{\frac{nT}{N}}^\tau \sigma\big(S_s\big)\, dB_s\right\|^p_{\mathbb{R}^d}\right]}_{\mathbf{B}} \\
&+ \underbrace{\mathbb{E}\left[\left\|\int_0^{\frac{nT}{N}} \mu\big(S_s\big) - \mu\big(\bar{\mathcal{S}}^N_s\big)\, ds\right\|^p_{\mathbb{R}^d}\right]}_{\mathbf{C}} + \underbrace{\mathbb{E}\left[\left\|\int_0^{\frac{nT}{N}} \sigma\big(S_s\big) - \sigma\big(\bar{\mathcal{S}}^N_s\big)\, dB_s\right\|^p_{\mathbb{R}^d}\right]}_{\mathbf{D}}\Bigg)\,,
\end{aligned} \tag{3.90}$$

using Lemma 3.1.1 and the growth estimate (3.20) we can bound the first two terms by

$$\begin{aligned}
\mathbf{A} &\le \Big(t - \tfrac{nT}{N}\Big)^{p-1} \int_{\frac{nT}{N}}^\tau \mathbb{E}\Big[\big\|\mu\big(S_s\big)\big\|^p_{\mathbb{R}^d}\Big]\, ds \\
&\le \Big(2\bar{K}\tfrac{T}{N}\Big)^p \left(1 + \mathbb{E}\left[\max_{0\le t\le T} \big\|S_t\big\|^p_{\mathbb{R}^d}\right]\right)
\end{aligned} \tag{3.91}$$

and

$$\begin{aligned}
\mathbf{B} &\le \Big(t - \tfrac{nT}{N}\Big)^{\frac{p}{2}-1} p^p \int_{\frac{nT}{N}}^\tau \mathbb{E}\Big[\big\|\sigma\big(S_s\big)\big\|^p_{\mathbb{R}^{d\times d}}\Big]\, ds \\
&\le \Big(\tfrac{T}{N}\Big)^{\frac{p}{2}} (2\bar{K}p)^p \left(1 + \mathbb{E}\left[\max_{0\le t\le T} \big\|S_t\big\|^p_{\mathbb{R}^d}\right]\right)
\end{aligned} \tag{3.92}$$

and using Lemma 3.1.1 and the Lipschitz assumption (3.3) we can bound the last two terms by

$$\begin{aligned}
\mathbf{C} &\le \Big(\tfrac{nT}{N}\Big)^{p-1} \int_0^{\frac{nT}{N}} \mathbb{E}\Big[\big\|\mu\big(S_s\big) - \mu\big(\bar{\mathcal{S}}^N_s\big)\big\|^p_{\mathbb{R}^d}\Big]\, ds \\
&\le T^{p-1}K^p \int_0^\tau \mathbb{E}\Big[\big\|S_s - \bar{\mathcal{S}}^N_s\big\|^p_{\mathbb{R}^d}\Big]\, ds
\end{aligned} \tag{3.93}$$

and

$$\begin{aligned}
\mathbf{D} &\le \Big(\tfrac{nT}{N}\Big)^{\frac{p}{2}-1} p^p \int_0^{\frac{nT}{N}} \mathbb{E}\Big[\big\|\sigma\big(S_s\big) - \sigma\big(\bar{\mathcal{S}}^N_s\big)\big\|^p_{\mathbb{R}^d}\Big]\, ds \\
&\le T^{\frac{p}{2}-1}(Kp)^p \int_0^\tau \mathbb{E}\Big[\big\|S_s - \bar{\mathcal{S}}^N_s\big\|^p_{\mathbb{R}^d}\Big]\, ds\,.
\end{aligned} \tag{3.94}$$

In view of (3.90), (3.91), (3.92), (3.93), (3.94) and Lemma 3.1.3 there exists a real number $\tilde{C} \in (0,\infty)$ depending only on p, K, $\bar{K}$ and T such that for every $N \in \mathbb{N}$, $\tau \in [0,T]$ it holds that

$$\mathbb{E}\Big[\big\|S_\tau - \bar{\mathcal{S}}_\tau^N\big\|_{\mathbb{R}^d}^p\Big] \leq \tilde{C}N^{-\frac{p}{2}}\Big(1 + \mathbb{E}\big[\|X\|_{\mathbb{R}^d}^p\big]\Big) + \tilde{C}\int_0^\tau \mathbb{E}\Big[\big\|S_s - \bar{\mathcal{S}}_s^N\big\|_{\mathbb{R}^d}^p\Big]\,ds\,. \tag{3.95}$$

Finally we can apply Grönwall's lemma and the definition of the constant extension of the Euler-Maruyama scheme to show that for every $N \in \mathbb{N}$ it holds that

$$\begin{aligned}&\max_{n\in\{0,1,\ldots,N\}} \mathbb{E}\left[\left\|S_{\frac{nT}{N}} - \mathcal{S}_n^N\right\|_{\mathbb{R}^d}^p\right] \leq \sup_{\tau\in[0,T]} \mathbb{E}\Big[\big\|S_\tau - \bar{\mathcal{S}}_\tau^N\big\|_{\mathbb{R}^d}^p\Big]\\ &\leq \sup_{\tau\in[0,T]} \tilde{C}N^{-\frac{p}{2}}\Big(1 + \mathbb{E}\big[\|X\|_{\mathbb{R}^d}^p\big]\Big)e^{\tilde{C}\tau} \leq \tilde{C}N^{-\frac{p}{2}}\Big(1 + \mathbb{E}\big[\|X\|_{\mathbb{R}^d}^p\big]\Big)e^{\tilde{C}T}\end{aligned} \tag{3.96}$$

and this proves the theorem. □

For the sake of completeness we also mention the numerically weak convergence rate for the Euler-Maruyama scheme.

Remark 3.4.4 (Numerically Weak Convergence Rate for the Euler-Maruyama Scheme)**.** *Under additional assumptions on X, φ, σ and μ there exists a real number $C \in (0,\infty)$ such that for every $N \in \mathbb{N}$ it holds that*

$$\max_{n\in\{0,1,\ldots,N\}} \Big|\mathbb{E}\Big[\varphi\Big(S_{\frac{nT}{N}}^X\Big)\Big] - \mathbb{E}\Big[\varphi\Big(\mathcal{S}_n^{N,X}\Big)\Big]\Big| \leq \frac{C}{N} \tag{3.97}$$

(cf. Kloeden [54, Theorem 14.5.1]).

Instead of the data $Z = (X,Y) = \big(X, \varphi\big(S_T^X\big)\big)$ we now consider the Learning Problem w.r.t. the approximated data $\mathcal{Z}^N = \big(X, \mathcal{Y}^N\big) = \big(X, \varphi\big(\mathcal{S}_N^{N,X}\big)\big)$ for sufficient large N. With Theorem 3.4.3 we can estimate the error we make by using the approximated data.

Proposition 3.4.5 (Approximated Learning Problem for the Kolmogorov PDE)**.** *Let $f \in \mathscr{C}^{1,2}([0,T]\times\mathbb{R}^d,\mathbb{R})$ be an at most polynomially growing solution to the Kolmogorov equation and let $X\colon \Omega \to [a,b]^d$ be a (continuously) uniformly distributed $\mathcal{G}_0$-measurable random vector, which is independent of $\sigma(B_t, t\in[0,T])$. For every $N \in \mathbb{N}$ let $\mathcal{S}^{N,X}\colon \{0,1,\ldots,N\}\times\Omega \to \mathbb{R}^d$ be the Euler-Maruyama approximation for the SDE* (3.48) *with step size $\frac{T}{N}$, define the data of the (with step size $\frac{T}{N}$) approximated Learning Problem for the Kolmogorov equation by*

$$\mathcal{Z}^N = \big(X, \mathcal{Y}^N\big) = \big(X, \varphi\big(\mathcal{S}_N^{N,X}\big)\big) \tag{3.98}$$

and denote by $\hat{\mathcal{F}}^N$ the regression function w.r.t. to $\big(X,\mathcal{Y}^N\big)$. Then there exists a real number $C \in (0,\infty)$ such that for every $N \in \mathbb{N}$ it holds that

(i) $\hat{\mathcal{F}}^N(x) = \mathbb{E}\big[\varphi\big(\mathcal{S}_N^{N,x}\big)\big]$ *for a.e. $x \in [a,b]^d$,*

(ii) $\sup_{x\in[a,b]^d}\big|\mathbb{E}\big[\varphi\big(\mathcal{S}_N^{N,x}\big)\big] - f(T,x)\big| \leq \frac{C}{\sqrt{N}}$ *and*

(iii) the function $[a,b]^d \ni x \mapsto \mathbb{E}\big[\varphi\big(\mathcal{S}_N^{N,x}\big)\big] \in \mathbb{R}$ is the unique minimizer of

$$\min_{F\in\mathscr{C}([a,b]^d,\mathbb{R})} \mathbb{E}\Big[\big|F(X) - \varphi\big(\mathcal{S}_N^{N,X}\big)\big|^2\Big]\,. \tag{3.99}$$

Proof. Note that Lemma 3.4.2 implies by the same argument as in equation (3.52) & (3.53) that for every $p \in [2, \infty)$ it holds that

$$\sup_{M\in\mathbb{N}} \left\|\mathcal{Y}^M\right\|_{L^p(\mathbb{P};|\cdot|)} = \sup_{M\in\mathbb{N}} E\Big[\big|\varphi\big(\mathcal{S}_M^{M,X}\big)\big|^p\Big]^{\frac{1}{p}} < \infty\,. \tag{3.100}$$

This shows that for every $N \in \mathbb{N}$ the output variable $\mathcal{Y}^N$ has finite variance and therefore the regression function $\hat{\mathcal{F}}^N$ is well-defined. Now we will prove the representation of the regression function in item (i). For every $x \in [a,b]^d$ let

$$\big(\Psi_n^{N,x}\big)_{n\in\{0,1,\ldots,N\}} \colon \{0,1,\ldots,n\} \times \mathscr{C}([0,T],\mathbb{R}^d) \to \mathbb{R}^d\,, \quad N \in \mathbb{N}\,, \tag{3.101}$$

be the mappings, which satisfy for every $N \in \mathbb{N}$, $n \in \{0,1,\ldots,N-1\}$, $u \in \mathscr{C}([0,T],\mathbb{R}^d)$ that $\Psi_0^{N,x}(u) = x$ and

$$\Psi_{n+1}^{N,x}(u) = \Psi_n^{N,x}(u) + \mu\big(\Psi_n^{N,x}(u)\big)\tfrac{T}{N} + \sigma\big(\Psi_n^{N,x}(u)\big)\Big(u\big(\tfrac{(n+1)T}{N}\big) - u\big(\tfrac{nT}{N}\big)\Big)\,. \tag{3.102}$$

Observe that for every $N \in \mathbb{N}$ it holds that

(i) $\Psi_N^{N,x}(B) = \mathcal{S}_N^{N,x}$,

(ii) $\Psi_N^{N,X}(B) = \mathcal{S}_N^{N,X}$ and

(iii) for every $u \in \mathscr{C}([0,T],\mathbb{R}^d)$ the mapping $[a,b]^d \ni x \mapsto \Psi_N^{N,x}(u) \in \mathbb{R}^d$ is continuous.

Furthermore note the fact that the Borel σ-algebra on $\mathscr{C}([0,T],\mathbb{R}^d)$ is generated by the evaluation functionals

$$\mathscr{C}([0,T],\mathbb{R}^d) \ni u \mapsto \mathrm{eval}_t(u) = u(t) \in \mathbb{R}^d \tag{3.103}$$

for every $t \in [0,T]$, i.e.

$$\mathcal{B}\big(\mathscr{C}([0,T],\mathbb{R}^d)\big) = \sigma(\mathrm{eval}_t, t \in [0,T]) \tag{3.104}$$

(cf. Aliprantis [2, Lemma 4.53]). This assures that for every $N \in \mathbb{N}$, $x \in [a,b]^d$ the mapping

$$\mathscr{C}([0,T],\mathbb{R}^d) \ni u \mapsto \Psi_N^{N,x}(u) \in \mathbb{R}^d \tag{3.105}$$

is $\mathcal{B}\big(\mathscr{C}([0,T],\mathbb{R}^d)\big)/\mathcal{B}(\mathbb{R}^d)$-measurable. Thus for every $N \in \mathbb{N}$ Lemma 2.3.9 (with $\Phi^x(u) = \varphi\big(\Psi_N^{N,x}(u)\big)$ for every $x \in [a,b]^d$ and $u \in \mathscr{C}([0,T],\mathbb{R}^d)$) and Theorem 2.2.1 yields that it holds that

$$\hat{\mathcal{F}}^N(x) = \mathbb{E}\big[\mathcal{Y}^N\big|X = x\big] = \mathbb{E}\big[\varphi\big(\mathcal{S}_N^{N,X}\big)\big|X = x\big] = \mathbb{E}\big[\varphi\big(\mathcal{S}_N^{N,x}\big)\big] \tag{3.106}$$

for a.e. $x \in [a,b]^d$. Observe that similar to equation (3.100) we can also deduce the fact that

$$\sup_{M\in\mathbb{N}} \sup_{x\in[a,b]^d} \mathbb{E}\Big[\big|\varphi\big(\mathcal{S}_M^{M,x}\big)\big|^p\Big]^{\frac{1}{p}} < \infty\,. \tag{3.107}$$

The continuity claim in item (iii) above and the continuity of φ imply that for all $N \in \mathbb{N}$, $\omega \in \Omega$ the function $[a,b]^d \ni x \mapsto \varphi\big(\mathcal{S}_N^{N,x}(\omega)\big) \in \mathbb{R}$ is continuous. In view of the uniform integrability convergence theorem (cf. Athreya [5, Theorem 2.5.10]) and (3.107) this demonstrates that for every $N \in \mathbb{N}$ the mapping

$$[a,b]^d \ni x \mapsto \mathbb{E}\big[\varphi\big(\mathcal{S}_N^{N,x}\big)\big] \in \mathbb{R} \tag{3.108}$$

is the unique continuous version of the regression function $\hat{\mathcal{F}}^N$ and therefore proves item (iii). For the proof of item (ii) let $S^x \colon [0,T] \times \Omega \to \mathbb{R}^d$ a be solution process to the SDE (3.48)

for every $x \in [a,b]^d$. Then the Feynman-Kac formula establishes that $f(T,x) = \mathbb{E}\big[\varphi(S_T^x)\big]$ for every $x \in \mathbb{R}^d$ and we observe that

$$\begin{aligned} &\sup_{x\in[a,b]^d} \big|\mathbb{E}\big[\varphi\big(\mathcal{S}_N^{N,x}\big)\big] - f(T,x)\big| = \sup_{x\in[a,b]^d} \big|\mathbb{E}\big[\varphi\big(\mathcal{S}_N^{N,x}\big)\big] - \mathbb{E}\big[\varphi\big(S_T^x\big)\big]\big| \\ &\leq \frac{1}{\sqrt{N}} \left(\sup_{M\in\mathbb{N}} \sup_{x\in[a,b]^d} \sqrt{M}\, \big|\mathbb{E}\big[\varphi\big(\mathcal{S}_M^{M,x}\big) - \varphi\big(S_T^x\big)\big]\big| \right). \end{aligned} \tag{3.109}$$

Next the fundamental theorem of calculus, the Cauchy-Schwarz inequality (cf., for instance, Cannarsa [17, Theorem 7.34 & Proposition 5.3]) and the assumption that φ has an at most polynomially growing gradient (3.4) yield that there exists a real number $c \in [1,\infty)$ such that for every $u, v \in \mathbb{R}^d$ it holds that

$$\begin{aligned} \big|\varphi(u) - \varphi(v)\big| &= \left| \int_0^1 \big\langle \nabla\varphi(v + s(u-v)), u - v \big\rangle_{\mathbb{R}^d} \, ds \right| \\ &\leq \|u - v\|_{\mathbb{R}^d} \sup_{s\in[0,1]} \|\nabla\varphi(v + s(u-v))\|_{\mathbb{R}^d} \\ &\leq c\, \|u - v\|_{\mathbb{R}^d} \big(1 + \|u\|_{\mathbb{R}^d} + \|v\|_{\mathbb{R}^d}\big)^c \end{aligned} \tag{3.110}$$

Eventually equation (3.110), the Cauchy-Schwarz inequality, Lemma 3.1.3 & 3.4.2 and the strong convergence rate of the Euler-Maruyama scheme (Theorem 3.4.3) assure that there exists a real number $C \in (0,\infty)$ such that we can bound the last term in equation (3.109) by

$$\begin{aligned} &\sup_{M\in\mathbb{N}} \sup_{x\in[a,b]^d} \sqrt{M}\, \big|\mathbb{E}\big[\varphi\big(\mathcal{S}_M^{M,x}\big) - \varphi\big(S_T^x\big)\big]\big| \\ &\leq \sup_{M\in\mathbb{N}} \sup_{x\in[a,b]^d} \sqrt{M}\, c\, \mathbb{E}\Big[\big\|\mathcal{S}_M^{M,x} - S_T^x\big\|_{\mathbb{R}^d} \big(1 + \big\|\mathcal{S}_M^{M,x}\big\|_{\mathbb{R}^d} + \big\|S_T^x\big\|_{\mathbb{R}^d}\big)^c \Big] \\ &\leq \sup_{M\in\mathbb{N}} \sup_{x\in[a,b]^d} \sqrt{M}\, c\, \mathbb{E}\Big[\big\|\mathcal{S}_M^{M,x} - S_T^x\big\|_{\mathbb{R}^d}^2 \Big]^{\frac{1}{2}} \mathbb{E}\Big[\big(1 + \big\|\mathcal{S}_M^{M,x}\big\|_{\mathbb{R}^d} + \big\|S_T^x\big\|_{\mathbb{R}^d}\big)^{2c} \Big]^{\frac{1}{2}} \\ &\leq c\, C \sup_{M\in\mathbb{N}} \sup_{x\in[a,b]^d} \sqrt{M}\, \mathbb{E}\Big[\big\|\mathcal{S}_M^{M,x} - S_T^x\big\|_{\mathbb{R}^d}^2 \Big]^{\frac{1}{2}} < \infty. \end{aligned} \tag{3.111}$$

This proves Proposition 3.4.5. □

Recall that in general we do not know the distribution of the (exact) data $Z = \big(X, \varphi\big(S_T^X\big)\big)$ of the Learning Problem for the Kolmogorov equation (Proposition 3.3.1) and even cannot compute realizations of the latter. But according to Proposition 3.4.5 for sufficiently large N the regression function $\hat{\mathcal{F}}^N$ w.r.t. the approximated data $\mathcal{Z}^N = \big(X, \varphi\big(\mathcal{S}_N^{N,X}\big)\big)$ is a sensible approximation to the solution of the Kolmogorov equation at time T and it is straightforward to obtain realizations of samples drawn from the distribution of $\mathcal{Z}^N$.

Example 3.4.6. *Let us continue with Example 3.3.2 and suppose we do not know the explicit solution to the stochastic differential equation. Let us partition the time interval* $[0,T]$ *into* $N \in \mathbb{N}$ *subintervals of equal length, then the Euler-Maruyama approximation of the SDE* (3.60) *with step size* $\frac{T}{N}$ *is defined as the stochastic process* $\mathcal{S}^{N,X} \colon \{0,1,\dots,N\} \times \Omega \to \mathbb{R}$ *which satisfies for every* $n \in \{0,1,\dots,N-1\}$ *that* $\mathcal{S}_0^{N,X} = X$ *and*

$$\mathcal{S}_{n+1}^{N,X} = \mathcal{S}_n^{N,X} + \tfrac{\bar{\sigma}^2 T}{2N} \mathcal{S}_n^{N,X} + \bar{\sigma} \mathcal{S}_n^{N,X} \Big(B_{\frac{(n+1)T}{N}} - B_{\frac{nT}{N}} \Big). \tag{3.112}$$

Based on Proposition 3.4.5 we can approximate the data for the Learning Problem by $\mathcal{Z}^N = \big(X, \varphi\big(\mathcal{S}_N^{N,X}\big)\big)$. *Observe that the increments of the Brownian motion*

$$B_{\frac{(n+1)T}{N}} - B_{\frac{nT}{N}} \tag{3.113}$$

are just centered normally distributed random variables with variance $\frac{T}{N}$ *independent of*

$$\sigma\big(B_t,\, 0 \le t \le \tfrac{nT}{N}\big)\,. \tag{3.114}$$

So in order to obtain a realization of a sample drawn from the distribution of the data $\mathcal{Z}^N$ *we*

(i) take a realization of the in $[a, b]$ *uniformly distributed random variable* X,

(ii) iteratively compute the realization of the Euler-Maruyama approximation for every step $n \in \{0, 1, \dots, N\}$ *by using realizations of independent centered normally distributed random variables with variance* $\frac{T}{N}$ *and*

(iii) evaluate $\varphi(x) = \sum_{j=0}^{s} c_j x^j$ *for the realization of the Euler-Maruyama approximation at the* N*-th step*

(see figure 3.1).

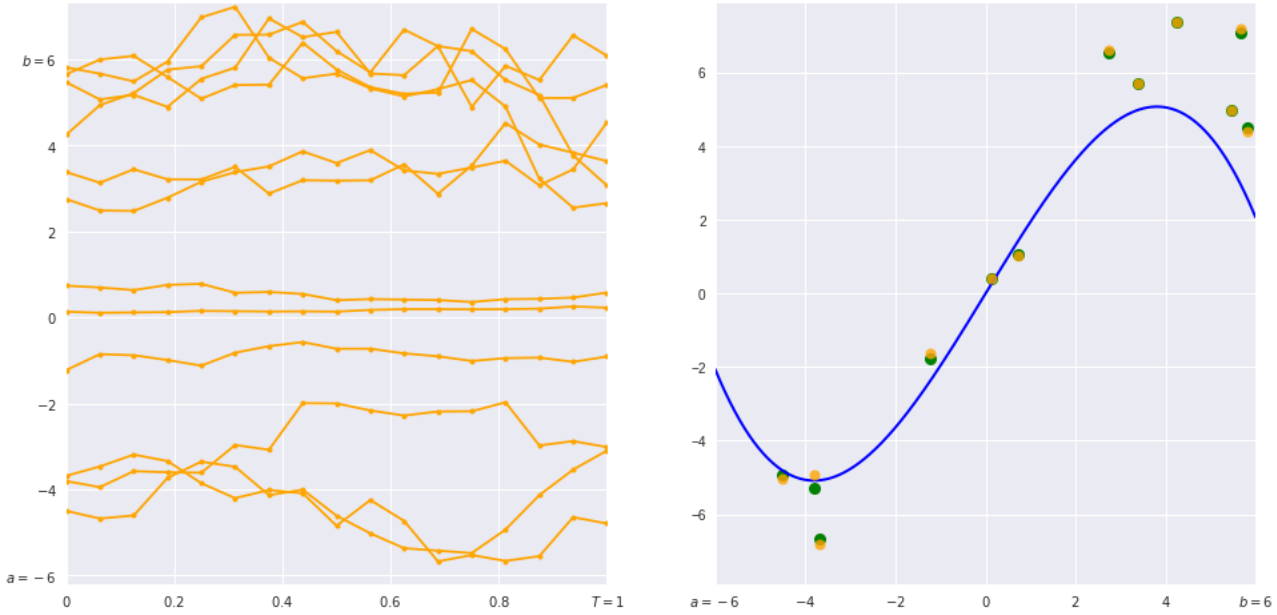

Figure 3.1: In accordance with Example 2.3.10 we choose $T = 1, \bar{\sigma} = 0.5, a = -6, b = 6, p = 3, c_0 = 0, c_1 = 1.77, c_2 = 0, c_3 = -0.015$. The plot shows $m = 10$ realizations of i.i.d. samples of the Euler-Maruyama approximation for $N = 16$, i.e. step size $\frac{1}{16}$, on the left (the points are connected by lines for visualization purposes) and corresponding realizations of samples from the data $\mathcal{Z}^N = \big(X, \varphi\big(\mathcal{S}_N^{N,X}\big)\big)$ on the right (orange). The realizations of the samples from the true data $Z = (X, Y)$ (green) as well as the regression function (blue) are shown as comparison (see also Figure 2.1).

3.5 Multilevel Monte Carlo Simulation

Recall that in Section 2.4 we motivated to use a Multilevel Learning approach in order to decrease the computational cost for the simulations. Furthermore Giles [29] originally suggested Multilevel Monte Carlo path simulations for estimating an expected value arising from a stochastic differential equation. Therefore analogously to Section 2.4 we propose to use Multilevel Euler-Maruyama simulations based on $L \in \mathbb{N}$ levels with an increasing number of steps. For every $l \in \mathbb{N}_0$ we denote by N_l the number of steps in the l-th level.

Proposition 3.5.1 (Multilevel Learning Problem for the Kolmogorov PDE)**.** *Let $f \in \mathscr{C}^{1,2}([0,T] \times \mathbb{R}^d, \mathbb{R})$ be an at most polynomially growing solution to the Kolmogorov equation and let $X\colon \Omega \to [a,b]^d$ be a (continuously) uniformly distributed $\mathcal{G}_0$-measurable random vector, which is independent of $\sigma(B_t, t \in [0,T])$. For every $l \in \mathbb{N}_0$ let $\mathcal{S}^{N_l,X}\colon \{0,1,\dots,N_l\} \times \Omega \to \mathbb{R}^d$ be the Euler-Maruyama approximation for the SDE (3.48) with step size $\frac{T}{N_l}$, define the 0-th level data*

$$\mathfrak{Z}^0 = \mathcal{Z}^{N_0} = \big(X, \mathcal{Y}^{N_0}\big) = \big(X, \varphi\big(\mathcal{S}_{N_0}^{N_0,X}\big)\big) \tag{3.115}$$

and the l-th level data

$$\mathfrak{Z}^l = \big(X, \mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}}\big) = \big(X, \varphi\big(\mathcal{S}_{N_l}^{N_l,X}\big) - \varphi\big(\mathcal{S}_{N_{l-1}}^{N_{l-1},X}\big)\big) \tag{3.116}$$

and denote by $\hat{\mathfrak{F}}^l$ the regression function w.r.t. the data $\mathfrak{Z}^l$. Then it holds that

(i) $\hat{\mathfrak{F}}^0(x) = \mathbb{E}\big[\varphi\big(\mathcal{S}_{N_0}^{N_0,x}\big)\big]$ *for a.e. $x \in [a,b]^d$ and*

(ii) the function $[a,b]^d \ni x \mapsto \mathbb{E}\big[\varphi\big(\mathcal{S}_{N_0}^{N_0,x}\big)\big] \in \mathbb{R}$ is the unique minimizer of

$$\min_{F \in \mathscr{C}([a,b]^d, \mathbb{R})} \mathbb{E}\Big[\big|F(X) - \varphi\big(\mathcal{S}_{N_0}^{N_0,X}\big)\big|^2\Big] \tag{3.117}$$

and for every $l \in \mathbb{N}$ it holds that

(iii) $\hat{\mathfrak{F}}^l(x) = \mathbb{E}\big[\varphi\big(\mathcal{S}_{N_l}^{N_l,x}\big) - \varphi\big(\mathcal{S}_{N_{l-1}}^{N_{l-1},x}\big)\big]$ *for a.e. $x \in [a,b]^d$ and*

(iv) the function $[a,b]^d \ni x \mapsto \mathbb{E}\big[\varphi\big(\mathcal{S}_{N_l}^{N_l,x}\big) - \varphi\big(\mathcal{S}_{N_{l-1}}^{N_{l-1},x}\big)\big] \in \mathbb{R}$ is the unique minimizer of

$$\min_{F \in \mathscr{C}([a,b]^d, \mathbb{R})} \mathbb{E}\left[\left|F(X) - \Big(\varphi\big(\mathcal{S}_{N_l}^{N_l,X}\big) - \varphi\big(\mathcal{S}_{N_{l-1}}^{N_{l-1},X}\big)\Big)\right|^2\right]. \tag{3.118}$$

For every $L \in \mathbb{N}$ define the data of the Multilevel Learning Problem for the Kolmogorov equation (with L levels) by

$$\big(\mathfrak{Z}^l\big)_{l=0}^L. \tag{3.119}$$

Then there exists a real number $C \in (0,\infty)$ such that for all $L \in \mathbb{N}$ it holds that

$$\sup_{x \in [a,b]^d} \left|\left(\mathbb{E}\big[\varphi\big(\mathcal{S}_{N_0}^{N_0,x}\big)\big] + \sum_{l=1}^{L} \mathbb{E}\big[\varphi\big(\mathcal{S}_{N_l}^{N_l,x}\big) - \varphi\big(\mathcal{S}_{N_{l-1}}^{N_{l-1},x}\big)\big]\right) - f(T,x)\right| \leq \frac{C}{\sqrt{N_L}}. \tag{3.120}$$

Proof. The statements of this Proposition are just a combination of Propositions 2.4.2 and 3.4.5. First observe that items (i) & (ii) follow directly from the corresponding statements in Proposition 3.4.5. Next for every $l \in \mathbb{N}$ the output data $\mathcal{Y}^{N_l} - \mathcal{Y}^{N_{l-1}} = \varphi\big(\mathcal{S}_{N_l}^{N_l,X}\big) - \varphi\big(\mathcal{S}_{N_{l-1}}^{N_{l-1},X}\big)$ suffices analogous moment bounds as the output data $\mathcal{Y}^{N_l} = \varphi\big(\mathcal{S}_{N_l}^{N_l,X}\big)$ (see (3.100)). The linearity of the expectation guarantees that we can mimic the proof of Proposition 3.4.5 in order to prove items (iii) & (iv) and also the claim in (3.120) as for every $L \in \mathbb{N}$ it holds that

$$\begin{aligned}\sup_{x\in[a,b]^d}\left|\left(\mathbb{E}\big[\varphi\big(\mathcal{S}_{N_0}^{N_0,x}\big)\big] + \sum_{l=1}^{L}\mathbb{E}\big[\varphi\big(\mathcal{S}_{N_l}^{N_l,x}\big) - \varphi\big(\mathcal{S}_{N_{l-1}}^{N_{l-1},x}\big)\big]\right) - f(T,x)\right| \\ = \sup_{x\in[a,b]^d}\left|\mathbb{E}\big[\varphi\big(\mathcal{S}_{N_L}^{N_L,x}\big)\big] - f(T,x)\right|.\end{aligned} \tag{3.121}$$

This concludes the proof of Proposition 3.5.1. □

In other words the previous proposition states that the regression function $\hat{\mathcal{F}}^{N_L}$ w.r.t. to the data $\big(X, \varphi\big(\mathcal{S}_{N_L}^{N_L,X}\big)\big)$ satisfies

$$\hat{\mathcal{F}}^{N_L}(x) = \sum_{l=0}^{L}\hat{\mathfrak{F}}^{l}(x) \tag{3.122}$$

for a.e. $x \in [a,b]^d$. It enables us to use the sum of the (unique) continuous versions of the regression functions $\sum_{l=0}^{L}\hat{\mathfrak{F}}^{l}(x)$ w.r.t. to the Multilevel data $\big(\mathfrak{Z}^l\big)_{l=0}^{L}$ as a suitable approximation to the solution of the Kolmogorov equation $f(T,x)$ at time T for all $x \in [a,b]^d$ and hence this Multilevel Learning approach presents a new alternative to the standard approach of Proposition 3.4.5. In accordance with the definition before Proposition 2.4.1 let us define samples for the empirical (Multilevel) Learning Problem.

Definition 3.5.2 (Samples of the Multilevel Learning Problem for the Kolmogorov PDE)**.** *Let $L \in \mathbb{N}$, $m_0, m_1, \dots, m_L \in \mathbb{N}$, let $B^{l,i}\colon [0,T]\times\Omega \to \mathbb{R}^d$, $l,i \in \mathbb{N}_0$, be independent standard $(\Omega, \mathcal{G}, \mathbb{P}, (\mathcal{G}_t)_{t\in[0,T]})$-Brownian motions with $B^{0,0} = B$, let $X_{l,i}\colon \Omega \to [a,b]^d$, $l,i \in \mathbb{N}_0$, be independent (continuously) uniformly distributed $\mathcal{G}_0$-measurable random vectors with $X_{0,0} = X$, which are independent of $\sigma\big(B_t^{l,i}, t \in [0,T], l,i \in \mathbb{N}_0\big)$, and for every $l,i \in \mathbb{N}_0$, $N \in \mathbb{N}$ let $\mathcal{S}^{N,l,i}\colon \{0,1,\dots,N\}\times\Omega \to \mathbb{R}^d$ be the Euler-Maruyama approximation for the SDE*

$$S_t^{l,i} = X_{l,i} + \int_0^t \mu\big(S_s^{l,i}\big)\,ds + \int_0^t \sigma\big(S_s^{l,i}\big)\,dB_s^{l,i}. \tag{3.123}$$

with step size $\frac{T}{N}$. Then we define the Multilevel samples $\big(\mathbf{Z}_l\big)_{l=0}^{L}$ for L levels with m_0 samples on level zero, m_1 samples on level one, …, m_L samples on level L by

$$\mathbf{Z}_0 = \Big(\big(X_{0,i}, \mathcal{Y}_{0,i}^{N_0}\big)\Big)_{i=1}^{m_0} = \Big(\big(X_{0,i}, \varphi\big(\mathcal{S}_{N_0}^{N_0,0,i}\big)\big)\Big)_{i=1}^{m_0} \tag{3.124}$$

and

$$\mathbf{Z}_l = \Big(\big(X_{l,i}, \mathcal{Y}_{l,i}^{N_l} - \mathcal{Y}_{l,i}^{N_{l-1}}\big)\Big)_{i=1}^{m_l} = \Big(\big(X_{l,i}, \varphi\big(\mathcal{S}_{N_l}^{N_l,l,i}\big) - \varphi\big(\mathcal{S}_{N_{l-1}}^{N_{l-1},l,i}\big)\big)\Big)_{i=1}^{m_l} \tag{3.125}$$

for $l \in \{1,2,\dots,L\}$. Let $\mathscr{H}$ be a valid hypothesis space, then for every $l \in \{0,1,\dots,L\}$ we denote by $\hat{\mathfrak{F}}_{\mathbf{Z}_l}$ the empirical target function w.r.t. to the samples $\mathbf{Z}_l$.

Note that for the standard approach according to Proposition 3.4.5 we obtain $m \in \mathbb{N}$ i.i.d. samples with precision level $l \in \mathbb{N}$, i.e. step size $\frac{T}{N_l}$, by defining

$$\mathbf{Z} = \Big(\big(X_i, \mathcal{Y}_i^{N_l}\big)\Big)_{i=1}^{m} = \Big(\big(X_{0,i}, \varphi\big(\mathcal{S}_{N_l}^{N_l,0,i}\big)\Big)_{i=1}^{m}. \tag{3.126}$$

Let us review the special case that the precision levels are chosen such that $N_0 \in \mathbb{N}$ and for every $l \in \mathbb{N}$ it holds that

$$N_l = sN_{l-1} \tag{3.127}$$

with $s \in \mathbb{N} \setminus \{1\}$. Then for every $l \in \mathbb{N}$ we can use the realizations of the Brownian motion increments, i.e. the realizations of the normally distributed random vectors, for the simulation of $\mathcal{S}_{N_l}^{N_l,X}$ to also calculate a (one level lower) simulation of $\mathcal{S}_{N_{l-1}}^{N_{l-1},X}$. This is done by adding s successive increments due to the fact that for all $n \in \{0, 1, \dots, N_{l-1}\}$ it holds that

$$B_{\frac{(n+1)T}{N_{l-1}}} - B_{\frac{nT}{N_{l-1}}} = \sum_{i=1}^{s} \left(B_{\frac{(sn+i)T}{N_l}} - B_{\frac{(sn+i-1)T}{N_l}} \right). \tag{3.128}$$

This shows that in this case the computational cost of simulating a Multilevel sample

$$\left(X_{l,i}, \mathcal{Y}_{l,i}^{N_l} - \mathcal{Y}_{l,i}^{N_{l-1}}\right) \tag{3.129}$$

(see (3.125)) is comparable to the cost of simulating a standard sample $\left(X_i, \mathcal{Y}_i^{N_l}\right)$ (see (3.126)). Passing to the empirical Multilevel Learning Problem we now show that assumptions (A) from Chapter 2.4 can be satisfied and in view of Proposition 2.4.3 investigate possible advantages of the Multilevel Learning approach. A calculation similar to the one in (3.111) assures that there exists a real number $C \in (0, \infty)$ such that for every $l \in \mathbb{N}_0$ it holds that

$$\left\| Y - \mathcal{Y}^{N_l} \right\|_{L^2(\mathbb{P};\|\cdot\|_{\mathbb{R}^d})} = \mathbb{E}\left[\left| \varphi\left(S_T^X\right) - \varphi\left(\mathcal{S}_{N_l}^{N_l,X}\right) \right|^2 \right]^{\frac{1}{2}} \leq \frac{C}{\sqrt{N_l}}. \tag{3.130}$$

Furthermore for simulating a realization of $\mathcal{Y}^{N_l}$ using the Euler-Maruyama scheme we need one realization of a uniformly distributed random vector, dN_l random normal realizations for the Brownian motion increments, in summary $2N_l + 1$ function evaluations of φ, σ and μ and $N_l(2d+2)+2$ additional floating point operations. Thus there exists a real number $\tilde{C} \in (0, \infty)$ such that the expected computational cost satisfies for every $l \in \mathbb{N}$ that

$$\mathbb{E}\left[\mathcal{C}^{N_l}\right] = \tilde{C} N_l. \tag{3.131}$$

We conclude that assumptions (A) in Section 2.4 are satisfied with

$$\alpha = \frac{1}{2}, \quad \gamma = 1. \tag{3.132}$$

Therefore at least in the extreme case $a = b$ Proposition 2.4.3 suggests that there exists $L \in \mathbb{N}$ and $m_0, m_1, \dots, m_L \in \mathbb{N}$ such that by defining the corresponding Multilevel samples $(\mathbf{Z}_l)_{l=0}^L$ and the corresponding empirical target functions $\hat{\mathfrak{F}}_{\mathbf{Z}_l}$ we obtain a mean squared error

$$\mathbb{E}\left[\left| \sum_{l=0}^{L} \hat{\mathfrak{F}}_{\mathbf{Z}_l}(X) - f(T,X) \right|^2 \right] = \mathcal{O}\left(\varepsilon^2\right) \tag{3.133}$$

with an expected overall computational cost for the simulations of

$$\mathbb{E}[\mathcal{C}] = \mathcal{O}\left(\varepsilon^{-2} (\log \varepsilon)^2\right) \tag{3.134}$$

for ε tending to zero. This is opposed to the standard approach which by defining suitable $l, m \in \mathbb{N}$, samples $\mathbf{Z} = \left((X_i, \mathcal{Y}_i^{N_l})\right)_{i=1}^m$ and the corresponding empirical target function $\hat{\mathcal{F}}_{\mathbf{Z}}^{N_l}$ obtains a mean squared error

$$\mathbb{E}\left[\left| \hat{\mathcal{F}}_{\mathbf{Z}}^{N_l}(X) - f(T,X) \right|^2 \right] = \mathcal{O}\left(\varepsilon^2\right) \tag{3.135}$$

with an expected overall computational cost for the simulations of

$$\mathbb{E}[\mathcal{C}] = \mathcal{O}\big(\varepsilon^{-4}\big) \tag{3.136}$$

for ε tending to zero (see equation (2.98)). Under additional assumptions one could employ the result on the weak convergence rate of the Euler-Maruyama scheme (Remark 3.4.4), but one still faces a computational cost of $\mathbb{E}[\mathcal{C}] = \mathcal{O}\big(\varepsilon^{-3}\big)$ (cf. Giles [29], Hutzenthaler [44] and the references mentioned therein). In view of (3.134), (3.136) and Proposition 3.5.1 we propose to take the sum of the empirical target functions w.r.t. the Multilevel samples as approximation of $f(T,x)$, i.e.

$$\sum_{l=0}^{L} \hat{\mathfrak{F}}_{\mathbf{Z}_l}(x) \approx f(T,x) \tag{3.137}$$

for every $x \in [a,b]^d$.

Example 3.5.3. *We will further investigate our previous example in the Multilevel setting. Let* $L = 3$ *and we define*

$$N_l = 2^l \tag{3.138}$$

for $l \in \{0,1,2,3\}$. *Then for* $l > 0$ *the Multilevel data takes the form*

$$\mathfrak{Z}^l = \big(X, \varphi\big(\mathcal{S}_{2^l}^{2^l,X}\big) - \varphi\big(\mathcal{S}_{2^{l-1}}^{2^{l-1},X}\big)\big) \tag{3.139}$$

and when computing a simulation of the coarse Euler-Maruyama approximation $\mathcal{S}_{2^{l-1}}^{2^{l-1},X}$ *we use the same Brownian motion trajectory as for the simulation of the fine scheme* $\mathcal{S}_{2^l}^{2^l,X}$ *(see* (3.128) *and Figure 3.2).*

Figure 3.2: The plot shows the fine Euler-Maruyama approximation (bright orange) and the corresponding coarse one (orange) for the Multilevel data $\mathfrak{Z}^3 = \big(X, \varphi\big(\mathcal{S}_8^{8,X}\big) - \varphi\big(\mathcal{S}_4^{4,X}\big)\big)$. The true trajectory of the solution to the stochastic differential equation is shown in red as a comparison.

Let us take the hypothesis space $\mathscr{H} = \mathscr{P}_6([a,b],\mathbb{R})$ *(see Example 2.3.10) and choose*

$$m_l = 100 \cdot 2^{3-l} \tag{3.140}$$

for $l \in \{0,1,2,3\}$*. We perform* 50 *independent trials and each time simulate realizations of the Multilevel samples* $(\mathbf{z}_l)_{l=0}^L$ *and compute the corresponding empirical target functions* $\hat{\mathfrak{F}}_{\mathbf{z}_l}$*,* $l \in \{0,1,2,3\}$*. In view of* (3.133) *and Proposition 3.5.1 we sum the latter to compute an approximation*

$$\sum_{l=0}^{L} \hat{\mathfrak{F}}_{\mathbf{z}_l}(x) \approx f(T,x) \tag{3.141}$$

to the solution to the Kolmogorov equation (3.57) *at time* T *for* $x \in [a,b]^d$*. As a comparison for each trial we also follow the standard approach due to Proposition 3.4.5 and* (3.135) *with* $N = 2^L = 8$ *and* $m = 400$ *realizations of samples* $\mathbf{z}$*. This corresponds to the same precision of the regression functions and the same number of random normal calls for both approaches (see also the introduction of Chapter 5). Denote the empirical target functions of the standard approach by* $\hat{\mathcal{F}}_{\mathbf{z}}^8$*. Figure 3.3 illustrates that for this comparable computational cost of the simulations the mean squared error over the trials is significantly smaller for* $\sum_{l=0}^3 \hat{\mathfrak{F}}_{\mathbf{z}_l}$ *than for* $\hat{\mathcal{F}}_{\mathbf{z}}^8$*.*

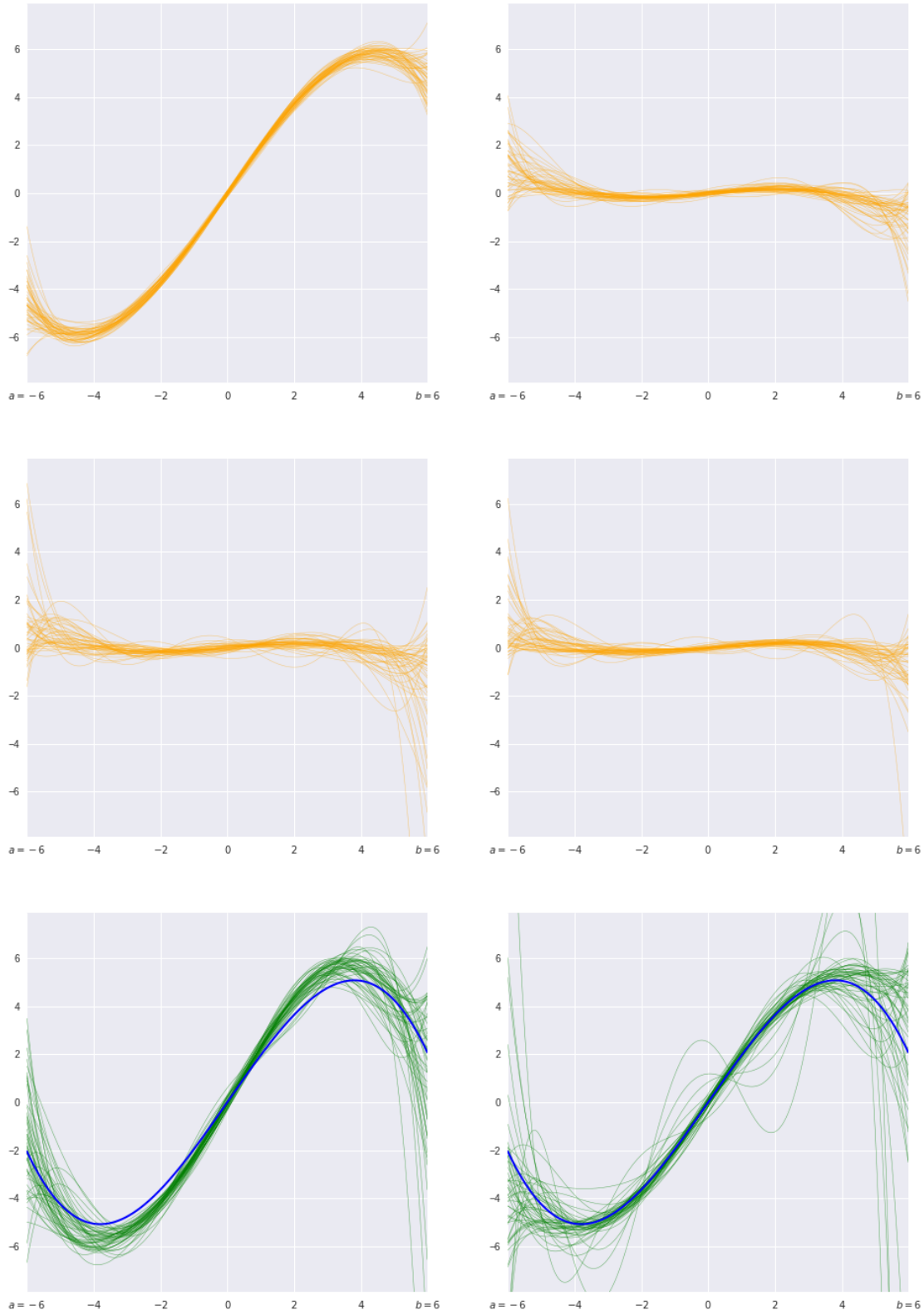

Figure 3.3: The first four plots show the empirical target functions w.r.t. to the realizations of each of the Multilevel samples $(\mathbf{z}_l)_{l=0}^3$ for 50 independent trials (orange). The plot on the bottom left and right shows for each trial $\sum_{l=0}^3 \hat{\mathfrak{F}}_{\mathbf{z}_l}$ and $\hat{\mathcal{F}}_{\mathbf{z}}^8$ respectively (green) and the regression function is shown as comparison (blue). Note that by using the same number of random normal calls the mean squared error over all trials equals 0.94 for the Multilevel case opposed to 5.24 in the standard approach.

Chapter 4

Neural Networks as Hypothesis Space

The following chapter offers a short overview of neural networks and their approximation properties. Especially when dealing with high-dimensional input data, that is large $d \in \mathbb{N}$ in the setting of the last chapters, we suggest a class of neural networks as a suitable hypothesis space $\mathscr{H}$ for our Learning Problem. Thereupon the chapter summarizes the Learning Problem for the Kolmogorov equation and explains the optimization algorithm for approximately solving the empirical Learning Problem in the hypothesis space of neural networks.

4.1 Definition

In this section we want to introduce the notion of artificial feed-forward neural networks (also known as multilayer perceptrons). The definition is based on a very simplified mathematical model of the neural structure in a human brain and follows standard literature (cf., for instance, Bishop [11] & [12, Chapter 5], Goodfellow [33], Haykin [38] and Caterini [18]). For every $M, N \in \mathbb{N}$ let us denote by $\mathscr{A}(\mathbb{R}^M, \mathbb{R}^N)$ the space of affine linear mappings from $\mathbb{R}^M$ to $\mathbb{R}^N$.

Definition 4.1.1 ((Deep) Artificial Feed-Forward Neural Network)**.** *Let $d, n, s \in \mathbb{N}$ and let $M_0, M_1, \dots, M_{s+1} \in \mathbb{N}$ with $M_0 = d$ and $M_{s+1} = n$. Let $\alpha \colon \mathbb{R} \to \mathbb{R}$ be a function and for every $i \in \{0, 1, \dots, s\}$ let $W_i \in \mathbb{R}^{M_{i+1} \times M_i}$, $b_i \in \mathbb{R}^{M_{i+1}}$ and denote by $A_i \in \mathscr{A}(\mathbb{R}^{M_i}, \mathbb{R}^{M_{i+1}})$ the affine linear mapping given by*

$$A_i(x) = W_i x + b_i \tag{4.1}$$

for every $x \in \mathbb{R}^{M_i}$. We define the function $\mathbb{F} : \mathbb{R}^d \to \mathbb{R}^n$ by

$$\mathbb{F}(x) = (A_s \circ \alpha \circ A_{s-1} \circ \alpha \circ A_{s-2} \circ \cdots \circ \alpha \circ A_0)(x) \tag{4.2}$$

for every $x \in \mathbb{R}^d$, where the function α is applied component-wise. We call $\mathbb{F}$ an (artificial feed-forward) neural network with activation function α, d units in the input layer, n units in the output layer, and M_i, $i \in \{1, 2, \dots, s\}$, units in each of the s hidden layers and refer to $s, d, M_1, \dots, M_s, n$ and the function α as the network architecture. We call W_i the weight matrix and b_i the biases in the i-th layer and refer to the collection of weight matrices and biases $\theta = \big((W_i, b_i)\big)_{i=0}^{s}$ as the parameters of the network. We say that a neural network is deep, if $s \geq 2$.

Note that the parameters of a neural network can be represented as a vector in $\mathbb{R}^\nu$ with

$$\nu = \sum_{i=0}^{s} M_{i+1} M_i + M_{i+1} = M_1(d+1) + n(M_s + 1) + \sum_{i=1}^{s-1} M_{i+1}(M_i + 1) \tag{4.3}$$

and we will often explicitly stress the dependence of $\mathbb{F}$ on θ by writing $\mathbb{F}_\theta$. The architecture of the network can be depicted as a directed acyclic graph like in Figure 4.1.

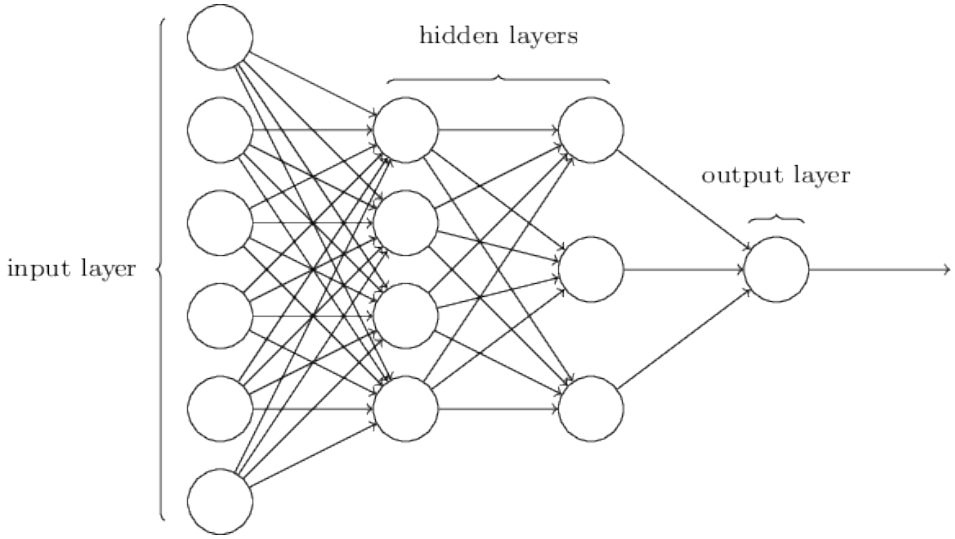

Figure 4.1: Graph of a deep neural network with $s = 2$, $d = 6$, $M_1 = 4$, $M_2 = 3$, $n = 1$, from Nielsen [62]

4.2 Properties

A neural network with continuous non-polynomial activation function can approximate all continuous functions on a compact subset of $\mathbb{R}^d$ arbitrary well with respect to the uniform norm provided that enough hidden units are available. This is a special case of the following more general theorem on the approximation capabilities of neural networks.

Theorem 4.2.1 (Universal Approximation Theorem)**.** *Let $d \in \mathbb{N}$, let $\alpha\colon \mathbb{R} \to \mathbb{R}$ be a locally bounded activation function and assume that the closure of the points of discontinuity of α is a Lebesgue null set, let*

$$\mathscr{N}_{(d,\alpha)} = \left\{A_1 \circ \alpha \circ A_0 : M \in \mathbb{N}, A_0 \in \mathscr{A}(\mathbb{R}^d, \mathbb{R}^M), A_1 \in \mathscr{A}(\mathbb{R}^M, \mathbb{R})\right\} \tag{4.4}$$

be the set of all neural networks with activation function α, d input units, one hidden layer (with an arbitrary number of hidden units) and one output unit. Then

(i) for every compact subset $\mathcal{D} \subseteq \mathbb{R}^d$, every $\varepsilon > 0$ and every $F \in \mathscr{C}(\mathbb{R}^d, \mathbb{R})$ there exists $\mathbb{F} \in \mathscr{N}_{(d,\alpha)}$ such that it holds that

$$\sup_{x \in \mathcal{D}} |F(x) - \mathbb{F}(x)| < \varepsilon \tag{4.5}$$

(ii) for every absolutely continuous (w.r.t. Lebesgue measure), compactly supported probability measure $\mathbb{P}$ on $\mathbb{R}^d$, every $\varepsilon > 0$, every $1 \leq p < \infty$, and every $F \in L^p(\mathbb{P}; |\cdot|)$ there exists $\mathbb{F} \in \mathscr{N}_{(d,\alpha)}$ such that it holds that

$$\|F(x) - \mathbb{F}(x)\|_{L^p(\mathbb{P};|\cdot|)} < \varepsilon \tag{4.6}$$

if and only if α is not an algebraic polynomial.

Proof. Leshno [57] and also many other sources with slightly different assumptions and statements, cf. Hornik [43], Cybenko [22], Scarselli [74] and Pinkus [65] for a survey. □

Observe that the theorem holds also for neural networks with an arbitrary number of hidden layers and output units as we can represent each output unit separately and approximate the identity in the other layers.

4.3 Proposed Algorithm

Based on this optimal approximation qualities (cf. also Bölcskei [14]), we suggest the set of neural networks with fixed architecture as an appropriate hypothesis space for our Learning Problem. However this is not a compact subspace of the space of bounded measurable functions, i.e. the conditions of a hypothesis space are not satisfied. We circumvent this by assuming that the activation function is continuous, the parameters of our network are bounded by a constant R and that the network is evaluated only on a compact subset $\mathcal{D}$.

Definition 4.3.1 (Hypothesis Space of Neural Networks)**.** *Let $s \in \mathbb{N}$, $d, M_1, \dots, M_s, n \in \mathbb{N}$, $R \in (0, \infty)$, let $\mathcal{D} \subseteq \mathbb{R}^d$ compact, let $\alpha \in \mathscr{C}(\mathbb{R}, \mathbb{R})$ non-polynomial and define $\nu \in \mathbb{N}$ by equation (4.3). Let $\mathbb{F}_\theta \colon \mathbb{R}^d \to \mathbb{R}^n$, $\theta \in \mathbb{R}^\nu$, be neural networks depending on the parameters θ with activation function α and M_i, $i \in \{1, 2, \dots, s\}$, units in each of the s hidden layers. Then we define the hypothesis space of neural networks with the given architecture as*

$$\mathscr{N} = \mathscr{N}_{s,d,M_1,M_2,\dots,M_s,n,\mathcal{D},R,\alpha} = \left\{ \mathbb{F}_\theta \big|_{\mathcal{D}} : \|\theta\|_{\mathbb{R}^\nu} \leq R \right\} \underset{compact}{\subseteq} \mathscr{B}(\mathcal{D}, \mathbb{R}^n)\,. \tag{4.7}$$

Although Theorem 4.2.1 shows that even neural networks with only one hidden layer can approximate every continuous function arbitrary well (given enough hidden units), recent practical applications show more success by employing deep neural networks for high-dimensional input data (cf., for instance, Hinton [41] and Krizhevsky [55]) and there is also mathematical evidence for this phenomenon (cf. Yarotsky [79] and Petersen [64]). Let us summarize our Learning Problem in the given context:

(i) We want to compute the solution $f(T, x)$ of the Kolmogorov type partial differential equation

$$\begin{cases} \frac{\partial f}{\partial t}(t,x) = \frac{1}{2} \operatorname{Trace}_{\mathbb{R}^d}\big(\sigma(x)[\sigma(x)]^* (\operatorname{Hess}_x f)(t,x)\big) + \big\langle \mu(x), (\nabla_x f)(t,x) \big\rangle_{\mathbb{R}^d} \\ f(0,x) = \varphi(x) \end{cases} \tag{4.8}$$

at time T for $x \in [a,b]^d$.

(ii) Assuming enough smoothness of f, φ, μ and σ this is equivalent to computing the expected value $\mathbb{E}\big[\varphi(S_T^x)\big]$, where S_T^x is the solution to the SDE

$$S_t^x = x + \int_0^t \mu\big(S_s^x\big)\, ds + \int_0^t \sigma\big(S_s^x\big)\, dW_s. \tag{4.9}$$

at time T with initial value $x \in [a,b]^d$.

(iii) This is equivalent to finding the (continuous version of the) regression function $\hat{F}$ w.r.t. the data $Z = \big(X, \varphi\big(S_T^X\big)\big)$ with X uniformly distributed in $[a,b]^d$, i.e. finding the unique minimizer of

$$\min_{F \in \mathscr{C}([a,b]^d, \mathbb{R})} \mathbb{E}\Big[\big| F(X) - \varphi\big(S_T^X\big) \big|^2 \Big] \tag{4.10}$$

(iv) As we lack knowledge of the solution to the stochastic differential equation, we approximate S_T^X by $\mathcal{S}_N^{N,X}$ using the Euler-Maruyama scheme with sufficient large N and consider the (continuous version of the) regression function $\hat{\mathcal{F}}^N$ w.r.t. the approximated data $\mathcal{Z}^N = \big(X, \varphi\big(\mathcal{S}_N^{N,X}\big)\big)$, i.e. the unique minimizer of

$$\min_{F \in \mathscr{C}([a,b]^d, \mathbb{R})} \mathbb{E}\Big[\big|F(X) - \varphi\big(\mathcal{S}_N^{N,X}\big)\big|^2\Big] \tag{4.11}$$

(v) As we have only access to a finite number of samples from the data $\mathcal{Z}^N$, we consider a Monte Carlo approximation of the expected value in equation (4.11) and we minimize in the hypothesis space of deep neural networks $\mathscr{H} = \mathscr{N} = \mathscr{N}_{s,d,M_1,M_2,\dots,M_s,n,\mathcal{D},R,\alpha}$ with $n = 1$, $s \geq 2$ and $\mathcal{D} = [a,b]^d$. To decrease the computational cost of the simulations we mimic the idea of Multilevel Monte Carlo approximation with $L \in \mathbb{N}$ levels of precision $N_0, N_1, \dots, N_L$. For each level $l \in \{0, 1, \dots, L\}$ we take $m_l \in \mathbb{N}$ samples

$$\begin{cases} \mathbf{Z}_0 = \Big(\big(X_{0,i}, \varphi\big(\mathcal{S}_{N_0}^{N_0,0,i}\big)\big)\Big)_{i=1}^{m_0}, & l = 0 \\ \mathbf{Z}_l = \Big(\big(X_{l,i}, \varphi\big(\mathcal{S}_{N_l}^{N_l,l,i}\big) - \varphi\big(\mathcal{S}_{N_{l-1}}^{N_{l-1},l,i}\big)\big)\Big)_{i=1}^{m_l}, & l = 1, 2, \dots, L \end{cases} \tag{4.12}$$

(see Definition 3.5.2) and seek to find the empirical target function $\hat{\mathfrak{F}}_{\mathbf{Z}_l}$ w.r.t. $\mathbf{Z}_l$, i.e. a minimizer of

$$\begin{cases} \min\limits_{F \in \mathscr{N}} \dfrac{1}{m_0} \displaystyle\sum_{i=1}^{m_0} \Big|F\big(X_{0,i}\big) - \varphi\big(\mathcal{S}_{N_0}^{N_0,0,i}\big)\Big|^2, & l = 0 \\ \min\limits_{F \in \mathscr{N}} \dfrac{1}{m_l} \displaystyle\sum_{i=1}^{m_l} \Big|F\big(X_{l,i}\big) - \Big(\varphi\big(\mathcal{S}_{N_l}^{N_l,l,i}\big) - \varphi\big(\mathcal{S}_{N_{l-1}}^{N_{l-1},l,i}\big)\Big)\Big|^2, & l = 1, 2, \dots, L\,. \end{cases} \tag{4.13}$$

Proposition 2.3.7, Proposition 3.5.1, Proposition 3.4.5 and Proposition 3.3.1 suggest that by controlling the approximation, model and sample errors we obtain that for a.e. $x \in [a,b]^d$ it holds that

$$\sum_{l=0}^{L} \hat{\mathfrak{F}}_{\mathbf{Z}_l}(x) \approx \sum_{l=0}^{L} \hat{\mathfrak{F}}^l(x) = \hat{\mathcal{F}}^{N_L}(x) \approx \hat{F}(x) = f(T,x)\,. \tag{4.14}$$

(vi) Minimizing the empirical error over $F \in \mathscr{N}$ (for a given outcome $\omega \in \Omega$) is equivalent to finding optimal parameters $\hat{\theta} \in \mathbb{R}^\nu$ (i.e. weights and biases) for a deep neural network with a given architecture. For each $l \in \{0, 1, \dots, L\}$ let $\mathbb{F}_\theta^l \colon [a,b]^d \to \mathbb{R}$, $\theta \in \mathbb{R}^\nu$, be neural networks depending on the parameters θ with activation function α and M_i, $i \in \{1, 2, \dots, s\}$, units in each of the s hidden layers. Then due to equation (4.13) for every $l \in \{0, 1, \dots, L\}$ we seek $\hat{\theta}_l \in \mathbb{R}^\nu$ as minimizer of

$$\begin{cases} \min\limits_{\|\theta\|_{\mathbb{R}^\nu} \leq R} \dfrac{1}{m_0} \displaystyle\sum_{i=1}^{m_0} \Big|\mathbb{F}_\theta^0\big(X_{0,i}(\omega)\big) - \varphi\big(\mathcal{S}_{N_0}^{N_0,0,i}(\omega)\big)\Big|^2, & l = 0 \\ \min\limits_{\|\theta\|_{\mathbb{R}^\nu} \leq R} \dfrac{1}{m_l} \displaystyle\sum_{i=1}^{m_l} \Big|\mathbb{F}_\theta^l\big(X_{l,i}(\omega)\big) - \Big(\varphi\big(\mathcal{S}_{N_l}^{N_l,l,i}(\omega)\big) - \varphi\big(\mathcal{S}_{N_{l-1}}^{N_{l-1},l,i}(\omega)\big)\Big)\Big|^2, & l = 1, 2, \dots, L\,. \end{cases} \tag{4.15}$$

The next section presents a method to approximately solve the latter minimization problems, i.e. to find $\tilde{\theta}_l \approx \hat{\theta}_l \in \mathbb{R}^\nu$, and under appropriate hypothesis we can think of

$$[a,b]^d \ni x \mapsto \sum_{l=0}^{L} \mathbb{F}^l_{\tilde{\theta}_l}(x) \in \mathbb{R} \tag{4.16}$$

as a suitable approximation of the function

$$[a,b]^d \ni x \mapsto f(T,x) \in \mathbb{R}\,. \tag{4.17}$$

There are also some modifications, extensions and comments to the above summary. First by starting in item (ii), one can also use the algorithm to compute $\mathbb{E}[\varphi(S_T^x)]$ (without the connection to the Kolmogorov equation), i.e. to solve certain problems involving the solutions of stochastic differential equations. Secondly, one can invoke other methods than the Euler-Maruyama scheme to approximate the solution of a stochastic differential equation (cf. Kloeden [54]). Lastly we could also allow for a different neural network architectures in each level.

4.4 Optimization

For the convenience of the reader let us generalize the optimization problems in (4.15). Throughout this chapter let $d, m, n, s \in \mathbb{N}$, $r \in \mathbb{N}_0$, $M_1, M_2, \ldots, M_s \in \mathbb{N}$, $\mathbb{R} \in (0,\infty)$, $\mathcal{D} \subset \mathbb{R}^d$ compact, $\alpha \in \mathscr{C}^r(\mathbb{R},\mathbb{R})$ non-polynomial and define ν by equation (4.3). Let $\mathbb{F}_\theta \colon \mathcal{D} \to \mathbb{R}^n$, $\theta \in \mathbb{R}^\nu$, be neural networks depending on the parameters θ with activation function α and M_i, $i \in \{1,2,\ldots,s\}$, units in each of the s hidden layers and let

$$\mathbf{z} = \big((x_i, y_i)\big)_{i=1}^m \in (\mathcal{D} \times \mathbb{R}^n)^m \tag{4.18}$$

denote m realizations of samples, then each of the optimization problems in (4.15) is of the following form (with $n = 1$, $\mathcal{D} = [a,b]^d$ and loss functions $\mathcal{L}_y(\tilde{y}) = \big|\tilde{y} - y\big|^2$ for every $\tilde{y}, y \in \mathbb{R}$).

Definition 4.4.1 (General Optimization Problem for Neural Networks)**.** *For every $y \in \mathbb{R}^n$ let*

$$\mathcal{L}_y \in \mathscr{C}^r(\mathbb{R}^n, [0,\infty)) \tag{4.19}$$

and for every $i \in \{1,2,\ldots,m\}$ define the functions $h_i \colon \mathbb{R}^\nu \to \mathbb{R}$ and $H \colon \mathbb{R}^\nu \to \mathbb{R}$ by

$$h_i(\theta) = \mathcal{L}_{y_i}\big(\mathbb{F}_\theta(x_i)\big) \tag{4.20}$$

and

$$H(\theta) = \frac{1}{m}\sum_{i=1}^m h_i(\theta) \tag{4.21}$$

for every $\theta \in \mathbb{R}^\nu$. The general optimization problem for the neural networks $\mathbb{F}_\theta$, $\theta \in \mathbb{R}^\nu$, with loss functions $\mathcal{L}_y$, $y \in \mathbb{R}^n$, and samples $\mathbf{z}$ asks for parameters $\hat{\theta} \in \mathbb{R}^\nu$ minimizing

$$\min_{\|\theta\|_{\mathbb{R}^\nu} \le R} H(\theta) = \min_{\|\theta\|_{\mathbb{R}^\nu} \le R} \frac{1}{m}\sum_{i=1}^m h_i(\theta) = \min_{\|\theta\|_{\mathbb{R}^\nu} \le R} \frac{1}{m}\sum_{i=1}^m \mathcal{L}_{y_i}\big(\mathbb{F}_\theta(x_i)\big)\,. \tag{4.22}$$

Recall that a (not necessary unique) minimizer $\hat{\theta}$ of (4.22) exists due to the fact that we chose the hypothesis space of neural networks $\mathscr{N}$ as a compact subspace of $\mathscr{B}(\mathcal{D}, \mathbb{R}^n)$ (see Definition 4.3.1) which in this case corresponds to the fact that we are minimizing a continuous function over a compact domain. Furthermore the assumptions on α and $\mathcal{L}_y$, $y \in \mathbb{R}^n$, imply that also the objective function H is r-times continuously differentiable. The General Optimization problem can be classified as a bound constrained nonlinear optimization problem which is thoroughly analyzed in the standard literature, for example Ruszczyński [73], Bertsekas [8] and Griva [35]. Let us present the plain vanilla version of an iterative algorithm for approximately solving (4.22), the so-called gradient descent algorithm (also known as steepest descent algorithm). Let us assume that R is sufficiently large and that we can ignore this bound constraint. As motivation recall that for $H \in \mathscr{C}^1(\mathbb{R}^\nu, \mathbb{R})$ the (instantaneous) rate of change at $\theta \in \mathbb{R}^\nu$ per unit of distance moved in the direction given by $v \in \mathbb{R}^\nu$, i.e. the directional derivative, equals

$$\lim_{h \to 0} \frac{H(\theta + hv) - H(\theta)}{h\|v\|_{\mathbb{R}^\nu}} = \frac{1}{\|v\|_{\mathbb{R}^\nu}} \left\langle \nabla H(\theta), v \right\rangle_{\mathbb{R}^\nu}. \tag{4.23}$$

Due to the Cauchy-Schwarz inequality the direction of (instananeous) steepest descent is therefore $-\gamma \nabla H(\theta)$ for some $\gamma \in (0, \infty)$.

Definition 4.4.2 (Gradient Descent Algorithm)**.** *Let $\gamma \in (0, \infty)$, $\theta^{(0)} \in \mathbb{R}^\nu$, let $H \in \mathscr{C}^1(\mathbb{R}^\nu, \mathbb{R})$, and let $\theta^{(k)} \in \mathbb{R}^\nu$, $k \in \mathbb{N}$, be a sequence such that for every $k \in \mathbb{N}$ it holds that*

$$\theta^{(k)} = \theta^{(k-1)} - \gamma \nabla H\big(\theta^{(k-1)}\big). \tag{4.24}$$

Then we call $\theta^{(k)}$, $k \in \mathbb{N}_0$, a gradient descent sequence for H with initial value $\theta^{(0)}$ and step size γ.

Note that in the context of neural networks the step size is often also referred to as learning rate. Under sufficient conditions on the function H the gradient descent algorithm converges at least to a critical point.

Theorem 4.4.3 (Convergence of the Gradient Descent Algorithm)**.** *Assume that $H \in \mathscr{C}^1(\mathbb{R}^\nu, \mathbb{R})$ is bounded from below and that its gradient is Lipschitz continuous with constant $K \in (0, \infty)$, i.e. for every $u, v \in \mathbb{R}^\nu$ it holds that*

$$\|\nabla H(u) - \nabla H(v)\|_{\mathbb{R}^\nu} \leq K \|u - v\|_{\mathbb{R}^\nu}. \tag{4.25}$$

For every step size

$$\gamma \in \big(0, \tfrac{1}{K}\big) \tag{4.26}$$

and every initial value $\theta^{(0)} \in \mathbb{R}^\nu$ it holds that the corresponding gradient descent sequence $\theta^{(k)}$, $k \in \mathbb{N}_0$, satisfies that

$$\lim_{k \to \infty} \nabla H\big(\theta^{(k)}\big) = 0 \tag{4.27}$$

Proof. Ruszczyński [73, Theorem 5.1]. □

Note that the objective function H in the general optimization problem for neural networks (Definition 4.4.1) is bounded from below by definition of the loss functions. Under additional assumptions one can show that the gradient descent sequence converges linearly to a minimizer, but there are examples with arbitrary slow rate of convergence (cf. Ruszczyński [73, Section 5.3]). One typical problem is the so-called zig-zagging in narrow curved valleys of the objective function which is illustrated in Figure 4.2.

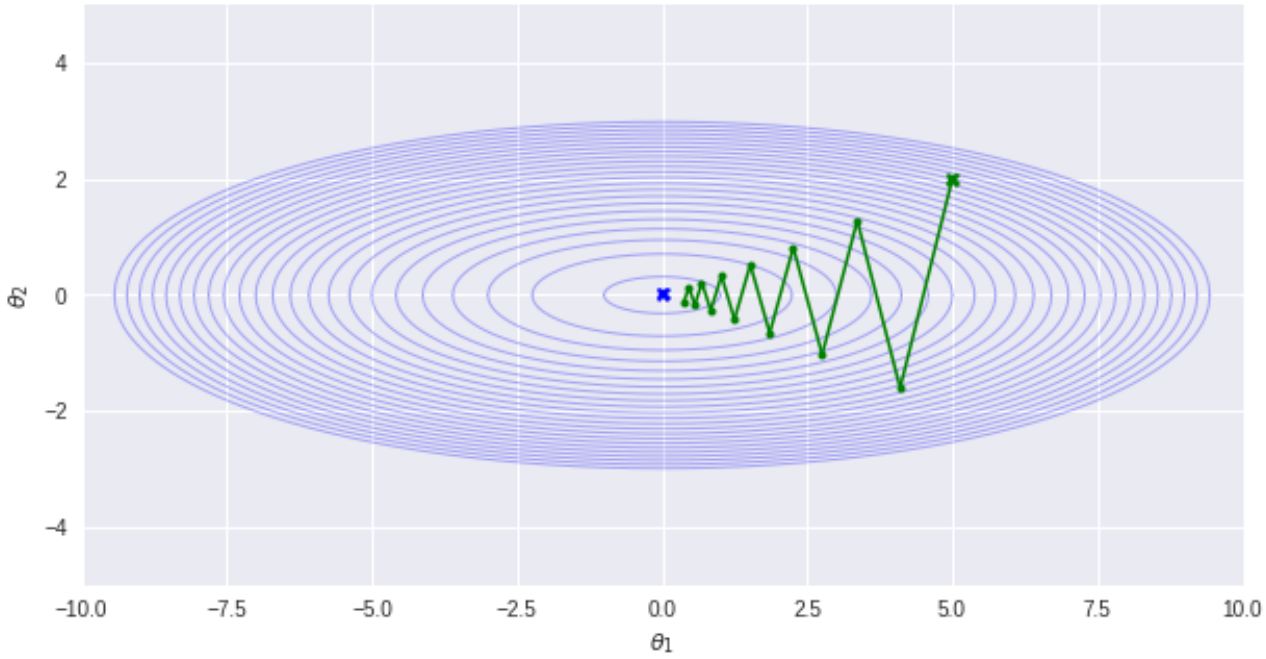

Figure 4.2: The plot shows level lines of the function $\mathbb{R}^2 \ni (\theta_1, \theta_2) \mapsto H(\theta_1, \theta_2) = \theta_1^2 + 10 \cdot \theta_2^2 \in \mathbb{R}$ (blue) and the first 13 steps of the gradient descent sequence for H with initial value $\theta^{(0)} = (5, 2)$ and step size $\gamma = 0.009$ (green). One can observe that the descent direction is perpendicular to the level lines. On the given domain one could choose the Lipschitz constant $K = 102$ and Theorem 4.4.3 guarantees convergence to the unique minimizer $(0, 0)$. But due to the narrow valley of the objective function, the gradient descent sequence proceeds in a zig-zagging manner and accordingly the convergence is very slow.

Before we go on to more advanced versions let us review the gradient descent algorithm in our neural network setting. In order to compute the gradient of

$$H = \frac{1}{m} \sum_{i=1}^{m} h_i \tag{4.28}$$

we need to calculate the gradient of

$$h_i(\theta) = \mathcal{L}_{y_i}\big(\mathbb{F}_\theta(x_i)\big) \tag{4.29}$$

w.r.t. $\theta \in \mathbb{R}^\nu$ for every $i \in \{1, 2, \ldots, m\}$. Intuitively for every $j \in \{0, 1, \ldots, \nu\}$, $i \in \{1, 2, \ldots, m\}$ the j-th coordinate of the gradient $\big(\nabla h_i\big)_j \in \mathbb{R}$ measures the (instantaneous) rate of change of the loss at sample (x_i, y_i) per unit change of the parameter θ_j. The computation of the gradients w.r.t. the parameters θ (i.e. the biases and weight matrices) can be done computationally efficient with the so-called backpropagation algorithm (cf. Bishop [12, Section 5.3] and Caterini [18, Section 3.2.3 & 4.1.2]). The latter makes use of the chain rule and the directed acyclic graph structure of the neural network to obtain the gradient of h_i at a cost of $\mathcal{O}(\nu)$ floating point operations for ν tending to infinity, i.e. it scales like the cost of a single evaluation of h_i at given parameters. In each step of the gradient descent algorithm one needs to calculate m gradients ∇h_i, $i \in \{0, 1, \ldots, m\}$, and therefore the total complexity of one gradient descent step is equal to $\mathcal{O}(m\nu)$ for m and ν tending to infinity. In practice one usually needs a lot

of samples (in order to obtain a small sample error, see the discussion after Proposition 2.3.7 and Theorem 2.3.8) and it would be desirable to have an algorithm with a computational cost which is not scaling with the number of samples m. As a motivation we note that the gradient descent algorithm often redundantly recomputes gradients for similar samples in one step. This leads to the definition of the mini-batch stochastic gradient descent algorithm, where in each step we take into account only $\bar{m} \in \mathbb{N}$ samples uniformly chosen from all m samples at random. In doing so we obtain an unbiased estimator of the gradient of H.

Definition 4.4.4 ((Mini-Batch) Stochastic Gradient Descent Algorithm)**.** *Let $m, \bar{m} \in \mathbb{N}$ with $\bar{m} < m$, let $\gamma \in (0, \infty)$, $\theta^{(0)} \in \mathbb{R}^\nu$ and for every $i \in \{1, 2, \dots, m\}$ let $h_i \in \mathscr{C}^1(\mathbb{R}^\nu, \mathbb{R})$ and define*

$$H = \frac{1}{m} \sum_{i=0}^{m} h_i \in \mathscr{C}^1(\mathbb{R}^\nu, \mathbb{R}) \tag{4.30}$$

and

$$\mathcal{I} = \left\{ (i_1, i_2, \dots, i_{\bar{m}}) \in \mathbb{N}^{\bar{m}} : 1 \leq i_1 < i_2 < \cdots < i_{\bar{m}} \leq m \right\}, \tag{4.31}$$

let

$$\iota^{(k)} = \left(\iota_1^{(k)}, \iota_2^{(k)}, \dots, \iota_{\bar{m}}^{(k)} \right) \colon \Omega \to \mathcal{I}, \quad k \in \mathbb{N}, \tag{4.32}$$

be independent uniformly distributed random vectors and let $\theta^{(k)} \colon \Omega \to \mathbb{R}^\nu$, $k \in \mathbb{N}$, be a sequence of random vectors such that for every $k \in \mathbb{N}$ it holds that

$$\theta^{(k)} = \theta^{(k-1)} - \gamma \frac{1}{\bar{m}} \sum_{j=1}^{\bar{m}} \nabla h_{\iota_j^{(k)}} \left(\theta^{(k-1)} \right). \tag{4.33}$$

Then we call $\theta^{(k)}$, $k \in \mathbb{N}_0$, a mini-batch stochastic gradient descent sequence for H with initial value $\theta^{(0)}$, batch-size $\bar{m}$ and step size γ. If $\bar{m} = 1$ we omit the prefix "mini-batch".

While the mini-batch stochastic gradient descent algorithm has decreased computational cost, the variance in the gradient estimates causes the gradient sequence to fluctuate heavily, which complicates the convergence to an exact minimizer (see Figure 4.3). Nevertheless one can also proof convergence under suitable conditions (cf. Bertsekas [9], Jentzen [47] and Shamir [76]) and these fluctuations can in some cases also prevent the algorithm from being stuck in unsatisfactory local minima. There are a lot of further optimization strategies, for instance introducing momentum, adaptive learning rates, early stopping, batch normalization and gradient noise and clipping. A good overview of possible variants and improvements of the (mini-batch) stochastic gradient descent algorithm are given in Ruder [72] and Goodfellow [33, Chapter 8]. Ultimately note that minimization algorithms that use second order information of the objective function (e.g. Newton's method, cf. Griva [35, Section 11.3]) are infeasible for neural networks in practice due to the large number of parameters.

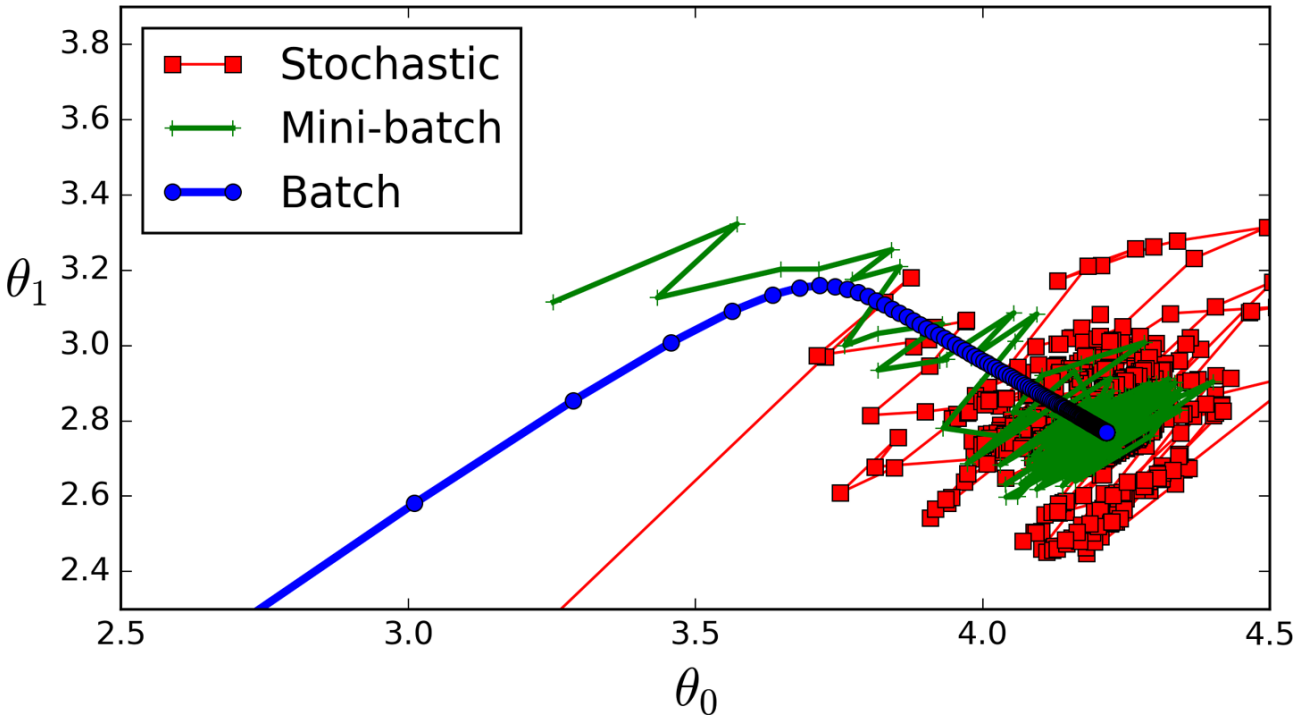

Figure 4.3: The figure depicts the typical fluctuations of the (mini-batch) stochastic gradient descent sequences near the minimizer opposed to the (standard) gradient descent sequence (blue), which ultimately stops at the minimizer. However each step of the latter takes approximately m-times the computational effort of a step from the stochastic gradient sequence (from Géron [28]).

Chapter 5

Numerical Results

For the following examples we slightly extend the framework in Section 4.4 and Section 4.3 in order to meet latest developments in the field of Deep Learning. In view of the optimization methods this includes the use of

(i) ADAM (adaptive moment estimation) optimizer with initial learning rate γ and batch-size $\bar{m}$ (cf. Kingma [50])

(ii) Batch Normalization with momentum λ (cf. Ioffe [45]) and

(iii) gradient clipping with threshold τ (cf. Goodfellow [33, Subsection 8.2.4])

for $\bar{m} \in \mathbb{N}$ and $\gamma, \lambda, \tau \in (0, \infty)$. Note that in practice we do not choose the sample size $m \in \mathbb{N}$ in advance since in each optimization step we only simulate the needed samples according to the batch-size $\bar{m}$. When we eventually obtain the desired accuracy and stop the algorithm at the k-th optimization step, this would correspond to a sample size of

$$m = k \cdot \bar{m} \,. \tag{5.1}$$

Furthermore we choose the samples of the uniformly distributed input variable X in the larger interval $[a - \Delta, b + \Delta]^d$ with $\Delta \in (0, \infty)$ for a better performance at the boundary. Recent research suggests to make use of the exponential linear unit activation function $\mathrm{ELU}\colon \mathbb{R} \to \mathbb{R}$ which is defined by

$$\mathrm{ELU}(x) = \begin{cases} x, & x > 0 \\ e^x - 1, & x \leq 0 \end{cases} \tag{5.2}$$

(cf. Clevert [19] and also Klambauer [51] for the *scaled* exponential linear unit activation function). For a neural network we denote by $\theta^{(0)}$ the initial parameters and by $\theta^{(k)}$ the parameters of the network after the k-th optimization step. Note that for the weights of the neural networks we use initial parameters which prevent variance scaling between the layers (cf. He [39] and Géron [28, Chapter 11]). We will neglect the bound on the parameters of the neural networks $R \in (0, \infty)$ as the representation of numbers is limited in an implementation anyway. In the Multilevel Learning approach let us take $L = 2$,

$$N_l = 4^{3+l} \tag{5.3}$$

time steps of the Euler-Maruyama scheme and batch size

$$\bar{m}_l = 256 \cdot 4^{2-l} \tag{5.4}$$

in order to simulate the data for the empirical Learning Problems for each level $l \in \{0,1,2\}$. The corresponding realizations of Multilevel samples we denote by $\big(\mathbf{z}_l\big)_{l=0}^2$ (see (3.125)). In the standard approach we choose

$$N = N_L = 4^5 \tag{5.5}$$

time steps, which results in the same precision for the regression functions $\hat{\mathcal{F}}^{N_L} = \sum_{l=0}^L \hat{\mathfrak{F}}^l$ (see equation (3.122)), and the associated realizations of samples $\mathbf{z}$ (see (3.126)). Further we take a batch-size of

$$\bar{m} = 256 \cdot 3 \tag{5.6}$$

corresponding to the same overall number of random normal calls in each descent step, namely

$$d \sum_{l=0}^{L} \bar{m}_l N_l = d(L+1) \cdot 256 \cdot 4^5 = d\bar{m}N\,. \tag{5.7}$$

To evaluate the performance of our algorithm we selectively compute

(i) the mean squared relative and absolute error (MSRE and MSE)

(ii) the relative $L1$-error (by Monte-Carlo integration)

between the real solution $f(T,x) = \mathbb{E}\big[\varphi(X_T^x)\big]$ (or a suitable approximation) and the neural network approximations to the empirical target functions (after the k-th optimization step)

$$\begin{cases} \sum_{l=0}^2 \mathbb{F}^l_{\theta^{(k)}}(x), & \text{for Multilevel Learning} \\ \mathbb{F}_{\theta^{(k)}}(x), & \text{for Standard Learning} \end{cases} \tag{5.8}$$

at uniformly distributed points in the interval $[a,b]^d$. The implementation (see Appendix .1) is written in the programming language Python using the open-source software library TensorFlow (cf. Abardi [1]) and executed on a p3.2xlarge Elastic Compute Cloud (EC2) instance hosted by Amazon Web Services (AWS).

5.1 Toy Example

Initially we will deal with our known Example 3.3.2 in the context of the proposed algorithm from Section 4.3. Of course for one dimensional problems like this there are much more efficient algorithms (and even explicit solutions), but we stick to this example for a complete picture, for explanatory purposes and as an indication of the behavior in more complex settings. Let $d = 1$, $p \in \mathbb{N}$, $a, \bar{\sigma}, c_0, \dots, c_p, T \in \mathbb{R}$, $b \in [a,\infty)$, let $\varphi\colon \mathbb{R} \to \mathbb{R}$ be the polynomial given by $\varphi(x) = \sum_{j=0}^p c_j x^j$ for every $x \in \mathbb{R}$ and recall that we seek to approximate the solution to the Kolmogorov equation

$$\begin{cases} \frac{\partial f}{\partial t}(t,x) = \frac{1}{2}\bar{\sigma}^2\big(x^2 \frac{\partial^2 f}{\partial x^2}(t,x) + x\frac{\partial f}{\partial x}(t,x)\big) \\ f(0,x) = \varphi(x) \end{cases} \tag{5.9}$$

at time T for $x \in [a,b]$. Like before we choose

$$T = 1,\ \bar{\sigma} = 0.5,\ a = -6,\ b = 6,\ p = 3,\ c_0 = 0,\ c_1 = 1.77,\ c_2 = 0,\ c_3 = -0.015\,. \tag{5.10}$$

Let

$$\begin{aligned} &\mathbb{F}^0 \in \mathscr{N}_{5,1,512,512,1024,512,512,1,[a,b],R,\mathrm{ELU}} \\ &\mathbb{F}^1, \mathbb{F}^2 \in \mathscr{N}_{5,1,128,128,256,128,128,1,[a,b],R,\mathrm{ELU}} \end{aligned} \tag{5.11}$$

and for each neural network $\mathbb{F}^l$, $l \in \{0,1,2\}$, we employ our optimization algorithm with

$$\gamma = 10^{-6},\ \lambda = 0.9,\ \tau = 0.3,\ \Delta = 0.2\,. \tag{5.12}$$

In doing so we obtain a sequence of parameters

$$\theta_l^{(k)}, \quad k \in \mathbb{N}\,, \tag{5.13}$$

which suggests

$$\sum_{l=0}^{2} \mathbb{F}^l_{\theta_l^{(k)}}(x) \approx f(T,x) \tag{5.14}$$

for $x \in [a,b]$ and large enough $k \in \mathbb{N}$. As a comparison we follow the standard approach with

$$\mathbb{F} \in \mathscr{N}_{5,1,512,512,1024,512,512,1,[a,b],R,\mathrm{ELU}} \tag{5.15}$$

and in the same setting as in equation (5.12) above we get a sequence of parameters $\theta^{(k)}$, $k \in \mathbb{N}$, suggesting

$$\mathbb{F}_{\theta^{(k)}}(x) \approx f(T,x) \tag{5.16}$$

for $x \in [a,b]$ and large enough $k \in \mathbb{N}$. Using the code in Appendix .1 the mean squared error between the neural network approximations w.r.t. the optimization time is depicted in Figure 5.1 and selected other errors can be compared in Table 5.1. Although our Multilevel approach takes more time for each descent step due to the increased cost of optimizing three neural networks simultaneously, at some stage it converges significantly faster and with less variance than the standard approach in this setting. This strengthens our conjecture of using the former approach, which seems to act like a regularizer on the learning process (cf. Goodfellow [33, Chapter 7] and Géron [28, Chapter 11] for commonly used regularization techniques).

	Standard approach			Multilevel approach		
Iter.	Time	MSE	rel. L1 Error	Time	MSE	rel. L1 Error
0	0.0	12.874+/-2.902	0.975+/-0.066	0.0	12.508+/-1.717	0.972+/-0.105
150	9.4	0.367+/-0.156	0.149+/-0.033	17.0	0.290+/-0.075	0.132+/-0.022
300	18.6	0.092+/-0.027	0.074+/-0.011	33.3	0.036+/-0.021	0.043+/-0.016
450	27.8	0.081+/-0.048	0.068+/-0.032	49.7	0.015+/-0.003	0.030+/-0.005
600	37.0	0.077+/-0.036	0.069+/-0.025	66.1	0.014+/-0.006	0.026+/-0.003
750	46.2	0.050+/-0.018	0.055+/-0.019	82.4	0.008+/-0.004	0.019+/-0.003
900	55.4	0.042+/-0.018	0.048+/-0.014	98.8	0.006+/-0.002	0.017+/-0.004
1050	64.6	0.060+/-0.030	0.058+/-0.023			
1200	73.8	0.045+/-0.023	0.049+/-0.017			
1350	82.9	0.036+/-0.017	0.044+/-0.018			
1500	92.1	0.041+/-0.019	0.046+/-0.013			

Table 5.1: Table of selected errors for both approaches evaluated at 50 uniformly distributed points in $[-6,6]$. The first number represents the mean and the second one the standard deviation over 5 independent trials.

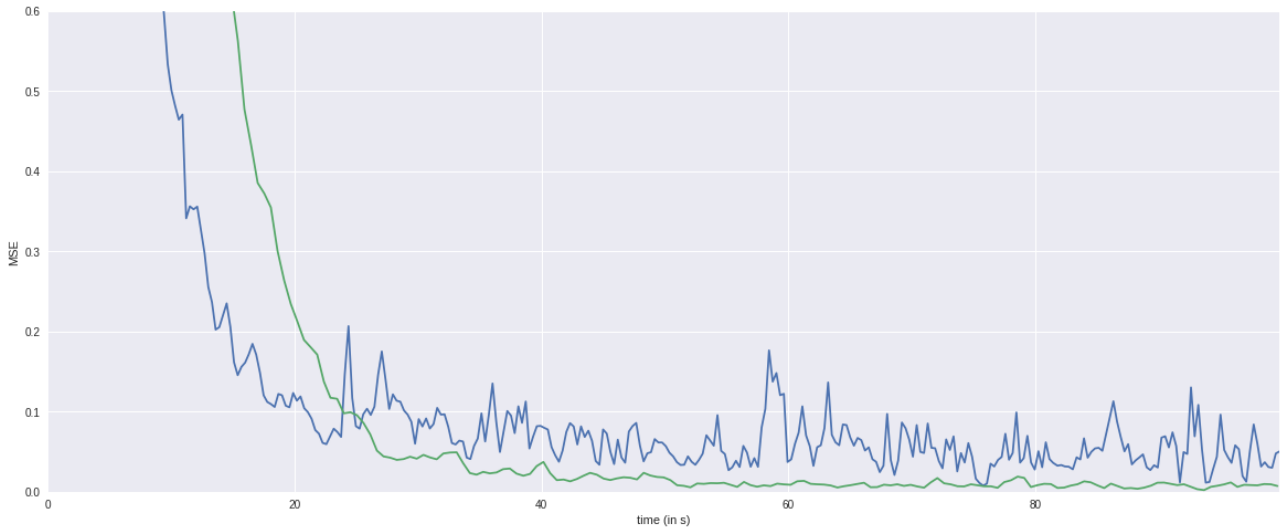

Figure 5.1: The plot shows the mean squared error (MSE) between the solution to the Kolmogorov PDE $f(T,x)$ and $\sum_{l=0}^{2} \mathbb{F}^{l}_{\theta_l^{(k)}}(x)$ (blue) or $\mathbb{F}_{\theta^{(k)}}(x)$ (green) evaluated at 50 uniformly distributed points in $[-6,6]$ for every fifth k in the first 100 seconds of optimization time.

5.2 Rainbow European Option

Let us investigate the behavior of our algorithm in a higher dimensional setting. Thereupon let $d \in \mathbb{N}$, $\mathcal{K} \in (0,\infty)$, let $\bar{\mu} \in \mathbb{R}^{d\times d}$ and $\bar{\sigma} \in \mathbb{R}^d$, for every $x \in [a,b]^d$ let $S^x = \big(S^{1,x},\dots,S^{d,x}\big)\colon [0,T]\times\Omega \to \mathbb{R}^d$ be a solution process to the SDE

$$S_t^{i,x} = x + \int_0^t \bar{\mu}_i S_s^{i,x}\,ds + \sum_{j=1}^{d}\int_0^t \bar{\sigma}_{ij} S_s^{i,x}\,dB_s^j, \quad i = 1,2,\dots,d \tag{5.17}$$

and define the function $\varphi\colon \mathbb{R}^d \to \mathbb{R}$ by

$$\varphi(x) = \max\left\{\max_{i=1,2,\dots,d} x_i - \mathcal{K}, 0\right\} \tag{5.18}$$

for every $x = (x_1,x_2,\dots,x_d) \in \mathbb{R}^d$. We are interested in approximating the function

$$[a,b]^d \ni x \to \mathbb{E}\big[\varphi\big(S_T^x\big)\big] \in \mathbb{R}\,, \tag{5.19}$$

which represents a financial model for pricing a Rainbow European Call (on max) option. This contract gives its owner the right, but not the obligation, to buy the maximal of d underlying financial assets S^x at a specified strike price $\mathcal{K}$ and at a given time T (cf. Glasserman [32, Chapter 3]). We denote by δ_{ij} the Kronecker delta and choose

$$d = 20,\ T = \frac{1}{12},\ a = 98,\ b = 102,\ \bar{\sigma}_{ij} = 0.01 + 0.03\cdot\delta_{ij},\ \bar{\mu}_i = 0.06,\ \mathcal{K} = 100 \tag{5.20}$$

for $i,j \in \{1,2,\dots,20\}$. Note that the function φ is not differentiable, but we can circumvent this problem by employing a more general version of the Feynman-Kac formula (see Remark 3.2.2). Furthermore we could again sample directly from the explicit solution of the SDE

$$S_T^{i,x} = x_i \exp\left(\big(\mu_i - \frac{1}{2}\sum_{j=1}^{d}\sigma_{ij}^2\big)T + \sum_{j=1}^{d}\sigma_{ij}W_T^j\right), \quad i = 1,2,\dots,d \tag{5.21}$$

(cf. Platen [66]) instead of using the Euler-Maruyama approximation, but we will only avail ourself of this opportunity to (approximatively) evaluate the function (5.19) and compute the errors. For the Multilevel approach we employ neural networks

$$\begin{aligned} \mathbb{F}^0 &\in \mathscr{N}_{5,20,128,1024,8192,1024,128,1,[a,b]^{20},R,\mathrm{ELU}} \\ \mathbb{F}^1, \mathbb{F}^2 &\in \mathscr{N}_{3,20,64,512,64,1,[a,b]^{20},R,\mathrm{ELU}} \end{aligned} \tag{5.22}$$

and in the standard case we take

$$\mathbb{F} \in \mathscr{N}_{5,20,128,1024,8192,1024,128,1,[a,b]^{20},R,\mathrm{ELU}} \,. \tag{5.23}$$

Moreover applying our optimization algorithm with

$$\gamma = 10^{-4},\ \lambda = 0.9,\ \tau = 0.3,\ \Delta = 0.2 \tag{5.24}$$

and using an exponential learning rate decay by the factor 0.1 every 4000 descent steps (see the second code in Appendix .1), we compare the MSE (see Figure 5.2) of the Multilevel and standard approach and list other errors in Tables 5.2 and 5.3. Furthermore we visualize the neural network structure in Figure 5.3. Although the Multilevel Learning approach again turns out to be advantageous in comparison to the standard approach, one must consider that for a complete picture one would need to compare the best architectures, optimization variants and batch-sizes for both approaches and take also in account implementation and hardware differences. While the former problem could be tackled using an exhaustive grid-search (cf. Géron [28, Chapter 11]) it was infeasible for this thesis. It seems that in certain settings the usual approach is at least equally successful reminding us that the advantages of the Multilevel approach can be proved only in an extreme case. Nevertheless the given examples demonstrate that the Multilevel Learning approach can possibly improve and accelerate the learning process significantly and should definitely be given a try, when tackling Learning Problems with computationally expensive simulations. To sum up, we see that our proposed algorithm from Section 4.3 provides a feasible way of solving problems involving high-dimensional stochastic differential equations (and in an equivalent way high-dimensional Kolmogorov equations) and seems not to underlie the curse of dimensionality.

Iter.	Time (in s)	MSE	MSRE	rel. L1 Error
0	0.0+/-0.0	10.044+/-0.964	1.0122+/-0.0931	0.988+/-0.046
1000	124.5+/-0.3	0.035+/-0.004	0.0036+/-0.0004	0.048+/-0.004
2000	248.7+/-0.5	0.019+/-0.004	0.0020+/-0.0004	0.034+/-0.004
3000	372.7+/-0.6	0.015+/-0.002	0.0016+/-0.0002	0.032+/-0.003
4000	496.9+/-0.8	0.012+/-0.002	0.0013+/-0.0002	0.028+/-0.002
5000	621.0+/-0.9	0.008+/-0.001	0.0008+/-0.0002	0.022+/-0.002
6000	745.1+/-0.9	0.007+/-0.001	0.0008+/-0.0001	0.022+/-0.002
7000	869.2+/-1.0	0.008+/-0.001	0.0009+/-0.0002	0.023+/-0.002
8000	993.2+/-1.2	0.007+/-0.001	0.0008+/-0.0001	0.022+/-0.002

Table 5.2: Table of selected errors for the Multilevel approach evaluated at 100 uniformly distributed points in $[98, 102]^{20}$. The first number represents the mean and the second one the standard deviation over 5 independent trials.

Iter.	Time (in s)	MSE	MSRE	rel. L1 Error
0	0.0+/-0.0	9.570+/-0.369	0.9672+/-0.0361	0.980+/-0.018
1000	66.1+/-0.2	0.083+/-0.018	0.0084+/-0.0017	0.073+/-0.009
2000	132.0+/-0.2	0.052+/-0.020	0.0055+/-0.0019	0.057+/-0.009
3000	197.9+/-0.2	0.050+/-0.019	0.0054+/-0.0023	0.056+/-0.009
4000	263.8+/-0.4	0.043+/-0.015	0.0045+/-0.0015	0.052+/-0.008
5000	329.8+/-0.4	0.014+/-0.004	0.0016+/-0.0005	0.030+/-0.004
6000	395.7+/-0.6	0.013+/-0.003	0.0014+/-0.0004	0.029+/-0.003
7000	461.7+/-0.6	0.014+/-0.002	0.0016+/-0.0003	0.030+/-0.002
8000	527.6+/-0.7	0.011+/-0.002	0.0012+/-0.0002	0.027+/-0.002
9000	593.4+/-0.8	0.009+/-0.002	0.0010+/-0.0002	0.025+/-0.002
10000	659.4+/-1.0	0.010+/-0.002	0.0011+/-0.0003	0.025+/-0.003
11000	725.3+/-1.0	0.010+/-0.002	0.0010+/-0.0003	0.025+/-0.003
12000	791.2+/-1.2	0.010+/-0.002	0.0011+/-0.0002	0.025+/-0.002
13000	857.1+/-1.3	0.009+/-0.002	0.0010+/-0.0002	0.025+/-0.002
14000	923.0+/-1.4	0.010+/-0.002	0.0010+/-0.0003	0.025+/-0.003
15000	989.0+/-1.5	0.009+/-0.002	0.0010+/-0.0002	0.025+/-0.002

Table 5.3: Table of selected errors for the standard approach evaluated at 100 uniformly distributed points in $[98, 102]^{20}$. The first number represents the mean and the second one the standard deviation over 5 independent trials.

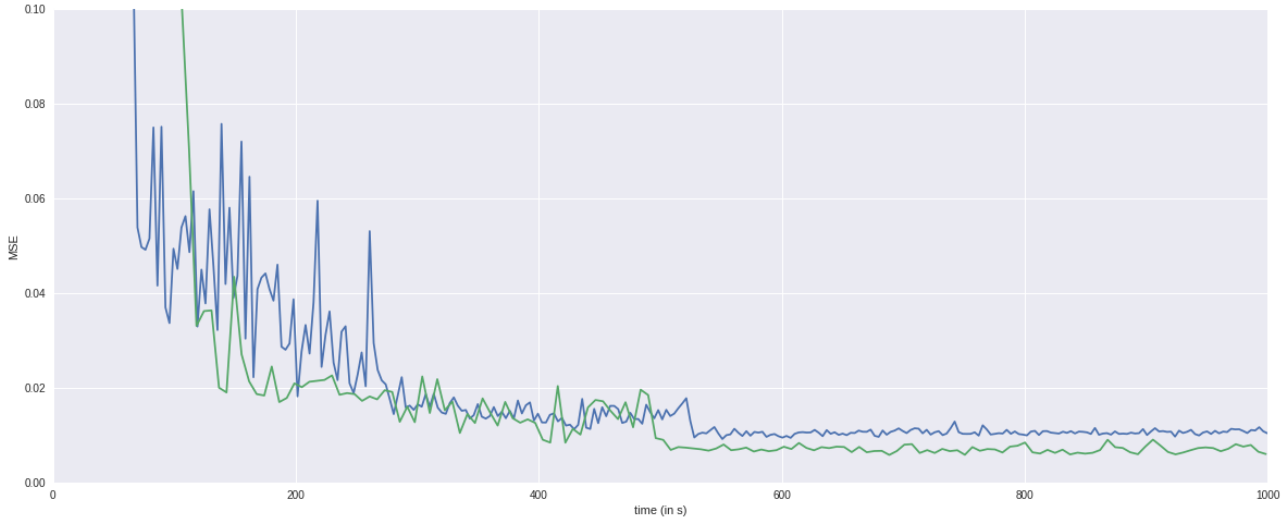

Figure 5.2: The plot shows the mean squared error between the (Monte Carlo approximated) solution and $\sum_{l=0}^{2} \mathbb{F}^{l}_{\theta_l^{(k)}}(x)$ (green) or $\mathbb{F}_{\theta^{(k)}}(x)$ (blue) evaluated at 100 uniformly distributed points in $[98, 102]^{20}$ for every fiftieth k in the first 1000 seconds of optimization time.

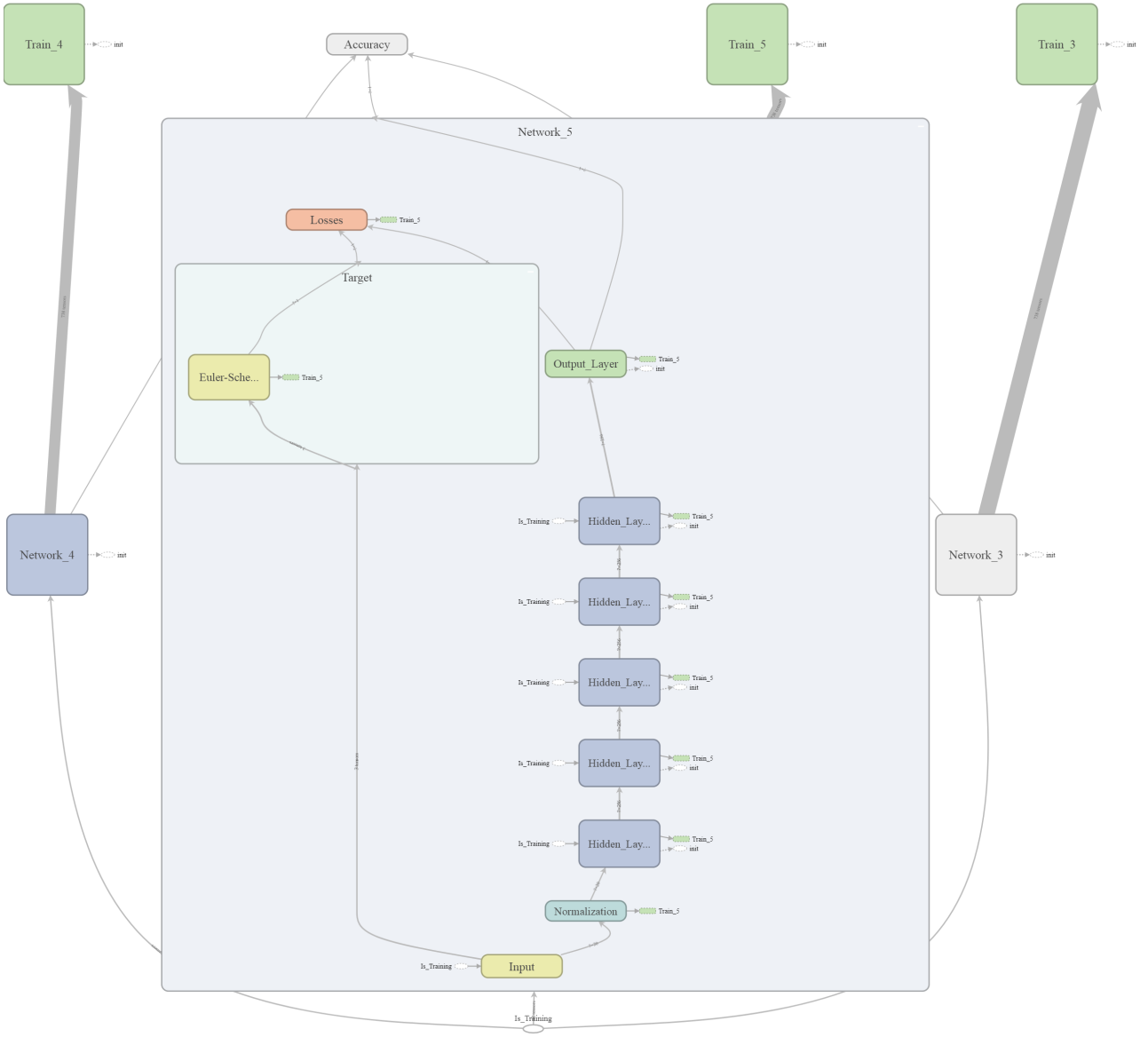

Figure 5.3: The figure visualizes the neural network structure for the Multilevel Learning approach and depicts the graph of the neural network in the finest level in more detail (created with TensorBoard, TensorFlow's visualization toolkit).

Chapter 6

Conclusion

This thesis successfully presented and merged three fields of (applied) mathematics, namely mathematical learning theory, stochastic analysis and foundations on neural networks. The resulting algorithm in Section 4.3 depicts a new and mathematically supported method for solving high-dimensional problems involving Kolmogorov equations and stochastic differential equations on a given domain. For the traditional numerical methods this usually represents an infeasible task. While we saw some promising examples of our proposed algorithm in Chapter 5, the framework in this thesis allows for extensive further applications and encourages more detailed mathematical analysis. First of all the Multilevel Learning approach in the first chapter could be applied and analyzed in many other Learning Problems, where large computational effort for computing simulations of samples is slowing down the empirical learning process. But even in our setting a continuing investigation on the behavior and performance of the two approaches would be interesting. This could be done numerically by an exhaustive grid-search over sensible parameters (with enough computational power), but we desire also more theoretical understanding on the convergence behavior. However there are ongoing mathematical attempts on understanding the sample and model errors of certain learning problems in the context of neural networks, which would lead to fruitful insights on suitable network architectures, sample sizes and approximation qualities. Beyond that we should mention additional improvements on the (Multilevel) Monte Carlo simulations by Quasi-Monte Carlo methods and importance sampling (cf. Caflisch [16], Glasserman [32, Section 4.6] and Giles [31]) and possible extensions to the case of second-order fully nonlinear partial differential equation (cf. Beck [6]). In summary the coupling of stochastic methods and generalization capabilities of neural networks is a far-reaching possibility to tackle the curse of dimensionality for solving partial differential equations and therefore enjoys much attention in recent research (cf. Sirignano [78], Khoo [49], E [25], Raissi [70]).

Appendix

.1 Source Codes

Python code for the Toy example from Section 5.1:

```
import tensorflow as tf
import numpy as np
from timeit import default_timer as timer
import pandas as pd

#general parameters
T=1.0 #time, where the solution is evaluated
sigma=0.5
mu=0.5*sigma**2
p_phi=[-0.015,0,1.77,0] #coefficients of polynomial phi
p_sol=[c*np.exp(0.5*(sigma*power)**2*T) for (c,power) in zip(p_phi,range(len(
    p_phi)-1,-1,-1))] #coefficients of explicit solution
a, b=-6., 6. #approximate the solution of the PDE in [a,b]^d
n_hidlayer_low=5 #number of hidden layers for the neural net in the lowest level
n_neurons_low=[512,512,1024,512,512] #[12,1024,1024,1024,128] #number of neurons
     for the hidden layers in the lowest level
n_hidlayer=5 #number of hidden layers for the neural nets in other levels
n_neurons=[128,128,256,128,128] #number of neurons for the hidden layers in the
    other levels
lr_rate=1e-6 #learning rate of the ADAM optimizer
valid_steps=5 #steps for validation of the metrics
w_initializer=tf.variance_scaling_initializer() #initializer of the weights
activation=tf.nn.elu #activation function
delta=0.2 #learn in slightly larger region
momentum=0.9 #momentum of batch normalization
gradient_clip=0.3 #gradient clipping

#function for building, fitting and evaulating the model
def fit(l_min,l_max,K,time,valid_points,real_sol,seed):
    #Euler-scheme for fine and coarse realizations
    def phi(x,lvl):
        with tf.variable_scope('Euler-Scheme'):
            with tf.variable_scope('Fine_Realization'):
                N_fine=tf.constant(4**lvl,name='Steps_fine',dtype=tf.int32)
                h_fine=tf.divide(T,tf.cast(N_fine,tf.float32),name='Step-
                    Size_fine')
                dw_fine=tf.random_normal(shape=[N_fine,tf.shape(x)[0],1],mean
                    =0.0,stddev=tf.sqrt(h_fine),name='DW')
                count=tf.constant(0,name='Count')
                def scheme_fine(i,realisation):
                    realisation+=(mu*h_fine+dw_fine[i,:,:]*sigma)*realisation
                    return [i+1,realisation]
                _, y_fine=tf.while_loop(lambda i,realisation: i<N_fine,
```

```
                scheme_fine,loop_vars=[count,x],name='Euler-Loop_fine')
            phi_fine=tf.reduce_sum(p_phi[3]+p_phi[2]*y_fine+p_phi[1]*y_fine**2+
                p_phi[0]*y_fine**3,axis=1,keepdims=True,name='Phi_fine')
            if lvl==l_min:
                return phi_fine
            else:
                with tf.variable_scope('Coarse_Realization'):
                    N_coarse=tf.constant(4**(lvl-1),name='Steps_coarse',dtype=tf
                        .int32)
                    h_coarse=tf.divide(T,tf.cast(N_coarse,tf.float32),name='Step
                        -Size_coarse')
                    dw_coarse=dw_fine[0::4,:,:]+dw_fine[1::4,:,:]+dw_fine
                        [2::4,:,:]+dw_fine[3::4,:,:]
                    def scheme_coarse(i,realisation):
                        realisation+=(mu*h_coarse+dw_coarse[i,:,:]*sigma)*
                            realisation
                        return [i+1,realisation]
                    _, y_coarse=tf.while_loop(lambda i,realisation: i<N_coarse,
                        scheme_coarse,loop_vars=[count,x],name='Euler-Loop_coarse
                        ')
                phi_coarse=tf.reduce_sum(p_phi[3]+p_phi[2]*y_coarse+p_phi[1]*
                    y_coarse**2+p_phi[0]*y_coarse**3,axis=1,keepdims=True,name='
                    Phi_coarse')
                phi_diff=tf.subtract(phi_fine,phi_coarse,name='Phi_diff')
                return phi_diff
    #function for building the neural networks
    def nn(valid_input,num_hidlayer,num_neurons,level,training):
        name_suffix=str(level)
        with tf.variable_scope('Network_'+name_suffix):
            with tf.variable_scope('Input'):
                batch_s=K*4**(l_max-level)
                input_layer=tf.cond(training,lambda: tf.random_uniform([batch_s
                    ,1],minval=a-delta,maxval=b+delta,name='Xi-Input'),lambda:
                    valid_input,name='Network-Input')
                prev_output=input_layer-(a+b)/2
            for n in range(num_hidlayer):
                with tf.variable_scope('Hidden_Layer%d' %(n+1)):
                    prev_output=tf.layers.dense(prev_output,num_neurons[n],
                        use_bias=False,kernel_initializer=w_initializer)
                    prev_output=tf.layers.batch_normalization(prev_output,
                        momentum=momentum,training=training)
                    prev_output=activation(prev_output)
            with tf.variable_scope('Output_Layer'):
                output=tf.layers.dense(prev_output,1,kernel_initializer=
                    w_initializer,name='Y_pred')
            with tf.variable_scope('Target'):
                z=phi(input_layer,level)
            with tf.variable_scope('Losses'):
                loss=tf.reduce_mean((z-output)**2,name='Loss')
        with tf.name_scope('Train_'+name_suffix):
            global_step=tf.Variable(0,trainable=False,name='Global_Step')
            learn_rate=tf.train.exponential_decay(lr_rate,global_step,5000,0.5,
                staircase=True,name='Learn_Rate')
            optimizer=tf.train.AdamOptimizer(learn_rate,name='Adam')
            grads_and_vars=optimizer.compute_gradients(loss)
            grads_and_vars=[(tf.clip_by_value(grad, -gradient_clip,
                gradient_clip), var) for grad, var in grads_and_vars if grad is
                not None]
```

```
79              update_ops=tf.get_collection(tf.GraphKeys.UPDATE_OPS, scope='
                    Network_'+name_suffix)
80              with tf.control_dependencies(update_ops):
81                  optimize=optimizer.apply_gradients(grads_and_vars,global_step=
                        global_step,name='Minimizer')
82          return output
83      #function for building the complete model
84      def build_model(valid_points,real_sol):
85          valid_input=tf.constant(valid_points,dtype=tf.float32)
86          is_training=tf.placeholder(tf.bool,name='Is_Training')
87          nn_outputs=[nn(valid_input,n_hidlayer,n_neurons,l,is_training) for l in
                range(l_min+1,l_max+1)]
88          nn_outputs.append(nn(valid_input,n_hidlayer_low,n_neurons_low,l_min,
                is_training))
89          with tf.variable_scope('Accuracy'):
90              f_real=tf.constant(real_sol,dtype=tf.float32,name='F_Real')
91              f_approx=tf.add_n(nn_outputs,name='F_Approximation')
92              abs_error=tf.abs(f_real-f_approx,name='Abs_Error')
93              abs_f_real=tf.abs(f_real,name='Abs_F_Real')
94              mse=tf.reduce_mean(abs_error**2,name='MSE')
95              rel_L1_error=tf.divide(tf.reduce_mean(abs_error),tf.reduce_mean(
                    abs_f_real),name='Rel_L1_Error')
96      #function for training the neural networks
97      def trainNN():
98          train_all=['Train_'+str(l)+'/Minimizer:0' for l in range(l_min,l_max+1)]
99          iteration=0
100         percentage=0
101         metrics_list=list()
102         fetch=['Accuracy/'+metric+':0' for metric in ['MSE','Rel_L1_Error']]
103         start=timer()
104         while timer()-start<time:
105             if ((iteration)%valid_steps)==0:
106                 runtime=timer()-start
107                 if (runtime/time)*100>percentage:
108                     print('------Progress: '+str(percentage)+'%',end='\r',flush=
                            True)
109                     percentage+=10
110                 metrics_valid=[iteration,runtime]
111                 metrics_valid.extend(sess.run(fetch,feed_dict={'Is_Training:0':
                        False}))
112                 metrics_list.append(metrics_valid)
113             sess.run(train_all,feed_dict={'Is_Training:0': True})
114             iteration+=1
115         return pd.DataFrame(metrics_list,columns=['Iteration','Time','MSE','rel.
                L1 Error'])
116
117     tf.reset_default_graph()
118     sess=tf.InteractiveSession()
119     tf.set_random_seed(seed)
120     build_model(valid_points,real_sol)
121     print('----Model build!')
122     tf.global_variables_initializer().run()
123     print('----Train Network(s)!')
124     metrics=trainNN()
125     sess.close()
126     return metrics
127
128 #Monte-Carlo approximation of the solution
```

```
def real_solution(seed):
    np.random.seed(seed)
    points=np.random.uniform(low=a,high=b,size=[num_points,1])
    solution=np.polyval(p_sol,points)
    return points,solution

#evaluate and compare the metrics
runtime=100 #runtime of the optimization algorithm in secondes
num_points=50 #number of evaluation points for the metrics
metrics_multilevel=list()
metrics_standard=list()
for seed in [2]: #range(5):
    print('\nSeed Nr. '+str(seed))
    points,solution=real_solution(seed)
    print('--Multilevel Learning:')
    lvl_min, lvl_max=3, 5 #4^(lvl_min) and 4^(lvl_max) are the number of time steps in the coarsest and finest levels respectively
    batch_mult=256 #multiplicator for the batch-size
    metric=fit(lvl_min,lvl_max,batch_mult,runtime,points,solution,seed)
    path=str(seed)+'_multilevel'
    metric.to_pickle(path)
    print('------Saved to '+path+' (pickled pandas DataFrame)')
    metrics_multilevel.append(metric)
    print('--Standard Learning:')
    lvl_min, lvl_max=5, 5
    batch_mult*=(lvl_max-lvl_min+1) #same number of random normal calls
    metric=fit(lvl_min,lvl_max,batch_mult,runtime,points,solution,seed)
    path=str(seed)+'_standard'
    metric.to_pickle(path)
    print('------Saved to '+path+' (pickled pandas DataFrame)')
    metrics_standard.append(metric)
print('\n\nFINISHED!') #open metrics_standard and metrics_multilevel - or use pd.read_pickle(path)
```

Python code for the Rainbow European Option example from Section 5.2:

```
import tensorflow as tf
import numpy as np
from timeit import default_timer as timer
import pandas as pd

#general parameters
d=20 #dimension (number of assets)
T=1./12. #expire time (one month)
sigma=0.01*np.ones([d,d])+0.03*np.eye(d)
mu=0.06*np.ones([d,1])
k=100 #strike price
a, b= 98, 102 #approx. the solution in [a,b]^d (initial value of the assets)
n_hidlayer_low=5 #number of hidden layers for the neural net in the lowest level
n_neurons_low=[128,1024,8192,1024,128] #number of neurons for the hidden layers in the lowest level
n_hidlayer=3 #number of hidden layers for the neural nets in the other levels
n_neurons=[64,512,64] #number of neurons for the hidden layers in the other levels
start_lr=1e-4 #initial learning rate of the ADAM optimizer
decay_steps=4000 #learning rate decay with factor 0.1
valid_steps=50 #steps for validation of the metrics
w_initializer=tf.variance_scaling_initializer() #initializer of the weights
activation=tf.nn.elu #activation function
```

```
delta=0.2 #learn in slightly larger region
momentum=0.9 #momentum of batch normalization
gradient_clip=0.3 #gradient clipping

#function for building, fitting and evaulating the model
def fit(l_min,l_max,K,time,valid_points,real_sol,seed):
    #Euler-scheme for fine and coarse realizations
    def phi(x,lvl):
        with tf.variable_scope('Euler-Scheme'):
            with tf.variable_scope('Fine_Realization'):
                N_fine=tf.constant(4**lvl,name='Steps_fine',dtype=tf.int32)
                h_fine=tf.divide(T,tf.cast(N_fine,tf.float32),name='Step-Size_fine')
                dw_fine=tf.random_normal(shape=[N_fine,tf.shape(x)[0],d],mean=0.0,stddev=tf.sqrt(h_fine),name='DW')
                count=tf.constant(0,name='Count')
                def scheme_fine(i,realisation):
                    realisation+=(mu_T*h_fine+tf.matmul(dw_fine[i,:,:],sigma_T))*realisation
                    return [i+1,realisation]
                _, y_fine=tf.while_loop(lambda i,realisation: i<N_fine, scheme_fine, loop_vars=[count,x],name='Euler-Loop_fine')
            phi_fine=tf.nn.relu(tf.reduce_max(y_fine, axis=1, keepdims=True)-k, name='Phi_fine')
            if lvl==l_min:
                return phi_fine
            else:
                with tf.variable_scope('Coarse_Realization'):
                    N_coarse=tf.constant(4**(lvl-1),name='Steps_coarse',dtype=tf.int32)
                    h_coarse=tf.divide(T,tf.cast(N_coarse,tf.float32),name='Step-Size_coarse')
                    dw_coarse=dw_fine[0::4,:,:]+dw_fine[1::4,:,:]+dw_fine[2::4,:,:]+dw_fine[3::4,:,:]
                    def scheme_coarse(i,realisation):
                        realisation+=(mu_T*h_coarse+tf.matmul(dw_coarse[i,:,:],sigma_T))*realisation
                        return [i+1,realisation]
                    _, y_coarse=tf.while_loop(lambda i,realisation: i<N_coarse, scheme_coarse, loop_vars=[count,x],name='Euler-Loop_coarse')
                phi_coarse=tf.nn.relu(tf.reduce_max(y_coarse, axis=1, keepdims=True)-k,name='Phi_coarse')
                phi_diff = tf.subtract(phi_fine,phi_coarse, name='Phi_diff')
                return phi_diff
    #function for building the neural networks
    def nn(valid_input,num_hidlayer,num_neurons,level,training):
        name_suffix=str(level)
        with tf.variable_scope('Network_'+name_suffix):
            with tf.variable_scope('Input'):
                batch_s=K*4**(l_max-level)
                input_layer=tf.cond(training,lambda: tf.random_uniform([batch_s,d],minval=a-delta,maxval=b+delta,name='Xi-Input'),lambda: valid_input,name='Network-Input')
            with tf.variable_scope('Normalization'):
                prev_output=input_layer-(a+b)/2
            for n in range(num_hidlayer):
                with tf.variable_scope('Hidden_Layer%d' %(n+1)):
```

```
66                      prev_output=tf.layers.dense(prev_output,num_neurons[n],
                            use_bias=False,kernel_initializer=w_initializer)
67                      prev_output=tf.layers.batch_normalization(prev_output,
                            momentum=momentum,training=training)
68                      prev_output=activation(prev_output)
69              with tf.variable_scope('Output_Layer'):
70                  output=tf.layers.dense(prev_output,1,kernel_initializer=
                        w_initializer,name='Y_pred')
71              with tf.variable_scope('Target'):
72                  z=phi(input_layer,level)
73              with tf.variable_scope('Losses'):
74                  loss=tf.reduce_mean((z-output)**2,name='Loss')
75          with tf.name_scope('Train_'+name_suffix):
76              global_step=tf.Variable(0,trainable=False,name='Global_Step')
77              learn_rate=tf.train.exponential_decay(start_lr,global_step,
                    decay_steps,0.1,staircase=True,name='Learn_Rate')
78              optimizer=tf.train.AdamOptimizer(learn_rate,name='Adam')
79              grads_and_vars=optimizer.compute_gradients(loss)
80              grads_and_vars=[(tf.clip_by_value(grad, -gradient_clip,
                    gradient_clip), var) for grad, var in grads_and_vars if grad is
                    not None]
81              update_ops=tf.get_collection(tf.GraphKeys.UPDATE_OPS, scope='
                    Network_'+name_suffix)
82              with tf.control_dependencies(update_ops):
83                  optimize=optimizer.apply_gradients(grads_and_vars,global_step=
                        global_step,name='Minimizer')
84          return output
85      #function for building the complete model
86      def build_model():
87          valid_input=tf.constant(valid_points,dtype=tf.float32)
88          is_training=tf.placeholder(tf.bool,name='Is_Training')
89          nn_outputs=[nn(valid_input,n_hidlayer,n_neurons,l,is_training) for l in
                range(l_min+1,l_max+1)]
90          nn_outputs.append(nn(valid_input,n_hidlayer_low,n_neurons_low,l_min,
                is_training))
91          with tf.variable_scope('Accuracy'):
92              f_real=tf.constant(real_sol,dtype=tf.float32,name='F_Real')
93              f_approx=tf.add_n(nn_outputs,name='F_Approximation')
94              abs_error=tf.abs(f_real-f_approx,name='Abs_Error')
95              abs_f_real=tf.abs(f_real,name='Abs_F_Real')
96              rel_abs_error=tf.divide(abs_error,abs_f_real,name='Rel_Abs_Error')
97              max_rel_error=tf.reduce_max(rel_abs_error,name='Max_Rel_Error')
98              mse=tf.reduce_mean(abs_error**2,name='MSE')
99              msre=tf.reduce_mean(rel_abs_error**2,name='MSRE')
100             rel_L1_error=tf.divide(tf.reduce_mean(abs_error),tf.reduce_mean(
                    abs_f_real),name='Rel_L1_Error')
101     #function for training the neural networks
102     def trainNN():
103         train_all=['Train_'+str(l)+'/Minimizer:0' for l in range(l_min,l_max+1)]
104         iteration=0
105         percentage=0
106         metrics_list=list()
107         fetch=['Accuracy/'+metric+':0' for metric in ['Max_Rel_Error','MSE','
                MSRE','Rel_L1_Error']]
108         start=timer()
109         while timer()-start<time:
110             if ((iteration)%valid_steps)==0:
111                 runtime=timer()-start
```

```
112                     if (runtime/time)*100>percentage:
113                         print('-------Progress: '+str(percentage)+'%',end='\r',flush=
                                True)
114                         percentage+=10
115                     metrics_valid=[iteration,runtime]
116                     metrics_valid.extend(sess.run(fetch,feed_dict={'Is_Training:0':
                            False}))
117                     metrics_list.append(metrics_valid)
118                 sess.run(train_all,feed_dict={'Is_Training:0': True})
119                 iteration+=1
120             return pd.DataFrame(metrics_list,columns=['Iteration','Time','Max. rel.
                    Error','MSE','MSRE','rel. L1 Error'])
121
122     tf.reset_default_graph()
123     sess=tf.InteractiveSession()
124     tf.set_random_seed(seed)
125     sigma_T=tf.constant(np.transpose(sigma),dtype=tf.float32)
126     mu_T=tf.constant(np.transpose(mu),dtype=tf.float32)
127     build_model()
128     print('-----Model build!')
129     tf.global_variables_initializer().run()
130     print('-----Train Network(s)!')
131     metrics=trainNN()
132     sess.close()
133     return metrics
134
135 #Monte-Carlo approximation of the solution
136 def MC_approx(seed):
137     tf.reset_default_graph()
138     sess=tf.InteractiveSession()
139     tf.set_random_seed(seed)
140     sigma_T=tf.constant(np.transpose(sigma),dtype=tf.float32)
141     mu_T=tf.constant(np.transpose(mu),dtype=tf.float32)
142     points=np.random.uniform(low=a,high=b,size=[num_points,d])
143     MC_sol=np.zeros([num_points,1])
144     for i in range(num_points):
145             if i%10==0:
146                 print('--Monte Carlo Simulations: '+str(i)+'/'+str(num_points),
                        end='\r',flush=True)
147             W=tf.random_normal(shape=[10000000,d],mean=0.0,stddev=tf.sqrt(T),
                    name='W')
148             MC_single=tf.reduce_mean(tf.nn.relu(tf.reduce_max(points[i,:]*tf.exp
                    ((mu_T-0.5*tf.reduce_sum(sigma_T**2,axis=0,keepdims=True))*T+tf.
                    matmul(W,sigma_T)),axis=1,keepdims=True)-k))
149             MC_sol[i,0]=sess.run(MC_single)
150     sess.close()
151     print('--Monte Carlo Simulations completed!')
152     return points,MC_sol
153
154 #evaluate and compare the metrics
155 runtime=1000 #runtime of the optimization algorithm in seconds
156 num_points=100 #number of evaluation points for the metrics
157 metrics_multilevel=list()
158 metrics_standard=list()
159 for seed in range(5):
160     print('\nSeed Nr. '+str(seed))
161     points,solution=MC_approx(seed)
162     print('--Multilevel Learning:')
```

```
    lvl_min, lvl_max=3, 5 #4^(lvl_min) and 4^(lvl_max) are the number of time steps in the coarsest and finest levels respectively
    batch_mult=256 #multiplicator for the batch-size
    metric=fit(lvl_min,lvl_max,batch_mult,runtime,points,solution,seed)
    path=str(seed)+'_multilevel'
    metric.to_pickle(path)
    print('------Saved to '+path+' (pickled pandas DataFrame)')
    metrics_multilevel.append(metric)
    print('--Standard Learning:')
    lvl_min, lvl_max=5, 5
    batch_mult*=(lvl_max-lvl_min+1) #same number of random normal calls
    metric=fit(lvl_min,lvl_max,batch_mult,runtime,points,solution,seed)
    path=str(seed)+'_standard'
    metric.to_pickle(path)
    print('------Saved to '+path+' (pickled pandas DataFrame)')
    metrics_standard.append(metric)
print('\n\nFINISHED!') #open metrics_standard and metrics_multilevel - or use pd.read_pickle(path)
```

Bibliography

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation* (2016), pp. 265–283.

[2] Aliprantis, C., and Border, K. *Infinite Dimensional Analysis: A Hitchhiker's Guide*. Springer, 2007.

[3] Arnold, L. *Stochastic differential equations*. A Wiley-Interscience publication. Wiley, 1974.

[4] Ash, R., and Doléans-Dade, C. *Probability and Measure Theory*. Harcourt/Academic Press, 2000.

[5] Athreya, K., and Lahiri, S. *Measure Theory and Probability Theory*. Springer Texts in Statistics. Springer, 2006.

[6] Beck, C., E, W., and Jentzen, A. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *arXiv:1709.05963* (2017), 56 pages.

[7] Bellman, R. *Dynamic Programming*. Princeton University Press, 2010.

[8] Bertsekas, D. *Nonlinear Programming*. Athena scientific optimization and computation series. Athena Scientific, 2016.

[9] Bertsekas, D. P., and Tsitsiklis, J. N. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization 10*, 3 (2000), 627–642.

[10] Billingsley, P. *Probability and Measure*. Wiley Series in Probability and Statistics. Wiley, 2012.

[11] Bishop, C. M. *Neural networks for pattern recognition*. Oxford university press, 1995.

[12] Bishop, C. M. *Pattern recognition and machine learning*. Information Science and Statistics. Springer, New York, 2006.

[13] Bobrowski, A. *Functional Analysis for Probability and Stochastic Processes: An Introduction*. Cambridge University Press, 2005.

[14] Bölcskei, H., Grohs, P., Kutyniok, G., and Petersen, P. Optimal approximation with sparsely connected deep neural networks. *arXiv:1705.01714* (2017), 36 pages.

[15] Boucheron, S., Lugosi, G., and Massart, P. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. OUP Oxford, 2013.

[16] Caflisch, R. E. Monte carlo and quasi-monte carlo methods. *Acta numerica 7* (1998), 1–49.

[17] Cannarsa, P., and D'Aprile, T. *Introduction to Measure Theory and Functional Analysis*. UNITEXT. Springer International Publishing, 2015.

[18] Caterini, A., and Chang, D. *Deep Neural Networks in a Mathematical Framework*. SpringerBriefs in Computer Science. Springer International Publishing, 2018.

[19] Clevert, D., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv:1511.07289* (2015), 14 pages.

[20] Cliffe, K. A., Giles, M. B., Scheichl, R., and Teckentrup, A. L. Multilevel monte carlo methods and applications to elliptic pdes with random coefficients. *Computing and Visualization in Science 14*, 1 (2011), 3–15.

[21] Cucker, F., and Smale, S. On the mathematical foundations of learning. *Bulletin of the American mathematical society 39*, 1 (2002), 1–49.

[22] Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems 2*, 4 (1989), 303–314.

[23] Da Prato, G., and Zabczyk, J. *Stochastic Equations in Infinite Dimensions*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2014.

[24] Durrett, R. *Brownian Motion and Martingales in Analysis*. Wadsworth & Brooks/Cole Mathematics Series. Wadsworth Advanced Books & Software, 1984.

[25] E, W., Han, J., and Jentzen, A. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics 5*, 4 (2017), 349–380.

[26] Friedman, A. *Stochastic Differential Equations and Applications*. Dover Books on Mathematics. Dover Publications, 2012.

[27] Gall, J. *Brownian Motion, Martingales, and Stochastic Calculus*. Graduate Texts in Mathematics. Springer International Publishing, 2016.

[28] Géron, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017.

[29] Giles, M. B. Multilevel Monte Carlo path simulation. *Oper. Res. 56*, 3 (2008), 607–617.

[30] Giles, M. B. Multilevel monte carlo methods. In *Monte Carlo and Quasi-Monte Carlo Methods 2012*. Springer, 2013, pp. 83–103.

[31] Giles, M. B., and Waterhouse, B. J. Multilevel quasi-monte carlo path simulation. *Advanced Financial Modelling, Radon Series on Computational and Applied Mathematics* (2009), 165–181.

[32] Glasserman, P. *Monte Carlo Methods in Financial Engineering*. Stochastic Modelling and Applied Probability. Springer New York, 2013.

[33] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

[34] Graham, C., and Talay, D. *Stochastic Simulation and Monte Carlo Methods: Mathematical Foundations of Stochastic Simulation*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg, 2013.

[35] Griva, I., Nash, S., and Sofer, A. *Linear and Nonlinear Optimization: Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2009.

[36] Hairer, M., Hutzenthaler, M., and Jentzen, A. Loss of regularity for Kolmogorov equations. *Ann. Probab. 43*, 2 (2015), 468–527.

[37] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York, 2013.

[38] Haykin, S. *Neural Networks and Learning Machines*. Pearson Education, 2011.

[39] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034.

[40] Heinrich, S. Multilevel monte carlo methods. In *International Conference on Large-Scale Scientific Computing* (2001), pp. 58–67.

[41] Hinton, G., Deng, L., Yu, D., Dahl, G. E., r. Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine 29*, 6 (2012), 82–97.

[42] Hofmann, N., Müller-Gronbach, T., and Ritter, K. Optimal approximation of stochastic differential equations by adaptive step-size control. *Math. Comp. 69*, 231 (2000), 1017–1034.

[43] Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks 4*, 2 (1991), 251 – 257.

[44] Hutzenthaler, M., and Jentzen, A. Convergence of the stochastic euler scheme for locally lipschitz coefficients. *Foundations of Computational Mathematics 11*, 6 (2011), 657–706.

[45] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167* (2015), 11 pages.

[46] James, G., Witten, D., Hastie, T., and Tibshirani, R. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York, 2013.

[47] Jentzen, A., Kuckuck, B., Neufeld, A., and von Wurstemberger, P. Strong error analysis for stochastic gradient descent optimization algorithms. *arXiv:1801.09324* (2018), 51 pages.

[48] Keller, A., Heinrich, S., and Niederreiter, H. *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Springer Berlin Heidelberg, 2007.

[49] Khoo, Y., Lu, J., and Ying, L. Solving parametric PDE problems with artificial neural networks. *arXiv:1707.03351* (2017), 12 pages.

[50] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014), 15 pages.

[51] Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems* (2017), pp. 972–981.

[52] Klenke, A. *Probability theory*, second ed. Universitext. Springer, London, 2014. A comprehensive course.

[53] Kloeden, P., Platen, E., and Schurz, H. *Numerical Solution of SDE Through Computer Experiments*. Universitext. Springer Berlin Heidelberg, 2012.

[54] Kloeden, P. E., and Platen, E. *Numerical solution of stochastic differential equations*, vol. 23 of *Applications of Mathematics (New York)*. Springer-Verlag, Berlin, 1992.

[55] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25* (2012), pp. 1097–1105.

[56] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature 521*, 7553 (2015), 436.

[57] Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks 6*, 6 (1993), 861–867.

[58] Maruyama, G. Continuous Markov processes and stochastic equations. *Rend. Circ. Mat. Palermo (2) 4* (1955), 48–90.

[59] Milstein, G. N. *Numerical integration of stochastic differential equations*, vol. 313 of *Mathematics and its Applications*. Kluwer Academic Publishers Group, Dordrecht, 1995. Translated and revised from the 1988 Russian original.

[60] Müller-Gronbach, T., Novak, E., and Ritter, K. *Monte Carlo-Algorithmen*. Springer-Lehrbuch. Springer Berlin Heidelberg, 2012.

[61] Müller-Gronbach, T., and Ritter, K. Minimal errors for strong and weak approximation of stochastic differential equations. In *Monte Carlo and quasi-Monte Carlo methods 2006*. Springer, Berlin, 2008, pp. 53–82.

[62] Nielsen, M. Neural networks and deep learning, 2015. [online; accessed March 05, 2018].

[63] Øksendal, B. *Stochastic differential equations*, sixth ed. Universitext. Springer-Verlag, Berlin, 2003. An introduction with applications.

[64] Petersen, P., and Voigtlaender, F. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *arXiv:1709.05289* (2017), 54 pages.

[65] Pinkus, A. Approximation theory of the mlp model in neural networks. *Acta Numerica 8* (1999), 143–195.

[66] Platen, E., and Rendek, R. Exact scenario simulation for selected multi-dimensional stochastic processes. *Communications on Stochastic Analysis* (2009).

[67] Poggio, T., and Smale, S. The mathematics of learning: Dealing with data. *Notices of the AMS 50*, 5 (2003), 537–544.

[68] Pollard, D. *A User's Guide to Measure Theoretic Probability*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2002.

[69] Protter, P. *Stochastic Integration and Differential Equations*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg, 2013.

[70] Raissi, M. Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations. *arXiv:1804.07010* (2018), 17 pages.

[71] Roman, S. *Advanced Linear Algebra*. Graduate Texts in Mathematics. Springer New York, 2007.

[72] Ruder, S. An overview of gradient descent optimization algorithms. *arXiv:1609.04747* (2016), 14 pages.

[73] Ruszczynski, A. *Nonlinear Optimization*. Princeton University Press, 2011.

[74] Scarselli, F., and Tsoi, A. C. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural Networks 11*, 1 (1998), 15 – 37.

[75] Schilling, R., Partzsch, L., and Böttcher, B. *Brownian Motion: An Introduction to Stochastic Processes*. De Gruyter Textbook. De Gruyter, 2012.

[76] Shamir, O., and Zhang, T. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning* (2013), pp. 71–79.

[77] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature 529* (2016), 484–489.

[78] Sirignano, J., and Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *arXiv:1708.07469* (2017), 16 pages.

[79] Yarotsky, D. Optimal approximation of continuous functions by very deep relu networks. *arXiv:1802.03620* (2018), 21 pages.

[80] Zwillinger, D. *CRC Standard Mathematical Tables and Formulae, 32nd Edition*. Advances in Applied Mathematics. CRC Press, 2011.