Deep Learning-Based Schemes for Solving High-Dimensional Nonlinear BSDEs and Their Uncertainty Quantification

BERGISCHE
UNIVERSITÄT
WUPPERTAL

Dissertation

University of Wuppertal
Faculty of Mathematics and Natural Sciences

submitted by
Lorenc Kapllani, M. Sc.
for the degree of

Doctor of Natural Sciences (Dr. rer. nat.)

supervised by
PD Dr. Long Teng

Doctoral Committee

PD Dr. Long Teng

Prof. Dr. Michael Günther

Prof. Dr. Peter Zaspel

Prof. Dr. Thomas Kruse

Prof. Dr. Matthias Ehrhardt

PD Dr. Matthias Rottmann

Wuppertal, February 25, 2025

Dedication

To my beloved family

Declaration

I hereby declare that this dissertation, titled "Deep Learning-Based Schemes for Solving High-Dimensional Nonlinear BSDEs and Their Uncertainty Quantification", represents my own work conducted under the supervision of PD Dr. Long Teng in the Applied and Computational Mathematics group, Faculty of Mathematics and Natural Sciences, at the University of Wuppertal. The research presented in this dissertation has been carried out by myself under the supervision of PD Dr. Long Teng. One paper included in this dissertation is the result of our collaboration with PD Dr. Matthias Rottmann.

I further declare that this dissertation has not been submitted for any other degree or qualification at any other university or institution.

All sources of information used in this dissertation have been duly acknowledged. Any assistance received from colleagues, including technical advice, discussions, or provision of facilities, has been acknowledged in the acknowledgment section of this dissertation.

Wuppertal, February 25, 2025

Lorenc Kapllani

Acknowledgements

I would like to express my deepest gratitude to all those who have supported me throughout my PhD journey in the Applied and Computational Mathematics group at the University of Wuppertal.

First and foremost, I am profoundly grateful to my supervisor, PD Dr. Long Teng, for his unwavering guidance, insightful feedback, and continuous encouragement. His expertise and dedication have been invaluable to my research and development as a scholar. Even during the most challenging times, our problem-solving approach helped us navigate obstacles and find solutions, making the journey manageable and rewarding.

I would also like to extend my heartfelt thanks to Prof. Dr. Matthias Ehrhardt and Prof. Dr. Michael Günther for their academic support and the enriching learning environment they provided. Special thanks go to PD Dr. Matthias Rottmann, with whom I had the pleasure of collaborating on a research article that forms a part of this dissertation. His collaboration and expertise greatly enhanced the quality of this work.

I am deeply grateful to the esteemed colleagues in our group for their guidance and support: Prof. Dr. Thomas Kruse, PD Dr. Jörg Kienitz, and PD Dr. Andreas Bartel. Your valuable insights and encouragement have been instrumental in my academic journey.

To my fellow PhD students and colleagues, thank you for the technical discussions, your kindness, and the many coffee breaks that provided much-needed rest and companionship. I appreciate Dr. Julia Ackermann, Anna Clevenhaus, Sarah Berkhahn, Dr. Friedemann Klaß, Konrad Kleinberg, Dr. Tatiana Kossaczka, Dr. Michelle Muniz, Prof. Dr. Pavel Petrov, Dr. Hanna Schilar, Kevin Schäfers and Daniel Walsken for the informal meetings, discussions, and the times we spent together during game-nights and going out for a drink. Our shared moments together with our esteemed professors, especially the numerous cakes for various occasions and just because, created a wonderful and supportive atmosphere. We also had a great time during our department events such as Christmas dinners and hikes.

Lastly, I owe my deepest gratitude to my family. Your unwavering support, love, and encouragement have been the bedrock of my strength throughout this challenging journey. Thank you for believing in me and standing by my side every step of the way. To my girlfriend, your love, patience, and understanding have been a constant source of comfort and motivation. Your belief in me and your encouragement during the highs and lows of this journey have meant a lot to me. Thank you for always being there and for your endless support.

Contents

List of Figures

List of Tables

List of Publications

1. L. KAPLLANI, *The Effect of the Number of Neural Networks on Deep Learning Schemes for Solving High Dimensional Nonlinear Backward Stochastic Differential Equations*, in M. Ehrhardt, M. Günther (eds.) Progress in Industrial Mathematics at ECMI 2021. ECMI 2021. Mathematics in Industry, vol. 39, Springer, Cham, 2022, https://doi.org/10.1007/978-3-031-11818-0_10.

2. L. KAPLLANI AND L. TENG, *Deep learning algorithms for solving high-dimensional nonlinear backward stochastic differential equations*, Discrete Contin. Dyn. Syst. - B, 29 (4), 2024, pp. 1695–1729, https://doi.org/10.3934/dcdsb.2023151.

3. L. KAPLLANI, L. TENG AND M. ROTTMANN, *Uncertainty quantification for deep learning-based schemes for solving high-dimensional backward stochastic differential equations*, Int. J. Uncertain. Quantif., 15 (3), 2025, pp. 55–94, https://doi.org/10.1615/Int.J.UncertaintyQuantification.2024053491.

4. L. KAPLLANI AND L. TENG, *A backward differential deep learning-based algorithm for solving high-dimensional nonlinear backward stochastic differential equations*, https://arxiv.org/abs/2404.08456, 2025, **accepted to be published in IMA J. Numer. Anal. with DOI: 10.1093/imanum/draf022**.

List of Presentations

1. (Online) European Consortium for Mathematics in Industry (ECMI), Wuppertal, Germany, April 2021.
2. 4th International Conference on Computational Finance (ICCF), Wuppertal, Germany, June 2022.
3. 16th German Probability and Statistics Days (GPSD), Essen, Germany, March 2023.
4. 15th European Summer School in Financial Mathematics, Delft, Netherlands, September 2023.
5. Central-European Conference on Scientific Computing (ALGORITMY), High Tatra Mountains, Slovakia, March 2024.

Abbreviations and List of Symbols

Abbreviations

(D)LDBSDE (Differential) Local deep BSDE scheme

(D)NN (Deep) neural network

(FB)SDE (Forward-backward) stochastic differential equation

(La)DBSDE (Locally additive) deep BSDE scheme

(P)C-LR (Piecewise) constant learning rate

(R)MSE (Root) mean squared error

ABM Arithmetic Brownian motion

AD Automatic differentiation

DBDP Deep backward dynamic programming scheme

DLBDP Differential learning backward dynamic programming scheme

GBM Geometric Brownian motion

HJB Hamilton-Jacobi-Bellman

i.i.d Independent and identically distributed

ITM, ATM, OTM In, at and out of the money

LSTM Long short term memory network

OSM One step Malliavin scheme

PDE Partial differential equation

RBSDE Reflected BSDE

ReLU Rectifier function

RNN Recurrent neural network

s.t. such that

SGD Stochastic gradient descent

STD Standard deviation

UQ Uncertainty quantification

w.r.t. with respect to

Roman symbols

A, a Affine transformations in DNN layer, drift function of the SDE

B, b Batch size, diffusion function of the SDE

C, c Positive constant

D, d Malliavin derivative, dimension

F Smooth bounded function

f, g Driver function of the BSDE, function at the terminal time of the BSDE

h Function in the Hilbert space

K, L Strike price, number of hidden layers (slight abuse of notation: used also as a Lipschitz constant or Hilbert space)

M, N Sample size for UQ model, number of discrete time points

P, p Total number of parameters in a DNN, probabilistic distribution

Q Number of runs (trainings) of the algorithm

$R, (R1, R2)$ Interest rate, interest rates for lending and borrowing

S Stock price

T, t Terminal time (maturity), time variable

u, v Function solving the PDE, vector

W Brownian motion

x Space variable

X, Y, Z Stochastic processes of the BSDE

Greek symbols

α Learning rate

β Empirical convergence rate

Δ, δ Uniform discretization of time interval $[0, T]$, dividend rate

ϵ, ε Square root of the expected squared error, expected squared error

η Number of neurons per hidden layer

Γ, γ Stochastic process related to Hessian matrix of the solution, function related to the Γ process

κ Index for the optimization step

λ Regularization parameter

μ Population mean

ν Positive constant

Ω, ω Sample space, sample from sample space Ω

Φ, ϕ, φ Standard normal cumulative distribution function, DNN functional form, smooth function bounded or has polynomial growth

ρ, ϱ Correlation coefficient, activation function

σ, ς Population STD, Spearman's rank correlation

τ Computation time of the algorithm (seconds)

Θ, θ Set of network parameters, total weights in a DNN

ξ Random variable

Subscripts

$+$ Positive values

$\mathfrak{b}, \mathfrak{p}$ Reference to bounded functions, functions with polynomial growth. Similarly when referencing other functions, e.g. a, b, f

$\mathfrak{i}$ Index, e.g. for $\mathfrak{i}$-th UQ model

$\mathcal{F}_t$ $\mathcal{F}_t$-measurable random variable

i, j Index for sample of the dataset of UQ model, sample (Brownian motion)

$l, (n, m), q$ Index for layer in a DNN, discrete time domain, trained algorithm

t Reference to a continuous time variable, e.g. X_t

x, y, z Reference to derivative w.r.t. variable x, y, z

Superscripts

$1, 2$ Reference to a property, e.g. C^1 continuous and one time differentiable

Δ Discrete version, e.g. $\left(X^{\Delta}, Y^{\Delta}, Z^{\Delta}\right)$ discrete version of processes (X, Y, Z)

$\mathfrak{l}$ Order of continuously differentiable functions

$\star$ Optimal solution

min, max For boundary values in the uniform distribution

nr Normalized data

r Relative metric

$train, valid, test$ Reference to training, validation or testing sample

y Reference to process Y, similarly for other processes (e.g. z_k for the k-th component of process Z)

Other symbols

$\check{\cdot}$ A transformed variable, e.g., $\check{X}_t = \ln(X_t)$ ln-transformation of X_t

$\Delta t, \Delta W_n$ Time increment, increment of the Brownian motion at discrete time interval $[t_n, t_{n+1}]$

$\ell, \boldsymbol{\ell}$ Binary label, multi-label

$\hat{\cdot}$ Estimated parameter, e.g., $\hat{\theta}$ estimated parameters of θ or $\hat{\sigma}$ for σ

$\mathfrak{H}$ Hidden state

$\mathfrak{K}$ Total number of optimization steps

$\mathbb{C}, \mathbb{D}$ Space of continuous and differentiable functions, space of Malliavin differentiable random variables

$\mathbb{E}$ Expectation

$\mathbb{H}$ Space of progressively measurable stochastic processes

$\mathbb{L}$ Space of measurable random variables

$\mathbb{N}$ Space of integers > 0

$\mathbb{P}$ Probability measure

$\mathbb{R}, \mathbb{S}$ Space of real numbers, space of continuous and progressively measurable stochastic processes

$\mathbf{0}_d, \mathbf{0}_{d,d}$ Vector, matrix of zeros of size d, $d \times d$

$\mathbf{1}_d, \mathbf{1}_{d,d}$ Vector, matrix of ones of size d, $d \times d$

$\mathbf{I}_d$ Identity matrix of size $d \times d$

$\mathbf{L}$ Loss function

$\mathbf{N}$ A list of N values

$\mathbf{x}, (\mathbf{y}, \mathbf{z})$ Random variable as input for DNN, label and the gradient related to the label

$\mathbf{X}, \mathbf{DX}$ Short notation for (X, Y, Z), its Malliavin derivative

$\mathbf{Y}$ Short notation for (Y_0, Z)

$\mathcal{A}$ Optimization algorithm, e.g. Adam optimizer

$\mathcal{B}, \mathcal{W}$ Bias vector, weight matrix of the DNN

$\mathcal{D}, \check{\mathcal{D}}$ Generated dataset for UQ model using DBSDE algorithm, LaDBSDE algorithm

$\mathcal{E}$ Total approximation error

$\mathcal{N}$,$\mathcal{U}$ Normal, uniform distribution

$\mathcal{S}$ Space of smooth random variables

$\nabla_x u, \mathrm{Hess}_x\, u$ Gradient, Hessian matrix of function u w.r.t. x

$\mathfrak{n}$	Number of problem parameters in the BSDE
$\bar{\cdot}$	Mean, e.g., $\overline{\tilde{\varepsilon}}$ mean of MSE values
$\mathfrak{q}$	Dimension
$\tilde{\cdot}$	Empirical measure, e.g. $\tilde{\mu}$ sample mean or $\tilde{\mathbf{L}}$ empirical loss
acc	Accuracy score
ep	Number of epochs
MRR	Mean reciprocal rank
pos	Position where a true label is found in a list of predicted labels
$rank$	Assigned rank to a list

Summary

Backward stochastic differential equations (BSDEs) are essential tools for modeling problems across various scientific domains, including finance, economics, physics, etc., due to their connection to partial differential equations (PDEs) through the well-known (nonlinear) Feynman-Kac formula. In case of option pricing, the solution pair of a BSDE represents the price and delta-hedging of an option. Under the Black-Scholes framework, such a BSDE is linear and the solution is given in a closed form. However, in most practical scenarios, BSDEs cannot be explicitly solved. Hence, advanced numerical techniques to approximate their solution become desired.

The classical numerical schemes, e.g. Fourier or cubature methods on spatial discretization suffer from the "curse of dimensionality", where the computational cost increases exponentially with the problems dimensionality. In case of option pricing, the BSDE exhibits the dimensionality with the number of underlying financial assets under consideration. Recently, several works have introduced innovative deep learning-based methods utilizing deep neural networks (DNNs) for solving high-dimensional nonlinear BSDEs.

In this thesis, we are concerned with developing efficient numerical schemes for solving high-dimensional nonlinear BSDEs using deep learning techniques. Due to the inherent uncertainty and potential errors in decision-making processes involving deep learning models, this thesis also considers uncertainty quantification (UQ) for these schemes.

The thesis is divided in three parts. The first part, outlined in Chapter 1 and based on our research in [57, 60], surveys the existing deep learning BSDE schemes, categorizing them into forward and backward deep learning schemes, as they are formulated forward or backward in time. The pioneering (forward) scheme, known as the deep BSDE scheme, has been shown in the literature to have several disadvantages. These include the potential to get stuck in poor local minima or even diverge, especially for a complex solution structure and a long terminal time. Furthermore, it tends to yield much better approximations of the BSDE at the initial time than at the other time points. To address these issues, we propose a novel forward scheme that overcomes them. To demonstrate the performance of the new algorithm, we present several high-dimensional nonlinear BSDEs, including examples from pricing problems in finance.

The second part of the thesis, presented in Chapter 2, is based on our research in [59]. The deep learning BSDE schemes often struggle to provide highly accurate gradient approximations, which are crucial for financial applications, especially concerning hedging strategies for option contracts. Therefore, in the second part, we propose a novel approach based on differential deep learning, where the DNN models are trained not only on the inputs and labels but also the differentials of the corresponding labels. Motivated by the efficiency of differential deep learning in approximating labels and their derivatives with respect to (w.r.t.) inputs, we introduce both backward and forward differential deep learning algorithms. We demonstrate, both theoretically and numerically, that the backward algorithm is more efficient compared to other contemporary backward deep learning methodologies. Additionally, we show that differential deep learning can

be extended to the forward deep learning schemes.

The third part, detailed in Chapter 3 based on our research in [61], focuses on the UQ of deep learning BSDE schemes. As these schemes are prone to noise and model errors, assessing its reliability before it can be used in practice is critical. In the context of pricing and hedging different contracts in finance, companies may incur significant financial losses due to poor judgments. As an example, we consider the forward deep learning schemes to study UQ and introduce a UQ model. Through various experiments, we demonstrate the reliability of the UQ model in estimating uncertainty for the class of forward deep learning schemes. The proposed UQ model is applicable to other deep learning BSDE schemes. It is worth highlighting that the work in this chapter is the first one about the development of a UQ model for deep learning BSDE schemes.

In Chapter 4, we conclude the thesis by summarizing the findings and outlining potential avenues for future research.

Chapter 1

Deep Learning BSDE Schemes

The aim of this chapter is to provide a brief overview of some basics related to BSDEs along with notable results about them. Moreover, we describe neural network (NN) architectures and deep learning techniques. Following this, we discuss two classes of deep learning schemes (forward and backward). The pioneering forward scheme, known as the deep BSDE method, has been reported in the literature to encounter several challenges, including getting stuck to a poor local minima or even diverging, especially for a complex solution structure and a long terminal time. Additionally, this scheme tends to produce more accurate approximations at the initial time than at later time points. To address these limitations, we propose a novel forward method. Through several numerical experiments, we show the improved performance of our new scheme compared to the deep BSDE scheme and other contemporary forward deep learning schemes. The chapter is based on our research conducted in [57, 60].

1.1 Introduction

BSDEs are important tools used to model problems in scientific fields due to their connections to PDEs and stochastic control problems via the nonlinear Feynman–Kac formula [62]. As an illustrative example of their applications in finance, it was demonstrated in [62] that the price and delta-hedging of an option can be represented by a BSDE. Such an approach via a BSDE has a couple of advantages when compared with the usual one of considering the associated PDE. Firstly, the delta-hedging strategy is inclusive in the BSDE solution. Secondly, many market models can be presented in terms of BSDEs, ranging from the Black-Scholes model to more advanced ones such as local volatility models [69], stochastic volatility models [26], jump-diffusion models [25], defaultable options [4], and many others. Thirdly, BSDEs can also be used in incomplete markets [62]. Furthermore, using BSDEs eliminates the need to switch to the so-called risk-neutral measure. Therefore, BSDEs represent a more intuitive and understandable approach for option pricing and hedging.

In this thesis, we consider the high-dimensional nonlinear decoupled forward-backward stochastic differential equation (FBSDE)

$$\begin{cases} X_t &= x_0 + \int_0^t a\left(s, X_s\right)\, ds + \int_0^t b\left(s, X_s\right)\, dW_s, \\ Y_t &= g\left(X_T\right) + \int_t^T f\left(s, \mathbf{X}_s\right)\, ds - \int_t^T Z_s\, dW_s, \end{cases} \forall\, t \in [0,T] \tag{1.1}$$

where $\mathbf{X}_t := (X_t, Y_t, Z_t)$, $W_t = \left(W_t^1, \ldots, W_t^d\right)^\top$ is a d-dimensional Brownian motion, $a : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$, $b : [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d\times d}$, $f : [0,T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{1\times d} \to \mathbb{R}$ is the driver function

and $g : \mathbb{R}^d \to \mathbb{R}$ is the terminal condition which depends on the final value X_T of the forward stochastic differential equation (SDE). Hence, the randomness in the BSDE is driven by the forward SDE. Usually, the coupled FBSDE is referred to as a FBSDE. Hence, to avoid confusion, we refer to the decoupled FBSDE (1.1) as a BSDE. We shall work under the standard well-posedness assumptions of [83] to ensure the existence of a unique solution pair of (1.1).

Under the Black-Scholes framework, the BSDE is linear and the solution is given in a closed form. However, in most practical scenarios, BSDEs cannot be explicitly solved. For instance, the Black-Scholes model under different interest rates for lending and borrowing [10] leads to a nonlinear BSDE for which finding an analytical solution becomes challenging. Hence, advanced numerical techniques to approximate their solutions are needed. In recent years, various numerical methods have been proposed for solving BSDEs, e.g., [11, 101, 35, 72, 103, 9, 74, 105, 34, 18, 104, 88, 87, 95, 96] and many others, see, e.g. [17] for a nice overview. However, most of them are not suitable for tackling high-dimensional BSDEs due to the well-recognized challenge known as the "curse of dimensionality". The computational cost associated with solving high-dimensional BSDEs grows exponentially with the increase in dimensionality. Some of the most important equations are naturally formulated in high dimensions, e.g. the Black-Scholes equation for option pricing that exhibits the dimensionality of the BSDE with the number of underlying financial assets. Some techniques such as parallel computing using GPU computing [36, 58] or sparse grid methods [100, 27, 13] have proven effective in solving only moderately dimensional BSDEs within a reasonable computation time.

Recently, different approaches have been proposed to solve high-dimensional BSDEs, which can be classified into three main categories. The first category involves multilevel Monte Carlo methods based on Picard iteration [23, 8, 47, 48, 24, 45, 46]. The second category includes tree-based methods [18, 93, 94], and the third one consists of deep learning-based methods using DNNs. The first deep learning-based scheme called the deep BSDE (we refer to it as the DBSDE scheme), was introduced in [22, 38]. The authors conducted numerical experiments with various examples, demonstrating the effectiveness of their proposed algorithm in high-dimensional settings. It proved proficient in delivering both accurate approximations of the solution and computational efficiency. Therefore, the method opened the door to solving BSDEs in hundreds of dimensions in a reasonable amount of time. Several articles have been published after the original publication of the DBSDE method, some adjusting, reformulating, or extending the algorithm [97, 12, 28, 52, 44, 67, 84, 6, 15, 73, 53, 85, 92, 30, 32, 54, 57, 1, 3, 33, 86, 60, 59], while others focused on error analysis [39, 56, 77] and uncertainty quantification [61].

The DBSDE scheme formulates the BSDE as a global optimization problem, where the process Z is parameterized using DNNs and the process Y is approximated from the Euler-Maruyama method applied to BSDE (1.1). The parameters of DNNs are then optimized using the stochastic gradient descent (SGD) algorithm on a terminal loss produced by the forward Euler-Maruyama discretization of the BSDE. Since the DBSDE scheme operate forward in time, we refer to it as a forward deep learning scheme. It has been pointed out in the literature that the DBSDE method suffers from the following disadvantages: 1) It can be stuck in poor local minima or even diverge, especially for a complex solution structure and a long terminal time, see, e.g., [44]. 2) It is capable of achieving much better approximations of the BSDE (1.1) at the initial time than at the other time points, although the solution of the BSDE is approximated pathwise along $[0, T]$, see [86]. Note that we refer as "good" local minima the one which may perform as well as a global minima, and can exist in considerable numbers given a NNs loss function. Otherwise we call it "poor" local minima. Using a single DNN for the process Z in the DBSDE scheme instead of multiple ones can only overcome the second disadvantage, as we demonstrated in [57]. Even other contemporary forward deep learning schemes such as [86] does not overcome both of these disadvantages.

Another class of deep learning schemes for solving high-dimensional BSDEs are the backward deep learning schemes, with the first algorithm developed in [44]. In this approach, the solution of (1.1) is approximated using two DNNs, and their parameters are backwardly optimized based on local optimization problems at each discrete time point. The scheme in [44] does not suffer from the disadvantages of the DBSDE scheme. However, it is more computationally costly than the forward deep learning schemes due to solving multiple local optimization problems.

Motivated by the disadvantages of the forward deep learning schemes [22, 86], we present a novel forward scheme in the first part of this thesis. The essential concept is to formulate the problem as a global optimization with local loss functions including the terminal condition. Our formulation is obtained by using the Euler-Maruyama discretization of the integrals and iterating it with the terminal condition, i.e., iterative time discretization, this might be seen also as a multi-step time discretization. The algorithm estimates the unknown solution (the process Y) using a DNN and its gradient (the process Z) via automatic differentiation (AD). These approximations are performed from the global minimization of the local loss functions defined at each time point from the iterative time discretization.

In [86], the author has introduced a similar strategy based on local loss functions arising from Euler-Maruyama discretization at each time interval, with the terminal condition included as an additional term in the loss function, i.e., the proposed algorithm attempt to match the discretized dynamics of the BSDE at each time interval. This approach achieves a good approximation of the processes Y and Z not only at the initial time but also at each time layer. Hence, it can overcome the second disadvantage of the DBSDE scheme. However, it still suffers from the first disadvantage, which will be demonstrated in our numerical experiments in Section 1.5. Note that it does not help the SGD algorithm in [22, 86] to converge to a good local minima just by considering another network architecture. For instance, the recurrent neural network (RNN) type architectures are specialized for learning long complex sequences. However, it has been pointed out in [44] that using RNN type architectures in the DBSDE scheme does not improve the results. Even when used in [86], the RNN architecture does not improve the results, which will be shown in Section 1.5. In our new formulation, using local losses including the terminal condition helps the SGD algorithm to converge to a good local minima.

This chapter is structured as follows. In Section 1.2, we introduce some preliminaries about BSDEs including NN architectures (DNNs and RNNs) and deep learning techniques. The forward deep learning schemes [22, 86] and our novel forward scheme are presented in Section 1.3. Section 1.4 describes the backward deep learning scheme [44], which is considered in more detail in the second part of the thesis. Section 1.5 demonstrates through various high-dimensional examples including pricing problems the improved performance of our new algorithm compared to other contemporary forward deep learning schemes [22, 86]. Finally, Section 1.6 concludes this chapter.

1.2 Preliminaries

1.2.1 Spaces and notation

Let $(\Omega, \mathcal{F}, \mathbb{P}, \{\mathcal{F}_t\}_{0\leq t\leq T})$ be a complete, filtered probability space. In this space a standard d-dimensional Brownian motion $\{W_t\}_{\leq t\leq T}$ is defined, such that (s.t.) the filtration $\{\mathcal{F}_t\}_{0\leq t\leq T}$ is the natural filtration of W_t. In what follows, all equalities concerning $\mathcal{F}_t$-measurable random variables are meant in the $\mathbb{P}$-a.s. sense and all expectations (unless stated otherwise) are meant under probability measure $\mathbb{P}$. Throughout the whole paper, we rely on the following notations

- $x \in \mathbb{R}^d$ as a column vector. $x \in \mathbb{R}^{1\times d}$ as a row vector.
- $|x|$ for the Frobenius norm of any $x \in \mathbb{R}^{d\times \mathfrak{q}}$. In the case of scalar and vector inputs, these coincide with the standard Euclidean norm.
- $\mathbb{S}^2\left([0,T]\times\Omega;\mathbb{R}^{d\times\mathfrak{q}}\right)$ for the space of continuous and progressively measurable stochastic processes $X:[0,T]\times\Omega\to\mathbb{R}^{d\times\mathfrak{q}}$ s.t. $\mathbb{E}\left[\sup_{0\le t\le T}|X_t|^2\right]<\infty$.
- $\mathbb{H}^2\left([0,T]\times\Omega;\mathbb{R}^{d\times\mathfrak{q}}\right)$ for the space of progressively measurable stochastic processes $Z:[0,T]\times\Omega\to\mathbb{R}^{d\times\mathfrak{q}}$ s.t. $\mathbb{E}\left[\int_0^T|Z_t|^2\,dt\right]<\infty$.
- $\mathbb{L}^2_{\mathcal{F}_t}\left(\Omega;\mathbb{R}^{d\times\mathfrak{q}}\right)$ for the space of $\mathcal{F}_t$-measurable random variable $\xi:\Omega\to\mathbb{R}^{d\times\mathfrak{q}}$ s.t. $\mathbb{E}\left[|\xi|^2\right]<\infty$.
- $L^2\left([0,T];\mathbb{R}^{\mathfrak{q}}\right)$ for the Hilbert space of deterministic functions $h:[0,T]\to\mathbb{R}^{\mathfrak{q}}$ s.t. $\int_0^T|h(t)|^2\,dt<\infty$.
- $\nabla_x f := \left(\frac{\partial f}{\partial x_1},\dots,\frac{\partial f}{\partial x_d}\right)\in\mathbb{R}^{1\times d}$ for the gradient of scalar-valued multivariate function $f(t,x,y,z)$ w.r.t. $x\in\mathbb{R}^d$, and analogously for $\nabla_y f\in\mathbb{R}$ and $\nabla_z f\in\mathbb{R}^{1\times d}$ w.r.t. $y\in\mathbb{R}$ and $z\in\mathbb{R}^{1\times d}$, respectively. Similarly, we denote the Jacobian matrix of a vector-valued function $u:\mathbb{R}^d\to\mathbb{R}^{\mathfrak{q}}$ by $\nabla_x u\in\mathbb{R}^{\mathfrak{q}\times d}$.
- $\mathrm{Hess}_x\,u\in\mathbb{R}^{d\times d}$ the Hessian matrix of a function $u:\mathbb{R}^d\to\mathbb{R}$.
- $\mathbb{C}^{\mathfrak{l}}_{\mathfrak{b}}\left(\mathbb{R}^d;\mathbb{R}^{\mathfrak{q}}\right)$ and $\mathbb{C}^{\mathfrak{l}}_{\mathfrak{p}}\left(\mathbb{R}^d;\mathbb{R}^{\mathfrak{q}}\right)$ for the set of $\mathfrak{l}$-times continuously differentiable functions $\varphi:\mathbb{R}^d\to\mathbb{R}^{\mathfrak{q}}$ s.t. all partial derivatives up to order $\mathfrak{l}$ are bounded or have polynomial growth, respectively.
- $\Delta=\{t_0,t_1,\dots,t_N\}$ is the time discretization of $[0,T]$ with $t_0=0<t_1<\ldots<t_N=T$, $\Delta t_n=t_{n+1}-t_n$ and $|\Delta|:=\max_{0\le n\le N-1}t_{n+1}-t_n$.
- $\mathbb{E}_n[Y]:=\mathbb{E}[Y|\mathcal{F}_{t_n}]$ for the conditional expectation w.r.t. the natural filtration, given the time partition Δ.
- $x^\top\in\mathbb{R}^{\mathfrak{q}\times d}$ for the transpose of any $x\in\mathbb{R}^{d\times\mathfrak{q}}$.
- $\mathrm{Tr}[x]$ for the trace of any $x\in\mathbb{R}^{d\times d}$.
- $\mathbf{0}_{d,d}$, $\mathbf{1}_{d,d}$ for $\mathbb{R}^{d\times d}$ matrices of all zeros and ones, respectively.

1.2.2 BSDEs

We recall some results on BSDE which are relevant for this work. For the functions in BSDE (1.1), we hierarchically structure the properties that they are assumed to fulfill.

AX1. *The initial condition $x_0\in\mathbb{L}^2_{\mathcal{F}_0}\left(\Omega;\mathbb{R}^d\right)$ and a,b satisfy a linear growth condition in x, i.e.,*

$$|a(t,x)|+|b(t,x)|\le C(1+|x|),$$

$\forall\, t\in[0,T], x\in\mathbb{R}^d$ and some constant $C>0$. Furthermore, a,b are uniformly Lipschitz continuous in the spatial variable, i.e.,

$$|a(t,x_1)-a(t,x_2)|+|b(t,x_1)-b(t,x_2)|\le L_{a,b}|x_1-x_2|$$

$\forall\, t\in[0,T], x_1,x_2\in\mathbb{R}^d$, for some constant $L_{a,b}>0$.

AX2. *Assumption AX1 holds. Moreover,* $a(t,0)$, $b(t,0)$ *are uniformly bounded* $\forall\ 0 \leq t \leq T$ *and* $a \in \mathbb{C}_{\mathfrak{b}}^{0,1}\left([0,T]\times\mathbb{R}^d;\mathbb{R}^d\right)$, $b \in \mathbb{C}_{\mathfrak{b}}^{0,1}\left([0,T]\times\mathbb{R}^d;\mathbb{R}^{d\times d}\right)$.

AY1. *The function* $f(t,x,y,z)$ *is uniformly Lipschitz continuous w.r.t* y *and* z, *i.e.*

$$|f(t,x,y_1,z_1) - f(t,x,y_2,z_2)| \leq L_f\left(|y_1-y_2| + |z_1-z_2|\right),$$

$\forall\,(t,x,y_1,z_1)$ *and* $(t,x,y_2,z_2) \in [0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^{1\times d}$, *for some constant* $L_f > 0$. *Moreover,* f, g *satisfy a quadratic growth condition in* x,*i.e.,*

$$|f(t,x,y,z)| + |g(x)| \leq C\left(1+|x|^2\right),$$

$\forall\,(t,x,y,z) \in [0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^{1\times d}$ *for some constant* $C > 0$.

AY2. *Assumption AY1 holds. Moreover,* $f \in \mathbb{C}_{\mathfrak{b}}^{0,1,1,1}\left([0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^{1\times d};\mathbb{R}\right)$ *and* $g \in \mathbb{C}_{\mathfrak{b}}^{1}\left(\mathbb{R}^d;\mathbb{R}\right)$.

In the following theorem, we state the well-known result on SDEs.

Theorem 1.2.1. *(Moment Estimates for SDEs [65])*
Assume that Assumption AX1 holds. Then the SDE in (1.1) has a unique strong solution $\{X_t\}_{0\leq t\leq T} \in \mathbb{S}^2\left([0,T]\times\Omega;\mathbb{R}^d\right)$ *and the following moment estimates hold:*

$$\mathbb{E}\left[\sup_{0\leq t\leq T}|X_t|^2\right] \leq C, \quad \mathbb{E}\left[\sup_{s\leq r\leq t}|X_r - X_s|^2\right] \leq C\,|t-s|,$$

where constant C *depends only on* T, d.

The well-posedness of the BSDE (1.1) is guaranteed by Assumption AY1. The following theorem guarantees the existence of a unique solution triple of (1.1).

Theorem 1.2.2. *(Properties of BSDEs [62])*
Assume that Assumptions AX1 and AY1 holds. Then the BSDE (1.1) admits a unique solution triple $\{X_t,Y_t,Z_t\}_{0\leq t\leq T} \in \mathbb{S}^2\left([0,T]\times\Omega;\mathbb{R}^d\right)\times\mathbb{S}^2\left([0,T]\times\Omega;\mathbb{R}\right)\times\mathbb{H}^2\left([0,T]\times\Omega;\mathbb{R}^{1\times d}\right)$.

Another result that is relevant for this work is the path regularity result of the processes Y and Z, which we state in the following theorem.

Theorem 1.2.3. *(Path regularity [49])*
Under Assumptions AX2 and AY2 the BSDE (1.1) admits a unique solution triple $\{X_t,Y_t,Z_t\}_{0\leq t\leq T} \in \mathbb{S}^2\left([0,T]\times\Omega;\mathbb{R}^d\right)\times\mathbb{S}^2\left([0,T]\times\Omega;\mathbb{R}\right)\times\mathbb{H}^2\left([0,T]\times\Omega;\mathbb{R}^{1\times d}\right)$. *Moreover, the following holds true:*

(i) There exist a constant $C > 0$ *s.t.* $\forall\, 0 \leq s \leq t \leq T$

$$\mathbb{E}\left[\sup_{s\leq r\leq t}|Y_r - Y_s|^2\right] \leq C\,|t-s|$$

(ii) There exist a constant $C > 0$ *s.t. for any partition* Δ *of* $[0,T]$

$$\sum_{n=1}^{N-1}\mathbb{E}\left[\int_{t_n}^{t_{n+1}}|Z_t - Z_{t_n}|^2\,dt\right] \leq C\,|\Delta|.$$

An important property of BSDEs is that they provide a probabilistic representation for the solution of a specific class of PDEs given by the nonlinear Feynman–Kac formula. Consider the semi-linear parabolic PDE

$$\frac{\partial u(t,x)}{\partial t} + \nabla_x u(t,x)\, a(t,x) + \frac{1}{2}\,\mathrm{Tr}\left[bb^\top \,\mathrm{Hess}_x\, u(t,x)\right] + f\left(t,x,u,\nabla_x u\, b\right)(t,x) = 0, \tag{1.2}$$

for all $(t,x) \in ([0,T] \times \mathbb{R}^d)$ and the terminal condition $u(T,x) = g(x)$. Assume that (1.2) has a classical solution $u \in \mathbb{C}_{\mathfrak{b}}^{1,2}\left([0,T]\times\mathbb{R}^d;\mathbb{R}\right)$ and the aforementioned standard Lipschitz assumptions of (1.1) are satisfied. Then the solution of (1.1) can be represented $\mathbb{P}$-a.s. by

$$Y_t = u\left(t,X_t\right), \quad Z_t = \nabla_x u\left(t,X_t\right) b\left(t,X_t\right) \quad \forall\, t \in [0,T)\,. \tag{1.3}$$

To approximate the function u (and its gradient), NNs are considered due to the approximation capability in high dimensions.

1.2.3 NNs as function approximators

NNs rely on the composition of simple functions, but provide an efficient way to approximate unknown functions.

We start introducing feedforward NNs or DNNs. Let $d, \mathfrak{q} \in \mathbb{N}$ be the input and output dimensions, respectively. We fix the global number of layers as $L+2$, $L \in \mathbb{N}$ the number of hidden layers each with $\eta \in \mathbb{N}$ neurons. The first layer is the input layer with d neurons and the last layer is the output layer with $\mathfrak{q}$ neurons. A DNN is a function $\phi(\mathbf{x};\theta) : \mathbb{R}^d \to \mathbb{R}^{\mathfrak{q}}$ composed of a sequence of simple functions, which can be expressed in the following form

$$\mathbf{x} \in \mathbb{R}^d \longmapsto A_{L+1}(\cdot;\theta(L+1)) \circ \varrho \circ A_L(\cdot;\theta(L)) \circ \varrho \circ \ldots \circ \varrho \circ A_1(\mathbf{x};\theta(1)) \in \mathbb{R}^{\mathfrak{q}},$$

where $\theta := (\theta(1),\ldots,\theta(L+1)) \in \mathbb{R}^P$ and P is the total number of network parameters, $x \in \mathbb{R}^d$ is called an input vector. Moreover, $A_l(\cdot;\theta(l)), l = 1,2,\ldots,L+1$ are affine transformations: $A_1(\cdot;\theta(1)) : \mathbb{R}^d \to \mathbb{R}^\eta$, $A_l(\cdot;\theta(l)), l = 2,\ldots,L : \mathbb{R}^\eta \to \mathbb{R}^\eta$ and $A_{L+1}(\cdot;\theta(L+1)) : \mathbb{R}^\eta \to \mathbb{R}^{\mathfrak{q}}$, represented by

$$A_l(v;\theta(l)) = \mathcal{W}_l v + \mathcal{B}_l, \quad v \in \mathbb{R}^{\eta_{l-1}},$$

where $\theta(l) := (\mathcal{W}_l, \mathcal{B}_l)$, $\mathcal{W}_l \in \mathbb{R}^{\eta_l \times \eta_{l-1}}$ is the weight matrix and $\mathcal{B}_l \in \mathbb{R}^{\eta_l}$ is the bias vector with $\eta_0 = d, \eta_{L+1} = \mathfrak{q}, \eta_l = \eta$ for $l = 1,\ldots,L$ and $\varrho : \mathbb{R} \to \mathbb{R}$ is a nonlinear function (called the activation function), and applied component-wise on the outputs of $A_l(\cdot;\theta(l))$. Common choices are $\tanh(\cdot), \sin(\cdot), \max(0,\cdot)$ etc. All these matrices $\mathcal{W}_l$ and vectors $\mathcal{B}_l$ form the parameters θ of the DNN and they have the dimension

$$P = \sum_{l=1}^{L+1} \eta_l(\eta_{l-1}+1) = \eta(d+1) + \eta(\eta+1)(L-1) + \mathfrak{q}(\eta+1),$$

for fixed $d, \mathfrak{q}, L$ and η. We denote by Θ the set of possible parameters for the DNN $\phi(\cdot;\theta)$ with $\theta \in \Theta$. The universal approximation theorem [41, 19] justifies the use of DNNs as function approximators.

Another NN architecture are the RNNs, which are specialized to learning long-term dependencies. Hence, it is naturally interesting to see whether RNNs can improve the forward deep learning schemes [22, 86] for solving BSDEs, in particular to overcome the disadvantages mentioned already. RNNs are a type of NN that allow previous outputs to be used as inputs with hidden states. More precisely, the standard RNNs [89] are defined as follows: let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathbb{R}^d$

be input vectors, the network computes hidden states $\mathfrak{H}_1, \mathfrak{H}_2, \ldots, \mathfrak{H}_N \in \mathbb{R}^\eta$, and predictions $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N, \in \mathbb{R}^\mathfrak{q}$, by the equations

$$\begin{aligned}\mathfrak{H}_n &= \varrho(\mathcal{W}_\mathfrak{H}\mathfrak{H}_{n-1} + \mathcal{W}_\mathbf{x}\mathbf{x}_n + \mathcal{B}_\mathfrak{H}),\\ \mathbf{y}_n &= \mathcal{W}_\mathbf{y}\mathfrak{H}_n + \mathcal{B}_\mathbf{y},\end{aligned}$$

where $n = 1, \ldots, N$, $\mathfrak{H}_0 = 0$ or considered as trainable parameter, $\theta = (\mathcal{W}_\mathfrak{H}, \mathcal{W}_\mathbf{x}, \mathcal{B}_\mathfrak{H}, \mathcal{W}_\mathbf{y}, \mathcal{B}_\mathbf{y}) \in \mathbb{R}^P$ are the trainable parameters and ϱ is the nonlinear activation function. Note that the standard RNNs are universal approximators as well, see [90]. If one shall think that $\mathfrak{H}_n$ depends only on the current input $\mathbf{x}_n$ and the last hidden state $\mathfrak{H}_{n-1}$, and suppose that the distribution over the hidden states is well-defined, the standard RNNs should preserve the Markovian property for the BSDEs. However, although the Markovian property can be preserved, our numerical results show that no improvements can be observed by using the standard RNNs in Section 1.5. Furthermore, we also check and find that the accuracy of the deep learning-based algorithms for solving BSDEs cannot be improved by using the advanced RNNs, e.g., Long Short-Term Memory (LSTM) networks or bidirectional RNNs. The reason could be that such NN architectures even violate the Markovian property for the BSDEs. For example, in the LSTM the output $\mathbf{y}_n$ depends not only on the input vector $\mathbf{x}_n$ (and parameters θ), but also the hidden state $\mathfrak{H}_{n-1}$ and/or $\mathfrak{H}_{n-2}, \ldots, \mathfrak{H}_0$, i.e., $\mathbf{x}_{n-1}$ and/or $\mathbf{x}_{n-2}, \ldots, \mathbf{x}_0$.

1.2.4 Supervised deep learning

We present supervised deep learning using DNNs as it is the main NN architecture used in this thesis. After the DNN architecture is defined, what determines the mapping of a certain input to an output are the parameters θ incorporated in the DNN model. These parameters need to be optimized s.t. the DNN approximates the unknown function which is called the training of the DNN. The loss function acts as the objective function to be minimized during the training procedure, in which the DNNs optimal set of parameters is searched.

Consider the training data sampled from some (unknown) multivariate joint distribution $(\mathbf{x}, \mathbf{y}) \sim p$, where the random variable $\mathbf{x} \in \mathbb{R}^d$ is referred as the input and the random variable $\mathbf{y} \in \mathbb{R}$ as the label. The goal (in a regression setting) is then to approximate the deterministic function $F(x) := \mathbb{E}^p[\mathbf{y}|\mathbf{x} = x]$ by DNN $\phi(\mathbf{x}; \theta)$ using $(\mathbf{x}, \mathbf{y}) \sim p$. The loss function measures how well the current approximation of the DNN is compared to the label. A common choice is the expected squared error, which is given as

$$\mathbf{L}(\theta) = \mathbb{E}^p\left[|\phi(\mathbf{x}; \theta) - \mathbf{y}|^2\right]. \tag{1.4}$$

Then the optimal parameters θ^* in (1.4) are given as

$$\theta^* \in \arg\min_{\theta \in \Theta} \mathbf{L}(\theta),$$

which can be estimated by using SGD-type algorithms.

1.3 Forward deep learning schemes

In order to formulate BSDE as a deep learning problem, the first step is to discretize the integrals in (1.1). Let us consider the time discretization Δ with uniform step size $\Delta t = t_{n+1} - t_n$.

For notational convenience we write $\Delta W_n = W_{t_{n+1}} - W_{t_n}$, $(X_n, Y_n, Z_n) = (X_{t_n}, Y_{t_n}, Z_{t_n})$ and $\left(X_n^\Delta, Y_n^\Delta, Z_n^\Delta\right)$ for the approximations. The well-known Euler-Maruyama scheme reads

$$X_{n+1}^\Delta = X_n^\Delta + a\left(t_n, X_n^\Delta\right)\Delta t + b\left(t_n, X_n^\Delta\right)\Delta W_n, \tag{1.5}$$

for $n = 0, 1, \ldots, N-1$ and $X_0^\Delta = x_0$. Since the Brownian motions are independent, $\Delta W_n \sim \mathcal{N}(\mathbf{0}_d,\, \Delta t\, \mathbf{I}_d)$, with $\mathbf{0}_d \in \mathbb{R}^d$ a vector of zeros and $\mathbf{I}_d \in \mathbb{R}^{d\times d}$ the identity matrix.

Next we apply the Euler-Maruyama scheme for the backward process. For the time interval $[t_n, t_{n+1}]$, the integral form of the backward process reads

$$Y_n = Y_{n+1} + \int_{t_n}^{t_{n+1}} f\left(s, \mathbf{X}_s\right)\, ds - \int_{t_n}^{t_{n+1}} Z_s\, dW_s,$$

which can be straightforwardly reformulated as

$$Y_{n+1} = Y_n - \int_{t_n}^{t_{n+1}} f\left(s, \mathbf{X}_s\right)\, ds + \int_{t_n}^{t_{n+1}} Z_s\, dW_s.$$

Applying the Euler-Maruyama scheme for the latter equation one obtains

$$Y_{n+1}^\Delta = Y_n^\Delta - f\left(t_n, \mathbf{X}_n^\Delta\right)\Delta t + Z_n^\Delta \Delta W_n, \tag{1.6}$$

for $n = 0, 1, \ldots, N-1$, where for notational convenience $\mathbf{X}_n^\Delta := \left(X_n^\Delta, Y_n^\Delta, Z_n^\Delta\right)$. By iterating (1.6) together with the terminal condition $g(X_N^\Delta)$, we have

$$Y_n^\Delta = g(X_N^\Delta) + \sum_{m=n}^{N-1}\left(f\left(t_m, \mathbf{X}_m^\Delta\right)\Delta t - Z_m^\Delta \Delta W_m\right), \tag{1.7}$$

for $n = 0, 1, \ldots, N-1$, which represents an iterative time discretization of

$$Y_n = g(X_T) + \int_{t_n}^{T} f\left(s, \mathbf{X}_s\right)\, ds - \int_{t_n}^{T} Z_s\, dW_s.$$

The discretized form defined by (1.7) plays an important role in our forward algorithm, as it will be presented in Section 1.3.3. The implicit equation (1.7) is assumed to be uniquely solvable. Furthermore, we assume that Assumptions AX1 and AY1 hold. Under such conditions, our algorithm approximates the solution of (1.7) by estimating Y^Δ with a NN and Z^Δ with the AD, and uses the SGD (in particular Adam optimizer) to find the root. The optimization algorithm minimizes the loss defined with (1.7) and finds the optimal parameters θ^* of the NN from the set Θ. In this way, the approximation of the solution to (1.7) has the optimization error by using the Adam algorithm, and the estimation error by using the empirical loss instead of the continuous loss. Note that this discretization is also used in [30] whose formulation is based on backward recursive local optimizations defined from (1.7) to estimate the solution and its gradient at each time step, namely a backward deep learning scheme. In our case, we consider a global optimization based on local losses obtained from (1.7) in a forward manner, hence a forward deep learning scheme.

After discretizing the integrals, the deep learning schemes are made fully implementable by approximating the unknown processes (Y_n^Δ, Z_n^Δ) in (1.5) for $n = 0, 1, \ldots, N$. These processes are parameterized using NNs, whose parameters are estimating by constructing an appropriate loss function. Next, we review how the forward deep learning methods in [22, 86] are constructed, and then present our new forward method.

1.3.1 The DBSDE scheme

The approximation of the discrete unknown processes $\left(Y_n^{\Delta}, Z_n^{\Delta}\right), n = 0, 1, \ldots, N$, in the DBSDE scheme [22] is designed as a stochastic control problem and can be presented as follows: starting from an initialization $Y_0^{\Delta,\theta} = \theta_0^y \in \mathbb{R}$ of Y_0^{Δ} and $Z_0^{\Delta,\theta} = \theta_0^z \in \mathbb{R}^{1\times d}$ of Z_0^{Δ}, and then using at each discrete time point $t_n, n = 1, 2, \ldots, N-1$, a different feedforward multilayer NN or DNN $\phi_n^z(\cdot;\theta_n^z) : \mathbb{R}^d \to \mathbb{R}^{1\times d}$ to approximate $Z_n^{\Delta} \in \mathbb{R}^{1\times d}$ as $Z_n^{\Delta,\theta}$, where the input of the network is the Markovian process $X_n^{\Delta} \in \mathbb{R}^d$. The approximation $\left\{Y_n^{\Delta,\theta}\right\}_{n=1}^{N}$ is calculated using the Euler-Maruyama method (1.6). Note that this algorithm forms a global DNN that consists of NNs defined at each time step, where the paths of $\left\{X_n^{\Delta}\right\}_{n=0}^{N}$ inclusive of $\{W_n\}_{n=0}^{N}$ are used as the input data. Note that W_n is an input to the NN as $\Delta W_{n-1} = W_n - W_{n-1}$ is used to calculate X_n^{Δ}. Then the final output $Y_N^{\Delta,\theta}$ depends on parameters $\theta = \left(\theta_0^y, \theta_0^z, \theta_1^z, \ldots, \theta_{N-1}^z\right)$. The output aims to match the terminal condition $g\left(X_N^{\Delta}\right)$ of the BSDE, and then optimizes the parameters θ over the parameter space Θ using the loss function

$$\mathbf{L}^{y,\Delta}(\theta) = \mathbf{E}\left[\left|Y_N^{\Delta,\theta} - g\left(X_N^{\Delta}\right)\right|^2\right]. \tag{1.8}$$

The final estimated parameters $\hat{\theta}$ are received after minimizing the loss function (1.8) using SGD-type algorithms, and provide the final approximation of $\left(Y_n^{\Delta}, Z_n^{\Delta}\right)$ as $\left(Y_n^{\Delta,\hat{\theta}}, Z_n^{\Delta,\hat{\theta}}\right)$ for $n = 0, 1, \ldots, N-1$ and $Y_N^{\Delta,\hat{\theta}}$. For the algorithmic framework we refer to [22]. Note that θ_0^y and θ_0^z are considered as learnable parameters in [22] and initialized by sampling from uniform distributions.

Remark 1.3.1. *As already mentioned, a strong drawback of the DBSDE scheme is that only (Y_0, Z_0) can be well approximated. The approximation of the processes Y and Z over the entire discrete time domain in the DBSDE scheme can be enhanced by utilizing a single DNN for the process Z [57], incorporating the time variable as an input to the network, instead of employing multiple DNNs at each discrete time point. This is because, for sufficiently regular solutions, the gradient between two close time steps should be similar for a given spatial point x. In the DBSDE scheme, there is no inherent connection between the gradients of successive and potentially close time steps. Moreover, by closely linking the parameters to the loss function using one DNN makes the resulting architecture easier to optimize and should perform better than the DBSDE scheme. As demonstrated in [12, 57] through various high-dimensional numerical examples, using a single DNN instead of multiple DNNs not only enhances numerical stability across the entire time domain but also reduces computational time. From the same reasoning for the process Y, such approach using one DNN is also considered in [86] and in our novel forward deep learning approach, where the DNN is used for the process Y and AD for Z.*

Due to the complexity of the problem formulation and the inherent challenges in ensuring strict regression error bounds for NNs using SGD methods, it is not straightforward to show that the total error resulting from minimizing the loss function (1.8) approaches zero as the mesh size becomes infinitely small. However, there are some articles about the convergence analysis of the DBSDE scheme, using different assumptions to guarantee their results. We refer the interested readers to [39] for an a posteriori error estimation of the scheme in the general case of coupled FBSDEs. For coupled FBSDEs with non-Lipschitz coefficients, refer to [56], and for fully-coupled drift coefficients, see [77]. It has been demonstrated in these works that as long as the loss function (1.8) is optimized to be close to zero under fine time discretization, the approximate solution is close to the true solution. Additionally, leveraging the universal approximation theorem, the authors show that NNs with appropriate parameters can yield

an approximately accurate numerical solution. However, these studies neglect the optimization error from the SGD algorithm (and the estimation error associated with minimizing the empirical version of the loss (1.8)), as accounting for it significantly increases the complexity of the analysis. We demonstrate in the numerical section, see also [44], that the DBSDE method even diverges for a complex solution structure and a long terminal time.

1.3.2 The local DBSDE scheme

To overcome the drawback of the DBSDE scheme that only (Y_0, Z_0) can be well approximated, the author in [86] proposed to formulate the BSDE problem based on a global optimization with local losses (we refer as Local Deep BSDE or LDBSDE scheme). The solution is approximated by using a DNN and its gradient via AD. These approximations are performed by the global minimization of local loss functions defined in terms of the dynamics of the BSDE at each discrete time point given by the Euler-Maruyama method (1.6) and the terminal condition included as an additional term. The algorithm is given as follows:

- Generate approximations X_{n+1}^{Δ} for $n = 0, 1, \dots, N-1$ using (1.5).
- At each discrete time point t_n, $n = 0, 1, \dots, N$, use one DNN $\phi^y(\cdot;\theta) : \mathbb{R}^{1+d} \to \mathbb{R}$ to approximate Y_n^{Δ} and Z_n^{Δ} using AD due to (1.3), where the input vector of the network is the time value $t_n \in \mathbb{R}_+$ and the Markovian process $X_n^{\Delta} \in \mathbb{R}^d$. More precisely

$$Y_n^{\Delta,\theta} = \phi^y(t_n, X_n^{\Delta};\theta), \quad Z_n^{\Delta,\theta} = \nabla_x \phi^y(t,x;\theta)\Big|_{(t,x)=(t_n,X_n^{\Delta})} b\left(t_n, X_n^{\Delta}\right).$$

- Train the parameters θ using a global loss function including local losses s.t. the dynamics of discretized BSDE (1.6) are satisfied at each time step, namely

$$\begin{aligned}
\mathbf{L}^{y,\Delta}(\theta) &= \sum_{n=0}^{N} \mathbf{L}_n^{y,\Delta}(\theta), \\
\mathbf{L}_n^{y,\Delta}(\theta) &= \mathbb{E}\left[\left|Y_{n+1}^{\Delta,\theta} - Y_n^{\Delta,\theta} + f\left(t_n, \mathbf{X}_n^{\Delta,\theta}\right)\Delta t - Z_n^{\Delta,\theta}\Delta W_n\right|^2\right], \quad n = 0, 1, \dots, N-1, \\
\mathbf{L}_N^{y,\Delta}(\theta) &= \mathbb{E}\left[\left|Y_N^{\Delta,\theta} - g\left(X_N^{\Delta}\right)\right|^2\right],
\end{aligned} \tag{1.9}$$

  where for notational convenience $\mathbf{X}_n^{\Delta,\theta} := \left(X_n^{\Delta}, Y_n^{\Delta,\theta}, Z_n^{\Delta,\theta}\right)$.
- Approximate the optimal parameters $\theta^* \in \arg\min_{\theta\in\Theta} \mathbf{L}^{y,\Delta}(\theta)$ using a SGD method and receive the final estimated parameters $\hat{\theta}$. Set the final approximation of $\left(Y_n^{\Delta}, Z_n^{\Delta}\right)$ as $\left(Y_n^{\Delta,\hat{\theta}}, Z_n^{\Delta,\hat{\theta}}\right)$ for $n = 0, 1, \dots, N$.

The convergence of the LDBSDE scheme (and a Milstein-version of the scheme) is discussed in [66], providing a similar a posteriori error type estimation as in [39] for the DBSDE scheme. The authors show that the error of these schemes is bounded by their respective loss function (1.9), and the loss functional converges sufficiently fast to zero guaranteeing that the error of the scheme vanishes in the limit. Such a situation is attainable through the universal approximation theorem of NNs. Note that the SGD method used to solve the optimization problem is assumed to not get trapped in a local minimum (which can be the case as we show in the numerical experiments). Furthermore, the authors highlight an important distinction: algorithms

like the DBSDE scheme require one DNN at each discrete time point, thereby limiting convergence results to spatial approximations for fixed time discretizations. In contrast, schemes such as the LDBSDE employ a single DNN defined across the entire state space, including the temporal dimension, avoiding the increase of DNNs and thus addressing this scalability challenge effectively.

1.3.3 The locally additive DBSDE scheme

The LDBSDE scheme improves the results of the DBSDE scheme for the approximations in the entire time domain. However, it can also get stuck in poor local minima as the DBSDE scheme especially for a complex solution structure and a long terminal time. Our idea is to consider a formulation based on a global optimization with local loss function, where each loss term includes the terminal condition. This is achieved by using the iterative time discretization (1.7). We refer to this as the Locally additive Deep BSDE (LaDBSDE) scheme as each local loss term is composed of a linear addition of nonlinear terms. The algorithm is given as follows:

- Generate approximations X_{n+1}^{Δ} for $n = 0, 1, \ldots, N-1$ using (1.5).
- At each discrete time point t_n, $n = 0, 1, \ldots, N-1$, use one DNN $\phi^y(\cdot;\theta) : \mathbb{R}^{1+d} \to \mathbb{R}$ to approximate Y_n^{Δ} and Z_n^{Δ} using AD due to (1.3), where the input vector of the network is the time value $t_n \in \mathbb{R}_+$ and the Markovian process $X_n^{\Delta} \in \mathbb{R}^d$. More precisely

$$Y_n^{\Delta,\theta} = \phi^y(t_n, X_n^{\Delta};\theta), \quad Z_n^{\Delta,\theta} = \nabla_x \phi^y(t,x;\theta)\Big|_{(t,x)=(t_n,X_n^{\Delta})} b\left(t_n, X_n^{\Delta}\right).$$

- Train the parameters θ using a global loss function including local losses s.t. the iterated dynamics of discretized BSDE (1.7) that always includes the terminal condition are satisfied at each discrete time point, namely

$$\begin{aligned} \mathbf{L}^{y,\Delta}(\theta) &= \sum_{n=0}^{N-1} \mathbf{L}_n^{y,\Delta}(\theta), \\ \mathbf{L}_n^{y,\Delta}(\theta) &= \mathbb{E}\left[\left| Y_n^{\Delta,\theta} - \sum_{m=n}^{N-1} \left(f\left(t_m, \mathbf{X}_m^{\Delta,\theta}\right)\Delta t - Z_m^{\Delta,\theta}\Delta W_m \right) - g\left(X_N^{\Delta}\right)\right|^2\right]. \end{aligned} \tag{1.10}$$

- Approximate the optimal parameters of $\theta^* \in \arg\min_{\theta\in\Theta} \mathbf{L}^{y,\Delta}(\theta)$ using a SGD method and receive the final estimated parameters $\hat{\theta}$. Set the final approximation of $\left(Y_n^{\Delta}, Z_n^{\Delta}\right)$ as $\left(Y_n^{\Delta,\hat{\theta}}, Z_n^{\Delta,\hat{\theta}}\right)$ for $n = 0, 1, \ldots, N-1$.

Remark 1.3.2. *One can consider two DNNs, one for the process Y and another for Z, which can result in better approximations compared to using AD for Z, see [44]. However, we aim to show that the improvements from the LaDBSDE scheme compared to the LDBSDE scheme result from the reformulation of loss function, specifically using (1.7) instead of (1.6). Therefore, we considered a similar architecture to that used in the LDBSDE scheme.*

We present the algorithmic framework (without using mini-batches and Adam optimizer) of the LaDBSDE scheme with one sample path indexed by j and learning rate α in Framework 1.3.1.

Framework 1.3.1. *Let $T, \alpha \in (0,\infty)$, $d, P, N \in \mathbb{N}$, $x_0 \in \mathbb{R}^d$, $a : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$, $b : [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d\times d}$, $f : [0,T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{1\times d} \to \mathbb{R}$ and $g : \mathbb{R}^d \to \mathbb{R}$ be functions, let $(\Omega, \mathcal{F}, \mathbb{P})$ be*

a probability space, let $W_j : [0,T] \times \Omega \to \mathbb{R}^d$, $j \in \mathbb{N}$, *be independent* d*-dimensional standard Brownian motions on* $(\Omega, \mathcal{F}, \mathbb{P})$, *let* $t_0, t_1, \ldots, t_N \in [0,T]$ *be real numbers with*

$$0 = t_0 < t_1 < \ldots < t_N = T,$$

for every $j \in \mathbb{N}$, *let* $X_j^\Delta : \{0,1,\ldots,N\} \times \Omega \to \mathbb{R}^d$ *be a stochastic process which satisfies for all* $n \in \{0,1,\ldots,N-1\}$, $\Delta W_{n,j} = W_{n+1,j} - W_{n,j}$ *that* $X_{0,j}^\Delta = x_0$ *and*

$$X_{n+1,j}^\Delta = X_{n,j}^\Delta + a\left(t_n, X_{n,j}^\Delta\right)\Delta t + b\left(t_n, X_{n,j}^\Delta\right)\Delta W_{n,j},$$

for every $\theta \in \mathbb{R}^P$, θ *parameters of a NN,* $n \in \{0,1,\ldots,N-1\}$, $\mathfrak{q} = 1$, $\varrho : \mathbb{R} \to \mathbb{R}$, $L, \eta \in \mathbb{N}$, *let* $\phi = (\phi(t,x;\theta))_{t\in\mathbb{R}_+, x\in\mathbb{R}^d, \theta\in\mathbb{R}^P} : \mathbb{R}^{d+1} \to \mathbb{R}^{\mathfrak{q}}$ *(*$\phi \in \mathbb{C}_{\mathfrak{b}}^{0,1}\left([0,T]\times\mathbb{R}^d; \mathbb{R}^{1\times d}\right)$*) be a family of functions generated by NNs, where the output is given as* $Y_{n,j}^{\Delta,\theta} = \phi(t_n, X_{n,j}^\Delta; \theta)$ *and let* $Z_{n,j}^{\Delta,\theta} = \nabla_x \phi(t,x;\theta)\Big|_{(t,x)=(t_n, X_{n,j}^\Delta)} b\left(t_n, X_{n,j}^\Delta\right)$, *for every* $j \in \mathbb{N}$, $n \in \{0,1,\ldots,N-1\}$ *let* $\varphi_{n,j} : \mathbb{R}^P \times \Omega \to \mathbb{R}$ *be the function which satisfies for all* $\theta \in \mathbb{R}^P$, $\omega \in \Omega$ *that*

$$\varphi_{n,j}(\theta,\omega) = \left| Y_{n,j}^{\Delta,\theta}(\omega) - \sum_{m=n}^{N-1}\left(f\left(t_m, \mathbf{X}_{m,j}^{\Delta,\theta}(\omega)\right)\Delta t - Z_{m,j}^{\Delta,\theta}(\omega)\Delta W_{m,j}(\omega)\right) - g\left(X_{N,j}^\Delta(\omega)\right)\right|^2,$$

for every $j \in \mathbb{N}$ *let* $\varphi_j : \mathbb{R}^P \times \Omega \to \mathbb{R}$ *be the function which satisfies for all* $\theta \in \mathbb{R}^P$, $\omega \in \Omega$ *that*

$$\varphi_j(\theta,\omega) = \sum_{n=0}^{N-1} \varphi_{n,j}(\theta,\omega),$$

for every $j \in \mathbb{N}$ *let* $(\nabla_\theta \varphi_j) : \mathbb{R}^P \times \Omega \to \mathbb{R}^P$ *be a function for all* $\omega \in \Omega$, $\theta \in \left\{v \in \mathbb{R}^P : \left(\mathbb{R}^P \ni w \mapsto \varphi_j(w,\omega) \in \mathbb{R} \text{ is differentiable at } v \in \mathbb{R}^P\right)\right\}$ *and* $\hat{\theta} : \mathbb{N} \times \Omega \to \mathbb{R}^P$ *be a stochastic process which satisfies for all* $\kappa \in \mathbb{N}$ *that*

$$\hat{\theta}^\kappa = \hat{\theta}^{\kappa-1} - \alpha(\nabla_\theta \varphi_j)\left(\hat{\theta}^{\kappa-1}\right).$$

The architecture of the LaDBSDE scheme is displayed in Figure 1.1. The flow of the information is represented by the direction of the arrows. The calculations can be broken down into three steps. In the first step, the samples of the forward SDE are calculated. The information used in this step is represented by the dotted lines. For instance, to calculate X_2^Δ, $(t_1, \Delta W_1, X_1^\Delta)$ is used, and $(t_{N-1}, \Delta W_{N-1}, X_{N-1}^\Delta)$ for X_N^Δ. The second step is to calculate the values $(Y_n^{\Delta,\theta}, Z_n^{\Delta,\theta})$ for $n = 0,1,\ldots,N-1$, using a DNN and the AD. The information needed for such calculations is represented by the solid lines. For example, the DNN uses as input (t_1, X_1^Δ) to calculate $Y_1^{\Delta,\theta}$. Using the AD we calculate the gradient in the spatial direction to obtain $Z_1^{\Delta,\theta}$. Finally, the local losses are calculated backwardly with the information presented by the dashed lines. To calculate $\mathbf{L}_{N-1}^{y,\Delta}$, the terminal condition $Y_N^\Delta = g(X_N^\Delta)$ and $(t_{N-1}, \Delta W_{N-1}, X_{N-1}^\Delta, Y_{N-1}^{\Delta,\theta}, Z_{N-1}^{\Delta,\theta})$ are used. For $\mathbf{L}_{N-2}^{y,\Delta}$, $(t_{N-2}, \Delta W_{N-2}, X_{N-2}^\Delta, Y_{N-2}^{\Delta,\theta}, Z_{N-2}^{\Delta,\theta})$ and the information form $\mathbf{L}_{N-1}^{y,\Delta}$ are used, namely Y_N^Δ and $(t_{N-1}, \Delta W_{N-1}, X_{N-1}^\Delta, Y_{N-1}^{\Delta,\theta})$. The same holds for the other loss terms. We use a reverse computation approach for the local loss functions because it is more efficient than the direct computation approach (where each local loss function, as defined in (1.10), is calculated sequentially from from $t = 0$ to $t = T$). The LaDBSDE algorithm calculating the final estimates $\left(Y_n^{\Delta,\hat{\theta}}, Z_n^{\Delta,\hat{\theta}}\right)$ for $n = 0,1,\ldots,N-1$, with mini-batches of size B and $\mathfrak{K}$ optimization steps of Adam optimizer using direct and reverse computation approaches of the

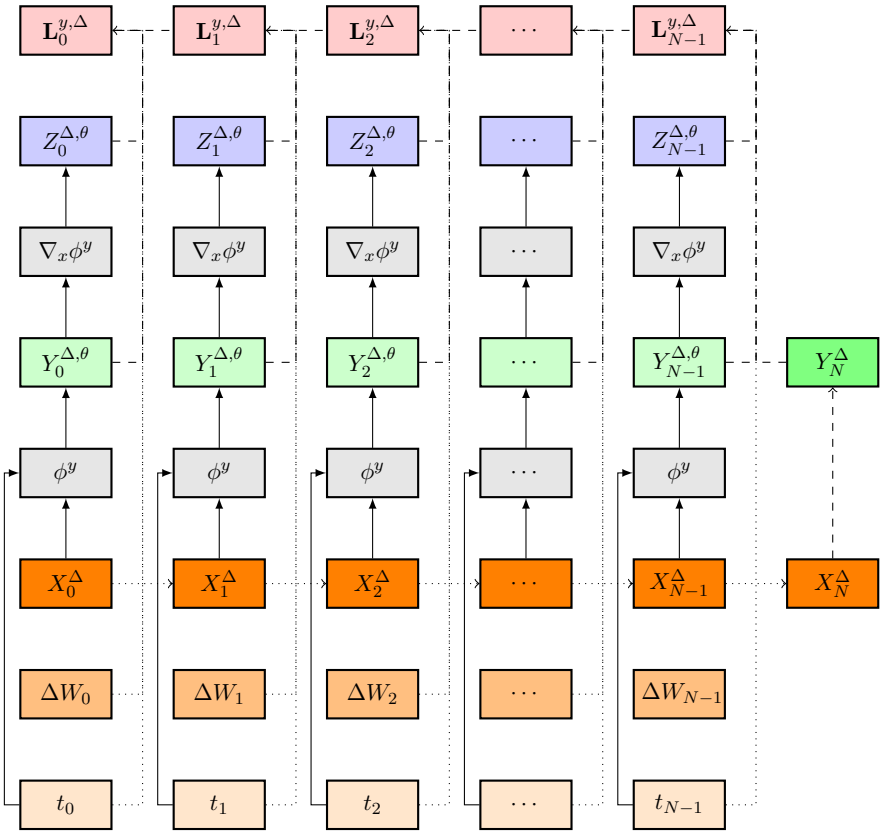


Figure 1.1: Architecture of the LaDBSDE scheme.

loss function (1.10) are given in Algorithm 1 and 2, respectively. Note that $\hat{\theta}$ is an estimation of θ^* due to the optimization error resulting from the Adam optimization algorithm and the estimation error from the empirical version of loss (1.10) given as

$$\tilde{\mathbf{L}}^{y,\Delta}\left(\hat{\theta}\right) = \sum_{n=0}^{N-1} \tilde{\mathbf{L}}_n^{y,\Delta}\left(\hat{\theta}\right),$$

$$\tilde{\mathbf{L}}_n^{y,\Delta}\left(\hat{\theta}\right) = \frac{1}{B}\sum_{j=1}^{B}\left|Y_{n,j}^{\Delta,\hat{\theta}} - \sum_{m=n}^{N-1}\left(f\left(t_m, \mathbf{X}_{m,j}^{\Delta,\hat{\theta}}\right)\Delta t - Z_{m,j}^{\Delta,\hat{\theta}}\Delta W_{m,j}\right) - g\left(X_{N,j}^{\Delta}\right)\right|^2,$$

for a batch size B. With Algorithm 2 the computation time of LaDBSDE is comparable to that of LDBSDE. An a posteriori error analysis for the LaDBSDE scheme can be conducted by following, for example, the methodology outlined in [66]. This is part of our ongoing research.

1.4 Backward deep learning schemes

As for the forward deep learning schemes, the first step is to discretize the integrals in BSDE (1.1), which are given from (1.5) and (1.6). The first backward deep learning approach is proposed in [44]. The discrete unknown processes $\left(Y_n^{\Delta}, Z_n^{\Delta}\right)$ in (1.6) are approximated using two DNNs. The DNN parameters are backwardly optimized at each discrete time step from the minimization of loss functions defined recursively by backward induction, namely local optimization at each discrete time point. The method is referred to as deep backward dynamic programming (DBDP) scheme, as it builds upon the backward dynamic programming relation (1.6). The

Algorithm 1: Algorithm of LaDBSDE scheme using a direct computation of the loss function (1.10)

Input: (N, d, T, x_0) - problem parameters
Input: (a, b, f, g) - functions of the BSDE
Input: $(\alpha, \mathfrak{K}, L, \eta, \varrho, B)$ - DNN hyperparameters
Output: $\left(\left\{ Y_n^{\Delta,\hat{\theta}} \right\}_{n=0}^{N-1}, \left\{ Z_n^{\Delta,\hat{\theta}} \right\}_{n=0}^{N-1} \right)$ - estimated solution
$\Delta t = \frac{T}{N}$
for $n = 0 : N$ **do**
  $t_n = n\,\Delta t$
end
$\mathfrak{q} = 1$ - DNN output dimension; input dimension $d + 1$
$\hat{\theta}^0$ - Xavier normal initializer [31]
Optimization or training part
for $\kappa = 1 : \mathfrak{K}$ **do**
  for $j = 1 : B$ **do**
    $X_{0,m}^{\Delta} = x_0$
    for $n = 0 : N - 1$ **do**
      Euler-Maruyama for the forward SDE
      $\Delta W_{n,j} \sim \mathcal{N}(\mathbf{0}_d, \Delta t\, \mathbf{I}_d)$
      $X_{n+1,j}^{\Delta} = X_{n,j}^{\Delta} + a\left(t_n, X_{n,j}^{\Delta}\right)\Delta t + b\left(t_n, X_{n,j}^{\Delta}\right)\Delta W_{n,j}$
      Use DNN with (L, η, ϱ) for Y and AD for Z
      $Y_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} = \phi^y\left(t_n, X_{n,j}^{\Delta}; \hat{\theta}^{\kappa-1}\right)$
      $Z_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} = \nabla_x \phi^y(t, x; \hat{\theta}^{\kappa-1})\Big|_{(t,x)=\left(t_n, X_{n,j}^{\Delta}\right)} b\left(t_n, X_{n,j}^{\Delta}\right)$
    end
  end
  Direct computation of the loss function (1.10)
  for $j = 1 : B$ **do**
    for $n = 0 : N - 1$ **do**
      $\tilde{Y}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} = g\left(X_{N,j}^{\Delta}\right)$
      for $m = n : N - 1$ **do**
        $\tilde{Y}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} = \tilde{Y}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} + f\left(t_m, \mathbf{X}_{m,j}^{\Delta,\hat{\theta}^{\kappa-1}}\right)\Delta t - Z_{m,j}^{\Delta,\hat{\theta}^{\kappa-1}}\Delta W_{m,j}$
      end
      $\tilde{\mathbf{L}}_{n,j}^{y,\Delta}(\hat{\theta}^{\kappa-1}) = \left| Y_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} - \tilde{Y}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} \right|^2$
    end
  end
  $\tilde{\mathbf{L}}^{y,\Delta}(\hat{\theta}^{\kappa-1}) = \sum_{n=0}^{N-1} \frac{1}{B} \sum_{j=1}^{B} \tilde{\mathbf{L}}_{n,j}^{y,\Delta}(\hat{\theta}^{\kappa-1})$
  Adam optimization step
  $\hat{\theta}^{\kappa}$ - trained parameters at step κ using Adam optimizer [64] for $\tilde{\mathbf{L}}^{y,\Delta}(\hat{\theta}^{\kappa-1})$
end
$\hat{\theta} = \hat{\theta}^{\mathfrak{K}}$ - final estimated DNN parameters after $\mathfrak{K}$ optimization steps

Algorithm 2: Algorithm of LaDBSDE scheme using a reverse computation of the loss function (1.10)

Input: (N, d, T, x_0) - problem parameters
Input: (a, b, f, g) - functions of the BSDE
Input: $(\alpha, \mathfrak{K}, L, \eta, \varrho, B)$ - DNN hyperparameters
Output: $\left(\left\{Y_n^{\Delta,\hat{\theta}}\right\}_{n=0}^{N-1}, \left\{Z_n^{\Delta,\hat{\theta}}\right\}_{n=0}^{N-1}\right)$ - estimated solution

$\Delta t = \frac{T}{N}$
for $n = 0 : N$ **do**
  $t_n = n\,\Delta t$
end
$\mathfrak{q} = 1$ - DNN output dimension; input dimension $d + 1$
$\hat{\theta}^0$ - Xavier normal initializer [31]
Optimization or training part
for $\kappa = 1 : \mathfrak{K}$ **do**
  for $j = 1 : B$ **do**
    $X_{0,m}^{\Delta} = x_0$
    for $n = 0 : N - 1$ **do**
      Euler-Maruyama for the forward SDE
      $\Delta W_{n,j} \sim \mathcal{N}(\mathbf{0}_d, \Delta t\, \mathbf{I}_d)$
      $X_{n+1,j}^{\Delta} = X_{n,j}^{\Delta} + a\left(t_n, X_{n,j}^{\Delta}\right)\Delta t + b\left(t_n, X_{n,j}^{\Delta}\right)\Delta W_{n,j}$
      Use DNN with (L, η, ϱ) for Y and AD for Z
      $Y_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} = \phi^y\left(t_n, X_{n,j}^{\Delta}; \hat{\theta}^{\kappa-1}\right)$
      $Z_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} = \nabla_x \phi^y(t, x; \hat{\theta}^{\kappa-1})\Big|_{(t,x)=\left(t_n, X_{n,j}^{\Delta}\right)} b\left(t_n, X_{n,j}^{\Delta}\right)$
    end
  end
  Reverse computation of the loss function (1.10)
  for $j = 1 : B$ **do**
    $\tilde{Y}_{N,j}^{\Delta,\hat{\theta}^{\kappa-1}} = g\left(X_{N,j}^{\Delta}\right)$
    for $n = N - 1 : 0$ **do**
      $\tilde{Y}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} = \tilde{Y}_{n+1,j}^{\Delta,\hat{\theta}^{\kappa-1}} + f\left(t_n, \mathbf{X}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}}\right)\Delta t - Z_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}}\Delta W_{n,j}$
      $\tilde{\mathbf{L}}_{n,j}^{y,\Delta}(\hat{\theta}^{\kappa-1}) = \left|Y_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} - \tilde{Y}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}}\right|^2$
    end
  end
  $\tilde{\mathbf{L}}^{y,\Delta}(\hat{\theta}^{\kappa-1}) = \sum_{n=0}^{N-1} \frac{1}{B} \sum_{j=1}^{B} \tilde{\mathbf{L}}_{n,j}^{y,\Delta}(\hat{\theta}^{\kappa-1})$
  Adam optimization step
  $\hat{\theta}^{\kappa}$ - trained parameters at step κ using Adam optimizer [64] for $\tilde{\mathbf{L}}^{y,\Delta}(\hat{\theta}^{\kappa-1})$
end
$\hat{\theta} = \hat{\theta}^{\mathfrak{K}}$ - final estimated DNN parameters after $\mathfrak{K}$ optimization steps

scheme can be formulated as follows:

- Generate approximations X_{n+1}^{Δ} for $n = 0, 1, \dots, N-1$ using (1.5).
- Set
$$Y_N^{\Delta,\hat{\theta}} := g(X_N^{\Delta}), \quad Z_N^{\Delta,\hat{\theta}} := \nabla_x g(X_N^{\Delta}) b(t_N, X_N^{\Delta}).$$
- For each discrete time point t_n, $n = N-1, N-2, \dots, 0$, use two DNNs $\phi_n^y(\cdot;\theta_n^y) : \mathbb{R}^d \to \mathbb{R}$ and $\phi_n^z(\cdot;\theta_n^z) : \mathbb{R}^d \to \mathbb{R}^{1\times d}$ to approximate the discrete processes $\left(Y_n^{\Delta}, Z_n^{\Delta}\right)$, respectively, where the input vector of the network is the Markovian process $X_n^{\Delta} \in \mathbb{R}^d$. More precisely
$$Y_n^{\Delta,\theta} = \phi_n^y(X_n^{\Delta};\theta_n^y), \quad Z_n^{\Delta,\theta} = \phi_n^z(X_n^{\Delta};\theta_n^z).$$
- Train the parameter set $\theta_n = (\theta_n^y, \theta_n^z)$ by constructing a loss function s.t. the dynamics of the discretized process Y given by (1.6) are fulfilled, namely
$$\mathbf{L}_n^{y,\Delta}(\theta_n) = \mathbb{E}\left[\left|Y_{n+1}^{\Delta,\hat{\theta}} - Y_n^{\Delta,\theta} + f\left(t_n, \mathbf{X}_n^{\Delta,\theta}\right)\Delta t_n - Z_n^{\Delta,\theta}\Delta W_n\right|^2\right]. \tag{1.11}$$
- Approximate the optimal parameters $\theta_n^* \in \arg\min_{\theta_n\in\Theta_n} \mathbf{L}_n^{y,\Delta}(\theta_n)$ using a SGD method and receive the estimated parameters $\hat{\theta}_n = \left(\hat{\theta}_n^y, \hat{\theta}_n^z\right)$. Then, define
$$Y_n^{\Delta,\hat{\theta}} := \phi_n^y\left(X_n^{\Delta};\hat{\theta}_n^y\right), \quad Z_n^{\Delta,\hat{\theta}} := \phi_n^z\left(X_n^{\Delta};\hat{\theta}_n^z\right).$$

The authors in [44] provided a similar scheme to the DBDP one, where the process Y is estimated by a DNN and Z via AD, similar to the LDBSDE or LaDBSDE schemes. Note that another backward deep learning scheme is developed in [30], where the authors used the iterative time discretization (1.7) instead of (1.6) for the backward part in (1.1). We consider the DBDP scheme and compare it to its differential deep learning counterpart in Chapter 2. The convergence of the DBDP scheme towards the solution (Y, Z) of (1.1) is provided in [44] together with a rate of convergence that depends on the discretization error from the Euler-Maruyama scheme and the approximation or model error by the DNNs, see Theorem 4.1 in [44].

1.5 Numerical results

In this section we illustrate the improved performance using the LaDBSDE scheme compared to the schemes DBSDE and LDBSDE. All the experiments below were run in PYTHON using TensorFlow on the PLEIADES cluster (no parallelization), which consists of 268 workernodes and additionally 5 GPU nodes with 8 NVidia HGX A100 GPUs (128 cores each, 2 TB memory, 16 GB per thread). We run the algorithms on the GPU nodes. For more information, see PLEIADES documentation[1].

In all the following examples, we consider similar hyperparameters for all schemes for a fair comparison.

For the DBSDE scheme, we keep the network architecture considered in [22]. In the implementations, $N-1$ DNNs are employed to calculate $Z_n^{\Delta,\theta}, n = 1, 2, \dots, N-1, \theta \in \mathbb{R}^P$. Each of the NNs has $L = 2$ hidden layers and $\eta = d + 10$ neurons per hidden layer. The authors [22] also

[1] https://pleiadesbuw.github.io/PleiadesUserDocumentation/

adopt batch normalization [50] right after each matrix multiplication and before activation. The rectifier function (ReLU) $\mathbb{R} \ni x \mapsto \max(0, x) \in [0, \infty)$ is used as the activation function ϱ for the hidden variables. All the weights are initialized using a normal or a uniform distribution without any pre-training. The choice of the dimension of the parameters is given as [22], i.e.,

$$P = d + 1 + (N-1)(2d(d+10) + (d+10)^2 + 4(d+10) + 2d).$$

For our scheme and the LDBSDE scheme, we adopt a similar network architecture as in [86], where the author considered a DNN with $L = 4$ hidden layers and $\eta = 256$ neurons. Based on this setting, the choice of the dimension of the parameters (including bias term) is given by

$$P = 256d + 198145. \tag{1.12}$$

Furthermore, $\mathbb{R} \ni x \mapsto \sin(x) \in [-1, 1]$ is used as activation function ϱ in [86] and the following learning rate decay approach:

$$\alpha_\kappa = 10^{\left(\mathbb{1}_{[0,20000]}(\kappa) + \mathbb{1}_{(20000,50000]}(\kappa) + \mathbb{1}_{(50000,80000]}(\kappa) - 6\right)},$$

for $\kappa = 1, 2, \ldots, \mathfrak{K}$, where $\mathfrak{K} = 100000$ is the total number of Adam optimizer steps. Instead of $\sin(x)$, we consider $\mathbb{R} \ni x \mapsto \tanh(x) \in [-1, 1]$. Note that from Framework 1.3.1 we require to optimize over differentiable DNNs, and using the classical ReLU function may lead to an explosion while calculating the numerical approximation of the Z process. Moreover, we use $L = 4$ hidden layers and $\eta = 10 + d$ neurons for the hidden layers. The dimension of the parameters (including bias term) for such network architecture is given by

$$P = 2d^2 + 56d + 361. \tag{1.13}$$

Compared to the complexity (1.12) given in [86], such parametrization of the NN gives a smaller complexity (1.13). For instance, considering an example in $d = 100$, the complexity based on equation (1.13) is decreased with a factor around 9 when compared to (1.12). In order to further reduce the computation time compared to the learning approach given in [86], we consider a piecewise-constant learning rate (PC-LR) decay with $\mathfrak{K} = 60000$ optimization steps of the Adam algorithm, with the learning rate α adjusted as follows

$$\alpha_\kappa = \begin{cases} 1\text{e}{-3}, & \text{for } 1 \leq \kappa \leq 20000, \\ 3\text{e}{-4}, & \text{for } 20000 < \kappa \leq 30000, \\ 1\text{e}{-4}, & \text{for } 30000 < \kappa \leq 4000, \\ 3\text{e}{-5}, & \text{for } 40000 < \kappa \leq 50000, \\ 1\text{e}{-5}, & \text{for } 50000 < \kappa \leq \mathfrak{K}. \end{cases}$$

This learning approach is also applied to the DBSDE algorithm, but with a learning rate ranging from $\alpha = 1\text{e}{-2}$ to $\alpha = 1\text{e}{-4}$. A batch size of $B = 128$ is used for each algorithm. Note that when dealing with a forward SDE represented by the Geometric Brownian Motion (GBM), we apply the ln-transformation.

We define the mean squared errors (MSEs) as performance metrics for a sample of size B:

$$\tilde{\varepsilon}_n^y := \frac{1}{B} \sum_{j=1}^{B} \left| Y_{n,j} - Y_{n,j}^{\Delta,\hat{\theta}} \right|^2, \quad \tilde{\varepsilon}_n^z := \frac{1}{B} \sum_{j=1}^{B} \left| Z_{n,j} - Z_{n,j}^{\Delta,\hat{\theta}} \right|^2, \tag{1.14}$$

where $Y_{n,j}$ and $Z_{n,j}$ are the values of the respective processes for sample j at time point t_n. To account the stochasticity of the underlying Brownian motion and the Adam optimizer, we

conduct $Q = 10$ independent runs (trainings) of the algorithms. We then define

$$\overline{\tilde{\varepsilon}}_n^y := \frac{1}{Q}\sum_{q=1}^{Q} \tilde{\epsilon}_{n,q}^y, \quad \overline{\tilde{\varepsilon}}_n^z := \frac{1}{Q}\sum_{q=1}^{Q} \tilde{\epsilon}_{n,q}^z, \tag{1.15}$$

as the mean MSE for the processes Y and Z. As a relative measure of the MSE, we consider

$$\tilde{\varepsilon}_n^{y,r} := \frac{1}{B}\sum_{j=1}^{B} \frac{\left|Y_{n,j} - Y_{n,j}^{\Delta,\hat{\theta}}\right|^2}{|Y_{n,j}|^2}, \quad \tilde{\varepsilon}_n^{z,r} := \frac{1}{B}\sum_{j=1}^{B} \frac{\left|Z_{n,j} - Z_{n,j}^{\Delta,\hat{\theta}}\right|^2}{|Z_{n,j}|^2}, \tag{1.16}$$

for the processes Y and Z, respectively. We select a testing sample of size $B^{test} = 1024$. The computation time (runtime) in seconds for one run of the algorithms is denoted as τ, and the average computation time over $Q = 10$ runs as

$$\overline{\tau} := \frac{1}{Q}\sum_{q=1}^{Q} \tau_q. \tag{1.17}$$

1.5.1 The simple bounded BSDE

We start with an example where the DBSDE method diverges.

Example 1.5.1. *The high-dimensional BSDE given in [44] reads*

$$\begin{cases} dX_t &= a\,dt + b\,dW_t, \\ X_0 &= x_0, \\ -dY_t &= \left(\left(\cos\left(\sum_{k=1}^d X_t^k\right) + 0.2\sin\left(\sum_{k=1}^d X_t^k\right)\right)\exp\left(\frac{T-t}{2}\right)\right. \\ & \quad \left. -\frac{1}{2}\left(\sin\left(\sum_{k=1}^d X_t^k\right)\cos\left(\sum_{k=1}^d X_t^k\right)\exp\left(T-t\right)\right)^2 + \frac{1}{2d}\left(Y_t\sum_{k=1}^d Z_t^k\right)^2\right) dt \\ & \quad -Z_t\,dW_t, \\ Y_T &= \cos\left(\sum_{k=1}^d X_t^k\right). \end{cases}$$

The analytical solution is given by

$$\begin{cases} Y_t &= \exp\left(\frac{T-t}{2}\right)\cos\left(\sum_{k=1}^d X_t^k\right), \\ Z_t &= -b\exp\left(\frac{T-t}{2}\right)\sin\left(\sum_{k=1}^d X_t^k\right)\mathbf{1}_{1,d}. \end{cases}$$

We begin with $d = 1$, $T = 2$, $a = 0.2$, $b = 1$ and $x_0 = 1$. In Table 1.1, we report the mean relative MSE values at t_0 for (Y_0, Z_0) from all the schemes, their average runtime (in seconds) and the empirical convergence rates β using $N \in \{4, 16, 64, 256\}$. The standard deviation (STD) of the relative MSE values at t_0 is given in the brackets. Actually, only a few hundreds optimization steps are needed to achieve a good approximation of (Y_0, Z_0). However, to obtain good approximations for the whole time domain, a high number of optimization steps is needed. From Table 1.1 we see that the DBSDE scheme diverges. The LDBSDE scheme converges to a poor local minima, the mean relative MSEs with $N = 256$ are around 14.20% and 1.64% for Y_0 and Z_0 respectively. Increasing the number of optimization steps ($\mathfrak{K} > 60000$) reduces the (sample) STD, but cannot improve the estimates in the LDBSDE scheme. We may think that each training of the algorithm converges around the same poor local minima. In order to numerically

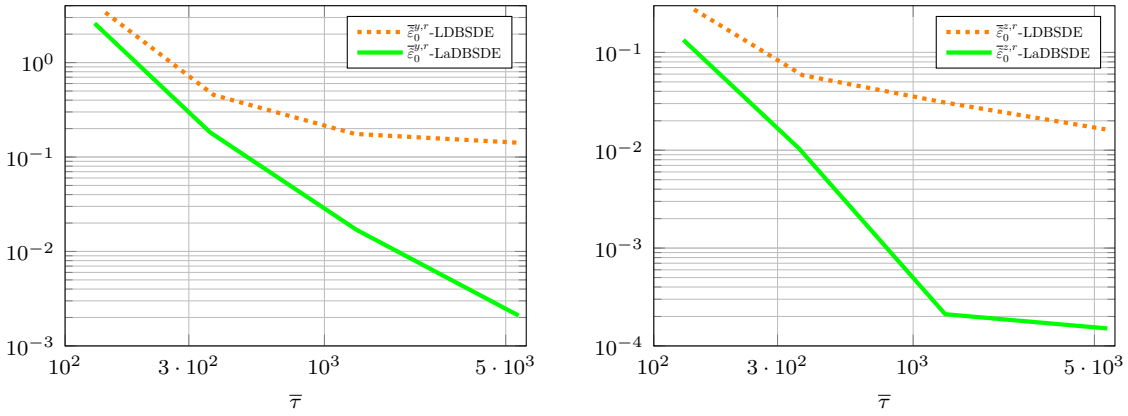
Metric	N = 4 DBSDE LDBSDE LDBSDE (RNN) LDBSDE (LSTM) LaDBSDE	N = 16 DBSDE LDBSDE LDBSDE (RNN) LDBSDE (LSTM) LaDBSDE	N = 64 DBSDE LDBSDE LDBSDE (RNN) LDBSDE (LSTM) LaDBSDE	N = 256 DBSDE LDBSDE LDBSDE (RNN) LDBSDE (LSTM) LaDBSDE	β
$\overline{\varepsilon}_0^{y,r}$	2.89e+00 (2.61e+00) 3.39e+00 (4.03e−02) 3.47e+00 (2.19e−02) 3.46e+00 (4.40e−03) 2.60e+00 (8.13e−03)	NaN 4.54e−01 (3.63e−03) 1.24e+00 (1.58e−01) 1.35e+00 (6.57e−03) 1.82e−01 (1.98e−01)	NaN 1.76e−01 (1.24e−02) 3.43e−01 (1.43e−01) 4.85e−01 (7.27e−02) 1.70e−02 (4.95e−03)	NaN 1.42e−01 (3.82e−02) 2.80e−01 (2.13e−01) 2.25e−01 (4.04e−02) 2.09e−03 (1.16e−03)	– 0.76 0.64 0.67 1.71
$\overline{\varepsilon}_0^{z,r}$	8.52e−01 (4.92e−01) 2.75e−01 (7.35e−03) 2.95e−01 (3.93e−03) 3.00e−01 (1.15e−03) 1.34e−01 (2.08e−03)	NaN 5.86e−02 (4.01e−04) 1.55e−01 (2.65e−02) 1.72e−01 (1.82e−03) 1.04e−02 (1.20e−02)	NaN 3.09e−02 (3.22e−03) 4.74e−02 (2.48e−02) 6.26e−02 (9.94e−03) 2.11e−04 (1.64e−04)	NaN 1.64e−02 (7.97e−03) 4.95e−02 (6.75e−02) 2.53e−02 (4.04e−03) 1.51e−04 (1.69e−04)	– 0.66 0.47 0.61 1.75
$\overline{\tau}$	1.45e+02 1.43e+02 1.04e+02 1.82e+02 1.30e+02	– 3.74e+02 2.46e+02 4.94e+02 3.63e+02	– 1.31e+03 8.06e+02 1.81e+03 1.33e+03	– 5.51e+03 3.33e+03 7.72e+03 5.61e+03	

Table 1.1: Mean relative MSE values, empirical convergence rates of (Y_0, Z_0) from DBSDE, LDBSDE (DNN, RNN or LSTM) and LaDBSDE schemes and their average runtimes in Example 1.5.1 for $d = 1$, $T = 2$ and $N \in \{4, 16, 64, 256\}$. The STD of the relative MSE values at t_0 is given in the brackets.

test that the RNN type architectures do not help the LDBSDE scheme to overcome the issue of poor local minima, we use the RNN and LSTM architectures, which are referred as LDBSDE (RNN) and LDBSDE (LSTM), respectively. Using the LSTM architecture in the LDBSDE scheme, the approximation errors are high, since the LSTM violates the Markovian property of the BSDEs. Even using the RNN in the LDBSDE scheme cannot improve the approximations. The LaDBSDE scheme gives smaller relative errors than the LDBSDE, 0.21% and 0.02% for Y_0 and Z_0, respectively. Moreover, our scheme has better empirical convergence rates than the LDBSDE scheme and achieves higher accuracy for comparable computation time, which holds for the following examples too. The empirical speed of convergence for Y_0 and Z_0 is displayed in Figure 1.2. Note that the approximation of Y_0 in [44] is more accurate than all the schemes (the results for Z_0 are missing) in this example. However, the algorithm in [44] is a backward deep learning scheme, which is based on local optimizations at each time step, see Section 1.4. Its computational cost should be much higher than all the DBSDE, LDBSDE and LaDBSDE schemes.

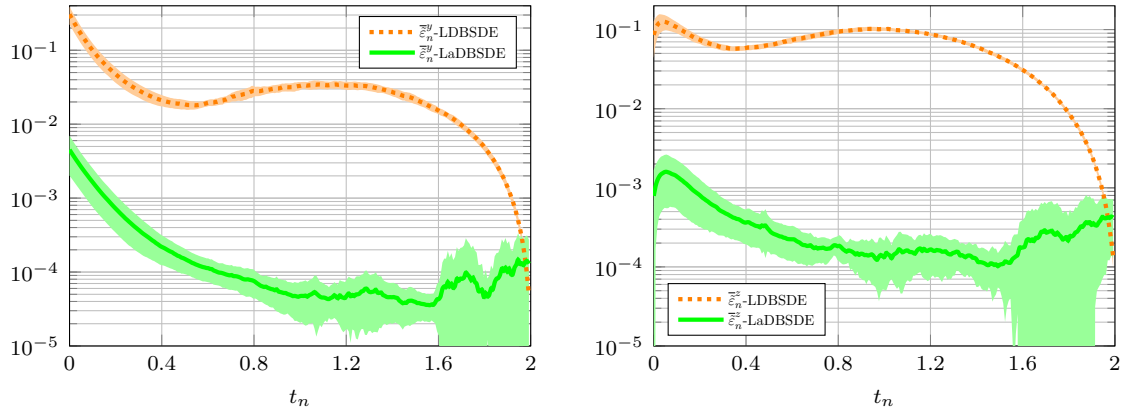
Next we compare the performances of LDBSDE and LaDBSDE for the entire time domain. Hence, using the testing sample across the discrete domain Δ, we visualize in Figure 1.3 the mean MSE values for each process (Y, Z). The STD of the MSE values is given in the shaded area. Note that the approximation for the entire time domain is not discussed in [22], and in [86] only Y is considered. From Figure 1.3 we see that LaDBSDE outperforms the LDBSDE scheme by providing smaller mean MSEs over the discrete time domain.

We consider the high-dimensional case by setting $d = 100$, $T = 1$, $a = \frac{0.2}{d}$, $b = \frac{1}{\sqrt{d}}$ and $x_0 = \mathbf{1}_d$. The mean relative MSE values, the empirical convergence rates for Y_0 and Z_0 and the average runtime of each scheme is reported in Table 1.2 by using $N \in \{2, 8, 32, 128\}$. The STD of the relative MSE values at t_0 is given in the brackets. In Figure 1.4, the empirical speed of convergence is displayed. In contrast to the 1-dimensional case, we observed from our experiments that



(a) Process Y. (b) Process Z.

Figure 1.2: The empirical speed of convergence for (Y_0, Z_0) from LDBSDE and LaDBSDE schemes in Example 1.5.1 for $d = 1$, $T = 2$ and $N \in \{4, 16, 64, 256\}$. The average runtime of the algorithms is given in seconds.



(a) Process Y. (b) Process Z.

Figure 1.3: Mean MSE values of the processes (Y, Z) from LDBSDE and LaDBSDE schemes over the discrete time points $\{t_n\}_{n=0}^{N-1}$ using the testing sample in Example 1.5.1 for $d = 1$, $T = 2$ and $N = 256$. The STD of MSE values is given in the shaded area.

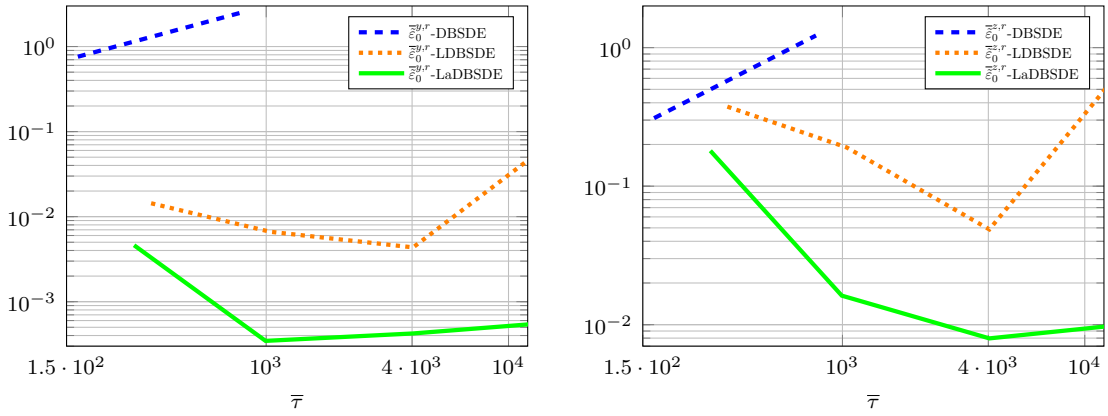
the DBSDE scheme gives good approximations for some trainings in this example for $d = 100$ and maturity $T = 1$, however for the other trainings the scheme diverged. The reason could be that the diffusion reduces due to the large value of dimensionality ($b = \frac{1}{\sqrt{d}}$), and the maturity is shorter than that in the case of one dimension. The DBSDE scheme only diverges by setting $T = 2$. The smallest mean relative MSEs are still given by the LaDBSDE scheme. To compare the approximations for the entire discrete time domain in the high-dimensional case, we display in Figure 1.5 the mean MSE values for each process (Y, Z) using the testing sample with $N = 128$. The STD of the MSE values is given in the shaded area. Our method shows better approximations of processes Y and Z on the entire discrete time domain compared to the LDBSDE scheme.

1.5.2 BSDE with quadratic control

Next we consider an example with a driver function in which the Z process grows quadratically.

Metric	N = 2 DBSDE LDBSDE LaDBSDE	N = 8 DBSDE LDBSDE LaDBSDE	N = 32 DBSDE LDBSDE LaDBSDE	N = 128 DBSDE LDBSDE LaDBSDE	β
$\overline{\varepsilon}_0^{y,r}$	7.59e−01 (4.02e−02) 1.44e−02 (2.36e−03) 4.61e−03 (2.03e−03)	2.48e+00 (4.24e+00) 6.63e−03 (4.38e−03) 3.46e−04 (4.17e−04)	NaN 4.35e−03 (2.38e−03) 4.23e−04 (4.13e−04)	NaN 8.75e−02 (1.38e−02) 5.78e−04 (5.62e−04)	– −0.36 0.44
$\overline{\varepsilon}_0^{z,r}$	3.10e−01 (9.25e−01) 3.75e−01 (2.19e−02) 1.80e−01 (1.23e−02)	1.22e+00 (2.28e+00) 1.91e−01 (2.52e−01) 1.62e−02 (9.84e−03)	NaN 4.87e−02 (2.68e−02) 7.97e−03 (3.79e−03)	NaN 9.24e−01 (2.09e−02) 1.02e−02 (3.91e−03)	– −0.10 0.67
$\overline{\tau}$	1.67e+02 3.35e+02 2.85e+02	7.78e+02 1.04e+03 9.94e+02	– 4.04e+03 4.04e+03	– 1.62e+04 1.62e+04	

Table 1.2: Mean relative MSE values, empirical convergence rates of (Y_0, Z_0) from DBSDE, LDBSDE and LaDBSDE schemes and their average runtimes in Example 1.5.1 for $d = 100$, $T = 1$ and $N \in \{2, 8, 32, 128\}$. The STD of the relative MSE values at t_0 is given in the brackets.



(a) Process Y.

(b) Process Z.

Figure 1.4: The empirical speed of convergence for (Y_0, Z_0) from DBSDE, LDBSDE and LaDBSDE schemes in Example 1.5.1 for $d = 100$, $T = 1$ and $N \in \{2, 8, 32, 128\}$. The average runtime of the algorithms is given in seconds.

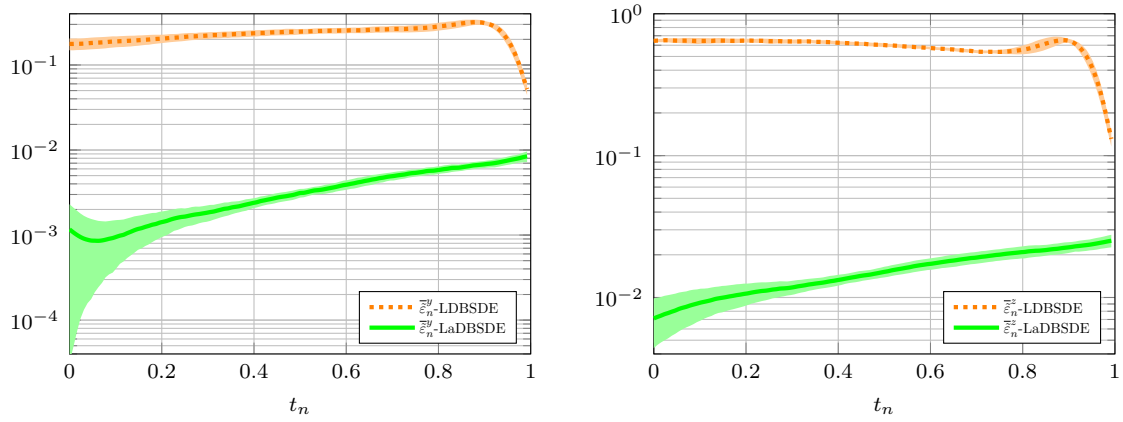
Example 1.5.2. *Consider the nonlinear BSDE (see [37], Section 5)*

$$\begin{cases} -dY_t &= \left(|Z_t|^2 - |\nabla_x \varphi(t, W_t)|^2 - \left(\frac{\partial}{\partial t} + \frac{1}{2}\operatorname{Tr}\left[\operatorname{Hess}_x\right]\right)\varphi(t, W_t)\right) dt - Z_t\, dW_t, \\ Y_T &= \sin\left(|W_T|^{2c}\right), \end{cases}$$

where $\varphi(t, W_t) = \sin\left(\left(T - t + |W_t|^2\right)^c\right)$. The analytic solution is

$$\begin{cases} Y_t &= \sin\left(\left(T - t + |W_t|^2\right)^c\right), \\ Z_t &= 2c\cos\left(\left(T - t + |W_t|^2\right)^c\right)\left(T - t + |W_t|^2\right)^{c-1} W_t^\top. \end{cases}$$

We choose $d = 100$, $T = 1$, $c = 0.4$ and report in Table 1.3 the mean relative MSE values of Y_0 and mean MSE values of Z_0 (since $Z_0 = (0, \ldots, 0)$) for $N \in \{2, 8, 32, 128\}$ from the DBSDE, LDBSDE and LaDBSDE schemes. The empirical convergence rates and the average runtimes of the algorithms are also provided. The STD of the relative MSE and MSE values is given in the brackets. We display the empirical speed of convergence in Figure 1.6. We observe comparable results for all the schemes for Y_0, while our scheme provides better approximations for Z_0.



(a) Process Y.

(b) Process Z.

Figure 1.5: Mean MSE values of the processes (Y, Z) from LDBSDE and LaDBSDE schemes over the discrete time points $\{t_n\}_{n=0}^{N-1}$ using the testing sample in Example 1.5.1 for $d = 100$, $T = 1$ and $N = 128$. The STD of MSE values is given in the shaded area.

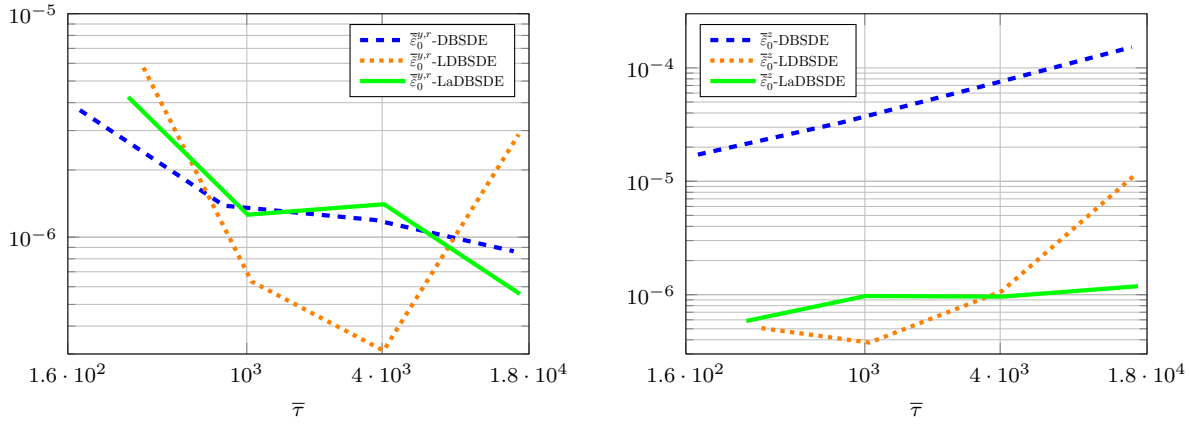
Metric	N = 2 DBSDE LDBSDE LaDBSDE	N = 8 DBSDE LDBSDE LaDBSDE	N = 32 DBSDE LDBSDE LaDBSDE	N = 128 DBSDE LDBSDE LaDBSDE	β
$\overline{\varepsilon}_0^{y,r}$	3.70e−06 (6.80e−07) 5.74e−06 (4.06e−06) 4.24e−06 (4.99e−06)	1.38e−06 (1.01e−06) 6.34e−07 (5.74e−07) 1.26e−06 (1.36e−06)	1.19e−06 (7.91e−07) 3.10e−07 (4.14e−07) 1.40e−06 (1.27e−06)	8.65e−07 (4.56e−07) 2.88e−06 (2.16e−06) 5.57e−07 (5.56e−07)	0.33 0.20 0.43
$\overline{\varepsilon}_0^z$	1.72e−05 (2.00e−06) 5.07e−07 (1.16e−07) 5.86e−07 (2.02e−07)	3.31e−05 (6.15e−06) 3.78e−07 (7.06e−08) 9.73e−07 (1.16e−07)	7.28e−05 (1.44e−05) 1.07e−06 (1.58e−07) 9.60e−07 (2.00e−07)	1.52e−04 (1.50e−05) 1.17e−05 (4.26e−06) 1.19e−06 (4.67e−07)	−0.53 −0.76 −0.15
$\overline{\tau}$	1.81e+02 3.47e+02 2.97e+02	7.93e+02 1.05e+03 1.01e+03	3.72e+03 4.06e+03 4.08e+03	1.54e+04 1.62e+04 1.64e+04	

Table 1.3: Mean relative MSE values of Y_0, mean MSE values of Z_0, empirical convergence rates from DBSDE, LDBSDE and LaDBSDE schemes and their average runtimes in Example 1.5.2 for $d = 100$ and $N \in \{2, 8, 32, 128\}$. The STD of the relative MSE and MSE values at t_0 is given in the brackets.

Moreover, the empirical convergence rates are higher from our scheme. In Figure 1.7, we display the mean MSE values for each process over discrete domain Δ using the testing sample and $N = 128$, where the STD of the MSE values is visualized in the shaded area. We see that the LaDBSDE scheme outperforms.

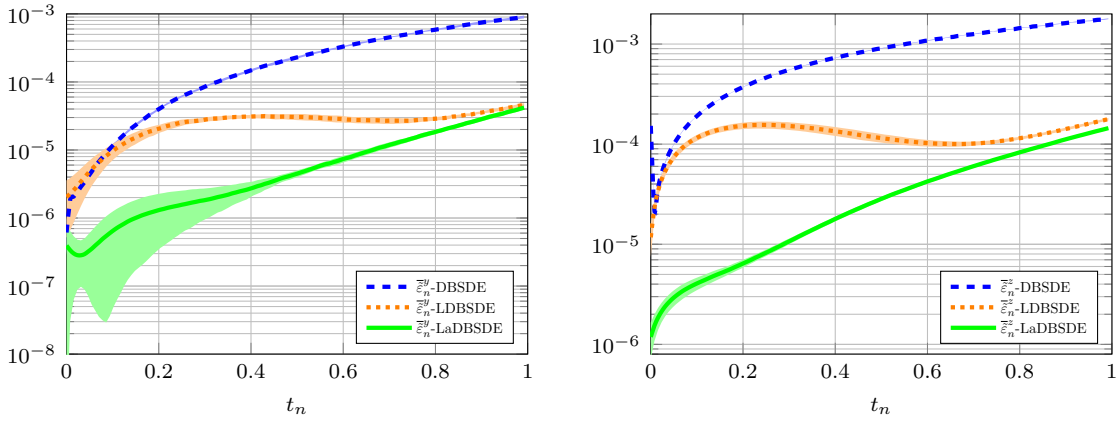
1.5.3 The Black-Scholes-Barenblatt BSDE

For the linear pricing problem we consider the Black-Scholes-Barenblatt type problem studied in [86].



(a) Process Y. (b) Process Z.

Figure 1.6: The empirical speed of convergence for (Y_0, Z_0) from DBSDE, LDBSDE and LaDBSDE schemes in Example 1.5.2 for $d = 100$ and $N \in \{2, 8, 32, 128\}$. The average runtime of the algorithms is given in seconds.



(a) Process Y. (b) Process Z.

Figure 1.7: Mean MSE values of the processes (Y, Z) from DBSDE, LDBSDE and LaDBSDE schemes over the discrete time points $\{t_n\}_{n=0}^{N-1}$ using the testing sample in Example 1.5.2 for $d = 100$ and $N = 128$. The STD of MSE values is given in the shaded area.

Example 1.5.3. *Consider the Black-Scholes-Barenblatt BSDE (see [86], Subsection 4.1)*

$$\begin{cases} dX_t &= bX_t\, dW_t, \\ X_0 &= x_0, \\ -dY_t &= -R\left(Y_t - \frac{1}{b}\sum_{k=1}^{d} Z_t^k\right) dt - Z_t\, dW_t, \\ Y_T &= |X_T|^2, \end{cases}$$

The analytic solution is

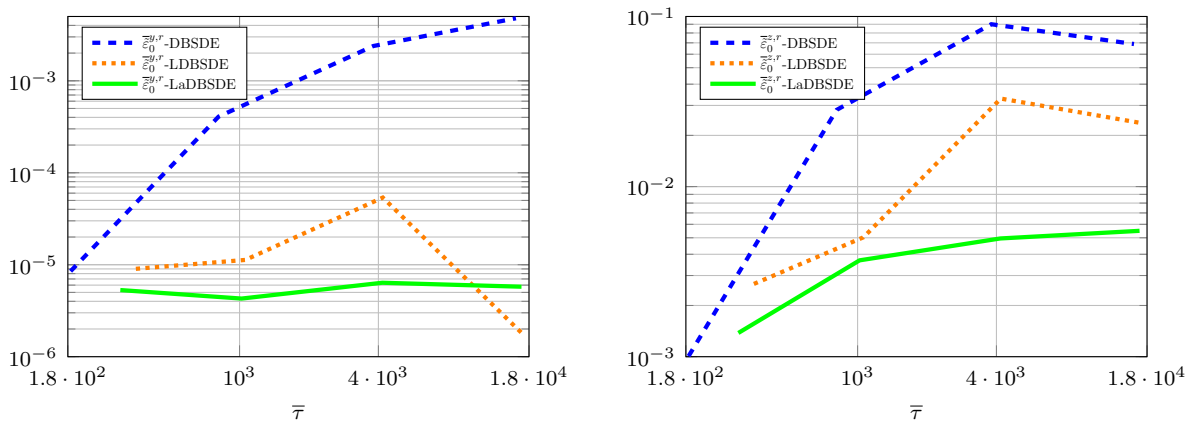
$$\begin{cases} Y_t &= \exp\left(\left(R + b^2\right)(T-t)\right)|X_t|^2, \\ Z_t &= 2b\exp\left(\left(R + b^2\right)(T-t)\right)(X_t^2)^\top. \end{cases}$$

We use $d = 100$, $T = 1$, $R = 0.05$, $b = 0.4$ and $x_0 = (1, 0.5, \ldots, 1, 0.5) \in \mathbb{R}^d$. Note that ln-transform is applied to the forward SDE. In Table 1.4, we report the mean relative MSE values

at t_0 for each process, the corresponding STD (given in the brackets), the empirical convergence rates and the algorithm average runtime using $N \in \{2, 8, 32, 128\}$. In Figure 1.8 we display the empirical speed of convergence. Our scheme provides the smallest mean relative MSEs for

Metric	N = 2 DBSDE LDBSDE LaDBSDE	N = 8 DBSDE LDBSDE LaDBSDE	N = 32 DBSDE LDBSDE LaDBSDE	N = 128 DBSDE LDBSDE LaDBSDE	β
$\overline{\varepsilon}_0^{y,r}$	8.49e−06 (3.48e−07) 9.02e−06 (3.30e−06) 5.30e−06 (2.31e−06)	4.07e−04 (6.63e−06) 1.13e−05 (8.65e−06) 4.28e−06 (3.92e−06)	2.39e−03 (3.06e−05) 5.37e−05 (1.01e−05) 6.37e−06 (6.13e−06)	4.78e−03 (3.19e−05) 1.79e−06 (2.45e−06) 5.76e−06 (4.53e−06)	−1.50 0.24 −0.05
$\overline{\varepsilon}_0^{z,r}$	1.01e−03 (6.07e−05) 2.68e−03 (4.02e−04) 1.38e−03 (3.40e−04)	2.82e−02 (4.99e−04) 5.03e−03 (2.09e−03) 3.68e−03 (8.08e−04)	9.00e−02 (1.41e−03) 3.29e−02 (3.87e−03) 4.96e−03 (8.10e−04)	6.89e−02 (2.13e−03) 2.37e−02 (3.25e−03) 5.49e−03 (9.05e−04)	−1.00 −0.61 −0.32
$\overline{\tau}$	1.85e+02 3.55e+02 3.04e+02	8.11e+02 1.06e+03 1.02e+03	3.80e+03 4.18e+03 4.16e+03	1.58e+04 1.68e+04 1.67e+04	

Table 1.4: Mean relative MSE values, empirical convergence rates of (Y_0, Z_0) from DBSDE, LDBSDE and LaDBSDE schemes and their average runtimes in Example 1.5.3 for $d = 100$ and $N \in \{2, 8, 32, 128\}$. The STD of the relative MSE values at t_0 is given in the brackets.



(a) Process Y.

(b) Process Z.

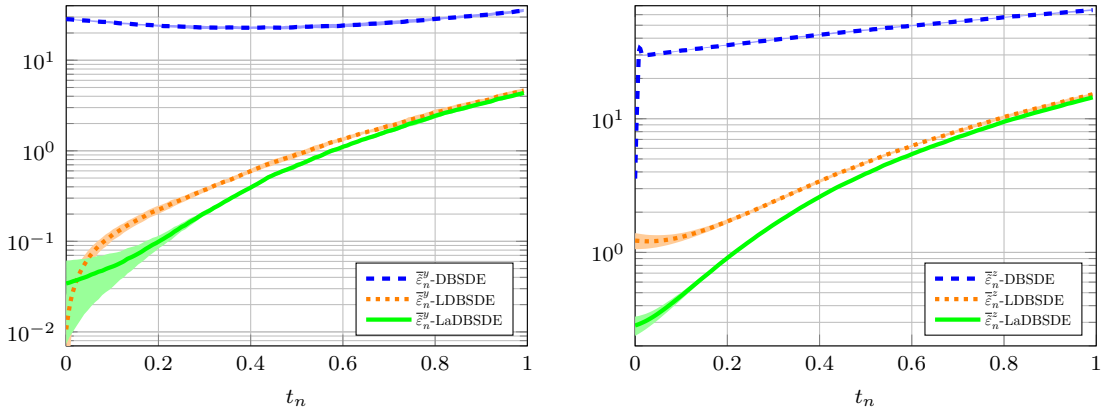
Figure 1.8: The empirical speed of convergence for (Y_0, Z_0) from DBSDE, LDBSDE and LaDBSDE schemes in Example 1.5.3 for $d = 100$ and $N \in \{2, 8, 32, 128\}$. The average runtime of the algorithms is given in seconds.

similar computation time. Using $N = 128$, we visualize the mean MSE values for each process over discrete domain Δ in Figure 1.9 for the testing sample (the STD of the MSE values is displayed in the shaded area). We see that our scheme outperforms the DBSDE and LDBSDE schemes in approximating the processes (Y, Z) over the entire discrete domain Δ.

1.5.4 Option pricing with different interest rates

For the nonlinear pricing problem, we consider pricing options under different interest rates, which has been addressed in e.g., [22, 23, 93, 94].

Example 1.5.4. *Consider the nonlinear pricing with different interest rates (see [10] and, e.g., [22, 23, 93, 94] where this example has been used as a test example for numerical methods*



(a) Process Y. (b) Process Z.

Figure 1.9: Mean MSE values of the processes (Y, Z) from DBSDE, LDBSDE and LaDBSDE schemes over the discrete time points $\{t_n\}_{n=0}^{N-1}$ using the testing sample in Example 1.5.3 for $d = 100$ and $N = 128$. The STD of MSE values is given in the shaded area.

for BSDEs)

$$\begin{cases} dX_t &= aX_t\,dt + bX_t\,dW_t, \\ X_0 &= x_0, \\ -dY_t &= \left(-R_1 Y_t - \frac{a-R_1}{b}\sum_{k=1}^d Z_t^k + (R_2-R_1)\max\left(\frac{1}{b}\sum_{k=1}^d Z_t^k - Y_t, 0\right)\right)dt \\ & \quad -Z_t\,dW_t, \\ Y_T &= \max\left(\max_{k=1,\dots,d}\left(X_T^k - K_1\right), 0\right) - 2\max\left(\max_{k=1,\dots,d}\left(X_T^k - K_2\right), 0\right). \end{cases}$$

The benchmark value with $d = 100$, $T = 0.5$, $a = 0.06$, $b = 0.2$, $R_1 = 0.04$, $R_2 = 0.06$, $K_1 = 120$, $K_2 = 150$ and $x_0 = (100, \dots, 100) \in \mathbb{R}^d$ is $Y_0 \doteq 21.2988$, which is computed using the multilevel Monte Carlo approach [23] with 7 Picard iterations and $Q = 10$ independent runs. For $N \in \{2, 8, 32, 128\}$, we show in Table 1.5 the approximation for Y_0 (the reference results for Z_0 are not available) from all the schemes and their average runtime. Specifically, we report the mean approximation of Y_0 defined as $\overline{Y}_0^{\Delta,\hat{\theta}} := \frac{1}{Q}\sum_{q=1}^Q Y_{0,q}^{\Delta,\hat{\theta}}$ and the mean relative MSE (their STD given in the brackets). The empirical convergence rate β^y is also provided. The empirical speed of convergence is visualized in Figure 1.10. Note that ln-transform is applied to the forward SDE. We see that both the LDBSDE and LaDBSDE schemes cannot perform better than the DBSDE scheme in this example. The reason is that both the driver function $f(t, x, y, z)$ and terminal condition $g(x)$ are not everywhere differentiable. However, our scheme still performs better than the LDBSDE scheme.

1.5.5 BSDE with non-additive diffusion

Finally, we consider an example where the noise in the forward SDE is non-additive.

Example 1.5.5. *Consider the BSDE with space-dependent diffusion coefficients (see [76], Sub-*

Metric	N = 2 DBSDE LDBSDE LaDBSDE	N = 8 DBSDE LDBSDE LaDBSDE	N = 32 DBSDE LDBSDE LaDBSDE	N = 128 DBSDE LDBSDE LaDBSDE	β^y
Y_0 [23]	21.2988				
$\overline{Y}_0^{\Delta,\hat{\theta}}$	21.0963 (2.76e−03) 21.3465 (1.35e−01) 21.4075 (1.50e−02)	21.0934 (3.82e−03) 21.7266 (1.07e−02) 21.4851 (1.07e−02)	21.0938 (6.61e−03) 21.7344 (1.84e−02) 21.5122 (1.83e−02)	21.1513 (1.03e−02) 21.7227 (2.03e−02) 21.5052 (2.01e−02)	
$\overline{\varepsilon}_0^{y,r}$	9.04e−05 (2.46e−06) 4.55e−05 (1.27e−04) 2.65e−05 (7.16e−06)	9.30e−05 (3.46e−06) 4.04e−04 (2.05e−05) 7.68e−05 (9.04e−06)	9.27e−05 (6.04e−06) 4.19e−04 (3.52e−05) 1.01e−04 (1.71e−05)	4.82e−05 (6.80e−06) 3.97e−04 (3.86e−05) 9.48e−05 (1.90e−05)	0.14 −0.47 −0.30
$\overline{\tau}$	1.84e+02 3.54e+02 3.03e+02	8.12e+02 1.07e+03 1.04e+03	3.84e+03 4.22e+03 4.20e+03	1.60e+04 1.69e+04 1.69e+04	

Table 1.5: Mean approximation of Y_0, its mean relative MSE, empirical convergence rate from DBSDE, LDBSDE and LaDBSDE schemes and their average runtimes in Example 1.5.4 for $d = 100$ and $N \in \{2, 8, 32, 128\}$. The STD of the approximations of Y_0 and its relative MSE values are given in the brackets.

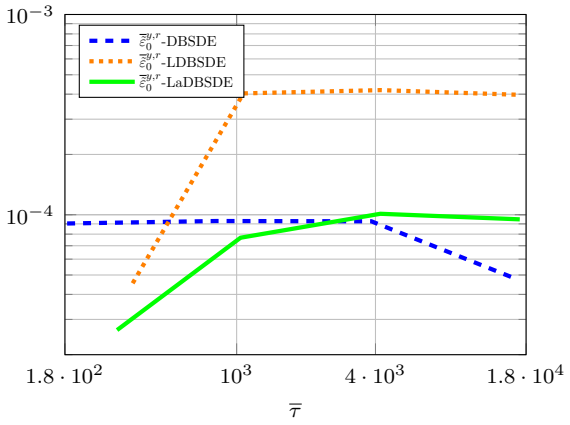


Figure 1.10: The empirical speed of convergence for Y_0 from DBSDE, LDBSDE and LaDBSDE schemes in Example 1.5.4 for $d = 100$ and $N \in \{2, 8, 32, 128\}$. The average runtime of the algorithms is given in seconds.

section 6.3)

$$
\begin{cases}
dX_t &= X_t \frac{1+X_t^2}{\left(2+X_t^2\right)^3}\, dt + \frac{1+X_t^2}{2+X_t^2}\, dW_t, \\
X_0 &= x_0, \\
-dY_t &= \left[\frac{1}{c_1(t+c_2)} \exp\left(-\frac{|X_t|^2}{c_1(t+c_2)}\right)\left(4\sum_{k=1}^d \frac{(X_t^k)^2\left(1+(X_t^k)^2\right)}{\left(2+(X_t^k)^2\right)^3} - \sum_{k=1}^d \frac{(X_t^k)^2}{t+c_2}\right.\right. \\
&\quad \left. + \sum_{k=1}^d \frac{\left(1+(X_t^k)^2\right)^2}{\left(2+(X_t^k)^2\right)^2}\left(1-2\frac{(X_t^k)^2}{c_1(t+c_2)}\right)\right) \\
&\quad \left. + \sqrt{\frac{1+Y_t^2+\exp\left(-2\frac{|X_t|^2}{c_1(t+c_2)}\right)}{1+2Y_t^2}} \sum_{k=1}^d Z_t^k \frac{X_t^k}{\left(2+(X_t^k)^2\right)^2}\right] dt - Z_t\, dW_t, \\
Y_T &= \exp\left(-\frac{|X_T|^2}{c_1(T+c_2)}\right),
\end{cases}
$$

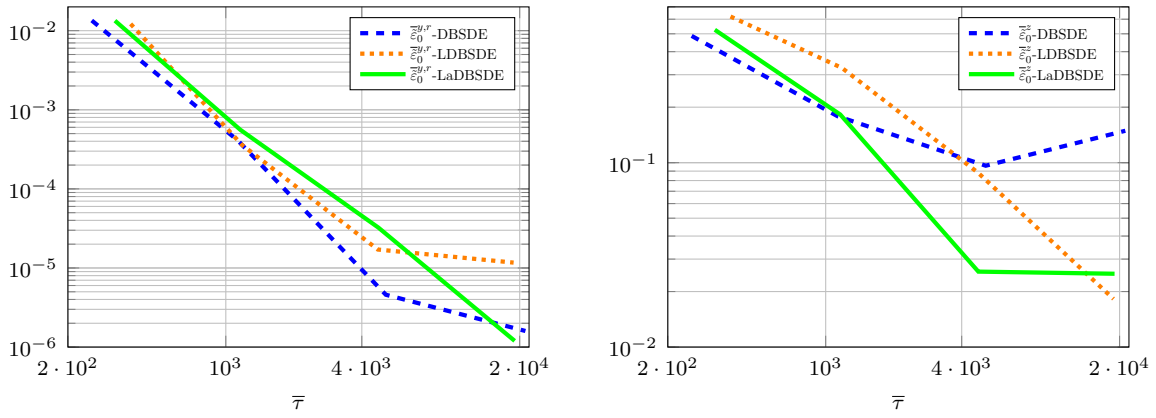
where $c_1, c_2 \in \mathbb{R}_+$. The analytical solution is given by

$$\begin{cases} Y_t &= \exp\left(-\frac{|X_t|^2}{c_1(t+c_2)}\right), \\ Z_t &= -\frac{2}{c_1(t+c_2)} \exp\left(-\frac{|X_t|^2}{c_1(t+c_2)}\right) \left(X_t \frac{1+X_t^2}{2+X_t^2}\right)^\top. \end{cases}$$

We choose $d = 100$, $T = 10$, $c_1 = 10d$, $c_2 = 1$ and $x_0 = \mathbf{1}_d$. In Table 1.6, we report the mean relative MSE values at t_0 for each process from all schemes, using $N \in \{2, 8, 32, 128\}$. The corresponding STD is given in the brackets. The average runtime of the algorithms and the empirical convergence rates are also included. The empirical speed of convergence is displayed in Figure 1.11. We observe that our scheme performs slightly better compared to the other

Metric	N = 2 DBSDE LDBSDE LaDBSDE	N = 8 DBSDE LDBSDE LaDBSDE	N = 32 DBSDE LDBSDE LaDBSDE	N = 128 DBSDE LDBSDE LaDBSDE	β
$\overline{\varepsilon}_0^{y,r}$	1.35e−02 (5.52e−05) 1.22e−02 (1.10e−04) 1.34e−02 (1.27e−04)	4.09e−04 (1.06e−05) 3.60e−04 (1.57e−05) 5.58e−04 (2.15e−05)	4.59e−06 (1.33e−06) 1.71e−05 (5.07e−06) 3.23e−05 (7.65e−06)	1.60e−06 (5.08e−07) 1.17e−05 (8.42e−06) 1.20e−06 (1.26e−06)	2.28 1.72 2.22
$\overline{\varepsilon}_0^{z,r}$	4.87e−01 (1.11e−02) 6.21e−01 (4.60e−03) 5.26e−01 (4.98e−03)	1.79e−01 (1.53e−02) 3.27e−01 (6.63e−03) 1.82e−01 (6.86e−03)	9.63e−02 (1.39e−02) 8.85e−02 (4.65e−03) 2.57e−02 (8.33e−03)	1.49e−01 (2.11e−02) 1.82e−02 (7.72e−03) 2.50e−02 (1.21e−02)	0.30 0.86 0.80
$\overline{\tau}$	2.55e+02 3.79e+02 3.23e+02	1.13e+03 1.18e+03 1.16e+03	5.11e+03 4.71e+03 4.74e+03	2.12e+04 1.88e+04 1.90e+04	

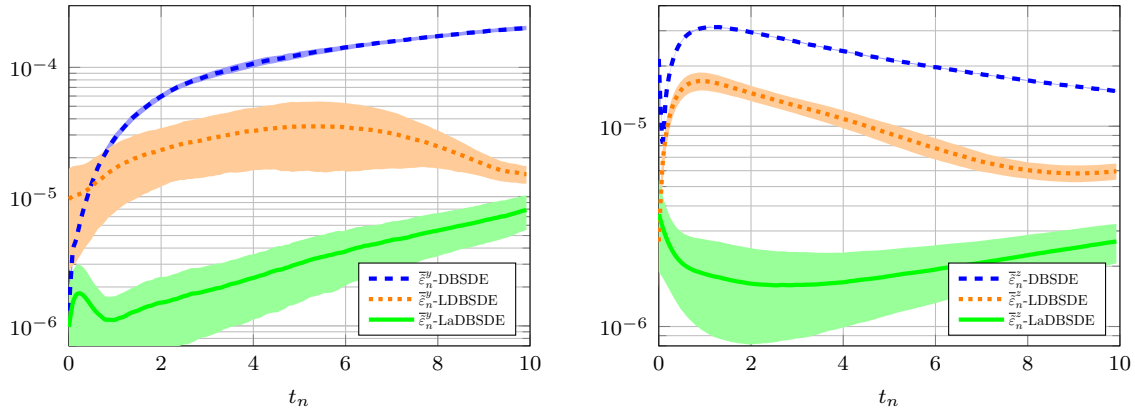
Table 1.6: Mean relative MSE values, empirical convergence rates of (Y_0, Z_0) from DBSDE, LDBSDE and LaDBSDE schemes and their average runtimes in Example 1.5.5 for $d = 100$ and $N \in \{2, 8, 32, 128\}$. The STD of the relative MSE values at t_0 is given in the brackets.



(a) Process Y. (b) Process Z.

Figure 1.11: The empirical speed of convergence for (Y_0, Z_0) from DBSDE, LDBSDE and LaDBSDE schemes in Example 1.5.5 for $d = 100$ and $N \in \{2, 8, 32, 128\}$. The average runtime of the algorithms is given in seconds.

schemes in approximating both Y_0 and Z_0. In Figure 1.12, we display using $N = 128$ the mean MSE values of (Y, Z) at each discrete time point (their STD given in the shaded area) using the testing sample. The LaDBSDE gives the best approximations of both processes on the entire discrete time domain.



(a) Process Y.

(b) Process Z.

Figure 1.12: Mean MSE values of the processes (Y, Z) from DBSDE, LDBSDE and LaDBSDE schemes over the discrete time points $\{t_n\}_{n=0}^{N-1}$ using the testing sample in Example 1.5.5 for $d = 100$ and $N = 128$. The STD of MSE values is given in the shaded area.

1.6 Conclusions

In this chapter we have proposed the LaDBSDE scheme as a forward deep learning algorithm to solve high-dimensional nonlinear BSDEs. It approximates the solution and its gradient based on a global minimization of a novel loss function, which uses local losses defined at each time step including the terminal condition. Our new formulation is achieved by iterating the Euler-Maruyama discretization of integrals with the terminal condition. The numerical results show that the proposed scheme LaDBSDE outperforms the existing forward deep learning schemes [22, 86] in the sense of that it does not get stuck in a poor local minima and provide a good approximation of the solution for the whole time domain.

Building upon the advancements introduced in this chapter, the next chapter introduces a new class of schemes based on differential deep learning [42], aimed to further enhance the accuracy of gradient and Hessian approximations. These improvements are particularly critical in financial applications, where such sensitivities play a pivotal role.

Chapter 2

Differential Deep Learning BSDE Schemes

The aim of this chapter is to introduce a new class of schemes for solving high-dimensional nonlinear BSDEs using differential deep learning [42], which provides higher accurate approximations of the gradient and the Hessian matrix of the solution compared to the class of deep learning schemes discussed in Chapter 1. High-accurate gradient approximations are particularly important in financial applications, where the gradient represents the hedging strategy for an option contract. The Hessian matrix corresponds to Γ sensitivity, which indicates a potential acceleration in changes in the option's value. To formulate the BSDE as a differential deep learning problem, we utilize Malliavin calculus. Thus, we begin by outlining the basics of Malliavin calculus, Malliavin differentiable BSDEs, and the differential deep learning technique. We then introduce a new backward differential deep learning scheme and provide a convergence analysis. Our idea can also be extended to forward deep learning schemes. We present a forward differential deep learning scheme as well. Finally, we demonstrate the efficiency of our proposed scheme through various high-dimensional examples, including option pricing problems, showing that they outperform contemporary deep learning methodologies. This chapter is based on our research presented in [59].

2.1 Introduction

High-accurate gradient approximations are of great significance, for example in finance, where the process Z represents the hedging strategy for an option contract. Except the works in [60, 76], other deep learning-based schemes do not discuss in detail the approximations for Z in high-dimensional spaces, as it is generally more challenging than approximating Y for BSDEs. In this chapter, we develop a class of schemes based on differential deep learning [42] that ensures high accuracy not only for the process Y, but also for the process Z.

Recall that the backward deep learning scheme in [44], namely the DBDP scheme, approximates the unknown solution pair of (1.1) using DNNs. The network parameters are optimized at each time step through the minimization of loss functions defined recursively via backward induction. More precisely, the loss is formulated from the Euler-Maruyama discretization of the BSDE at each time interval. Such formulation gives an implicit approximation of the process Z. Hence, the SGD algorithm lacks explicit information about Z, which impacts its approximation accuracy. To address this, we enhance the SGD algorithm by providing it with additional information to achieve accurate approximations of Z. As stated, we make use of differential deep learning [42],

a general extension of supervised deep learning. In this framework, the DNN model is trained not only on inputs and labels but also on differentials of labels w.r.t. inputs. Differential deep learning offers an efficient approximation not only of the labels but also of their derivatives when compared to traditional supervised deep learning. We use Malliavin calculus to formulate the BSDE problem as a differential deep learning problem. By applying the Malliavin derivative to a BSDE, the Malliavin derivatives of the solution pair (Y, Z) of the BSDE satisfy themselves another BSDE, resulting thus in a system of BSDEs. This formulation also requires estimating the Hessian matrix of the solution. In the context of option pricing, this matrix corresponds to Γ-hedging.

Our backward differential deep learning method works as follows. Firstly, we discretize the system of BSDEs using the Euler-Maruyama method. Subsequently, we utilize DNNs to approximate the unknown solution of these BSDEs, requiring the estimation of the triple of the processes (Y, Z, Γ). The network parameters are optimized backwardly at each time step by minimizing a loss function defined as a weighted sum of the dynamics of the discretized BSDE system. Through this way, SGD is equipped with explicit information about the dynamics of the process Z. As a result, our method can yield more accurate approximations than the DBDP scheme not only for the process Z, but also for the process Γ. The computation time of our scheme is higher compared to the DBDP scheme. Note that the authors in [76] also used the Malliavin derivative to improve the accuracy of Z. However, their method significantly differs from ours, as they only employ supervised deep learning. Their approach requires training the BSDE system separately, which gives a higher computational cost compared to our method. This is demonstrated in our numerical experiments. Furthermore, our approach using differential deep learning can be straightforwardly extended not only to the backward deep learning schemes, but also to forward deep learning schemes [22, 86, 60]. In contrast, the scheme presented in [76] cannot be integrated into such methodologies, as it cannot be formulated as a global optimization problem. To the best of our knowledge, only [71] applies differential deep learning to solve high-dimensional PDEs, where the authors consider the associated dual stochastic control problem instead of working with BSDEs.

The outline of this chapter is organized as follows. In the next section, we review the basics of Malliavin calculus and Malliavin differentiable BSDEs, followed by an introduction to the technique of differential deep learning. Section 2.3 presents the backward differential deep learning algorithm and its convergence analysis. In Section 2.4 we apply differential deep learning to the forward scheme [86]. The numerical experiments presented in Section 2.5 confirm the theoretical results and show high accuracy of the solution, its gradient, and the Hessian matrix of the solution over different option pricing problems. Finally, Section 2.6 concludes this chapter.

2.2 Malliavin differentiable BSDEs and differential deep learning

2.2.1 Malliavin calculus

We shall use techniques of the stochastic calculus of variations. To this end, we use the following notation. For more details, we refer the reader to [79]. Let $\mathcal{S}$ be the space of smooth random variables of the form

$$\xi = \varphi\left(\int_0^T h_1(t)dW_t, \dots, \int_0^T h_d(t)dW_t\right)$$

where $\varphi \in \mathbb{C}_{\mathfrak{p}}^{\infty}\left(\mathbb{R}^d;\mathbb{R}\right)$, $h_1,\dots,h_d \in L^2\left([0,T];\mathbb{R}^{\mathfrak{q}}\right)$. The Malliavin derivative of smooth random variable $\xi \in \mathcal{S}$ is the $\mathbb{R}^{1\times\mathfrak{q}}$-valued stochastic process given by

$$D_s\xi := \sum_{k=1}^{d} \frac{\partial \varphi}{x_k}\left(\int_0^T h_1(t)dW_t,\dots,\int_0^T h_d(t)dW_t\right)h_k(s).$$

We define the domain of D in $\mathbb{L}^2_{\mathcal{F}_T}$ as $\mathbb{D}^{1,2}\left(\Omega;\mathbb{R}\right)$, meaning that $\mathbb{D}^{1,2}$ is the closure of the class of smooth random variables $\mathcal{S}$ w.r.t. the norm

$$\|\xi\|_{\mathbb{D}^{1,2}} := \left(\mathbb{E}\left[|\xi|^2 + \int_0^T |D_s\xi|^2\, ds\right]\right)^{\frac{1}{2}}.$$

Note that in case of vector valued Malliavin differentiable random variables $\xi = (\xi_1,\dots,\xi_{\mathfrak{q}})$, $\xi \in \mathbb{D}^{1,2}\left(\Omega;\mathbb{R}^{\mathfrak{q}}\right)$, its Malliavin derivative $D_s\xi \in \mathbb{R}^{\mathfrak{q}\times\mathfrak{q}}$ is the matrix-valued stochastic process.

The following lemma represents the Malliavin chain rule, which can be extended to Lipschitz continuous functions.

Lemma 2.2.1. *(Malliavin chain rule [79])*
Let $F \in \mathbb{C}_{\mathfrak{b}}^{1}\left(\mathbb{R}^d;\mathbb{R}^{\mathfrak{q}}\right)$. Suppose that $\xi \in \mathbb{D}^{1,2}\left(\Omega;\mathbb{R}^d\right)$. Then $F(\xi) \in \mathbb{D}^{1,2}\left(\Omega;\mathbb{R}^{\mathfrak{q}}\right)$, and for each $0 \le s \le T$

$$D_sF(\xi) = \nabla_x F(\xi) D_s\xi.$$

2.2.2 Malliavin differentiable BSDEs

For the functions in BSDE (1.1), we continue presenting additional assumptions that they are assumed to fulfill when considering Malliavin derivative of BSDE (1.1).

AX3. *Assumption AX2 holds. Moreover, $a \in \mathbb{C}_{\mathfrak{b}}^{0,2}\left([0,T]\times\mathbb{R}^d;\mathbb{R}^d\right)$, $b \in \mathbb{C}_{\mathfrak{b}}^{0,2}\left([0,T]\times\mathbb{R}^d;\mathbb{R}^{d\times d}\right)$ and there exist and positive constant $C > 0$ s.t.*

$$v^\top b(t,x)b(t,x)^\top v \ge C|v|^2, \quad x,v \in \mathbb{R}^d, t \in [0,T].$$

AY3. *Assumption AY2 holds. Moreover, $f \in \mathbb{C}_{\mathfrak{b}}^{0,2,2,2}\left([0,T]\times\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^{1\times d};\mathbb{R}\right)$ and $g \in \mathbb{C}_{\mathfrak{b}}^{2}\left(\mathbb{R}^d;\mathbb{R}\right)$.*

We collect some Malliavin differentiability results on BSDEs, as we are interested on BSDEs s.t. their solution triple $\{X_t,Y_t,Z_t\}_{0\le t\le T}$ is differentiable in the Malliavin sense. The results are stated in the following theorems.

Theorem 2.2.1. *(Malliavin differentiability of SDEs [79])*
Assume that Assumption AX2 holds. Then $\forall\, t \in [0,T]$, $X_t \in \mathbb{D}^{1,2}\left(\Omega;\mathbb{R}^d\right)$ and its Malliavin derivative admits a continuous version $\{D_sX_t\}_{0\le s,t\le T} \in \mathbb{S}^2\left([0,T]\times\Omega;\mathbb{R}^{d\times d}\right)$ satisfying for $0 \le s \le t \le T$ the SDE

$$D_sX_t = \mathbb{1}_{s\le t}\left\{b\left(s,X_s\right) + \int_s^t \nabla_x a\left(r,X_r\right) D_sX_r dr + \int_s^t \nabla_x b\left(r,X_r\right) D_sX_r dW_r\right\},$$

where $\nabla_x b$ denotes a $\mathbb{R}^{d\times d\times d}$-valued tensor. Moreover, there exists a constant $C > 0$ s.t.

$$\sup_{s\in[0,T]} \mathbb{E}\left[\sup_{t\in[s,T]} |D_sX_t|^2\right] \le C, \quad \mathbb{E}\left[|D_sX_r - D_sX_t|^2\right] \le C\,|r-t|\,.$$

Theorem 2.2.2. *(Malliavin differentiability of BSDEs [62])*
Assume that Assumptions AX2 and AY2 holds. Then the solution triple $\{X_t, Y_t, Z_t\}_{0\leq t\leq T}$ *of (1.1) verify that* $\forall\, t \in [0,T]$ $Y_t \in \mathbb{D}^{1,2}(\Omega;\mathbb{R})$, $Z_t \in \mathbb{D}^{1,2}\left(\Omega;\mathbb{R}^{1\times d}\right)$, X *satisfies the statement of Theorem 1.2.3, and a version of* $\{D_sY_t\}_{0\leq s,t\leq T} \in \mathbb{S}^2\left([0,T]\times\Omega;\mathbb{R}^{1\times d}\right)$, $\{D_sZ_t\}_{0\leq s,t\leq T} \in \mathbb{H}^2\left([0,T]\times\Omega;\mathbb{R}^{d\times d}\right)$ *satisfy the BSDE*

$$
\begin{aligned}
&D_sY_t = \mathbf{0}_{1,d}, \quad D_sZ_t = \mathbf{0}_{d,d}, \quad 0\leq t < s \leq T,\\
&D_sY_t = \nabla_x g\left(X_T\right) D_sX_T + \int_t^T \left(\nabla_x f\left(r,\mathbf{X}_r\right) D_sX_r + \nabla_y f\left(r,\mathbf{X}_r\right) D_sY_r + \nabla_z f\left(r,\mathbf{X}_r\right) D_sZ_r\right)dr\\
&\qquad - \int_t^T \left((D_sZ_r)^\top dW_r\right)^\top, \quad 0\leq s\leq t\leq T.
\end{aligned}
$$

Moreover, D_tY_t *defined by the above equation is a version of* Z_t $\mathbb{P}$*-a.s.* $\forall\, t\in[0,T]$.

The final important result that is relevant for this work is the path regularity result of the processes Y and Z, which we state in the following theorem by extending Theorem 1.2.3.

Theorem 2.2.3. *(Path regularity [49])*
Under Assumptions AX3 and AY3 the following holds true:

(i) There exist a constant $C>0$ *s.t.* $\forall\, 0\leq s\leq t\leq T$

$$\mathbb{E}\left[\sup_{s\leq r\leq t}\left|Y_r - Y_s\right|^2\right] \leq C\left|t-s\right|$$

(ii) There exist a constant $C>0$ *s.t.* $\forall\, 0\leq s\leq t\leq T$

$$\mathbb{E}\left[\sup_{s\leq r\leq t}\left|Z_r - Z_s\right|^2\right] \leq C\left|t-s\right|$$

In particular, there exists a continuous modification of the process Z.

2.2.3 Differential deep learning

One of the biggest challenges w.r.t. finding the optimal parameter set of the DNN is to avoid learning training data-specific patterns, namely overfitting, and rather enforcing better generalization of the fitted models. Hence, regularization approaches have been developed for DNNs to avoid overfitting and thus improve the performance of the model. Such approaches penalize certain norms of the parameters θ, expressing a *preference* for θ. Differential deep learning [42] has the same motivation as regularization, namely to improve the accuracy of the model. This is achieved by not expressing a *preference*, but *correctness*, in particular enforcing differential *correctness*. It assumes that the derivative of the label w.r.t. input is known. Let us consider the function $F_x(x) = \nabla_x F(x)$ and the random variable $\mathbf{z} := F_x(\mathbf{x}) \in \mathbb{R}^{1\times d}$. The goal in differential deep learning is to approximate the label function $F(x)$ by DNN $\phi(\mathbf{x};\theta)$ using data $(\mathbf{x},\mathbf{y},\mathbf{z}) \sim p$ and minimizing the extended loss function (1.4) given as

$$\mathbf{L}(\theta) = \mathbb{E}^p\left[\left|\phi\left(\mathbf{x};\theta\right) - \mathbf{y}\right|^2\right] + \lambda\mathbb{E}^p\left[\left|\nabla_x\phi\left(\mathbf{x};\theta\right) - \mathbf{z}\right|^2\right], \tag{2.1}$$

where $\nabla_x\phi$ is calculated using AD and $\lambda \in \mathbb{R}_+$. Our numerical experiments indicated that approximating the derivatives using AD resulted in worse performance compared to utilizing a

separate DNN. This is consistent with the results in [44]. Therefore, we chose to employ a separate DNN for the derivatives, namely we consider a slightly different formulation of differential deep learning compared to [42]. We use one DNN $\phi^y(\mathbf{x};\theta^y)$ to approximate the function $F(x)$ and another $\phi^z(\mathbf{x};\theta^z)$ for $F_x(x)$, and rewrite the loss function (2.1) as

$$\mathbf{L}(\theta) = \lambda_1 \mathbb{E}^p\left[\left|\phi^y(\mathbf{x};\theta^y) - \mathbf{y}\right|^2\right] + \lambda_2 \mathbb{E}^p\left[\left|\phi^z(\mathbf{x};\theta^z) - \mathbf{z}\right|^2\right], \tag{2.2}$$

where $\theta = (\theta^y, \theta^z)$, $\lambda_1, \lambda_2 \in [0,1]$ and $\lambda_1 + \lambda_2 = 1$. Then the optimal parameters θ^* in (2.2) are given as

$$\theta^* \in \arg\min_{\theta\in\Theta} \mathbf{L}(\theta),$$

estimated using a SGD method. Since the derivatives are integrated in the loss function (2.2) as an additional term, we consider this modification to remain within the framework of differential deep learning.

2.3 Backward differential deep learning schemes

In this section, we introduce the proposed backward differential deep learning method and provide its convergence analysis.

2.3.1 The backward differential deep dynamic programming scheme

In order to formulate the BSDE (1.1) as a differential learning problem, we firstly discretize the integrals in the resulting BSDE system given as

$$X_t = x_0 + \int_0^t a(s, X_s)\, ds + \int_0^t b(s, X_s)\, dW_s, \tag{2.3}$$

$$Y_t = g(X_T) + \int_t^T f(s, \mathbf{X}_s)\, ds - \int_t^T Z_s\, dW_s, \tag{2.4}$$

$$D_s X_t = \mathbb{1}_{s\le t}\Big[b(s, X_s) + \int_s^t \nabla_x a(r, X_r)\, D_s X_r dr + \int_s^t \nabla_x b(r, X_r)\, D_s X_r dW_r\Big], \tag{2.5}$$

$$D_s Y_t = \mathbb{1}_{s\le t}\Big[\nabla_x g(X_T)\, D_s X_T + \int_t^T f_D(r, \mathbf{X}_r, \mathbf{D}_s\mathbf{X}_r)\, dr - \int_t^T \Big((D_s Z_r)^\top dW_r\Big)^\top\Big], \tag{2.6}$$

where we introduced the notations $\mathbf{D}_s\mathbf{X}_t := (D_sX_t, D_sY_t, D_sZ_t)$ and $f_D(t, \mathbf{X}_t, \mathbf{D}_s\mathbf{X}_t) := \nabla_x f(t, \mathbf{X}_t)\, D_sX_t + \nabla_y f(t, \mathbf{X}_t)\, D_sY_t + f(t, \mathbf{X}_t)\, D_sZ_t \ \forall\, 0 \le s, t \le T$. Note that the solution of the above BSDE system is a pair of triples of stochastic processes $\{(X_t, Y_t, Z_t)\}_{0\le t\le T}$ and $\{(D_sX_t, D_sY_t, D_sZ_t)\}_{0\le s,t\le T}$ s.t. (2.3)-(2.6) holds $\mathbb{P}$-a.s.

Let us consider the time discretization Δ. For notational convenience we write $(D_nX_m, D_nY_m, D_nZ_m) = (D_{t_n}X_{t_m}, D_{t_n}Y_{t_m}, D_{t_n}Z_{t_m})$ and $\left(D_nX_m^\Delta, D_nY_m^\Delta, D_nZ_m^\Delta\right)$ for the approximations, where $0 \le n, m \le N$. The Euler-Maruyama scheme for the forward SDE (2.3) and the backward process (2.4) is given in Section 1.3 in (1.5) and (1.6), respectively.

Next, we discretize the BSDE for the Malliavin derivatives, i.e. (2.5)-(2.6) in a similar manner. The Malliavin derivative (2.5) approximated by the Euler-Maruyama method gives the estimates

$$D_nX_m^\Delta = \begin{cases} \mathbb{1}_{n=m} b\left(t_n, X_n^\Delta\right), \quad 0 \le m \le n \le N, \\ D_nX_{m-1}^\Delta + \nabla_x a\left(t_{m-1}, X_{m-1}^\Delta\right) D_nX_{m-1}^\Delta \Delta t_{m-1} \\ +\nabla_x b\left(t_{m-1}, X_{m-1}^\Delta\right) D_nX_{m-1}^\Delta \Delta W_{m-1}, \quad 0 \le n < m \le N. \end{cases} \tag{2.7}$$

From $[t_n, t_{n+1}]$ (2.6) is given as

$$D_n Y_n = D_n Y_{n+1} + \int_{t_n}^{t_{n+1}} f_D\left(s, \mathbf{X}_s, \mathbf{D}_n \mathbf{X}_s\right) ds - \int_{t_n}^{t_{n+1}} \left(\left(D_n Z_s\right)^\top dW_s\right)^\top. \tag{2.8}$$

Using Euler-Maruyama scheme in (2.8), we get

$$D_n Y_n^\Delta = D_n Y_{n+1}^\Delta + f_D\left(t_n, \mathbf{X}_n^\Delta, \mathbf{D}_n \mathbf{X}_n^\Delta\right) \Delta t_n - \left(\left(D_n Z_n^\Delta\right)^\top \Delta W_n\right)^\top, \tag{2.9}$$

with $\mathbf{D}_n \mathbf{X}_n^\Delta := \left(D_n X_n^\Delta, D_n Y_n^\Delta, D_n Z_n^\Delta\right)$. Given the Markov property of the underlying processes, the Malliavin chain rule (Lemma 2.2.1) implies that

$$D_n Y_m = \nabla_x y\left(t_m, X_m\right) D_n X_m, \quad D_n Z_m = \nabla_x z\left(t_m, X_m\right) D_n X_m =: \gamma\left(t_m, X_m\right) D_n X_m, \tag{2.10}$$

for some deterministic functions $y : [0,T] \times \mathbb{R}^d \to \mathbb{R}$ and $z : [0,T] \times \mathbb{R}^d \to \mathbb{R}^{1\times d}$, where $\gamma : [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d\times d}$ is the Jacobian matrix of $z\left(t_m, X_m\right)$. Note that from the Feynman-Kac relation (1.3) we have that $z\left(t_m, X_m\right) = \nabla_x y\left(t_m, X_m\right) b\left(t_m, X_m\right)$. Hence, one can write that $D_n Y_m = z\left(t_m, X_m\right) b^{-1}\left(t_m, X_m\right) D_n X_m$. Using Theorem 2.2.2, we have that (2.9) becomes

$$Z_n^\Delta = Z_{n+1}^\Delta b^{-1}\left(t_{n+1}, X_{n+1}^\Delta\right) D_n X_{n+1}^\Delta + f_D\left(t_n, \mathbf{X}_n^\Delta, \mathbf{D}_n \mathbf{X}_n^\Delta\right) \Delta t_n - \left(\left(\Gamma_n^\Delta D_n X_n^\Delta\right)^\top \Delta W_n\right)^\top, \tag{2.11}$$

where due to the aforementioned relations $f_D\left(t, \mathbf{X}_n^\Delta, \mathbf{D}_n \mathbf{X}_n^\Delta\right) = \nabla_x f\left(t_n, \mathbf{X}_n^\Delta\right) D_n X_n^\Delta + \nabla_y f\left(t_n, \mathbf{X}_n^\Delta\right) Z_n^\Delta + \nabla_z f\left(t_n, \mathbf{X}_n^\Delta\right) \Gamma_n^\Delta D_n X_n^\Delta$.

After discretizing the integrals, our scheme is made fully implementable at each discrete time point t_n by an appropriate function approximator to estimate the discrete unknown processes $\left(Y_n^\Delta, Z_n^\Delta, \Gamma_n^\Delta\right)$ in (1.6) and (2.11). We estimate these unknown processes using DNNs and propose the following scheme:

- Generate approximations X_{n+1}^Δ for $n = 0, 1, \ldots, N-1$ via (1.5) and its discrete Malliavin derivative $D_n X_n^\Delta$, $D_n X_{n+1}^\Delta$ using (2.7).
- Set

$$Y_N^{\Delta,\hat{\theta}} := g(X_N^\Delta), \quad Z_N^{\Delta,\hat{\theta}} := \nabla_x g(X_N^\Delta) b(t_N, X_N^\Delta), \quad \Gamma_N^{\Delta,\hat{\theta}} := \left[\nabla_x (\nabla_x g\, b)\right](t_N, X_N^\Delta).$$

- For each discrete time point t_n, $n = N-1, N-2, \ldots, 0$, use three DNNs $\phi_n^y(\cdot;\theta_n^y) : \mathbb{R}^d \to \mathbb{R}$, $\phi_n^z(\cdot;\theta_n^z) : \mathbb{R}^d \to \mathbb{R}^{1\times d}$ and $\phi_n^\gamma(\cdot;\theta_n^\gamma) : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ to approximate the discrete processes $\left(Y_n^\Delta, Z_n^\Delta, \Gamma_n^\Delta\right)$, respectively, where the input vector of the network is the Markovian process $X_n^\Delta \in \mathbb{R}^d$. More precisely

$$Y_n^{\Delta,\theta} = \phi_n^y(X_n^\Delta;\theta_n^y), \quad Z_n^{\Delta,\theta} = \phi_n^z(X_n^\Delta;\theta_n^z), \quad \Gamma_n^{\Delta,\theta} = \phi_n^\gamma(X_n^\Delta;\theta_n^\gamma).$$

- Train the parameter set $\theta_n = (\theta_n^y, \theta_n^z, \theta_n^\gamma)$ using the differential learning approach by constructing a loss function - as in (2.2) - s.t. the dynamics of the discretized process Y and Z given by (1.6) and (2.11) are fulfilled, namely

$$\begin{aligned}
\mathbf{L}_n^\Delta\left(\theta_n\right) &:= \lambda_1 \mathbf{L}_n^{y,\Delta}\left(\theta_n\right) + \lambda_2 \mathbf{L}_n^{z,\Delta}\left(\theta_n\right), \\
\mathbf{L}_n^{y,\Delta}\left(\theta_n\right) &= \mathbb{E}\left[\left|Y_{n+1}^{\Delta,\hat{\theta}} - Y_n^{\Delta,\theta} + f\left(t_n, \mathbf{X}_n^{\Delta,\theta}\right) \Delta t_n - Z_n^{\Delta,\theta} \Delta W_n\right|^2\right], \\
\mathbf{L}_n^{z,\Delta}\left(\theta_n\right) &:= \mathbb{E}\left[\left|Z_{n+1}^{\Delta,\hat{\theta}} b^{-1}\left(t_{n+1}, X_{n+1}^\Delta\right) D_n X_{n+1}^\Delta - Z_n^{\Delta,\theta}\right.\right. \\
&\qquad \left.\left. + f_D\left(t_n, \mathbf{X}_n^{\Delta,\theta}, \mathbf{D}_n \mathbf{X}_n^{\Delta,\theta}\right) \Delta t_n - \left(\left(\Gamma_n^{\Delta,\theta} D_n X_n^\Delta\right)^\top \Delta W_n\right)^\top\right|^2\right],
\end{aligned} \tag{2.12}$$

where for notational convinience $\mathbf{D}_n\mathbf{X}_n^{\Delta,\theta} := \left(D_nX_n^\Delta, Z_n^{\Delta,\theta}, \Gamma_n^{\Delta,\theta} D_nX_n^\Delta\right)$.

- Approximate the optimal parameters $\theta_n^* \in \arg\min_{\theta_n\in\Theta_n} \mathbf{L}_n^\Delta(\theta_n)$ using a SGD method and receive the estimated parameters $\hat{\theta}_n = \left(\hat{\theta}_n^y, \hat{\theta}_n^z, \hat{\theta}_n^\gamma\right)$. Then, define

$$Y_n^{\Delta,\hat{\theta}} := \phi_n^y\left(X_n^\Delta; \hat{\theta}_n^y\right), \quad Z_n^{\Delta,\hat{\theta}} := \phi_n^z\left(X_n^\Delta; \hat{\theta}_n^z\right), \quad \Gamma_n^{\Delta,\hat{\theta}} := \phi_n^\gamma\left(X_n^\Delta; \hat{\theta}_n^\gamma\right).$$

Note that the DBDP scheme in [44] can be considered as a special case of our scheme by choosing $\lambda_1 = 1$ and $\lambda_2 = 0$. We refer to our scheme as DLBDP (differential learning backward dynamic programming) scheme, where $\lambda_1 = \frac{1}{d+1}$ and $\lambda_2 = \frac{d}{d+1}$ are considered due to dimensionality of the processes Y and Z, a common practice in differential deep learning [42].

Our scheme offers several advantages over the DBDP scheme and other well-known deep learning approaches [22, 30, 86, 60]:

(i) By explicitly incorporating the dynamics of the process Z via the BSDE (2.6) in the loss function (2.12), we enhance the accuracy of Z approximations through the SGD method.

(ii) Additionally, the inclusion of the process Γ in the loss function through BSDE (2.6) allows for better estimation of Γ within the DLBDP scheme compared to the deep learning-based schemes, where AD is required for approximation of Γ.

The scheme in [76] – called the one step Malliavin (OSM) scheme – also uses the Malliavin derivative to improve the accuracy of Z in the DBDP method. Hence, in the numerical experiments, we compare our approach with both the schemes DBDP and OSM. The latter can be formulated as follows:

- Generate approximations X_{n+1}^Δ for $n = 0, 1, \ldots, N-1$ of SDE part in (2.3) via (1.5) and its discrete Malliavin derivative $D_nX_n^\Delta$, $D_nX_{n+1}^\Delta$ using (2.7).

- Set

$$Y_N^{\Delta,\hat{\theta}} = g(X_N^\Delta), \quad Z_N^{\Delta,\hat{\theta}} = \nabla_x g(X_N^\Delta) b(t_N, X_N^\Delta), \quad \Gamma_N^{\Delta,\hat{\theta}} = \left[\nabla_x(\nabla_x g\, b)\right](t_N, X_N^\Delta).$$

- For each discrete time point t_n, $n = N-1, N-2, \ldots, 0$, consider two optimization problems. In the first one, use two independent DNNs $\phi_n^z(\cdot;\theta_n^z) : \mathbb{R}^d \to \mathbb{R}^{1\times d}$ and $\phi_n^\gamma(\cdot;\theta_n^\gamma) : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ to approximate the discrete processes $\left(Z_n^\Delta, \Gamma_n^\Delta\right)$, respectively, where the input vector of the network is the Markovian process $X_n^\Delta \in \mathbb{R}^d$. More precisely

$$Z_n^{\Delta,\theta} = \phi_n^z(X_n^\Delta; \theta_n^z), \quad \Gamma_n^{\Delta,\theta} = \phi_n^\gamma(X_n^\Delta; \theta_n^\gamma).$$

- Train the parameter set $\theta_n = (\theta_n^z, \theta_n^\gamma)$ using a loss function such that the dynamics of the discretized process Z given by (2.11) (with the Malliavin derivative of the driver function evaluated at time points t_n and t_{n+1} [76]) are fulfilled, namely

$$\mathbf{L}_n^{z,\Delta}(\theta_n) = \mathbb{E}\left[\left| Z_{n+1}^{\Delta,\hat{\theta}} b^{-1}\left(t_{n+1}, X_{n+1}^\Delta\right) D_nX_{n+1}^\Delta - Z_n^{\Delta,\theta} \right.\right.$$
$$\left.\left. + f_D\left(t_{n+1}, \mathbf{X}_{n+1}^{\Delta,\hat{\theta}}, \mathbf{D}_n\mathbf{X}_{n+1,n}^{\Delta,\hat{\theta}}\right)\Delta t_n - \left(\left(\Gamma_n^{\Delta,\theta} D_nX_n^\Delta\right)^\top \Delta W_n\right)^\top \right|^2\right],$$

where $\mathbf{X}_{n+1}^{\Delta,\hat{\theta}} = \left(X_{n+1}^{\Delta}, Y_{n+1}^{\Delta,\hat{\theta}}, Z_{n+1}^{\Delta,\hat{\theta}}\right)$ and $\mathbf{D}_n\mathbf{X}_{n+1,n}^{\Delta,\hat{\theta}} := \left(D_nX_{n+1}^{\Delta}, D_nY_{n+1}^{\Delta,\hat{\theta}}, \Gamma_n^{\Delta,\theta}D_nX_n^{\Delta}\right)$. Approximate the optimal parameters $\theta_n^* \in \arg\min_{\theta_n\in\Theta_n} \mathbf{L}_n^{z,\Delta}(\theta_n)$ using a SGD method and receive the estimated parameters $\hat{\theta}_n = \left(\hat{\theta}_n^z, \hat{\theta}_n^\gamma\right)$. Then, define

$$Z_n^{\Delta,\hat{\theta}} = \phi_n^z\left(X_n^{\Delta}; \hat{\theta}_n^z\right), \quad \Gamma_n^{\Delta,\hat{\theta}} = \phi_n^\gamma\left(X_n^{\Delta}; \hat{\theta}_n^\gamma\right).$$

- For the second optimization problem, use another DNN $\phi_n^y(\cdot;\theta_n^y) : \mathbb{R}^d \to \mathbb{R}^{1\times d}$ to approximate the discrete processes Y_n^{Δ}, namely

$$Y_n^{\Delta,\theta} = \phi_n^y(X_n^{\Delta};\theta_n^y).$$

  Train the parameters θ_n^y using a loss function such that the dynamics of the discretized process Y given by (1.6) are fulfilled, namely

$$\mathbf{L}_n^{y,\Delta}(\theta_n^y) = \mathbb{E}\left[\left|Y_{n+1}^{\Delta,\hat{\theta}} - Y_n^{\Delta,\theta} + f\left(t_n, X_n^{\Delta}, Y_n^{\Delta,\theta}, Z_n^{\Delta,\hat{\theta}}\right)\Delta t_n - Z_n^{\Delta,\hat{\theta}}\Delta W_n\right|^2\right].$$

  Approximate the optimal parameters $\theta_n^{*,y} \in \arg\min_{\theta_n^y\in\Theta_n^y} \mathbf{L}_n^{y,\Delta}(\theta_n^y)$ using a SGD method and receive the estimated parameters $\hat{\theta}_n^y$. Then, define

$$Y_n^{\Delta,\hat{\theta}} = \phi_n^y\left(X_n^{\Delta}; \hat{\theta}_n^y\right).$$

In comparison to the OSM scheme, our approach demonstrates the following advantages:

(i) Since the OSM scheme employs supervised deep learning, it requires solving two optimization problems per time step - one for BSDE (2.4) and another for BSDE (2.6) - to approximate the unknown processes (Y, Z, Γ). Consequently, the computational cost of the OSM scheme is up to twice as high as that of our scheme. This is demonstrated in our numerical experiments.

(ii) Our scheme can be seamlessly extended not only to the backward deep learning approaches (as the DBDP scheme), but also to forward deep learning approaches, such as [22, 86, 60]. This is discussed in Section 2.4. The OSM scheme cannot be integrated to such schemes, as it cannot be formulated as a global optimization problem.

2.3.2 Convergence analysis

The main goal of this section is to prove the convergence of the DLBDP scheme towards the solution (Y, Z, Γ) of the BSDE system (2.3)-(2.6), and provide a rate of convergence that depends on the discretization error from the Euler-Maruyama scheme and the approximation or model error by the DNNs.

For the functions figuring in the BSDE system (2.3)-(2.6), the following assumptions are in place.

AX4. *Assumption AX3 holds, with the Malliavin derivative $|D_sb(t, X_t)| \le C$ $\mathbb{P}$-a.s. for $0 \le s \le t \le T$. The functions $a(t,x)$ and $b(t,x)$ are $1/2$-Hölder continuous in time.*

AY4. *Assumption AY3 holds. Moreover, $g \in \mathbb{C}_{\mathfrak{b}}^{2+\mathfrak{l}}\left(\mathbb{R}^d;\mathbb{R}\right)$, $\mathfrak{l} > 0$. The function $f(t,x,y,z)$ and its partial derivatives $\nabla_x f$, $\nabla_y f$ and $\nabla_z f$ are all $1/2$-Hölder continuous in time.*

We emphasise that our assumption for the SDE is less restrictive than that in [76], where arithmetic Brownian motion (ABM) is assumed. When pricing and hedging options, usually the stock dynamics are modeled by the GBM. To ensure the applicability of our convergence analysis in such cases, we consider in the numerical section the ln-transformation of stock prices. Consequently, we obtain a drift and diffusion function that satisfy Assumption AX4, thereby ensuring that our theoretical analysis holds in the numerical experiments. Moreover, in case of more advanced models than the GBM, if the Malliavin derivative of $b(t, X_t)$ is bounded, our analysis still holds.

The following lemma is a consequence of the considered assumptions.

Lemma 2.3.1. *Under Assumptions AX4 and AY4, the Malliavin derivatives* $(D_s X_t, D_s Y_t, D_s Z_t)$ *are bounded* $\mathbb{P}$*-a.s. for* $0 \leq s \leq t \leq T$.

Proof. Due to Assumption AX4, we have that $|D_s X_t| \leq C$ $\mathbb{P}$-a.s. for $0 \leq s \leq t \leq T$ using [16], Lemma 4.2 as $|D_s b(t, X_t)| \leq C$ $\mathbb{P}$-a.s. for $0 \leq s \leq t \leq T$. Moreover, the parabolic PDE (1.3) has a classical solution $u \in \mathbb{C}_{\mathfrak{b}}^{1,2}\left([0,T] \times \mathbb{R}^d; \mathbb{R}\right)$ (see [21], Theorem 2.1). The boundedness of $(D_s Y_t, D_s Z_t)$ follows after using the relations (2.10). □

From the mean-value theorem, for $f \in \mathbb{C}_{\mathfrak{b}}^{0,2,2,2}\left([0,T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{1\times d}; \mathbb{R}\right)$, we have that f and all its first-order derivatives in (x, y, z) are Lipschitz continuous. Therefore, the following holds (using also Assumption AY4 and Lemma 2.3.1)

$$\begin{aligned} |f(t_1, \mathbf{x}_1) - f(t_2, \mathbf{x}_2)| &\leq L_f \left(|t_1 - t_2|^{\frac{1}{2}} + |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| \right), \\ |f_D(t_1, \mathbf{x}_1, \mathbf{D}\mathbf{x}_1) - f_D(t_2, \mathbf{x}_2, \mathbf{D}\mathbf{x}_2)| &\leq L_{f_D} \Big(|t_1 - t_2|^{\frac{1}{2}} + |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| \\ &\quad + |Dx_1 - Dx_2| + |Dy_1 - Dy_2| + |Dz_1 - Dz_2| \Big), \end{aligned} \tag{2.13}$$

with $\mathbf{x}_{\mathfrak{i}} = (x_{\mathfrak{i}}, y_{\mathfrak{i}}, z_{\mathfrak{i}})$, $\mathbf{D}\mathbf{x}_{\mathfrak{i}} = (Dx_{\mathfrak{i}}, Dy_{\mathfrak{i}}, Dz_{\mathfrak{i}})$ and $t_{\mathfrak{i}} \in [0,T]$, $x_{\mathfrak{i}} \in \mathbb{R}^d$, $y_{\mathfrak{i}} \in \mathbb{R}$, $z_{\mathfrak{i}}, Dy_{\mathfrak{i}} \in \mathbb{R}^{1\times d}$, $Dx_{\mathfrak{i}}, Dz_{\mathfrak{i}} \in \mathbb{R}^{d\times d}$, where $L_f, L_{f_D} > 0$ and $\mathfrak{i} = 1, 2$.

Under Assumptions AX4 and AY4, by Theorems 1.2.1, 2.2.1 and 2.2.3, we have that the processes (X, Y, Z, DX, DY) are all mean-squared continuous in time, more specific, there exists some constant $C > 0$ s.t. $\forall\, s, r, t \in [0,T]$

$$\begin{aligned} &\mathbb{E}\left[|X_t - X_s|^2\right] \leq C\,|t-s|\,, \quad \mathbb{E}\left[|D_s X_t - D_s X_r|^2\right] \leq C\,|t-r|\,, \\ &\mathbb{E}\left[|Y_t - Y_s|^2\right] \leq C\,|t-s|\,, \quad \mathbb{E}\left[|Z_t - Z_s|^2\right] \leq C\,|t-s|\,, \quad \mathbb{E}\left[|D_s Y_t - D_s Y_r|^2\right] \leq C\,|t-r|\,. \end{aligned} \tag{2.14}$$

From Assumptions AX4, AY4 and Lemma 2.3.1, we also see for $0 \leq s \leq t \leq T$ that

$$\mathbb{E}\left[\int_0^T |f_D\left(t, \mathbf{X}_t, \mathbf{D}_s \mathbf{X}_t\right)|^2 \, dt\right] < \infty. \tag{2.15}$$

Moreover, we have the well-known error estimate that the Euler-Maruyama approximations in (1.5) admit to

$$\max_{0\leq n\leq N-1} \mathbb{E}\left[\sup_{t_n \leq t \leq t_{n+1}} \left|X_t - X_{t_n}^{\Delta}\right|^2\right] = \mathcal{O}\left(|\Delta|\right), \tag{2.16}$$

under Assumption AX1 and the Hölder continuity assumption in AX4 (see [102], Theorem 5.3.1), where the notation $\mathcal{O}\left(|\Delta|\right)$ means that $\limsup_{|\Delta|\to 0} |\Delta|^{-1}\, \mathcal{O}\left(|\Delta|\right) < \infty$. Note that under

Assumption AX2 and the Hölder continuity assumption in AX4, it can be showed that the Euler-Maruyama Malliavin derivative approximations $D_n X^{\Delta}_{n+1}$ in (2.7) admit to similar error estimates as in (2.16)

$$\mathbb{E}\left[\left|D_n X_{n+1} - D_n X^{\Delta}_{n+1}\right|^2\right] = \mathcal{O}\left(|\Delta|\right). \tag{2.17}$$

Let us introduce the $\mathbb{L}^2$-regularity of DZ:

$$\varepsilon^{DZ}\left(|\Delta|\right) := \mathbb{E}\left[\sum_{n=0}^{N-1}\int_{t_n}^{t_{n+1}}\left|D_n Z_r - \widehat{DZ}_n\right|^2 dr\right], \tag{2.18}$$

with

$$\widehat{DZ}_n := \frac{1}{\Delta t_n}\mathbb{E}_n\left[\int_{t_n}^{t_{n+1}} D_n Z_s ds\right]$$

the $\mathbb{L}^2$-projection of the corresponding Malliavin derivative w.r.t. the $\mathcal{F}_{t_n}$ σ-algebra. Based on relations in (2.10), we have that

$$D_n Z_r = \Gamma_r D_n X_r, \quad D_m Z_r = \Gamma_r D_m X_r, \quad \forall\ t_n, t_m < r.$$

Subsequently, using Lemma 2.3.1, the mean-squared continuity in time of DX given by Theorem 2.2.1, and that the terminal condition of the Malliavin BSDE (2.6) is Lipschitz continuous (due to Assumption AY4), we have that

$$\varepsilon^{DZ}\left(|\Delta|\right) = \mathcal{O}\left(|\Delta|\right),$$

after applying [101] (Theorem 3.1).

We now define

$$\begin{cases} \hat{Y}^{\Delta}_n & := \mathbb{E}_n\left[Y^{\Delta,\hat{\theta}}_{n+1}\right] + f\left(t_n, \hat{\mathbf{X}}^{\Delta}_n\right)\Delta t_n, \\ \hat{Z}^{\Delta}_n & := \mathbb{E}_n\left[Z^{\Delta,\hat{\theta}}_{n+1} b^{-1}\left(t_{n+1}, X^{\Delta}_{n+1}\right) D_n X^{\Delta}_{n+1}\right] + f_D\left(t_n, \hat{\mathbf{X}}^{\Delta}_n, \mathbf{D}_n\hat{\mathbf{X}}^{\Delta}_n\right)\Delta t_n, \\ \hat{\Gamma}^{\Delta}_n & := \frac{1}{\Delta t_n}\mathbb{E}_n\left[\Delta W_n Z^{\Delta,\hat{\theta}}_{n+1} b^{-1}\left(t_{n+1}, X^{\Delta}_{n+1}\right) D_n X^{\Delta}_{n+1}\right] b^{-1}\left(t_n, X^{\Delta}_n\right), \end{cases} \tag{2.19}$$

for $n = 0, \ldots, N-1$, where $\hat{\mathbf{X}}_n := \left(X^{\Delta}_n, \hat{Y}^{\Delta}_n, \hat{Z}^{\Delta}_n\right)$ and $\mathbf{D}_n\hat{\mathbf{X}}_n := \left(D_n X^{\Delta}_n, \hat{Z}^{\Delta}_n, \hat{\Gamma}^{\Delta}_n b(t_n, X^{\Delta}_n)\right)$. Note that $\hat{Y}^{\Delta}_n$ and $\hat{Z}^{\Delta}_n$ in (2.19) are calculated by taking $\mathbb{E}_n[\cdot]$ in (1.6) and (2.11), where $\mathbb{E}_n\left[\hat{Z}^{\Delta}_n \Delta W_n\right] = 0$ and $\mathbb{E}_n\left[\hat{\Gamma}^{\Delta}_n b(t_n, X^{\Delta}_n)\Delta W_n\right] = 0$. Moreover, $\hat{\Gamma}^{\Delta}_n$ in (2.19) is calculated by multiplying both sides of (2.11) with ΔW_n, where $\mathbb{E}_n\left[\Delta W_n f_D\left(t_n, \hat{\mathbf{X}}^{\Delta}_n, \mathbf{D}_n\hat{\mathbf{X}}^{\Delta}_n\right)\right] = 0$. Finally, applying the Itô isometry gives $\hat{\Gamma}^{\Delta}_n$ in (2.19).

By the Markov property of the underlying processes, there exist some deterministic functions $\hat{y}_n$, $\hat{z}_n$ and $\hat{\gamma}_n$ s.t.

$$\hat{Y}^{\Delta}_n = \hat{y}_n\left(X^{\Delta}_n\right), \quad \hat{Z}^{\Delta}_n = \hat{z}_n\left(X^{\Delta}_n\right), \quad \hat{\Gamma}^{\Delta}_n = \hat{\gamma}_n\left(X^{\Delta}_n\right), \quad n = 0, \ldots, N-1. \tag{2.20}$$

Moreover, by the martingale representation theorem, there exists an $\mathbb{R}^{d\times d}-$valued square integrable process $D_n\hat{Z}_t$ s.t.

$$D_n Y^{\Delta,\hat{\theta}}_{n+1} = \hat{Z}^{\Delta}_n - f_D\left(t_n, \hat{\mathbf{X}}^{\Delta}_n, \mathbf{D}_n\hat{\mathbf{X}}^{\Delta}_n\right)\Delta t_n + \int_{t_n}^{t_{n+1}}\left(\left(D_n\hat{Z}_s\right)^{\top} dW_s\right)^{\top}, \tag{2.21}$$

and by Itô isometry, we have

$$D_n\hat{Z}^{\Delta}_n = \hat{\Gamma}^{\Delta}_n b(t_n, X^{\Delta}_n) = \frac{1}{\Delta t_n}\mathbb{E}_n\left[\int_{t_n}^{t_{n+1}} D_n\hat{Z}_s ds\right], \quad n = 0, \ldots, N-1.$$

Hence, $D\hat{Z}^{\Delta}$ is an $\mathbb{L}^2-$projection of $D\hat{Z}$. Moreover, $\hat{Z}^{\Delta}$ is an $\mathbb{L}^2-$projection of $\hat{Z}$ s.t.

$$Y_{n+1}^{\Delta,\hat{\theta}} = \hat{Y}_n^{\Delta} - f\left(t_n, \hat{\mathbf{X}}_n^{\Delta}\right)\Delta t_n + \int_{t_n}^{t_{n+1}} \hat{Z}_s dW_s. \tag{2.22}$$

Finally, we define the approximation errors of $\hat{y}_n$, $\hat{z}_n$ and $\hat{\gamma}_n$ by the DNNs ϕ_n^y, ϕ_n^z and ϕ_n^{γ} defined as

$$\begin{aligned}
\varepsilon_n^y &:= \inf_{\theta_n^y \in \Theta_n^y} \mathbb{E}\left[\left|\hat{y}_n\left(X_n^{\Delta}\right) - \phi_n^y\left(X_n^{\Delta};\theta_n^y\right)\right|^2\right], \\
\varepsilon_n^z &:= \inf_{\theta_n^z \in \Theta_n^z} \mathbb{E}\left[\left|\hat{z}_n\left(X_n^{\Delta}\right) - \phi_n^z\left(X_n^{\Delta};\theta_n^z\right)\right|^2\right], \\
\varepsilon_n^{\gamma} &:= \inf_{\theta_n^{\gamma} \in \Theta_n^{\gamma}} \mathbb{E}\left[\left|\left(\hat{\gamma}_n\left(X_n^{\Delta}\right) - \phi_n^{\gamma}\left(X_n^{\Delta};\theta_n^{\gamma}\right)\right) b(t_n, X_n^{\Delta})\right|^2\right],
\end{aligned} \tag{2.23}$$

for $n = 0, \ldots, N-1$. The goal is now to find an upper bound of the total approximation error of the DLBDP scheme defined as

$$\begin{aligned}
\mathcal{E}\left[(Y,Z,\Gamma), \left(Y^{\Delta,\hat{\theta}}, Z^{\Delta,\hat{\theta}}, \Gamma^{\Delta,\hat{\theta}}\right)\right] := &\max_{0\le n\le N} \mathbb{E}\left[\left|Y_n - Y_n^{\Delta,\hat{\theta}}\right|^2\right] + \max_{0\le n\le N} \mathbb{E}\left[\left|Z_n - Z_n^{\Delta,\hat{\theta}}\right|^2\right] \\
&+ \mathbb{E}\left[\sum_{n=0}^{N-1} \int_{t_n}^{t_{n+1}} \left|D_n Z_s - D_n Z_n^{\Delta,\hat{\theta}}\right|^2 ds\right],
\end{aligned}$$

in terms of the discretization error (from the Euler-Maruyama scheme) and the approximation errors (2.23) by the DNNs, where $D_n Z_s - D_n Z_n^{\Delta,\hat{\theta}} = \Gamma_s b(s) - \Gamma_n^{\Delta,\hat{\theta}} b(t_n)$ due to relations (2.10) and Assumption AX4.

Theorem 2.3.1. *(Consistency of DLBDP scheme). Under Assumptions AX4 and AY4, there exist a constant $C > 0$ independent of the time partition Δ, s.t. the total approximation error of the DLBDP scheme admits to*

$$\begin{aligned}
\mathcal{E}\left[(Y,Z,\Gamma), \left(Y^{\Delta,\hat{\theta}}, Z^{\Delta,\hat{\theta}}, \Gamma^{\Delta,\hat{\theta}}\right)\right] \le C \Bigg\{ &|\Delta| + \varepsilon^{DZ}(|\Delta|) + N\sum_{n=0}^{N-1}\left(\lambda_1\varepsilon_n^y + \lambda_2\varepsilon_n^z\right) \\
&+ \sum_{n=0}^{N-1}\left(\lambda_2\varepsilon_n^y + \lambda_1\varepsilon_n^z + \lambda_2\varepsilon_n^{\gamma}\right)\Bigg\}.
\end{aligned}$$

Proof. In the following, C denotes a positive generic constant independent of Δ, which may take different values from line to line.

Step 1. Let us fix $n \in \{0, 1, \ldots, N-1\}$. For the time interval $[t_n, t_{n+1}]$ in (2.4), after taking $\mathbb{E}_n[\cdot]$ and using the relation for $\hat{Y}_n^{\Delta}$ in (2.19), we get

$$Y_n - \hat{Y}_n^{\Delta} = \mathbb{E}_n\left[Y_{n+1} - Y_{n+1}^{\Delta,\hat{\theta}}\right] + \mathbb{E}_n\left[\int_{t_n}^{t_{n+1}} f(s, \mathbf{X}_s) - f\left(t_n, \hat{\mathbf{X}}_n^{\Delta}\right) ds\right].$$

Using the Jensen inequality for the second term above and then the $L^2([0,T];\mathbb{R})$ Cauchy-Schwarz inequality, we have

$$\left|Y_n - \hat{Y}_n^{\Delta}\right| \le \left|\mathbb{E}_n\left[Y_{n+1} - Y_{n+1}^{\Delta,\hat{\theta}}\right]\right| + (\Delta t_n)^{\frac{1}{2}}\left(\mathbb{E}_n\left[\int_{t_n}^{t_{n+1}} \left|f(s,\mathbf{X}_s) - f\left(t_n, \hat{\mathbf{X}}_n^{\Delta}\right)\right|^2 ds\right]\right)^{\frac{1}{2}}.$$

Using again the Jensen inequality for the first term above and the Young inequality of the form

$$(c_1+c_2)^2 \leq (1+\nu\Delta t_n)\, c_1^2 + \left(1+\frac{1}{\nu\Delta t_n}\right) c_2^2, \quad \nu>0, \tag{2.24}$$

we get (after taking $\mathbb{E}[\cdot]$) that

$$\begin{aligned}\mathbb{E}\left[\left|Y_n-\hat{Y}_n^{\Delta}\right|^2\right] &\leq (1+\nu\Delta t_n)\,\mathbb{E}\left[\left|Y_{n+1}-Y_{n+1}^{\Delta,\hat{\theta}}\right|^2\right] \\ &\quad+\frac{1}{\nu}(1+\nu\Delta t_n)\,\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|f(s,\mathbf{X}_s)-f\left(t_n,\hat{\mathbf{X}}_n^{\Delta}\right)\right|^2 ds\right].\end{aligned}$$

The Lipschitz and Hölder continuity of f in (2.13) and the inequality $\left(\sum_{\mathrm{i}=1}^4 c_{\mathrm{i}}\right)^2 \leq 4\sum_{\mathrm{i}=1}^4 c_{\mathrm{i}}^2$ yields

$$\begin{aligned}&\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|f(s,\mathbf{X}_s)-f\left(t_n,\hat{\mathbf{X}}_n^{\Delta}\right)\right|^2 ds\right] \\ &\leq 4L_f^2\left(\int_{t_n}^{t_{n+1}}|s-t_n|\,ds+\int_{t_n}^{t_{n+1}}\mathbb{E}\left[\left|X_s-X_n^{\Delta}\right|^2\right]ds+\int_{t_n}^{t_{n+1}}\mathbb{E}\left[\left|Y_s-\hat{Y}_n^{\Delta}\right|^2\right]ds\right. \\ &\quad\left.+\int_{t_n}^{t_{n+1}}\mathbb{E}\left[\left|Z_s-\hat{Z}_n^{\Delta}\right|^2\right]ds\right)\end{aligned} \tag{2.25}$$

Due to the mean squared continuities (2.14) and the inequality $(c_1+c_2)^2 \leq 2\left(c_1^2+c_2^2\right)$, we have

$$\begin{aligned}\int_{t_n}^{t_{n+1}}\mathbb{E}\left[\left|X_s-X_n^{\Delta}\right|^2\right]ds &= \int_{t_n}^{t_{n+1}}\mathbb{E}\left[\left|(X_s-X_n)+\left(X_n-X_n^{\Delta}\right)\right|^2\right]ds, \\ &\leq \int_{t_n}^{t_{n+1}}\mathbb{E}\left[\left(|X_s-X_n|+\left|X_n-X_n^{\Delta}\right|\right)^2\right]ds, \\ &\leq C|\Delta|^2+2\Delta t_n\mathbb{E}\left[\left|X_n-X_n^{\Delta}\right|^2\right].\end{aligned}$$

Performing similar calculations for other terms in (2.25), we gather

$$\begin{aligned}\mathbb{E}\left[\left|Y_n-\hat{Y}_n^{\Delta}\right|^2\right] &\leq (1+\nu\Delta t_n)\,\mathbb{E}\left[\left|Y_{n+1}-Y_{n+1}^{\Delta,\hat{\theta}}\right|^2\right] \\ &\quad+\frac{4L_f^2}{\nu}(1+\nu\Delta t_n)\left\{C|\Delta|^2+2\Delta t_n\left(\mathbb{E}\left[\left|Y_n-\hat{Y}_n^{\Delta}\right|^2\right]+\mathbb{E}\left[\left|Z_n-\hat{Z}_n^{\Delta}\right|^2\right]\right)\right\},\end{aligned} \tag{2.26}$$

where we also used (2.16).

Step 2. By taking $\mathbb{E}_n[\cdot]$ in (2.8) and using the relation for $\hat{Z}_n^{\Delta}$ in (2.19), we have

$$Z_n-\hat{Z}_n^{\Delta}=\mathbb{E}_n\left[D_nY_{n+1}-D_nY_{n+1}^{\Delta,\hat{\theta}}\right]+\mathbb{E}_n\left[\int_{t_n}^{t_{n+1}} f_D(s,\mathbf{X}_s,\mathbf{D}_n\mathbf{X}_s)-f_D\left(t_n,\hat{\mathbf{X}}_n^{\Delta},\mathbf{D}_n\hat{\mathbf{X}}_n^{\Delta}\right)ds\right],$$

where $Z_n=D_nY_n$ due to Theorem 2.2.2. Similarly as in the previous step, namely applying Jensen inequality for the second term above, using the $L^2\left([0,T];\mathbb{R}^d\right)$ Cauchy-Schwarz inequality and the Young inequality (2.24), we have

$$\begin{aligned}\mathbb{E}\left[\left|Z_n-\hat{Z}_n^{\Delta}\right|^2\right] &\leq (1+\nu\Delta t_n)\,\mathbb{E}\left[\left|\mathbb{E}_n\left[D_nY_{n+1}-D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right] \\ &\quad+\frac{1}{\nu}(1+\nu\Delta t_n)\,\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|f_D(s,\mathbf{X}_s,\mathbf{D}_n\mathbf{X}_s)-f_D\left(t_n,\hat{\mathbf{X}}_n^{\Delta},\mathbf{D}_n\hat{\mathbf{X}}_n^{\Delta}\right)\right|^2 ds\right].\end{aligned}$$

From the Lipschitz and Hölder continuity of f_D in (2.13) and the inequality $\left(\sum_{\mathrm{i}=1}^{7} c_{\mathrm{i}}\right)^2 \leq 8\sum_{\mathrm{i}=1}^{7} c_{\mathrm{i}}^2$, we get

$$\begin{aligned}
&\mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left|f_D\left(s, \mathbf{X}_s, \mathbf{D}_n\mathbf{X}_s\right) - f_D\left(t_n, \hat{\mathbf{X}}_n^{\Delta}, \mathbf{D}_n\hat{\mathbf{X}}_n^{\Delta}\right)\right|^2 ds\right] \\
&\leq 8L_{f_D}^2\left(\int_{t_n}^{t_{n+1}} |s-t_n|\, ds + \int_{t_n}^{t_{n+1}} \mathbb{E}\left[\left|X_s - X_n^{\Delta}\right|^2\right] ds + \int_{t_n}^{t_{n+1}} \mathbb{E}\left[\left|Y_s - \hat{Y}_n^{\Delta}\right|^2\right] ds\right. \\
&\quad + \int_{t_n}^{t_{n+1}} \mathbb{E}\left[\left|Z_s - \hat{Z}_n^{\Delta}\right|^2\right] ds + \int_{t_n}^{t_{n+1}} \mathbb{E}\left[\left|D_nX_s - D_nX_n^{\Delta}\right|^2\right] ds \\
&\quad \left. + \int_{t_n}^{t_{n+1}} \mathbb{E}\left[\left|D_nY_s - D_n\hat{Y}_n^{\Delta}\right|^2\right] ds + \int_{t_n}^{t_{n+1}} \mathbb{E}\left[\left|D_nZ_s - D_n\hat{Z}_n^{\Delta}\right|^2\right] ds\right).
\end{aligned}$$

The mean squared continuities (2.14) and the inequality $(c_1+c_2)^2 \leq 2\left(c_1^2+c_2^2\right)$ yields

$$\begin{aligned}
\mathbb{E}\left[\left|Z_n - \hat{Z}_n^{\Delta}\right|^2\right] &\leq (1+\nu\Delta t_n)\,\mathbb{E}\left[\left|\mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right] \\
&\quad + \frac{8L_{f_D}^2}{\nu}(1+\nu\Delta t_n)\left\{C|\Delta|^2 + 2\Delta t_n\left(\mathbb{E}\left[\left|X_n - X_n^{\Delta}\right|^2\right] + \mathbb{E}\left[\left|Y_n - \hat{Y}_n^{\Delta}\right|^2\right]\right.\right. \\
&\quad \left. + \mathbb{E}\left[\left|Z_n - \hat{Z}_n^{\Delta}\right|^2\right]\right) + 2\Delta t_n\left(\mathbb{E}\left[\left|D_nX_n - D_nX_n^{\Delta}\right|^2\right] + \mathbb{E}\left[\left|D_nY_n - D_n\hat{Y}_n^{\Delta}\right|^2\right]\right) \\
&\quad \left. + \mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left|D_nZ_s - D_n\hat{Z}_n^{\Delta}\right|^2 ds\right]\right\}.
\end{aligned}$$

By using (2.16), $D_nY_n - D_n\hat{Y}_n^{\Delta} = Z_n - \hat{Z}_n^{\Delta}$ and $\mathbb{E}\left[\left|D_nX_n - D_nX_n^{\Delta}\right|^2\right] = 0$ (due to Assumption AX4), we have

$$\begin{aligned}
\mathbb{E}\left[\left|Z_n - \hat{Z}_n^{\Delta}\right|^2\right] &\leq (1+\nu\Delta t_n)\,\mathbb{E}\left[\left|\mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right] \\
&\quad + \frac{8L_{f_D}^2}{\nu}(1+\nu\Delta t_n)\left\{C|\Delta|^2 + 2\Delta t_n\left(\mathbb{E}\left[\left|Y_n - \hat{Y}_n^{\Delta}\right|^2\right] + 2\mathbb{E}\left[\left|Z_n - \hat{Z}_n^{\Delta}\right|^2\right]\right)\right. \\
&\quad \left. + \mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left|D_nZ_s - D_n\hat{Z}_n^{\Delta}\right|^2 ds\right]\right\}.
\end{aligned}$$

By the definition of $\widehat{DZ}_n$ in (2.18), the last term above is given as

$$\mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left|D_nZ_s - D_n\hat{Z}_n^{\Delta}\right|^2 ds\right] = \mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left|D_nZ_s - \widehat{DZ}_n\right|^2 ds\right] + \Delta t_n\mathbb{E}\left[\left|\widehat{DZ}_n - D_n\hat{Z}_n^{\Delta}\right|^2\right]. \tag{2.27}$$

Hence, we get

$$\begin{aligned}
\mathbb{E}\left[\left|Z_n - \hat{Z}_n^{\Delta}\right|^2\right] &\leq (1+\nu\Delta t_n)\,\mathbb{E}\left[\left|\mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right] \\
&\quad + \frac{8L_{f_D}^2}{\nu}(1+\nu\Delta t_n)\left\{C|\Delta|^2 + 2\Delta t_n\left(\mathbb{E}\left[\left|Y_n - \hat{Y}_n^{\Delta}\right|^2\right] + 2\mathbb{E}\left[\left|Z_n - \hat{Z}_n^{\Delta}\right|^2\right]\right)\right. \\
&\quad \left. + \mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left|D_nZ_s - \widehat{DZ}_n\right|^2 ds\right] + \Delta t_n\mathbb{E}\left[\left|\widehat{DZ}_n - D_n\hat{Z}_n^{\Delta}\right|^2\right]\right\}.
\end{aligned} \tag{2.28}$$

Next, we find an upper bound for $\Delta t_n \mathbb{E}\left[\left|\widehat{DZ}_n - D_n\hat{Z}_n^\Delta\right|^2\right]$ in (2.28). By multiplying both sides of (2.8) with ΔW_n, taking $\mathbb{E}_n[\cdot]$, using the Itô's isometry and (2.18), we have together with (2.19)

$$\begin{aligned}\Delta t_n\left(\widehat{DZ}_n - D_n\hat{Z}_n^\Delta\right) = \mathbb{E}_n\left[\Delta W_n\left(D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}} - \mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right)\right] \\ + \mathbb{E}_n\left[\Delta W_n\int_{t_n}^{t_{n+1}} f_D\left(s,\mathbf{X}_s,\mathbf{D}_n\mathbf{X}_s\right)ds\right],\end{aligned}$$

after rewriting the relation for $\hat{\Gamma}_n^\Delta$ in (2.19) as $\Delta t_n D_n\hat{Z}_n^\Delta = \mathbb{E}_n\left[\Delta W_n D_nY_{n+1}^{\Delta,\hat{\theta}}\right]$ and that $\mathbb{E}_n\left[\Delta W_n\mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right] = 0$. The conditional $\mathbb{L}^2\left(\Omega;\mathbb{R}^d\right)$ Cauchy-Schwarz inequality in the Frobenius norm and the independence of Brownian motion increments implies

$$\begin{aligned}\Delta t_n\left|\widehat{DZ}_n - D_n\hat{Z}_n^\Delta\right| \le (d\Delta t_n)^{\frac{1}{2}}\left(\mathbb{E}_n\left[\left|D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}} - \mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right]\right)^{\frac{1}{2}} \\ + (d\Delta t_n)^{\frac{1}{2}}\left(\mathbb{E}_n\left[\left|\int_{t_n}^{t_{n+1}} f_D\left(s,\mathbf{X}_s,\mathbf{D}_n\mathbf{X}_s\right)ds\right|^2\right]\right)^{\frac{1}{2}}.\end{aligned}$$

By applying the $L^2\left([0,T];\mathbb{R}^d\right)$ Cauchy-Schwarz inequality for the last term above, using the inequality $(c_1+c_2)^2 \le 2\left(c_1^2+c_2^2\right)$ and the law of total expectation yields

$$\begin{aligned}\Delta t_n\mathbb{E}\left[\left|\widehat{DZ}_n - D_n\hat{Z}_n^\Delta\right|^2\right] \le 2d\left(\mathbb{E}\left[\left|D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right|^2\right] - \mathbb{E}\left[\left|\mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right]\right) \\ + 2d\Delta t_n\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|f_D\left(s,\mathbf{X}_s,\mathbf{D}_n\mathbf{X}_s\right)\right|^2 ds\right].\end{aligned} \tag{2.29}$$

Using the upper bound (2.29) in (2.28) and choosing $\nu \equiv \check{\nu} = 16dL_{f_D}^2$, this implies

$$\begin{aligned}\mathbb{E}\left[\left|Z_n - \hat{Z}_n^\Delta\right|^2\right] \le (1+\check{\nu}\Delta t_n)\,\mathbb{E}\left[\left|D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right|^2\right] \\ + \frac{1}{2d}(1+\check{\nu}\Delta t_n)\left\{C|\Delta|^2 + 2\Delta t_n\left(\mathbb{E}\left[\left|Y_n - \hat{Y}_n^\Delta\right|^2\right] + 2\mathbb{E}\left[\left|Z_n - \hat{Z}_n^\Delta\right|^2\right]\right)\right. \\ \left. + \mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|D_nZ_s - \widehat{DZ}_n\right|^2 ds\right]\right\} \\ + (1+\check{\nu}\Delta t_n)\,\Delta t_n\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|f_D\left(s,\mathbf{X}_s,\mathbf{D}_n\mathbf{X}_s\right)\right|^2 ds\right].\end{aligned} \tag{2.30}$$

Step 3. Combining (2.26) and (2.30) gives

$$
\begin{aligned}
&\mathbb{E}\left[\left|Y_n-\hat{Y}_n^{\Delta}\right|^2\right]+\mathbb{E}\left[\left|Z_n-\hat{Z}_n^{\Delta}\right|^2\right]\\
&\le(1+\nu\Delta t_n)\,\mathbb{E}\left[\left|Y_{n+1}-Y_{n+1}^{\Delta,\hat{\theta}}\right|^2\right]+(1+\check{\nu}\Delta t_n)\,\mathbb{E}\left[\left|D_nY_{n+1}-D_nY_{n+1}^{\Delta,\hat{\theta}}\right|^2\right]\\
&\quad+\frac{4L_f^2}{\nu}\,(1+\nu\Delta t_n)\left\{C|\Delta|^2+\Delta t_n\left(\mathbb{E}\left[\left|Y_n-\hat{Y}_n^{\Delta}\right|^2\right]+\mathbb{E}\left[\left|Z_n-\hat{Z}_n^{\Delta}\right|^2\right]\right)\right\}\\
&\quad+\frac{1}{2d}\,(1+\check{\nu}\Delta t_n)\left\{C|\Delta|^2+4\Delta t_n\left(\mathbb{E}\left[\left|Y_n-\hat{Y}_n^{\Delta}\right|^2\right]+\mathbb{E}\left[\left|Z_n-\hat{Z}_n^{\Delta}\right|^2\right]\right)\right.\\
&\quad\left.+\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|D_nZ_s-\widehat{DZ}_n\right|^2ds\right]\right\}\\
&\quad+(1+\check{\nu}\Delta t_n)\,\Delta t_n\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|f_D\left(s,\mathbf{X}_s,\mathbf{D}_n\mathbf{X}_s\right)\right|^2ds\right].
\end{aligned}
$$

Moreover,

$$
\begin{aligned}
&\left(1-\left(\frac{4L_f^2}{\nu}\,(1+\nu\Delta t_n)+\frac{2}{d}\,(1+\check{\nu}\Delta t_n)\right)\Delta t_n\right)\left(\mathbb{E}\left[\left|Y_n-\hat{Y}_n^{\Delta}\right|^2\right]+\mathbb{E}\left[\left|Z_n-\hat{Z}_n^{\Delta}\right|^2\right]\right)\\
&\le(1+\nu\Delta t_n)\,\mathbb{E}\left[\left|Y_{n+1}-Y_{n+1}^{\Delta,\hat{\theta}}\right|^2\right]+(1+\check{\nu}\Delta t_n)\,\mathbb{E}\left[\left|D_nY_{n+1}-D_nY_{n+1}^{\Delta,\hat{\theta}}\right|^2\right]\\
&\quad+\frac{4L_f^2}{\nu}\,(1+\nu\Delta t_n)\,C|\Delta|^2+\frac{1}{2d}\,(1+\check{\nu}\Delta t_n)\left\{C|\Delta|^2+\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|D_nZ_s-\widehat{DZ}_n\right|^2ds\right]\right\}\\
&\quad+(1+\check{\nu}\Delta t_n)\,\Delta t_n\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|f_D\left(s,\mathbf{X}_s,\mathbf{D}_n\mathbf{X}_s\right)\right|^2ds\right].
\end{aligned}
$$

Then, for any given $\nu>0$ and $|\Delta|$ sufficiently small:

$$
\begin{aligned}
&\mathbb{E}\left[\left|Y_n-\hat{Y}_n^{\Delta}\right|^2\right]+\mathbb{E}\left[\left|Z_n-\hat{Z}_n^{\Delta}\right|^2\right]\\
&\le(1+C|\Delta|)\,\mathbb{E}\left[\left|Y_{n+1}-Y_{n+1}^{\Delta,\hat{\theta}}\right|^2\right]+(1+C|\Delta|)\,\mathbb{E}\left[\left|D_nY_{n+1}-D_nY_{n+1}^{\Delta,\hat{\theta}}\right|^2\right]+C|\Delta|^2\\
&\quad+C\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|D_nZ_s-\widehat{DZ}_n\right|^2ds\right]+C|\Delta|\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|f_D\left(s,\mathbf{X}_s,\mathbf{D}_n\mathbf{X}_s\right)\right|^2ds\right].
\end{aligned}
\tag{2.31}
$$

Using the Young inequality in the form

$$
(c_1+c_2)^2\ge(1-\nu)\,c_1^2+\left(1-\frac{1}{\nu}\right)c_2^2\ge(1-\nu)\,c_1^2-\frac{1}{\nu}c_2^2,\quad\nu>0,
\tag{2.32}
$$

we have for $\nu=|\Delta|$ that

$$
\begin{aligned}
\mathbb{E}\left[\left|Y_n-\hat{Y}_n^{\Delta}\right|^2\right]&=\mathbb{E}\left[\left|Y_n-Y_n^{\Delta,\hat{\theta}}+Y_n^{\Delta,\hat{\theta}}-\hat{Y}_n^{\Delta}\right|^2\right]\\
&\ge(1-|\Delta|)\,\mathbb{E}\left[\left|Y_n-Y_n^{\Delta,\hat{\theta}}\right|^2\right]-\frac{1}{|\Delta|}\mathbb{E}\left[\left|Y_n^{\Delta,\hat{\theta}}-\hat{Y}_n^{\Delta}\right|^2\right],\\
\mathbb{E}\left[\left|Z_n-\hat{Z}_n^{\Delta}\right|^2\right]&=\mathbb{E}\left[\left|Z_n-Z_n^{\Delta,\hat{\theta}}+Z_n^{\Delta,\hat{\theta}}-\hat{Z}_n^{\Delta}\right|^2\right]\\
&\ge(1-|\Delta|)\,\mathbb{E}\left[\left|Z_n-Z_n^{\Delta,\hat{\theta}}\right|^2\right]-\frac{1}{|\Delta|}\mathbb{E}\left[\left|Z_n^{\Delta,\hat{\theta}}-\hat{Z}_n^{\Delta}\right|^2\right].
\end{aligned}
$$

Then, for small enough $|\Delta|$, (2.31) becomes

$$\begin{aligned}
&\mathbb{E}\left[\left|Y_n - Y_n^{\Delta,\hat{\theta}}\right|^2\right] + \mathbb{E}\left[\left|Z_n - Z_n^{\Delta,\hat{\theta}}\right|^2\right] \\
&\leq (1+C|\Delta|)\,\mathbb{E}\left[\left|Y_{n+1} - Y_{n+1}^{\Delta,\hat{\theta}}\right|^2\right] + (1+C|\Delta|)\,\mathbb{E}\left[\left|D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right|^2\right] + C|\Delta|^2 \\
&\quad + C\mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left|D_nZ_s - \widehat{DZ}_n\right|^2 ds\right] + C|\Delta|\mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left|f_D\left(s, \mathbf{X}_s, \mathbf{D}_n\mathbf{X}_s\right)\right|^2 ds\right] \\
&\quad + CN\mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - Y_n^{\Delta,\hat{\theta}}\right|^2\right] + CN\mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - Z_n^{\Delta,\hat{\theta}}\right|^2\right].
\end{aligned}$$

Using the discrete Grönwall lemma in the above equation, we get

$$\begin{aligned}
&\max_{0\leq n\leq N}\mathbb{E}\left[\left|Y_n - Y_n^{\Delta,\hat{\theta}}\right|^2\right] + \max_{0\leq n\leq N}\mathbb{E}\left[\left|Z_n - Z_n^{\Delta,\hat{\theta}}\right|^2\right] \\
&\leq C\mathbb{E}\left[\left|g(X_T) - g\left(X_N^{\Delta}\right)\right|^2\right] + C\mathbb{E}\left[\left|\nabla_x g(X_T)D_{N-1}X_N - \nabla_x g\left(X_N^{\Delta}\right)D_{N-1}X_N^{\Delta}\right|^2\right] \\
&\quad + C|\Delta| + C\sum_{n=0}^{N-1}\mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left|D_nZ_s - \widehat{DZ}_n\right|^2 ds\right] \\
&\quad + C|\Delta|\sum_{n=0}^{N-1}\mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left|f_D\left(s, \mathbf{X}_s, \mathbf{D}_n\mathbf{X}_s\right)\right|^2 ds\right] + CN\sum_{n=0}^{N-1}\mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - Y_n^{\Delta,\hat{\theta}}\right|^2\right] \\
&\quad + CN\sum_{n=0}^{N-1}\mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - Z_n^{\Delta,\hat{\theta}}\right|^2\right].
\end{aligned}$$

The second term in the above inequality can be written as

$$\begin{aligned}
&\mathbb{E}\left[\left|\nabla_x g(X_T)D_{N-1}X_N - \nabla_x g\left(X_N^{\Delta}\right)D_{N-1}X_N^{\Delta}\right|^2\right] \\
&= \mathbb{E}\left[\left|\left(\nabla_x g(X_T) - \nabla_x g\left(X_N^{\Delta}\right)\right)D_{N-1}X_N + \nabla_x g\left(X_N^{\Delta}\right)\left(D_{N-1}X_N - D_{N-1}X_N^{\Delta}\right)\right|^2\right], \\
&\leq 2\mathbb{E}\left[\left|\nabla_x g(X_T) - \nabla_x g\left(X_N^{\Delta}\right)\right|^2 \left|D_{N-1}X_N\right|^2\right] + 2\mathbb{E}\left[\left|\nabla_x g\left(X_N^{\Delta}\right)\right|^2 \left|D_{N-1}X_N - D_{N-1}X_N^{\Delta}\right|^2\right], \\
&\leq C\mathbb{E}\left[\left|\nabla_x g(X_T) - \nabla_x g\left(X_N^{\Delta}\right)\right|^2\right] + C\mathbb{E}\left[\left|D_{N-1}X_N - D_{N-1}X_N^{\Delta}\right|^2\right]
\end{aligned}$$

where we used the submultiplicative property of the Frobenius norm and $(c_1+c_2)^2 \leq 2\left(c_1^2+c_2^2\right)$ for the first inequality, the boundedness of DX and $\nabla_x g(X)$ for the second inequality. Therefore, we get from (2.17) that

$$\mathbb{E}\left[\left|\nabla_x g(X_T)D_{N-1}X_N - \nabla_x g\left(X_N^{\Delta}\right)D_{N-1}X_N^{\Delta}\right|^2\right] \leq C\mathbb{E}\left[\left|\nabla_x g(X_T) - \nabla_x g\left(X_N^{\Delta}\right)\right|^2\right] + C|\Delta|. \tag{2.33}$$

From (2.15), the $\mathbb{L}^2$-regularity of DZ (2.18) and the inequality (2.33), we have

$$\begin{aligned}
&\max_{0\leq n\leq N}\mathbb{E}\left[\left|Y_n - Y_n^{\Delta,\hat{\theta}}\right|^2\right] + \max_{0\leq n\leq N}\mathbb{E}\left[\left|Z_n - Z_n^{\Delta,\hat{\theta}}\right|^2\right] \\
&\leq C\mathbb{E}\left[\left|g(X_T) - g\left(X_N^{\Delta}\right)\right|^2\right] + C\mathbb{E}\left[\left|\nabla_x g(X_T) - \nabla_x g\left(X_N^{\Delta}\right)\right|^2\right] + C|\Delta| + C\varepsilon^{DZ}\left(|\Delta|\right) \\
&\quad + CN\sum_{n=0}^{N-1}\mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - Y_n^{\Delta,\hat{\theta}}\right|^2\right] + CN\sum_{n=0}^{N-1}\mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - Z_n^{\Delta,\hat{\theta}}\right|^2\right].
\end{aligned} \tag{2.34}$$

Step 4. Let us fix $n \in \{0, 1, \ldots, N-1\}$. By using relations (2.21) and (2.22), and recalling the definition of $\left(\hat{Z}_n^\Delta, D_n \hat{Z}_n^\Delta\right)$ as an $\mathbb{L}^2-$projection of $\left(\hat{Z}_t, D_n \hat{Z}_t\right)$, we can rewrite the loss function (2.12) as

$$\begin{aligned} \mathbf{L}_n^\Delta(\theta_n) = \lambda_1 \left(\hat{\mathbf{L}}_n^{y,\Delta}(\theta_n) + \mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left| \hat{Z}_s - \hat{Z}_n^\Delta \right|^2 ds \right] \right) \\ + \lambda_2 \left(\hat{\mathbf{L}}_n^{z,\Delta}(\theta_n) + \mathbb{E}\left[\int_{t_n}^{t_{n+1}} \left| D_n \hat{Z}_s - D_n \hat{Z}_n^\Delta \right|^2 ds \right] \right), \end{aligned} \tag{2.35}$$

where

$$\begin{aligned} \hat{\mathbf{L}}_n^{y,\Delta}(\theta_n) := \mathbb{E}\left[\left| \hat{Y}_n^\Delta - \phi_n^y\left(X_n^\Delta; \theta_n^y\right) + \left(f\left(t_n, \mathbf{X}_n^{\Delta,\theta}\right) - f\left(t_n, \hat{\mathbf{X}}_n^\Delta\right) \right) \Delta t_n \right|^2 \right] \\ + \Delta t_n \mathbb{E}\left[\left| \hat{Z}_n^\Delta - \phi_n^z\left(X_n^\Delta; \theta_n^z\right) \right|^2 \right], \end{aligned} \tag{2.36}$$

and

$$\begin{aligned} \hat{\mathbf{L}}_n^{z,\Delta}(\theta_n) := \mathbb{E}\left[\left| \hat{Z}_n^\Delta - \phi_n^z\left(X_n^\Delta; \theta_n^z\right) + \left(f_D\left(t_n, \mathbf{X}_n^{\Delta,\theta}, \mathbf{D}_n \mathbf{X}_n^{\Delta,\theta}\right) \right.\right.\right. \\ \left.\left.\left. - f_D\left(t_n, \hat{\mathbf{X}}_n^\Delta, \mathbf{D}_n \hat{\mathbf{X}}_n^\Delta\right) \right) \Delta t_n \right|^2 \right] \\ + \Delta t_n \mathbb{E}\left[\left| \left(\hat{\Gamma}_n^\Delta - \phi_n^\gamma\left(X_n^\Delta; \theta_n^\gamma\right) \right) b\left(t_n, X_n^\Delta\right) \right|^2 \right]. \end{aligned} \tag{2.37}$$

By using the Young inequality (2.24) in (2.36) we have

$$\begin{aligned} \hat{\mathbf{L}}_n^{y,\Delta}(\theta_n) \leq (1 + \nu \Delta t_n) \mathbb{E}\left[\left| \hat{Y}_n^\Delta - \phi_n^y\left(X_n^\Delta; \theta_n^y\right) \right|^2 \right] + \left(1 + \frac{1}{\nu \Delta t_n}\right) \Delta t_n^2 \\ \mathbb{E}\left[\left| f\left(t_n, \mathbf{X}_n^{\Delta,\theta}\right) - f\left(t_n, \hat{\mathbf{X}}_n^\Delta\right) \right|^2 \right] + \Delta t_n \mathbb{E}\left[\left| \hat{Z}_n^\Delta - \phi_n^z\left(X_n^\Delta; \theta_n^z\right) \right|^2 \right]. \end{aligned}$$

From the Lipschitz and Hölder continuity of f (2.13) and the inequality $(c_1 + c_2)^2 \leq 2\left(c_1^2 + c_2^2\right)$, we get

$$\begin{aligned} \mathbb{E}\left[\left| f\left(t_n, \mathbf{X}_n^{\Delta,\theta}\right) - f\left(t_n, \hat{\mathbf{X}}_n^\Delta\right) \right|^2 \right] \leq 2 L_f^2 \left(\mathbb{E}\left[\left| \hat{Y}_n^\Delta - \phi_n^y\left(X_n^\Delta; \theta_n^y\right) \right|^2 \right] \right. \\ \left. + \mathbb{E}\left[\left| \hat{Z}_n^\Delta - \phi_n^z\left(X_n^\Delta; \theta_n^z\right) \right|^2 \right] \right). \end{aligned}$$

Hence, (2.36) is bounded by

$$\hat{\mathbf{L}}_n^{y,\Delta}(\theta_n) \leq (1 + C\Delta t_n) \mathbb{E}\left[\left| \hat{Y}_n^\Delta - \phi_n^y\left(X_n^\Delta; \theta_n^y\right) \right|^2 \right] + C \Delta t_n \mathbb{E}\left[\left| \hat{Z}_n^\Delta - \phi_n^z\left(X_n^\Delta; \theta_n^z\right) \right|^2 \right]. \tag{2.38}$$

By performing similar steps for (2.37) (using Lipschitz and Hölder continuity of f_D instead of f (2.13)), we get

$$\begin{aligned} \hat{\mathbf{L}}_n^{z,\Delta}(\theta_n) \leq C \Delta t_n \mathbb{E}\left[\left| \hat{Y}_n^\Delta - \phi_n^y\left(X_n^\Delta; \theta_n^y\right) \right|^2 \right] + (1 + C\Delta t_n) \mathbb{E}\left[\left| \hat{Z}_n^\Delta - \phi_n^z\left(X_n^\Delta; \theta_n^z\right) \right|^2 \right] \\ + C \Delta t_n \mathbb{E}\left[\left| \left(\hat{\Gamma}_n^\Delta - \phi_n^\gamma\left(X_n^\Delta; \theta_n^\gamma\right) \right) b(t_n, X_n^\Delta) \right|^2 \right]. \end{aligned} \tag{2.39}$$

We define $\hat{\mathbf{L}}_n^{\Delta} := \lambda_1 \hat{\mathbf{L}}_n^{y,\Delta} + \lambda_2 \hat{\mathbf{L}}_n^{z,\Delta}$. Using the bounds in (2.38) and (2.39) yields

$$\begin{aligned}
\hat{\mathbf{L}}_n^{\Delta}(\theta_n) \leq \lambda_1 (1 + C\Delta t_n)\, \mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - \phi_n^y\left(X_n^{\Delta};\theta_n^y\right)\right|^2\right] + \lambda_2 C\Delta t_n \mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - \phi_n^y\left(X_n^{\Delta};\theta_n^y\right)\right|^2\right] \\
+ \lambda_2 (1 + C\Delta t_n)\, \mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - \phi_n^z\left(X_n^{\Delta};\theta_n^z\right)\right|^2\right] + \lambda_1 C\Delta t_n \mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - \phi_n^z\left(X_n^{\Delta};\theta_n^z\right)\right|^2\right] \\
+ \lambda_2 C\Delta t_n \mathbb{E}\left[\left|\left(\hat{\Gamma}_n^{\Delta} - \phi_n^{\gamma}\left(X_n^{\Delta};\theta_n^{\gamma}\right)\right) b(t_n, X_n^{\Delta})\right|^2\right].
\end{aligned} \tag{2.40}$$

On the other hand, by using the Young inequality (2.32) for $\nu \equiv \nu \Delta t_n$, we get

$$\begin{aligned}
\hat{\mathbf{L}}_n^{y,\Delta}(\theta_n) &\geq (1 - \nu\Delta t_n)\mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - \phi_n^y\left(X_n^{\Delta};\theta_n^y\right)\right|^2\right] - \frac{1}{\nu\Delta t_n}\Delta t_n^2 \mathbb{E}\left[\left|f\left(t_n, \mathbf{X}_n^{\Delta,\theta}\right) - f\left(t_n, \hat{\mathbf{X}}_n^{\Delta}\right)\right|^2\right] \\
&\quad + \Delta t_n \mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - \phi_n^z\left(X_n^{\Delta};\theta_n^z\right)\right|^2\right], \\
&\geq (1 - \nu\Delta t_n)\, \mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - \phi_n^y\left(X_n^{\Delta};\theta_n^y\right)\right|^2\right] - \frac{2L_f^2 \Delta t_n}{\nu}\left(\mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - \phi_n^y\left(X_n^{\Delta};\theta_n^y\right)\right|^2\right]\right. \\
&\quad \left. + \mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - \phi_n^z\left(X_n^{\Delta};\theta_n^z\right)\right|^2\right]\right) + \Delta t_n \mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - \phi_n^z\left(X_n^{\Delta};\theta_n^z\right)\right|^2\right]
\end{aligned}$$

where the Lipschitz and Hölder continuity of f (2.13) and the inequality $(c_1 + c_2)^2 \leq 2\left(c_1^2 + c_2^2\right)$ are used for the second inequality. By choosing $\nu \equiv \nu^{\star} = 4L_f^2$, we get

$$\hat{\mathbf{L}}_n^{y,\Delta}(\theta_n) \geq (1 - C\Delta t_n)\, \mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - \phi_n^y\left(X_n^{\Delta};\theta_n^y\right)\right|^2\right] + \frac{\Delta t_n}{2}\mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - \phi_n^z\left(X_n^{\Delta};\theta_n^z\right)\right|^2\right], \tag{2.41}$$

and by performing similar steps for (2.37) (using Lipschitz and Hölder continuity of f_D instead of f (2.13)) yields

$$\begin{aligned}
\hat{\mathbf{L}}_n^{z,\Delta}(\theta_n) \geq (1 - C\Delta t_n)\, \mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - \phi_n^z\left(X_n^{\Delta};\theta_n^z\right)\right|^2\right] - \frac{\Delta t_n}{2}\mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - \phi_n^y\left(X_n^{\Delta};\theta_n^y\right)\right|^2\right] \\
+ \frac{\Delta t_n}{2}\mathbb{E}\left[\left|\left(\hat{\Gamma}_n^{\Delta} - \phi_n^{\gamma}\left(X_n^{\Delta};\theta_n^{\gamma}\right)\right) b(t_n, X_n^{\Delta})\right|^2\right].
\end{aligned} \tag{2.42}$$

By using the bounds in (2.41) and (2.42), we have for $\hat{\mathbf{L}}_n^{\Delta}$ that

$$\begin{aligned}
\hat{\mathbf{L}}_n^{\Delta}(\theta_n) &\geq \lambda_1 (1 - C\Delta t_n)\, \mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - \phi_n^y\left(X_n^{\Delta};\theta_n^y\right)\right|^2\right] - \frac{\lambda_2}{2}\Delta t_n \mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - \phi_n^y\left(X_n^{\Delta};\theta_n^y\right)\right|^2\right] \\
&\quad + \lambda_2 (1 - C\Delta t_n)\, \mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - \phi_n^z\left(X_n^{\Delta};\theta_n^z\right)\right|^2\right] + \frac{\lambda_1}{2}\Delta t_n \mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - \phi_n^z\left(X_n^{\Delta};\theta_n^z\right)\right|^2\right] \\
&\quad + \lambda_2 \frac{\Delta t_n}{2}\mathbb{E}\left[\left|\left(\hat{\Gamma}_n^{\Delta} - \phi_n^{\gamma}\left(X_n^{\Delta};\theta_n^{\gamma}\right)\right) b(t_n, X_n^{\Delta})\right|^2\right], \\
&\geq \lambda_1 (1 - C\Delta t_n)\, \mathbb{E}\left[\left|\hat{Y}_n^{\Delta} - \phi_n^y\left(X_n^{\Delta};\theta_n^y\right)\right|^2\right] + \lambda_2 (1 - C\Delta t_n)\, \mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - \phi_n^z\left(X_n^{\Delta};\theta_n^z\right)\right|^2\right] \\
&\quad + \lambda_2 \frac{\Delta t_n}{2}\mathbb{E}\left[\left|\left(\hat{\Gamma}_n^{\Delta} - \phi_n^{\gamma}\left(X_n^{\Delta};\theta_n^{\gamma}\right)\right) b(t_n, X_n^{\Delta})\right|^2\right].
\end{aligned} \tag{2.43}$$

Step 5. Let us fix $n \in \{0, 1, \ldots, N-1\}$. We assume that $\hat{\theta}_n$ is a perfect approximation of the optimal parameters $\theta_n^{\star} = \left(\theta_n^{y,\star}, \theta_n^{z,\star}, \theta_n^{\gamma,\star}\right) \in \arg\min_{\theta_n \in \Theta_n} \mathbf{L}_n^{\Delta}(\theta_n)$ so that $Y_n^{\Delta,\hat{\theta}} = \phi_n^y\left(\cdot;\theta_n^{y,\star}\right)$,

$Z_n^{\Delta,\hat{\theta}} = \phi_n^z\left(\cdot;\theta_n^{z,\star}\right)$ and $\Gamma_n^{\Delta,\hat{\theta}} = \phi_n^\gamma\left(\cdot;\theta_n^{\gamma,\star}\right)$. In other words, we assume that the SGD method is not trapped in a local minimum, and we neglect the estimation error resulting from minimizing an empirical loss function. We also have that $\theta_n^\star \in \arg\min_{\theta_n\in\Theta_n} \hat{\mathbf{L}}_n^\Delta(\theta_n)$ from (2.35). Hence, $\hat{\mathbf{L}}_n^\Delta(\theta_n^\star) \le \hat{\mathbf{L}}_n^\Delta(\theta_n)$ for any θ_n. By using the upper bound (2.40) and the lower bound (2.43) of $\hat{\mathbf{L}}_n^\Delta(\theta_n)$, we then have for all θ_n

$$
\begin{aligned}
&\lambda_1(1-C\Delta t_n)\,\mathbb{E}\left[\left|\hat{Y}_n^\Delta - Y_n^{\Delta,\hat{\theta}}\right|^2\right] + \lambda_2(1-C\Delta t_n)\,\mathbb{E}\left[\left|\hat{Z}_n^\Delta - Z_n^{\Delta,\hat{\theta}}\right|^2\right]\\
&+\lambda_2\frac{\Delta t_n}{2}\mathbb{E}\left[\left|\left(\hat{\Gamma}_n^\Delta - \Gamma_n^{\Delta,\hat{\theta}}\right)b(t_n,X_n^\Delta)\right|^2\right] \le \hat{\mathbf{L}}_n^\Delta(\theta_n^\star) \le \hat{\mathbf{L}}_n^\Delta(\theta_n) \le\\
&\lambda_1(1+C\Delta t_n)\,\mathbb{E}\left[\left|\hat{Y}_n^\Delta - \phi_n^y\left(X_n^\Delta;\theta_n^y\right)\right|^2\right] + \lambda_2 C\Delta t_n\mathbb{E}\left[\left|\hat{Y}_n^\Delta - \phi_n^y\left(X_n^\Delta;\theta_n^y\right)\right|^2\right]\\
&+\lambda_2(1+C\Delta t_n)\,\mathbb{E}\left[\left|\hat{Z}_n^\Delta - \phi_n^z\left(X_n^\Delta;\theta_n^z\right)\right|^2\right] + \lambda_1 C\Delta t_n\mathbb{E}\left[\left|\hat{Z}_n^\Delta - \phi_n^z\left(X_n^\Delta;\theta_n^z\right)\right|^2\right]\\
&+\lambda_2 C\Delta t_n\mathbb{E}\left[\left|\left(\hat{\Gamma}_n^\Delta - \phi_n^\gamma\left(X_n^\Delta;\theta_n^\gamma\right)\right)b(t_n,X_n^\Delta)\right|^2\right].
\end{aligned}
$$

For Δt_n sufficiently small satisfying $C\Delta t_n \le \frac{1}{2}$ and using (2.20), we have

$$
\begin{aligned}
&\lambda_1\mathbb{E}\left[\left|\hat{Y}_n^\Delta - Y_n^{\Delta,\hat{\theta}}\right|^2\right] + \lambda_2\mathbb{E}\left[\left|\hat{Z}_n^\Delta - Z_n^{\Delta,\hat{\theta}}\right|^2\right] + \lambda_2\Delta t_n\mathbb{E}\left[\left|\left(\hat{\Gamma}_n^\Delta - \Gamma_n^{\Delta,\hat{\theta}}\right)b(t_n,X_n^\Delta)\right|^2\right]\\
&\le \lambda_1 C\varepsilon_n^y + \lambda_2 C\Delta t_n\varepsilon_n^y + \lambda_2 C\varepsilon_n^z + \lambda_1 C\Delta t_n\varepsilon_n^z + \lambda_2 C\Delta t_n\varepsilon_n^\gamma.
\end{aligned}
\tag{2.44}
$$

After inserting the last inequality into (2.34), we obtain

$$
\begin{aligned}
&\max_{0\le n\le N}\mathbb{E}\left[\left|Y_n - Y_n^{\Delta,\hat{\theta}}\right|^2\right] + \max_{0\le n\le N}\mathbb{E}\left[\left|Z_n - Z_n^{\Delta,\hat{\theta}}\right|^2\right]\\
&\le C\mathbb{E}\left[\left|g(X_T) - g\left(X_N^\Delta\right)\right|^2\right] + C\mathbb{E}\left[\left|\nabla_x g(X_T) - \nabla_x g\left(X_N^\Delta\right)\right|^2\right] + C|\Delta| + C\varepsilon^{DZ}(|\Delta|)\\
&\quad + CN\lambda_1\sum_{n=0}^{N-1}\varepsilon_n^y + C\lambda_2\sum_{n=0}^{N-1}\varepsilon_n^y + CN\lambda_2\sum_{n=0}^{N-1}\varepsilon_n^z + C\lambda_1\sum_{n=0}^{N-1}\varepsilon_n^z + C\lambda_2\sum_{n=0}^{N-1}\varepsilon_n^\gamma.
\end{aligned}
\tag{2.45}
$$

This completes the proof of the consistency of processes Y and Z in Theorem 2.3.1.

Step 6. It now remains to prove the consistency for the process Γ. Let us fix $n \in \{0,1,\dots,N-1\}$. Using (2.29) in (2.27), we get

$$
\begin{aligned}
&\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|D_nZ_s - D_n\hat{Z}_n^\Delta\right|^2 ds\right]\\
&\le \mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|D_nZ_s - \widehat{DZ}_n\right|^2 ds\right] + 2d\left(\mathbb{E}\left[\left|D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right|^2\right]\right.\\
&\quad\left.-\mathbb{E}\left[\left|\mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right]\right) + 2d\Delta t_n\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|f_D\left(s,\mathbf{X}_s,\mathbf{D}_n\mathbf{X}_s\right)\right|^2 ds\right].
\end{aligned}
$$

Summing from $n = 0, \dots, N-1$, using (2.15) and (2.18) gives

$$
\begin{aligned}
&\mathbb{E}\left[\sum_{n=0}^{N-1}\int_{t_n}^{t_{n+1}}\left|D_nZ_s - D_n\hat{Z}_n^{\Delta}\right|^2 ds\right] \\
&\leq \varepsilon^{DZ}\left(|\Delta|\right) + C|\Delta| + 2d\mathbb{E}\left[\left|D_{N-1}Y_N - D_{N-1}Y_N^{\Delta,\hat{\theta}}\right|^2\right] \\
&\quad + 2d\sum_{n=1}^{N-1}\left(\mathbb{E}\left[\left|D_{n-1}Y_n - D_{n-1}Y_n^{\Delta,\hat{\theta}}\right|^2\right] - \mathbb{E}\left[\left|\mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right]\right), \qquad (2.46) \\
&\leq \varepsilon^{DZ}\left(|\Delta|\right) + C|\Delta| + C\mathbb{E}\left[\left|\nabla_x g\left(X_N\right) - \nabla_x g\left(X_N^{\Delta}\right)\right|^2\right] \\
&\quad + 2d\sum_{n=1}^{N-1}\left(\mathbb{E}\left[\left|D_{n-1}Y_n - D_{n-1}Y_n^{\Delta,\hat{\theta}}\right|^2\right] - \mathbb{E}\left[\left|\mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right]\right),
\end{aligned}
$$

where the summation index is changed for the last summation in the first inequality, and (2.33) is used in the second inequality.

Using similar steps as for (2.33), we have that

$$
\begin{aligned}
&\mathbb{E}\left[\left|D_{n-1}Y_n - D_{n-1}Y_n^{\Delta,\hat{\theta}}\right|^2\right] - \mathbb{E}\left[\left|\mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right] \\
&\leq C\mathbb{E}\left[\left|Z_n - Z_n^{\Delta,\hat{\theta}}\right|^2\right] + C|\Delta| - \mathbb{E}\left[\left|\mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right].
\end{aligned}
$$

Moreover, using

$$
\left(1 - |\Delta|\right)\mathbb{E}\left[\left|Z_n - Z_n^{\Delta,\hat{\theta}}\right|^2\right] - \frac{1}{|\Delta|}\mathbb{E}\left[\left|Z_n^{\Delta,\hat{\theta}} - \hat{Z}_n^{\Delta}\right|^2\right] \leq \mathbb{E}\left[\left|Z_n - \hat{Z}_n^{\Delta}\right|^2\right]
$$

and (2.30), we have for $|\Delta|$ small enough:

$$
\begin{aligned}
&\mathbb{E}\left[\left|D_{n-1}Y_n - D_{n-1}Y_n^{\Delta,\hat{\theta}}\right|^2\right] - \mathbb{E}\left[\left|\mathbb{E}_n\left[D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right]\right|^2\right] \\
&\leq C\mathbb{E}\left[\left|D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right|^2\right] + C|\Delta| + C|\Delta|\mathbb{E}\left[\left|Y_n - \hat{Y}_n^{\Delta}\right|^2\right] + C|\Delta|\mathbb{E}\left[\left|Z_n - \hat{Z}_n^{\Delta}\right|^2\right] \\
&\quad + C\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|D_nZ_s - \widehat{DZ}_n\right|^2 ds\right] + C|\Delta|\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|f_D\left(s, \mathbf{X}_s, \mathbf{D}_n\mathbf{X}_s\right)\right|^2 ds\right] \\
&\quad + CN\mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - Z_n^{\Delta,\hat{\theta}}\right|^2\right].
\end{aligned}
$$

Hence, (2.46) becomes

$$
\begin{aligned}
&\mathbb{E}\left[\sum_{n=0}^{N-1}\int_{t_n}^{t_{n+1}}\left|D_nZ_s - D_n\hat{Z}_n^{\Delta}\right|^2 ds\right] \\
&\leq C\varepsilon^{DZ}\left(|\Delta|\right) + C|\Delta| + C\mathbb{E}\left[\left|\nabla_x g\left(X_N\right) - \nabla_x g\left(X_N^{\Delta}\right)\right|^2\right] \\
&\quad + C\sum_{n=1}^{N-1}\mathbb{E}\left[\left|D_nY_{n+1} - D_nY_{n+1}^{\Delta,\hat{\theta}}\right|^2\right] + C|\Delta|\sum_{n=1}^{N-1}\mathbb{E}\left[\left|Y_n - \hat{Y}_n^{\Delta}\right|^2\right] \\
&\quad + C|\Delta|\sum_{n=1}^{N-1}\mathbb{E}\left[\left|Z_n - \hat{Z}_n^{\Delta}\right|^2\right] + CN\sum_{n=1}^{N-1}\mathbb{E}\left[\left|\hat{Z}_n^{\Delta} - Z_n^{\Delta,\hat{\theta}}\right|^2\right],
\end{aligned}
$$

where we used (2.15) and (2.18). From (2.31) and (2.34), we have that

$$\begin{aligned}
&\mathbb{E}\left[\sum_{n=0}^{N-1}\int_{t_n}^{t_{n+1}}\left|D_nZ_s-D_n\hat{Z}_n^{\Delta}\right|^2ds\right]\\
&\leq C\mathbb{E}\left[\left|g(X_T)-g\left(X_N^{\Delta}\right)\right|^2\right]+C\mathbb{E}\left[\left|\nabla_x g(X_T)-\nabla_x g\left(X_N^{\Delta}\right)\right|^2\right]+C|\Delta|+C\varepsilon^{DZ}\left(|\Delta|\right) \qquad (2.47)\\
&\quad+CN\sum_{n=0}^{N-1}\mathbb{E}\left[\left|\hat{Y}_n^{\Delta}-Y_n^{\Delta,\hat{\theta}}\right|^2\right]+CN\sum_{n=0}^{N-1}\mathbb{E}\left[\left|\hat{Z}_n^{\Delta}-Z_n^{\Delta,\hat{\theta}}\right|^2\right].
\end{aligned}$$

Finally, using

$$\begin{aligned}
\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|D_nZ_s-D_nZ_n^{\Delta,\hat{\theta}}\right|^2ds\right]&\leq 2\mathbb{E}\left[\int_{t_n}^{t_{n+1}}\left|D_nZ_s-D_n\hat{Z}_n^{\Delta}\right|^2ds\right]\\
&\quad+2\Delta t_n\mathbb{E}\left[\left|D_n\hat{Z}_n^{\Delta}-D_nZ_n^{\Delta,\hat{\theta}}\right|^2ds\right],
\end{aligned}$$

summing over $n=0,\dots,N-1$ and applying the inequalities (2.44) and (2.47), we derive the proof of the consistency of process Γ as expressed in (2.45). This concludes the proof of Theorem 2.3.1. □

Remark 2.3.1. *According to Theorem 2.3.1, the total approximation error of the DLBDP scheme consists of four terms. The first term correspond to*

(i) the strong approximation of the terminal condition and its gradient, depending on the Euler-Maruyama scheme and the functions $(g(x),\nabla_x g(x))$,

(ii) the strong approximation of the Euler-Maruyama scheme and the path regularity of the processes (Y,Z), see Theorem 2.2.3.

The second term represents the $\mathbb{L}^2$-regularity of DZ. All the aforementioned terms converge to zero as $|\Delta|$ goes to zero, with a rate of $|\Delta|$ when Assumptions AX4 and AY4 are satisfied. For the last two terms, the better the DNNs are able to estimate the functions (2.20), the smaller is their contribution in the total approximation error. Note that from the universal approximation theorem [41, 19], the approximation error from the DNNs can be made arbitrarily small for a sufficiently large number of hidden neurons. It is crucial noting that, in contrast to both the DBDP scheme and the method outlined in [76], the DLBDP scheme provides a means to manage the impact of the DNN's approximation error. This is accomplished by selecting the values of λ_1 and λ_2, resulting in improved accuracy for the processes (Y,Z,Γ), as we demonstrate in Section 2.5.

2.4 Forward differential deep learning schemes

In this section, we introduce a forward differential deep learning scheme, which extends the well-known LDBSDE scheme [86] described in Section 1.3.2.

As mentioned before, a drawback of the forward deep learning methods such as the LDBSDE scheme is their struggle to provide highly accurate first and second-order gradient approximations, namely the process Z and Γ. This limitation becomes apparent in the loss function (1.9) of the LDBSDE scheme, as the SGD algorithm lacks the explicit information about the dynamics of Z and does not explicitly include Γ. In a differential deep learning problem, the loss

function requires explicit information about the labels and their derivatives with respect to inputs [42]. Therefore, transforming the BSDE into a differential deep learning problem provides the necessary information to the SGD algorithm. This is done by using the Malliavin calculus.

Applying the Malliavin derivative to (1.1) yields another BSDE. These are given in (2.2)-(2.6). As for the LDBSDE scheme, we firstly discretize the integrals in such BSDE system using the Euler-Maruyama method, providing the approximations given in Section 1.3 in (1.5) and (1.6), and Section 2.3.1, (2.7) and (2.11). After discretizing the integrals, our scheme is made fully implementable by approximating the unknown processes $\left(Y_n^\Delta, Z_n^\Delta, \Gamma_n^\Delta\right)$ in the discrete BSDE system (1.6) and (2.11) using three DNNs for $n = 0, 1, \ldots, N$. We refer to our scheme as differential LDBSDE (DLDBSDE), which works as follows:

- Generate approximations X_{n+1}^Δ for $n = 0, 1, \ldots, N-1$ using (1.5) and its discrete Malliavin derivative $D_n X_n^\Delta, D_n X_{n+1}^\Delta$ using (2.7).
- At each discrete time point t_n, $n = 0, 1, \ldots, N$, use DNNs $\phi^y(\cdot;\theta^y) : \mathbb{R}^{1+d} \to \mathbb{R}$, $\phi^z(\cdot;\theta^z) : \mathbb{R}^{1+d} \to \mathbb{R}^{1\times d}$ and $\phi^\gamma(\cdot;\theta^\gamma) : \mathbb{R}^{1+d} \to \mathbb{R}^{d\times d}$ to approximate the discrete processes $\left(Y_n^\Delta, Z_n^\Delta, \Gamma_n^\Delta\right)$, respectively, where the input vector of the network is the time value $t_n \in \mathbb{R}_+$ and the Markovian process $X_n^\Delta \in \mathbb{R}^d$, namely
$$Y_n^{\Delta,\theta} = \phi^y(t_n, X_n^\Delta; \theta^y), \quad Z_n^{\Delta,\theta} = \phi^z(t_n, X_n^\Delta; \theta^z), \quad \Gamma_n^{\Delta,\theta} = \phi^\gamma(t_n, X_n^\Delta; \theta^\gamma).$$
- Train the parameters $\theta = (\theta^y, \theta^z, \theta^\gamma)$ using a global differential loss type function including local losses such that the dynamics of discretized BSDE system (1.6) and (2.11) are satisfied at each time step, namely
$$\begin{aligned}
\mathbf{L}^\Delta(\theta) &= \lambda_1 \mathbf{L}^{y,\Delta}(\theta) + \lambda_2 \mathbf{L}^{z,\Delta}(\theta) \\
\mathbf{L}^{y,\Delta}(\theta) &= \mathbb{E}\left[\sum_{n=0}^{N-1} \left|Y_{n+1}^{\Delta,\theta} - Y_n^{\Delta,\theta} + f\left(t_n, \mathbf{X}_n^{\Delta,\theta}\right)\Delta t - Z_n^{\Delta,\theta}\Delta W_n\right|^2 + \left|Y_N^{\Delta,\theta} - g\left(X_N^\Delta\right)\right|^2\right] \\
\mathbf{L}^{z,\Delta}(\theta_n) &= \mathbb{E}\left[\sum_{n=0}^{N-1} \left|D_n Y_{n+1}^{\Delta,\theta} - Z_n^{\Delta,\theta} + f_D\left(t_n, \mathbf{X}_n^{\Delta,\theta}, \mathbf{D}_n \mathbf{X}_n^{\Delta,\theta}\right)\Delta t - \left(\left(\Gamma_n^{\Delta,\theta} D_n X_n^\Delta\right)^\top \Delta W_n\right)^\top\right|^2 \right. \\
&\qquad \left. + \left|Z_N^{\Delta,\theta} - g_x\left(X_N^\Delta\right) b\left(t_N, X_N^\Delta\right)\right|^2\right],
\end{aligned} \tag{2.48}$$
where $D_n Y_{n+1}^{\Delta,\theta} = Z_{n+1}^{\Delta,\theta} b^{-1}\left(t_{n+1}, X_{n+1}^\Delta\right) D_n X_{n+1}^\Delta$.
- Approximate the optimal parameters $\theta^* \in \arg\min_{\theta\in\Theta} \mathbf{L}^\Delta(\theta)$ using a SGD method and receive the final estimated parameters $\hat{\theta}$. Set the final approximation of $\left(Y_n^\Delta, Z_n^\Delta, \Gamma_n^\Delta\right)$ as $\left(Y_n^{\Delta,\hat{\theta}}, Z_n^{\Delta,\hat{\theta}}, \Gamma_n^{\Delta,\hat{\theta}}\right)$ for $n = 0, 1, \ldots, N$.

Note that LDBSDE scheme can be considered as a special case of our scheme by choosing $\lambda_1 = 1$ and $\lambda_2 = 0$, and using one DNN for Y and AD for the processes Z and Γ. Similarly to the DLBDP scheme, we chose $\lambda_1 = \frac{1}{d+1}$ and $\lambda_2 = \frac{d}{d+1}$. The difference between the DLDBSDE scheme and the DLBDP scheme is outlined in Remark 2.4.1, and its convergence analysis in Remark 2.4.2.

Remark 2.4.1. *The DLDBSDE scheme significantly differs from the DLBDP scheme. Unlike the latter, which relied on local optimization, the former is based on global optimization.*

Remark 2.4.2. *The convergence of the LDBSDE scheme is discussed in [66], providing an a posteriori error estimation similar to [39] for the DBSDE scheme. The authors demonstrate that the error of the LDBSDE scheme is bounded by its respective loss function* (1.9)*, and the loss functional converges sufficiently fast to zero, ensuring that the error of the scheme vanishes in the limit. This result is achievable through the universal approximation theorem [41, 19] of NNs. An a posteriori error analysis for the DLDBSDE scheme can be conducted by following the methodology in [66] and Section 2.3.2. The latter includes the additional assumptions, namely Assumptions AX4 and AY4 needed for ensuring the boundedness of the Malliavin derivatives. It also provides the extra steps required to address the discretization error introduced by the Euler-Maruyama method and the model/approximation error from the DNNs* (ϕ^z, ϕ^γ) *associated with the second term in the loss function* (2.48)*. This is part of our ongoing research.*

2.5 Numerical results

The numerical results for forward-type schemes have been presented in the previous chapter. Therefore, in this chapter, we focus on the numerical results of backward-type schemes. Specifically, we illustrate the improved performance of the DLBDP scheme compared to the DBDP scheme not only when approximating the solution, but also its gradient and the Hessian matrix. Moreover, we show that our scheme achieves similar accuracy compared to the OSM scheme for less computation time. As high-accurate gradient approximations are of great importance in finance, we consider linear and nonlinear option pricing examples. All the experiments below were conducted using PYTHON and TensorFlow on the PLEIADES cluster's GPU nodes.

In all the following examples, we consider the same hyperparameters for our scheme and both the DBDP and OSM schemes for a fair comparison. For the DNNs, we choose $L = 2$ hidden layers and $\eta = 100 + d$ neurons per hidden layer. The input is normalized based on the true moments. The input is not normalized at discrete time point t_0, as the standard deviation is zero. A hyperbolic tangent activation $\tanh(\cdot)$ is applied on each hidden layer. It is crucial to mention that one can't apply batch normalization for the hidden layers as AD is required to approximate the process Γ in the DBDP scheme. This is because using batch normalization creates dependence for the gradients in the batch, since it normalizes across the batch dimension. Using the method tf.GradientTape.batch_jacobian to approximate Γ when the DNN that approximates Z involves tf.keras.layers.BatchNormalization layers returns something with the expected shape, but its contents have an unclear meaning, see TensorFlow documentation[1], batch Jacobian section. Therefore, batch normalization is emitted not only in the DBDP scheme, but also in our scheme to ensure a fair comparison. For the SGD iterations, we use the Adam optimizer with a PC-LR approach. We choose a batch size of $B = 1024$ for each of κ optimization steps. At the discrete time point t_{N-1}, we consider $\mathfrak{K} = 24000$ optimization steps, where the learning rate α is adjusted as follows

$$\alpha_\kappa = \begin{cases} 1\text{e}{-2}, & \text{for } 1 \leq \kappa \leq 2000, \\ 3\text{e}{-3}, & \text{for } 2000 < \kappa \leq 4000, \\ 1\text{e}{-3}, & \text{for } 4000 < \kappa \leq 8000, \\ 3\text{e}{-4}, & \text{for } 8000 < \kappa \leq 12000, \\ 1\text{e}{-4}, & \text{for } 12000 < \kappa \leq 16000, \\ 3\text{e}{-5}, & \text{for } 16000 < \kappa \leq 20000, \\ 1\text{e}{-5}, & \text{for } 20000 < \kappa \leq \mathfrak{K}. \end{cases}$$

[1] https://www.tensorflow.org/guide/advanced_autodiff#batch_jacobian

For the next discrete time points (i.e., $t_{N-2}, \dots, t_0$), we make use of the transfer learning approach, and reduce the number of optimization steps to $\mathfrak{K} = 10000$, and use the following learning rates

$$\alpha_\kappa = \begin{cases} 1\text{e}{-3}, & \text{for } 1 \leq \kappa \leq 2000, \\ 3\text{e}{-4}, & \text{for } 2000 < \kappa \leq 4000, \\ 1\text{e}{-4}, & \text{for } 4000 < \kappa \leq 6000, \\ 3\text{e}{-5}, & \text{for } 6000 < \kappa \leq 8000, \\ 1\text{e}{-5}, & \text{for } 8000 < \kappa \leq \mathfrak{K}. \end{cases}$$

The gradient of the driver function f w.r.t each variable (x, y, z) and the function g w.r.t to variable x are calculated by using AD, namely tf.GradientTape in TensorFlow. For the gradient of the function representing Z_t (when available) in (1.3) w.r.t to variable x, tf.GradientTape.batch_jacobian is used. Note that we consider a uniform time discretization Δ of $[0, T]$. The DLBDP algorithm (without ln-transformation) calculating the final estimates $\left(Y_n^{\Delta,\hat{\theta}}, Z_n^{\Delta,\hat{\theta}}\right)$ for $n = N-1, \dots, 1, 0$, is given in Algorithm 3 when using the aforementioned learning rate decay and transfer learning approaches. The parameters $\hat{\theta}$ are an estimation of θ^* due to the optimization error resulting from the Adam optimization algorithm and the estimation error from the empirical version of loss (2.12) given as

$$\begin{aligned} \tilde{\mathbf{L}}_n^\Delta(\theta_n) &= \lambda_1 \tilde{\mathbf{L}}_n^{y,\Delta}(\theta_n) + \lambda_2 \tilde{\mathbf{L}}_n^{z,\Delta}(\theta_n), \\ \tilde{\mathbf{L}}_n^{y,\Delta}(\theta_n) &= \frac{1}{B}\sum_{j=1}^{B} \left| Y_{n+1,j}^{\Delta,\hat{\theta}} - \phi_n^y\left(X_{n,j}^\Delta; \theta_n^y\right) + f\left(t_n, \mathbf{X}_{n,j}^{\Delta,\theta}\right)\Delta t - \phi_n^z\left(X_{n,j}^\Delta; \theta_n^z\right)\Delta W_{n,j} \right|^2, \\ \tilde{\mathbf{L}}_n^{z,\Delta}(\theta_n) &= \frac{1}{B}\sum_{j=1}^{B} \Bigg| Z_{n+1,j}^{\Delta,\hat{\theta}} b^{-1}\left(t_{n+1}, X_{n+1,j}^\Delta\right) D_n X_{n+1,j}^\Delta - \phi_n^z\left(X_{n,j}^\Delta; \theta_n^z\right) \\ &\quad + f_D\left(t_n, \mathbf{X}_{n,j}^{\Delta,\theta}, \mathbf{D}_n \mathbf{X}_{n,j}^{\Delta,\theta}\right)\Delta t - \left(\left(\phi_n^\gamma\left(X_{n,j}^\Delta; \theta_n^\gamma\right) D_n X_{n,j}^\Delta\right)^\top \Delta W_{n,j}\right)^\top \Bigg|^2, \end{aligned}$$

for a batch size B.

As performance metrics for a sample with the size B, we use the MSE, mean MSE, and its relative measure as defined in (1.14), (1.15) and (1.16), respectively, where for the process Γ we have that, e.g., the MSE is given as

$$\tilde{\varepsilon}_n^\gamma := \frac{1}{B}\sum_{j=1}^{B} \left| \Gamma_{n,j} - \Gamma_{n,j}^{\Delta,\hat{\theta}} \right|^2.$$

The average computation time over $Q = 10$ runs of the algorthms is given as in (1.17).

2.5.1 The Black-Scholes BSDE

We start with a linear BSDE - the Black-Scholes BSDE - which is used for pricing of European options.

Example 2.5.1. *The high-dimenisonal Black-Scholes BSDE reads [100]*

$$\begin{cases} dX_t^k &= (a_k - \delta_k) X_t^k\, dt + b_k X_t^k\, dW_t^k, \\ X_0^k &= x_0^k, \quad k = 1, \dots, d, \\ -dY_t &= -\left(RY_t + \sum_{k=1}^d \frac{a_k - R + \delta_k}{b_k} Z_t^k\right) dt - Z_t\, dW_t, \\ Y_T &= \left(\prod_{k=1}^d \left(X_T^k\right)^{c_k} - K\right)^+, \end{cases}$$

Algorithm 3: Algorithm of DLBDP scheme

Input: (N, d, T, x_0) - problem parameters; (a, b, f, g) - functions of the BSDE

Input: $(\alpha, \mathfrak{K}, L, \eta, \varrho, B, \lambda_1, \lambda_2)$ - DNN hyperparameters

Output: $\left(Y_n^{\Delta,\hat{\theta}}, Z_n^{\Delta,\hat{\theta}}, \Gamma_n^{\Delta,\hat{\theta}}\right)$, $n = 0, \dots, N-1$ - estimated solution

for $n = N-1 : 0$ **do**

$t_n = n\,\Delta t$, $\Delta t = \frac{T}{N}$

$\hat{\theta}_n^0 = \left(\hat{\theta}_n^{y,0}, \hat{\theta}_n^{z,0}, \hat{\theta}_n^{\gamma,0}\right)$ - Xavier normal initializer [31]

if $n < N-1$ **then**

$\hat{\theta}_n^0 = \hat{\theta}_{n+1}$ - Transfer learning approach

end

for $\kappa = 1 : \mathfrak{K}$ **do**

for $j = 1 : B$ **do**

$X_{0,j}^{\Delta} = x_0$

for $n = 0 : N-1$ **do**

$$X_{n+1,j}^{\Delta} = X_{n,j}^{\Delta} + a\left(t_n, X_{n,j}^{\Delta}\right)\Delta t + b\left(t_n, X_{n,j}^{\Delta}\right)\Delta W_{n,j},$$
$$\Delta W_{n,j} \sim \mathcal{N}(\mathbf{0}_d, \Delta t\,\mathbf{I}_d)$$
$$D_n X_{n,j}^{\Delta} = b\left(t_n, X_{n,j}^{\Delta}\right)$$
$$D_n X_{n+1,j}^{\Delta} = D_n X_{n,j}^{\Delta} + \nabla_x a\left(t_n, X_{n,j}^{\Delta}\right) D_n X_{n,j}^{\Delta}\Delta t + \nabla_x b\left(t_n, X_{n,j}^{\Delta}\right) D_n X_{n,j}^{\Delta}\Delta W_{n,j}$$

end

Use three DNNs with hyperparamters (L, η, ϱ) **and input** $X_{n,j}^{\Delta}$**:**

$$Y_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} = \phi_n^y\left(\cdot; \hat{\theta}_n^{y,\kappa-1}\right), Z_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} = \phi_n^z\left(\cdot; \hat{\theta}_n^{z,\kappa-1}\right), \Gamma_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} = \phi_n^\gamma\left(\cdot; \hat{\theta}_n^{\gamma,\kappa-1}\right)$$

if $n = N-1$ **then**

$$Y_{n+1,j}^{\Delta,\hat{\theta}} = g(X_{N,j}^{\Delta}), \quad Z_{n+1,j}^{\Delta,\hat{\theta}} = \nabla_x g(X_{N,j}^{\Delta}) b(t_N, X_{N,j}^{\Delta})$$

else

$$Y_{n+1,j}^{\Delta,\hat{\theta}} = \phi_{n+1}^y\left(X_{n+1,j}^{\Delta}; \hat{\theta}_{n+1}^y\right), \quad Z_{n+1,j}^{\Delta,\hat{\theta}} = \phi_{n+1}^z\left(X_{n+1,j}^{\Delta}; \hat{\theta}_{n+1}^z\right)$$

end

end

Empirical loss function (2.12)**:**

$$\tilde{\mathbf{L}}_n^{y,\Delta}\left(\hat{\theta}_n^{\kappa-1}\right) = \frac{1}{B}\sum_{j=1}^{B}\left|Y_{n+1,j}^{\Delta,\hat{\theta}} - Y_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} + f\left(t_n, \mathbf{X}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}}\right)\Delta t - Z_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}}\Delta W_{n,j}\right|^2$$

$$\tilde{\mathbf{L}}_n^{z,\Delta}\left(\hat{\theta}_n^{\kappa-1}\right) = \frac{1}{B}\sum_{j=1}^{B}\Bigg|Z_{n+1,j}^{\Delta,\hat{\theta}} b^{-1}\left(t_{n+1}, X_{n+1,j}^{\Delta}\right) D_n X_{n+1,j}^{\Delta} - Z_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} + f_D\left(t_n, \mathbf{X}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}}, \mathbf{D}_n \mathbf{X}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}}\right)\Delta t - \left(\left(\Gamma_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} D_n X_{n,j}^{\Delta}\right)^\top \Delta W_{n,j}\right)^\top\Bigg|^2$$

$$\tilde{\mathbf{L}}_n^{\Delta}\left(\hat{\theta}_n^{\kappa-1}\right) = \lambda_1 \tilde{\mathbf{L}}_n^{y,\Delta}\left(\hat{\theta}_n^{\kappa-1}\right) + \lambda_2 \tilde{\mathbf{L}}_n^{z,\Delta}\left(\hat{\theta}_n^{\kappa-1}\right)$$

$\hat{\theta}_n^{\kappa}$ - trained parameters at step κ using Adam [64] with α_κ

end

$\hat{\theta}_n = \hat{\theta}_n^{\mathfrak{K}}$ - final estimated DNN parameters at t_n after $\mathfrak{K}$ optimization steps

end

where $c_k > 0$ and $\sum_{k=1}^{d} c_k = 1$. Note that a_k represents the return rate of the stock X_t^k, b_k the volatility of the stock returns, δ_k is its dividend rate, and x_0^k is the price of the stock at $t = 0$. Moreover, X_T is the price of the stocks at time T, which denotes the maturity of the option contract. The value K represents the contract's strike price. Finally, R corresponds to the risk-free interest rate. The analytic solution (the option price Y_t and the delta-hedging strategy Z_t) is given by

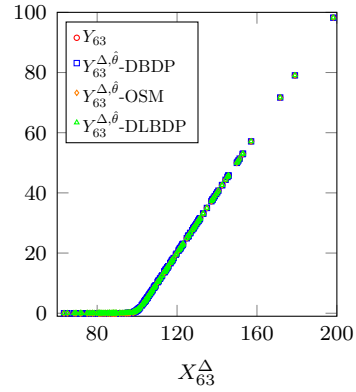
$$\begin{cases} Y_t &= u(t, X_t) = \exp\left(-\check{\delta}\,(T-t)\right) \prod_{k=1}^{d} \left(X_t^k\right)^{c_k} \Phi\left(\check{d}_1\right) - \exp\left(-R\,(T-t)\right) K \Phi\left(\check{d}_2\right), \\ Z_t^k &= \frac{\partial u}{\partial x_k} b_k X_t^k = c_k \exp\left(-\check{\delta}\,(T-t)\right) \prod_{k=1}^{d} \left(X_t^k\right)^{c_k} \Phi\left(\check{d}_1\right) b_k, \quad k = 1, \ldots, d, \\ \check{d}_1 &= \frac{\ln\left(\frac{\prod_{k=1}^{d}\left(X_t^k\right)^{c_k}}{K}\right) + \left(R - \check{\delta} + \frac{\check{b}^2}{2}\right)(T-t)}{\check{b}\sqrt{T-t}}, \\ \check{d}_2 &= \check{d}_1 - \check{b}\sqrt{T-t}, \\ \check{b}^2 &= \sum_{k=1}^{d} (b_k c_k)^2, \quad \check{\delta} = \sum_{k=1}^{d} c_k \left(\delta_k + \frac{b_k^2}{2}\right) - \frac{\check{b}^2}{2}, \end{cases} \tag{2.49}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function. The analytical solution $\Gamma_t = \nabla_x \left(\nabla_x u\,(t, X_t)\, b\,(t, X_t)\right)$ is calculated by using AD. As we mentioned in Section 2.3.2, when dealing with a forward SDE represented by the GBM, we apply the ln-transformation to ensure that the theoretical analysis is applicable to our numerical experiments. We define $\check{X}_t := \ln(X_t)$ and $\check{u}(t, \check{X}_t) := u(t, X_t)$. Using the Feynman-Kac formula, we write the Black-Scholes BSDE in the ln-domain

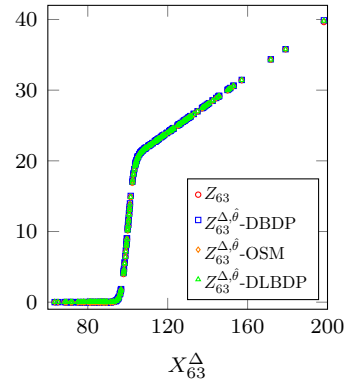
$$\begin{cases} d\check{X}_t^k &= \left(a_k - \delta_k - \frac{1}{2} b_k^2\right) dt + b_k\, dW_t^k, \\ \check{X}_0^k &= \ln\left(x_0^k\right), \quad k = 1, \ldots, d, \\ -d\check{Y}_t &= -\left(R\check{Y}_t + \sum_{k=1}^{d} \frac{a_k - R + \delta_k}{b_k} \check{Z}_t^k\right) dt - \check{Z}_t\, dW_t, \\ \check{Y}_T &= \left(\exp\left(\sum_{k=1}^{d} c_k \check{X}_T^k\right) - K\right)^+. \end{cases} \tag{2.50}$$

The ln-transformation simplifies the Malliavin derivatives as $D_n X_n^k = b_k X_n^k$, $D_n X_{n+1}^k = b_k X_{n+1}^k$ and $D_n \check{X}_n^k = D_n \check{X}_{n+1}^k = b_k$ for $k = 1, \ldots, d$. Note that $\left(\check{Y}_t, \check{Z}_t\right) = (Y_t, Z_t)$ since $\check{Y}_t = \check{u}(t, \check{X}_t) = u(t, X_t) = Y_t$ and $\check{Z}_t^k = \frac{\partial \check{u}}{\partial \check{x}_k} b_k = \frac{\partial u}{\partial x_k} b_k X_t^k = Z_t^k$ for $k = 1, \ldots, d$. Hence, we can compare the approximated solution of (2.50) in the ln-domain with the exact solution of Example 2.5.1 given in (2.49). In case of the process Γ, we have that $\check{\Gamma}_t^{k,j} \frac{1}{X_t^j} = \Gamma_t^{k,j}$ for $k, j = 1, \ldots, d$. In the following tests, for $k = 1, \ldots, d$, we set $x_0^k = 100$, $a_k = 0.05$, $b_k = 0.2$, $R = 0.03$, $c_k = \frac{1}{d}$ and $\delta_k = 0$. Moreover, we set $K = 100$, $T = 1$ and $d \in \{1, 10, 50\}$.

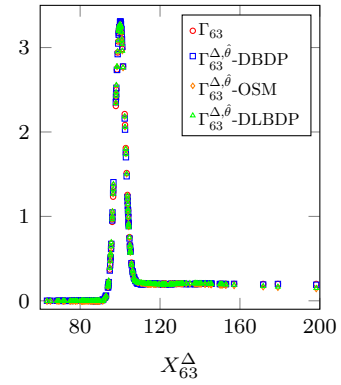
We start with $d = 1$ as we can also visualize the exact and approximated values of each process over the discrete time domain. In Figure 2.1, we display the exact and approximated value of the processes (Y, Z, Γ) from the first run of DBDP, OSM and DLBDP at arbitrary discrete time points $(t_2, t_{32}, t_{63}) = (0.0312, 0.5000, 0.9844)$ using $N = 64$. Moreover, we show only 256 out of 1024 testing samples for better visualization. Our scheme outperforms the DBDP scheme in approximating the process Γ, particularly as we approach t_0. The OSM scheme exhibits similar approximations as our method. It is difficult to observe any improvement from DLBDP scheme for the processes Y and Z in Figure 2.1. Therefore, to provide a clearer comparison using the entire testing sample across the discrete domain Δ, we visualize in Figure 2.2 the mean MSE values for each process for $d \in \{1, 10, 50\}$. The STD of the MSE values is given in the shaded area. Firstly, we compare our scheme with the DBDP scheme. For the case of $d = 1$, Figure 2.2c clearly shows a substantial improvement in approximating the process Γ across the



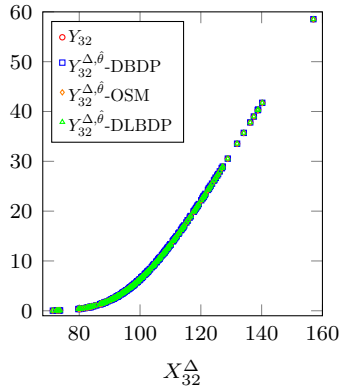
(a) Process Y at $t_{63} = 0.9844$.



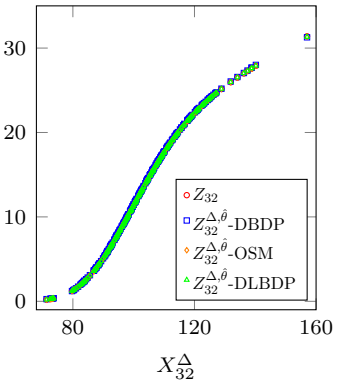
(b) Process Z at $t_{63} = 0.9844$.



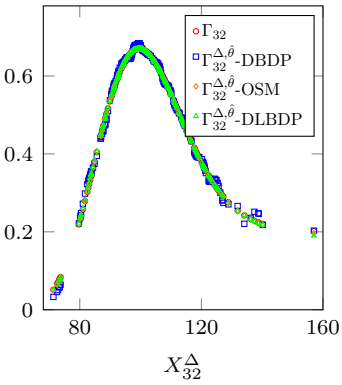
(c) Process Γ at $t_{63} = 0.9844$.



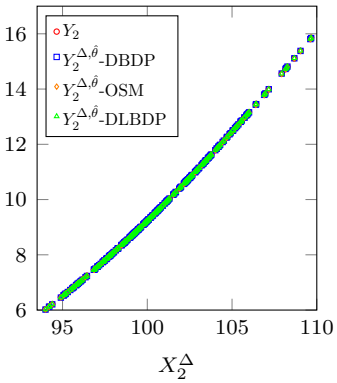
(d) Process Y at $t_{32} = 0.5000$.



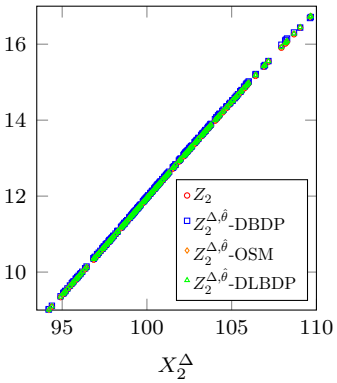
(e) Process Z at $t_{32} = 0.5000$.



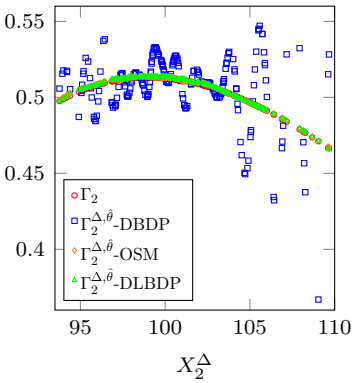
(f) Process Γ at $t_{32} = 0.5000$.



(g) Process Y at $t_2 = 0.0312$.

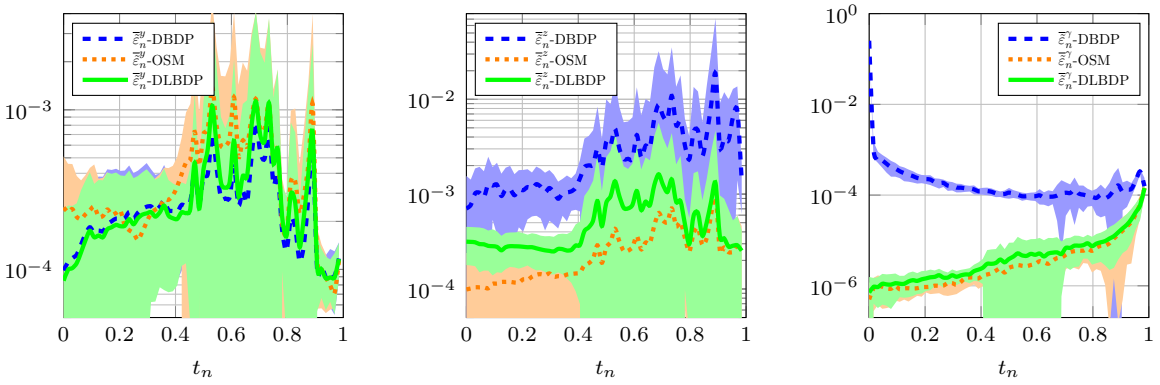


(h) Process Z at $t_2 = 0.0312$.



(i) Process Γ at $t_2 = 0.0312$.

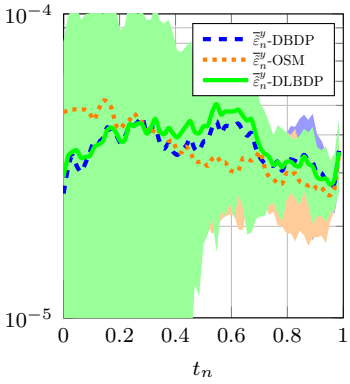
Figure 2.1: Exact and approximated values of the processes (Y, Z, Γ) from the first run of DBDP, OSM and DLBDP schemes at discrete time points $(t_2, t_{32}, t_{63}) = (0.0312, 0.5000, 0.9844)$ using 256 samples of the testing sample in Example 2.5.1, for $d = 1$ and $N = 64$.



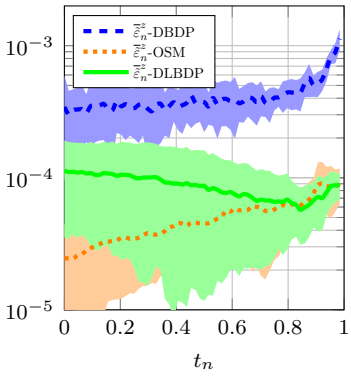
(a) Process Y, $d = 1$.

(b) Process Z, $d = 1$.

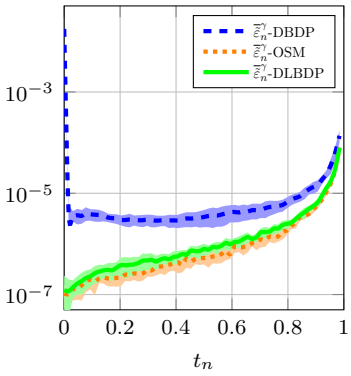
(c) Process Γ, $d = 1$.



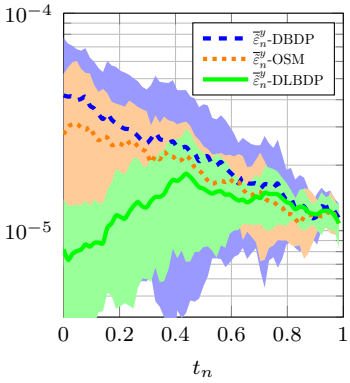
(d) Process Y, $d = 10$.



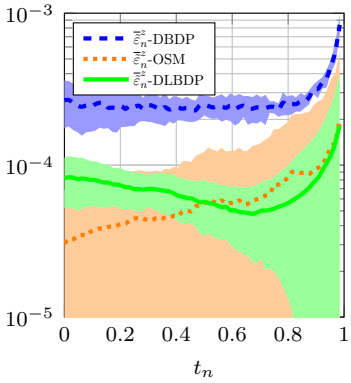
(e) Process Z, $d = 10$.



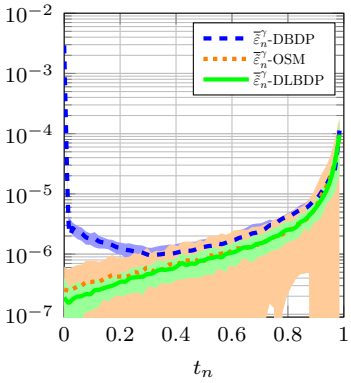
(f) Process Γ, $d = 10$.



(g) Process Y, $d = 50$.



(h) Process Z, $d = 50$.



(i) Process Γ, $d = 50$.

Figure 2.2: Mean MSE values of the processes (Y, Z, Γ) from DBDP, OSM and DLBDP schemes over the discrete time points $\{t_n\}_{n=0}^{N-1}$ using the testing sample in Example 2.5.1, for $d \in \{1, 10, 50\}$ and $N = 64$. The STD of MSE values is given in the shaded area.

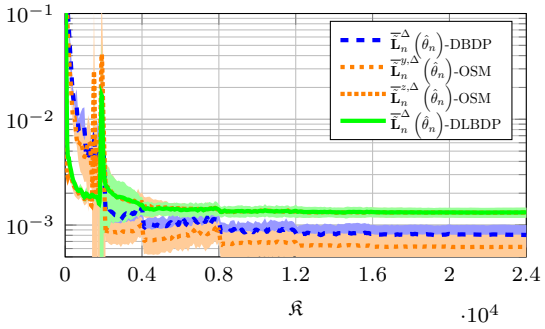
discrete time points $\{t_n\}_{n=0}^{N-1}$ achieved by our scheme compared to the DBDP scheme. Furthermore, Figure 2.2b demonstrates that the DLBDP scheme also outperforms in approximating the process Z. However, there is no improvement achieved with our scheme for the process Y, as shown in Figure 2.2a. As the dimension increases to $d = 10$ and $d = 50$, our scheme further exhibits a higher accuracy for approximating the processes (Z, Γ). Moreover, an improvement in approximating the process Y is evident for $d = 50$ from the DLBDP scheme compared to DBDP scheme, as displayed in Figure 2.2g. The approximations from our scheme and the OSM scheme are comparable. Specifically, both schemes yield similar approximations for the process Γ, while the OSM scheme performs better for the process Z, and our scheme gives higher accuracy for the process Y.

Next, we report in Table 2.1 the mean relative MSE of each process at t_0 while varying N for $d \in \{1, 10, 50\}$, along with the average computation time from both the DBDP, OSM and DLBDP schemes. The STD of the relative MSE values at t_0 is given in the brackets. The mean relative MSE of (Y_0, Z_0) decreases as N increases for each dimension in all schemes. This trend is also observed for Γ_0 in the OSM and DLBDP schemes, but not in the DBDP scheme, which actually diverges. Note that the mean relative MSE values start to flatten out for $N = 64$, indicating that the overall contribution of the approximation error from the DNNs increases for higher N and becomes larger than the discretization error. This is consistent with the error analysis in Section 2.3.2 (see Theorem 4.1 for the DBDP scheme [44] and Theorem 5.2 for the OSM scheme [76]). Compared to the DBDP scheme, our approach consistently yields the smallest mean relative MSE for each process, especially as the dimension increases. Both the OSM and DLBDP schemes provide overall comparable approximations. The average computation time of the DLBDP algorithm is higher compared to that of the DBDP algorithm. Note that we compare the computational time of all schemes including the computation of Γ at each optimization step. In [76] it is mentioned that the runtime of their algorithm is roughly double of the DBDP one, as it requires solving two optimization problems per discrete time step. Since in the second optimization problem only the parameters of the DNN for the process Y are optimized, one can reasonably infer that our algorithm may be up to twice as fast as the one proposed in [76]. This is observed in Table 2.1 when comparing the computation of the OSM and DLBDP schemes, especially as d and N increases (the algorithm's complexity grows due to the higher number of network parameters with increasing dimensionality and the increased number of optimization problems with larger N).

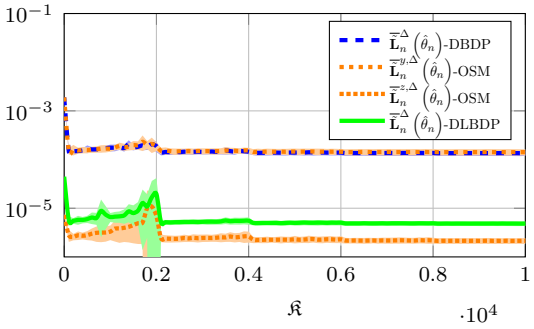
To train the algorithms, we set a high number of optimization steps (and a high number of hidden neurons) as described in the beginning of Section 2.5 s.t. the same hyperparameters are used for each example. However, the computation time of the algorithms can be reduced, e.g., by reducing the number of optimization steps. This can be seen in Figure 2.3, where we display the mean loss and MSE values of each process for all the algorithms using a validation sample $B^{valid} = 1024$, at discrete time points (t_{32}, t_{63}) in case of $d = 50$ and using $N = 64$. The mean loss is defined as $\overline{\tilde{\mathbf{L}}}_n^{\Delta}\left(\hat{\theta}_n\right) := \frac{1}{Q}\sum_{q=1}^{Q}\tilde{\mathbf{L}}_{n,q}^{\Delta}\left(\hat{\theta}_n\right)$. The STD of the loss and MSE values is given in the shaded area. By choosing for instance $\mathfrak{K} = 16000$ at t_{63} and $\mathfrak{K} = 5000$ at other discrete time points, the runtime of the algorithms is substantially reduced with almost an insignificant loss of accuracy.

2.5.2 Option pricing with different interest rates

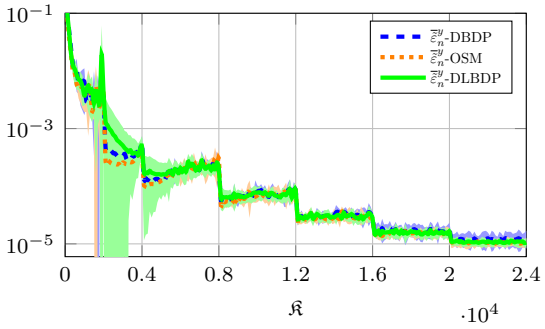
We now consider the pricing problem involving a European option in a financial market where the different interest rates for borrowing and lending are different, which is represented by a nonlinear BSDE.



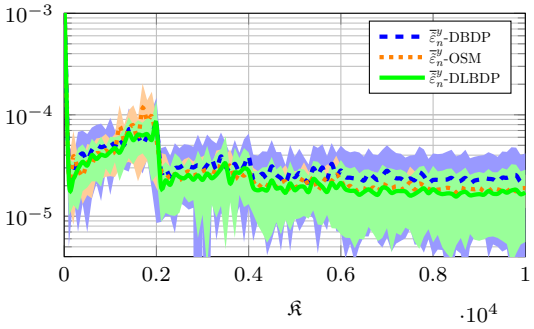
(a) Loss, $t_{63} = 0.9844$.



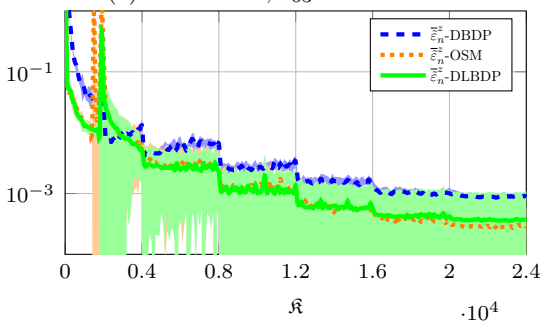
(b) Loss, $t_{32} = 0.5000$.



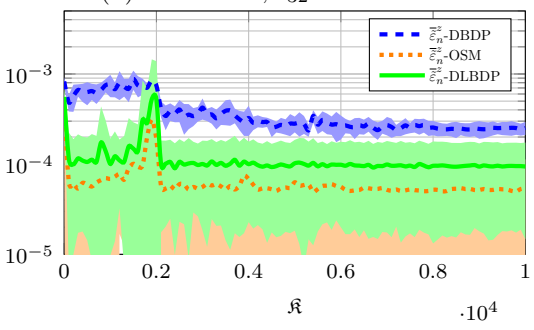
(c) Process Y, $t_{63} = 0.9844$.



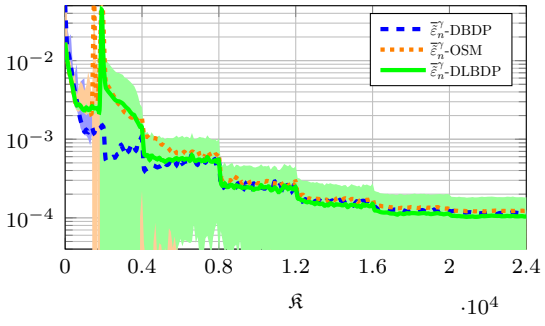
(d) Process Y, $t_{32} = 0.5000$.



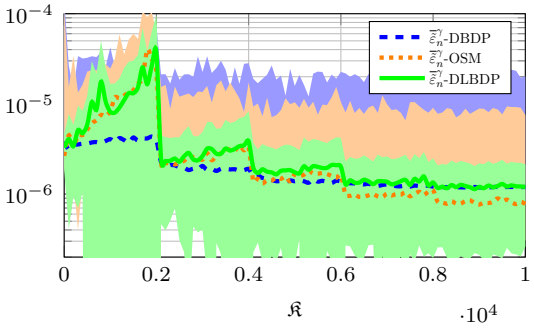
(e) Process Z, $t_{63} = 0.9844$.



(f) Process Z, $t_{32} = 0.5000$.



(g) Process Γ, $t_{63} = 0.9844$.



(h) Process Γ, $t_{32} = 0.5000$.

Figure 2.3: Mean loss and MSE values of the process (Y, Z, Γ) from DBDP, OSM and DLBDP schemes at discrete time points $(t_{32}, t_{63}) = (0.5000, 0.9844)$ using the validation sample in Example 2.5.1, for $d = 50$ and $N = 64$. The STD of the loss and MSE values is given in the shaded area.

Metric	N = 2 DBDP OSM DLBDP	N = 8 DBDP OSM DLBDP	N = 32 DBDP OSM DLBDP	N = 64 DBDP OSM DLBDP
(a) $d = 1$.				
$\overline{\varepsilon}_0^{y,r}$	1.31e−05 (1.50e−05) 4.66e−05 (3.55e−05) 8.47e−06 (9.42e−06)	3.57e−06 (1.63e−06) 4.44e−06 (3.93e−06) 3.29e−06 (4.06e−06)	2.93e−06 (4.08e−06) 8.82e−07 (1.79e−06) 3.14e−06 (4.20e−06)	1.11e−06 (1.66e−06) 2.76e−06 (3.01e−06) 9.59e−07 (1.69e−06)
$\overline{\varepsilon}_0^{z,r}$	3.20e−03 (3.58e−04) 2.54e−06 (3.06e−06) 9.46e−04 (1.28e−04)	2.04e−04 (3.06e−05) 8.94e−07 (9.66e−07) 7.47e−05 (1.20e−05)	1.91e−05 (9.24e−06) 2.14e−06 (2.55e−06) 5.79e−06 (1.56e−06)	4.93e−06 (5.90e−06) 6.90e−07 (9.79e−07) 2.20e−06 (9.52e−07)
$\overline{\varepsilon}_0^{\gamma,r}$	1.16e+00 (1.55e−02) 5.59e−05 (1.18e−05) 8.10e−04 (6.58e−05)	9.94e−01 (1.49e−03) 6.51e−06 (5.69e−06) 7.36e−05 (2.24e−05)	9.89e−01 (5.47e−03) 1.79e−06 (2.12e−06) 4.93e−06 (4.87e−06)	9.86e−01 (1.01e−02) 1.96e−06 (2.52e−06) 2.77e−06 (3.33e−06)
$\overline{\tau}$	2.14e+02 3.44e+02 2.68e+02	6.60e+02 1.03e+03 7.65e+02	2.84e+03 4.56e+03 3.16e+03	6.83e+03 1.15e+04 7.39e+03
(b) $d = 10$.				
$\overline{\varepsilon}_0^{y,r}$	4.06e−04 (1.03e−04) 6.28e−04 (1.01e−04) 4.09e−05 (3.03e−05)	1.98e−05 (1.27e−05) 4.07e−05 (2.76e−05) 8.83e−06 (5.46e−06)	4.72e−06 (6.36e−06) 1.36e−05 (1.45e−05) 4.10e−06 (4.06e−06)	2.68e−06 (3.85e−06) 4.94e−06 (3.56e−06) 3.05e−06 (5.51e−06)
$\overline{\varepsilon}_0^{z,r}$	1.77e−02 (5.69e−04) 1.05e−05 (7.64e−06) 5.65e−03 (2.01e−04)	1.08e−03 (1.53e−04) 1.67e−06 (2.15e−06) 4.14e−04 (3.96e−05)	7.79e−05 (1.85e−05) 1.16e−06 (1.34e−06) 2.44e−05 (1.01e−05)	2.58e−05 (1.88e−05) 1.84e−06 (1.49e−06) 8.51e−06 (5.85e−06)
$\overline{\varepsilon}_0^{\gamma,r}$	1.00e+00 (2.47e−03) 2.18e−04 (4.63e−05) 6.80e−04 (6.43e−05)	1.00e+00 (5.17e−04) 1.07e−05 (9.20e−06) 8.53e−06 (3.48e−06)	1.00e+00 (8.77e−04) 6.08e−06 (2.64e−06) 6.85e−06 (2.96e−06)	1.00e+00 (1.73e−03) 5.94e−06 (2.85e−06) 6.99e−06 (6.45e−06)
$\overline{\tau}$	2.72e+02 5.14e+02 4.08e+02	1.03e+03 1.89e+03 1.35e+03	7.40e+03 1.39e+04 8.44e+03	2.47e+04 4.73e+04 2.64e+04
(c) $d = 50$.				
$\overline{\varepsilon}_0^{y,r}$	5.47e−03 (3.72e−04) 3.64e−03 (4.10e−04) 2.23e−05 (1.94e−05)	4.20e−04 (9.11e−05) 2.55e−04 (5.89e−05) 8.12e−06 (7.46e−06)	4.67e−05 (3.80e−05) 1.45e−05 (1.13e−05) 4.15e−06 (6.77e−06)	1.48e−05 (1.24e−05) 9.79e−06 (8.85e−06) 2.90e−06 (2.04e−06)
$\overline{\varepsilon}_0^{z,r}$	5.75e−02 (1.27e−03) 1.55e−03 (2.65e−04) 2.28e−02 (4.33e−04)	4.15e−03 (3.36e−04) 4.06e−05 (1.64e−05) 1.49e−03 (6.05e−05)	2.75e−04 (6.49e−05) 6.51e−06 (5.62e−06) 1.04e−04 (2.51e−05)	8.27e−05 (2.85e−05) 9.42e−06 (1.17e−05) 2.54e−05 (9.21e−06)
$\overline{\varepsilon}_0^{\gamma,r}$	1.00e+00 (2.75e−05) 2.24e−02 (1.83e−03) 6.17e−02 (1.84e−03)	1.00e+00 (2.34e−04) 1.25e−04 (8.82e−05) 1.33e−03 (2.13e−04)	1.00e+00 (2.85e−04) 6.59e−05 (7.35e−05) 1.19e−04 (1.13e−04)	1.00e+00 (1.69e−04) 8.93e−05 (1.23e−04) 6.56e−05 (7.64e−05)
$\overline{\tau}$	5.65e+02 2.75e+03 2.47e+03	2.83e+03 9.77e+03 7.77e+03	2.88e+04 7.32e+04 4.67e+04	1.12e+05 2.54e+05 1.47e+05

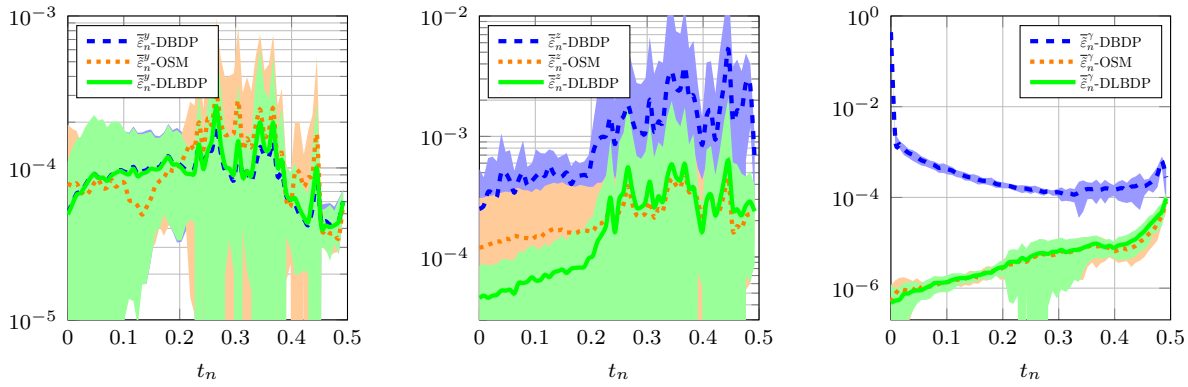
Table 2.1: Mean relative MSE values of (Y_0, Z_0, Γ_0) from DBDP, OSM and DLBDP schemes and their average runtimes in Example 2.5.1 for $d \in \{1, 10, 50\}$ and $N \in \{2, 8, 32, 64\}$. The STD of the relative MSE values at t_0 is given in the brackets.

Example 2.5.2. *The high-dimensional nonlinear BSDE for pricing European options with different interest rates reads*

$$\begin{cases} dX_t &= aX_t\,dt + bX_t\,dW_t, \\ X_0 &= x_0, \\ -dY_t &= \left(-R_1 Y_t - \frac{a-R_1}{b}\sum_{k=1}^d Z_t^k + (R_2 - R_1)\max\left(\frac{1}{b}\sum_{k=1}^d Z_t^k - Y_t, 0\right)\right) dt \\ &\quad -Z_t\,dW_t, \\ Y_T &= g(X_T), \end{cases}$$

where R_1 and R_2 are the interest rates for lending and borrowing, respectively, and $g(x)$ is the payoff function. Note that instead of solving the above BSDE directly, we solve the transformed BSDE in the ln-domain.

In the case of $d = 1$, we consider an European call option with $g(X_T) = (X_T - K)^+$. This setting agrees with the setting in [35] (Section 6.3.1), where it is noted that solving the above nonlinear BSDE is the same as solving the linear BSDE in Example 2.5.1 with $R = R_2$. This is a good example to compare the approximation of the process (Y, Z, Γ) in case of a nonlinear BSDE from all algorithms with the exact solution (given in (2.49)) on the entire discrete time domain. We set $T = 0.5$, $K = 100$, $x_0 = 100$, $a = 0.06$, $b = 0.2$, $R_1 = 0.04$ and $R_2 = 0.06$. In Figure 2.4, we display the mean MSE values for each process over discrete domain Δ using the testing sample and $N = 64$, where the STD of the MSE values is visualized in the shaded area. We see that



(a) Process Y. (b) Process Z. (c) Process Γ.

Figure 2.4: Mean MSE values of the processes (Y, Z, Γ) from DBDP, OSM and DLBDP schemes over the discrete time points $\{t_n\}_{n=0}^{N-1}$ using the testing sample in Example 2.5.2, for $d = 1$ and $N = 64$. The STD of MSE values is given in the shaded area.

our scheme outperforms the DBDP scheme in approximating the processes (Z, Γ) on the entire discrete domain Δ, similarly as in Example 2.5.1. Moreover, our scheme outperforms the OSM scheme in approximating the process Z in this example. In Table 2.2, we report the mean relative MSE values at t_0 for each process and the algorithm average runtime using $N \in \{2, 8, 32, 64\}$. The STD of the relative MSE values at t_0 is given in the brackets. The same conclusion can be drawn that the DLBDP scheme yields convergent results for $N \in \{2, 8, 32, 64\}$ for each process (whereas DBDP diverges for Γ_0) and outperforms the DBDP scheme. Similarly to Example 2.5.1, our scheme and the OSM scheme yield overall comparable approximations, except for $N = 64$, where our scheme performs better for the process Z. Additionally, the DLBDP scheme exhibits smaller runtimes compared to the OSM scheme.

Next, we test all schemes in the case of $d = 50$, using the payoff function

$$Y_T = \max\left(\max_{k=1,\dots,d}\left(X_T^k - K_1\right), 0\right) - 2\max\left(\max_{k=1,\dots,d}\left(X_T^k - K_2\right), 0\right),$$

where $K_1 = 120$ and $K_2 = 150$. Note that $a = 0.06$, $b = 0.2$ and $x_0 = 100\mathbf{1}_{50}$. The benchmark value is $Y_0 \doteq 17.9743$, which is computed using the multilevel Monte Carlo approach [23] with 7 Picard iterations and $Q = 10$ independent runs. For $N \in \{2, 8, 32, 64\}$, we show in Table 2.3 the approximation for Y_0 (the reference results for Z_0 are not available) from all algorithms and their average runtime. More precisely, we report the mean approximation of Y_0 defined as $\overline{Y}_0^{\Delta,\hat{\theta}} := \frac{1}{Q}\sum_{q=1}^{Q} Y_{0,q}^{\Delta,\hat{\theta}}$, the mean relative MSE and their STD given in the brackets. We

Metric	N = 2 DBDP OSM DLBDP	N = 8 DBDP OSM DLBDP	N = 32 DBDP OSM DLBDP	N = 64 DBDP OSM DLBDP
$\overline{\widehat{\varepsilon}}_0^{y,r}$	3.85e−06 (4.16e−06) 1.33e−05 (1.61e−05) 3.76e−06 (4.31e−06)	2.69e−06 (2.94e−06) 3.66e−06 (4.60e−06) 2.71e−06 (2.91e−06)	2.97e−06 (3.17e−06) 1.42e−06 (2.22e−06) 2.93e−06 (3.25e−06)	9.91e−07 (1.13e−06) 1.55e−06 (2.22e−06) 9.65e−07 (1.19e−06)
$\overline{\widehat{\varepsilon}}_0^{z,r}$	1.97e−04 (7.14e−05) 2.87e−06 (4.45e−06) 2.77e−05 (1.78e−05)	1.75e−05 (1.10e−05) 9.31e−07 (9.62e−07) 1.44e−06 (8.18e−07)	3.07e−06 (3.13e−06) 1.51e−06 (1.83e−06) 6.97e−07 (7.76e−07)	1.67e−06 (1.90e−06) 7.98e−07 (1.26e−06) 3.05e−07 (2.59e−07)
$\overline{\widehat{\varepsilon}}_0^{\gamma,r}$	1.09e+00 (1.67e−02) 9.53e−07 (1.48e−06) 3.25e−04 (3.80e−05)	1.01e+00 (3.16e−02) 1.51e−06 (1.33e−06) 2.51e−05 (1.05e−05)	9.96e−01 (1.44e−03) 1.52e−06 (1.52e−06) 1.24e−06 (1.26e−06)	9.96e−01 (1.43e−03) 1.12e−06 (1.52e−06) 1.08e−06 (1.61e−06)
$\overline{\tau}$	2.14e+02 3.11e+02 2.35e+02	6.63e+02 1.00e+03 7.37e+02	2.85e+03 4.57e+03 3.15e+03	6.85e+03 1.15e+04 7.42e+03

Table 2.2: Mean relative MSE values of (Y_0, Z_0, Γ_0) from DBDP, OSM and DLBDP schemes and their average runtimes in Example 2.5.2 for $d = 1$ and $N \in \{2, 8, 32, 64\}$. The STD of the relative MSE values at t_0 is given in the brackets.

Metric	N = 2 DBDP OSM DLBDP	N = 8 DBDP OSM DLBDP	N = 32 DBDP OSM DLBDP	N = 64 DBDP OSM DLBDP
Y_0 [23]	17.9743			
$\overline{Y}_0^{\Delta,\hat{\theta}}$	17.5602 (4.11e−01) 17.6537 (2.57e−01) 17.8329 (1.83e−01)	17.7981 (4.50e−01) 17.5056 (7.75e−01) 17.4669 (6.58e−01)	17.9276 (5.15e−01) 17.8351 (3.88e−01) 17.9714 (1.63e−01)	17.9112 (4.91e−01) 17.8865 (8.77e−02) 17.9117 (9.41e−02)
$\overline{\widehat{\varepsilon}}_0^{y,r}$	1.05e−03 (1.48e−03) 5.23e−04 (5.25e−04) 1.65e−04 (2.77e−04)	7.24e−04 (1.79e−03) 2.54e−03 (5.66e−03) 2.14e−03 (3.50e−03)	8.29e−04 (1.40e−03) 5.27e−04 (1.08e−03) 8.22e−05 (7.96e−05)	7.58e−04 (8.88e−04) 4.77e−05 (9.41e−05) 3.95e−05 (4.65e−05)
$\overline{\tau}$	5.54e+02 2.60e+03 2.36e+03	2.82e+03 9.74e+03 7.67e+03	2.87e+04 7.30e+04 4.67e+04	1.12e+05 2.55e+05 1.47e+05

Table 2.3: Mean approximation of Y_0, its mean relative MSE from DBDP, OSM and DLBDP schemes and their average runtimes in Example 2.5.2 for $d = 50$ and $N \in \{2, 8, 32, 64\}$. The STD of the approximations of Y_0 and its relative MSE values are given in the brackets.

observe that our scheme consistently provides higher accurate approximations of Y_0 for the 50-dimensional nonlinear BSDE in Example 2.5.2 compared to the other schemes, resulting in smaller relative MSE value. The DBDP scheme achieves the shortest computation time, while our scheme is faster than the OSM scheme. Note that the mean relative MSE can be further reduced by increasing the number of hidden neurons or layers provided that the optimization error is sufficiently small.

2.5.3 The Hamilton-Jacobi-Bellman equation

The next example is a Hamilton-Jacobi-Bellman (HJB) equation which admits a semi-explicit solution [22]. In finance, specifically portfolio optimization, solving the HJB equation provides insights into the optimal investment strategy that maximizes expected utility of the investors terminal wealth. Hence, the process Y is related to the wealth of the portfolio and the process Z the holding on each asset.

Example 2.5.3. *The high-dimensional HJB BSDE reads*

$$\begin{cases} dX_t &= b\, dW_t, \\ X_0 &= x_0, \\ -dY_t &= -\sum_{k=1}^{d} \left(\frac{Z_t^k}{b}\right)^2 dt - Z_t\, dW_t, \\ Y_T &= g(X_T). \end{cases}$$

This BSDE admits the semi-explicit solution as given in [22]

$$Y_t = u(t, X_t) = -\ln\left(\mathbb{E}\left[\exp\left(-g(X_t + b\,(W_T - W_t))\right)\right]\right).$$

The semi-explicit solution of (Z_t, Γ_t) is calculated using relations (2.10). Note that it is quite time consuming to approximate highly accurate pathwise reference solutions (Y_t, Z_t, Γ_t) for $t \in [0, T]$. Hence, we only calculate a reference solution at t_0. We set $T = 0.5$, $d = 50$, $X_0 = \mathbf{1}_d$, $b = \sqrt{0.2}$ and $g(x) = \ln\left(\frac{1}{2}\left(1 + |x|^2\right)\right)$. Using 10^7 Brownian motion samples and 50 independent runs, we calculate the mean approximations of (Y_0, Z_0, Γ_0) and use as reference values to test the accuracy of all schemes. In Table 2.4, we report the relative MSE values at t_0 for each process, the corresponding STD (given in the brackets) and the algorithm average runtime using $N \in \{2, 8, 32, 64\}$. Our scheme consistently outperforms the DBDP scheme in approximating (Z_0, Γ_0),

Metric	N = 2 DBDP OSM DLBDP	N = 8 DBDP OSM DLBDP	N = 32 DBDP OSM DLBDP	N = 64 DBDP OSM DLBDP
$\overline{\varepsilon}_0^{y,r}$	5.59e−07 (3.65e−07) 9.56e−07 (5.80e−07) 4.56e−07 (2.99e−07)	2.45e−07 (1.68e−07) 1.36e−07 (1.12e−07) 2.67e−07 (2.29e−07)	1.21e−07 (1.70e−07) 1.75e−07 (1.83e−07) 2.08e−07 (2.11e−07)	2.07e−07 (1.63e−07) 1.67e−07 (1.48e−07) 2.45e−07 (4.04e−07)
$\overline{\varepsilon}_0^{z,r}$	2.62e−04 (4.98e−05) 5.47e−05 (1.83e−05) 6.30e−05 (6.83e−06)	5.22e−04 (1.29e−04) 5.56e−05 (2.85e−05) 1.02e−04 (1.97e−05)	6.19e−04 (1.34e−04) 4.98e−05 (2.29e−05) 9.09e−05 (1.49e−05)	7.56e−04 (9.84e−05) 6.84e−05 (4.51e−05) 1.08e−04 (3.50e−05)
$\overline{\varepsilon}_0^{\gamma,r}$	9.35e−01 (1.02e−02) 5.20e−04 (2.81e−05) 2.99e−04 (1.03e−05)	9.65e−01 (1.28e−02) 4.34e−04 (2.60e−05) 8.28e−04 (2.41e−05)	8.18e−01 (1.15e−03) 5.05e−04 (2.72e−05) 1.14e−03 (3.43e−05)	8.41e−01 (4.51e−03) 6.32e−04 (4.10e−05) 1.50e−03 (4.43e−05)
$\overline{\tau}$	5.15e+02 2.53e+03 2.45e+03	2.38e+03 8.90e+03 7.52e+03	2.09e+04 5.64e+04 3.91e+04	7.80e+04 1.85e+05 1.13e+05

Table 2.4: Mean relative MSE values of (Y_0, Z_0, Γ_0) from DBDP, OSM and DLBDP schemes and their average runtimes in Example 2.5.3 for $d = 50$ and $N \in \{2, 8, 32, 64\}$. The STD of the relative MSE values at t_0 is given in the brackets.

particularly for Γ_0. The OSM scheme performs slightly better than the DLBDP scheme in this example, but has a higher computation time.

2.5.4 The Black-Scholes extended with local volatility

Our next example is taken from [87] in order to demonstrate the effectiveness of our scheme in case of a time dependent diffusion function. Consider an European call option as in Example 2.5.1, where each underlying asset follows a GBM with time dependent drift and diffusion.

Example 2.5.4. *The high-dimenisonal Black-Scholes BSDE with local volatility reads [87]*

$$\begin{cases} dX_t &= (a(t)-\delta)\,X_t\,dt + b(t)X_t\,dW_t, \\ X_0 &= x_0, \\ -dY_t &= -\left(RY_t + \sum_{k=1}^d \frac{a(t)-R+\delta}{b(t)} Z_t^k\right)\,dt - Z_t\,dW_t, \\ Y_T &= \left(\prod_{k=1}^d \left(X_T^k\right)^{c_k} - K\right)^+, \end{cases}$$

where for $a(t)$ and $b(t)$ we choose the following periodic functions

$$a(t) = a_0 + a_1 \sin\left(\frac{2\pi}{C_1}t\right) + a_2 \sin\left(\frac{2\pi}{C_2}t\right),$$
$$b(t) = b_0 + b_1 \sin\left(\frac{2\pi}{C_1}t\right) + b_2 \sin\left(\frac{2\pi}{C_2}t\right).$$

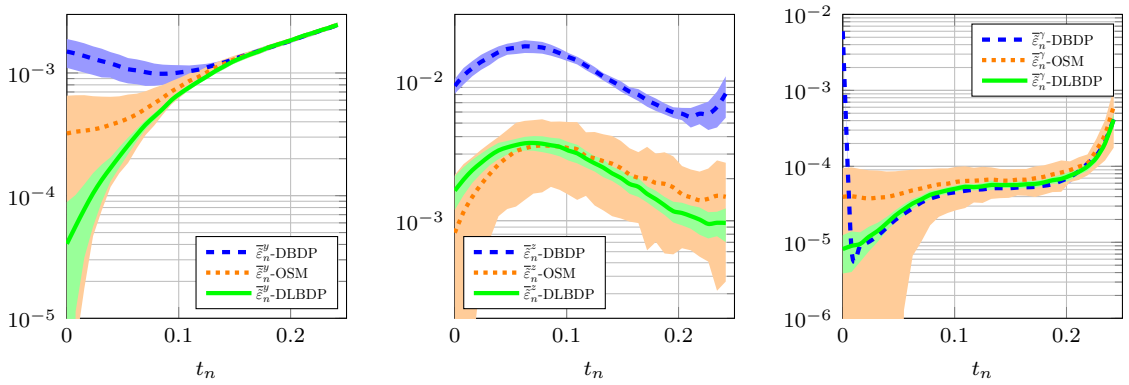
The exact solution of this local volatility model is given by the Black-Scholes formula with volatility parameter $\bar{b} = \sqrt{\frac{1}{T-t}\int_t^T b(s)^2 ds}$. More precisely, the exact solution is given by (2.49) with

$$\check{b} = \sum_{k=1}^d (\bar{b}c_k)^2, \quad \check{\delta} = \sum_{k=1}^d c_k\left(\delta_k + \frac{\bar{b}^2}{2}\right) - \frac{\check{b}^2}{2}, \quad Z_t^k = \frac{\partial u}{\partial x_k} b(t) X_t^k.$$

We apply the ln-transformation in this example, which is similar as in the case of Example 2.5.1. Moreover, we set $T = 0.25$, $d = 50$ and the other following parameter values

$$X_0 = 100, K = 100, R = 0.1, a_0 = 0.2, a_1 = 0.1, a_2 = 0.02,$$
$$c_k = \frac{1}{d}, \delta = 0, C_1 = 1, C_2 = 0.25, b_0 = 0.25, b_1 = 0.125, b_2 = 0.025.$$

Using $N = 32$, the mean MSE values for each process over discrete domain Δ are visualized in Figure 2.5 for the testing sample. The STD of the MSE values is displayed in the shaded area. Compared to previous examples, we notice significant improvements from our scheme not only



(a) Process Y. (b) Process Z. (c) Process Γ.

Figure 2.5: Mean MSE values of the processes (Y, Z, Γ) from DBDP, OSM and DLBDP schemes over the discrete time points $\{t_n\}_{n=0}^{N-1}$ using the testing sample in Example 2.5.4, for $d = 50$ and $N = 32$. The STD of MSE values is given in the shaded area.

in approximating the process Z but also the process Y compared to the DBDP scheme. In the case of the process Γ, such improvements are evident only near t_0. Interestingly, the DLBDP

scheme outperfrms the OSM scheme in this example for the processes Y and Γ, while providing comparable approximations of the process Z.

In Table 2.5, we report the mean relative MSE values at t_0 for each process from all schemes, using $N \in \{2, 8, 16, 32\}$. The corresponding STD is given in the brackets. The average runtime of the algorithms are also included. Our scheme gives overall the smallest relative MSE values. In

Metric	N = 2 DBDP OSM DLBDP	N = 8 DBDP OSM DLBDP	N = 16 DBDP OSM DLBDP	N = 32 DBDP OSM DLBDP
$\overline{\varepsilon}_0^{y,r}$	9.00e−03 (4.22e−04) 5.92e−03 (2.85e−04) 1.89e−04 (5.49e−05)	6.00e−03 (5.16e−04) 4.36e−04 (1.26e−04) 7.62e−06 (7.10e−06)	2.05e−03 (1.81e−04) 2.15e−04 (1.34e−04) 2.18e−05 (2.98e−05)	5.88e−04 (1.55e−04) 1.26e−04 (1.30e−04) 1.59e−05 (1.89e−05)
$\overline{\varepsilon}_0^{z,r}$	1.77e−01 (1.57e−03) 6.99e−02 (7.43e−04) 1.14e−01 (8.91e−04)	2.78e−02 (1.28e−03) 3.47e−03 (3.31e−04) 9.62e−03 (5.48e−04)	6.86e−03 (5.04e−04) 4.34e−04 (1.40e−04) 1.79e−03 (3.18e−04)	1.57e−03 (1.91e−04) 1.40e−04 (2.23e−04) 2.80e−04 (7.59e−05)
$\overline{\varepsilon}_0^{\gamma,r}$	1.00e+00 (5.28e−04) 3.92e−01 (3.52e−03) 4.72e−01 (3.49e−03)	1.00e+00 (6.55e−05) 2.99e−04 (1.83e−04) 1.78e−03 (6.86e−04)	1.00e+00 (2.25e−04) 3.19e−03 (1.36e−03) 8.91e−04 (4.64e−04)	1.00e+00 (9.98e−04) 6.63e−03 (9.26e−03) 1.36e−03 (7.11e−04)
$\overline{\tau}$	5.61e+02 2.71e+03 2.41e+03	2.79e+03 9.62e+03 7.78e+03	8.52e+03 2.47e+04 1.76e+04	2.80e+04 7.14e+04 4.59e+04

Table 2.5: Mean relative MSE values of (Y_0, Z_0, Γ_0) from DBDP, OSM and DLBDP schemes and their average runtimes in Example 2.5.4 for $d = 50$ and $N \in \{2, 8, 16, 32\}$. The STD of the relative MSE values at t_0 is given in the brackets.

this example, the improvement in approximating Y_0 is more evident than in previous examples.

2.5.5 BSDE with non-additive diffusion

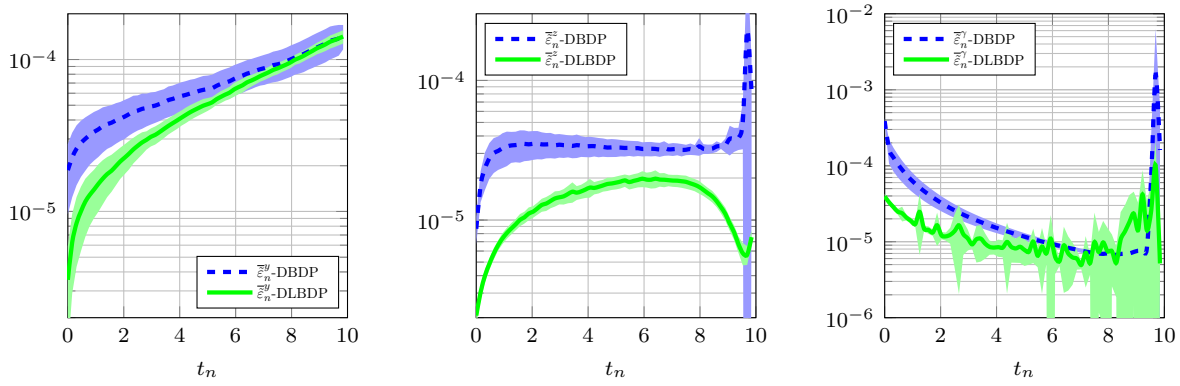
We now consider the non-symmetric example in [76] to demonstrate the performance of our scheme when the noise in the forward SDE is non-additive. The BSDE and its analytical solution are given in Example 1.5.5.

We choose $d = 50$, $T = 10$, $c_1 = 10d$, $c_2 = 1$ and $x_0 = \mathbf{1}_d$. In Figure 2.6, we display the mean MSE values for each process over discrete domain Δ using the testing sample and $N = 64$, where the STD of the MSE values is visualized in the shaded area. Note that for $N = 64$ the approximations from the OSM scheme are not available, because the scheduled scripts in the GPU nodes of PLEIADES cluster have a time limit of 3 days. Therefore, only the approximations from the DBDP and DLBDP schemes are displayed. Our scheme clearly outperforms the DBDP scheme in approximating each process during the entire discrete time domain.

In Table 2.6, we report the mean MSE values (due to small magnitude of the exact solution) at t_0 for each process and the algorithm average runtime using $N \in \{2, 8, 32, 64\}$. The STD of the relative MSE values at t_0 is given in the brackets. The same conclusions as can be drawn with our scheme when compared to the DBDP and OSM schemes even in the case of a more general diffusion term.

2.5.6 The Black-Scholes BSDE with correlated noise

Finally, we test all the schemes using an example with correlated noise. Specifically, we consider a European max call option within the Black-Scholes framework for a basket of stocks with



(a) Process Y. (b) Process Z. (c) Process Γ.

Figure 2.6: Mean MSE values of the processes (Y, Z, Γ) from DBDP and DLBDP schemes over the discrete time points $\{t_n\}_{n=0}^{N-1}$ using the testing sample in Example 1.5.5, for $d = 50$ and $N = 64$. The STD of MSE values is given in the shaded area.

Metric	N = 2 DBDP OSM DLBDP	N = 8 DBDP OSM DLBDP	N = 32 DBDP OSM DLBDP	N = 64 DBDP OSM DLBDP
$\overline{\widehat{\varepsilon}}_0^y$	1.03e−02 (1.81e−04) 1.56e−02 (6.28e−04) 1.23e−02 (8.73e−04)	1.27e−04 (5.35e−05) 7.01e−04 (2.19e−05) 5.07e−04 (8.91e−05)	8.79e−06 (8.09e−06) 5.03e−05 (1.20e−05) 4.46e−06 (4.97e−06)	1.87e−05 (9.08e−06) NA 3.55e−06 (2.37e−06)
$\overline{\widehat{\varepsilon}}_0^z$	1.69e−04 (6.23e−06) 6.32e−05 (5.03e−06) 1.21e−04 (2.07e−05)	7.31e−05 (8.25e−06) 1.81e−05 (7.95e−07) 1.31e−05 (2.70e−06)	1.60e−05 (2.71e−06) 3.09e−06 (3.84e−07) 2.60e−06 (6.37e−07)	8.66e−06 (1.94e−06) NA 2.07e−06 (3.29e−07)
$\overline{\widehat{\varepsilon}}_0^\gamma$	4.82e−04 (1.32e−05) 2.85e−04 (2.78e−05) 7.87e−04 (3.78e−05)	4.84e−04 (5.67e−05) 7.83e−05 (2.04e−06) 3.25e−04 (2.24e−05)	4.03e−04 (6.19e−05) 1.11e−05 (1.33e−06) 7.10e−05 (4.08e−06)	3.87e−04 (3.46e−05) NA 3.95e−05 (3.24e−06)
$\overline{\tau}$	5.79e+02 3.95e+03 3.16e+03	3.22e+03 1.41e+04 1.04e+04	3.36e+04 9.47e+04 5.92e+04	1.26e+05 NA 1.78e+05

Table 2.6: Mean MSE values of (Y_0, Z_0, Γ_0) from DBDP, OSM and DLBDP schemes and their average runtimes in Example 1.5.5 for $d = 50$ and $N \in \{2, 8, 32, 64\}$. The STD of the relative MSE values at t_0 is given in the brackets. The approximations for $N = 64$ from the OSM scheme are not available (NA) due to large computation time (more than 3 days).

distinct parameters (expected return, volatility, and correlation). The dynamics of the stocks are therefore given as

$$\begin{cases} dX_t^k &= (a_k - \delta_k)\, X_t^k\, dt + b_k X_t^k\, dW_t^k, \\ X_0^k &= x_0^k, \quad k = 1, \dots, d, \\ dW_t^k dW_t^j &= \rho_{k,j} dt, \quad k, j = 1, \dots, d, \quad \rho_{k,k} = 1. \end{cases}$$

By applying the Cholesky decomposition to the correlation matrix $(\rho_{k,j})_{k,j=1,\dots,d}$ and transforming the stock dynamics into the ln-domain, the corresponding BSDE is given as follows.

Example 2.5.5. *The high-dimenisonal BSDE for an European max call option in the ln-domain*

reads

$$\begin{cases} d\check{X}_t^k &= \left(a_k - \delta_k - \frac{1}{2}b_k^2\right)\,dt + b_k \sum_{j=1}^d \check{\rho}_{k,j}\, d\check{W}_t^{\,j}, \\ \check{X}_0^k &= \log\left(x_0^k\right), \quad k = 1, \ldots, d, \\ -dY_t &= -\left(RY_t + \sum_{k=1}^d \left(a_k - R + \delta_k\right)\varphi\left(Z_t^k\right)\right)\,dt - Z_t\, d\check{W}_t, \\ Y_T &= \left(\max_{k=1,\ldots,d} \exp\left(\check{X}_T\right) - K\right)^+, \end{cases}$$

where $\check{W}_t$ are d independent Brownian motions, $\check{\rho}_{k,j}$ represents the elements of the lower triangular matrix from Cholesky decomposition of $(\rho_{k,j})_{k,j=1,\ldots,d}$ and

$$\varphi(Z_t^k) = \begin{cases} \frac{Z_t^k}{b_k \check{\rho}_{k,k}}, & k = d, \\ \frac{Z_t^k - \sum_{j=k+1}^d b_j \check{\rho}_{j,k} F(Z_t^j)}{b_k \check{\rho}_{k,k}}, & k \neq d. \end{cases}$$

We set $d = 20$, $T = 0.5$, $R = 0.05$, $K = 100$ and $\delta_k = 0$ for $k = 1, \ldots, d$. The expected returns, volatilities, and correlation matrix are generated randomly. Specifically, $x_0^k \sim \mathcal{U}[K - 0.05K, K + 0.05K]$, $a^k \sim \mathcal{U}[0.01, 0.1]$ and $b^k \sim \mathcal{U}[0.05, 0.3]$ for $k = 1, \ldots, d$. The correlation matrix is sampled from $\mathcal{U}[-1, 1]$, ensuring that it is symmetric, positive definite, and has diagonal elements equal to one. To compute a benchmark value of Y_0, we use the Monte Carlo method (under the exact solution of the stock dynamics in the ln-domain) with 10^7 Brownian motion samples and 50 independent runs. Table 2.7 reports the mean approximation of Y_0, the mean relative MSE values, and the average runtime for all schemes using $N \in \{2, 8, 32, 64\}$. Standard deviations are provided in parentheses. Our method gives the best approximations of the benchmark option

Metric	N = 2 DBDP OSM DLBDP	N = 8 DBDP OSM DLBDP	N = 32 DBDP OSM DLBDP	N = 64 DBDP OSM DLBDP
Y_0 Monte Carlo	33.4819			
$\overline{Y}_0^{\Delta,\hat{\theta}}$	33.5478 (3.63e−02) 33.3931 (1.30e−01) 34.1471 (1.87e−01)	33.4500 (9.82e−02) 33.5005 (9.42e−02) 33.6575 (6.70e−02)	33.3932 (1.87e−01) 33.4690 (7.40e−02) 33.4367 (4.56e−02)	33.4664 (1.38e−01) 33.4449 (5.60e−02) 33.4542 (4.05e−02)
$\overline{\tilde{\varepsilon}}_0^{y,r}$	5.06e−06 (4.23e−06) 2.20e−05 (3.38e−05) 4.26e−04 (2.03e−04)	9.50e−06 (7.34e−06) 8.22e−06 (1.69e−05) 3.15e−05 (2.42e−05)	3.82e−05 (5.52e−05) 5.04e−06 (7.56e−06) 3.68e−06 (5.20e−06)	1.72e−05 (1.75e−05) 4.01e−06 (3.41e−06) 2.14e−06 (2.34e−06)
$\overline{\tau}$	3.34e+02 7.53e+02 5.80e+02	1.52e+03 3.09e+03 2.19e+03	1.35e+04 2.66e+04 1.57e+04	4.71e+04 9.35e+04 5.13e+04

Table 2.7: Mean approximation of Y_0, its mean relative MSE from DBDP, OSM and DLBDP schemes and their average runtimes in Example 2.5.5 for $d = 20$ and $N \in \{2, 8, 32, 64\}$. The STD of the approximations of Y_0 and its relative MSE values are given in the brackets.

value compared to the DBDP and OSM schemes, showcasing its robustness in high-dimensional, non-symmetric settings. The errors for (Z_0, Γ_0) are not reported due to the lack of highly accurate benchmarks. However, based on the previous examples, similar conclusions can be drawn for (Z_0, Γ_0).

2.6 Conclusions

In this chapter, we introduced a new class of schemes which utilizes the differential deep learning approach to solve high-dimensional nonlinear BSDEs. By applying Malliavin calculus, we

transform the BSDEs into a differential deep learning problem. This transformation results in a system of BSDEs that requires the estimation of the solution, its gradient, and the Hessian matrix, given by the triple of processes (Y, Z, Γ) in the BSDE system. To approximate this solution triple, we discretize the integrals within the system using the Euler-Maruyama method and parameterize their discrete version using DNNs. The DNN parameters are iteratively optimized backwardly at each time step by minimizing a differential learning type loss function, constructed as a weighted sum of the dynamics of the discretized BSDE system. An error analysis is conducted to demonstrate the convergence of the proposed algorithm. Our formulation provides additional information to the SGD method to give more accurate approximations compared to deep learning-based approaches, as our loss function includes not only the dynamics of the process Y but also Z. We show that the introduced differential deep learning-based approach can be used to other deep learning based schemes, e.g., in [86], but also others [22, 60]. The proficiency of our algorithm in terms of accuracy or computational efficiency is demonstrated through numerous numerical experiments involving pricing and hedging nonlinear options up to 50 dimensions. The proposed algorithm holds promise for applications in pricing and hedging financial derivatives in high-dimensional settings.

Building on the advancements in solving high-dimensional BSDEs using differential and deep learning techniques, the next chapter shifts focus to UQ for such schemes, addressing the challenges of evaluating uncertainties inherent in their implementation.

Chapter 3

UQ for Deep Learning BSDE Schemes

The aim of this chapter is to study UQ for a class of deep learning BSDE schemes. More precisely, we review the sources of uncertainty involved in the schemes and numerically study the impact of different sources. Usually, the STD of the approximate solutions obtained from multiple runs of the algorithm with different datasets is calculated to address the uncertainty. This approach is computationally quite expensive, especially for high-dimensional problems. Hence, we develop a UQ model that efficiently estimates the STD of the approximate solution using only a single run of the algorithm. The model also estimates the mean of the approximate solution, which can be leveraged to initialize the algorithm and improve the optimization process. We begin by introducing the sources of uncertainty in deep learning BSDE schemes, where we take as an example the pioneering DBSDE scheme. Moreover, the UQ model is introduced based on this scheme. Finally, we demonstrate through various numerical experiments the UQ model's reliability in estimating mean and STD not only for the DBSDE scheme, but also for other deep learning BSDE schemes. Additionally, we show multiple practical implications of using the UQ model. The chapter is based on our research conducted in [61]. It is worth highlighting that the work in this chapter is the first development of a UQ model for deep learning BSDE schemes. There remains significant work to be undertaken in this area.

3.1 Introduction

Since the predictions of models that use deep learning in decision-making processes are prone to noise and model errors [55], assessing the model's reliability before it can be used in practice is critical. An example of such decisions is the pricing and hedging of different contracts in finance. Companies may suffer from significant financial losses as a result of poor judgments. Thus, it is highly desirable to understand the uncertainties in deep learning BSDE schemes and develop methods to quantify them.

The pricing and hedging of option contracts is represented by the solution of the BSDE (1.1), see [62]. In case of a basket option, such BSDE is high-dimensional, as the dimensionality is related to the number of assets in the basket. When including market imperfections such as different interest rates for lending and borrowing [10], the BSDE becomes non-linear. As already explained in Chapter 1, the first deep learning-based scheme for solving high-dimensional non-linear BSDEs is the DBSDE scheme [22]. Due to the universal approximation capability [41, 19] of NNs, the objective function can be effectively optimized in practice. Therefore, the function

values of interest (the unknown solution and its gradient) are obtained quite accurately. Some convergence analysis of the DBSDE scheme have been studied, e.g., see [39, 56, 77] for the error analysis (utilizing universal approximation capability of NNs) and [98] for its gradient convergence (under a restrictive choice of NN setting). However, their analysis does not consider all the error or uncertainty sources (quite difficult to analyze), which in practical applications can't be disentangled, as we show in the numerical experiments. Thus, quantifying the uncertainty of the DBSDE scheme – and other differential or deep learning BSDE schemes – becomes crucial for practical applications.

As an example, we consider the pioneering DBSDE scheme to study UQ and introduce our UQ model. As presented in Section 1.3.1, the authors in the DBSDE scheme reformulate the BSDE as a stochastic control problem and use the Euler-Maruyama method to discretize the integrals. The unknown functions (solution at initial time and its gradient on the whole time domain) are estimated using DNNs. The parameters of DNNs are then optimized using the SGD algorithm. The DBSDE method incorporates various sources of uncertainty, including finite time discretization, restrictive choice of DNN specifications, the lack of convergence guarantees of the SGD algorithm, and finite sample size during stochastic optimization. It is crucial to identify and quantify these different sources of uncertainty in the DBSDE scheme for practical applications. Therefore, we review these sources of uncertainty and numerically demonstrate the impact of different sources. As mentioned before, our numerical experiments show that it is practically challenging to disentangle them, emphasizing the importance of quantifying uncertainty before using the scheme in practice. Usually, the STD of the approximate solutions obtained from multiple runs of the DBSDE algorithm with different datasets is calculated to account for the uncertainty in a given prediction, see [22]. This approach is computationally expensive, especially in high-dimensional cases due to the complexity of the DBSDE scheme itself. As the dimensionality increases, so does the number of network parameters that need to be optimized within each run of the scheme. While the ensemble size remains fixed regardless of the dimension, it adds a constant factor to the overall complexity of the method. Therefore, we develop a UQ model to estimate the STD of the approximate solution without requiring multiple runs. Several techniques have been proposed in the literature to quantify uncertainty, such as Monte Carlo Dropout [29], Monte Carlo DropConnnect [75], deep ensembles [70, 82], Flipout-based variational inference [99], Markov Chain Monte Carlo [68], and many others [40, 81, 80]. A review of these techniques can be found in [2]. To the best of our knowledge, there are no applications or developments of UQ models specifically for deep learning BSDE schemes. Hence, we develop a UQ model with the aim of addressing this gap.

Our UQ model is based on a commonly used approach for quantifying uncertainty in heteroscedastic nonlinear regression, see [5] for heteroscedastic least square type regression methods and [78, 70, 63] for heteroscedastic NN regression methods. We make the assumption that the residuals or errors of the DBSDE scheme follow a normal distribution with zero mean and the STD depending on the chosen parameter set of the discretized BSDE. This is a standard assumption in heteroscedastic regression. In our method, we use a DNN to learn two functions that estimate the mean and STD of the approximate solution. To train the DNN, we construct a dataset of independent and identically distributed (i.i.d) samples, which includes different parameter sets of the discretized BSDE and the corresponding approximate solutions obtained from the DBSDE algorithm. After generating a moderate number of samples, we optimize the network parameters by minimizing the negative ln-likelihood. Our UQ model provides an estimate of the STD of the approximate solution in a more computationally efficient manner compared to running multiple iterations of the algorithm per BSDE parameter set. Additionally, the estimated mean of the approximate solution from our model can be used to initialize the algorithm and improve the optimization process. Note that Bayesian (variational) methods could be used for

estimating the uncertainty of the DBSDE scheme, without requiring an additional little processing DNN as in our approach to model the uncertainty. However, integrating Bayesian methods into the DBSDE scheme require changes to its network architecture (e.g. the addition of dropout layers or even more drastic changes). Our aim is to perform UQ efficiently within the existing structure of the scheme, without introducing changes to it. Bayesian approaches typically come with increased computational cost. The deep ensemble method, which serves as our baseline, involves training the DBSDE scheme e.g. 10 times. The STD of these 10 approximations is then used as a measure of uncertainty. Although effective, this method is computationally expensive due to multiple trainings/DBSDE solves required. Our approach requires only one run of the DBSDE scheme (plus the pre-processing step to gather data to train the model) and provides STD estimates equivalent to those of 4 or more DBSDE runs in the deep ensemble approach, as demonstrated in our numerical experiments. Using our method in a continual learning manner, it provides a substantial reduction in computational cost. Note that our model is applicable not only to the DBSDE scheme but also to other deep learning and differential deep learning BSDE schemes for solving BSDEs. In addition to the DBSDE scheme, we apply our UQ model to the LaDBSDE scheme [60], which exhibits better convergence than the DBSDE scheme. Our numerical experiments demonstrate that the UQ model produces reliable estimates of the mean and STD of the approximate solution for both the DBSDE and LaDBSDE schemes. Moreover, we show multiple practical implications of using the UQ model. Firstly, the estimated STD captures multiple sources of uncertainty, demonstrating its effectiveness in quantifying the uncertainty. Secondly, the UQ model illustrates the improved performance of the LaDBSDE scheme in comparison to the DBSDE scheme based on the corresponding estimated STD values. Finally, our UQ model can be utilized to determine DNN hyperparameter values for which the respective scheme performs well, e.g. the number of discretization points.

The remainder of this chapter is organized as follows. In Section 3.2, we discuss the sources of uncertainty in the DBSDE scheme. The UQ model that estimates the STD of the approximate solution is provided in Section 3.3. In Section 3.4, we analyze the practical impact of different sources of uncertainty in the DBSDE scheme and demonstrate the effectiveness of our UQ model through numerical tests for both the DBSDE and LaDBSDE schemes. Finally, in Section 3.5, we conclude this chapter.

3.2 Uncertainty decomposition in the DBSDE scheme

In this section, we first reformulate the BSDE (1.1) as a stochastic control problem and then describe the DBSDE scheme for this problem. This forms the foundation for discussing afterward the sources of uncertainty in the DBSDE scheme. Similarly, one can identify the sources of uncertainty in other (differential) deep learning BSDE schemes.

3.2.1 A suitable stochastic control problem to represent the DBSDE scheme

The DBSDE scheme solves the associated stochastic control problem to the BSDE (1.1), which can be formulated as

$$\begin{aligned} &\inf_{Y_0\in\mathbb{L}^2_{\mathcal{F}_0}(\Omega;\mathbb{R}),\, Z\in\mathbb{H}^2\left([0,T]\times\Omega;\mathbb{R}^{1\times d}\right)} \mathbf{L}\left(Y_0, Z\right), \\ s.t.\quad & X_t = x_0 + \int_0^t a\left(s, X_s\right)\, ds + \int_0^t b\left(s, X_s\right)\, dW_s, \\ & Y_t = Y_0 - \int_0^t f\left(s, \mathbf{X}_s\right)\, ds + \int_0^t Z_s\, dW_s, \end{aligned} \tag{3.1}$$

for $t \in [0, T]$, where

$$\mathbf{L}\left(Y_0, Z\right) = \mathbb{E}\left[\left|g(X_T) - Y_T\right|^2\right].$$

The solution $\{(X_t, Y_t, Z_t)\}_{0\le t\le T}$ of (1.1) is a minimizer of (3.1) since the loss function attains zero when it is evaluated at the solution. In addition, the wellposedness of the BSDEs (under the usual regularity conditions [62]) ensures the existence and uniqueness of the minimizer. Due to (1.3), a function approximator for $u : \mathbb{R}^d \to \mathbb{R}$ and $\nabla_x u\, b : [0, T] \times \mathbb{R}^d \to \mathbb{R}^{1\times d}$ to approximate the unknown solution $Y_0 = u(t_0, x_0)$ and $Z_t = \nabla_x u\left(t, X_t\right) b\left(t, X_t\right)\ \forall\ t \in [0, T]$ is needed. Due to their approximation capability in high dimensions, the authors in [22] considered DNNs in the DBSDE scheme.

As already discussed, the first step to formulate (3.1) as a deep learing problem is to discretize the integrals (given by the Euler-Maruyama scheme in (1.5) and (1.6)). The discretized counterpart of (3.1) is given as

$$\begin{aligned} &\inf_{Y_0^\Delta\in\mathbb{L}^2_{\mathcal{F}_0}(\Omega;\mathbb{R}),\, Z^\Delta\in\mathbb{H}^{\Delta,2}\left(\{0,1,\ldots,N-1\}\times\Omega;\mathbb{R}^{1\times d}\right)} \mathbf{L}^\Delta\left(Y_0^\Delta, Z^\Delta\right), \\ s.t.\quad & X_0^\Delta = x_0, \\ & X_{n+1}^\Delta = X_n^\Delta + a\left(t_n, X_n^\Delta\right)\Delta t + b\left(t_n, X_n^\Delta\right)\Delta W_n, \\ & Y_{n+1}^\Delta = Y_n^\Delta - f\left(t_n, \mathbf{X}_n^\Delta, \right)\Delta t + Z_n^\Delta \Delta W_n, \end{aligned} \tag{3.2}$$

for $n = 0, 1, \ldots, N-1$, where

$$\mathbf{L}^\Delta\left(Y_0^\Delta, Z^\Delta\right) = \mathbb{E}\left[\left|g(X_N^\Delta) - Y_N^\Delta\right|^2\right],$$

and $\mathbb{H}^{\Delta,2}\left(\{0,1,\ldots,N\}\times\Omega;\mathbb{R}^{1\times d}\right)$ is the space of progressively measurable discrete stochastic processes $Z^\Delta : \{0,1,\ldots,N\}\times\Omega \to \mathbb{R}^{1\times d}$ s.t. $\mathbb{E}\left[\sum_{n=0}^N \left|Z_{t_n}^\Delta\right|^2 \Delta t\right] < \infty$. The authors in [22] considered DNNs, namely $\phi_0^y : \mathbb{R}^d \to \mathbb{R}$ and $\phi_n^z : \mathbb{R}^d \to \mathbb{R}^{1\times d}$ for $n = 0, 1, \ldots, N-1$ to estimate the solution of (3.2). The fully implementable DBSDE scheme for a sample of size B obtained from the Brownian motion increments ΔW_n is finally given as

$$\begin{aligned} &\min_{\theta\in\Theta} \tilde{\mathbf{L}}^\Delta\left(\phi_0^y\left(x_0;\theta_0^y\right), \phi^z\left(X^\Delta;\theta^z\right)\right), \\ s.t.\quad & X_0^\Delta = x_0, \quad Y_0^{\Delta,\theta} = \phi_0^y(x_0;\theta_0^y), \\ & X_{n+1,j}^\Delta = X_{n,j}^\Delta + a\left(t_n, X_{n,j}^\Delta\right)\Delta t + b\left(t_n, X_{n,j}^\Delta\right)\Delta W_{n,j}, \\ & Z_{n,j}^{\Delta,\theta} = \phi_n^z(X_{n,j}^\Delta;\theta_n^z), \\ & Y_{n+1,j}^{\Delta,\theta} = Y_{n,j}^{\Delta,\theta} - f\left(t_n, \mathbf{X}_{n,j}^{\Delta,\theta}\right)\Delta t + Z_{n,j}^{\Delta,\theta}\Delta W_{n,j}, \end{aligned} \tag{3.3}$$

for $n = 0, 1, \dots, N-1$, $j = 1, \dots, B$, $\Delta W_{n,j} \sim \mathcal{N}(\mathbf{0}_d, \Delta t\, \mathbf{I}_d)$ and

$$\tilde{\mathbf{L}}^{\Delta}\left(\phi_0^y(x_0;\theta_0^y), \phi^z(X^{\Delta};\theta^z)\right) = \frac{1}{B}\sum_{j=1}^{B}\left|g(X_{N,j}^{\Delta}) - Y_{N,j}^{\Delta,\theta}\right|^2.$$

As stated in Section 1.3.1 $Y_0^{\Delta,\theta} = \theta_0^y \in \mathbb{R}$ and $Z_0^{\Delta,\theta} = \theta_0^z \in \mathbb{R}^{1\times d}$ are considered as learnable parameters in [22], which are initialized by sampling from uniform distributions. Furthermore, $N-1$ DNNs are employed to calculate $Z_n^{\Delta,\theta}$ for $n = 1, 2, \dots, N-1$.

In the following sections, we introduce all the sources of uncertainty in the DBSDE scheme and propose a UQ model to estimate the uncertainty. It is important to emphasize that our approach is applicable not only to the DBSDE scheme but also to the LaDBSDE scheme (as we demonstrate in the numerical experiments) and potentially to other (differential) deep learning BSDE schemes as well.

3.2.2 Sources of uncertainty in the DBSDE scheme

For notational convenience, we consider to outline the multiple sources of uncertainty in the DBSDE scheme

- $Y_0^{\star} := \arg\min_{\mathbf{Y}\in\mathbb{L}^2_{\mathcal{F}_0}(\Omega;\mathbb{R})\times\mathbb{H}^2\left([0,T]\times\Omega;\mathbb{R}^{1\times d}\right)} \mathbf{L}(\mathbf{Y})$ the solution to (3.1), where $\mathbf{Y} = (Y_0, Z)$.
- $\mathbf{Y}^{\Delta,\star} := \arg\min_{\mathbf{Y}^{\Delta}\in\mathbb{L}^2_{\mathcal{F}_0}(\Omega;\mathbb{R})\times\mathbb{H}^{\Delta,2}\left(\{0,1,\dots,N-1\}\times\Omega;\mathbb{R}^{1\times d}\right)} \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\Delta}\right)$ the solution to (3.2), where $\mathbf{Y}^{\Delta} = \left(Y_0^{\Delta}, Z^{\Delta}\right)$.
- $\theta^{\star} := \arg\min_{\theta\in\Theta} \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\theta}\right)$ the optimal parameters of parameterized version of (3.2) using DNNs, where $\mathbf{Y}^{\theta} = \left(\phi_0^y(x_0;\theta_0^y), \phi^z(X^{\Delta};\theta^z)\right)$.
- $\tilde{\theta}^{\star} := \arg\min_{\theta\in\Theta} \tilde{\mathbf{L}}^{\Delta}\left(\mathbf{Y}^{\theta}\right)$ the optimal parameters in (3.3).
- $\mathcal{A} : \mathbb{N} \to \Theta$ the optimization algorithm and $\hat{\theta} = \mathcal{A}(B)$ an estimate for the sample of size B. Then $\mathbf{Y}^{\hat{\theta}} = \left(Y_0^{\Delta,\hat{\theta}}, Z^{\Delta,\hat{\theta}}\right)$ is an estimate of (3.3) after applying optimizer $\mathcal{A}$.
- $\varphi^{\Delta} : \mathbb{L}^2_{\mathcal{F}_0}(\Omega;\mathbb{R}) \times \mathbb{H}^{\Delta,2}\left(\{0,\dots,N-1\}\times\Omega;\mathbb{R}^{1\times d}\right) \to \mathbb{L}^2_{\mathcal{F}_0}(\Omega;\mathbb{R}) \times \mathbb{H}^2\left([0,T]\times\Omega;\mathbb{R}^{1\times d}\right)$ is an interpolation scheme. We choose φ^{Δ} s.t. $\mathbf{L}^{\Delta} - \mathbf{L}\circ\varphi^{\Delta} = 0$.

These uncertainties stem from factors such as finite time discretization, restrictive choice of DNN specifications, the lack of convergence guarantees of the SGD algorithm, and finite sample sizes. To systematically capture these sources, we employ a telescopic sum approach. More precisely, we decompose the total error of the DBSDE scheme – given by evaluating its final estimates $\mathbf{Y}^{\hat{\theta}}$ in (3.3) – by consecutively adding and removing the solutions defined above evaluated in (3.1), (3.2) or (3.3). This yields the following error decomposition of the DBSDE scheme

$$\begin{aligned}
\mathbf{L}\left(\varphi^{\Delta}\left(\mathbf{Y}^{\hat{\theta}}\right)\right) \leq\ & \mathbf{L}(\mathbf{Y}^{\star}) \\
&+ \mathbf{L}\left(\varphi^{\Delta}\left(\mathbf{Y}^{\Delta,\star}\right)\right) - \mathbf{L}\left(\mathbf{Y}^{\star}\right) && \text{(discretization error)} \\
&+ \left|\mathbf{L}^{\Delta}\left(\mathbf{Y}^{\Delta,\star}\right) - \mathbf{L}\left(\varphi^{\Delta}\left(\mathbf{Y}^{\Delta,\star}\right)\right)\right| \\
&+ \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\theta^{\star}}\right) - \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\Delta,\star}\right) && \text{(model/approximazion error)} \\
&+ \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\tilde{\theta}^{\star}}\right) - \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\theta^{\star}}\right) && \text{(estimation error)}
\end{aligned}$$

$$
\begin{aligned}
&+ \left|\tilde{\mathbf{L}}^{\Delta}\left(\mathbf{Y}^{\tilde{\theta}^\star}\right) - \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\tilde{\theta}^\star}\right)\right| && \text{(sampling error)}\\
&+ \tilde{\mathbf{L}}^{\Delta}\left(\mathbf{Y}^{\hat{\theta}}\right) - \tilde{\mathbf{L}}^{\Delta}\left(\mathbf{Y}^{\tilde{\theta}^\star}\right) && \text{(optimization error)}\\
&+ \left|\mathbf{L}^{\Delta}\left(\mathbf{Y}^{\hat{\theta}}\right) - \tilde{\mathbf{L}}^{\Delta}\left(\mathbf{Y}^{\hat{\theta}}\right)\right| && \text{(sampling error)}\\
&+ \left|\mathbf{L}\left(\varphi^{\Delta}\left(\mathbf{Y}^{\hat{\theta}}\right)\right) - \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\hat{\theta}}\right)\right|.
\end{aligned}
$$

where the interpolant φ^{Δ} is used to evaluate a discrete solution to the continuous formulation (3.1). Since $\mathbf{L}^{\Delta} - \mathbf{L} \circ \varphi^{\Delta} = 0$, we have

$$
\begin{aligned}
\mathbf{L}\left(\varphi^{\Delta}\left(\mathbf{Y}^{\hat{\theta}}\right)\right) &\leq \mathbf{L}\left(\varphi^{\Delta}\left(\mathbf{Y}^{\Delta,\star}\right)\right) && (3.4)\\
&+ \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\theta^\star}\right) - \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\Delta,\star}\right) && (3.5)\\
&+ \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\tilde{\theta}^\star}\right) - \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\theta^\star}\right) && (3.6)\\
&+ \left|\tilde{\mathbf{L}}^{\Delta}\left(\mathbf{Y}^{\tilde{\theta}^\star}\right) - \mathbf{L}^{\Delta}\left(\mathbf{Y}^{\tilde{\theta}^\star}\right)\right| && (3.7)\\
&+ \tilde{\mathbf{L}}^{\Delta}\left(\mathbf{Y}^{\hat{\theta}}\right) - \tilde{\mathbf{L}}^{\Delta}\left(\mathbf{Y}^{\tilde{\theta}^\star}\right) && (3.8)\\
&+ \left|\mathbf{L}^{\Delta}\left(\mathbf{Y}^{\hat{\theta}}\right) - \tilde{\mathbf{L}}^{\Delta}\left(\mathbf{Y}^{\hat{\theta}}\right)\right|. && (3.9)
\end{aligned}
$$

In the decomposition above, each term corresponds to a specific source of uncertainty. The naming convention we adopt here is motivated by [43]. The term (3.4) captures the discretization error associated with the Euler-Maruyama method. Note that in numerical methods approximating conditional expectations on spatial discretization, the discretization error is only a source of bias and not uncertainty. It is quantified by the difference between the BSDE solution and its discrete approximation. However, in the DBSDE scheme, it is a source of bias and uncertainty. For instance, a coarser discretization introduces more variance (the Brownian motion increments $\Delta W_n \sim \mathcal{N}(\mathbf{0}_d, \Delta t\, \mathbf{I}_d)$) but also systematic errors. When implementing the algorithm, a DNN architecture has to be chosen. Hence, (3.5) represents the model or approximation error [7, 51]. Since the scheme optimizes the empirical loss, (3.6) denotes the estimation error, as the empirical loss is only an estimate of the true loss. Selecting an SGD-type algorithm to optimize the DNN parameters introduces the optimization error, represented by (3.8). Finally, (3.7) and (3.9) correspond to the sampling errors. Note that the errors (3.4)-(3.9) may influence each other. For instance, a small model error (3.5) achieved by a complex DNN architecture could result in an increased optimization error (3.8), e.g. due to SGD getting stuck in poor local minima.

There are several studies that have conducted a theoretical analysis of the error terms associated with the DBSDE method. In [39], the authors provide a posteriori error estimation of the scheme for the general case of coupled FBSDEs (see [56] for FBSDEs with non-Lipschitz coefficients and [77] for fully-coupled drift coefficients). It is demonstrated that the error converges to zero given the universal approximation capability of NNs. Specifically, the error decreases as discretization error (3.4) and model error (3.5) diminish by increasing the number of time steps and neurons/layers. However, their analysis does not consider estimation error (3.6), sampling errors (3.7) and (3.9), and optimization error (3.8). The latter is quite difficult to analyze, and its current understanding is limited, mainly due to the non-convexity of the loss function in deep learning BSDE schemes. The authors in [98] provide a theoretical guarantee for the convergence of SGD iterations in the DBSDE scheme, but under a very restrictive choice of NN settings. On the other hand, the estimation and sampling errors can be negligible thanks to the soft memory limitation of a single SGD step (a new sample of a chosen batch size is generated after each optimization step). Hence, one can pass many realizations of the underlying Brownian motion

throughout the optimization cycle. While the magnitude of such error terms can be negligible, one is limited in practice to select a large number of discretization points and neurons/layers due to memory limitation and the high increase in computation time (the scheme uses $N-1$ DNNs). Moreover, while a high value of N can reduce discretization error, it can also result in a larger error due to the increased number of DNNs and the accumulation of propagated errors over time, as we demonstrate in the numerical section. Hence, although theoretically one expects uncertainty only from the optimization error, in practical implementation, the discretization and model errors also contribute to the uncertainty in the DBSDE scheme. The practical impact of these sources of uncertainty is discussed in the numerical section.

3.3 UQ model

In practice, it is challenging to disentangle the sources of uncertainty in the DBSDE scheme, as we demonstrate in the numerical experiments. Consequently, quantifying the uncertainty of the DBSDE scheme becomes crucial for practical applications. In this section, we develop a UQ model to estimate the uncertainty. After applying the DBSDE scheme, we obtain the random variables $Y_0^{\Delta,\hat{\theta}} \in \mathbb{R}$ and $Z_0^{\Delta,\hat{\theta}} \in \mathbb{R}^{1\times d}$ that approximate Y_0 and Z_0, respectively. To evaluate the quality of such approximations, one can use the expected squared error when the exact solution is known. This metric accounts for all the error sources in the DBSDE scheme and is calculated as

$$\epsilon^y := \sqrt{\mathbb{E}\left[\left(Y_0^{\Delta,\hat{\theta}} - Y_0\right)^2\right]} \in \mathbb{R}_+, \quad \epsilon^{z_k} := \sqrt{\mathbb{E}\left[\left(Z_0^{\Delta,\hat{\theta},k} - Z_0^k\right)^2\right]} \in \mathbb{R}_+,$$

for $k = 1, \dots, d$. However, (Y_0, Z_0) is usually unknown, the STD of the approximate solutions

$$\sigma^y := \sqrt{\mathbb{E}\left[\left(Y_0^{\Delta,\hat{\theta}} - \mu^y\right)^2\right]} \in \mathbb{R}_+, \quad \sigma^{z_k} := \sqrt{\mathbb{E}\left[\left(Z_0^{\Delta,\hat{\theta},k} - \mu^{z_k}\right)^2\right]} \in \mathbb{R}_+,$$

is often used, where $k = 1, \dots, d$, $\mu^y := \mathbb{E}\left[Y_0^{\Delta,\hat{\theta}}\right] \in \mathbb{R}$ and $\mu^{z_k} := \mathbb{E}\left[Z_0^{\Delta,\hat{\theta},k}\right] \in \mathbb{R}$. To compute the STD (and the expected squared error when the exact solution (Y_0, Z_0) is available), Q runs of the DBSDE algorithm must be done. The STD and the expected squared error are used as the benchmark in our experiments. To this end, we have the root mean squared error (RMSE) and the ensemble (biased sample) STD as

$$\tilde{\epsilon}^y := \sqrt{\frac{1}{Q}\sum_{q=1}^{Q}\left(Y_{0,q}^{\Delta,\hat{\theta}} - Y_0\right)^2}, \quad \tilde{\sigma}^y := \sqrt{\frac{1}{Q}\sum_{q=1}^{Q}\left(Y_{0,q}^{\Delta,\hat{\theta}} - \tilde{\mu}^y\right)^2},$$

for Y_0 and

$$\tilde{\epsilon}^{z_k} := \sqrt{\frac{1}{Q}\sum_{q=1}^{Q}\left(Z_{0,q}^{\Delta,\hat{\theta},k} - Z_0^k\right)^2}, \quad \tilde{\sigma}^{z_k} := \sqrt{\frac{1}{Q}\sum_{q=1}^{Q}\left(Z_{0,q}^{\Delta,\hat{\theta},k} - \tilde{\mu}^{z_k}\right)^2},$$

for Z_0, $k = 1, \dots, d$, where $\left(Y_{0,q}^{\Delta,\hat{\theta}}, Z_{0,q}^{\Delta,\hat{\theta}}\right)$ represents the approximated solutions from the q-th run of the algorithm, $\tilde{\mu}^y := \frac{1}{Q}\sum_{q=1}^{Q} Y_{0,q}^{\Delta,\hat{\theta}}$ and $\tilde{\mu}^{z_k} := \frac{1}{Q}\sum_{q=1}^{Q} Z_{0,q}^{\Delta,\hat{\theta},k}$ are the ensemble (sample) means of the approximate solution. Note that one can use the ensemble unbiased STD. However, the estimate of the STD from the UQ model is biased (as a maximum likelihood estimate). Therefore, for the purpose of comparison in numerical experiments, it is more convenient to use the ensemble

biased STD. Usually, $Q = 10$ is used, which is computationally expensive in high dimensions. Therefore, we propose a UQ model to estimate the STDs for $\left(Y_0^{\Delta,\hat{\theta}}, Z_0^{\Delta,\hat{\theta}}\right)$ by using only $Q = 1$ run of the algorithm. The model is based on an approach commonly used to quantify uncertainty in heteroscedastic nonlinear regression. We make the assumption that the errors of the DBSDE scheme follow a normal distribution with zero mean and the STD depending on the parameter set of the discretized BSDE (such as T, x_0, Δt, etc.). This assumption aligns with the standard practice in heteroscedastic regression. To train the UQ model, we construct a dataset of length M with i.i.d samples $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i\}_{i=1}^M$. Here, $\mathbf{x}_i \in \mathbb{R}^{\mathfrak{n}}$ represents $\mathfrak{n}$ parameters of the discretized BSDE, for example, $\mathbf{x}_i := (x_{0,i}, T_i, \Delta t_i)$, and $(\mathbf{y}_i, \mathbf{z}_i) := \left(Y_0^{\Delta,\hat{\theta}}(\mathbf{x}_i), Z_0^{\Delta,\hat{\theta}}(\mathbf{x}_i)\right) \in \mathbb{R} \times \mathbb{R}^{1\times d}$ are the approximations of $(Y_0(\mathbf{x}_i), Z_0(\mathbf{x}_i))$ obtained from the DBSDE scheme using BSDE parameter set $\mathbf{x}_i$. We use uniform distributions to select the BSDE parameter set $\mathbf{x}_i$

$$x_{0,i} \sim \mathcal{U}\left[x_0^{min}, x_0^{max}\right], \quad T_i \sim \mathcal{U}\left[T^{min}, T^{max}\right],$$

where (x_0^{min}, x_0^{max}) and (T^{min}, T^{max}) are the boundaries of the corresponding uniform distributions for x_0 and T. The training and test data for our UQ model stem from the same distribution. Consequently, our method cannot be expected to generalize to out-of-distribution data, which is a common challenge in deep learning in general. We propose adopting a continual learning approach, which is however beyond the scope of this dissertation and remains future work. For instance, an out-of-distribution detection method could be applied to the parameter space of the BSDE (e.g. kernel density estimation). If a given BSDE parameter set is identified as in-distribution, our current UQ model is applied. For out-of-distribution parameter sets, we fall back to the ensemble baseline approach. This allows us to incorporate new data into our UQ model, enabling it to be retrained and refined for future predictions. One can use the entire dataset $\mathcal{D}$ to build a learning algorithm that considers Y_0 and Z_0 as pairs (the BSDE solution at t_0 is the pair (Y_0, Z_0)). However, this approach may introduce increased complexity for the learning algorithm, mainly because the magnitudes of the solutions for $\mathbf{y}$ and $\mathbf{z}$ over different BSDE parameter sets $\mathbf{x}$ can differ significantly. Additionally, assumptions regarding their correlation might be necessary. Hence, we divide the dataset $\mathcal{D}$ into two datasets, $\mathcal{D}^y = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^M$ and $\mathcal{D}^z = \{\mathbf{x}_i, \mathbf{z}_i\}_{i=1}^M$, and develop two different learning algorithms. Given the input feature $\mathbf{x}$, we use one DNN to model the probabilistic predictive distribution $p^{y,\theta}(\mathbf{y}|\mathbf{x})$ and another for $p^{z,\theta}(\mathbf{z}|\mathbf{x})$. More precisely, we treat each observed value as a sample from a Gaussian distribution (multivariate Gaussian for Z_0), with the mean and STD as functions of the BSDE parameter set, namely

$$\mathbf{y}_i \sim \mathcal{N}(\mu^y(\mathbf{x}_i), (\sigma^y)^2(\mathbf{x}_i)), \quad \mathbf{z}_i \sim \mathcal{N}(\mu^z(\mathbf{x}_i), \mathbf{I}_d(\sigma^z)^2(\mathbf{x}_i)), \tag{3.10}$$

and allow the networks to calculate their final estimates as $(\hat{\mu}^y(\mathbf{x}_i), \hat{\sigma}^y(\mathbf{x}_i)) \in \mathbb{R} \times \mathbb{R}_+$ and $(\hat{\mu}^z(\mathbf{x}_i), \hat{\sigma}^z(\mathbf{x}_i)) \in \mathbb{R}^d \times \mathbb{R}_+^d$. These calculations are performed by minimizing the negative ln-likelihood

$$\tilde{\mathbf{L}}^y(\theta) = -\ln\left(p^{y,\theta}(\mathbf{y}|\mathbf{x})\right) = \frac{1}{M}\sum_{i=1}^{M}\left(\ln\left(\sigma^{y,\theta}(\mathbf{x}_i)\right) + \frac{1}{2}\frac{\left(\mathbf{y}_i - \mu^{y,\theta}(\mathbf{x}_i)\right)^2}{(\sigma^{y,\theta})^2(\mathbf{x}_i)}\right) + C,$$

for Y_0 and assuming that the covariance matrix of $Z_0^{\Delta,\hat{\theta}}$ is diagonal, then

$$\begin{aligned}\tilde{\mathbf{L}}^z(\theta) = -\ln\left(p^{z,\theta}(\mathbf{z}|\mathbf{x})\right) = \frac{1}{M}\sum_{i=1}^{M}\Bigg(&\sum_{k=1}^{d}\ln\left(\sigma^{z_k,\theta}(\mathbf{x}_i)\right) \\ &+ \frac{1}{2}\left(\mathbf{z}_i - \mu^{z,\theta}(\mathbf{x}_i)\right)^\top\left((\sigma^{z,\theta})^2(\mathbf{x}_i)\right)^{-1}\left(\mathbf{z}_i - \mu^{z,\theta}(\mathbf{x}_i)\right)\Bigg) + C,\end{aligned}$$

for Z_0, where $C > 0$ are constants and $\left(\mu^\theta, \sigma^\theta\right)$ are the parameterized version of unknown parameters (μ, σ) using DNNs with parameters θ. The algorithmic framework of our UQ model is provided in the following section.

3.4 Numerical results

In this section, we take the DBSDE scheme as an example to illustrate the impact of different sources of uncertainty in the scheme and apply our UQ model to both the DBSDE and LaDBSDE schemes. All the experiments below were conducted using PYTHON and TensorFlow on the PLEIADES cluster.

3.4.1 Experimental setup

The dataset $\mathcal{D}$ of length M for the UQ model is generated as follows:

- For sample i, $i = 1, \ldots, M$, generate a BSDE parameter set $\mathbf{x}_i$, e.g. $\mathbf{x}_i = (x_{0,i}, T_i, \Delta t_i)$ using uniform distributions.
- After specifying the hyperparameters of the DBSDE scheme, run the algorithm for the BSDE with parameter set $\mathbf{x}_i$ to calculate the approximations $(\mathbf{y}_i, \mathbf{z}_i) = \left(Y_0^{\Delta,\hat{\theta}}(\mathbf{x}_i), Z_0^{\Delta,\hat{\theta}}(\mathbf{x}_i)\right)$.
- Collect $(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i)$ and perform the same steps for each i.

A more detailed formulation is given in Algorithm 4. Note that in order to efficiently generate the dataset $\mathcal{D}$, we consider a straightforward parallelization of Algorithm 4, namely one core is used to simulate the DBSDE scheme for one BSDE parameter set. Hence, multiple cores provide a parallel generation of the dataset $\mathcal{D}$. The implementation of Algorithm 4 follows the hyperparameters considered in [22] for the DBSDE scheme, see also Section 1.5. We use Algorithm 5 and 6 to estimate the parameters in (3.10) for Y_0 and Z_0, respectively. For the implementation of Algorithm 5, we first split the dataset $\mathcal{D}^y$ into training, validation, and testing samples, whose sizes are denoted by M^{train}, M^{valid} and M^{test}, respectively. Note that the validation set is used to validate the performance of our UQ model when tuning its hyperparameters. The input layer of the DNN has $\mathfrak{n}$ neurons, and the output layer has 2 neurons. The first neuron in the output layer estimates the mean of the approximate solution $\mu^y(\mathbf{x})$, and the second neuron in the output layer estimates the STD of the approximate solution $\sigma^y(\mathbf{x})$, where the softplus activation function $\varrho(x) = \ln(1 + e^x) \in (0, \infty)$ is applied to obtain positive estimates. The ReLU activation function is used for the L^y hidden layers. Note that it is appropriate to choose $\eta^y > \mathfrak{q}^y$. The input data $\mathbf{x}$ is normalized based on the training data. We use the Adam optimizer with learning rate α^y, a batch size B^y, L2 regularization with parameter λ^y, and a specified number of epochs ep^y. The hyperparameters are set in a similar fashion for the implementation of Algorithm 6.

To visually and quantitatively compare our estimates of the mean and STD of the approximate solution to benchmark values, such as the exact solution, RMSE, ensemble mean, and ensemble STD, we consider both linear and nonlinear BSDEs with available analytical solutions. We take the Black-Scholes BSDE as a linear 1-dimensional example, which is used for pricing the European options. It is given in Section 2.5.1, Example 2.5.1, which is given in $d = 1$ as:

Algorithm 4: Algorithm generating dataset $\mathcal{D}$ for UQ model

Input: $\left(N, M, d, x_0^{min}, x_0^{max}, T^{min}, T^{max}, Y_0^{min}, Y_0^{max}\right)$ - problem related parameters
Input: (a, b, f, g) - functions of BSDE
Input: $(\alpha, L, \eta, \varrho, B, \mathfrak{K})$ - DNN hyperparameters in DBSDE scheme
Output: $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i\}_{i=1}^{M}$ - Dataset for UQ model
for $i = 1 : M$ **do**
  $x_{0,i} \sim \mathcal{U}\left[x_0^{min},\, x_0^{max}\right]$
  $T_i \sim \mathcal{U}\left[T^{min},\, T^{max}\right]$
  $\Delta t_i = \frac{T_i}{N}$
  $\mathbf{x}_i = (x_{0,i}, T_i, \Delta t_i)$
  for $n = 0 : N$ **do**
    $t_n = n\,\Delta t_i$
  end
  Initialize parameter set θ
  $Y_0^{\Delta,\hat{\theta}^0}(\mathbf{x}_i) = \hat{\theta}_0^{y,0} \sim \mathcal{U}\left[Y_0^{min},\, Y_0^{max}\right]$
  $Z_0^{\Delta,\hat{\theta}^0}(\mathbf{x}_i) = \hat{\theta}_0^{z,0} \sim \mathcal{U}\left[-\mathbf{1}_{1,d},\, \mathbf{1}_{1,d}\right]$ - $\mathbf{1}_{1,d} \in \mathbb{R}^{1\times d}$ vector of all ones
  $\left(\hat{\theta}_1^{z,0}, \ldots, \hat{\theta}_{N-1}^{z,0}\right)$- Xavier normal initializer [31]
  $\hat{\theta}^0 = \left(\hat{\theta}_0^{y,0}, \hat{\theta}_0^{z,0}, \hat{\theta}_1^{z,0}, \ldots, \hat{\theta}_{N-1}^{z,0}\right)$
  Optimization or training part
  for $\kappa = 1 : \mathfrak{K}$ **do**
    for $j = 1 : B$ **do**
      $X_{0,j}^{\Delta} = x_{0,i}$
      for $n = 0 : N-1$ **do**
        Euler-Maruyama for the forward SDE
        $\Delta W_{n,j} \sim \mathcal{N}(\mathbf{0}_d, \Delta t_i\, \mathbf{I}_d)$
        $X_{n+1,j}^{\Delta} = X_{n,j}^{\Delta} + a\left(t_n, X_{n,j}^{\Delta}\right)\Delta t_i + b\left(t_n, X_{n,j}^{\Delta}\right)\Delta W_{n,j}$
        Use DNN with (L, η, ϱ) **for** Z **and Euler-Maruyama for** Y
        $\mathfrak{q} = d$ - DNN output dimension; input dimension d
        if $n < N-1$ **then**
          $Z_{n+1,j}^{\Delta,\hat{\theta}^{\kappa-1}} = \phi_{n+1}^{z}\left(X_{n+1,j}^{\Delta}; \hat{\theta}_{n+1}^{z,\kappa-1}\right)$
          $Y_{n+1,j}^{\Delta,\hat{\theta}^{\kappa-1}} = Y_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} - f\left(t_n, \mathbf{X}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}}\right)\Delta t_i + Z_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}}\Delta W_{n,j}$
        else
          $Y_{n+1,j}^{\Delta,\hat{\theta}^{\kappa-1}} = Y_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}} - f\left(t_n, \mathbf{X}_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}}\right)\Delta t_i + Z_{n,j}^{\Delta,\hat{\theta}^{\kappa-1}}\Delta W_{n,j}$
        end
      end
    end
    $\tilde{\mathbf{L}}^{\Delta}\left(\hat{\theta}^{\kappa-1}\right) = \frac{1}{B}\sum_{j=1}^{B}\left|g(X_{N,j}^{\Delta}) - Y_{N,j}^{\Delta,\hat{\theta}^{\kappa-1}}\right|^2$
    Adam optimization step
    $\hat{\theta}^{\kappa}$ - trained parameters with Adam optimizer [64], learning rate α
  end
  $\hat{\theta} = \hat{\theta}^{\mathfrak{K}}$ - final estimated parameters after $\mathfrak{K}$ optimization steps
  $(\mathbf{y}_i, \mathbf{z}_i) = \left(Y_0^{\Delta,\hat{\theta}}(\mathbf{x}_i), Z_0^{\Delta,\hat{\theta}}(\mathbf{x}_i)\right)$
end

Algorithm 5: Algorithm estimating the parameters of (3.10) for Y_0

Input: $\left(M, \mathfrak{n}, M^{valid}, M^{test}, \mathcal{D}^y\right)$ - parameters and dataset for UQ model

Input: $(\alpha^y, L^y, \eta^y, \varrho^y, B^y, ep^y, \lambda^y)$ - DNN hyperparameters

Output: $\left(\mu^{y,\hat{\theta}}(\mathbf{x}), \sigma^{y,\hat{\theta}}(\mathbf{x})\right)$ - estimates of UQ model for Y_0

Split $\mathcal{D}^y$ into training, validation and testing samples

$M^{train} = M - M^{valid} - M^{test}$

$\left(\mathbf{x}^{train}, \mathbf{y}^{train}\right)$

$\left(\mathbf{x}^{valid}, \mathbf{y}^{valid}\right)$

$\left(\mathbf{x}^{test}, \mathbf{y}^{test}\right)$

Normalize input data x based on training statistics

$\left(\mathbf{x}^{train,nr}, \mathbf{x}^{valid,nr}, \mathbf{x}^{test,nr}\right)$

$\mathfrak{q}^y = 2$ - DNN output dimension; input dimension $\mathfrak{n}$

Initialize parameters θ

$\hat{\theta}^0$- Xavier normal initializer [31]

$\kappa = 0$

for *each epoch until* ep^y **do**

  for $j = 1 : \frac{M^{train}}{B^y}$ **do**

    $\kappa = \kappa + 1$

    Batch data

    $\mathbf{x}^{train,nr} = \left\{\mathbf{x}_i^{train,nr}\right\}_{i=(j-1)\,B^y+1}^{j\,B^y}$

    Use DNN with (L^y, η^y, ϱ^y) to estimate parameters in (3.10) for Y_0

    $\mu^{y,\hat{\theta}^{\kappa-1}}\left(\mathbf{x}^{train,nr}\right), \sigma^{y,\hat{\theta}^{\kappa-1}}\left(\mathbf{x}^{train,nr}\right) = \phi^y\left(\mathbf{x}^{train,nr}; \hat{\theta}^{\kappa-1}\right)$

    Calculate loss including L2 regularization

    $\tilde{\mathbf{L}}^y\left(\hat{\theta}^{\kappa-1}\right) =$

    $$\frac{1}{B^y}\sum_{i=(j-1)\,B^y+1}^{j\,B^y}\left(\ln\left(\sigma^{y,\hat{\theta}^{\kappa-1}}\left(\mathbf{x}_i^{train,nr}\right)\right) + \frac{1}{2}\frac{\left(\mathbf{y}_i^{train} - \mu^{y,\hat{\theta}^{\kappa-1}}\left(\mathbf{x}_i^{train,nr}\right)\right)^2}{\left(\sigma^{y,\hat{\theta}^{\kappa-1}}\right)^2\left(\mathbf{x}_i^{train,nr}\right)}\right) + \lambda^y\sum_{l=1}^{L^y}\left|\hat{\mathcal{W}}_l^{\kappa-1}\right|^2$$

    Adam optimization step

    $\hat{\theta}^\kappa$ - trained parameters with Adam optimizer [64], learning rate α^y

  end

end

$\hat{\theta} = \hat{\theta}^\kappa$ - final estimated parameters of DNN after ep^y epochs, each with $\frac{M^{train}}{B^y}$ number of batches

$\left(\mu^{y,\hat{\theta}}(\mathbf{x}), \sigma^{y,\hat{\theta}}(\mathbf{x})\right)$ - estimated parameters of (3.10) for Y_0, $\mathbf{x}$ training, validation or testing sample

Algorithm 6: Algorithm estimating the parameters of (3.10) for Z_0

Input: $\left(M, \mathfrak{n}, M^{valid}, M^{test}, \mathcal{D}^z\right)$ - parameters and dataset for UQ model

Input: $(\alpha^z, L^z, \eta^z, \varrho^z, B^z, ep^z, \lambda^z)$ - DNN hyperparameters

Output: $\left(\mu^{z,\hat{\theta}}(\mathbf{x}), \sigma^{z,\hat{\theta}}(\mathbf{x})\right)$ - estimates of UQ model for Z_0

Split $\mathcal{D}^z$ into training, validation and testing samples

$M^{train} = M - M^{valid} - M^{test}$

$\left(\mathbf{x}^{train}, \mathbf{z}^{train}\right)$

$\left(\mathbf{x}^{valid}, \mathbf{z}^{valid}\right)$

$\left(\mathbf{x}^{test}, \mathbf{z}^{test}\right)$

Normalize input data x based on training statistics

$\left(\mathbf{x}^{train,nr}, \mathbf{x}^{valid,nr}, \mathbf{x}^{test,nr}\right)$

$\mathfrak{q}^z = 2d$ - DNN output dimension; input dimension $\mathfrak{n}$

Initialize parameters θ

$\hat{\theta}^0$- Xavier normal initializer [31]

$\kappa = 0$

for *each epoch until* ep^z **do**

  for $j = 1 : \frac{M^{train}}{B^z}$ **do**

    $\kappa = \kappa + 1$

    Batch data

    $\mathbf{x}^{train,nr} = \left\{\mathbf{x}_i^{train,nr}\right\}_{i=(j-1)\,B^z+1}^{j\,B^z}$

    Use DNN with (L^z, η^z, ϱ^z) to estimate parameters in (3.10) for Z_0

    $\mu^{z,\hat{\theta}^{\kappa-1}}\left(\mathbf{x}^{train,nr}\right), \sigma^{z,\hat{\theta}^{\kappa-1}}\left(\mathbf{x}^{train,nr}\right) = \phi^z\left(\mathbf{x}^{train,nr}; \hat{\theta}\kappa{-}1\right)$

    Calculate loss including L2 regularization

$$\tilde{\mathbf{L}}^z\left(\hat{\theta}^{\kappa-1}\right) = \frac{1}{B^z}\sum_{i=(j-1)\,B^z+1}^{j\,B^z}\Bigg(\sum_{k=1}^{d}\ln\left(\sigma^{z_k,\hat{\theta}^{\kappa-1}}\left(\mathbf{x}_i^{train,nr}\right)\right) + \frac{1}{2}\left(\mathbf{z}_i^{train} - \mu^{z,\hat{\theta}^{\kappa-1}}\left(\mathbf{x}_i^{train,nr}\right)\right)^{\top}\left(\left(\sigma^{z,\hat{\theta}^{\kappa-1}}\right)^2\left(\mathbf{x}_i^{train,nr}\right)\right)^{-1} \left(\mathbf{z}_i^{train} - \mu^{z,\hat{\theta}^{\kappa-1}}\left(\mathbf{x}_i^{train,nr}\right)\right)\Bigg) + \lambda^z\sum_{l=1}^{L^z}\left|\hat{\mathcal{W}}_l^{\kappa-1}\right|^2$$

    Adam optimization step

    $\hat{\theta}^{\kappa}$ - trained parameters with Adam optimizer [64], learning rate α^z

  end

end

$\hat{\theta} = \hat{\theta}^{\kappa}$ - final estimated parameters of DNN after ep^z epochs, each with $\frac{M^{train}}{B^z}$ number of batches

$\left(\mu^{z,\hat{\theta}}(\mathbf{x}), \sigma^{z,\hat{\theta}}(\mathbf{x})\right)$ - estimated parameters of (3.10) for Z_0, $\mathbf{x}$ training, validation or testing sample

Example 3.4.1. *The Black-Scholes BSDE in $d = 1$ reads [105]*

$$\begin{cases} dS_t &= (a-\delta)\, S_t\, dt + bS_t\, dW_t, \quad S_0 = s_0, \\ -dY_t &= -\left(RY_t + (a - R + \delta)\, \frac{Z_t}{b}\right) dt - Z_t\, dW_t, \\ Y_T &= (S_T - K)_+ . \end{cases}$$

For the nonlinear case, we consider the nonlinear high-dimensional Burgers type BSDE.

Example 3.4.2. *The high-dimensional Burgers type BSDE reads [22]*

$$\begin{cases} dX_t &= b\, dW_t, \quad X_0 = 0, \\ -dY_t &= \left(\frac{b}{d} Y_t - \frac{2d+b^2}{2bd}\right) \left(\sum_{k=1}^{d} Z_t^k\right) dt - Z_t\, dW_t, \\ Y_T &= \frac{\exp\left(T + \frac{1}{d}\sum_{k=1}^{d} X_T^k\right)}{1+\exp\left(T+\frac{1}{d}\sum_{k=1}^{d} X_T^k\right)}, \end{cases}$$

where, $W_t = (W_t^1, W_t^2, \ldots, W_t^d)^\top$, $X_t = (X_t^1, X_t^2, \ldots, X_t^d)^\top$ and $Z_t = (Z_t^1, Z_t^2, \ldots, Z_t^d)$. The analytic solution is given by

$$\begin{cases} Y_t &= \frac{\exp\left(t + \frac{1}{d}\sum_{k=1}^{d} X_t^k\right)}{1+\exp\left(t+\frac{1}{d}\sum_{k=1}^{d} X_t^k\right)}, \\ Z_t &= \frac{b}{d} \frac{\exp\left(t + \frac{1}{d}\sum_{k=1}^{d} X_t^k\right)}{\left(1+\exp\left(t+\frac{1}{d}\sum_{k=1}^{d} X_t^k\right)\right)^2} \mathbf{1}_{1,d}, \end{cases}$$

where $\mathbf{1}_{1,d} \in \mathbb{R}^{1\times d}$ a row vector of ones. At t_0 we have that $(Y_0, Z_0) = \left(0.5, \frac{b}{4d}\mathbf{1}_{1,d}\right)$.

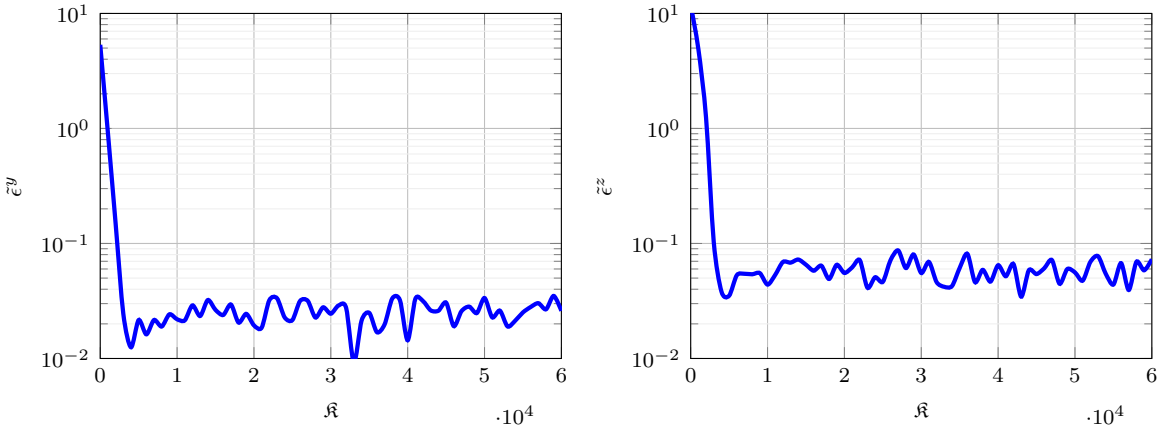
The following experiments are organized as follows. Firstly, we illustrate the impact of the sources of uncertainty in the DBSDE scheme for both examples by visualizing the effect of different errors on the uncertainty. Secondly, we assess the performance of our UQ model for the mean and STD of the approximate solution by comparing them with the benchmark values. Additionally, we determine the number of runs of the DBSDE algorithm for which the ensemble STD is comparable to the estimated STD. Furthermore, the computational cost of generating the training data for the UQ model is evaluated. To demonstrate the applicability of the UQ model to other deep learning-based BSDE schemes, we apply it to the LaDBSDE scheme. Finally, we show the practical implications of our UQ model.

3.4.2 The impact of the sources of uncertainty in the DBSDE scheme

To demonstrate the impact of the sources of uncertainty in the DBSDE scheme, we fix the parameter set of the BSDE and vary the hyperparameters of the DBSDE scheme that affect the corresponding source of uncertainty. Initially, we focus on the estimation and optimization errors. By using a high number of optimization steps $\mathfrak{K}$ and different learning rate approaches, the effect of these errors on the uncertainty of the DBSDE scheme is analyzed. Afterward, we investigate the impact of the model error by increasing the number of hidden neurons η. Lastly, the number of discretization points N is varied in order to study how the discretization error contributes to the uncertainty of the DBSDE scheme.

For the parameter values $T = 1, K = 100, S_0 = 100, a = 0.05, b = 0.2, R = 0.03$ and $\delta = 0$ in Example 3.4.1, the exact solution is $(Y_0, Z_0) = (9.4134, 11.9741)$. In Example 3.4.2, we chose $d = 50$ and fix $b = 25, T = 0.25$. The exact solution is $(Y_0, Z_0) = (0.5, 0.125\mathbf{1}_{1,50})$. For the DBSDE

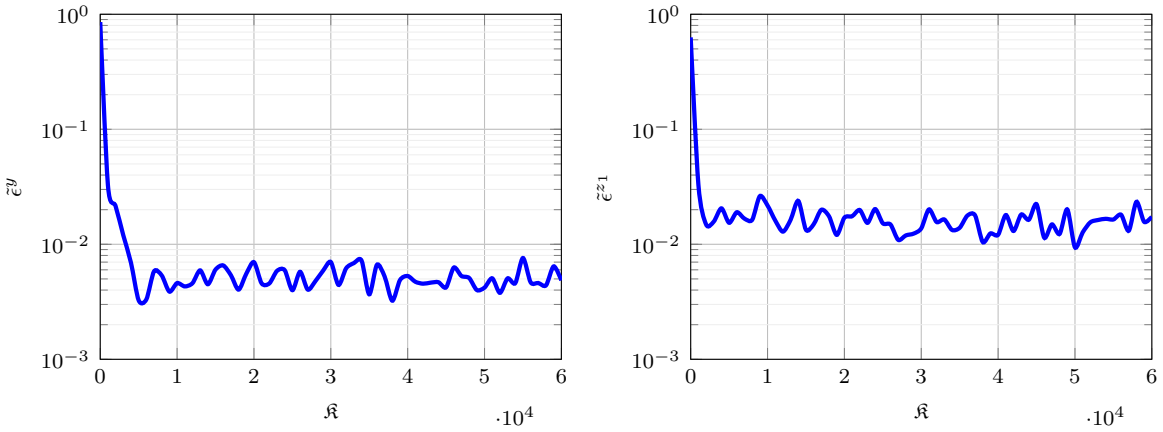
algorithm, we start with the following hyperparameters: a constant learning rate approach (C-LR) with $\mathfrak{K} = 60000$ optimization steps of the Adam algorithm, a learning rate $\alpha = 1\text{e}{-2}$, and a batch size of $B = 128$. To analyze the effect of estimation and optimization errors on the RMSE, we plot the RMSE values in Figures 3.1 and 3.2 for Examples 3.4.1 and 3.4.2, respectively, for increasing values of $\mathfrak{K}$. The RMSE values of Z_0 are plotted only for the first component in Example 3.4.2 as it is similar for the other components in our experiments. We use $N = 32$ discretization points and $Q = 10$ runs of the DBSDE algorithm. Note that each run of the DBSDE algorithm involves a different seed for generating the dataset and different initialization values of DNN parameters. As $\mathfrak{K}$ increases, the sum of estimation and optimization



(a) RMSE values for Y_0.

(b) RMSE values for Z_0.

Figure 3.1: RMSE values are plotted for Example 3.4.1 for increasing $\mathfrak{K}$, where $T = 1, K = 100, S_0 = 100, a = 0.05, b = 0.2, R = 0.03$ and $\delta = 0$.



(a) RMSE values for Y_0.

(b) RMSE values for Z_0^1.

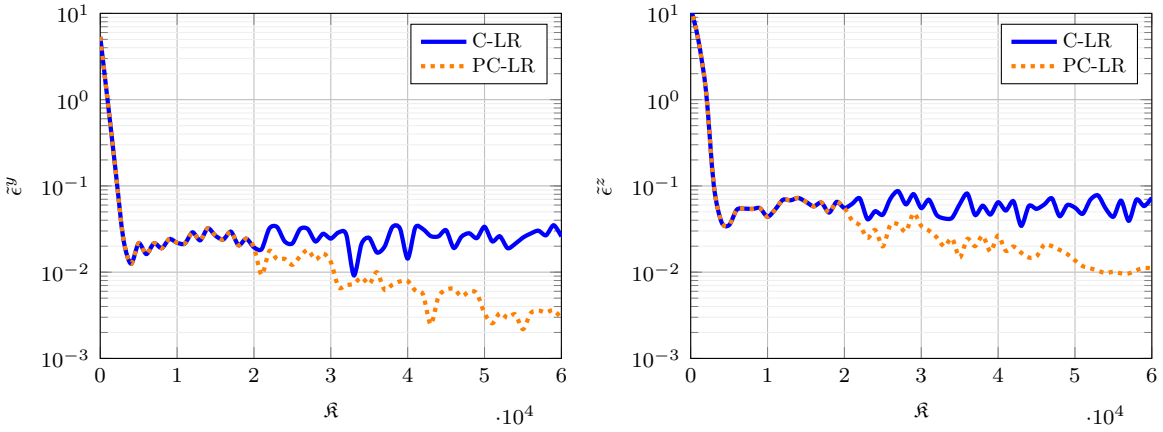
Figure 3.2: RMSE values are plotted for Example 3.4.2 while increasing $\mathfrak{K}$, where $T = 0.25$ and $b = 25$.

errors decreases for both examples as expected. This is because the DBSDE algorithm uses a new sample of size 128 after each optimization step, and the optimizer tends to perform better with more training data and optimization steps. This reduction in RMSE is evident until around 5000 optimization steps. However, for $\mathfrak{K} > 5000$, the RMSE plateaus due to other error sources, such as model and discretization errors, which are higher than the optimization error. Note that the RMSE values for Example 3.4.2 are lower than those for Example 3.4.1 because the exact

solution has a smaller value. To further reduce the optimization error, we use a PC-LR approach with the following learning rates

$$\alpha_\kappa = \begin{cases} 1\text{e}{-2}, & \text{for } 1 \leq \kappa \leq 20000, \\ 3\text{e}{-3}, & \text{for } 20000 < \kappa \leq 30000, \\ 1\text{e}{-3}, & \text{for } 30000 < \kappa \leq 40000, \\ 3\text{e}{-4}, & \text{for } 40000 < \kappa \leq 50000, \\ 1\text{e}{-4}, & \text{for } 50000 < \kappa \leq \mathfrak{K}. \end{cases}$$

We compare the RMSE values using C-LR and PC-LR in Figure 3.3 for Example 3.4.1. A similar behavior is observed for Example 3.4.2, see Figure A.1 in Appendix A.



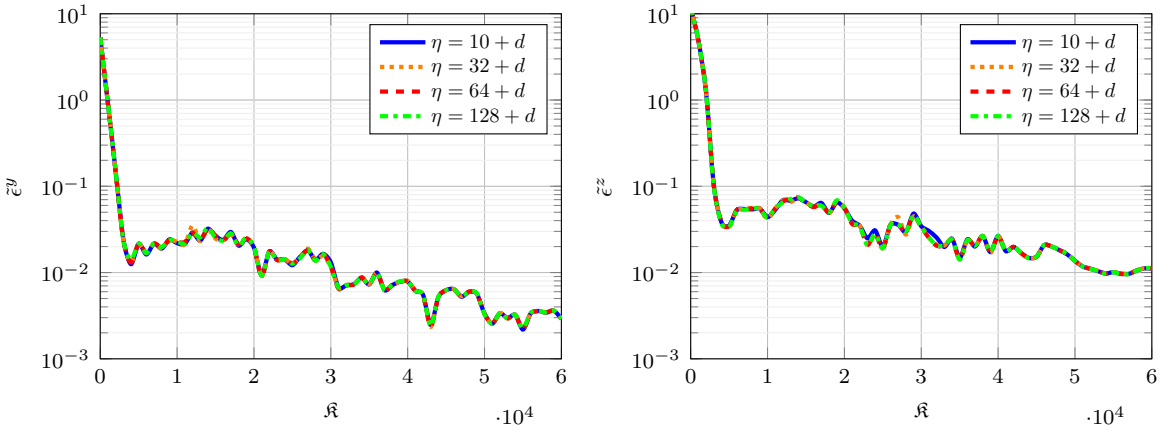
(a) RMSE values for Y_0.

(b) RMSE values for Z_0.

Figure 3.3: RMSE values are plotted for Example 3.4.1 using different learning rate approaches, where $T = 1, K = 100, S_0 = 100, a = 0.05, b = 0.2, R = 0.03$ and $\delta = 0$.

Next, we consider the model error. To try reduce the model error, one can increase the number of hidden neurons η or the number of hidden layers L. We report the RMSE values for $\eta \in \{10+d, 32+d, 64+d, 128+d\}$ in Figures 3.4 and 3.5 using PC-LR, for Examples 3.4.1 and 3.4.2, respectively. We observe that the RMSE does not decrease in both examples.

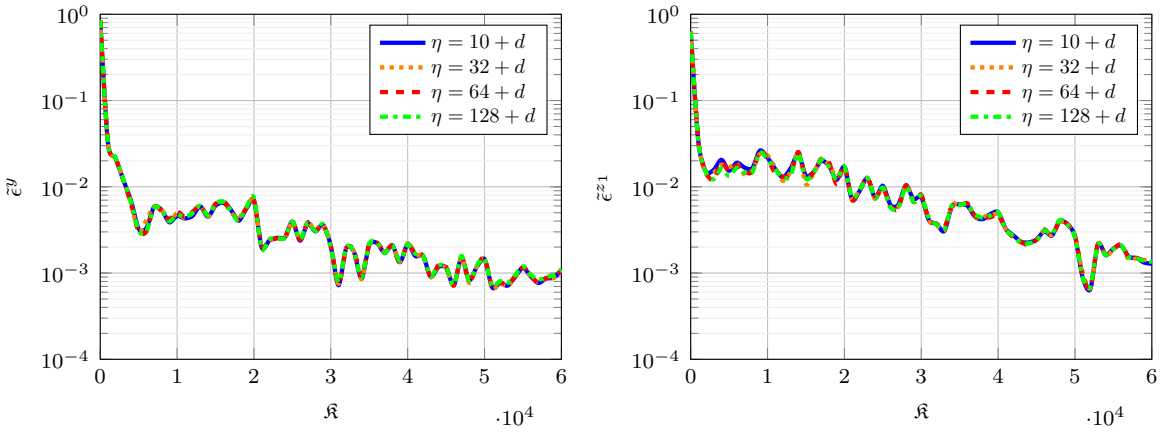
Finally, we consider the discretization error, which appears to be larger than the model error. To visualize this, we use the PC-LR approach while setting $\eta = 128 + d$, and plot the RMSE values in Figure 3.6 for $N \in \{2, 8, 32, 128, 256, 512, 1024\}$ in the case of Example 3.4.1. When considering the approximation of Y_0, the RMSE decreases with increasing N until $N = 32$, after which it does not reduce anymore. This phenomenon is even worse for the approximation of Z_0, as it starts to increase. It is worth noting that approximating Z is generally more challenging than approximating Y for BSDEs. While a higher value of N can reduce the discretization error, it can also lead to a larger error due to the increased number of DNNs and network parameters to be optimized. Additionally, the propagated errors over time in the DBSDE scheme become larger with a higher value of N. Such behaviour is more evident in Example 3.4.2, where the RMSE start to significantly increase for both Y_0 and Z_0 after $N = 32$, see Figure A.2 in Appendix A. This is due to the higher dimensionality, and the nonlinearity of the driver function in Example 3.4.2. To provide further clarity, we display the RMSE values and the absolute errors $\tilde{\epsilon}_q^y = |Y_{0,q}^{\Delta,\hat{\theta}} - Y_0|$ and $\tilde{\epsilon}_q^z = |Z_{0,q}^{\Delta,\hat{\theta}} - Z_0|$ from the q-th run for $N \in \{32, 1024\}$ in Figure 3.7 for Example 3.4.1. The variation of absolute errors from different runs around the corresponding RMSE values indicates that the increase in RMSE in Figure 3.6 for $N > 32$ is caused mainly



(a) RMSE values for Y_0.

(b) RMSE values for Z_0.

Figure 3.4: RMSE values are plotted for Example 3.4.1 using $\eta \in \{10+d, 32+d, 64+d, 128+d\}$, where $T = 1, K = 100, S_0 = 100, a = 0.05, b = 0.2, R = 0.03$ and $\delta = 0$.



(a) RMSE values for Y_0.

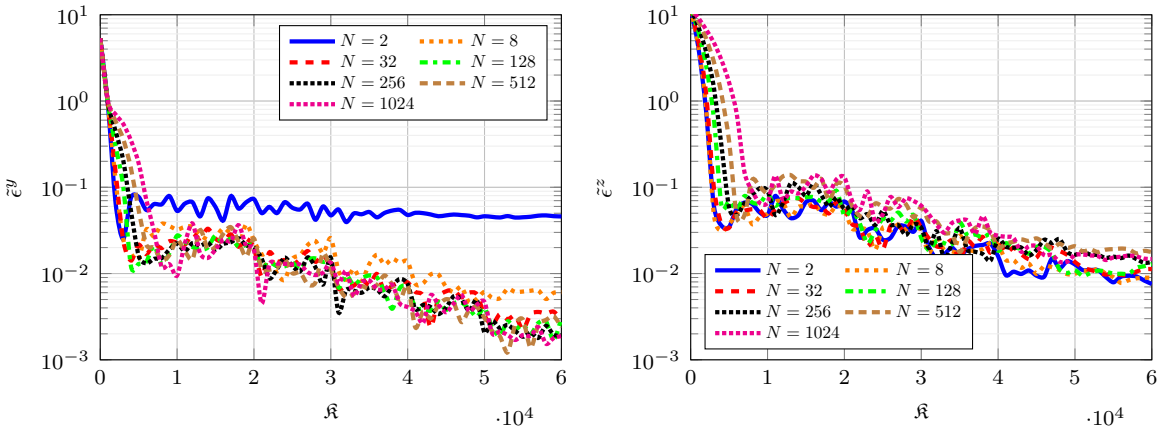
(b) RMSE values for Z_0^1.

Figure 3.5: RMSE values are plotted for Example 3.4.2 using $\eta \in \{10+d, 32+d, 64+d, 128+d\}$, where $T = 0.25$ and $b = 25$.

by the propagated errors (same is observed for Example 3.4.2, see Figure A.3 in Appendix A). Note that choosing $\eta > 128 + d$ does not reduce the RMSE. Hence, disentangling the sources of uncertainty is practically challenging, making it essential to quantify the uncertainty of the DBSDE scheme for practical applications.

3.4.3 Performance of the UQ model

In this section, we evaluate the quality of the mean and STD of the approximate solution estimated from our UQ model for both the DBSDE and LaDBSDE schemes. We start with the DBSDE scheme and consider Examples 3.4.1 and 3.4.2. In each example, the following structure is used. Firstly, the dataset for training and evaluating the UQ model is generated. To gain insights into the RMSE and ensemble STD, we visualize these values and determine an appropriate metric for assessing the accuracy of the ensemble STD in approximating the RMSE. Secondly, we focus on evaluating the accuracy of the estimated STD from the UQ model in approximating the RMSE, and compare it with the performance of the ensemble STD.



(a) RMSE values for Y_0.

(b) RMSE values for Z_0.

Figure 3.6: RMSE values are plotted for Example 3.4.1 using $N \in \{2, 8, 32, 128, 256, 512, 1024\}$, where $T = 1, K = 100, S_0 = 100, a = 0.05, b = 0.2, R = 0.03$ and $\delta = 0$.

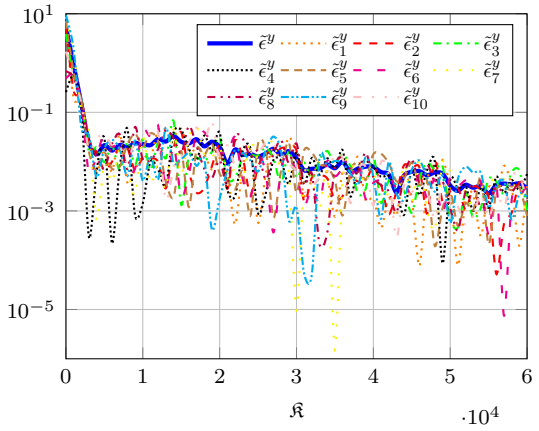
Additionally, we determine the number of DBSDE runs that the estimated STD achieves the same quality as the ensemble STD. The computational cost for training the UQ model is assessed in terms of the number of DBSDE runs needed to train it. Finally, we evaluate the accuracy of the estimated mean from the UQ model in approximating the exact solution, and compare it with the performance of the ensemble mean. Regarding the LaDBSDE scheme, only Example 3.4.2 is analyzed to show the applicability of the UQ model, following a similar structure as for the DBSDE scheme.

3.4.3.1 The DBSDE scheme for the Black-Scholes example

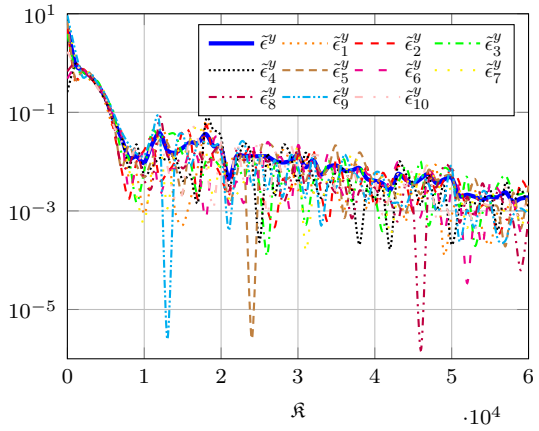
The dataset $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i\}_{i=1}^M$ for the UQ model is generated using Algorithm 4, where $\mathbf{x}_i$ includes the parameter set of the Black-Scholes BSDE, i.e. $\mathbf{x}_i = (a_i, b_i, S_{0,i}, R_i, \delta_i, K_i, T_i)$. Note that $\mathbf{y}_i = Y_0^{\Delta,\hat{\theta}}(\mathbf{x}_i)$ and $\mathbf{z}_i = Z_0^{\Delta,\hat{\theta}}(\mathbf{x}_i)$ are the approximated solutions given by the DBSDE algorithm for the parameter set $\mathbf{x}_i$ of the Black-Scholes BSDE. The parameter set $\mathbf{x}_i$ is generated using uniform distributions, where the bounds are chosen to account for all different scenarios of a European (call) option, namely in the money (ITM), at the money (ATM), and out of the money (OTM). To calculate $(\mathbf{y}_i, \mathbf{z}_i)$, we set $\mathfrak{K} = 30000$, $\alpha = 1\text{e}{-2}$ and $B = 128$ for the DBSDE algorithm. We generate 3 datasets where we treat T and N differently for each dataset. In dataset $\mathcal{D}_1$, we fix the values of T and N. In dataset $\mathcal{D}_2$, we vary T, but we keep the step size Δt fixed. In the last dataset $\mathcal{D}_3$, both T and Δt vary. This way, we analyze the performance of our UQ model in the case where the maturity value and discretization error vary. Additionally, we analyze the cases where the maturity value or the discretization error is fixed. The range of values for the parameters on each dataset is given in Table 3.1, where we choose $a = 0.05, \delta = 0$, and $K = 100$. We consider $M = 2560$ different BSDE parameter sets and run the DBSDE algorithm $Q = 10$

Dataset	Parameter range					
	b	S_0	R	T	N	Δt
$\mathcal{D}_1$	$[0.1, 0.4]$	$[K-20, K+20]$	$[0.001, 0.1]$	0.25	10	0.025
$\mathcal{D}_2$	$[0.1, 0.4]$	$[K-20, K+20]$	$[0.001, 0.1]$	$[\frac{1}{12}, 1]$	$\frac{T}{\Delta t}$	0.025
$\mathcal{D}_3$	$[0.1, 0.4]$	$[K-20, K+20]$	$[0.001, 0.1]$	$[\frac{1}{12}, 1]$	16	$\frac{T}{N}$

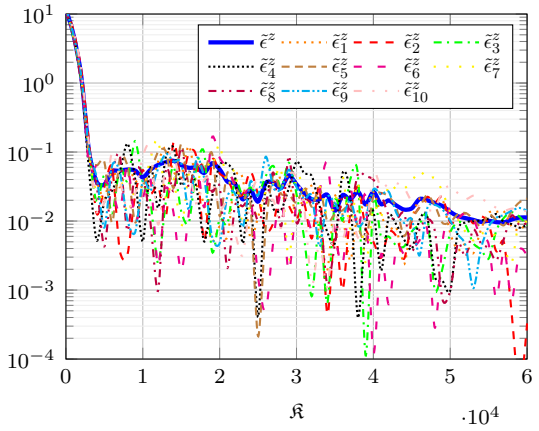
Table 3.1: Parameter range for Example 3.4.1.



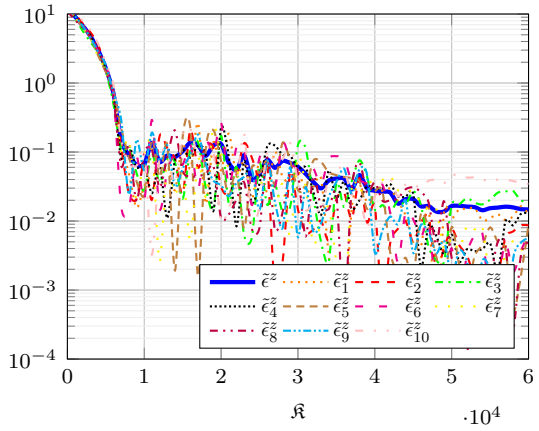
(a) RMSE values and the absolute error from each of $Q = 10$ DBSDE runs for Y_0 with $N = 32$.



(b) RMSE values and the absolute error from each of $Q = 10$ DBSDE runs for Y_0 with $N = 1024$.



(c) RMSE values and the absolute error from each of $Q = 10$ DBSDE runs for Z_0 with $N = 32$.



(d) RMSE values and the absolute error from each of $Q = 10$ DBSDE runs for Z_0 with $N = 1024$.

Figure 3.7: RMSE values and the absolute errors from each of $Q = 10$ DBSDE runs are plotted for Example 3.4.1 using $N \in \{32, 1024\}$, where $T = 1, K = 100, S_0 = 100, a = 0.05, b = 0.2, R = 0.03$ and $\delta = 0$.

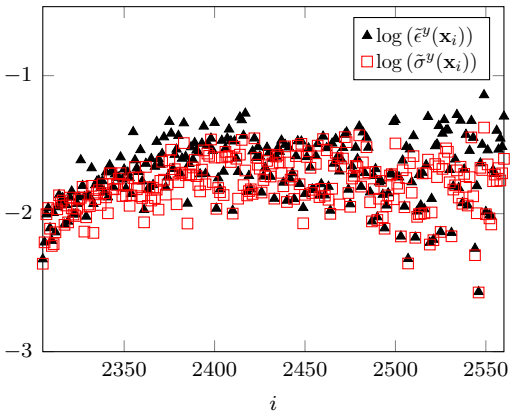
times for each BSDE parameter set. Thus, we construct a dataset of length $M = 2560$ for the UQ model by selecting the BSDE parameter set and the corresponding approximated solutions from the first run of the DBSDE algorithm. We also calculate benchmark values such as the RMSE, ensemble mean, and ensemble STD for each BSDE parameter set. Note that the datasets $\mathcal{D}_\mathfrak{j}$ are split into $\mathcal{D}^y_\mathfrak{j}$ and $\mathcal{D}^z_\mathfrak{j}$ for $\mathfrak{j} = 1, 2, 3$ in order to built the UQ model for Y_0 and Z_0. To gain insights into the benchmark values (the RMSE and ensemble STD), we display these values in Figure 3.8 and 3.9, sorted by the value of the exact solution (Y_0 and Z_0 respectively) for each dataset in the log-domain. Since the exact solution is different for each BSDE parameter set $\mathbf{x}$, we also display their relative estimates, where,

$$\tilde{\epsilon}^{y,r}(\mathbf{x}) := \frac{\tilde{\epsilon}^y(\mathbf{x})}{|Y_0(\mathbf{x})|}, \quad \tilde{\sigma}^{y,r}(\mathbf{x}) := \frac{\tilde{\sigma}^y(\mathbf{x})}{|\tilde{\mu}^y(\mathbf{x})|},$$

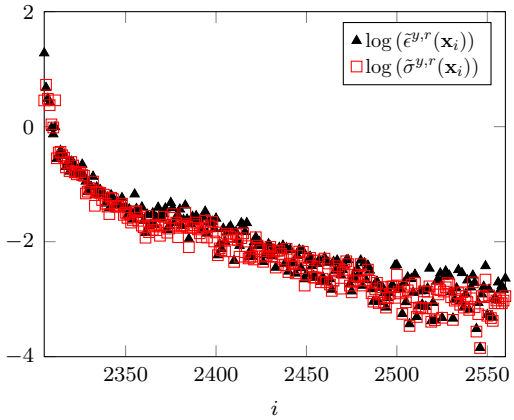
are the corresponding relative estimates for Y_0 and

$$\tilde{\epsilon}^{z,r}(\mathbf{x}) := \frac{\tilde{\epsilon}^z(\mathbf{x})}{|Z_0(\mathbf{x})|}, \quad \tilde{\sigma}^{z,r}(\mathbf{x}) := \frac{\tilde{\sigma}^z(\mathbf{x})}{|\tilde{\mu}^z(\mathbf{x})|},$$

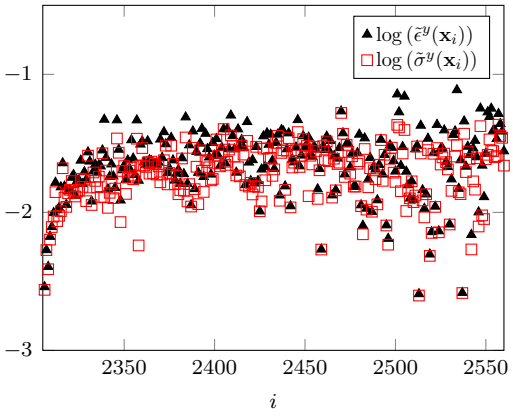
are the corresponding relative measures for Z_0. Note that we show only the last 256 out of 2560 values for better visualization. The RMSE, ensemble STD, and also their relative values



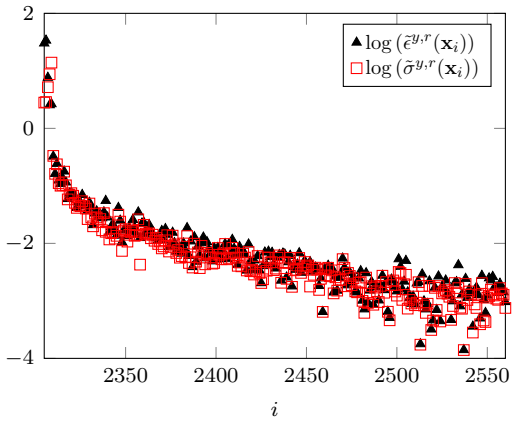
(a) Estimates for $\mathcal{D}_1^y$.



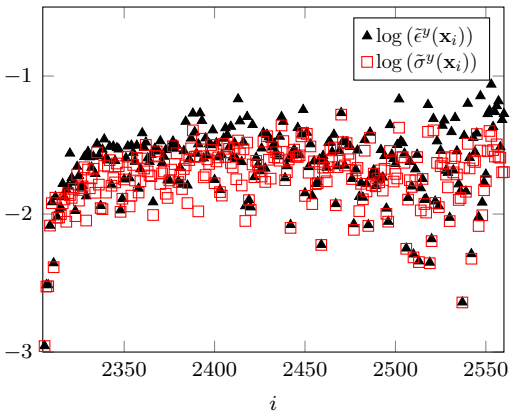
(b) Relative estimates for $\mathcal{D}_1^y$.



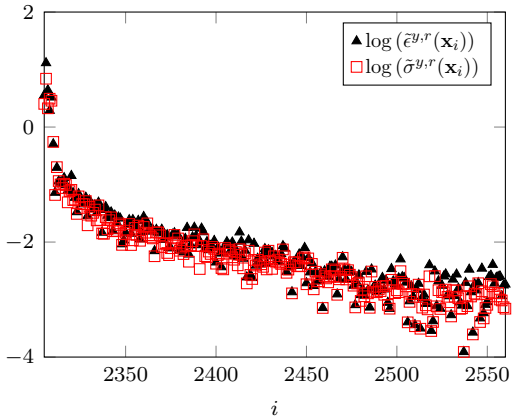
(c) Estimates for $\mathcal{D}_2^y$.



(d) Relative estimates for $\mathcal{D}_2^y$.



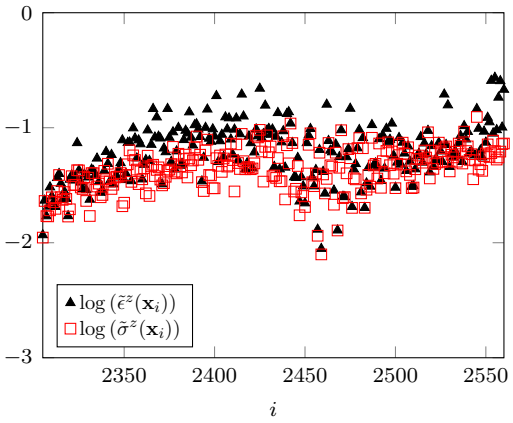
(e) Estimates for $\mathcal{D}_3^y$.



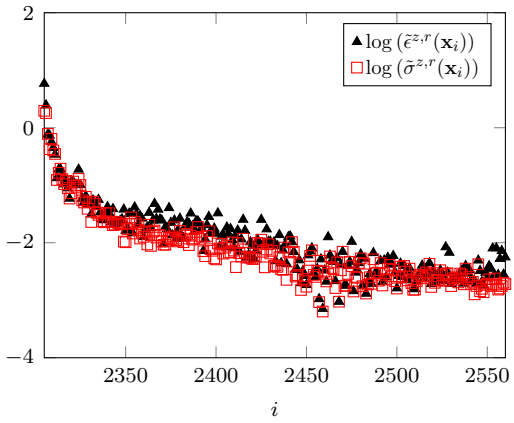
(f) Relative estimates for $\mathcal{D}_3^y$.

Figure 3.8: RMSE, the ensemble STD and their relative estimates for $\mathcal{D}_j^y$, $j = 1, 2, 3$ sorted by the value of the exact solution in Example 3.4.1.

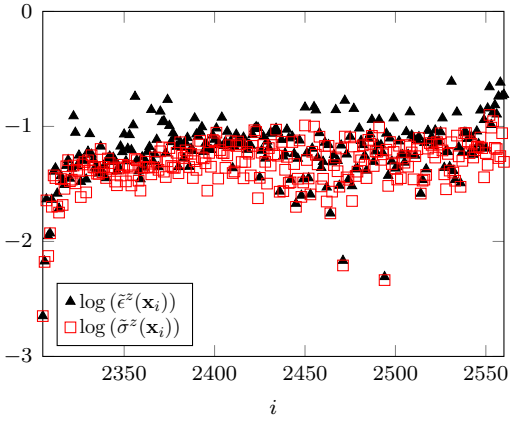
exhibit a strong positive correlation. Hence, we use the correlation to quantify the strength of the relationship between the RMSE and ensemble STD values. The correlation values in the log-domain are reported in Table 3.2. Note that all the following calculations for evaluating the estimated STD from the UQ model are conducted in the log-domain throughout this section.



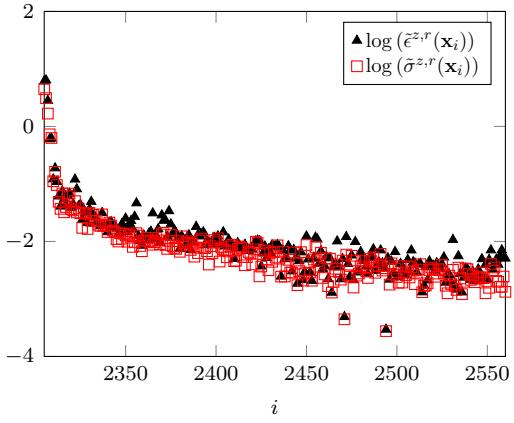
(a) Estimates for $\mathcal{D}_1^z$.



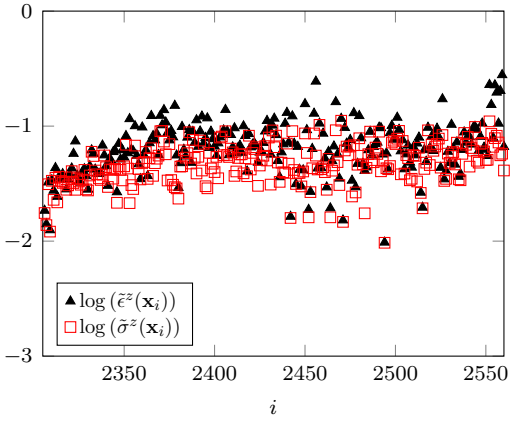
(b) Relative estimates for $\mathcal{D}_1^z$.



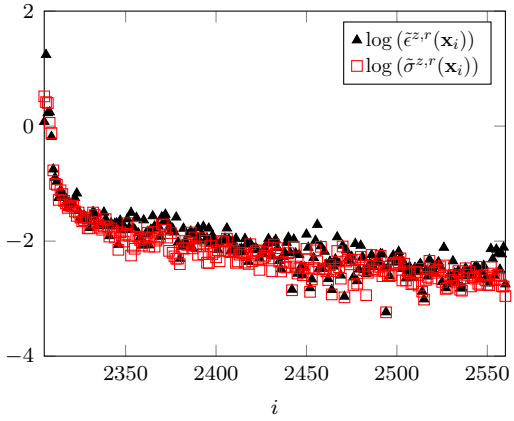
(c) Estimates for $\mathcal{D}_2^z$.



(d) Relative estimates for $\mathcal{D}_2^z$.



(e) Estimates for $\mathcal{D}_3^z$.



(f) Relative estimates for $\mathcal{D}_3^z$.

Figure 3.9: RMSE, the ensemble STD and their relative estimates for $\mathcal{D}_j^z$, $j = 1, 2, 3$ sorted by the value of the exact solution in Example 3.4.1.

We observe that the relative values provide a more reasonable measure than the absolute ones as the exact solution varies for each BSDE parameter set $\mathbf{x}$. Therefore, the relative values are used to evaluate the performance of the UQ model in estimating the STD of the approximate solution. Furthermore, we find that the cases with high relative RMSE values in Figure 3.8 and 3.9 correspond to deep OTM options, for which the DBSDE algorithm may produce negative estimates of the option price Y_0 or its delta hedging strategy Z_0, indicating divergence. The

Measure	Correlation	Dataset		
		$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$
Absolute measure for Y_0	$\rho\left(\log(\tilde{\epsilon}^y(\mathbf{x})), \log(\tilde{\sigma}^y(\mathbf{x}))\right)$	0.9260	0.9338	0.9284
Relative measure for Y_0	$\rho\left(\log(\tilde{\epsilon}^{y,r}(\mathbf{x})), \log(\tilde{\sigma}^{y,r}(\mathbf{x}))\right)$	0.9860	0.9835	0.9761
Absolute measure for Z_0	$\rho\left(\log(\tilde{\epsilon}^z(\mathbf{x})), \log(\tilde{\sigma}^z(\mathbf{x}))\right)$	0.8491	0.8397	0.8034
Relative measure for Z_0	$\rho\left(\log(\tilde{\epsilon}^{z,r}(\mathbf{x})), \log(\tilde{\sigma}^{z,r}(\mathbf{x}))\right)$	0.9750	0.9638	0.9554

Table 3.2: Correlation between the RMSE and ensemble STD values, and their relative values for $\mathcal{D}_\mathfrak{j}$, $\mathfrak{j} = 1, 2, 3$ in Example 3.4.1.

number of such cases is given in Table 3.3. To estimate the STD and mean of the approximate

Condition for divergence	Dataset		
	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$
$Y_0^{\Delta,\hat{\theta}}(\mathbf{x}) < 0$ or $Z_0^{\Delta,\hat{\theta}}(\mathbf{x}) < 0$	126	62	60

Table 3.3: Number of diverged cases of the DBSDE scheme for $\mathcal{D}_\mathfrak{j}$, $\mathfrak{j} = 1, 2, 3$ in Example 3.4.1.

solution for Y_0 and Z_0, we use Algorithm 5 and 6, respectively. Note that there is no evidence against the normality assumption for our UQ model, see Appendix B. The datasets $\mathcal{D}_\mathfrak{j}$ are split into training, validation, and testing samples, where we set $M^{valid} = M^{test} = 256$, and the rest for training, $M^{train} = M - M^{valid} - M^{test}$. To account for deviations in our results, we repeat each experiment 10 times, training 10 different UQ models, and provide the mean as well as the STD. For $\mathcal{D}_1$, $\mathfrak{n} = 3$, since $\mathbf{x}_i = (b_i, S_{0,i}, R_i)$ is the BSDE parameter set, $\mathfrak{n} = 5$ for $\mathcal{D}_2$ and $\mathcal{D}_3$ where (T, N) and $(T, \Delta t)$ are also varied, respectively. We use $\eta^y = \eta^z = 128$ and $L^y = L^z = 2$. The learning rate α, number of epochs ep, batch size B, and L2 regularization parameter λ are tuned. Based on the performance of the UQ model in the validation sample, the fine-tuned hyperparameters for Y_0 and Z_0 in dataset $\mathcal{D}_1$ are: $B = 128$, $\lambda = 3\text{e}{-2}$, and a PC-LR approach with $\alpha \in \{1\text{e}{-3}, 3\text{e}{-4}, 1\text{e}{-4}, 3\text{e}{-5}, 1\text{e}{-5}\}$ and $ep = 1000$ for $\alpha = 1\text{e}{-3}$ and $ep = 500$ for other learning rates. For the other datasets, the only change is in the PC-LR approach, where $ep = 100$ for $\alpha < 1\text{e}{-3}$. Note that all the following results are shown for the testing sample. To evaluate the quality of the estimated STD, we report in Table 3.4 the correlation between the relative RMSE and ensemble STD values $\rho\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\tilde{\sigma}^r(\mathbf{x}))\right)$, as well as the mean correlation between the relative RMSE and estimated STD values $\overline{\rho}\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\hat{\sigma}^r(\mathbf{x}))\right)$, where the averaging corresponds to the number of repetitions of our experiment computed by

$$\overline{\rho}\left(\log(\tilde{\epsilon}^{y,r}(\mathbf{x})), \log(\hat{\sigma}^{y,r}(\mathbf{x}))\right) := \frac{1}{10}\sum_{\mathfrak{i}=1}^{10} \rho\left(\log(\tilde{\epsilon}^{y,r}(\mathbf{x})), \log(\hat{\sigma}_\mathfrak{i}^{y,r}(\mathbf{x}))\right),$$

$$\overline{\rho}\left(\log(\tilde{\epsilon}^{z,r}(\mathbf{x})), \log(\hat{\sigma}^{z,r}(\mathbf{x}))\right) := \frac{1}{10}\sum_{\mathfrak{i}=1}^{10} \rho\left(\log(\tilde{\epsilon}^{z,r}(\mathbf{x})), \log(\hat{\sigma}_\mathfrak{i}^{z,r}(\mathbf{x}))\right).$$

Moreover, $\hat{\sigma}^{y,r}(\mathbf{x})$ and $\hat{\sigma}^{z,r}(\mathbf{x})$ represents the estimated relative STD values from 10 different trained UQ models for Y_0 and Z_0, respectively. The index $\mathfrak{i}$ corresponds to the values estimated from the $\mathfrak{i}$-th trained UQ model. The STD of the correlation is given in the brackets. Note that training 10 UQ models in this example, as well as in the following, is done only to test the robustness of our approach and to assess the statistical validity of our results. The correlation values for the relative ensemble STD and the mean correlation values for the relative estimated STD from our UQ model are very close for Y_0 and Z_0. This demonstrates that the relative estimated STD effectively approximates the relative ensemble STD. Moreover, we determine the number of

		Dataset		
UQ approach	Metric	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$
Ensemble for Y_0	$\rho\left(\log(\tilde{\epsilon}^{y,r}(\mathbf{x})), \log(\tilde{\sigma}^{y,r}(\mathbf{x}))\right)$	0.9893	0.9817	0.9898
UQ model for Y_0	$\overline{\rho}\left(\log(\tilde{\epsilon}^{y,r}(\mathbf{x})), \log(\hat{\sigma}^{y,r}(\mathbf{x}))\right)$	0.9852 (0.0006)	0.9604 (0.0049)	0.9575 (0.0074)
Ensemble for Z_0	$\rho\left(\log(\tilde{\epsilon}^{z,r}(\mathbf{x})), \log(\tilde{\sigma}^{z,r}(\mathbf{x}))\right)$	0.9669	0.9657	0.9581
UQ model for Z_0	$\overline{\rho}\left(\log(\tilde{\epsilon}^{z,r}(\mathbf{x})), \log(\hat{\sigma}^{z,r}(\mathbf{x}))\right)$	0.9473 (0.0021)	0.9151 (0.0108)	0.8934 (0.0137)

Table 3.4: Correlation between the relative RMSE and ensemble STD values, and the mean correlation between the relative RMSE and estimated STD values from the UQ model for $\mathcal{D}_\mathfrak{j}$, $\mathfrak{j} = 1, 2, 3$ using the testing sample in Example 3.4.1. The STD of the correlation is given in the brackets.

runs of the DBSDE algorithm for which the relative ensemble STD is approximately equal to the relative estimated STD. Therefore, we display in Figure 3.10 the correlation between the relative RMSE and ensemble STD values $\rho\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\tilde{\sigma}^r(\mathbf{x}))\right)$ for different DBSDE runs, the mean correlation between the relative RMSE and estimated STD values $\overline{\rho}\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\hat{\sigma}^r(\mathbf{x}))\right)$, and their intersection. The shaded area gives the STD of the correlation. We observe that the relative estimated STD from the UQ model is as good as the relative ensemble STD calculated from around $Q = 9$ runs of the DBDSE algorithm for Y_0 and $Q = 5$ for Z_0 in dataset $\mathcal{D}_1$. When the maturity T is also varied in dataset $\mathcal{D}_2$, the relative estimated STD is as good as the relative ensemble STD calculated from around $Q = 4$ runs of the DBDSE algorithm for Y_0 and Z_0. The same can be concluded for the last dataset $\mathcal{D}_3$ where the maturity T and the step size Δt are also varied. Hence, using a training dataset of length $M^{train} = 2048$ to train the UQ model, the relative estimated STD can perform as well as the relative ensemble STD of at least $Q = 4$ DBSDE runs.

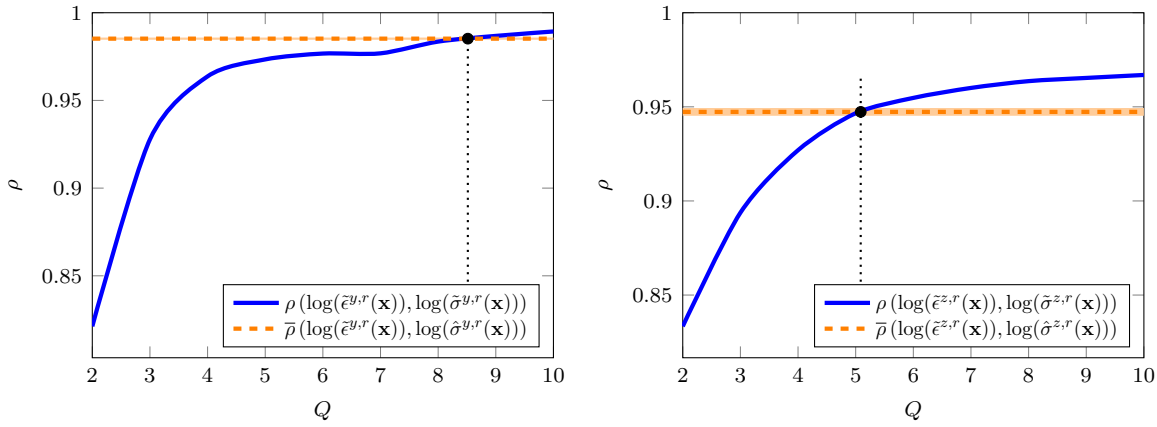
Using a larger M^{train}, the UQ model is expected to provide a better estimate of the relative STD. However, this pre-processing step to gather training data for the model increases the computational cost as the number of DBSDE runs needed to train the UQ model is equal to M^{train}. To show such a trade-off, we display in Figure 3.11 the mean correlation between the relative RMSE and estimated STD values $\overline{\rho}\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\hat{\sigma}^r(\mathbf{x}))\right)$ while increasing the number of DBSDE runs to train the UQ model from 10% to 100% of M^{train}. We observe that the UQ model can give a good estimate of the relative STD even when trained with 1024 DBSDE runs.

Our UQ model does not only estimates the STD of the approximate solution but also its mean. We use the RMSE to measure the quality of the estimated mean compared to the ensemble mean and the expected (exact) solution, which are presented in Table 3.5. The mean RMSE ($\overline{RMSE}$) corresponds to the number of repetitions of our experiment computed by

$$\overline{RMSE}\left(Y_0(\mathbf{x}), \hat{\mu}^y(\mathbf{x})\right) := \frac{1}{10}\sum_{\mathfrak{i}=1}^{10} RMSE\left(Y_0(\mathbf{x}), \hat{\mu}^y_\mathfrak{i}(\mathbf{x})\right),$$

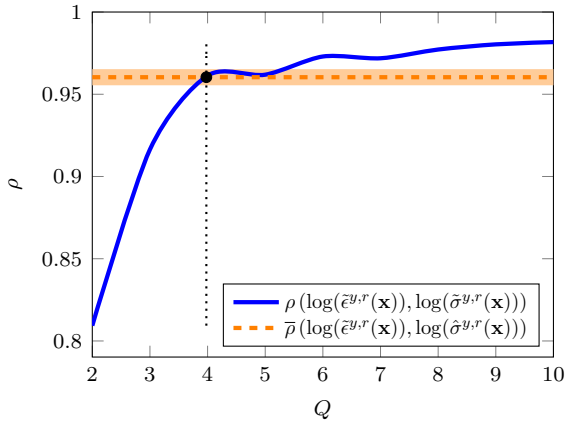
$$\overline{RMSE}\left(Z_0(\mathbf{x}), \hat{\mu}^z(\mathbf{x})\right) := \frac{1}{10}\sum_{\mathfrak{i}=1}^{10} RMSE\left(Z_0(\mathbf{x}), \hat{\mu}^z_\mathfrak{i}(\mathbf{x})\right),$$

where $\hat{\mu}^y(\mathbf{x})$ and $\hat{\mu}^z(\mathbf{x})$ represents the estimated mean values from 10 different trained UQ models for Y_0 and Z_0, respectively. The index $\mathfrak{i}$ corresponds to the values estimated from the $\mathfrak{i}$-th trained UQ model. The STD of the RMSE is given in the brackets. The RMSE between the exact solution and ensemble mean values $RMSE\left(Y_0(\mathbf{x}), \tilde{\mu}^y(\mathbf{x})\right)$, and the mean RMSE between the exact solution and estimated mean values $\overline{RMSE}\left(Y_0(\mathbf{x}), \hat{\mu}^y(\mathbf{x})\right)$ are very close. The same can be concluded for Z_0. Hence, the estimated mean given by our UQ model can be used as

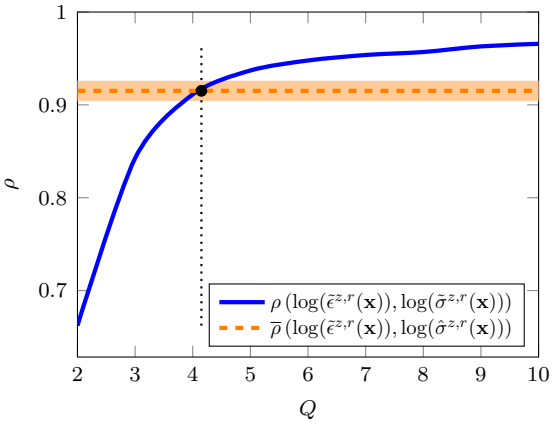


(a) Dataset $\mathcal{D}_1^y$.

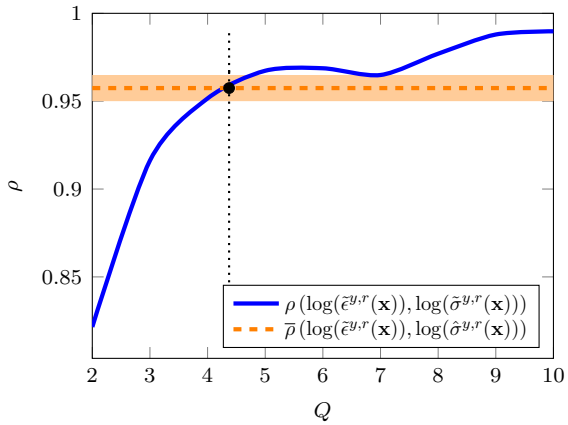
(b) Dataset $\mathcal{D}_1^z$.



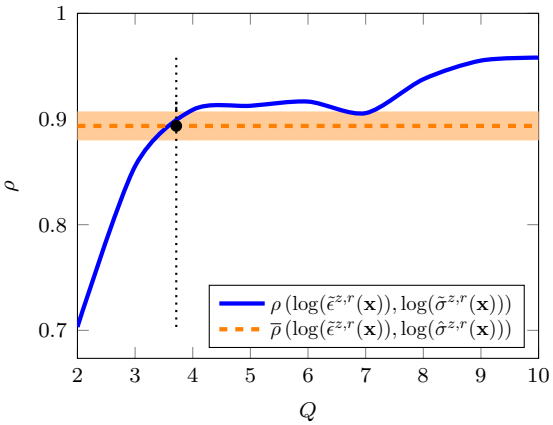
(c) Dataset $\mathcal{D}_2^y$.



(d) Dataset $\mathcal{D}_2^z$.



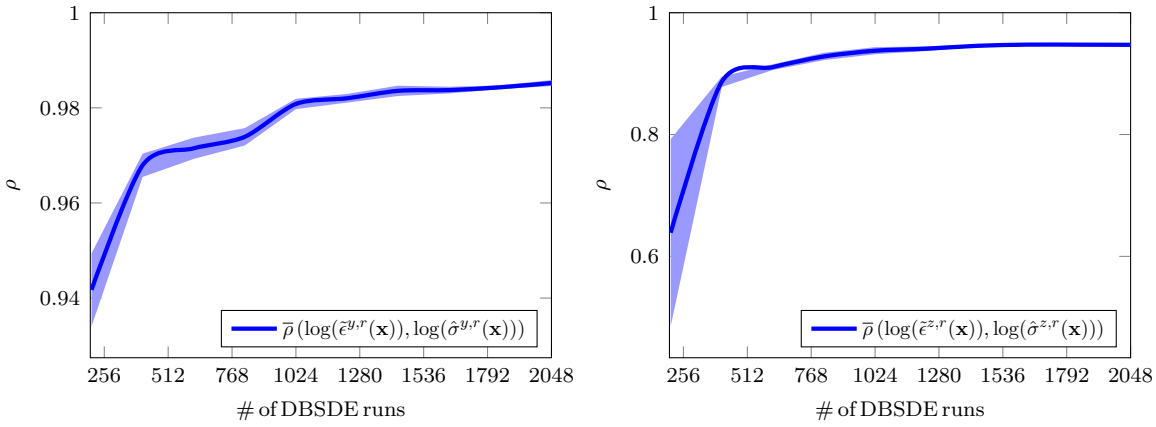
(e) Dataset $\mathcal{D}_3^y$.



(f) Dataset $\mathcal{D}_3^z$.

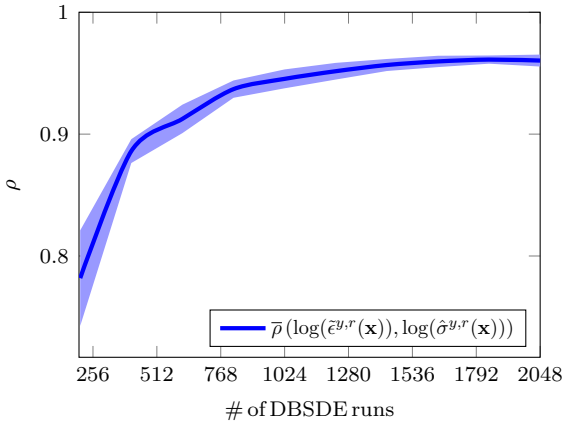
Figure 3.10: Correlation between the relative RMSE and ensemble STD values for different DBSDE runs, and the mean correlation between the relative RMSE and estimated STD values from the UQ model (STD of correlation given in the shaded area) for $\mathcal{D}_\mathrm{j}$, $\mathrm{j} = 1, 2, 3$ using the testing sample in Example 3.4.1. The black dot defines their intersection.

highly accurate initializers of $\left(Y_0^{\Delta,\hat{\theta}}, Z_0^{\Delta,\hat{\theta}}\right)$ in the DBSDE algorithm, instead of initializing them randomly using uniform distributions.

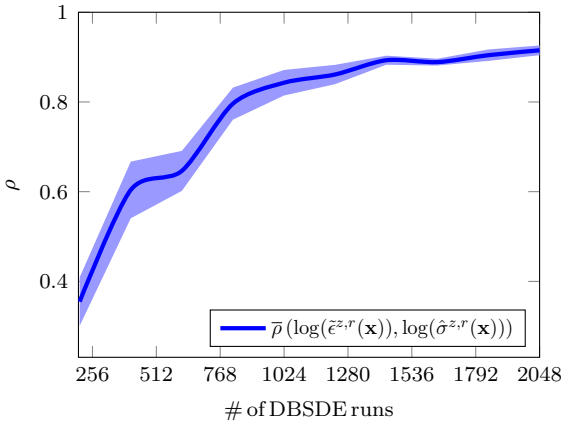


(a) Dataset $\mathcal{D}_1^y$.

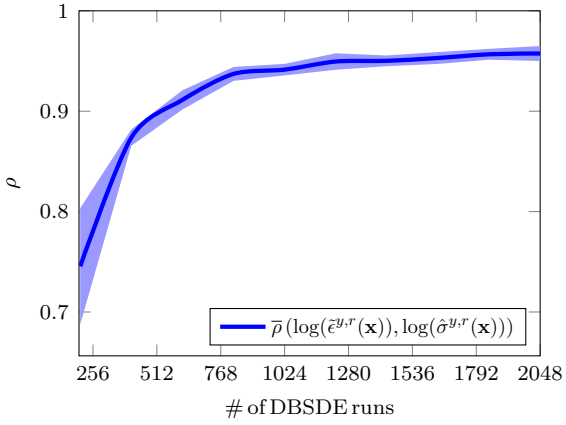
(b) Dataset $\mathcal{D}_1^z$.



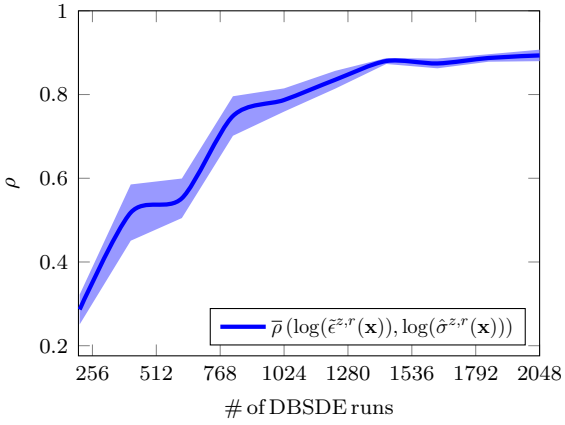
(c) Dataset $\mathcal{D}_2^y$.



(d) Dataset $\mathcal{D}_2^z$.



(e) Dataset $\mathcal{D}_3^y$.



(f) Dataset $\mathcal{D}_3^z$.

Figure 3.11: Mean correlation between the relative RMSE and estimated STD values from the UQ model while increasing the number of DBSDE runs to train the model from 10% to 100% of M^{train} for $\mathcal{D}_\mathrm{j}$, $\mathrm{j} = 1, 2, 3$ using the testing sample in Example 3.4.1. The STD of the correlation is given in the shaded area.

3.4.3.2 The DBSDE scheme for the Burgers type example

We generate dataset $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i\}_{i=1}^M$ using Algorithm 4, where $\mathbf{x}_i$ now includes the parameter set of the Burgers type BSDE, namely $\mathbf{x}_i = (b_i, T_i)$, and $\mathbf{y}_i \in \mathbb{R}$ and $\mathbf{z}_i \in \mathbb{R}^{1\times d}$ the corresponding

		Dataset		
UQ approach	Metric	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$
Ensemble for Y_0	$RMSE\left(Y_0(\mathbf{x}), \tilde{\mu}^y(\mathbf{x})\right)$	1.72e−2	1.70e−02	2.04e−02
UQ model for Y_0	$\overline{RMSE}\left(Y_0(\mathbf{x}), \hat{\mu}^y(\mathbf{x})\right)$	1.97e−2 (9.89e−04)	3.29e−02 (1.34e−03)	3.35e−02 (1.44e−03)
Ensemble for Z_0	$RMSE\left(Z_0(\mathbf{x}), \tilde{\mu}^z(\mathbf{x})\right)$	6.47e−02	6.00e−02	6.02e−02
UQ model for Z_0	$\overline{RMSE}\left(Z_0(\mathbf{x}), \hat{\mu}^z(\mathbf{x})\right)$	7.05e−02 (2.30e−03)	8.27e−02 (4.71e−03)	8.17e−02 (7.11e−03)

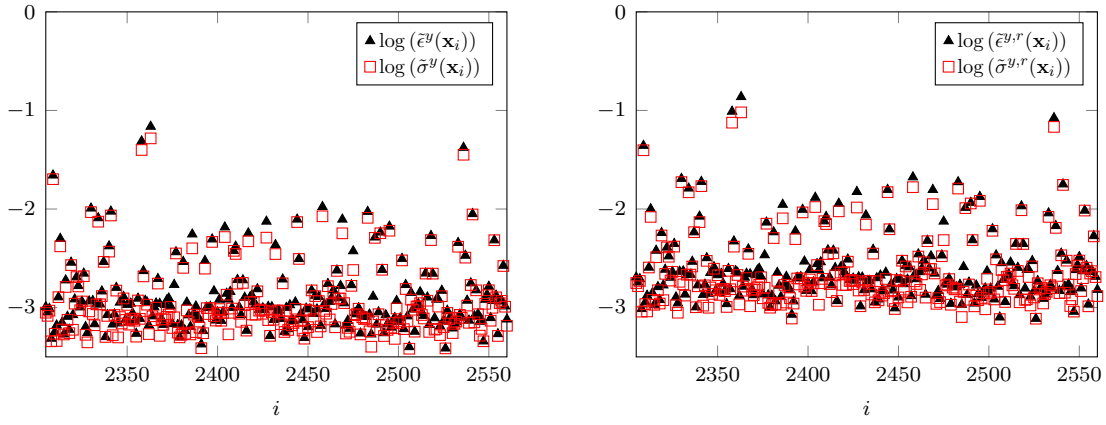
Table 3.5: RMSE between the exact solution and ensemble mean values, and the mean RMSE between the exact solution and estimated mean values from the UQ model for $\mathcal{D}_\mathrm{j}$, $\mathrm{j} = 1, 2, 3$ using the testing sample in Example 3.4.1. The STD of the RMSE is given in the brackets.

approximate solution for $Y_0(\mathbf{x}_i)$ and $Z_0(\mathbf{x}_i)$. We keep the same hyperparameter values of the DBSDE algorithm as in Example 3.4.1 to generate the dataset $\mathcal{D}$ and consider only varying the BSDE parameter set as done for the dataset $\mathcal{D}_3$ in Example 3.4.1, namely $(b, T, \Delta t)$ are varied. In Table 3.6 the range of parameter values is reported. We set again $M = 2560$ and $Q = 10$

Dataset	Parameter range			
	b	T	N	Δt
$\mathcal{D}$	$[0.2, 40]$	$\left[\frac{1}{12}, 0.3\right]$	32	$\frac{T}{N}$

Table 3.6: Parameter range for Example 3.4.2.

for each BSDE parameter set, and split the dataset $\mathcal{D}$ into $\mathcal{D}^y$ and $\mathcal{D}^z$. The RMSE, ensemble STD, and their relative values for Y_0 and Z_0^1 are displayed in Figures 3.12 and 3.13, sorted by the value of the corresponding exact solution. Note that for Z_0, only the results for the first component are shown (with similar behavior observed for other components of Z_0). We find

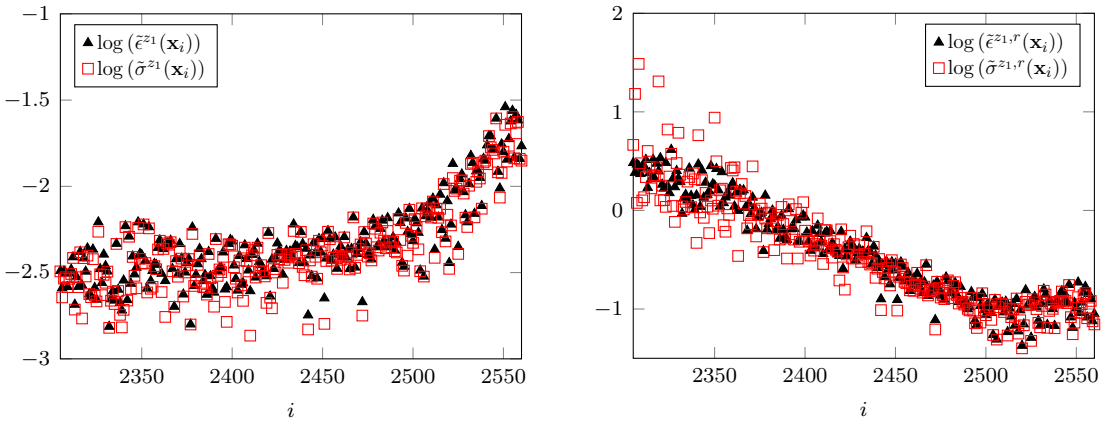


(a) Estimates for $\mathcal{D}^y$.

(b) Relative estimates for $\mathcal{D}^y$.

Figure 3.12: RMSE, the ensemble STD and their relative estimates for $\mathcal{D}^y$ sorted by the value of the exact solution in Example 3.4.2.

that the DBSDE algorithm produces negative approximations of Z_0 in a significant number of cases (1270 cases), especially for small values of b ($b \approx 0.2$). Additionally, for large values of b and T ($b > 40$ and $T > 0.3$), the relative RMSE values become very large. Similar to the previous example, Figures 3.12 and 3.13 demonstrate a strong positive correlation among the RMSE, ensemble STD, and their relative values. The correlation values in the log-domain are reported in Table 3.7 for dataset $\mathcal{D}$.



(a) Estimates for $\mathcal{D}^z$.

(b) Relative estimates for $\mathcal{D}^z$.

Figure 3.13: RMSE, the ensemble STD and their relative estimates for $\mathcal{D}^z$ sorted by the value of the exact solution in Example 3.4.2.

Measure	Correlation	Dataset $\mathcal{D}$
Absolute measure for Y_0	$\rho\left(\log(\tilde{\epsilon}^y(\mathbf{x})), \log(\tilde{\sigma}^y(\mathbf{x}))\right)$	0.9815
Relative measure for Y_0	$\rho\left(\log(\tilde{\epsilon}^{y,r}(\mathbf{x})), \log(\tilde{\sigma}^{y,r}(\mathbf{x}))\right)$	0.9813
Absolute measure for Z_0^1	$\rho\left(\log(\tilde{\epsilon}^{z_1}(\mathbf{x})), \log(\tilde{\sigma}^{z_1}(\mathbf{x}))\right)$	0.9900
Relative measure for Z_0^1	$\rho\left(\log(\tilde{\epsilon}^{z_1,r}(\mathbf{x})), \log(\tilde{\sigma}^{z_1,r}(\mathbf{x}))\right)$	0.9353

Table 3.7: Correlation between the RMSE and ensemble STD values, and their relative values for $\mathcal{D}$ in Example 3.4.2.

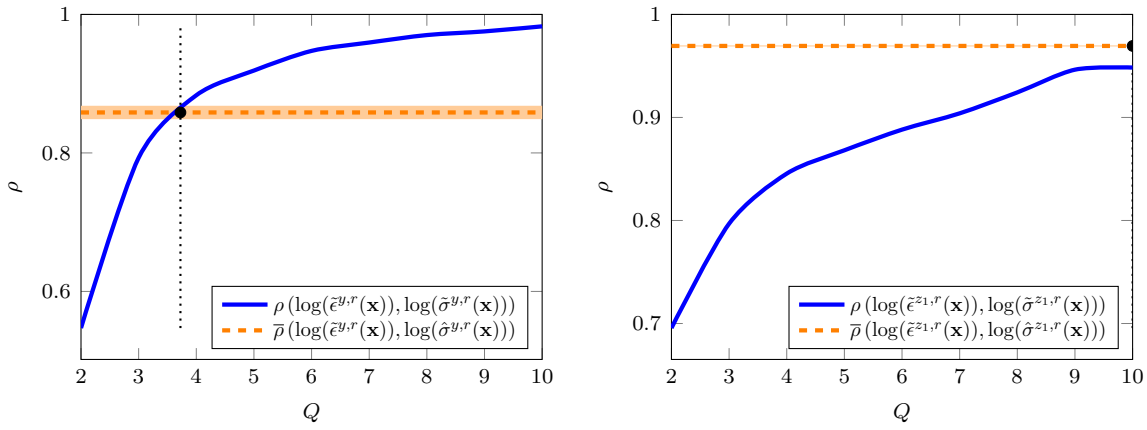
To train the UQ model, we follow the same procedure as in Example 3.4.1. We again choose a testing and validation sample of 256 and use the rest for training the UQ model. Note that $\mathfrak{n} = 3$ since $\mathbf{x}_i = (b_i, T_i, \Delta t_i)$. Based on the validation sample, the fine-tuned hyperparameters for Y_0 are as follows: $B^y = 128$, $\lambda^y = 1\text{e}{-3}$, and a PC-LR approach with $\alpha^y \in \{1\text{e}{-3}, 3\text{e}{-4}, 1\text{e}{-4}, 3\text{e}{-5}, 1\text{e}{-5}\}$ and $ep^y = 5000$ for $\alpha^y = 1\text{e}{-3}$ and $ep^y = 500$ for the other learning rates. For Z_0^1, the fine-tuned hyperparameters are: $B^z = 128$, $\lambda^z = 3\text{e}{-2}$, and a PC-LR approach with $\alpha^z = \alpha^y$ and $ep^z = 1000$ for $\alpha^z = 1\text{e}{-3}$ and $ep^z = 100$ for the other learning rates. In Table 3.8, we present the correlation between the relative RMSE and ensemble STD values $\rho\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\tilde{\sigma}^r(\mathbf{x}))\right)$, as well as the mean correlation between the relative RMSE and estimated STD values $\overline{\rho}\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\hat{\sigma}^r(\mathbf{x}))\right)$. The correlation values for the relative ensemble

UQ approach	Metric	Dataset $\mathcal{D}$
Ensemble for Y_0	$\rho\left(\log(\tilde{\epsilon}^{y,r}(\mathbf{x})), \log(\tilde{\sigma}^{y,r}(\mathbf{x}))\right)$	0.9828
UQ model for Y_0	$\overline{\rho}\left(\log(\tilde{\epsilon}^{y,r}(\mathbf{x})), \log(\hat{\sigma}^{y,r}(\mathbf{x}))\right)$	0.8584 (0.0096)
Ensemble for Z_0^1	$\rho\left(\log(\tilde{\epsilon}^{z_1,r}(\mathbf{x})), \log(\tilde{\sigma}^{z_1,r}(\mathbf{x}))\right)$	0.9484
UQ model for Z_0^1	$\overline{\rho}\left(\log(\tilde{\epsilon}^{z_1,r}(\mathbf{x})), \log(\hat{\sigma}^{z_1,r}(\mathbf{x}))\right)$	0.9694 (0.0005)

Table 3.8: Correlation between the relative RMSE and ensemble STD values, and the mean correlation between the relative RMSE and estimated STD values from the UQ model for $\mathcal{D}$ using the testing sample in Example 3.4.2. The STD of the correlation is given in the brackets.

STD and the mean correlation values for the relative estimated STD from our UQ model are close for Y_0 and even better for Z_0^1, indicating that our UQ model can provide highly accurate estimates of the relative ensemble STD also in high dimensions. In Figure 3.14, we display the

correlation between the relative RMSE and ensemble STD values $\rho\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\tilde{\sigma}^r(\mathbf{x}))\right)$ for different DBSDE runs, the mean correlation between the relative RMSE and estimated STD values $\overline{\rho}\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\hat{\sigma}^r(\mathbf{x}))\right)$, and their intersection. The relative estimated STD from the



(a) Dataset $\mathcal{D}^y$.

(b) Dataset $\mathcal{D}^z$.

Figure 3.14: Correlation between the relative RMSE and ensemble STD values for different DBSDE runs, and the mean correlation between the relative RMSE and estimated STD values from the UQ model (STD of correlation given in the shaded area) for $\mathcal{D}$ using the testing sample in Example 3.4.2. The black dot defines their intersection.

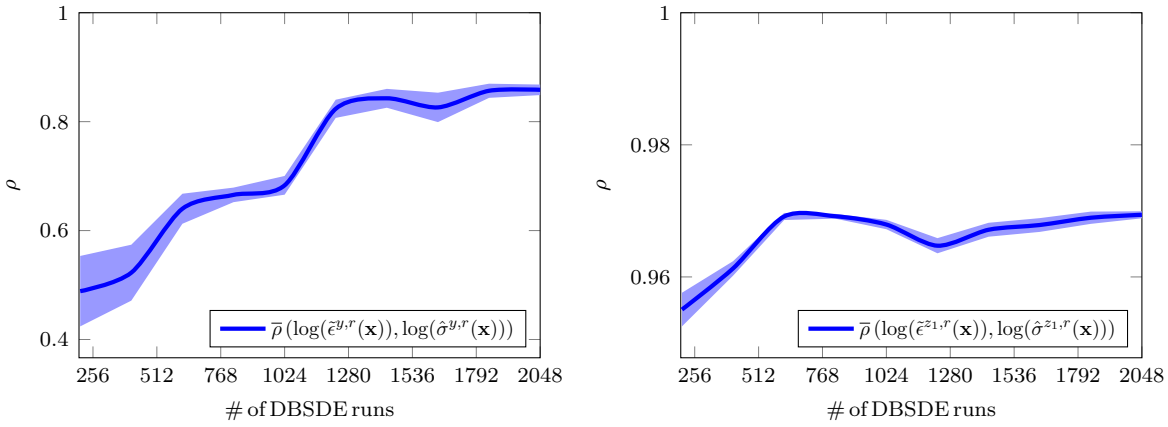
UQ model achieves the quality of the relative ensemble STD calculated from around $Q = 4$ runs of the DBDSE algorithm for Y_0 and more than $Q = 10$ for Z_0^1. In this example, the performance of the UQ model for Z_0 improves compared to the previous example. One possible explanation for this improvement is the exact solution in this example, which slightly varies for Z_0 across different BSDE parameter sets $\mathbf{x}$, namely Example 3.4.2 is less challenging than Example 3.4.1 for Z_0.

To show the computation cost of generating the training data for the UQ model, we display in Figure 3.15 the mean correlation between the relative RMSE and estimated STD values $\overline{\rho}\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\hat{\sigma}^r(\mathbf{x}))\right)$ while increasing the number of DBSDE runs to train the model. Even when training the UQ model with around 1024 DBSDE runs, a good estimate of the STD is achieved.

Next, we examine the performance of our UQ model for the mean of the approximate solution. We calculate the RMSE between the exact solution and ensemble mean values $RMSE\left(Y_0(\mathbf{x}), \tilde{\mu}^y(\mathbf{x})\right)$, as well as the mean RMSE between the exact solution and estimated mean values $\overline{RMSE}\left(Y_0(\mathbf{x}), \hat{\mu}(\mathbf{x})\right)$ for Y_0, and similarly for Z_0^1. The corresponding values are reported in Table 3.9. Based on the results, we can conclude that the estimated means given by

UQ approach	Metric	Dataset $\mathcal{D}$
Ensemble for Y_0	$RMSE\left(Y_0(\mathbf{x}), \tilde{\mu}^y(\mathbf{x})\right)$	3.80e−03
UQ model for Y_0	$\overline{RMSE}\left(Y_0(\mathbf{x}), \hat{\mu}^y(\mathbf{x})\right)$	1.12e−03 (1.24e−04)
Ensemble for Z_0^1	$RMSE\left(Z_0^1(\mathbf{x}), \tilde{\mu}^{z_1}(\mathbf{x})\right)$	2.82e−03
UQ model for Z_0^1	$\overline{RMSE}\left(Z_0^1(\mathbf{x}), \hat{\mu}^{z_1}(\mathbf{x})\right)$	9.08e−04 (1.92e−04)

Table 3.9: RMSE between the exact solution and ensemble mean values, and the mean RMSE between the exact solution and estimated mean values from the UQ model for $\mathcal{D}$ using the testing sample in Example 3.4.2. The STD of the RMSE is given in the brackets.



(a) Dataset $\mathcal{D}^y$. (b) Dataset $\mathcal{D}^z$.

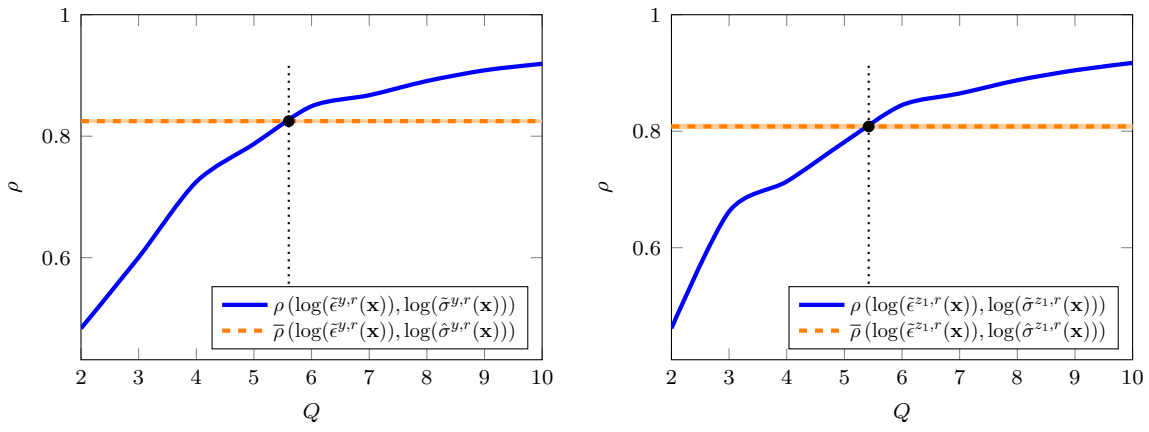
Figure 3.15: Mean correlation between the relative RMSE and estimated STD values from the UQ model while increasing the number of DBSDE runs to train the model from 10% to 100% of M^{train} for $\mathcal{D}$ using the testing sample in Example 3.4.2. The STD of the correlation is given in the shaded area.

our UQ model can serve as highly accurate initializers for $\left(Y_0^{\Delta,\hat{\theta}}, Z_0^{\Delta,\hat{\theta}}\right)$ in the DBSDE algorithm also in high dimensions.

3.4.3.3 The LaDBSDE scheme for the Burgers type example

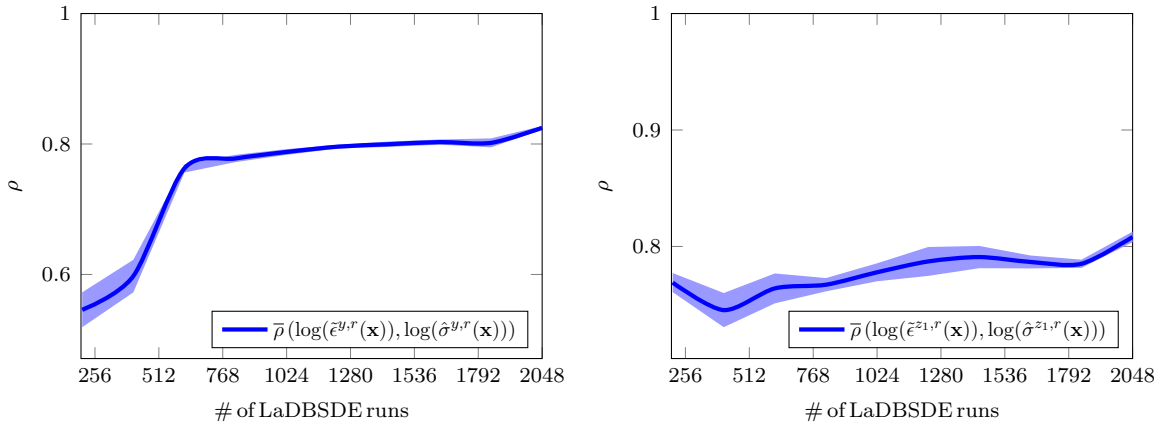
We now demonstrate that the proposed UQ model can be applied to other deep learning-based BSDE schemes, specifically the LaDBSDE scheme [60]. As application, we select the Burgers type BSDE, due to its nonlinearity and higher dimensionality. We choose the hyperparameters for the LaDBSDE scheme similarly to those used in the DBSDE scheme. More precisely, we consider $\mathfrak{K} = 30000$, $\alpha = 1\text{e}{-3}$, $B = 128$, $\eta = 10 + d$, $L = 4$, and $\varrho(x) = \tanh(x)$. Batch normalization is applied after each matrix multiplication and before activation functions. The Adam optimizer is used as an SGD-type algorithm. Using BSDE parameter set $\mathbf{x}$ as outlined in Table 3.6, we apply the LaDBSDE scheme and collect the corresponding approximation of $Y_0(\mathbf{x})$ and $Z_0(\mathbf{x})$. To distinguish this dataset from the one generated by the DBSDE scheme, we denote it as $\check{\mathcal{D}}$.

We split the dataset $\check{\mathcal{D}}$ into $\check{\mathcal{D}}^y$ and $\check{\mathcal{D}}^z$ to train and test the UQ model for Y_0 and Z_0, respectively. Using a validation sample of 256, the fine-tuned hyperparameters of the UQ model for Y_0 are: $B^y = 32$, $\lambda^y = 3\text{e}{-3}$, and a PC-LR approach with $\alpha^y \in \{1\text{e}{-3}, 3\text{e}{-4}, 1\text{e}{-4}, 3\text{e}{-5}, 1\text{e}{-5}\}$ and $ep^y = 1000$ for $\alpha^y = 1\text{e}{-3}$ and $ep^y = 100$ for the other learning rates. For Z_0^1, we have $B^z = 128$, $\lambda^z = 1\text{e}{-2}$, and a PC-LR approach with $\alpha^z = \alpha^y$ and $ep^z = 1000$ for $\alpha^z = 1\text{e}{-3}$ and $ep^z = 500$ for the other learning rates. In Figure 3.16, we display the correlation between the relative RMSE and ensemble STD values $\rho\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\tilde{\sigma}^r(\mathbf{x}))\right)$ for different LaDBSDE runs, the mean correlation between the relative RMSE and estimated STD values $\overline{\rho}\left(\log(\tilde{\epsilon}^r(\mathbf{x})), \log(\hat{\sigma}^r(\mathbf{x}))\right)$, and their intersection. The relative estimated STD from the UQ model achieves the quality of the relative ensemble STD calculated from $Q = 6$ runs of the LaDBDSE algorithm for Y_0 and around $Q = 5$ for Z_0^1 (similar performance for other components of Z_0). This demonstrates that the UQ model can be applied to other deep learning-based BSDE schemes, which work in a similar manner to the DBSDE scheme. The computational cost to train the UQ model in the case of the LaDBSDE scheme is shown in Figure 3.17. We can draw the same conclusions as for the



(a) Dataset $\check{\mathcal{D}}^y$. (b) Dataset $\check{\mathcal{D}}^z$.

Figure 3.16: Correlation between the relative RMSE and ensemble STD values for different LaDBSDE runs, and the mean correlation between the relative RMSE and estimated STD values from the UQ model (STD of correlation given in the shaded area) for $\check{\mathcal{D}}$ using the testing sample in Example 3.4.2. The black dot defines their intersection.



(a) Dataset $\check{\mathcal{D}}^y$. (b) Dataset $\check{\mathcal{D}}^z$.

Figure 3.17: Mean correlation between the relative RMSE and estimated STD values from the UQ model while increasing the number of LaDBSDE runs from 10% to 100% of M^{train} for $\check{\mathcal{D}}$ using the testing sample in Example 3.4.2. The STD of the correlation is given in the shaded area.

DBSDE scheme from the results obtained in this case.

In Table 3.10, we present the RMSE between the exact solution and ensemble mean values $RMSE\left(Y_0(\mathbf{x}), \tilde{\mu}^y(\mathbf{x})\right)$, as well as the mean RMSE between the exact solution and estimated mean values $\overline{RMSE}\left(Y_0(\mathbf{x}), \hat{\mu}^y(\mathbf{x})\right)$ for Y_0 (similar for Z_0^1). These values are very close. The same conclusion can be drawn for Z_0^1. This indicates that our UQ model can also provide highly accurate approximations of the mean of the approximate solution for the LaDBSDE scheme. Moreover, the RMSE values are smaller than those in Table 3.9 for Y_0 and Z_0^1. Hence, our UQ model can identify on average the improved approximations provided by the LaDBSDE scheme compared to the DBSDE scheme.

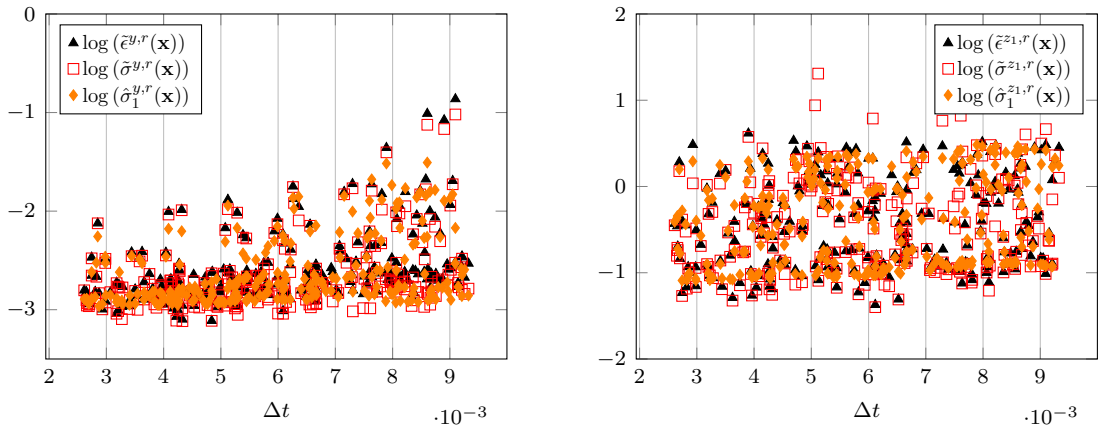
UQ approach	Metric	Dataset $\check{D}$
Ensemble for Y_0	$RMSE\left(Y_0(\mathbf{x}), \tilde{\mu}^y(\mathbf{x})\right)$	1.32e−03
UQ model for Y_0	$\overline{RMSE}\left(Y_0(\mathbf{x}), \hat{\mu}^y(\mathbf{x})\right)$	1.11e−03 (1.93e−05)
Ensemble for Z_0^1	$RMSE\left(Z_0^1(\mathbf{x}), \tilde{\mu}^{z_1}(\mathbf{x})\right)$	1.35e−03
UQ model for Z_0^1	$\overline{RMSE}\left(Z_0^1(\mathbf{x}), \hat{\mu}^{z_1}(\mathbf{x})\right)$	1.14e−03 (7.65e−05)

Table 3.10: RMSE between the exact solution and ensemble mean values, and the mean RMSE between the exact solution and estimated mean values from the UQ model for $\check{\mathcal{D}}$ using the testing sample in Example 3.4.2. The STD of the RMSE is given in the brackets.

3.4.4 Practical implications of the UQ model

In this section, we study what sources of uncertainty can be captured by our UQ model and we demonstrate its applicability to downstream tasks.

We start by analyzing the sources of uncertainty that our UQ model can effectively capture. It can be expected that the estimated STD captures uncertainty due to the optimization heuristic as well as the uncertainty due to data sampling. The reason is that each training of the DBSDE scheme provides a slightly different solution, which is therefore reflected by the training dataset of the UQ model. The differences in the solutions are caused by: 1) The random initialization of the parameters θ of the DNNs in the DBSDE scheme, which leads to different gradient descent iterations. 2) The sampling from the distribution of Brownian motion increments in the DBSDE scheme, which also leads to different gradient approximations. However, it is less clear about the uncertainty stemming from the discretization error, as it might bias the approximations provided by the DBSDE scheme. To illustrate the behavior of the relative RMSE, ensemble STD, and estimated STD values across varying Δt values, we display these measures in Figure 3.18 using the testing data in dataset $\mathcal{D}$ from Example 3.4.2. Note that in this section we use only the relative



(a) Dataset $\mathcal{D}^y$. (b) Dataset $\mathcal{D}^z$.

Figure 3.18: Relative RMSE, ensemble STD and estimated STD values from the UQ model for increasing value of Δt for $\mathcal{D}$ using the testing sample in Example 3.4.2.

estimated STD from the first trained UQ model (out of our ensemble of 10 models considered before for evaluation). As Δt decreases, the bias from the discretization error decreases, and the relative estimated STD improves in approximating the relative RMSE. For larger values of Δt, the bias grows, but the STD also increases. Therefore, the trend of the STD remains consistent with the RMSE, indicating that the relative estimated STD remains reasonable across different values of Δt for approximating the relative RMSE. The same is observed for the relative

ensemble STD. To measure the strength and direction of the monotonic relationship between the relative RMSE and estimated STD values across Δt values, we consider Spearman's rank correlation (ς). This metric is calculated as

$$\varsigma\left(\tilde{\epsilon}^r(\mathbf{x}), \hat{\sigma}^r(\mathbf{x})\right) := 1 - \frac{6\sum_{i=1}^{M^{test}} rank(\tilde{\epsilon}^r(\mathbf{x}))_i - rank(\hat{\sigma}^r(\mathbf{x}))_i}{M^{test}\left((M^{test})^2 - 1\right)},$$

where, e.g. $rank(\tilde{\epsilon}^r(\mathbf{x}))_i$ is the assigned rank to $\tilde{\epsilon}^r(\mathbf{x}_i)$. Note that $\varsigma\left(\tilde{\epsilon}^r(\mathbf{x}), \tilde{\sigma}^r(\mathbf{x})\right)$ is calculated similarly. The rank correlation values are displayed in Table 3.11. The high positive rank corre-

UQ approach	Rank correlation	Dataset $\mathcal{D}$
Ensemble for Y_0	$\varsigma\left(\tilde{\epsilon}^{y,r}(\mathbf{x}), \tilde{\sigma}^{y,r}(\mathbf{x})\right)$	0.9282
UQ model for Y_0	$\varsigma\left(\tilde{\epsilon}^{y,r}(\mathbf{x}), \hat{\sigma}_1^{y,r}(\mathbf{x})\right)$	0.6977
Ensemble for Z_0^1	$\varsigma\left(\tilde{\epsilon}^{z_1,r}(\mathbf{x}), \tilde{\sigma}^{z_1,r}(\mathbf{x})\right)$	0.9780
UQ model for Z_0^1	$\varsigma\left(\tilde{\epsilon}^{z_1,r}(\mathbf{x}), \hat{\sigma}_1^{z_1,r}(\mathbf{x})\right)$	0.9608

Table 3.11: Rank correlation between the relative RMSE, ensemble STD, and estimated STD values for $\mathcal{D}$ using the testing sample in Example 3.4.2.

lation values indicate that the relative estimated STD from our UQ model can reflect multiple sources of uncertainty, including the uncertainty caused by the discretization error.

Next, we aim to determine whether our UQ model can detect the enhanced performance of the LaDBSDE scheme over the DBSDE scheme for each BSDE parameter set, rather than just considering the average performance as shown before. For this purpose, the accuracy score (acc) for the testing sample of datasets $\mathcal{D}$ and $\check{\mathcal{D}}$ of Example 3.4.2 is considered. We define binary labels to calculate it. For the relative RMSE, we consider

$$\ell_{\tilde{\epsilon}}^r(\mathbf{x}_i) := \begin{cases} 1 & \text{if } \tilde{\epsilon}^{r,LaDBSDE}(\mathbf{x}_i) < \tilde{\epsilon}^{r,DBSDE}(\mathbf{x}_i), \\ 0 & \text{otherwise,} \end{cases}$$

for the BSDE parameter set $\mathbf{x}_i$. Similarly, we define binary labels $\ell_{\tilde{\sigma}}^r(\mathbf{x})$ and $\ell_{\hat{\sigma}}^r(\mathbf{x}_i)$ for the relative ensemble STD and estimated STD, respectively. The accuracy score between the labels of the relative RMSE and estimated STD values represents the number of BSDE parameter sets in which the smallest relative RMSE and estimated STD values are achieved from the same scheme, divided by the total number of BSDE parameter sets, i.e.

$$acc\left(\ell_{\tilde{\epsilon}}^r(\mathbf{x}), \ell_{\hat{\sigma}}^r(\mathbf{x})\right) := \frac{1}{M^{test}} \sum_{i=1}^{M^{test}} \mathbb{1}_{\ell_{\tilde{\epsilon}}^r(\mathbf{x}_i) = \ell_{\hat{\sigma}}^r(\mathbf{x}_i)}.$$

Similarly, we calculate the accuracy score between the labels of relative RMSE and ensemble STD values $acc\left(\ell_{\tilde{\epsilon}}^r(\mathbf{x}), \ell_{\tilde{\sigma}}^r(\mathbf{x})\right)$ and report them in Table 3.12. The accuracy score of almost 1 for Z_0^1

UQ approach	Accuracy score	Datasets $\mathcal{D}$ and $\check{\mathcal{D}}$
Ensemble for Y_0	$acc\left(\ell_{\tilde{\epsilon}}^{y,r}(\mathbf{x}), \ell_{\tilde{\sigma}}^{y,r}(\mathbf{x})\right)$	0.9023
UQ model for Y_0	$acc\left(\ell_{\tilde{\epsilon}}^{y,r}(\mathbf{x}), \ell_{\hat{\sigma}_1}^{y,r}(\mathbf{x})\right)$	0.8008
Ensemble for Z_0^1	$acc\left(\ell_{\tilde{\epsilon}}^{z_1,r}(\mathbf{x}), \ell_{\tilde{\sigma}}^{z_1,r}(\mathbf{x})\right)$	0.9922
UQ model for Z_0^1	$acc\left(\ell_{\tilde{\epsilon}}^{z_1,r}(\mathbf{x}), \ell_{\hat{\sigma}_1}^{z_1,r}(\mathbf{x})\right)$	0.9922

Table 3.12: Accuracy score between the binary labels of the relative RMSE, ensemble STD, and estimated STD values from $\mathcal{D}$ and $\check{\mathcal{D}}$ using the testing sample in Example 3.4.2.

implies that the relative estimated STD from our UQ model illustrates enhanced performance when comparing the DBSDE and LaDBSDE schemes across the entire testing sample. This observation is valid only for 80% of the testing sample for Y_0.

Moreover, as the RMSE increases due to propagated errors with increasing N (from a certain value of N depending on the BSDE parameter set values), it is of interest to determine whether the UQ model can identify the value of N at which the algorithm attains the smallest RMSE based on the estimated STD. To investigate this, we generate a dataset $\mathcal{D}^{\mathbf{N}}$ similar to $\mathcal{D}$ in Table 3.6 with a fixed maturity $T = 0.3$, and each sampled BSDE parameter set is solved for $\mathbf{N} = \{2, 8, 32, 128\}$. The dataset $\mathcal{D}^{\mathbf{N}}$ consists of 2560 BSDE parameter sets, resulting in a total number of samples $M = 10240$. We choose $M^{train} = 8192$ and $M^{valid} = M^{test} = 1024$. Note that $\mathfrak{n} = 2$ since $\mathbf{x}_i = (b_i, N)$, $N \in \mathbf{N}$. We use the same hyperparameters for the UQ model as for dataset $\mathcal{D}$ and train only one model. To evaluate the accuracy score, we define the binary multi-label

$$\boldsymbol{\ell}^r_{\tilde{\epsilon}}(\mathbf{x}_i) := \begin{cases} \{1,0,0,0\} & \text{if } N^{*,r}_{\tilde{\epsilon}}(\mathbf{x}_i) = 2, \\ \{0,1,0,0\} & \text{if } N^{*,r}_{\tilde{\epsilon}}(\mathbf{x}_i) = 8, \\ \{0,0,1,0\} & \text{if } N^{*,r}_{\tilde{\epsilon}}(\mathbf{x}_i) = 32, \\ \{0,0,0,1\} & \text{if } N^{*,r}_{\tilde{\epsilon}}(\mathbf{x}_i) = 128, \end{cases}$$

for the relative RMSE, where $N^{*,r}_{\tilde{\epsilon}}(\mathbf{x}_i) := \arg\min_{N \in \mathbf{N}} \tilde{\epsilon}^r(b_i, N)$. The binary multi-labels for the relative ensemble STD $\boldsymbol{\ell}^r_{\tilde{\sigma}}(\mathbf{x}_i)$ and estimated STD $\boldsymbol{\ell}^r_{\hat{\sigma}}(\mathbf{x}_i)$ are defined similarly. The accuracy score values between these multi-labels for the testing sample of $\mathcal{D}^{\mathbf{N}}$ are presented in Table 3.13. We observe that the accuracy score values between the multi-labels of the relative RMSE and

UQ approach	Accuracy score	Dataset $\mathcal{D}^{\mathbf{N}}$
Ensemble for Y_0	$acc\left(\boldsymbol{\ell}^{y,r}_{\tilde{\epsilon}}(\mathbf{x}), \boldsymbol{\ell}^{y,r}_{\tilde{\sigma}}(\mathbf{x})\right)$	0.6836
UQ model for Y_0	$acc\left(\boldsymbol{\ell}^{y,r}_{\tilde{\epsilon}}(\mathbf{x}), \boldsymbol{\ell}^{y,r}_{\hat{\sigma}_1}(\mathbf{x})\right)$	0.3594
Ensemble for Z^1_0	$acc\left(\boldsymbol{\ell}^{z_1,r}_{\tilde{\epsilon}}(\mathbf{x}), \boldsymbol{\ell}^{z_1,r}_{\tilde{\sigma}}(\mathbf{x})\right)$	0.8008
UQ model for Z^1_0	$acc\left(\boldsymbol{\ell}^{z_1,r}_{\tilde{\epsilon}}(\mathbf{x}), \boldsymbol{\ell}^{z_1,r}_{\hat{\sigma}_1}(\mathbf{x})\right)$	0.5313

Table 3.13: Accuracy score between the multi-labels of the relative RMSE, ensemble STD, and estimated STD values from $\mathcal{D}^{\mathbf{N}}$ using the testing sample in Example 3.4.2.

estimated STD values $acc\left(\boldsymbol{\ell}^r_{\tilde{\epsilon}}(\mathbf{x}), \boldsymbol{\ell}^r_{\tilde{\sigma}}(\mathbf{x})\right)$ are between 0.4 and 0.5. This indicates that the relative estimated STD correctly predicted the value of N with the smallest relative RMSE for around 40% or 50% of the BSDE parameter sets in the testing sample.

The accuracy score serves as a restrictive metric, requiring each predicted label $\boldsymbol{\ell}^r_{\tilde{\sigma}}(\mathbf{x})$ or $\boldsymbol{\ell}^r_{\hat{\sigma}}(\mathbf{x})$ to exactly match the true label $\boldsymbol{\ell}^r_{\tilde{\epsilon}}(\mathbf{x})$. Hence, it doesn't tolerate partial errors. For instance, if the N value with the smallest relative RMSE coincides with the one having the second smallest relative estimated STD, the prediction is counted as incorrect. This rigid evaluation fails to consider the order of predicted labels. For this purpose, we consider the mean reciprocal rank metric (MRR), since there is only one relevant label per sample. It measures the effectiveness of a model in ranking a list of predicted labels based on their relevance to the only true label. In our case, the true label is $N^{*,r}_{\tilde{\epsilon}}(\mathbf{x})$. The predicted ones are denoted by $\mathbf{N}^{sort,r}_{\tilde{\sigma}}(\mathbf{x})$ and $\mathbf{N}^{sort,r}_{\hat{\sigma}}(\mathbf{x})$, the ascending sorted $\mathbf{N}$ values for the BSDE parameter set $\mathbf{x}$ based on the value of relative ensemble STD and estimated STD, respectively. Hence, $MRR\left(N^{*,r}_{\tilde{\epsilon}}(\mathbf{x}), \mathbf{N}^{sort,r}_{\hat{\sigma}}(\mathbf{x})\right)$ is given by

$$MRR\left(N^{*,r}_{\tilde{\epsilon}}(\mathbf{x}), \mathbf{N}^{sort,r}_{\hat{\sigma}}(\mathbf{x})\right) := \frac{1}{256}\sum_{i=1}^{256} \frac{1}{pos\left(N^{*,r}_{\tilde{\epsilon}}(\mathbf{x}_i), \mathbf{N}^{sort,r}_{\hat{\sigma}}(\mathbf{x}_i)\right)},$$

where $pos\left(N^{*,r}_{\tilde{\epsilon}}(\mathbf{x}_i), \mathbf{N}^{sort,r}_{\hat{\sigma}}(\mathbf{x}_i)\right)$ gives the position where the true label $N^{*,r}_{\tilde{\epsilon}}(\mathbf{x}_i)$ is found in the

list of predicted labels $\mathbf{N}_{\hat{\sigma}}^{sort,r}(\mathbf{x}_i)$. The mean reciprocal rank values are reported in Table 3.14. We observe that, on average, our UQ model can show that the smallest relative RMSE is achieved

UQ approach	Mean reciprocal rank	Dataset $\mathcal{D}^{\mathbf{N}}$
Ensemble for Y_0	$MRR\left(N_{\tilde{\epsilon}}^{*,y,r}(\mathbf{x}), \mathbf{N}_{\tilde{\sigma}}^{sort,y,r}(\mathbf{x})\right)$	0.8268
UQ model for Y_0	$MRR\left(N_{\hat{\epsilon}}^{*,y,r}(\mathbf{x}), \mathbf{N}_{\hat{\sigma}_1}^{sort,y,r}(\mathbf{x})\right)$	0.6149
Ensemble for Z_0^1	$MRR\left(N_{\tilde{\epsilon}}^{*,z_1,r}(\mathbf{x}), \mathbf{N}_{\tilde{\sigma}}^{sort,z_1,r}(\mathbf{x})\right)$	0.8952
UQ model for Z_0^1	$MRR\left(N_{\hat{\epsilon}}^{*,z_1,r}(\mathbf{x}), \mathbf{N}_{\hat{\sigma}_1}^{sort,z_1,r}(\mathbf{x})\right)$	0.74584

Table 3.14: Mean reciprocal rank between the N value with the smallest relative RMSE and the ascending sorted $\mathbf{N}$ values based on the relative ensemble STD and estimated STD values from $\mathcal{D}^{\mathbf{N}}$ using the testing sample in Example 3.4.2.

for the N value of either the first or second smallest relative estimated STD.

3.5 Conclusions

In this chapter, we investigated the sources of uncertainty in the deep learning-based BSDE schemes and develop a UQ model based on heteroscedastic nonlinear regression to estimate the uncertainty. We applied the UQ model to the pioneering scheme developed in [22] and the one in [60]. The STD of the approximate solution captures the uncertainty, which is usually estimated by performing multiple runs of the algorithm with different datasets. This approach is quite computationally expensive, especially in high-dimensional cases. Our UQ model estimates the STD much cheaper, namely using a single run of the algorithm. Under the assumption of normally distributed errors with zero mean and the STD depending on the parameter set of the discretized BSDE, we employ a DNN to learn two functions that estimate the mean and STD of the approximate solution. The DNN is trained using a dataset of i.i.d. samples, consisting of various parameter sets of the discretized BSDE and their corresponding approximated solutions from a single run of the algorithm. The network parameters are optimized by minimizing the negative ln-likelihood. The STD is thus estimated much cheaper. Furthermore, the estimated mean can be leveraged to initialize the algorithm, improving the optimization process. Our numerical results demonstrate that the proposed UQ model provides reliable estimates of the mean and STD of the approximate solution for both considered schemes, even in high-dimensional cases. The estimated STD captures various sources of uncertainty, showcasing its capability in quantifying the uncertainty. Moreover, the UQ model illustrates the improved performance of the LaDBSDE scheme compared to the DBSDE scheme based on the corresponding estimated STD values. Finally, it can also identify the hyperparameters that yield a well-performing scheme.

Chapter 4

Conclusions and outlook

In this thesis, we developed novel numerical schemes for solving high-dimensional nonlinear BSDEs using differential and deep learning techniques, with a particular focus on applications in finance. Additionally, we delved into UQ to assess the reliability of these methods before being used in practice.

In Chapter 1, we reviewed contemporary deep learning BSDE schemes, categorizing them into forward and backward types, with a focus on the disadvantages as discussed in the literature. More specifically, the pioneering forward scheme [22] has been shown to suffer from two main drawbacks: 1) It can get stuck in a poor local minima or even diverge for a complex solution structure and a long terminal time. 2) It is capable in achieving much better approximation at $t = 0$, although pathwise solution is approximated. The contemporary forward scheme [86] can overcome only the second disadvantage. We showed numerically that even using RNN type architectures in [86] – which are NN architectures specialized for learning long complex sequences – does not help the scheme to converge to a good local minima. To overcome these limitations, we proposed a new forward scheme by reformulating the optimization problem. Our scheme employs a loss function that minimizes local losses defined at each time step, iterating the Euler-Maruyama discretization of integrals with the terminal condition. Through various numerical experiments including pricing problems, we demonstrated that our approach outperforms existing forward deep learning schemes [22, 86] by not getting stuck in a poor local minima and providing robust approximations across the entire time domain.

A common issue of deep learning BSDE schemes is their struggle to provide high-accurate first- and second-order gradient approximations, which are critical for financial applications. In Chapter 2, we addressed this by introducing a new class of schemes based on differential deep learning. Motivated by the fact that differential deep learning can provide an efficient approximation of the labels and their derivatives w.r.t. inputs, we reformulated the BSDE as a differential deep learning problem by using Malliavin calculus. This transformation required to estimate the solution, its gradient, and the Hessian matrix, represented by the triple of processes (Y, Z, Γ) in the BSDE system. To approximate this solution triple, we discretized the integrals within the system using the Euler-Maruyama method and parameterized their discrete version using DNNs. Depending on how the network parameters are optimized – either local or global optimization – we provided both backward and forward differential deep learning schemes. We stared with the backward scheme, where the DNN parameters are iteratively optimized backwardly at each time step by minimizing a differential learning type loss function, constructed as a weighted sum of the dynamics of the discretized BSDE system. A convergence analysis is performed to validate the accuracy and reliability of the proposed algorithm. In case of the forward scheme, only the algorithm is presented. The network parameters here are optimized

by globally minimizing a differential learning loss function, defined as a weighted sum of the dynamics of the discretized BSDE system that incorporate local loss functions. The proficiency of our new backward algorithm in terms of accuracy is demonstrated through numerous numerical experiments involving pricing and hedging nonlinear options in high dimensions. The proposed methods show potential for application in pricing and hedging financial derivatives in high-dimensional settings.

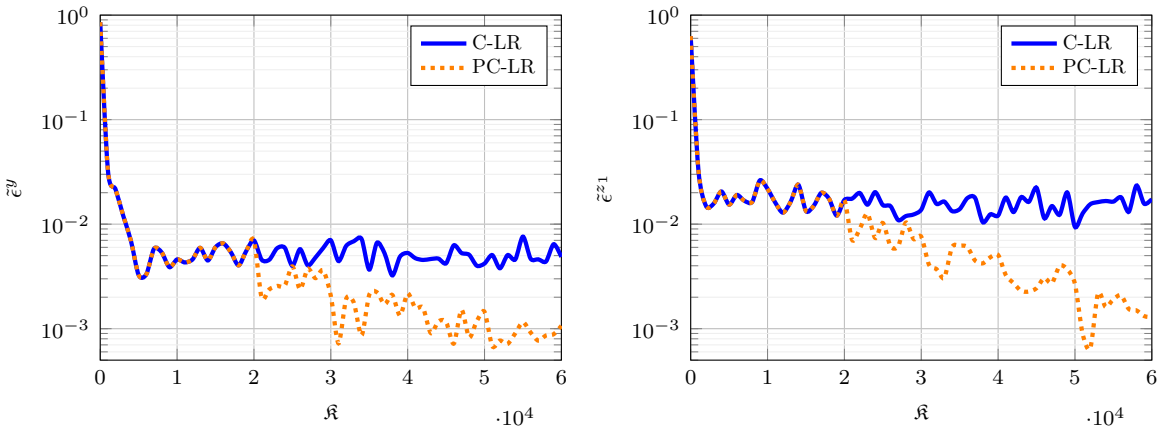
In Chapter 3, we explored the uncertainties inherent in deep learning BSDE schemes and proposed a UQ model leveraging heteroscedastic nonlinear regression to quantify these uncertainties. We applied the model to the pioneering scheme [22] as well as the approach presented in [60]. Traditionally, uncertainty is addressed by calculating the STD of approximate solutions through multiple runs of the algorithm on varying datasets – a process that becomes very computationally expensive in high-dimensional scenarios. In contrast, our UQ model efficiently estimates the STD using only a single algorithm run. Assuming normally distributed errors with zero mean and a parameter-dependent STD, we employed a DNN to estimate both the mean and STD of the approximate solution. The network is trained on i.i.d. samples, where each sample comprised parameter sets of the discretized BSDE and their corresponding approximate solutions from a single run. By minimizing the negative ln-likelihood, the DNN parameters are optimized, enabling efficient estimation of the STD. Additionally, the estimated mean can be utilized to initialize the algorithm, enhancing its optimization process. Numerical experiments demonstrated that the proposed UQ model reliably estimated the mean and STD for both schemes, even in high-dimensional cases. The estimated STD effectively captured various uncertainty sources, highlighting the model's robustness in quantifying uncertainty. Furthermore, the results illustrated the superior performance of [60] over [22], as indicated by the STD estimates. The UQ model also proved useful in identifying optimal hyperparameters for achieving a good-performing scheme.

This thesis covers a few existing and novel algorithms for solving high-dimensional BSDEs, and provides the first development of a UQ model for these schemes. However, several areas require further investigation. Future research directions, as outlined in the chapters, include conducting an error analysis of the scheme presented in Chapter 1, the forward one in Chapter 2, and addressing out-of-distribution data for the UQ model as suggested in Chapter 3. An intriguing open question concerns the assumptions about the boundedness of the Malliavin derivatives in Chapter 2, particularly where the Malliavin derivative of the diffusion term is assumed to be bounded in the forward SDE. For more general diffusion terms, advanced techniques, such as the truncation technique mentioned in [14], would be necessary. Additionally, exploring other option pricing and hedging problems, such as American type options, would require considering reflected BSDEs (RBSDEs). This presents further challenges, as adopting the developed schemes to RBSDEs requires approximating an additional process. It is called the reflecting process, which keep the solution Y_t of the BSDE going below the barrier $g(X_t)$, namely the American option price should not go below the payoff at time t. Moreover, extra terms in the loss function would be needed to enforce this constraint. This becomes even more complex and interesting when considering differential deep learning schemes, where the Malliavin derivative of the reflecting process must be taken into account.

Appendix A

Impact of the sources of uncertainty for the Burgers type BSDE

In this section, we visualize the effect of different errors on the RMSE for Example 3.4.2. The impact of the optimization error is shown in Figure A.1 using C-LR and PC-LR approaches. For the discretization error, see Figure A.2. The effect of the optimization error and propagated

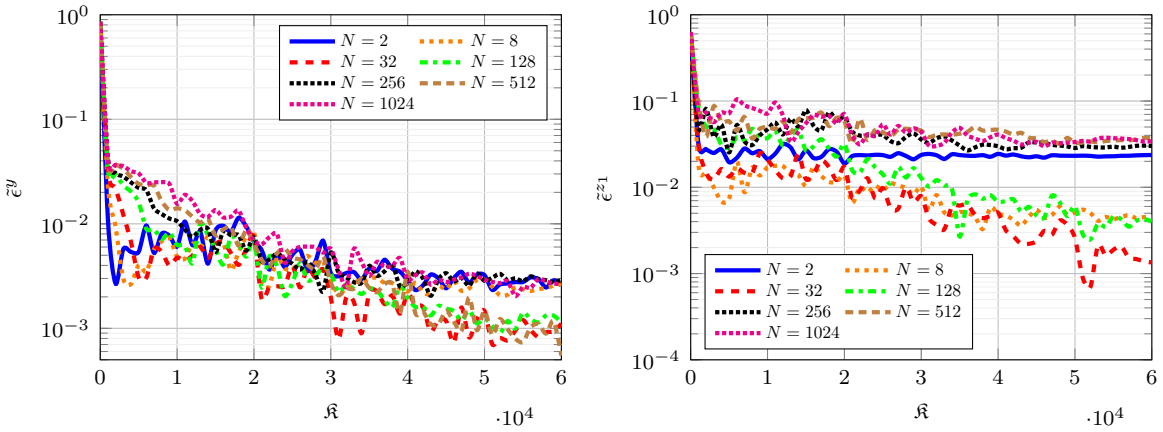


(a) RMSE values for Y_0.

(b) RMSE values for Z_0^1.

Figure A.1: RMSE values are plotted for Example 3.4.2 using different learning rate approaches, where $T = 0.25$ and $b = 25$.

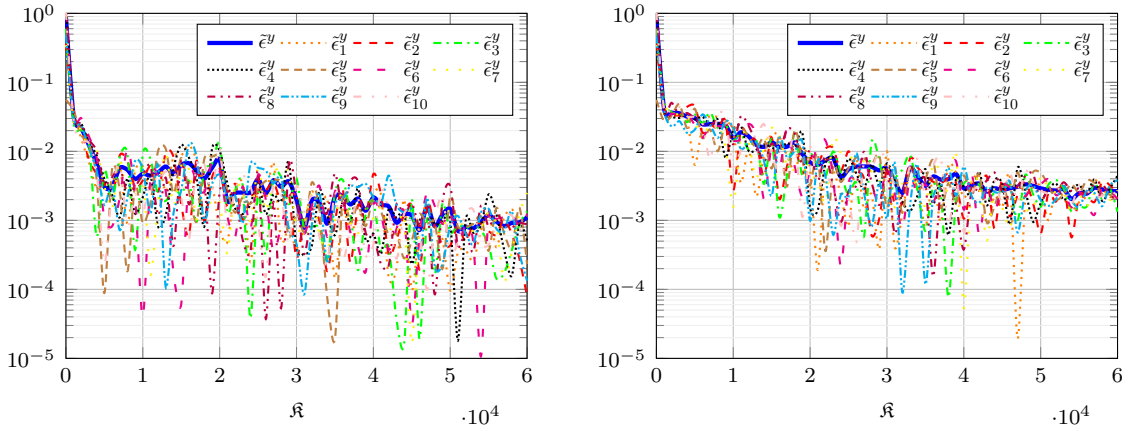
errors over time are displayed in Figure A.3.



(a) RMSE values for Y_0.

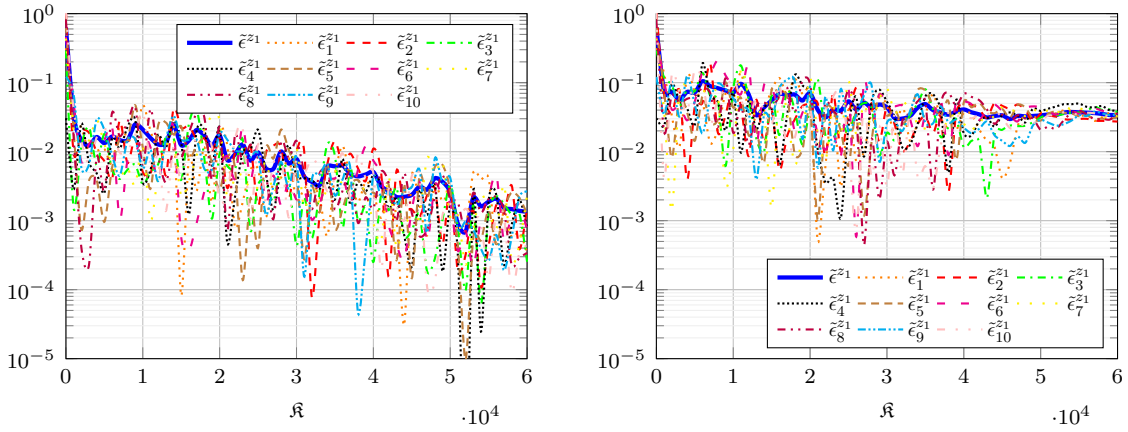
(b) RMSE values for Z_0^1.

Figure A.2: RMSE values are plotted for Example 3.4.2 using $N \in \{2, 8, 32, 128, 256, 512, 1024\}$, where $T = 0.25$ and $b = 25$.



(a) RMSE values and the absolute error for each of $Q = 10$ DBSDE runs for Y_0 with $N = 32$.

(b) RMSE values and the absolute error for each of $Q = 10$ DBSDE runs for Y_0 with $N = 1024$.



(c) RMSE values and the absolute error for each of $Q = 10$ DBSDE runs for Z_0^1 with $N = 32$.

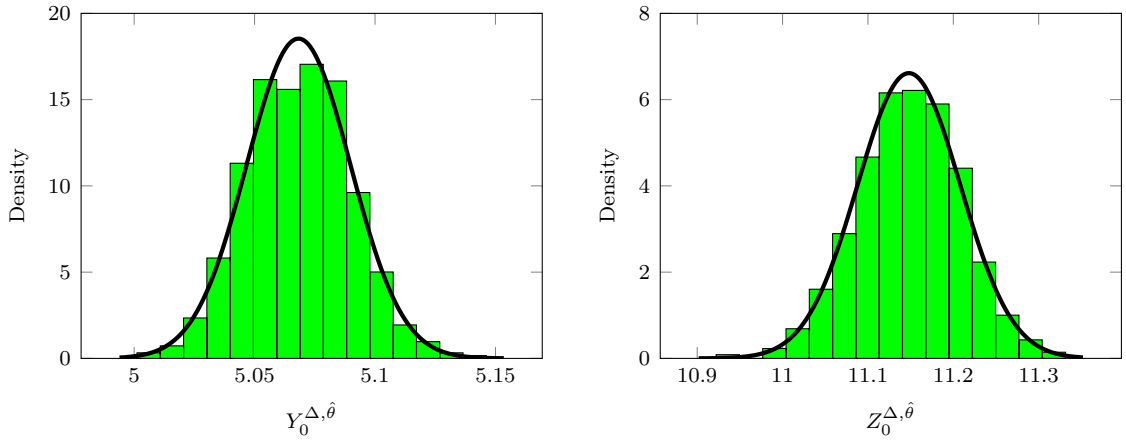
(d) RMSE values and the absolute error for each of $Q = 10$ DBSDE runs for Z_0^1 with $N = 1024$.

Figure A.3: RMSE values and the absolute errors from each of $Q = 10$ DBSDE runs are plotted for Example 3.4.2 using $N \in \{32, 1024\}$, where $T = 0.25$ and $b = 25$.

Appendix B

Normality assumption of the error distribution

In this section, we conduct a test to assess the normality of the error distribution in (3.10) for Example 3.4.1. For the parameter values $T = 0.33, K = 100, S_0 = 100, a = 0.05, b = 0.2, R = 0.03$ and $\delta = 0$, the exact solution is $(Y_0, Z_0) = (5.0679, 11.1420)$. Using $N = 16$, $\mathfrak{K} = 30000$, $\alpha = 1\text{e}{-2}$ and conducting $Q = 1280$ independent runs of the DBSDE algorithm, we display the empirical distribution of the approximations in Figure B.1. The observed empirical distributions



(a) $Y_0^{\Delta,\hat{\theta}}$, where $(\tilde{\mu}^y, \tilde{\sigma}^y) = (5.0682, 0.0215)$.

(b) $Z_0^{\Delta,\hat{\theta}}$, where $(\tilde{\mu}^z, \tilde{\sigma}^z) = (11.1480, 0.0603)$.

Figure B.1: Empirical distribution of the approximate solution (3.10) in Example 3.4.1 for parameter set $T = 0.33, K = 100, S_0 = 100, a = 0.05, b = 0.2, R = 0.03$ and $\delta = 0$. The curve represents a fitted normal distribution to the data.

exhibit a Gaussian shape. To further assess the normality, we perform the Shapiro-Wilk [91] and D'Agostino and Pearson's [20] tests for the assessment of normality. The p-values obtained from these tests are presented in Table B.1. With a significance level of 0.05, we conclude that there is no evidence to reject the assumption of normal distribution for the approximate solutions $Y_0^{\Delta,\hat{\theta}}$ and $Z_0^{\Delta,\hat{\theta}}$.

	Shapiro-Wilk	D'Agostino-Pearson
$Y_0^{\Delta,\hat{\theta}}$	0.2975	0.4794
$Z_0^{\Delta,\hat{\theta}}$	0.3957	0.2435

Table B.1: p-value of the statistical tests in Example 3.4.1 for parameter set $T = 0.33, K = 100, S_0 = 100, a = 0.05, b = 0.2, R = 0.03$ and $\delta = 0$.

Bibliography

[1] L. Abbas-Turki, S. Crépey, B. Saadeddine, and W. Sabbagh, *Pathwise XVAs: The Direct Scheme*, (2022), https://perso.lpsm.paris/~crepey/papers/MABSDE.pdf.

[2] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi, *A review of uncertainty quantification in deep learning: Techniques, applications and challenges*, Inf. Fusion, 76 (2021), pp. 243–297, https://doi.org/10.1016/j.inffus.2021.05.008.

[3] K. Andersson, A. Andersson, and C. W. Oosterlee, *Convergence of a Robust Deep FBSDE Method for Stochastic Control*, SIAM J. Sci. Comput., 45 (2023), pp. A226–A255, https://doi.org/10.1137/22M1478057.

[4] S. Ankirchner, C. Blanchet-Scalliet, and A. Eyraud-Loisel, *Credit risk premia and quadratic BSDEs with a single jump*, Int. J. Theor. Appl. Finance., 13 (2010), pp. 1103–1129, https://doi.org/10.1142/s0219024910006133.

[5] S. L. Beal and L. B. Sheiner, *Heteroscedastic nonlinear regression*, Technometrics, 30 (1988), pp. 327–338, https://doi.org/10.1080/00401706.1988.10488406.

[6] C. Beck, S. Becker, P. Cheridito, A. Jentzen, and A. Neufeld, *Deep Splitting Method for Parabolic PDEs*, SIAM J. Sci. Comput., 43 (2021), pp. A3135–A3154, https://doi.org/10.1137/19M1297919.

[7] C. Beck, A. Jentzen, and B. Kuckuck, *Full error analysis for the training of deep neural networks*, Infin. Dimens. Anal. Quantum Probab. Relat. Top., 25 (2022), p. 2150020, https://doi.org/10.1142/S021902572150020X.

[8] S. Becker, R. Braunwarth, M. Hutzenthaler, A. Jentzen, and P. von Wurstemberger, *Numerical Simulations for Full History Recursive Multilevel Picard Approximations for Systems of High-Dimensional Partial Differential Equations*, Commun. Comput. Phys., 28 (2020), pp. 2109–2138, https://doi.org/10.4208/cicp.OA-2020-0130.

[9] C. Bender and J. Zhang, *Time discretization and Markovian iteration for coupled FBSDEs*, Ann. Appl. Probab., 18 (2008), pp. 143–177, https://doi.org/10.1214/07-aap448.

[10] Y. Z. Bergman, *Option Pricing with Differential Interest Rates*, Rev. Financ. Stud., 8 (1995), pp. 475–500, https://doi.org/10.1093/rfs/8.2.475.

[11] B. Bouchard and N. Touzi, *Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations*, Stoch. Process Their Appl., 111 (2004), pp. 175–206, https://doi.org/10.1016/j.spa.2004.01.001.

[12] Q. Chan-Wai-Nam, J. Mikael, and X. Warin, *Machine learning for semi linear PDEs*, J. Sci. Comput., 79 (2019), pp. 1667–1712, https://doi.org/10.1007/s10915-019-00908-3.

[13] J.-F. Chassagneux, J. Chen, N. Frikha, and C. Zhou, *A learning scheme by sparse grids and Picard approximations for semilinear parabolic PDEs*, IMA J. Numer. Anal., 43 (2023), pp. 3109–3168, https://doi.org/10.1093/imanum/drac066.

[14] J.-F. Chassagneux and A. Richou, *Numerical simulation of quadratic BSDEs*, Ann. Appl. Probab., 26 (2016), pp. 262 – 304, https://doi.org/10.1214/14-AAP1090.

[15] Y. Chen and J. W. Wan, *Deep neural network framework based on backward stochastic differential equations for pricing and hedging American options in high dimensions*, Quant. Finance, 21 (2021), pp. 45–67, https://doi.org/10.1080/14697688.2020.1788219.

[16] P. Cheridito and K. Nam, *BSDEs with terminal conditions that have bounded Malliavin derivative*, J. Funct. Anal., 266 (2014), pp. 1257–1285, https://doi.org/10.1016/j.jfa.2013.12.004.

[17] J. Chessari, R. Kawai, Y. Shinozaki, and T. Yamada, *Numerical methods for backward stochastic differential equations: A survey*, Probab. Surv., 20 (2023), pp. 486–567, https://doi.org/10.1214/23-PS18.

[18] D. Crisan and K. Manolarakis, *Solving Backward Stochastic Differential Equations Using the Cubature Method: Application to Nonlinear Pricing*, SIAM J. Financial Math., 3 (2012), pp. 534–571, https://doi.org/10.1137/090765766.

[19] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Math. Control Signal Systems, 2 (1989), pp. 303–314, https://doi.org/10.1007/BF02551274.

[20] R. D'Agostino and E. S. Pearson, *Tests for departure from normality. Empirical results for the distributions of b^2 and $\sqrt{b^1}$*, Biometrika, 60 (1973), pp. 613–622, https://doi.org/10.1093/biomet/60.3.613.

[21] F. Delarue and S. Menozzi, *A forward–backward stochastic algorithm for quasi-linear PDEs*, Ann. Appl. Probab., 16 (2006), pp. 140 – 184, https://doi.org/10.1214/105051605000000674.

[22] W. E, J. Han, and A. Jentzen, *Deep Learning-Based Numerical Methods for High-Dimensional Parabolic Partial Differential Equations and Backward Stochastic Differential Equations*, Commun. Math. Stat., 5 (2017), pp. 349–380, https://doi.org/10.1007/s40304-017-0117-6.

[23] W. E, M. Hutzenthaler, A. Jentzen, and T. Kruse, *On Multilevel Picard Numerical Approximations for High-Dimensional Nonlinear Parabolic Partial Differential Equations and High-Dimensional Nonlinear Backward Stochastic Differential Equations*, J. Sci. Comput., 79 (2019), pp. 1534–1571, https://doi.org/10.1007/s10915-018-00903-0.

[24] W. E, M. Hutzenthaler, A. Jentzen, and T. Kruse, *Multilevel Picard iterations for solving smooth semilinear parabolic heat equations*, Partial Differ. Equ. Appl., 2 (2021), pp. 1–31, https://doi.org/10.1007/s42985-021-00089-5.

[25] A. Eyraud-Loisel, *Backward stochastic differential equations with enlarged filtration: Option hedging of an insider trader in a financial market with jumps*, Stoch. Process Their Appl., 115 (2005), pp. 1745–1763, https://doi.org/10.1016/j.spa.2005.05.006.

[26] A. Fahim, N. Touzi, and X. Warin, *A probabilistic numerical method for fully nonlinear parabolic PDEs*, Ann. Appl. Probab., 21 (2011), pp. 1322–1364, https://doi.org/10.1214/10-aap723.

[27] Y. Fu, W. Zhao, and T. Zhou, *Efficient spectral sparse grid approximations for solving multi-dimensional forward backward SDEs*, Discrete Contin. Dyn. Syst. - B, 22 (2017), pp. 3439–3458, https://doi.org/10.3934/dcdsb.2017174.

[28] M. Fujii, A. Takahashi, and M. Takahashi, *Asymptotic Expansion as Prior Knowledge in Deep Learning Method for High dimensional BSDEs*, Asia-Pac. Financ. Mark., 26 (2019), pp. 391–408, https://doi.org/10.1007/s10690-019-09271-7.

[29] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, in Proc. 33-rd Int. Conf. Mach. Learn., vol. 48, PMLR, 2016, pp. 1050–1059, https://proceedings.mlr.press/v48/gal16.html.

[30] M. Germain, H. Pham, and X. Warin, *Approximation Error Analysis of Some Deep Backward Schemes for Nonlinear PDEs*, SIAM J. Sci. Comput., 44 (2022), pp. A28–A56, https://doi.org/10.1137/20M1355355.

[31] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in Proc. AISTATS 13-th Int. Conf. Artif. Intell. Stat., vol. 9, PMLR, 2010, pp. 249–256, https://proceedings.mlr.press/v9/glorot10a.html.

[32] A. Gnoatto, M. Patacca, and A. Picarelli, *A deep solver for BSDEs with jumps*, 2022, https://arxiv.org/abs/2211.04349.

[33] A. Gnoatto, A. Picarelli, and C. Reisinger, *Deep xVA Solver: A Neural Network–Based Counterparty Credit Risk Management Framework*, SIAM J. Financial Math., 14 (2023), pp. 314–352, https://doi.org/10.1137/21M1457606.

[34] E. Gobet and C. Labart, *Solving BSDE with Adaptive Control Variate*, SIAM J. Numer. Anal., 48 (2010), pp. 257–277, https://doi.org/10.1137/090755060.

[35] E. Gobet, J.-P. Lemor, and X. Warin, *A regression-based Monte Carlo method to solve backward stochastic differential equations*, Ann. Appl. Probab., 15 (2005), pp. 2172–2202, https://doi.org/10.1214/105051605000000412.

[36] E. Gobet, J. G. López-Salas, P. Turkedjiev, and C. Vázquez, *Stratified Regression Monte-Carlo Scheme for Semilinear PDEs and BSDEs with Large Scale Parallelization on GPUs*, SIAM J. Sci. Comput., 38 (2016), pp. C652–C677, https://doi.org/10.1137/16m106371x.

[37] E. Gobet and P. Turkedjiev, *Linear regression MDP scheme for discrete backward stochastic differential equations under general conditions*, Math. Comp., 85 (2016), pp. 1359–1391, https://doi.org/10.1090/mcom/3013.

[38] J. Han, A. Jentzen, and W. E, *Solving high-dimensional partial differential equations using deep learning*, Proc. Natl. Acad. Sci. U.S.A., 115 (2018), pp. 8505–8510, https://doi.org/10.1073/pnas.1718942115.

[39] J. Han and J. Long, *Convergence of the deep BSDE method for coupled FBSDEs*, Probab. Uncertain. Quant. Risk, 5 (2020), https://doi.org/10.1186/s41546-020-00047-w.

[40] D. Hendrycks and K. Gimpel, *A baseline for detecting misclassified and out-of-distribution examples in neural networks*, 2018, https://arxiv.org/abs/1610.02136.

[41] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators*, Neural Netw., 2 (1989), pp. 359–366, https://doi.org/10.1016/0893-6080(89)90020-8.

[42] B. Huge and A. Savine, *Differential Machine Learning*, 2020, https://arxiv.org/abs/2005.02347.

[43] E. Hüllermeier and W. Waegeman, *Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods*, Mach. Learn., 110 (2021), pp. 457–506, https://doi.org/10.1007/s10994-021-05946-3.

[44] C. Huré, H. Pham, and X. Warin, *Deep backward schemes for high-dimensional nonlinear PDEs*, Math. Comput., 89 (2020), pp. 1547–1579, https://doi.org/10.1090/mcom/3514.

[45] M. Hutzenthaler, A. Jentzen, and T. Kruse, *Overcoming the curse of dimensionality in the numerical approximation of parabolic partial differential equations with gradient-dependent nonlinearities*, Found. Comput. Math., 22 (2022), pp. 905–966, https://doi.org/10.1007/s10208-021-09514-y.

[46] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen, *Overcoming the curse of dimensionality in the numerical approximation of backward stochastic differential equations*, J. Numer. Math., 31 (2023), pp. 1–28, https://doi.org/10.1515/jnma-2021-0111.

[47] M. Hutzenthaler, A. Jentzen, T. Kruse, T. A. Nguyen, and P. von Wurstemberger, *Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations*, Proc. R. Soc. A., 476 (2020), p. 20190630, https://doi.org/10.1098/rspa.2019.0630.

[48] M. Hutzenthaler and T. Kruse, *Multilevel Picard approximations of high-dimensional semilinear parabolic differential equations with gradient-dependent nonlinearities*, SIAM J. Numer. Anal., 58 (2020), pp. 929–961, https://doi.org/10.1137/17M1157015.

[49] P. Imkeller and G. D. Reis, *Path regularity and explicit convergence rate for BSDE with truncated quadratic growth*, Stoch. Process Their Appl., 120 (2010), pp. 348–379, https://doi.org/10.1016/j.spa.2009.11.004.

[50] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in International conference on machine learning, vol. 37, PMLR, 2015, pp. 448–456, https://proceedings.mlr.press/v37/ioffe15.html.

[51] A. Jentzen and A. Riekert, *Strong overall error analysis for the training of artificial neural networks via random initializations*, Commun. Math. Stat., (2023), https://doi.org/10.1007/s40304-022-00292-9.

[52] S. Ji, S. Peng, Y. Peng, and X. Zhang, *Three Algorithms for Solving High-Dimensional Fully Coupled FBSDEs Through Deep Learning*, IEEE Intell. Syst., 35 (2020), pp. 71–84, https://doi.org/10.1109/MIS.2020.2971597.

[53] S. Ji, S. Peng, Y. Peng, and X. Zhang, *A novel control method for solving high-dimensional Hamiltonian systems through deep neural networks*, 2021, https://arxiv.org/abs/2111.02636.

[54] S. Ji, S. Peng, Y. Peng, and X. Zhang, *A deep learning method for solving stochastic optimal control problems driven by fully-coupled FBSDEs*, 2022, https://arxiv.org/abs/2204.05796.

[55] H. Jiang, B. Kim, M. Guan, and M. Gupta, *To trust or not to trust a classifier*, Adv. Neural Inf. Process. Syst., 31 (2018), https://proceedings.neurips.cc/paper_files/paper/2018/file/7180cffd6a8e829dacfc2a31b3f72ece-Paper.pdf.

[56] Y. Jiang and J. Li, *Convergence of the Deep BSDE method for FBSDEs with non-Lipschitz coefficients*, Probab. Uncertain. Quant. Risk, 6 (2021), pp. 391–408, https://doi.org/10.3934/puqr.2021019.

[57] L. Kapllani, *The Effect of the Number of Neural Networks on Deep Learning Schemes for Solving High Dimensional Nonlinear Backward Stochastic Differential Equations*, in M. Ehrhardt, M. Günther (eds) Progress in Industrial Mathematics at ECMI 2021. ECMI 2021. Mathematics in Industry(), vol. 39, Springer, Cham, 2022, https://doi.org/10.1007/978-3-031-11818-0_10.

[58] L. Kapllani and L. Teng, *Multistep schemes for solving backward stochastic differential equations on GPU*, J. Math. Ind., 12 (2022), https://doi.org/10.1186/s13362-021-00118-3.

[59] L. Kapllani and L. Teng, *A backward differential deep learning-based algorithm for solving high-dimensional nonlinear backward stochastic differential equations*, 2024, https://arxiv.org/abs/2404.08456.

[60] L. Kapllani and L. Teng, *Deep learning algorithms for solving high-dimensional nonlinear backward stochastic differential equations*, Discrete Contin. Dyn. Syst. - B, 29 (2024), pp. 1695–1729, https://doi.org/10.3934/dcdsb.2023151.

[61] L. Kapllani, L. Teng, and M. Rottmann, *Uncertainty quantification for deep learning-based schemes for solving high-dimensional backward stochastic differential equations*, Int. J. Uncertain. Quantif., 15 (2025), pp. 55–94, https://doi.org/10.1615/Int.J.UncertaintyQuantification.2024053491.

[62] N. E. Karoui, S. Peng, and M. C. Quenez, *Backward Stochastic Differential Equations in Finance*, Math. Financ., 7 (1997), pp. 1–71, https://doi.org/10.1111/1467-9965.00022.

[63] A. Kendall and Y. Gal, *What uncertainties do we need in bayesian deep learning for computer vision?*, Adv. Neural Inf. Process. Syst., 30 (2017), https://proceedings.neurips.cc/paper_files/paper/2017/file/2650d6089a6d640c5e85b2b88265dc2b-Paper.pdf.

[64] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017, https://arxiv.org/abs/1412.6980.

[65] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Springer, 2013, https://doi.org/10.1007/978-3-662-12616-5.

[66] C. Knochenhauer, O. Hager, C. Reimers, L. Schnell, F. T. Seifried, and M. Würschmidt, *Convergence Rates for a Deep Learning Algorithm for Semilinear PDEs*, Available at SSRN, (2021), https://ssrn.com/abstract=3981933.

[67] S. Kremsner, A. Steinicke, and M. Szölgyenyi, *A Deep Neural Network Algorithm for Semilinear Elliptic PDEs with Applications in Insurance Mathematics*, Risks, 8 (2020), p. 136, https://doi.org/10.3390/risks8040136.

[68] M. A. Kupinski, J. W. Hoppin, E. Clarkson, and H. H. Barrett, *Ideal-observer computation in medical imaging with use of Markov-chain Monte Carlo techniques*, J. Opt. Soc. Am. A, 20 (2003), pp. 430–438, https://doi.org/10.1364/JOSAA.20.000430.

[69] C. Labart and J. Lelong, *A Parallel Algorithm for solving BSDEs-Application to the pricing and hedging of American options*, 2011, https://arxiv.org/abs/1102.4666v1.

[70] B. Lakshminarayanan, A. Pritzel, and C. Blundell, *Simple and scalable predictive uncertainty estimation using deep ensembles*, Adv. Neural Inf. Process. Syst., 30 (2017), https://proceedings.neurips.cc/paper_files/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.

[71] W. Lefebvre, G. Loeper, and H. Pham, *Differential learning methods for solving fully nonlinear PDEs*, Digit. Finance, 5 (2023), pp. 183–229, https://doi.org/10.1007/s42521-023-00077-x.

[72] J.-P. Lemor, E. Gobet, and X. Warin, *Rate of convergence of an empirical regression method for solving generalized backward stochastic differential equations*, Bernoulli, 12 (2006), pp. 889–916, https://doi.org/10.3150/bj/1161614951.

[73] J. Liang, Z. Xu, and P. Li, *Deep learning-based least squares forward-backward stochastic differential equation solver for high-dimensional derivative pricing*, Quant. Finance, 21 (2021), pp. 1309–1323, https://doi.org/10.1080/14697688.2021.1881149.

[74] J. Ma, J. Shen, and Y. Zhao, *On Numerical Approximations of Forward-Backward Stochastic Differential Equations*, SIAM J. Numer. Anal., 46 (2008), pp. 2636–2661, https://doi.org/10.1137/06067393x.

[75] A. Mobiny, P. Yuan, S. K. Moulik, N. Garg, C. C. Wu, and H. V. Nguyen, *Dropconnect is effective in modeling uncertainty of bayesian deep networks*, Sci. Rep., 11 (2021), p. 5458, https://doi.org/10.1038/s41598-021-84854-x.

[76] B. Negyesi, K. Andersson, and C. W. Oosterlee, *The One Step Malliavin scheme: new discretization of BSDEs implemented with deep learning regressions*, IMA J. Numer. Anal., (2024), p. drad092, https://doi.org/10.1093/imanum/drad092.

[77] B. Negyesi, Z. Huang, and C. W. Oosterlee, *Generalized convergence of the deep BSDE method: a step towards fully-coupled FBSDEs and applications in stochastic control*, 2024, https://arxiv.org/abs/2403.18552.

[78] D. A. Nix and A. S. Weigend, *Estimating the mean and variance of the target probability distribution*, in Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94), vol. 1, IEEE, 1994, pp. 55–60, https://doi.org/10.1109/ICNN.1994.374138.

[79] D. Nualart, *The Malliavin Calculus and Related Topics*, vol. 1995, Springer, 2006, https://doi.org/10.1007/3-540-28329-3.

[80] P. Oberdiek, G. Fink, and M. Rottmann, *UQGAN: A Unified Model for Uncertainty Quantification of Deep Classifiers trained via Conditional GANs*, Adv. Neural Inf. Process. Syst., 35 (2022), pp. 21371–21385, https://proceedings.neurips.cc/paper_files/paper/2022/file/8648e249887ccb0fe8c067d596e35b40-Paper-Conference.pdf.

[81] P. Oberdiek, M. Rottmann, and H. Gottschalk, *Classification uncertainty of deep neural networks based on gradient information*, in L. Pancioni, F. Schwenker, E. Trentin (eds) Artificial Neural Networks in Pattern Recognition. ANNPR 2018. Lecture Notes in Computer Science(), vol. 11081, Springer, Cham, 2018, https://doi.org/10.1007/978-3-319-99978-4_9.

[82] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, *Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift*, Adv. Neural Inf. Process. Syst., 32 (2019), https://proceedings.neurips.cc/paper_files/paper/2019/file/8558cb408c1d76621371888657d2eb1d-Paper.pdf.

[83] E. Pardoux and S. Peng, *Adapted solution of a backward stochastic differential equation*, Syst. Control. Lett., 14 (1990), pp. 55–61, https://doi.org/10.1016/0167-6911(90)90082-6.

[84] M. Pereira, Z. Wang, I. Exarchos, and E. A. Theodorou, *Learning Deep Stochastic Optimal Control Policies using Forward-Backward SDEs*, 2021, https://arxiv.org/abs/1902.03986.

[85] H. Pham, X. Warin, and M. Germain, *Neural networks-based backward scheme for fully nonlinear PDEs*, SN Partial Differ. Equ. Appl., 2 (2021), https://doi.org/10.1007/s42985-020-00062-8.

[86] M. Raissi, *Forward–backward stochastic neural networks: deep learning of high-dimensional partial differential equations*, in Peter Carr Gedenkschrift: Research Advances in Mathematical Finance, World Scientific, 2024, pp. 637–655, https://doi.org/10.1142/9789811280306_0018.

[87] M. Ruijter and C. Oosterlee, *Numerical Fourier method and second-order Taylor scheme for backward SDEs in finance*, Appl. Numer. Math., 103 (2016), pp. 1–26, https://doi.org/10.1016/j.apnum.2015.12.003.

[88] M. J. Ruijter and C. W. Oosterlee, *A Fourier Cosine Method for an Efficient Computation of Solutions to BSDEs*, SIAM J. Sci. Comput., 37 (2015), pp. A859–A889, https://doi.org/10.1137/130913183.

[89] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, Nature, 323 (1986), pp. 533–536, https://doi.org/10.1038/323533a0.

[90] A. M. Schäfer and H. G. Zimmermann, *Recurrent neural networks are universal approximators*, In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds) Artificial Neural Networks – ICANN 2006. ICANN 2006. Lecture Notes in Computer Science, 4131 (2006), pp. 632–640, https://doi.org/10.1007/11840817_66.

[91] S. S. Shapiro and M. B. Wilk, *An analysis of variance test for normality (complete samples)*, Biometrika, 52 (1965), pp. 591–611, https://doi.org/10.2307/2333709.

[92] A. Takahashi, Y. Tsuchida, and T. Yamada, *A new efficient approximation scheme for solving high-dimensional semilinear PDEs: Control variate method for Deep BSDE solver*, J. Comput. Phys., 454 (2022), p. 110956, https://doi.org/10.1016/j.jcp.2022.110956.

[93] L. Teng, *A Review of Tree-Based Approaches to Solving Forward–Backward Stochastic Differential Equations*, J. Comput. Finance, 25 (2021), https://doi.org/10.21314/JCF.2021.010.

[94] L. Teng, *Gradient boosting-based numerical methods for high-dimensional backward stochastic differential equations*, Appl. Math. Comput., 426 (2022), p. 127119, https://doi.org/10.1016/j.amc.2022.127119.

[95] L. Teng, A. Lapitckii, and M. Günther, *A multi-step scheme based on cubic spline for solving backward stochastic differential equations*, Appl. Numer. Math., 150 (2020), pp. 117–138, https://doi.org/10.1016/j.apnum.2019.09.016.

[96] L. Teng and W. Zhao, *High-order combined multi-step scheme for solving forward backward stochastic differential equations*, J. Sci. Comput., 87 (2021), https://doi.org/10.1007/s10915-021-01505-z.

[97] H. Wang, H. Chen, A. Sudjianto, R. Liu, and Q. Shen, *Deep Learning-Based BSDE Solver for Libor Market Model with Application to Bermudan Swaption Pricing and Hedging*, 2018, https://arxiv.org/abs/1807.06622.

[98] Z. Wang and S. Tang, *Gradient Convergence of Deep Learning-Based Numerical Methods for BSDEs*, Chin. Ann. Math., B, 42 (2021), pp. 199–216, https://doi.org/10.1007/s11401-021-0253-x.

[99] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, *Flipout: Efficient pseudo-independent weight perturbations on mini-batches*, 2018, https://arxiv.org/abs/1803.04386.

[100] G. Zhang, *A Sparse-Grid Method for Multi-Dimensional Backward Stochastic Differential Equations*, J. Comput. Math., 31 (2013), pp. 221–248, https://doi.org/10.4208/jcm.1212-m4014.

[101] J. Zhang, *A numerical scheme for BSDEs*, Ann. Appl. Probab., 14 (2004), pp. 459–488, https://doi.org/10.1214/aoap/1075828058.

[102] J. Zhang, *Backward Stochastic Differential Equations*, Springer, 2017, https://doi.org/10.1007/978-1-4939-7256-2.

[103] W. Zhao, L. Chen, and S. Peng, *A New Kind of Accurate Numerical Method for Backward Stochastic Differential Equations*, SIAM J. Sci. Comput., 28 (2006), pp. 1563–1581, https://doi.org/10.1137/05063341x.

[104] W. Zhao, Y. Fu, and T. Zhou, *New Kinds of High-Order Multistep Schemes for Coupled Forward Backward Stochastic Differential Equations*, SIAM J. Sci. Comput., 36 (2014), pp. A1731–A1751, https://doi.org/10.1137/130941274.

[105] W. Zhao, G. Zhang, and L. Ju, *A Stable Multistep Scheme for Solving Backward Stochastic Differential Equations*, SIAM J. Numer. Anal., 48 (2010), pp. 1369–1394, https://doi.org/10.1137/09076979x.