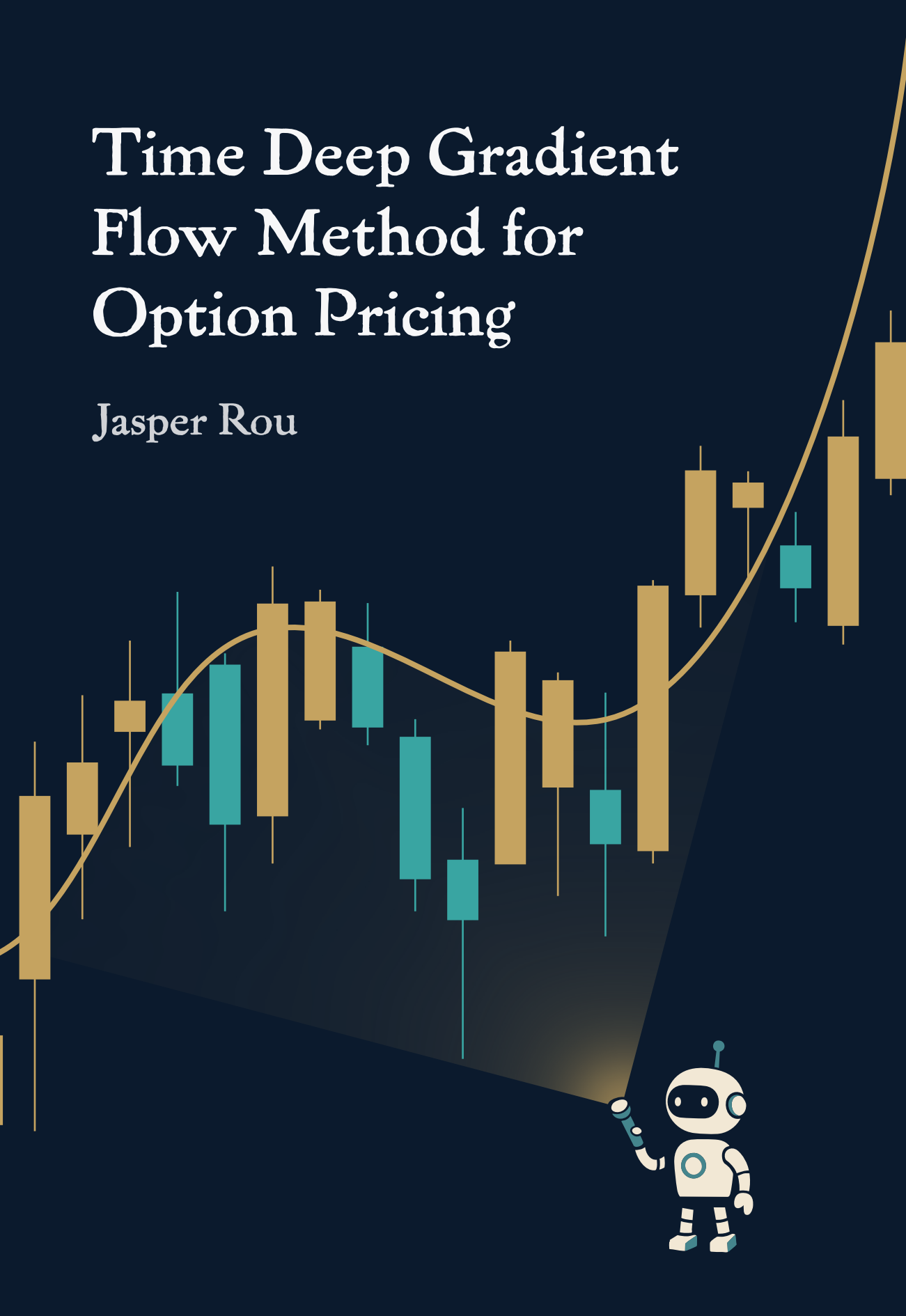
Time Deep Gradient Flow Method for Option Pricing

Jasper Rou



Time Deep Gradient Flow Method for Option Pricing

Time Deep Gradient Flow Method for Option Pricing

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen;
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
13 januari 2025 om 15:00 uur

door

Jasper Gijsbert ROU

Master of Science in Applied Mathematics, TU Delft, Nederland
geboren te Veenendaal, Nederland

Dit proefschrift is goedgekeurd door de promotoren.

Samenstelling promotiecommissie bestaat uit:

Rector magnificus,	voorzitter
Prof. dr. A. Papapantoleon	TU Delft, *promotor*
Prof. dr. F.H.J. Redig	TU Delft, *promotor*
onafhankelijke leden:	
Prof. dr. ir. M.C. Veraar	TU Delft
Prof. dr. ir. C.W. Oosterlee	Universiteit Utrecht
Prof. dr. M. Rosenbaum	École Polytechnique
Dr. C. Bayer	Weierstrass Institute
Prof. dr. ir. G. Jongbloed	TU Delft, *reservelid*

The research in this dissertation was funded by a DIAM Fast Track PhD Scholarship.

Printed by: ProefschriftMaken

Cover by: Sterre Lutz

ISBN 978-94-6534-090-6

An electronic copy of this dissertation is available at
`https://repository.tudelft.nl/`.

Of course! I can help writing a PhD dissertation.

ChatGPT

Contents

Summary

Pricing financial derivatives such as options is a key challenge in financial mathematics. A common approach is to express the option price as the solution to a partial differential equation (PDE). As option pricing models become increasingly complicated, the resulting PDEs become high-dimensional, rendering classical numerical methods computationally infeasible. Recent advances suggest that deep learning may offer efficient alternatives for solving such problems.

This dissertation develops a novel deep learning approach for pricing options, which can efficiently handle high-dimensional problems. We call our method the Time Deep Gradient Flow (TDGF). The TDGF reformulates the PDE as a time-discretized variational problem, which is then approximated using deep neural networks. We focus on three option pricing models: the Black–Scholes model with constant volatility; the Heston model with stochastic volatility and the lifted Heston model, a Markovian approximation of rough volatility. We compare our method in these models with classical numerical methods such as Monte Carlo and Fourier methods and another deep learning method called the Deep Galerkin Method (DGM).

In Chapter 3, we develop the TDGF for pricing European options in diffusion models resulting from Markovian approximations of rough volatility models. The option pricing PDE is reformulated as an energy minimization problem, which is approximated in a time stepping fashion by deep neural networks. The proposed scheme respects the asymptotic behavior of option prices for large levels of moneyness and adheres to a priori known bounds for option prices. A series of numerical examples assesses the accuracy and efficiency of the proposed method, with particular focus in the lifted Heston model.

Chapter 4 extends the TDGF to pricing American put options in up to five dimensions under the Black—Scholes and Heston model. We modify the TDGF method to handle the free-boundary partial differential equation inherent in American options. We carefully design the sampling strategy during training to enhance performance. Our results demonstrate that both TDGF and DGM achieve high accuracy while significantly outperforming conventional Monte Carlo methods in terms of computational speed. In particular, TDGF tends to be faster during training than DGM.

In Chapter 5, we consider the convergence of the TDGF. We prove two things. First, there exists a neural network that converges to the solution of the PDE. This proof consists of three parts: convergence of the time stepping; equivalence of the solution of the discretized PDE and the minimizer of the variational formulation and the approximation of the minimizer by a neural network by using a version of the universal approximation theorem. Second, we prove that when training the network we converge to the correct solution. This proof consists of two parts: as the number of neurons goes to infinity, we converge to some gradient flow, and as

the training time goes to infinity, this gradient flow converges to the solution.

In Chapter 6, we verify empirically the results from the previous chapter by showing that using more resources leads to a smaller error. Further, we analyze the practical performance of TDGF and DGM in solving PDEs in the Black–Scholes and Heston models. We determine their empirical convergence rates and training time as functions of (i) the number of layers, (ii) the number of nodes per layer, (iii) the number of epochs, and (iv) the number of samples in each epoch. For the TDGF, we also consider the order of the discretization scheme and the number of time steps.

Samenvatting

Het waarderen van financiële derivaten zoals opties is een belangrijke uitdaging binnen de financiële wiskunde. Een gangbare benadering is om de optieprijs uit te drukken als de oplossing van een partiële differentiaalvergelijking (PDV). Naarmate optieprijsmodellen steeds complexer worden, worden de bijbehorende PDV's hoog-dimensionaal, waardoor klassieke numerieke methoden computationeel onhaalbaar worden. Recente ontwikkelingen suggereren dat deep learning efficiënte alternatieven kan bieden voor het oplossen van dergelijke problemen.

Dit proefschrift ontwikkelt een nieuwe deep learning-benadering voor het waarderen van opties, die efficiënt met hoog-dimensionale problemen kan omgaan. We noemen onze methode de Time Deep Gradient Flow (TDGF). De TDGF herformuleert de PDV als een in de tijd gediscretiseerd variatieprobleem, dat vervolgens wordt benaderd met behulp van diepe neurale netwerken. We richten ons op drie optieprijsmodellen: het Black—Scholes–model met constante volatiliteit; het Heston-model met stochastische volatiliteit en het lifted Heston-model, een Markoviaanse benadering van ruwe volatiliteit. We vergelijken onze methode in deze modellen met klassieke numerieke methoden zoals Monte Carlo- en Fourier-methoden en met een andere deep learning-methode, de Deep Galerkin Method (DGM).

In Hoofdstuk 3 ontwikkelen we de TDGF voor het waarderen van Europese opties in diffusie modellen die voortkomen uit Markoviaanse benaderingen van modellen met ruwe volatiliteit. De PDV voor optieprijzen wordt herformuleerd als een energieminimalisatieprobleem, dat stap voor stap in de tijd wordt benaderd met diepe neurale netwerken. Het voorgestelde schema respecteert het asymptotische gedrag van optieprijzen bij hoge niveaus van moneyness en houdt rekening met vooraf bekende grenzen voor optieprijzen. Een reeks numerieke voorbeelden evalueert de nauwkeurigheid en efficiëntie van de voorgestelde methode, met bijzondere aandacht voor het lifted Heston-model.

Hoofdstuk 4 breidt de TDGF uit naar het waarderen van Amerikaanse putopties in maximaal vijf dimensies binnen het Black-–Scholes- en Heston-model. We passen de TDGF-methode aan om de vrijrand-PDV die eigen is aan Amerikaanse opties te kunnen hanteren. We ontwerpen zorgvuldig de samplingstrategie tijdens het trainen om de prestaties te verbeteren. Onze resultaten tonen aan dat zowel TDGF als DGM een hoge nauwkeurigheid behalen, terwijl ze conventionele Monte Carlo-methoden aanzienlijk overtreffen qua rekensnelheid. In het bijzonder blijkt TDGF sneller te trainen dan DGM.

In Hoofdstuk 5 behandelen we de convergentie van de TDGF. We bewijzen twee zaken. Ten eerste bestaat er een neuraal netwerk dat convergeert naar de oplossing van de PDV. Dit bewijs bestaat uit drie delen: de convergentie van de tijdsstappen; de equivalentie van de oplossing van de gediscretiseerde PDV en de minimizer van de

variatieformulering; en de benadering van deze minimizer door een neuraal netwerk met behulp van een versie van de universele benaderingstelling. Ten tweede bewijzen we dat het trainen van het netwerk leidt tot de correcte oplossing. Dit bewijs bestaat uit twee delen: als het aantal neuronen naar oneindig gaat, convergeren we naar een bepaalde gradient flow, en als de trainingstijd naar oneindig gaat, convergeert deze gradient flow naar de oplossing.

In Hoofdstuk 6 verifiëren we empirisch de resultaten uit het vorige hoofdstuk door aan te tonen dat het gebruik van meer middelen leidt tot een kleinere fout. Daarnaast analyseren we de praktische prestaties van TDGF en DGM bij het oplossen van PDV's in het Black—Scholes- en Heston-model. We bepalen hun empirische convergentiesnelheden en trainingstijden als functie van (i) het aantal lagen, (ii) het aantal knooppunten per laag, (iii) het aantal epochs, en (iv) het aantal samples per epoch. Voor de TDGF beschouwen we ook de orde van het discretisatieschema en het aantal tijdsstappen.

1 Introduction

Every month, trillions of dollars of financial products are traded on exchanges around the world [86]. In addition to stocks or shares, you can also trade a contract that derives its value from the performance of an underlying product, called a derivative. Identifying derivative prices accurately and quickly is fundamental for the financial world.

1.1 Determining option prices

The most widely traded derivative in the financial markets is the stock option. This derivative gives the owner the right, but not the obligation, to buy or sell a stock at a specified price on or before a specified date. The specified price is the strike price K and the specified time is the maturity time T. If the contract gives the right to buy, it is a call option, while if it gives the right to sell, it is a put option.

Consider a call option at maturity time. If the stock is worth more than the strike price, we can exercise our right and buy the stock for the strike price. Then we make a profit that is equal to the difference between the stock and the strike price. On the other hand, if the stock is worth less than the strike price, we do not have any obligation to buy, and we do not exercise our right letting the stock expire worthless. Fig. 1.1 shows a plot of our payoff.

If there is still some time until maturity and the stock is worth less than the strike price, there is still a probability that the price of the stock increases above the strike price. Therefore, the option is still worth some amount of money. Since we cannot look into the future to see how stock prices evolve, it is impossible to determine exactly this amount. We can provide an educated guess with an accurate model of the option price.

1.2 Option pricing models

The first mathematical attempt to model of the price of derivatives was made by Bachelier [8] in 1900. He modeled the evolution of an asset as the sum of a deterministic term and a random term containing a Brownian motion. This model was revisited by Samuelson [80] in 1965 to prevent asset prices becoming negative. In

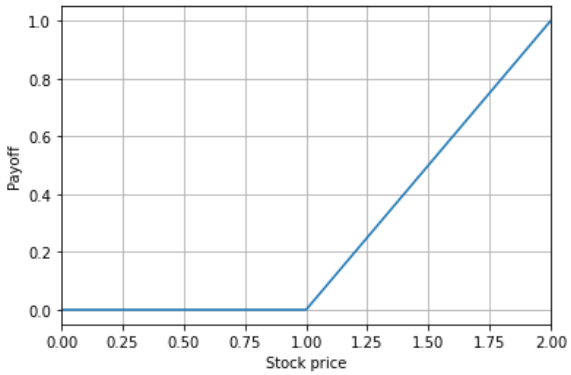


Figure 1.1: The value of an call option as a function of the underlying stock price at the maturity time with a strike price of 1.

his model, the price of an asset is a geometric Brownian motion, with as a drift the constant interest rate r and as a diffusion the constant volatility σ.

In 1973, Black and Scholes [23] showed an explicit formula for the price of an option in this model in terms of K, T, r, σ and the price of the stock. This model can, therefore, quickly provide an estimate of the option price and is still popular today under the name Black-Scholes model. However, looking at market prices of options on a single stock for different strike prices and maturity times, we need different volatilities to obtain accurate prices. Hence the volatility cannot be a constant for a single stock.

Therefore, other financial models have appeared in which the volatility is no longer constant but follows its own stochastic process. These models are called stochastic volatility models. A famous and widely used example is the model of Heston [49] from 1993. Compared to the Black-Scholes model, these models are more realistic but do not have an analytical solution. Recent research has shown that these stochastic volatility models do not yet completely accurately model the stock market. In the models, the stock price is too smooth compared to the wild behavior the real world market can show.

Recently, a new and even more complicated class of models emerged in which the volatility varies more rapidly than in the classical stochastic volatility models [40]. These models are called rough volatility models and show close agreement with market observations. The volatility can vary much since the Brownian motion driving it is replaced by a fractional Brownian motion. The drawback of this more realistic model is that it is not Markovian anymore: we cannot predict the future evolution of the stock anymore by just looking at its current state, but need to consider the whole past trajectory.

To address this problem, Abi Jaber and El Euch [3] designed Markovian approximations of rough volatility models. In these lifted models, the volatility is a linear combination of sub-volatilities, each evolving according to a regular Brownian motion. In this way, these models gain Markovianity, but at the cost of becoming

multidimensional. Therefore, there is a need for methods that can determine option prices in high-dimensional settings.

1.3 Solvers

Efficient numerical methods to rapidly price derivatives are a key issue in financial mathematics. Developed methods are among others Fourier methods, in which the option price is derived from the characteristic function of the stock price, and Monte Carlo methods, in which the expected payoff of the derivative is determined by sampling many paths. Another method is to write the price of an option as the solution to a partial differential equation (PDE), which can be solved using, for example, finite difference methods [71].

Unfortunately, all these conventional methods do not scale well for more complicated or high-dimensional problems. They discretize the space with grid points and to keep their accuracy, they need to use exponentially many grid points. Therefore, they become exponentially slow in higher dimensions. This phenomenon is called the curse of dimensionality.

A promising alternative is neural network algorithms. Neural networks are trained to match randomly sampled input variables with the correct output. Once a neural network finishes training, the output is an analytical function of the input. The price of an option for certain input variables can therefore be calculated fast. Furthermore, the number of input variables can be increased without making the network much slower. Therefore, neural network algorithms are suitable for high-dimensional problems.

Neural networks are becoming a key instrument in financial mathematics. Applications involve, among other things, optimal control [82], risk management [44], mean field games, and option pricing. Calculating option prices with neural networks can be done using a data-driven approach [65] or by solving the option pricing PDE. In the latter case, many different approaches have been developed [45]. One popular PDE method is the Deep Galerkin Method (DGM) [81], minimizing some residual error. The DGM is easy to implement and easy to generalize to different settings.

However, two major issues can be identified with the DGM. First, the choice of error type is arbitrary and is not guaranteed to be the best choice. Second, in higher-order PDEs the number of derivatives grows exponentially with the dimension causing the training of the network to become slow. Although Sirignano and Spiliopoulos [81] provide a Monte Carlo method for the fast computation of second derivatives, this solution is not elegant.

A big concern in machine learning is explainability. Neural network algorithms are often referred to as 'black-box' algorithms since it is unclear how they relate the input to the output. Especially in finance, practitioners are reluctant to use methods that they do not understand and are also not allowed to by regulators. Furthermore, there are few guarantees that the solution obtained is correct. General results in this regard are hard to obtain.

Much of the theoretical work on neural networks is based on the universal approx-

imation theorem, which says that neural networks can approximate any function on some bounded domain [52]. However, this theorem does not tell us anything about how to find this approximating neural network. Results in this direction are, for example, obtained by Jiang, Sirignano, and Cohen [60], who showed that the DGM is guaranteed to converge to the correct solution, and by Assabumrungrat, Minami, and Hirano [7], who empirically examined the accuracy of some deep PDE methods for option pricing.

1.4 Aim and scope

This study develops a new neural network method to solve PDEs in the context of option pricing. We call this method the Time Deep Gradient Flow method (TDGF).

The key idea of the TDGF is to reformulate the PDE as an energy minimization problem. This reformulation addresses the two problems related to the DGM. First, since it is already a minimization problem, we do not need to choose a norm. Second, during the reformulation the order of the derivatives reduces, therefore speeding up the training process.

These advantages come at the cost of an additional approximation. To reformulate the PDE, we discretize the PDE in time and split the operator in a symmetric and non-symmetric part. We then approximate the solution of this minimization problem with a neural network, in which we use information about the solution in our network architecture.

The scope of this study is two-fold. On the one hand, we study applications and show that our method can be used in many different settings. On the other hand, we analyze the accuracy of our method and show that using more computational resources leads to a better solution. Chapters 3 and 4 study the applications. In Chapter 3 we consider a simple derivative, but in a complicated model. In Chapter 4 we consider simpler models but more complicated derivatives. Chapters 5 and 6 analyze the accuracy. In Chapter 5, we consider a theoretical analysis of the accuracy of our method. In Chapter 6, we consider a numerical analysis of the accuracy of our method. A more detailed description of the rest of the thesis follows below.

1.5 Thesis outline

Chapter 2 begins by defining the mathematical background of the research. First, we present an overview of the different option pricing methods we consider in this dissertation: Monte Carlo, Fourier and PDEs. Second, we provide an overview of the different models for option pricing that we focus on in this research: Black-Scholes, Heston and lifted Heston.

Chapter 3 explains how to apply the TDGF in (rough) diffusion models. It introduces the key elements of the method and examples of how to apply the method in each of the different option pricing models. It explains the implementation details and architecture of the neural network as well. Furthermore, it provides numerical results for these different models, both in terms of accuracy and computation time.

Chapter 4 explains how to apply the TDGF in the context of American options. American options are a different variant of options that are harder to price. In this chapter, we explain how we can adapt the TDGF so that it can also solve the option price in the American case. We also apply the TDGF in a new multidimensional setting: the case in which we have more than one underlying stock.

Chapter 5 provides convergence results of the method. First, we show that we can find a neural network that approximates the option pricing PDE. In this process, we also provide a general version of the universal approximation theorem. Second, we provide global convergence proofs of the method in both network size and training time.

Chapter 6 provides an empirical investigation of the accuracy of the model. We consider an error analysis of the method with respect to the training and network parameters. We compare TDGF and DGM for the different parameters and do this comparison in different option pricing models.

2

Mathematical background

In this chapter, we provide the mathematical background necessary for the subsequent four chapters. In Section 2.1 we introduce the notation used throughout this dissertation. In Section 2.2 we explore the methods for pricing an option that exist in the literature. In Section 2.3 we explain the different option pricing models that we use throughout the dissertation.

2.1 Notation

Let $(\Omega, \mathcal{F}, \mathbb{P})$ denote a probability space. Let $\mathbb{F} = (\mathcal{F}_t)_{t\in[0,T]}$ denote a filtration of the sigma algebra $\mathcal{F}$. Let W denote a Brownian motion adapted to the filtration.

Definition 2.1. *Let (S, Σ) be a measurable space, and $X : [0, T] \times \Omega \to S$ be a stochastic process. The process X is adapted to the filtration $\mathbb{F}$ if the random variable $X_t : \Omega \to S$ is an $(\mathcal{F}_t, \Sigma)$-measurable function for all $t \in [0, T]$.*

For an arbitrary space $\mathcal{H}$, $\|\cdot\|_{\mathcal{H}}$ denotes the norm, $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ denotes the inner product, and $w_m \xrightarrow{\mathcal{H}} w$ denotes the weak convergence on this space. We abbreviate spaces and norms as $\mathcal{H} = \mathcal{H}(\mathbb{R}^d)$.

For $1 \leq p < \infty$, $L^p(\Omega)$ is the space of functions such that

$$\|f\|_{L^p(\Omega)} = \left(\int_\Omega |f(x)|^p \mathrm{d}x\right)^{\frac{1}{p}} < \infty.$$

Let $C_c^k\left(\mathbb{R}^d\right)$ denote the space of functions with compact support and continuous partial derivatives up to order k. For $1 \leq p < \infty$, $W_0^{k,p}(\Omega)$ is the Sobolev space with norm

$$\|f\|_{W_0^{k,p}(\Omega)} = \left(\sum_{|\alpha|\leq k} \int_\Omega |D^\alpha f(x)|^p \mathrm{d}x\right)^{\frac{1}{p}} < \infty,$$

with $D^\alpha f$ the weak derivative of f and α a multi-index. Let us introduce the shorthand notation $\mathcal{H}_0^k = W_0^{k,2}$ and let $\mathcal{H}^{-1}$ denote the dual space of $\mathcal{H}_0^1$.

Let $\mathcal{V} \subset \mathcal{H} \subset \mathcal{V}^*$ denote a Gelfand triple, in which $\mathcal{H}$ is a separable Hilbert space, $\mathcal{V}$ is a Banach space and $\mathcal{V}^*$ is the topological dual of $\mathcal{V}$.

Definition 2.2 (Self-adjoint operator)**.** *An operator $\mathcal{L} : \mathcal{V} \to \mathcal{V}^*$ is self-adjoint if*

$$\langle \mathcal{L}u, v\rangle_{\mathcal{V}^*,\mathcal{V}} = \langle \mathcal{L}v, u\rangle_{\mathcal{V}^*,\mathcal{V}} \quad \text{for all } \ u, v \in \mathcal{V}.$$

2

Remark 2.1. *The inner product $\langle \mathcal{L}u, v\rangle_{\mathcal{V}^*,\mathcal{V}}$ means that $\mathcal{L}u$ acts on v as a functional. An important example of this is: $\mathcal{V} = \mathcal{H}_0^1(\mathbb{R}^d)$, $\mathcal{V}^* = \mathcal{H}^{-1}(\mathbb{R}^d)$, $\mathcal{H} = L^2(\mathbb{R}^d)$, and $\mathcal{L} = -\Delta$, where Δ denotes the Laplace operator. Then, we define the functional $\mathcal{L}u$ as follows*

$$\langle \mathcal{L}u, v\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} = \langle -\Delta u, v\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} := \langle \nabla u, \nabla v\rangle_{L^2}.$$

Φ denotes the cumulative distribution function of the standard normal distribution:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} \mathrm{d}t.$$

For two $\mathbb{R}^{d\times d}$-valued matrices A and B, $\odot$ denotes the element-wise multiplication:

$$A \odot B = \left(a^{ij}\right)_{i,j=1,\dots,d} \odot \left(b^{ij}\right)_{i,j=1,\dots,d} = \left(a^{ij} b^{ij}\right)_{i,j=1,\dots,d} \in \mathbb{R}^{d\times d}.$$

For normed vector spaces V and W and U an open subset of V, the function $f : U \to W$ is Fréchet differentiable at $x \in U$ if there exists a bounded linear operator $A : V \to W$ such that

$$\lim_{\|h\|_V \to 0} \frac{\|f(x+h) - f(x) - Ah\|_W}{\|h\|_V} = 0.$$

If there exists such an operator A, it is unique and we denote by $\mathcal{D}f = A$ the Fréchet derivative $\mathcal{D}$ of f at x.

2.2 Option pricing

An option is a contract that gives the owner the right, but not the obligation, to buy or sell an asset at a specified price on or before a specified date. If the option gives the right to buy, it is a *call option*. If the option gives the right to sell, it is a *put option*. If the option can only be exercised at maturity, it is *European*. If the option can be exercised at any time before or at maturity, it is *American*. The specified price is the strike price and is denoted by K. The specified date is the maturity and is denoted by T.

If at exercise time, the asset S is higher than the strike price K, the owner can buy the asset for K and earn $S - K$. If at exercise time, the asset is worth less than the strike price K, the owner does not exercise and earn 0. This payoff is summarized in the function $\Psi_{\text{call}}(S) = (S - K)^+$. Similarly, for a put option, the payoff is $\Psi_{\text{put}}(S) = (K - S)^+$.

In an arbitrage-free market, there is no risk-free profit. Therefore, the price of a European option $u(t, S)$ that pays $\Psi(S)$ at maturity should at time t be equal to the

expected payoff under the equivalent martingale measure at maturity discounted at the risk-free rate r: $u(t,S) = \mathrm{e}^{-r(T-t)}\mathbb{E}\left[\Psi(S_T)|S_t = S\right]$.

In order to evaluate this expectation, we need to know the behavior of the stock. It is impossible to know the future behavior of the stock, but in general we can model it as a combination of a deterministic and a random part with an Itô process.

Definition 2.3. *An Itô process is an adapted stochastic process that can be written as the sum of an integral with respect to a Brownian motion and an integral with respect to time:*

$$X_t = X_0 + \int_0^t \mu_s \mathrm{d}s + \int_0^t \sigma_s \mathrm{d}W_s, \tag{2.1}$$

for an integrable process μ and a predictable and W-integrable process σ .

If $\sigma_t \in L^2([0,T]\times\Omega)$, then $\int_0^t \sigma_s \mathrm{d}W_s$ for $t \in [0,T]$ is a martingale with respect to the filtration $\mathbb{F}$ [62]. The differential form of equation (2.1) is

$$\mathrm{d}X_t = \mu_t \mathrm{d}t + \sigma_t \mathrm{d}W_t, \quad X_0 > 0. \tag{2.2}$$

2.2.1 Monte Carlo methods

A intuitive, easy to implement and widely used method is Monte Carlo, which relies on repeated simulation to obtain an estimate of the expectation. First, use a discretization in time, with K time steps: $\Delta t = \frac{T}{K}$, $\mu_k = \mu_{\frac{nT}{K}}$ and $\sigma_k = \sigma_{\frac{kT}{K}}$. Then we can estimate the trajectory of a stock S_t, with $S_0 = S$ by the Euler scheme,

$$\begin{aligned} \hat{S}_{k+1} &= \hat{S}_k + \mu_k \Delta t + \sigma_k \Delta W, \\ \hat{S}_0 &= S. \end{aligned}$$

Increments of a Brownian motion are normally distributed with mean 0 and variance Δt:

$$\Delta W \sim \mathcal{N}(0, \Delta t).$$

The Monte Carlo method samples M different paths $\hat{S}^m$ and approximates the option value by

$$u(0,S) = \mathrm{e}^{-rT}\mathbb{E}\left[\Psi(S_T)|S_0 = S\right] \approx \mathrm{e}^{-rT}\frac{1}{M}\sum_{m=1}^{M}\Psi\left(\hat{S}_K^m\right).$$

2.2.2 Characteristic functions

A second method of option pricing is to compute the conditional characteristic function of the stock price. We can then use the conditional characteristic function to compute option prices using Fourier-based methods such as the COS method [34]. For affine models, it is relatively easy to compute the conditional characteristic function.

Definition 2.4. *An $\mathbb{R}^d$-valued process $\mathbf{X}$ is affine if it has the dynamics*

$$\mathrm{d}\mathbf{X}_t = \beta(\mathbf{X}_t)\mathrm{d}t + \sigma(\mathbf{X}_t)\mathrm{d}\mathbf{W}_t,$$

with $\mathbf{W}_t$ a vector of independent Brownian motions, and

$$\begin{aligned}\beta(\mathbf{X}_t) &= b(t) + \beta_1 X_t^1 + ... + \beta_d X_t^d,\\ \sigma(\mathbf{X}_t)\sigma(\mathbf{X}_t)^\mathsf{T} &= a(t) + \alpha_1 X_t^1 + ... + \alpha_d X_t^d,\end{aligned}$$

for some $b, \beta_i \in \mathbb{R}^d$ and $a, \alpha_i \in \mathbb{R}^{d\times d}$, i=1,...,d.

Theorem 2.1. *Suppose $\mathbf{X}$ is an $\mathbb{R}^d$-valued affine process. Then the conditional moment generating function of $\mathbf{X}$ is of the form*

$$\mathbb{E}\left[\mathrm{e}^{\langle \mathbf{u}, \mathbf{X}_T\rangle}|\mathcal{F}_t\right] = \mathrm{e}^{\phi(T-t,\mathbf{u})+\langle\psi(T-t,\mathbf{u}),\mathbf{X}_t\rangle}, \tag{2.3}$$

with $\mathbf{u} \in \mathbb{R}^d$ if and only if and ϕ and ψ satisfy the Ricatti equations

$$\begin{aligned}\frac{\partial\psi_i(t,\mathbf{u})}{\partial t} &= \langle\psi(t,\mathbf{u}),\beta_i\rangle + \frac{1}{2}\langle\psi(t,\mathbf{u}),\alpha_i\psi(t,\mathbf{u})\rangle, && i = 1,...,d,\\ \psi_i(0,\mathbf{u}) &= u_i, && i = 1,...,d,\\ \frac{\partial\phi(t,\mathbf{u})}{\partial t} &= \langle\psi(t,\mathbf{u}),b(T-t)\rangle + \frac{1}{2}\langle\psi(t,\mathbf{u}),a(T-t)\psi(t,\mathbf{u})\rangle,\\ \phi(0,\mathbf{u}) &= 0.\end{aligned}$$

Proof. By Filipović [35, Theorem 2.13] with $c = \gamma = m = \mu = 0$,

$$\mathbb{E}\left[\mathrm{e}^{\langle \mathbf{u}, \mathbf{X}_T\rangle}|\mathcal{F}_\tau\right] = \mathrm{e}^{\tilde\phi(\tau,T,\mathbf{u})+\langle\tilde\psi(\tau,T,\mathbf{u}),\mathbf{X}_\tau\rangle}$$

with $\tilde\phi$ and $\tilde\psi$ solving the Ricatti equaitons

$$\begin{aligned}-\frac{\partial\tilde\psi_i(\tau,T,\mathbf{u})}{\partial\tau} &= \langle\tilde\psi(\tau,T,\mathbf{u}),\beta_i\rangle + \frac{1}{2}\langle\tilde\psi(\tau,T,\mathbf{u}),\alpha_i\tilde\psi(\tau,T,\mathbf{u})\rangle,\\ \tilde\psi(T,T,\mathbf{u}) &= u_i,\\ -\frac{\partial\tilde\phi(\tau,T,\mathbf{u})}{\partial\tau} &= \langle\tilde\psi(\tau,T,\mathbf{u}),b(\tau)\rangle + \frac{1}{2}\langle\tilde\psi(\tau,T,\mathbf{u}),a(\tau)\tilde\psi(\tau,T,\mathbf{u})\rangle\\ \tilde\phi(T,T,\mathbf{u}) &= 0.\end{aligned}$$

The result follows by taking $\phi(t,\mathbf{u}) = \tilde\phi(T-t,T,\mathbf{u})$ and $\psi_i(t,\mathbf{u}) = \tilde\psi_i(T-t,T,\mathbf{u})$. □

The conditional characteristic function is the conditional moment generating function, substituting $i\mathbf{u}$ for $\mathbf{u}$.

2.2.3 Partial differential equations

A third method for option pricing is to rewrite the option price as the solution of a partial differential equation (PDE).

Definition 2.5. *Let $0 = t_0 < t_1 < ... < t_K = t$ be a partition and $h = t_k - t_{k-1} = \frac{t}{K}$ the mesh size. The covariation of two processes X and Y is*

$$\langle X, Y \rangle_t = \lim_{h \to 0} \sum_{k=1}^{K} \left(X_{t_k} - X_{t_{k-1}} \right) \left(Y_{t_k} - Y_{t_{k-1}} \right).$$

The quadratic variation is the covariation of a process with itself.

For a Brownian motion, the quadratic variation is [62]

$$\langle W, W \rangle_t = \lim_{h \to 0} \sum_{k=1}^{K} \left(W_{t_k} - W_{t_{k-1}} \right)^2 = \lim_{h \to 0} \sum_{k=1}^{K} t_k - t_{k-1} = \lim_{h \to 0} t = t,$$

while for time t

$$\langle t, t \rangle_t = \lim_{h \to 0} \sum_{k=1}^{K} \left(t_k - t_{k-1} \right)^2 = \lim_{h \to 0} th = 0.$$

More generally, an Itô process of the form (2.1) has quadratic variation

$$\langle X, X \rangle_t = \int_0^t \sigma_s^2 \mathrm{d}s.$$

For Itô processes the famous Itô's formula [71] holds.

Lemma 2.1 (Itô's formula)**.** *Let $\mathbf{X}$ be an $\mathbb{R}^d$-valued Itô process and $f : \mathbb{R}^d \to \mathbb{R}$ a $\mathcal{C}^2$-function. Then*

$$\mathrm{d}f(\mathbf{X}_t) = \sum_{i=1}^{d} \frac{\partial f}{\partial X_t^i} \mathrm{d}X_t^i + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial X_t^i \partial X_t^j} \mathrm{d}\langle X_t^i, X_t^j \rangle. \tag{2.4}$$

If f depends on t as well, then

$$\mathrm{d}f(t, \mathbf{X}_t) = \frac{\partial f}{\partial t} \mathrm{d}t + \sum_{i=1}^{d} \frac{\partial f}{\partial X_t^i} \mathrm{d}X_t^i + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial X_t^i \partial X_t^j} \mathrm{d}\langle X_t^i, X_t^j \rangle.$$

Substituting equation (2.2) in (2.4) gives

$$\begin{aligned}
\mathrm{d}f(X_t) &= \sum_{i=1}^{d} \frac{\partial f}{\partial X_t^i} \left(\mu_t^i \mathrm{d}t + \sigma_t^i \mathrm{d}W_t^i \right) + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial X_t^i \partial X_t^j} \sigma_t^i \sigma_t^j \mathrm{d}t \\
&= \left(\sum_{i=1}^{d} \frac{\partial f}{\partial X_t^i} \mu_t^i + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial X_t^i \partial X_t^j} \sigma_t^i \sigma_t^j \right) \mathrm{d}t + \sum_{i=1}^{d} \frac{\partial f}{\partial X_t^i} \sigma_t^i \mathrm{d}W_t^i \\
&= \mathcal{A}f \mathrm{d}t + \text{local martingale}.
\end{aligned}$$

$\mathcal{A}$ is the infinitesimal generator of the process X.

Theorem 2.2 (Feynman–Kac Theorem)**.** *Let* $\mathbf{X}$ *be an* $\mathbb{R}^d$*-valued Itô process and* $\mathcal{A}$ *its infinitesimal generator. Then* u *satisfies the PDE*

$$\begin{aligned} \frac{\partial u}{\partial t} + \mathcal{A}u &= ru, & (t, \mathbf{x}) &\in [0,T] \times \mathbb{R}^d, \\ u(T, \mathbf{x}) &= \Psi(\mathbf{x}), & \mathbf{x} &\in \mathbb{R}^d \end{aligned} \tag{2.5}$$

if and only if

$$u(t, \mathbf{x}) = \mathrm{e}^{-r(T-t)} \mathbb{E}\left[\Psi(\mathbf{X}_T) | \mathbf{X}_t = \mathbf{x}\right].$$

Thus, the price of a European option satisfies PDE (2.5) with $\mathcal{A}$ the infinitesimal generator of the stock price process and Ψ the terminal payoff.

2.3 Models

In this section we explain the different option pricing models that we use throughout the thesis. In Section 2.3.1 we treat the Black–Scholes model, in Section 2.3.2 the Heston model and in Section 2.3.3 the rough and lifted Heston model.

2.3.1 Black–Scholes model

The most famous model for option pricing is the Black and Scholes [23] model. In this model, the dynamics of the stock price S are a geometric Brownian motion:

$$\mathrm{d}S_t = rS_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t, \quad S_0 > 0, \tag{2.6}$$

with $\sigma \in \mathbb{R}_+$ a parameter indicating how much the stock fluctuates, called the volatility and $r \in \mathbb{R}_+$ the risk-free rate.

Let $f : \mathbb{R} \to \mathbb{R}$ be a $\mathcal{C}^2$-function. Then by Itô's formula

$$\begin{aligned} \mathrm{d}f(S_t) &= \frac{\partial f}{\partial S}\left(rS_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t\right) + \frac{1}{2}\frac{\partial^2 f}{\partial S^2} \mathrm{d}\langle S_t, S_t \rangle \\ &= \left(\frac{\partial f}{\partial S} rS_t + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S_t^2\right) \mathrm{d}t + \text{local martingale}. \end{aligned}$$

So by the Feynman–Kac Theorem the price of an option $u(t, S)$ satisfies the PDE:

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 u}{\partial S^2} + rS\frac{\partial u}{\partial S} - ru &= 0, \\ u(T, S) &= \Psi(S). \end{aligned}$$

We consider time to maturity instead of time and denote it by t as well. Then the PDE becomes

$$\begin{aligned} \frac{\partial u}{\partial t} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 u}{\partial S^2} - rS\frac{\partial u}{\partial S} + ru &= 0, \\ u(0, S) &= \Psi(S). \end{aligned}$$

This PDE has an exact solution [71]:

$$\begin{aligned} u_{\text{call}}(t,S) &= S\Phi(d_1) - K\mathrm{e}^{-rt}\Phi(d_2), \\ u_{\text{put}}(t,S) &= K\mathrm{e}^{-rt}\Phi(-d_2) - S\Phi(-d_1). \end{aligned} \tag{2.7}$$

with

$$d_1 = \frac{\log\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)t}{\sigma\sqrt{t}}, \quad d_2 = d_1 - \sigma\sqrt{t}.$$

2.3.2 Heston model

The Heston [49] model is a popular stochastic volatility model with dynamics

$$\begin{aligned} \mathrm{d}S_t &= rS_t\mathrm{d}t + \sqrt{V_t}S_t\mathrm{d}W_t, & S_0 > 0, \\ \mathrm{d}V_t &= \lambda(\kappa - V_t)\mathrm{d}t + \eta\sqrt{V_t}\mathrm{d}B_t, & V_0 > 0. \end{aligned} \tag{2.8}$$

Here V is the variance process, B a Brownian motion correlated to W with correlation ρ, and $\lambda, \kappa, \eta \in \mathbb{R}_+$. Following similar arguments as in the Black–Scholes case, the price of an option satisfies the PDE:

$$\begin{aligned} \frac{\partial u}{\partial t} + \mathcal{A}u &= ru, \\ u(T,S,V) &= \Psi(S); \end{aligned}$$

with

$$\mathcal{A}f = rS\frac{\partial f}{\partial S} + \lambda(\kappa - V)\frac{\partial f}{\partial V} + \frac{1}{2}S^2V\frac{\partial^2 f}{\partial S^2} + \frac{1}{2}\eta^2V\frac{\partial^2 f}{\partial V^2} + \rho\eta SV\frac{\partial^2 f}{\partial S\partial V}.$$

Switching to time to maturity, the price of an option satisfies:

$$\begin{aligned} \frac{\partial u}{\partial t} - \mathcal{A}u + ru &= 0, \\ u(0,S,V) &= \Psi(S). \end{aligned}$$

Unlike the Black–Scholes model, this PDE does not have an analytical solution. However, we can compute the characteristic function. Since W_t and B_t are two Brownian motions with correlation ρ, we can write $W_t = \rho B_t + \sqrt{1-\rho^2}\mathrm{d}B_t^{\perp}$, with $B^{\perp}$ a Brownian motion independent of B. Let $X = \log(S)$, then the affine process $\mathbf{X} = (X, V)$ has dynamics

$$\mathrm{d}\mathbf{X}_t = \begin{bmatrix} \left(r - \frac{1}{2}V_t\right) \\ \lambda(\kappa - V_t) \end{bmatrix}\mathrm{d}t + \begin{bmatrix} \sqrt{V_t}\rho & \sqrt{V_t}\sqrt{1-\rho^2} \\ \eta\sqrt{V_t} & 0 \end{bmatrix}\mathrm{d}\begin{bmatrix} B_t \\ B_t^{\perp} \end{bmatrix}.$$

Then the moment generating function of $\mathbf{X}$, with $\mathbf{u} = (u, v)$, is of the form (2.3) with [49]:

$$\begin{aligned}
\phi(t, \mathbf{u}) &= rut + \frac{\lambda\kappa}{\eta^2}\left((\lambda - \rho\eta u - D)t - 2\log\left(\frac{1 - g\mathrm{e}^{-Dt}}{1-g}\right)\right), \\
\psi_1(t, \mathbf{u}) &= u, \\
\psi_2(t, \mathbf{u}) &= \frac{\lambda - \rho\eta u - D}{\eta^2}\frac{1 - \mathrm{e}^{-Dt}}{1 - g\mathrm{e}^{-Dt}}, \\
g &= \frac{\lambda - \rho\eta u - D}{\lambda - \rho\eta u + D}, \\
D &= \sqrt{(\rho\eta u - \lambda)^2 + \eta^2(u - u^2)}.
\end{aligned} \tag{2.9}$$

2.3.3 Rough/Lifted Heston model

The Heston model requires difficult modifications, such as making the parameters time dependent, in order to adequately calibrate the model to financial market data. The rough Heston model, in which the volatility process is driven by a fractional Brownian motion, is an alternative in which only a few (constant) parameters are required in order to adequately calibrate the model; see Gatheral et al. [40] and Bayer, Friz, and Gatheral [12]. The dynamics of the variance process in the rough Heston model are:

$$V_t = V_0 + \int_0^t K(t-s)\left(\lambda(\kappa - V_s)\mathrm{d}s + \eta\sqrt{V_s}\mathrm{d}B_s\right), \tag{2.10}$$

with the fractional kernel $K(t) = \frac{t^{H-\frac{1}{2}}}{\Gamma\left(H+\frac{1}{2}\right)}$, $\lambda, \kappa, \eta \in \mathbb{R}_+$ and $H \in \left(0, \frac{1}{2}\right)$ the Hurst parameter. This model is not Markovian, due to the presence of the fractional Brownian motion in the dynamics, hence the pricing and hedging of derivatives becomes a challenging task. In particular, we do not have access to a PDE of the form (2.5) for the rough Heston model.

The lifted Heston model is a Markovian approximation of the rough Heston model, see Abi Jaber and El Euch [3], that is interesting as a model in its own right; see also Abi Jaber [1]. The lifted Heston model converges to the rough Heston model, and the option prices in the lifted Heston model converge to the option prices in the rough Heston model. The essence of the model is to rewrite the kernel as a Laplace transform and to approximate this kernel with discretization.

Define $c_H = \frac{1}{\Gamma\left(H+\frac{1}{2}\right)\Gamma\left(\frac{1}{2}-H\right)}$ and measure $\mu(\mathrm{d}x) = c_H x^{-H-\frac{1}{2}}\mathrm{d}x$. Then

$$\begin{aligned}
\int_0^\infty \mathrm{e}^{-xt}\mu(\mathrm{d}x) &= c_H\int_0^\infty \mathrm{e}^{-xt}x^{-H-\frac{1}{2}}\mathrm{d}x = c_H\int_0^\infty \mathrm{e}^{-u}\left(\frac{u}{t}\right)^{-H-\frac{1}{2}}\frac{1}{t}\mathrm{d}u \\
&= c_H t^{H-\frac{1}{2}}\int_0^\infty \mathrm{e}^{-u}u^{-H-\frac{1}{2}}\mathrm{d}u = c_H t^{H-\frac{1}{2}}\Gamma\left(\frac{1}{2}-H\right) = K(t).
\end{aligned}$$

So $K(t)$ is written as the Laplace transform of μ. In the lifted Heston model, we

replace μ by a finite sum of Dirac measures. The approximate kernel is

$$K^n(t) = \sum_{i=1}^{n} c_i^n \mathrm{e}^{-\gamma_i t},$$

with

$$c_i^n = \left(r_n^{1-\alpha} - 1\right) \frac{r_n^{(\alpha-1)\left(1+\frac{n}{2}\right)} r_n^{(1-\alpha)i}}{\Gamma(\alpha)\Gamma(2-\alpha)}, \quad \gamma_i^n = \frac{1-\alpha}{2-\alpha} \frac{r_n^{2-\alpha}-1}{r_n^{1-\alpha}-1} r_n^{i-1-\frac{n}{2}}, \quad \alpha = H + \frac{1}{2}.$$

Theoretical results suggest taking $r_n = 1 + 10n^{-0.9}$. Numerical results suggest that $n = 20$ and $r_n = 2.5$ gives a good approximation to the rough Heston model [1]. Replacing $K(t)$ by $K^n(t)$ in (2.10) gives

$$V_t^n = g^n(t) + \int_0^t \sum_{i=1}^{n} c_i \mathrm{e}^{-\gamma_i(t-s)} \left(-\lambda V_s^n \mathrm{d}s + \eta\sqrt{V_s^n}\mathrm{d}B_s\right) = g^n(t) + \sum_{i=1}^{n} c_i^n V_t^{n,i},$$

with

$$g^n(t) = V_0 + \lambda\kappa \sum_{i=1}^{n} c_i^n \int_0^t \mathrm{e}^{-\gamma_i^n(t-s)}\mathrm{d}s,$$
$$V_t^{n,i} = \int_0^t \mathrm{e}^{-\gamma_i(t-s)} \left(-\lambda V_s \mathrm{d}s + \eta\sqrt{V_s}\mathrm{d}B_s\right).$$

The last term is the solution of an Ornstein–Uhlenbeck process [37] with dynamics:

$$\mathrm{d}V_t^{n,i} = -\left(\gamma_i^n V_t^{n,i} + \lambda V_t^n\right)\mathrm{d}t + \eta\sqrt{V_t^n}\mathrm{d}B_t, \quad V_0^{n,i} = 0.$$

Indeed, applying Itô's formula to the function $f\left(V_t^{n,i}, t\right) = V_t^{n,i}\mathrm{e}^{\gamma_i^n t}$ gives

$$\mathrm{d}f = \frac{\partial f}{\partial t}\mathrm{d}t + \frac{\partial f}{\partial V_t^{n,i}}\mathrm{d}V_t^{n,i} + \frac{1}{2}\frac{\partial^2 f}{\partial\left(V_t^{n,i}\right)^2}\mathrm{d}\left\langle V_t^{n,i}, V_t^{n,i}\right\rangle = \gamma_i^n V_t^{n,i}\mathrm{e}^{\gamma_i^n t}\mathrm{d}t + \mathrm{e}^{\gamma_i^n t}\mathrm{d}V_t^{n,i}$$
$$= \mathrm{e}^{\gamma_i^n t}\left(-\lambda V_t \mathrm{d}t + \eta\sqrt{V_t}\mathrm{d}W_t\right).$$

Integrating from 0 to t gives

$$V_t^{n,i}\mathrm{e}^{\gamma_i^n t} - V_0^{n,i} = \int_0^t \mathrm{e}^{\gamma_i^n s}\left(-\lambda V_s \mathrm{d}s + \eta\sqrt{V_s}\mathrm{d}B_s\right)$$
$$\implies V_t^{n,i} = V_0^{n,i}\mathrm{e}^{-\gamma_i^n t} + \int_0^t \mathrm{e}^{-\gamma_i^n(t-s)}\left(-\lambda V_s \mathrm{d}s + \eta\sqrt{V_s}\mathrm{d}B_s\right).$$

For n = 1, $c_1^1 = 1$ and $\gamma_1^1 = 0$, the lifted Heston model degenerates into the standard Heston model (2.8).

Lifted Heston PDE

Consider the lifted Heston dynamics that we derived:

$$\begin{aligned} \mathrm{d}S_t &= rS_t\mathrm{d}t + \sqrt{V_t^n}S_t\mathrm{d}W_t, & S_0 > 0,\\ \mathrm{d}V_t^{n,i} &= -\left(\gamma_i^n V_t^{n,i} + \lambda V_t^n\right)\mathrm{d}t + \eta\sqrt{V_t^n}\mathrm{d}B_t, & V_0^{n,i} = 0,\\ V_t^n &= g^n(t) + \sum_{i=1}^n c_i^n V_t^{n,i}, \\ g^n(t) &= V_0 + \lambda\kappa\sum_{i=1}^n c_i^n \int_0^t \mathrm{e}^{-\gamma_i^n(t-s)}\mathrm{d}s, \end{aligned} \tag{2.11}$$

with $\lambda, \eta, c_i^n, \gamma_i^n, V_0, \kappa \in \mathbb{R}_+$, and W, B two correlated (standard) Brownian motions with correlation coefficient ρ. The variance factors $V^{n,i}$ start from zero and share the same one-dimensional Brownian motion B. Let $f : \mathbb{R}^{n+1} \to \mathbb{R}$ be a $\mathcal{C}^2$-function. Denote $f_S = \frac{\partial f}{\partial S}$ and $f_{V^i} = \frac{\partial f}{\partial V^{n,i}}$. Then Itô's formula gives

$$\begin{aligned} \mathrm{d}f\left(S_t, V_t^{n,1}, ..., V_t^{n,n}\right) =& f_S\mathrm{d}S_t + \sum_{i=1}^n f_{V^i}\mathrm{d}V_t^{n,i} + \frac{1}{2}f_{SS}\mathrm{d}\left\langle S_t, S_t\right\rangle \\ &+ \sum_{i=1}^n f_{SV^i}\mathrm{d}\left\langle S_t, V_t^{n,i}\right\rangle + \frac{1}{2}\sum_{i,j=1}^n f_{V^iV^j}\mathrm{d}\left\langle V_t^{n,i}, V_t^{n,i}\right\rangle \\ =& f_S rS_t\mathrm{d}t - \sum_{i=1}^n f_{V^i}\left(\gamma_i^n V_t^{n,i} + \lambda V_t^n\right)\mathrm{d}t + \frac{1}{2}f_{SS}V_t^n\left(S_t\right)^2\mathrm{d}t \\ &+ \sum_{i=1}^n f_{SV^i}\eta\rho V_t^n S_t\mathrm{d}t + \frac{1}{2}\sum_{i,j=1}^n f_{V^iV^j}\eta^2 V_t^n\mathrm{d}t \\ &+ \text{local martingale.} \end{aligned}$$

Then the Feynman–Kac Theorem gives that the price of an option u with payoff $\Psi(S_T)$ satisfies

$$\begin{aligned} \frac{\partial u}{\partial t} + \tilde{\mathcal{A}}u &= ru,\\ u(T, S, V^{n,1}, ..., V^{n,n}) &= \Psi(S), \end{aligned}$$

with

$$\begin{aligned} \tilde{\mathcal{A}}f =& rSf_S - \sum_{i=1}^n\left(\gamma_i^n V^{n,i} + \lambda V_t^n\right)f_{V^i} \\ &+ \frac{1}{2}V_t^n S^2 f_{SS} + \eta\rho V_t^n S\sum_{i=1}^n f_{SV^i} + \frac{\eta^2}{2}V_t^n\sum_{i,j=1}^n f_{V^iV^j}. \end{aligned}$$

Switching to time to maturity, we have to be particularly careful in this case, because the function g^n in V^n depends on the time t. Let us denote

$$\tilde{V}_\tau^n := g^n(T-\tau) + \sum_{i=1}^n c_i^n V_\tau^{n,i}.$$

Then for time to maturity t, the price of the option u satisfies

$$\begin{aligned} \frac{\partial u}{\partial t} - \mathcal{A}u + ru = 0, \\ u(0, S, V^{n,1}, ..., V^{n,n}) = \Psi(S), \end{aligned}$$

with

$$\begin{aligned} \mathcal{A}f =& rSf_S - \sum_{i=1}^{n} \left(\gamma_i^n V^{n,i} + \lambda \tilde{V}_t^n\right) f_{V^i} \\ &+ \frac{1}{2}\tilde{V}_t^n S^2 f_{SS} + \eta\rho\tilde{V}_t^n S \sum_{i=1}^{n} f_{SV^i} + \frac{\eta^2}{2}\tilde{V}_t^n \sum_{i,j=1}^{n} f_{V^iV^j}. \end{aligned}$$

Lifted Heston characteristic function

Next, we derive the characteristic function in the lifted Heston model. To apply the method from Section 2.2.2, we need an affine model. The S with the lifted Heston dynamics (2.11) is not affine. As in the Heston model, we transform to the logarithm of the stock price. Defining $X = \log(S)$ transforms (2.11) to

$$\begin{aligned} \mathrm{d}X_t &= \left(r - \frac{1}{2}V_t^n\right)\mathrm{d}t + \sqrt{V_t^n}\mathrm{d}W_t, & X_0 \in \mathbb{R}, \\ \mathrm{d}V_t^{n,i} &= -\left(\gamma_i^n V_t^{n,i} + \lambda V_t^n\right)\mathrm{d}t + \eta\sqrt{V_t^n}\mathrm{d}B_t, & V_0^{n,i} = 0, \\ V_t^n &= g^n(t) + \sum_{i=1}^{n} c_i^n V_t^{n,i}. \end{aligned} \tag{2.12}$$

As in the Heston case, we can write $W_t = \rho B_t + \sqrt{1-\rho^2}\mathrm{d}B_t^{\perp}$, with $B^{\perp}$ a Brownian motion independent of B. In matrix notation, (2.12) becomes

$$\mathrm{d}\mathbf{X}_t = \mathrm{d}\begin{bmatrix} X_t \\ V_t^{n,1} \\ \vdots \\ V_t^{n,n} \end{bmatrix} = \beta(\mathbf{X}_t)\mathrm{d}t + \sigma(\mathbf{X}_t)\mathrm{d}\mathbf{B}_t,$$

with

$$\beta(\mathbf{X}_t) = \begin{bmatrix} r - \frac{1}{2}\left(g^n(t) + \sum_{i=1}^n c_i^n V_t^{i,n}\right), \\ -\gamma_1^n V_t^{n,1} - \lambda\left(g^n(t) + \sum_{i=1}^n c_i^n V_t^{i,n}\right) \\ \vdots \\ -\gamma_n^n V_t^{n,n} - \lambda\left(g^n(t) + \sum_{i=1}^n c_i^n V_t^{i,n}\right) \end{bmatrix},$$

$$\sigma(\mathbf{X}_t) = \begin{bmatrix} \rho\sqrt{g^n(t) + \sum_{i=1}^n c_i^n V_t^{n,i}} & \sqrt{1-\rho^2}\sqrt{g^n(t) + \sum_{i=1}^n c_i^n V_t^{n,i}} \\ \eta\sqrt{g^n(t) + \sum_{i=1}^n c_i^n V_t^{n,i}} & 0 \\ \vdots & \vdots \\ \eta\sqrt{g^n(t) + \sum_{i=1}^n c_i^n V_t^{n,i}} & 0 \end{bmatrix}.$$

Then

$$\beta(\mathbf{X}_t) = b(t) + \beta_0 X_t + \beta_1 V_t^{n,1} + ... + \beta_n V_t^{n,n} = b(t) + B\mathbf{X},$$
$$\sigma(\mathbf{X}_t)\sigma(\mathbf{X}_t)^\mathsf{T} = a(t) + \alpha_0 X_t + \alpha_1 V_t^{n,1} + ... + \alpha_n V_t^{n,n},$$

with

$$b(t) = \begin{bmatrix} r - \frac{1}{2}g^n(t) \\ -\lambda g^n(t) \\ \vdots \\ -\lambda g^n(t) \end{bmatrix}, \quad \beta_0 = \mathbf{0}, \quad \beta_1 = \begin{bmatrix} -\frac{1}{2}c_1^n \\ -\gamma_1^n - \lambda c_1^n \\ -\lambda c_1^n \\ \vdots \\ -\lambda c_1^n \end{bmatrix}, ..., \beta_n = \begin{bmatrix} -\frac{1}{2}c_n^n \\ -\lambda c_n^n \\ \vdots \\ -\lambda c_n^n \\ -\gamma_n^n - \lambda c_n^n \end{bmatrix},$$
$$a(t) = g^n(t)\Sigma, \qquad \alpha_0 = \mathbf{0}, \quad \alpha_i = c_i^n\Sigma \text{ for } i = 1, ..., n,$$

with

$$\Sigma = \begin{bmatrix} 1 & \eta\rho & ... & \eta\rho \\ \eta\rho & \eta^2 & ... & \eta^2 \\ \vdots & \vdots & \ddots & \vdots \\ \eta\rho & \eta^2 & ... & \eta^2 \end{bmatrix}.$$

So the process $\mathbf{X}$ is affine. Then by Theorem 2.1 the moment generating function of the lifted Heston model is

$$\mathbb{E}\left[\mathrm{e}^{\langle \mathbf{u}, \mathbf{X}_T\rangle} | \mathcal{F}_t\right] = \mathrm{e}^{\phi(T-t,\mathbf{u}) + \langle \psi(T-t,\mathbf{u}), \mathbf{X}_t\rangle},$$

in which ϕ and ψ satisfy Ricatti equations. For ψ_0

$$\frac{\partial \psi_0(t,\mathbf{u})}{\partial t} = 0, \quad \psi_0(0,\mathbf{u}) = u_0 \implies \psi_0(t,u) = u_0.$$

Then for $i = 1, ..., n$

$$\begin{aligned}\frac{\partial \psi_i(t,\mathbf{u})}{\partial t} =& -\gamma_i^n \psi_i(t,\mathbf{u}) - \lambda c_i^n \sum_{j=1}^n \psi_j(t,\mathbf{u}) - \frac{1}{2} c_i^n u_0 \\ &+ c_i^n \left(\frac{1}{2} u_0^2 + u_0 \eta\rho \sum_{j=1}^n \psi_j(t,\mathbf{u}) + \frac{\eta^2}{2} \left(\sum_{j=1}^n \psi_j(t,\mathbf{u}) \right)^2 \right) \\ =& -\gamma_i^n \psi_i(t,\mathbf{u}) + c_i^n F\left(u_0, \sum_{j=1}^n \psi_j(t,\mathbf{u}) \right), \quad \psi_i(0,\mathbf{u}) = u_i,\end{aligned}$$

with

$$F(u,v) = \frac{1}{2}\left(u^2 - u\right) + (u\eta\rho - \lambda)\, v + \frac{\eta^2}{2} v^2.$$

Then finally

$$\frac{\partial \phi(t,\mathbf{u})}{\partial t} = r u_0 + g^n(T-t) F\left(u_0, \sum_{j=1}^n \psi_j(t,\mathbf{u}) \right), \quad \phi(0,\mathbf{u}) = 0.$$

3

Time Deep Gradient Flow method for option pricing in (rough) diffusion models

In the previous chapter we have seen different models in which we can price options. Furthermore, we have seen different methods how to solve the option price in these models. In the more complicated rough volatility models, the classical methods become slow, justifying the need for neural network algorithms. We have already seen one popular method called the Deep Galerkin Method (DGM), but this method has two issues. First, the choice of the type of error is arbitrary and is not guaranteed to be the best choice. Second, in higher-order partial differential equations (PDEs) the number of derivatives grows exponentially with the dimensions causing the training of the network to become slow.

In this chapter, we develop a novel deep learning approach for pricing European options in diffusion models, which can efficiently handle high-dimensional problems resulting from Markovian approximations of rough volatility models. We call our method the Time Deep Gradient Flow method (TDGF). The option pricing PDE is reformulated as an energy minimization problem, which is approximated in a time stepping fashion by deep artificial neural networks. The proposed scheme respects the asymptotic behavior of option prices for large levels of moneyness and adheres to a priori known bounds for option prices. A series of numerical examples assesses the accuracy and efficiency of the proposed method, with particular focus in the lifted Heston model.

3.1 Introduction

Stochastic volatility models have been popular in the mathematical finance literature because they allow to accurately model and reproduce the shape of implied volatility

This chapter is based on A. Papapantoleon and J. Rou. A time-stepping deep gradient flow method for option pricing in (rough) diffusion models. *Quantitative Finance*, pages 1–12, 2025.

smiles for a single maturity. They require though certain modifications, such as making the parameters time or maturity dependent, in order to reproduce a whole volatility surface; see *e.g.* the comprehensive books by Gatheral [38] or Bergomi [22]. The class of rough volatility models, in which the volatility process is driven by a fractional Brownian motion, offers an attractive alternative to classical volatility models, since they allow to reproduce many stylized facts of asset and option prices with only a few (constant) parameters; see *e.g.* the seminal articles by Gatheral et al. [40] and Bayer et al. [12], and the recent volume by Bayer, Friz, Fukasawa, Gatheral, Jacquier, and Rosenbaum [16]. The presence of fractional Brownian motion in the dynamics of rough volatility models yields that the volatility process is not a semimartingale, and the model is not Markovian, which means that the simulation of the dynamics, or the pricing and hedging of derivatives, become challenging tasks. This development has attracted increasing attention from the mathematical finance community and has led to the creation of several innovative methods to tackle these challenges.

Markovian approximation of fractional processes is a popular method for tackling the challenges arising in these models, since they allow to relate rough volatility models to their classical counterparts. Abi Jaber and El Euch [2, 3] have designed multifactor stochastic volatility models approximating rough volatility models and enjoying a Markovian structure. Zhu, Loeper, Chen, and Langrené [89] have showed that the rough Bergomi model can be well approximated by the forward-variance Bergomi model with appropriately chosen weights and mean reversion speed parameters, which has the Markovian property. Bayer and Breneis [9, 10] have studied Markovian approximations of stochastic Volterra equations using an d-dimensional diffusion process defined as the solution to a system of (ordinary) stochastic differential equations. Cuchiero and Teichmann [28] have provided existence, uniqueness and approximation results for Markovian lifts of affine rough volatility models of general (jump) diffusion type. Moreover, Bonesini, Callegaro, Grasselli, and Pagès [24] have proposed a theoretical framework that exploits convolution kernels to transform a Volterra path-dependent (non-Markovian) stochastic process into a standard (Markovian) diffusion process.

Novel simulation schemes have also been developed in order to speed up Monte Carlo methods for rough volatility models. Bennedsen, Lunde, and Pakkanen [21] have developed a hybrid simulation scheme for Brownian semistationary processes, and applied this scheme to the rough Bergomi model. This method was further refined by McCrickerd and Pakkanen [67] who developed variance reduction techniques for conditionally log-normal models, and by Fukasawa and Hirano [36] who optimized the use of random numbers through orthogonal projections. Moreover, Gatheral [39] has developed hybrid quadratic-exponential schemes for the simulation of rough affine forward variance models, while Bayer and Breneis [11] have developed efficient and accurate simulation schemes for the rough Heston model based on low-dimensional Markovian representations.

Deep learning and neural network methods have also been applied in the field of rough stochastic volatility modeling. Horvath, Muguruza, and Tomas [54] presented a neural network-based method that performs the calibration task for the

full implied volatility surface in (rough) volatility models. Jacquier and Oumgari [56] introduced a deep learning–based algorithm to evaluate options in affine rough stochastic volatility models. Jacquier and Zuric [58] constructed a numerical algorithm based on deep learning to solve path-dependent PDEs that arise in the context of rough volatility.

Many other methods have been used and applied to rough stochastic volatility models. El Euch and Rosenbaum [33] have computed the characteristic function of the log price in the rough Heston model, thus opening the door for the application of Fourier transform methods for option pricing. Bayer, Ben Hammouda, and Tempone [14] proposed the application of adaptive sparse grids and quasi Monte Carlo methods for option pricing in the rough Bergomi model, while Bayer, Friz, Gassiat, Martin, and Stemper [15] have applied Hairer's theory of regularity structures in order to analyze rough volatility models. In addition, Bayer, Friz, Gulisashvili, Horvath, and Stemper [13], Horvath, Jacquier, and Lacombe [53] and Jacquier and Pannier [57], among others, have studied asymptotic properties of rough volatility models in various regimes.

The connection between PDEs and option pricing is deep and well studied in the literature. Several methods have been developed in order to price options in diffusion or stochastic volatility models, especially in high-dimensional settings, using sparse grids, asymptotic expansions, finite difference and finite elements methods; see *e.g.* Düring and Fournié [30], Hilber, Matache, and Schwab [50], Reisinger and Wittum [76] and the comprehensive book by Hilber, Reichmann, Schwab, and Winter [51].

The recent advances in deep learning methods have also created an intense interest in the solution of PDEs by deep learning and neural network methods. Sirignano and Spiliopoulos [81] proposed to solve high-dimensional PDEs by approximating the solution with a deep neural network which is trained to satisfy the differential operator, initial condition, and boundary conditions. Raissi, Perdikaris, and Karniadakis [75] introduced physics informed neural networks, *i.e.* neural networks that are trained to solve supervised learning tasks while respecting any given law of physics described by general nonlinear PDEs. Van der Meer, Oosterlee, and Borovykh [83] discussed the choice of a norm in the loss function for the training of neural networks. Han, Jentzen, and E [47] introduced a deep learning-based approach that can handle general high-dimensional parabolic PDEs by reformulating the PDE using backward stochastic differential equations and approximating the gradient of the unknown solution by neural networks. E and Yu [32] proposed a deep learning-based method for numerically solving variational problems, particularly the ones that arise from PDEs. Liao and Ming [63] proposed a method to deal with the essential boundary conditions encountered in the deep learning-based numerical solvers for PDEs. Georgoulis, Loulakis, and Tsiourvas [41] considered the approximation of initial and boundary value problems involving high-dimensional, dissipative evolution PDEs using a deep gradient flow neural network framework. In similar spirit, Park, Kim, Son, and Hwang [73] proposed a deep learning-based minimizing movement scheme for approximating the solutions of PDEs. The work of [41] was followed up by Georgoulis, Papapantoleon, and Smaragdakis [42], who developed an implicit-explicit deep minimizing movement method for pricing Euro-

pean basket options written on assets that follow jump-diffusion dynamics. Let us refer the reader to the forthcoming book by Jentzen, Kuckuck, and von Wurstemberger [59], who provide an introduction to the topic of deep learning algorithms and a survey of the (vast) literature.

The starting point for our research is the existence of a Markovian approximation for a rough volatility model or, more generally, another Markovian model that describes empirical data well; see *e.g.* Rømer [77] and Abi Jaber and Li [4]. Inspired by Georgoulis et al. [41], we consider the option pricing PDE, introduce a time stepping discretization of the time derivative, and reformulate the spatial derivatives as an energy minimization problem; the main difference with Georgoulis et al. [41] is the presence of the drift (first-order) term, which we treat in an explicit fashion in the discretization. This gradient flow formulation of the PDE gives rise to a suitable cost function for the training, via stochastic gradient descent, of deep neural networks that approximate the solution of the PDE. The main advantages of this method are twofold: on the one hand, the neural network from the previous time step is a good initial guess for the neural network optimization in the current time step, thus reducing the number of training stages necessary. On the other hand, the energy formulation allows to consider only first-order derivatives, thus reducing the training time in each time step / training stage significantly. In addition, we use available information about the qualitative behavior of option prices in order to assist the training of the neural network. More specifically, the architecture of the neural network respects the asymptotic behavior of option prices for large levels of moneyness, and adheres to a priori known bounds for option prices. The empirical results show that the proposed method is accurate, with a small overall error in the order of 10^{-3} and comparable with other deep learning methods, while the training times are significantly smaller compared to similar methods and do not grow significantly with the dimension of the underlying space.

This chapter is organized as follows: in Section 3.2, we revisit the option pricing PDE for Markovian diffusion models. In Section 3.3, we explain how we develop our method for solving option pricing PDEs, with focus on the time discretization and energy formulation. In Section 3.4, we revisit classical models for asset prices, with particular focus in the lifted Heston model. In Section 3.5, we explain certain details for the design and implementation of the method. In Section 3.6, we present and discuss the outcome of our numerical experiments, while Section 3.7 concludes this chapter.

3.2 Problem formulation

Let S denote the price process of a financial asset that evolves according to a (Markovian) diffusion model, and consider a European vanilla derivative on S with payoff $\Psi(S_T)$ at maturity time $T > 0$. Using the fundamental theorem of asset pricing and the Feynman–Kac formula, the price of this derivative can be written as the solution to a PDE in these models. Indeed, let $u : [0, T] \times \Omega \to \mathbb{R}$ denote the price of this derivative, with $\Omega \subseteq \mathbb{R}^{n+1}$ and t the time to maturity. Then, u solves the

general PDE

$$\begin{aligned} \frac{\partial}{\partial t}u(t,\mathbf{x}) + \mathcal{A}u(t,\mathbf{x}) + ru(t,\mathbf{x}) = 0, \qquad & (t,\mathbf{x}) \in (0,T]\times\Omega, \\ u(0,\mathbf{x}) = \Psi(\mathbf{x}), \qquad & \mathbf{x}\in\Omega, \end{aligned} \tag{3.1}$$

with $\mathcal{A}$ a second-order differential operator of the form

$$\mathcal{A}u = -\sum_{i,j=0}^{n} a^{ij}\frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=0}^{n}\beta^i \frac{\partial u}{\partial x_i}. \tag{3.2}$$

The coefficients a^{ij}, β^i of the generator $\mathcal{A}$ are directly related to the dynamics of the stochastic process S and can, in general, depend on the time and the spatial variables. For the moment, let us prescribe homogeneous Dirichlet boundary conditions on $\partial\Omega$.

We rewrite the PDE (3.1) as an energy minimization problem, therefore we need to split the operator in a symmetric and an asymmetric part. The symmetric part is then associated to a Dirichlet energy. Using the product rule, we can rewrite $\mathcal{A}$ as follows

$$\begin{aligned} \mathcal{A}u &= \sum_{i=0}^{n}\beta^i\frac{\partial u}{\partial x_i} - \sum_{i,j=0}^{n}\frac{\partial}{\partial x_j}\left(a^{ij}\frac{\partial u}{\partial x_i}\right) + \sum_{i,j=0}^{n}\frac{\partial a^{ij}}{\partial x_j}\frac{\partial u}{\partial x_i} \\ &= \sum_{i=0}^{n}\left(\beta^i + \sum_{j=0}^{n}\frac{\partial a^{ij}}{\partial x_j}\right)\frac{\partial u}{\partial x_i} - \sum_{i,j=0}^{n}\frac{\partial}{\partial x_j}\left(a^{ij}\frac{\partial u}{\partial x_i}\right). \end{aligned}$$

Define $b^i = \beta^i + \sum_{j=0}^{n}\frac{\partial a^{ij}}{\partial x_j}$. Then, $\mathcal{A}$ takes the form

$$\mathcal{A}u = -\nabla\cdot(A\nabla u) + \mathbf{b}\cdot\nabla u, \tag{3.3}$$

with

$$A = \begin{bmatrix} a^{00} & a^{10} & \dots & a^{n0} \\ a^{01} & a^{11} & \dots & a^{n1} \\ \vdots & \vdots & \ddots & \vdots \\ a^{0n} & a^{1n} & \dots & a^{nn} \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b^0 \\ b^1 \\ \vdots \\ b^n \end{bmatrix}. \tag{3.4}$$

The order of the derivatives with respect to x_i and x_j in the second derivative term can be interchanged in (3.2), hence we can always choose $a^{ij} = a^{ji}$. Therefore, A is a symmetric matrix.

3.3 Energy minimization and a Deep Gradient Flow method

Let us now explain how we develop our deep neural network method for solving the PDE (3.1). We proceed in three steps. First, we discretize the time derivative in

Section 3.3.1 and introduce a time stepping scheme. Second, we rewrite the solution of the PDE as an energy minimization problem in Section 3.3.2, which gives rise to a suitable cost function. Third, we approximate the minimizer by a neural network, which is trained using stochastic gradient descent, in Section 3.3.3

3.3.1 Time discretization

A key ingredient of our method is that we do not use time as an input variable for the neural network, thus training a global space-time network for solving the PDE, as in Sirignano and Spiliopoulos [81] and Raissi et al. [75]. Instead, following Georgoulis et al. [41], we wish to train a neural network for each time step. The neural network from the previous time step is a good initial guess for the neural network optimization in the current time step, thus reducing the number of training stages necessary.

Let us divide the time interval $(0, T]$ into K equally spaced intervals $(t_{k-1}, t_k]$, with $h = t_k - t_{k-1} = \frac{1}{K}$ for $k = 0, 1, \ldots, K$. Let U^k denote the approximation to the solution of the PDE $u(t_k, \mathbf{x})$ at time step t_k, using the following time discretization scheme

$$\frac{U^k - U^{k-1}}{h} - \nabla \cdot \left(A \nabla U^k\right) + \mathbf{b} \cdot \nabla U^k + r U^k = 0,$$
$$U^0 = \Psi.$$

In order to be able to rewrite the PDE as an energy minimization problem, we wish to treat the asymmetric part as a constant function. Therefore, we substitute the function U^k in the asymmetric part with its value from the previous time point, and thus consider the backward differentiation scheme

$$\begin{aligned} \frac{U^k - U^{k-1}}{h} - \nabla \cdot \left(A \nabla U^k\right) + F\left(U^{k-1}\right) + r U^k = 0, \\ U^0 = \Psi, \end{aligned} \tag{3.5}$$

with $F(u) = \mathbf{b} \cdot \nabla u$. Convergence analysis of this scheme appears in Section 5.3.2.

Remark 3.1. *Nonuniform grids can also be used for the discretization of time, in order to employ a finer grid initially when the solution is less smooth. The numerical experiments in Section 3.4 show that this refinement is not necessary in the examples we consider.*

3.3.2 Variational formulation

We would like to find an energy functional $I(u)$ such that U^k is a critical point of I. Let us rewrite (3.5) as follows:

$$\left(U^k - U^{k-1}\right) + h\left(-\nabla \cdot \left(A \nabla U^k\right) + r U^k + F\left(U^{k-1}\right)\right) = 0, \quad U^0 = \Psi. \tag{3.6}$$

Then, we have the following result, which is generalized in Theorem 5.3.

Lemma 3.1. *Assume that A is symmetric and positive semidefinite. Then, U^k solves (3.6) if and only if U^k minimizes*

$$I^k(u) = \frac{1}{2}\left\|u - U^{k-1}\right\|^2_{L^2(\Omega)} + h\left(\int_\Omega \frac{1}{2}\left((\nabla u)^\mathsf{T} A \nabla u + ru^2\right) + F\left(U^{k-1}\right) u \mathrm{d}x\right).$$

Proof. Let v be a smooth function that is zero on the boundary. Then, for all such v, consider the function

$$\begin{aligned} i^k(\tau) =& I^k\left(U^k + \tau v\right) \\ =& \int_\Omega \frac{1}{2}\left(U^k + \tau v - U^{k-1}\right)^2 \mathrm{d}x \\ &+ h\int_\Omega \frac{1}{2}\left(\left(\nabla\left(U^k + \tau v\right)\right)^\mathsf{T} A \nabla\left(U^k + \tau v\right) + r\left(U^k + \tau v\right)^2\right) \\ &+ F\left(U^{k-1}\right)\left(U^k + \tau v\right) \mathrm{d}x, \end{aligned}$$

for $\tau \in \mathbb{R}$. U^k minimizes I^k if and only if $\tau = 0$ minimizes i^k. Therefore,

$$\begin{aligned} 0 =& (i^k)'(0) \\ =& \left[\int_\Omega \left(U^k + \tau v - U^{k-1}\right) v \mathrm{d}x + h\int_\Omega \frac{1}{2}(\nabla v)^\mathsf{T} A \nabla\left(U^k + \tau v\right)\right. \\ &\left. + \frac{1}{2}\left(\nabla\left(U^k + \tau v\right)\right)^\mathsf{T} A \nabla v + r\left(U^k + \tau v\right) v + F\left(U^{k-1}\right) v \mathrm{d}x\right]_{\tau=0} \\ =& \int_\Omega \left(U^k - U^{k-1}\right) v + \frac{h}{2}\left((\nabla v)^\mathsf{T} A \nabla U^k + \left(\nabla U^k\right)^\mathsf{T} A \nabla v\right) + hrU^k v \\ &+ hF\left(U^{k-1}\right) v \mathrm{d}x \\ =& \int_\Omega \left(U^k - U^{k-1}\right) v + hA\nabla U^k \cdot \nabla v + h\left(rU^k + F\left(U^{k-1}\right)\right) v \mathrm{d}x \\ =& \int_\Omega \left(\left(U^k - U^{k-1}\right) + h\left(-\nabla \cdot \left(A\nabla U^k\right) + rU^k + F\left(U^{k-1}\right)\right)\right) v \mathrm{d}x, \end{aligned}$$

where in the second to last equality we used that A is symmetric, and in the last equality we used the divergence theorem. This equality holds for all v if and only if (3.6) holds. The second derivative is positive; indeed,

$$\left(i^k\right)''(\tau) = (1 + rh)\int_\Omega v^2 \mathrm{d}x + h\int_\Omega (\nabla v)^\mathsf{T} A \nabla v \mathrm{d}x > 0,$$

since A is positive semidefinite. Therefore, $\tau = 0$ is indeed the minimizer. □

3.3.3 Neural network approximation

Let $f^k(\mathbf{x}; \theta)$ denote a neural network approximation of U^k with trainable parameters θ. Applying a Monte Carlo approximation to the integrals, the discretized cost

functional takes the form

$$L^k(\theta;\mathbf{x}) = \frac{|\Omega|}{2M}\sum_{m=1}^{M}\left(f^k(\mathbf{x}_m;\theta) - f^{k-1}(\mathbf{x}_m)\right)^2 + \frac{h|\Omega|}{M}\sum_{m=1}^{M}\left[\frac{1}{2}\left(\left(\nabla f^k(\mathbf{x}_m;\theta)\right)^\top A\nabla f^k(\mathbf{x}_m;\theta) + r\left(f^k(\mathbf{x}_m;\theta)\right)^2\right) + \left(\mathbf{b}\cdot\nabla f^{k-1}(\mathbf{x}_m)\right) f^k(\mathbf{x}_m;\theta)\right].$$

Here, M denotes the number of samples $\mathbf{x}_m$.

In order to minimize this cost function, we use a stochastic gradient descent-type algorithm, *i.e.* an iterative scheme of the form:

$$\theta_{n+1} = \theta_n - \alpha_n \nabla_\theta L^k(\theta_n;\mathbf{x});$$

more specifically, we use the Adam algorithm [61]. The hyperparameter α_n is the step size of our update, called the learning rate. An overview of the TDGF method appears in Algorithm 1.

Algorithm 1 Time Deep Gradient Flow method

```
Initialize θ_0^0.
Set f^0(x;θ) = Ψ(x).
for each time step k = 1,...,K do
    Initialize θ_0^k = θ^{k-1}.
    for each sampling stage n = 1,...,N do
        Generate M random points x_m for training.
        Calculate the cost functional L^k(θ_n^k; x_i) for the sampled points.
        Take a descent step θ_{n+1}^k = θ_n^k − α_n ∇_θ L^k(θ_n^k; x_i).
    end for
end for
```

3.4 Examples

Let us now outline the different models to which we apply our method, and derive the appropriate form of the generator in the option pricing PDE for each of these models. In particular, we treat the Black–Scholes model in Section 3.4.1, the Heston model in Section 3.4.2, and the lifted Heston model in Section 3.4.3.

3.4.1 Black–Scholes model

The Black–Scholes model has one spatial variable x, corresponding to the asset price. In Section 2.3.1 we derived that the generator in the Black–Scholes model, in the form (3.2), equals

$$\mathcal{A}u = -\frac{1}{2}\sigma^2 x^2\frac{\partial^2 u}{\partial x^2} - rx\frac{\partial u}{\partial x},$$

with $r, \sigma \in \mathbb{R}_+$ the risk-free rate and the volatility respectively. Applying the chain rule,

$$\mathcal{A}u = -\frac{\partial}{\partial x}\left(\frac{1}{2}\sigma^2 x^2 \frac{\partial u}{\partial x}\right) + \sigma^2 x \frac{\partial u}{\partial x} - rx\frac{\partial u}{\partial x}.$$

Therefore, the operator $\mathcal{A}$ takes the form (3.3) with the coefficients in (3.4) provided by

$$\begin{aligned} a &= \frac{1}{2}\sigma^2 x^2, \\ b &= (\sigma^2 - r)x. \end{aligned}$$

We check that the operator A satisfies the assumptions of Lemma 3.1. Let $\zeta \in \mathbb{R}$. Then

$$\zeta^\mathsf{T} A \zeta = \frac{1}{2}\sigma^2 x^2 \zeta^2 \geq 0.$$

So the matrix A in the Black–Scholes case is indeed positive semidefinite.

3.4.2 Heston model

The Heston model has two spatial variables (x, v), corresponding to the asset price and the variance. In Section 2.3.2 we derived that the generator corresponding to the Heston model in the form (3.2), equals

$$\mathcal{A}u = -rx\frac{\partial u}{\partial x} - \lambda(\kappa - v)\frac{\partial u}{\partial v} - \frac{1}{2}x^2 v \frac{\partial^2 u}{\partial x^2} - \frac{1}{2}\eta^2 v \frac{\partial^2 u}{\partial v^2} - \rho\eta x v \frac{\partial^2 u}{\partial x \partial v}.$$

Here, ρ is the correlation between the Brownian motions driving S and V and $\lambda, \kappa, \eta \in \mathbb{R}_+$. Applying the chain rule again,

$$\begin{aligned} \mathcal{A}u = &- rx\frac{\partial u}{\partial x} - \lambda(\kappa - v)\frac{\partial u}{\partial v} - \frac{\partial}{\partial x}\left(\frac{1}{2}x^2 v \frac{\partial u}{\partial x}\right) + xv\frac{\partial u}{\partial x} - \frac{\partial}{\partial v}\left(\frac{1}{2}\eta^2 v \frac{\partial u}{\partial v}\right) + \frac{1}{2}\eta^2 \frac{\partial u}{\partial v} \\ &- \frac{\partial}{\partial x}\left(\frac{1}{2}\rho\eta x v \frac{\partial u}{\partial v}\right) + \frac{1}{2}\rho\eta v \frac{\partial u}{\partial v} - \frac{\partial}{\partial v}\left(\frac{1}{2}\rho\eta x v \frac{\partial u}{\partial x}\right) + \frac{1}{2}\rho\eta x \frac{\partial u}{\partial x}. \end{aligned}$$

Therefore, the operator $\mathcal{A}$ takes the form (3.3) with the coefficients in (3.4) provided by

$$\begin{aligned} a^{00} &= \frac{1}{2}x^2 v, \\ a^{10} = a^{01} &= \frac{1}{2}\rho\eta x v, \\ a^{11} &= \frac{1}{2}\eta^2 v, \\ b^0 &= \left(-r + v + \frac{1}{2}\rho\eta\right)x, \\ b^1 &= \lambda(v - \kappa) + \frac{1}{2}\eta^2 + \frac{1}{2}\rho\eta v. \end{aligned}$$

Let $\zeta = (\zeta_1, \zeta_2)^\mathsf{T} \in \mathbb{R}^2$. Then

$$\begin{aligned}\zeta^\mathsf{T} A \zeta &= \frac{1}{2}\left(vx^2\zeta_1^2 + 2\eta\rho vx\zeta_1\zeta_2 + \eta^2 v\zeta_2^2\right) \geq \frac{|\rho|v}{2}\left(x^2\zeta_1^2 \pm 2\eta x\zeta_1\zeta_2 + \eta^2\zeta_2^2\right) \\ &= \frac{|\rho|v}{2}\left(x\zeta_1 \pm \eta\zeta_2\right)^2 \geq 0.\end{aligned}$$

Therefore, the matrix A in the Heston model is also positive semidefinite.

3

3.4.3 Lifted Heston model

The lifted Heston model has $n+1$ spatial variables $(x, v_1, \ldots, v_n)$, corresponding to the asset price S and the n variance processes $V^{n,i}$ in (2.11). In Section 2.3.3 we derived that the generator of the model in the form (3.2) equals

$$\mathcal{A}u = -rxu_x + \sum_{i=1}^n \left(\gamma_i^n v_i + \lambda\tilde{V}_t^n\right) u_{v_i} - \frac{1}{2}\tilde{V}_t^n x^2 u_{xx} - \eta\rho\tilde{V}_t^n x \sum_{i=1}^n u_{xv_i} - \frac{\eta^2}{2}\tilde{V}_t^n \sum_{i,j=1}^n u_{v_iv_j},$$

with

$$\begin{aligned}\tilde{V}_\tau^n &= g^n(T-\tau) + \sum_{i=1}^n c_i^n v_i, \\ g^n(t) &= V_0 + \lambda\kappa \sum_{i=1}^n c_i^n \int_0^T \mathrm{e}^{-\gamma_i^n(t-s)}\mathrm{d}s,\end{aligned}$$

$\lambda, \eta, c_i^n, \gamma_i^n, V_0, \kappa \in \mathbb{R}_+$, and ρ the correlation between the Brownian motions driving the asset price and the volatilities. Then, applying the chain rule again,

$$\begin{aligned}\mathcal{A}u =& -rxu_x + \sum_{i=1}^n \left(\gamma_i^n v_i + \lambda\tilde{V}_\tau^n\right) u_{v_i} - \left(\frac{1}{2}\tilde{V}_\tau^n x^2 u_x\right)_x + \tilde{V}_\tau^n x u_x \\ &- \frac{1}{2}\sum_{i=1}^n \left(\eta\rho\tilde{V}_\tau^n x u_{v_i}\right)_x + \frac{1}{2}\eta\rho\tilde{V}_\tau^n \sum_{i=1}^n u_{v_i} - \frac{1}{2}\sum_{i=1}^n \left(\eta\rho\tilde{V}_\tau^n x u_x\right)_{v_i} + \frac{1}{2}\eta\rho x\sum_{i=1}^n c_i^n u_x \\ &- \sum_{i,j=1}^n \left(\frac{\eta^2}{2}\tilde{V}_\tau^n u_{v_i}\right)_{v_j} + \frac{\eta^2}{2}\sum_{i,j=1}^n c_j^n u_{v_i} \\ =& \left(\tilde{V}_\tau^n x - rx + \frac{1}{2}\eta\rho x\sum_{i=1}^n c_i^n\right) u_x \\ &+ \sum_{i=1}^n \left(\gamma_i^n v_i + \lambda\tilde{V}_\tau^n + \frac{1}{2}\eta\rho\tilde{V}_\tau^n + \frac{\eta^2}{2}\sum_{j=1}^n c_j^n\right) u_{v_i} - \left(\frac{1}{2}\tilde{V}_\tau^n x^2 u_x\right)_x \\ &- \frac{1}{2}\sum_{i=1}^n \left(\eta\rho\tilde{V}_\tau^n x u_{v_i}\right)_x - \frac{1}{2}\sum_{i=1}^n \left(\eta\rho\tilde{V}_\tau^n x u_x\right)_{v_i} - \sum_{i,j=1}^n \left(\frac{\eta^2}{2}\tilde{V}_\tau^n u_{v_i}\right)_{v_j}.\end{aligned}$$

Therefore, the operator $\mathcal{A}$ takes the form (3.3) with the coefficients in (3.4) provided by

$$\begin{aligned}
a^{00} &= \frac{1}{2}\tilde{V}_\tau^n x^2, \\
a^{i0} = a^{0i} &= \frac{1}{2}\eta\rho\tilde{V}_\tau^n x, && i = 1, \dots, n, \\
a^{ij} &= \frac{\eta^2}{2}\tilde{V}_\tau^n, && i, j = 1, \dots, n, \\
b^0 &= \left(\tilde{V}_\tau^n - r + \frac{1}{2}\eta\rho\sum_{i=1}^n c_i^n\right) x, \\
b^i &= \left(\gamma_i^n v_i + \lambda\tilde{V}_\tau^n + \frac{1}{2}\eta\rho\tilde{V}_\tau^n + \frac{\eta^2}{2}\sum_{j=1}^n c_j^n\right), && i = 1, \dots, n.
\end{aligned}$$

Let $\zeta = (\zeta_0, ..., \zeta_n)^\mathsf{T} \in \mathbb{R}^{n+1}$. Then

$$\begin{aligned}
\zeta^\mathsf{T} A\zeta &= (\zeta_0, ..., \zeta_n)^\mathsf{T} \begin{bmatrix} \frac{1}{2}\tilde{V}_t^n x^2\zeta_0 + \frac{1}{2}\eta\rho\tilde{V}_t^n x\sum_{i=1}^n \zeta_i \\ \frac{1}{2}\eta\rho\tilde{V}_t^n x\zeta_0 + \frac{\eta^2}{2}\tilde{V}_t^n\sum_{j=1}^n \zeta_j \\ \vdots \\ \frac{1}{2}\eta\rho\tilde{V}_t^n x\zeta_0 + \frac{\eta^2}{2}\tilde{V}_t^n\sum_{j=1}^n \zeta_j \end{bmatrix} \\
&= \frac{1}{2}\tilde{V}_t^n x^2\zeta_0^2 + \frac{1}{2}\eta\rho\tilde{V}_t^n x\zeta_0\sum_{i=1}^n \zeta_i + \sum_{i=1}^n \zeta_i\left(\frac{1}{2}\eta\rho\tilde{V}_t^n x\zeta_0 + \frac{\eta^2}{2}\tilde{V}_t^n\sum_{j=1}^n \zeta_j\right) \\
&= \frac{1}{2}\tilde{V}_t^n x^2\zeta_0^2 + \eta\rho\tilde{V}_t^n x\zeta_0\sum_{i=1}^n \zeta_i + \frac{\eta^2}{2}\tilde{V}_t^n\left(\sum_{i=1}^n \zeta_i\right)^2 \\
&= \frac{1}{2}\tilde{V}_t^n\left(x^2\zeta_0^2 + 2\eta\rho x\zeta_0\sum_{i=1}^n \zeta_i + \eta^2\left(\sum_{i=1}^n \zeta_i\right)^2\right).
\end{aligned}$$

Since $\frac{1}{2}\tilde{V}_t^n \geq 0$, $\zeta_0^2 \geq 0$, $\eta^2\left(\sum_{i=1}^n \zeta_i\right)^2 \geq 0$ and $|\rho| \leq 1$,

$$\begin{aligned}
\zeta^\mathsf{T} A\zeta &\geq \frac{|\rho|}{2}\tilde{V}_t^n\left(x^2\zeta_0^2 \pm 2\eta x\zeta_0\sum_{i=1}^n \zeta_i + \eta^2\left(\sum_{i=1}^n \zeta_i\right)^2\right) = \frac{|\rho|}{2}\tilde{V}_t^n\left(x\zeta_0 \pm \eta\sum_{i=1}^n \zeta_i\right)^2 \\
&\geq 0.
\end{aligned}$$

Therefore, the matrix A in the lifted Heston model is also positive semidefinite.

3.5 Implementation details

Let us now describe some details about the design of the neural network architecture and the implementation of the numerical method. Motivated by Georgoulis et al.

[42], we would like to use information about the option price in order to facilitate the training of the neural network. On the one hand, using the lower no-arbitrage bound of a call option, *i.e.* $u(t,x) \geq (x - K\mathrm{e}^{-rt})^+$, the neural network learns the difference between the option price and this bound, instead of the option price itself. This difference turns out to be an easier function to approximate than the option price.

On the other hand, after some large level of asset price / moneyness, the option price grows linearly with the asset price / moneyness. Let us denote this level by x_p. Therefore, instead of letting the neural network learn the option price at points larger than x_p, we approximate this price by adding the difference between x_p and this point to the option value at x_p:

$$u(x_p + y) = u(x_p) + y, \quad y > 0.$$

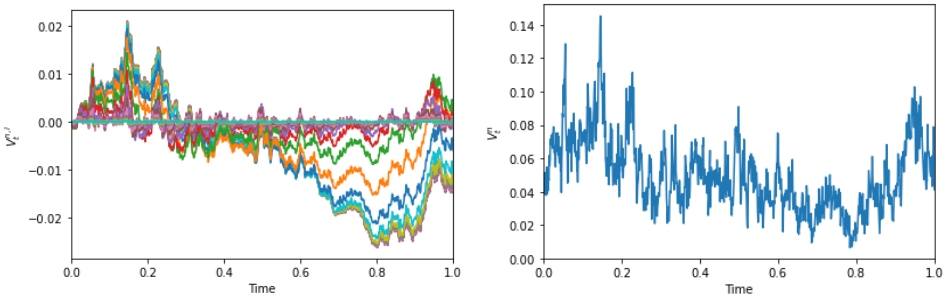
Moreover, we consider the moneyness $\frac{S}{K}$ as an input variable instead of treating the asset price S and the strike price K separately, in order to reduce the size of the parameter space.

The architecture of the neural network for the TDGF method is inspired by the DGM of Sirignano and Spiliopoulos [81], hence we call the hidden layers "DGM layers". Overall, we set the following:

$$\begin{aligned}
X^1 &= \sigma_1\left(W^1\mathbf{x} + b^1\right), \\
Z^l &= \sigma_1\left(U^{z,l}\mathbf{x} + W^{z,l}X^l + b^{z,l}\right), && l = 1,\dots,L, \\
G^l &= \sigma_1\left(U^{g,l}\mathbf{x} + W^{g,l}X^l + b^{g,l}\right), && l = 1,\dots,L, \\
R^l &= \sigma_1\left(U^{r,l}\mathbf{x} + W^{r,l}X^l + b^{r,l}\right), && l = 1,\dots,L, \\
H^l &= \sigma_1\left(U^{h,l}\mathbf{x} + W^{h,l}\left(X^l \odot R^l\right) + b^{h,l}\right), && l = 1,\dots,L, \\
X^{l+1} &= \left(1 - G^l\right) \odot H^l + Z^l \odot X^l, && l = 1,\dots,L, \\
f(\mathbf{x};\theta) &= \left(x - K\mathrm{e}^{-rt}\right)^+ + \sigma_2\left(WX^{L+1} + b\right),
\end{aligned} \tag{3.7}$$

Here, L is the number of hidden layers, σ_i is the activation function for $i = 1, 2$. Moreover, $\mathbf{x}$ denotes the vector of inputs, which are the asset price and the variances (in the stochastic volatility models), and then $\mathbf{x}_1 = x$. In the numerical experiments, we have used 3 layers and 50 neurons per layer. The activation functions we have selected are the hyperbolic tangent function, $\sigma_1(x) = \tanh(x)$, and the softplus function, $\sigma_2(x) = \log\left(\mathrm{e}^x + 1\right)$, which guarantees that the option price remains above the no-arbitrage bound.

We consider a maturity of $T = 1.0$ year, and use the parameters sets for the Heston and the lifted Heston model from Heston [49, Table 1] and Abi Jaber and El Euch [3, page 321]. We set the number of time steps equal to $K = 100$, use 2000 sampling stages in each time step, while in each sampling stage we take 600 samples per dimension, *i.e.* $M = 600(n+1)$ samples, following the recommendations in Georgoulis et al. [41]. Here, n denotes the number of variance processes, in the case of stochastic volatility models. As for the sampling domain, we consider the moneyness $x \in [0.01, 3.0]$ and the Heston volatility $V \in [0.001, 0.1]$.



(a) The factors $V_t^{20,i}$ of the variance process. (b) The total variance process V_t^{20}.

Figure 3.1: Monte Carlo simulation of the variance processes in the lifted Heston model for $H = 0.1$, $\lambda = 0.3$, $\eta = 0.1$, $\theta = 0.05$ and $V_0 = 0.05$.

In the lifted Heston model, we have to be particularly careful. Although the variance process V^n should remain nonnegative, some factors might become negative [1]. In Fig. 3.1 we plot a Monte Carlo simulation of the individual and total variance processes. Indeed, the individual factor become negative while the total variance process always stays nonnegative.

The variance of an Ornstein–Uhlenbeck process with dynamics

$$\mathrm{d}X_t = \theta(\mu - X_t)\mathrm{d}t + \sigma \mathrm{d}W_t,$$

is

$$\mathrm{Var}(X_t) = \frac{\sigma^2}{2\theta}\left(1 - \mathrm{e}^{-2\theta t}\right).$$

If we take $V_t^n = g^n(t)$ constant in the dynamics of the volatility processes, we approximate the variance as

$$\mathrm{Var}(V_t^{n,i}) \approx \frac{\eta^2 g^n(t)}{2\gamma_i^n}\left(1 - \mathrm{e}^{-2\gamma_i^n t}\right) =: \mathrm{var}_i(t).$$

We sample the lifted Heston volatilities $V^{n,i} \in \left[-V_i^{\mathrm{high}}, V_i^{\mathrm{high}}\right]$, with

$$V_i^{\mathrm{high}} = 3\sqrt{\mathrm{var}_i(T)}.$$

After sampling $V^{n,i}$ uniformly, we calculate V^n and discard all combinations that result in a negative V^n.

We set $x_p = 2.0$. In the optimization stage, we use the Adam algorithm [61] with a learning rate $\alpha = 3 \times 10^{-4}$, $(\beta_1, \beta_2) = (0.9, 0.999)$ and zero weight decay. Finally, the training is performed on the DelftBlue supercomputer [29], using a single NVidia Tesla V100S GPU.

3.6 Numerical results

Let us now present and discuss the outcome of certain numerical experiments involving all models outlined in Section 3.4. We are interested in both the accuracy,

in Section 3.6.1, and the speed, in Section 3.6.2, of the numerical method developed in this chapter.

We compare the TDGF method with another popular deep learning method for solving PDEs, the DGM of Sirignano and Spiliopoulos [81]. In the DGM approach, the solution of the PDE is translated into a quadratic minimization problem, which is minimized using stochastic gradient descent. In our setting, this minimization takes the form:

$$\left\|u_t - \nabla \cdot (A\nabla u) + \mathbf{b} \cdot \nabla u + ru\right\|^2_{L^2([0,T]\times\Omega)} + \left\|u(0,\mathbf{x}) - \Psi(\mathbf{x})\right\|^2_{L^2(\Omega)} \to \min.$$

For large levels of asset price / moneyness, the option price grows linearly with the asset price / moneyness. Therefore, we add a loss term punishing the derivative of the option price with respect to the stock deviating from 1 at $x = 3.0$.

In order to ensure a fair comparison between the two methods, we use 200,000 sampling stages for the DGM. In each stage, we use $M = 600(n + 2)$ samples for the PDE in the interior domain and $M = 600(n + 1)$ samples for the initial and boundary conditions. Moreover, we use the same network architecture as for the TDGF with the same number of layers and neurons per layer, and also use the Adam optimizer.

In order to compare the accuracy of the two methods, we need a reference value. In the Black–Scholes model, in the case of a European call option, the option pricing PDE has an exact solution, see (2.7).

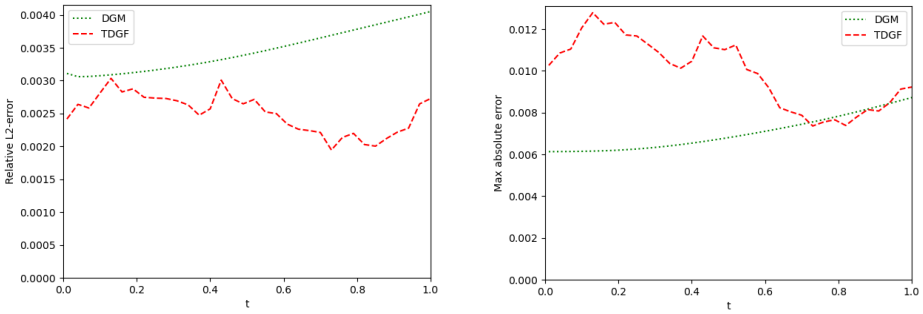
In the Heston model, the option pricing PDE does not admit an analytical solution, but we have an analytical expression for the characteristic function; see (2.9). Using the characteristic function, we compute the reference price with the COS method of Fang and Oosterlee [34], with $N = 512$ terms and truncation domain $\left[-10\tau^{\frac{1}{4}}, 10\tau^{\frac{1}{4}}\right]$.

Finally, in the lifted Heston model, the characteristic function does not have an analytical expression but is known up to the solution of a system of Riccati equations; see Section 2.3.3. We solve these equations using an implicit-explicit discretization scheme with 500 time steps, after which we apply the COS method again to compute the reference price.

3.6.1 Accuracy

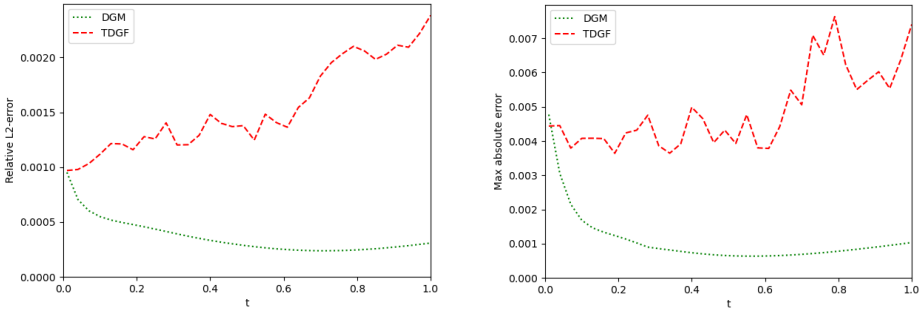
In the Black–Scholes model, we have only one input variable, the asset price S. Fig. 3.2 presents the relative L^2- and the absolute maximum error for the TDGF and the DGM methods in the Black–Scholes model. These errors are calculated over 47 evenly spaced grid points in the interval $[0.01, 3]$. Both methods have a comparable (small) error.

In the Heston model, we have two input variables, the asset price S and the variance V. Fig. 3.3 presents the relative L^2- and the absolute maximum error for the TDGF and the DGM methods in the Heston model. These figures are presented for fixed $V_0 = 0.03$, however other values of V_0 between 0.01 and 0.09 provide similar results. The DGM is more accurate than the TDGF method, although the TDGF



(a) Relative L^2-error. (b) Maximal absolute error.

Figure 3.2: Errors of the two methods in the Black–Scholes model against time, with interest rate $r = 0.05$ and volatility $\sigma = 0.25$.



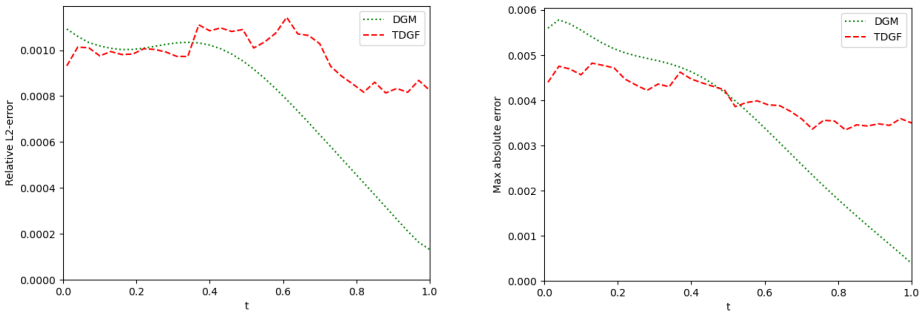
(a) Relative L^2-error (b) Maximal absolute error

Figure 3.3: Error of the two methods in the Heston model against time, with $r = 0.0$ and $\eta = 0.1$, $\rho = 0.0$, $\kappa = 0.01$, $V_0 = 0.03$ and $\lambda = 2.0$.

is just as accurate as in the Black–Scholes model, and the error is any case small, in the order of 10^{-3}.

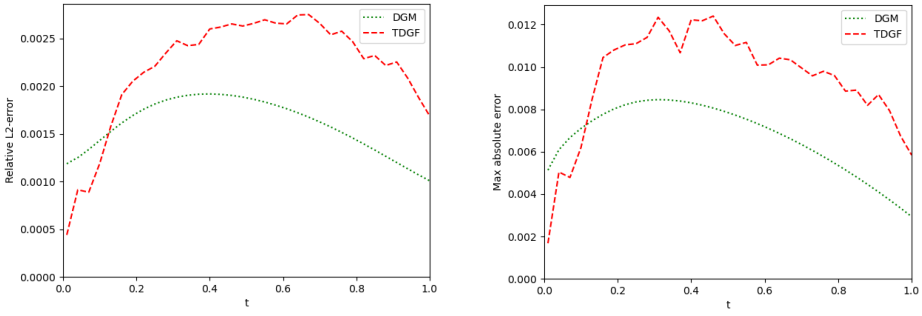
In the lifted Heston model, we have $n + 1$ input variables, the asset price S and the variances $V^{n,i}$, with $i = 1, \ldots, n$. Fig. 3.4 presents the relative L^2- and the maximum absolute error for the TDGF and the DGM methods in this model, in the case $n = 1$, using the same parameters as in the Heston model before. The numerical results are comparable for the two methods, and both methods even seem to perform slightly better than in the previous example.

Fig. 3.5 presents the relative L^2- and the maximum absolute error for the TDGF and the DGM methods again in the lifted Heston model, for the same parameter set as previously, but now with $n = 20$ variance factors. The errors of both methods are comparable and, although they have increased compared to the $n = 1$ case, they are still low, in the order of 10^{-3}.



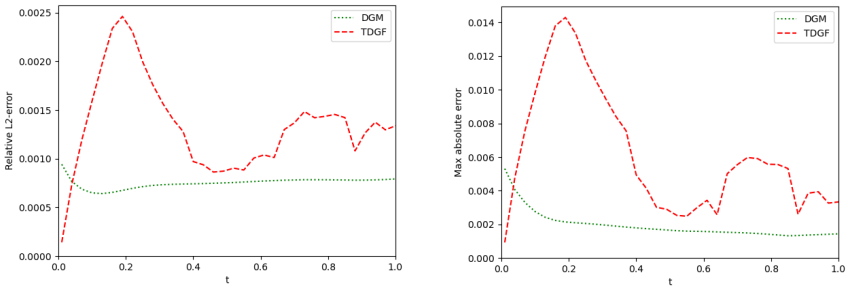
(a) Relative L^2-error (b) Maximal absolute error

Figure 3.4: Errors of the two methods in the lifted Heston model with $n = 1$ variance process against time, with $r = 0.0$, $\eta = 0.1$, $\rho = 0.0$, $\kappa = 0.01$, $V_0 = 0.01$ and $\lambda = 2.0$.



(a) Relative L^2-error (b) Maximal absolute error

Figure 3.5: Errors of the two methods in the lifted Heston model with $n = 20$ variance process against time, with $r = 0.0$, $\eta = 0.1$, $\rho = 0.0$, $\kappa = 0.01$, $V_0 = 0.01$ and $\lambda = 2.0$.



(a) Relative L^2-error

(b) Maximal absolute error

Figure 3.6: Errors of the two methods in the lifted Heston model with $n = 1$ variance process against time, with $r = 0.0$, $\eta = 0.3$, $\rho = -0.7$, $\kappa = 0.02$, $V_0 = 0.02$ and $\lambda = 0.3$.

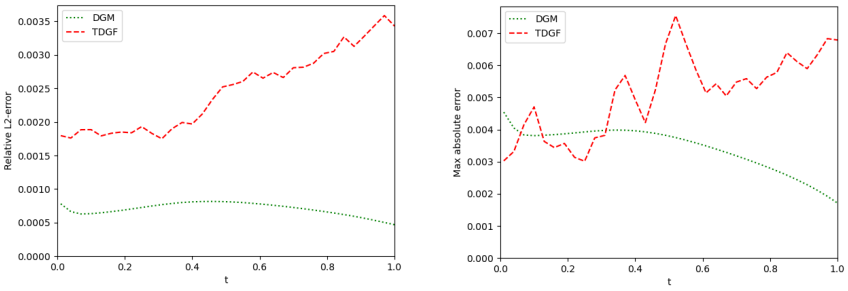
Fig. 3.6 presents the relative L^2- and the absolute maximum error for the two methods in the lifted Heston model, now for the parameter set suggested by Abi Jaber [1] with $n = 1$. The main difference here is that the correlation between the asset price and the variance processes is no longer zero. The error in both methods has slightly increased with respect to the previous example, with the DGM performing slightly better than the TDGF. However, the overall error is still small, in the order of 10^{-3}.

Finally, Fig. 3.7 presents the relative L^2- and the absolute maximum error for the two methods in the lifted Heston model, for the same parameter set as in the previous two figures, but now with $n = 5$ and $\rho = -0.2$. The combination of increased dimension (*i.e.* 5 variance processes compared to one before) and decreased correlation results in the errors of both methods being similar to the previous case, and still small, in the order of 10^{-3}. However, as the number of variance processes increase while the correlation is nonzero, then the errors in both methods are increasing rapidly, and we can no longer accommodate *e.g.* 20 variance factors as we did in the uncorrelated case.

3.6.2 Training and computational times

Table 3.1 summarizes the training times for the TDGF and the DGM methods in the different models. As we have expected, due to the time stepping and the absence of a second derivative in the cost function, the training of the TDGF method is faster than for the DGM method. Moreover, the training time for the TDGF method does not increase as the dimension of the problem (*i.e.* the number of variance processes) increases. On the contrary, the training time for the DGM method increases significantly with the number of dimensions, and becomes more than 4 times slower in the lifted Heston model when the dimension of the variance processes increases from 1 to 20.

Table 3.2 presents the computational times for the TDGF and the DGM methods in all models. The computational times are taken as the average over 34 time points



(a) Relative L^2-error (b) Maximal absolute error

Figure 3.7: Errors of the two methods in the lifted Heston model with $n = 5$ variance processes against time, with $r = 0.0$, $\eta = 0.3$, $\rho = -0.2$, $\kappa = 0.02$, $V_0 = 0.02$ and $\lambda = 0.3$.

Model	BS	Heston	LH, $n = 1$	LH, $n = 5$	LH, $n = 20$
DGM	7.0×10^3	12.2×10^3	12.5×10^3	21.0×10^3	54.3×10^3
TDGF	3.8×10^3	5.8×10^3	5.9×10^3	6.3×10^3	7.4×10^3

Table 3.1: Training time in seconds of the different methods for a European call option in the different models.

of computing the option prices on a grid of 47 moneyness points. Even though we take an average, there is still a high variance in the computational times. Both methods are significantly faster than the COS method in the lifted Heston model, in which we do not know the characteristic function explicitly and a system of Riccati equations needs to be solved for every evaluation of the characteristic function.

The possibility of splitting the computation of an option price in a slow, offline phase (training) and a fast, online one (computation), is one of the advantages of deep learning methods such as the TDGF proposed here and the DGM. However, when changing the model parameters, then the TDGF and the DGM need to be retrained, thus the overall time (training and computation) is slower than the evaluation of the COS method. The development of "deep parametric PDE" methods as in Glau and Wunderlich [43] would be an interesting avenue to explore in that respect. Finally, the TDGF requires more memory than the DGM, around 10-100MB, as it has to store a separate neural network for each time step, whereas the DGM needs only one neural network for all times; the increase in speed though justifies this cost.

3.7 Conclusion

We have developed in this chapter a novel deep learning method for option pricing in diffusion models. Starting from the option pricing PDE, using a backward differentiation time stepping scheme and results from the calculus of variations, we can rewrite the PDE as an energy minimization problem for a suitable energy func-

Model	BS	Heston	LH, $n = 1$	LH, $n = 5$	LH, $n = 20$
Exact/COS	0.0015	0.013	9.1	9.5	10.0
DGM	0.0086	0.0015	0.0043	0.0052	0.0015
TDGF	0.0018	0.0018	0.0019	0.0019	0.0018

Table 3.2: Computational time in seconds of the different methods for a European call option in the different models.

tional. The time stepping provides a good initial guess for the next step in the optimization procedure, while the energy formulation yields an equation that only has first derivatives, instead of second ones. The energy minimization problem offers a suitable candidate for a cost function, and the PDE is solved by training a neural network using stochastic gradient descent-type optimizers (*e.g.* Adam). We have considered as examples for our method the Black–Scholes model, the Heston model and the lifted Heston model, and derived the appropriate representations for the PDE in all these cases.

Overall, the TDGF method developed here performs well in the numerical experiments, with an error in the order of 10^{-3}. Moreover, the training and computation (evaluation) times for the TDGF method are faster and more consistent compared to the popular DGM method. However, when the dimension of the variance processes in the lifted Heston model increases and the correlation increases simultaneously, then the error also increases. This increase in the error is due to the diffusion term becoming smaller, which means that the explicit term in the decomposition (3.5) becomes the dominant term.

4

Time Deep Gradient Flow method for pricing American options

In the previous chapter, we developed a new neural network method to price options, called the Time Deep Gradient Flow method (TDGF), and used it to price a European option on a single asset in rough volatility models. Besides European options, which can only be exercised at maturity, there also exist American options which can be priced at any time before maturity. Furthermore, besides options on a single asset, there also exist options that depend on multiple different assets. These two extensions make the pricing of the option more complicated.

In this chapter, we explore neural network-based methods for pricing multidimensional American put options under the Black—Scholes and Heston model, extending up to five dimensions. We focus on two approaches: the TDGF and the Deep Galerkin Method (DGM). We extend the TDGF method to handle the free-boundary partial differential equation (PDE) inherent in American options. We carefully design the sampling strategy during training to enhance performance. Both TDGF and DGM achieve high accuracy while outperforming conventional Monte Carlo methods in terms of computational speed. In particular, TDGF tends to be faster during training than DGM.

4.1 Introduction

Pricing options is a fundamental problem in financial mathematics. In addition to European options, which can only be exercised at maturity, there exist American options, which can be exercised at any time before maturity. This early exercise feature introduces additional complexity, making the pricing of American options

This chapter is based on J. Rou. Time deep gradient flow method for pricing American options. Preprint arXiv:2507.17606, 2025.

more challenging than European options. One of the first successful methods for pricing American options is the binomial options pricing model introduced by Cox, Ross, and Rubinstein [27]. Another widely used approach formulates the price of an American option as the solution to a PDE with a free boundary or a system of variational inequalities; see Myneni [69] for a comprehensive overview.

As the number of underlying assets increases, the option pricing problem becomes high-dimensional, necessitating more efficient numerical methods. Clarke and Parrott [26] describe a multigrid procedure for a fast iterative solution to the pricing of American options. Longstaff and Schwartz [66] proposed a powerful simulation-based technique that approximates the value of American options using least squares regression. Ikonen and Toivanen [55] explored five distinct methods for pricing American options: the projected SOR method, a projected multigrid method, an operator splitting method, a penalty method, and a component-wise splitting method. For an overview of simulation-based methods, see Belomestny and Schoenmakers [20].

The development of deep learning has introduced new and powerful ways to solve the problem of pricing American options. Pioneering work in this direction includes Becker, Cheridito, and Jentzen [17, 18], Becker, Cheridito, Jentzen, and Welti [19], who developed deep learning approaches to learn optimal exercise strategies, pricing, and hedging of American options in high-dimensional settings. Other notable contributions include Herrera, Krach, Ruyssen, and Teichmann [48], who demonstrated the potential of randomized neural networks to outperform traditional deep neural networks and standard basis functions in approximating solutions to optimal stopping problems; Nwankwo, Umeorah, Ware, and Dai [70], who proposed a deep learning framework based on the Landau transformation to handle the free-boundary problem in American option pricing; and Peng, Wei, and Wei [74], who introduced a deep penalty method.

Sirignano and Spiliopoulos [81] proposed the DGM, which accurately solves high-dimensional free-boundary PDEs. Recently, Papapantoleon and Rou [72] introduced the TDGF method as a more efficient alternative to DGM to solve PDEs arising from European option pricing problems. In this chapter, we extend the TDGF method to handle free-boundary problems, allowing it to price American options. We compare the performance of DGM and TDGF in pricing American put options under the Black–Scholes and Heston model with up to five underlying assets, evaluating both accuracy and computational efficiency.

The remainder of the Chapter is organized as follows. Section 4.2, formulates the problem by defining the system of variational inequalities associated with American options and presenting the multidimensional Black–Scholes and Heston model. Section 4.3 describes the extension of the TDGF method to American options and introduces the specific neural network architecture and sampling methods used. Section 4.4 presents numerical results that compare accuracy and computational efficiency. Finally, Section 4.5 summarizes our findings.

4.2 Problem formulation

This section formulates the problem. Section 4.2.1 defines the system of variational inequalities associated with American options. Section 4.2.2 presents the multidimensional Black–Scholes model. Section 4.2.3 presents the multidimensional Heston model.

4.2.1 American options

Let $\mathbf{S} = (S_1, S_2, ..., S_d)$ denote the price processes of d financial assets that evolve according to a diffusion model, and consider an American derivative on $\mathbf{S}$ with payoff $\Psi(\mathbf{S}_t)$ at any time $t < T$, with maturity time $T > 0$. Let $u : [0,T] \times \Omega \to \mathbb{R}$ denote the price of the American derivative, with $\Omega \subseteq \mathbb{R}^d$ and t the time to maturity. Then, u solves the system of inequalities [51]:

$$\begin{aligned} \frac{\partial u}{\partial t} + \mathcal{A}u + ru &\geq 0, && (t,\mathbf{x}) \in [0,T] \times \Omega, \\ u(t,\mathbf{x}) &\geq \Psi(\mathbf{x}), && (t,\mathbf{x}) \in [0,T] \times \Omega, \\ u(0,\mathbf{x}) &= \Psi(\mathbf{x}), && \mathbf{x} \in \Omega, \\ \left(\frac{\partial u}{\partial t} + \mathcal{A}u + ru\right)(u(t,\mathbf{x}) - \Psi(\mathbf{x})) &= 0, && (t,\mathbf{x}) \in [0,T] \times \Omega, \end{aligned} \tag{4.1}$$

with $\mathcal{A}$ a second-order differential operator of the form

$$\mathcal{A}u = -\sum_{i,j=1}^{d} a^{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^{d} \beta^i \frac{\partial u}{\partial x_i}. \tag{4.2}$$

The coefficients a^{ij}, β^i of the generator $\mathcal{A}$ relate directly to the dynamics of the stochastic processes $\mathbf{S}$ and can, in general, depend on the time and the spatial variables.

Problem (4.1) is equivalent to the free-boundary problem:

$$\begin{aligned} \max\left\{-\frac{\partial u}{\partial t} - \mathcal{A}u - ru, \Psi(\mathbf{x}) - u(t,\mathbf{x})\right\} &= 0, \\ u(0,\mathbf{x}) &= \Psi(\mathbf{x}). \end{aligned}$$

The TDGF reformulates the PDE as an energy minimization problem, which is then approximated in a time-stepping fashion by deep neural networks. In order to write the PDE as an energy minimization problem, we need to split the operator in a symmetric and an (asymmetric) remainder part. Following Chapter 3, we can rewrite the operator $\mathcal{A}$ as

$$\mathcal{A}u = -\nabla \cdot (A \nabla u) + \mathbf{b} \cdot \nabla u, \tag{4.3}$$

with a symmetric positive semidefinite matrix

$$A = \begin{bmatrix} a^{11} & \dots & a^{d1} \\ \vdots & \ddots & \vdots \\ a^{1d} & \dots & a^{dd} \end{bmatrix} \quad \text{and vector} \quad \mathbf{b} = \begin{bmatrix} b^1 \\ \vdots \\ b^d \end{bmatrix}. \tag{4.4}$$

4.2.2 Multidimensional Black–Scholes model

In the model by Black and Scholes [23], the dynamics of the stock price S follow a geometric Brownian motion. Suppose we have d assets, each following the Black–Scholes model:

$$\mathrm{d}S_i(t) = rS_i(t)\mathrm{d}t + \sigma_i S_i(t)\mathrm{d}W_i(t)_t, \quad S_i(0) > 0,$$

with $r > 0$ the risk-free rate, $\sigma_i > 0$ the volatility of asset S_i and $[W_1(t), ..., W_d(t)]$ Brownian motions with correlation matrix

$$\begin{bmatrix} 1 & \rho_{12} & ... & \rho_{1d} \\ \rho_{12} & 1 & ... & \rho_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1d} & \rho_{2d} & ... & 1 \end{bmatrix}.$$

The generator corresponding to these dynamics, in the form (4.2), equals

$$\mathcal{A}u = -\sum_{i=1}^{d} rS_i \frac{\partial u}{\partial S_i} - \frac{1}{2}\sum_{i=1,j=1}^{d} \sigma_i\sigma_j S_i S_j \rho_{ij} \frac{\partial^2 u}{\partial S_i \partial S_j}.$$

Applying the product rule gives:

$$\begin{aligned}
\mathcal{A}u =& -\sum_{i=1}^{d} rS_i \frac{\partial u}{\partial S_i} - \frac{1}{2}\sum_{i=1}^{d} \sigma_i^2 S_i^2 \frac{\partial^2 u}{\partial S_i^2} - \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i} \sigma_i\sigma_j S_i S_j \rho_{ij} \frac{\partial^2 u}{\partial S_i \partial S_j} \\
=& -\sum_{i=1}^{d} rS_i \frac{\partial u}{\partial S_i} - \frac{1}{2}\sum_{i=1}^{d} \frac{\partial}{\partial S_i}\left(\sigma_i^2 S_i^2 \frac{\partial u}{\partial S_i}\right) + \sum_{i=1}^{d} \sigma_i^2 S_i \frac{\partial u}{\partial S_i} \\
& - \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i} \frac{\partial}{\partial S_j}\left(\sigma_i\sigma_j S_i S_j \rho_{ij} \frac{\partial u}{\partial S_i}\right) + \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i} \sigma_i\sigma_j S_i \rho_{ij} \frac{\partial u}{\partial S_i} \\
=& \sum_{i=1}^{d}\left(\sigma_i^2 + \frac{1}{2}\sum_{j\neq i} \sigma_i\sigma_j\rho_{ij} - r\right) S_i \frac{\partial u}{\partial S_i} - \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial}{\partial S_j}\left(\sigma_i\sigma_j S_i S_j \rho_{ij} \frac{\partial u}{\partial S_i}\right).
\end{aligned}$$

Therefore, the operator $\mathcal{A}$ takes the form (4.3) with the coefficients in (4.4) provided by

$$\begin{aligned}
a^i &= \frac{1}{2}\sigma_i\sigma_j S_i S_j \rho_{ij}, && i = 1, ..., d, \\
b^i &= \left(\sigma_i^2 + \frac{1}{2}\sum_{j\neq i} \sigma_i\sigma_j\rho_{ij} - r\right) S_i, && i = 1, ..., d.
\end{aligned}$$

4.2.3 Multidimensional Heston model

The model by Heston [49] is a popular stochastic volatility model. In d dimensions the dynamics of asset S and variance process V are

$$\begin{aligned} \mathrm{d}S_i(t) &= rS_i(t)\mathrm{d}t + \sqrt{V_i(t)}S_i(t)\mathrm{d}W_i(t), & S_i(0) > 0, \\ \mathrm{d}V_i(t) &= \lambda_i\left(\kappa_i - V_i(t)\right)\mathrm{d}t + \eta_i\sqrt{V_i(t)}\mathrm{d}B_i(t), & V_i(0) > 0. \end{aligned}$$

Here and $\lambda, \kappa, \eta \in \mathbb{R}_+$ and $[W_1(t), ..., W_d(t), B_1(t), ..., B_d(t)]$ are Brownian motions, with correlation matrix:

$$\begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1d} & \hat{\rho}_{11} & \hat{\rho}_{12} & \cdots & \hat{\rho}_{1d} \\ \rho_{12} & 1 & \cdots & \rho_{2d} & \hat{\rho}_{21} & \hat{\rho}_{22} & \cdots & \hat{\rho}_{2d} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{1d} & \rho_{2d} & \cdots & 1 & \hat{\rho}_{d1} & \hat{\rho}_{d2} & \cdots & \hat{\rho}_{dd} \\ \hat{\rho}_{11} & \hat{\rho}_{21} & \cdots & \hat{\rho}_{d1} & 1 & \tilde{\rho}_{12} & \cdots & \tilde{\rho}_{1d} \\ \hat{\rho}_{12} & \hat{\rho}_{22} & \cdots & \hat{\rho}_{d2} & \tilde{\rho}_{12} & 1 & \cdots & \tilde{\rho}_{2d} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{\rho}_{1d} & \hat{\rho}_{2d} & \cdots & \hat{\rho}_{dd} & \tilde{\rho}_{1d} & \tilde{\rho}_{2d} & \cdots & 1 \end{bmatrix},$$

with ρ_{ij} the correlation between W_i and W_j, $\tilde{\rho}_{ij}$ the correlation between B_i and B_j and $\hat{\rho}_{ij}$ the correlation between B_i and W_j.

Let $f\left(S_1(t), ..., S_d(t), V_1(t), ..., V_d(t)\right) : \mathbb{R}^{2d} \to \mathbb{R}$ be a $\mathcal{C}^2$-function. Then Itô's formula gives

$$\begin{aligned} &\mathrm{d}f\left(S_1(t), ..., S_d(t), V_1(t), ..., V_d(t)\right) \\ &= \sum_{i=1}^{d} \frac{\partial f}{\partial S_i}\mathrm{d}S_i + \sum_{i=1}^{d} \frac{\partial f}{\partial V_i}\mathrm{d}V_i + \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial S_i \partial S_j}\mathrm{d}\left\langle S_i, S_j\right\rangle + \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial S_i \partial V_j}\mathrm{d}\left\langle S_i, V_j\right\rangle \\ &\quad + \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial V_i \partial V_j}\mathrm{d}\left\langle V_i, V_j\right\rangle \\ &= \sum_{i=1}^{d} \frac{\partial f}{\partial S_i} rS_i\mathrm{d}t + \sum_{i=1}^{d} \frac{\partial f}{\partial V_i}\lambda_i\left(\kappa_i - V_i\right)\mathrm{d}t + \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial S_i \partial S_j}\sqrt{V_i V_j}S_i S_j \rho_{ij}\mathrm{d}t \\ &\quad + \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial S_i \partial V_j}\sqrt{V_i V_j}S_i \eta_j \hat{\rho}_{ij}\mathrm{d}t + \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial V_i \partial V_j}\eta_i \eta_j \sqrt{V_i V_j}\tilde{\rho}_{ij}\mathrm{d}t + \text{martingale}. \end{aligned}$$

Then the generator corresponding to these dynamics, in the form (4.2), equals

$$\begin{aligned} \mathcal{A}u = &- \sum_{i=1}^{d} \frac{\partial u}{\partial S_i} rS_i - \sum_{i=1}^{d} \frac{\partial u}{\partial V_i}\lambda_i\left(\kappa_i - V_i\right) - \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2 u}{\partial S_i \partial S_j}\sqrt{V_i V_j}S_i S_j \rho_{ij} \\ &- \sum_{i,j=1}^{d} \frac{\partial^2 u}{\partial S_i \partial V_j}\sqrt{V_i V_j}S_i \eta_j \hat{\rho}_{ij} - \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2 u}{\partial V_i \partial V_j}\eta_i \eta_j \sqrt{V_i V_j}\tilde{\rho}_{ij}. \end{aligned}$$

Applying the product rule gives:

$$
\begin{aligned}
\mathcal{A}u = &-\sum_{i=1}^{d}\frac{\partial u}{\partial S_i}rS_i - \sum_{i=1}^{d}\frac{\partial u}{\partial V_i}\lambda_i\left(\kappa_i - V_i\right) - \frac{1}{2}\sum_{i=1}^{d}\frac{\partial^2 u}{\partial S_i^2}V_iS_i^2 \\
&- \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i}\frac{\partial^2 u}{\partial S_i \partial S_j}\sqrt{V_iV_j}S_iS_j\rho_{ij} - \sum_{i=1}^{d}\frac{\partial^2 u}{\partial S_i \partial V_i}V_iS_i\eta_i\hat{\rho}_{ii} \\
&- \sum_{i=1}^{d}\sum_{j\neq i}\frac{\partial^2 u}{\partial S_i \partial V_j}\sqrt{V_iV_j}S_i\eta_j\hat{\rho}_{ij} - \frac{1}{2}\sum_{i=1}^{d}\frac{\partial^2 u}{\partial V_i^2}\eta_i^2V_i \\
&- \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i}\frac{\partial^2 u}{\partial V_i \partial V_j}\eta_i\eta_j\sqrt{V_iV_j}\tilde{\rho}_{ij} \\
= &-\sum_{i=1}^{d}\frac{\partial u}{\partial S_i}rS_i - \sum_{i=1}^{d}\frac{\partial u}{\partial V_i}\lambda_i\left(\kappa_i - V_i\right) - \frac{1}{2}\sum_{i=1}^{d}\frac{\partial}{\partial S_i}\left(\frac{\partial u}{\partial S_i}V_iS_i^2\right) + \sum_{i=1}^{d}\frac{\partial u}{\partial S_i}V_iS_i \\
&- \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i}\frac{\partial}{\partial S_j}\left(\frac{\partial u}{\partial S_i}\rho_{ij}\sqrt{V_iV_j}S_iS_j\right) + \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i}\frac{\partial u}{\partial S_i}\rho_{ij}\sqrt{V_iV_j}S_i \\
&- \frac{1}{2}\sum_{i=1}^{d}\frac{\partial}{\partial S_i}\left(\frac{\partial u}{\partial V_i}V_iS_i\eta_i\hat{\rho}_{ii}\right) + \frac{1}{2}\sum_{i=1}^{d}\frac{\partial u}{\partial V_i}V_i\eta_i\hat{\rho}_{ii} \\
&- \frac{1}{2}\sum_{i=1}^{d}\frac{\partial}{\partial V_i}\left(\frac{\partial u}{\partial S_i}V_iS_i\eta_i\hat{\rho}_{ii}\right) + \frac{1}{2}\sum_{i=1}^{d}\frac{\partial u}{\partial S_i}S_i\eta_i\hat{\rho}_{ii} \\
&- \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i}\frac{\partial}{\partial S_i}\left(\frac{\partial u}{\partial V_j}\sqrt{V_iV_j}S_i\eta_j\hat{\rho}_{ij}\right) + \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i}\frac{\partial u}{\partial V_j}\sqrt{V_iV_j}\eta_j\hat{\rho}_{ij} \\
&- \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i}\frac{\partial}{\partial V_j}\left(\frac{\partial u}{\partial S_i}\sqrt{V_iV_j}S_i\eta_j\hat{\rho}_{ij}\right) + \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i}\frac{\partial u}{\partial S_i}\frac{\sqrt{V_i}}{2\sqrt{V_j}}S_i\eta_j\hat{\rho}_{ij} \\
&- \frac{1}{2}\sum_{i=1}^{d}\frac{\partial}{\partial V_i}\left(\frac{\partial u}{\partial V_i}\eta_i^2V_i\right) + \frac{1}{2}\sum_{i=1}^{d}\frac{\partial u}{\partial V_i}\eta_i^2 \\
&- \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i}\frac{\partial}{\partial V_i}\left(\frac{\partial u}{\partial V_j}\eta_i\eta_j\sqrt{V_iV_j}\tilde{\rho}_{ij}\right) + \frac{1}{2}\sum_{i=1}^{d}\sum_{j\neq i}\frac{\partial u}{\partial V_j}\eta_i\eta_j\frac{1}{2\sqrt{V_i}}\sqrt{V_j}\tilde{\rho}_{ij}
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{i=1}^{d} \left(-r + V_i + \frac{1}{2} \sum_{j \neq i} \rho_{ij} \sqrt{V_i V_j} + \frac{1}{2} \eta_i \hat{\rho}_{ii} + \frac{1}{4} \sum_{j \neq i} \sqrt{\frac{V_i}{V_j}} \eta_j \hat{\rho}_{ij} \right) S_i \frac{\partial u}{\partial S_i} \\
&+ \sum_{i=1}^{d} \left(\lambda_i \left(V_i - \kappa_i \right) + \frac{1}{2} \sum_{j=1}^{d} \sqrt{V_i V_j} \eta_i \hat{\rho}_{ji} + \frac{1}{2} \eta_i^2 + \frac{1}{4} \sum_{j \neq i} \eta_i \eta_j \sqrt{\frac{V_i}{V_j}} \tilde{\rho}_{ij} \right) \frac{\partial u}{\partial V_i} \\
&- \frac{1}{2} \sum_{i=1}^{d} \frac{\partial}{\partial S_i} \left(\frac{\partial u}{\partial S_i} V_i S_i^2 \right) - \frac{1}{2} \sum_{i=1}^{d} \sum_{j \neq i} \frac{\partial}{\partial S_j} \left(\frac{\partial u}{\partial S_i} \rho_{ij} \sqrt{V_i V_j} S_i S_j \right) \\
&- \frac{1}{2} \sum_{i=1}^{d} \frac{\partial}{\partial S_i} \left(\frac{\partial u}{\partial V_i} V_i S_i \eta_i \hat{\rho}_{ii} \right) - \frac{1}{2} \sum_{i=1}^{d} \frac{\partial}{\partial V_i} \left(\frac{\partial u}{\partial S_i} V_i S_i \eta_i \hat{\rho}_{ii} \right) \\
&- \frac{1}{2} \sum_{i=1}^{d} \sum_{j \neq i} \frac{\partial}{\partial S_i} \left(\frac{\partial u}{\partial V_j} \sqrt{V_i V_j} S_i \eta_j \hat{\rho}_{ij} \right) - \frac{1}{2} \sum_{i=1}^{d} \sum_{j \neq i} \frac{\partial}{\partial V_j} \left(\frac{\partial u}{\partial S_i} \sqrt{V_i V_j} S_i \eta_j \hat{\rho}_{ij} \right) \\
&- \frac{1}{2} \sum_{i=1}^{d} \frac{\partial}{\partial V_i} \left(\frac{\partial u}{\partial V_i} \eta_i^2 V_i \right) - \frac{1}{2} \sum_{i=1}^{d} \sum_{j \neq i} \frac{\partial}{\partial V_i} \left(\frac{\partial u}{\partial V_j} \eta_i \eta_j \sqrt{V_i V_j} \tilde{\rho}_{ij} \right)
\end{aligned}
$$

Therefore, the operator $\mathcal{A}$ takes the form (4.3) with the coefficients in (4.4) provided by

$$
\begin{aligned}
a^{ij} &= \frac{1}{2} \rho_{ij} \sqrt{V_i V_j} S_i S_j, \quad i, j = 1, ..., d, \\
a^{i,j+d} = a^{j+d,i} &= \frac{1}{2} \sqrt{V_i V_j} S_i \eta_j \hat{\rho}_{ij}, \quad i, j = 1, ..., d, \\
a^{i+d,j+d} &= \frac{1}{2} \eta_i \eta_j \sqrt{V_i V_j} \tilde{\rho}_{ij}, \quad i, j = 1, ..., d,
\end{aligned}
$$

$$
b^i = \left(-r + V_i + \frac{1}{2} \sum_{j \neq i} \rho_{ij} \sqrt{V_i V_j} + \frac{1}{2} \eta_i \hat{\rho}_{ii} + \frac{1}{4} \sum_{j \neq i} \sqrt{\frac{V_i}{V_j}} \eta_j \hat{\rho}_{ij} \right) S_i, \quad i = 1, ..., d,
$$

$$
b^{i+d} = \lambda_i \left(V_i - \kappa_i \right) + \frac{1}{2} \sum_{j=1}^{d} \sqrt{V_i V_j} \eta_i \hat{\rho}_{ji} + \frac{1}{2} \eta_i^2 + \frac{1}{4} \sum_{j \neq i} \eta_i \eta_j \sqrt{\frac{V_i}{V_j}} \tilde{\rho}_{ij}, \quad i = 1, ..., d.
$$

4.3 Methodology

This section provides the details on how to solve the problem from the previous section. Section 4.3.1 describes the extension of the TDGF method to American options. Section 4.3.2 introduces the specific neural network architecture used. Section 4.3.3 introduces the specific sampling methods used.

4.3.1 Time Deep Gradient Flow method

The TDGF is a neural network method to efficiently solve high-dimensional PDEs [41, 72]. Let us divide the time interval $[0, T]$ into K equally spaced intervals $(t_{k-1}, t_k]$, with $h = t_k - t_{k-1} = \frac{1}{K}$ for $k = 0, 1, \ldots, K$. By first discretizing the

PDE in time and then rewriting the discretized PDE as an energy functional we can approximate the solution to the PDE

$$\begin{aligned}\frac{\partial u}{\partial t} - \nabla \cdot (A \nabla u) + \mathbf{b} \cdot \nabla u + ru = 0,\\ u(0) = \Psi,\end{aligned}$$

by

$$\begin{aligned}u(t_k, \mathbf{x}) \approx & U^k = \arg\min I^k(u),\\ I^k(u) = & \frac{1}{2} \left\| u - U^{k-1} \right\|^2_{L^2(\Omega)} + h \left(\int_\Omega \frac{1}{2} \left((\nabla u)^\mathsf{T} A \nabla u + r u^2 \right) + F\left(U^{k-1}\right) u \mathrm{d}x \right),\\ U^0 = & \Psi.\end{aligned}$$

Let $f^k(\mathbf{x}; \theta)$ denote a neural network approximation of U^k with trainable parameters θ. Applying a Monte Carlo approximation to the integrals, the discretized cost functional takes the form

$$L^k(\theta; \mathbf{x}) = \frac{|\Omega|}{2M} \sum_{m=1}^{M} \left(f^k(\mathbf{x}_m; \theta) - f^{k-1}(\mathbf{x}_m) \right)^2 + h N^k(\theta; \mathbf{x}),$$

with

$$\begin{aligned}N^k(\theta; \mathbf{x}) = & \frac{|\Omega|}{M} \sum_{m=1}^{M} \left[\frac{1}{2} \left(\left(\nabla f^k(\mathbf{x}_m; \theta) \right)^\mathsf{T} A \nabla f^k(\mathbf{x}_m; \theta) + r \left(f^k(\mathbf{x}_m; \theta) \right)^2 \right) \right.\\ & \left. + \left(\mathbf{b} \cdot \nabla f^{k-1}(\mathbf{x}_m) \right) f^k(\mathbf{x}_m; \theta) \right].\end{aligned}$$

Here, M denotes the number of samples $\mathbf{x}_m$. From equation (4.1), the PDE is satisfied if the solution u is strictly larger than the payoff Ψ. Therefore, we only train the PDE on the part of the domain where the solution is above the payoff.

In order to minimize this cost function, we use a stochastic gradient descent-type algorithm, *i.e.* an iterative scheme of the form:

$$\theta_{n+1} = \theta_n - \alpha_n \nabla_\theta L^k(\theta_n; \mathbf{x}).$$

The hyperparameter α_n is the step size of our update, called the learning rate. An overview of the TDGF appears in Algorithm 2.

4.3.2 Architecture

Let us now describe some details about the design of the neural network architecture and the implementation of the numerical method. As in Section 3.5, we would like to use information about the option price in order to facilitate the training of the neural network. The price of an American option can be decomposed in two (positive) values: the intrinsic value and the continuation value. The intrinsic value

Algorithm 2 Time Deep Gradient Flow method for American Options

Initialize θ_0^0.
for each sampling stage n **do**
  Generate random points $\mathbf{x}_m$ for training.
  Calculate the cost functional $L^0(\theta_n^0;\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M}\left(f^0(\mathbf{x}_m;\theta_n^0) - \Psi(\mathbf{x}_m)\right)^2$ for the selected points.
  Take a descent step $\theta_{n+1}^0 = \theta_n^0 - \alpha_n \nabla_\theta L^0(\theta_n^0;\mathbf{x})$.
end for
for each time step $k = 1,\dots,K$ **do**
  Initialize $\theta_0^k = \theta^{k-1}$.
  for each sampling stage n **do**
    Generate random points $\mathbf{x}_m$ for training.
    Select the points $\mathbf{x}_m$ where $f^k(\mathbf{x}_m) > \Psi(\mathbf{x}_m)$.
    Calculate the cost functional $L^k(\theta_n^k;\mathbf{x})$ for the selected points.
    Take a descent step $\theta_{n+1}^k = \theta_n^k - \alpha_n \nabla_\theta L^k(\theta_n^k;\mathbf{x})$.
  end for
end for

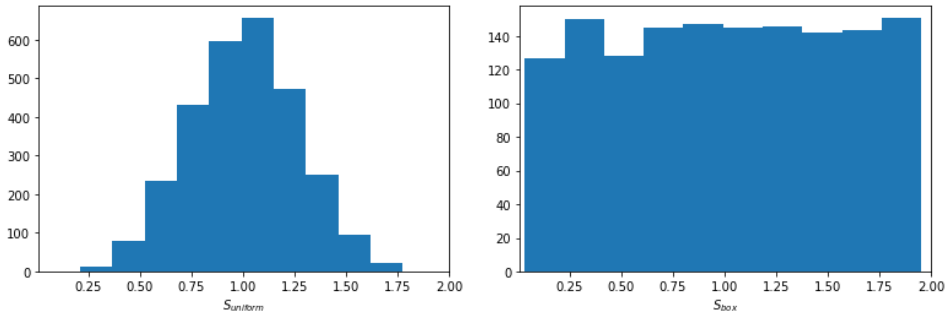
is the value of the option if we exercise, which we know to be Ψ. The continuation value is the value of the option if we do not exercise and let the stock continue following the PDE. From the second line of (4.1) we know $u \geq \Psi$. The neural network learns the continuation value, instead of the option price itself.

The architecture of the neural network for the TDGF method in case of American options follows (3.7). Overall, we set:

$$\begin{aligned}
X^1 &= \sigma_1\left(W^1\mathbf{x} + b^1\right), \\
Z^l &= \sigma_1\left(U^{z,l}\mathbf{x} + W^{z,l}X^l + b^{z,l}\right), && l = 1,\dots,L, \\
G^l &= \sigma_1\left(U^{g,l}\mathbf{x} + W^{g,l}X^l + b^{g,l}\right), && l = 1,\dots,L, \\
R^l &= \sigma_1\left(U^{r,l}\mathbf{x} + W^{r,l}X^l + b^{r,l}\right), && l = 1,\dots,L, \\
H^l &= \sigma_1\left(U^{h,l}\mathbf{x} + W^{h,l}\left(X^l \odot R^l\right) + b^{h,l}\right), && l = 1,\dots,L, \\
X^{l+1} &= \left(1 - G^l\right) \odot H^l + Z^l \odot X^l, && l = 1,\dots,L, \\
f(\mathbf{x};\theta) &= \Psi + \sigma_2\left(WX^{L+1} + b\right).
\end{aligned}$$

Here, L is the number of hidden layers, σ_i is the activation function for $i = 1, 2$. In the numerical experiments, we use 3 layers and 50 neurons per layer. The activation functions are the hyperbolic tangent function, $\sigma_1(x) = \tanh(x)$, and the softplus function, $\sigma_2(x) = \log\left(e^x + 1\right)$, which guarantees that the option price remains above the no-arbitrage bound.

We consider a maturity of $T = 1.0$ year, and take the number of time steps equal to $K = 100$. We use 2000 sampling stages in each time step. For the optimization we use Adam algorithm [61] with a learning rate $\alpha = 3\times10^{-4}$, $(\beta_1, \beta_2) = (0.9, 0.999)$ and zero weight decay. The training is performed on the DelftBlue supercomputer [29], using a single NVidia Tesla V100S GPU.



(a) Uniform (b) 19 boxes

Figure 4.1: Histogram of the moneyness for 5 dimensions with 2850 samples for different sampling methods.

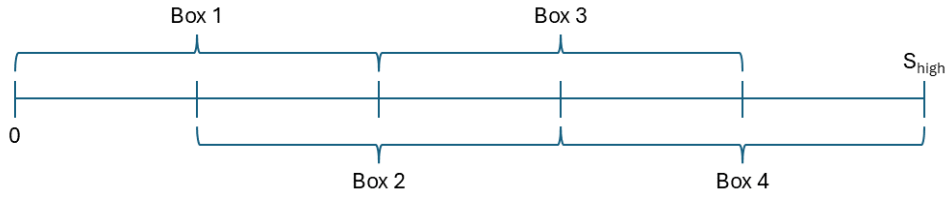


Figure 4.2: Sampling domain with 4 boxes.

4.3.3 Sampling

For the sampling we have to be particularly careful in the multidimensional case. We assign equal weight to each asset, therefore the moneyness of the option is the average of the individual moneynesses:

$$S = \frac{1}{d}\sum_{i=1}^{d} S_i.$$

If we sample each S_i uniformly we obtain a histogram of the moneyness as in Fig. 4.1a. There are barely samples at the edges of the domain, therefore the network does not learn the solution in this area.

To cope with this issue, we split the domain in $n-1$ smaller boxes

$$\left[0, \frac{2S_{high}}{n}\right], \left[\frac{S_{high}}{n}, \frac{3S_{high}}{n}\right], ..., \left[\frac{(n-2)S_{high}}{n}, S_{high}\right]$$

and take samples from each box separately. Fig. 4.2 displays an example of the domain and boxes for $n = 5$. Fig. 4.1b displays the moneyness using this box sampling.

For the TDGF, during training of the time steps we are only concerned with points where the neural network is larger than the payoff. Therefore, for the TDGF we

apply initial training with box sampling and during the time steps we apply uniform sampling. In each sampling stage we take 30 samples per box per dimension ($30d$ for Black–Scholes, $60d$ for Heston) and use 19 boxes.

In the experiments, we choose the parameters such that in the Black–Scholes model, the continuation value is larger than in the Heston model. In the Heston model, the continuation value is already zero for moneyness larger than 1.5, while for Black–Scholes, the continuation value can be positive for moneyness beyond 2. Numerical experiments suggest therefore that for better results we consider the sampling domain of the moneyness $S \in [0.01, 3.0]$ for Black–Scholes and $S \in [0.01, 2.0]$ for Heston. The domain of the Heston volatility is $V \in [0.001, 0.1]$.

4.4 Numerical results

Since we do not consider dividends and assume $r \geq 0$ the best exercise strategy for an American call option is to wait until maturity. Therefore, the price of an American call is the same as a European call [68]. So we consider an American basket put with equal weights for each asset: $\Psi(\mathbf{S}) = \left(K - \frac{1}{d}\sum_{i=1}^{d} S_i\right)^+$.

We compare the TDGF method with the DGM [81]. In the DGM approach, we minimize the L^2-error of the free-boundary PDE:

$$\left\| \max\left\{ -\frac{\partial u}{\partial t} - \mathcal{A}u - ru, \Psi(\mathbf{x}) - u(t,\mathbf{x}) \right\} \right\|^2_{L^2([0,T]\times\Omega)} + \|u(0,\mathbf{x}) - \Psi(\mathbf{x})\|^2_{L^2(\Omega)}\,.$$

To have a fair comparison between the two methods, we use 200,000 sampling stages and the same learning rate $\alpha = 3 \times 10^{-4}$ for the DGM.

In order to evaluate the accuracy of the two methods, we need a reference value. We compute 1,000 Monte Carlo paths with 1,000 time steps and apply the method of Longstaff and Schwartz [66], which applies a polynomial regression of order 4 on the paths where the intrinsic value is positive.

When evaluating, we plot the continuation value against the moneyness on an equidistant grid of 47 points where the moneyness and volatility in each dimension are the same.

For the correlation matrices, we take the correlation between two different stocks $\rho_{ij} = \frac{1}{2}$; between a stock and its variance $\hat{\rho}_{ii} = -\frac{1}{2}$; between two different variances 0 and between a stock and a different variance, 0. Then the correlation matrices in the Black–Scholes and Heston model are positive semi-definite and therefore valid correlation matrices.

4.4.1 Accuracy

Fig. 4.3 presents the difference between the option price and the payoff against moneyness in the two-dimensional Black–Scholes model. Fig. 4.4 presents the difference between the option price and the payoff against moneyness in the five-dimensional Black–Scholes model. All three methods display similar values and therefore both DGM and TDGF give accurate results.

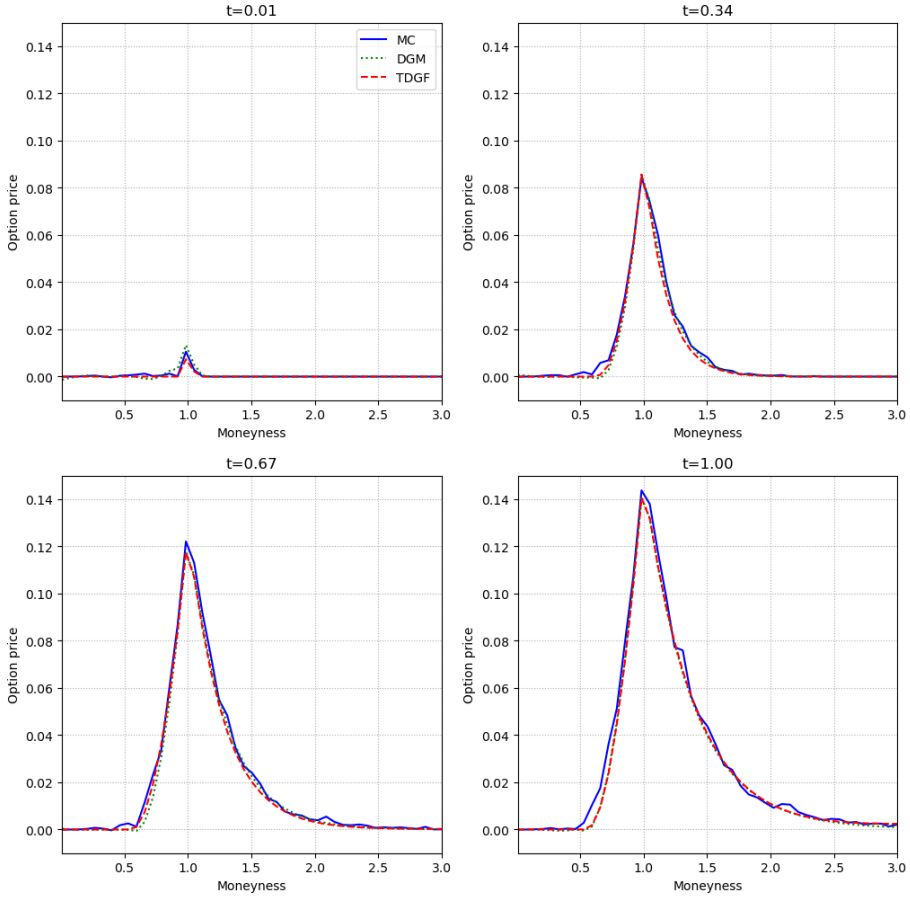


Figure 4.3: Difference between the option price and the payoff in the two-dimensional Black–Scholes model against the moneyness of the stock, compared to the DGM and Monte Carlo with Longstaff–Schwartz methods, for four different times to maturity with $r = 0.05$ and $\sigma_i = 0.5$ and $\rho_{ij} = 0.5$ for each i and j.

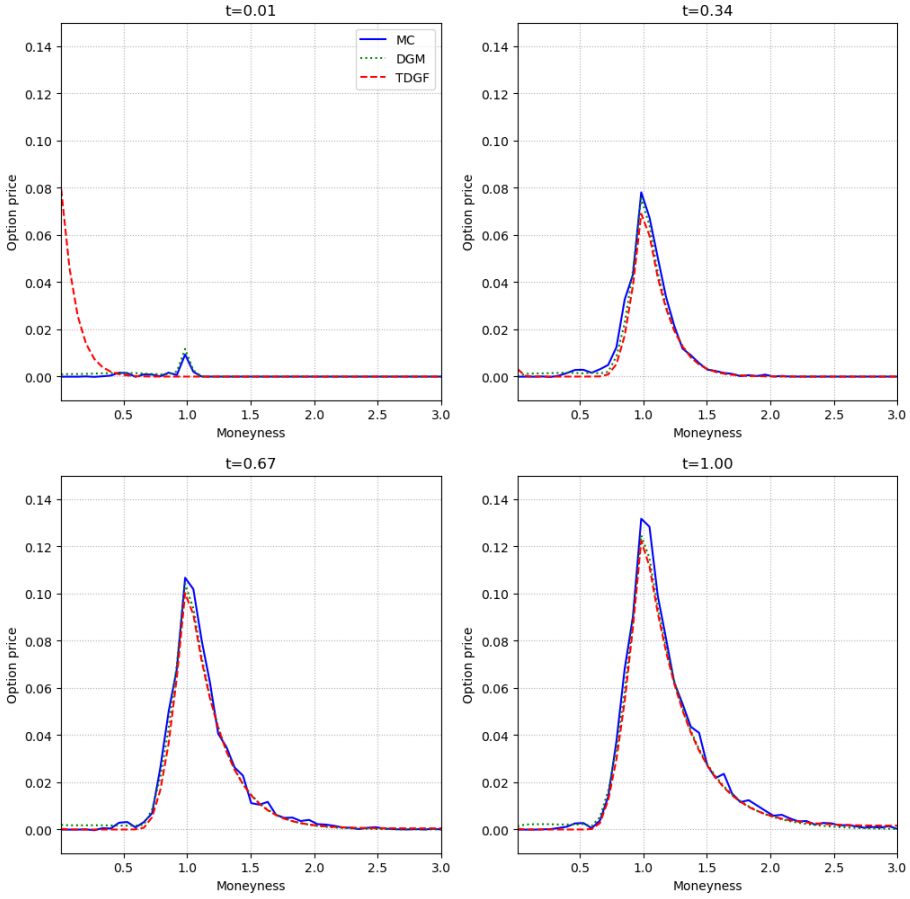


Figure 4.4: Difference between the option price and the payoff in the five-dimensional Black–Scholes model against the moneyness of the stock, compared to the DGM and Monte Carlo with Longstaff–Schwartz methods, for four different times to maturity with $r = 0.05$ and $\sigma_i = 0.5$ and $\rho_{ij} = 0.5$ for each i and j.

Model	BS $d = 2$	BS $d = 5$	Heston $d = 2$	Heston $d = 5$
DGM	8293	16174	17997	41718
TDGF	4543	6583	7138	12881

Table 4.1: Training time in seconds of the different methods for an American put option in the different models.

Model	BS $d = 2$	BS $d = 5$	Heston $d = 2$	Heston $d = 5$
MC	4.6	4.5	6.0	6.6
DGM	0.0015	0.0024	0.0015	0.0016
TDGF	0.0018	0.0017	0.0016	0.0017

Table 4.2: Computational time in seconds of the different methods for an American put option in the different models.

Fig. 4.5 presents the difference between the option price and the payoff against moneyness in the two-dimensional Heston model. Fig. 4.6 presents the difference between the option price and the payoff against moneyness in the five-dimensional Heston model. All three methods display similar values and therefore both DGM and TDGF give accurate results.

4.4.2 Training and computational times

Table 4.1 summarizes the training times for the TDGF and the DGM methods in the different models. As expected, due to the time stepping and the absence of a second derivative in the cost function, the training of the TDGF method is faster than for the DGM method.

Table 4.2 presents the computational times for the TDGF and the DGM in all models. The computational times are the average over 34 computations at different time points. Both methods are faster than the Monte Carlo method.

4.5 Conclusion

In this chapter, we explored neural network-based methods for pricing multidimensional American put options under the Black–Scholes and Heston models, extending up to five dimensions. We focused on two approaches: the TDGF method and the DGM.

We extended the TDGF method to handle the free-boundary PDE inherent in American options. We restricted training to the region where the PDE holds and incorporated the lower bound constraint directly into the network architecture. Additionally, we carefully designed the sampling strategy during training to enhance performance.

Both TDGF and DGM achieve high accuracy while outperforming conventional Monte Carlo methods in terms of computational speed. Notably, TDGF tends to be faster during training than DGM.

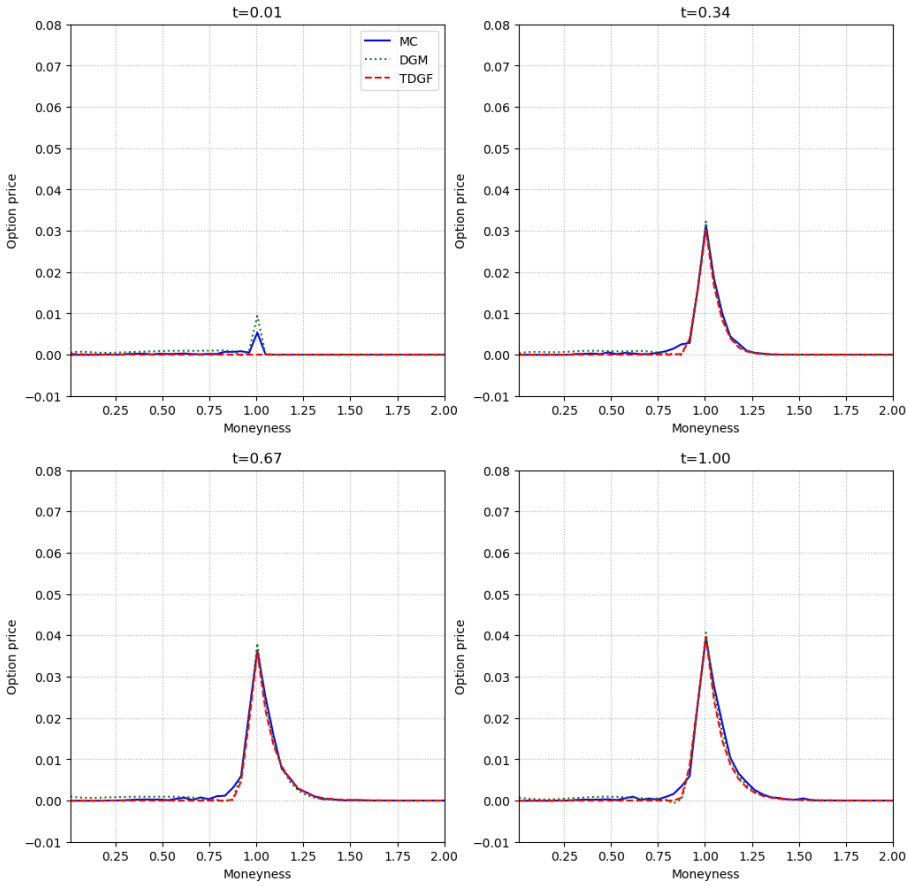


Figure 4.5: Difference between the option price and the payoff in the two-dimensional Heston model against the moneyness of the stock, compared to the DGM and Monte Carlo with Longstaff–Schwartz methods, for four different times to maturity with $r = 0.05$ and $\eta_i = 0.1$, $\rho_{ij} = 0.5$, $\hat{\rho}_{ii} = -0.5$, $\kappa_i = 0.01$, $V_0^i = 0.05$ and $\lambda_i = 2.0$ for each i and j.

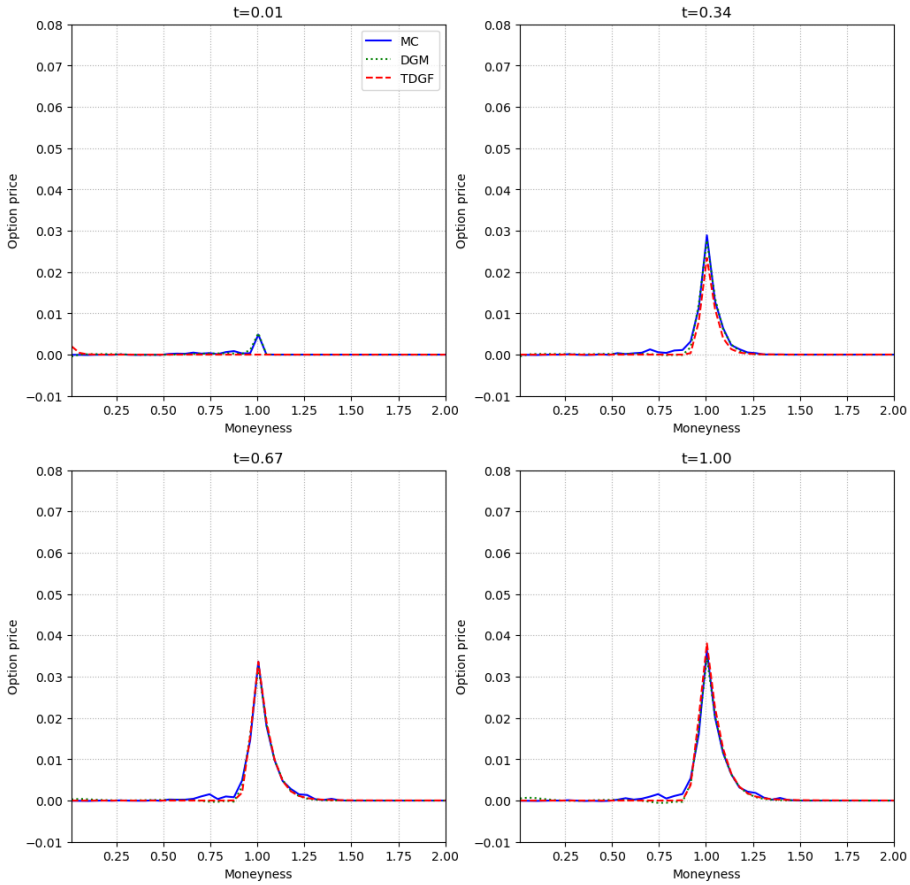


Figure 4.6: Difference between the option price and the payoff in the five-dimensional Heston model against the moneyness of the stock, compared to the DGM and Monte Carlo with Longstaff–Schwartz methods, for four different times to maturity with $r = 0.05$ and $\eta_i = 0.1$, $\rho_{ij} = 0.5$, $\hat{\rho}_{ii} = -0.5$, $\kappa_i = 0.01$, $V_0^i = 0.05$ and $\lambda_i = 2.0$ for each i and j.

5

Convergence of the generalization error for Deep Gradient Flow methods for PDEs

In the previous two chapters, we have developed a novel deep learning approach for pricing European options in diffusion models which we call the Time Deep Gradient Flow method (TDGF). We have assessed the accuracy and efficiency of the proposed method in a series of numerical examples. In Chapter 3 a simple one-dimensional European option in the complicated lifted Heston model and in Chapter 4 a multidimensional American option in the simpler Black–Scholes and Heston model. However, since TDGF is a neural network algorithm, it can be referred to as a 'black box' algorithm, since it is not clear how it relates the input to the output. Therefore, there are few guarantees that the TDGF can find the correct solution.

In this chapter, we consider the convergence of the TDGF. We prove two things. First, that there exists a neural network converging to the solution of the partial differential equation (PDE). This proof consists of three parts: 1) convergence of the time stepping; 2) equivalence of the solution of the discretized PDE and the minimizer of the variational formulation and 3) the approximation of the minimizer by a neural network by using a version of the universal approximation theorem. Second, we prove that when training the network we converge to the correct solution. This proof consists of two parts: 1) as the number of neurons goes to infinity we converge to some gradient flow and 2) as the training time goes to infinity this gradient flow converges to the solution.

This chapter is based on C. Liu, A. Papapantoleon, and J. Rou. Convergence of the generalization error for deep gradient flow methods for PDEs. To appear.

5.1 Introduction

Deep learning methods for the solution of high-dimensional partial (integro-) differential equations (P(I)DEs) have gained tremendous popularity in the last few years, since they can tackle equations in dimensions that cannot be handled by classical methods, such as finite difference and finite element schemes. This ability allows the modeling of more realistic phenomena across various scientific fields, including engineering, biology, economics and finance. The seminal articles of Sirignano and Spiliopoulos [81] on the Deep Galerkin Method (DGM) and of Raissi et al. [75] on physics-informed neural networks (PINNs), have laid the foundation for a variety of methods for solving high-dimensional PDEs.

In the present chapter, we study deep gradient flow – sometimes also called deep minimizing movement methods – for the solution of PDEs. Let us consider the PDE

$$\begin{aligned} u_t + \mathcal{A}u = 0, \quad (t,x) \in [0,T] \times \Omega, \\ u(0,x) = \Psi(x), \quad x \in \partial\Omega, \end{aligned} \tag{5.1}$$

with $\mathcal{A}$ a differential operator, Ψ a function related to the nature of the problem at hand, T is a (finite) time horizon and $\Omega \subseteq \mathbb{R}^d$ is the domain of the PDE.

The deep gradient flow methods (DGFs) for the solution of high-dimensional PDEs of the form (5.1) translate the PDE into an energy minimization problem, and can be described in the following manner:

$$u^\star_{\theta,n} = \arg\min_{v \in \mathcal{C}^n_\theta} \int \ell(v(x)) \mathrm{d}x, \tag{5.2}$$

see E and Yu [31], Georgoulis et al. [41, 42], Papapantoleon and Rou [72]. Here, $\mathcal{C}^n_\theta$ denotes the space of neural networks with n neurons with θ the set of trainable parameters, while ℓ denotes the energy functional associated to the operator $\mathcal{A}$. The minimization problem (5.2) is solved using stochastic gradient descent or one of its variants.

Let $u^\star$ denote the unique solution of (5.1). We would like to analyze and study the difference between the true solution of (5.1) and the solution computed by the DGFs, *i.e.* by the outcome of the minimization problem (5.2). This difference is the *generalization error* in the machine learning literature:

$$\mathcal{E}_{\text{gen}} = \|u^\star - u^\star_{\theta,n}\|.$$

The generalization error can be decomposed in three separate components:

- the *quadrature error* $\mathcal{E}_{\text{quad}}$, which refers to how well Monte Carlo or another quadrature method approximates the integral in (5.2);
- the *approximation error* $\mathcal{E}_{\text{approx}}$, which refers to how well the neural network v can approximate the continuous function u that solves the (5.1);
- the *training error* $\mathcal{E}_{\text{train}}$, which refers to how well GD or SGD approximate the true solution of the minimization problem (5.2).

Then, we have the error decomposition

$$\mathcal{E}_{\text{gen}} = \mathcal{E}_{\text{train}} + \mathcal{E}_{\text{quad}} + \mathcal{E}_{\text{approx}},$$

and the aim of the present chapter is to study these errors and show that, as the numbers of neurons tends to infinity and the training time also tends to infinity, then the generalization error tends to zero, and the outcome of the DGFs indeed approximates the solution of the PDE.

The quadrature error is the most well-understood error of the three, thus this chapter focuses on the other two errors. Moreover, our method also induces an discretization error, from the time-stepping scheme. However, as this error is also well-studied and understood, we have chosen to omit it from the discussion here.

The first part of this chapter focuses on the analysis of the approximation error: we show that there exists a neural network that approximates the solution of the PDE. This result uses ideas from PDE theory, optimization and the calculus of variations, and is inspired by the seminal paper of Sirignano and Spiliopoulos [81], in which an analogous result is shown for the DGM method. The second part of this chapter focuses on the analysis of the training error, and we show that as the number of neurons tend to infinity and the training time also tends to infinity, then the outcome of the DGF tends to the true solution of the PDE. This result is inspired by the work of the Jiang et al. [60], in which they show analogous results for the DGM and PINN methods. The combination of these two results, yields that the generalization error also tends to zero.

5.2 Deep Gradient Flow methods for PDEs

Let us start by providing an overview of DGFs for the solution of PDEs. These methods have gained increased popularity in the literature because they can efficiently handle high-dimensional PDEs stemming from physics, engineering and finance, see *e.g.* E and Yu [31], Liao and Ming [63], Georgoulis et al. [41] Park et al. [73] and Papapantoleon and Rou [72] for differential operators, and Georgoulis et al. [42] for an integro-differential operator. DGFs reformulate the PDE as an energy minimization problem, which is then approximated in a time-stepping fashion by deep artificial neural networks. This method results in a loss function that is tailor made to the PDE at hand, avoids the use of a second derivative which is computationally costly, and reduces the training time compared to, for instance, the DGM of Sirignano and Spiliopoulos [81]; see *e.g.* Georgoulis et al. [41, Sec. 5].

Let $u(t,x) : [0,T] \times \mathbb{R}^d \to \mathbb{R}$ be the solution of the following partial (integro-) differential equation:

$$u_t + \mathcal{A}u = 0, \quad u(0) = u_0, \tag{5.3}$$

with $\mathcal{A}$ an operator from $\mathcal{V}$ to $\mathcal{V}^*$ and $u_0 \in \mathcal{H}$ is the initial condition.

In order to write the PDE as an energy minimization problem, we need to split the operator in a symmetric and an (asymmetric) remainder part:

$$\mathcal{A}u = \mathcal{L}u + F(u), \tag{5.4}$$

with $\mathcal{L}$ a self-adjoint, linear operator and F is a (possibly nonlinear) operator from $\mathcal{V}$ to $\mathcal{V}^*$. This PDE is then discretized using, for example, the backward Euler differentiation scheme, which yields

$$\frac{U^k - U^{k-1}}{h} + \mathcal{L}U^k + F\Big(U^{k-1}\Big) = 0, \quad U^0 = u_0,$$

in which U^k denotes the approximation to the solution of the PDE $u(t_k)$ at time step t_k, on an appropriate grid. The variational formulation of this equation yields an energy functional $I^k(v)$ such that U^k is a critical point of I^k, with

$$I^k(v) = \frac{1}{2}\left\|v - U^{k-1}\right\|_{L^2}^2 + \frac{h}{2}\left\langle \mathcal{L}v, v\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} + h\left\langle F\left(U^{k-1}\right), v\right\rangle_{L^2}.$$

The function v is approximated by artificial neural networks which are trained using the stochastic gradient descent (SGD) algorithm, or one of its variants, while the functional I^k provides a loss function for the SGD iterations which is tailor made for this problem. The aim of this chapter is to show that this procedure converges to the true solution $u^\star$ of the PDE (5.3).

5

Next, we present examples of PDEs that have been treated by DGFs and their applications.

Example 5.1 (Heat equation)**.** *The simplest example that fits in this framework is the heat equation, which reads*

$$u_t = \kappa\Delta u, \quad \kappa > 0,$$

subject to an initial condition. Then $\mathcal{A} = \mathcal{L} = -\kappa\Delta$ *and* $F(u) = 0$.

Example 5.2. *Georgoulis et al. [41] consider dissipative evolution PDEs of the following form*

$$u_t - \nabla\cdot(A\nabla u) = F,$$

subject to appropriate initial and terminal conditions, with A a symmetric, uniformly positive definite and bounded diffusion tensor and F is a suitable function. Then, $\mathcal{A} = \mathcal{L} = -\nabla\cdot(A\nabla u)$.

Example 5.3 (Option pricing PDEs)**.** *The option pricing PDEs from Sections 3.4.1 to 3.4.3 fit naturally in this setting. In the Black and Scholes [23] model, for example,*

$$\mathcal{L}u = -\frac{\sigma^2}{2}\Delta u + ru \quad \text{and} \quad F(u) = \Big(\frac{\sigma^2}{2} - r\Big)\nabla u.$$

Here r and σ are positive parameters that denote the risk-free interest rate and the asset volatility respectively.

In the Heston [49], the option pricing PDE in this model takes the form (5.4) *with*

$$\mathcal{L}u = -\nabla\cdot(A\nabla u) + ru \quad \text{and} \quad F(u) = \mathbf{b}\cdot\nabla u, \tag{5.5}$$

with

$$A = \frac{V}{2}\begin{bmatrix} S^2 & \eta\rho S \\ \eta\rho S & \eta^2 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} (V - r + \frac{1}{2}\rho\eta)S \\ \kappa(V-\theta) + \frac{1}{2}\eta\rho V + \frac{\eta^2}{2} \end{bmatrix}.$$

Here, S denotes the asset price process, V the variance process, η denotes the volatility of the volatility, ρ the correlation between the Brownian motions driving the asset price and the variance process, θ the long term variance and κ the reversion rate of the variance to θ.

Example 5.4 (Option pricing PIDEs)**.** *Certain classes of partial integro-differential equation (PIDEs) arising in the pricing of financial derivatives can also be casted in this framework, in particular when the integro-differential operator is not "symmetrized". Let us consider, for example, the multidimensional Merton model as described in Georgoulis et al. [42]. Then, the PIDE arising for the pricing of basket options can be described using* (5.5)*, with the operator $\mathcal{L}$ retaining the same structure, while the function F takes now the form*

$$F(u) = \mathbf{b} \cdot \nabla u - \lambda \int_{\mathbb{R}^d} \big(u\left(x\mathrm{e}^z\right) - u(x)\big)\nu(\mathrm{d}z),$$

with ν the multivariate normal density function.

Example 5.5 (Allen–Cahn equation)**.** *Park et al. [73] consider the example of the two-dimensional Allen–Cahn equation:*

$$u_t = \Delta u - \epsilon^{-2} W'(u),$$

with appropriate initial and boundary conditions, with W a double well potential; for instance, $W(u) = \frac{(u^2-1)^2}{4}$. Then $\mathcal{L}u = -\Delta u + \epsilon^{-2}W'(u)$ and $F(u) = 0$.

5.3 Convergence of the approximation error

In this section, we show that the approximation error of the DGF converges to zero, *i.e.* we consider a neural network with a single layer and prove that as the number of nodes in the network tends to infinity, there exists a neural network that converges to the solution of the PDE. This proof consists of several steps. First, we show that the problem is well-posed in Section 5.3.1. Second, we prove convergence of the time-stepping scheme in Section 5.3.2. Third, we prove the equivalence of the discretized PDE and the minimization of the variational formulation in Section 5.3.3. Fourth, we prove a version of the universal approximation theorem in Section 5.3.4. Finally, we deduce the convergence of the neural network approximation to the solution of the minimization problem by using this theorem in Section 5.3.5.

In the sequel, we consider the following Gelfand triple: $\mathcal{V} = \mathcal{H}_0^1(\mathbb{R}^d)$, $\mathcal{V}^* = \mathcal{H}^{-1}(\mathbb{R}^d)$ and $\mathcal{H} = L^2(\mathbb{R}^d)$. Let us consider the PDE (5.3)–(5.4) and assume that the operators $\mathcal{L}$ and F satisfy the following conditions.

Assumption 5.1. *(CON) Assume that the operators $\mathcal{L}$ and F satisfy the following inequalities, for any $u, v \in \mathcal{H}_0^1(\mathbb{R}^d)$,*

$$\left|\langle \mathcal{L}u, v\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}\right| \leq M \left\|u\right\|_{\mathcal{H}_0^1} \left\|v\right\|_{\mathcal{H}_0^1} \quad \text{and} \quad \left\|F(u)\right\|_{L^2} \leq M \left\|u\right\|_{\mathcal{H}_0^1},$$

with $M > 0$ is a constant.

Assumption 5.2. *(GÅ) The operator $\mathcal{L}$ satisfies the Gårding inequality,* i.e. *there exist constants $\lambda_1 > 0, \lambda_2 \geq 0$ such that, for any $u \in \mathcal{H}_0^1(\mathbb{R}^d)$, holds*

$$\langle \mathcal{L}u, u\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} \geq \lambda_1 \|u\|^2_{\mathcal{H}_0^1} - \lambda_2 \|u\|^2_{L^2}.$$

Assumption 5.3. *(SA) The operator $\mathcal{L}$ is self-adjoint and positive definite.*

Assumption 5.4. *(LIP) The operator F satisfies an estimate of the form*

$$\|F(v) - F(w)\|_{\mathcal{H}^{-1}} \leq \lambda \|v - w\|_{\mathcal{H}^1} + \mu \|v - w\|_{L^2},$$

for all $v, w \in \left\{v \in \mathcal{H}_0^1 : \min_x \|u(x) - v\|_{\mathcal{H}^1} \leq 1\right\}$, with $\lambda < 1$ and $\mu \in \mathbb{R}$.

Remark 5.1. *The examples of PDEs considered in the previous section typically satisfy these assumptions. More details, focusing on the option pricing PDEs of Example 5.3, are deferred to Section 5.A.3.*

5.3.1 Well-posedness

Let us first discuss the existence and uniqueness of solutions for equation (5.3).

5

Theorem 5.1 (Well-posedness)**.** *Assume that the operators $\mathcal{L}$ and F satisfy Assumptions 5.1 and 5.2, then equation* (5.3) *admits a unique weak solution $u \in L^2\left((0,T);\mathcal{H}_0^1(\mathbb{R}^d)\right) \cap \mathcal{H}^1\left((0,T);\mathcal{H}^{-1}(\mathbb{R}^d)\right)$, that satisfies*

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle u, v\rangle_{L^2} + \langle \mathcal{L}u, v\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} + \langle F(u), v\rangle_{L^2} = 0$$

for any $v \in \mathcal{H}_0^1(\mathbb{R}^d)$ and $u(0) = u_0$.

Proof. According to Hilber et al. [51, Theorem 3.2.2], we only need to verify that the bilinear form $\langle \mathcal{A}u, v\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}$ is continuous and satisfies the "Gårding inequality", with

$$\langle \mathcal{A}u, v\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} = \langle \mathcal{L}u, v\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} + \langle F(u), v\rangle_{L^2}.$$

The continuity follows directly from Assumption 5.1 and the Cauchy–Schwarz inequality, since

$$\begin{aligned}\left|\langle \mathcal{A}u, v\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}\right| &\leq \left|\langle \mathcal{L}u, v\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}\right| + |\langle F(u), v\rangle_{L^2}| \\ &\leq M\left[\|u\|_{\mathcal{H}_0^1}\|v\|_{\mathcal{H}_0^1} + \|u\|_{\mathcal{H}_0^1}\|v\|_{L^2}\right] \leq 2M\|u\|_{\mathcal{H}_0^1}\|v\|_{\mathcal{H}_0^1}.\end{aligned}$$

Let us also verify that the bilinear form satisfies the Gårding inequality,

$$\begin{aligned}\left|\langle \mathcal{A}u, u\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}\right| &\geq \left|\langle \mathcal{L}u, u\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}\right| - |\langle F(u), u\rangle_{L^2}| \\ &\geq \lambda_1\|u\|^2_{\mathcal{H}_0^1} - \lambda_2\|u\|^2_{L^2} - M\|u\|_{\mathcal{H}_0^1}\|u\|_{L^2} \\ &\geq \lambda_1\|u\|^2_{\mathcal{H}_0^1} - \lambda_2\|u\|^2_{L^2} - M\left(\frac{\lambda_1}{2M}\|u\|^2_{\mathcal{H}_0^1} + \frac{M}{2\lambda_1}\|u\|^2_{L^2}\right) \\ &= \frac{\lambda_1}{2}\|u\|^2_{\mathcal{H}_0^1} - \left(\lambda_2 + \frac{M^2}{2\lambda_1}\right)\|u\|^2_{L^2}.\end{aligned}$$

In the second step we used Assumptions 5.1 and 5.2 and the Cauchy–Schwarz inequality and in the third step the Young inequality with $\varepsilon = \frac{\lambda_1}{M}$. □

5.3.2 Time stepping

The second step is to discretize the PDE in time and prove that this discretization converges to the true solution as the time step tends to zero. Consider the PDE in formulation (5.3)–(5.4):

$$u_t + \mathcal{L}u + F(u) = 0, \quad u(0) = u_0.$$

Let us divide $[0, T]$ in K intervals $(t_{k-1}, t_k]$ with step size $h = t_k - t_{k-1} = \frac{1}{K}$. Let U^k denote the approximation of $u(t_k)$ using the backward Euler discretization scheme:

$$\frac{U^k - U^{k-1}}{h} + \mathcal{L}U^k + F\left(U^{k-1}\right) = 0, \quad U^0 = u_0. \tag{5.6}$$

Theorem 5.2. *Assume that the operators $\mathcal{L}$ and F satisfy Assumptions 5.1 to 5.4. Then, there exists a constant C independent of h and k such that, for h sufficiently small, holds*

$$\max_{0 \le k \le K} \left\| u(t_k) - U^k \right\|_{L^2} \le Ch.$$

Proof. The proof follows directly from Theorem 2.1 in Akrivis, Crouzeix, and Makridakis [6]. Indeed, using that $U^0 = u\,(0)$, we can show by direct, but tedious, calculations that the assumptions of [6, p. 523] are satisfied for $\lambda < 1$ and $q = 1$. □

5.3.3 Weak formulation and uniqueness of minimizer

The third step is to reformulate equation (5.6) as a variational problem and prove that its solution is equivalent to the minimization of an energy functional. Let us first rewrite (5.6) as follows

$$\left(U^k - U^{k-1}\right) + h\left(\mathcal{L}U^k + F\left(U^{k-1}\right)\right) = 0, \quad U^0 = u_0. \tag{5.7}$$

We want to find an energy functional $I^k(u)$ such that U^k is a critical point of I^k. Consider the following functional I^k on $\mathcal{H}_0^1(\mathbb{R}^d)$

$$\begin{aligned} I^k(u) &= \frac{1}{2}\left\| u - U^{k-1} \right\|_{L^2}^2 + \frac{h}{2}\left\langle \mathcal{L}u, u \right\rangle_{\mathcal{H}^{-1}, \mathcal{H}_0^1} + h\left\langle F\left(U^{k-1}\right), u \right\rangle_{L^2} \\ &=: \mathcal{M}^k(u) + \mathcal{G}^k(u), \end{aligned} \tag{5.8}$$

in which

$$\begin{aligned} \mathcal{M}^k(u) &= \frac{1}{2}\|u\|_{L^2}^2 + \frac{h}{2}\left\langle \mathcal{L}u, u \right\rangle_{\mathcal{H}^{-1}, \mathcal{H}_0^1}, \\ \mathcal{G}^k(u) &= -\left\langle u, U^{k-1} \right\rangle_{L^2} + \frac{1}{2}\left\| U^{k-1} \right\|_{L^2}^2 + h\left\langle F\left(U^{k-1}\right), u \right\rangle_{L^2}. \end{aligned}$$

Here, $\mathcal{G}^k$ is a linear functional and $\mathcal{M}^k$ is a nonlinear (quadratic) term.

Theorem 5.3. *Assume that the operators $\mathcal{L}$ and F satisfy Assumptions 5.1 to 5.4 and that $0 < h < \frac{1}{2\lambda_2}$, with λ_2 the constant from Assumption 5.2. Then, the minimizer of* (5.8) *is the unique solution of* (5.7) *in $\mathcal{H}_0^1(\mathbb{R}^d)$.*

The proof of this theorem is based on the following two preparatory results.

Lemma 5.1. *Consider the setting of Theorem 5.3. Then, the functional I^k is bounded from below and, for any $w_* \in \mathcal{H}_0^1(\mathbb{R}^d)$ and sequence $w_m \xrightarrow{\mathcal{H}_0^1} w_*$,*

$$\liminf_{m\to\infty} I^k(w_m) \geq I^k(w_*).$$

Proof. Let us first prove that I^k is bounded from below. Using the Cauchy–Schwarz inequality and the inequality $\alpha\beta \leq \alpha^2/4 + \beta^2$,

$$\begin{aligned}
\mathcal{G}^k(u) &\geq -\left\|U^{k-1}\right\|_{L^2} \|u\|_{L^2} + \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2 - h\left\|F\left(U^{k-1}\right)\right\|_{L^2} \|u\|_{L^2} \\
&= -\|u\|_{L^2}\left(\left\|U^{k-1}\right\|_{L^2} + h\left\|F\left(U^{k-1}\right)\right\|_{L^2}\right) + \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2 \\
&\geq -\frac{1}{4}\|u\|_{L^2}^2 - \left(\left\|U^{k-1}\right\|_{L^2} + h\left\|F\left(U^{k-1}\right)\right\|_{L^2}\right)^2 + \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2.
\end{aligned}$$

Hence, the functional I^k satisfies

$$\begin{aligned}
I^k(u) =& \mathcal{M}^k(u) + \mathcal{G}^k(u) \\
\geq& \frac{1}{4}\|u\|_{L^2}^2 + \frac{h}{2}\langle \mathcal{L}u, u\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} - \left(\{\}\right)\left\|U^{k-1}\right\|_{L^2} + h\left\|F\left(U^{k-1}\right)\right\|_{L^2}\Big)^2 \\
&+ \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2.
\end{aligned}$$

Using Assumption 5.2 and the condition $h < \frac{1}{2\lambda_2}$, $I^k(u)$ is bounded below by

$$\begin{aligned}
I^k(u) &\geq \frac{1}{4}\|u\|_{L^2}^2 + \frac{h}{2}\langle \mathcal{L}u, u\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} - R_k^{(1)} \\
&\geq \left(\frac{1}{4} - \frac{h\lambda_2}{2}\right)\|u\|_{L^2}^2 + \frac{h\lambda_1}{2}\|u\|_{\mathcal{H}_0^1}^2 - R_k^{(1)},
\end{aligned} \tag{5.9}$$

with the term $R_k^{(1)}$, defined below, independent of u and finite

$$R_k^{(1)} := \left(\left\|U^{k-1}\right\|_{L^2} + h\left\|F\left(U^{k-1}\right)\right\|_{L^2}\right)^2 - \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2. \tag{5.10}$$

As for the second part, consider $w_* \in \mathcal{H}_0^1(\mathbb{R}^d)$ and a sequence $(w_m)_m$ such that $w_m \xrightarrow{\mathcal{H}_0^1} w_*$ as $m \to \infty$. Then, by the definition of weak convergence

$$\frac{1}{2}\langle w_m, w_*\rangle_{L^2} + \frac{h}{2}\langle \mathcal{L}w_*, w_m\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} \xrightarrow[m\to\infty]{} \frac{1}{2}\|w_*\|_{L^2}^2 + \frac{h}{2}\langle \mathcal{L}w_*, w_*\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1},$$

while for the linear part

$$\mathcal{G}^k(w_m) \xrightarrow[m\to\infty]{} \mathcal{G}^k(w_*).$$

Consider now the functional

$$I^k(w_m) + I^k(w_*) - I^k(w_* - w_m) = \underbrace{\langle w_m, w_* \rangle_{L^2} + h \langle \mathcal{L} w_*, w_m \rangle_{\mathcal{H}^{-1}, \mathcal{H}_0^1} + 2\mathcal{G}^k(w_m)}_{\longrightarrow 2I^k(w_*)}. \tag{5.11}$$

$$\mathcal{M}^k(u) = \frac{1}{2}\|u\|_{L^2}^2 + \frac{h}{2}\langle \mathcal{L}u, u \rangle_{\mathcal{H}^{-1}, \mathcal{H}_0^1} \geq \left(\frac{1}{2} - \frac{h\lambda_2}{2}\right)\|u\|_{L^2}^2 + \frac{h\lambda_1}{2}\|u\|_{\mathcal{H}_0^1}^2 \geq 0, \tag{5.12}$$

from Assumption 5.2 and $h < \frac{1}{2\lambda_2}$. Then, taking the limit as $m \to \infty$ on both sides of (5.11) and using (5.12),

$$\liminf_{m\to\infty} I^k(w_m) + I^k(w_*) - \underbrace{\liminf_{m\to\infty} I^k(w_* - w_m)}_{\geq 0} \geq 2I^k(w_*),$$

which implies $\liminf_{m\to\infty} I^k(w_m) \geq I^k(w_*)$. □

Proposition 5.1. *Consider the setting of Theorem 5.3. Let $U^{k-1} \in \mathcal{H}_0^1(\mathbb{R}^d)$, then there exists a unique minimizer in $\mathcal{H}_0^1(\mathbb{R}^d)$ of the functional I^k.*

Proof. Let us first show the uniqueness of the minimizer of the functional I^k. Let $w_1, w_2 \in \mathcal{H}_0^1(\mathbb{R}^d)$ be two minimizers of I^k then, using Assumptions 5.2 and 5.3,

$$\begin{aligned} & I^k(w_1) + I^k(w_2) - 2I^k\left(\frac{w_1 + w_2}{2}\right) \\ =& \frac{1}{4}\|w_1 - w_2\|_{L^2}^2 + \frac{h}{4}\langle \mathcal{L}(w_1 - w_2), w_1 - w_2 \rangle_{\mathcal{H}^{-1}, \mathcal{H}_0^1} \\ \geq& \left(\frac{1}{4} - \frac{h\lambda_2}{4}\right)\|w_1 - w_2\|_{L^2}^2 + \frac{h\lambda_1}{4}\|w_1 - w_2\|_{\mathcal{H}_0^1}^2 \overset{(5.12)}{\geq} 0, \end{aligned}$$

which is 0 if and only if $w_1 = w_2$ almost everywhere. Otherwise, $I^k\left(\frac{w_1+w_2}{2}\right)$ is smaller than $I^k(w_1)$, which is a contradiction.

Next, we show the existence of a minimizer for I^k. Define the bounded set $\mathcal{B}^k \subset \mathcal{H}_0^1(\mathbb{R}^d)$ via

$$\mathcal{B}^k := \left\{ f \in \mathcal{H}_0^1(\mathbb{R}^d) \middle| \left(\frac{1}{4} - \frac{h\lambda_2}{2}\right)\|f\|_{L^2}^2 + \frac{h\lambda_1}{2}\|f\|_{\mathcal{H}_0^1}^2 \leq R_k^{(1)} + \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2 \right\},$$

with the constant $R_k^{(1)}$ is defined in (5.10). Consider an $f \notin \mathcal{B}^k$ then, using inequality (5.9), $I^k(f) \geq \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2$. Using that $0 \in \mathcal{B}^k$, $I^k(0) = \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2$, and that I^k is bounded from below,

$$\inf_{w\in\mathcal{B}^k} I^k(w) = \inf_{w\in\mathcal{H}_0^1} I^k(w) > -\infty.$$

Let us now choose $w_m \in \mathcal{B}^k$ such that $I^k(w_m) \to \inf_{w\in\mathcal{B}^k} I^k(w)$. Let us also define w_* as the weak limit of w_m in $\mathcal{H}_0^1$. Then, by Lemma 5.1,

$$\inf_{w\in\mathcal{H}_0^1} I^k(w) = \inf_{w\in\mathcal{B}^k} I^k(w) = \liminf_{m\to\infty} I^k(w_m) \geq I^k(w_*).$$

The last inequality readily implies $I^k(w_*) = \inf_{w\in\mathcal{H}_0^1} I^k(w)$. □

Proof of Theorem 5.3. Consider the homogeneous equation

$$\frac{w}{h} + \mathcal{L}w = 0.$$

Multiplying with w on each side and integrating, implies $\frac{1}{h}\|w\|_{L^2}^2 + \langle \mathcal{L}w, w\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} = 0$. Using Assumption 5.2 and $h < \frac{1}{2\lambda_2}$, yields that $w = 0$. Therefore the homogeneous equation only has the solution $w = 0$ in $\mathcal{H}_0^1(\mathbb{R}^d)$. Thus, the solution of (5.7) is unique.

Assume that U^k minimizes I^k, and let v be a smooth function. Consider the function

$$\begin{aligned} i^k(\tau) =& I^k\left(U^k + \tau v\right) \\ =& \frac{1}{2}\left\|U^k + \tau v - U^{k-1}\right\|_{L^2}^2 + \frac{h}{2}\left\langle \mathcal{L}(U^k + \tau v), U^k + \tau v\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} \\ &+ h\left\langle F\left(U^{k-1}\right), U^k + \tau v\right\rangle_{L^2}, \end{aligned}$$

for $\tau \in \mathbb{R}$. Since U^k minimizes I^k, $\tau = 0$ should minimize i^k. Hence,

$$\begin{aligned} 0 =& \left(i^k\right)'(0) \\ =& \left\langle U^k - U^{k-1}, v\right\rangle_{L^2} + \frac{h}{2}\Big(\left\langle \mathcal{L}U^k, v\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} + \left\langle \mathcal{L}v, U^k\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}\Big) \\ &+ h\left\langle F\left(U^{k-1}\right), v\right\rangle_{L^2} \\ =& \left\langle U^k - U^{k-1}, v\right\rangle_{L^2} + h\left\langle \mathcal{L}U^k, v\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} + h\left\langle F\left(U^{k-1}\right), v\right\rangle_{L^2}. \end{aligned}$$

In the last equality we used that $\mathcal{L}$ is self-adjoint. This equality must hold for all v, thus (5.7) holds. Finally, by Assumption 5.2,

$$\left(i^k\right)''(\tau) = \|v\|_{L^2}^2 + h\langle \mathcal{L}v, v\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} \geq (1 - h\lambda_2)\|v\|_{L^2}^2 + h\lambda_1\|v\|_{\mathcal{H}_0^1}^2 > 0.$$

Therefore, $\tau = 0$ is indeed the minimizer. □

5.3.4 Neural network approximation and a version of the Universal Approximation Theorem

We use a neural network to approximate the solution of the PDE (5.6) or, more specifically, the solution of the optimization problem $\min_{u\in\mathcal{H}_0^1} I^k(u)$ in (5.8). The fourth step is to consider a more general problem: show that any function $v \in$

$\mathcal{H}_0^1(\mathbb{R}^d)$ can be approximated by a neural network. Hornik [52] proved that a different class of neural networks, see Remark 5.3, is dense in $\mathcal{H}_0^1(\Omega)$, for some bounded domain $\Omega \subseteq \mathbb{R}^d$. However, in our case, the domain equals $\mathbb{R}^d$, therefore we need a tailor-made version of the Universal Approximation Theorem.

Definition 5.1 (Activation function)**.** *An activation function is a function $\psi : \mathbb{R}^d \to \mathbb{R}$ such that $\psi \in C_c^\infty(\mathbb{R}^d)$ and $\int_{\mathbb{R}^d} \psi(x)\mathrm{d}x \neq 0$.*

Definition 5.2 (Neural network)**.** *Let ψ be an activation function, then we define*

$$\mathcal{C}^n(\psi) = \left\{ \zeta : \mathbb{R}^d \to \mathbb{R} \;\middle|\; \zeta(x) = \sum_{i=1}^n \beta_i \psi(\alpha_i x + c_i) \right\},$$

as the class of neural networks with a single hidden layer and n hidden units. The vector of weights and biases equals

$$\theta = (\beta_1, \ldots, \beta_n, \alpha_1, \ldots, \alpha_n, c_1, \ldots, c_n) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{d \times n},$$

with $\alpha_i \neq 0$ for all $i \in \{1, \ldots, n\}$, thus the dimension of the parameter space equals $(2+d)\,n$. Moreover, we set $\mathcal{C}(\psi) = \cup_{n \geq 1} \mathcal{C}^n(\psi)$.

Remark 5.2. *In the sequel, we consider PDEs that take values in $\mathbb{R}^d$, thus choosing an activation function ψ in $C_c^\infty(\mathbb{R}^d)$ is convenient. Then, we require that $\alpha_i \neq 0$, otherwise $\psi(\alpha_i x + c_i)$ is a constant, which is not integrable on $\mathbb{R}^d$.*

Remark 5.3. *Hornik [52] introduced a class of neural networks of the form $\xi(x) = \sum_{i=1}^n \beta_i \phi(a_i \cdot x + c_i)$ with $\phi : \mathbb{R} \to \mathbb{R}$ and $a_i \in \mathbb{R}^d$. The dimension of the parameter space in this case equals again $(2+d)\,n$. However, this kind of neural network does not belong to $L^2(\mathbb{R}^d)$. In fact, it is not possible to prove that this class of neural networks is dense in $\mathcal{H}_0^1(\mathbb{R}^d)$, even if ϕ has compact support. Consider, for example, the case $d = 2$, set $n = 1$, $\phi = \mathbf{1}_{[-1,1]}$, $a_1 = (1, -1)$. Then $\|\phi(\alpha_1 \cdot)\|_{L^2} = \int_{\mathbb{R}^2} \mathbf{1}_{|x-y| \leq 1} \mathrm{d}x\mathrm{d}y$, which is the area of an unbounded belt, and therefore equal to ∞.*

Theorem 5.4. *Let ψ be an activation function, then the space of neural networks $\mathcal{C}(\psi)$ is dense in $\mathcal{H}_0^1(\mathbb{R}^d)$.*

The proof of this theorem builds on the proof of the next two lemmata.

Lemma 5.2. *Let ψ be an activation function. Let g be a continuous function. Suppose that, for any $\zeta \in \mathcal{C}(\psi)$, holds $\int_{\mathbb{R}^d} \zeta(x) g(x) \mathrm{d}x = 0$. Then $g = 0$.*

Remark 5.4. *Since $\psi \in C_c^\infty(\mathbb{R}^d)$, any function ζ in $\mathcal{C}(\psi)$ has compact support. Hence $\int_{\mathbb{R}^d} \zeta(x) g(x) \mathrm{d}x$ is well-defined.*

Proof. Let $g \in C(\mathbb{R}^d)$, $x \in \mathbb{R}^d$, $0 < \varepsilon \leq 1$, and define

$$\Psi^\varepsilon(g)(x) := \int_{\mathbb{R}^d} \varepsilon^{-d} \psi\left(\frac{x-y}{\varepsilon}\right) g(y)\,\mathrm{d}y.$$

We would like to show that

$$\lim_{\varepsilon\to 0}\Psi^{\varepsilon}(g)(x) = cg(x),$$

with $c = -\int_{\mathbb{R}^d}\psi(x)\mathrm{d}x$. Using a change of variables twice,

$$\begin{aligned}\Psi^{\varepsilon}(g)(x) &= \int_{\mathbb{R}^d}\varepsilon^{-d}\psi\left(\frac{x-y}{\varepsilon}\right)g(y)\,\mathrm{d}y \overset{z=x-y}{=} -\int_{\mathbb{R}^d}\varepsilon^{-d}\psi\left(\frac{z}{\varepsilon}\right)g(x-z)\,\mathrm{d}z\\ &\overset{m=\varepsilon^{-1}z}{=} -\int_{\mathbb{R}^d}\psi(m)\,g(x-\varepsilon m)\,\mathrm{d}m = -\int_K\psi(m)\,g(x-\varepsilon m)\,\mathrm{d}m,\end{aligned}$$

with K the (compact) support of ψ. Since z is a vector, $m = \frac{z}{\varepsilon}$ yields $\mathrm{d}m = \varepsilon^{-d}\mathrm{d}z$. Then, using the dominated convergence theorem,

$$\lim_{\varepsilon\to 0}\Psi^{\varepsilon}(g)(x) = \lim_{\varepsilon\to 0}\left(-\int_K\psi(m)\,g(x-\varepsilon m)\,\mathrm{d}m\right) = cg(x).$$

5

Now, consider any $\zeta\in\mathcal{C}(\psi)$ such that $\int_{\mathbb{R}^d}\zeta(y)g(y)\mathrm{d}y = 0$. Then, for any $x\in\mathbb{R}^d$, setting $n=1$, $\beta=\varepsilon^{-d}$, $\alpha=\varepsilon^{-1}$ and $c=\frac{x}{\varepsilon}$ in the definition of $\mathcal{C}(\psi)$,

$$\int_{\mathbb{R}^d}\varepsilon^{-d}\psi\left(\frac{x-y}{\varepsilon}\right)g(y)\,\mathrm{d}y = 0.$$

We conclude the proof by sending $\varepsilon\to 0$ and using that $c\neq 0$, by definition of an activation function. □

Lemma 5.3. *Let w be a function on $C^\infty(\mathbb{R}^d)$ with support on the unit sphere, defined as*

$$w(x) = \begin{cases} c\exp\left(\dfrac{-1}{1-|x|^2}\right), & \text{if } |x|<1,\\ 0, & \text{if } |x|\geq 1,\end{cases}$$

with c a constant such that the integral of w equals 1. Let $f\in L^1_{\mathrm{loc}}(\mathbb{R}^d)$, and introduce

$$J^{\varepsilon}f(x) = w_{\varepsilon}*f(x) = \int_{\mathbb{R}^d}w_{\varepsilon}(y)\,f(x-y)\,\mathrm{d}y,$$

with $w_{\varepsilon} = \varepsilon^{-d}w\left(\frac{x}{\varepsilon}\right)$. Then, for any $\varphi\in C_c^\infty$ and $f\in L^1_{\mathrm{loc}}(\mathbb{R}^d)$,

$$\lim_{\varepsilon\to 0}\langle\varphi, J^{\varepsilon}f\rangle_{L^2} = \langle\varphi, f\rangle_{L^2}.$$

Remark 5.5. *The convolution of w_{ε} with f is convenient, because then $J^{\varepsilon}f$ is infinitely differentiable.*

Proof. Let us first rewrite $\langle \varphi, J^\varepsilon f \rangle_{L^2}$ as follows

$$\begin{aligned}
\langle \varphi, J^\varepsilon f \rangle_{L^2} &= \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \varphi(x) w_\varepsilon(y) f(x-y) \,\mathrm{d}y\mathrm{d}x \\
&= \varepsilon^{-d} \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \varphi(x) w\left(\frac{y}{\varepsilon}\right) f(x-y) \,\mathrm{d}y\mathrm{d}x \\
&\overset{z=y/\varepsilon}{=} \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \varphi(x) w(z) f(x-\varepsilon z) \,\mathrm{d}z\mathrm{d}x \\
&= \int_K \int_K \varphi(x) w(z) f(x-\varepsilon z) \,\mathrm{d}z\mathrm{d}x
\end{aligned}$$

with K a compact set that contains the support of w and φ. Hence, by the dominated convergence theorem and Lusin's theorem, letting $\varepsilon \to 0$ and using that the integral of w equals 1,

$$\lim_{\varepsilon \to 0} \langle \varphi, J^\varepsilon f \rangle_{L^2} = \lim_{\varepsilon \to 0} \int_K \int_K \varphi(x) w(z) f(x-\varepsilon z) \,\mathrm{d}z\mathrm{d}x = \langle \varphi, f \rangle_{L^2}. \qquad \square$$

Proof of Theorem 5.4. $\mathcal{C}(\psi) \subset \mathcal{H}_0^1(\mathbb{R}^d)$, since $\psi \in C_c^\infty(\mathbb{R}^d)$. Assume that $\mathcal{C}(\psi)$ is not dense in $\mathcal{H}_0^1(\mathbb{R}^d)$ then, as a corollary of the Hahn–Banach extension theorem, see *e.g.* van Neerven [85, Corollary 4.12], there exists a nonzero continuous linear functional G on $\mathcal{H}_0^1(\mathbb{R}^d)$ such that for any $\zeta \in \mathcal{C}(\psi)$,

$$G(\zeta) = 0.$$

Using the Riesz representation theorem, there exists a $g \neq 0$ in $\mathcal{H}_0^1(\mathbb{R}^d)$, such that for any $f \in \mathcal{H}_0^1(\mathbb{R}^d)$,

$$\langle f, g \rangle_{\mathcal{H}_0^1(\mathbb{R}^d)} = G(f).$$

Therefore $\langle \zeta, g \rangle_{L^2} + \langle \nabla\zeta, \nabla g \rangle_{L^2} = 0$. Let us denote $g_1^\varepsilon = J^\varepsilon g$ and $g_2^\varepsilon = J^\varepsilon \nabla g$, for convenience. Consider the inner product of these functions with ζ, then

$$\begin{aligned}
&\langle \zeta, g_1^\varepsilon \rangle_{L^2} + \langle \nabla\zeta, g_2^\varepsilon \rangle_{L^2} \\
&= \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \zeta(x) w_\varepsilon(y) g(x-y) \,\mathrm{d}y\mathrm{d}x + \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \nabla\zeta(x) \cdot [w_\varepsilon(y) \nabla g(x-y)] \,\mathrm{d}y\mathrm{d}x \\
&= \int_{\mathbb{R}^d} \left(\int_{\mathbb{R}^d} \zeta(x) g(x-y) \,\mathrm{d}x \right) w_\varepsilon(y) \,\mathrm{d}y \\
&\quad + \int_{\mathbb{R}^d} \left(\int_{\mathbb{R}^d} \nabla\zeta(x) \cdot \nabla g(x-y) \,\mathrm{d}x \right) w_\varepsilon(y) \,\mathrm{d}y \\
&\overset{x=z+y}{=} \int_{\mathbb{R}^d} \left(\int_{\mathbb{R}^d} \zeta(z+y) g(z) \,dz \right) w_\varepsilon(y) \,\mathrm{d}y \\
&\quad + \int_{\mathbb{R}^d} \left(\int_{\mathbb{R}^d} \nabla\zeta(z+y) \cdot \nabla g(z) \,dz \right) w_\varepsilon(y) \,\mathrm{d}y \\
&= \int_{\mathbb{R}^d} \left(\langle \zeta(\cdot + y), g \rangle_{L^2} + \langle \nabla\zeta(\cdot + y), \nabla g \rangle_{L^2} \right) w_\varepsilon(y) \,\mathrm{d}y = 0.
\end{aligned}$$

Since $\zeta \in \mathcal{C}(\psi)$, it has compact support and we can apply Fubini's theorem in the second step, while we can also use that $\zeta(\cdot + y) \in \mathcal{C}(\psi)$ in the last step. Hence $\langle \zeta, g_1^\varepsilon \rangle_{L^2} + \langle \nabla\zeta, g_2^\varepsilon \rangle_{L^2} = 0$. Then, using integration by parts,

$$\langle \zeta, g_1^\varepsilon - \nabla \cdot (g_2^\varepsilon) \rangle_{L^2} = 0.$$

Since $g_1^\varepsilon - \nabla \cdot (g_2^\varepsilon)$ is continuous, see *e.g.* van Neerven [85, Proposition 11.1], by Lemma 5.2, $g_1^\varepsilon - \nabla \cdot (g_2^\varepsilon) = 0$. Then, for any $f \in C_c^\infty(\mathbb{R}^d)$,

$$\langle f, g_1^\varepsilon \rangle_{L^2} + \langle \nabla f, g_2^\varepsilon \rangle_{L^2} = \langle f, g_1^\varepsilon - \nabla \cdot (g_2^\varepsilon) \rangle_{L^2} = 0.$$

Using Lemma 5.3, for any $f \in C_c^\infty(\mathbb{R}^d)$,

$$G(f) = \langle f, g \rangle_{L^2} + \langle \nabla f, \nabla g \rangle_{L^2} = \lim_{\varepsilon \to 0} \langle f, g_1^\varepsilon \rangle_{L^2} + \langle \nabla f, g_2^\varepsilon \rangle_{L^2} = 0.$$

Since $C_c^\infty(\mathbb{R}^d)$ is dense in $\mathcal{H}_0^1$ with norm $\|\cdot\|_{\mathcal{H}_0^1}$, G is a zero functional on $\mathcal{H}_0^1$, which is a contradiction. □

5

5.3.5 Convergence of the minimizer

The final step in this section, is to show that the minimizer converges to the solution of the PDE, which immediately yields that the approximation error of the method converges to zero. Theorem 5.3 yields that the solution of equation (5.7) is equivalent to the minimizer of the functional I^k in (5.8). Here, we show that this minimizer can be approximated by a neural network as defined in Definition 5.2. The convergence to the solution of PDE (5.3)–(5.4), follows by an application of Theorem 5.2.

Theorem 5.5. *Let $(w_m)_{m\in\mathbb{N}}$ be a sequence in $\mathcal{H}_0^1(\mathbb{R}^d)$ and w_* be the minimizer of I^k. Then*

$$\lim_{m\to\infty} \|w_m - w_*\|_{\mathcal{H}_0^1} = 0 \quad \text{if and only if} \quad \lim_{m\to\infty} I^k(w_m) = I^k(w_*).$$

Remark 5.6. *Therefore, we can select the approximation sequence (w_m) from the space of neural networks $\mathcal{C}(\psi)$. Using that $\mathcal{C}(\psi)$ is dense in $\mathcal{H}_0^1(\mathbb{R}^d)$, an approximation sequence always exists.*

Remark 5.7. *For an arbitrary $u \in \mathcal{H}_0^1(\mathbb{R}^d)$,*

$$\left| I^k(u_m) - I^k(u) \right| \to 0,$$

does not imply $\|u_m - u\|_{L^2} \to 0$. Consider, for example, $F = 0$, then I^k is quadratic and we can always choose $u_m = -u$ since $I^k(u) = I^k(-u)$.

Proposition 5.2 (Continuity)**.** *Assume that the operators $\mathcal{L}$ and F satisfy Assumptions 5.1 and 5.3, then the functional I^k is continuous,* i.e. *for any $f, u \in \mathcal{H}_0^1(\mathbb{R}^d)$, holds*

$$\left| I^k(f) - I^k(u) \right| \leq (1 + hM) \|f - u\|_{\mathcal{H}_0^1} \left(\|f + u\|_{\mathcal{H}_0^1} + \left\| U^{k-1} \right\|_{\mathcal{H}_0^1} \right).$$

Proof. Using the definition of the energy functional in (5.8), and that $\mathcal{L}$ is linear and self-adjoint and the Caucy–Schwarz inequality,

$$\begin{aligned}
\left|I^k(f) - I^k(u)\right| = &\left|\frac{1}{2}\left\|f - U^{k-1}\right\|_{L^2}^2 + \frac{h}{2}\left\langle \mathcal{L}f, f\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} + h\left\langle F\left(U^{k-1}\right), f\right\rangle_{L^2}\right. \\
&\left. -\frac{1}{2}\left\|u - U^{k-1}\right\|_{L^2}^2 - \frac{h}{2}\left\langle \mathcal{L}u, u\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} - h\left\langle F\left(U^{k-1}\right), u\right\rangle_{L^2}\right| \\
\leq &\frac{1}{2}\Big|\underbrace{\|f\|_{L^2}^2 - \|u\|_{L^2}^2 - 2\left\langle f-u, U^{k-1}\right\rangle_{L^2}}_{=\langle f-u, f+u-2U^{k+1}\rangle}\Big| \\
&+ \frac{h}{2}\left|\langle \mathcal{L}(f-u), f+u\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}\right| + h\left|\left\langle F\left(U^{k-1}\right), f-u\right\rangle_{L^2}\right| \\
\leq &\frac{1}{2}\|f-u\|_{L^2}\left\|f+u-2U^{k-1}\right\|_{L^2} + \frac{h}{2}\left|\langle \mathcal{L}(f-u), f+u\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}\right| \\
&+ h\left|\left\langle F\left(U^{k-1}\right), f-u\right\rangle_{L^2}\right|. \qquad (5.13)
\end{aligned}$$

Moreover, using Assumption 5.1 and the Cauchy–Schwarz inequality again,

$$\begin{aligned}
&\frac{h}{2}\left|\langle \mathcal{L}(f-u), f+u\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}\right| + h\left|\left\langle F\left(U^{k-1}\right), f-u\right\rangle_{L^2}\right| \\
\leq &\frac{hM}{2}\|f-u\|_{\mathcal{H}_0^1}\|f+u\|_{\mathcal{H}_0^1} + hM\left\|U^{k-1}\right\|_{\mathcal{H}_0^1}\|f-u\|_{L^2} \\
\leq &hM\left(\|f+u\|_{\mathcal{H}_0^1} + \left\|U^{k-1}\right\|_{\mathcal{H}_0^1}\right)\|f-u\|_{\mathcal{H}_0^1}, \qquad (5.14)
\end{aligned}$$

while from the triangle inequality,

$$\frac{1}{2}\|f-u\|_{L^2}\left\|f+u-2U^{k-1}\right\|_{L^2} \leq \|f-u\|_{\mathcal{H}_0^1}\left(\|f+u\|_{\mathcal{H}_0^1} + \left\|U^{k-1}\right\|_{\mathcal{H}_0^1}\right). \qquad (5.15)$$

Replacing (5.14) and (5.15) into (5.13) completes the proof. □

Proof of Theorem 5.5. Assume that

$$\|w_m - w_*\|_{\mathcal{H}_0^1} \to 0.$$

Then, the sequence $\|w_m - w_*\|_{\mathcal{H}_0^1}$ is bounded by some constant $C > 0$. Using Proposition 5.2,

$$\left|I^k(w_m) - I^k(w_*)\right| \leq \|w_m - w_*\|_{\mathcal{H}_0^1} C\left(1 + \left\|U^{k-1}\right\|_{\mathcal{H}_0^1}\right) \to 0.$$

Thus $I^k(w_m) \to I^k(w_*)$.

Next, we prove that $I^k(w_m) \to I^k(w_*)$ implies that $w_m \to w_*$ in $\mathcal{H}_0^1$. Let us first notice that $w_m \rightharpoonup w_*$. Otherwise, there exists a subsequence (w_{m_i}), an $\varepsilon > 0$ and a nonzero functional f such that $|f[w_{m_i}] - f[w_*]| \geq \varepsilon$. Since (w_{m_i}) is bounded in $\mathcal{H}_0^1$ (otherwise $I^k(w_{m_i})$ is unbounded), it is pre-weakly compact (which means it has a weakly convergent subsequence, see *e.g.* van Neerven [85, Corollary

5

4.56]). Let us denote one of its weak limits by w_{**}. Using Lemma 5.1, $I^k(w_*) = \lim_{i\to\infty} I^k(w_{m_i}) \geq I^k(w_{**})$. This inequality implies $w_* = w_{**}$ by the uniqueness of the minimizer. Hence $w_m \rightharpoonup w_*$, a contradiction with $|f[w_{m_i}] - f[w_*]| \geq \varepsilon$. Therefore, $w_m \rightharpoonup w_*$.

Now, since $I^k(w_m) \to I^k(w_*)$, $\mathcal{G}^n(w_m) \to \mathcal{G}^n(w_*)$, and $\mathcal{M}^k(w_m) \to \mathcal{M}^k(w_*)$,

$$\frac{1}{2}\|w_m\|_{L^2}^2 + \frac{h}{2}\langle \mathcal{L}w_m, w_m\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} \to \frac{1}{2}\|w_*\|_{L^2}^2 + \frac{h}{2}\langle \mathcal{L}w_*, w_*\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}.$$

This convergence implies

$$\begin{aligned}&\left(\frac{1}{2} - \frac{h\lambda_2}{2}\right)\|w_m - w_*\|_{L^2}^2 + \frac{h\lambda_2}{2}\|w_m - w_*\|_{\mathcal{H}_0^1}^2\\ \leq &\frac{1}{2}\|w_m - w_*\|_{L^2}^2 + \frac{h}{2}\langle \mathcal{L}(w_m - w_*), w_m - w_*\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} \to 0,\end{aligned}$$

since $w_m \rightharpoonup w_*$. Therefore, by Assumption 5.2, we conclude $\|w_m - w_*\|_{\mathcal{H}_0^1} \to 0$. □

5

5.4 Convergence of the training error

In this section, we show that, for each fixed time step k, the trained neural network converges to the true solution of the discretized PDE (5.6) as the number of neurons and the training time tend to infinity. Therefore, using the convergence of the time-stepping scheme, we can conclude the convergence of the training error.

5.4.1 Convergence of the trained neural network

In this subsection, we analyze the training of the neural network for the DGF as a function of the number of neurons n. In particular, we would like to study the process of the parameters θ^n, such that the neural network introduced in Definition 5.2, approximates the solution of the discretized PDE (5.6). We show that this process satisfies a gradient flow equation as the number of neurons tends to infinity, the "wide network limit".

Let us denote the parameters of the neural network by $\theta^n = \left(\beta^i, \alpha^i, c^i\right)_{i=1}^n \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{d\times n}$. Moreover, for $\frac{1}{2} < \delta < 1$, let us introduce a neural network

$$V^n(\theta^n; x) = \frac{1}{n^\delta}\sum_{i=1}^n \hat{\beta}^{i,n}\psi\left(\hat{\alpha}^{i,n}x + \hat{c}^{i,n}\right),$$

in accordance with Definition 5.2, with the "clipped" parameters defined as follows:

$$\begin{aligned}\hat{\alpha}^{i,n} &= \begin{cases}\left(r_n \wedge \alpha^i\right) \vee \frac{1}{r_n}, & \text{for } \alpha^i > 0,\\ \left(\frac{-1}{r_n} \wedge \alpha^i\right) \vee (-r_n), & \text{for } \alpha^i < 0,\end{cases}\\ \hat{\beta}^{i,n} &= \left(r_n \wedge \beta^i\right) \vee (-r_n),\\ \hat{c}^{i,n} &= \left(r_n \wedge c^i\right) \vee (-r_n),\end{aligned}$$

for some r_n increasing with n. We restrict the domain of the parameters (β^i, α^i, c^i) to $[-r_n, r_n]$ which converges to $\mathbb{R}$ as $n \to \infty$, and for α^i we also need to subtract the ball $\left(\frac{-1}{r_n}, \frac{1}{r_n}\right)$. Gradient clipping is in accordance with deep learning literature, see, for example, Zhang, He, Sra, and Jadbabaie [87] and Goodfellow, Bengio, and Courville [46, Ch. 10 and 11].

Next, let us introduce the gradient descent dynamics for the training process of the parameters θ^n, with t denoting the training time. The neural network $V^n(\theta^n; \cdot)$ should minimize the loss functional I^k in (5.8) of the DGF. Hence, the dynamic of θ_t^n should match the gradient of $I^k\left(V^n; \cdot\right)$:

$$\begin{aligned}\frac{\mathrm{d}\theta_t^n}{\mathrm{d}t} &= -\eta_n \nabla_\theta I^k\left(V^n\left(\theta_t^n; x\right)\right) \\ &= -\eta_n \left\langle \mathcal{D}I^k\left(V_t^n\right), \nabla_\theta V_t^n \right\rangle_{\mathcal{H}_0^1}, \end{aligned} \tag{5.16}$$

with learning rate $\eta_n = n^{2\delta-1}$ and for any $u, v \in \mathcal{H}_0^1(\mathbb{R}^d)$,

$$\left\langle \mathcal{D}I^k\left(v\right), u\right\rangle_{\mathcal{H}_0^1} = \left\langle v - U^{k-1}, u\right\rangle_{L^2} + h\left\langle \mathcal{L}v, u\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} + h\left\langle F\left(U^{k-1}\right), u\right\rangle_{L^2}. \tag{5.17}$$

We obtain a coordinate dynamic $(\theta_t^n)_{t\geq 0} = (\theta_t^{i,n})_{t\geq 0} = \left(\beta_t^{i,n}, \alpha_t^{i,n}, c_t^{i,n}\right)_{t\geq 0}$. This dynamic depends on the number of hidden layers n of the neural network, since the parameters that optimally approximate a function depend on the number of parameters we use. We use a random initialization for this process, independent of n, denoted by:

$$\left(\beta_0^{i,n}, \alpha_0^{i,n}, c_0^{i,n}\right) = \left(\beta_0^i, \alpha_0^i, c_0^i\right) = \theta_0^i.$$

Assumption 5.5. *(NNI) The parameters $\beta_0^i, \alpha_0^i, c_0^i$ that initialize the neural network are i.i.d. random variables that satisfy:*

(i) *β_0^i is a symmetric random variable with finite second moment: $\mathbb{E}\left[\left|\beta_0^i\right|^2\right] < \infty$;*

(ii) *$\alpha_0^i \neq 0$ $\mathbb{P}$-almost surely and $\mathbb{E}\left[\left|\alpha_0^i\right|^{d+7} + \left|\alpha_0^i\right|^{-d-2}\right] < \infty$;*

(iii) *c_0^i is an $\mathbb{R}^d$-valued random variable and $\mathbb{E}\left[\left|c_0^i\right|^{d+7}\right] < \infty$;*

(iv) *α_0^i, c_0^i, have full support,* i.e. *for any Borel set $A \subset \mathbb{R}$ and $B \subset \mathbb{R}^d$ with positive Lebesgue measure, $\mathbb{P}\left(\alpha_0^i \in A\right)$ and $\mathbb{P}\left(c_0^i \in B\right)$ are positive.*

Using the chain rule, the dynamic $V_t^n(x) = V^n\left(\theta_t^n; x\right)$ satisfies the following equation

$$\begin{aligned}\frac{\mathrm{d}V_t^n(x)}{\mathrm{d}t} &= \nabla_\theta V^n\left(\theta_t^n; x\right) \cdot \frac{\mathrm{d}\theta_t^n}{\mathrm{d}t} = -\eta_n \nabla_\theta V^n\left(\theta_t^n; x\right) \cdot \nabla_\theta I^k\left(V^n\left(\theta_t^n; x\right)\right) \\ &= -\left\langle \mathcal{D}I^k\left(V_t^n\right), Z_t^n(x, \cdot)\right\rangle_{\mathcal{H}_0^1}, \end{aligned} \tag{5.18}$$

with $V_0^n(x) = V^n(\theta_0^n; x)$ and

$$\begin{aligned} Z_t^n(x,y) &= \eta_n \nabla_\theta V_t^n(x) \cdot \nabla_\theta V_t^n(y) \\ &= \frac{1}{n}\sum_{i=1}^n \nabla_\theta \hat{\beta}_t^{i,n} \psi\left(\hat{\alpha}_t^{i,n} x + \hat{c}_t^{i,n}\right) \cdot \nabla_\theta \hat{\beta}_t^{i,n} \psi\left(\hat{\alpha}_t^{i,n} y + \hat{c}_t^{i,n}\right). \end{aligned}$$

We expect (5.18) to converge to the following gradient flow

$$\frac{\mathrm{d}V_t(x)}{\mathrm{d}t} = -\left\langle \mathcal{D}I^k(V_t), Z(x,\cdot)\right\rangle_{\mathcal{H}_0^1}, \tag{5.19}$$

with $V_0 = 0$ and

$$Z(x,y) = \mathbb{E}\big[\nabla_\theta \beta_0^1 \psi\left(\alpha_0^1 x + c_0^1\right) \cdot \nabla_\theta \beta_0^1 \psi\left(\alpha_0^1 y + c_0^1\right)\big],$$

while the inner product of the Fréchet derivative of the loss functional with another functional is defined in (5.17). The gradient flow (5.19) is an infinite-dimensional ODE that governs the dynamics of the wide network limit of the neural network during the training process and, obviously, the right-hand side depends on the loss function of the gradient flow method for the solution of PDEs in (5.8). The kernel $Z(\cdot,\cdot)$ is not the standard neural tangent kernel, as it also depends on the loss functional, which further complicates the analysis. The right-hand side of (5.19), using (5.17), takes the following form:

$$\begin{aligned} \mathcal{T}(v)(x) &:= \left\langle \mathcal{D}I^k(v), Z(x,\cdot)\right\rangle_{\mathcal{H}_0^1} \\ &= \left\langle v - U^{k-1}, Z(x,\cdot)\right\rangle_{L^2} + h\left\langle \mathcal{L}v, Z(x,\cdot)\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} \\ &\quad + h\left\langle F\left(U^{k-1}\right), Z(x,\cdot)\right\rangle_{L^2}. \end{aligned}$$

The analysis of this operator plays a crucial role in the next subsection, in which we study the long term behavior of this gradient flow.

Let us introduce the following shorthand notation:

$$\mathcal{X}(\theta; x) := \nabla_\theta \beta \psi(\alpha x + c) \quad \text{and} \quad \mathcal{X}^n(\theta; x) := \nabla_\theta \hat{\beta} \psi(\hat{\alpha} x + \hat{c})$$

for some generic parameters $\theta = (\beta, \alpha, c) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^d$, with $(\hat{\beta}, \hat{\alpha}, \hat{c})$ denoting the clipped version of these parameters. Moreover, in order to simplify the notation, let us set

$$X(x) := \mathcal{X}\left(\theta_0^1; x\right), \quad X^n(x) := \mathcal{X}^n\left(\theta_0^1; x\right) \quad \text{and} \quad X_t^{i,n}(x) := \mathcal{X}^n\left(\theta_t^{i,n}; x\right).$$

Then, using this notation,

$$\begin{aligned} Z_t^n(x,y) &= \frac{1}{n}\sum_{i=1}^n X_t^{i,n}(x) \cdot X_t^{i,n}(y), \\ Z(x,y) &= \mathbb{E}\left[X(x) \cdot X(y)\right]. \end{aligned}$$

This representation invites us to use the law of large numbers to conclude that $Z_t^n(x,y) \to Z(x,y)$ as $n \to \infty$. So we have an intuitive connection between the gradient flow in (5.18) that the neural network follows during the training process, and the corresponding "wide network limit" in (5.19).

The main result of this subsection follows: as the number of neurons tends to infinity during the training process, then the neural network V^n converges to the wide network limit V, which satisfies the gradient flow (5.19).

Theorem 5.6. *Assume that the neural network satisfies Assumption 5.5, and let r_n increase with n, while $r_n \leq \log n$. Moreover, assume that the operators of the PDE* (5.3)–(5.4) *satisfy Assumptions 5.1 and 5.2. Then, the dynamic* (5.18) *converges to the gradient flow* (5.19) *as $n \to \infty$,* i.e. *for any $T > 0$,*

$$\sup_{0 \leq t \leq T} \mathbb{E}\left[\left\|V_t^n - V_t\right\|_{\mathcal{H}_0^1}\right] \xrightarrow[n\to\infty]{} 0.$$

Lemma 5.4. *Assume that the neural network satisfies Assumption 5.5, then*

$$\mathbb{E}\left[\|V_0^n\|_{\mathcal{H}_0^1}\right] \leq C^{(1)} n^{\frac{1}{2}-\delta},$$

with $C^{(1)} = \mathbb{E}\left[\left|\beta_0^i\right|^2\right]^{\frac{1}{2}} \left(\mathbb{E}\left[\left|\alpha_0^i\right|^{-d}\right] + \mathbb{E}\left[\left|\alpha_0^i\right|^{2-d}\right] + 2\right)^{\frac{1}{2}} \|\psi\|_{\mathcal{H}_0^1}$.

Proof. Let us denote a neuron by $Y^i(x) := \hat{\beta}_0^i \psi\left(\hat{\alpha}_0^i x + \hat{c}_0^i\right)$, then

$$V_0^n = \frac{1}{n^\delta} \sum_{i=1}^n Y^i(x)$$

and

$$\mathbb{E}\left[\|V_0^n\|_{\mathcal{H}_0^1}^2\right] = \mathbb{E}\left[\int_{\mathbb{R}^d} |V_0^n|^2 + |\nabla_x V_0^n|^2 \,\mathrm{d}x\right]. \tag{5.20}$$

Let us first compute the value of the cross terms, for $i \neq j$, which equals

$$\begin{aligned}\mathbb{E}\left[\int_{\mathbb{R}^d} Y^i(x) Y^j(x) \mathrm{d}x\right] &= \mathbb{E}\left[\int_{\mathbb{R}^d} \hat{\beta}_0^i \psi\left(\hat{\alpha}_0^i x + \hat{c}_0^i\right) \hat{\beta}_0^j \psi\left(\hat{\alpha}_0^j x + \hat{c}_0^j\right) \mathrm{d}x\right] \\ &= \int_{\mathbb{R}^d} \mathbb{E}\left[\hat{\beta}_0^i\right] \mathbb{E}\left[\hat{\beta}_0^j\right] \mathbb{E}\left[\psi\left(\hat{\alpha}_0^i x + \hat{c}_0^i\right) \psi\left(\hat{\alpha}_0^j x + \hat{c}_0^j\right)\right] \mathrm{d}x = 0.\end{aligned}$$

Here, we can apply Fubini's theorem since ψ has compact support and the parameters θ are bounded, while the random variables β_0^i are symmetric, hence their expectation is zero. Therefore, we can bound the L^2-norm of V_0^n by

$$\begin{aligned}\mathbb{E}\left[\int_{\mathbb{R}^d} |V_0^n|^2 \,\mathrm{d}x\right] &= \frac{1}{n^{2\delta}} \mathbb{E}\left[\int_{\mathbb{R}^d} \sum_{i=1}^n \left|\hat{\beta}_0^i\right|^2 \left|\psi\left(\hat{\alpha}_0^i x + \hat{c}_0^i\right)\right|^2 \mathrm{d}x\right] \\ &= \frac{1}{n^{2\delta-1}} \mathbb{E}\left[\left|\hat{\beta}_0^i\right|^2 \int_{\mathbb{R}^d} \left|\hat{\alpha}_0^i\right|^{-d} |\psi(y)|^2 \,\mathrm{d}y\right] \\ &\leq \frac{1}{n^{2\delta-1}} \mathbb{E}\left[\left|\hat{\beta}_0^i\right|^2\right] \mathbb{E}\left[\left|\hat{\alpha}_0^i\right|^{-d}\right] \int_{\mathbb{R}^d} |\psi(y)|^2 \,\mathrm{d}y. \end{aligned} \tag{5.21}$$

The derivative of V_0^n can be expressed as

$$\nabla_x V_0^n = \frac{1}{n^\delta} \sum_{i=1}^n \nabla_x Y^i(x) = \frac{1}{n^\delta} \sum_{i=1}^n \hat{\beta}_0^i \hat{\alpha}_0^i \left(\nabla\psi\right)\left(\hat{\alpha}_0^i x + \hat{c}_0^i\right).$$

Hence, we can analogously bound the L^2-norm of the derivative of V_0^n by

$$\begin{aligned}\mathbb{E}\left[\int_{\mathbb{R}^d} \left|\nabla_x V_0^n\right|^2 \mathrm{d}x\right] &= \frac{1}{n^{2\delta-1}} \mathbb{E}\left[\int_{\mathbb{R}^d} \left|\hat{\beta}_0^i\right|^2 \left|\hat{\alpha}_0^i\right|^2 \left|\left(\nabla\psi\right)\left(\hat{\alpha}_0^i x + \hat{c}_0^i\right)\right|^2 \mathrm{d}x\right] \\ &\leq \frac{1}{n^{2\delta-1}} \mathbb{E}\left[\left|\hat{\beta}_0^i\right|^2\right] \mathbb{E}\left[\left|\hat{\alpha}_0^i\right|^{2-d}\right] \int_{\mathbb{R}^d} \left|\nabla\psi(x)\right|^2 \mathrm{d}x. \end{aligned} \tag{5.22}$$

Applying bounds (5.21) and (5.22) to (5.20), and using Jensen's inequality and that $\left|\hat{\beta}_0^i\right| \leq \left|\beta_0^i\right|$, yields

$$\begin{aligned}&\mathbb{E}\left[\left\|V_0^n\right\|_{\mathcal{H}_0^1}\right] \leq \sqrt{\mathbb{E}\left[\left\|V_0^n\right\|_{\mathcal{H}_0^1}^2\right]} \\ \leq& n^{\frac{1}{2}-\delta} \sqrt{\mathbb{E}\left[\left|\beta_0^i\right|^2\right] \mathbb{E}\left[\left|\hat{\alpha}_0^i\right|^{-d}\right] \int_{\mathbb{R}^d} \left|\psi\right|^2 \mathrm{d}x + \mathbb{E}\left[\left|\beta_0^i\right|^2\right] \mathbb{E}\left[\left|\hat{\alpha}_0^i\right|^{2-d}\right] \int_{\mathbb{R}^d} \left|\nabla\psi\right|^2 \mathrm{d}x} \\ \leq& n^{\frac{1}{2}-\delta} \mathbb{E}\left[\left|\beta_0^i\right|^2\right]^{\frac{1}{2}} \left(\mathbb{E}\left[\left|\hat{\alpha}_0^i\right|^{-d}\right] + \mathbb{E}\left[\left|\hat{\alpha}_0^i\right|^{2-d}\right]\right)^{\frac{1}{2}} \sqrt{\int_{\mathbb{R}^d} \left|\psi\right|^2 \mathrm{d}x + \int_{\mathbb{R}^d} \left|\nabla\psi\right|^2 \mathrm{d}x}.\end{aligned}$$

If $|\alpha| \leq r_n$, then $|\hat{\alpha}|^{-1} \leq |\alpha|^{-1}$ and if $|\alpha| > r_n$, then $|\hat{\alpha}|^{-1} = r_n^{-1}$. Therefore, for $d = 1, 2$,

$$\mathbb{E}\left[\left|\hat{\alpha}_0^i\right|^{-d}\right] + \mathbb{E}\left[\left|\hat{\alpha}_0^i\right|^{2-d}\right] \leq \mathbb{E}\left[\left|\alpha_0^1\right|^{-d} + (r_n)^{-d} + \left|\alpha_0^1\right|^{2-d}\right],$$

while for $d \geq 3$,

$$\mathbb{E}\left[\left|\hat{\alpha}_0^i\right|^{-d}\right] + \mathbb{E}\left[\left|\hat{\alpha}_0^i\right|^{2-d}\right] \leq \mathbb{E}\left[\left|\alpha_0^1\right|^{-d} + (r_n)^{-d} + \left|\alpha_0^1\right|^{2-d} + (r_n)^{2-d}\right]. \qquad \square$$

Proof of Theorem 5.6. Using (5.18) and (5.19), we need to estimate the following difference:

$$\begin{aligned}V_t^n(x) - V_t(x) =& V_0^n(x) - V_0(x) \\ &+ \int_0^t \left\langle \mathcal{D}I^k\left(V_s\right), Z(x,\cdot)\right\rangle_{\mathcal{H}_0^1} - \left\langle \mathcal{D}I^k\left(V_s^n\right), Z_s^n\left(x,\cdot\right)\right\rangle_{\mathcal{H}_0^1} \mathrm{d}s \\ =& \underbrace{V_0^n(x) - V_0(x)}_{(a1)} + \int_0^t \underbrace{\left\langle \mathcal{D}I^k\left(V_s\right) - \mathcal{D}I^k\left(V_s^n\right), Z(x,\cdot)\right\rangle_{\mathcal{H}_0^1}}_{(a2)} \\ &+ \underbrace{\left\langle \mathcal{D}I^k\left(V_s^n\right), Z(x,\cdot) - Z_0^n\left(x,\cdot\right)\right\rangle_{\mathcal{H}_0^1}}_{(a3)} \\ &+ \underbrace{\left\langle \mathcal{D}I^k\left(V_s^n\right), Z_0^n\left(x,\cdot\right) - Z_s^n\left(x,\cdot\right)\right\rangle_{\mathcal{H}_0^1}}_{(a4)} \mathrm{d}s.\end{aligned}$$

The proof is now separated in several steps, corresponding to the estimation of each of the terms above.
Step (a1). $V_0(x) = 0$ by definition, hence, using Lemma 5.4,

$$\mathbb{E}\left[\|V_0^n(x) - V_0(x)\|_{\mathcal{H}_0^1}\right] \leq C^{(1)} n^{\frac{1}{2}-\delta}.$$

Step (a2). In order to separate the random terms V and X, we define another probability space Ω' and probability measure $\mathbb{P}'$ such that $X(\omega')$ under $\mathbb{P}'$ has the same distribution as X under $\mathbb{P}$. Then, we can rewrite $(a2)$ as follows

$$\begin{aligned}(a2) &= \left\langle \mathcal{D}I^k(V_s) - \mathcal{D}I^k(V_s^n), \mathbb{E}[X \cdot X(x)]\right\rangle_{\mathcal{H}_0^1} \\ &= \int_{\Omega'} \left\langle \mathcal{D}I^k(V_s) - \mathcal{D}I^k(V_s^n), X(\omega')\right\rangle_{\mathcal{H}_0^1} X(x)(\omega')\,\mathbb{P}'(\mathrm{d}\omega').\end{aligned}$$

Hence, we can bound the $\mathcal{H}_0^1$-norm of (a2) by

$$\begin{aligned}\|(a2)\|_{\mathcal{H}_0^1} &\leq \int_{\Omega'} \left\| \left\langle \mathcal{D}I^k(V_s) - \mathcal{D}I^k(V_s^n), X(\omega')\right\rangle_{\mathcal{H}_0^1} \cdot X(x)(\omega')\right\|_{\mathcal{H}_0^1} \mathbb{P}(\mathrm{d}\omega') \\ &= \int_{\Omega'} \left| \left\langle \mathcal{D}I^k(V_s) - \mathcal{D}I^k(V_s^n), X(\omega')\right\rangle_{\mathcal{H}_0^1}\right| \|X(\omega')\|_{\mathcal{H}_0^1} \mathbb{P}(\mathrm{d}\omega') \\ &\overset{\text{Lemma 5.5}}{\leq} K\|V_s - V_s^n\|_{\mathcal{H}_0^1} \int_{\Omega'} \|X(\omega')\|_{\mathcal{H}_0^1}^2 \mathbb{P}(\mathrm{d}\omega') \\ &\overset{\text{Lemma 5.8}}{=} K\|V_s - V_s^n\|_{\mathcal{H}_0^1} \mathbb{E}\left[\|X\|_{\mathcal{H}_0^1}^2\right] \\ &\leq C_\psi \|V_s - V_s^n\|_{\mathcal{H}_0^1}.\end{aligned}$$

Here, K is the constant from Lemma 5.5 and C_ψ is another constant that depends on the activation function ψ and may change from line to line.
Step (a3). Let us rewrite the term $(a3)$ as follows:

$$\begin{aligned}(a3) = &\underbrace{\frac{1}{n}\sum_{i=1}^n \left\langle \mathcal{D}I^k(V_s^n), X_0^{i,n} \cdot X_0^{i,n}(x) - \mathbb{E}\left[X_0^{i,n} \cdot X_0^{i,n}(x)\right]\right\rangle_{\mathcal{H}_0^1}}_{(a3.1)} \\ &+ \underbrace{\left\langle \mathcal{D}I^k(V_s^n), \mathbb{E}[X^n \cdot X^n(x) - X \cdot X(x)]\right\rangle_{\mathcal{H}_0^1}}_{(a3.2)}.\end{aligned}$$

We can use X^n instead of $X_0^{i,n}$ in the second term since, by Assumption 5.5, these two terms have the same expectation. Then, we can further separate the term $(a3.2)$ as follows:

$$\begin{aligned}(a3.2) = &\underbrace{\left\langle \mathcal{D}I^k(V_s^n), \mathbb{E}[(X^n - X) \cdot X^n(x)]\right\rangle_{\mathcal{H}_0^1}}_{(a3.21)} \\ &+ \underbrace{\left\langle \mathcal{D}I^k(V_s^n), \mathbb{E}[X \cdot (X^n(x) - X(x))]\right\rangle_{\mathcal{H}_0^1}}_{(a3.22)}.\end{aligned}$$

A similar computation as for the term $(a2)$, yields

$$\begin{aligned}\|(a3.21)\|_{\mathcal{H}_0^1} &\leq \int_{\Omega'} \left|\left\langle \mathcal{D}I^k\left(V_s^n\right), X^n\left(\omega'\right) - X\left(\omega'\right)\right\rangle_{\mathcal{H}_0^1}\right| \|X^n\left(\omega'\right)\|_{\mathcal{H}_0^1} \mathbb{P}\left(\mathrm{d}\omega'\right) \\ &\overset{\text{Lemma 5.5}}{\leq} K\left(1 + \|V_s^n\|_{\mathcal{H}_0^1}\right) \int_{\Omega'} \|X^n\left(\omega'\right) - X\left(\omega'\right)\|_{\mathcal{H}_0^1} \|X^n\left(\omega'\right)\|_{\mathcal{H}_0^1} \mathbb{P}\left(\mathrm{d}\omega'\right) \\ &= K\left(1 + \|V_s^n\|_{\mathcal{H}_0^1}\right) \mathbb{E}\left[\|X^n - X\|_{\mathcal{H}_0^1} \|X^n\|_{\mathcal{H}_0^1}\right].\end{aligned}$$

Analogously,

$$\|(a3.22)\|_{\mathcal{H}_0^1} \leq K\left(1 + \|V_s^n\|_{\mathcal{H}_0^1}\right) \mathbb{E}\left[\|X^n - X\|_{\mathcal{H}_0^1} \|X\|_{\mathcal{H}_0^1}\right].$$

5

Overall, combining the two bounds and then using Lemma 5.6, then the Cauchy–Schwarz inequality, and finally Lemmas 5.8 and 5.9,

$$\begin{aligned}\mathbb{E}\left[\|(a3.2)\|_{\mathcal{H}_0^1}\right] &\leq 2K\mathbb{E}\left[1 + \|V_s^n\|_{\mathcal{H}_0^1}\right] \mathbb{E}\left[\|X^n - X\|_{\mathcal{H}_0^1} \left(\|X^n\|_{\mathcal{H}_0^1} + \|X\|_{\mathcal{H}_0^1}\right)\right] \\ &\leq C_\psi \mathbb{E}\left[\|X^n - X\|_{\mathcal{H}_0^1} \left(\|X^n\|_{\mathcal{H}_0^1} + \|X\|_{\mathcal{H}_0^1}\right)\right] \\ &\leq C_\psi \mathbb{E}\left[\|X^n - X\|_{\mathcal{H}_0^1}^2\right]^{\frac{1}{2}} \mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^2 + \|X\|_{\mathcal{H}_0^1}^2\right]^{\frac{1}{2}} \leq C_\psi\, \varepsilon_n^{\frac{1}{2}},\end{aligned}$$

with ε_n defined in Lemma 5.9 and equal to

$$\varepsilon_n = \mathbb{E}\left[\|X^n - X\|_{\mathcal{H}_0^1}^2\right].$$

On the other hand, using Lemma 5.5 the norm of $(a3.1)$ can be bounded by

$$\begin{aligned}&\|(a3.1)\|_{L^2} \\ =&\frac{1}{n}\left(\int_{\mathbb{R}^d} \left\langle \mathcal{D}I^k\left(V_s^n\right), \sum_{i=1}^n \left(X_0^{i,n} \cdot X_0^{i,n}\left(x\right) - \mathbb{E}\left[X_0^{i,n} \cdot X_0^{i,n}\left(x\right)\right]\right)\right\rangle_{\mathcal{H}_0^1}^2 \mathrm{d}x\right)^{\frac{1}{2}} \\ \leq&\frac{K}{n}\left(1 + \|V_s^n\|_{\mathcal{H}_0^1}\right)\left(\int_{\mathbb{R}^d} \left\|\sum_{i=1}^n \left(X_0^{i,n} \cdot X_0^{i,n}\left(x\right) - \mathbb{E}\left[X_0^{i,n} \cdot X_0^{i,n}\left(x\right)\right]\right)\right\|_{\mathcal{H}_0^1}^2 \mathrm{d}x\right)^{\frac{1}{2}}.\end{aligned}$$

Then, using the Cauchy–Schwarz inequality and Lemma 5.6,

$$\begin{aligned}\mathbb{E}\left[\|(a3.1)\|_{L^2}\right] \leq & \frac{K}{n}\mathbb{E}\left[\left(1+\|V_s^n\|_{\mathcal{H}_0^1}\right)^2\right]^{\frac{1}{2}} \\ & \times \mathbb{E}\left[\int_{\mathbb{R}^d}\left\|\sum_{i=1}^n\left(X_0^{i,n}\cdot X_0^{i,n}(x)-\mathbb{E}\left[X_0^{i,n}\cdot X_0^{i,n}(x)\right]\right)\right\|_{\mathcal{H}_0^1}^2 \mathrm{d}x\right]^{\frac{1}{2}} \\ \leq & \frac{C_\psi}{n}\mathbb{E}\left[\int_{\mathbb{R}^d}\left\|\sum_{i=1}^n\left(X_0^{i,n}\cdot X_0^{i,n}(x)-\mathbb{E}\left[X_0^{i,n}\cdot X_0^{i,n}(x)\right]\right)\right\|_{\mathcal{H}_0^1}^2 \mathrm{d}x\right]^{\frac{1}{2}} \\ = & \frac{C_\psi}{\sqrt{n}}\mathbb{E}\left[\int_{\mathbb{R}^d}\|X^n\cdot X^n(x)-\mathbb{E}\left[X^n\cdot X^n(x)\right]\|_{\mathcal{H}_0^1}^2 \mathrm{d}x\right]^{\frac{1}{2}}.\end{aligned}$$

The last equality follows because X^n and $X_0^{i,n}$ are equally distributed by Assumption 5.5, while $X^{i,n}\cdot X^{i,n}(x)$ are i.i.d. variables that satisfy

$$\mathbb{E}\left[\left\langle X^{i,n}\cdot X^{i,n}(x), X^{j,n}\cdot X^{j,n}(x)\right\rangle_{\mathcal{H}_0^1}\right]=0, \quad \text{for } i\neq j.$$

Using the triangle inequality,

$$\begin{aligned}&\mathbb{E}\left[\int_{\mathbb{R}^d}\|X^n\cdot X^n(x)-\mathbb{E}\left[X^n\cdot X^n(x)\right]\|_{\mathcal{H}_0^1}^2 \mathrm{d}x\right] \\ &\leq 2\mathbb{E}\left[\int_{\mathbb{R}^d}\|X^n\cdot X^n(x)\|_{\mathcal{H}_0^1}^2 \mathrm{d}x\right]+2\int_{\mathbb{R}^d}\|\mathbb{E}\left[X^n\cdot X^n(x)\right]\|_{\mathcal{H}_0^1}^2 \mathrm{d}x \\ &\leq 2\mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^2\|X^n\|_{L^2}^2\right]+2\mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^2\right]\mathbb{E}\left[\|X^n\|_{L^2}^2\right].\end{aligned}$$

Then, combining the last two inequalities,

$$\mathbb{E}\left[\|(a3.1)\|_{L^2}\right] \leq \frac{C_\psi}{\sqrt{n}}\left(\mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^2\|X^n\|_{L^2}^2\right]^{\frac{1}{2}}+\mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^2\right]^{\frac{1}{2}}\mathbb{E}\left[\|X^n\|_{L^2}^2\right]^{\frac{1}{2}}\right).$$

We can analogously estimate the term $\mathbb{E}\left[\|\nabla_x(a3.1)\|_{L^2}\right]$:

$$\begin{aligned}\mathbb{E}\left[\|(a3.1)\|_{\mathcal{H}_0^1}\right] &\leq \frac{C_\psi}{\sqrt{n}}\left(\mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^4\right]^{\frac{1}{2}}+\mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^2\right]\right) \leq \frac{C_\psi}{\sqrt{n}}\mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^4\right]^{\frac{1}{2}} \\ &\overset{\text{Lemma 5.7}}{\leq} \frac{C_\psi\left(r_n\right)^{8+d}}{\sqrt{n}}.\end{aligned}$$

Overall, we finish this step by concluding

$$\mathbb{E}\left[\|(a3)\|_{\mathcal{H}_0^1}\right] \leq C_\psi\left(\frac{\left(r_n\right)^{8+d}}{\sqrt{n}}+\varepsilon_n^{\frac{1}{2}}\right).$$

Step (a4). By definition of Z_t^n,

$$\begin{aligned}(a4) =& \frac{1}{n}\sum_{i=1}^{n}\left\langle \mathcal{D}I^k\left(V_s^n\right), X_0^{i,n}\right\rangle_{\mathcal{H}_0^1}\cdot X_0^{i,n}(x) - \frac{1}{n}\sum_{i=1}^{n}\left\langle \mathcal{D}I^k\left(V_s^n\right), X_s^{i,n}\right\rangle_{\mathcal{H}_0^1}\cdot X_s^{i,n}(x)\\ =& \frac{1}{n}\sum_{i=1}^{n}\left\langle \mathcal{D}I^k\left(V_s^n\right), X_0^{i,n} - X_s^{i,n}\right\rangle_{\mathcal{H}_0^1}\cdot X_0^{i,n}(x)\\ &- \frac{1}{n}\sum_{i=1}^{n}\left\langle \mathcal{D}I^k\left(V_s^n\right), X_s^{i,n}\right\rangle_{\mathcal{H}_0^1}\cdot\left(X_s^{i,n}(x) - X_0^{i,n}(x)\right).\end{aligned}$$

Using first Lemma 5.5 and then Lemmas 5.7 and 5.10,

$$\begin{aligned}\|(a4)\|_{\mathcal{H}_0^1} &\le \frac{K}{n}\left(1+\|V_s^n\|_{\mathcal{H}_0^1}\right)\sum_{i=1}^{n}\left(\left\|X_s^{i,n}\right\|_{\mathcal{H}_0^1}+\left\|X^{i,n}\right\|_{\mathcal{H}_0^1}\right)\left\|X_s^{i,n}-X^{i,n}\right\|_{\mathcal{H}_0^1}\\ &\le \frac{C_\psi}{n}\left(r_n\right)^{8+d}\left(1+\|V_s^n\|_{\mathcal{H}_0^1}\right)\sum_{i=1}^{n}\left|\theta_s^{i,n}-\theta_0^i\right|^{\frac{1}{2}}.\end{aligned}$$

Hence we can bound its expectation by

$$\begin{aligned}\mathbb{E}\left[\|(a4)\|_{\mathcal{H}_0^1}\right] &\le \frac{C_\psi}{n}\left(r_n\right)^{8+d}\mathbb{E}\left[\left(\|V_s^n\|_{\mathcal{H}_0^1}+1\right)^2\right]^{\frac{1}{2}}\mathbb{E}\left[\left(\sum_{i=1}^{n}\left|\theta_s^{i,n}-\theta_0^i\right|^{\frac{1}{2}}\right)^2\right]^{\frac{1}{2}}\\ &\le C_\psi n^{-\frac{1}{2}}\left(r_n\right)^{8+d}\mathbb{E}\left[\left(\|V_s^n\|_{\mathcal{H}_0^1}+1\right)^2\right]^{\frac{1}{2}}\mathbb{E}\left[\left|\theta_s^{1,n}-\theta_0^1\right|\right]^{\frac{1}{2}}\\ &\overset{\text{Lemmas 5.6 and 5.11}}{\le} C_\psi\sqrt{t}n^{\frac{\delta}{2}-1}\left(r_n\right)^{10+2d}.\end{aligned}$$

Final step. Combining the previous steps,

$$\begin{aligned}\mathbb{E}\left[\|V_t^n - V_t\|_{\mathcal{H}_0^1}\right] \le& \mathbb{E}\left[\|(a1)\|_{\mathcal{H}_0^1}\right] + \mathbb{E}\left[\left\|\int_0^t (a2)+(a3)+(a4)\,\mathrm{d}s\right\|_{\mathcal{H}_0^1}\right]\\ \le& \mathbb{E}\left[\|(a1)\|_{\mathcal{H}_0^1}\right]\\ &+\int_0^t \mathbb{E}\left[\|(a2)\|_{\mathcal{H}_0^1}\right]+\mathbb{E}\left[\|(a3)\|_{\mathcal{H}_0^1}\right]+\mathbb{E}\left[\|(a4)\|_{\mathcal{H}_0^1}\right]\mathrm{d}s\\ \le& M\int_0^t \mathbb{E}\left[\|V_s^n - V_s\|_{\mathcal{H}_0^1}\right]\mathrm{d}s\\ &+C^{(1)}n^{\frac{1}{2}-\delta}+T^{\frac{3}{2}}C_\psi\left(\frac{\left(r_n\right)^{8+d}}{\sqrt{n}}+\varepsilon_n^{\frac{1}{2}}+n^{\frac{\delta}{2}-1}\left(r_n\right)^{10+2d}\right).\end{aligned}$$

Hence, using Grönwall's inequality, we conclude

$$\begin{aligned}\mathbb{E}\left[\|V_t^n - V_t\|_{\mathcal{H}_0^1}\right] &\le \mathrm{e}^{MT}\left(C^{(1)}n^{\frac{1}{2}-\delta}+T^{\frac{3}{2}}C_\psi\left(\frac{\left(r_n\right)^{8+d}}{\sqrt{n}}+\varepsilon_n^{\frac{1}{2}}+n^{\frac{\delta}{2}-1}\left(r_n\right)^{10+2d}\right)\right)\\ &\xrightarrow[n\to\infty]{} 0.\end{aligned}$$

□

5.4.2 Long time behavior of the gradient flow

In this subsection, we prove that the wide network limit of the trained neural network, *i.e.* the process V defined in (5.19), converges to the global minimizer, denoted by w_*, of the loss function I^k of the DGF for the solution of PDEs. This result, combined with the convergence of the time-stepping scheme, proves then the convergence of the training error.

Theorem 5.7. *Assume that the neural network satisfies Assumption 5.5 and the coefficients of the PDE* (5.3)–(5.4) *satisfy Assumptions 5.1 and 5.2. Then,*

$$\lim_{t\to\infty} \|V_t - w_*\|_{\mathcal{H}_0^1} = 0.$$

Let us start by rewriting the dynamics of the gradient flow V in (5.19) as follows:

$$\begin{aligned} \frac{\mathrm{d}\left(V_t - w_*\right)(x)}{\mathrm{d}t} &= \frac{\mathrm{d}\left(V_t\right)(x)}{\mathrm{d}t} = -\left\langle \mathcal{D}I^k\left(V_t - w_* + w_*\right), Z(x,\cdot)\right\rangle_{\mathcal{H}_0^1} \\ &= -\widetilde{\mathcal{T}}\left(V_t - w_*\right)(x), \end{aligned} \tag{5.23}$$

with

$$\begin{aligned} \widetilde{\mathcal{T}}(v) :=& \mathcal{T}(v + w_*) \\ =& \left\langle v + w_* - U^{k-1}, Z(x,\cdot)\right\rangle_{L^2} + h\left\langle \mathcal{L}(v+w_*), Z(x,\cdot)\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} \\ &+ h\left\langle F\left(U^{k-1}\right), Z(x,\cdot)\right\rangle_{L^2} \\ =& \left\langle v, Z(x,\cdot)\right\rangle_{L^2} + h\left\langle \mathcal{L}v, Z(x,\cdot)\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} \\ &+ \left\langle w_* - U^{k-1} + h\left(\mathcal{L}w_* + F\left(U^{k-1}\right)\right), Z(x,\cdot)\right\rangle_{L^2} \\ =& \left\langle v, Z(x,\cdot)\right\rangle_{L^2} + h\left\langle \mathcal{L}v, Z(x,\cdot)\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}. \end{aligned}$$

We work with $\widetilde{\mathcal{T}}$ in the sequel, because $\mathcal{T}$ is not linear ($\mathcal{T}(0) \neq 0$). Next, let us define another inner product, such that $\widetilde{\mathcal{T}}$ becomes positive semidefinite. Indeed, for any $u, v \in \mathcal{H}_0^1\left(\mathbb{R}^d\right)$, set

$$\langle v, u\rangle_{\widetilde{\mathcal{H}}_0^1} := \langle v, u\rangle_{L^2} + h\left\langle \mathcal{L}v, u\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1},$$

then, using Assumptions 5.1 and 5.2,

$$\begin{aligned} \|u\|^2_{\widetilde{\mathcal{H}}_0^1} = \langle u, u\rangle_{\widetilde{\mathcal{H}}_0^1} &= \langle u, u\rangle_{L^2} + h\left\langle \mathcal{L}u, u\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} \\ &\begin{cases} \leq (1 + hM)\|u\|^2_{\mathcal{H}_0^1} \\ \geq h\lambda_1\|u\|^2_{\mathcal{H}_0^1} + (1 - h\lambda_2)\|u\|^2_{L^2} \geq h\lambda_1\|u\|^2_{\mathcal{H}_0^1}. \end{cases} \end{aligned}$$

Hence, this inner product induces a norm on $\mathcal{H}_0^1\left(\mathbb{R}^d\right)$, denoted by $\|\cdot\|_{\widetilde{\mathcal{H}}_0^1}$, which is equivalent to $\|\cdot\|_{\mathcal{H}_0^1}$. In this case, we can rewrite $\widetilde{\mathcal{T}}(v)(x) = \langle v, Z(x,\cdot)\rangle_{\widetilde{\mathcal{H}}_0^1}$.

Proposition 5.3. *Assume that the neural network satisfies Assumption 5.5 and the coefficients of the PDE* (5.3)–(5.4) *satisfy Assumption 5.1. Then, $\widetilde{\mathcal{T}}$ is a self-adjoint, positive definite and trace class operator on $\mathcal{H}_0^1\left(\mathbb{R}^d\right)$ with inner product $\langle\cdot,\cdot\rangle_{\widetilde{\mathcal{H}}_0^1}$* i.e. *for any $u, v \in \mathcal{H}_0^1\left(\mathbb{R}^d\right)$ holds*

$$\left\langle \widetilde{\mathcal{T}}(v), u \right\rangle_{\widetilde{\mathcal{H}}_0^1} = \left\langle v, \widetilde{\mathcal{T}}(u) \right\rangle_{\widetilde{\mathcal{H}}_0^1}, \quad \left\langle \widetilde{\mathcal{T}}(v), v \right\rangle_{\widetilde{\mathcal{H}}_0^1} > 0 \ \text{ for } v \neq 0, \quad \text{and}$$
$$\sum_{i=1}^{\infty} \left\langle \widetilde{\mathcal{T}}(\mathrm{e}_i), \mathrm{e}_i \right\rangle_{\widetilde{\mathcal{H}}_0^1} < \infty,$$

with $\{\mathrm{e}_i\}_{i=1}^{\infty}$ an orthogonal basis on $\mathcal{H}_0^1(\mathbb{R}^d)$ under the norm $\langle\cdot,\cdot\rangle_{\widetilde{\mathcal{H}}_0^1}$.

Proof. Let us first verify that $\widetilde{\mathcal{T}}$ is self-adjoint and positive definite. Using that

$$\widetilde{\mathcal{T}}(v)(x) = \langle v, Z(x,\cdot)\rangle_{\widetilde{\mathcal{H}}_0^1} = \mathbb{E}\left[\langle v, X\rangle_{\widetilde{\mathcal{H}}_0^1} \cdot X\right],$$

taking the inner product yields

$$\left\langle \widetilde{\mathcal{T}}(v), u \right\rangle_{\widetilde{\mathcal{H}}_0^1} = \mathbb{E}\left[\langle v, X\rangle_{\widetilde{\mathcal{H}}_0^1} \cdot \langle u, X\rangle_{\widetilde{\mathcal{H}}_0^1}\right] = \left\langle v, \widetilde{\mathcal{T}}(u) \right\rangle_{\widetilde{\mathcal{H}}_0^1},$$
$$\left\langle \widetilde{\mathcal{T}}(v), v \right\rangle_{\widetilde{\mathcal{H}}_0^1} = \mathbb{E}\left[\left|\langle v, X\rangle_{\widetilde{\mathcal{H}}_0^1}\right|^2\right] \geq 0.$$

Next, let us verify that $\mathbb{E}\left[\left|\langle v, X\rangle_{\widetilde{\mathcal{H}}_0^1}\right|^2\right] = 0$ only if $v = 0$. The first marginal of X is $\psi\left(\alpha_0^1 x + c_0^1\right)$ and we know from Assumption 5.5 that the random variables α_0^1 and c_0^1 have full support. Hence, $\mathbb{E}\left[\left|\langle v, X\rangle_{\widetilde{\mathcal{H}}_0^1}\right|^2\right] = 0$ implies $\langle v, w\rangle_{\widetilde{\mathcal{H}}_0^1} = 0$ for any $w \in \mathcal{C}(\psi)$. Using that $\mathcal{C}(\psi)$ is dense in $\mathcal{H}_0^1\left(\mathbb{R}^d\right)$ with the norm $\|\cdot\|_{\mathcal{H}_0^1}$, see Theorem 5.4, then it is dense with the norm $\|\cdot\|_{\widetilde{\mathcal{H}}_0^1}$ as well. Therefore, $v = 0$.

Finally, let us show that $\widetilde{\mathcal{T}}$ is a trace class operator. Using Parseval's identity and Lemma 5.8,

$$\sum_{i=1}^{\infty} \left\langle \widetilde{\mathcal{T}}(\mathrm{e}_i), \mathrm{e}_i \right\rangle_{\widetilde{\mathcal{H}}_0^1} = \sum_{i=1}^{\infty} \mathbb{E}\left[\left|\langle \mathrm{e}_i, X\rangle_{\widetilde{\mathcal{H}}_0^1}\right|^2\right] = \mathbb{E}\left[\|X\|_{\widetilde{\mathcal{H}}_0^1}^2\right] \leq C\mathbb{E}\left[\|X\|_{\mathcal{H}_0^1}^2\right] < \infty. \quad \square$$

Proof of Theorem 5.7. Using van Neerven [85, Proposition 14.13], every trace class operator is compact and positive definite. Therefore, we can do a spectral decomposition for the operator $\widetilde{\mathcal{T}}$. Hence, there exists an orthogonal basis $\{\tilde{\mathrm{e}}_i\}_{i=1}^{\infty}$, such that

$$\widetilde{\mathcal{T}}(\tilde{\mathrm{e}}_i) = \gamma_i \tilde{\mathrm{e}}_i,$$

with $\gamma_1 \geq \gamma_2 \geq \cdots > 0$. Set $h_t^i := \langle V_t - w_*, \tilde{\mathrm{e}}_i\rangle_{\widetilde{\mathcal{H}}_0^1}$. Then, using (5.23),

$$\frac{\mathrm{d}h_t^i}{\mathrm{d}t} = \frac{\langle \mathrm{d}(V_t - w_*), \tilde{\mathrm{e}}_i\rangle_{\widetilde{\mathcal{H}}_0^1}}{\mathrm{d}t} = -\left\langle \widetilde{\mathcal{T}}(V_t - w_*), \tilde{\mathrm{e}}_i \right\rangle_{\widetilde{\mathcal{H}}_0^1} = -\left\langle V_t - w_*, \widetilde{\mathcal{T}}\tilde{\mathrm{e}}_i \right\rangle_{\widetilde{\mathcal{H}}_0^1}$$
$$= -\gamma_i h_t^i.$$

Therefore, $h_t^i = \mathrm{e}^{-\gamma_i t} h_0^i$. Hence, using Parseval's identity again,

$$\|V_t - w_*\|^2_{\widetilde{\mathcal{H}}_0^1} = \sum_{i=1}^{\infty} \left(h_t^i\right)^2 = \sum_{i=1}^{\infty} \mathrm{e}^{-2\gamma_i t} \left(h_0^i\right)^2,$$

which converges to 0 because $\gamma_i > 0$ and $\sum_{i=1}^{\infty} \left(h_0^i\right)^2 = \|w_*\|^2_{\widetilde{\mathcal{H}}_0^1} < \infty$. Finally, since the norm $\|\cdot\|_{\widetilde{\mathcal{H}}_0^1}$ is equivalent to $\|\cdot\|_{\mathcal{H}_0^1}$, we conclude

$$\lim_{t\to\infty} \|V_t - w_*\|_{\mathcal{H}_0^1} = 0. \qquad \square$$

5.A Auxiliary results

5.A.1 Functional inequalities and norm estimates

In the first part of the appendix, we show that the Fréchet derivative of the loss function is continuous, and we also prove that the neural network and its wide network limit are bounded in the $\mathcal{H}_0^1$-norm.

Lemma 5.5 (Continuity of the Fréchet derivative)**.** *Assume that the operators of the PDE* (5.3)–(5.4) *satisfy Assumption 5.1. Then, the Fréchet derivative of the loss function is continuous,* i.e. *there exists a constant $K > 0$, such that for any u, v, w in $\mathcal{H}_0^1(\mathbb{R}^d)$ holds*

$$\left|\left\langle \mathcal{D}I^k(v), u\right\rangle_{\mathcal{H}_0^1} - \left\langle \mathcal{D}I^k(w), u\right\rangle_{\mathcal{H}_0^1}\right| \le K \|v - w\|_{\mathcal{H}_0^1} \|u\|_{\mathcal{H}_0^1}.$$

In particular, by choosing $w = 0$, we have

$$\left|\left\langle \mathcal{D}I^k(v), u\right\rangle_{\mathcal{H}_0^1}\right| \le K \left(1 + \|v\|_{\mathcal{H}_0^1}\right) \|u\|_{\mathcal{H}_0^1}.$$

Proof. Using the definition of the Fréchet derivative in (5.17), the triangle and Cauchy–Schwarz inequalities and Assumption 5.1,

$$\begin{aligned}\left|\left\langle \mathcal{D}I^k(v), u\right\rangle_{\mathcal{H}_0^1} - \left\langle \mathcal{D}I^k(w), u\right\rangle_{\mathcal{H}_0^1}\right| &= \left|\langle v - w, u\rangle_{L^2} + \frac{h}{2}\langle \mathcal{L}(v-w), u\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}\right| \\ &\le \|v - w\|_{L^2} \|u\|_{L^2} + \frac{hM}{2} \|v - w\|_{\mathcal{H}_0^1} \|u\|_{\mathcal{H}_0^1} \\ &\le K \|v - w\|_{\mathcal{H}_0^1} \|u\|_{\mathcal{H}_0^1}.\end{aligned}$$

Setting $w = 0$ and using again the Cauchy–Schwarz inequality and Assumption 5.1, yields

$$\begin{aligned}\left|\left\langle \mathcal{D}I^k(0), u\right\rangle_{\mathcal{H}_0^1}\right| &= \left|-\left\langle U^{k-1}, u\right\rangle_{L^2} + h\left\langle F(U^{k-1}), u\right\rangle_{L^2}\right| \\ &\le \left\|U^{k-1}\right\|_{L^2} \|u\|_{L^2} + h \left\|F(U^{k-1})\right\|_{L^2} \|u\|_{L^2} \le K \|u\|_{\mathcal{H}_0^1},\end{aligned}$$

since $U^{k-1} \in \mathcal{H}_0^1$. Then, combining the two results and using the triangle inequality,

$$\begin{aligned}\left|\left\langle \mathcal{D}I^k(v), u\right\rangle_{\mathcal{H}_0^1}\right| &= \left|\left\langle \mathcal{D}I^k(v), u\right\rangle_{\mathcal{H}_0^1} - \left\langle \mathcal{D}I^k(0), u\right\rangle_{\mathcal{H}_0^1} + \left\langle \mathcal{D}I^k(0), u\right\rangle_{\mathcal{H}_0^1}\right| \\ &\leq \left|\left\langle \mathcal{D}I^k(v), u\right\rangle_{\mathcal{H}_0^1} - \left\langle \mathcal{D}I^k(0), u\right\rangle_{\mathcal{H}_0^1}\right| + \left|\left\langle \mathcal{D}I^k(0), u\right\rangle_{\mathcal{H}_0^1}\right| \\ &\leq K\left(1 + \|v\|_{\mathcal{H}_0^1}\right)\|u\|_{\mathcal{H}_0^1}. \qquad \square\end{aligned}$$

Lemma 5.6. *Assume that the neural network satisfies Assumption 5.5 and the operators of the PDE* (5.3)–(5.4) *satisfy Assumptions 5.1 and 5.2. Then, for all* $t \geq 0$,

$$\mathbb{E}\left[\|V_t^n\|_{\mathcal{H}_0^1}^2\right] \leq C_\psi \quad \text{and} \quad \|V_t\|_{\mathcal{H}_0^1}^2 \leq C_\psi,$$

with C_ψ *a positive constant that only depends on the activation function* ψ.

Proof. Let us first show that $I^k(V_t^n)$ is not increasing in t. According to (5.16),

$$\frac{\mathrm{d}I^k(V_t^n)}{\mathrm{d}t} = \nabla_\theta I^k(V_t^n) \cdot \frac{\mathrm{d}\theta_t^n}{\mathrm{d}t} = -\eta_n \left|\nabla_\theta I^k\left(V^n(\theta_t^n; x)\right)\right|^2 \leq 0.$$

This inequality readily implies $I^k(V_t^n) \leq I^k(V_0^n)$. Using Assumption 5.1, the Cauchy–Schwarz inequality and $ab \leq \frac{a^2+b^2}{2}$,

$$\begin{aligned}I^k(u) &= \frac{1}{2}\left\|u - U^{k-1}\right\|_{L^2}^2 + \frac{h}{2}\left\langle \mathcal{L}u, u\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} + h\left\langle F\left(U^{k-1}\right), u\right\rangle_{L^2} \\ &= \frac{1}{2}\|u\|_{L^2}^2 + \frac{h}{2}\left\langle \mathcal{L}u, u\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} + \left\langle hF\left(U^{k-1}\right) - U^{k-1}, u\right\rangle_{L^2} + \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2 \\ &\leq \frac{1}{2}\|u\|_{L^2}^2 + \frac{hM}{2}\|u\|_{\mathcal{H}_0^1}^2 + \left\|hF(U^{k-1}) - U^{k-1}\right\|_{L^2}\|u\|_{L^2} + \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2 \\ &\leq \|u\|_{L^2}^2 + \frac{hM}{2}\|u\|_{\mathcal{H}_0^1}^2 + \frac{1}{2}\left\|hF(U^{k-1}) - U^{k-1}\right\|_{L^2}^2 + \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2 \\ &= C_1\|u\|_{\mathcal{H}_0^1}^2 + C_2.\end{aligned}$$

Moreover, using Assumption 5.2, the Cauchy–Schwarz inequality again and the inequality

$$\|m\|_{L^2}\|n\|_{L^2} \leq \frac{\lambda}{2}\|m\|_{L^2}^2 + \frac{1}{2\lambda}\|v\|_{L^2}^2 \text{ with } \lambda = \frac{h\lambda_1}{2},$$

$$
\begin{aligned}
I^k(u) =& \frac{1}{2}\|u\|_{L^2}^2 + \frac{h}{2}\left\langle \mathcal{L}u, u\right\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} + \left\langle hF\left(U^{k-1}\right) - U^{k-1}, u\right\rangle_{L^2} + \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2 \\
\geq& \left(\frac{1}{2} - \frac{h\lambda_2}{2}\right)\|u\|_{L^2}^2 + \frac{h\lambda_1}{2}\|u\|_{\mathcal{H}_0^1}^2 - \left\|hF\left(U^{k-1}\right) - U^{k-1}\right\|_{L^2}\|u\|_{L^2} \\
& - \frac{1}{2}\left\|U^{k-1}\right\|_{L_2}^2 \\
\geq& \left(\frac{1}{2} - \frac{h\lambda_2}{2}\right)\|u\|_{L^2}^2 + \frac{h\lambda_1}{4}\|u\|_{\mathcal{H}_0^1}^2 - \frac{1}{h\lambda_1}\left\|hF\left(U^{k-1}\right) - U^{k-1}\right\|_{L^2}^2 \\
& - \frac{1}{2}\left\|U^{k-1}\right\|_{L^2}^2 \\
=& C_3\|u\|_{\mathcal{H}_0^1}^2 - C_4.
\end{aligned}
$$

Therefore, since $I^k\left(V_t^n\right)$ is not increasing with t and using Lemma 5.4,

$$
\begin{aligned}
\mathbb{E}\left[\|V_t^n\|_{\mathcal{H}_0^1}^2\right] &\leq \mathbb{E}\left[\frac{1}{C_3}I^k\left(V_t^n\right) + \frac{C_4}{C_3}\right] \leq \mathbb{E}\left[\frac{1}{C_3}I^k\left(V_0^n\right) + \frac{C_4}{C_3}\right] \\
&\leq \mathbb{E}\left[\frac{C_1}{C_3}\|V_0^n\|_{\mathcal{H}_0^1}^2 + \frac{C_2 + C_4}{C_3}\right] \leq C_\psi.
\end{aligned}
$$

Using similar arguments,

$$
\frac{\mathrm{d}I^k\left(V_t\right)}{\mathrm{d}t} = \left\langle \mathcal{D}I^k\left(V_t\right), \frac{\mathrm{d}V_t}{\mathrm{d}t}\right\rangle_{\mathcal{H}_0^1} = -\left|\left\langle \mathcal{D}I^k\left(V_t\right), \mathbb{E}\left[\nabla_\theta \beta_0 \psi\left(\alpha_0 x + c_0\right)\right]\right\rangle_{\mathcal{H}_0^1}\right|^2 \leq 0.
$$

This inequality yields that $I^k\left(V_t\right) \leq I^k\left(V_0\right) = I^k(0)$, hence $\|V_t\|_{\mathcal{H}_0^1}^2 \leq C_\psi$. □

5.A.2 Gradient θ estimates

This subsection contains several useful results concerning gradient estimates of the neurons of the neural network with respect to its parameters. Moreover, we show that gradients of the neurons are Lipschitz continuous with respect to the parameters θ of the network, that they converge to their "unclipped" analogs for large n, and we estimate the distance between the parameters as the training progresses.

Lemma 5.7 ($\mathcal{H}_0^1$-boundedness of X^n)**.** *Let $\theta \in \mathbb{R}\times\mathbb{R}\times\mathbb{R}^d$, then X^n is bounded in $\mathcal{H}_0^1$,* i.e. *there exists a constant $C_\psi > 0$ such that*

$$
\|X^n(\theta)\|_{\mathcal{H}_0^1}^2 \leq C_\psi\left(r_n\right)^{d+8}.
$$

Proof. The derivatives of a neuron with respect to its parameters equal

$$
\begin{aligned}
\frac{\partial}{\partial\beta}\hat{\beta}\psi\left(\hat{\alpha}x + \hat{c}\right) &= \psi\left(\hat{\alpha}x + \hat{c}\right)\mathbf{1}_{\{|\beta|\leq r_n\}}, \\
\frac{\partial}{\partial\alpha}\hat{\beta}\psi\left(\hat{\alpha}x + \hat{c}\right) &= \hat{\beta}x^{\mathsf{T}}\left(\nabla\psi\right)\left(\hat{\alpha}x + \hat{c}\right)\mathbf{1}_{\{\frac{1}{r_n}\leq|\alpha|\leq r_n\}}, \\
\frac{\partial}{\partial c}\hat{\beta}\psi\left(\hat{\alpha}x + \hat{c}\right) &= \hat{\beta}\left(\nabla\psi\right)\left(\hat{\alpha}x + \hat{c}\right)\mathbf{1}_{\{|c|\leq r_n\}}.
\end{aligned}
$$

Therefore, we obtain the bound

$$\begin{aligned}|X^n(\theta)| &= \left|\frac{\partial}{\partial\beta}\hat{\beta}\psi\left(\hat{\alpha}x+\hat{c}\right)+\frac{\partial}{\partial\alpha}\hat{\beta}\psi\left(\hat{\alpha}x+\hat{c}\right)+\frac{\partial}{\partial c}\hat{\beta}\psi\left(\hat{\alpha}x+\hat{c}\right)\right| \\ &\leq \left|\psi\left(\hat{\alpha}x+\hat{c}\right)\mathbf{1}_{\{|\beta|\leq r_n\}}\right| + \left|\hat{\beta}x^{\mathsf{T}}\left(\nabla\psi\right)\left(\hat{\alpha}x+\hat{c}\right)\mathbf{1}_{\{\frac{1}{r_n}\leq|\alpha|\leq r_n\}}\right| \\ &\quad + \left|\hat{\beta}\left(\nabla\psi\right)\left(\hat{\alpha}x+\hat{c}\right)\mathbf{1}_{\{|c|\leq r_n\}}\right| \\ &\leq \left|\psi\left(\hat{\alpha}x+\hat{c}\right)\right| + r_n\left|x\cdot\nabla\psi\left(\hat{\alpha}x+\hat{c}\right)\right| + r_n\left|\left(\nabla\psi\right)\left(\hat{\alpha}x+\hat{c}\right)\right|.\end{aligned}$$

The second term above can be bounded by

$$\begin{aligned}r_n\left|x\cdot\nabla\psi\left(\hat{\alpha}x+\hat{c}\right)\right| &\leq r_n(\hat{\alpha}_n)^{-1}\Big(\left|(\hat{\alpha}_n x+\hat{c})\cdot\nabla\psi\left(\hat{\alpha}x+\hat{c}\right)\right| + \left|\hat{c}\cdot\nabla\psi\left(\hat{\alpha}x+\hat{c}\right)\right|\Big) \\ &\leq C_\psi(r_n)^3.\end{aligned}$$

Therefore, using that $|\hat{\alpha}|\geq(r_n)^{-1}$,

$$\begin{aligned}\int_{\mathbb{R}^d}|X^n(\theta)|^2\,\mathrm{d}x &\leq \int_{\mathbb{R}^d}\left|\psi\left(\hat{\alpha}x+\hat{c}\right)\right|^2\mathrm{d}x + (r_n)^2\int_{\mathbb{R}^d}|x|^2\left|\left(\nabla\psi\right)\left(\hat{\alpha}x+\hat{c}\right)\right|^2\mathrm{d}x \\ &\quad + (r_n)^2\int_{\mathbb{R}^d}\left|\left(\nabla\psi\right)\left(\hat{\alpha}x+\hat{c}\right)\right|^2\mathrm{d}x \\ &\overset{(y=\hat{\alpha}x)}{=} |\hat{\alpha}|^{-d}\int_{\mathbb{R}^d}\left|\psi\left(y+\hat{c}\right)\right|^2\mathrm{d}y \\ &\quad + (r_n)^2|\hat{\alpha}|^{-d-2}\int_{\mathbb{R}^d}|y|^2\left|\left(\nabla\psi\right)\left(y+\hat{c}\right)\right|^2\mathrm{d}y \\ &\quad + (r_n)^2|\hat{\alpha}|^{-d}\int_{\mathbb{R}^d}\left|\left(\nabla\psi\right)\left(y+\hat{c}\right)\right|^2\mathrm{d}y \\ &\overset{(z=y+\hat{c})}{\leq} (r_n)^d\int_{\mathbb{R}^d}\left|\psi\left(z\right)\right|^2\mathrm{d}z + (r_n)^{d+4}\int_{\mathbb{R}^d}|z-\hat{c}|^2\left|\left(\nabla\psi\right)\left(z\right)\right|^2\mathrm{d}z \\ &\quad + (r_n)^{d+2}\int_{\mathbb{R}^d}\left|\left(\nabla\psi\right)\left(z\right)\right|^2\mathrm{d}z \\ &\leq C_\psi(r_n)^{d+6}\end{aligned}$$

Analogously we can estimate the gradient,

$$\begin{aligned}|\nabla_x X^n(\theta)| \leq& 2r_n\left|\nabla\psi\left(\hat{\alpha}x+\hat{c}\right)\right| + (r_n)^2|x|\left|\left(\mathcal{D}^2\psi\right)\left(\hat{\alpha}x+\hat{c}\right)\right| \\ &+ (r_n)^2\left|\left(\mathcal{D}^2\psi\right)\left(\hat{\alpha}x+\hat{c}\right)\right|,\end{aligned}$$

and hence

$$\int_{\mathbb{R}^d}|\nabla_x X^n(\theta)|^2\,\mathrm{d}x \leq C_\psi(r_n)^{d+8}.$$

Finally, we conclude

$$\|X^n(\theta)\|^2_{\mathcal{H}^1_0} = \int_{\mathbb{R}^d}|X^n(\theta)|^2+|\nabla_x X^n(\theta)|^2\,\mathrm{d}x \leq C_\psi\left(r_n\right)^{d+8}. \qquad \square$$

Lemma 5.8 ($\mathcal{H}_0^1$-boundedness of X)**.** *Let $\theta \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^d$, and assume that the neural network satisfies Assumption 5.5. Then, X is bounded in $\mathcal{H}_0^1$,* i.e. *there exists a constant $C_\psi > 0$ such that*

$$\|X(\theta)\|_{\mathcal{H}_0^1}^2 \leq C_\psi \left(1+\beta^2\right)\left(1+c^2\right)\left(|\alpha|^{-d} + |\alpha|^{-d-2} + |\alpha|^{2-d}\right).$$

Moreover,

$$\mathbb{E}\left[\|X\|_{\mathcal{H}_0^1}^2\right] < \infty \quad \text{and} \quad \sup_{n\geq 1} \mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^2\right] < \infty.$$

Proof. Let us first consider the L^2-norm of $X(\theta)$. As in the proof of Lemma 5.7,

$$\begin{aligned}
\|X(\theta)\|_{L^2}^2 \leq & \int_{\mathbb{R}^d} |\psi(\alpha x + c)|^2 \,\mathrm{d}x + \beta^2 \int_{\mathbb{R}^d} |x|^2 \,|(\nabla\psi)(\alpha x + c)|^2 \,\mathrm{d}x \\
& + \beta^2 \int_{\mathbb{R}^d} |(\nabla\psi)(\alpha x + c)|^2 \,\mathrm{d}x \\
= & |\alpha|^{-d} \int_{\mathbb{R}^d} |\psi(z)|^2 \,\mathrm{d}z + \beta^2 |\alpha|^{-d-2} \int_{\mathbb{R}^d} |z-c|^2 \,|(\nabla\psi)(z)|^2 \,\mathrm{d}z \\
& + \beta^2 |\alpha|^{-d} \int_{\mathbb{R}^d} |(\nabla\psi)(z)|^2 \,\mathrm{d}z \\
\leq & C_\psi \left(1+\beta^2\right)\left(1+|c|^2\right)\left(|\alpha|^{-d} + |\alpha|^{-d-2}\right),
\end{aligned}$$

For the equality in the second step we used the change of variables $z = \alpha x + c$. Analogously,

$$\|\nabla X(\theta)\|_{L^2}^2 \leq C_\psi \left(1+\beta^2\right)\left(1+|c|^2\right)\left(|\alpha|^{-d} + |\alpha|^{2-d}\right).$$

Combining these two results, we recover the $\mathcal{H}_0^1$-estimate of $X(\theta)$. Taking expectations on the $\mathcal{H}_0^1$-estimate of $X(\theta)$ and using Assumption 5.5,

$$\begin{aligned}
\mathbb{E}\left[\|X\|_{\mathcal{H}_0^1}^2\right] &\leq C_\psi \mathbb{E}\left[1+\left|\beta_0^1\right|^2\right] \mathbb{E}\left[1+\left|c_0^1\right|^2\right] \mathbb{E}\left[\left|\alpha_0^1\right|^{-d} + \left|\alpha_0^1\right|^{-d-2} + \left|\alpha_0^1\right|^{2-d}\right] \\
&< \infty,
\end{aligned}$$

while using the $\mathcal{H}_0^1$-estimate of $X^n(\theta)$ from the previous lemma,

$$\begin{aligned}
\mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^2\right] &\leq C_\psi \mathbb{E}\left[1+\left|\hat{\beta}_0^1\right|^2\right] \mathbb{E}\left[1+\left|\hat{c}_0^1\right|^2\right] \mathbb{E}\left[\left|\hat{\alpha}_0^1\right|^{-d} + \left|\hat{\alpha}_0^1\right|^{-d-2} + \left|\hat{\alpha}_0^1\right|^{2-d}\right] \\
&\leq C_\psi \underbrace{\mathbb{E}\left[1+\left|\beta_0^1\right|^2\right] \mathbb{E}\left[1+\left|c_0^1\right|^2\right]}_{<\infty} \mathbb{E}\left[\left|\hat{\alpha}_0^1\right|^{-d} + \left|\hat{\alpha}_0^1\right|^{-d-2} + \left|\hat{\alpha}_0^1\right|^{2-d}\right].
\end{aligned}$$

Using a similar reasoning as in the proof of Lemma 5.4, if $d = 1, 2$,

$$\begin{aligned}
&\mathbb{E}\left[\left|\hat{\alpha}_0^1\right|^{-d} + \left|\hat{\alpha}_0^1\right|^{-d-2} + \left|\hat{\alpha}_0^1\right|^{2-d}\right] \\
\leq &\mathbb{E}\left[\left|\alpha_0^1\right|^{-d} + \left|\alpha_0^1\right|^{-d-2} + (r_n)^{-d} + (r_n)^{-d-2} + \left|\alpha_0^1\right| + 1\right] < \infty,
\end{aligned}$$

and if $d \geq 3$,

$$\begin{aligned}&\mathbb{E}\left[\left|\hat{\alpha}_0^1\right|^{-d} + \left|\hat{\alpha}_0^1\right|^{-d-2} + \left|\hat{\alpha}_0^1\right|^{2-d}\right]\\ \leq&\mathbb{E}\left[\left|\alpha_0^1\right|^{-d} + \left|\alpha_0^1\right|^{-d-2} + (r_n)^{-d} + (r_n)^{-d-2} + \left|\alpha_0^1\right|^{2-d} + (r_n)^{2-d}\right] < \infty.\end{aligned}$$

Therefore, $\sup_{n\geq 1}\mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^2\right] < \infty$. □

Lemma 5.9. *Assume that the neural network satisfies Assumption 5.5. Then*

$$\varepsilon_n := \mathbb{E}\left[\|X^n - X\|_{\mathcal{H}_0^1}^2\right] \xrightarrow[n\to\infty]{} 0.$$

Proof. Let us decompose ε_n as follows,

$$\begin{aligned}\varepsilon_n =&\mathbb{E}\Big[\|X^n - X\|_{\mathcal{H}_0^1}^2 \Big(\mathbf{1}_{|\beta_0^1|\leq r_n, |\alpha_0^1|\leq r_n, |c_0^1|\leq r_n} + \mathbf{1}_{|\beta_0^1|> r_n, |\alpha_0^1|\leq r_n, |c_0^1|\leq r_n}\\ &+\mathbf{1}_{|\alpha_0^1|> r_n, |c_0^1|\leq r_n} + \mathbf{1}_{|c_0^1|> r_n}\Big)\Big],\end{aligned}$$

and then we treat each summand separately.
Term 1. By definition $\|X^n - X\|_{\mathcal{H}_0^1}^2 \mathbf{1}_{\{|\beta_0^1|\leq r_n, |\alpha_0^1|\leq r_n, |c_0^1|\leq r_n\}} = 0$.
Term 2. Let $|\beta_0^1| > r_n$ and $|\alpha_0^1| \leq r_n$, then $\|X^n\|_{\mathcal{H}_0^1} \leq \|X\|_{\mathcal{H}_0^1}$. Hence,

$$\mathbb{E}\left[\|X^n - X\|_{\mathcal{H}_0^1}^2 \mathbf{1}_{\{|\beta_0^1|> r_n, |\alpha_0^1|\leq r_n, |c_0^1|\leq r_n\}}\right] \leq 2\mathbb{E}\left[\|X\|_{\mathcal{H}_0^1}^2 \mathbf{1}_{\{|\beta_0^1|> r_n, |\alpha_0^1|\leq r_n, |c_0^1|\leq r_n\}}\right],$$

which converges to 0 by the dominated convergence theorem.
Terms 3 & 4. The following inequality holds in this case

$$\begin{aligned}&\|X^n - X\|_{\mathcal{H}_0^1}^2 \left(\mathbf{1}_{\{|\alpha_0^1|> r_n, |c_0^1|\leq r_n\}} + \mathbf{1}_{|c_0^1|> r_n}\right)\\ &\leq 2\left(\|X^n\|_{\mathcal{H}_0^1}^2 + \|X\|_{\mathcal{H}_0^1}^2\right)\left(\mathbf{1}_{\{|\alpha_0^1|> r_n, |c_0^1|\leq r_n\}} + \mathbf{1}_{|c_0^1|> r_n}\right).\end{aligned}$$

Using the dominated convergence theorem,

$$\mathbb{E}\left[\|X\|_{\mathcal{H}_0^1}^2 \left(\mathbf{1}_{\{|\alpha_0^1|> r_n, |c_0^1|\leq r_n\}} + \mathbf{1}_{|c_0^1|> r_n\}}\right)\right] \xrightarrow[n\to\infty]{} 0.$$

Moreover, applying Lemma 5.7,

$$\|X^n\|_{\mathcal{H}_0^1}^2 \left(\mathbf{1}_{\{|\alpha_0^1|> r_n\}} + \mathbf{1}_{\{|c_0^1|> r_n\}}\right) \leq C_\psi (r_n)^{d+8}\left(\mathbf{1}_{\{|\alpha_0^1|> r_n\}} + \mathbf{1}_{\{|c_0^1|> r_n\}}\right),$$

which converges to 0 almost surely. Therefore, using the dominated convergence theorem once again,

$$\begin{aligned}\mathbb{E}\left[\|X^n\|_{\mathcal{H}_0^1}^2 \left(\mathbf{1}_{\{|\alpha_0^1|> r_n\}} + \mathbf{1}_{\{|c_0^1|> r_n\}}\right)\right] &\leq \mathbb{E}\left[\left|\alpha_0^1\right|^{d+8} \mathbf{1}_{\{|\alpha_0^1|> r_n\}} + \left|c_0^1\right|^{d+8} \mathbf{1}_{\{|c_0^1|> r_n\}}\right]\\ &\xrightarrow[n\to\infty]{} 0.\end{aligned}$$

□

Lemma 5.10 (θ-Lipschitz continuity)**.** *Let $\theta, \theta' \in \mathbb{R}\times\mathbb{R}\times\mathbb{R}^d$, then X^n is Lipschitz continuous in $\mathcal{H}_0^1$,* i.e. *there exists a constant $C_\psi > 0$ such that*

$$\|X^n(\theta) - X^n(\theta')\|_{\mathcal{H}_0^1} \leq C_\psi \left(r_n\right)^{4+\frac{d}{2}} |\theta - \theta'|^{\frac{1}{2}} .$$

Proof. As in the proof of Lemma 5.7,

$$\begin{aligned}\|X^n(\theta) - X^n(\theta')\|_{\mathcal{H}_0^1} \leq & \|\psi\left(\hat{\alpha}\cdot+\hat{c}\right) - \psi\left(\hat{\alpha}'\cdot+\hat{c}'\right)\|_{\mathcal{H}_0^1} \\ & + \left\|\hat{\beta}\left(\cdot\nabla\psi\right)\left(\hat{\alpha}\cdot+\hat{c}\right) - \hat{\beta}'\cdot\left(\nabla\psi\right)\left(\hat{\alpha}'\cdot+\hat{c}'\right)\right\|_{\mathcal{H}_0^1} \\ & + \left\|\hat{\beta}\left(\nabla\psi\right)\left(\hat{\alpha}\cdot+\hat{c}\right) - \hat{\beta}'\left(\nabla\psi\right)\left(\hat{\alpha}'\cdot+\hat{c}'\right)\right\|_{\mathcal{H}_0^1} . \end{aligned} \tag{5.24}$$

Since $|\hat{\alpha}|, |\hat{\beta}|, |\hat{c}| \leq r_n$, $|\hat{\alpha}| \geq r_n^{-1}$ and $\psi \in C_c^\infty\left(\mathbb{R}^d\right)$, and therefore Lipschitz,

$$\begin{aligned}&\int_{\mathbb{R}^d} \left|\psi\left(\hat{\alpha}x+\hat{c}\right) - \psi\left(\hat{\alpha}'x+\hat{c}'\right)\right|^2 \mathrm{d}x \\ \leq & \int_{\mathbb{R}^d} C_\psi \left|\left(\hat{\alpha}-\hat{\alpha}'\right)x + \hat{c} - \hat{c}'\right| \left|\psi\left(\hat{\alpha}x+\hat{c}\right) - \psi\left(\hat{\alpha}'x+\hat{c}'\right)\right| \mathrm{d}x \\ \leq & C_\psi |\theta-\theta'| \int_{\mathbb{R}^d} (1+|x|)\left(\left|\psi\left(\hat{\alpha}x+\hat{c}\right)\right| + \left|\psi\left(\hat{\alpha}'x+\hat{c}'\right)\right|\right) \mathrm{d}x.\end{aligned}$$

Using the change of variables $y = \hat{\alpha}x + \hat{c}$,

$$\begin{aligned}\int_{\mathbb{R}^d} (1+|x|)\left|\psi\left(\hat{\alpha}x+\hat{c}\right)\right| \mathrm{d}x &= |\hat{\alpha}|^{-d} \int_{\mathbb{R}^d} |\psi(y)| \,\mathrm{d}y + |\hat{\alpha}|^{-d-1} \int_{\mathbb{R}^d} |y-\hat{c}| \left|\psi\left(y\right)\right| dy \\ &\leq C_\psi\left(r_n^d + r_n^{d+2}\right),\end{aligned}$$

and we obtain the bound

$$\int_{\mathbb{R}^d} \left|\psi\left(\hat{\alpha}x+\hat{c}\right) - \psi\left(\hat{\alpha}'x+\hat{c}'\right)\right|^2 \mathrm{d}x \leq C_\psi |\theta-\theta'| \, r_n^{d+2}.$$

Analogously, using that $\nabla\psi$ is Lipschitz as well,

$$\begin{aligned}&\int_{\mathbb{R}^d} \left|\hat{\alpha}\nabla\psi\left(\hat{\alpha}x+\hat{c}\right) - \hat{\alpha}'\nabla\psi\left(\hat{\alpha}'x+\hat{c}'\right)\right|^2 \mathrm{d}x \\ \leq & 2\int_{\mathbb{R}^d} \left|\hat{\alpha}-\hat{\alpha}'\right|^2 \left|\nabla\psi\left(\hat{\alpha}x+\hat{c}\right)\right|^2 \mathrm{d}x + 2\int_{\mathbb{R}^d} \left|\hat{\alpha}'\right|^2 \left|\nabla\psi\left(\hat{\alpha}x+\hat{c}\right) - \nabla\psi\left(\hat{\alpha}'x+\hat{c}'\right)\right|^2 \mathrm{d}x \\ \leq & 4r_n |\theta-\theta'| \int_{\mathbb{R}^d} \left|\nabla\psi\left(\hat{\alpha}x+\hat{c}\right)\right|^2 \mathrm{d}x + 2r_n^2 \int_{\mathbb{R}^d} \left|\nabla\psi\left(\hat{\alpha}x+\hat{c}\right) - \nabla\psi\left(\hat{\alpha}'x+\hat{c}'\right)\right|^2 \mathrm{d}x \\ \leq & C_\psi |\theta-\theta'| \left(r_n\right)^{d+4}.\end{aligned}$$

Therefore, we arrive at the following bound for the $\mathcal{H}_0^1$-norm of this term

$$\|\psi\left(\hat{\alpha}\cdot+\hat{c}\right) - \psi\left(\hat{\alpha}'\cdot+\hat{c}'\right)\|_{\mathcal{H}_0^1} \leq C_\psi |\theta-\theta'|^{1/2} \left(r_n\right)^{d/2+2}.$$

We can similarly estimate the other two terms in (5.24):

$$\left\|\hat{\beta}\left(\cdot\nabla\psi\right)\left(\hat{\alpha}\cdot+\hat{c}\right)-\hat{\beta}'\cdot\left(\nabla\psi\right)\left(\hat{\alpha}'\cdot+\hat{c}'\right)\right\|_{\mathcal{H}_0^1}$$
$$+\left\|\hat{\beta}\left(\nabla\psi\right)\left(\hat{\alpha}\cdot+\hat{c}\right)-\hat{\beta}'\left(\nabla\psi\right)\left(\hat{\alpha}'\cdot+\hat{c}'\right)\right\|_{\mathcal{H}_0^1}\leq C_\psi\left|\theta-\theta'\right|^{1/2}\left(r_n\right)^{d/2+4}.$$

We conclude

$$\left\|X^n(\theta)-X^n\left(\theta'\right)\right\|_{\mathcal{H}_0^1}\leq C_\psi\left(r_n\right)^{4+\frac{d}{2}}\left|\theta-\theta'\right|^{\frac{1}{2}}. \qquad \square$$

Lemma 5.11. *Assume that the neural network satisfies Assumption 5.5. Then, for* $t\geq 0$*,*

$$\mathbb{E}\left[\left|\theta_t^{i,n}-\theta_0^{i,n}\right|\right]\leq C_\psi t n^{\delta-1}\left(r_n\right)^{\frac{d}{2}+4}.$$

Remark 5.8. *This lemma yields that, when* n *is large, then the value* $\theta_t^{i,n}$ *of the evolution of the parameters of the neural network does not differ significantly from its initial value* $\theta_0^{i,n}$*.*

Proof. By (5.16),

$$\theta_t^{i,n}-\theta_0^{i,n}=-\eta_n\int_0^t\left\langle\mathcal{D}I^k\left(V_s^n\right),\nabla_{\theta^i}V_s^n\right\rangle_{\mathcal{H}_0^1}\mathrm{d}s,$$

thus, their squared difference equals

$$\begin{aligned}\left|\theta_t^{i,n}-\theta_0^{i,n}\right|^2&=\left|\eta_n\right|^2\left|\int_0^t\left\langle\mathcal{D}I^k\left(V_s^n\right),\nabla_{\theta^i}V_s^n\right\rangle_{\mathcal{H}_0^1}\mathrm{d}s\right|^2\\&\leq n^{4\delta-2}t\int_0^t\left\langle\mathcal{D}I^k\left(V_s^n\right),\nabla_{\theta^i}V_s^n\right\rangle_{\mathcal{H}_0^1}^2\mathrm{d}s.\end{aligned}$$

Using Lemma 5.5,

$$\left|\left\langle\mathcal{D}I^k\left(V_s^n\right),\nabla_{\theta^i}V_s^n\right\rangle_{\mathcal{H}_0^1}\right|^2\leq K\left(1+\left\|V_s^n\right\|_{\mathcal{H}_0^1}^2\right)\left\|\nabla_{\theta^i}V_s^n\right\|_{\mathcal{H}_0^1}^2.$$

Using that

$$\left|\nabla_{\theta^i}V^n\left(\theta_t^n;x\right)\right|=\frac{1}{n^\delta}\left|\mathcal{X}^n\left(\theta_t^{i,n};x\right)\right|,$$

and applying Lemma 5.7,

$$\left\|\nabla_{\theta^i}V_t^n\right\|_{\mathcal{H}_0^1}^2\leq\frac{C_\psi}{n^{2\delta}}\left(r_n\right)^{d+8}.$$

Moreover, from Lemma 5.6,

$$\mathbb{E}\left[\left\|V_t^n\right\|_{\mathcal{H}_0^1}^2\right]\leq C_\psi.$$

Hence, we can bound the norm of the square difference by

$$\begin{aligned}\mathbb{E}\left[\left|\theta_t^{i,n}-\theta_0^{i,n}\right|^2\right]&\leq C_\psi t n^{2(\delta-1)}\left(r_n\right)^{d+8}\int_0^t\mathbb{E}\left[\left\|V_s^n\right\|_{\mathcal{H}_0^1}^2+1\right]\mathrm{d}s\\&\leq C_\psi t^2 n^{2(\delta-1)}\left(r_n\right)^{d+8}.\end{aligned}$$

The result follows now using Jensen's inequality. $\square$

5.A.3 Examples

Proposition 5.4. *The Black–Scholes PDE from Example 5.3 satisfies Assumptions 5.1 and 5.2*

Proof. In the Black–Scholes PDE,

$$\mathcal{L}u = -\frac{\sigma^2}{2}\frac{\partial^2 u}{\partial x^2} + ru,$$
$$F(u) = \left(\frac{\sigma^2}{2} - r\right)\frac{\partial u}{\partial x}.$$

Then the energy functional is

$$I^k(u) = \frac{1}{2}\left\|u - U^{k-1}\right\|_{L^2}^2 + \frac{h}{2}\int_{\mathbb{R}} \frac{\sigma^2}{2}\left(\frac{\partial u}{\partial x}\right)^2 + ru^2 dx + h\int_{\mathbb{R}} F\left(U^{k-1}\right) u dx.$$

Using the triangle and Cauchy–Schwarz inequality

$$\begin{aligned}
\left|\langle \mathcal{L}u, v\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1}\right| &= \left|\int_{\mathbb{R}} \frac{\sigma^2}{2}\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + ruv dx\right| \leq \left|\frac{\sigma^2}{2}\right|\left|\int_{\mathbb{R}} \frac{\partial u}{\partial x}\frac{\partial v}{\partial x} dx\right| + |r|\left|\int_{\mathbb{R}} uv dx\right| \\
&\leq \left|\frac{\sigma^2}{2}\right|\left\|\frac{\partial u}{\partial x}\right\|_{L^2}\left\|\frac{\partial v}{\partial x}\right\|_{L^2} + |r|\,\|u\|_{L^2}\,\|v\|_{L^2} \\
&\leq \left(\left|\frac{\sigma^2}{2}\right| + |r|\right)\|u\|_{\mathcal{H}_0^1}\,\|v\|_{\mathcal{H}_0^1} \\
\|F(u)\|_{L^2} &= \left\|\left(\frac{\sigma^2}{2} - r\right)\frac{\partial u}{\partial x}\right\|_{L^2} \leq \left|\frac{\sigma^2}{2} - r\right|\|u\|_{\mathcal{H}_0^1}.
\end{aligned}$$

So Assumption 5.1 is satisfied with $M = \left|\frac{\sigma^2}{2}\right| + |r|$. Furthermore,

$$\langle \mathcal{L}u, u\rangle_{\mathcal{H}^{-1},\mathcal{H}_0^1} = \int_{\mathbb{R}} \frac{\sigma^2}{2}\left(\frac{\partial u}{\partial x}\right)^2 + ru^2 dx \geq \left(\frac{\sigma^2}{2} + r\right)\|u\|_{\mathcal{H}_0^1}^2.$$

So Assumption 5.2 is satisfied with $\lambda_1 = \frac{\sigma^2}{2} + r > 0$ and $\lambda_2 = 0$. □

6

Error analysis of deep PDE solvers for option pricing

By the previous chapter, there exists a neural network that converges to the solution of the partial differential equation (PDE). Furthermore given enough resources in terms of network size and training time when training, the network converges to the correct solution. In practical situations, however, we do not have infinite resources. For the real-world applicability of deep learning PDE solvers it is essential to understand their empirical and quantitative accuracy well.

In this chapter, we verify empirically the results from the previous chapter by showing that using more resources leads to a smaller error. Furthermore, we offer actionable insights into the utility of Deep PDE solvers for practical option pricing implementation. Through comparative experiments in both the Black–Scholes and the Heston model, we assess the empirical performance of two neural network algorithms to solve PDEs: the Deep Galerkin Method (DGM) and the Time Deep Gradient Flow method (TDGF). We determine their empirical convergence rates and training time as functions of (i) the number of layers, (ii) the number of nodes per layer, (iii) the number of epochs, and (iv) the number of samples in each epoch. For the TDGF, we also consider the order of the discretization scheme and the number of time steps.

6.1 Introduction

Option pricing is a fundamental problem in finance. Since the seminal work of Black and Scholes [23], numerous mathematical models and computational approaches have been developed to determine option prices. One common approach formulates the price of an option as the solution to a PDE, which can be solved numerically using methods such as finite differences or finite elements. However, traditional grid-based methods suffer from the curse of dimensionality: the number of grid

This chapter is based on J. Rou. Error analysis of deep PDE solvers for option pricing. Preprint arXiv:2505.05121, 2025.

points grows exponentially with the dimension of the problem. This challenge is particularly acute in high-dimensional settings, such as basket options or Markovian approximations of rough volatility models [3, 72].

Neural networks provide a promising alternative by efficiently approximating solutions to PDEs. Once trained, they can generate option prices rapidly, bypassing the limitations of conventional numerical methods. Deep learning-based approaches have been successfully applied in other financial contexts, such as risk management [25], portfolio optimization [88], and optimal stopping [17], as well as data-driven methods to price options [65]. Given their potential, various deep learning-based PDE solvers have been proposed [45], including Backward Stochastic Differential Equation (BSDE) methods [47], DGMs [81] and TDGF [41, 72]. However, their empirical accuracy remains not sufficiently well understood, which limits their practical adoption in financial applications.

This chapter focuses on two neural network methods for solving PDEs: the DGM and the TDGF. Our primary objective is to assess their empirical accuracy. For theoretical convergence analyses, see the work of Jiang et al. [60] for DGM and Liu, Papapantoleon, and Rou [64] for TDGF. For empirical studies of other deep PDE solvers, such as BSDE-based methods, see the work of Assabumrungrat et al. [7].

To provide actionable insights into the applicability of deep PDE solvers for option pricing, we systematically analyze the impact of key parameters on accuracy and training time. First, we investigate the effect of training by varying the number of sampling stages and the number of samples. Second, we investigate the effect of the size of the neural network by varying the number of layers and the number of nodes per layer. For a comparison of different architectures, see the work of Van Mieghem, Papapantoleon, and Papazoglou-Hennig [84]. Finally, for TDGF, we also examine the discretization order and the number of time steps.

Our main findings are: the L^2-error decreases almost linearly with the number of sampling stages; the number of layers tends to decrease the error, but not with a clear rate; and increasing the number of time steps decreases the L^2-error with the second-order method decreasing quicker than the first-order method. These three parameters increase the training time linearly. The number of samples and nodes per layer did not show a clear relationship with neither the L^2-error nor the training time. The first three parameters concern training stages done sequentially, while the other two concern training stages which can be done in parallel.

The remainder of this chapter is structured as follows. Section 6.2 introduces the two neural network-based PDE solvers. Section 6.3 outlines the option pricing models under consideration: Black–Scholes and Heston. Section 6.4 details the implementation aspects. Section 6.5 presents the numerical results for each of the five parameters. Finally, Section 6.6 summarizes our findings and conclusions.

6.2 Neural network methods

This section explains the two neural network methods used in this chapter. Section 6.2.1 elaborates on the TDGF and Section 6.2.2 on the DGM.

6.2.1 Time Deep Gradient Flow method

The TDGF is a neural network method to efficiently solve high-dimensional PDEs [41, 42, 72]. Consider the general PDE

$$\frac{\partial}{\partial t}u(t,\mathbf{x}) + \mathcal{A}u(t,\mathbf{x}) + ru(t,\mathbf{x}) = 0, \quad (t,\mathbf{x}) \in [0,T]\times\Omega,$$
$$u(0,\mathbf{x}) = \Psi(\mathbf{x}), \quad \mathbf{x}\in\Omega,$$

with $\mathcal{A}$ a second-order differential operator of the form

$$\mathcal{A}u = -\sum_{i,j=1}^{d} a^{ij}\frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^{d}\beta^i \frac{\partial u}{\partial x_i}.$$

Using the splitting method from Chapter 3, $\mathcal{A}$ can be rewritten in the form

$$\mathcal{A}u = -\nabla\cdot(A\nabla u) + \mathbf{b}\cdot\nabla u, \tag{6.1}$$

with a symmetric an positive semidefinite matrix

$$A = \begin{bmatrix} a^{11} & a^{21} & \dots & a^{d1} \\ a^{21} & a^{22} & \dots & a^{d2} \\ \vdots & \vdots & \ddots & \vdots \\ a^{1d} & a^{2d} & \dots & a^{dd} \end{bmatrix} \quad \text{and vector} \quad \mathbf{b} = \begin{bmatrix} b^1 \\ b^2 \\ \vdots \\ b^d \end{bmatrix}. \tag{6.2}$$

Let us divide the time interval $(0,T]$ into K equally spaced intervals $(t_{k-1}, t_k]$, with $h = t_k - t_{k-1} = \frac{1}{K}$ for $k = 0, 1, \dots, K$. Let U^k denote the approximation to the solution of the PDE $u(t_k,\mathbf{x})$ at time step t_k, using either a first- or second-order discretization scheme [5]

$$\frac{U^k - U^{k-1}}{h} - \nabla\cdot\left(A\nabla U^k\right) + F\left(U^{k-1}\right) + rU^k = 0,$$
$$\frac{\frac{3}{2}U^k - 2U^{k-2} + \frac{1}{2}U^{k-1}}{h} - \nabla\cdot\left(A\nabla U^k\right) + 2F\left(U^{k-1}\right) - F\left(U^{k-2}\right) + rU^k = 0,$$

with $F(u) = \mathbf{b}\cdot\nabla u$, $U^0 = \Psi$ and in the second-order scheme we take U^1 from the first-order scheme. Then we can rewrite the discretized PDE as an energy functional [42, 72]

$$U^k = \underset{u\in\mathcal{H}^1}{\arg\min}\, I^k(u),$$

with the energy functionals

$$\begin{aligned} I_1^k(u) =& \frac{1}{2}\left\|u - U^{k-1}\right\|^2_{L^2(\Omega)} + h\left(\int_\Omega \frac{1}{2}\left((\nabla u)^\mathsf{T} A\nabla u + ru^2\right) + F\left(U^{k-1}\right)u\mathrm{d}x\right), \\ I_2^k(u) =& \frac{1}{2}\left\|u - \frac{4}{3}U^{k-1} + \frac{1}{3}U^{k-2}\right\|^2_{L^2(\Omega)} \\ &+ \frac{2h}{3}\left(\int_\Omega \frac{1}{2}\left((\nabla u)^\mathsf{T} A\nabla u + ru^2\right) + \left(2F\left(U^{k-1}\right) - F\left(U^{k-2}\right)\right)u\mathrm{d}x\right), \end{aligned} \tag{6.3}$$

for the first- and second-order discretization respectively. Let $f^k(\mathbf{x};\theta)$ denote a neural network approximation of U^k with trainable parameters θ. Applying a Monte Carlo approximation to the integrals, the discretized cost functional takes the form

$$L_n^k(\theta;\mathbf{x}) = \frac{|\Omega|}{2M}\sum_{m=1}^{M}\left(f^k(\mathbf{x}_m;\theta) + \sum_{j=1}^{n}\alpha_n^j f^{k-j}(\mathbf{x}_m)\right)^2 + \beta_n h N_n^k(\theta;\mathbf{x}), \tag{6.4}$$

with

$$\begin{aligned} N_n^k(\theta;\mathbf{x}) =& \frac{|\Omega|}{M}\sum_{m=1}^{M}\left[\frac{1}{2}\left(\left(\nabla f^k(\mathbf{x}_m;\theta)\right)^{\mathsf{T}} A\nabla f^k(\mathbf{x}_m;\theta) + r\left(f^k(\mathbf{x}_m;\theta)\right)^2\right)\right. \\ &\left. + \left(\mathbf{b}\cdot\sum_{j=1}^{n}\gamma_n^j\nabla f^{k-j}(\mathbf{x}_m)\right) f^k(\mathbf{x}_m;\theta)\right]. \end{aligned}$$

Here, M denotes the number of samples $\mathbf{x}_m$; $n \in \{1,2\}$ the order of the discretization and α_n^j, β_n and γ_n^j are the corresponding coefficients.

In order to minimize this cost function, we use a stochastic gradient descent-type algorithm, *i.e.* an iterative scheme of the form:

$$\theta_{n+1} = \theta_n - \alpha\nabla_\theta L^k(\theta_n;\mathbf{x}). \tag{6.5}$$

The hyperparameter α is the step size of our update, called the learning rate. An overview of the TDGF method appears in Algorithm 1 in Chapter 3.

6.2.2 Deep Galerkin Method

We compare the TDGF method with a popular deep learning method for solving PDEs: the DGM of Sirignano and Spiliopoulos [81]. In the DGM approach, we minimize the square L^2-error of the PDE:

$$\left\|\frac{\partial u}{\partial t} - \nabla\cdot(A\nabla u) + \mathbf{b}\cdot\nabla u + ru\right\|^2_{L^2([0,T]\times\Omega)} + \|u(0,x) - \Psi(x)\|^2_{L^2(\Omega)}.$$

Then the cost functional for the neural network approximation $f(t,\mathbf{x};\theta)$ of u, takes the form

$$\begin{aligned} L(\theta;t,\mathbf{x}) =& \frac{T|\Omega|}{M_1}\sum_{m=1}^{M_1}\left[\partial_t f(t,\mathbf{x}_m;\theta) - \nabla\cdot(A\nabla f(t,\mathbf{x}_m;\theta)) + \mathbf{b}\cdot\nabla f(t,\mathbf{x}_m;\theta)\right. \\ &\left. + rf(t,\mathbf{x}_m;\theta)\right]^2 + \frac{|\Omega|}{M_2}\sum_{m=1}^{M_2}\left[f(0,\mathbf{x}_m;\theta) - \Psi(\mathbf{x}_m)\right]^2. \end{aligned}$$

The solution of the PDE is approximated by a neural network using stochastic gradient descent as in equation (6.5). Contrary to the TDGF there is no time stepping. Instead of training a neural network for each time step, there is one neural network with time as input parameter. An overview of the DGM appears in Algorithm 3.

Algorithm 3 Deep Galerkin Method

1: Initialize θ_0.
2: **for** each sampling stage $n = 1, ..., N$ **do**
3: Generate M random points $(t_m, \mathbf{x}_m)$ for training.
4: Calculate the cost functional $L(\theta_n; t, \mathbf{x})$ for the selected points.
5: Take a descent step $\theta_{n+1} = \theta_n - \alpha \nabla_\theta L(\theta_n; t, \mathbf{x})$.
6: **end for**

6.3 Option pricing models

This section explains the two option pricing models in which we solve the pricing PDE. Section 6.3.1 elaborates on the Black–Scholes model and Section 6.3.2 on the Heston model.

6.3.1 Black–Scholes model

In Section 3.4.1 we derived that the operator $\mathcal{A}$ in the Black–Scholes model takes the form (6.1) with the coefficients in (6.2) provided by

$$\begin{aligned} a &= \frac{1}{2}\sigma^2 S^2, \\ b &= (\sigma^2 - r)S, \end{aligned}$$

with $r, \sigma \in \mathbb{R}_+$ the risk-free rate and the volatility respectively.

6.3.2 Heston model

In Section 3.4.2 we derived that the operator $\mathcal{A}$ takes the form (6.1) with the coefficients in (6.2) provided by

$$\begin{aligned} a^{11} &= \frac{1}{2}S^2 V, \\ a^{21} = a^{12} &= \frac{1}{2}\rho\eta S V, \\ a^{22} &= \frac{1}{2}\eta^2 V, \\ b^1 &= \left(-r + V + \frac{1}{2}\rho\eta\right) S, \\ b^2 &= \lambda(V - \kappa) + \frac{1}{2}\eta^2 + \frac{1}{2}\rho\eta V, \end{aligned}$$

Here ρ is the correlation coefficient between the Brownian motions driving S and V and $\lambda, \kappa, \eta \in \mathbb{R}_+$.

6.4 Implementation details

For the architecture we use the architecture in (3.7) including the use of information about the option price in order to facilitate the training of the neural network. We

use as activation functions the hyperbolic tangent function, $\sigma_1(x) = \tanh(x)$, and the softplus function, $\sigma_2(x) = \log(e^x + 1)$, which guarantees that the option price remains above the no-arbitrage bound.

We consider the effect of five parameters on the error: the number of sampling stages, N in Algorithm 1; the number of samples, M in equation (6.4); the number of layers; the number of nodes per layer; and for the TDGF also the number of time steps, K in Algorithm 1. In the last case we consider both the first- and second-order discretization scheme in equation (6.3). As the default parameter set we take 600 samples per dimension in each sampling stage. To obtain the total number of samples from this number, multiply by 2 for DGM in the Black–Scholes (time and stock price) and TDGF in the Heston model (stock price and volatility) and multiply by 3 for the DGM in the Heston model (time, stock price and volatility). The default network size is 3 layers and 50 nodes per layers. For the TDGF we take 100 time steps and 500 sampling stages in each time step and for the DGM we take 100,000 sampling stages. After this many sampling stages the error does not decrease further.

For both DGM and TDGF we use the Adam optimizer [61] with a learning rate $\alpha = 3 \times 10^{-4}$, $(\beta_1, \beta_2) = (0.9, 0.999)$ and zero weight decay. The training is performed on the DelftBlue supercomputer [29], using one seventh instance of a NVidia Tesla A100 GPU. We run each problem for five different random seed and compare the average error of the five runs.

As the modeling problem we take the price of a European call option with interest rate $r = 0.05$ and maturity $T = 1.0$ year. We consider the Black–Scholes model with volatility $\sigma = 0.25$ and the Heston model with $\eta = 0.1$, $\rho = 0.0$, $\kappa = 0.01$ and $\lambda = 2.0$. For the domain Ω we consider $S \in [0.01, 3.0]$ and $V \in [0.001, 0.1]$. The solution of the Black–Scholes PDE with these parameters together with the solution produced by the TDGF with the default training parameters is in Fig. 6.1.

6.5 Numerical results

In the next subsections we vary one of the parameters while keeping the others constant at the default value. We compute the L^2-error on an equidistant grid of 47 points in each dimension on the domain.

6.5.1 Sampling stages

First, we consider the number of sampling stages. For the TDGF we vary the number of sampling stages per time step from 16 to 500. For the DGM we vary the number of sampling stages from 2048 to 100,000. After this many sampling stages, the error does not decrease anymore in the Black–Scholes model. In the Heston model, the error seems to quit decreasing quicker for both methods. The fitted convergence rates for both methods and both models are in Table 6.1. All convergence rates are slightly larger than -1. The plots of the L^2-error on linear and log scale are in Figs. 6.2 to 6.5 together with the training time. The training time increases linearly with the number of sampling stages.

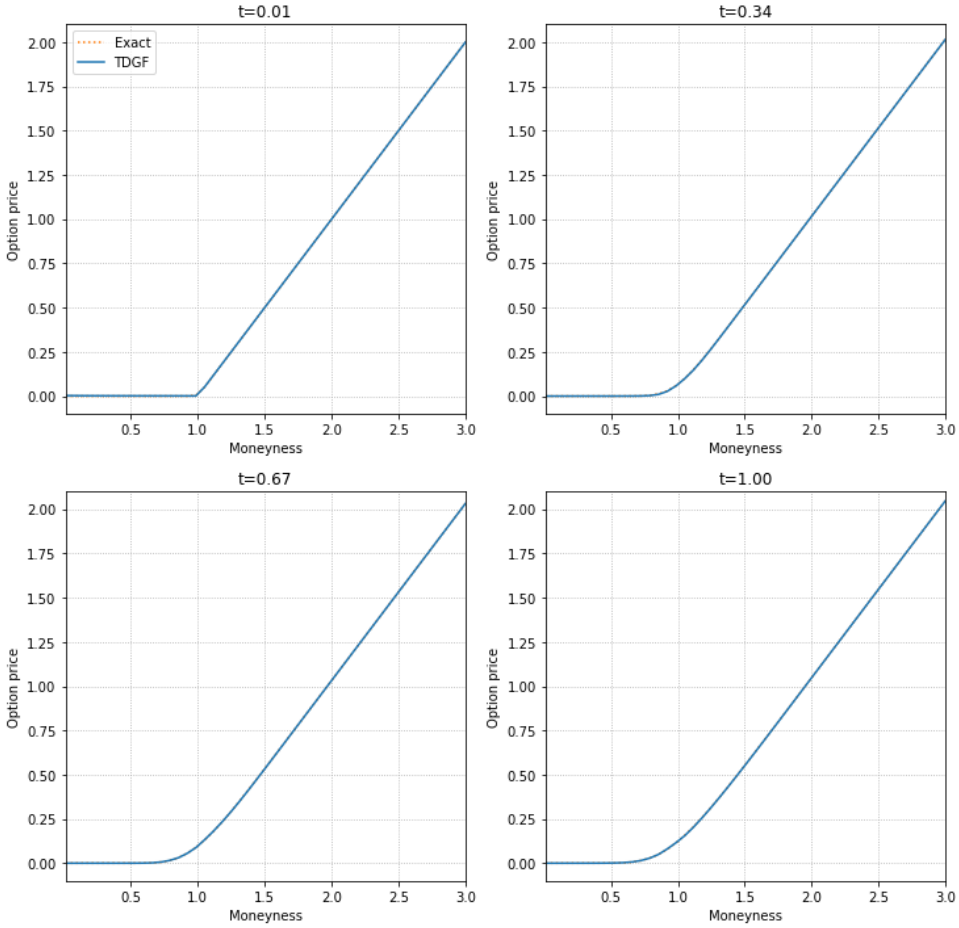


Figure 6.1: Exact price of European call option with $r = 0.05$, $\sigma = 0.25$ and $T = 1.0$ compared to the price of computed by the TDGF.

Method	Black–Scholes	Heston
TDGF	-0.91	-0.63
DGM	-0.73	-0.75

Table 6.1: Convergence rates for the number of sampling stages.

Method	Black–Scholes	Heston
TDGF	-0.27	-0.11
DGM	-0.12	0.2

Table 6.2: Convergence rates for the number of samples.

Method	Black–Scholes	Heston
TDGF	-0.63	-0.47
DGM	-0.33	-0.70

Table 6.3: Convergence rates for the number of layers.

6.5.2 Samples

Second, we consider the number of samples per dimension in each sampling stage. We vary the number of samples per dimension from 16 to 600. The fitted convergence rates for both methods and both models are in Table 6.2. In general, it is hard to draw conclusions. For the TDGF the rates are slightly negative, but far from -0.5, which would be the expected rate of convergence for Monte Carlo sampling. For the DGM the rates are larger and the error does not decrease as uniformly with the number of samples as for the TDGF. The plots of the L^2-error on linear and log scale are in Figs. 6.6 to 6.9 together with the training time. The number of samples does not have a big impact on the training time.

6.5.3 Layers

Third, we consider the number of layers of the neural network. We vary the number for layers from 1 to 4. The fitted convergence rates for both methods and both models are in Table 6.3. The rates vary but are all negative so, in general, more layers improves the result. The plots of the L^2-error on linear and log scale are in Figs. 6.10 to 6.13 together with the training time. The training time increases linearly with the number of layers.

6.5.4 Nodes per layer

Fourth, we consider the number of nodes per layers of the neural network. We vary the number of layers from 10 to 50. The fitted convergence rates for both methods and both models are in Table 6.4. The rates vary across different methods and models and are even positive for the DGM. The plots of the L^2-error on linear and log scale are in Figs. 6.14 to 6.17 together with the training time. The number of nodes per layers does not have a big impact on the training time.

6.5.5 Time steps

Fifth and final, we consider the number of time steps. We vary the number of time steps from 2 to 25. The fitted convergence rates for both models and for both first- and second-order time stepping are in Table 6.5. After 25 time steps,

Method	Black–Scholes	Heston
TDGF	-1.11	-0.39
DGM	0.15	0.07

Table 6.4: Convergence rates for the number of nodes per layer.

Method	Black–Scholes	Heston
$O(1)$	-0.29	-0.15
$O(2)$	-0.56	-0.25

Table 6.5: Convergence rates for the number of time steps

the second-order scheme does not improve any further, but the first-order scheme does. The rates for the Black–Scholes are lower than for Heston. In both cases $O(2)$ outperforms $O(1)$. The plots of the L^2-error on linear and log scale are in Figures Figs. 6.18 and 6.19 together with the training time. The training time grows linearly with the number of time steps with the second-order method growing faster than the first-order method.

6.6 Conclusion

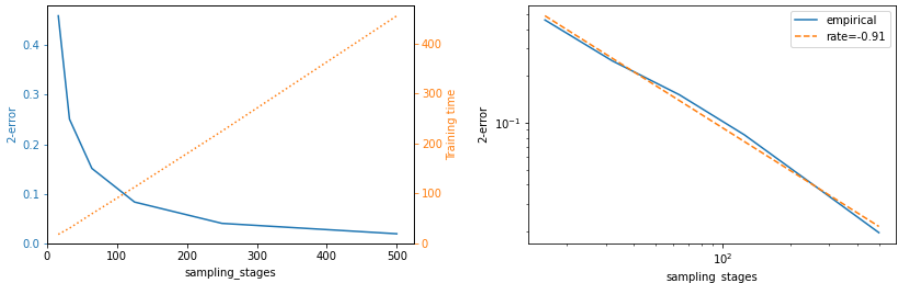
This research analyzed the error of two neural network methods to solve option pricing PDEs: TDGF and DGM. We determined the empirical convergence rates of the L^2-error of five parameters in both the Black–Scholes and the Heston model. We also considered the effect of these parameters on the training time. Based on the experiments we can give some recommendations that can assist anyone who want to use the methods in a practical setting.

- For both the TDGF and the DGM, the L^2-error decreases almost linearly with the number of sampling stages up to some point where it stops converging. Since the training time grows linearly with the the number of sampling stages, it would be optimal to stop at this point. Unfortunately, there is no method to locate this point beforehand and we recommend choosing the number of sampling stages based on whether speed or accuracy is more important in the practical setting.
- For the TDGF the L^2-error decreases slightly with the number of samples. Since the number of samples does not influence the training time, we recommend using a large number of samples like six hundred per dimension or even more.
- For the DGM the L^2-error does not decrease with the number of samples. Therefore, it is hard to give any recommendation.
- For both TDGF and DGM, the number of layers tends to decrease the error, but not with a clear rate. One layer is clearly not enough, but four layers does not improve the results compared to two or three layers. Since the number of

layers has a big influence on the training time, we recommend using two or three layers.

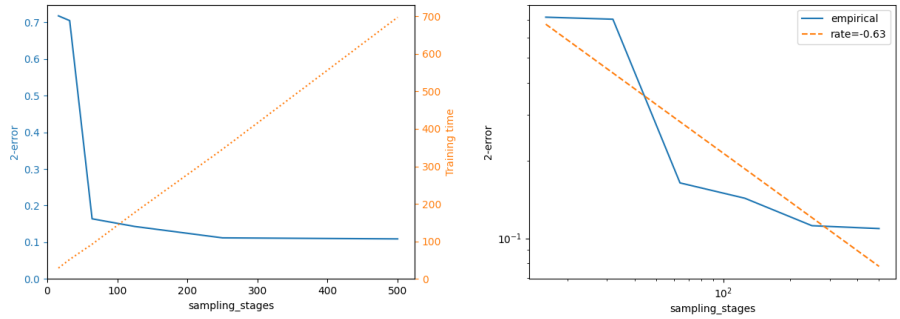
- For the TDGF, the L^2-error decreases with the number of nodes per layer. Since the number of nodes per layer does not influence the training time, we recommend choosing a large number of nodes per layer like forty of fifty.
- For the DGM, the L^2-error does not decrease with the number of nodes per layer. We recommend choosing a smaller number of nodes per layer like thirty.
- For the TDGF, the number of time steps decreases the L^2-error. Using a second-order time stepping method, the error decreases quicker than using a first-order method. We recommend using the second-order time stepping method. Since the training time increases linearly with the number of time steps, we again recommend choosing the number of time steps based on whether speed or accuracy is more important in the practical setting.

6.A Error plots



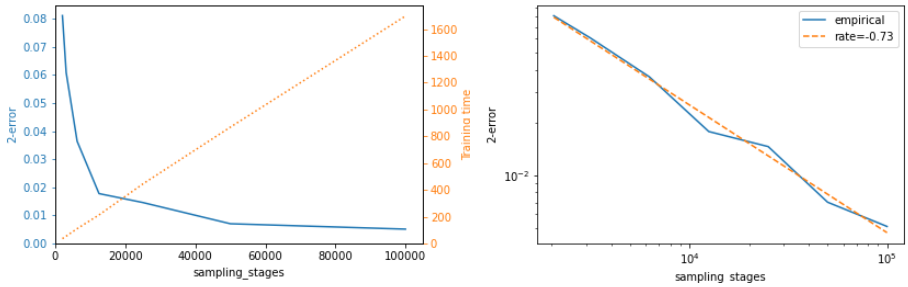
(a) Linear scale (b) Logarithmic scale

Figure 6.2: L^2-error of the TDGF for a call option in the Black–Scholes model against number of sampling stages varying from 16 to 500.



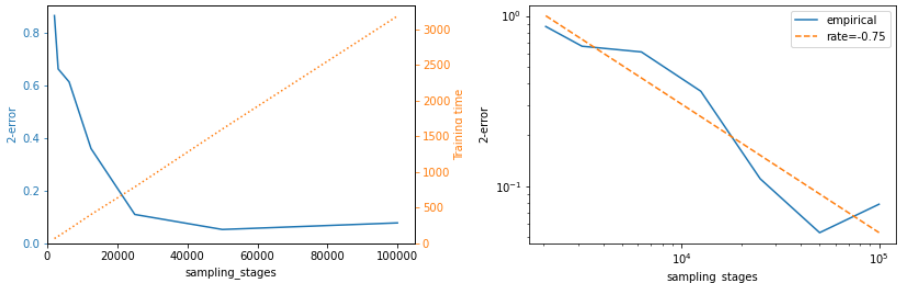
(a) Linear scale (b) Logarithmic scale

Figure 6.3: L^2-error of the TDGF for a call option in the Heston model against number of sampling stages varying from 16 to 500.



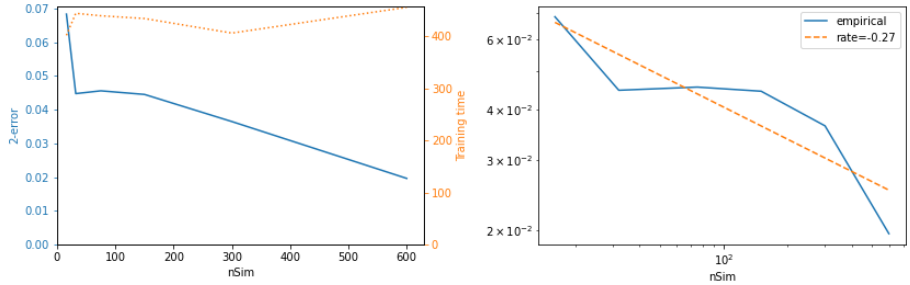
(a) Linear scale (b) Logarithmic scale

Figure 6.4: L^2-error of the DGM for a call option in the Black–Scholes model against number of sampling stages varying from 2048 to 100,000.



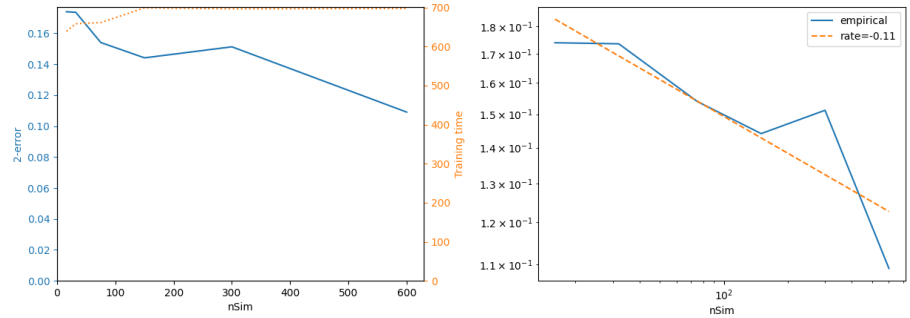
(a) Linear scale (b) Logarithmic scale

Figure 6.5: L^2-error of the DGM for a call option in the Heston model against number of sampling stages varying from 2048 to 100,000.



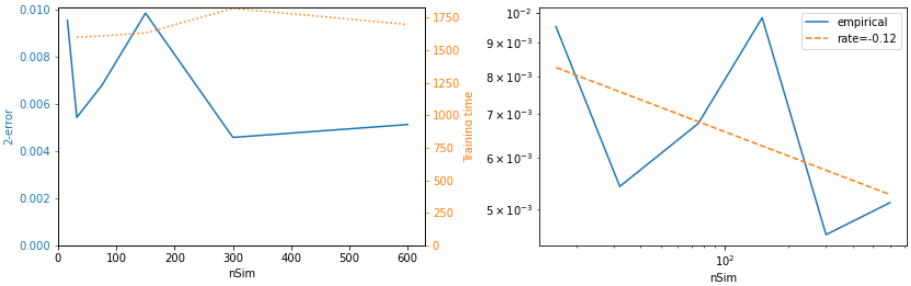
(a) Linear scale (b) Logarithmic scale

Figure 6.6: L^2-error of the TDGF for a call option in the Black–Scholes model against number of samples varying from 16 to 600.



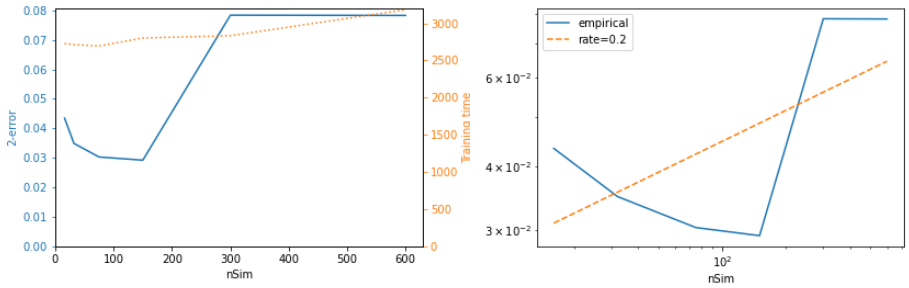
(a) Linear scale (b) Logarithmic scale

Figure 6.7: L^2-error of the TDGF for a call option in the Heston model against number of samples varying from 16 to 600.



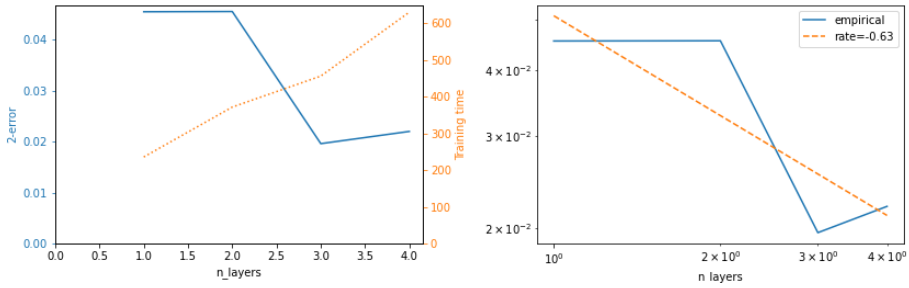
(a) Linear scale (b) Logarithmic scale

Figure 6.8: L^2-error of the DGM for a call option in the Black–Scholes model against number of samples varying from 16 to 600.



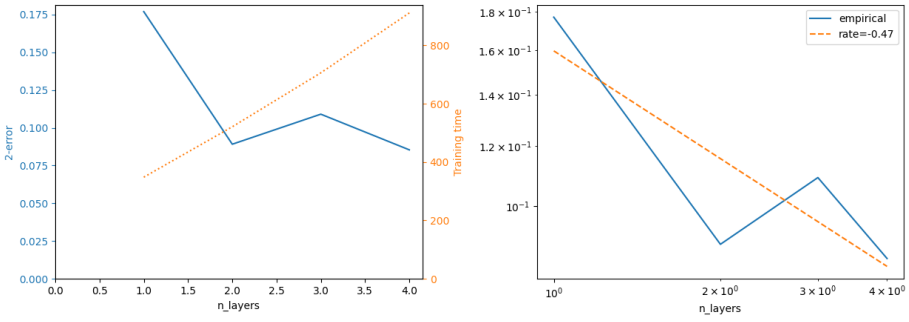
(a) Linear scale (b) Logarithmic scale

Figure 6.9: L^2-error of the DGM for a call option in the Heston model against number of samples varying from 16 to 600.



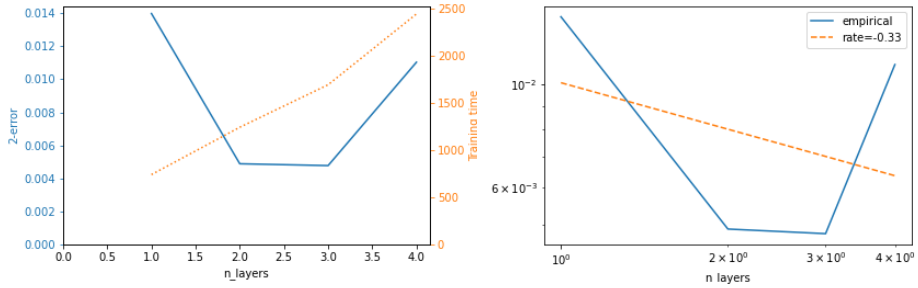
(a) Linear scale (b) Logarithmic scale

Figure 6.10: L^2-error of the TDGF for a call option in the Black–Scholes model against number of layers varying from 1 to 4.



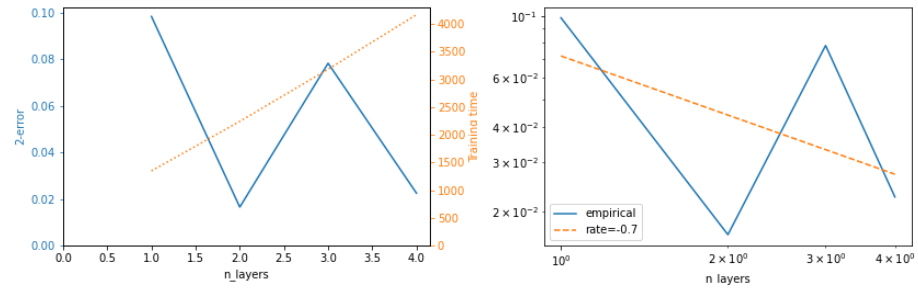
(a) Linear scale (b) Logarithmic scale

Figure 6.11: L^2-error of the TDGF for a call option in the Heston model against number of layers varying from 1 to 4.



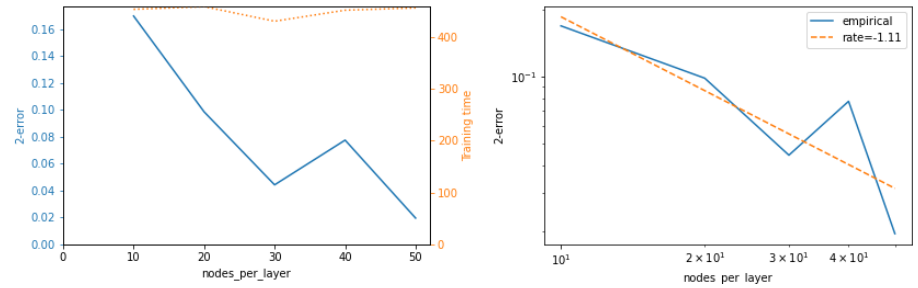
(a) Linear scale (b) Logarithmic scale

Figure 6.12: L^2-error of the DGM for a call option in the Black–Scholes model against number of layers varying from 1 to 4.



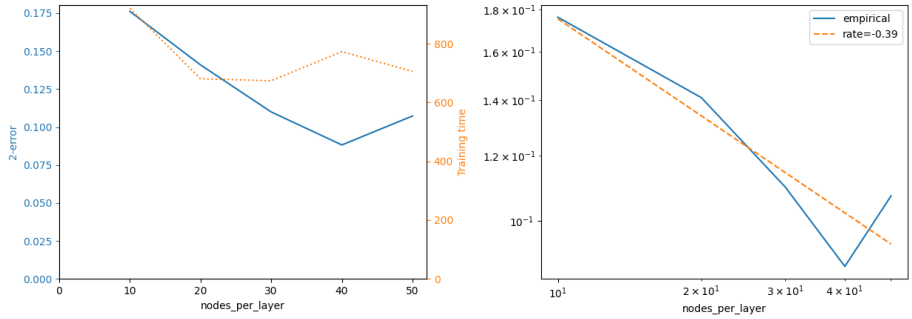
(a) Linear scale (b) Logarithmic scale

Figure 6.13: L^2-error of the DGM for a call option in the Heston model against number of layers varying from 1 to 4.



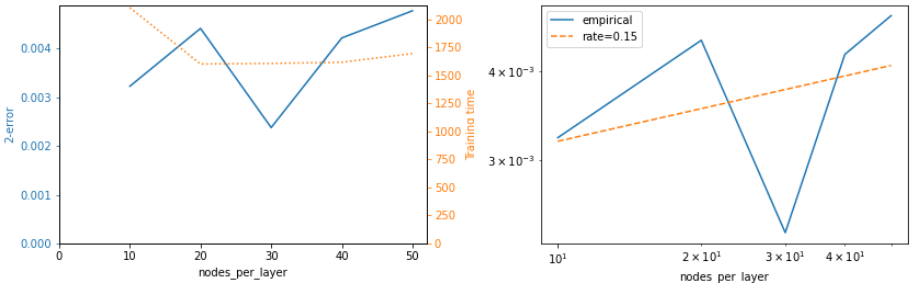
(a) Linear scale (b) Logarithmic scale

Figure 6.14: L^2-error of the TDGF for a call option in the Black–Scholes model against number of nodes per layer varying from 10 to 50.



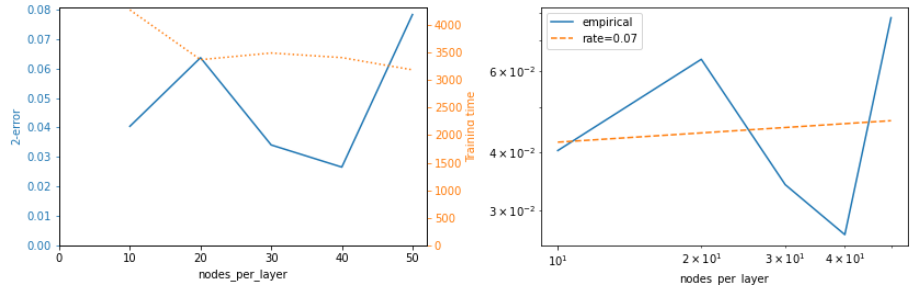
(a) Linear scale (b) Logarithmic scale

Figure 6.15: L^2-error of the TDGF for a call option in the Heston model against number of nodes per layer varying from 10 to 50.



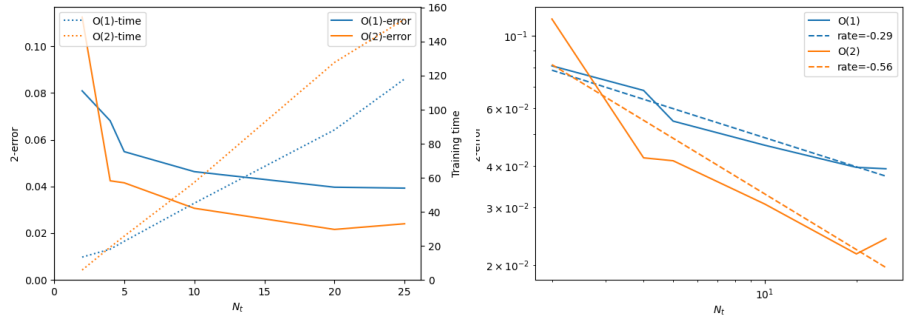
(a) Linear scale (b) Logarithmic scale

Figure 6.16: L^2-error of the DGM for a call option in the Black–Scholes model against number of nodes per layer varying from 10 to 50.



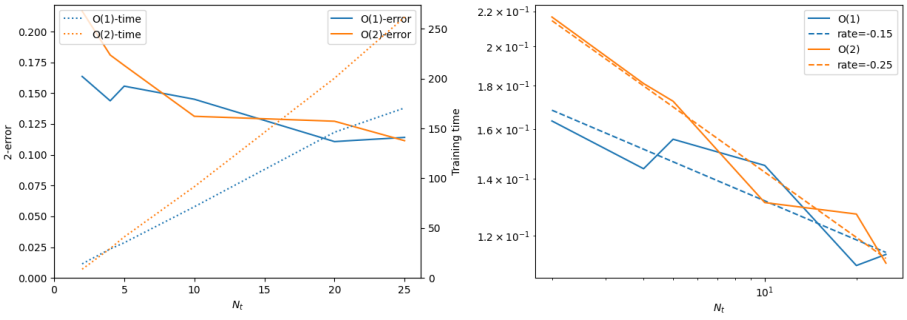
(a) Linear scale (b) Logarithmic scale

Figure 6.17: L^2-error of the DGM for a call option in the Heston model against number of nodes per layer varying from 10 to 50.



(a) Linear scale (b) Logarithmic scale

Figure 6.18: L^2-error of the TDGF for a call option in the Black–Scholes model against number of time steps varying from 2 to 25.



(a) Linear scale (b) Logarithmic scale

Figure 6.19: L^2-error of the TDGF for a call option in the Heston model against number of time steps varying from 2 to 25.

Bibliography

[1] E. Abi Jaber. Lifting the Heston model. *Quantitative Finance*, 19(12):1995–2013, 2019.

[2] E. Abi Jaber and O. El Euch. Markovian structure of the Volterra Heston model. *Statistics & Probability Letters*, 149:63–72, 2019.

[3] E. Abi Jaber and O. El Euch. Multifactor approximation of rough volatility models. *SIAM Journal on Financial Mathematics*, 10(2):309–349, 2019.

[4] E. Abi Jaber and S. Li. Volatility models in practice: Rough, path-dependent, or markovian? *Mathematical Finance*, 2025.

[5] G. Akrivis and Y.-S. Smyrlis. Implicit-explicit bdf methods for the kuramoto–sivashinsky equation. *Applied numerical mathematics*, 51(2-3):151–169, 2004.

[6] G. Akrivis, M. Crouzeix, and C. Makridakis. Implicit-explicit multistep methods for quasilinear parabolic equations. *Numerische Mathematik*, 82:521–541, 1999.

[7] R. Assabumrungrat, K. Minami, and M. Hirano. Error analysis of option pricing via deep PDE solvers: Empirical study. In *2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 329–336. IEEE, 2024.

[8] L. Bachelier. Théorie de la spéculation. *Annales scientifiques de l'École normale supérieure*, 3(17):21–86, 1900.

[9] C. Bayer and S. Breneis. Markovian approximations of stochastic Volterra equations with the fractional kernel. *Quantitative Finance*, 23(1):53–70, 2023.

[10] C. Bayer and S. Breneis. Weak Markovian approximations of rough Heston. Preprint arXiv:2309.07023, 2023.

[11] C. Bayer and S. Breneis. Efficient option pricing in the rough Heston model using weak simulation schemes. *Quantitative Finance*, 24(9):1247–1261, 2024.

[12] C. Bayer, P. Friz, and J. Gatheral. Pricing under rough volatility. *Quantitative Finance*, 16(6):887–904, 2016.

[13] C. Bayer, P. Friz, A. Gulisashvili, B. Horvath, and B. Stemper. Short-time near-the-money skew in rough fractional volatility models. *Quantitative Finance*, 19: 779–798, 2019.

[14] C. Bayer, C. Ben Hammouda, and R. Tempone. Hierarchical adaptive sparse grids and quasi-Monte Carlo for option pricing under the rough Bergomi model. *Quantitative Finance*, 20:1457–1473, 2020.

[15] C. Bayer, P. Friz, P. Gassiat, J. Martin, and B. Stemper. A regularity structure for rough volatility. *Mathematical Finance*, 30:782–832, 2020.

[16] C. Bayer, P. Friz, M. Fukasawa, J. Gatheral, A. Jacquier, and M. Rosenbaum. *Rough Volatility*. Society for Industrial and Applied Mathematics, 2023.

[17] S. Becker, P. Cheridito, and A. Jentzen. Deep optimal stopping. *Journal of Machine Learning Research*, 20(74):1–25, 2019.

[18] S. Becker, P. Cheridito, and A. Jentzen. Pricing and hedging American-style options with deep learning. *Journal of Risk and Financial Management*, 13(7): 158, 2020.

[19] S. Becker, P. Cheridito, A. Jentzen, and T. Welti. Solving high-dimensional optimal stopping problems using deep learning. *European Journal of Applied Mathematics*, 32(3):470–514, 2021.

[20] D. Belomestny and J. Schoenmakers. *Advanced Simulation-Based Methods for Optimal Stopping and Control: With Applications in Finance*. Springer, 2018.

[21] M. Bennedsen, A. Lunde, and M. Pakkanen. Hybrid scheme for Brownian semistationary processes. *Finance and Stochastics*, 21:931–965, 2017.

[22] L. Bergomi. *Stochastic Volatility Modeling*. CRC Press, Boca Raton, FL, 2016.

[23] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[24] O. Bonesini, G. Callegaro, M. Grasselli, and G. Pagès. From elephant to goldfish (and back): memory in stochastic Volterra processes. Preprint, arXiv:2306.02708, 2023.

[25] H. Buehler, L. Gonon, J. Teichmann, and B. Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

[26] N. Clarke and K. Parrott. Multigrid for American option pricing with stochastic volatility. *Applied Mathematical Finance*, 6(3):177–195, 1999.

[27] J. C. Cox, S. A. Ross, and M. Rubinstein. Option pricing: A simplified approach. *Journal of financial Economics*, 7(3):229–263, 1979.

[28] C. Cuchiero and J. Teichmann. Generalized Feller processes and Markovian lifts of stochastic Volterra processes: the affine case. *Journal of Evolution Equations*, 20(4):1301–1348, 2020.

[29] Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 2). `https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2`, 2024.

[30] B. Düring and M. Fournié. High-order compact finite difference scheme for option pricing in stochastic volatility models. *Journal of Computational and Applied Mathematics*, 236:4462–4473, 2012.

[31] W. E and B. Yu. The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6:1–12, 2018.

[32] W. E and B. Yu. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6:1–12, 2018.

[33] O. El Euch and M. Rosenbaum. The characteristic function of rough Heston models. *Mathematical Finance*, 29:3–38, 2019.

[34] F. Fang and C. Oosterlee. A novel pricing method for European options based on Fourier-cosine series expansions. *SIAM Journal on Scientific Computing*, 31(2):826–848, 2009.

[35] D. Filipović. Time-inhomogeneous affine processes. *Stochastic Processes and their Applications*, 115(4):639–659, 2005.

[36] M. Fukasawa and A. Hirano. Refinement by reducing and reusing random numbers of the hybrid scheme for Brownian semistationary processes. *Quantitative Finance*, 21:1127–1146, 2021.

[37] C. Gardiner. *Handbook of stochastic methods*, volume 3. springer Berlin, 1985.

[38] J. Gatheral. *The Volatility Surface: A Practitioner's Guide*. Wiley, 2012.

[39] J. Gatheral. Efficient simulation of affine forward variance models. *Risk*, February, 2022.

[40] J. Gatheral, T. Jaisson, and M. Rosenbaum. Volatility is rough. *Quantitative finance*, 18(6):933–949, 2018.

[41] E. Georgoulis, M. Loulakis, and A. Tsiourvas. Discrete gradient flow approximations of high dimensional evolution partial differential equations via deep neural networks. *Communications in Nonlinear Science and Numerical Simulation*, 117:106893, 2023.

[42] E. Georgoulis, A. Papapantoleon, and C. Smaragdakis. A deep implicit-explicit minimizing movement method for option pricing in jump-diffusion models. Preprint, arXiv:2401.06740, 2024.

[43] K. Glau and L. Wunderlich. The deep parametric pde method and applications to option pricing. *Applied Mathematics and Computation*, 432:127355, 2022.

[44] A. Gnoatto, A. Picarelli, and C. Reisinger. Deep xVA solver: A neural network–based counterparty credit risk management framework. *SIAM Journal on Financial Mathematics*, 14(1):314–352, 2023.

[45] L. Gonon, A. Jentzen, B. Kuckuck, S. Liang, A. Riekert, and P. von Wurstemberger. An overview on machine learning methods for partial differential equations: from physics informed neural networks to deep operator learning. Preprint, arXiv:2408.13222, 2024.

[46] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[47] J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[48] C. Herrera, F. Krach, P. Ruyssen, and J. Teichmann. Optimal stopping via randomized neural networks. *Frontiers of Mathematical Finance*, 3(1):31–77, 2024.

[49] S. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies*, 6 (2):327–343, 1993.

[50] N. Hilber, A.-M. Matache, and C. Schwab. Sparse wavelet methods for option pricing under stochastic volatility. *Journal of Computational Finance*, 8(4), 2005.

[51] N. Hilber, O. Reichmann, C. Schwab, and C. Winter. *Computational Methods for Quantitative Finance: Finite Element Methods for Derivative Pricing*. Springer Science & Business Media, 2013.

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

[53] B. Horvath, A. Jacquier, and C. Lacombe. Asymptotic behaviour of randomised fractional volatility models. *Journal of Applied Probability*, 56:496–523, 2019.

[54] B. Horvath, A. Muguruza, and M. Tomas. Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models. *Quantitative Finance*, 21:11–27, 2021.

[55] S. Ikonen and J. Toivanen. Efficient numerical methods for pricing American options under stochastic volatility. *Numerical Methods for Partial Differential Equations: An International Journal*, 24(1):104–126, 2008.

[56] A. Jacquier and M. Oumgari. Deep curve-dependent PDEs for affine rough volatility. *SIAM Journal on Financial Mathematics*, 14(2):353–382, 2023.

[57] A. Jacquier and A. Pannier. Large and moderate deviations for stochastic Volterra systems. *Stochastic Processes and their Applications*, 149:142–187, 2022.

[58] A. Jacquier and Z. Zuric. Random neural networks for rough volatility. Preprint, arXiv:2305.01035, 2023.

[59] A. Jentzen, B. Kuckuck, and P. von Wurstemberger. *Mathematical Introduction to Deep Learning: Methods, Implementations, and Theory*. (forthcoming), 2023.

[60] D. Jiang, J. Sirignano, and S. Cohen. Global convergence of deep galerkin and PINNs methods for solving partial differential equations. Preprint, arXiv:2305.06000, 2023.

[61] D. Kingma and J. Ba. Adam: A method for stochastic optimization. Preprint, arXiv:1412.6980, 2014.

[62] H.-H. Kuo. *Introduction to Stochastic Integration*. Springer-Verlag New York, 2006. doi: https://doi.org/10.1007/0-387-31057-6.

[63] Y. Liao and P. Ming. Deep Nitsche method: Deep Ritz method with essential boundary conditions. Preprint, arXiv:1912.01309, 2019.

[64] C. Liu, A. Papapantoleon, and J. Rou. Convergence of the generalization error for deep gradient flow methods for PDEs. To appear.

[65] S. Liu, C. Oosterlee, and S. Bohte. Pricing options and computing implied volatilities using neural networks. *Risks*, 7(1):16, 2019.

[66] F. Longstaff and E. Schwartz. Valuing American options by simulation: a simple least-squares approach. *The review of financial studies*, 14(1):113–147, 2001.

[67] R. McCrickerd and M. Pakkanen. Turbocharging Monte Carlo pricing for the rough Bergomi model. *Quantitative Finance*, 18:1877–1886, 2018.

[68] M. Musiela and M. Rutkowski. *Martingale methods in financial modelling*, volume 36. Springer Science & Business Media, 2006.

[69] R. Myneni. The pricing of the American option. *The Annals of Applied Probability*, pages 1–23, 1992.

[70] C. Nwankwo, N. Umeorah, T. Ware, and W. Dai. Deep learning and American options via free boundary framework. *Computational Economics*, 64(2):979–1022, 2024.

[71] C. Oosterlee and L. Grzelak. *Mathematical Modeling and Computation in Finance: With Exercises and Python and Matlab Computer Codes*. World Scientific, 2019.

[72] A. Papapantoleon and J. Rou. A time-stepping deep gradient flow method for option pricing in (rough) diffusion models. *Quantitative Finance*, pages 1–12, 2025.

[73] M. S. Park, C. Kim, H. Son, and H. J. Hwang. The deep minimizing movement scheme. *Journal of Computational Physics*, 494:112518, 2023.

[74] Y. Peng, P. Wei, and W. Wei. Deep penalty methods: A class of deep learning algorithms for solving high dimensional optimal stopping problems. arXiv preprint arXiv:2405.11392, 2024.

[75] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[76] C. Reisinger and G. Wittum. Efficient hierarchical approximation of high-dimensional option pricing problems. *SIAM Journal on Scientific Computing*, 29(1):440–458, 2007.

[77] S. E. Rømer. Empirical analysis of rough and classical stochastic volatility models to the SPX and VIX markets. *Quantitative Finance*, 22:1805–1838, 2022.

[78] J. Rou. Error analysis of deep PDE solvers for option pricing. Preprint arXiv:2505.05121, 2025.

[79] J. Rou. Time deep gradient flow method for pricing American options. Preprint arXiv:2507.17606, 2025.

[80] P. Samuelson. Rational theory of warrant pricing. *Industrial Management Review*, 6(2):13–31, 1965.

[81] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[82] J. Sirignano and K. Spiliopoulos. Asymptotics of reinforcement learning with neural networks. *Stochastic Systems*, 12(1):2–29, 2022.

[83] R. Van der Meer, C. Oosterlee, and A. Borovykh. Optimally weighted loss functions for solving PDEs with neural networks. *Journal of Computational and Applied Mathematics*, 405:113887, 2022.

[84] L. Van Mieghem, A. Papapantoleon, and J. Papazoglou-Hennig. Machine learning for option pricing: an empirical investigation of network architectures. arXiv preprint arXiv:2307.07657, 2023.

[85] J. van Neerven. *Functional Analysis*. Cambridge University Press, 2022.

[86] World Federation of Exchanges. FY 2023 market highlights. `https://www.world-exchanges.org/our-work/articles/fy-2023-market-highlights-report`, 2023.

[87] J. Zhang, T. He, S. Sra, and A. Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020.

[88] Z. Zhang, S. Zohren, and S. Roberts. Deep learning for portfolio optimization. arXiv preprint arXiv:2005.13665, 2020.

[89] Q. Zhu, G. Loeper, W. Chen, and N. Langrené. Markovian approximation of the rough Bergomi model for Monte Carlo option pricing. *Mathematics*, 9(5): 528, 2021.

Acknowledgements

First, I would like to express my thanks to my supervisor Antonis. You were always good at keeping an overview and I appreciate that you encouraged me to give many talks and made sure I was not teaching too much. I enjoyed the time we spent at conferences and during my visit to you in Athens. I am honored to be your first PhD here in Delft.

Second, I would like to thank Frank for being my second promotor. Even though you were not much involved in the scientific part, I appreciate your input during the progress meeting. Furthermore, you were always willing to listen and give advice when I needed it, which helped me a lot.

Third, I would like to thank Mark Veraar, Kees Oosterlee, Mathieu Rosenbaum and Christian Bayer for being in the committee. Many thanks for the feedback on my thesis and questions during the defence.

I would like to thank Chenguang for being my co-author. I learned a lot from working together and the paper that we wrote is much better than I could have done myself.

Of course I would also like to thank all my other colleagues. Thanks to Anne, Bart, Berend, Victor and most of all Jonathan for sharing an office. Thanks to Ardjen for helping me deciding on and obtaining my next job. Thanks to Balint and Gijs for the fun times we had at conferences. Thanks to Hidde, Jan-Tino, Marc, Serena, Stefan, Vicente Yago en Yanyan for the fun lunches and evenings we had together.

Last, I would like to thank my family and friends for their support. Thanks to Marcus for being paranimph. Thanks to Sterre for being paranimph, designing the cover and listening to my complaints about my job every day.

Curriculum Vitæ

Jasper Gijsbert ROU

16-10-1999	Born in Veenendaal, Netherlands.

Education

2011–2017	Secondary school, *cum laude* Christelijk Lyceum Veenendaal
2017–2020	Bachelor in Applied Mathematics & Applied Physics, *cum laude* Delft University of Technology
2020–2022	Master in Applied Mathematics, *cum laude* Delft University of Technology
2022–2025	PhD. Applied Mathematics Delft University of Technology
Thesis:	Time Deep Gradient Flow Method for Option Pricing
Promotor:	Prof. dr. A. Papapantoleon

List of Publications

- A. Papapantoleon and J. Rou. A time-stepping deep gradient flow method for option pricing in (rough) diffusion models. *Quantitative Finance*, pages 1–12, 2025.
- J. Rou. Error analysis of deep PDE solvers for option pricing. Preprint arXiv:2505.05121, 2025.
- J. Rou. Time deep gradient flow method for pricing American options. Preprint arXiv:2507.17606, 2025.
- C. Liu, A. Papapantoleon, and J. Rou. Convergence of the generalization error for deep gradient flow methods for PDEs. To appear.