

Analysis of Neural-network-based PDE-solving Algorithms



InFoMM
Industrially Focused
Mathematical Modelling

Deqing Jiang
University College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Hilary 2024

To my parents, Mrs. Liying Yao and Mr. Hua Jiang.

Acknowledgements

The road of doing mathematical research is winding with highs and lows. I am lucky to have this opportunity to do my research in our department, and I cherish those fabulous and cheerful moments I have had along this journey, and those hard moments which taught me a lesson.

I am truly grateful to my splendid supervisors, Prof. Samuel N. Cohen and Prof. Justin Sirignano, of course not only for their responsible supervision, but also for their charming characters, for how they cared and encouraged me, and for all the time and effort we have spent together. Without their unlimited warm support, I probably would have quit halfway through COVID time. They steered me to this final stop instead.

I would like to thank my CDT, my college, and the Alan Turing Institute for having generously funded my studies. InFoMM CDT also provided important training opportunities for me, which broadened my understanding of various topics and enhanced my capability to carry out my studies.

I have received valuable feedback and suggestions on our work from my transfer and confirmation examiners, audiences listening to my seminar presentations, my friends studying related subjects, and my peers in the department. I appreciate their kind help.

Finally, let me express my gratitude to everyone accompanying me in this rapidly changing world, without whom this thesis, as my own little miracle, may never come into reality.

Abstract

Neural networks are powerful function approximation tools in scientific computing. Thanks to recent advances in the modern electronic engineering industry, computational resources are more accessible than ever to train neural networks to perform various tasks. In recent years, algorithms that numerically solve partial differential equations (PDEs) using neural networks have become popular, as these methods can overcome the curse of dimensionality, which prevents classical methods from solving high-dimensional equations.

Despite their practical success, the mathematical properties of these algorithms have not yet been well-studied. Compared to other typical neural network training tasks, solving PDEs requires matching boundary conditions and PDE operators simultaneously. The unboundedness of PDE operators (in L^2) also makes the analysis even harder.

In this thesis, we aim to provide a rigorous mathematical analysis of related algorithms in several aspects. A new algorithm (Q-PDE) for solving PDEs is proposed, and a few numerical examples are given. The training process of the neural network approximator is characterized, for which we also prove that an infinite-dimensional ODE models the limiting behavior. A novel neural tangent kernel is described, which drives the ODE system. By applying proper functional analysis tools, we can show the convergence of neural networks to PDE solutions in the limit regime. In addition, we model the dynamics of policy iteration methods with neural networks that solve PDEs originating from stochastic control problems by reformulating them as a sequence of linear PDEs.

Contents

1	Introduction	1
1.1	Background	1
1.2	Deep-learning-based methods	4
1.2.1	Direct methods	5
1.2.2	Probabilistic methods	8
1.2.3	Learning alternative formulations	10
1.2.4	Learning operators	13
1.3	Mathematical tools	16
1.3.1	Neural networks as function approximators	16
1.3.2	Mean-field analysis of neural networks	19
1.3.3	Kernel methods	23
1.3.4	Analysis in Banach spaces and Hilbert spaces	25
1.4	Layout and main aims	26
1.5	Statement of originality	27
2	Q-PDE algorithm	28
2.1	The Q-PDE Algorithm	30
2.1.1	Assumptions on the PDE	31
2.1.2	Neural Network Configuration	34
2.1.3	Training Algorithm	36
2.1.3.1	Main results	38
2.1.3.2	Similarity to Q-learning	39
2.2	Preliminary analysis of the training algorithm	40
2.2.0.1	Dynamics of Q_t^N	40
2.2.0.2	Limit kernel	41
2.3	Convergence to the limit ODE	46
2.3.1	Characterizing the difference between kernels	47
2.3.1.1	Difference between kernel $B_t^N(x, y)$ and $B_0^N(x, y)$	47

2.3.1.2	Difference between $B_0^N(x, y)$ and $B(x, y)$	49
2.3.1.3	Difference between kernel $B_t^N(x, y)$ and $B(x, y)$. . .	50
2.3.2	Convergence of initial approximator Q_0^N to Q_0	50
2.3.3	Convergence of the approximator Q_t^N to Q_t	51
2.4	Convergence for large training time	55
2.5	Numerical experiments	59
2.5.1	Discretization of the continuous-time algorithm	59
2.5.2	Test equation I: Survival time of a Brownian motion	60
2.5.2.1	1-dimensional case	61
2.5.2.2	20-dimensional case	62
2.5.3	Test equation II: A nonconstant coefficient PDE	67
2.5.3.1	Monte Carlo solution	67
2.5.3.2	Neural network approximator configuration	68
2.5.3.3	Test outcome	68
2.6	Proofs and additional information	69
2.6.1	Proof of Lemma 2.1.3	69
2.6.2	Proof of Lemma 2.2.1	71
2.6.3	Proof of Lemma 2.3.1	73
2.6.4	Proof of Lemma 2.3.2	76
2.6.5	Proof of Lemma 2.3.4	76
2.6.6	Existence of ODE solutions	77
2.6.7	Monotonicity for linear elliptic PDE	78
2.6.8	Configuration of the numerical test cases	80
2.6.8.1	Table of hyperparameters	80
2.6.8.2	Initialization of neural networks	80
2.6.8.3	Learning process	81

3 Convergence analysis of deep Galerkin method (DGM) & Physics-informed neural networks (PINNs) 82

3.1	Introduction	82
3.2	Mathematical Framework	85
3.3	Deep Learning Algorithms for Solving PDEs	86
3.3.1	Convergence of the trained neural network as the number of hidden units $N \rightarrow \infty$	89
3.4	Analysis of the kernel function	90
3.5	Global convergence of the limit ODE as $t \rightarrow \infty$	94

3.6	Global convergence of PINNs algorithm	95
3.6.1	On related matters of DGM and PINNs	99
3.6.1.1	Convergence rates	99
3.6.1.2	Comparing our analysis with empirical numerical results	100
3.7	Proofs and additional information	101
3.7.1	Proof of Theorem 3.3.7	101
3.7.2	Exponential convergence of original PINNs objective	108
3.7.3	A proposed uniformly stable activation function	109
4	Actor-critic methods for Hamilton–Jacobi–Bellman PDEs	112
4.1	Actor-critic methods	114
4.2	Single-step update of the control with neural networks	116
4.2.1	Neural network control	117
4.2.2	Wide limit of the neural network control	118
4.2.3	Practical issues	122
4.3	Online actor-critic methods	123
4.3.1	Actor U_θ and critic Q_ϕ	124
4.3.2	Continuous-time online actor-critic method	125
4.3.2.1	Wide limit of the actor-critic pair	127
4.3.2.2	Open problems	129
4.3.3	Appendix: proof of Lemma 4.3.6	130
4.4	Numerical examples	137
4.4.1	Test case I: linear drift	138
4.4.1.1	Neural network training configuration	139
4.4.1.2	Test results	139
4.4.2	Test case II: cosine drift	140
4.4.2.1	Neural network training configuration	141
4.4.2.2	Test results	141
5	Conclusions & future works	143
	Bibliography	145

List of Figures

1.1	Example of a feedforward neural network with $M = 3$ hidden layers	17
2.1	Loss level during and after training	61
2.2	Approximate and exact solutions in 1d case	62
2.3	Average loss level during training with ADAM	63
2.4	Average loss level during training with SGD	63
2.5	Relative and absolute errors of the approximate solution, in terms of distance from the origin.	64
2.6	Actual loss level of the NN solution during training, compared to exact solution	64
2.7	Training loss level of NNs with 6 different number of nodes	65
2.8	Computational time of NNs with different number of nodes in the range $[2^2, 2^{15}]$ on a single node of GPU A100	66
2.9	Training loss of Q-PDE approximator solving (2.126) and its absolute error level	68
2.10	Training loss of Q-PDE approximator solving (2.126) and its absolute error level	69
4.1	A diagram of how actor-critic method updates controls $\{u_k\}$ and value estimates $\{q_k\}$ iteratively from step k to $k + 1$	115
4.2	Trained critics and actors after each iteration	139
4.3	Cost function and its relative error	139
4.4	Neural network control and its error	140
4.5	Trained critics and actors after each iteration	141
4.6	Cost function and its relative error	142
4.7	Neural network control and its error	142

Chapter 1

Introduction

In recent years, the application of neural networks has revolutionized many areas including scientific computing. Many neural-network-based algorithms have been proposed for tackling one of the main challenges in scientific computing: solving high-dimensional partial differential equations numerically. Despite their practical success, the mathematical analysis for such algorithms remains open. In this thesis, we aim to (i) propose a new algorithm for solving a class of partial differential equations; (ii) rigorously establish limiting behavior of neural-network-based PDE approximations; (iii) prove their convergence to exact solutions under general assumptions; (iv) demonstrate solutions of equations originating from stochastic control problems with these algorithms.

1.1 Background

Partial differential equations (PDEs) are one of the most commonly applied mathematical tools for describing system behaviors in physics, engineering sciences, mathematical finance, and many other areas. Finding solutions, or approximation of solutions, remains a core challenge for pure and applied mathematics. The mathematical theory of PDE has been extensively researched and is still rapidly expanding, and an indication of this can be seen from the immense array of classical literature on this topic in different forms, for instance, [30, 1, 33, 21, 77]. However, in many practical cases, as analytical solutions are unavailable it is necessary to approximate them numerically via machine-aided approaches.

Traditionally, two categories of numerical schemes for solving PDEs are adopted, namely finite difference (FD) and finite element (FE) methods. Finite difference methods allow derivatives in a problem to be approximated by difference quotients.

One first discretizes the domain of the problem to form proper grids, before approximations of the PDE are constructed by taking values from adjacent grid points, which form a stencil. By doing this, a (linear) PDE is converted into a sequence of linear equations that are solvable via matrix computations. Truncation error, the error between the exact solution and the approximated solution assuming perfect machine arithmetic, depends on the mesh size of the grid chosen. For example, when a Crank–Nicolson scheme [22] is applied to solve the heat equation, its truncation error is of the scale

$$\text{error} \propto \mathcal{O}(\Delta t^2 + \Delta x^2)$$

where Δt and Δx denote the mesh size of time and spatial grids respectively.

Finite element methods take a different approach, converting PDEs to their equivalent weak formulation. Typically, after formulating a grid (or mesh) that may consist of triangles, squares, or sometimes polygons of the domain, one chooses a finite basis of trial functions which are piecewise linear or piecewise polynomial functions of compact support, in correspondence with the predetermined mesh of the object. This process eliminates spatial derivatives from the original PDE, and consequently, the problem is decomposed into a set of ordinary differential equations (ODE) or algebraic equations that are solved via numerical linear algebra tools. Its error bound, for trial functions of polynomial order p and geometric scale h , in general satisfies [144]

$$\text{error} \propto \mathcal{O}(h^{p+1}).$$

While the mathematical analysis of FD and FE methods has been intensively conducted, it becomes clear that both methods could suffer from the **curse of dimensionality** (CoD). Mesh sizes are kept small in each dimension to guarantee approximation precision, and hence the number of grid points or trial functions required grows exponentially as the dimension of the problem goes large. In general, these two methods are incapable of solving problems of high dimensions as otherwise computational costs shall exceed the practical limit.

Since mesh-free methods may overcome the curse of dimensionality, function approximation tools derived from machine learning, especially neural networks, are natural candidates for higher dimensional problems. Recent findings also back the claim that neural networks can overcome this obstacle to achieve the required solution accuracy for various PDE problems, [41, 40, 53, 127, 36], to list a few. For ε -error L^2 neural network approximation of d -dimensional PDE solution u denoted by $u_{\varepsilon,d}$, the size of $u_{\varepsilon,d}$ (i.e. the number of units in the neural network) satisfies

$$\text{size}(u_{\varepsilon,d}) < Cd^k \varepsilon^{-k},$$

where $C, k > 0$ are constants invariant of dimension d and error bar ε for the same class of PDEs. This implies that the number of units required for the neural network increases at most polynomially with respect to d and ε . Notice that, for this result neural networks with all different number of hidden layers are considered. If we only considers single-layer neural networks, then this result may not hold. However, we mainly focuses on single-layer neural networks in this thesis, as multi-layer neural networks are much hard to analyze mathematically.

Machine learning, especially deep learning [81, 37], has revolutionized image recognition [126, 148], natural language processing [35, 115], gaming [137, 9, 85], and a broader set of other areas [131, 62, 106]. These areas require modeling nonlinear functions of high-dimensional inputs such as image pixels, voice signals, sentence series, etc. Neural networks, with different architectures, have shown their capability to tackle such tasks and have gained huge success. At its core, one takes the following steps to train a neural network for its designed purpose:

1. Choose an architecture f , and its parametrization class:

$$F : \{f_\theta : \theta \in \Theta\}.$$

2. Choose a cost functional $c : F \rightarrow \mathbb{R}$ and an optimization scheme to find

$$\theta^* \approx \arg \min_{\theta \in \Theta} c(f_\theta).$$

This general approach is also applicable for approximating PDE solutions. To access PDE solution u , one first selects a proper architecture such that $u \in F$, or at least $\|u - u^F\| < \epsilon$ for $\epsilon \ll 1$ where u^F is the best projection of u on F . Often, as prior knowledge of PDE solution is limited, some initial guesses are heuristically accepted and hyper-parameter tuning of different architectures is much required. The next critical step is to set a loss functional c and an optimization method; these form the main bodies of different PDE-solving algorithms. In the following subsection, we shall review some of these algorithms in detail.

Despite these methods' practical success in various applications, their mathematical properties, including numerical stability and convergence analysis, have not been well-studied. Specifically, most existing literature studies the existence of a neural network to approximate the solution with required precision, which reflects only the first part of the algorithm design. However, in this thesis, we closely examine the second part, the optimization scheme of these algorithms in some cases. We model the training flow of approximators as dynamic processes and provide proof of convergence under generic assumptions.

1.2 Deep-learning-based methods

In this subsection, we briefly review several categories of existing deep-learning-based PDE-solving algorithms that are related to the topics covered in this thesis. In earlier works, deep neural networks have been utilized to assist conventional numerical methods. Many of these are limited to low-dimensional problems as they cannot overcome the curse of dimensionality.

In [82, 34, 156], neural networks are applied to improve the effectiveness of the FD method. The CINT method [129] combines the classical Galerkin method with neural networks to solve initial boundary value problems. In [101, 102], the solution of ODEs is formed by a combination of splines whose coefficients are determined by training a neural network. [78, 79, 97] propose to solve differential equations with neural networks trained to minimize squared loss over meshed collocation points. Since they require pre-fixed mesh grids for discretization to solve equations, they are limited to low-dimensional problems. Taking a similar approach to these methods, the method to approximate the solutions by a single neural network in [26] could not be generalized to high-dimensional problem settings.

In the last two decades, neural networks have become much more popular in scientific computing areas, and a large number of new algorithms have come into view. Mainly, these algorithms can be categorized into four classes:

- Direct methods: Form the original problem as an optimization problem and seek to find a numerical solution directly.
- Probabilistic methods: Solutions of PDEs related to stochastic processes' value functionals can be approximated by neural networks on simulated trajectories of Monte-Carlo samples.
- Learning alternative forms: Neural networks are trained to learn the PDEs' weak solutions, energy forms, etc.
- Learning the operator: Given that data of PDE solutions are available, neural networks could be set to learn the operator mapping from the equations to these solutions and then generalize to PDEs with similar formulations.

In the scope of this thesis, we mainly focus on the mathematical analysis of methods in the first category. However, these approaches to the training of PDE-solving neural networks share a very similar loss minimization methodology at their core. Therefore, the analysis of algorithms across different categories has common features – in

the future, the analysis framework in this thesis may be generalizable to the other categories of algorithms listed above.

The following concise introduction to different algorithm categories aims to help readers understand recent advances in this subject. Nevertheless, this review is far from being exhaustive. For other articles reviewing these subjects, we refer readers to [32, 65, 153, 4, 23] and references therein.

1.2.1 Direct methods

We first introduce direct methods. Different from classic numerical methods, these methods no longer require mesh grids to carry out matrix computations on stencils. Taking advantage of recent years' computational resource boosting, neural networks learn to match the behavior of the PDE operators together with the boundary conditions by tuning their parameters on the fly. Let us take a second-order, potentially nonlinear PDE as an example:

$$\begin{aligned}\partial_t V(t, x) + \mathcal{L}V(t, x) &= 0, \quad (t, x) \in [0, T] \times \Omega \\ V(0, x) &= v_0(x), \quad x \in \Omega \\ V(t, x) &= h(t, x), \quad (t, x) \in [0, T] \times \partial\Omega,\end{aligned}\tag{1.1}$$

where $\Omega \subset \mathbb{R}^d$ is a compact domain and $\partial\Omega$ denotes its boundary. Given a solution to this PDE exists, to approximate this solution, naturally one may minimize the following cost functional for $f \in \mathcal{D}(\mathcal{L}) \cap L^2([0, T] \times \Omega)$:

$$c(f) := \|\partial_t f + \mathcal{L}f\|_{L^2([0, T] \times \Omega)}^2 + \|f(0, \cdot) - v_0\|_{L^2(\Omega)}^2 + \|f - h\|_{L^2([0, T] \times \partial\Omega)}^2.\tag{1.2}$$

This cost quantifies how well f satisfies the differential operator, together with the boundary condition and the initial condition. Intuitively, if $c(f) = 0$ then f represents a solution of the PDE.

Now, consider a class of parametrized functions, which may typically be neural networks: $\{f_\theta : \theta \in \Theta\}$. The optimization task of finding the minimizer of the cost functional within this function class becomes

$$\min_{\theta} J(\theta) := \|\partial_t f_\theta + \mathcal{L}f_\theta\|_{L^2([0, T] \times \Omega)}^2 + \|f_\theta(0, \cdot) - v_0\|_{L^2(\Omega)}^2 + \|f_\theta - h\|_{L^2([0, T] \times \partial\Omega)}^2.\tag{1.3}$$

In [138], this choice of cost functional J is justified. If one has $J(\theta_n) \rightarrow 0$ as $n \rightarrow \infty$, then under some mild assumptions it is guaranteed that $f_{\theta_n} \rightarrow V$ in $L^p(\Omega)$ for $p < 2$, where V is the unique solution of equation (1.1). Usually, this learning objective is hard to evaluate directly, hence some discretization is still required.

Deep Galerkin Method (DGM) [138] randomly samples points in the domain and approximately evaluates objective (1.3) in each iteration step; hence, it is mesh-free. To solve the sample PDE problem (1.1) numerically, DGM takes the following steps:

1. Choose neural network architecture f , initialize its parametrization $\theta = \theta_0$ with some distributions $\theta_0 \sim \nu$.
2. Generate random samples $s_1 = \{(t_i, x_i)\}_{1 \leq i \leq n_1}$ from $[0, T] \times \Omega$, $s_2 = \{x_j\}_{1 \leq j \leq n_2}$ from Ω and $s_3 = \{(t_k, x_k)\}_{1 \leq k \leq n_3}$ from $[0, T] \times \partial\Omega$ with respect to some probability densities respectively.
3. Compute the empirical squared loss at these random points $s = \cup_i s_i$:

$$L(\theta, s) := \frac{1}{|s_1|} \sum_i \left| \partial_t f_\theta(t_i, x_i) + \mathcal{L}f_\theta(t_i, x_i) \right|^2 + \frac{\lambda_I}{|s_2|} \sum_j \left| f_\theta(0, x_j) - v_0(x_j) \right|^2 + \frac{\lambda_B}{|s_3|} \sum_k \left| f_\theta(t_k, x_k) - h(t_k, x_k) \right|^2, \quad (1.4)$$

where λ_I and λ_B are the multipliers controlling the importance of errors in initial and boundary conditions.

4. Take optimization step with gradient term $\nabla_\theta L$ and chosen learning rate α :

$$\tilde{\theta} = \theta - \alpha \nabla_\theta L(\theta, s). \quad (1.5)$$

5. Update θ by $\tilde{\theta}$, then repeat steps 2 to 5 until the stopping criterion is met.

Notice that the gradient taken in step 4, $\nabla_\theta L(\theta, s)$, is an unbiased estimator of the gradient taken for the original functional if the sampling measure of L is the same one as in (1.3). Another critical point is that θ may only converge to a local minimum as the objective is non-convex in the parameter space, similar to other deep-learning tasks. Nevertheless, in Chapter 3, we will rigorously prove that as the number of units in the (single-layer) approximator f_θ goes large, the landscape shall eventually flat out – Neural networks will be able to reach the global optimality point through training via gradient descent as their representation power grows.

Physics-informed neural networks (PINNs) [123, 122] take an approach similar to DGM, except for a few differences. PINNs may incorporate additional information into the training process. For instance, if experimental sensors can provide

measured data of the solution u , which could be velocity, temperature, or flux density at some specified points $\{(t_i, x_i)\}$ directly, then to train the neural network approximator, users can add extra error terms $\|f_\theta(t_i, x_i) - u(t_i, x_i)\|_2^2$ into loss (1.4). This possibly speeds up the convergence to the approximator. Another key difference is that originally PINNs may keep using the same batch of sample points. Once a batch of points inside the domain and at the boundary are sampled, they will be used throughout the training session with no reshuffling.

Using the same batch of samples makes analyzing PINNs easier since the training is confined to a finite set of points. However, this feature also makes PINNs lose some of the convergence guarantees in theory. For instance, let us consider a second-order linear elliptic PDE with Dirichlet boundary condition:

$$\begin{aligned} \mathcal{A}u(x) &= h(x), \quad x \in \Omega \\ u(x) &= g(x), \quad x \in \partial\Omega. \end{aligned} \tag{1.6}$$

Given a randomly drawn, but then fixed finite set of points $\{x_j\} \subset \Omega$ and $\{y_k\} \subset \partial\Omega$, a perfectly trained PINNs approximator f_{θ^*} should satisfy

$$\sum_j (\mathcal{A}f_{\theta^*}(x_j) - h(x_j))^2 + \sum_k (f_{\theta^*}(y_k) - g(y_k))^2 = 0. \tag{1.7}$$

This does not guarantee that f_{θ^*} is close to solution u , in general. Consider the closed ball $\mathcal{B}_\epsilon(x_j) \subset \Omega$ where ϵ is small enough such that $\mathcal{B}_\epsilon(x_j) \cap \{x_i\}_{i \neq j} = \emptyset$. There exist infinitely-many choices of smooth functions ε such that ε is compactly supported only by the complement of $\cup_j \mathcal{B}_\epsilon(x_j)$ and that $\varepsilon(x) = 0$ for $x \in \partial\Omega$. Then, if we denote $\tilde{f}_{\theta^*} := f_{\theta^*} + \varepsilon$, similar to (1.7) one obtains

$$\sum_j (\mathcal{A}\tilde{f}_{\theta^*}(x_j) - h(x_j))^2 + \sum_k (\tilde{f}_{\theta^*}(y_k) - g(y_k))^2 = 0. \tag{1.8}$$

This suggests that PINNs cannot distinguish between the solution of the PDE and a perturbation of it without further assumptions. For this reason, existing analysis of PINNs' stationary points and consequent convergence results may require extra global regularity terms added to the neural network's optimization objective (1.4), for instance, Hölder regularity [132] or L^1 regularity [27]. Under these confinements, together with some extra assumptions on the PDE, the global minimum of PINNs' objective converges to the solution of the PDE as the number of neural network units goes to infinity. Another approach to specifying PINNs' capability to find PDE solutions takes Barron spaces [154] into consideration, and it is typically applicable

for cases where the neural network approximators are single-layer. Barron spaces consist of real-valued functions admitting a representation of the form

$$f(x) = \int_{\Omega} c\sigma(w \cdot x + b)\rho(dc, dw, db), \quad (1.9)$$

where ρ is a probability measure on parametrization space Ω , and activation function σ is chosen as the ReLU function [112]. It can be seen as a continuum analog of single-layer neural networks as the number of units goes to infinity. The Barron norm also provides a bound of Barron functions' generalization error (via Rademacher complexity). In [133], the error between PINNs' global minimizer and the true solution is bound by a term given by Rademacher complexity. In [16], a complexity estimate of the PDEs' solution in Barron spaces is established. In [95], a similar bound is given by terms including the path norm of the neural network. In [95], it is also shown that for neural networks that are large enough, following gradient descent (GD), with a positive probability, PINNs' training objective will decay quickly under some assumptions. Possible failure modes of PINNs under different problem setting are studied in, for instance, [31, 124, 75, 149, 151, 50].

PINNs method has various extensions and variants for different problem settings, for instance, XPINNs [56], nn-PINNs [96], Δ -PINNs [20], etc [67, 116, 57, 155, 136]. For their applications to solving physics and engineering problems, see [104, 125, 61, 98, 10, 58] for fluid dynamics, [11, 107, 17] for diffusion problems, [70, 130] for biological modeling, and [64, 91, 161, 108, 135, 47, 66] for problems from other areas, just to name a few.

The mathematical properties of the PINNs method remain open problems in many aspects. In Chapter 3, we study the convergence of PINNs in a vanilla setting, solving linear PDEs with a single-layer neural network approximator. We hope that it paves the way to understanding PINNs behavior for solving non-linear equations with multi-layer networks in the future.

1.2.2 Probabilistic methods

Certain PDE families are related to the expected values of some stochastic processes. For linear parabolic PDEs, one may apply the Feynman–Kac formula to evaluate the PDE solution at any given location via simulation. For example, consider the following PDE

$$\begin{aligned} \partial_t V(t, x) + \mathcal{A}V(t, x) + f(t, x) &= 0, \quad t \in [0, T] \\ V(T, x) &= h(x), \end{aligned} \quad (1.10)$$

where the operator $\mathcal{A}$ satisfies

$$\mathcal{A}u(t, x) := \sum_i b_i(t, x) \partial_i u(t, x) + \frac{1}{2} \sum_{j,k} \sigma \sigma^\top_{jk}(t, x) \partial_{jk} u(t, x) \quad (1.11)$$

for $u \in \mathcal{D}(\mathcal{A})$. Under mild assumptions, one observes that

$$V(t, x) = \mathbb{E}_x \left[h(X_T) + \int_t^T f(s, X_s) ds \right] \quad (1.12)$$

where X_t satisfies the stochastic differential equation (SDE) [114]

$$dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t, \quad X_t = x. \quad (1.13)$$

Therefore, $V(t, x)$ can be evaluated via Monte Carlo simulation of the process $\{X_s\}_{s \geq t}$. In [6], neural networks are applied to find solutions for parametric families of Kolmogorov partial differential equations via this general approach. Going beyond this, one may consider semi-linear parabolic PDEs:

$$\begin{aligned} \partial_t u(t, x) + \mathcal{A}u(t, x) + f(t, x, u(t, x), \sigma^\top(t, x) \nabla u(t, x)) &= 0, \quad t \in [0, T] \\ u(T, x) &= h(x). \end{aligned} \quad (1.14)$$

Backward stochastic differential equations (BSDEs) give such PDEs a probabilistic representation (nonlinear Feynman–Kac formula) [117, 29]. For the BSDE

$$\begin{aligned} X_t &= X_0 + \int_0^t b(s, X_s) ds + \int_0^t \sigma(s, X_s) dW_s, \\ Y_t &= h(X_T) + \int_t^T f(s, X_s, Y_s, Z_s) ds - \int_t^T Z_s \cdot dW_s, \quad 0 \leq t \leq T, \end{aligned} \quad (1.15)$$

its solution is an adapted process $\{(X_t, Y_t, Z_t)\}_{0 \leq t \leq T}$ satisfying $Y_t = u(t, X_t)$ and $Z_t = \sigma^\top(t, X_t) \nabla u(t, X_t)$ for $t \in [0, T]$. This result also reflects the Pontryagin maximum principle for stochastic control problems [119]. For the variational problem

$$\begin{aligned} \inf_{Y_0, \{Z_t\}} \mathbb{E} |h(X_T) - Y_t|^2 \\ \text{s.t. } X_t &= X_0 + \int_0^t b(s, X_s) ds + \int_0^t \sigma(s, X_s) dW_s, \\ Y_t &= Y_0 - \int_0^t f(s, X_s, Y_s, Z_s) ds + \int_0^t Z_s \cdot dW_s, \end{aligned} \quad (1.16)$$

has a unique minimizer which is the solution to the PDE (1.14). The **Deep BSDE** method [44, 43] solves the PDE via solving this equivalent variational formulation.

After time discretization via an Euler–Maruyama scheme on a given grid $0 = t_0 < t_1 < \dots < t_N = T$, the variational problem is converted to

$$\begin{aligned}
& \inf_{\psi_0, \{\phi_n\}_{n=0}^{N-1}} \mathbb{E} |h(X_T) - Y_t|^2 \\
& \text{s.t. } Y_0 = \psi_0(X_0) \\
& X_{t_{i+1}} = X_{t_i} + b(t_i, X_{t_i})\Delta t_i + \sigma(t_i, X_{t_i})\Delta W_i, \\
& Z_{t_i} = \phi_i(X_{t_i}), \\
& Y_{t_{i+1}} = Y_{t_i} - f(t_i, X_{t_i}, Y_{t_i}, Z_{t_i})\Delta t_i + Z_{t_i} \cdot \Delta W_i.
\end{aligned} \tag{1.17}$$

Deep BSDE applies a set of neural networks ψ and $\{\phi_i\}$ to approximate the unknown mappings $X_0 \rightarrow u(0, Y_0)$ and $X_t \rightarrow \sigma^\top(t, X_t)\nabla u(t, X_t)$ respectively. All these sub-networks together form a deep neural network as a whole. This network takes paths $\{X_{t_i}\}$ and Brownian increments $\{\Delta W_i\}$ as input arguments and gives the output $\tilde{u}(\{X_{t_i}\}, \{\Delta W_i\})$ as an approximation to $u(T, X_T)$. The neural network’s optimization objective now becomes

$$J(\psi, \{\phi_i\}) = \mathbb{E} \left[|h(X_T) - \tilde{u}|^2 \right], \tag{1.18}$$

and a stochastic gradient descent (SGD) algorithm for neural network training can be applied to the objective above with each batch of simulated paths of X and ΔW . Convergence of the Deep BSDE algorithm is studied in [45, 60]. Another approach, **Deep Backward Dynamic Programming** (DBDP) [52] utilizes backward induction

$$Y_{t_i} = Y_{t_{i+1}} + f(t_i, X_{t_i}, Y_{t_i}, Z_{t_i})\Delta t_i - Z_{t_i} \cdot \Delta W_i \tag{1.19}$$

instead, and at each time step t_i approximates the processes (Y_{t_i}, Z_{t_i}) with neural networks trained with X_{t_i} as input. We refrain from further discussion of the extensions and variants of Deep BSDE and DBDP methods here and refer readers to [32] and references therein.

1.2.3 Learning alternative formulations

PDEs can often be solved by transforming the original problem into equivalent or weaker formulations. In this subsection, we review a few examples to demonstrate that deep-learning-based methods can exploit this approach for various PDE classes.

The deep Ritz method [157] seeks to solve the variational problem

$$\min_{u \in H} I(u) \tag{1.20}$$

where H is the set of trial functions, and the variational form satisfies

$$I(u) = \int_{\Omega} \left(-fu + \frac{1}{2} \|\nabla u\|^2 \right) d\mu, \quad (1.21)$$

where μ is the Lebesgue measure over domain Ω , and f is a given function representing external forcing to the system. The deep Ritz method makes use of a neural network approximator $u(x; \theta)$ with activation function nodes $\phi : x \rightarrow \max\{x^3, 0\}$ to minimize the loss functional $L(\theta) := I(u(\cdot; \theta))$. At each training step, a mini-batch of N points is randomly sampled, and the empirical loss

$$L_N(\theta) := \frac{1}{N} \sum_{i=1}^N \left(-f(x_i)u(x_i; \theta) + \frac{1}{2} \|\nabla u(x_i; \theta)\|^2 \right). \quad (1.22)$$

is trained to be minimized via the gradient descent direction. Observe that the potential form (1.21) is related to the solution of the following Poisson equation

$$\begin{aligned} -\Delta u(x) &= f(x), & x \in \Omega \\ u(x) &= u_0(x), & x \in \partial\Omega \end{aligned} \quad (1.23)$$

in the sense that u is the minimizer of (1.21) subject to $u = u_0$ at the boundary. To solve this equation, an extra regularity term

$$\beta \int_{\partial\Omega} |u(x; \theta) - u_0(x)|^2 \mu(dx) \quad (1.24)$$

is added to loss functional $L(\theta)$, where $\beta \gg 1$ forces the approximator to learn the boundary condition. Numerical examples of the algorithm are shown in [157] and its convergence is analyzed in [111].

Deep least-square methods [12] focus on solving the following second-order elliptic PDE:

$$-\nabla \cdot (A\nabla u) + Xu = f \quad \text{in } \Omega \quad (1.25)$$

with boundary conditions

$$u = g_D, \text{ on } \Gamma_D \quad \text{and} \quad -\mathbf{n} \cdot A\nabla u = g_N, \text{ on } \Gamma_N. \quad (1.26)$$

Here $A(x)$ denotes a symmetric matrix-valued function in $L^2(\Omega)^{d \times d}$, X is a linear differential operator of order up to one, and $\mathbf{n}$ is the outward unit normal vector. It introduces a flux variable $\sigma := -A\nabla u$ and the problem is rewritten as a first-order one:

$$\nabla \cdot \sigma + Xu = f, \text{ in } \Omega \quad \text{and} \quad \sigma + A\nabla u = 0, \text{ in } \Omega, \quad (1.27)$$

with the same set of boundary conditions. Two separate networks σ_θ, u_η are applied to learn σ and u respectively. By the fact that the interior and boundary norms are not on the same scale, this method uses a balanced least-square functional as the learning objective:

$$\begin{aligned} \mathcal{G}(\sigma_\theta, u_\eta) := & \left\| \nabla \cdot \sigma_\theta + Xu_\eta - f \right\|_{0,\Omega}^2 + \left\| A^{-1/2} \sigma_\theta + A^{1/2} \nabla u_\eta \right\|_{0,\Omega}^2 + \alpha_D \left\| u_\eta - g_D \right\|_{1/2,\Gamma_D}^2 \\ & + \alpha_N \left\| \mathbf{n} \cdot \sigma_\theta - g_N \right\|_{-1/2,\Gamma_N}^2. \end{aligned} \quad (1.28)$$

In practice, as the Sobolev norms $\|\cdot\|_{1/2}$ and $\|\cdot\|_{-1/2}$ are not computationally feasible, they are approximated by weighted L^2 norms with local weights on a batch of sampled points.

Weak adversarial networks (WANs) [159] take a different approach. These exploit weak solutions for PDEs. Consider the following elliptic PDE boundary value problem (BVP):

$$\begin{aligned} \mathcal{A}u &= 0, \text{ in } \Omega \\ \mathcal{B}u &= 0, \text{ on } \partial\Omega. \end{aligned} \quad (1.29)$$

For test functions in $\mathcal{H}_0^1(\Omega)$, after integration by parts, the weak formulation of the PDE becomes

$$\langle \mathcal{A}u, \psi \rangle_2 = 0, \quad \forall \psi \in \mathcal{H}_0^1(\Omega) \quad (1.30)$$

with boundary conditions remaining unchanged. Consider the mapping $\mathcal{H}_0^1(\Omega) \rightarrow \mathbb{R} : \psi \rightarrow \mathcal{A}u(\psi) = \langle \mathcal{A}u, \psi \rangle$, for fixed u this is a linear functional on $\mathcal{H}_0^1(\Omega)$. Its operator norm, induced from the L^2 norm is defined as $\|\mathcal{A}u\|_{op} := \sup\{\langle \mathcal{A}u, \psi \rangle_2 / \|\psi\|_2 \mid 0 \neq \psi \in \mathcal{H}_0^1(\Omega)\}$. It is proved that $\tilde{u}$ is a weak solution of the PDE iff $\|\mathcal{A}\tilde{u}\|_{op} = 0$ and the boundary condition is satisfied. Since the optimization problems

$$\min_{u \in \mathcal{H}^1} \|\mathcal{A}\tilde{u}\|_{op} \quad \text{and} \quad \min_{u \in \mathcal{H}^1} \max_{\psi \in \mathcal{H}_0^1} |\langle \mathcal{A}u, \psi \rangle_2 / \|\psi\|_2| \quad (1.31)$$

are equivalent, WANs methodologically adopt an idea similar to generative adversarial networks (GANs) [38] to approximate a weak solution. Two neural networks u_θ and ψ_η are introduced, where the latter works as a trial function that performs the maximization task and the former learns to minimize the operator norm. The loss functional

$$L_I(\theta, \eta) := \log|\langle \mathcal{A}[u_\theta], \psi_\eta \rangle_2|^2 - \log\|\psi_\eta\|_2^2 \quad (1.32)$$

is aligned with the min-max objective to be solved via neural networks' training

$$\min_{\theta} \max_{\eta} \left\{ L_I(\theta, \eta) + \alpha L_B(\theta) \right\}, \quad (1.33)$$

in which $L_B(\theta)$ represents the boundary cost functional measuring error of u_{θ} fitting the boundary conditions and α is a pre-fixed balancing parameter. In each epoch, gradient terms are computed, and θ, η take gradient descent and ascent directions respectively. Friedrichs learning method [14] takes an approach similar to WANNs.

Some other methods also take advantage of variational forms of PDEs. In [68], a neural network parameterizes the committor function. Its weights are trained by optimizing the variational form. Combining PINNs and the variational form, in [67], domain decomposition is applied and local test functions are selected. The neural network is trained to optimize the sum of variational losses.

1.2.4 Learning operators

Under some circumstances, users seek to approximate solutions for a parametric family of equations simultaneously, where some data of parameter-solution pairs are available. For instance, consider the following ODE system:

$$\frac{dy}{dx}(x) = g(y(x), x, u(x)), \quad y(0) = y_0, \quad (1.34)$$

where $u \in V$, and V is a compact subset of $C[0, 1]$. $y : [0, 1] \rightarrow \mathbb{R}$ is the solution of the equation, implicitly depends on the choice of u . Conventional ODE solvers and algorithms from the above three categories require repeating the same computational procedures to obtain a solution for the ODE once u is modified. In a PDE problem setting, consider a generic family of PDE BVPs:

$$\begin{aligned} \mathcal{L}_a u(x) &= f(x), & x \in \Omega \\ u(x) &= 0, & x \in \partial\Omega \end{aligned} \quad (1.35)$$

for $a \in \mathcal{A}$, where $\mathcal{A}$ is the Banach space for parameterization. Let us assume that the solution $u : \Omega \rightarrow \mathbb{R}$ belongs to a Banach space $\mathcal{U}$, and $f \in \mathcal{U}^*$, the dual space of $\mathcal{U}$. Operator $\mathcal{L}_a : \mathcal{A} \rightarrow L(\mathcal{U}, \mathcal{U}^*)$ maps each parameterization of the PDE a to a linear operator $\mathcal{U} \rightarrow \mathcal{U}^*$.

In this case, to learn the solution of this parametric PDE family, the map that solvers are essentially modeling is $\mathcal{S} : \mathcal{A} \rightarrow \mathcal{U}$ which maps $a \rightarrow u$. Given that a training data set $\{(a_i, u_i)\}_{i=1}^N$ is available, one can train (a system of) neural networks to approximate this mapping via a supervised learning approach. This neural network

system $\mathcal{S}_\theta$ takes parameterization a as input and outputs an approximation of the solution u_θ . The Bochner norm characterizes the error of this approximation:

$$\left\| \mathcal{S}_\theta - \mathcal{S} \right\|_{L^2(\mathcal{A}, \mathcal{U}, \mu)}^2 := \int_{\mathcal{A}} \left\| \mathcal{S}_\theta(a) - \mathcal{S}(a) \right\|_{\mathcal{U}}^2 \mu(da) = \int_{\mathcal{A}} \left\| \mathcal{S}_\theta(a) - u_a \right\|_{\mathcal{U}}^2 \mu(da). \quad (1.36)$$

On the data samples $\{(a_i, u_i)\}_{i=1}^N$, one solves the empirical risk minimization problem

$$\min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N \left\| \mathcal{S}_\theta(a_i) - u_i \right\|_{\mathcal{U}}^2. \quad (1.37)$$

The universal approximation theorem of operators [15] shows that neural networks can represent operators on a compact set. For continuous non-polynomial activation function σ , Banach space X , suppose $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact subsets respectively, $V \subset C(K_1)$ is a compact subset of continuous functions, and G is a non-linear continuous operator $G : V \rightarrow C(K_2)$. For any error level $\epsilon > 0$, there exists positive integers n, p, m constants $c_i^k, w_{i,j}^k, b_i^k, z_k \in \mathbb{R}$, $v_k \in \mathbb{R}^d$, $x_j \in K_1$ for $1 \leq i \leq m$, $1 \leq j \leq n$, $1 \leq k \leq p$ such that

$$\left| G(u)(y) - \sum_{k=1}^p \sum_{i=1}^m c_i^k \sigma \left(\sum_{j=1}^n w_{i,j}^k u(x_j) + b_i^k \right) \sigma(v_k \cdot y + z_k) \right| \leq \epsilon \quad (1.38)$$

for any $u \in V$ and $y \in K_2$.

To realize this approximation result, deep operator networks (DeepONets) [89] consist of two networks. The “branch” network encodes the input function u at a fixed number of input locations $\{x_j\}$, which effectively learns the

$$\sum_{i=1}^m c_i^k \sigma \left(\sum_{j=1}^n w_{i,j}^k u(x_j) + b_i^k \right) \quad (1.39)$$

component of the approximation in (1.38), and the “trunk” network approximates the following

$$\sigma(v_k \cdot y + z_k) \quad (1.40)$$

part. Given input function $u \in V$, the branch network, consisting of several sub-networks, takes a vector $[u(x_1), \dots, u(x_n)]^\top$ of function values as input and output $\xi_i \in \mathbb{R}$ from each of the i -th sub-network. Meanwhile, the branch net takes $y \in K_2 \subset \mathbb{R}^d$ at its input and returns a set of weights $[t_1, \dots, t_p]^\top$ for each of the branch sub-network. Merging them together, approximation of $G(u)(y)$ is hence given by

$$G(u)(y) \approx \sum_{i=1}^p \xi_i t_i. \quad (1.41)$$

For error estimates and convergence analysis of DeepONets, see [80, 25, 99].

Neural operators (NOs) [74] take another approach, where essentially they learn to express integral operators. As an example, for elliptic operator $\mathcal{L}_a = -\text{div}(a\nabla\cdot)$, the solution of the PDE family (1.35) is represented as

$$u(x) = \int_{\Omega} G_a(x, y) f(y) dy \quad (1.42)$$

where G_a denotes the Green's function, which is the unique solution to the problem

$$\mathcal{L}_a G_a(x, \cdot) = \delta_x(\cdot). \quad (1.43)$$

Thus, to learn the mapping $a \rightarrow u$, one learns the mapping from a to the integral operator induced by $G_a(\cdot, \cdot)$ as equivalent. The proposed architecture of neural operators $\mathcal{S}_{\theta} : \mathcal{A} \rightarrow \mathcal{U}$ has the following components:

1. Lifting $\mathcal{Q}$: Map input a to its first hidden representation ε_1 , performing a fully local operation.
2. Kernel Integration: Map each hidden representation to the next one $\varepsilon_i \rightarrow \varepsilon_{i+1}$, by summing a bias function b , a local linear operator W , and a non-local integral kernel operator $\mathcal{K}$.
3. Projection $\mathcal{P}$: Map the last hidden representation ε_T to $u_{\theta} \in \mathcal{U}$, performing a fully local operation.

This outlined structure mimics a multi-layer neural network, and $\mathcal{S}_{\theta}$ has the high-level representation

$$\mathcal{S}_{\theta} := \mathcal{P} \circ \sigma_T(W_T + \mathcal{K}_T - b_T) \circ \dots \circ \sigma_1(W_1 + \mathcal{K}_1 - b_1) \circ \mathcal{Q}. \quad (1.44)$$

For each single kernel integration layer, ε_i is mapped into ε_{i+1} following

$$\varepsilon_{i+1}(x) = \sigma\left(W_i \varepsilon_i(\Pi_i x) + \int_{\Omega} k_i(x, y) \varepsilon_i(y) \mu_i(dy) + b_i(x)\right), \quad (1.45)$$

where Π_i represents a fixed local mapping, often identity. Several specifications of NOs apply the integral operator $\mathcal{K}_i$ part $\int_{\Omega} k_i(x, y) \varepsilon_i(y) \mu_i(dy)$ differently. The **graph neural operator** (GNO) [3] evaluates the integral of a fixed set of points with a parametrized kernel

$$\mathcal{K}\varepsilon(x) = \frac{1}{|\mathcal{N}|} \sum_{y \in \mathcal{N}} k_{\theta}(x, y) \varepsilon(y), \quad (1.46)$$

while the well-known **Fourier neural operator** (FNO) [87] applies both the Fourier transform $\mathcal{F}$ and its inverse to obtain

$$\mathcal{K}\varepsilon(x) = \left(\mathcal{F}^{-1}(R_\theta) \cdot \mathcal{F}(\varepsilon) \right)(x), \quad (1.47)$$

where R_θ is the Fourier transform of a periodic function k parametrized by θ . As an example of FNO’s application, in [76] it is applied to obtain high-resolution weather forecasting results. For error analysis of FNO, see [73]. For a detailed comparison of DeepONet and FNO, see [90].

Neural operator methods have several other variants as well, for instance, wavelet neural operator (WNO) [147], low-rank neural operator (LNO) [74], etc. PINNs and neural operators are combined in [88, 150], where neural operator solutions are trained by PINNs for fine-tuning. Many of their mathematical properties remain open problems as well. However, we also point out that operator learning at its core is a class of supervised learning problems on an infinite-dimensional space.

1.3 Mathematical tools

In this subsection, we briefly go over some of the mathematical tools in the scope of this thesis. We start with the mathematical properties of neural networks, and the specific type of neural network we study in this paper. Next, we introduce the mean-field type analysis of neural networks, which characterizes the limit behavior of neural networks when the number of units goes to infinity. Asymptotically, the evolution of the neural network’s training trajectory shall converge to a limit dynamic system, often an infinite-dimensional ODE or PDE. Then, kernel methods, especially the idea of *neural tangent kernel* are covered. Lastly, we list some functional analysis concepts that are important to the ideas explored in this thesis.

1.3.1 Neural networks as function approximators

The concept of artificial neural networks has a long history, tracing back to the 1950s. In the early 2010s, the application of neural networks revolutionized image classification areas as it outperformed conventional methods by a huge margin, and neural networks have become extremely popular in numerous areas since then.

Different architectures of neural networks are preferred for a variety of tasks. For instance, convolutional neural network (CNN) [86] and residual network (ResNet) [46] are better for image classification and general computer vision tasks as they are refined for distinguishing the edge of objects from pixels and maintaining a low error

rate, and recurrent neural networks [103, 158] are capable of time series analysis and other sequential tasks as the recurrent part contains informational memory from past sequence data. In this thesis, we focus on feed-forward neural networks (FNNs), a vanilla type of neural network.

Definition 1.3.1. (*Fully-connected FNN*) A fully-connected feed-forward neural network of M layers $\psi_{\theta}^M : \mathbb{R}^d \rightarrow \mathbb{R}$ is a composition of non-linear functions:

$$\psi_{\theta}^M(x) := c^{\top} h^M \circ h^{M-1} \circ \dots \circ h^1(x), \quad (1.48)$$

where $h^k(x) = \sigma(\mathbf{W}^k x + \mathbf{b}^k)$ with $\mathbf{W}^k \in \mathbb{R}^{m_k \times m_{k-1}}$, $\mathbf{b}^k \in \mathbb{R}^{m_k}$ for $k \in \{1, 2, \dots, M\}$, and $c \in \mathbb{R}^{m_M}$, $m_0 = d$. The map σ is a non-linear activation function applied componentwise. The notation h^k represents the k -th (hidden) layer of the network of width m_k . $\theta := \{c, \{\mathbf{W}^k\}_{1 \leq k \leq M}, \{\mathbf{b}^k\}_{1 \leq k \leq M}\}$ gives the parameterization of the neural network.

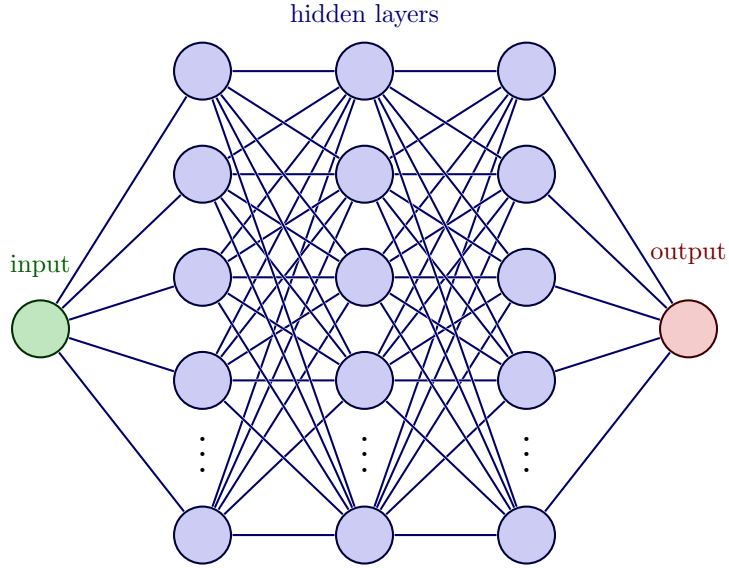


Figure 1.1: Example of a feedforward neural network with $M = 3$ hidden layers

Mathematically, FNNs are parameterized functions formed by the compositions of non-linear functions. Recursively, they can be defined for any finite number of layers. Multi-layer neural networks may learn differently from single-layer neural networks [121, 113], and outperform single-layer opponents for some tasks, see for instance [7]. However, in the remainder of this thesis, we will focus on single-layer FNNs, as their training is mathematically tractable. Understanding their behavior should pave

the way for understanding the evolution of neural networks with more sophisticated structures. Single-layer neural networks (of N units) admit a simplified expression

$$\psi_\theta(x) = \sum_{i=1}^N c_i \sigma(w^i \cdot x + b^i), \quad (1.49)$$

which can be seen as a linear combination of m separate basis function nodes. Thus, some of its properties are revealed from a functional analysis perspective.

A natural question is whether neural networks have the universal power to approximate arbitrary functions in L^2 space or even $\mathcal{H}^2$ Sobolev spaces, and the answer is yes. In [24, 49, 48], universal approximation theorems for neural networks are given, and an extended version for multi-layer networks is proved in [93]. Similar to the Stone–Weierstrass theorem for real polynomial families, on a compact set Ω , given an arbitrary continuous function $f \in C(\Omega)$, for any precision $\epsilon > 0$, there exists N_ϵ and parameterization set θ_ϵ such that

$$\|\psi_{\theta_\epsilon} - f\|_{L^\infty(\Omega)} < \epsilon. \quad (1.50)$$

Since continuous functions are dense in $L^2(\Omega)$, the class of neural networks is also dense. A similar result also holds for the Sobolev $\mathcal{H}^2$ norm. For function $h \in \mathcal{H}^2(\Omega)$, for any error level $\epsilon > 0$, there exists N_ϵ and parameterization set θ_ϵ such that

$$\|\psi_{\theta_\epsilon} - h\|_{\mathcal{H}^2(\Omega)} < \epsilon, \quad (1.51)$$

if the activation function σ is smooth and bounded. This universal approximation result consists of the foundation of training neural networks to solve (second-order) PDEs. If the solution of a PDE on a compact set is in $\mathcal{H}^2$, then neural networks can approximate it to any given precision. However, this does not imply an accurate approximation is achievable via training. The approximator may be trapped in a local minimum and fail to find the global optimizer.

In some cases, approximators require extra components to satisfy certain conditions. For instance, for a neural network ψ_θ , one may multiply by $\eta(x) := 1 - \|x\|_2^2$ to obtain $\Psi_\theta = \eta \cdot \psi_\theta$ to construct a neural-network-based function that satisfies $\Psi \equiv 0$ on the unit sphere. Such a function will help match the solution of a PDE with a Dirichlet boundary condition (in the unit ball). We shall return to prove the representation property of such Ψ_θ in Chapter 2.

Lastly, the training of neural networks requires parameters to be updated via gradient descent (GD), or stochastic gradient descent (SGD) and its variants, for

instance, AdaGrad [28] or ADAM [69]. As PDE operators are unbounded in the L^2 -operator norm, to learn PDE solutions, the training process of these parameters can be unstable, leading to a gradient explosion phenomenon similar to the examples given in [37]. To avoid that, Gradient clipping procedures or special activation function designs can be introduced for training stability purposes. In Chapters 2 and 3, we will introduce our findings on this matter.

1.3.2 Mean-field analysis of neural networks

The mean-field limit characterizes neural networks' behavior when the number of network units $N \rightarrow \infty$. Of course, different scaling factors (e.g. $\frac{1}{N}$ in (1.53) and $\frac{1}{\sqrt{N}}$ in (1.71)) result in different convergence modes, including almost sure convergence, or convergence in probability and distribution, etc. For the single-layer network ψ^N in (1.49), given that σ is bounded, $\{c^i, w^i\}$ are i.i.d samples, and that $\mathbb{E}[c] = 0$, $\mathbb{V}[c] < \infty$, by the law of large numbers and the central limit theorem we have

$$\lim_{N \rightarrow \infty} \frac{\psi^N(x)}{N} = 0, \text{ a.s.} \quad \text{and} \quad \lim_{N \rightarrow \infty} \frac{\psi^N(x)}{\sqrt{N}} \sim \mathcal{N}\left(0, \mathbb{V}[c\sigma(w \cdot x)]\right), \text{ dist.} \quad (1.52)$$

Different scaling factors over N , together with different learning rates α^N , provide different types of mean-field limit results.

Consider training a single-layer N -unit neural network $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}$

$$\varphi_\theta^N(x) := \frac{1}{N} \sum_{i=1}^N c^i \sigma(w^i \cdot x) \quad (1.53)$$

to solve the following regression problem

$$\min_{\theta} L^N(\theta) := \frac{1}{2} \mathbb{E}_{(Y,X) \sim \pi} \left[|Y - \varphi_\theta^N(X)|^2 \right], \quad (1.54)$$

where π is a probability measure on $\mathbb{R} \times \mathbb{R}^d$. Suppose that parameters θ of φ_θ^N are updated via vanilla SGD with constant learning rate $\alpha > 0$ and single random sample (Y_k, X_k) at the k -th iteration step:

$$\begin{aligned} c_{k+1}^i &= c_k^i + \frac{\alpha}{N} (Y_k - \varphi_{\theta_k}^N(X_k)) \sigma(w_k^i \cdot X_k), \\ w_{k+1}^{i,k} &= w_k^{i,j} + \frac{\alpha}{N} (Y_k - \varphi_{\theta_k}^N(X_k)) c_k^i \sigma'(w_k^i \cdot X_k) x_k^j, \end{aligned} \quad (1.55)$$

for $1 \leq i \leq N$, $1 \leq j \leq d$. Then, if one considers the empirical measure of parameters

$$\nu_k^N(dc, dw) = \frac{1}{N} \sum_{i=1}^N \delta_{c_k^i, w_k^i}(dc, dw), \quad (1.56)$$

the neural network at time k can be rewritten as a linear functional of $\nu_k^N(dc, dw)$:

$$\varphi_{\theta_k}^N = \langle c\sigma(w \cdot x), \nu_k^N \rangle. \quad (1.57)$$

In [140], it is then proved that under certain technical assumptions, the scaled empirical measure $\mu_t^N := \nu_{\lfloor Nt \rfloor}^N$ converges in distribution to a limit measure $\bar{\mu}_t$ in the Skorokhod space $D_E([0, T])$ for $E = \mathcal{M}(\mathbb{R}^{1+d})$, where the limit measure is the unique deterministic solution to the measure-valued equation

$$\begin{aligned} & \langle f, \bar{\mu}_t \rangle \\ &= \langle f, \bar{\mu}_0 \rangle + \int_0^t \left(\int_{\mathcal{X} \times \mathcal{Y}} \alpha(y - \langle c' \sigma(w' \cdot x), \bar{\mu}_s \rangle) \langle \nabla(c\sigma(w \cdot x)) \cdot \nabla f, \bar{\mu}_s \rangle \pi(dx, dy) ds, \right. \\ & \quad \left. \right) \end{aligned} \quad (1.58)$$

for every $f \in C_b^2(\mathbb{R}^{1+d})$, and where $\nabla f := (\nabla_c f, \nabla_w f)$ and $\bar{\mu}_0$ is the limit in distribution $\mu_0^N \rightarrow \bar{\mu}_0$ as $N \rightarrow \infty$. Substituting f by $c\sigma(w \cdot x)$ in (1.58), this describes the evolution of the neural network's limit as $N \rightarrow \infty$. In addition, the loss functional satisfies

$$\lim_{N \rightarrow \infty} L^N(\theta_{\lfloor Nt \rfloor}) = \frac{1}{2} \int_{\mathcal{X} \times \mathcal{Y}} [y - \bar{g}(x, p(t, \cdot))]^2 \pi(dx, dy), \quad (1.59)$$

where

$$\bar{g}(x, p(t, \cdot)) := \int_{\mathbb{R}} c\sigma(w \cdot x) p(t, c, w) dc dw, \quad (1.60)$$

with $p(t, \theta)$ satisfying the PDE

$$\begin{aligned} & \frac{\partial p(t, \theta)}{\partial t} = -\alpha \operatorname{div}_{\theta}(p(t, \theta)) \nabla_{\theta} \left[v(\theta, p(t, \cdot)) \right], \\ & v(\theta, p(t, \cdot)) = \int_{\mathcal{X} \times \mathcal{Y}} \left(y - \int_{\mathbb{R}^{d+1}} c' \sigma(w' \cdot x) p(t, c', w') dc' dw' \right) c\sigma(w \cdot x) \pi(dx, dy). \end{aligned} \quad (1.61)$$

The limit of the objective function (1.54) converges to (1.59).

Some of these convergence results can be extended to multi-layer networks [142]. In [142], by carefully choosing the learning rate for second-layer parameters and analyzing their dynamics, the authors establish a limit ODE for two-layer neural networks. Then, it is proved that a two-layer network recovers the global minimum in the limit regime by showing that this limit ODE converges to the global optimal point as $t \rightarrow \infty$. The authors also reveal a key difference between multi-layer neural networks and single-layer ones. Traditional mean-field approaches applied in

[140, 105] for single-layer networks cannot be used for multi-layer ones since nested measures for these multi-layer neural networks are not closed (see Section 4.3 of [142] for detailed derivations). In this thesis, we analyze the behavior of neural networks for solving PDEs, which form different objective functions from those regression tasks discussed in [142]. PDE operators are typically not L^2 -bounded, and to deal with this we introduced many truncation components to stabilize the training process. These components are not necessary for neural network training for regression tasks. In future research, with finer analytical approaches, our analysis could be extended to PDE-solving tasks with multi-layer neural networks, and proving an ODE limit for these neural networks is possible.

The (scaled) fluctuation process $\eta_t^N := \sqrt{N}(\mu_t^N - \bar{\mu}_t)$ with μ_t in (1.58) is further studied in [139]. It is proved that under mild assumptions, η_t^N converges in distribution to $\bar{\eta}_t$, which satisfies a stochastic partial differential equation (SPDE):

$$\begin{aligned} \langle f, \bar{\eta}_t \rangle &= \langle f, \bar{\eta}_0 \rangle + \int_0^t \int_{\mathcal{X} \times \mathcal{Y}} \alpha \left(y - \langle c\sigma(w \cdot x), \bar{\mu}_s \rangle \right) \times \langle \nabla c\sigma(w \cdot x) \cdot \nabla f, \bar{\eta}_s \rangle \pi(dx, dy) ds \\ &\quad - \int_0^t \int_{\mathcal{X} \times \mathcal{Y}} \alpha \langle c\sigma(w \cdot x), \bar{\eta}_s \rangle \langle \nabla c\sigma(w \cdot x) \cdot \nabla f, \bar{\mu}_s \rangle \pi(dx, dy) ds + \langle f, \bar{M}_t \rangle, \end{aligned} \quad (1.62)$$

where $\bar{\eta}_0 := \lim_{N \rightarrow \infty} \sqrt{N}(\mu_t^N - \bar{\mu}_t)$ and $\bar{M}_t$ is a given zero-mean Gaussian process. Asymptotically, the training process (of performing the regression task) satisfies

$$\mu_t^N \approx \mu_t + \frac{1}{\sqrt{N}} \bar{\eta}_t. \quad (1.63)$$

In [105], the authors consider a scaled version of the neural network with scaling factor α

$$\hat{f}_\theta^N(x) := \frac{\alpha}{N} \sum_{i=1}^N c_i \sigma(w^i \cdot x). \quad (1.64)$$

Under stronger (uniform boundedness and other) assumptions, similar limit dynamic results are obtained. In addition, a bound for the error between the mean-field residual dynamics and the linearized residual dynamics is given. The distribution ρ_t^α satisfies the rescaled distributional dynamics

$$\begin{aligned} \partial_t \rho_t^\alpha &= \frac{1}{\alpha} \nabla_\theta \cdot \left(\rho_t^\alpha \nabla_\theta \Psi_\alpha(\theta, \rho_t^\alpha) \right) \\ \Psi_\alpha(\theta, \rho) &:= -\mathbb{E} \left[c\sigma(w \cdot x) y \right] + \alpha \int \mathbb{E} \left[c\sigma(w \cdot x) c' \sigma(w' \cdot x) \right] \rho(dc', dw'). \end{aligned} \quad (1.65)$$

The mean-field residual dynamics for $u_t^\alpha := \mathbb{E}[y|x] - \alpha \int c\sigma(w \cdot x)\rho(dc, dw)$ satisfy

$$\partial_t u_t^\alpha(x) = -(\mathcal{H}_{\rho_t^\alpha})u_t^\alpha(x), \quad (1.66)$$

where $\mathcal{H}_\rho$ is the integral operator induced by the kernel

$$H_\rho(x, x') := \int \langle \nabla_\theta c\sigma(w \cdot x), \nabla_\theta c\sigma(w \cdot x') \rangle \rho(dw). \quad (1.67)$$

At the same time, the linearized residual dynamics are written as

$$\partial_t u_t^*(x) = -\mathcal{H}_{\rho_0} u_t^*(x), \quad (1.68)$$

and under certain assumptions, it is proved that

$$\begin{aligned} \|u_t^\alpha - u_t^*\|_{L^2} &\leq \frac{C}{\alpha}, \\ \mathbb{E}\left[|Y - \alpha \int c\sigma(w \cdot x)\rho_t^\alpha(dw)|^2\right] &\leq \left(\|u_t^*\|_{L^2} + \frac{C}{\alpha}\right)^2. \end{aligned} \quad (1.69)$$

Furthermore, the convergence of the rescaled residual dynamics to its limit is uniform:

$$\lim_{\alpha \rightarrow +\infty} \sup_{t \in [0, T]} \|u_t^\alpha - u_t^*\|_{L^2} = 0. \quad (1.70)$$

Such an analysis framework is not limited to standard regression problems. It can be generalized to other learning tasks, for instance, deep reinforcement learning. In [141], the mean-field limit of online Q-learning algorithm [145] is studied. Consider a single-layer Q-network

$$Q_\theta^N(x, a) := \frac{1}{\sqrt{N}} \sum_{i=1}^N c^i \sigma(w^i \cdot (x, a)) \quad (1.71)$$

which learns to minimize the Bellman error

$$L^N(\theta) := \sum_{(x, a) \in \mathcal{X} \times \mathcal{A}} \left[(Y(x, a) - Q_\theta^N(x, a))^2 \right] \pi(x, a) \quad (1.72)$$

where $Y(x, a) = r(x, a) + \gamma \sum_{x' \in \mathcal{X}} \max_{a' \in \mathcal{A}} Q_\theta^N(x', a') p(x'|x, a)$ and π is a probability measure on $\mathcal{X} \times \mathcal{A}$. With online Q-learning, neural network Q is updated by

$$\theta_{k+1} = \theta_k + \alpha_k^N \left(r(x_k, a_k) + \gamma \max_{a' \in \mathcal{A}} Q_{\theta_k}^N(x_{k+1}, a') - Q_{\theta_k}^N(x_k, a_k) \right) \nabla_\theta Q_{\theta_k}(x_k, a_k). \quad (1.73)$$

Similar to (1.56) and (1.58) for the regression case, if one denotes $\nu_k^N = \frac{1}{N} \sum_{i=1}^N \delta_{c_k^i, w_k^i}$, $\mu_t^N = \nu_{[Nt]}^N$ and

$$h_t^N(x, a) := Q_{\theta_{[Nt]}^N}(x, a), \quad (1.74)$$

then (μ_t^N, h_t^N) converges in distribution in the Skorokhod space $D_E([0, T])$ for $E = \mathcal{M}(\mathbb{R}^{d+1}) \times \mathbb{R}^{|\mathcal{X} \times \mathcal{A}|}$ as $N \rightarrow \infty$, to the limit process (μ_t, h_t) which satisfies the random limit ODE:

$$\begin{aligned} h_t(x, a) &= h_0(x, a) + \\ &\int_0^t \sum_{(x', a') \in \mathcal{X} \times \mathcal{A}} \pi(x', a') A_{x, a; x', a'} \left(r(x', a') + \gamma \sum_{z \in \mathcal{X}} \max_{a'' \in \mathcal{A}} h_s(z, a'') p(z | x', a') - h_s(x', a') \right) ds \\ h_0(x, a) &= \mathcal{G}(x, a) \\ \langle f, \mu_t \rangle &= \langle f, \mu_0 \rangle \end{aligned} \tag{1.75}$$

for all $f \in C_b^2(\mathbb{R}^{d+1})$, where A is a positive definite matrix such that for $s, s' \in \mathcal{X} \times \mathcal{A}$ it has entry

$$A_{s, s'} = \alpha \langle \sigma(w \cdot s) \sigma(w \cdot s'), \mu_0 \rangle + \alpha \langle c^2 \sigma'(w \cdot s) \sigma'(w \cdot s') s \cdot s', \mu_0 \rangle. \tag{1.76}$$

In (1.75), $\mu_t = \mu_0$ in the weak sense. This is due to the neural network's scaling factor $\frac{1}{\sqrt{N}}$ and its training mechanism. From (1.75) it is then proved under some conditions that the neural network approximator Q learns the true value function in its limit:

$$\lim_{t \rightarrow \infty} \sup_{(x, a) \in \mathcal{X} \times \mathcal{A}} |h_t(x, a) - V(x, a)| = 0. \tag{1.77}$$

In Chapters 2 and 3, we study the mean-field limit of neural network approximators for the case where they learn to approximate solutions of PDEs. There, we typically study convergence in a gradient descent setting, which leads to a simplified analogy to SGD results introduced in this subsection. We show that with proper scaling, the training dynamics of learning to solve PDEs converge to its limit characterized by an infinite-dimensional ODE and that the solution of the ODE converges to the global minimizer one seeks to find.

1.3.3 Kernel methods

To start with, we consider a standard linear regression problem from $X = [x_1, \dots, x_N]$ to $Y = [y_1, \dots, y_N]$:

$$(\text{LR}) \quad \min_w \|Y - wX\|_2^2 = \sum_{i=1}^N \|y_i - w \cdot x_i\|^2. \tag{1.78}$$

The global minimizer for the problem is $\hat{w} = YX^\top (XX^\top)^{-1}$ and consequently the optimal predictor is

$$\hat{y}(x) = \hat{w} \cdot x = YX^\top (XX^\top)^{-1} x. \tag{1.79}$$

Now consider the kernel regression problem on the same data set $\{(x_i, y_i)\}_{i=1}^N$:

$$(KR) \quad \min_w \left\{ \sum_{i=1}^N \|y_i - \langle w, \phi(x_i) \rangle_H\|^2 \right\} \quad (1.80)$$

where $\phi : \mathbb{R}^d \rightarrow H$ lifts data x into a Hilbert space H , often a feature space. If we denote $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ be a kernel function such that $K(x, x') = \langle \phi(x), \phi(x') \rangle_H$, then K induces a positive semi-definite matrix on the data set $\hat{K}$ with each of its entries $\hat{K}(i, j) = K(x_i, x_j)$. Similar to the linear regression case, one obtains $\hat{w} = \sum [y \hat{K}^\top]_i \phi(x_i)$ and the predictor is

$$\hat{y}(x) = Y \hat{K}^\top \hat{K}(X, x) \quad (1.81)$$

where $\hat{K}(X, x) = [K(x_1, x), \dots, K(x_n, x)]^\top$. Ordinary linear regression can be seen as a special case of kernel regression where ϕ is simply the identity. Going beyond classic regression settings, kernels also characterize the behavior of neural networks. Intuitively, in a neural network regression problem

$$(NN) \quad \min_c \left\{ \sum_{i=1}^N \|y_i - \varphi_\theta(x_i)\|^2 \right\} \quad (1.82)$$

where $\varphi_\theta := \frac{1}{\sqrt{N}} \sum_{i=1}^N c^i \sigma(w^i \cdot x)$, the predictor can be written down in the form with kernel $K(x, x') = \frac{1}{N} \sum_{i=1}^N \sigma(w_i \cdot x) \sigma(w_i \cdot x')$. In practice, as one usually not only optimizes over c but also over w , the kernel is a bit more complicated than that derived above, and does not remain a constant as training goes on.

We now formally introduce the **neural tangent kernel** (NTK) [55]. Consider a fully-connected FNN α_θ with L layers, n_L units at each layer:

$$\begin{aligned} \alpha^0(x; \theta) &= x \\ \alpha^l(x; \theta) &= \frac{1}{\sqrt{n_l}} W^l \sigma(\alpha^{l-1}(x; \theta)) + \beta b^l. \end{aligned} \quad (1.83)$$

We initialize each of its parameters by i.i.d $\mathcal{N}(0, 1)$ random variables. It admits $P = \sum_{l=0}^{L-1} (n_l + 1) n_{l+1}$ parameters, and its *neural tangent kernel* is defined by

$$K^L(\theta)(x, x') = \sum_{i=1}^P \partial_{\theta_i} \alpha_\theta(x) \otimes \partial_{\theta_i} \alpha_\theta(x'). \quad (1.84)$$

It is then proved that as the width of each layer $n_1, \dots, n_L \rightarrow \infty$, the NTK at initialization (training time $k = 0$) converges to a *deterministic* limit kernel in probability:

$$K^L(0) \rightarrow K_\infty^L \otimes Id_{n_l} \quad (1.85)$$

where the scalar kernel K_∞^L is recursively given by

$$\begin{aligned} K_\infty^1(x, x') &= \Sigma^1(x, x') \\ K_\infty^{m+1}(x, x') &= \Sigma^{k+1}(x, x') + K_\infty^m(x, x') \hat{\Sigma}^{m+1}(x, x'), \end{aligned} \quad (1.86)$$

with

$$\begin{aligned} \Sigma^1(x, x') &= \frac{1}{n_0} x \cdot x' + \beta^2, \\ \Sigma^{m+1}(x, x') &= \mathbb{E}_{f \sim \mathcal{N}(0, \Sigma^m)} \left[\sigma(f(x)) \sigma(f(x')) \right] + \beta^2, \\ \bar{\Sigma}^{m+1}(x, x') &= \mathbb{E}_{f \sim \mathcal{N}(0, \Sigma^m)} \left[\sigma'(f(x)) \sigma'(f(x')) \right]. \end{aligned} \quad (1.87)$$

In addition, under certain boundedness conditions, it is proved that, uniformly for any training time $0 \leq t \leq T$, the NTK at time t remains as the same constant kernel:

$$K^L(t) \rightarrow K_\infty^L \otimes Id_{n_l}. \quad (1.88)$$

The limit ODE in (1.75) for deep Q-learning also coincides with this result, in which A represents the NTK's limit as a constant kernel. In Chapter 3, we develop the NTK limit for PDE-solving tasks (DGM & PINNs) under a single-layer setting. We show that the kernel function for PDE-solving NTK is given by

$$S(x, x') = \mathbb{E}_{\theta \sim \mu} \left[\nabla_\theta [\mathcal{L}(c\sigma(w \cdot x + b))] \cdot \nabla_\theta [\mathcal{L}(c\sigma(w \cdot x' + b))] \right] \quad (1.89)$$

where the PDE operator $\mathcal{L}$ shows up and fundamentally changes the form of the NTK together with its representation properties. To the best of the author's knowledge, this is the first time a universal NTK form is proposed for characterizing neural network's evolution of solving general linear PDE problems.

1.3.4 Analysis in Banach spaces and Hilbert spaces

In Chapters 2 and 3, we use Sobolev spaces defined for a compact domain $\Omega \subset \mathbb{R}^d$

$$\mathcal{H}^p = \left\{ f \in L^2(\Omega, \mu) : \|f\|_{\mathcal{H}^p} := \left(\sum_{|\alpha| \leq p} \|D^\alpha f\|_{L^2} \right) < \infty \right\}, \quad (1.90)$$

as a basic tool for analyzing the solutions of PDEs. For properties of $\mathcal{H}^2$ and $\mathcal{H}_0^1$ spaces, we refer readers to [1]. Also in Chapter 2, we study monotone PDE operators, where for any $f_1, f_2 \in \mathcal{H}^2(\Omega) \cap \mathcal{H}_0^1(\Omega)$ the operator $\mathcal{L}$ satisfies

$$\langle f_1 - f_2, \mathcal{L}f_1 - \mathcal{L}f_2 \rangle_{L^2} \leq -\gamma \|f_1 - f_2\|_{L^2}^2 \quad (1.91)$$

for a constant $\gamma > 0$. [134] serves as an excellent reference for monotone operators and their properties.

In Chapters 2 and 3, we make use of Hilbert–Schmidt integral operators:

$$\mathcal{T}f(x) := \int_{\Omega} k(x, z)f(z)\mu(dz) \quad (1.92)$$

induced by a Hilbert–Schmidt kernel k where

$$\int_{\Omega} \int_{\Omega} |k(x, z)|^2 \mu(dx)\mu(dz) < \infty. \quad (1.93)$$

As k is symmetric, $\mathcal{T}$ is self-adjoint and hence the spectral theorem applies. For properties of operator $\mathcal{T}$ and its orthogonal basis, see [19]. Finally, in the remaining chapters, we use function-valued ODEs to describe the limit behavior of neural networks. For ODEs in Banach or Hilbert spaces, we refer readers to [71].

1.4 Layout and main aims

The remainder of this thesis is organized as follows. Chapter 2 is based on [18]. We develop a new numerical method for solving elliptic-type PDEs by adapting the Q-learning algorithm in reinforcement learning. To solve PDEs with Dirichlet boundary condition, our “Q-PDE” algorithm is mesh-free and therefore has the potential to overcome the curse of dimensionality. Using a neural tangent kernel (NTK) approach, we prove that the neural network approximator for the PDE solution, trained with the Q-PDE algorithm, converges to the trajectory of an infinite-dimensional ordinary differential equation (ODE) as the number of hidden units $\rightarrow \infty$. For monotone PDEs (i.e. those given by monotone operators, which may be nonlinear), despite the lack of a spectral gap in the NTK, we then prove that the limit neural network, which satisfies the infinite-dimensional ODE, ergodically converges in L^2 to the PDE solution as the training time $\rightarrow \infty$. More generally, we can prove that any fixed point of the wide-network limit for the Q-PDE algorithm is a solution of the PDE (not necessarily under the monotone condition). The numerical performance of the Q-PDE algorithm is studied for several elliptic PDEs.

Chapter 3 is based on [59]. We prove global convergence for one of the commonly used deep learning algorithms for solving PDEs, the Deep Galerkin Method (DGM). DGM trains a neural network approximator to solve the PDE using stochastic gradient descent. We prove that, as the number of hidden units in the single-layer network goes to infinity (i.e., in the “wide network limit”), the trained neural network converges to the solution of an infinite-dimensional linear ordinary differential equation

(ODE). The PDE residual of the limiting approximator converges to zero as the training time $\rightarrow \infty$. Under mild assumptions, this convergence also implies that the neural network approximator converges to the solution of the PDE. A closely related class of deep learning methods for PDEs is Physics Informed Neural Networks (PINNs). Using the same mathematical techniques, we can prove a similar global convergence result for the PINN neural network approximators. Both proofs require analyzing a kernel function in the limit ODE governing the evolution of the limit neural network approximator. A key technical challenge is that the kernel function, which is a composition of the PDE operator and the neural tangent kernel (NTK) operator, lacks a spectral gap, therefore requiring a careful analysis of its properties.

Chapter 4 consists of a discussion about solving PDEs coming from control problems. We show that as certain non-linear equations can be solved by solving a set of linear equations and updating the control iteratively, neural networks can be applied to represent and update the control. In the first half of this chapter, we show that neural networks can learn to achieve a pointwise maximization target with our objective function design, which is critical for single-step updates in policy iteration. In its second half, we also provide a rigorous analysis of the continuum analogy of the online actor-critic algorithms, where the actor network representing the control and the critic network approximating the value function are trained simultaneously.

Chapter 5 concludes the thesis and contains thoughts on possible future works.

1.5 Statement of originality

Chapter 2 is based on collaborative work with the author’s supervisors, Prof. Samuel N. Cohen and Prof. Justin Sirignano, in which the author was responsible for the bulk of the mathematical results, numerical implementation, and presentation of the work. The rest chapters are the author’s independent work under the supervision of the author’s supervisors.

Chapter 2

Q-PDE algorithm

In this chapter, we train a neural network to solve a PDE, using a different training algorithm, for which we give a rigorous proof of the convergence to a NTK limiting regime (as the width of the neural network increases). This NTK approach yields a particularly simple dynamic under our training algorithm, allowing us to prove convergence to the solution of the PDE (as training time increases), despite the poorly behaved spectral properties of the NTK, for those PDEs given by monotone operators. Our Q-PDE algorithm is substantially different from the gradient descent algorithm studied in [55], [83], [84], since a PDE operator will be directly applied to the neural network to evaluate the parameter updates.

High dimensional partial differential equations (PDEs) are widely used in many applications in physics, engineering, and finance. It is challenging to numerically solve high-dimensional PDEs, as traditional finite difference methods become computationally intractable due to the curse of dimensionality. In the past decade, deep learning has become a revolutionary tool for a number of different areas including image recognition, natural language processing, and scientific computing. The idea of solving PDEs with deep neural networks has been rapidly developed in recent years and has achieved promising performance in solving real-world problems, as reviewed in Section 1.2.1.

A number of deep-learning-based PDE solving algorithms, for instance the deep Galerkin method (DGM; [138]) and physics-informed neural networks (PINNs) ([122]), have been proposed. Inspired by the Q-learning method in reinforcement learning, we propose a new “Q-PDE” algorithm which approximates the solution of PDEs with an artificial neural network. In this paper, we prove that for monotone PDEs the Q-PDE training process gives an approximation which converges to the solution of a certain limit equation as the number of hidden units in a single-layer neural network goes to

infinity. Furthermore, we prove that the limit neural network converges strongly to the solution of the PDE that the algorithm is trying to solve.

Q-learning ([152]) is a well-known algorithm for computing the value function of the optimal policy in reinforcement learning. Deep Q-learning (DQN) ([109]), a neural-network-based Q-learning algorithm, has successfully learned to play Atari games ([110]) and subsequently has become widely-used for many applications ([163]). The Q-learning algorithm updates its value function approximator following a biased gradient estimate computed from input data samples. We propose an algorithm which is similar to Q-learning in the sense that we update the parameters of a neural network via a biased gradient flow in continuous time. As far as the authors are aware, this is the first time that a Q-learning algorithm has been developed to solve PDEs directly.

The universal approximation theorem ([24]) indicates that the family of single-layer neural networks are powerful function approximators. However, the universal approximation theorem only states that there exists a neural network which can approximate continuous functions arbitrarily well; it does not suggest how to identify the parameters for such a neural network. When the number of units in our neural network becomes large, it is, however, possible to obtain asymptotic limits, for example the “neural tangent kernel” limit ([55]), which gives a variant of the law of large numbers for infinitely wide neural networks. We will use this approach to study the performance of the biased gradient flow as a training algorithm, and show that the wide-network limit satisfies an infinite-dimensional ordinary differential equation (ODE).

We apply our Q-PDE approach to second order nonlinear PDEs with Dirichlet boundary conditions. In particular, we are able to give strong convergence results when the differential operator satisfies a strong monotonicity condition. Monotone PDEs arise in various applications, particularly in PDEs arising from stochastic modeling – the generators of ergodic stochastic processes are monotone (when evaluated with their stationary distributions), which suggests a variety of possible applications of our approach. Further, the subdifferentials of convex functionals (i.e. of maps from functions to the real line) are monotone; this suggests that monotone PDEs may be a particularly well-suited class of equations for gradient methods, as they correspond to (a generalization of) minimizations of convex functionals. We will see that, given this monotonicity assumption, we can prove that the limit Q-PDE algorithm will converge (strongly in L^2) to the solution of the monotone PDE. More generally, we can prove that any fixed point of the wide-network limit for the Q-PDE algorithm is a solution of the PDE (not necessarily under the monotone condition).

The layout of this chapter is as follows. The necessary properties of the PDEs which we study, the architecture of the neural networks we use, and the Q-PDE training algorithm we propose are presented in Section 2.1. Section 2.2 derives useful properties of the limiting system, while Section 2.3 contains a rigorous proof that the training process converges to this limit. Analysis of the limit neural network, in particular the proof of convergence to the solution of the PDE, is presented in Section 2.4. Numerical results are presented in Section 2.5.

In summary, this paper will provide the following:

- A new neural network training algorithm for second order PDEs (of the form given by Assumption 2.1.13), based on Q-learning (Algorithm 2.1.20), and designed to ensure Dirichlet boundary conditions are enforced.
- Proofs of various functional-analytic properties of the neural tangent kernel limit of this algorithm (Section 2.2).
- A rigorous convergence proof of the neural-tangent kernel limit for a wide single-layer network (Section 2.3). This limit gives a simple deterministic dynamical system characterizing the neural network approximation during training (Definition 2.3.8).
- Proof that any fixed point of the limit neural network training dynamics is a solution of the PDE (Theorem 2.4.1).
- A strong convergence proof for the limit neural network as the training time $t \rightarrow \infty$, when considering a PDE given by a monotone operator (Section 2.4).
- Numerical examples to demonstrate effectiveness of the training algorithm (Section 2.5).

A more precise summary of our mathematical results is given in Section 2.1.3.1.

2.1 The Q-PDE Algorithm

In this section, we first of all state our assumptions on the PDE and its domain. Then we describe the neural network approximator and our Q-PDE algorithm.

2.1.1 Assumptions on the PDE

We consider a class of second-order nonlinear PDEs on a bounded open domain $\Omega \subset \mathbb{R}^n$ with Dirichlet boundary conditions:

$$\begin{cases} \mathcal{L}u = 0 & \text{in } \Omega \\ u = f & \text{on } \partial\Omega. \end{cases} \quad (2.1)$$

We assume a measure ¹ μ is given on Ω , with continuous Radon–Nikodym density $d\mu/d\text{Leb}$ bounded away from 0 and ∞ (where Leb denotes Lebesgue measure). It follows that $\mu(\Omega) < \infty$ and $\mu(A) > 0$ for all open sets $A \subset \Omega$; all integrals on Ω will be taken with respect to this measure (unless otherwise indicated).

We will study strong Sobolev solutions to the PDE (2.1); that is, we are interested in solutions $u \in \mathcal{H}^2$, where

$$\mathcal{H}^p = \left\{ f \in L^2(\Omega, \mu) : \|f\|_{\mathcal{H}^p} := \left(\sum_{|\alpha| \leq p} \|D^\alpha f\|_{L^2}^2 \right)^{1/2} < \infty \right\}, \quad (2.2)$$

where Du is the weak derivative of u and where the boundary condition $u|_{\partial\Omega} = f$ is understood in the usual trace-operator sense in $\mathcal{H}^1$.

For notational simplicity, we write $\mathcal{H}_{(0)}^2 = \mathcal{H}^2 \cap \mathcal{H}_0^1$ for the subspace of functions $u \in \mathcal{H}^2$ such that $u|_{\partial\Omega} = 0$, that is, with boundary trace zero. This remains a Hilbert space under the $\mathcal{H}^2$ inner product. We write $\|\cdot\|$ (without a subscript) for the L^2 norm and similarly for the inner product. We refer to [30] or [1] for further details and the general theory of these spaces and concepts.

The class of PDEs we will consider are those for which $\mathcal{L}$, f and Ω satisfy some (strong) regularity conditions, which we now detail. We first of all present the assumptions and lemmas for the domain and the boundary condition.

Assumption 2.1.1. *The boundary $\partial\Omega$ is $C^{3,\alpha}$, for some $\alpha \in (0, 1)$, that is, three times continuously differentiable, with α -Hölder continuous third derivative.*

Assumption 2.1.2 (Auxiliary function η). *There exists a (known) function $\eta \in C_b^3(\mathbb{R}^n)$, which satisfies $0 < \eta < 1$ in Ω , and $\eta = 0$ on $\partial\Omega$. Furthermore, its first order derivative does not vanish at the boundary (that is, for $x \in \partial\Omega$ and $\mathbf{n}_x$ an outward unit normal vector at x , we have $\langle \nabla\eta(x), \mathbf{n}_x \rangle \neq 0$).*

¹For example, μ can be taken to be the Lebesgue measure on $\Omega \subset \mathbb{R}^n$, and the analytic theory we consider is essentially the same as that when using the Lebesgue measure (i.e. the Sobolev norms using μ are equivalent to those using Leb). We allow for more general μ , as this will roughly correspond to a choice of sampling scheme in our numerical algorithm, and gives us a degree of flexibility when considering the monotonicity of $\mathcal{L}$.

Using Assumptions 2.1.1 and 2.1.2, we have the following useful result, which allows us to approximate functions in $\mathcal{H}_{(0)}^2$. Its proof is given in Appendix 2.6.1.

Lemma 2.1.3. *1. The set of functions $C^3(\overline{\Omega}) \cap C_0(\overline{\Omega})$ is dense in $\mathcal{H}_{(0)}^2 = \mathcal{H}^2 \cap \mathcal{H}_0^1$ (under the $\mathcal{H}^2$ topology).*

2. For any function $u \in C^3(\overline{\Omega}) \cap C_0(\overline{\Omega})$, the function $\tilde{u} = u/\eta$ is in $C_b^2(\Omega) \subset \mathcal{H}^2$.

In addition to these assumptions on $\mathcal{L}$, we make the following assumptions on the boundary value.

Assumption 2.1.4 (Interpolation of the boundary condition function). *There exists a (known) function $\bar{f} \in \mathcal{H}^2$ such that $\bar{f}|_{\partial\Omega} = f$ (or, more precisely, if $T : \mathcal{H}^1(\Omega) \rightarrow L^2(\partial\Omega)$ is the boundary trace operator, we have $T(\bar{f}) = f$). In the rest of this paper, for notational simplicity, we identify f with its extension $\bar{f}$ defined on $\overline{\Omega}$.*

Remark 2.1.5. The existence of some $\bar{f}$ satisfying Assumption 2.1.4 is guaranteed, given a solution $u \in \mathcal{H}^2$ exists (as we could take $\bar{f} = u$). Practically, we assume not only that $\bar{f}$ exists, but that it is possible for us to use it as part of our numerical method to find u . In many cases, this is a mild and natural assumption, for instance when we have $f = 0$, where we can simply take $\bar{f} = 0$.

Now we present the assumptions of the PDE itself.

Assumption 2.1.6 (Lipschitz continuity of $\mathcal{L}$). *There exists a constant C such that for any $f_1, f_2 \in \mathcal{H}^2 \cap C^2$ and any $x \in \Omega$, the (nonlinear differential) operator $\mathcal{L}$ satisfies*

$$\begin{aligned} |\mathcal{L}f_1(x) - \mathcal{L}f_2(x)| &\leq C \left[|f_1(x) - f_2(x)| + \sum_{i=1}^n |\partial_{x_i} f_1(x) - \partial_{x_i} f_2(x)| \right. \\ &\quad \left. + \frac{1}{2} \sum_{i,j=1}^n |\partial_{x_i x_j}^2 f_1(x) - \partial_{x_i x_j}^2 f_2(x)| \right]. \end{aligned} \quad (2.3)$$

Assumption 2.1.7 (Integrability of $\mathcal{L}$ at zero). *Taking $f_0 \equiv 0$, we have $\mathcal{L}f_0 \in L^2$.*

Remark 2.1.8. Under Assumptions 2.1.6 and 2.1.7, $\mathcal{L}$ is a Lipschitz operator $\mathcal{L} : \mathcal{H}^2 \rightarrow L^2$.

Our algorithm will also make use of the following auxiliary function, which allows us to control behaviour of functions near the boundary of Ω .

In order to prove convergence of our scheme for large training time, we will particularly focus on PDEs given by strongly monotone operators (cf. [8]), which for convenience we take with the following sign convention.

Assumption 2.1.9 (Strong L^2 -monotonicity of $\mathcal{L}$). *There exists a constant $\gamma > 0$ such that for any $f_1, f_2 \in \mathcal{H}_{(0)}^2$ the operator $\mathcal{L}$ satisfies*

$$\langle f_1 - f_2, \mathcal{L}f_1 - \mathcal{L}f_2 \rangle \leq -\gamma \|f_1 - f_2\|_{L^2}^2, \quad (2.4)$$

where $\langle \cdot, \cdot \rangle$ is the $L^2(\mu)$ inner product.

For examples of such a monotone PDE operator, check (2.118), or, for instance, the following d -dimensional equation

$$1 + \|x\|^2 - df(x) + \mathcal{A}_{OU}f(x) = 0, \quad x \in B_1(0) \quad (2.5)$$

with boundary condition $f = 0$, where $B_1(0)$ is the unit ball, and $\mathcal{A}_{OU}$ is the infinitesimal generator of a d -dimensional OU process

$$dX_t = (a - X_t)dt + \sigma dW_t^d \quad (2.6)$$

where W^d is a d -dimensional Brownian motion, $a \in B_1(0)$ and $\sigma \neq 0$. While these assumptions are somewhat restrictive, they are general enough to allow for the case $\mathcal{L}u = \mathcal{A}u - \gamma u + r$, for r any L^2 function, where $\mathcal{A}$ is the generator of a sufficiently nice Feller process X and $\gamma > 0$, for an appropriate choice of sampling distribution μ ; see Section 2.6.7 for further discussion. In this case, the solution u can be expressed (using the Feynman–Kac theorem) in terms of the stochastic process,

$$u(x) = \mathbb{E} \left[e^{-\gamma\tau} f(X_\tau) + \int_0^\tau e^{-\gamma t} r(X_t) dt \mid X_0 = x \right] \quad (2.7)$$

where $\tau = \inf\{t \geq 0 : X_t \notin \Omega\}$.

Further, these assumptions are also sufficiently general to allow some nonlinear PDEs of interest, for example Hamilton–Jacobi–Bellman equations under a Cordes condition, as discussed by [143]. The assumption of monotonicity is also connected to Lyapunov stability analysis, as it is easy to check that strong monotonicity is equivalent to stating that $v \mapsto \frac{1}{2}\|v\|_{L^2}^2$ is a Lyapunov function for the infinite-dimensional dynamical system $dv/dt = \mathcal{L}v + \gamma v$. For more general discussion of monotone operators, and their connection to analysis of the traditional Galerkin method, we refer to [160].

Our final assumption on the PDE is that a solution exists.

Assumption 2.1.10 (Existence of solutions). *There exists a (unique) solution $u \in \mathcal{H}^2$ to (2.1).*

Remark 2.1.11. We note that, assuming strong monotonicity (Assumption 2.1.9), if a solution exists then it is guaranteed to be unique: if u, u' are two solutions, then $\mathcal{L}u = \mathcal{L}u'$, hence $0 = \langle u - u', \mathcal{L}u - \mathcal{L}u' \rangle \leq -\gamma \|u - u'\|^2$, and so $u = u'$ almost everywhere.

While existence of solutions in $\mathcal{H}^2$ is a strong assumption, it is often satisfied for weak solutions to elliptic equations, given the well-known elliptic regularity results which ensure solutions are sufficiently smooth.

Remark 2.1.12. The algorithm we present below can easily be extended to higher-order PDEs. To do this, further smoothness assumptions (on the activation function σ , continuation value f and auxiliary function η) and higher moments of the initial weights w^i in the neural network are needed, and one argument (Lemma 2.1.3) needs to be extended using alternative approaches. In this paper, for the sake of concreteness (and to avoid unduly long derivations), we present our results for PDEs up to second-order; the details of the extension to the general case are left as a tedious exercise for the reader.

Summary of assumptions 2.1.13. For ease of reference, we now summarize the assumptions we have made on our PDE:

1. $\mathcal{L}$ is a nonlinear second-order differential operator, Lipschitz (as a map $\mathcal{H}^2(\Omega) \rightarrow L^2(\Omega)$), square integrable at zero (Assumptions 2.1.6, 2.1.7).
2. The boundary value f has a known continuation to a function in $\mathcal{H}^2(\Omega)$ (Assumption 2.1.4).
3. $\Omega \subset \mathbb{R}^n$ is bounded and open, and has a $C^{3,\alpha}$ boundary $\partial\Omega$, which is the zero level set of a known auxiliary function $\eta \in C_b^3(\mathbb{R}^n)$, and η has nonzero derivative on the boundary (Assumptions 2.1.1, 2.1.2)
4. $\mathcal{L}$ is strongly monotone, and there exists an $\mathcal{H}^2(\Omega)$ solution to the PDE (Assumptions 2.1.9, 2.1.10).

We emphasise that only parts 1–3 of this assumption will be used, except in Section 2.4, where the strong monotonicity and existence of solutions will also be needed.

2.1.2 Neural Network Configuration

We will approximate the solution to the PDE (2.1) with a neural network, which will be trained with a deep Q-learning inspired algorithm. In particular, we will study a

single-layer neural network with N hidden units:

$$S^N(x; \theta) = \frac{1}{N^\beta} \sum_{i=1}^N c^i \sigma(w^i \cdot x + b^i), \quad (2.8)$$

where the scaling factor $\beta \in (\frac{1}{2}, 1)$ and $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is a non-linear scalar function. The parameters θ are defined as $\theta := \{c^i, w^i, b^i\}_{i=1, \dots, N}$ where $c^i, b^i \in \mathbb{R}$, and $w^i \in \mathbb{R}^n$. The function η can then be used to design a neural network model which automatically satisfies the boundary condition, introduced by [100]:

$$Q^N(x; \theta) := S^N(x; \theta) \eta(x) + f(x). \quad (2.9)$$

Assumption 2.1.14 (Activation function). *The activation function $\sigma \in C_b^4(\mathbb{R})$ is non-constant, where C_b^k is the space of functions with k -th order bounded continuous derivatives.*

Remark 2.1.15. This assumption coincides with the condition of Theorem 4 of [48], and guarantees that the functions σ generate neural nets which are dense in the Sobolev space $\mathcal{H}^2$. (Hornik shows, under this condition, the stronger result of density in $\mathcal{H}^4$; we focus on $\mathcal{H}^2$, but require additional bounded derivatives as part of our proof.) We will particularly make use of this to establish Lemma 2.2.7.

Before training begins (i.e. at $t = 0$), the neural network parameters are randomly initialized. The random initialization satisfies the assumptions described below.

Assumption 2.1.16 (Neural network initialization). *The initialization of the parameters θ_0 , for all $i \in \{1, 2, \dots, N\}$, satisfies:*

- *The parameters c_0^i, w_0^i, b_0^i are independent random variables.*
- *The random variables c_0^i are bounded, $|c_0^i| < K_0$, and $\mathbb{E}[c_0^i] = 0$.*
- *The distribution of the random variables w_0^i, b_0^i has full support. That is, for any open set $D \subseteq \mathbb{R}^{n+1}$, we have $\mathbb{P}((w_0^i, b_0^i) \in D) > 0$.*
- *For any indices $i, k \in \{1, \dots, N\}$, we have $\mathbb{E}[|(w_0^i)_k|^3] < \infty$, $\mathbb{E}[|b_0^i|] < \infty$.*

2.1.3 Training Algorithm

We now present our algorithm for training the neural network $Q^N(x; \theta)$ to solve the PDE (2.1). At training time t , the parameters of the neural network are denoted $\theta_t = \{c_t^i, w_t^i, b_t^i\}_{i=1, \dots, N}$. For simplicity, we denote $S^N(\cdot; \theta_t)$ as S_t^N , and $Q^N(\cdot; \theta_t)$ as Q_t^N . Then,

$$Q_t^N = S_t^N \eta + f. \quad (2.10)$$

Our goal is to design an algorithm to train the approximator Q^N to find the solution of the PDE. One approach, see for instance [138], is to train the model Q^N to minimize the average PDE residual:

$$\int_{\Omega} \left| \mathcal{L}Q^N(x) \right|^2 \mu(dx). \quad (2.11)$$

To improve integrability, we will smoothly truncate this objective function, and use the result to motivate our training algorithm.

Definition 2.1.17 (Smooth truncation function). *The functions $\{\psi^N\}_{N \in \mathbb{N}}$ are a family of smooth truncation functions if, for some $\delta \in (0, \frac{1-\beta}{2})$,*

- $\psi^N \in \mathbb{C}_b^2(\mathbb{R})$ is increasing on $\mathbb{R}$.
- $|\psi^N|$ is bounded by $2N^\delta$.
- $\psi^N(x) = x$ for $x \in [-N^\delta, N^\delta]$.
- $|(\psi^N)'| \leq 1$ on $\mathbb{R}$.
- $F^N := (\psi^N) \cdot (\psi^N)'$ is uniformly Lipschitz continuous on $\mathbb{R}$ for all $N \in \mathbb{N}$.

For simplicity, we will usually describe the family ψ^N as a (smooth) truncation function. A simple example of a truncation function is as follows: take $\delta = (1-\beta)/4 > 0$ and

$$\psi^N(x) = \int_0^x g^N(y) dy, \quad \text{where} \quad g^N(x) = \begin{cases} e^{-(|x| - N^\delta)^2} & \text{if } |x| \geq N^\delta, \\ 1 & \text{if } |x| < N^\delta. \end{cases} \quad (2.12)$$

Remark 2.1.18. It is easy to see that, for any smooth truncation function, the related function $F^N(x) = (\psi^N) \cdot (\psi^N)'$ satisfies

$$|F^N(x) - x| \leq 2|x| \mathbb{1}_{\{|x| \geq N^\delta\}} \text{ for all } x \in \mathbb{R}. \quad (2.13)$$

We will use a truncation function ψ^N to modify (2.11), and will consider minimizing the truncated objective:

$$\int_{\Omega} [\psi^N(\mathcal{L}Q^N(x))]^2 \mu(dx). \quad (2.14)$$

To minimise a loss function like (2.14), one can apply gradient-descent based methods.

However, in practice, computing the gradient of $\mathcal{L}(Q_t^N(\cdot; \theta))$ with respect to θ does not lead to an algorithm permitting simple analysis, as natural properties of the differential operator $\mathcal{L}$ are not directly preserved. Instead, we introduce a biased gradient estimator as follows:

Definition 2.1.19. *The sequence of functions*

$$G^N(\theta_t) = - \int_{\Omega} \left[\psi^N(\mathcal{L}Q_t^N(x)) \cdot (\psi^{N'})(\mathcal{L}Q_t^N(x)) \right] \nabla_{\theta} Q_t^N(x) \mu(dx) \quad (2.15)$$

are called the biased gradient estimators for the truncated loss. These can be approximated using Monte Carlo sampling, as

$$\hat{G}_M^N(\theta_t) = - \frac{1}{M} \sum_{i=1}^M \left[\psi^N(\mathcal{L}Q_t^N(x_i)) \cdot (\psi^{N'})(\mathcal{L}Q_t^N(x_i)) \right] \nabla_{\theta} Q_t^N(x_i) \quad (2.16)$$

where x_i are independent samples from the distribution μ .

Our analysis will focus on the biased gradient estimator G^N , however in our numerical implementation (in order to avoid the curse of dimensionality) we will use the Monte Carlo estimates $\hat{G}_M^N$ for large M .

We shall see that this biased gradient is a continuous-space, PDE analog of the classic Q-learning algorithm. In summary, our “Q-PDE” algorithm for solving PDEs is:

Algorithm 2.1.20 (Q-PDE Algorithm). We fix a family of smooth truncation functions $\{\psi^N\}_{N \in \mathbb{N}}$ and, for each value of $N \in \mathbb{N}$, we proceed as follows:

1. Randomly initialize the parameters θ_0 , as specified in Assumption 2.1.16.
2. Train the neural network via the *biased gradient flow*

$$\frac{d\theta_t}{dt} = -\alpha_t^N G^N(\theta_t). \quad (2.17)$$

with G^N as in (2.15) and learning rate

$$\alpha_t^N = \alpha^N = \alpha N^{2\beta-1}, \quad (2.18)$$

where α is a positive constant.

The approximate solution to the PDE at training time t is Q_t^N , as defined by (2.9).

In the remainder of this paper, we will study this Q-PDE algorithm for solving PDEs both mathematically and numerically. First, we prove that the neural network model Q_t^N converges to the solution of an infinite-dimensional ODE as the number of hidden units $N \rightarrow \infty$. Then, we study the limit ODE, and prove that it converges to the true solution of the PDE as the training time $t \rightarrow \infty$. Finally, in Section 2.5, we demonstrate numerically that the algorithm performs well in practice with a finite number of hidden units. For high dimensional PDEs, it is impossible to form mesh-grids to apply conventional finite difference methods. However, our method is mesh-free. The crucial feature for implementing our algorithm in practice is to evaluate the integral term G^N . As discussed, we can do this using the Monte Carlo estimate $\hat{G}_M^N$; therefore our algorithm has the potential to solve high-dimensional PDEs.

Remark 2.1.21. It is worth observing that, given this choice of biased gradient flow, there is generally no guarantee that our algorithm corresponds to the gradient descent direction applied to any potential function. The key advantage of the use of this biased gradient is that it separates the derivatives in θ from the differential operator $\mathcal{L}$, which allows the underlying properties of the PDE to be preserved more simply. We will see that this results in particularly simple dynamics in the wide-network limit.

2.1.3.1 Main results

Our main mathematical results are the following:

Theorem 2.1.22 (cf. Theorem 2.3.10). *For a single-layer neural-network PDE approximator Q^N with N units trained using the Q-PDE algorithm, at each training time $t \geq 0$, as $N \rightarrow \infty$, in expectation Q_t^N converges to Q_t (in $\mathcal{H}^2$) given by a deterministic $\mathcal{H}^2$ -valued ordinary differential equation*

$$\frac{dQ_t}{dt} = \alpha \mathcal{B}\mathcal{L}(Q_t) \quad \text{with} \quad Q_0 = f.$$

Here, $\mathcal{B}$ is the integral operator given by

$$(\mathcal{B}v)(y) = \int_{\Omega} B(x, y)v(x)\mu(dx)$$

with $B(x, y) = \eta(x)\eta(y)A(x, y)$ for $x, y \in \Omega$ and

$$A(x, y) = \mathbb{E}_{c, w, b} \left[\sigma(w \cdot x + b) \sigma(w \cdot y + b) + c^2 \sigma'(w \cdot x + b) \sigma'(w \cdot y + b) (x \cdot y + 1) \right]$$

where the expectation is taken with respect to the distribution of the random initialization of neural network parametrization c , w , and b satisfying Assumption 2.1.16.

Theorem 2.1.23. *The limiting dynamical system Q_t has the following convergence properties:*

- *If Q_t converges in $\mathcal{H}^1$ to a fixed point Q , then Q is the solution to the PDE (2.1). (Theorem 2.4.1)*
- *If $\mathcal{L}$ is a monotone operator, and u is the solution to the PDE (2.1), then for almost every sequence $t_k \rightarrow \infty$, we have $Q_{t_k} \rightarrow u$ in L^2 . (Theorem 2.4.2)*
- *If $\mathcal{L}$ is a Gateaux differentiable monotone operator then, for almost every sequence $t_k \rightarrow \infty$, we have $\mathcal{BL}Q_{t_k} \rightarrow 0$. (Theorem 2.4.3)*

2.1.3.2 Similarity to Q-learning

Our training algorithm's biased gradient flow is inspired by the classic Q-learning algorithm in a reinforcement learning setting. The Q-learning algorithm approximates the value function of the optimal policy for Markov decision problems (MDPs). It seeks to minimize the error of a parametric approximator, such as a neural network, $R(\cdot, \cdot; \theta)$:

$$L(\theta) = \sum_{(s,a) \in \mathcal{S} \times \mathcal{X}} [Y(s, a; \theta) - R(s, a; \theta)]^2 \pi(s, a). \quad (2.19)$$

Here $\mathcal{S}$ and $\mathcal{X}$ are the finite state and action spaces of the MDP and π is a probability mass function which is strictly positive for every $(s, a) \in \mathcal{S} \times \mathcal{X}$. Y is the target Bellman function

$$Y(s, a; \theta) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} \max_{a' \in \mathcal{X}} R(s', a'; \theta) p(s'|s, a). \quad (2.20)$$

Q-learning updates the parametric approximator R via $\theta_{k+1} = \theta_k + \alpha_k^N U_k$ where U is a biased estimator of the gradient of $(Y - R)^2$. Since the transition density $p(s'|s, a)$ is unknown, $Y - R$ is estimated using random samples from the Markov chain (s_k, a_k) :

$$\tilde{Y}(s_k, a_k) - R(s_k, a_k; \theta_k) := r(s_k, a_k) + \gamma \max_{a' \in \mathcal{X}} R(s_{k+1}, a'; \theta_k) - R(s_k, a_k; \theta_k). \quad (2.21)$$

The Q-learning algorithm treats Y (and $\tilde{Y}$) as a constant and takes a derivative only with respect to the last term in (2.21). The Q-learning update direction U_k is:

$$-U_k =: \left(\tilde{Y}(s_k, a_k) - R(s_k, a_k; \theta_k) \right) \nabla_{\theta_k} [-R(s_k, a_k; \theta_k)] \quad (2.22)$$

We emphasize that the term $\nabla_{\theta} \tilde{Y}$ is not included, which means U_k is a biased estimator for the direction of steepest descent.

Returning to the objective function for the Q-PDE algorithm, the gradient of the loss function (2.14) is

$$\int_{\Omega} \left[\psi^N(\mathcal{L}Q_t^N(x)) \psi^{N'}(\mathcal{L}Q_t^N(x)) \right] \nabla_{\theta} \mathcal{L}Q_t^N(x) \mu(dx). \quad (2.23)$$

The update direction G^N in (2.15) differs from (2.23), as $\nabla_{\theta} \mathcal{L}Q_t^N(x)$ has been replaced with $\nabla_{\theta}(-Q_t^N(x))$. This is, conceptually, the same update direction as used in the Q-learning algorithm.

2.2 Preliminary analysis of the training algorithm

We begin our analysis of the training algorithm (2.17) by proving several useful bounds, in particular, that the neural network parameters c_t and w_t are bounded in expectation. The proof is provided in Appendix 2.6.2. We recall that n is the dimension of our underlying space, that is, $\Omega \subset \mathbb{R}^n$.

Lemma 2.2.1 (Boundedness of parameters). *For all $T > 0$, there exists a deterministic constant $C > 0$ such that, for all $N \in \mathbb{N}$, all $0 \leq t \leq T$, all $i \in \{1, 2, \dots, N\}$, and all $k \in \{1, 2, \dots, n\}$,*

$$|c_t^i| < C, \quad \mathbb{E}|(w_t^i)_k| < C, \quad \text{and} \quad \mathbb{E}|b_t^i| < C. \quad (2.24)$$

2.2.0.1 Dynamics of Q_t^N

We next consider the evolution of the output of the neural network Q_t^N as a function of the training time t . Define B_t^N as the symmetric, positive semi-definite kernel function

$$B_t^N(x, y) = \eta(x) \eta(y) A_t^N(x, y), \quad (2.25)$$

where A_t^N is the *neural tangent kernel* ([55])

$$\begin{aligned} A_t^N(x, y) &= \left(\nabla_{\theta} S_t^N(x) \right)^{\top} \left(\nabla_{\theta} S_t^N(y) \right) \\ &= \frac{1}{N} \sum_{i=1}^N \left[\sigma(w_t^i \cdot x + b_t^i) \sigma(w_t^i \cdot y + b_t^i) \right. \\ &\quad \left. + c_t^{i^2} \sigma'(w_t^i \cdot x + b_t^i) \sigma'(w_t^i \cdot y + b_t^i) (x \cdot y + 1) \right]. \end{aligned} \quad (2.26)$$

For any $y \in \overline{\Omega}$, we can derive the evolution of value $Q_t^N(y)$ using the chain rule:

$$\frac{dQ_t^N(y)}{dt} = \nabla_{\theta} Q_t^N(y) \cdot \frac{d\theta}{dt}. \quad (2.27)$$

The RHS of (2.27) can be evaluated using (2.9), (2.17), and Fubini's theorem, which yields

$$\frac{dQ_t^N(y)}{dt} = \alpha \int_{\Omega} F^N(\mathcal{L}Q_t^N(x)) B_t^N(x, y) \mu(dx), \quad (2.28)$$

where F^N is as in Definition 2.1.17.

2.2.0.2 Limit kernel

Equation (2.28) shows that the kernel B^N has a key role in the dynamics of Q^N . We now characterize the limit of B^N as $N \rightarrow \infty$. In Section 2.3.1, we will study this convergence in detail.

At time $t = 0$, the parameters c_0^i, w_0^i, b_0^i are independently sampled. Therefore, for each $(x, y) \in \Omega$, by the strong law of large numbers $A_0^N(x, y)$ converges almost surely to $A(x, y)$, where

$$A(x, y) = \mathbb{E}_{c, w, b} [\sigma(w \cdot x + b) \sigma(w \cdot y + b) + c^2 \sigma'(w \cdot x + b) \sigma'(w \cdot y + b) (x \cdot y + 1)]. \quad (2.29)$$

It follows that $B_0^N(x, y)$ converges almost surely to $B(x, y)$:

$$\lim_{N \rightarrow \infty} B_0^N(x, y) = B(x, y) := \eta(x) \eta(y) A(x, y). \quad (2.30)$$

We can write

$$B(x, y) = \eta(x) \eta(y) \mathbb{E}_{c, w, b} [\sigma(w \cdot x + b) \sigma(w \cdot y + b) + c^2 U(x) \cdot U(y)], \quad (2.31)$$

where the (w, b) -dependent vector function U is given by $U(x) = \sigma'(w \cdot x + b)(x + 1/\sqrt{n})$. As $\sigma \in C_b^4(\mathbb{R})$, we know that A, A^N, B and B^N are all uniformly continuous in (x, y) , so the above a.s. convergences hold for all $(x, y) \in \Omega$ simultaneously.

Lemma 2.2.2. *There exists a constant $M > 0$ such that, for any $(x, y) \in \bar{\Omega} \times \bar{\Omega}$,*

$$|B(x, y)| + \sum_k |\partial_{y_k} B(x, y)| + \sum_{k, l} |\partial_{y_k y_l}^2 B(x, y)| \leq M. \quad (2.32)$$

Proof. Taking B as defined in (2.30), as $\eta \in C_b^3$, $\sigma \in C_b^4$ and each term of $\{c_t^i\}_{i \in \mathbb{N}}$ is uniformly bounded, we have $|B(x, y)| < C$. The partial derivative of B is

$$\begin{aligned} & \partial_{y_k} B(x, y) \\ &= \eta(x) (\partial_{y_k} \eta(y)) A(x, y) + \eta(x) \eta(y) (\partial_{y_k} A(x, y)) \\ &= \eta(x) \partial_{y_k} \eta(y) A(x, y) + \eta(x) \eta(y) \mathbb{E}_{c, w, b} \left[\sigma(w \cdot x + b) \sigma'(w \cdot y + b) w_k \right. \\ & \quad \left. + c^2 \sigma'(w \cdot x + b) [w_k \sigma''(w \cdot y + b) x \cdot y + \sigma'(w \cdot y + b) x_k] \right]. \end{aligned} \quad (2.33)$$

As before, our assumptions on η , σ and $\{c_t^i\}_{i \in \mathbb{N}}$ guarantee that the terms in $\partial_{y_k} B(x, y)$ are uniformly bounded for all $x, y \in \bar{\Omega}$, so $|\partial_{y_k} B(x, y)| < C$; similarly for $\partial_{y_k y_l}^2 B(x, y)$. The result follows. $\square$

Given we have a kernel B , it is natural to define the corresponding integral operator, $v \mapsto \int_{\Omega} B(x, y)v(x)\mu(dx)$. It will be convenient to define this on a slightly larger space than L^2 , to account for the effect of the function η .

Definition 2.2.3. We define the function space $L_{\eta}^2 := \{f : \Omega \rightarrow \mathbb{R} \mid \eta f \in L^2\}$, and observe (as η is bounded) that $L^2 \subset L_{\eta}^2$.

Definition 2.2.4. The operator $\mathcal{B} : L_{\eta}^2 \rightarrow L^2$ is defined by

$$(\mathcal{B}v)(y) = \int_{\Omega} B(x, y)v(x)\mu(dx) \quad (2.34)$$

with B as in (2.30).

Lemma 2.2.5. For any $v \in L_{\eta}^2$, we have $\mathcal{B}v \in \mathcal{H}_{(0)}^2$, or more specifically, $\mathcal{B}v \in C_b^2(\bar{\Omega})$ and $(\mathcal{B}v)(y) = 0$ for all $y \in \partial\Omega$.

Proof. From the definition of $\mathcal{B}v$, we have $(\mathcal{B}v)(y) = \eta(y) \int_{\Omega} A(x, y)\eta(x)v(x)\mu(dx)$. Lemma 2.2.2 and the smoothness of σ and η clearly imply that $\mathcal{B}v \in C_b^2(\bar{\Omega}) \subset \mathcal{H}^2$. We know that $\eta(x) = 0$ for $x \in \partial\Omega$, so it is clear that $(\mathcal{B}v)(y) = 0$ for all $y \in \partial\Omega$. As $\mathcal{H}_0^1$ is the kernel (in $\mathcal{H}^1$) of the boundary trace operator, and for smooth functions the trace operator is simply evaluation of the function on the boundary, it follows that $\mathcal{B}v \in \mathcal{H}_0^1$. We conclude that $\mathcal{B}v \in \mathcal{H}_{(0)}^2 = \mathcal{H}^2 \cap \mathcal{H}_0^1$. $\square$

Lemma 2.2.6. The linear map $\mathcal{B} : L_{\eta}^2 \rightarrow \mathcal{H}^2$ is Lipschitz, in particular, there exists $C > 0$ such that, for any $v \in L_{\eta}^2$, we have $\|\mathcal{B}v\|_{\mathcal{H}^2} \leq C\|\eta \cdot v\|_{L^2}$.

Proof. By definition, with D^{α} denoting the differential operator with respect to the y argument, $\|\mathcal{B}v\|_{\mathcal{H}^2}^2 = \sum_{|\alpha| \leq 2} \|D^{\alpha}\mathcal{B}v\|_{L^2}^2$. By the boundedness of η , A and their derivatives, there exists a constant K such that for any multi-index α , $|D^{\alpha}[\eta(y)A(x, y)]| < K$ for any $x, y \in \Omega$. Hence, exchanging the order of integration (with respect to x) and differentiation (with respect to y) by the dominated convergence theorem,

$$\begin{aligned} & \left| (D^{\alpha}\mathcal{B}v)(y) \right| \\ &= \left| D^{\alpha} \left[\int_{\Omega} \eta(y)A(x, y)\eta(x)v(x)\mu(dx) \right] \right| = \left| \int_{\Omega} D^{\alpha}[\eta(y)A(x, y)]\eta(x)v(x)\mu(dx) \right| \\ &\leq \int_{\Omega} \left| D^{\alpha}[\eta(y)A(x, y)]\eta(x)v(x) \right| \mu(dx) \leq K \int_{\Omega} |\eta(x)v(x)| \mu(dx). \end{aligned} \quad (2.35)$$

By Jensen's inequality, there is a constant $C > 0$ such that

$$|(D^\alpha \mathcal{B}v)(y)|^2 \leq K^2 \left(\int_{\Omega} |\eta(x)v(x)| \mu(dx) \right)^2 \leq C \int_{\Omega} |\eta(x)v(x)|^2 \mu(dx) = C \|\eta \cdot v\|_{L^2}^2. \quad (2.36)$$

Integrating over Ω , we see that $\|D^\alpha \mathcal{B}v\|_{L^2}^2 \leq (C\mu(\Omega)) \|\eta \cdot v\|_{L^2}^2$. Summing over indices α yields the result. $\square$

Our next challenge is to prove that the image of $\mathcal{B}$ is dense in $\mathcal{H}_{(0)}^2$. This will allow us to represent the solutions to our PDE in terms of $\mathcal{B}$, and is closely related to the universal approximation theorems of [24], [48], and others.

Lemma 2.2.7. *For a function $g \in \mathcal{H}^2(\Omega)$, if the equality*

$$\int_{\Omega} \sum_{|\alpha| \leq 2} D^\alpha \sigma(w \cdot x + b) D^\alpha g(x) \mu(dx) = 0 \quad (2.37)$$

holds for all $w \in \mathbb{R}^n$, $b \in \mathbb{R}$, then $g \equiv 0$.

Proof. Since μ is a finite measure and $\sigma \in C_b^2$ (cf. Assumption 2.1.14) and non-constant, from Theorem 4 in [48] we have that the linear span of $\{\sigma(w \cdot x + b)\}_{w,b \in \mathbb{R}}$ is dense in $\mathcal{H}^2$. Hence, there is no nontrivial element of $\mathcal{H}^2$ orthogonal to $\sigma(w \cdot x + b)$ for all w, b . In other words, for $g \in \mathcal{H}^2$, if the inner product $\langle \sigma_{w,b}, g \rangle_{\mathcal{H}^2} = 0$ for all w, b , then $g = 0$, which is the stated result. $\square$

Lemma 2.2.8. *Define $S : \mathcal{H}^2 \rightarrow \mathcal{H}^2$ by $(Sh)(y) := \int_{\Omega} \sum_{|\alpha| \leq 2} D_x^\alpha h(x) D_x^\alpha A(x, y) \mu(dx)$. Then $Sh = 0$ if and only if $h = 0$.*

Proof. We first observe that, as $\sigma \in C_b^4(\mathbb{R})$, $D_x^\alpha A(x, y)$ is clearly a C_b^2 function of y for all x . Consider the inner product between Sh and h on $\mathcal{H}^2$:

$$\begin{aligned} \langle Sh, h \rangle_{\mathcal{H}^2} &= \sum_{|\alpha| \leq 2} \langle D^\alpha Sh, D^\alpha h \rangle_{L^2} \\ &= \int_{\Omega^2} \sum_{|\alpha_1| \leq 2, |\alpha_2| \leq 2} D_x^{\alpha_1} h(x) D_y^{\alpha_2} h(y) \left(D_x^{\alpha_1} D_y^{\alpha_2} A(x, y) \right) \mu(dx) \mu(dy) \\ &= \mathbb{E}_{c,w,b} \left[\int_{\Omega^2} \sum_{|\alpha_1| \leq 2, |\alpha_2| \leq 2} \left\{ D_x^{\alpha_1} h(x) D_y^{\alpha_2} h(y) \right. \right. \\ &\quad \times \left. \left(D_x^{\alpha_1} D_y^{\alpha_2} \left[\sigma(w \cdot x + b) \sigma(w \cdot y + b) + c^2 U(x) \cdot U(y) \right] \right) \mu(dx) \mu(dy) \right\} \right]. \end{aligned} \quad (2.38)$$

Notice that

$$\begin{aligned}
0 &\leq \mathbb{E}_{c,w,b} \left[\left\| c \int_{\Omega} \sum_{|\alpha| \leq 2} D_x^\alpha h(x) D_x^\alpha U(x) \mu(dx) \right\|^2 \right] \\
&\mathbb{E}_{c,w,b} \left[\int_{\Omega^2} \sum_{|\alpha_1| \leq 2, |\alpha_2| \leq 2} \left\{ D_x^{\alpha_1} h(x) D_y^{\alpha_2} h(y) \left(D_x^{\alpha_1} D_y^{\alpha_2} (c^2 U(x) \cdot U(y)) \right) \mu(dx) \mu(dy) \right\} \right]
\end{aligned} \tag{2.39}$$

Combining (2.38) with (2.39) we derive

$$\langle Sh, h \rangle_{\mathcal{H}^2} \geq \mathbb{E}_{c,w,b} \left[\left(\int_{\Omega} \sum_{|\alpha| \leq 2} D_x^\alpha h(x) D_x^\alpha \sigma(w \cdot x + b) \mu(dx) \right)^2 \right] \geq 0. \tag{2.40}$$

By Lemma 2.2.7, for $h \neq 0$, there exists $\epsilon > 0$, $w^* \in \mathbb{R}^n$, $b^* \in \mathbb{R}$ such that $0 < \epsilon = \left| \int_{\Omega} \sum_{|\alpha| \leq 2} D_x^\alpha h(x) D_x^\alpha \sigma(w^* \cdot x + b^*) \mu(dx) \right|$. As this integral is continuous with respect to the parameters w, b , there exists $\delta > 0$ such that for any $(w, b) \in R_\delta := \{(w, b) : \|w - w^*\| + |b - b^*| \leq \delta\}$, we know $\left| \int_{\Omega} \sum_{|\alpha| \leq 2} D_x^\alpha h(x) D_x^\alpha \sigma(w \cdot x + b) \mu(dx) \right| \geq \frac{\epsilon}{2}$. Therefore, from (2.40) we have

$$\langle Sh, h \rangle_{\mathcal{H}^2} \geq \mathbb{E}_{c,w,b} \left[\left(\int_{\Omega} \sum_{|\alpha| \leq 2} D_x^\alpha h(x) D_x^\alpha \sigma(w \cdot x + b) \mu(dx) \right)^2 \right] \geq \mathbb{E}_{c,w,b} \left[\frac{\epsilon^2}{4} \mathbb{1}_{R_\delta} \right] > 0. \tag{2.41}$$

□

Lemma 2.2.9. *Define the operator $\mathcal{A} : L^2 \rightarrow \mathcal{H}^2$ by $(\mathcal{A}f)(y) = \int_{\Omega} A(x, y) f(x) \mu(dx)$. The image of $\mathcal{A}$ is dense in $\mathcal{H}^2$, that is, for any $u \in \mathcal{H}^2$, there exists a sequence $\{v_k\}_{k \in \mathbb{N}}$ in L^2 such that $\lim_{k \rightarrow \infty} \|\mathcal{A}v_k - u\|_{\mathcal{H}^2} = 0$.*

Proof. By the smoothness of the kernel A , we have $\text{im}(\mathcal{A}) := \{\mathcal{A}v : v \in L^2\} \subset \mathcal{H}^2$. By definition, for any $h, g \in \mathcal{H}^2$, the inner product between $\mathcal{A}h$ and g in $\mathcal{H}^2$ space is

$$\begin{aligned}
\langle \mathcal{A}g, h \rangle_{\mathcal{H}^2} &= \sum_{|\alpha| \leq 2} \langle D^\alpha \mathcal{A}g, D^\alpha h \rangle_{L^2} = \int_{\Omega} \sum_{|\alpha| \leq 2} D_x^\alpha h(x) \int_{\Omega} D_x^\alpha A(x, y) g(y) \mu(dy) \mu(dx) \\
&= \int_{\Omega} g(y) \int_{\Omega} \sum_{|\alpha| \leq 2} D_x^\alpha h(x) D_x^\alpha A(x, y) \mu(dx) \mu(dy) = \langle g, Sh \rangle_{L^2}.
\end{aligned} \tag{2.42}$$

We introduce the adjoint operator of $\mathcal{A}$ on $\mathcal{H}^2$, denoted $\mathcal{A}^*$, and write

$$\langle \mathcal{A}g, h \rangle_{\mathcal{H}^2} = \langle g, \mathcal{A}^*h \rangle_{\mathcal{H}^2} = \langle g, Sh \rangle_{L^2}. \tag{2.43}$$

By Lemma 2.2.8, $Sh = 0$ if and only if $h = 0$. Therefore, by setting $g = Sh$ in (2.43), we have $\langle g, \mathcal{A}^*h \rangle_{\mathcal{H}^2} = \langle Sh, Sh \rangle_{L^2} > 0$ for any non-zero h . Thus, $\ker(\mathcal{A}^*) := \{h \in \mathcal{H}^2 : \mathcal{A}^*h = 0\} = \{0\}$. Write $\overline{\text{im}(\mathcal{A})}$ for the closure in $\mathcal{H}^2$ of $\text{im}(\mathcal{A})$. We recall that the ortho-complement in $\mathcal{H}^2$ of $\text{im}(\mathcal{A})$ is $\ker(\mathcal{A}^*)$, so we can decompose the space $\mathcal{H}^2$ as $\mathcal{H}^2 = \overline{\text{im}(\mathcal{A})} \oplus \ker(\mathcal{A}^*)$. We know that $\ker(\mathcal{A}^*) = \{0\}$ in $\mathcal{H}^2$, and hence conclude that $\text{im}(\mathcal{A})$ is dense in $\mathcal{H}^2$. $\square$

Theorem 2.2.10 (Density in $\mathcal{H}_{(0)}^2$). *The image of $\mathcal{B}$ is dense in $\mathcal{H}_{(0)}^2$, that is, for any $u \in \mathcal{H}_{(0)}^2$, there exists a sequence $\{v_k\}_{k \in \mathbb{N}}$ in L_η^2 such that $\lim_{k \rightarrow \infty} \|\mathcal{B}v_k - u\|_{\mathcal{H}^2} = 0$.*

Proof. For a function $u \in C^3(\overline{\Omega}) \cap C_0(\overline{\Omega})$, we define the function $\tilde{u} = u/\eta$. By Lemma 2.1.3(ii), $\tilde{u} \in C_b^2(\Omega) \subset \mathcal{H}^2$. Since the image of $\mathcal{A}$ is dense in $\mathcal{H}^2$, there exists a sequence $w_k \in \mathcal{H}^2$ such that $\|\mathcal{A}w_k - \tilde{u}\|_{\mathcal{H}^2} \rightarrow 0$. Consequently, the boundedness of η and its derivatives shows $\|\eta(\mathcal{A}w_k - \tilde{u})\|_{\mathcal{H}^2} = \|\mathcal{B}(w_k/\eta) - u\|_{\mathcal{H}^2} \rightarrow 0$. Since $C^3(\overline{\Omega}) \cap C_0(\overline{\Omega})$ is dense in $\mathcal{H}_{(0)}^2$ (under the $\mathcal{H}^2$ topology, Lemma 2.1.3(i)), we conclude that the image of $\mathcal{B}$ is dense in $\mathcal{H}_{(0)}^2$. $\square$

Remark 2.2.11. The above result shows that it is, in principle, possible to approximate the PDE solution u using an infinite neural network, with the boundary condition enforced by η . It also justifies the boundary-matching method proposed by [100]. In particular, as $u - (1 - \eta)f \in \mathcal{H}_{(0)}^2$, there exists a sequence $v_k \in L_\eta^2$ such that $\mathcal{B}v_k + (1 - \eta)f \rightarrow u$ in $\mathcal{H}^2$. Of course, whether a particular training algorithm yields such a sequence is a separate question.

We collect the remaining key properties of $\mathcal{B}$ into the following lemma.

Lemma 2.2.12. *The linear operator $\mathcal{B} : L_\eta^2 \rightarrow \mathcal{H}^2$ has the following properties:*

1. $\mathcal{B}$ is strictly positive definite, and induces a norm $\|\cdot\|_{\mathcal{B}}$ on L_η^2 , given by $\|v\|_{\mathcal{B}} := \sqrt{\langle \mathcal{B}v, v \rangle_{L^2}}$. In particular, $\|v\|_{\mathcal{B}} < \infty$ for all $v \in L_\eta^2$.
2. There exists a constant $\lambda > 0$ such that $\|\mathcal{B}v\|_{L^2}^2 \leq \lambda \|v\|_{\mathcal{B}}^2$.

Proof. For $v \in L_\eta^2$, let $g := \eta v \in L^2$. By definition,

$$\begin{aligned}
\langle v, \mathcal{B}v \rangle &= \langle v, \eta \mathcal{A}g \rangle = \langle g, \mathcal{A}g \rangle = \int_{\Omega^2} g(x)g(y)A(x,y)\mu(dx)\mu(dy) \\
&= \int_{\Omega^2} g(x)g(y)\mathbb{E}_{c,w,b}[\sigma(w \cdot x + b)\sigma(w \cdot y + b)]\mu(dx)\mu(dy) \\
&\quad + \int_{\Omega^2} g(x)g(y)\mathbb{E}_{c,w,b}[c^2 U(x) \cdot U(y)]\mu(dx)\mu(dy) \\
&= \mathbb{E}_{c,w,b} \left[\left(\int_{\Omega} \sigma(w \cdot x + b)g(x)\mu(dx) \right)^2 + \left\| c \int_{\Omega} U(x)g(x)\mu(dx) \right\|^2 \right] \\
&\geq \mathbb{E}_{c,w,b} \left[\left(\int_{\Omega} \sigma(w \cdot x + b)g(x)\mu(dx) \right)^2 \right] \geq 0.
\end{aligned} \tag{2.44}$$

For $v \neq 0$, we know $g \neq 0$. With Assumption 2.1.14, by Theorem 5 in [48], there exists $w^* \in \mathbb{R}^2$, $b^* \in \mathbb{R}$ such that $|\int_{\Omega} \sigma(w^* \cdot x + b^*)g(x)\mu(dx)| = \epsilon > 0$. As the integral is continuous with respect to parameters w, b there exists $\delta > 0$ such that for any (w, b) in the set $R_\delta := \{(w, b) : \|w - w^*\| + |b - b^*| < \delta\}$, we have $|\int_{\Omega} \sigma(w \cdot x + b)g(x)\mu(dx)| > \epsilon/2$. Therefore $\langle v, \mathcal{B}v \rangle \geq \mathbb{E}_{c,w,b}[(\int_{\Omega} \sigma(w \cdot x + b)g(x)\mu(dx))^2] \geq \mathbb{E}_{c,w,b}[\mathbb{1}_{R_\delta} \frac{\epsilon^2}{4}] > 0$; so $\mathcal{B}$ defines a norm as stated.

From the above calculations, we also see that $\mathcal{A}$ is a positive-definite Hilbert–Schmidt integral operator on L^2 . Therefore, its eigenfunctions span the entire L^2 space and the spectral theorem applies, in particular $\mathcal{A}$ has nonnegative eigenvalues bounded above. Suppose λ is the supremum of the eigenvalues of $\mathcal{A}$, we have $\|\mathcal{A}g\|_{L^2} \leq \lambda \langle g, \mathcal{A}g \rangle$. Thus, as $|\eta(x)| < 1$, we know $\|\mathcal{B}v\|_{L^2}^2 = \|\eta \mathcal{A}(\eta v)\|_{L^2}^2 \leq \|\mathcal{A}(\eta v)\|_{L^2}^2 \leq \lambda \langle \eta v, \mathcal{A}(\eta v) \rangle = \lambda \langle v, \mathcal{B}v \rangle = \lambda \|v\|_{\mathcal{B}}^2$. $\square$

2.3 Convergence to the limit ODE

In this section, we seek to understand the behavior of our approximator Q_t^N when $N \rightarrow \infty$. We prove that the pre-limit process, $\{Q_t^N(\cdot)\}_{0 \leq t \leq T}$, converges to a limiting process $\{Q_t(\cdot)\}_{0 \leq t \leq T}$, in an appropriate space of functions. In this section, we will only use Assumptions 2.1.13(i–iii), in particular we will not need the assumption that $\mathcal{L}$ is monotone or that a solution to the PDE exists in $\mathcal{H}^2$. The challenge here is that our operator $\mathcal{L}$ is not generally the gradient of any potential function, and is an unbounded operator (in the L^2 or supremum norms), and so some care is needed.

In the following subsections, we first bound the difference between the kernels $B_t^N(x, y)$ and $B(x, y)$ in a convenient sense. We then show that, as N becomes large, the neural network approximator converges to the solution of an infinite dimensional ODE. Our main convergence result is Theorem 2.3.10.

2.3.1 Characterizing the difference between kernels

We characterize the (second order) difference between two kernels at (x, y) by

$$\begin{aligned} H(A_1, A_2)(x, y) := & |A_1(x, y) - A_2(x, y)| + \sum_k |\partial_{y_k} A_1(x, y) - \partial_{y_k} A_2(x, y)| \\ & + \sum_{k,l} |\partial_{y_k y_l}^2 A_1(x, y) - \partial_{y_k y_l}^2 A_2(x, y)|. \end{aligned} \quad (2.45)$$

Note that partial derivatives are only taken with respect to y components. From Lemma 2.2.2, there exists a constant M such that, for all $x, y \in \Omega$,

$$H(B, 0)(x, y) \leq M. \quad (2.46)$$

Similarly, the (second order) difference between two smooth functions is characterized by

$$\begin{aligned} G(f_1, f_2)(x) := & |f_1(x) - f_2(x)| + \sum_k |\partial_{x_k} f_1(x) - \partial_{x_k} f_2(x)| \\ & + \sum_{k,l} |\partial_{x_k x_l}^2 f_1(x) - \partial_{x_k x_l}^2 f_2(x)|. \end{aligned} \quad (2.47)$$

2.3.1.1 Difference between kernel $B_t^N(x, y)$ and $B_0^N(x, y)$

In this subsection, we characterize the difference between kernel B_t^N and B_0^N . We denote the expectation taken with respect to randomized initialization by $\mathbb{E}_{c_0, w_0, b_0}[\cdot]$. The proofs of the following lemmas are included in the appendix.

Lemma 2.3.1. *There exists $C > 0$ such that, for all $i, j \in \{1, 2, \dots, n\}$, for all $0 \leq t \leq T$, and for all $N > 0$,*

$$|A_t^N(x, y) - A_0^N(x, y)| \leq CN^{\delta+\beta-1}, \quad (2.48)$$

$$|\partial_{y_i} A_t^N(x, y) - \partial_{y_i} A_0^N(x, y)| \leq CN^{\delta+\beta-1} \left[1 + \frac{\sum_{k=1}^N |(w_0^k)_i|}{N} \right], \quad (2.49)$$

$$|\partial_{y_i y_j}^2 A_t^N(x, y) - \partial_{y_i y_j}^2 A_0^N(x, y)| \leq CN^{\delta+\beta-2} \left(\sum_{k=1}^N (1 + |(w_0^k)_i|)(1 + |(w_0^k)_j|) \right). \quad (2.50)$$

Lemma 2.3.2. *There exists $C > 0$ such that, for all $0 \leq t \leq T$ and all $N > 0$, the expected difference between A_t^N and A_0^N satisfies*

$$\mathbb{E}_{c_0, w_0, b_0} \left[|H(A_t^N, A_0^N)(x, y)|^2 \right] \leq CN^{2(\delta+\beta-1)}. \quad (2.51)$$

After characterizing the difference between A_t^N and A_0^N , we estimate the difference between the kernels B_t^N and B_0^N .

Lemma 2.3.3. *There exists $C > 0$ such that, for all $0 \leq t \leq T$ and all $N > 0$*

$$\mathbb{E}_{c_0, w_0, b_0} \left[N^{2\delta} \int_{\Omega^2} |H(B_t^N, B_0^N)(x, y)|^2 \mu(dx) \mu(dy) \right] \leq C N^{2(2\delta + \beta - 1)}. \quad (2.52)$$

Proof. Since $B_t^N(x, y) = \eta(x)\eta(y)A_t^N(x, y)$, by the Cauchy–Schwarz inequality and the assumption $\eta \in C_b^3$, for some constant $C > 0$,

$$\begin{aligned} |H(B_t^N, B_0^N)(x, y)|^2 &\leq C \left[|B_t^N(x, y) - B_0^N(x, y)|^2 + \sum_k |\partial_{y_k} B_t^N(x, y) - \partial_{y_k} B_0^N(x, y)|^2 \right. \\ &\quad \left. + \sum_{k,l} |\partial_{y_k y_l} B_t^N(x, y) - \partial_{y_k y_l} B_0^N(x, y)|^2 \right]. \end{aligned} \quad (2.53)$$

By the product rule, we have

$$\partial_{y_i} B_t^N(x, y) = \eta(x) A_t^N(x, y) \partial_{y_i} \eta(y) + \eta(x) \eta(y) \partial_{y_i} A_t^N(x, y), \quad (2.54)$$

$$\begin{aligned} \partial_{y_i y_j}^2 B_t^N(x, y) &= \eta(x) A_t^N(x, y) \partial_{y_i y_j}^2 \eta(y) + \eta(x) \partial_{y_j} A_t^N(x, y) \partial_{y_i} \eta(y) \\ &\quad + \partial_{y_i} A_t^N(x, y) \partial_{y_j} \eta(y) + \eta(x) \eta(y) \partial_{y_i y_j}^2 A_t^N(x, y). \end{aligned} \quad (2.55)$$

As $\eta \in C_b^3$, there exists a constant $C > 0$ independent of training time t , such that

$$|B_t^N(x, y) - B_0^N(x, y)| \leq C |A_t^N(x, y) - A_0^N(x, y)|, \quad (2.56)$$

$$|\partial_{y_i} B_t^N(x, y) - \partial_{y_i} B_0^N(x, y)| \leq C \left[|A_t^N(x, y) - A_0^N(x, y)| + |\partial_{y_i} A_t^N(x, y) - \partial_{y_i} A_0^N(x, y)| \right], \quad (2.57)$$

and

$$\begin{aligned} &|\partial_{y_i y_j}^2 B_t^N(x, y) - \partial_{y_i y_j}^2 B_0^N(x, y)| \\ &\leq C \left[|A_t^N(x, y) - A_0^N(x, y)| + |\partial_{y_i} A_t^N(x, y) - \partial_{y_i} A_0^N(x, y)| \right. \\ &\quad \left. + |\partial_{y_j} A_t^N(x, y) - \partial_{y_j} A_0^N(x, y)| + |\partial_{y_i y_j}^2 A_t^N(x, y) - \partial_{y_i y_j}^2 A_0^N(x, y)| \right]. \end{aligned} \quad (2.58)$$

Substituting (2.56), (2.57) and (2.58) in (2.53),

$$\begin{aligned} |H(B_t^N, B_0^N)(x, y)|^2 &\leq C \left[|A_t^N(x, y) - A_0^N(x, y)|^2 + \sum_k |\partial_{y_k} A_t^N(x, y) - \partial_{y_k} A_0^N(x, y)|^2 \right. \\ &\quad \left. + \sum_{k,l} |\partial_{y_k y_l}^2 A_t^N(x, y) - \partial_{y_k y_l}^2 A_0^N(x, y)|^2 \right]. \end{aligned} \quad (2.59)$$

By Lemma 2.3.2, there exists a constant $C > 0$ such that for all $N \in \mathbb{N}$ and any time $0 \leq t \leq T$ $\mathbb{E}_{c_0, w_0, b_0} [|H(B_t^N, B_0^N)|^2] \leq CN^{2(\delta+\beta-1)}$. Thus, by Tonelli's theorem, we have

$$\mathbb{E}_{c_0, w_0, b_0} \left[\int_{\Omega^2} |H(B_t^N, B_0^N)(x, y)|^2 \mu(dx) \mu(dy) \right] \leq CN^{2(\delta+\beta-1)}. \quad (2.60)$$

Multiplying by $N^{2\delta}$ on both sides concludes the result. $\square$

2.3.1.2 Difference between $B_0^N(x, y)$ and $B(x, y)$

We now characterize the difference between the kernels B_0^N and B . The law of large numbers implies B_0^N is converging to B almost surely; here we obtain a bound on its speed of convergence. Again, the proof of the following lemma is given in the appendix.

Lemma 2.3.4. *There exists $C > 0$ such that, for any $j, k, l \in \{1, 2, \dots, n\}$, $\forall N > 0$,*

$$\begin{aligned} \mathbb{E}_{c_0, w_0, b_0} [|A_0^N(x, y) - A(x, y)|^2] &\leq \frac{C}{N}, \\ \mathbb{E}_{c_0, w_0, b_0} [|\partial_{y_i} A_0^N(x, y) - \partial_{y_i} A(x, y)|^2] &\leq \frac{C}{N}, \\ \mathbb{E}_{c_0, w_0, b_0} [|\partial_{y_i y_j}^2 A_0^N(x, y) - \partial_{y_i y_j}^2 A(x, y)|^2] &\leq \frac{C}{N}. \end{aligned} \quad (2.61)$$

Lemma 2.3.5. *There exists $C > 0$ such that, for all $N > 0$,*

$$\mathbb{E}_{c_0, w_0, b_0} \left[N^{2\delta} \int_{\Omega^2} |H(B_0^N, B)(x, y)|^2 \mu(dx) \mu(dy) \right] \leq \frac{C}{N^{1-2\delta}}. \quad (2.62)$$

Proof. Similar to (2.53), we derive

$$\begin{aligned} |H(B_0^N, B)(x, y)|^2 &\leq C \left[|A_0^N(x, y) - A(x, y)|^2 + \sum_k |\partial_{y_k} A_0^N(x, y) - \partial_{y_k} A(x, y)|^2 \right. \\ &\quad \left. + \sum_{k, l} |\partial_{y_k y_l}^2 A_0^N(x, y) - \partial_{y_k y_l}^2 A(x, y)|^2 \right]. \end{aligned} \quad (2.63)$$

Therefore, Lemma 2.3.4 guarantees $\mathbb{E}_{c_0, w_0, b_0} [|H(B_0^N, B)(x, y)|^2] \leq C/N$. By Tonelli's theorem, we have

$$\begin{aligned} \mathbb{E}_{c_0, w_0, b_0} \left[N^{2\delta} \int_{\Omega^2} |H(B_0^N, B)(x, y)|^2 \mu(dx) \mu(dy) \right] \\ = \int_{\Omega^2} N^{2\delta} \mathbb{E}_{c_0, w_0, b_0} [|H(B_0^N, B)(x, y)|^2] \mu(dx) \mu(dy) \leq \frac{C}{N^{1-2\delta}}. \end{aligned} \quad (2.64)$$

$\square$

2.3.1.3 Difference between kernel $B_t^N(x, y)$ and $B(x, y)$

Combining the results from the above two subsections, we provide one of our key lemmas.

Lemma 2.3.6. *The kernels B_t^N and B satisfy*

$$\lim_{N \rightarrow \infty} \mathbb{E}_{c_0, w_0, b_0} \left[N^\delta \int_0^T \int_{\Omega^2} |H(B_u^N, B)(x, y)|^2 \mu(dx) \mu(dy) du \right] = 0. \quad (2.65)$$

Proof. By the triangle inequality,

$$H(B_u^N, B)(x, y) \leq H(B_u^N, B_0^N)(x, y) + H(B_0^N, B)(x, y). \quad (2.66)$$

Therefore,

$$|H(B_u^N, B)(x, y)|^2 \leq 2|H(B_u^N, B_0^N)(x, y)|^2 + 2|H(B_0^N, B)(x, y)|^2. \quad (2.67)$$

Combining Lemmas 2.3.3 and 2.3.5, we have, for any $0 \leq t \leq T$,

$$N^{2\delta} \mathbb{E}_{c_0, w_0, b_0} \left[\int_{\Omega^2} |H(B_u^N, B)(x, y)|^2 \mu(dx) \mu(dy) \right] \leq CN^{2(2\delta+\beta-1)} + CN^{2\delta-1}. \quad (2.68)$$

Integrating with respect to time,

$$\begin{aligned} & \mathbb{E}_{c_0, w_0, b_0} \left[N^\delta \int_0^T \int_{\Omega^2} |H(B_u^N, B)(x, y)|^2 \mu(dx) \mu(dy) du \right] \\ & \leq CT(N^{2(2\delta+\beta-1)} + N^{2\delta-1}), \end{aligned} \quad (2.69)$$

which converges to zero as $N \rightarrow \infty$. $\square$

2.3.2 Convergence of initial approximator Q_0^N to Q_0

In this subsection, we show that the randomly initialized approximator Q_0^N converges to its limit, given by $Q_0 := f$ as the number of hidden units N goes to infinity.

Lemma 2.3.7. *The initial approximator Q_0^N satisfies*

$$\lim_{N \rightarrow \infty} \mathbb{E}_{c_0, w_0, b_0} [\|Q_0^N - Q_0\|_{\mathcal{H}^2}^2] = 0. \quad (2.70)$$

Proof. By definition, for any indices k, l ,

$$\begin{aligned} Q_0^N(x) - Q_0(x) &= \frac{\eta(x)}{N^\beta} \sum_{i=1}^N c_0^i \sigma(w_0^i \cdot x + b_0^i), \\ \partial_{x_k}(Q_0^N - Q_0)(x) &= \frac{\eta(x)}{N^\beta} \sum_{i=1}^N c_0^i \sigma'(w_0^i \cdot x + b_0^i)(w_0^i)_k + \frac{\partial_{x_k} \eta(x)}{N^\beta} \sum_{i=1}^N c_0^i \sigma(w_0^i \cdot x + b_0^i), \end{aligned} \quad (2.71)$$

and

$$\begin{aligned}
& \partial_{x_k x_l}^2 (Q_0^N - Q_0)(x) \\
&= \frac{\eta(x)}{N^\beta} \sum_{i=1}^N c_0^i \sigma''(w_0^i \cdot x + b_0^i) (w_0^i)_k (w_0^i)_l + \frac{\partial_{x_l} \eta(x)}{N^\beta} \sum_{i=1}^N c_0^i \sigma'(w_0^i \cdot x + b_0^i) (w_0^i)_k \\
&+ \frac{\partial_{x_k} \eta(x)}{N^\beta} \sum_{i=1}^N c_0^i \sigma'(w_0^i \cdot x + b_0^i) (w_0^i)_l + \frac{\partial_{x_k x_l}^2 \eta(x)}{N^\beta} \sum_{i=1}^N c_0^i \sigma(w_0^i \cdot x + b_0^i).
\end{aligned} \tag{2.72}$$

As $\{c_0^i, w_0^i, b_0^i\}$ are independent for different $i \in \{1, 2, \dots, N\}$, we have, for a constant $C > 0$ which may vary from line to line,

$$\mathbb{E}_{c_0, w_0, b_0} \left[|Q_0^N(x) - Q_0(x)|^2 \right] = \frac{\eta(x)^2}{N^{2\beta-1}} \mathbb{E}_{c, w, b} [c^2 \sigma(w \cdot x + b)^2] \leq \frac{C}{N^{2\beta-1}}, \tag{2.73}$$

and

$$\begin{aligned}
& \mathbb{E}_{c_0, w_0, b_0} \left[|\partial_{x_k} Q_0^N(x) - \partial_{x_k} Q_0(x)|^2 \right] \\
& \leq \frac{2}{N^{2\beta-1}} \mathbb{E}_{c, w, b} \left[\eta(x)^2 c^2 \sigma'(w \cdot x + b)^2 w_k^2 + (\partial_{x_k} \eta(x))^2 c^2 \sigma(w \cdot x + b)^2 \right] \leq \frac{C}{N^{2\beta-1}}.
\end{aligned} \tag{2.74}$$

Similarly,

$$\mathbb{E}_{c_0, w_0, b_0} \left[|\partial_{x_k x_l}^2 Q_0^N(x) - \partial_{x_k x_l}^2 Q_0(x)|^2 \right] \leq \frac{C}{N^{2\beta-1}}. \tag{2.75}$$

Summing over all indices, then integrating over Ω , gives

$$\mathbb{E}_{c_0, w_0, b_0} \left[\|Q_0^N - Q_0\|_{\mathcal{H}^2}^2 \right] \leq C N^{1-2\beta}. \tag{2.76}$$

As $\beta > 1/2$, we have the desired convergence. $\square$

2.3.3 Convergence of the approximator Q_t^N to Q_t

Definition 2.3.8 (Wide network limit). *For $t \geq 0$, we define the wide network limit Q_t by the infinite-dimensional ODE*

$$\frac{dQ_t}{dt}(y) = \alpha \int_{\Omega} \mathcal{L}(Q_t)(x) B(x, y) \mu(dx), \quad \text{for all } y \in \overline{\Omega}, \tag{2.77}$$

with initial value $Q_0(y) = f(y)$, for all $y \in \overline{\Omega}$. Equivalently, we can write

$$\frac{dQ_t}{dt} = \alpha \mathcal{BL}(Q_t), \quad Q_0 = f. \tag{2.78}$$

Theorem 2.3.9. *The limit ODE (2.78) admits a unique solution in $\mathcal{H}^2$.*

Proof. From Lemma 2.2.6, $\mathcal{B}$ is uniformly Lipschitz as a map $L^2 \rightarrow \mathcal{H}^2$. By Assumption 2.1.6, $\mathcal{L}$ is uniformly Lipschitz as a map $\mathcal{H}^2 \rightarrow L^2$. Hence, the operator $\mathcal{BL} : \mathcal{H}^2 \mapsto \mathcal{H}^2$ is uniformly Lipschitz. From the Picard–Lindelöf theorem (see, for example, Theorem 2.2.1 in [71]), we know that (2.78) admits a unique solution with $Q_t \in \mathcal{H}^2$ for all $t \geq 0$. $\square$

We next prove that Q_t^N converges to Q_t , justifying the name ‘wide network limit’.

Theorem 2.3.10 (Convergence to the limit process Q_t). *For any fixed time $0 \leq t \leq T$,*

$$\lim_{N \rightarrow \infty} \mathbb{E}_{c_0, w_0, b_0} [\|Q_t^N - Q_t\|_{\mathcal{H}^2}] = 0. \quad (2.79)$$

In particular, the boundary value of Q_t is given by $Q_t|_{\partial\Omega} = f$.

Proof. From the dynamics of Q_t^N and Q ((2.28) and (2.77)) we have

$$Q_t^N(y) - Q_0^N(y) = \alpha \int_0^t \int_{\Omega} F^N(\mathcal{L}Q_u^N(x)) B_u^N(x, y) \mu(dx) du, \quad (2.80)$$

$$Q_t(y) - Q_0(y) = \alpha \int_0^t \int_{\Omega} \mathcal{L}Q_u(x) B(x, y) \mu(dx) du. \quad (2.81)$$

Differentiate $Q_t^N(y)$ and $Q_t(y)$ with respect to y_k to obtain

$$\partial_{y_k} Q_t^N(y) - \partial_{y_k} Q_0^N(y) = \alpha \int_0^t \int_{\Omega} F^N(\mathcal{L}Q_u^N(x)) \partial_{y_k} B_u^N(x, y) \mu(dx) du, \quad (2.82)$$

$$\partial_{y_k} Q_t(y) - \partial_{y_k} Q_0(y) = \alpha \int_0^t \int_{\Omega} \mathcal{L}Q_u(x) \partial_{y_k} B(x, y) \mu(dx) du. \quad (2.83)$$

Twice-differentiate Q_t^N and Q_t with respect to y_k and y_l to obtain:

$$\partial_{y_k y_l}^2 Q_t^N(y) - \partial_{y_k y_l}^2 Q_0^N(y) = \alpha \int_0^t \int_{\Omega} F^N(\mathcal{L}Q_u^N(x)) \partial_{y_k y_l}^2 B_u^N(x, y) \mu(dx) du, \quad (2.84)$$

$$\partial_{y_k y_l}^2 Q_t(y) - \partial_{y_k y_l}^2 Q_0(y) = \alpha \int_0^t \int_{\Omega} \mathcal{L}Q_u(x) \partial_{y_k y_l}^2 B(x, y) \mu(dx) du. \quad (2.85)$$

Subtracting (2.80) from (2.81), we have

$$\begin{aligned} & |Q_t^N(y) - Q_t(y)| - |Q_0^N(y) - Q_0(y)| \\ & \leq \alpha \int_0^t \left| \int_{\Omega} F(Q_u^N)(x) B_u^N(x, y) - \mathcal{L}Q_u(x) B(x, y) \mu(dx) \right| du \\ & \leq \alpha \int_0^t \left| \int_{\Omega} F^N(\mathcal{L}Q_u^N(x)) (B_u^N(x, y) - B(x, y)) \mu(dx) \right| du \\ & \quad + \alpha \int_0^t \left| \int_{\Omega} (F^N(\mathcal{L}Q_u^N(x)) - F^N(\mathcal{L}Q_u(x))) B(x, y) \mu(dx) \right| du \\ & \quad + \alpha \int_0^t \left| \int_{\Omega} (F^N(\mathcal{L}Q_u(x)) - \mathcal{L}Q_u(x)) B(x, y) \mu(dx) \right| du. \end{aligned} \quad (2.86)$$

Subtracting (2.82) from (2.83), we have

$$\begin{aligned}
& |\partial_{y_k} Q_t^N(y) - \partial_{y_k} Q_t(y)| - |\partial_{y_k} Q_0^N(y) - \partial_{y_k} Q_0(y)| \\
& \leq \alpha \int_0^t \left| \int_{\Omega} F^N(\mathcal{L}Q_u^N(x)) \left(\partial_{y_k} B_u^N(x, y) - \partial_{y_k} B(x, y)(x, y) \right) \mu(dx) \right| du \\
& \quad + \alpha \int_0^t \left| \int_{\Omega} \left(F^N(\mathcal{L}Q_u^N(x)) - F^N(\mathcal{L}Q_u(x)) \right) \partial_{y_k} B(x, y) \mu(dx) \right| du \\
& \quad + \alpha \int_0^t \left| \int_{\Omega} \left(F^N(\mathcal{L}Q_u(x)) - \mathcal{L}Q_u(x) \right) \partial_{y_k} B(x, y) \mu(dx) \right| du.
\end{aligned} \tag{2.87}$$

Similarly, subtracting (2.84) from (2.85), we have

$$\begin{aligned}
& |\partial_{y_k y_l}^2 Q_t^N(y) - \partial_{y_k y_l}^2 Q_t(y)| - |\partial_{y_k y_l}^2 Q_0^N(y) - \partial_{y_k y_l}^2 Q_0(y)| \\
& \leq \alpha \int_0^t \left| \int_{\Omega} F^N(\mathcal{L}Q_u^N(x)) \left(\partial_{y_k y_l}^2 B_u^N(x, y) - \partial_{y_k y_l}^2 B(x, y)(x, y) \right) \mu(dx) \right| du \\
& \quad + \alpha \int_0^t \left| \int_{\Omega} \left(F^N(\mathcal{L}Q_u^N(x)) - F^N(\mathcal{L}Q_u(x)) \right) \partial_{y_k y_l}^2 B(x, y) \mu(dx) \right| du \\
& \quad + \alpha \int_0^t \left| \int_{\Omega} \left(F^N(\mathcal{L}Q_u(x)) - \mathcal{L}Q_u(x) \right) \partial_{y_k y_l}^2 B(x, y) \mu(dx) \right| du.
\end{aligned} \tag{2.88}$$

As stated in Definition 2.1.17, the function $F^N = (\psi^N) \cdot (\psi^N)'$ satisfies a global Lipschitz condition. Therefore, for some $C > 0$,

$$\begin{aligned}
& \left| F^N(\mathcal{L}Q_t^N(x)) - F^N(\mathcal{L}Q_t(x)) \right| < C \left| \mathcal{L}Q_t^N(x) - \mathcal{L}Q_t(x) \right| \\
& \leq C \left(|Q_t^N(x) - Q_t(x)| + \sum_k |\partial_{x_k} Q_t^N(x) - \partial_{x_k} Q_t(x)| + \sum_{k,l} |\partial_{x_k x_l}^2 Q_t^N(x) - \partial_{x_k x_l}^2 Q_t(x)| \right) \\
& = CG(Q_t^N, Q_t)(x).
\end{aligned} \tag{2.89}$$

Summing (2.86), (2.87) and (2.88) over all indices k and l , we have

$$\begin{aligned}
& G(Q_t^N, Q_t)(y) \\
& = |Q_t^N(y) - Q_t(y)| + \sum_k |\partial_{x_k} Q_t^N(y) - \partial_{x_k} Q_t(y)| + \sum_{k,l} |\partial_{x_k x_l}^2 Q_t^N(y) - \partial_{x_k x_l}^2 Q_t(y)| \\
& \leq G(Q_t^N, Q_0)(y) + CN^\delta \int_0^t \int_{\Omega} H(B_u^N, B)(x, y) \mu(dx) du \\
& \quad + C \int_0^t \int_{\Omega} G(Q_u^N, Q_u)(x) H(B, 0)(x, y) \mu(dx) du \\
& \quad + C \int_0^t \int_{\Omega} |F^N(\mathcal{L}Q_u(x)) - \mathcal{L}Q_u(x)| H(B, 0)(x, y) \mu(dx) du.
\end{aligned} \tag{2.90}$$

Here, the coefficient N^δ comes from the term F^N , as by Assumption 2.1.17, $|F^N| = |(\psi^N) \cdot (\psi^N)'| \leq 2N^\delta$. As mentioned in Remark 2.1.18, we also know $|F^N(x) - x| \leq 2|x|\mathbb{1}_{\{|x| \geq N^\delta\}}$. As $H(B, 0)$ is uniformly bounded (Lemma 2.2.2), from inequality (2.90), for some $C > 0$ we have

$$\begin{aligned} G(Q_t^N, Q_t)(y) &\leq G(Q_t^N, Q_0)(y) + CN^\delta \int_0^t \int_\Omega H(B_u^N, B)(x, y) \mu(dx) du \\ &\quad + C \int_0^t \int_\Omega G(Q_u^N, Q_u)(x) \mu(dx) du \\ &\quad + C \int_0^t \int_\Omega |\mathcal{L}Q_u(x)| \mathbb{1}_{\{|\mathcal{L}Q_u(x)| \geq N^\delta\}} \mu(dx) du. \end{aligned} \quad (2.91)$$

By multiplying $G(Q_t^N, Q_t)(y)$ in inequality (2.91), integrating on both sides, and applying the Cauchy–Schwarz inequality, from (2.91) we derive

$$\begin{aligned} &\int_\Omega |G(Q_t^N, Q_t)(y)|^2 \mu(dy) \\ &\leq C \left(\int_\Omega |G(Q_t^N, Q_t)(y)|^2 \mu(dy) \right)^{\frac{1}{2}} \left[\left(\int_\Omega |G(Q_0^N, Q_0)(y)|^2 \mu(dy) \right)^{\frac{1}{2}} \right. \\ &\quad + N^\delta \int_0^t \left(\int_{\Omega^2} |H(B_u^N, B)(x, y)|^2 \mu(dx) \mu(dy) \right)^{\frac{1}{2}} du \\ &\quad + \int_0^t \int_\Omega |G(Q_u^N, Q_u)(x)|^2 \mu(dx) du \\ &\quad \left. + \int_0^t \int_\Omega |\mathcal{L}Q_u(x)| \mathbb{1}_{\{|\mathcal{L}Q_u(x)| \geq N^\delta\}} \mu(dx) du \right]. \end{aligned} \quad (2.92)$$

We write

$$\begin{aligned} J^N &:= \left(\int_\Omega |G(Q_0^N, Q_0)(y)|^2 \mu(dy) \right)^{\frac{1}{2}} + N^\delta \int_0^T \int_{\Omega^2} |H(B_u^N, B)(x, y)|^2 \mu(dx) \mu(dy) du \\ &\quad + \int_0^T \int_\Omega |\mathcal{L}Q_u(x)| \mathbb{1}_{\{|\mathcal{L}Q_u(x)| \geq N^\delta\}} \mu(dx) du. \end{aligned} \quad (2.93)$$

Then the inequality (2.91) can be formulated as

$$\left(\int_\Omega |G(Q_t^N, Q_t)(y)|^2 \mu(dy) \right)^{\frac{1}{2}} \leq J^N + C \int_0^t \left(\int_\Omega |G(Q_u^N, Q_u)(x)|^2 \mu(dx) \right)^{\frac{1}{2}} du. \quad (2.94)$$

Since

$$G(Q_t^N, Q_t)(y) = |Q_t^N(y) - Q_t(y)| + \sum_k |\partial_{y_k}(Q_t^N - Q_t)(y)| + \sum_{k,l} |\partial_{y_k, y_l}^2(Q_t^N - Q_t)(y)|, \quad (2.95)$$

by a simple quadratic-mean inequality we have

$$\|Q_t^N - Q_t\|_{\mathcal{H}^2}^2 \leq \int_{\Omega} |G(Q_t^N, Q_t)(y)|^2 \mu(dy) \leq (n^2 + n + 1) \|Q_t^N - Q_t\|_{\mathcal{H}^2}^2. \quad (2.96)$$

Consequently, from (2.94) we derive

$$\|Q_t^N - Q_t\|_{\mathcal{H}^2} \leq J^N + C \int_0^t \|Q_u^N - Q_u\|_{\mathcal{H}^2} du. \quad (2.97)$$

By applying Grönwall's inequality, (2.97) yields the exponential bound $\int_0^t \|Q_u^N - Q_u\|_{\mathcal{H}^2} du \leq \frac{J^N}{C} (e^{Ct} - 1)$.

It remains to bound J^N in (2.93), as N goes to infinity. By Lemma 2.3.7 and (2.96), $\mathbb{E}_{c_0, w_0, b_0} [\int_{\Omega} |G(Q_0^N, Q_0)(y)|^2 \mu(dy)] \rightarrow 0$. By the dominated convergence theorem, we obtain $\int_0^T \int_{\Omega} |\mathcal{L}Q_u(x)| \mathbb{1}_{\{|\mathcal{L}Q_u(x)| \geq N^\delta\}} \mu(dx) du \rightarrow 0$. Finally, by Lemma 2.3.6, we show that $\mathbb{E}_{c_0, w_0, b_0} [N^\delta \int_0^T \int_{\Omega^2} |H(B_u^N, B)(x, y)|^2 \mu(dx) \mu(dy) du] \rightarrow 0$. Therefore, $\mathbb{E}_{c_0, w_0, b_0} [J^N] \rightarrow 0$ as $N \rightarrow \infty$ and consequently, from the above exponential bound,

$$\lim_{N \rightarrow \infty} \mathbb{E}_{c_0, w_0, b_0} \left[\int_0^T \|Q_u^N - Q_u\|_{\mathcal{H}^2} du \right] = 0. \quad (2.98)$$

Substituting this result back into inequality (2.97) and taking the expectation with respect to the random initialization leads to $\lim_{N \rightarrow \infty} \mathbb{E}_{c_0, w_0, b_0} [\|Q_t^N - Q_t\|_{\mathcal{H}^2}] = 0$ for any time $t \leq T$.

Finally, observe that $Q_t^N|_{\partial\Omega} = f$ by construction, so almost sure convergence in $\mathcal{H}^2$ (for a subsequence of N) and continuity of the boundary trace operator imply $Q_t|_{\partial\Omega} = f$. $\square$

Remark 2.3.11. The convergence mode in the above theorem can be strengthened to an almost sure convergence by the Glivenko–Contelli theorem. We refrain from further discussing these results here.

2.4 Convergence for large training time

In the previous section, we showed that, as the neural network is made wider, its training process converges to a process that satisfies an infinite-dimensional ODE (2.78). In particular, the wide-limit of the approximate solution has the dynamics $\frac{dQ_t}{dt} = \alpha \mathcal{BL}(Q_t)$. This limit process can therefore be regarded as an approximation of the setting where we use a large (single-layer) neural network, and our problem (in the limiting setting) is transformed into the study of this infinite-dimensional dynamical system. The following theorem is an easy consequence.

Theorem 2.4.1. *If the wide-network limit converges in $\mathcal{H}^1$ to a fixed point Q , then Q is a solution to the PDE (2.1).*

Proof. For a fixed point Q , we know $\mathcal{BL}(Q) = 0$. From Lemma 2.2.12, this implies that $\mathcal{L}(Q) = 0$, so the PDE dynamics are satisfied. As Q^N satisfies the boundary conditions $Q_t^N|_{\partial\Omega} = f$, and $Q_t^N \rightarrow Q_t$ in $\mathcal{H}^2$ for each t , and $Q_t \rightarrow Q$ in $\mathcal{H}^1$ we see that Q must satisfy the boundary condition also (by $\mathcal{H}^1$ -continuity of the boundary trace operator). $\square$

In this section, we will consider the simple case where $\mathcal{L}$ is a monotone operator. This implies that the dynamical system $dv/dt = \mathcal{L}v$ converges in L^2 exponentially quickly, which suggests the dynamics of Q will be well behaved. However, the presence of the operator $\mathcal{B}$ leads to some difficulties in our analysis. Nevertheless, we will show that, in this setting, our approximation Q_t converges to the true solution u of the PDE, at least for a generic subsequence of times.

We assume $\alpha = 1$ in (2.18) for notational simplicity in this section, without loss of generality (as this is a simple rescaling of time).

Theorem 2.4.2. *Let u be the solution to the PDE (2.1) under our assumptions (Assumption 2.1.13, now including part iv), and assume the neural network configuration satisfies Assumptions 2.1.14 and 2.1.16. Then the wide network limit $\{Q_t\}_{t \geq 0}$ of the Q -PDE algorithm (Definitions 2.1.20, 2.3.8) satisfies*

$$\frac{1}{t} \int_0^t \|Q_s - u\|_{L^2}^2 ds \rightarrow 0 \text{ as } t \rightarrow \infty. \quad (2.99)$$

In particular, there exists a set $A \subset [0, \infty)$ with $\lim_{t \rightarrow \infty} (t^{-1} \text{Leb}(A \cap [0, t])) = 0$ such that, for all sequences $t_k \rightarrow \infty$ which do not take values in A , we have the (strong) L^2 convergence

$$\|Q_{t_k} - u\|_{L^2} \rightarrow 0.$$

Proof. We begin by supposing we have a decomposition

$$Q_0 = f = u + \mathcal{B}v_0 + g, \quad (2.100)$$

for some $v_0 \in L_\eta^2$ and $g \in \mathcal{H}^2$. As $u \in \mathcal{H}^2$, we can choose v_0 arbitrarily and then find g using (2.100). Consider the process v solving the ODE

$$\frac{dv_t}{dt} = \mathcal{L}(\mathcal{B}v_t + u + g) \quad (2.101)$$

with initial value v_0 as in (2.100). As detailed in Appendix 2.6.6, as $\mathcal{B}$ and $\mathcal{L}$ are both Lipschitz continuous in appropriate spaces, a standard Picard–Lindelöf argument shows that v is uniquely defined in L^2_η , for all $t \geq 0$.

By differentiating $\mathcal{B}v_t$ (and using dominated convergence to move the derivative through $\mathcal{B}$), we obtain

$$\frac{d(\mathcal{B}v_t + u + g)}{dt} = \frac{d(\mathcal{B}v_t)}{dt} = \mathcal{B}\mathcal{L}(\mathcal{B}v_t + u + g); \quad \mathcal{B}v_0 + u + g = Q_0. \quad (2.102)$$

In particular, as the ODE defining Q has a unique solution in $\mathcal{H}^2$ (Theorem 2.3.9), we have the identity $Q_t = \mathcal{B}v_t + u + g$.

Using the Young inequality $2\langle x, y \rangle \leq \gamma\|x\|^2 + (4/\gamma)\|y\|^2$ for $\gamma > 0$, together with the fact $\mathcal{L}(u) = 0$ and our assumption that $\mathcal{L}$ is strongly monotone and Lipschitz, for some constant $k > 0$ we have

$$\begin{aligned} \frac{d}{dt}\|v_t\|_{\mathcal{B}}^2 &= \frac{d}{dt}\langle \mathcal{B}v_t, v_t \rangle = 2\langle \mathcal{B}v_t, \mathcal{L}(\mathcal{B}v_t + u + g) \rangle \\ &= 2\langle \mathcal{B}v_t, \mathcal{L}(\mathcal{B}v_t + u + g) - \mathcal{L}(u + g) \rangle + 2\langle \mathcal{B}v_t, \mathcal{L}(u + g) - \mathcal{L}(u) \rangle \\ &\leq -2\gamma\|\mathcal{B}v_t\|_{L^2}^2 + \gamma\|\mathcal{B}v_t\|_{L^2}^2 + \frac{4}{\gamma}\|\mathcal{L}(u + g) - \mathcal{L}(u)\|_{L^2}^2 \\ &\leq -\gamma\|\mathcal{B}v_t\|_{L^2}^2 + \gamma k\|g\|_{\mathcal{H}^2}^2. \end{aligned} \quad (2.103)$$

From Lemma 2.2.12 there exists $\lambda > 0$ such that, for all $v \in L^2_\eta$, $\|\mathcal{B}v_t\|_{L^2}^2 \leq \lambda\|v_t\|_{\mathcal{B}}^2$. Using this value of λ , as $\|g\|_{\mathcal{H}^2}^2$ is a constant, $\frac{d}{dt}(\lambda\|v_t\|_{\mathcal{B}}^2 - k\|g\|_{\mathcal{H}^2}^2) \leq -\gamma\lambda(\|\mathcal{B}v_t\|_{L^2}^2 - k\|g\|_{\mathcal{H}^2}^2)$. Using the definition of λ and integrating,

$$\begin{aligned} \left(\|\mathcal{B}v_t\|_{L^2}^2 - k\|g\|_{\mathcal{H}^2}^2\right) &\leq \left(\lambda\|v_t\|_{\mathcal{B}}^2 - k\|g\|_{\mathcal{H}^2}^2\right) \\ &\leq \left(\lambda\|v_0\|_{\mathcal{B}}^2 - k\|g\|_{\mathcal{H}^2}^2\right) - \gamma\lambda \int_0^t \left(\|\mathcal{B}v_s\|_{L^2}^2 - k\|g\|_{\mathcal{H}^2}^2\right) ds. \end{aligned} \quad (2.104)$$

Writing $h_t = \int_0^t (\|\mathcal{B}v_s\|_{L^2}^2 - k\|g\|_{\mathcal{H}^2}^2) ds$, we see $dh_t/dt \leq (\lambda\|v_0\|_{\mathcal{B}}^2 - k\|g\|_{\mathcal{H}^2}^2) - \gamma\lambda h_t$, and hence Grönwall's inequality yields

$$h_t = \int_0^t \left(\|\mathcal{B}v_s\|_{L^2}^2 - k\|g\|_{\mathcal{H}^2}^2\right) ds \leq (\lambda\|v_0\|_{\mathcal{B}}^2 - k\|g\|_{\mathcal{H}^2}^2) \int_0^t e^{-\gamma\lambda(t-s)} ds. \quad (2.105)$$

Recalling that $Q_t - u = \mathcal{B}v_t + g$ and

$$\|\mathcal{B}v + g\|_{L^2}^2 \leq 2\|\mathcal{B}v\|_{L^2}^2 + 2\|g\|_{\mathcal{H}^2}^2 = 2(\|\mathcal{B}v\|_{L^2}^2 - k\|g\|_{\mathcal{H}^2}^2) + (2k + 2)\|g\|_{\mathcal{H}^2}^2, \quad (2.106)$$

we conclude that, for some constant $C > 0$,

$$\int_0^t \|Q_s - u\|_{L^2}^2 ds = \int_0^t \|\mathcal{B}v_s + g\|_{L^2}^2 ds \leq Ct\|g\|_{\mathcal{H}^2}^2 + C\|v_0\|_{\mathcal{B}}^2. \quad (2.107)$$

This inequality must hold for all choices of g and v_0 satisfying (2.100), and we can choose them to optimize (2.107). We observe that $Q_0 - u \in \mathcal{H}_{(0)}^2$, as $Q_0, u \in \mathcal{H}^2$ and they have the same boundary value. For every $\epsilon > 0$, by Theorem 2.2.10, there exists a choice of $v_0^{(\epsilon)} \in L_\eta^2$ such that $\|Q_0 - u - \mathcal{B}v_0^{(\epsilon)}\|_{\mathcal{H}^2} \leq \epsilon$. We further recall that $\|v_0^{(\epsilon)}\|_{\mathcal{B}}^2 < \infty$ for $v_0^{(\epsilon)} \in L_\eta^2$.

Defining $g^{(\epsilon)} = Q_0 - u - \mathcal{B}v_0^{(\epsilon)}$, (2.107) yields the limit as $t \rightarrow \infty$

$$\frac{1}{t} \int_0^t \|Q_s - u\|_{L^2}^2 ds \leq C \|g^{(\epsilon)}\|_{\mathcal{H}^2}^2 + \frac{C}{t} \|v_0^{(\epsilon)}\|_{\mathcal{B}}^2 \rightarrow C\epsilon. \quad (2.108)$$

As $\epsilon > 0$ was arbitrary, we conclude that $\lim_{t \rightarrow \infty} \left\{ \frac{1}{t} \int_0^t \|Q_s - u\|_{L^2}^2 ds \right\} = 0$.

To obtain the final statement, set $\beta(t) = \sup_{T \geq t} \left\{ \frac{1}{T} \int_0^T \|Q_s - u\|_{L^2}^2 ds \right\}$. Observe that β is a nonincreasing positive function with $\lim_{t \rightarrow \infty} \beta(t) = 0$. Defining the set $A = \{t : \|Q_t - u\|^2 > \sqrt{\beta(t)}\}$, Markov's inequality yields

$$\begin{aligned} \frac{1}{t} \text{Leb}(A \cap [0, t]) &= \frac{1}{t} \text{Leb}\left(\left\{s \leq t : \frac{\|Q_s - u\|_{L^2}^2}{\sqrt{\beta(s)}} > 1\right\}\right) \leq \frac{1}{t} \int_0^t \frac{\|Q_s - u\|_{L^2}^2}{\sqrt{\beta(s)}} ds \\ &\leq \frac{1}{\sqrt{\beta(t)}} \left(\frac{1}{t} \int_0^t \|Q_s - u\|_{L^2}^2 ds\right) \leq \sqrt{\beta(t)} \rightarrow 0. \end{aligned} \quad (2.109)$$

□

The above result only considers the convergence of Q . In the case where $\mathcal{L}$ is Gateaux differentiable (which is certainly the case when $\mathcal{L}$ is linear), we can give a similar result for the convergence of $\mathcal{L}Q_t$, involving the positive-definite kernel $\mathcal{B}$.

Theorem 2.4.3. *Suppose $\mathcal{L} : \mathcal{H}^2 \rightarrow L^2$ is Gateaux differentiable and the conditions of Theorem 2.4.2 hold. Then there exists $K > 0$ such that $\int_0^t \|\mathcal{B}\mathcal{L}Q_s\|_{L^2}^2 ds \leq K$. In particular, there exists a set $A \subset [0, \infty)$ with $\text{Leb}(A) < \infty$ such that for all sequences $t_k \rightarrow \infty$ which do not take values in A , we have $\|\mathcal{B}\mathcal{L}Q_{t_k}\|_{L^2} \rightarrow 0$.*

Proof. For any $w \in L^2$, $h \in \mathcal{H}^2$ we write $\partial\mathcal{L}(w; h) = \lim_{\epsilon \rightarrow 0} \{(\mathcal{L}(w + h) - \mathcal{L}(w))/\epsilon\}$ for the Gateaux derivative in direction h evaluated at w , the limit being taken in L^2 .

We first observe that, as $\mathcal{L}$ is strongly monotone, for any $h \in \mathcal{H}^2$,

$$\langle h, \partial\mathcal{L}(w; h) \rangle = \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} \langle h, \mathcal{L}(w + \epsilon h) - \mathcal{L}(w) \rangle \leq -\gamma \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon^2} \|\epsilon h\|_{L^2}^2 = -\gamma \|h\|_{L^2}^2. \quad (2.110)$$

Now write $v_t = \mathcal{L}Q_t$. As $Q_t \in \mathcal{H}^2$, we know that $\mathcal{B}v_t = \mathcal{B}\mathcal{L}Q_t \in \mathcal{H}^2$ (as $\mathcal{L} : \mathcal{H}^2 \rightarrow L^2$ and $\mathcal{B} : L^2 \rightarrow \mathcal{H}^2$). By the chain rule, we have $dv_t/dt = \partial\mathcal{L}(Q_t; dQ_t/dt) = \partial\mathcal{L}(Q_t; \mathcal{B}\mathcal{L}Q_t) = \partial\mathcal{L}(Q_t; \mathcal{B}v_t)$. Therefore,

$$\frac{d}{dt} \|v_t\|_{\mathcal{B}}^2 = 2 \left\langle \mathcal{B}v_t, \frac{dv_t}{dt} \right\rangle = 2 \left\langle \mathcal{B}v_t, \partial\mathcal{L}(Q_t; \mathcal{B}v_t) \right\rangle \leq -2\gamma \|\mathcal{B}v_t\|_{L^2}^2. \quad (2.111)$$

Lemma 2.2.12 gives the bound $\|\mathcal{B}v_t\|_{L^2}^2 \leq \lambda\|v_t\|_{\mathcal{B}}^2$, from which we obtain

$$\|\mathcal{B}v_t\|_{L^2}^2 \leq \lambda\|v_t\|_{\mathcal{B}}^2 \leq \lambda\|v_0\|_{\mathcal{B}}^2 - 2\lambda\gamma \int_0^t \|\mathcal{B}v_s\|^2 ds. \quad (2.112)$$

By Grönwall's inequality, we conclude

$$\int_0^t \|\mathcal{B}v_s\|_{L^2}^2 ds \leq \lambda\|v_0\|_{\mathcal{B}}^2 \int_0^t e^{-2\lambda\gamma(t-s)} ds \leq \frac{\|v_0\|_{\mathcal{B}}^2}{2\gamma} =: K < \infty. \quad (2.113)$$

Taking $A = \{t : \|\mathcal{B}v_t\|_{L^2}^2 > 1/t\}$, the final stated property is obtained from Markov's inequality, similarly to in Theorem 2.4.2. $\square$

Corollary 2.4.4. *Under the conditions and notation of Theorem 2.4.3, the sequence $\mathcal{L}Q_{t_k}$ converges weakly to zero in L^2 if and only if it remains bounded in L^2 .*

Proof. Recall that a sequence x_n converges L^2 -weakly to zero if and only if it is L^2 -bounded and there exists an L^2 -dense set $\mathcal{Y}$ such that $\langle y, x_n \rangle \rightarrow 0$ for all $y \in \mathcal{Y}$.

We know from Theorem 2.4.3 that $\mathcal{B}\mathcal{L}Q_{t_k} \rightarrow 0$ strongly in L^2 so, for any $v \in L^2$, $\langle \mathcal{B}v, \mathcal{L}Q_{t_k} \rangle = \langle v, \mathcal{B}\mathcal{L}Q_{t_k} \rangle \rightarrow 0$. In particular, we see that $\langle y, Q_{t_k} \rangle \rightarrow 0$ for any $y \in \text{im}(\mathcal{B})$. We know from Theorem 2.2.10 that $\text{im}(\mathcal{B})$ is dense in $\mathcal{H}^2$, and hence in L^2 . The result follows. $\square$

2.5 Numerical experiments

In this section, we present numerical results where we apply our algorithm to solve a family of partial differential equations. The approximator matches the solution of the differential equation closely in a relatively short period of training time in these test cases.

2.5.1 Discretization of the continuous-time algorithm

As a reminder, in a continuous time setting, the algorithm follows a biased gradient flow: $d\theta_t/dt = -\alpha_t^N G_t^N(\theta_t)$, based on the biased gradient estimator

$$G_t^N(\theta_t) = \int_{\Omega} 2\psi(\mathcal{L}Q_t^N(x))\psi'(\mathcal{L}Q_t^N(x))\nabla_{\theta}(-\gamma Q_t^N(x))\mu(dx). \quad (2.114)$$

A natural discretization of our algorithm with which we train our approximator is

$$\theta_{t+1} - \theta_t = -\alpha_t^N \hat{G}_{M,t}^N, \quad (2.115)$$

where

$$G_{M,t}^{\hat{N}} = \frac{1}{M} \sum_{i=1}^M 2\psi(\mathcal{L}Q_t^N(x_t^i))\psi'(\mathcal{L}Q_t^N(x_t^i))\nabla_{\theta}(-\gamma Q_t^N(x_t^i)) \quad (2.116)$$

is an unbiased estimator of G_t^N , given $\{x_t^i\}_{i=1,2,\dots,M}$ the random grid independently sampled from distribution μ at time step t , and α_t^N is the learning rate. More advanced discretization methods, for example the ADAM adaptive gradient descent rule ([69]) can also be used, and (as is common in gradient based optimization problems) give a noticeable improvement in performance.

The Q-learning algorithm for solving PDEs as follows:

Algorithm 1: Q-PDE Algorithm

Parameters: Hyper-parameters of the single-layer neural network; Domain Ω ; PDE operator $\mathcal{L}$; Boundary condition at $\partial\Omega$; Sampling measure μ ; Number of Monte Carlo points M ;

Upper bound of training time T ;

Initialise: Neural net S^N ; Auxiliary function η ; Approximator Q^N based on S^N and η ; Smoothing function ψ^N ; Learning rate scheduler $\{\alpha_t^N\}_{t \geq 0}$;

Stopping criteria ϵ ; Current time $t = 0$.

while $err \geq \epsilon$ and $t \leq T$ **do**

Sample M points in Ω using μ , $\{x_i\}$;
 Compute biased gradient estimator $G_{M,t}^N$ using (2.116);
 Update neural network parameters via (2.115);
 Compute $err = \frac{1}{M} \sum_{i=1}^M \psi^N(\mathcal{L}Q_t^N(x_i))^2$;
 Update time t ;

end

Return approximator Q_t^N .

We implement this algorithm using PyTorch (with backpropagation to compute gradients in both parameters and the state variable x) and the standard ADAM optimization scheduler for α .²

2.5.2 Test equation I: Survival time of a Brownian motion

Suppose Ω is the n -dimensional unit ball. For n -dimensional Brownian motion X_t starting from point $x \in \mathbb{R}^n$, we wish to compute

$$u(x) := \mathbb{E} \left[\int_0^{\tau_{\Omega}} e^{-\gamma t} dt \middle| X_0 = x \right] \quad (2.117)$$

where τ_{Ω} is the first exit time of X for domain Ω . By the Feynman–Kac theorem (applied to the process X stopped at τ_{Ω}), this expectation is given by the solution of

²The implementation is available at <https://github.com/DeqingJ/QPDE>.

the following PDE:

$$\begin{cases} 1 - \gamma u + \frac{1}{2}\Delta u &= 0 & \text{in } \Omega \\ u &= 0 & \text{on } \partial\Omega, \end{cases} \quad (2.118)$$

where Δu is the Laplacian of u . We consider this problem as a good test case as its explicit solution is known to us, allowing us to check if our algorithm works for a high-dimensional version of this PDE. We initialize a single neural network S^N equipped with sigmoid activation function, and introduce auxiliary function $\eta(x) = 1 - \|x\|^2$. The approximator $Q = S \cdot \eta$ is trained to fit the solution. As shown in the following subsections, through our algorithm, the approximator Q learns to have the solution of the PDE with different discounting factor in different dimensions. For detailed configuration of our approximator, see Appendix 2.6.8.

2.5.2.1 1-dimensional case

For dimension $n = 1$, domain $\Omega = (-1, 1)$. The exact solution of this differential equation is $u(x) = \frac{1}{\gamma} + c_1(e^{\sqrt{2\gamma}x} + e^{-\sqrt{2\gamma}x})$ where $c_1 = -1/(\gamma(e^{-\sqrt{2\gamma}} + e^{\sqrt{2\gamma}}))$. For this subsection, set $\gamma = 0.1$. To keep track of the training progress, we monitor the average loss level at time t , that is $e_t \approx \int_{\Omega} [\mathcal{L}Q_t]^2 d\mu$ where μ is the Lebesgue measure on Ω , which can be estimated using our sample of evaluation points at each time.

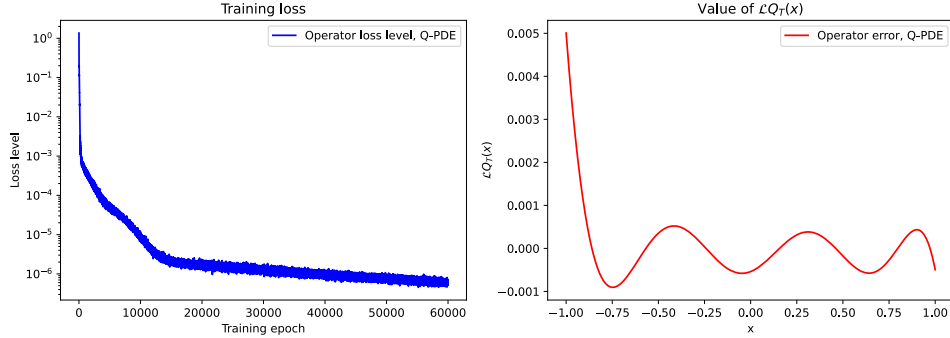


Figure 2.1: Loss level during and after training

The left side of Figure 2.1 shows how the average operator loss decays during training. The ‘elbow’ shape of the loss curve can be a result of our learning rate choice which decays ($\sim \frac{1}{1+t}$) as training proceeds and the landscape in the parameter space. In the first few hundred steps, L^2 loss decays faster. In later stages, loss decays much slower as the neural network requires a lot more steps to further improve its performance. The right side of it plots the operator error at terminal time

$\mathcal{L}Q_T^N(x)$. For a perfect fit, the operator loss $\mathcal{L}Q_T(x)$ should be zero across the entire domain. We observe that for our approximator, the norm of this loss remains small and fluctuates around 0. In Figure 2.2, we compare our approximator Q_τ^N with the exact solution. Here τ indicates the training time at which we observed the minimal loss e_τ , which occurs for τ close to the terminal time T . (Given the fluctuations observed in the loss, choosing this minimal point can give a noticeable qualitative improvement in performance.) The relative error of our approximation at x , given by $|(Q_\tau^N(x) - u(x))/u(x)|$, remains lower than 0.02%.

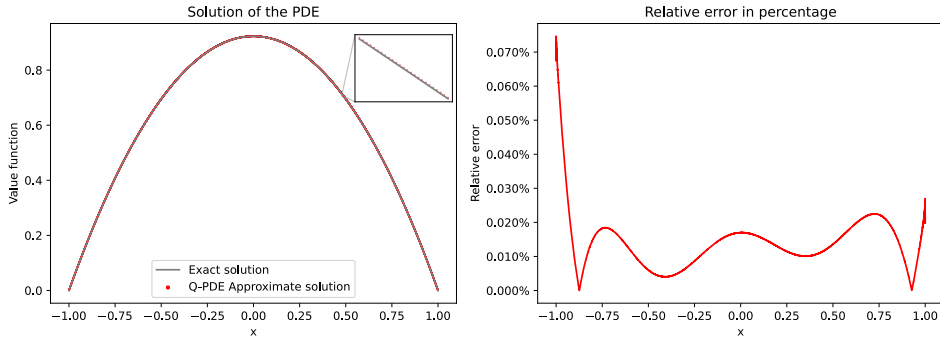


Figure 2.2: Approximate and exact solutions in 1d case

2.5.2.2 20-dimensional case

Now we consider the 20-dimensional case. High-dimensional PDEs are typically hard to solve, as mesh grids are computationally unaffordable in these cases. However, as a mesh-free method, our algorithm can be applied. We numerically solve the PDE (2.118) for discounting factor $\gamma = 0.2$ via our algorithm and compare the result with the exact solution, given by

$$u(x) = \frac{1}{\gamma} \left(1 - \frac{J_9(i\sqrt{2\gamma}\|x\|)}{\|x\|^9 J_9(i\sqrt{2\gamma})} \right), \quad (2.119)$$

where J_9 is the ninth-order Bessel function of the first kind. We also compare the training results of our method to the results from the Deep Galerkin Method (which does stochastic gradient descent with an unbiased gradient estimate) as a benchmark.

Our approximator trained with the Q-PDE algorithm is able to accurately approximate the solution of the 20-dimensional PDE. Left of Figure 2.3 shows that the operator loss is less than 10^{-5} after training for 40,000 epochs, for both Q-PDE and DGM. For both methods, we observe cyclic behavior in the loss during training, which

appears similar to the slingshot effect discussed in [146], which is potentially a consequence of momentum accumulation of the ADAM optimizer. A full understanding of this effect (which we have not observed when using a simple gradient descent optimizer) is left for future work. In comparison, the erratic convergence effect is absent in Figure 2.4, where SGD optimizer is applied with other training configurations remain the same. With the same number of training steps, the averaged training loss of SGD is much larger than that of ADAM. Hence, though ADAM brings in oscillation effects, it still outperforms SGD in this case.

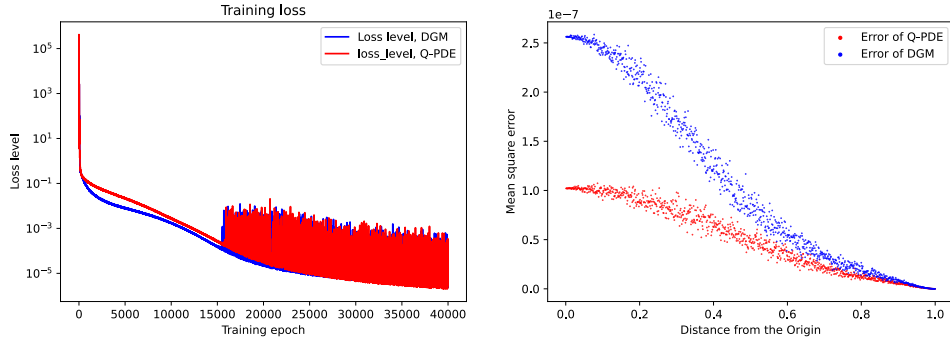


Figure 2.3: Average loss level during training with ADAM

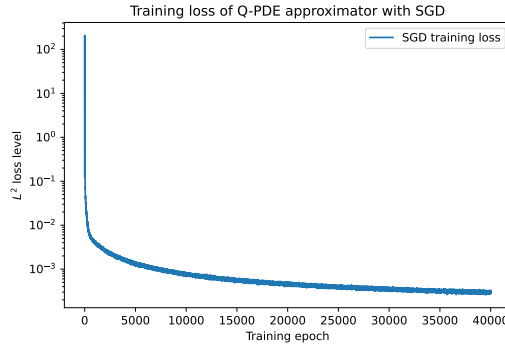


Figure 2.4: Average loss level during training with SGD

The solution u of this PDE is radially symmetric, that is, it can be represented by a function which depends on $\|x\|$ rather than x . For the right side of Figure 2.3, we verify that our approximate solution is close to this function for different $\|x\| \in [0, 1]$. For $0 \leq r \leq 1$, we randomly generate 100 sample points $\{x_{j,r}\}_{j=1,\dots,100}$ on the sphere $\|x\| = r$ and compute the mean-squared error of the set of values $\{Q_\tau^N(x_{i,r})\}$ as compared to the exact solution. It is shown that for different r , the

mean-squared error of the neural network approximator is consistently small. Q-PDE solution obtains smaller error than our DGM solution, but they are at the same scale.

For $0 \leq r \leq 1$, we also consider the maximum relative error, defined as

$$R_r = \max_{x_{i,r}} \left\{ \left| \frac{Q_\tau^N(x_{i,r}) - u(x_{i,r})}{u(x_{i,r})} \right| \right\}. \quad (2.120)$$

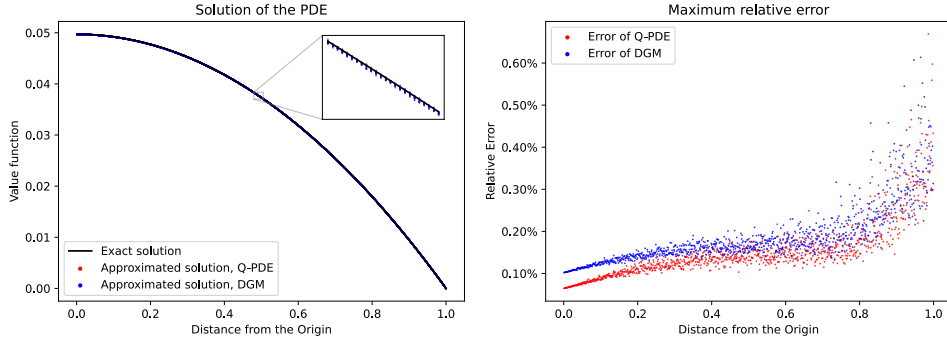


Figure 2.5: Relative and absolute errors of the approximate solution, in terms of distance from the origin.

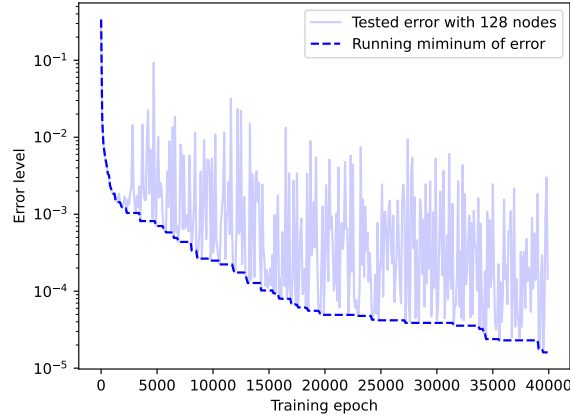


Figure 2.6: Actual loss level of the NN solution during training, compared to exact solution

The left plot of Figure 2.5 implies that this error remains small for all $0 \leq r \leq 1$. Again, DGM solution admits slightly larger relative error. Meanwhile, as DGM solution is trained with exact gradients which are computationally more expensive, consequently, it takes 5 times longer to train. Both DGM and Q-PDE solutions reach

high accuracy. We plot a scatter plot for $\{Q_\tau^N(w_i)\}$ in the right side in Figure 2.5 and compare with the exact solution u . The approximate solutions closely matches the exact solution.

In Figure 2.6, the averaged error level over 10^4 uniformly sampled points in Ω (comparing to the exact solution) is shown. The error also fluctuates erratically, similar to the effect we observed on training losses. The minimum of error decays as training goes on, and reaches $\sim 10^{-5}$ at the end of it. As a sanity check, notice that since the test PDE is (uniformly) elliptic, the training residual term e and neural network approximator Q satisfies the PDE

$$\begin{aligned}\mathcal{L}(Q - u) &= e, & \Omega, \\ Q - u &= 0, & \partial\Omega.\end{aligned}\tag{2.121}$$

Here, $\mathcal{L} := -\gamma + \frac{1}{2}\Delta$ is the PDE operator, and $Q - u$ is the error term. Therefore, by Shauder estimation introduced in [30] there exists constant $C > 0$ such that

$$\|Q - u\|_2 \leq C\|e\|_2\tag{2.122}$$

which coincides with the results we have in Figure 2.6.

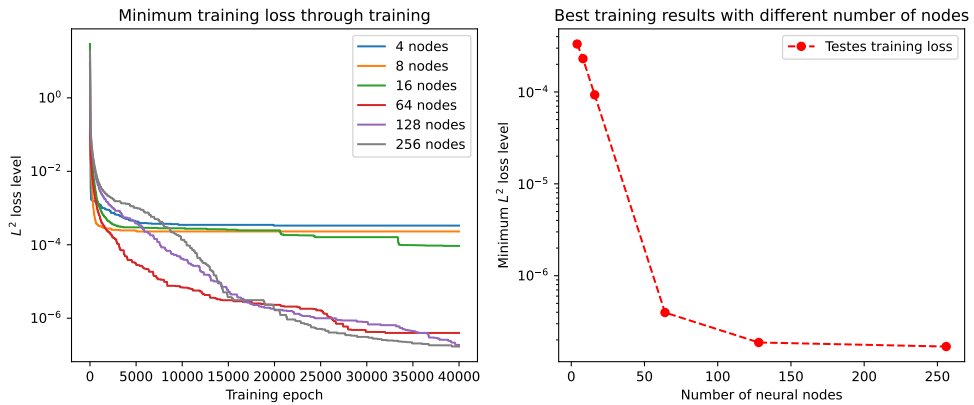
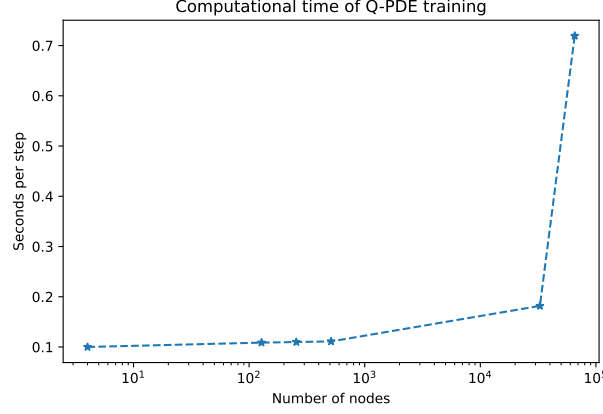


Figure 2.7: Training loss level of NNs with 6 different number of nodes

Accuracy results for single-layer neural networks with different number of neuron nodes are tested, and they are shown in Figure 2.7. For a small number of nodes, increasing the number of nodes significantly makes the approximation more accurate. However, as the number of nodes further increases, the accuracy of single-layer approximator only improves marginally. Apart from MC error (notice that the loss is sampled on 10^4 points) which potentially affects the results, it is worth pointing out

that it reflects the neural network approximation power result introduced on page 2, Chapter 1. Practically, single-layer networks may not be the best option of doing approximations, as the number of nodes required for higher accuracy may expand very rapidly.



2.5.3 Test equation II: A nonconstant coefficient PDE

In this section, we consider a slightly more complicated test case. Suppose Ω is the n -dimensional unit ball. The total running cost of n -dimensional process X_t is given by

$$V(x) = \mathbb{E}_x \left[\int_0^{\tau_\Omega} e^{-\gamma t} (1 + \kappa \sin(X_t^1) X_t^2) dt \right], \quad (2.123)$$

where X_t^i is the i -th component of X_t . X_t is governed by the following SDE

$$dX_t = \sigma(X_t) dW_t, \quad X_0 = x \in \Omega \quad (2.124)$$

where W_t is n -dimensional Brownian motion, and volatility term $\sigma(X_t)$ is given by

$$\sigma(X_t) = \text{diag}\{1, 1, X_t^3, 1, \dots, 1\}. \quad (2.125)$$

By Feynman–Kac formula, value function $V(x)$ is the solution of the linear, nonconstant coefficient PDE

$$1 + \kappa \sin(x_1) x_2 - \gamma V(x) + \sum_{i=1}^n \sigma_{ii}^2(x) \partial_{ii}^2 V(x) = 0, \quad x \in \Omega \quad (2.126)$$

$$V(x) = 0, \quad x \in \partial\Omega.$$

Remark 2.5.2. The source terms and the coefficient terms are not constant and not symmetric, therefore the solution of this PDE is no longer radially symmetric in n dimension. However, this PDE is still monotone, hence it serves as an effective example of checking our algorithm’s functionality. Again, we are not showing that single-layer networks are a good choice for approximating high-dimensional PDE solutions, but are solely examining whether this algorithm, equipped with single-layer neural networks, can work for a monotone PDE whose solution does not have an one-dimensional representation.

2.5.3.1 Monte Carlo solution

While a closed-form solution is not available, we apply MC simulations to obtain an approximation of PDE solution. Process (2.124) is simulated via Euler-Maruyama scheme

$$\Delta X_{t_n} = \sigma(X_{t_n}) \Delta W_{t_n} \quad (2.127)$$

and the integral is approximated by its discretization on each sample paths

$$\sum_j \mathbf{1}(t_j < \tau) e^{-\gamma t_j} (1 + \kappa \sin(X_{t_j}^1) X_{t_j}^2) \Delta t_j. \quad (2.128)$$

2.5.3.2 Neural network approximator configuration

In the following numerical test, we choose dimension $n = 6$, and PDE parameters $\kappa = 1, \gamma = 0.2$. The neural network approximator is constructed by $Q = S \cdot \eta$ where S is a single-layer neural network with 256 nodes, parameters initialized in match with Assumption 2.1.16. $\eta(x) = 1 - \|x\|_2^2$ is the auxiliary function pasting the boundary condition into approximation. To train the approximator, ADAM optimizer is applied and the initial learning rate is set at $l = 0.1$.

2.5.3.3 Test outcome

Given that the closed-form solution is not available, we compared the approximation of training neural network with MC simulation results. The training process (with the same reason explained by the end of last subsection, our code is not optimized and hence takes much longer than it would take after code restructuring) takes ~ 1.5 hrs on a single consumer GPU to finish. For MC solution, we uniformly sampled 200 points in the domain and computed the error on each of these points. Each MC solution point is calculated on 10^5 sample paths with time step size $\Delta t = 10^{-5}$. Each MC sample takes ~ 2 minutes to run on the same GPU, and the total MC computing time is ~ 7 hrs.

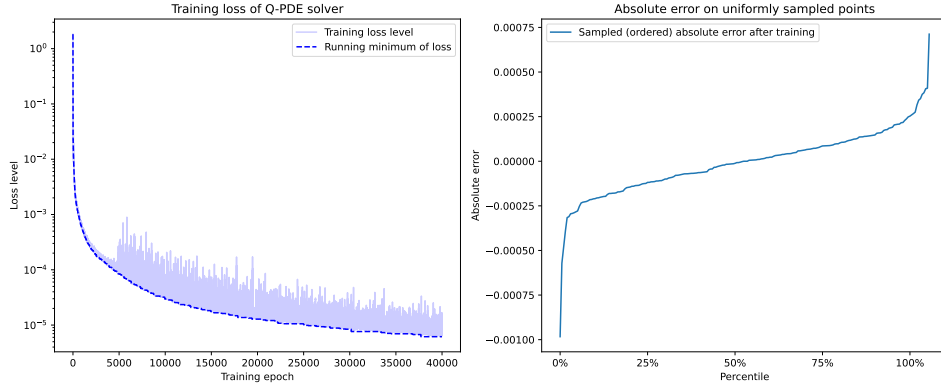


Figure 2.9: Training loss of Q-PDE approximator solving (2.126) and its absolute error level

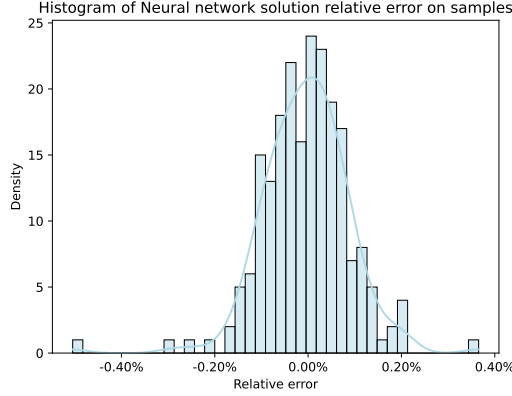


Figure 2.10: Training loss of Q-PDE approximator solving (2.126) and its absolute error level

On the left side of Figure 2.9, the loss over training periods is displayed. After training, training loss level $\sim 10^{-5}$ to 10^{-6} is achieved. On the right side, the percentile plot of absolute error comparing to the MC solution is given. On these samples, the absolute error ranges between $\pm 10^{-3}$, which indicates that we have obtained an approximation at an acceptable error level. In Figure 2.10, a histogram of the approximator's relative error is shown. Except for 2 points with larger relative error, on most of the points the relative error level remains bounded by $\pm 0.25\%$. It is possible to expand the number of neuron nodes in the approximator and pursue a better accuracy. However, similar to the previous test case, we expect the number of nodes required to grow rapidly, as potentially single-layer networks are not the best class for approximator.

2.6 Proofs and additional information

In our proofs, the constants K , C may vary from line to line.

2.6.1 Proof of Lemma 2.1.3

Proof. We first prove³ statement *i*. Suppose that $u \in \mathcal{H}^2(\Omega) \cap \mathcal{H}_0^1(\Omega)$. Consider the Poisson problem solved by u :

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \quad (2.129)$$

³Thanks to Endre Süli for suggesting the proof of statement *i* given here.

where $f := -\Delta u \in L^2(\Omega)$. Let $f_n \in C_0^\infty(\Omega)$ be a sequence with $f_n \rightarrow f \in L^2(\Omega)$. Then consider the Poisson problem

$$\begin{cases} -\Delta u_n = f_n & \text{in } \Omega, \\ u_n = 0 & \text{on } \partial\Omega. \end{cases} \quad (2.130)$$

Since (by Assumption 2.1.1) $\partial\Omega \in C^{3,\alpha}$ for a Hölder exponent $\alpha \in (0, 1)$ and $f_n \in C_0^\infty(\Omega) \subset C^{1,\alpha}(\overline{\Omega})$, by the Schauder theory for elliptic boundary-value problems (see, for example, Theorems 6.14 and 6.19 in [33]), there exists a unique solution $u_n \in C^{3,\alpha}(\overline{\Omega}) \cap C_0(\overline{\Omega})$ to this boundary-value problem. As

$$\begin{cases} -\Delta(u - u_n) = (f - f_n) & \text{in } \Omega, \\ u - u_n = 0 & \text{on } \partial\Omega, \end{cases} \quad (2.131)$$

and $\partial\Omega \in C^{3,\alpha}$ (and therefore also C^2), and $u - u_n \in \mathcal{H}^2(\Omega)$, it follows from Theorem 3.1.2.1 and Remark 3.1.2.2 in [39], that for some $C > 0$

$$\|u - u_n\|_{\mathcal{H}^2} \leq C \|\Delta(u - u_n)\|_{L^2} = C \|f - f_n\|_{L^2} \rightarrow 0 \quad (2.132)$$

as $n \rightarrow \infty$. Thus, we have constructed a sequence of functions $u_n \in C^{3,\alpha}(\overline{\Omega}) \cap \mathcal{H}_0^1$ which converges to $u \in \mathcal{H}^2(\Omega)$ in the $\mathcal{H}^2$ norm. Therefore, $C^{3,\alpha}(\overline{\Omega}) \cap C_0(\overline{\Omega})$ is dense in $\mathcal{H}^2(\Omega) \cap \mathcal{H}_0^1(\Omega)$. As

$$\left(C^{3,\alpha}(\overline{\Omega}) \cap C_0(\overline{\Omega})\right) \subset \left(C^3(\overline{\Omega}) \cap C_0(\overline{\Omega})\right), \quad (2.133)$$

we conclude that $C^3(\overline{\Omega}) \cap C_0(\overline{\Omega})$ is also dense in $\mathcal{H}^2(\Omega) \cap \mathcal{H}_0^1(\Omega)$.

We now prove statement *ii*. Take an arbitrary $u \in C^3(\overline{\Omega}) \cap C_0(\overline{\Omega})$, and define $\tilde{u} = u/\eta$. It is clear that $\tilde{u}|_\Omega$ is in $C^3(\Omega)$, from classical rules of calculus, as $\eta|_\Omega > 0$. In particular, $\tilde{u}$ and its derivatives are bounded on any closed subset of Ω , so we only need to consider the behavior of $\tilde{u}$ in a neighborhood of $\partial\Omega$.

We know that η has nonvanishing derivative on $\partial\Omega$, and equals zero on $\partial\Omega$, which implies that $|\langle \nabla\eta(x), \mathbf{n}_x \rangle| > 0$ for each $x \in \partial\Omega$, where $\mathbf{n}_x$ is the outward pointing normal at x . As $\partial\Omega$ is compact, this implies that

$$\inf_{x \in \partial\Omega} |\langle \nabla\eta(x), \mathbf{n}_x \rangle| > \epsilon, \quad \text{for some } \epsilon > 0. \quad (2.134)$$

As $u \in C^3(\overline{\Omega}) \cap C_0(\overline{\Omega})$, for any $x \in \partial\Omega$, we also have $\limsup_{x' \rightarrow x} |u(x')| = 0$.

Consider a smooth path $\rho : [0, 1) \rightarrow \Omega$ with $\lim_{t \rightarrow 1} \rho(t) = x \in \partial\Omega$. For a function g with domain Ω (e.g. $g = \eta$, u or $\tilde{u}$), we write g_ρ for the composition $g \circ \rho$ with domain $[0, 1)$. Suppose ρ does not approach $\partial\Omega$ tangentially, in particular,

$$\lim_{t \rightarrow 1} \langle d\rho/dt, \mathbf{n}_x \rangle \geq \epsilon \quad \text{and} \quad \|d\rho/dt\| \equiv 1. \quad (2.135)$$

As $\langle \nabla \eta(x), y \rangle = 0$ for all $y \perp \mathbf{n}_x$, we have

$$\lim_{t \rightarrow 1} |\eta'_\rho(t)| = |\langle \nabla \eta(x), \mathbf{n}_x \rangle| \langle d\rho/dt, \mathbf{n}_x \rangle > \epsilon^2. \quad (2.136)$$

Using L'Hôpital's rule, we can determine the behaviour of $\tilde{u}_\rho$ (and its derivatives) as $t \rightarrow 1$. In particular, by repeated application of the rule, as $t \rightarrow 1$

$$\tilde{u}_\rho = \frac{u_\rho}{\eta_\rho} \rightarrow \frac{u'_\rho}{\eta'_\rho} \quad (2.137)$$

$$\tilde{u}'_\rho = \frac{\eta_\rho u'_\rho - \eta'_\rho u_\rho}{\eta_\rho^2} \rightarrow \frac{u''_\rho}{2\eta'_\rho} - \frac{\eta''_\rho}{2\eta'_\rho} \tilde{u}_\rho \quad (2.138)$$

$$\tilde{u}''_\rho = \frac{\eta_\rho^2 u''_\rho - \eta_\rho \eta''_\rho u_\rho - 2\eta_\rho \eta'_\rho u'_\rho + 2(\eta'_\rho)^2 u_\rho}{\eta_\rho^3} \rightarrow \frac{u'''_\rho}{3\eta'_\rho} - \frac{\eta'''_\rho}{3\eta'_\rho} \tilde{u}_\rho + \frac{\eta''_\rho}{\eta'_\rho} \tilde{u}'_\rho \quad (2.139)$$

Considering $t \approx 1$, as $|\eta'_\rho| > \epsilon^2$, the right hand side of each of these terms is bounded, with a uniform bound for all paths under consideration. Writing $D_h(\tilde{u})$ for the directional derivative of $\tilde{u}$ in a direction h , we know $\|\tilde{u}'_\rho\| = \|D_{d\rho/dt}(\tilde{u})\|$ and $\|\tilde{u}''_\rho\| = \|D_{d\rho/dt}^2(\tilde{u})\|$. We can therefore find a single value $K > 0$, and take a ball B_x around each $x \in \partial\Omega$, such that, within each B_x , for all unit vectors h satisfying $\langle h, \mathbf{n}_x \rangle > \epsilon$,

$$|\tilde{u}| + |D_h(\tilde{u})| + |D_h^2(\tilde{u})| \leq K. \quad (2.140)$$

As $\Omega \setminus (\cup_{x \in \partial\Omega} B_x)$ is compact, we know that, for K sufficiently large, (2.140) also holds on $\Omega \setminus (\cup_{x \in \partial\Omega} B_x)$, for all unit vectors h . We conclude that (2.140) holds on all of Ω , that is, $\tilde{u}$, and its directional first and second derivatives (except possibly in tangent directions at the boundary), are uniformly bounded.

However, as $\partial\Omega$ is a $C^{3,\alpha}$ boundary (in particular, for all ϵ sufficiently small and any $x \in \partial\Omega$ the set $\{y \in \mathbb{R}^n : \langle y, \mathbf{n}_x \rangle \geq \epsilon\}$ is a convex cone containing a linear basis for $\mathbb{R}^n$), this implies that the first and second derivatives of $\tilde{u}$ are also uniformly bounded on Ω . We conclude that $\tilde{u} \in C_b^2(\Omega)$. $\square$

2.6.2 Proof of Lemma 2.2.1

Proof. (1) From the training algorithm, calculating the entries in vector $\nabla_\theta Q_t^N(x)$, we have

$$\frac{\partial Q_t^N}{\partial c^i}(x) = \frac{\eta(x)\sigma(w_t^i \cdot x + b_t^i)}{N^\beta}. \quad (2.141)$$

Therefore, by (2.17), applying the chain rule we derive

$$\frac{dc_t^i}{dt} = \alpha N^{\beta-1} \int_\Omega F^N(\mathcal{L}Q_t^N(x)) \eta(x) \sigma(w_t^i \cdot x + b_t^i) \mu(dx). \quad (2.142)$$

By the boundedness of the functions F^N , η and σ , the RHS of (2.142) is bounded:

$$\left| \int_{\Omega} F^N(\mathcal{L}Q_t^N(x))\eta(x)\sigma(w_t^i \cdot x + b_t^i)\mu(dx) \right| < KN^{\delta}, \quad (2.143)$$

where K is a constant. Therefore, we conclude that

$$|c_t^i| < |c_0^i| + |c_t^i - c_0^i| < K_0 + \alpha N^{\delta+\beta-1}tK < C. \quad (2.144)$$

(2) Similar to above, considering the w^i entries in the vector $\nabla_{\theta}Q_t^N(x)$, we have

$$\frac{\partial Q_t^N}{\partial (w_t^i)_k}(x) = \frac{\eta(x)\sigma'(w_t^i \cdot x + b_t^i)c_t^i x_k}{N^{\beta}}. \quad (2.145)$$

Therefore w_t^i satisfies:

$$\frac{d(w_t^i)_k}{dt} = \alpha N^{\beta-1} \int_{\Omega} F^N(\mathcal{L}Q_t^N(x))\eta(x)c_t^i \sigma'(w_t^i \cdot x + b_t^i)x_k \mu(dx). \quad (2.146)$$

By the boundedness of F^N , η , σ' and c_t^i , the RHS of (2.146) is bounded by

$$\left| \int_{\Omega} F^N(\mathcal{L}Q_t^N(x))\eta(x)c_t^i \sigma'(w_t^i \cdot x + b_t^i)x_k \mu(dx) \right| < KN^{\delta}, \quad (2.147)$$

where K is a constant. Therefore,

$$|(w_t^i)_k| < |(w_0^i)_k| + |(w_t^i)_k - (w_0^i)_k| < |(w_0^i)_k| + \alpha N^{\delta+\beta-1}tK < |(w_0^i)_k| + C. \quad (2.148)$$

Taking an expectation on both sides, we obtain

$$\mathbb{E}|(w_t^i)_k| < C. \quad (2.149)$$

(3) Considering the b^i entries in the vector $\nabla_{\theta}Q_t^N(x)$, we have

$$\frac{\partial Q_t^N}{\partial b_t^i}(x) = \frac{\eta(x)\sigma'(w_t^i \cdot x + b_t^i)c_t^i}{N^{\beta}}. \quad (2.150)$$

Therefore, b_t^i satisfies

$$\frac{db_t^i}{dt} = \alpha N^{\beta-1} \int_{\Omega} F^N(\mathcal{L}Q_t^N(x))\eta(x)c_t^i \sigma'(w_t^i \cdot x + b_t^i)\mu(dx). \quad (2.151)$$

By the boundedness of F^N , η , σ' and c_t^i , the RHS of (2.146) is bounded by

$$\left| \int_{\Omega} F^N(\mathcal{L}Q_t^N(x))\eta(x)c_t^i \sigma'(w_t^i \cdot x + b_t^i)\mu(dx) \right| < KN^{\delta}, \quad (2.152)$$

where K is a constant. Therefore,

$$|b_t^i| < |b_0^i| + |b_t^i - b_0^i| < |b_0^i| + \alpha N^{\delta+\beta-1}tK < |b_0^i| + C. \quad (2.153)$$

Taking an expectation on both sides, we obtain

$$\mathbb{E}|b_t^i| < C. \quad (2.154)$$

□

2.6.3 Proof of Lemma 2.3.1

Proof. (1) Notice that, since

$$\begin{aligned}
& |A_t^N(x, y) - A_0^N(x, y)| \\
&= \left| \frac{1}{N} \sum_{i=1}^N \left[\sigma(w_t^i \cdot x + b_t^i) \sigma(w_t^i \cdot y + b_t^i) - \sigma(w_0^i \cdot x + b_0^i) \sigma(w_0^i \cdot y + b_0^i) \right. \right. \\
&\quad \left. \left. + x \cdot y (C_t^i)^2 \sigma'(w_t^i \cdot x + b_t^i) \sigma'(w_t^i \cdot y + b_t^i) \right. \right. \\
&\quad \left. \left. - x \cdot y (C_0^i)^2 \sigma'(w_0^i \cdot x + b_0^i) \sigma'(w_0^i \cdot y + b_0^i) \right] \right|, \tag{2.155}
\end{aligned}$$

We have

$$\begin{aligned}
& |A_t^N(x, y) - A_0^N(x, y)| \\
&\leq \frac{1}{N} \sum_{i=1}^N \left[\left| \sigma(w_t^i \cdot x + b_t^i) \sigma(w_t^i \cdot y + b_t^i) - \sigma(w_0^i \cdot x + b_0^i) \sigma(w_0^i \cdot y + b_0^i) \right| \right. \\
&\quad \left. + |x \cdot y| |c_t^i|^2 - c_0^i|^2 \|\sigma'(w_t^i \cdot x + b_t^i) \sigma'(w_t^i \cdot y + b_t^i)\| \right. \\
&\quad \left. + c_0^i|^2 |x \cdot y| \|\sigma'(w_t^i \cdot x + b_t^i) \sigma'(w_t^i \cdot y + b_t^i) - \sigma'(w_0^i \cdot x + b_0^i) \sigma'(w_0^i \cdot y + b_0^i)\| \right]. \tag{2.156}
\end{aligned}$$

For a smooth function $f : \mathbb{R} \rightarrow \mathbb{R}$, applying the mean value theorem with respect to w_t^i , there exists $w^* \in \mathbb{R}^n$ such that

$$\begin{aligned}
& f(w_t^i \cdot x + b) f(w_t^i \cdot y + b) - f(w_0^i \cdot x + b) f(w_0^i \cdot y + b) \\
&= (w_t^i - w_0^i) \cdot \left[f'(w^* \cdot x + b) f(w^* \cdot y + b) x + f(w^* \cdot x + b) f'(w^* \cdot y + b) y \right]. \tag{2.157}
\end{aligned}$$

By the fact that $\sigma \in C_b^4(\mathbb{R})$ and $|c_u^i| < C$ for any $u \in [0, T]$, we have

$$\begin{aligned}
& |A_t^N(x, y) - A_0^N(x, y)| \leq \frac{C}{N} \sum_{i=1}^N \left[(\|x\|_1 + \|y\|_1) \|w_t^i - w_0^i\|_1 + |x \cdot y| |c_t^i - c_0^i| \right. \\
&\quad \left. + \|w_t^i - w_0^i\|_1 (\|x\|_1 + \|y\|_1) |x \cdot y| + (1 + |x \cdot y|) |b_t^i - b_0^i| \right]. \tag{2.158}
\end{aligned}$$

Therefore, by (2.143), (2.152) and (2.147), the increments of c_t , b_t and w_t are bounded by $CN^{\delta+\beta-1}$. Therefore, we have

$$|A_t^N(x, y) - A_0^N(x, y)| \leq CN^{\delta+\beta-1} \left[(\|x\|_1 + \|y\|_1) (1 + |x \cdot y|) + |x \cdot y| \right] \leq CN^{\delta+\beta-1}. \tag{2.159}$$

(2) By definition,

$$\begin{aligned}\partial_{y_i} A_t^N(x, y) &= \frac{1}{N} \sum_{k=1}^N \left[\sigma(w_t^k \cdot x + b_t^k) \sigma'(w_t^k \cdot x + b_t^k) (w_t^k)_i \right. \\ &\quad + c_t^{k^2} \sigma'(w_t^k \cdot x + b_t^k) \sigma'(w_t^k \cdot y + b_t^k) x_i \\ &\quad \left. + c_t^{k^2} \sigma'(w_t^k \cdot x + b_t^k) \sigma''(w_t^k \cdot y + b_t^k) x \cdot y (w_t^k)_i \right].\end{aligned}\tag{2.160}$$

Similarly to in (2.155), we split the difference between $\partial_{y_i} A_t^N(x, y)$ and $\partial_{y_i} A_0^N(x, y)$ into terms

$$\begin{aligned}&|\partial_{y_i} A_t^N(x, y) - \partial_{y_i} A_0^N(x, y)| \\ &\leq \frac{1}{N} \sum_{k=1}^N \left[\left| \sigma(w_t^k \cdot x + b_t^k) \sigma'(w_t^k \cdot x + b_t^k) (w_t^k)_i - \sigma(w_0^k \cdot x + b_0^k) \sigma'(w_0^k \cdot x + b_0^k) (w_0^k)_i \right| \right. \\ &\quad + \left| c_t^{k^2} \sigma'(w_t^k \cdot x + b_t^k) \sigma'(w_t^k \cdot y + b_t^k) x_i - c_0^{k^2} \sigma'(w_0^k \cdot x + b_0^k) \sigma'(w_0^k \cdot y + b_0^k) x_i \right| \\ &\quad + \left| c_t^{k^2} \sigma'(w_t^k \cdot x + b_t^k) \sigma''(w_t^k \cdot y + b_t^k) x \cdot y (w_t^k)_i \right. \\ &\quad \left. - c_0^{k^2} \sigma'(w_0^k \cdot x + b_0^k) \sigma''(w_0^k \cdot y + b_0^k) x \cdot y (w_0^k)_i \right| \Big].\end{aligned}\tag{2.161}$$

Applying the mean value theorem with respect to w_t^i to each term on the RHS of (2.161), using the fact that $\sigma \in C_b^4(\mathbb{R})$ and that $|c_t^i| < C$ for any $t \in [0, T]$, we have

$$\begin{aligned}&\left| \partial_{y_i} A_t^N(x, y) - \partial_{y_i} A_0^N(x, y) \right| \\ &\leq \frac{C}{N} \sum_{k=1}^N \left[|(w_t^k)_i - (w_0^k)_i| + |(w_0^k)_i| (\|x\|_1 + \|y\|_1) \|w_t^k - w_0^k\|_1 + |c_t^i - c_0^i| |x_i| \right. \\ &\quad + |x_i| (\|x\|_1 + \|y\|_1) \|w_t^k - w_0^k\|_1 + |(w_t^k)_i - (w_0^k)_i| |x \cdot y| + |c_t^i - c_0^i| |(w_0^k)^i| |x \cdot y| \\ &\quad \left. + |(w_0^k)^i| |x \cdot y| (\|x\|_1 + \|y\|_1) \|w_t^k - w_0^k\|_1 + (1 + |x \cdot y|) |b_t^k - b_0^k| \right].\end{aligned}$$

Therefore, by the boundedness of $|c_t^k - c_0^k|$, $|b_t^k - b_0^k|$, $|(w_t^k)_i - (w_0^k)_i|$ and $\|w_t^k - w_0^k\|_1$ we derive

$$\begin{aligned}\left| \partial_{y_i} A_t^N(x, y) - \partial_{y_i} A_0^N(x, y) \right| &\leq CN^{\beta-1} \left[1 + \frac{\sum_{k=1}^N |(w_0^k)_i|}{N} (|x \cdot y| + 1) (1 + \|x\|_1 + \|y\|_1) \right. \\ &\quad \left. + |x_i| (1 + \|x\|_1 + \|y\|_1) + |x \cdot y| \right] \\ &\leq CN^{\delta+\beta-1}.\end{aligned}$$

By definition,

$$\begin{aligned}\partial_{y_i y_j}^2 A_t^N(x, y) = & \frac{1}{N} \sum_{k=1}^N \left[\sigma(w_t^k \cdot x + b_t^k) \sigma''(w_t^k \cdot x + b_t^k) (w_t^k)_i (w_t^k)_j \right. \\ & + c_t^{k^2} \sigma'(w_t^k \cdot x + b_t^k) \sigma''(w_t^k \cdot y + b_t^k) x_i (w_t^k)_j \\ & + c_t^{k^2} \sigma'(w_t^k \cdot x + b_t^k) \sigma''(w_t^k \cdot y + b_t^k) x_j (w_t^k)_i \\ & \left. + c_t^{k^2} \sigma'(w_t^k \cdot x + b_t^k) \sigma'''(w_t^k \cdot y + b_t^k) x \cdot y (w_t^k)_i (w_t^k)_j \right].\end{aligned}$$

The difference between $\partial_{y_i y_j}^2 A_t^N$ and $\partial_{y_i y_j}^2 A_0^N$ can be split into terms

$$\begin{aligned}& |\partial_{y_i y_j}^2 A_t^N(x, y) - \partial_{y_i y_j}^2 A_0^N(x, y)| \\ & \leq \frac{1}{N} \sum_{k=1}^N \left[\left| \sigma(w_t^k \cdot x + b_t^k) \sigma''(w_t^k \cdot x + b_t^k) (w_t^k)_i (w_t^k)_j \right. \right. \\ & \quad \left. - \sigma(w_0^k \cdot x + b_0^k) \sigma''(w_0^k \cdot x + b_0^k) (w_0^k)_i (w_0^k)_j \right| \\ & \quad + \left| c_t^{k^2} \sigma'(w_t^k \cdot x + b_t^k) \sigma''(w_t^k \cdot y + b_t^k) x_i (w_t^k)_j \right. \\ & \quad \left. - c_0^{k^2} \sigma'(w_0^k \cdot x + b_0^k) \sigma''(w_0^k \cdot y + b_0^k) x_i (w_0^k)_j \right| \\ & \quad + \left| c_t^{k^2} \sigma'(w_t^k \cdot x + b_t^k) \sigma''(w_t^k \cdot y + b_t^k) x_j (w_t^k)_i \right. \\ & \quad \left. - c_0^{k^2} \sigma'(w_0^k \cdot x + b_0^k) \sigma''(w_0^k \cdot y + b_0^k) x_j (w_0^k)_i \right| \\ & \quad + \left| c_t^{k^2} \sigma'(w_t^k \cdot x + b_t^k) \sigma'''(w_t^k \cdot y + b_t^k) x \cdot y (w_t^k)_i (w_t^k)_j \right. \\ & \quad \left. - c_0^{k^2} \sigma'(w_0^k \cdot x + b_0^k) \sigma'''(w_0^k \cdot y + b_0^k) x \cdot y (w_0^k)_i (w_0^k)_j \right| \Big].\end{aligned}\tag{2.162}$$

We apply the mean value theorem to each term above, leading to

$$\begin{aligned}& |\partial_{y_i y_j}^2 A_t^N(x, y) - \partial_{y_i y_j}^2 A_0^N(x, y)| \\ & \leq C N^{\beta-1} \left[\frac{\sum_{k=1}^N |(w_0^k)_i| |(w_0^k)_j|}{N} (|x \cdot y| + 1) (\|x\|_1 + \|y\|_1 + 1) \right. \\ & \quad + \frac{\sum_{k=1}^N |(w_0^k)_i|}{N} [(1 + |x_j|)(1 + \|x\|_1 + \|y\|_1) + |x \cdot y|] \\ & \quad \left. + \frac{\sum_{k=1}^N |(w_0^k)_j|}{N} [(1 + |x_i|)(1 + \|x\|_1 + \|y\|_1) + |x \cdot y|] + 1 + |x_i| + |x_j| + |y| \right] \\ & \leq C N^{\delta+\beta-1}.\end{aligned}\tag{2.163}$$

□

2.6.4 Proof of Lemma 2.3.2

Proof. Taking squares and then expectations on both sides of inequalities (2.48), (2.49), and (2.50) gives

$$\begin{aligned}\mathbb{E}_{c_0, w_0, b_0} [|A_t^N(x, y) - A_0^N(x, y)|^2] &\leq CN^{2(\delta+\beta-1)}, \\ \mathbb{E}_{c_0, w_0, b_0} [|\partial_{y_k} A_t^N(x, y) - \partial_{y_k} A_0^N(x, y)|^2] &\leq CN^{2(\delta+\beta-1)}, \\ \mathbb{E}_{c_0, w_0, b_0} [|\partial_{y_k y_l}^2 A_t^N(x, y) - \partial_{y_k y_l}^2 A_0^N(x, y)|^2] &\leq CN^{2(\delta+\beta-1)}.\end{aligned}\tag{2.164}$$

Summing these inequalities for all $k, l \in \{1, 2, \dots, n\}$ and applying the Cauchy–Schwarz inequality gives the result. $\square$

2.6.5 Proof of Lemma 2.3.4

Proof. First note that

$$\begin{aligned}\mathbb{E}_{c_0, w_0, b_0} [A_0^N(x, y)] &= \mathbb{E}_{c_0, w_0, b_0} \left[\frac{1}{N} \sum_{k=1}^N \sigma(w_0^k \cdot x + b_0^k) \sigma(w_0^k \cdot y + b_0^k) \right. \\ &\quad \left. + c_0^2 x \cdot y \sigma'(w_0^k \cdot x + b_0^k) \sigma'(w_0^k \cdot y + b_0^k) \right] = A(x, y).\end{aligned}\tag{2.165}$$

We can also calculate

$$\begin{aligned}\mathbb{E}_{c_0, w_0, b_0} [|A_0^N(x, y) - A(x, y)|^2] &= \text{Var}[A_0^N(x, y)] = \frac{1}{N} \text{Var}[U(x; c, w, b) \cdot U(y; c, w, b)] \\ &\leq \frac{1}{N} \mathbb{E}_{c, w, b} |U(x; c, w, b) \cdot U(y; c, w, b)|^2 \leq \frac{C}{N}.\end{aligned}\tag{2.166}$$

For the partial derivatives, for any fixed $x, y \in \Omega$,

$$\begin{aligned}&|\partial_{y_i} (\sigma(w \cdot x + b) \sigma(w \cdot y + b) + c^2 \sigma'(w \cdot x + b) \sigma'(w \cdot y + b) x \cdot y)| \\ &= |\sigma(w \cdot x + b) \sigma'(w \cdot y + b) w_i \\ &\quad + c^2 \sigma'(w \cdot x + b) \sigma'(w \cdot y + b) x_i + c^2 x \cdot y \sigma'(w \cdot x + b) \sigma''(w \cdot y + b) w_i| \end{aligned}\tag{2.167}$$

and

$$\begin{aligned}&|\partial_{y_i y_j}^2 (\sigma(w \cdot x + b) \sigma(w \cdot y + b) + c^2 \sigma'(w \cdot x + b) \sigma'(w \cdot y + b) x \cdot y)| \\ &= |\sigma(w \cdot x + b) \sigma''(w \cdot y + b) w_i w_j + c^2 \sigma'(w \cdot x + b) \sigma''(w \cdot y + b) (x_i w_j + x_j w_i) \\ &\quad + c^2 x \cdot y \sigma'(w \cdot x + b) \sigma'''(w \cdot y + b) w_i w_j|.\end{aligned}\tag{2.168}$$

These are locally bounded around $(x, y) \in \Omega^2$, so by Leibniz's integral rule, we swap the expectation and the differential operator

$$\begin{aligned}\partial_{y_i} A(x, y) &= \partial_{y_i} \mathbb{E}_{c,w,b} [\sigma(w \cdot x + b) \sigma(w \cdot y + b) + c^2 \sigma'(w \cdot x + b) \sigma'(w \cdot y + b) x \cdot y] \\ &= \mathbb{E}_{c,w,b} [\partial_{y_i} (\sigma(w \cdot x + b) \sigma(w \cdot y + b) + c^2 \sigma'(w \cdot x + b) \sigma'(w \cdot y + b) x \cdot y)] \\ &= \mathbb{E}_{c_0, w_0, b_0} [\partial_{y_i} A_0^N(x, y)],\end{aligned}\tag{2.169}$$

and

$$\begin{aligned}\partial_{y_i y_j}^2 A(x, y) &= \partial_{y_i y_j}^2 \mathbb{E}_{c,w,b} [\sigma(w \cdot x + b) \sigma(w \cdot y + b) + c^2 \sigma'(w \cdot x + b) \sigma'(w \cdot y + b) x \cdot y] \\ &= \mathbb{E}_{c,w,b} [\partial_{y_i y_j}^2 (\sigma(w \cdot x + b) \sigma(w \cdot y + b) + c^2 \sigma'(w \cdot x + b) \sigma'(w \cdot y + b) x \cdot y)] \\ &= \mathbb{E}_{c_0, w_0, b_0} [\partial_{y_i y_j}^2 A_0^N(x, y)].\end{aligned}\tag{2.170}$$

Therefore, we can bound the first-derivative variance $\text{Var}[\partial_{y_i} A_0^N(x, y)]$:

$$\begin{aligned}\mathbb{E}_{c_0, w_0, b_0} |\partial_{y_i} A_0^N(x, y) - \partial_{y_i} A(x, y)|^2 &= \frac{1}{N} \text{Var}[\partial_{y_i} (U(x; c, w, b) \cdot U(y; c, w, b))] \\ &\leq \frac{1}{N} \mathbb{E}_{c,w,b} |\partial_{y_i} (U(x; c, w, b) \cdot U(y; c, w, b))|^2 \\ &\leq \frac{1}{N} \mathbb{E}_{c,w,b} [(C|w_i| + C|x_i| + C|x \cdot y||w_i|)^2] \\ &\leq \frac{C}{N}.\end{aligned}\tag{2.171}$$

and similarly the second derivative variance $\text{Var}[\partial_{y_i y_j}^2 A_0^N(x, y)]$:

$$\begin{aligned}\mathbb{E}_{c_0, w_0, b_0} |\partial_{y_i y_j}^2 A_0^N(x, y) - \partial_{y_i y_j}^2 A(x, y)|^2 &= \frac{1}{N} \text{Var}[\partial_{y_i y_j}^2 (U(x; c, w, b) \cdot U(y; c, w, b))] \\ &\leq \frac{1}{N} \mathbb{E}_{c,w,b} |\partial_{y_i y_j}^2 (U(x; c, w, b) \cdot U(y; c, w, b))|^2 \\ &\leq \frac{1}{N} \mathbb{E}_{c,w,b} [(C|w_i w_j| + C|x_i w_j| + C|x_j w_i| + C|x \cdot y||w_i w_j|)^2] \\ &\leq \frac{C}{N}.\end{aligned}\tag{2.172}$$

□

2.6.6 Existence of ODE solutions

Lemma 2.6.1. *Under the assumptions of Theorem 2.4.2, for any $q \in \mathcal{H}^2$, and any initial value $v_0 \in L_\eta^2$, the ODE*

$$\frac{dv_t}{dt} = \mathcal{L}(\mathcal{B}v_t + q)\tag{2.173}$$

admits a unique solution v taking values in L_η^2 .

Proof. Consider the process $w_t = \eta v_t$. This has initial value $w_0 = \eta v_0 \in L^2$, and satisfies the dynamics

$$\frac{dw_t}{dt} = \eta \mathcal{L}(\mathcal{B}(w_t/\eta) + q) =: G(w_t). \quad (2.174)$$

From Lemma 2.2.6, the map $w \mapsto \mathcal{B}(w/\eta)$ is uniformly Lipschitz as a map $L^2 \rightarrow \mathcal{H}^2$. By Assumption 2.1.6, $\mathcal{L}$ is uniformly Lipschitz as a map $\mathcal{H}^2 \rightarrow L^2$. By Assumption 2.1.2, η is bounded on Ω . Therefore, $G : L^2 \rightarrow L^2$ is uniformly Lipschitz.

From the Picard–Lindelöf theorem (see, for example, [71, Theorem 2.2.1]), we know that (2.174) admits a unique solution with $w_t \in L^2$ for all $t \geq 0$. Using this, together with the fact $\eta > 0$ on Ω , we see that there is a unique $v_t = w_t/\eta$ satisfying the original equation, with values in L^2_η . $\square$

2.6.7 Monotonicity for linear elliptic PDE

In this subsection, we give a brief proof of monotonicity of a class of elliptic linear differential operators, with particular choices of sampling measure μ . Our results in this appendix are not exhaustive, but demonstrate some of the flexibility of the equations we have considered.

Lemma 2.6.2. *Consider an autonomous Markov process X satisfying the SDE*

$$dX_t = \nu(X_t)dt + \sigma(X_t)dW_t \quad (2.175)$$

where $\nu : \bar{\Omega} \rightarrow \mathbb{R}^n$ and $\sigma : \bar{\Omega} \rightarrow \mathbb{R}^{n \times n}$ and W is an $\mathbb{R}^n$ -valued standard Brownian motion. Suppose further that ν and σ both vanish outside Ω . For an absolutely continuous measure μ on Ω , with density f_μ , let ν , σ and f_μ be sufficiently smooth that

$$\gamma^* = \sup_{x \in \Omega} \left\{ \frac{-\sum_i \frac{\partial}{\partial x_i} [\nu_i(x) f_\mu(x)] + \sum_{i,j} \frac{\partial^2}{\partial x_i \partial x_j} [\frac{1}{2} (\sigma \sigma^\top)_{ij}(x) f_\mu(x)]}{f_\mu(x)} \right\} < \infty. \quad (2.176)$$

Then the generator of X satisfies, for all $u \in \mathcal{H}^2$,

$$\left\langle u, \sum_i \nu_i \frac{\partial u}{\partial x_i} + \frac{1}{2} \sum_{i,j} (\sigma \sigma^\top)_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} \right\rangle \leq \frac{\gamma^*}{2} \|u\|, \quad (2.177)$$

where the inner product and norm are both taken in $L^2(\Omega, \mu)$.

Proof. Consider a copy of X initialized with $X_0 \sim \mu$. The Fokker–Plank equation states that the density of X_0 satisfies $f(x, 0) = f_\mu$ and

$$\frac{\partial}{\partial t} f(x, t) = - \sum_i \frac{\partial}{\partial x_i} [\nu_i(x) f(x, t)] + \sum_{i,j} \frac{\partial^2}{\partial x_i \partial x_j} \left[\frac{1}{2} (\sigma \sigma^\top)_{ij}(x) f(x, t) \right] \quad (2.178)$$

and hence, using the definition of γ^* ,

$$\frac{\partial}{\partial t} (e^{-\gamma^* t} f(x, t)) \leq 0. \quad (2.179)$$

As $f(x, t)$ is positive, we can define an operator

$$T_t u(x) := \mathbb{E} \left[u(X_t^x) e^{-\frac{\gamma^*}{2} t} \middle| X_0 = x \right] \quad (2.180)$$

which satisfies, by Jensen’s inequality,

$$\|T_t u\|_{L^2(\mu)} \leq \int_{\Omega} u^2(t) e^{-\gamma^* t} f(t, x) dx \leq \int_{\Omega} u^2(t) f_\mu(x) dx = \|u\|_{L^2(\mu)}^2 \quad (2.181)$$

so the operator T_t is a contraction. It is easy to verify that T_t is also a strongly continuous semigroup, and its generator is given by $u \mapsto -\frac{\gamma^*}{2} u + \mathcal{A}u$, where $\mathcal{A}$ is the generator of X . By the Lumer–Phillips theorem ([94]), we know that the generator of T_t is dissipative on $L^2(\Omega, \mu)$, that is

$$\left\langle u, -\frac{\gamma^*}{2} u + \mathcal{A}u \right\rangle = \left\langle u, -\frac{\gamma^*}{2} u + \sum_i \nu_i \frac{\partial u}{\partial x_i} + \frac{1}{2} \sum_{i,j} (\sigma \sigma^\top)_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} \right\rangle \leq 0. \quad (2.182)$$

Rearrangement yields the result. $\square$

Remark 2.6.3. This result immediately shows that if μ is a stationary distribution for the Markov process, then $\gamma^* = 0$ and the generator of X is automatically L^2 -dissipative. (This result is known, see, for example, [63, Chapter 20].) This suggests that this will often be a wise choice for μ in linear problems, if it is known explicitly.

By manipulating the choice of measure μ , or equivalently setting μ equal to Lebesgue measure and considering the equivalent PDE

$$\tilde{\mathcal{L}}u := f(x) \mathcal{L}u = 0, \quad (2.183)$$

this result can be leveraged usefully, provided the drift of X does not grow quickly. Similar results for other bounds on b can also be obtained.

Lemma 2.6.4. Consider a process X as in Lemma 2.6.2. Suppose that σ is constant and there exist bounded functions $g_i : \mathbb{R} \rightarrow \mathbb{R}$ for $i \leq n$ such that, for all $x \in \Omega$,

$$\gamma \geq - \sum_i \left(\frac{\partial}{\partial x_i} \nu_i(x) + g(x_i) \nu_i(x) \right) + \sum_{ij} g_i(x_i) g_j(x_j) (\sigma \sigma^\top)_{ij}. \quad (2.184)$$

Then the generator of X satisfies $\langle u, \mathcal{A}u \rangle \leq \frac{\gamma}{2} \|u\|$, where μ is the measure on Ω with Lebesgue density

$$f_\mu = C \exp \left(\sum_i \int_0^{x_i} g_i(\zeta) d\zeta \right),$$

for C a normalizing constant. In particular, this implies that the operator

$$\mathcal{L}u = r - \gamma u + \mathcal{A}u \quad (2.185)$$

is strongly monotone, in the sense of Assumption 2.1.9.

Proof. We calculate

$$\begin{aligned} & - \sum_i \frac{\partial}{\partial x_i} [\nu_i(x) f_\mu(x)] + \sum_{i,j} \frac{\partial^2}{\partial x_i \partial x_j} [(\sigma \sigma^\top)_{ij} f_\mu(x)] \\ &= - \sum_i \left(\frac{\partial}{\partial x_i} \nu_i(x) f_\mu(x) + g_i(x_i) \nu_i(x) f_\mu(x) \right) + \sum_{i,j} g_i(x_i) g_j(x_j) (\sigma \sigma^\top)_{ij} f_\mu(x) \\ &\leq \gamma f_\mu(x). \end{aligned} \quad (2.186)$$

Therefore, in (2.176) we have $\gamma^* \leq \gamma$, and the result follows from (2.177). $\square$

Remark 2.6.5. In one dimension, (2.184) is clearly satisfied whenever we know that $4\sigma^2(\gamma + \nu') \geq -\nu^2$, by taking $g(x) = -\nu(x)/(2\sigma^2)$.

2.6.8 Configuration of the numerical test cases

2.6.8.1 Table of hyperparameters

Dimension	Method	Layer	Units	Optimizer	Numer of MC samples
1	Q-PDE	1	64	ADAM	$l_{MC}=1\text{k}, u_{MC}=2\text{k}$
20	Q-PDE	1	256	ADAM	$l_{MC}=2\text{k}, u_{MC}=10\text{k}$
20	DGM	1	256	ADAM	$l_{MC}=2\text{k}, u_{MC}=10\text{k}$

2.6.8.2 Initialization of neural networks

Parameters of the single-layer net S_0 are randomly sampled: c_0^i are i.i.d sampled from uniform distribution $\mathcal{U}[-1, 1]$; w_0^i and b_0^i are i.i.d sampled from Gaussian distribution $\mathcal{N}(0, I_d)$ and $\mathcal{N}(0, 1)$ where I_d is the identity matrix of dimension d .

2.6.8.3 Learning process

We use $Q_t := S_t \cdot \eta$ as the approximator, where $\eta(x) := 1 - \|x\|^2$. We apply the built-in ADAM optimizer in our test with initial learning rate $l_0 = 0.5$, and the learning rate decays as $l_t = l_0/(1 + t/200)$. In each step, we sample MC samples for gradient estimate. As a larger number of MC sample points reduces the random error, we linearly increase the number M_t of MC points to be sampled at each step as $M_t = \text{round}(l_{MC} + (u_{MC} - l_{MC})t/T)$, where T is the terminal number of training steps.

Chapter 3

Convergence analysis of deep Galerkin method (DGM) & Physics-informed neural networks (PINNs)

3.1 Introduction

Deep learning methods have become widely used for solving high-dimensional PDEs and modeling physics data governed by PDEs. Although low-dimensional PDEs can be efficiently solved with existing numerical techniques, such as finite difference methods, high-dimensional PDEs are computationally intractable due to the curse of dimensionality. An alternative approach that has been widely employed is to approximate the PDE solution with a neural network and then train the neural network with stochastic gradient descent to satisfy the PDE and its boundary conditions – e.g., the deep Galerkin method (DGM) in [138]. A similar method – physics-informed neural networks (PINNs) in [122] – was developed to model physics data by training a neural network to both satisfy the corresponding governing PDE and match a sparse set of experimental observations. Both methods share a common feature of training a neural network with (stochastic) gradient descent to satisfy an objective function with PDE. Numerous other articles in the literature have also explored solving PDEs with neural networks (see [4] for an overview). Solving PDEs with neural network approximators is a natural idea that has been considered in different forms for decades, for instance, [82], [78], [97] and [128]. These papers propose to use neural networks to solve differential equations by estimating neural network solutions on an a priori fixed mesh.

Due to the universal approximation properties of neural networks [48], it is clear that *there exists* a neural network which can approximate the solution to a given PDE (interpreting the PDE solution as a function in the Sobolev space $\mathcal{H}^2$). Recently, important results were proven by [132] and [27], showing that the global minimizer of PINNs with Hölder and ridge regularization will converge to the solution of the PDE as the scale of the network goes to infinity.

In [95], a gradient descent training algorithm for neural net approximations for PDEs is considered, under different assumptions than in our paper and with different mathematical techniques. In particular, for a particular choice of activation functions, assuming non-degeneracy of the Gram matrix associated with the PDE kernel at finitely many (randomly chosen) points, they argue that the gradient descent method will converge as training time $\rightarrow \infty$ to a regularized approximation of the PDE. Under the assumption that the PDE source term has bounded Rademacher complexity, the resulting sequence of approximations will converge. However, it is not possible to interchange these two limiting steps (the number of points and number of training steps) using their analysis. Our approach instead proves that the trained neural network converges to an infinite-dimensional evolution equation and then directly proves convergence of the limit network to the PDE solution as the training time $\rightarrow \infty$, using eigenfunction decomposition analysis. We do not assume that the eigenvalues of the corresponding NTK kernel are uniformly lower bounded (which is not true as the number of hidden units in the network $\rightarrow \infty$) and we do not make assumptions on the Rademacher complexity of the PDE.

A theoretical analysis for solving PDEs in Reproducing Kernel Hilbert Spaces (RKHS) with applications to neural networks is given in [92]. Their analysis requires the assumption that the PDE operator has the same eigenfunctions as the kernel, which is not satisfied by many common activation functions used in deep learning. Our proof does not require this assumption and therefore our convergence theorem applies to a broad class of neural network models.

Convergence analysis for the optimization of neural network approximators to PDEs must address several mathematical challenges. First, the neural network is non-convex in its parameters, which is further exacerbated by applying a PDE operator to the neural network in the objective function. Consequently, as the number of hidden units $\rightarrow \infty$, the standard neural tangent kernel (NTK) does not arise [55]. Instead, the kernel function involves the PDE operator, requiring the development of new mathematical analysis. Finally, as is also true for the standard NTK setting, the kernel lacks a spectral gap, which makes analysis of infinite-dimensional systems

(such as approximators to PDEs) challenging. Our proof leverages a careful analysis of the eigendecomposition of the limit ODE and its kernel function.

For both the DGM and PINN algorithms, we prove that, as the number of hidden units in the single-layer network goes to infinity (i.e., in the “wide network limit”), the trained neural network converges to the solution of an infinite-dimensional linear ODE. The PDE residual of the limiting approximator converges to zero as the training time $\rightarrow \infty$. Under mild assumptions, this convergence also implies that the neural network approximator converges to the solution of the PDE. [151] prove that, under some assumptions, as the number of neurons goes to infinity, the training process of PINNs will converge to a process characterized by a kernel matrix. However, [151] does not prove global convergence of the neural network approximator.

Our paper provides a rigorous mathematical analysis of the DGM and PINN training process for solving PDEs with neural networks. In summary, the key contributions of this paper are:

1. We prove that as the number of hidden units in the neural network $\rightarrow \infty$ (i.e., in the “wide network limit”), the training process of the neural approximator trained to minimize the PDE residual converges to an infinite-dimensional linear ODE characterized by a kernel function.
2. The kernel function is different than the standard NTK kernel and involves the PDE operator.
3. We prove that even though the kernel is only positive semi-definite and there is no spectral gap, the objective function (i.e., the PDE residual of the wide-limit neural network) converges to zero as the training time $t \rightarrow \infty$. This result establishes global convergence. Furthermore, under an additional mild assumption, the wide-limit neural network converges to the PDE solution.

The paper is organized as follows. In Section 3.2, we introduce the class of PDEs that will be considered. Section 3.3 describes the neural network training algorithm for solving PDEs and then proves that the neural network approximator converges to the limit ODE as the number of hidden units $\rightarrow \infty$. Section 3.5 studies the properties of the kernel function that characterizes the limit ODE. Then, we prove that the PDE residual converges to zero as the training time $\rightarrow \infty$. Then, it is proven that – with an additional mild assumption on the PDE – the wide-limit neural network also converges to the PDE solution. In Section 3.6, we prove global convergence for the PINN algorithm. Lemmas, corollaries, and theorems are presented in the main part of the paper. All mathematical proofs are in the Appendix 3.7.

3.2 Mathematical Framework

We will study the convergence of neural network algorithms – such as DGM and PINNs – for solving PDEs. In particular, we will analyze the convergence of such algorithms for the following class of second-order linear PDEs with Dirichlet boundary conditions, where $\mathcal{A}$ is a second-order linear PDE operator satisfies the Assumption 3.2.4:

$$\begin{cases} \mathcal{A}v &= h, & \text{in } \Omega \\ v &= f, & \text{on } \partial\Omega, \end{cases} \quad (3.1)$$

where $\Omega \in \mathbb{R}^d$ is a compact set with a smooth boundary. We will study strong Sobolev solutions to the PDE (3.3); that is, we are interested in solutions $u \in \mathcal{H}^2$, where for a finite measure μ on Ω ,

$$\mathcal{H}^p = \left\{ f \in L^2(\Omega, \mu) : \|f\|_{\mathcal{H}^p} := \left(\sum_{|\alpha| \leq p} \|D_\alpha f\|_{L^2} \right)^2 < \infty \right\}, \quad (3.2)$$

where Du is the weak derivative of u (see [30]). We assume μ is equivalent to Lebesgue measure, and the logarithm of its Radon–Nikodym derivative is bounded (which ensures it generates the same $\mathcal{H}^2$ space as Lebesgue measure). For notational convenience, we will write $\mathcal{H}_{(0)}^2 = \mathcal{H}^2 \cap \mathcal{H}_0^1$, representing the $\mathcal{H}^2$ functions with zero value on the boundary, equipped with the $\mathcal{H}^2$ norm.

We make the following (standard) assumptions on our problem:

Assumption 3.2.1 (Smoothness of the boundary Ω). *The boundary $\partial\Omega$ is $C^{3,\alpha}$ for some $\alpha \in (0, 1)$; i.e., three times continuously differentiable with α -Hölder continuous derivatives of order 3.*

Assumption 3.2.2 (Auxiliary function η). *There exists a (known) function $\eta \in C_b^3(\mathbb{R}^n)$, which satisfies $\eta > 0$ in Ω , and $\eta = 0$ on $\partial\Omega$. Furthermore, its first order derivative does not vanish at the boundary (that is, for $x \in \partial\Omega$ and $\mathbf{n}_x$ an outward unit normal vector at x , we have $\nabla\eta(x) \cdot \mathbf{n}_x \neq 0$).*

Assumption 3.2.3 (Interpolation of the boundary condition function). *There exists a (known) function $\bar{f} \in \mathcal{H}^2$ such that $\bar{f}|_{\partial\Omega} = f$. In the rest of this paper, we identify f with its extension $\bar{f}$ defined on $\bar{\Omega}$ for notational simplicity.*

We can reformulate the PDE as

$$\begin{cases} \mathcal{A}u &= g, & \text{in } \Omega \\ u &= 0, & \text{on } \partial\Omega, \end{cases} \quad (3.3)$$

where $u := v - f$ and $g := h - \mathcal{A}f$. Finally, we assume that the PDE operator satisfies a certain type of Lipschitz condition:

Assumption 3.2.4 (Lipschitz condition). *There exists a constant $k > 0$ such that for any $f_1, f_2 \in C^2 \cap \mathcal{H}^2$ and any $x \in \Omega$, the linear operator $\mathcal{A}$ satisfies*

$$|\mathcal{A}f_1(x) - \mathcal{A}f_2(x)| \leq k \left[\sum_{0 \leq |\alpha| \leq 2} |D_\alpha f_1(x) - D_\alpha f_2(x)| \right]. \quad (3.4)$$

Remark 3.2.5 (Lipschitz condition in $\mathcal{H}^2$). Given $\mathcal{A}$ is linear, by setting $f_2 = 0$, squaring and taking integral on the domain, we can see that there exists constant $C > 0$ that

$$\|\mathcal{A}f\|_{L^2} \leq C\|f\|_{\mathcal{H}^2} \quad (3.5)$$

holds for smooth function f . Then by taking a sequence of functions g_i in the Schwartz function space approaching f in $\mathcal{H}^2$ norm, we see that $\mathcal{A}$ is Lipschitz in $\mathcal{H}^2$. That is, there exists a constant $C > 0$

$$\|\mathcal{A}f_1 - \mathcal{A}f_2\|_{L^2} \leq C\|f_1 - f_2\|_{\mathcal{H}^2}. \quad (3.6)$$

3.3 Deep Learning Algorithms for Solving PDEs

Deep learning PDE algorithms – such as DGM and PINNs – train a neural network approximator to satisfy the PDE and its boundary conditions using either gradient descent or stochastic gradient descent.

Consider the following single-layer neural network with N hidden units S^N :

$$S^N(x; \theta^N) = \frac{1}{N^\beta} \sum_{i=1}^N c^i \sigma(w^i x + b^i), \quad (3.7)$$

where $\frac{1}{N^\beta}$ is a normalization factor and $\frac{1}{2} < \beta < 1$. We train a neural network Q^N to approximate the solution u to the PDE where

$$Q^N(x; \theta^N) := \eta(x) S^N(x; \theta^N) = \eta(x) \cdot \frac{1}{N^\beta} \sum_{i=1}^N c^i \sigma(w^i x + b^i). \quad (3.8)$$

Here $\eta(x)$ is a fixed function which vanishes on the boundary $x \in \partial\Omega$; therefore Q^N automatically satisfies the boundary conditions of the PDE (3.3). This method, which was first introduced by [100], simplifies the training of the neural network model. The parameters $\theta^N = (c^i, w^i, b^i)_{i=1}^N$ must be trained using gradient descent to satisfy the

PDE in the interior of the domain. Specifically, we will minimize the PDE residual error for the neural network by minimizing the following objective function:

$$J(\theta^N) = \|\mathcal{A}Q^N - g\|_{L^2(\mu)}^2 := \int_{\Omega} [\mathcal{A}Q^N(x; \theta^N) - g(x)]^2 \mu(dx), \quad (3.9)$$

where μ is a sampling measure (satisfying the regularity assumptions stated after (3.2) above). If the residual term $\mathcal{A}Q^N(x; \theta^N) - g(x)$ equals zero for all $x \in \Omega$, then $Q^N = u$ is the solution of the PDE. We will minimize the objective function (3.9) using gradient descent with clipping. The gradient of (3.9) is:

$$\nabla_{\theta} J(\theta^N) = \int_{\Omega} [\mathcal{A}Q^N(x; \theta^N) - g(x)] \nabla_{\theta} \mathcal{A}Q^N(x; \theta^N) \mu(dx). \quad (3.10)$$

Gradient clipping is widely used in deep learning, see for instance [162], [118] and chapters 10 and 11 of [37]. The continuous-time gradient descent training with clipping is given by:

$$\frac{d\theta_t^N}{dt} = -\alpha^N G^N(\theta_t^N), \quad (3.11)$$

where the learning rate is $\alpha^N = N^{2\beta-1}$ and

$$G^N(\theta_t^N) = - \int_{\Omega} \psi^N(\mathcal{A}Q^N(x; \theta_t^N) - g(x)) \Phi^N(\nabla_{\theta} \mathcal{A}Q^N(x; \theta_t^N)) \mu(dx). \quad (3.12)$$

Here Φ^N is a vector function that applies elementwise clipping with the function ϕ^N . For each entry of vector $\nabla_{\theta} \mathcal{A}Q^N$, we clip its value with the scalar function ϕ^N .

In practice, (3.11) can be approximated by discretizing in time and, at each time step, generating Monte Carlo samples from the measure μ to approximate the integral, which is highly computationally efficient even for high-dimensional PDEs. This is also equivalent to the stochastic gradient descent version of (3.11). Although not investigated in this paper, standard methods can be used to prove that stochastic gradient – using the correct learning rate – will converge to the continuous-time gradient flow (3.11); for example, weak convergence analysis such as in [141] and [139] could be used.

Let $Q_t^N(x) = Q^N(x; \theta_t^N)$ be the neural network at training time t with N hidden units. We will analyze the trained neural network Q_t^N as the number of hidden units $N \rightarrow \infty$ and the training time $t \rightarrow \infty$. First, we prove that the trained neural network Q_t^N will converge to the solution of an infinite-dimensional ODE as $N \rightarrow \infty$. That is, in the “wide limit” where the number of hidden units $\rightarrow \infty$, Q_t^N converges to

the solution of an ODE. Then, we prove that the wide-limit neural network converges to the global minimizer of the objective function (with zero PDE residual) as the training time $t \rightarrow \infty$. Under additional mild assumptions, this global minimizer is also a solution to the PDE. These convergence results can also be proven for the PINNs algorithm for solving PDEs; see Section 3.6.

Our convergence results will be proven under the following assumptions on the neural network architecture:

Assumption 3.3.1 (Activation function). *The activation function $\sigma \in C_b^4(\mathbb{R})$ is non-constant.*

Assumption 3.3.2 (Neural network initialization). *The initialization of the parameters θ_0^N , for all $i \in \{1, 2, \dots, N\}$, satisfies:*

- *The parameters c_0^i, w_0^i, b_0^i are i.i.d. random variables.*
- *The random variables c_0^i are bounded, $|c_0^i| < K_0$, and $\mathbb{E}[c_0^i] = 0$.*
- *The distribution of the random variables w_0^i, b_0^i has full support. That is, for any open set $D \subseteq \mathbb{R}^{n+1}$, we have $\mathbb{P}((w_0^i, b_0^i) \in D) > 0$.*
- *The moments $\mathbb{E}[|(w_0^i)_k|^3]$ and $\mathbb{E}[|b_0^i|]$ are bounded where $(w_0^i)_k$ is the k -element of w_0^i .*

Definition 3.3.3 (Smooth clipping function). *A function class $\{h^N\}_{N \in \mathbb{N}^+}$ forms a family of smooth clipping functions with parameter $\gamma > 0$ if for any $N \in \mathbb{N}^+$*

- *$h^N \in \mathbb{C}_b^2(\mathbb{R})$ is increasing on $\mathbb{R}$.*
- *$|h^N|$ is bounded by $2N^\gamma$.*
- *$h^N(x) = x$ for $x \in [-N^\gamma, N^\gamma]$.*
- *$|(h^N)'| \leq 1$ on $\mathbb{R}$.*

Assumption 3.3.4. *Functions $\{\psi^N\}_{N \in \mathbb{N}^+}$ and $\{\phi^N\}_{N \in \mathbb{N}^+}$ are families of smooth clipping functions with parameter δ and $\epsilon - \beta$ where $\epsilon > \delta > 0$, $\beta \in (\frac{1}{2}, 1)$ and $\epsilon + \delta < \frac{1-\beta}{2}$.*

Lemma 3.3.5. *There exists a constant k independent of N such that the change of each component of c^i, w^i, b^i from its initial condition (e.g. $|c_t^i - c_0^i|$) is bounded by $ktN^{2\beta-1+\delta+\epsilon-\beta} = ktN^{\beta+\delta+\epsilon-1}$.*

Proof. Notice that in (3.11), three terms α^N , $\psi^N(\mathcal{A}Q^N - g)$ and $\Phi^N(\nabla_\theta \mathcal{A}Q^N)$ are bounded by $N^{2\beta-1}$, N^δ and $N^{\epsilon-\beta}$ respectively. $\square$

3.3.1 Convergence of the trained neural network as the number of hidden units $N \rightarrow \infty$

In this section, we analyze the evolution of the neural network Q_t^N as it is trained to minimize the PDE residual in the objective function $J(\theta^N)$. We can prove that, as the number of hidden units $N \rightarrow \infty$, the neural network $Q_t^N(y)$ will converge to the solution $Q_t(y)$ of an infinite-dimensional linear ODE. By the chain rule, the dynamics of Q^N satisfy

$$\begin{aligned} \frac{dQ_t^N}{dt}(y) &= \nabla_{\theta} Q_t^N(y) \cdot \frac{d\theta^N}{dt} \\ &= - \int_{\Omega} \psi^N(\mathcal{A}Q_t^N(x) - g(x)) [\alpha^N \Phi^N(\nabla_{\theta} \mathcal{A}Q_t^N(x))] \cdot \nabla_{\theta} Q_t^N(y) \mu(dx). \end{aligned} \quad (3.13)$$

We are interested in studying the limit of the dynamics of the neural network Q_t^N as $N \rightarrow \infty$. Specifically, we will prove that Q_t^N will converge to Q_t as $N \rightarrow \infty$ where Q_t satisfies

$$\frac{dQ_t}{dt}(y) = - \int_{\Omega} [\mathcal{A}Q_t(x) - g(x)] U(x, y) \mu(dx), \quad Q_0 = 0, \quad (3.14)$$

where the function U is

$$U(x, y) := \mathbb{E}_{c, w, b} \left[\nabla_{c, w, b} \mathcal{A}[\eta(x) c \sigma(x; w, b)] \cdot \nabla_{c, w, b} [\eta(y) c \sigma(y; w, b)] \right], \quad (3.15)$$

where the random variable (c, w, b) has the same distribution as (c_0^i, w_0^i, b_0^i) .

The ODE (3.14) is an infinite-dimensional linear ODE governing the evolution of the wide-limit neural network (i.e., a neural network with an “infinite” number of hidden units) during training. The right-hand side (RHS) of the ODE involves integral over the PDE residual $[\mathcal{A}Q_t(x) - g(x)]$ weighted by a kernel $U(x, y)$. It is important to notice that the kernel $U(x, y)$ is not the standard NTK kernel: it involves the PDE operator $\mathcal{A}$, which significantly complicates its analysis.

One of the consequences of the presence of the PDE operator $\mathcal{A}$ in the kernel $U(x, y)$ is that $U(x, y)$ is asymmetric. This is a key difference from the standard NTK kernel, which is symmetric.

Define the integral operator $\mathcal{U} : L^2 \rightarrow \mathcal{H}_{(0)}^2 \subset L^2$ by

$$\mathcal{U}f := \int_{\Omega} f(x) U(x, y) \mu(dx). \quad (3.16)$$

Note that it is straightforward to check that $U(x, y) = 0$ for $y \in \partial\Omega$ and that U is C_b^2 with respect to y , which ensures that $\mathcal{U}f$ takes values in $\mathcal{H}_{(0)}^2$. Using this notation,

the limit ODE (3.14) can be rewritten as a linear equation in $\mathcal{H}_{(0)}^2$:

$$\frac{dQ_t}{dt} = -\mathcal{U}[\mathcal{A}Q_t - g], \quad Q_0 = 0. \quad (3.17)$$

Lemma 3.3.6. *The ODE (3.17) admits a unique solution in $\mathcal{H}_{(0)}^2$.*

Proof. By Assumptions 2.1.6, 2.1.14 and 2.1.16, the operator $\mathcal{U}$ is Lipschitz in $\mathcal{H}^2$ norm. Therefore it admits a unique solution in $\mathcal{H}^2$. It is easy to verify that, as its initial condition is in $\mathcal{H}_{(0)}^2$, and $\mathcal{U}$ has codomain $\mathcal{H}_{(0)}^2$, the solution lives in the Hilbert subspace $\mathcal{H}_{(0)}^2$. $\square$

Now we present one of this paper's main results. The trajectory of Q_t^N during training, in the limit $N \rightarrow \infty$, can be characterized by the wide-limit network Q_t which satisfies the infinite-dimensional ODE (3.17).

Theorem 3.3.7. *For any $t \geq 0$, the neural network Q_t^N converges to Q_t in $\mathcal{H}^2$:*

$$\lim_{N \rightarrow \infty} \mathbb{E}[\|Q_t^N - Q_t\|_{\mathcal{H}^2}] = 0. \quad (3.18)$$

3.4 Analysis of the kernel function

In order to prove global convergence as $t \rightarrow \infty$ for the limit ODE (3.17), we first must prove some key properties for the integral operator. Specifically, we will study the properties of the operator $\mathcal{S} = \mathcal{A}\mathcal{U}$.

Definition 3.4.1 (Operator $\mathcal{S}$). *The operator $\mathcal{S} : L^2 \rightarrow L^2$ is defined by*

$$\mathcal{S}f := \mathcal{A}\mathcal{U}f = \mathcal{A}\left(\int_{\Omega} f(x)U(x, \cdot)\mu(dx)\right) \quad (3.19)$$

Definition 3.4.2 (Kernel S). *The kernel S is defined by*

$$S(x, y) := \mathbb{E}_{c, w, b} \left[\nabla_{c, w, b} \mathcal{A}[\eta(x)c\sigma(x; w, b)] \cdot \nabla_{c, w, b} \mathcal{A}[\eta(y)c\sigma(y; w, b)] \right]. \quad (3.20)$$

By symmetry of second derivatives (Clairaut–Schwarz–Young theorem) we know that $S(x, \cdot) := \mathcal{A}U(x, \cdot)$ and hence $\mathcal{S}f = \int_{\Omega} f(x)S(x, \cdot)\mu(dx)$.

While $U(x, y)$ is asymmetric, $S(x, y)$ is symmetric. The symmetric kernel S depends upon the interaction of PDE operator $\mathcal{A}$ applied to the activation function σ . It will next be proven (Lemma 3.4.6) that the operator $\mathcal{S}$ is discriminatory in the image set of $\mathcal{A}$. This is an important property that will later be leveraged in the global convergence proof.

Lemma 3.4.3. *The kernel S is uniformly bounded. That is, there exists a constant $k > 0$ such that for any $(x, y) \in \Omega^2$ we know $|S(x, y)| \leq k$.*

Proof. By definition

$$\begin{aligned} S(x, y) = \mathbb{E} & \left[\mathcal{A}[\eta(x)\sigma_{w,b}(x)]\mathcal{A}[\eta(y)\sigma_{w,b}(y)] + \sum_{i=1}^d \mathcal{A}[c\eta(x)x_i\sigma'_{w,b}(x)]\mathcal{A}[c\eta(y)y_i\sigma'_{w,b}(y)] \right. \\ & \left. + \mathcal{A}[c\eta(x)\sigma'_{w,b}(x)]\mathcal{A}[c\eta(y)\sigma'_{w,b}(y)] \right]. \end{aligned} \quad (3.21)$$

By the fact that $\mathcal{A}$ is Lipschitz, and that η , $\sigma_{w,b}$ and their partial derivatives are all bounded, there exists constant $k_1 > 0$, $k_2 > 0$ such that

$$|\mathcal{A}[\eta(x)\sigma_{w,b}(x)]| \leq k \sum_{\alpha} |D_{\alpha}\eta(x)\sigma_{w,b}(x)| \leq k_1 \sum_{1 \leq i, j \leq d} |w_i| + |w_i w_j| + 1. \quad (3.22)$$

Similarly, since c is bounded and Ω is bounded

$$\begin{aligned} |\mathcal{A}[c\eta(x)\sigma'_{w,b}(x)]| & \leq k_1 \sum_{1 \leq i, j \leq d} |w_i| + |w_i w_j| + 1 \\ |\mathcal{A}[c\eta(x)x_i\sigma'_{w,b}(x)]| & \leq k_1 \sum_{1 \leq i, j \leq d} |w_i| + |w_i w_j| + 1. \end{aligned} \quad (3.23)$$

Therefore,

$$|S(x, y)| \leq \mathbb{E}_{c,w,b} [(d+2)k_1^2 \left(\sum_{1 \leq i, j \leq d} |w_i| + |w_i w_j| + 1 \right)^2] \leq k_2. \quad (3.24)$$

□

Lemma 3.4.4. *The integral operator $\mathcal{S}$ is Hilbert–Schmidt. In addition, $\mathcal{S}$ is self-adjoint and positive semi-definite.*

Proof. Since $S(x, y)$ is uniformly bounded, we have $\iint_{\Omega^2} |S(x, y)|^2 \mu(dx) \mu(dy) < \infty$. Therefore $\mathcal{S}$ is Hilbert–Schmidt. And since $S(x, y)$ is symmetric, the operator $\mathcal{S}$ is self-adjoint with respect to the L^2 inner product. Now it remains to prove that $\mathcal{S}$ is positive semi-definite. For $f \in L^2$

$$\begin{aligned} \langle f, \mathcal{S}f \rangle & = \iint_{\Omega^2} f(x)S(x, y)f(y)d\mu(x)d\mu(y) \\ & = \iint f(x)\mathbb{E}_{c,w,b} [\nabla_{c,w,b}\mathcal{A}[\eta(x)c\sigma(x; w, b)] \cdot \nabla_{c,w,b}\mathcal{A}[\eta(y)c\sigma(y; w, b)]] f(y)\mu(dx)\mu(dy) \end{aligned} \quad (3.25)$$

By Tonelli's theorem, swapping the order of expectation and the integral gives

$$\begin{aligned}
& \langle f, \mathcal{S}f \rangle \\
&= \mathbb{E} \left[\iint_{\Omega^2} f(x) \nabla_{c,w,b} \mathcal{A}[\eta(x)c\sigma(x;w,b)] \cdot \nabla_{c,w,b} \mathcal{A}[\eta(y)c\sigma(y;w,b)] f(y) \mu(dx) \mu(dy) \right] \\
&\geq \mathbb{E}_{c,w,b} \left[\iint_{\Omega^2} f(x) \nabla_c \mathcal{A}[\eta(x)c\sigma(x;w,b)] \cdot \nabla_c \mathcal{A}[\eta(y)c\sigma(y;w,b)] f(y) \mu(dx) \mu(dy) \right] \\
&= \mathbb{E}_{c,w,b} \left[\iint_{\Omega^2} f(x) \mathcal{A}[\eta(x)\sigma(x;w,b)] \mathcal{A}[\eta(y)\sigma(y;w,b)] f(y) \mu(dx) \mu(dy) \right] \\
&= \mathbb{E}_{c,w,b} \left[\left(\int_{\Omega} f(x) \mathcal{A}[\eta(x)\sigma(x;w,b)] \mu(dx) \right)^2 \right] \geq 0.
\end{aligned} \tag{3.26}$$

□

Lemma 3.4.5 (Spectral decomposition). *The integral operator $\mathcal{S}$ is compact, in particular, there exists an orthogonal basis of L^2 , $\{\varepsilon_i\}_{i \in \mathbb{N}^+} \cup \{\nu_i\}_{i \in \mathbb{N}^+}$ such that*

$$\mathcal{S}\varepsilon_i = \lambda_i \varepsilon_i, \quad \mathcal{S}\nu_i = 0, \tag{3.27}$$

where $\lambda_1 \geq \lambda_2 \geq \dots > 0$.

Proof. As $\mathcal{S}$ is a Hilbert–Schmidt integral operator, $\mathcal{S}$ is compact. Since $\mathcal{S}$ is self-adjoint, the spectral theorem applies. From Lemma 3.4.4, we see that $\mathcal{S}$ is positive semi-definite. Thus, its eigenvalues are real, non-negative, and concentrate only at zero. We use $\{\nu_i\}$ to represent the eigenfunctions of the zero eigenvalue, and $\{\varepsilon_i\}$ for eigenfunctions of positive eigenvalues. □

Lemma 3.4.6 (Projection on $\ker(\mathcal{S})$). *For $h \in L^2$, if $\mathcal{S}h = 0$ then $\langle h, \mathcal{A}f \rangle = 0$ for any $f \in \mathcal{H}_{(0)}^2$.*

Proof. We recall without proof the following technical result.

Lemma 3.4.7 (Lemma 5 in [59]). *Given Assumptions 3.2.1 and 3.2.2:*

1. *The set of functions $C^3(\overline{\Omega}) \cap C_0(\overline{\Omega})$ is dense in $\mathcal{H}_{(0)}^2 = \mathcal{H}^2 \cap \mathcal{H}_0^1$ (under the $\mathcal{H}^2$ topology).*
2. *For any function $u \in C^3(\overline{\Omega}) \cap C_0(\overline{\Omega})$, the function $\tilde{u} = u/\eta$ is in $C_b^2(\Omega) \subset \mathcal{H}^2$.*

We now proceed to the proof of Lemma 3.4.6. By (2.49), we have

$$\langle h, \mathcal{S}h \rangle = \mathbb{E}_{c,w,b} \left[\left(\int_{\Omega} h(x) \mathcal{A}[\eta(x)\sigma(x;w,b)] \mu(dx) \right)^2 \right] = 0. \tag{3.28}$$

Therefore

$$\int_{\Omega} h(x) \mathcal{A}[\eta(x) \sigma(x; w, b)] \mu(dx) = 0 \quad (3.29)$$

for any (c, w, b) by the continuity of the objective with respect to parameters c, w, b .

Since μ is a finite measure, from Theorem 4 in [48] we have that the linear span of $\{\sigma(w \cdot x + b)\}_{w, b \in \mathbb{R}}$ is dense in $\mathcal{H}^2$. For a general $f \in C^3(\bar{\Omega}) \cap C_0(\bar{\Omega})$, by Lemma 2.1.3(ii), we can approximate the function f/η within the linear span of $\{\sigma(w \cdot x + b)\}_{w, b \in \mathbb{R}}$. Multiplying by η , by Lemma 2.1.3(i) it follows that the function class $\{\eta(x) \sigma(x; w, b)\}$ is dense in the function space $\mathcal{H}_{(0)}^2(\Omega)$, which consists of $\mathcal{H}^2$ function with boundary value zero. For any $f \in \mathcal{H}_{(0)}^2$, there exists a function sequence $\{F_N : \eta \sum_{i=1}^N c_i \sigma_{w, b}\}_{N \geq 1}$ such that

$$\lim_{N \rightarrow \infty} \|F_N - f\|_{\mathcal{H}^2} = 0. \quad (3.30)$$

Therefore,

$$\langle h, \mathcal{A}f \rangle = \int_{\Omega} h(x) \mathcal{A}f \mu(dx) = \lim_{N \rightarrow \infty} \langle h, \mathcal{A}F_N \rangle = 0. \quad (3.31)$$

□

Remark 3.4.8. If u is a solution to the PDE (3.3), Lemma 3.4.6 implies that for any $t \geq 0$, the residual of our approximator $\mathcal{A}Q_t - g = \mathcal{A}[Q_t - u]$ has zero projection on the eigenfunction family $\{\nu_i\}$ since $\mathcal{S}\nu_i = 0$.

Corollary 3.4.9. *Assuming the PDE (3.3) admits a solution $u \in \mathcal{H}_{(0)}^2$, any stationary point $Q^* \in \mathcal{H}_{(0)}^2$ of the limit ODE (3.14) is a solution of the PDE (3.3). That is, any stationary point of the limit training algorithm (3.14) is a global minimizer and a solution of (3.3).*

Proof. Suppose that Q_* is a stationary point. Then we have

$$\mathcal{U}[\mathcal{A}Q^* - g] = 0. \quad (3.32)$$

Then

$$\mathcal{S}\mathcal{A}[Q^* - u] = \mathcal{S}[\mathcal{A}Q^* - g] = \mathcal{A}\mathcal{U}[\mathcal{A}Q^* - g] = 0. \quad (3.33)$$

By Lemma 3.4.6, the inner product term $\langle \mathcal{A}[Q^* - u], \mathcal{A}f \rangle = 0$ for any $f \in \mathcal{H}_{(0)}^2$. Therefore, by taking $f = Q^* - u$, we have $\|\mathcal{A}Q^* - g\|_2^2 = \langle \mathcal{A}[Q^* - u], \mathcal{A}[Q^* - u] \rangle = 0$. At the same time, since $Q^* \in \mathcal{H}_{(0)}^2$ satisfies the zero boundary condition, we conclude that $Q_* = u$ is the solution of the PDE. □

Remark 3.4.10. The existence of solutions is needed in this result, for example, consider the trivial case where $\mathcal{A}u \equiv 0$ and $g \neq 0$. Then $\frac{d}{dt}Q_t = 0$, but $Q_t = Q_0$ does not solve the PDE.

3.5 Global convergence of the limit ODE as $t \rightarrow \infty$

By analyzing the PDE residual term's projection on the eigenfunctions of $\mathcal{S}$, we can prove that the PDE residual (which is the objective function that is being minimized) converges to zero as the training time $t \rightarrow \infty$.

Applying $\mathcal{A}$ with respect to y on both sides of the limit ODE (3.17) yields

$$\frac{d\mathcal{A}Q_t}{dt} = \frac{d[\mathcal{A}Q_t - g]}{dt} = -\mathcal{S}[\mathcal{A}Q_t - g]. \quad (3.34)$$

Theorem 3.5.1 (Convergence of PDE residual under DGM). *Assuming the PDE (3.3) admits a solution in $\mathcal{H}_{(0)}^2$, the PDE residual for the wide-limit neural network Q_t converges to zero:*

$$\lim_{t \rightarrow \infty} \|\mathcal{A}Q_t - g\|_{L^2(\mu)} = 0. \quad (3.35)$$

Proof. In (3.34), multiplying $\mathcal{A}Q_t - g$ on both sides and integrating with respect to $\mu(dy)$:

$$\frac{d\|\mathcal{A}[Q_t - u]\|_2^2}{dt} = -2\langle \mathcal{A}[Q_t - u], \mathcal{S}\mathcal{A}[Q_t - u] \rangle \leq 0 \quad (3.36)$$

with strict inequality unless $\|\mathcal{A}Q_t - g\|_2 = 0$ which corresponds to PDE's solution.

Consider $\mathcal{A}Q_t - g = \mathcal{A}[Q_t - u]$ projected on $\{\varepsilon_i\}_{i \in \mathbb{N}^+} \cup \{\nu_i\}_{i \in \mathbb{N}^+}$. By Lemma 3.4.6, we have

$$\mathcal{A}Q_t - g = \sum_i h_t^i \varepsilon_i + \sum_j 0 \nu_j = \sum_i h_t^i \varepsilon_i. \quad (3.37)$$

Therefore $\|\mathcal{A}Q_t - g\|_2^2 = \sum_i h_t^{i2}$. Now consider its projection on each ε_i

$$\begin{aligned} \frac{d}{dt} \langle \mathcal{A}Q_t - g, \varepsilon_i \rangle &= \left\langle \frac{d\mathcal{A}Q_t - g}{dt}, \varepsilon_i \right\rangle = \langle -\mathcal{S}[\mathcal{A}Q_t - g], \varepsilon_i \rangle = \langle \mathcal{A}Q_t - g, -\mathcal{S}\varepsilon_i \rangle \\ &= -\lambda_i \langle \mathcal{A}Q_t - g, \varepsilon_i \rangle. \end{aligned} \quad (3.38)$$

Therefore

$$\frac{d}{dt} \langle \mathcal{A}Q_t - g, \varepsilon_i \rangle = \frac{d}{dt} [h_t^i] = -\lambda_i h_t^i. \quad (3.39)$$

Consequently $h_t^i = h_0^i e^{-\lambda_i t}$ whose absolute value decays exponentially, and $|h_t^i| \leq |h_0^i|$ for any $t \geq 0$ and any i .

Now, by the dominated convergence theorem

$$\lim_{t \rightarrow \infty} \|\mathcal{A}Q_t - g\|_2^2 = \lim_{t \rightarrow \infty} \sum_i |h_t^i|^2 = \sum_i \lim_{t \rightarrow \infty} |h_t^i|^2 = 0. \quad (3.40)$$

□

This result establishes global convergence of the (wide-limit) training algorithm since the objective function is the PDE residual. However, the convergence of the PDE residual to zero does not necessarily guarantee that $\|Q_t - u\|_{L^2(\mu)}$ as $t \rightarrow \infty$ where u is the solution of the PDE (3.3). We show that, under a mild additional assumption on the PDE operator $\mathcal{A}$, we can guarantee that Q_t converges to the solution of the PDE u .

Assumption 3.5.2 (Bounded inverse of $\mathcal{A}$). *The inverse of $\mathcal{A}$ is a bounded operator on $L^2 \rightarrow L^2$. That is, there exists a constant $k > 0$, such that for any $g \in L^2$, the PDE (3.3) has a unique solution $u \in \mathcal{H}_{(0)}^2$ satisfying*

$$\|u\|_2 \leq k\|g\|_{L^2(\mu)}.$$

Remark 3.5.3. Second-order uniformly elliptic PDEs naturally have this property. For reference, see Theorem 6, Chapter 6 of [30].

Theorem 3.5.4 (Convergence of Q_t under DGM). *If $\mathcal{A}$ has a bounded inverse, Q_t converges to the solution u of the PDE:*

$$\lim_{t \rightarrow \infty} \|Q_t - u\|_{L^2(\mu)} = 0.$$

Proof. Writing $\mathcal{A}^{-1}$ for the inverse operator of $\mathcal{A}$, we have

$$\|Q_t - u\|_2 = \|\mathcal{A}^{-1}[\mathcal{A}[Q_t - u]]\|_2 \leq k\|\mathcal{A}[Q_t - u]\|_2 = k\|\mathcal{A}Q_t - g\|_2 \rightarrow 0. \quad (3.41)$$

□

3.6 Global convergence of PINNs algorithm

In many real-world engineering applications, the PDE solution can be observed at a sparse set of points in the interior of the domain Ω . We consider a PINN algorithm which trains a neural network model to predict the solution $u(x)$ for all $x \in \Omega$ by minimizing (via gradient descent) both the PDE residual sampled by μ on a random set of points in each epoch as well as the distance between the neural network and the observations at the finite set of sparse points. It is assumed that the boundary condition is known.

Let us denote these extra observation points in Ω as $\mathbf{x} := \{x_i\}_{i=1,2,\dots,M}$ where the observations are $u(x_i) = u_i$ for $1 \leq i \leq M$. Define the measure $\mu_{\mathbf{x}} = \frac{1}{M} \sum_{i=1}^M \delta_{x_i}$ and the kernel function

$$B(x, y) := \mathbb{E}_{c,w,b} \left[\nabla_{c,w,b}[c\eta(x)\sigma_{w,b}(x)] \cdot \nabla_{c,w,b}[c\eta(y)\sigma_{w,b}(y)] \right], \quad (3.42)$$

and the corresponding ‘integral’ operator $\mathcal{B} : L^2(\mu_{\mathbf{x}}) \equiv \mathbb{R}^M \rightarrow L^2(\mu)$ by

$$\mathcal{B}\mathbf{v}(y) := \frac{1}{M} \sum_{i=1}^M v_i B(x_i, y). \quad (3.43)$$

For any operator $\mathcal{C}$, we define $\bar{\mathcal{C}}$ to be the evaluation of $\mathcal{C}$ on each of our training points,

$$\bar{\mathcal{C}}\mathbf{v} = [\mathcal{C}\mathbf{v}(x_1), \dots, \mathcal{C}\mathbf{v}(x_M)]^\top \in \mathbb{R}^M,$$

in particular

$$\bar{\mathcal{B}}\mathbf{v} := \left[\frac{1}{M} \sum_{i=1}^M v_i B(x_i, x_1), \dots, \frac{1}{M} \sum_{i=1}^M v_i B(x_i, x_M) \right]^\top, \quad (3.44)$$

and for generic $g \in L^2(\mu)$ and $\mathbf{v} \in \mathbb{R}^M$, direct calculation shows that

$$\begin{aligned} \langle \bar{\mathcal{U}}g, \mathbf{v} \rangle_{L^2(\mu_{\mathbf{x}})} &= \frac{1}{M} \sum_i (\mathcal{U}g)(x_i) v_i \\ &= \frac{1}{M} \sum_i \int \mathbb{E} \left[v_i g(x) \nabla_{c,w,b} \mathcal{A}[c\eta(x) \sigma_{w,b}(x)] \cdot \nabla_{c,w,b} [c\eta(x_i) \sigma_{w,b}(x_i)] \right] d\mu \\ &= \langle g, \mathcal{A}\mathcal{B}\mathbf{v} \rangle_{L^2(\mu)}, \end{aligned} \quad (3.45)$$

in other words, $\bar{\mathcal{U}} : L^2(\mu) \rightarrow \mathbb{R}^M$ is the adjoint of $\mathcal{A}\mathcal{B} : \mathbb{R}^M \rightarrow L^2(\mu)$, so we write $\mathcal{A}\mathcal{B} = \bar{\mathcal{U}}^*$.

The PINN objective function is

$$\begin{aligned} J(\theta^N) &= \|\mathcal{A}Q_t^N - g\|_{L^2(\mu)}^2 + \|Q_t^N - u\|_{L^2(\mu_{\mathbf{x}})}^2 \\ &= \int_{\Omega} (\mathcal{A}Q_t^N - g)^2 d\mu + \int_{\Omega} (Q_t^N - u)^2 d\mu_{\mathbf{x}}. \end{aligned} \quad (3.46)$$

Informally, the wide-limit ODE (as $N \rightarrow \infty$) for the neural network trained with continuous-time gradient descent is:

$$\frac{dQ_t}{dt} = -\mathcal{U}[\mathcal{A}Q_t - g] - \mathcal{B}[Q_t - u]. \quad (3.47)$$

Applying the operator $\mathcal{A}$ and recalling $\mathcal{A}\mathcal{B}$ is the adjoint of $\bar{\mathcal{U}} : L^2(\mu) \rightarrow \mathbb{R}^M$, we derive that

$$\frac{d}{dt} \begin{bmatrix} \mathcal{A}Q_t - g \\ \mathbf{Q}_t - \mathbf{u} \end{bmatrix} = - \begin{bmatrix} \mathcal{S} & \bar{\mathcal{U}}^* \\ \bar{\mathcal{U}} & \bar{\mathcal{B}} \end{bmatrix} \begin{bmatrix} \mathcal{A}Q_t - g \\ \mathbf{Q}_t - \mathbf{u} \end{bmatrix}, \quad (3.48)$$

where, on the left-hand side of the equation, the first term $\mathcal{A}Q_t - g \in L^2(\mu)$ and the second term $\mathbf{Q}_t - \mathbf{u} \in \mathbb{R}^M$ is

$$\mathbf{Q}_t - \mathbf{u} = \begin{bmatrix} Q_t(x_1) - u(x_1) \\ Q_t(x_2) - u(x_2) \\ \vdots \\ Q_t(x_M) - u(x_M) \end{bmatrix}. \quad (3.49)$$

In light of this, we define the operator $\mathcal{V} : L^2(\mu) \times \mathbb{R}^M \rightarrow L^2(\mu) \times \mathbb{R}^M$

$$\mathcal{V}f := \begin{bmatrix} \mathcal{S} & \bar{\mathcal{U}}^* \\ \bar{\mathcal{U}} & \bar{\mathcal{B}} \end{bmatrix} f. \quad (3.50)$$

Given the properties of $\mathcal{S}$ (Lemma 3.4.4), it is easy to verify that $\mathcal{V}$ is Hilbert–Schmidt, self-adjoint and positive semi-definite. It will not generally be strictly positive definite. However, using the representation properties of neural networks, we can prove that $(\mathcal{A}Q_t - g, \mathbf{Q}_t - \mathbf{u})^\top$ in (3.48) has zero projection on the kernel set of $\mathcal{V}$.

Lemma 3.6.1 (Projection on $\ker(\mathcal{V})$). *For $h \in L^2(\mu) \times \mathbb{R}^M$, if $\mathcal{V}h = 0$ then $\langle h, \mathcal{V}m \rangle = 0$ for any $m = [\mathcal{A}f - g, f(x_1) - u(x_1), \dots, f(x_M) - u(x_M)]^\top$ where $f \in \mathcal{H}_{(0)}^2$.*

Proof. Let us begin with the following remark.

Remark 3.6.2. The setting we consider has a small modification as compared to the original PINNs formulation: we consider the limit ODE for an algorithm which samples points at random using the measure μ instead of using an *a priori* fixed set of points. We study this setting because if the PDE residual is minimized only on a fixed sparse set of points, then there is no guarantee that the approximator will converge to the solution – given any function, it is possible to perturb the function locally around each measurement point to construct functions which will have zero residual at these points; such functions will generally have no relation to the solution of the PDE. For similar reasons, we assume that the boundary value of the PDE is perfectly known.

Denote by $\varrho = \begin{bmatrix} \varrho^d \\ \varrho^p \end{bmatrix}$ an eigenfunction of $\mathcal{V}$ which has an eigenvalue zero:

$$\mathcal{V} \begin{bmatrix} \varrho^d \\ \varrho^p \end{bmatrix} = \mathcal{V}\varrho = 0\varrho = \begin{bmatrix} \mathcal{S} & \bar{\mathcal{U}}^* \\ \bar{\mathcal{U}} & \bar{\mathcal{B}} \end{bmatrix} \begin{bmatrix} \varrho^d \\ \varrho^p \end{bmatrix}. \quad (3.51)$$

We now show that

$$\left\langle \varrho, \begin{bmatrix} \mathcal{A}f - g \\ \mathbf{f} - \mathbf{u} \end{bmatrix} \right\rangle = 0, \quad (3.52)$$

for any $f \in \mathcal{H}_{(0)}^2$. This is because, similar to (2.49), we have

$$\langle \varrho, \mathcal{V}\varrho \rangle \geq \mathbb{E} \left[\left(\int_{\Omega} \varrho^d(x) \mathcal{A}[\eta(x) \sigma_{w,b}(x)] \mu(dx) + \int_{\Omega} \varrho^p(x) \eta(x) \sigma_{w,b}(x) d\mu_{\mathbf{x}}(x) \right)^2 \right] \geq 0. \quad (3.53)$$

Therefore, $\mathcal{V}\varrho = 0$ implies that for any (w, b) pairs

$$\int_{\Omega} \varrho^d(x) \mathcal{A}[\eta(x) \sigma(x; w, b)] \mu(dx) + \int_{\Omega} \varrho^p(x) \eta(x) \sigma(x; w, b) d\mu_{\mathbf{x}}(x) = 0. \quad (3.54)$$

Now, for any $f - u \in \mathcal{H}_{(0)}^2$, there exists a function sequence $\{F_N : \eta \sum_{i=1}^N c_i \sigma_{w,b}\}_{N \geq 1}$ such that

$$\lim_{N \rightarrow \infty} \|F_N - (f - u)\|_{\mathcal{H}^2}^2 = 0, \quad (3.55)$$

and simultaneously

$$\lim_{N \rightarrow \infty} \|F_N - (f - u)\|_{L^2(\mu(\mathbf{x}))}^2 = \frac{1}{M} \sum_{i=1}^M [F_N(x_i) - (f(x_i) - u(x_i))]^2 = 0, \quad (3.56)$$

we have

$$\int_{\Omega} \varrho^d(x) \mathcal{A}F^N(x) \mu(dx) + \int_{\Omega} \varrho^p(x) F^N(x) d\mu_{\mathbf{x}}(x) = 0. \quad (3.57)$$

This implies that

$$\int_{\Omega} \varrho^d(x) \mathcal{A}[f - u](x) \mu(dx) + \int_{\Omega} \varrho^p(x) [f - u](x) d\mu_{\mathbf{x}}(x) = 0, \quad (3.58)$$

and hence (3.52) is proven. $\square$

Theorem 3.6.3 (Global convergence of PINNs objective). *Assume that the PDE (3.3) admits a continuous $\mathcal{H}^2$ solution. The objective function $\|\mathcal{A}Q_t - g\|_{L^2(\mu)}^2 + \|Q_t - u\|_{L^2(\mu_{\mathbf{x}})}^2$ then converges to zero:*

$$\lim_{t \rightarrow \infty} \left(\|\mathcal{A}Q_t - g\|_{L^2(\mu)}^2 + \|Q_t - u\|_{L^2(\mu_{\mathbf{x}})}^2 \right) = 0. \quad (3.59)$$

Proof. From (3.48) we see that

$$\frac{d}{dt} \left[\|\mathcal{A}Q_t - g\|_{L^2(\mu)}^2 + \|Q_t - u\|_{L^2(\mu_{\mathbf{x}})}^2 \right] = - \begin{bmatrix} \mathcal{A}Q_t - g \\ \mathbf{Q}_t - \mathbf{u} \end{bmatrix}^{\top} \mathcal{V} \begin{bmatrix} \mathcal{A}Q_t - g \\ \mathbf{Q}_t - \mathbf{u} \end{bmatrix} \leq 0, \quad (3.60)$$

and that the equality holds iff $Q_t = u$. Therefore, the optimization objective is decreasing.

Consider $\tilde{\mathbf{Q}}_t = [\mathcal{A}Q_t - g, \mathbf{Q}_t - \mathbf{u}]^\top$ projected on $\{\vartheta\}_{i \in \mathbb{N}^+} \cup \{\varrho_j\}_{j \in \mathbb{N}^+}$. By Lemma 3.6.1, we have

$$\tilde{\mathbf{Q}}_t = \sum_i h_t^i \vartheta_i + \sum_j 0 \varrho_j = \sum_i h_t^i \vartheta_i. \quad (3.61)$$

Therefore $\|\mathcal{A}Q_t - g\|_{L^2(\mu)}^2 + \|Q_t - u\|_{L^2(\mu_{\mathbf{x}})}^2 = \sum_i h_t^{i^2}$. Now consider its projection on each ϑ_i

$$\frac{d}{dt} \langle \tilde{\mathbf{Q}}_t, \vartheta_i \rangle = \left\langle \frac{d\tilde{\mathbf{Q}}_t}{dt}, \vartheta_i \right\rangle = \langle -\mathcal{V}\tilde{\mathbf{Q}}_t, \vartheta_i \rangle = \langle \tilde{\mathbf{Q}}_t, -\mathcal{V}\vartheta_i \rangle = -\lambda_i \langle \tilde{\mathbf{Q}}_t, \vartheta_i \rangle. \quad (3.62)$$

Therefore

$$\frac{d}{dt} \langle \tilde{\mathbf{Q}}_t, \vartheta_i \rangle = \frac{d}{dt} [h_t^i] = -\lambda_i h_t^i. \quad (3.63)$$

Consequently $h_t^i = h_0^i e^{-\lambda_i t}$ whose absolute value decays exponentially, and $|h_t^i| \leq |h_0^i|$ for any $t \geq 0$ and any i .

Now, by the dominated convergence theorem

$$\lim_{t \rightarrow \infty} \|\mathcal{A}Q_t - g\|_{L^2(\mu)}^2 + \|Q_t - u\|_{L^2(\mu_{\mathbf{x}})}^2 = \lim_{t \rightarrow \infty} \sum_i |h_t^i|^2 = \sum_i \lim_{t \rightarrow \infty} |h_t^i|^2 = 0. \quad (3.64)$$

□

Theorem 3.6.4 (Global convergence of PINNs). *If $\mathcal{A}$ has an L^2 bounded inverse and the PDE (3.3) admits a continuous $\mathcal{H}^2$ solution, then Q_t converges to the solution of the PDE*

$$\lim_{t \rightarrow \infty} \left(\|Q_t - u\|_{L^2(\mu)}^2 + \|Q_t - u\|_{L^2(\mu_{\mathbf{x}})}^2 \right) = 0.$$

Proof. As the existence of an inverse guarantees that the PDE (3.3) admits a (unique) solution, it is clear from Theorem 3.6.3 that $\|Q_t - u\|_{L^2(\mu_{\mathbf{x}})} \rightarrow 0$ and $\|\mathcal{A}Q_t - g\|_{L^2(\mu)}^2 \rightarrow 0$. As in the proof of Theorem 3.5.4, the convergence of the residual implies the convergence of Q_t to u , given $\mathcal{A}^{-1}$ is a bounded operator. □

3.6.1 On related matters of DGM and PINNs

3.6.1.1 Convergence rates

Earlier, in Section 3.5 and the first part of Section 3.6, we have proved convergence rates for the projection of the solution onto the eigenfunctions. We have an exponential convergence rate for these projections. However, the convergence rate depends upon the eigenvalue – As we have highlighted, the kernel of the neural network has

no spectral gap, therefore its eigenmodes concentrate at 0^+ , thus we cannot prove an overall convergence rate in theory. In practice, the rate of convergence depends on the architecture of neural networks which implicitly influences the kernel property, the problem setting which decides the distribution of eigenfunctions, etc. For empirical studies on this matter, readers could check these papers listed at the end of Section 1.2.1, and references therein.

3.6.1.2 Comparing our analysis with empirical numerical results

Both PINNs and DGM are extremely popular, with their application shown up in a huge number of papers in not only applied mathematics, but also in different sectors of natural sciences and engineering fields. For a review of these applications, check the references listed in the second to last paragraph of Section 1.2.1, and also enormous lists of references there in.

The difference between PINNs and DGM is also briefly covered in this thesis, earlier in Chapter 1. As we have explained, when PINNs apply to use a fixed grid of points, the minimization of its training loss does not guarantee that PINNs find a good approximation of the PDE solution, as these points do not form a stencil. Therefore, to guarantee the convergence of PINNs, additional penalty terms are possibly required, and we are not covering this topic here in this chapter. However, readers can also check the references listed after equation (1.8) for this matter.

Another important topic related to both our research and the existing numerical literature is the failure modes of PINNs and DGM. There is also a vast list of good literature already examining this problem, see for instance [31, 124, 75, 149, 151, 50]. Some of these papers examine the high-frequency modes of PINNs and DGM, which coincides with the lack of convergence rate shown in our proof: Those high-frequency modes deny possible convergence rates in theory, and also may stop these algorithms from practically working on specific cases. From the author's perspective, we try to show that, as the scale (width) of the neural network grows, in its limit the global optimizer is achievable. Though we are not able to provide a certain convergence rate, this is a positive result as it suggests that performance improvement is possible when more neurons are added in.

3.7 Proofs and additional information

3.7.1 Proof of Theorem 3.3.7

The main idea of the proof is the split the difference into multiple residual terms and provide a bound for each of them. Then we apply Grönwall's inequality to provide an estimate of the difference between Q^N and Q . In this proof, constant C may vary from line to line, but it remains invariant with the number of neurons N and training time t . Expectations are taken with respect to random initialization of neural network parameters $\{c^i, w^i, b^i\}_{i=1, \dots, N}$.

We start with the following auxiliary lemmas which provide bounds for residual terms in our estimate.

Lemma 3.7.1. *We define residual term $M_1 :=$*

$$\int_0^T N^{2\beta+\delta-1} \left(\int_{\Omega^2} |\Phi^N(\nabla_{\theta} \mathcal{A}Q_s^N(x)) \cdot \sum_{\alpha} D_{\alpha} \nabla_{\theta}(Q_s^N - Q_0^N)(y)|^2 \mu(dx) \mu(dy) \right)^{\frac{1}{2}} ds, \quad (3.65)$$

It satisfies

$$\lim_{N \rightarrow \infty} \mathbb{E} M_1 = 0. \quad (3.66)$$

Proof. By Jensen's inequality, it suffices to prove that for any indices α

$$\int_0^T N^{2\beta+\delta-1} \left(\int \mathbb{E} \left[|\Phi^N(\nabla_{\theta} \mathcal{A}Q_s^N(x)) \cdot D_{\alpha} \nabla_{\theta}(Q_s^N - Q_0^N)(y)|^2 \right] \mu(dx) \mu(dy) \right)^{\frac{1}{2}} ds \rightarrow 0. \quad (3.67)$$

Notice that $|\Phi^N|$ is bounded elementwise by $N^{\epsilon-\beta}$. We also notice that by the mean value theorem and that η, σ , and their derivatives are up to polynomial growth

$$\begin{aligned} \|D_{\alpha}^y(\nabla_{\theta} Q_s^N - \nabla_{\theta} Q_0^N)(y)\|_1 &\leq \frac{C}{N^{\beta}} \sum_{i=1}^N \left((\|w_s^i - w_0^i\| + \|b_s^i - b_0^i\| \right. \\ &\quad \left. + \|c_s^i - c_0^i\|) f(w_0^i, c_0^i, b_0^i, y) \right) \end{aligned} \quad (3.68)$$

where f is a polynomial up to 3rd order. We notice that $\|w_u^i - w_0^i\| + \|b_u^i - b_0^i\| + \|c_u^i - c_0^i\| \leq uCN^{\epsilon+\delta+\beta-1}$. Therefore

$$\mathbb{E} \left[|\Phi^N(\nabla_{\theta} \mathcal{A}Q_s^N(x)) \cdot D_{\alpha}(\nabla_{\theta} Q_s^N - \nabla_{\theta} Q_0^N)(y)|^2 \right] \leq CN^{2(2\epsilon+\delta-\beta)}. \quad (3.69)$$

Therefore, the left-hand side of (2.27) is smaller than $CN^{2\delta+2\epsilon+\beta-1}$, which goes to 0 as $N \rightarrow \infty$. $\square$

Lemma 3.7.2. *We define residual term*

$$M_2 := \int_0^T N^{2\beta+\delta-1} \left(\int \left| (\Phi^N(\nabla_\theta \mathcal{A}Q_s^N(x)) - \Phi^N(\nabla_\theta \mathcal{A}Q_0^N(x))) \cdot \sum_\alpha D_\alpha \nabla_\theta Q_0^N(y) \right|^2 \mu(dx)\mu(dy) \right)^{\frac{1}{2}} ds \quad (3.70)$$

It satisfies

$$\lim_{N \rightarrow \infty} \mathbb{E} M_2 = 0. \quad (3.71)$$

Proof. Similarly, it suffices to show that for any α that

$$\int_0^T N^{2\beta+\delta-1} \left(\int_{\Omega^2} \mathbb{E} \left[\left| (\Phi^N(\nabla_\theta \mathcal{A}Q_s^N(x)) - \Phi^N(\nabla_\theta \mathcal{A}Q_0^N(x))) \cdot D_\alpha \nabla_\theta Q_0^N(y) \right|^2 \right] \mu(dx)\mu(dy) \right)^{\frac{1}{2}} ds \rightarrow 0. \quad (3.72)$$

As, elementwise, $|\phi^N(x) - \phi^N(y)| \leq |x - y|$, we have

$$\begin{aligned} \left| (\Phi^N(\nabla_\theta \mathcal{A}Q_s^N(x)) - \Phi^N(\nabla_\theta \mathcal{A}Q_0^N(x))) \cdot D_\alpha \nabla_\theta Q_0^N(y) \right| &\leq \left| \nabla_\theta \mathcal{A}Q_s^N(x) - \nabla_\theta \mathcal{A}Q_0^N(x) \right| \\ &\quad \cdot \left| D_\alpha \nabla_\theta Q_0^N(y) \right| \end{aligned} \quad (3.73)$$

Then, since

$$\begin{aligned} \left| (\nabla_\theta \mathcal{A}Q_s^N(x) - \nabla_\theta \mathcal{A}Q_0^N(x)) \cdot D_\alpha \nabla_\theta Q_0^N(y) \right| &\leq \frac{C}{N^{2\beta}} \sum_{i=1}^N \left((\|w_s^i - w_0^i\| + \|b_s^i - b_0^i\| \right. \\ &\quad \left. + \|c_s^i - c_0^i\|) g(w_0^i, c_0^i, b_0^i, x, y) \right) \end{aligned} \quad (3.74)$$

where g is a polynomial up to 3rd order. Since $\|w_u^i - w_0^i\| + \|b_u^i - b_0^i\| + \|c_u^i - c_0^i\| \leq uCN^{\epsilon+\delta+\beta-1}$, we have

$$\mathbb{E} \left[\left| (\Phi^N(\nabla_\theta \mathcal{A}Q_s^N(x)) - \Phi^N(\nabla_\theta \mathcal{A}Q_0^N(x))) \cdot D_\alpha \nabla_\theta Q_0^N(y) \right|^2 \right] \leq CN^{2(\epsilon+\delta-\beta)}. \quad (3.75)$$

The left-hand side of (3.72) is bounded by $CN^{\epsilon+\delta+\beta-1}$, which goes to 0 as N goes to infinity. $\square$

Lemma 3.7.3. *We define residual term*

$$M_3 := \int_0^T N^{2\beta+\delta-1} \left(\int_{\Omega^2} |(\Phi^N(\nabla_\theta \mathcal{A}Q_0^N(x)) - \nabla_\theta \mathcal{A}Q_0^N(x)) \cdot \sum_{\alpha} D_\alpha \nabla_\theta Q_0^N(y)|^2 \mu(dx)\mu(dy) \right)^{\frac{1}{2}} ds. \quad (3.76)$$

We have

$$\lim_{N \rightarrow \infty} \mathbb{E} M_3 = 0. \quad (3.77)$$

Proof. By Jensen's inequality, it suffices to show that

$$\int_0^T N^{2\beta+\delta-1} \left(\int_{\Omega^2} \mathbb{E} \left[|(\Phi^N(\nabla_\theta \mathcal{A}Q_0^N(x)) - \nabla_\theta \mathcal{A}Q_0^N(x)) \cdot D_\alpha \nabla_\theta Q_0^N(y)|^2 \right] \mu(dx)\mu(dy) \right)^{\frac{1}{2}} ds \rightarrow 0 \quad (3.78)$$

for any indices α where $|\alpha| \leq 2$. Let us denote the k -th unit of the initial approximator Q_0^N as q^k , where $q^k = c_0^k N^{-\beta} \eta \sigma_{w_0^k, b_0^k}$. Then $\mathbb{E}[|(\Phi^N(\nabla_\theta \mathcal{A}Q_0^N(x)) - \nabla_\theta \mathcal{A}Q_0^N(x)) \cdot D_\alpha \nabla_\theta Q_0^N(y)|^2]$

$$\begin{aligned} &= \mathbb{E} \left[\left(\sum_{k=1}^N [\Phi^N(\nabla_\theta \mathcal{A}q^k(x)) - \nabla_\theta \mathcal{A}q^k(x)] \cdot \nabla_\theta D_\alpha q^k(y) \right)^2 \right] \\ &= N \mathbb{E} \left[(\Phi^N(\nabla_\theta \mathcal{A}q(x)) - \nabla_\theta \mathcal{A}q(x)) \cdot \nabla_\theta D_\alpha q(y) \right]^2 + N(N-1) \mathbb{E} \left[\Phi^N(\nabla_\theta \mathcal{A}q(x)) \right. \\ &\quad \left. - \nabla_\theta \mathcal{A}q(x) \right] \cdot \nabla_\theta D_\alpha q(y) \Big]^2. \end{aligned} \quad (3.79)$$

Then, by definition

$$\begin{aligned} &\Phi^N(\nabla_\theta \mathcal{A}q(x)) - \nabla_\theta \mathcal{A}q(x) \cdot \nabla_\theta D_\alpha q(y) \\ &= [\phi^N(\mathcal{A}[N^{-\beta} \eta \sigma_{w,b}](x)) - \mathcal{A}[N^{-\beta} \eta \sigma_{w,b}](x)] \cdot D_\alpha [N^{-\beta} \eta \sigma_{w,b}](y) \\ &\quad + \sum_{i=1}^d [\phi^N(\mathcal{A}[N^{-\beta} c x_i \eta \sigma'_{w,b}](x)) - \mathcal{A}[N^{-\beta} c x_i \eta \sigma'_{w,b}](x)] \cdot D_\alpha [N^{-\beta} c y_i \eta \sigma'_{w,b}](y) \\ &\quad + [\phi^N(\mathcal{A}[N^{-\beta} c \eta \sigma'_{w,b}](x)) - \mathcal{A}[N^{-\beta} c \eta \sigma'_{w,b}](x)] \cdot D_\alpha [N^{-\beta} c \eta \sigma'_{w,b}](y). \end{aligned} \quad (3.80)$$

We introduce a uniform bound for $|D_\alpha[\eta \sigma_{w,b}(x)]|$, $|D_\alpha[c x_i \eta \sigma_{w,b}(x)]|$ and also the term $|D_\alpha[c \eta \sigma_{w,b}(x)]|$ for any indices α and any x . By the fact that $\mathcal{A}$ is Lipschitz,

$$|\mathcal{A}[\eta \sigma_{w,b}](x)| \leq k_0 \sum_{0 \leq |\alpha| \leq 2} |D_\alpha[\eta \sigma_{w,b}](x)|. \quad (3.81)$$

Notice that

$$\begin{aligned}
|D_\alpha[\eta\sigma_{w,b}](x)| &= \left| \sum_{\alpha_1+\alpha_2=\alpha} D_{\alpha_1}\eta \cdot D_{\alpha_2}\sigma_{w,b}(x) \right| \leq k_\eta \left| \sum_{\alpha_1+\alpha_2=\alpha} D_{\alpha_2}\sigma_{w,b}(x) \right| \\
&\leq k_\eta \left| \sum_{0 \leq |\alpha_2| \leq 2} D_{\alpha_2}\sigma_{w,b}(x) \right| \\
&\leq k_\eta k_\sigma k_\Omega \left(1 + \sum_{i=1}^d |(w)_i| + \sum_{i,j=1}^d |(w)_i(w)_j| \right).
\end{aligned} \tag{3.82}$$

Therefore

$$|\mathcal{A}[\eta\sigma_{w,b}](x)| \leq k_0 \sum_{0 \leq |\alpha| \leq 2} |D_\alpha[\eta\sigma_{w,b}](x)| \leq k \left(1 + \sum_{i=1}^d |(w)_i| + \sum_{i,j=1}^d |(w)_i(w)_j| \right). \tag{3.83}$$

Similar results hold for $|D_\alpha[cx_i\eta\sigma_{w,b}]|$ and $|D_\alpha[c\eta\sigma_{w,b}]|$. We define $f(w) := k(1 + \sum_{i=1}^d |(w)_i| + \sum_{i,j=1}^d |(w)_i(w)_j|)$. Now $|\phi^N(\mathcal{A}[N^{-\beta}\eta\sigma_{w,b}](x)) - \mathcal{A}[N^{-\beta}\eta\sigma_{w,b}](x)|$

$$\begin{aligned}
&\leq N^{-\beta} \left[|\mathcal{A}[\eta\sigma_{w,b}](x)| - N^\epsilon \right] \mathbf{1}_{\{|\mathcal{A}[\eta\sigma_{w,b}](x)| \geq N^\epsilon\}}(x) \\
&\leq N^{-\beta} [f(w) - N^\epsilon] \mathbf{1}_{\{f(w) \geq N^\epsilon\}}(x).
\end{aligned} \tag{3.84}$$

Subsequently,

$$|\Phi^N(\nabla_\theta \mathcal{A}q(x)) - \nabla_\theta \mathcal{A}q(x)] \cdot \nabla_\theta D_\alpha q(y)| \leq N^{-2\beta}(d+2)[f(w) - N^\epsilon] \mathbf{1}_{\{f(w) \geq N^\epsilon\}}(x). \tag{3.85}$$

Therefore

$$N\mathbb{E} \left[\left(\Phi^N(\nabla_\theta \mathcal{A}q(x)) - \nabla_\theta \mathcal{A}q(x)] \cdot \nabla_\theta D_\alpha q(y) \right)^2 \right] \leq N^{1-4\beta}(d+2)\mathbb{E}[f(w)^2] \leq kN^{1-4\beta}. \tag{3.86}$$

Meanwhile $\mathbb{E}[\Phi^N(\nabla_\theta \mathcal{A}q(x)) - \nabla_\theta \mathcal{A}q(x)] \cdot \nabla_\theta D_\alpha q(y)$

$$\begin{aligned}
&\leq \mathbb{E}[N^{-2\beta}(d+2)[f(w) - N^\epsilon] \mathbf{1}_{\{f(w) \geq N^\epsilon\}}(x)] \leq N^{-2\beta}(d+2)\mathbb{E}[f(w) \mathbf{1}_{\{f(w) \geq N^\epsilon\}}(x)] \\
&\leq k_0 N^{-2\beta-\epsilon} \mathbb{E}[f(w)^2] \leq kN^{-2\beta-\epsilon}.
\end{aligned} \tag{3.87}$$

Hence

$$N(N-1)\mathbb{E} \left[\Phi^N(\nabla_\theta \mathcal{A}q(x)) - \nabla_\theta \mathcal{A}q(x)] \cdot \nabla_\theta D_\alpha q(y) \right]^2 \leq kN^{2-4\beta-2\epsilon}. \tag{3.88}$$

Combining (3.86) and (3.88), we conclude that

$$M_3 \leq k(N^{\delta-1/2} + N^{\delta-\epsilon}) \rightarrow 0 \tag{3.89}$$

as $N \rightarrow \infty$. $\square$

Lemma 3.7.4. *We define residual term*

$$M_4 := \mathbb{E} \left[\int_0^T N^\delta \left(\int_{\Omega^2} |N^{2\beta-1} \nabla_\theta \mathcal{A} Q_0^N(x) \cdot \sum_\alpha D_\alpha \nabla_\theta Q_0^N(y) - \sum_\alpha D_\alpha^y U(x, y)|^2 \mu(dx) \mu(dy) \right)^{\frac{1}{2}} ds \right]. \quad (3.90)$$

It satisfies

$$\lim_{N \rightarrow \infty} \mathbb{E} M_4 = 0. \quad (3.91)$$

Proof. It suffices to prove that for any indices α

$$\int_0^T N^\delta \left(\int \mathbb{E} \left[|N^{2\beta-1} \nabla_\theta \mathcal{A} Q_0^N(x) \cdot D_\alpha \nabla_\theta Q_0^N(y) - D_\alpha^y U(x, y)|^2 \right] \mu(dx) \mu(dy) \right)^{\frac{1}{2}} ds \quad (3.92)$$

converges to 0 as $N \rightarrow \infty$. By the strong law of large numbers:

$$\lim_{N \rightarrow \infty} N^{2\beta-1} \nabla_\theta \mathcal{A} Q_0^N(x) \cdot D_\alpha \nabla_\theta Q_0^N(y) = D_\alpha^y U(x, y). \quad (3.93)$$

Therefore

$$\begin{aligned} & \mathbb{E} \left[|N^{2\beta-1} \nabla_\theta \mathcal{A} Q_0^N(x) \cdot D_\alpha \nabla_\theta Q_0^N(y) - D_\alpha^y U(x, y)|^2 \right] \\ &= \text{Var} [N^{2\beta-1} \nabla_\theta \mathcal{A} Q_0^N(x) \cdot D_\alpha \nabla_\theta Q_0^N(y)] = \frac{1}{N} \text{Var} [D_\alpha^y U_{c,w,b}(x, y)]. \end{aligned} \quad (3.94)$$

Consequently, the left-hand side of (3.92) is bounded by $CN^{\delta-1}$ which vanishes as N goes to infinity. $\square$

Lemma 3.7.5. *We have*

$$\lim_{N \rightarrow \infty} \int_0^T \int_\Omega |\psi^N(\mathcal{L} Q_s(x)) - \mathcal{L} Q_s(x)| \mu(dx) ds = 0. \quad (3.95)$$

Proof. Notice that

$$|\psi^N(\mathcal{L} Q_s(x)) - \mathcal{L} Q_s(x)| \leq |\mathcal{L} Q_s(x)| \mathbf{1}_{\{|\mathcal{L} Q_s| > N^\delta\}}(x). \quad (3.96)$$

By the dominated convergence theorem, we conclude our proof. $\square$

Lemma 3.7.6. *We have*

$$\lim_{N \rightarrow \infty} \mathbb{E} \left[\left(\int_\Omega \left[\sum_\alpha |D_\alpha(Q_0^N - Q_0)(y)| \right]^2 \mu(dy) \right)^{\frac{1}{2}} \right] = 0. \quad (3.97)$$

Proof. By Jensen's inequality, it suffices to show that for any α

$$\int_{\Omega} \mathbb{E}[D_{\alpha}(Q_0^N - Q_0)(y)^2] \mu(dy) \rightarrow 0. \quad (3.98)$$

Since $\mathbb{E}[Q_0^N] = Q_0 = 0$ holds for all y we have

$$\mathbb{E}[D_{\alpha}(Q_0^N - Q_0)(y)^2] = \text{Var}[D_{\alpha}Q_0^N(y)] = N^{1-2\beta} \text{Var}[D_{\alpha}c\sigma(wy + b)] \leq CN^{1-2\beta}. \quad (3.99)$$

Integrating with respect to $\mu(y)$, and letting N go to infinity finish the proof. $\square$

Now we present our main proof.

Proof of Theorem 3.3.7. To simplify our notations, we set $\mathcal{L}Q := \mathcal{A}Q - g$. In the integral form, we have

$$Q_t^N(y) = Q_0^N(y) - \int_0^t \int_{\Omega} \psi^N(\mathcal{L}Q_s^N(x)) \alpha^N \Phi^N(\nabla_{\theta} \mathcal{A}Q_s^N(x)) \cdot \nabla_{\theta} Q_s^N(y) \mu(dx) ds. \quad (3.100)$$

Similarly, at the same time

$$Q_t(y) = Q_0(y) - \int_0^t \int_{\Omega} \mathcal{L}Q_s(x) U(x, y) \mu(dx) ds. \quad (3.101)$$

Subtracting (3.101) from (3.100) and taking partial derivative with respect to y with indices α give

$$\begin{aligned} & |D_{\alpha}(Q_t^N - Q_t)(y)| \\ & \leq \int_0^t \int_{\Omega} \left| \psi^N(\mathcal{L}Q_s^N(x)) \alpha^N \Phi^N(\nabla_{\theta} \mathcal{A}Q_s^N(x)) \cdot D_{\alpha} \nabla_{\theta} Q_s^N(y) - \mathcal{L}Q_s^N(x) D_{\alpha}^y U(x, y) \right| \\ & \quad \mu(dx) ds + |D_{\alpha}(Q_0^N - Q_0)(y)|, \end{aligned} \quad (3.102)$$

where the right-hand side can be further bounded by

$$\begin{aligned}
&\leq \int_0^t \int_{\Omega} \left| \psi^N(\mathcal{L}Q_s^N(x)) \alpha^N \Phi^N(\nabla_{\theta} \mathcal{A}Q_s^N(x)) \cdot D_{\alpha}(\nabla_{\theta} Q_s^N - \nabla_{\theta} Q_0^N)(y) \right| \mu(dx) ds \\
&\quad + \int_0^t \int_{\Omega} \left| \psi^N(\mathcal{L}Q_s^N(x)) \alpha^N (\Phi^N(\nabla_{\theta} \mathcal{A}Q_s^N(x)) - \Phi^N(\nabla_{\theta} \mathcal{A}Q_0^N(x)) \cdot \right. \\
&\quad \left. D_{\alpha} \nabla_{\theta} Q_0^N(y) \right| \mu(dx) ds \\
&\quad + \int_0^t \int_{\Omega} \left| \psi^N(\mathcal{L}Q_s^N(x)) \alpha^N (\Phi^N(\nabla_{\theta} \mathcal{A}Q_0^N(x)) - \nabla_{\theta} \mathcal{A}Q_0^N(x)) \cdot D_{\alpha} \nabla_{\theta} Q_0^N(y) \right| \mu(dx) ds \\
&\quad + \int_0^t \int_{\Omega} \left| \psi^N(\mathcal{L}Q_s^N(x)) [\alpha^N \nabla_{\theta} \mathcal{A}Q_0^N(x) \cdot D_{\alpha} \nabla_{\theta} Q_0^N(y) - D_{\alpha}^y U(x, y)] \right| \mu(dx) ds \\
&\quad + \int_0^t \int_{\Omega} \left| [\psi^N(\mathcal{L}Q_s^N(x)) - \psi^N(\mathcal{L}Q_s(x))] D_{\alpha}^y U(x, y) \right| \mu(dx) ds \\
&\quad + \int_0^t \int_{\Omega} \left| [\psi^N(\mathcal{L}Q_s(x)) - \mathcal{L}Q_s(x)] D_{\alpha}^y U(x, y) \right| \mu(dx) ds + |D_{\alpha}(Q_0^N - Q_0)(y)|.
\end{aligned} \tag{3.103}$$

Now by the fact that $|\psi^N| < N^{\delta}$, $|D_{\alpha}^y U(x, y)| < C$ we have

$$\begin{aligned}
&|D_{\alpha}(Q_t^N - Q_t)(y)| \\
&\leq \int_0^t \int_{\Omega} N^{2\beta+\delta-1} \left| \Phi^N(\nabla_{\theta} \mathcal{A}Q_s^N(x)) \cdot D_{\alpha}(\nabla_{\theta} Q_s^N - \nabla_{\theta} Q_0^N)(y) \right| \mu(dx) ds \\
&\quad + \int_0^t \int_{\Omega} N^{2\beta+\delta-1} \left| (\Phi^N(\nabla_{\theta} \mathcal{A}Q_s^N(x)) - \Phi^N(\nabla_{\theta} \mathcal{A}Q_0^N(x)) \cdot D_{\alpha} \nabla_{\theta} Q_0^N(y) \right| \mu(dx) ds \\
&\quad + \int_0^t \int_{\Omega} N^{2\beta+\delta-1} \left| (\Phi^N(\nabla_{\theta} \mathcal{A}Q_0^N(x)) - \nabla_{\theta} \mathcal{A}Q_0^N(x)) \cdot D_{\alpha} \nabla_{\theta} Q_0^N(y) \right| \mu(dx) ds \\
&\quad + \int_0^t \int_{\Omega} N^{\delta} \left| N^{2\beta-1} \nabla_{\theta} \mathcal{A}Q_0^N(x) \cdot D_{\alpha} \nabla_{\theta} Q_0^N(y) - D_{\alpha}^y U(x, y) \right| \mu(dx) ds \\
&\quad + C \int_0^t \int_{\Omega} \left| \psi^N(\mathcal{L}Q_s^N(x)) - \psi^N(\mathcal{L}Q_s(x)) \right| \mu(dx) ds \\
&\quad + C \int_0^t \int_{\Omega} \left| \psi^N(\mathcal{L}Q_s(x)) - \mathcal{L}Q_s(x) \right| \mu(dx) ds + |D_{\alpha}(Q_0^N - Q_0)(y)|.
\end{aligned} \tag{3.104}$$

Let us denote $V_t^N(y) := \sum_{|\alpha| \leq 2} |D_{\alpha}(Q_t^N - Q_t)(y)|$. Then by summing inequality (3.104) with respect to all indices and integrating with respect to $\mu(y)$

$$\int_{\Omega} V_t^N(y)^2 \mu(dy) \leq \left(\int_{\Omega} V_t^N(y)^2 \mu(dy) \right)^{\frac{1}{2}} \left[C \int_0^t \left(\int_{\Omega} V_s^N(x)^2 \mu(dx) \right)^{\frac{1}{2}} ds + M \right] \tag{3.105}$$

where $M := M_1 + M_2 + M_3 + M_4 + M_5 + M_6$ denotes the sum of residual terms where M_1 to M_4 have been defined in previous lemmas and

$$\begin{aligned} M_5 &:= \int_0^T \int_{\Omega} |\psi^N(\mathcal{L}Q_s(x)) - \mathcal{L}Q_s(x)| \mu(dx) ds, \\ M_6 &:= \left(\int_{\Omega} \left[\sum_{\alpha} |D_{\alpha}(Q_0^N - Q_0)(y)|^2 \mu(dy) \right] \right)^{\frac{1}{2}}. \end{aligned} \quad (3.106)$$

Then by Grönwall's inequality, since from (3.105) we have

$$\left(\int_{\Omega} V_t^N(y)^2 \mu(dy) \right)^{\frac{1}{2}} \leq C \int_0^t \left(\int_{\Omega} V_s^N(x)^2 \mu(dx) \right)^{\frac{1}{2}} ds + M, \quad (3.107)$$

we derive that

$$\int_0^t \left(\int_{\Omega} V_s^N(x)^2 \mu(dx) \right)^{\frac{1}{2}} ds \leq \frac{M}{C} e^{Ct}. \quad (3.108)$$

Taking expectation on both sides, by Lemmas 3.7.1, 3.7.2, 3.7.3, 3.7.4, 3.7.5, 3.7.6 we bound each of the terms in M . Hence

$$\lim_{N \rightarrow \infty} \mathbb{E} \left[\int_0^t \left(\int_{\Omega} V_s^N(x)^2 \mu(dx) \right)^{\frac{1}{2}} ds \right] = 0. \quad (3.109)$$

By (3.107) we have for any $0 \leq s \leq T$

$$\lim_{N \rightarrow \infty} \mathbb{E} \left[\left(\int_{\Omega} V_s^N(x)^2 \mu(dx) \right)^{\frac{1}{2}} \right] = 0. \quad (3.110)$$

Notice that $\int_{\Omega} V_s^N(x)^2 \mu(dx) \geq \|Q_s^N - Q_s\|_{\mathcal{H}^2}^2$, we conclude that for any $0 \leq s \leq T$

$$\lim_{N \rightarrow \infty} \mathbb{E} \left[\|Q_s^N - Q_s\|_{\mathcal{H}^2} \right] = 0. \quad (3.111)$$

□

3.7.2 Exponential convergence of original PINNs objective

As stated in Chapter 1, original PINNs use the same batch of data points inside the domain and optimize its loss functional only on this same fixed set of points. We define the ‘discretized’ version of the operator $\mathcal{U}$ on a finite set of points as

$$\hat{\mathcal{U}}f = \int_{\Omega} f(x) U(x, \cdot) d\nu(x) \quad (3.112)$$

where $\nu = \frac{1}{M} \sum_{i=1}^M \delta_{x_i}$. Here $\{x_1, \dots, x_M\}$ are M randomly sampled data points in Ω before training iteration starts. Subsequently, the ‘discretized’ operator $\hat{\mathcal{S}}$ is defined by

$$\hat{\mathcal{S}}f := \mathcal{A}\hat{\mathcal{U}}f = \int_{\Omega} f(x)S(x, \cdot)d\nu(x). \quad (3.113)$$

Informally, the wide-limit ODE for the original PINNs’ limit is

$$\frac{dQ_t}{dt} = -\hat{\mathcal{U}}[\mathcal{A}Q_t - g], \quad Q_0 = q_0. \quad (3.114)$$

Applying $\mathcal{A}$ on both sides, multiplying by $\frac{1}{2}\mathcal{A}Q_t - g$, and integrating with respect to empirical measure ν show

$$\begin{aligned} \frac{d\|e_t\|_2^2}{dt} &= \frac{d\|\mathcal{A}Q_t - g\|_{L^2(\nu)}^2}{dt} = -\langle \mathcal{A}Q_t - g, \hat{\mathcal{S}}[\mathcal{A}Q_t - g] \rangle_{L^2(\nu)} \\ &= -e_t \hat{\mathcal{S}} e_t^\top \end{aligned} \quad (3.115)$$

where e_t is the PDE residual vector $e_t = [\mathcal{A}Q_t(x_1) - g(x_1), \dots, \mathcal{A}Q_t(x_M) - g(x_M)]$ and $\hat{\mathcal{S}}$ is the induced positive semi-definite matrix where

$$\hat{\mathcal{S}}(i, j) = S(x_i, x_j). \quad (3.116)$$

Similar to the proof in the previous section, e_t is orthogonal to the kernel set of matrix $\hat{\mathcal{S}}$ for all $t \geq 0$. Hence, $e_t = \exp(-\hat{\mathcal{S}}t)e_0 \rightarrow 0$ exponentially fast, where the convergence rate is lower-bounded by the smallest non-zero eigenvalue of $\hat{\mathcal{S}}$, $\lambda_{\min} > 0$.

3.7.3 A proposed uniformly stable activation function

We introduce clipping functions in gradient terms to avoid gradient explosion, as differential operators are not bounded. In this subsection, we propose a class of activation functions with which training neural networks for solving second-order linear PDE residual minimization tasks become stable.

Suppose that twice-differentiable, non-linear, bounded function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ satisfies:

$$|x^p D_k \sigma(x)| < C \quad (3.117)$$

for all $x \in \mathbb{R}$, for some $C > 0$ for $p = 0, 1, 2$ and $k = 0, 1, 2, 3$. We define an activation function $\Sigma : \mathbb{R}^d \rightarrow \mathbb{R}$

$$\Sigma^d(x; w, b) = \prod_{i=1}^d \sigma(w_i x_i + b_i) \quad (3.118)$$

where x_i is the i -th element of x .

Lemma 3.7.7 (Discriminatory). *Function class $\Sigma(\cdot; w, b)$ is discriminatory on any compact subset of $\mathbb{R}^d$.*

Proof. Since σ is non-linear and bounded, it is discriminatory on $\mathbb{R}$. Then inductively, Σ is discriminatory on cylinder sets in $\mathbb{R}^2, \mathbb{R}^3 \dots, \mathbb{R}^d$. $\square$

Now we define a single-layer network

$$\psi_{\theta}^N(x) = \frac{1}{N} \sum_{i=1}^N c^i \Sigma(x; w^i, b^i). \quad (3.119)$$

We assume that the domain Ω satisfies that for all $x \in \Omega$, its coordinate $\min_i \{x_i\} > C > 0$, we also assume that once $b^i \in L^2$ are randomly initialized, they remain fixed throughout training. Then we have the following result:

Lemma 3.7.8 (Stable for learning the Laplacian). *While minimizing the loss functional*

$$L(\theta) = \int_{\Omega} \left| \Delta \psi_{\theta}^N - f \right|^2 \mu(dx), \quad (3.120)$$

via gradient descent with a constant learning rate, if f is bounded on the closure of Ω , then the parameters of ψ^N remains stable.

Proof. The gradient of $L(\theta)$ satisfies

$$\nabla_{\theta} L = \int_{\Omega} \left(\Delta \psi_{\theta}^N - f \right) \nabla_{\theta} [\Delta \psi^N] \mu(dx). \quad (3.121)$$

Notice that by

$$\Delta \psi^N = \frac{1}{N} \sum_{k=1}^d \left(\sum_{i=1}^N c^i (w_k^i)^2 \sigma''(w_k^i x_k + b_k^i) \prod_{j \neq k} \sigma(w_j^i x_j + b_j^i) \right), \quad (3.122)$$

and the fact that

$$\left| w^2 \sigma''(wx + b) \right| = \frac{w^2 x^2}{x^2} \left| \sigma''(wx + b) \right| \leq \frac{2(wx + b)^2 + 2b^2}{x^2} \left| \sigma''(wx + b) \right| \leq C(1 + b^2), \quad (3.123)$$

we derive the boundedness of $|\Delta \psi^N|$

$$\left| \Delta \psi^N - f \right| \leq C \left(1 + \frac{\sum_{i=1}^N |c^i| (1 + \|b^i\|_2^2)}{N} \right). \quad (3.124)$$

By (3.123) and the fact that $|\sigma|$ is bounded, we have

$$\left| \nabla_{c^i} \Delta \psi^N \right| \leq \frac{C}{N} (1 + \|b^i\|_2^2). \quad (3.125)$$

Therefore,

$$\begin{aligned} \left| \frac{dc^i}{dt} \right| &\leq |\alpha| \int_{\Omega} \left| \Delta \psi^N - f \right| \left| \nabla_{c^i} \Delta \psi^N \right| \mu(dx) \leq \frac{C(1 + \|b^i\|_2^2)}{N} \left(1 + \frac{\sum_{i=1}^N |c^i| (1 + \|b^i\|_2^2)}{N} \right) \\ &\leq C \left(\frac{1}{N} + \frac{\|b^i\|_2^2}{N} \right) \left(1 + \frac{1 + \sum_{i=1}^N \|b^i\|_2^2}{N} \sum_{i=1}^N |c^i| \right). \end{aligned} \quad (3.126)$$

Subsequently, summing over i leads to:

$$\frac{d}{dt} \sum_{i=1}^N |c^i| \leq C \left(1 + \frac{\sum_{i=1}^N \|b^i\|_2^2}{N} \right) \left(1 + \frac{1 + \sum_{i=1}^N \|b^i\|_2^2}{N} \sum_{i=1}^N |c^i| \right). \quad (3.127)$$

Therefore

$$\sum_{i=1}^N |c_t^i| \leq \left(\sum_{i=1}^N |c_0^i| \right) \exp(C^* t) < \infty, \quad (3.128)$$

where $C^* = C \left(\frac{\sum_{i=1}^N \|b^i\|_2^2}{N} \right)$ is a constant depending on

$$\frac{\sum_{i=1}^N \|b^i\|_2^2}{N}. \quad (3.129)$$

Similarly, for w^i one can derive

$$\left| \nabla_{w_k^i} \Delta \psi^N \right| \leq \frac{C|c^i|}{N} (1 + |b_k^i| + |b_k^i|^2). \quad (3.130)$$

Then

$$\left| \frac{dw_k^i}{dt} \right| \leq C \left| \Delta \psi^N - f \right| \left| \nabla_{w_k^i} \Delta \psi^N \right| \leq C^{**} < \infty, \quad (3.131)$$

where $C^{**} = C \left(\frac{\sum_{i=1}^N \|b^i\|_2^2}{N}, \sum_{i=1}^N |c^i| \right)$ is a constant depending on $\frac{\sum_{i=1}^N \|b^i\|_2^2}{N}$ and $\sum_{i=1}^N |c^i|$. Since $\sum_{i=1}^N |c^i|$ is bounded up to any given time T , $\frac{dw_k^i}{dt}$ is bounded by a constant invariant of time. Thus, c^i, w^i remain stable through training. $\square$

Remark 3.7.9. With our proposed activation function, we no longer need the clipping functions to ensure the stability of the training process for learning the Laplacian. Gradient explosion is avoided. And of course, this result can be applied to any second-order PDE operators. Here we take the Laplacian operator as an example solely for readability reasons.

Chapter 4

Actor-critic methods for Hamilton–Jacobi–Bellman PDEs

Hamilton–Jacobi–Bellman (HJB) PDEs comprise an important class of nonlinear PDEs, connecting with stochastic optimal control problems. HJB PDEs are closely related to the dynamic programming principle (DPP) [5], and their solutions represent optimal expected values for controlled processes. Deep learning has been applied to stochastic control problems of different forms, for instance, [42, 13, 51, 120]. In this chapter, we study a neural-network-based algorithm for solving HJB PDEs that are related to certain control problems. Rather than solving the nonlinear PDE directly, this method decomposes the nonlinear PDE problem into a series of linear PDEs and seeks to approach the solution of the nonlinear PDE via iteratively finding the optimal control of the control problem.

In the first half of this chapter, we consider a continuous-time, continuous-space control problem where the controlled system is a (time-homogeneous) $\mathbb{R}^d$ -valued process. Let W_t be a d' -dimensional Brownian Motion defined on a filtered probability space $\{\bar{\Omega}, \mathcal{F}, (\mathcal{F}_t)_{t \geq 0}, \mathbb{P}\}$. The controlled SDE, for given measurable functions $b : \mathbb{R}^d \times A \rightarrow \mathbb{R}^d$ and $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d'}$, is

$$dX_t = b(X_t, u_t)dt + \Phi(X_t)dW_t, \quad X_0 = x \in \Omega \subset \mathbb{R}^d. \quad (4.1)$$

Here $\{u_t\}_{t \geq 0}$ is the control process which belongs to the space of admissible controls $\mathcal{U}$ such that u_t is $\mathcal{F}_t$ -adapted, valued in a separable metric space A . For simplicity, we consider that $A \subseteq \mathbb{R}$. Let $c : (\mathbb{R}^d, A) \rightarrow \mathbb{R}^+$ and $g : \mathbb{R}^d \rightarrow \mathbb{R}$ be measurable functions, and consider the total discounted cost functional

$$J^u(x) = \mathbb{E}_x \left[\int_0^\tau e^{-\gamma s} c(X_s, u_s) ds + e^{-\gamma \tau} g(X_\tau) \right], \quad (4.2)$$

where τ is the first exit time $\tau = \inf_{t \geq 0} \{X_t \notin \Omega\}$. Then the optimal value function

$$V(x) = \inf_u J^u(x) \quad (4.3)$$

is given as the (viscosity) solution of the HJB equation

$$\begin{aligned} \inf_{u \in A} \left\{ \sum_i b_i(x, u) \partial_i V(x) + c(x, u) \right\} + \frac{1}{2} \sum_{i,j} \Phi \Phi^\top_{i,j}(x) \partial_{i,j}^2 V(x) - \gamma V(x) &= 0, \quad x \in \Omega \\ V(x) &= g(x), \quad x \in \partial\Omega. \end{aligned} \quad (4.4)$$

Conventionally, to solve equation (4.4), one first discretizes the operators by finite difference (or finite element) methods. Then the discretized algebraic equations are solved by the Bellman–Howard iteration algorithm (policy iteration) or Newton’s method. These methods suffer from the curse of dimensionality, hence they are incapable of solving high-dimensional HJB equations.

Adapting the idea from Chapters 1 to 3, a natural alternative for solving such HJB PDEs is to introduce neural networks and reformulate the PDE problem as a numerical optimization one. Training a neural network f_θ to solve equation (4.4) via DGM or PINNs, the optimization objective is

$$\begin{aligned} J(\theta) &= \left\| \inf_{u \in \mathcal{U}} \left\{ \sum_i b_i(\cdot, u) \partial_i f_\theta + c(\cdot, u) \right\} + \frac{1}{2} \sum_{i,j} \Phi \Phi^\top_{i,j} \partial_{i,j}^2 f_\theta - \gamma f_\theta \right\|_{L^2(\Omega)}^2 + \\ &+ \left\| f_\theta - g \right\|_{\nu(\partial\Omega)}^2. \end{aligned} \quad (4.5)$$

where ν is a properly chosen norm at the boundary. However, such a formulation may encounter several obstacles. Firstly, performing this optimization task requires computing the gradient term

$$\nabla_\theta \left[\inf_{u \in \mathcal{U}} \left\{ \sum_i b_i(x, u) \partial_i f_\theta(x) + c(x, u) \right\} \right] \quad (4.6)$$

but this is possibly not well-defined, as the infimum is not smooth. At the same time, even if this term is defined properly, the non-linearity of the PDE introduces additional non-convexity and makes computations more costly. As a result, neural networks may be trapped in local minima far away from optimality after hundreds of expensive iteration steps.

However, for a fixed admissible control $u \in \mathcal{U}$, the value function J^u of the corresponding controlled process $\{X_t^u\}$ satisfies a (high-dimensional) linear PDE instead,

which can be much easier to solve:

$$\sum_i b_i(x, u) \partial_i V(x) + \frac{1}{2} \sum_{i,j} \Phi \Phi_{i,j}^\top(x) \partial_{i,j}^2 V(x) + c(x, u) - \gamma V(x) = 0, \quad x \in \Omega \quad (4.7)$$

$$V(x) = g(x), \quad x \in \partial\Omega.$$

This PDE is solvable via applying DGM or PINNs methods, with convergence guarantees given in earlier chapters. Also, one can obtain the solution of the PDE (4.7) directly with Monte Carlo simulations which approximate the reward function J^u in (4.2). Since, for any fixed admissible control u , an approximation of its value function V^u is accessible, two straightforward questions are how to find a better control from V^u and whether, in the end, the optimal control or an ϵ -optimal control can be found.

4.1 Actor-critic methods

Stochastic control problems are closely related to reinforcement learning (RL) [145] problems. In the past decade, deep reinforcement learning has revolutionized the approaches to solving decision-making tasks. As one of the most famous examples, AlphaGo and its variants [137] have dominated the Chinese chess game, Go, and outperformed all human players in this area. It suggests that methods originating from deep RL have the potential for solving high-dimensional decision-making problems upon proper algorithm design and training.

In an RL context, a decision-making problem is typically modeled by a Markov decision process (MDP). One of the most significant differences between control problems and RL problems is that in RL problems, the transition probability function is often unknown prior to training. If the transition probability model is given, finding the optimal control is equivalent to solving a HJB equation.

Actor-critic methods [72, 145] are a branch of iterative algorithms in RL. This class of algorithms performs two tasks iteratively: i) Given a control u , approximate its value function V^u ; ii) Given an approximation of V^u , update control $u \rightarrow u'$ such that u' outperforms u or is closer to optimal control u^* . The actor part refers to the control u , and the critic part assesses the value of the current control. Such a methodology can be transferred to solving HJB PDEs. The solution of the HJB PDE represents the value function of the optimal control. Intuitively, while the actor that represents an admissible control of the control problem is optimized over training iterations, the critic is correspondingly approaching the solution of the PDE we are looking for.

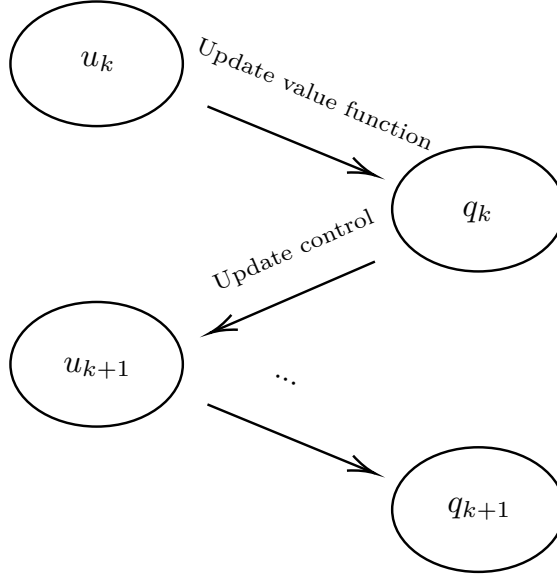


Figure 4.1: A diagram of how actor-critic method updates controls $\{u_k\}$ and value estimates $\{q_k\}$ iteratively from step k to $k + 1$.

Various approaches exist for training the actor and critic components. For instance, in [164], the critic network $\hat{V}$ is trained via a least-square temporal difference (LSTD) method, which makes use of the fact that for (feedback) control function class $u : \Omega \rightarrow A$, value functions similar to J^u in (4.2) satisfy the DPP

$$J^u(X_0) = \int_0^{\tilde{\tau}} e^{-\gamma s} c(X_s, u(X_s)) ds - \int_0^{\tilde{\tau}} e^{-\gamma s} \nabla V(X_s)^\top \Phi(X_s) dW_s + e^{-\gamma \tilde{\tau}} V(X_{\tilde{\tau}}) \quad (4.8)$$

for any $X_0 \in \Omega$ and stopping time $\tilde{\tau} \leq \tau$. The actor is updated via a direct policy gradient approach: set

$$\hat{J}(u) = \mathbb{E}_{X_0 \sim \mu, u} \left[\int_0^{T \wedge \tau} c(X_s, u(X_s)) e^{-\gamma s} ds + \hat{V}(X_{T \wedge \tau}) e^{-\gamma T \wedge \tau} \right] \quad (4.9)$$

and derive

$$\begin{aligned} \frac{\delta \hat{J}}{\delta u} = \mathbb{E}_{X_0 \sim \mu, u} \left[\int_0^{T \wedge \tau} \frac{\delta c(X_s, u(X_s))}{\delta u} e^{-\gamma s} ds + \mathbf{1}_{\{\tau < T\}} c(X_\tau, u(X_\tau)) e^{-\gamma \tau} \frac{\delta \tau}{\delta u} \right. \\ \left. + \nabla \hat{V}(X_{T \wedge \tau}) e^{-\gamma(T \wedge \tau)} \frac{\delta X_{T \wedge \tau}}{\delta u} + \mathbf{1}_{\{\tau < T\}} (\mathcal{L}^u - \gamma) \hat{V}(X_\tau) e^{-\gamma \tau} \frac{\delta \tau}{\delta u} \right], \end{aligned} \quad (4.10)$$

before updating the actor parameters via gradient descent accordingly.

In the next section, we propose a different actor-critic approach based on the policy iteration method in [54]. In [54], the value function of the current control

is derived by solving the corresponding linear PDE boundary value problem with neural networks before the control is updated by taking pointwise minimization or maximization with respect to the Hamiltonian. However, it is not trivial to perform such pointwise operations for neural networks and to represent these results in a practical format. A natural candidate class is neural networks, but they naturally minimize L^2 losses rather than L^∞ losses. Hence, an adaptation to the approach in [54] is of practical significance.

4.2 Single-step update of the control with neural networks

The algorithm proposed in [54] is originally for solving Hamilton–Bellman–Jacobi–Isaacs (HJBI) boundary value problems arising from stochastic games, and it can be similarly formulated to solve our problem (4.4). First, given a fixed admissible control u , a neural network approximator

$$Q_\phi \in C_b^2(\Omega) \quad (4.11)$$

is utilized to approximate the value function V^u which solves the linear PDE (4.7). Second, the pointwise minimization step

$$u^*(x) = \arg \min_u \left\{ \sum_i b_i(x, u) \partial_i Q_\phi(x) + c(x, u) \right\}, \quad x \in \Omega \quad (4.12)$$

generates a new admissible control $u^* \in \mathcal{U}$, and iteratively Q_ϕ is updated to match the value function of u^* via solving its linear PDE. These two steps are alternately applied, yielding a sequence of approximations to the HJB PDE solution and the optimal control. This choice of pointwise update is justified by the fact that if Q_ϕ perfectly matches the value function J^u , then for the pointwise minimizer control

$$\tilde{u}^*(x) = \arg \min_u \left\{ \sum_i b_i(x, u) \partial_i J^u(x) + c(x, u) \right\}, \quad x \in \Omega \quad (4.13)$$

the comparison principle guarantees that

$$J^{\tilde{u}^*}(x) \leq J^u(x), \quad \forall x \in \Omega. \quad (4.14)$$

In other words, the control is improved uniformly. However, as we have pointed out in the last section, in practice, it's impossible to store an infinite table of control values, and hence a rigorous pointwise update is infeasible.

4.2.1 Neural network control

We introduce a single-layer neural network with N hidden units to represent our feedback control $U_\theta^N : \Omega \rightarrow A$

$$U_\theta^N(x) = \frac{1}{N^{\beta_u}} \sum_{i=1}^N h^i \sigma(v^i \cdot x + z^i), \quad (4.15)$$

where $\theta = \{h_i, v_i, z_i\}_{i=1}^N$ is the parameterization of the network, initialized at training time $t = 0$ in the same way as in Assumption 2.1.16 and $\beta_u > 0$ is a scaling factor in $[\frac{1}{2}, 1]$. In the remainder of this section, we further assume that the components in the control problem are bounded:

Assumption 4.2.1. *Functions $b_i, c \in C_b^2(\mathbb{R}^{d+1})$, $\Phi \Phi_{i,j}^\top \in C_b^2(\mathbb{R}^d)$ for all indices i, j .*

One may consider take U_θ^N to minimize an alternative objective, which is already been considered in [2] as follows

$$\min_{\theta} \int_{\Omega} H(U_\theta^N, Q_\phi) d\mu, \quad (4.16)$$

where μ is by default the Lebesgue measure and the Hamiltonian

$$H(u, q)(x) := \mathcal{A}^u q(x) + c(x, u) - \gamma q(x), \quad (4.17)$$

with $\mathcal{A}^u$ being the generator of the controlled process, for $f \in \mathcal{D}(\mathcal{A}^u)$

$$\mathcal{A}^u f(x) := \sum_i b_i(x, u) \partial_i f(x) + \frac{1}{2} \sum_{i,j} \sigma \sigma^\top_{i,j}(x) \partial_{i,j}^2 f(x). \quad (4.18)$$

The authors of [2] has implemented this idea and presented a few good numerical examples with it, while leaving the theoretical analysis of this approach still open. Here we try to fill in their gap by conducting some analysis of this approach in this section. Intuitively, rather than minimizing the Hamiltonian pointwisely for every $x \in \Omega$, we choose to minimize its averaged level over the entire domain. We aim to verify whether (under certain convexity assumptions) the trained control U_θ^N converges to the pointwise minimizer u^* given in (4.12).

Since the critic network Q_ϕ remains fixed while training the control, the gradient of the objective (4.16) with respect to network parameters is

$$\begin{aligned} & \nabla_{\theta} \left[\int_{\Omega} H(U_\theta^N, Q_\phi)(x) dx \right] \\ &= \int_{\Omega} \left(\sum_i \partial_u b_i(x, U_\theta^N(x)) \partial_{x_i} Q_\phi(x) + \partial_u c(x, U_\theta^N(x)) \right) \nabla_{\theta_u} U_\theta^N(x) dx, \end{aligned} \quad (4.19)$$

and we study the training of the neural network control U_θ^N via a continuous-time gradient descent approximation:

$$\frac{d\theta_t}{dt} = -l^N \nabla_\theta \left[\int_\Omega H(U_\theta^N, Q_\phi)(x) dx \right], \quad (4.20)$$

where $l^N = N^{2\beta_u-1}$ is the learning rate. By Assumption 4.2.1, there exists a constant $C > 0$ such that

$$\left| \sum_i \partial_u b_i(x, U_\theta^N(x)) \partial_{x_i} Q_\phi(x) + \partial_u c(x, U_\theta^N(x)) \right| < C, \quad \forall x \in \Omega. \quad (4.21)$$

Hence, from (4.19) and (4.20) one similarly derive that for any fixed training time $t > 0$ there exists a constant $C < \infty$ such that parameters

$$|h^i| < C, \quad \mathbb{E}[|(v^i)_j|] < C, \quad \mathbb{E}[|(z^i)|] < C \quad (4.22)$$

for all indices i, j . Here the expectation is again taken with respect to the random initialization of the neural network parameterization. Similar to previous chapters, by the chain rule, we can derive the evolution of the control over time

$$\frac{dU_{\theta_t}^N}{dt}(y) = - \int_\Omega \left(\sum_i \partial_u b_i(x, U_{\theta_t}^N(x)) \partial_{x_i} Q_\phi(x) + \partial_u c(x, U_{\theta_t}^N(x)) \right) A_t^N(x, y) dx \quad (4.23)$$

where A_t^N is the NTK (similar to (2.26))

$$A_t^N(x, y) := \frac{1}{N} \sum_{k=1}^N \nabla_{h^k, v^k, z^k} [h_t^k \sigma(v_t^k \cdot x + z_t^k)] \cdot \nabla_{h^k, v^k, z^k} [h_t^k \sigma(v_t^k \cdot y + z_t^k)]. \quad (4.24)$$

Again, by the law of large numbers, we have the almost sure convergence

$$\lim_{N \rightarrow \infty} A_0^N(x, y) = A(x, y) \quad (4.25)$$

where $A(x, y)$ is the time-invariant NTK limit given by

$$A(x, y) = \mathbb{E}_{h, v, z} \left[\nabla_{h, v, z} [h \sigma(v \cdot x + z)] \cdot \nabla_{h, v, z} [h \sigma(v \cdot y + z)] \right]. \quad (4.26)$$

4.2.2 Wide limit of the neural network control

As $N \rightarrow \infty$, the evolution of the neural network U_θ^N can be characterized by an infinite-dimensional ODE:

$$\begin{aligned} \frac{dU_t}{dt}(y) &= - \int_\Omega \left(\sum_i \partial_u b_i(x, U_t(x)) \partial_{x_i} Q_\phi(x) + \partial_u c(x, U_t(x)) \right) A(x, y) dx \\ &= - \int_\Omega \partial_u H(U_t, Q_\phi)(x) A(x, y) dx, \end{aligned} \quad (4.27)$$

with $U_0 = 0$. We denote $\sum_i \partial_u b_i(x, U_t(x)) \partial_{x_i} Q_\phi(x) + \partial_u c(x, U_t(x))$ by $\partial_u H(U_t, Q_\phi)(x)$ to simplify our notations. Similar to the convergence to wide-limit ODEs we have established in Chapter 2, (2.3.8), and Chapter 3, (3.14) we have

Lemma 4.2.2. *For all training time $t \geq 0$, the trained control converges to its limit as network width $N \rightarrow \infty$, in the sense that*

$$\mathbb{E} \left[\left\| U_{\theta_t}^N - U_t \right\|_{L^2(\Omega)} \right] \rightarrow 0. \quad (4.28)$$

The expectation is taken with respect to the neural network's random initialization.

Proof. This convergence result is special case of the Theorem 2.3.10. Hence we skip its proof here. $\square$

We now require the following convexity assumption, under which we can finish our convergence derivation.

Assumption 4.2.3. *The Hamiltonian $H(U, Q_\phi)(x)$ is strictly convex and Lipschitz continuous with respect to U for all $x \in \Omega$. And, in particular, there is a unique $U^* : \Omega \rightarrow A$ that minimizes $H(u, Q_\phi)(x)$ for all $x \in \Omega$.*

Proposition 4.2.4. *If a control $\tilde{U}$ satisfies*

$$\partial_u H(\tilde{U}, Q_\phi)(x) = 0 \quad \text{a.s.} \quad (4.29)$$

then we have $\tilde{U} = u^*$ a.s.

Proof. By Assumption 4.2.3, the Hamiltonian is strictly convex. Therefore, its unique stationary point is the minimizer u^* . $\square$

Proposition 4.2.5. *The term $\partial_u H(U, Q_\phi)$ is uniformly bounded in L^∞ for all $U \in \mathcal{U}$. There exists a constant $C > 0$ such that*

$$|\partial_u H(U, Q_\phi)(x)| < C \quad (4.30)$$

for all $x \in \Omega$.

Proof. It is a direct result given by Assumption 4.2.1 and (4.11). $\square$

It is straightforward to show the following stationarity result of limit ODE (4.27).

Lemma 4.2.6. *Under Assumption 4.2.3, the only stationary point of (4.27) is u^* .*

Proof. A stationary point of U_t in (4.27), satisfies $\frac{d}{dt}U_t = 0$ a.s. Therefore, by the fact that kernel A is positive definite, we conclude that $\partial_u H(U_t, Q_\phi) = 0$ a.s. Hence, by Assumption 4.2.3, $U_t = u^*$. $\square$

Now, taking a total derivative of $H(U, Q_\phi)$ with respect to U gives

$$dH(U, Q_\phi)(y) = \partial_u H(U, Q_\phi)(y) dU. \quad (4.31)$$

Hence, we can further obtain

$$\frac{dH(U_t, Q_\phi)}{dt}(y) = -\partial_u H(U_t, Q_\phi)(y) \int_{\Omega} \partial_u H(U_t, Q_\phi)(x) A(x, y) dx. \quad (4.32)$$

Therefore

$$\frac{d}{dt} \int_{\Omega} H(U_t, Q_\phi)(y) dy = - \int_{\Omega} \int_{\Omega} \partial_u H(U_t, Q_\phi)(x) \partial_u H(U_t, Q_\phi)(y) A(x, y) dx dy \leq 0. \quad (4.33)$$

Since the activation function σ is discriminatory, the kernel A is positive definite (see Lemma 2.2.12), and the last equality holds if and only if $\partial_u H(U_t, Q_\phi) = 0$ almost everywhere. Notice that $H(U, Q_\phi)$ is lower-bounded by $H(U, Q_\phi)(y) \geq H(u^*, Q_\phi)(y)$ for all $y \in \Omega$, therefore

$$\int_{\Omega} H(U_t, Q_\phi)(y) dy \geq \int_{\Omega} H(u^*, Q_\phi)(y) dy > -\infty. \quad (4.34)$$

As it is a decaying but lower-bounded functional, from (4.32) we conclude that

$$0 \leq \int_{\Omega} \int_{\Omega} \partial_u H(U_t, Q_\phi)(x) \partial_u H(U_t, Q_\phi)(y) A(x, y) dx dy \rightarrow 0 \quad (4.35)$$

except for $t \in \mathcal{T}$ where $\mathcal{T} \subset [0, +\infty)$ is some set of finite measure. Subsequently, as $t \rightarrow \infty$

$$\|\partial_u H(U_t, Q_\phi)\|_A \rightarrow 0 \quad (4.36)$$

except for $t \in \mathcal{T}$. Here $\|\cdot\|_A$ is the metric induced by kernel A

$$\|h\|_A := \int_{\Omega} h(x) h(y) A(x, y) \mu(dx) \mu(dy). \quad (4.37)$$

Consider the eigenmodes of A , $\{e_i\}_{i \geq 0}$, with eigenvalues $\{\lambda_i\}_{i \geq 0}$ where $\lambda_1 \geq \lambda_2 \geq \dots > 0$ concentrate at 0. We write

$$\partial_u H(U_t, Q_\phi) = \sum_i a_t^i e_i, \quad (4.38)$$

and hence

$$\|\partial_u H(U_t, Q_\phi)\|_A^2 = \sum_i (a_t^i)^2 \lambda_i \rightarrow 0 \quad (4.39)$$

as $t \rightarrow \infty$ except on $\mathcal{T}$. Then by (4.39) we see that

$$\langle e_i, \partial_u H(U_t, Q_\phi) \rangle = a_t^i \rightarrow 0 \quad (4.40)$$

for all i (except on $\mathcal{T}$). Because by spectral theorem, $\{e_i\}$ forms a basis in L^2 , we have

$$\partial_u H(U_t, Q_\phi) \xrightarrow{L^2} 0 \quad (4.41)$$

except on $\mathcal{T}$.

Assumption 4.2.7. *For L^∞ -bounded sequence $\partial_u(f_t, Q_\phi)$, The following A -norm condition*

$$\|\partial_u(f_t, Q_\phi)\|_A \rightarrow 0, \quad t \rightarrow \infty \quad (4.42)$$

implies that there exists a subsequence $\{t_k\}$ such that

$$\|\partial_u(f_{t_k}, Q_\phi)\|_{L^1} \rightarrow 0, \quad k \rightarrow \infty. \quad (4.43)$$

Now, by (4.36) and Assumption 4.2.7 we have $U_t \rightarrow u^*$ in $L^1(\Omega)$ on a subsequence $\{t_k\} \rightarrow \infty$. Combining the results above, we have the following convergence theorem demonstrating that neural network control learns the single-step pointwise minimization control u^* for a given neural network critic Q_ϕ .

Theorem 4.2.8. *For any $\epsilon > 0$, there exists an ϵ -precision neural network control update $U_{t_\epsilon}^{N_\epsilon}$ such that*

$$\mathbb{E} \left[\left\| U_{t_\epsilon}^{N_\epsilon} - u^* \right\|_{L^1(\Omega)} \right] \leq \epsilon \quad (4.44)$$

where the expectation is taken with respect to the random initialization of control U , t_ϵ is the training time needed in the current step, and N_ϵ is the complexity of the network.

Remark 4.2.9. The above theorem shows that u^* can be approximated by training a neural network, with any required precision level, under the convexity assumption. Without Assumption 4.2.3, neural networks are not guaranteed to find the global optimal update u^* but rather may get stuck at local minima where $\partial_u H(U, Q_\phi) = 0$ a.s. This is one of the major weaknesses of our neural network approach.

Remark 4.2.10. Our results so far indicate that neural networks can perform single-step updates of the actor under certain assumptions in a policy iteration setting. Combining these with the convergence results given in [54], in its limit, the policy iteration method based on critic Q_ϕ and actor U_θ converges to the optimal value function which solves the HJB PDE. However, a lack of convergence rate of U_θ to u^* suggests we cannot establish a convergence rate for the entire algorithm. Also, in practice, discretization of gradient descent steps brings in error terms. Our current analysis does not deal with this error, but only an idealization. This is a very interesting problem to analyze in the future, but covered by this thesis.

4.2.3 Practical issues

Similar to the deep Galerkin method's objective (3.9), the optimization objective (4.16) needs discretization before any practical implementation. One can easily apply Monte Carlo sampling in each epoch to approximate the integration

$$\frac{m(\Omega)}{N_m} \sum_{i=1}^{N_m} H(U_\theta^N, Q_\phi)(x_i) \approx \int_{\Omega} H(U_\theta^N, Q_\phi) d\mu. \quad (4.45)$$

In each epoch, N_m random samples are taken in Ω and reshuffled at the end. Another point is that in (4.19), partial derivatives of Q_ϕ are involved in the computation of the gradient term. However, by our proofs in Chapters 2 and 3, we only have $Q_\phi \approx J^u$ in $L^2(\Omega)$, hence our approximation is matching the value function but not its derivatives. In this case, we apply integration by parts (IBPs) to get rid of these partial derivative terms

$$\begin{aligned} & \int_{\Omega} \left(\sum_i \partial_u b_i(x, U_\theta^N(x)) \partial_{x_i} Q_\phi(x) + \partial_u c(x, U_\theta^N(x)) \right) \nabla_\theta U_\theta^N(x) \mu(dx) \\ &= \int_{\partial\Omega} Q_\phi \nabla_\theta U_\theta^N \left[\partial_u b(\cdot, U_\theta^N) \cdot \hat{n} \right] dS - \int_{\Omega} Q_\phi \nabla_x \cdot \left[\partial_u b(\cdot, U_\theta^N) \nabla_\theta U_\theta^N \right] d\mu, \end{aligned} \quad (4.46)$$

where the divergence form above equals

$$\begin{aligned} & \nabla_x \cdot \left[\partial_u b(x, U_\theta^N(x)) \nabla_\theta U_\theta^N(x) \right] \\ &= \sum_i \left(\partial_{u, x_i} b(x, U_\theta^N(x)) \nabla_{\theta_u} U_\theta^N(x) + \partial_{uu} b(x, U_\theta^N(x)) \partial_{x_i} U_\theta^N(x) \nabla_\theta U_\theta^N(x) \right. \\ & \quad \left. + \partial_u b(x, U_\theta^N(x)) \nabla_\theta [\partial_{x_i} U_\theta^N](x) \right). \end{aligned} \quad (4.47)$$

Hence, $\mathcal{H}^1$ approximation of the value function (required in [54]) can be substituted by a L^2 approximation Q_ϕ for the training of our actor U_θ^N .

4.3 Online actor-critic methods

The approach taken in Section 4.2 is often categorized as an ‘offline’ method [145], as the value Q_ϕ remains fixed while training U_θ . In contrast, ‘online’ methods allow both components to be updated simultaneously. For instance, before U_θ converges to the pointwise minimization control u^* in Section 4.2, after a few (or only one or two) epochs, the critic Q_ϕ is re-trained to match the value function of updated control J^{U_θ} . In this section, we model the training process of such online approaches for solving the following HJB PDE.

The idea behind this is that we have shown in previous chapters that critic neural networks can learn the value function for fixed actors by solving corresponding PDEs, and actor networks can learn to update a policy for fixed critics. Both ways are viable given enough long training time, and naturally the further step is to combine these two in an online manner.

Let us consider a generalized case where both the drift and diffusion parts are controlled. We introduce the following controlled process defined on $\{\bar{\Omega}, \mathcal{F}, (\mathcal{F}_t)_{t \geq 0}, \mathbb{P}\}$ with measurable coefficients $b : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$ and $\Phi : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^{d \times d'}$

$$dX_t = b(X_t, u_t)dt + \Phi(X_t, u_t)dW_t, \quad X_0 = x \in \Omega \subset \mathbb{R}^d, \quad (4.48)$$

where the admissible control $u \in \mathcal{U}$ takes value in $\mathbb{R}$. Now we consider the same cost functional J^u as in (4.2). Then the optimal value function $V(x) = \inf_u J^u(x)$ is given as the solution of the HJB equation

$$\inf_{u \in \mathcal{U}} \left\{ \sum_i b_i(x, u) \partial_i V(x) + \frac{1}{2} \sum_{i,j} \Phi \Phi^\top_{i,j}(x, u) \partial_{i,j}^2 V(x) + c(x, u) \right\} - \gamma V(x) = 0, \quad x \in \Omega$$

$$V(x) = g(x), \quad x \in \partial\Omega. \quad (4.49)$$

For notational simplicity, we denote for f in the domain of the operator

$$\mathcal{L}^u f(x) := \sum_i b_i(x, u) \partial_i f(x) + \frac{1}{2} \sum_{i,j} \Phi \Phi^\top_{i,j}(x, u) \partial_{i,j}^2 f(x) - \gamma f(x) + c(x, u). \quad (4.50)$$

Similar to Assumption 4.2.1, in the remainder of this section, we assume that

Assumption 4.3.1. *Functions $b_i, \Phi \Phi^\top_{i,j}, c \in C_b^2(\mathbb{R}^{d+1})$ for all indices i, j .*

Under the boundedness assumption, we immediately have the following Lipschitz properties.

Proposition 4.3.2. *There exists a constant $C > 0$ such that:*

1. For any $x \in \Omega$, any admissible control $u \in \mathcal{U}$, function $f_1, f_2 \in \mathcal{D}(\mathcal{L}^u)$

$$|\partial_u \mathcal{L}^u f_1(x) - \partial_u \mathcal{L}^u f_2(x)| \leq C \sum_{|\alpha| \leq 2} |D_\alpha f_1(x) - D_\alpha f_2(x)|, \quad (4.51)$$

2. For any $x \in \Omega$, any admissible control u_1, u_2 and $f \in \mathcal{D}(\mathcal{L}^{u_1}) \cap \mathcal{D}(\mathcal{L}^{u_2})$

$$|\partial_u \mathcal{L}^{u_1} f(x) - \partial_u \mathcal{L}^{u_2} f(x)| \leq C |u_1(x) - u_2(x)| \sum_{|\alpha| \leq 2} |D_\alpha f(x)|. \quad (4.52)$$

4.3.1 Actor U_θ and critic Q_ϕ

Trying to solve equation (4.49), we first define our neural network actor U_θ^N and neural network critic Q_ϕ . We use the same set of notations to represent our feedback control $U_\theta^N : \Omega \rightarrow \mathbb{R}$

$$U_\theta^N(x) := \frac{1}{N^{\beta_u}} \sum_{i=1}^N h^i \sigma(v^i \cdot x + z^i), \quad (4.53)$$

and the critic network $Q_\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ with N^* units $Q_\phi^{N^*} := S_\phi^{N^*} \cdot \eta + g$, which is generated by a separate neural network

$$S_\phi^{N^*}(x) := \frac{1}{N^{*\beta_q}} \sum_{i=1}^{N^*} c^i \sigma(w^i \cdot x + b^i). \quad (4.54)$$

Here, $\phi = \{c^i, w^i, b^i\}$ parameterizes the critic network, and η is again the auxiliary function (see Chapter 2) that matches the boundary condition. Without loss of generality, we assume that both the actor network and the critic network apply the same activation function σ . In addition, we assume that the configuration of these two networks satisfies the following assumption.

Assumption 4.3.3 (Neural network configuration). *The activation function $\sigma \in C_b^3(\mathbb{R})$ is non-constant. The initialization of the parameters θ, ϕ for all indices satisfies:*

- The parameters $c_0^i, w_0^i, b_0^i, h_0^i, v_0^i, z_0^i$ are independent random variables.
- The random variables c_0^i, h_0^i are bounded, $|c_0^i| < K_0$, $|h_0^i| < K_0$ and $\mathbb{E}[c_0^i] = \mathbb{E}[h_0^i] = 0$.
- The distribution of the random variables w_0^i, b_0^i and v_0^i, z_0^i have full support. That is, for any open set $D \subseteq \mathbb{R}^{n+1}$,

$$\mathbb{P}((w_0^i, b_0^i) \in D) > 0, \quad \mathbb{P}((v_0^i, z_0^i) \in D) > 0. \quad (4.55)$$

- For all indices i , we have the moment bounds $\mathbb{E}[\|w_0^i\|_2^3] < \infty$, $\mathbb{E}[\|v_0^i\|_2^3] < \infty$, $\mathbb{E}[\|b_0^i\|] < \infty$, $\mathbb{E}[\|z_0^i\|] < \infty$.

In a discrete-time setting, the training of the actor and critic networks goes iteratively. Given the current actor U_θ , the critic Q_ϕ aims to learn the value function J^{U_θ} via learning to minimize the PDE loss

$$\int_{\Omega} |\mathcal{L}^{U_\theta} Q_\phi|^2 d\mu \quad (4.56)$$

with DGM or Q-PDE algorithm steps, or the regular squared loss

$$\int_{\Omega} |Q_\phi - J^{U_\theta}|^2 d\mu \quad (4.57)$$

when Monte Carlo simulation for J^{U_θ} is available. And for fixed critic Q_ϕ , the actor minimizes the objective in (4.16). A pseudo-code description of the online actor-critic algorithm is given below:

Algorithm 2: Online Actor-Critic Algorithm

Parameters: Hyper-parameters of actor and critic networks; Domain Ω ; SDE coefficients b, Φ ; Cost function c , Boundary values g ; Sampling law μ ; Stopping criteria ϵ ; Number of iterations in each training step k ;
Initialize: Neural net S^{N^*} and U^N ; Auxiliary function η ; Learning rate scheduler; Current time $t = 0$;

Iteratively execute the following two steps until stopping criteria are met:

Critic steps: Fix actor u . Do k iterations of either:

- Minimizing the associated linear PDE loss (4.56) or
- Simulating the value function J^U (and its partial derivatives if possible) and learning the simulated value function via minimizing (4.57).

Update value function $Q \leftarrow Q'$;

Actor steps: Fix critic Q . Do k iterations of minimizing objective (4.16) with an optimizer (GD/SGD/ADAM).

Update actor $U \leftarrow U'$;

4.3.2 Continuous-time online actor-critic method

We examine a continuum analogy of the online actor-critic algorithm in the rest of this chapter. We assume that the actor and the critic are trained in continuous time and both of the networks are simultaneously updated following a gradient descent

approach:

$$\begin{aligned}\frac{d\phi_t}{dt} &= -l_q^{t,N} G_q(\theta_t, \phi_t), \\ \frac{d\theta_t^u}{dt} &= -l_u^{t,N} \tilde{G}_u(\theta_t, \phi_t).\end{aligned}\tag{4.58}$$

In (4.58), l_q and l_u are the learning rates for the critic network and the actor network respectively which we will specify shortly. We have the gradient of (4.16) G_u :

$$\begin{aligned}G_u(\theta_t, \phi_t) &= \nabla_\theta \left[\int_\Omega H(U_{\theta_t}^N, Q_{\phi_t}^{N*}) d\mu \right] \\ &= \int_\Omega \left(\sum_i \partial_u b_i(x, U_{\theta_t}^N(x)) \partial_i Q_{\phi_t}^{N*}(x) + \frac{1}{2} \sum_{i,j} \partial_u \Phi \Phi_{ij}^I(x, U_{\theta_t}(x)) \partial_{ij} Q_{\phi_t}^{N*}(x) \right. \\ &\quad \left. + \partial_u c(x, U_{\theta_t}^N(x)) \right) \nabla_\theta U_{\theta_t}^N(x) \mu(dx).\end{aligned}\tag{4.59}$$

which we stabilize to give the quantity $\tilde{G}_u$ appearing in (4.58),

$$\begin{aligned}\tilde{G}_u(\theta_t, \phi_t) &= \int_\Omega \psi^N \left(\sum_i \partial_u b_i(x, U_{\theta_t}^N(x)) \partial_i Q_{\phi_t}^{N*}(x) + \frac{1}{2} \sum_{i,j} \partial_u \Phi \Phi_{ij}^I(x, U_{\theta_t}(x)) \partial_{ij} Q_{\phi_t}^{N*}(x) \right. \\ &\quad \left. + \partial_u c(x, U_{\theta_t}^N(x)) \right) \nabla_\theta U_{\theta_t}^N(x) \mu(dx).\end{aligned}\tag{4.60}$$

In (4.60), ψ is a truncation function as introduced in Chapters 2 and 3, see Definition 2.1.17. Returning to (4.58), G_q is the gradient of the critic network's optimization objective of the algorithm user's choice. If the critic is trained by learning J^U directly via Monte Carlo then

$$G_q(\theta_t, \phi_t) = \nabla_\phi \left[\int_\Omega \psi^{N*} (Q_{\phi_t}^{N*} - J^{U_{\theta_t}})^2 d\mu \right] = \int_\Omega F^{N*} (Q_{\phi_t}^N - J^{U_{\theta_t}}) \nabla_\phi Q_{\phi_t}^{N*} d\mu, \tag{4.61}$$

where $F = \psi \cdot \psi'$ (see also Definition 2.1.17). If the critic is trained by minimizing the PDE loss via the deep Galerkin method, then similar to the setting in equation (3.12) in Chapter 3, the biased gradient term is given by

$$G_q(\theta_t, \phi_t) = \int_\Omega \psi^{N*} (\mathcal{L}^{U_{\theta_t}^N} Q_{\phi_t}^N) \psi^{N*} \left(\nabla_\phi [\mathcal{L}^{U_{\theta_t}^N} Q_{\phi_t}^N] \right) d\mu, \tag{4.62}$$

where with a slight abuse of notation we let $\psi^k(\mathbf{f}) = (\psi^k(f_1), \dots, \psi^k(f_d))$ for $\mathbf{f} = (f_1, \dots, f_d) \in \mathbb{R}^d$. If the critic is trained by the Q-PDE algorithm, which is reasonable when the PDE family $\mathcal{L}^u$ is monotone, then similar to (2.15) the gradient term is

$$G_q(\theta_t, \phi_t) = \int_\Omega F^{N*} (\mathcal{L}^{U_{\theta_t}^N} Q_{\phi_t}^N) \nabla_\phi (-Q_{\phi_t}^N) d\mu. \tag{4.63}$$

To form a neural tangent kernel, learning rates l_q and l_u in (4.58) are given by $l_q = N^{2\beta_q-1}$ and $l_u = N^{2\beta_u-1}\alpha_t$.

Remark 4.3.4. The choice of learning rate $\alpha_t > 0$ is quite flexible. By introducing $\alpha_t = \frac{1}{1+t}$ in the actor's learning rate, we aim to slow down the evolution of the actor which helps the critic learn the value function of the moving actor. Other choices are also viable. In the remainder of the chapter, we prove convergence results for $\alpha_t = \frac{1}{1+t}$, however, such convergence results also hold for other learning rate schedulers.

By the chain rule, we derive the evolution of U_θ and Q_ϕ over time, given by

$$\frac{dU_{\theta_t}^N}{dt}(y) = -\frac{1}{1+t} \int_{\Omega} \psi^N(\partial_u H(U_{\theta_t}^N, Q_{\phi_t}^N)(x)) \left(N^{2\beta_u-1} \nabla_{\theta} U_{\theta_t}^N(x) \cdot \nabla_{\theta} U_{\theta_t}^N(y)^{\top} \right) \mu(dx) \quad (4.64)$$

and

$$\begin{aligned} \frac{dQ_{\phi_t}^N(y)}{dt} &= \frac{d\theta_q^t}{dt} \cdot \nabla_{\phi} Q_{\phi_t}^N(y)^{\top} \\ &= \int_{\Omega} F^{N*}(\mathcal{L}^{U_{\theta_t}^N} Q_{\phi_t}^N(x)) \left(N^{2\beta_q-1} \nabla_{\phi} Q_{\phi_t}^N(x) \cdot \nabla_{\phi} Q_{\phi_t}^N(y)^{\top} \right) \mu(dx). \end{aligned} \quad (4.65)$$

for all $y \in \Omega$. Equation (4.65) is derived for the case where the critic is updated by following (4.63). If the critic is trained via (4.61) or (4.62), the dynamics can also be derived accordingly.

4.3.2.1 Wide limit of the actor-critic pair

Again, we show that, as $N, N^* \rightarrow \infty$, the dynamics of U_θ and Q_ϕ converge to infinite-dimensional ODEs. Informally, when the critic is trained with (4.63), the wide-limit ODE we wish to introduce is

$$\begin{aligned} \frac{dQ_t}{dt} &= \mathcal{B} \mathcal{L}^{U_t} Q_t, & Q_0 &= 0, \\ \frac{dU_t}{dt} &= -\frac{\mathcal{K} \partial_u H(U_t, Q_t)}{1+t}, & U_0 &= 0, \end{aligned} \quad (4.66)$$

where $\mathcal{B}$ (very similar to that in previous chapters) is the integral operator defined by

$$(\mathcal{B}f)(y) := \int_{\Omega} f(x) B(x, y) \mu(dx) \quad (4.67)$$

for $f \in \mathcal{D}(\mathcal{B})$. Here $B(x, y)$ is defined as $B(x, y) := \eta(x)\eta(y)\bar{A}(x, y)$ where

$$\bar{A}(x, y) := \mathbb{E}_{c,w,b} \left[\nabla_{c,w,b}(c\sigma(w \cdot x + b)) \cdot \nabla_{c,w,b}(c\sigma(w \cdot y + b)) \right] \quad (4.68)$$

is the pointwise limit of the NTK for the component network S_t^N of the critic network Q_{ϕ_t}

$$\begin{aligned}\bar{A}_t^N(x, y) &= \left(\nabla_{\phi} S_t^N(x) \right)^{\top} \left(\nabla_{\phi} S_t^N(y) \right) \\ &= \frac{1}{N} \sum_{i=1}^N \left[\sigma(w_t^i \cdot x + b_t^i) \sigma(w_t^i \cdot y + b_t^i) + c_t^{i2} \sigma'(w_t^i \cdot x + b_t^i) \sigma'(w_t^i \cdot y + b_t^i) (x \cdot y + 1) \right].\end{aligned}\tag{4.69}$$

We also denote

$$B_t^N(x, y) = \eta(x) \eta(y) \bar{A}_t^N(x, y).\tag{4.70}$$

$\mathcal{K}$ is the Hilbert–Schmidt integral operator

$$(\mathcal{K}f)(y) := \int_{\Omega} A(x, y) f(x) \mu(dx)\tag{4.71}$$

for $f \in L^2(\Omega)$ where $A(x, y)$ is given in (4.26). We have the following assumption and subsequent convergence lemma for the actor-critic pair.

Assumption 4.3.5 (Existence of a unique solution). *The wide limit ODE (4.66) taking values in $\mathcal{H}^2(\Omega) \times L^2(\Omega)$ admits a unique solution $\{(Q_t, U_t)\}$ for all $t \geq 0$.*

Lemma 4.3.6 (Wide limit of the actor-critic pair). *The approximator $Q_{\phi_t}^{N^*}$ and actor $U_{\theta_t}^N$ in (4.64) and (4.65) converge to their limit (4.66) as $N, N^* \rightarrow \infty$ for any $t \geq 0$, in the sense that:*

$$\lim_{N^* \rightarrow \infty} \mathbb{E} \left[\|Q_{\phi_t}^{N^*} - Q_t\|_{\mathcal{H}^2}^2 \right] = 0, \quad \lim_{N \rightarrow \infty} \mathbb{E} \left[\|U_{\theta_t}^N - U_t\|_{L^2}^2 \right] = 0.\tag{4.72}$$

We present the rigorous (if rather tedious) proof of Lemma 4.3.6 at the end of this chapter. If the critic network follows (4.61) or (4.62), then we have different wide limit ODEs respectively. For (4.61), its limit ODE pair is

$$\begin{aligned}\frac{dQ_t}{dt} &= -\tilde{\mathcal{K}} \mathcal{L}^{U_t} Q_t, & Q_0 &= 0, \\ \frac{dU_t}{dt} &= -\frac{\mathcal{K} \partial_u H(U_t, Q_t)}{1 + t}, & U_0 &= 0,\end{aligned}\tag{4.73}$$

where $\tilde{\mathcal{K}}$ is also a Hilbert–Schmidt integral operator

$$(\tilde{\mathcal{K}}f)(y) := \int_{\Omega} \tilde{A}(x, y) f(x) \mu(dx)\tag{4.74}$$

for $f \in L^2(\Omega)$ where

$$\tilde{A}(x, y) = \mathbb{E}_{c, w, b} \left[\nabla_{c, w, b} [c\sigma(w \cdot x + b)] \cdot \nabla_{c, w, b} [c\sigma(w \cdot y + b)] \right]. \quad (4.75)$$

For (4.62), its wide limit ODE pair is given by

$$\begin{aligned} \frac{dQ_t}{dt} &= -\mathcal{M}_t \mathcal{L}^{U_t} Q_t, & Q_0 &= 0, \\ \frac{dU_t}{dt} &= -\frac{\mathcal{K} \partial_u H(U_t, Q_t)}{1+t}, & U_0 &= 0, \end{aligned} \quad (4.76)$$

where

$$(\mathcal{M}_t f)(y) := \int_{\Omega} f(x) M_t(x, y) \mu(dx) \quad (4.77)$$

and

$$M_t(x, y) := \mathbb{E}_{c, w, b} \left[\nabla_{c, w, b} \left((\mathcal{A}^{U_t} - \gamma) [\eta(x) c\sigma(x; w, b)] \right) \cdot \nabla_{c, w, b} [\eta(y) c\sigma(y; w, b)] \right]. \quad (4.78)$$

Lemma 4.3.6 also holds for the convergence from pre-limit dynamics (4.61) and (4.62) to their wide limit ODEs (4.73) and (4.76). Being very similar to the current proof, we skip the proof of these two cases for readability reasons.

4.3.2.2 Open problems

The global convergence of the wide limit ODEs in these three cases above remains an open problem under our studies. Not many tools are available for these infinite-dimensional ODEs taking value in Hilbert spaces. Also, both the functions and the optimization objectives change over time and affect each other nonlinearly. Typically, in the case where the critic is trained by DGM, the kernel in (4.76) also changes over time and we may lose track of how its eigenvalue and eigenfunctions change accordingly. There is no guarantee that they change continuously as the operator has an infinite set of them. In addition, the actor and the critic are trained to minimize different objectives through gradient descent, and hence it is not easy to establish an estimate to show that the system is dissipative for certain energy forms. Nevertheless, we at least have the following stationary point theorem which indicates that the only stationary point of the limit ODE system solves the PDE we wish to solve in the first place.

Theorem 4.3.7 (Stationary point solves the PDE). *Suppose that HJB equation (4.49) admits a classical solution, and function pair $(\tilde{Q}, \tilde{U})$ is a fixed point of the ODE (4.66). Then $\tilde{U}$ minimizes the Hamiltonian H and $\tilde{Q}$ is the optimal value function for the control problem.*

Proof. Since $\mathcal{B}\mathcal{L}^{\tilde{U}}\tilde{Q} = 0$ almost everywhere, by the discriminatory property of kernel B , we have $\mathcal{L}^{\tilde{U}}\tilde{Q} = 0$ while $\tilde{Q}$ satisfies the boundary condition. Therefore, $\tilde{Q}$ is the value function of the controlled process. Now, since $H(Q, U)$ is globally convex with respect to U , $\partial_u H = 0$ implied that $\tilde{U}$ is the global minimizer of H . Now, by the verification theorem, we conclude that $\tilde{U}$ is the optimal control and $\tilde{Q}$ is its corresponding value function. $\square$

4.3.3 Appendix: proof of Lemma 4.3.6

In our proofs, we first fix an arbitrary training time bound $T > 0$. Constants K, C may vary from line to line. Without loss of generality, we also assume that $N = N^*$. We introduce the following lemmas before we present our main proof.

Lemma 4.3.8 (Stability). *Following the Actor-Critic PDE scheme, there exists a constant C such that for $0 \leq t \leq T$, critic network parameters satisfy*

$$|c_t^i| < C, \quad \mathbb{E}|w_t^i| < C, \quad \mathbb{E}|b_t^i| < C, \quad (4.79)$$

and actor network parameters satisfy

$$|h_t^i| < C, \quad \mathbb{E}|v_t^i| < C, \quad \mathbb{E}|z_t^i| < C. \quad (4.80)$$

Proof. The proof for Lemma 4.3.8 is a repetition of that for Lemma 2.2.1 and Lemma 3.3.5. Hence we skip its details here. $\square$

Lemma 4.3.9 (Uniform bound on Q_t). *There exists a constant $C > 0$ such that for any $y \in \Omega$, $0 \leq t \leq T$ we have*

$$\sum_{\alpha} |D_{\alpha} Q_t(y)| \leq C. \quad (4.81)$$

Proof. Since

$$Q_t(y) = \int_0^t \int_{\Omega} \mathcal{L}^{U_s} Q_s(x) B(x, y) \mu(dx) ds, \quad (4.82)$$

By the boundedness of kernel B and its partial derivatives, taking partial derivatives give

$$\begin{aligned}
\sum_{\alpha} |D_{\alpha} Q_t(y)| &\leq \int_0^t \int_{\Omega} |\mathcal{L}^{U_s} Q_s(x)| \left| \sum_{\alpha} D_{\alpha}^y B(x, y) \right| \mu(dx) ds \\
&\leq C \int_0^t \int_{\Omega} |\mathcal{L}^{U_s} Q_s(x) - \mathcal{L}^{U_s} \mathbf{0}(x)| + |\mathcal{L}^{U_s} \mathbf{0}(x)| \mu(dx) ds \\
&\leq C \int_0^t \int_{\Omega} \sum_{\alpha} |D_{\alpha} Q_t(x)| \mu(dx) ds + C \int_0^t \int_{\Omega} |c_{U_s}(x)| \mu(dx) ds \\
&\leq C \int_0^t \int_{\Omega} \sum_{\alpha} |D_{\alpha} Q_t(x)| \mu(dx) ds + K.
\end{aligned} \tag{4.83}$$

Applying Gronwall's inequality finishes the proof. $\square$

Lemma 4.3.10. *We define*

$$V_0^N(y) := |U_0^N(y) - U_0(y)| + \sum_{|\alpha| \leq 2} |D_{\alpha}(Q_0^N - Q_0)(y)| = |U_0^N(y)| + \sum_{|\alpha| \leq 2} |D_{\alpha}(Q_0^N)(y)|. \tag{4.84}$$

Then, as $N \rightarrow \infty$ we have

$$\mathbb{E} \left[\left(\int_{\Omega} V_0^N(y)^2 \mu(dy) \right)^{\frac{1}{2}} \right] \rightarrow 0. \tag{4.85}$$

Proof. By Jensen's inequality, it suffices to show that

$$\int_{\Omega} \mathbb{E} \left[V_0^N(y)^2 \right] \mu(dy) \rightarrow 0. \tag{4.86}$$

Notice that

$$\mathbb{E}[U_0^N(y)] = 0, \quad \mathbb{E}[D_{\alpha} Q_0^N(y)] = 0. \tag{4.87}$$

for arbitrary α . Then

$$\mathbb{E}[U_0^N(y)^2] = \text{Var}[U_0^N(y)] = N^{1-2\beta_u} \text{Var}_{h,v,z}[h\sigma(vx + z)] < CN^{1-2\beta_u} \tag{4.88}$$

and

$$\mathbb{E}[D_{\alpha} Q_0^N(y)^2] = \text{Var}[D_{\alpha} Q_0^N(y)] = N^{1-2\beta_q} \text{Var}_{c,w,b}[c\sigma(wx + b)] < CN^{1-2\beta_q}. \tag{4.89}$$

Therefore

$$\begin{aligned}
\int_{\Omega} \mathbb{E} \left[V_0^N(y)^2 \right] \mu(dy) &\leq \int_{\Omega} C(\mathbb{E}[U_0^N(y)^2] + \sum_{|\alpha| \leq 2} \mathbb{E}[D_{\alpha} Q_0^N(y)^2]) \mu(dy) \\
&\leq C(N^{1-2\beta_u} + N^{1-2\beta_q}) \rightarrow 0.
\end{aligned} \tag{4.90}$$

$\square$

Without loss of generality, on the top of Assumption 4.3.3, we further assume that the parameters (c, w, b) and (h, v, z) are sampled under the same joint distribution. In this case, for each $(x, y) \in \Omega$, by the strong law of large numbers these two kernels converge almost surely to the same deterministic kernel $A(x, y)$ in (4.26):

$$\lim_{N \rightarrow \infty} A_0^N(x, y) = \lim_{N \rightarrow \infty} \bar{A}_0^N(x, y) = A(x, y), \quad a.s. \quad (4.91)$$

Lemma 4.3.11. *As $N \rightarrow \infty$ we have*

$$\mathbb{E} \left[\int_0^T \left(\int_{\Omega^2} N^{2\delta} |A_s^N(x, y) - A(x, y)|^2 \mu(dx) \mu(dy) ds \right)^{\frac{1}{2}} ds \right] \rightarrow 0. \quad (4.92)$$

Proof. By Fubini's theorem and Jensen's inequality, it suffices to prove that

$$\int_0^T \left(\int_{\Omega^2} N^{2\delta} \mathbb{E}[|A_s^N(x, y) - A(x, y)|^2] \mu(dx) \mu(dy) \right)^{\frac{1}{2}} ds \rightarrow 0. \quad (4.93)$$

First, we notice that by the mean value theorem and the fact that η , σ , and their derivatives are up to polynomial growth

$$|A_0(x, y) - A_u^N(x, y)| \leq \frac{C}{N} \sum_{i=1}^N \left((\|h_u^i - h_0^i\| + \|v_u^i - v_0^i\| + \|z_u^i - z_0^i\|) f(h_0^i, v_0^i, z_0^i, x, y) \right) \quad (4.94)$$

where f is a polynomial up to the second order. We notice that for time $0 \leq u \leq T$, $\|h_u^i - h_0^i\| + \|v_u^i - v_0^i\| + \|z_u^i - z_0^i\| \leq uCN^{\delta+\beta-1}$. Next, we have

$$\mathbb{E}|A_0^N(x, y) - A(x, y)|^2 = \mathbb{E}|A_0^N(x, y) - \mathbb{E}[A_0^N(x, y)]|^2 = \text{Var}[A_0^N(x, y)] < \frac{C}{N}. \quad (4.95)$$

Combining the inequalities above

$$N^\delta \left(\int_{\Omega^2} \mathbb{E}|A_u^N(x, y) - A(x, y)|^2 \mu(dx) \mu(dy) \right)^{\frac{1}{2}} \leq CuN^{2\delta+\beta-1} + CN^{\delta-\frac{1}{2}}. \quad (4.96)$$

Integrating with respect to time variable u and letting N go to infinity, we conclude the proof. $\square$

Lemma 4.3.12. *For B in (4.67) and B^N in (4.70), we denote*

$$S_s^N(x, y) = \sum_{|\alpha| \leq 2} |D_\alpha^y B(x, y) - D_\alpha^y B_s^N(x, y)|. \quad (4.97)$$

Then, as $N \rightarrow \infty$ we have

$$\mathbb{E} \left[\int_0^T \left(\int_{\Omega^2} N^{2\delta} S_s^N(x, y)^2 \mu(dx) \mu(dy) \right)^{\frac{1}{2}} ds \right] \rightarrow 0. \quad (4.98)$$

Proof. Similarly, by Jensen's inequality, it suffices to show that

$$\int_0^T \left(\int_{\Omega^2} \mathbb{E} \left[N^{2\delta} S_s^N(x, y)^2 \right] \mu(dx) \mu(dy) \right)^{\frac{1}{2}} ds \quad (4.99)$$

Since for $S_s^N(x, y)$ we have

$$S_s^N(x, y) = \sum_{|\alpha| \leq 2} |D_\alpha^y B(x, y) - D_\alpha^y B_s^N(x, y)|, \quad (4.100)$$

it suffices to show that for arbitrary α

$$N^\delta \int_0^t \left(\int_{\Omega^2} \mathbb{E} |D_\alpha^y B(x, y) - D_\alpha^y B_u^N(x, y)|^2 \mu(dx) \mu(dy) \right)^{\frac{1}{2}} du \rightarrow 0. \quad (4.101)$$

First, we notice that by the mean value theorem and the fact that η , σ , and their derivatives are up to polynomial growth

$$\begin{aligned} & |D_\alpha^y B_0(x, y) - D_\alpha^y B_u^N(x, y)| \\ & \leq \frac{C}{N} \sum_{i=1}^N \left((\|w_u^i - w_0^i\| + \|b_u^i - b_0^i\| + \|c_u^i - c_0^i\|) f(w_0^i, c_0^i, b_0^i, x, y) \right) \end{aligned} \quad (4.102)$$

where f is a polynomial up to $|\alpha|$ -th order. We notice that $\|w_u^i - w_0^i\| + \|b_u^i - b_0^i\| + \|c_u^i - c_0^i\| \leq uN^{\delta+\beta-1}$. Therefore

$$\left(\int_{\Omega^2} \mathbb{E} |D_\alpha^y B_0^N(x, y) - D_\alpha^y B_u^N(x, y)|^2 \mu(dx) \mu(dy) \right)^{\frac{1}{2}} \leq CuN^{\delta+\beta-1}. \quad (4.103)$$

Next, we have

$$\begin{aligned} \mathbb{E} |D_\alpha^y B_0^N(x, y) - D_\alpha^y B(x, y)|^2 &= \mathbb{E} |D_\alpha^y B_0^N(x, y) - \mathbb{E}[D_\alpha^y B_0^N(x, y)]|^2 = \text{Var}[D_\alpha^y B_0^N(x, y)] \\ &< \frac{C}{N}. \end{aligned} \quad (4.104)$$

Combining the inequalities above

$$N^\delta \left(\int_{\Omega^2} \mathbb{E} |D_\alpha^y B(x, y) - D_\alpha^y B_u^N(x, y)|^2 \mu(dx) \mu(dy) \right)^{\frac{1}{2}} \leq CuN^{2\delta+\beta-1} + CN^{\delta-\frac{1}{2}} \rightarrow 0. \quad (4.105)$$

Integrating with respect to time t we conclude the proof. $\square$

Proof of Lemma 4.3.6. With slight abuse of notations, we denote $Q_t^N = Q_{\phi_t}^N$ and $U_t^N = U_{\theta_t}^N$. By integrating the dynamics of $\frac{dQ_t^N}{dt}$ with respect to time, we have

$$Q_t^N(y) = Q_0^N(y) + \int_0^t \int_{\Omega} F^N(\mathcal{L}^{U_s^N} Q_s^N(x)) B_s^N(x, y) \mu(dx) ds \quad (4.106)$$

and

$$Q_t(y) = Q_0(y) + \int_0^t \int_{\Omega} \mathcal{L}^{U_s} Q_s(x) B(x, y) \mu(dx) ds. \quad (4.107)$$

Subtracting the second equation from the first and taking α -th order derivatives with respect to y gives

$$\begin{aligned} D_{\alpha}(Q_t^N - Q_t)(y) &= D_{\alpha}(Q_0^N - Q_0)(y) + \int_0^t \int_{\Omega} F^N(\mathcal{L}^{U_s^N} Q_s^N(x)) D_{\alpha}^y B_s^N(x, y) \\ &\quad - \mathcal{L}^{U_s} Q_s(x) D_{\alpha}^y B(x, y) \mu(dx) ds. \end{aligned} \quad (4.108)$$

We are splitting the integral on the right-hand side into four terms and bound each of them separately.

$$\begin{aligned} &|D_{\alpha}(Q_t^N - Q_t)(y)| \\ &\leq |D_{\alpha}(Q_0^N - Q_0)(y)| + \int_0^t \int_{\Omega} |F^N(\mathcal{L}^{U_s^N} Q_s^N(x))| |D_{\alpha}^y B_s^N(x, y) - D_{\alpha}^y B_s(x, y)| \mu(dx) ds \\ &\quad + \int_0^t \int_{\Omega} |F^N(\mathcal{L}^{U_s^N} Q_s^N(x)) - F^N(\mathcal{L}^{U_s^N} Q_s(x))| |D_{\alpha}^y B(x, y)| \mu(dx) ds \\ &\quad + \int_0^t \int_{\Omega} |F^N(\mathcal{L}^{U_s^N} Q_s(x)) - F^N(\mathcal{L}^{U_s} Q_s(x))| |D_{\alpha}^y B(x, y)| \mu(dx) ds \\ &\quad + \int_0^t \int_{\Omega} |F^N(\mathcal{L}^{U_s} Q_s^N(x)) - \mathcal{L}^{U_s^N} Q_s(x)| |D_{\alpha}^y B(x, y)| \mu(dx) ds. \end{aligned} \quad (4.109)$$

Therefore, by the fact that $|F^N(x)| \leq N^{\delta}$ and that there exists a constant C such that $|D_{\alpha}^y B(x, y)| \leq C$ we have

$$\begin{aligned} &|D_{\alpha}(Q_t^N - Q_t)(y)| \\ &\leq |D_{\alpha}(Q_0^N - Q_0)(y)| + \int_0^t \int_{\Omega} N^{\delta} |D_{\alpha}^y B_s^N(x, y) - D_{\alpha}^y B_s(x, y)| \mu(dx) ds \\ &\quad + C \int_0^t \int_{\Omega} |F^N(\mathcal{L}^{U_s^N} Q_s^N(x)) - F^N(\mathcal{L}^{U_s^N} Q_s(x))| \mu(dx) ds \\ &\quad + C \int_0^t \int_{\Omega} |U_s^N(x) - U_s(x)| \sum_{\alpha} |D_{\alpha} Q_s(x)| \mu(dx) ds \\ &\quad + \int_0^t \int_{\Omega} |\mathcal{L}^{U_s} Q_s(x)| \mathbf{1}_{\{|\mathcal{L}^{U_s} Q_s| \geq N^{\delta}\}}(x) \mu(dx) ds \\ &\leq |D_{\alpha}(Q_0^N - Q_0)(y)| + \int_0^t \int_{\Omega} N^{\delta} |D_{\alpha}^y B_s^N(x, y) - D_{\alpha}^y B_s(x, y)| \mu(dx) ds \\ &\quad + C \int_0^t \int_{\Omega} \sum_{|\alpha| \leq 2} |D_{\alpha}(Q_s^N - Q_s)(x)| + |U_s^N(x) - U_s(x)| \mu(dx) ds \\ &\quad + \int_0^t \int_{\Omega} |\mathcal{L}^{U_s} Q_s(x)| \mathbf{1}_{\{|\mathcal{L}^{U_s} Q_s| \geq N^{\delta}\}}(x) \mu(dx) ds. \end{aligned} \quad (4.110)$$

Similarly, for actor process U_t^N we have

$$U_t^N(y) = U_0^N(y) - \int_0^t \frac{1}{1+s} \int_{\Omega} \psi^N(\partial_u H(U_s^N, Q_s^N)(x)) \bar{A}_s^N(x, y) \mu(dx) ds, \quad (4.111)$$

$$U_t(y) = U_0(y) - \int_0^t \frac{1}{1+s} \int_{\Omega} \partial_u H(U_s, Q_s)(x) A_s(x, y) \mu(dx) ds. \quad (4.112)$$

Subtracting (4.112) from (4.110), splitting the integral term on the right-hand side, and taking absolute value give

$$\begin{aligned} & |U_t^N(y) - U_t(y)| \\ & \leq |U_0^N(y) - U_0(y)| + \int_0^t \frac{1}{1+s} \int_{\Omega} |\psi^N(\partial_u H(U_s^N, Q_s^N)(x))| |\bar{A}_s^N(x, y) - A(x, y)| \mu(dx) ds \\ & \quad + \int_0^t \frac{1}{1+s} \int_{\Omega} |\psi^N(\partial_u H(U_s^N, Q_s^N)(x)) - \psi^N(\partial_u H(U_s, Q_s)(x))| |A(x, y)| \mu(dx) ds \\ & \quad + \int_0^t \frac{1}{1+s} \int_{\Omega} |\psi^N(\partial_u H(U_s, Q_s)(x)) - \partial_u H(U_s, Q_s)(x)| |A(x, y)| \mu(dx) ds \\ & \leq |U_0^N(y) - U_0(y)| + \int_0^t \frac{N^\delta}{1+s} \int_{\Omega} |\bar{A}_s^N(x, y) - A(x, y)| \mu(dx) ds \\ & \quad + \int_0^t \frac{1}{1+s} \int_{\Omega} |\partial_u H(U_s^N, Q_s^N)(x) - \partial_u H(U_s, Q_s)(x)| |A(x, y)| \mu(dx) ds \\ & \quad + \int_0^t \frac{1}{1+s} \int_{\Omega} |\partial_u H(U_s, Q_s)(x)| \mathbf{1}_{\{|\partial_u H(U_s, Q_s)| \geq N^\delta\}}(x) |A(x, y)| \mu(dx) ds. \end{aligned} \quad (4.113)$$

Notice that by Assumption 3.2.1 and Assumption 3.2.2

$$\begin{aligned} & |\partial_u H(U_s^N, Q_s^N)(x) - \partial_u H(U_s, Q_s)(x)| \\ & \leq |\partial_u H(U_s^N, Q_s^N)(x) - \partial_u H(U_s^N, Q_s)(x)| + |\partial_u H(U_s^N, Q_s)(x) - \partial_u H(U_s, Q_s)(x)| \\ & \leq C \sum_{|\alpha| \leq 2} |D_\alpha(Q_s^N - Q_s)(x)| + C |U_s^N(x) - U_s(x)| \sum_{|\alpha| \leq 2} |D_\alpha Q_s(x)|. \end{aligned} \quad (4.114)$$

Now by the fact that $|A(x, y)|$ and $\sum_{|\alpha| \leq 2} |D_\alpha Q_s(x)|$ are uniformly bounded, and that $\frac{1}{1+t^d} < 1$, from (4.113) we derive

$$\begin{aligned} |U_t^N(y) - U_t(y)| & \leq |U_0^N(y) - U_0(y)| + \int_0^t \int_{\Omega} N^\delta |\bar{A}_s^N(x, y) - A(x, y)| \mu(dx) ds \\ & \quad + C \int_0^t \int_{\Omega} \sum_{|\alpha| \leq 2} |D_\alpha(Q_s^N - Q_s)(x)| + |U_s^N(x) - U_s(x)| \mu(dx) ds \\ & \quad + C \int_0^t \int_{\Omega} |\partial_u H(U_s, Q_s)(x)| \mathbf{1}_{\{|\partial_u H(U_s, Q_s)| \geq N^\delta\}}(x) \mu(dx) ds. \end{aligned} \quad (4.115)$$

Summing up the inequalities (4.110) for all indices α and (4.115), we have

$$V_t^N(y) \leq \bar{J}_t^N(y) + C \int_0^t \int_{\Omega} V_s^N(x) \mu(dx) ds, \quad (4.116)$$

where

$$V_t^N(y) := |U_t^N(y) - U_t(y)| + \sum_{|\alpha| \leq 2} |D_{\alpha}(Q_t^N - Q_t)(y)|, \quad (4.117)$$

and residual term

$$\begin{aligned} \bar{J}_t^N(y) := & C \left[\int_0^t \int_{\Omega} N^{\delta} |\bar{A}_s^N(x, y) - A(x, y)| \mu(dx) ds + \int_0^t \int_{\Omega} N^{\delta} S_s^N(x, y) dx ds \right. \\ & + \int_0^t \int_{\Omega} |\mathcal{L}^{U_s} Q_s(x)| \mathbf{1}_{|\mathcal{L}^{U_s} Q_s| \geq N^{\delta}}(x) \\ & \left. + |\partial_u H(U_s, Q_s)(x)| \mathbf{1}_{|\partial_u H(U_s, Q_s)| \geq N^{\delta}}(x) \mu(dx) ds + V_0^N(y) \right]. \end{aligned} \quad (4.118)$$

where $S_s^N(x, y) := \sum_{|\alpha| \leq 2} |D_{\alpha} B_s^N(x, y) - D_{\alpha} B_s(x, y)|$. Multiplying $V_t^N(y)$ on both sides in (4.116) and integrating with respect to measure μ give

$$\begin{aligned} \int_{\Omega} V_t^N(y)^2 \mu(dy) & \leq \int_{\Omega} V_t^N(y) \bar{J}_t^N(y) \mu(dy) + \int_0^t \int_{\Omega^2} V_s^N(x) V_t^N(y) \mu(dx) \mu(dy) ds \\ & \leq C \left(\int_{\Omega} V_t^N(y)^2 \mu(dy) \right)^{\frac{1}{2}} \left[J_T^N + \int_0^t \left(\int_{\Omega} V_s^N(y)^2 \mu(dy) \right)^{\frac{1}{2}} ds \right]. \end{aligned} \quad (4.119)$$

Here the second line is given by Cauchy inequality where the residual term J_T^N can be written as

$$\begin{aligned} J_T^N = & \left(\int_{\Omega} V_0^N(y)^2 \mu(dy) \right)^{\frac{1}{2}} + \int_0^T \left(\int_{\Omega^2} N^{2\delta} |\bar{A}_s^N(x, y) - A(x, y)|^2 \mu(dx) \mu(dy) ds \right)^{\frac{1}{2}} \\ & + \int_0^T \left(\int_{\Omega^2} N^{2\delta} S_s^N(x, y)^2 \mu(dx) \mu(dy) \right)^{\frac{1}{2}} ds + \int_0^T \int_{\Omega} |\mathcal{L}^{U_s} Q_s(x)| \mathbf{1}_{\{|\mathcal{L}^{U_s} Q_s| \geq N^{\delta}\}}(x) \\ & + \left| \partial_u H(U_s, Q_s)(x) \right| \mathbf{1}_{\{|\partial_u H(U_s, Q_s)| \geq N^{\delta}\}}(x) \mu(dx) ds. \end{aligned} \quad (4.120)$$

Therefore

$$\left(\int_{\Omega} V_t^N(y)^2 \mu(dy) \right)^{\frac{1}{2}} \leq J_T^N + C \int_0^t \left(\int_{\Omega} V_s^N(y)^2 \mu(dy) \right)^{\frac{1}{2}} ds. \quad (4.121)$$

By Gronwall's inequality and taking expectation on both sides, for $0 \leq t \leq T$

$$\mathbb{E} \left[\int_0^t \left(\int_{\Omega} V_s^N(y)^2 \mu(dy) \right)^{\frac{1}{2}} ds \right] \leq \frac{\mathbb{E}[J_T^N]}{C} (e^{Ct} - 1). \quad (4.122)$$

Now, by Lemma 4.3.10, Lemma 4.3.11, Lemma 4.3.12 and the dominated convergence theorem, $\lim_{N \rightarrow \infty} \mathbb{E}[J_T^N] = 0$. Since $\mathbb{E}[J_T^N] \rightarrow 0$ as $N \rightarrow \infty$, we have

$$\mathbb{E} \left[\int_0^t \left(\int_{\Omega} V_s^N(y)^2 \mu(dy) \right)^{\frac{1}{2}} ds \right] \rightarrow 0. \quad (4.123)$$

Therefore by (4.121)

$$\mathbb{E} \left[\left(\int_{\Omega} V_t^N(y)^2 \mu(dy) \right)^{\frac{1}{2}} \right] \rightarrow 0. \quad (4.124)$$

Notice that

$$\begin{aligned} \int_{\Omega} V_t^N(y)^2 \mu(dy) &\geq \frac{1}{2} \int_{\Omega} \left(|U_t^N(y) - U_t(y)|^2 + \sum_{|\alpha| \leq 2} |D_{\alpha}(Q_t^N - Q_t)(y)|^2 \right) \mu(dy) \\ &\geq \frac{1}{2} (\|U_t^N - U_t\|_2^2 + \|Q_t^N - Q_t\|_{\mathcal{H}^2}^2). \end{aligned} \quad (4.125)$$

As our training time bound $T \geq t$ is arbitrary, we conclude our proof that for any $t \geq 0$

$$\mathbb{E} \left[\|U_t^N - U_t\|_2^2 + \|Q_t^N - Q_t\|_{\mathcal{H}^2}^2 \right] \rightarrow 0. \quad (4.126)$$

□

4.4 Numerical examples

In this section, we test our method on two test cases. In [2], the authors already provided a few good test examples with the same method discussed in this chapter. In their paper, they implemented the algorithm mainly for problems where time variable is involved. For concrete examples on solving Merton problem, optimal execution problem, we refer readers to [2]. In this section, we provide examples that are static, with boundary condition already matched automatically by the critic. These two examples are primary, but they are useful for us to check whether our algorithm works in simple case. Also, we hope that they can help build the foundation of further development of the current algorithm, and further analysis of the algorithm in future research.

4.4.1 Test case I: linear drift

Consider the following controlled process

$$dX_t = u_t dt + dW_t, \quad X_0 = x \in [-1, 1], \quad (4.127)$$

whose discounted total cost to be minimized is given by:

$$J(x) = \mathbb{E}_x \left[\int_0^\tau e^{-\gamma t} (\kappa + u_t^2) dt \right]. \quad (4.128)$$

This simple cost consists of two components, the first one depends on how long the process stays inside the interval, and the second one penalizes the ‘energy’ it consumes for escaping. The ODE for the cost of a given feedback control $\mathbf{u}(x)$ is given by:

$$\kappa + u^2(x) - \gamma J(x) + u(x)J'(x) + \frac{1}{2}J''(x) = 0, \quad J(-1) = J(1) = 0. \quad (4.129)$$

The optimal cost function is given by

$$\min_u \{ \kappa + u^2(x) - \gamma J(x) + u(x)J'(x) + \frac{1}{2}J''(x) = 0 \}, \quad J(-1) = J(1) = 0. \quad (4.130)$$

Minimizing over u gives $u = -\frac{1}{2}J'$ and hence the equation is formulated as

$$\kappa - \gamma J - \left(\frac{J'}{2}\right)^2 + \frac{1}{2}J'' = 0, \quad J(-1) = J(1) = 0. \quad (4.131)$$

For certain κ and γ , the solution to the equation is quadratic and the optimal control is consequently linear. However, for general κ and γ , the optimal control is non-linear and the cost function is not quadratic.

We solve the problem by training our critic and action pair. The critic, Q , is set to learn the cost function in (4.128) where control U is a separate neural network, taking current state x and maps into a real value as the control status $U(x)$. For actor U , it learns to minimize the integrated Hamiltonian, which is equivalent to minimizing (over u)

$$\int_{[0,1]} u(x)^2 + u(x)Q'(x)dx. \quad (4.132)$$

Essentially, the actor is learning to match $-\frac{1}{2}Q'$ across the entire domain. After a few iterations, when the critic-actor pair converges and we test its values against the finite difference solution we obtained by solving (4.131) on a mesh grid.

4.4.1.1 Neural network training configuration

Actor is initially set as $u = 2$. The critic network is given by a single-layer 64-unit neural network, and the actor network is a separate neural network with the same structure. For offline setting, at each stage, the actor or the critic is trained until meeting stopping criteria before the other part is iterated.

4.4.1.2 Test results

We test our results with $\kappa = 1, \gamma = 0.2$. In Figure 4.2, we observed that both the critic and the actor gradually changes to converge to its stationary point. The dynamics of these changes and how different initialization effects these changes are interesting for future numerical research.

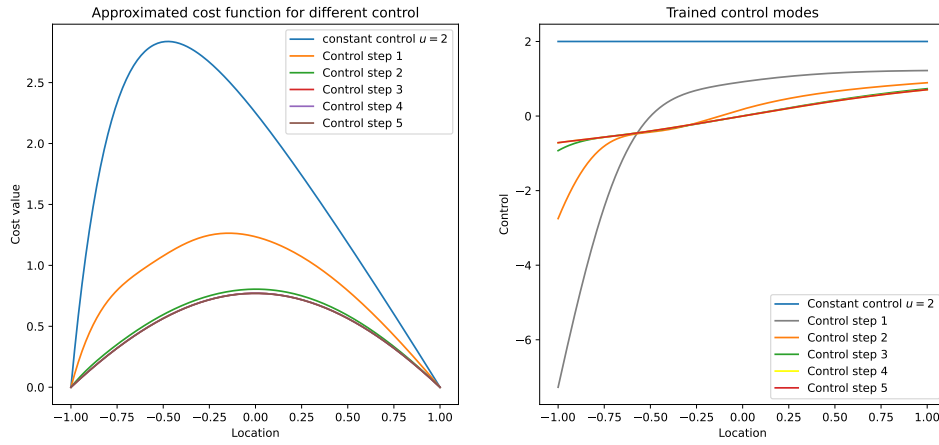


Figure 4.2: Trained critics and actors after each iteration

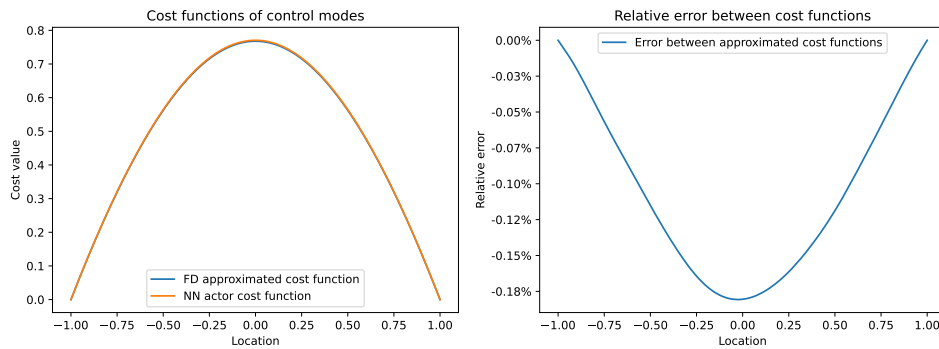


Figure 4.3: Cost function and its relative error

In Figure 4.3, the neural network solution is compared with the finite difference solution. Two solutions match with good accuracy, leaving a maximum relative error level below 0.2%. In Figure 4.4, the neural network control matches the nonlinear shape also with acceptable accuracy, only leaving larger error at the margins with scale $\sim 10^{-3}$.

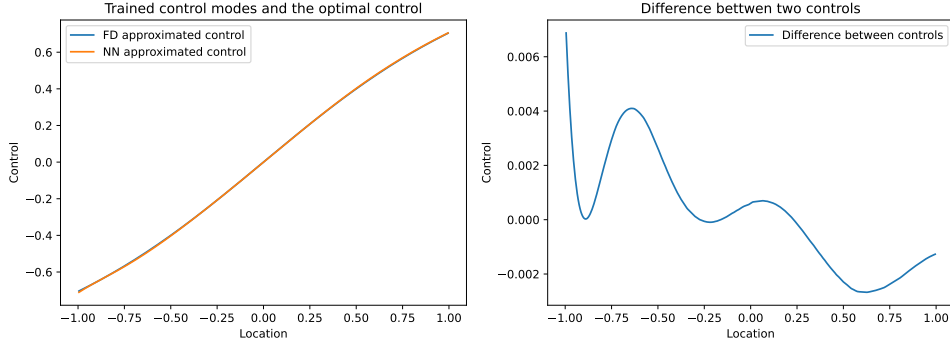


Figure 4.4: Neural network control and its error

4.4.2 Test case II: cosine drift

In this section, we consider a slightly different controlled process

$$dX_t = \cos(u_t)dt + dW_t, \quad X_0 = x \in [-1, 1], \quad (4.133)$$

with the same form of total cost to be minimized:

$$J(x) = \mathbb{E}_x \left[\int_0^\tau e^{-\gamma t} (\kappa + u_t^2) dt \right]. \quad (4.134)$$

This time, the intuition is different. For $x > 0$, since we know $\cos(u_t) \leq 1$ and $u_t^2 \geq 0$, for $x \geq 0$ it is easy to verify that the optimal control is $u(x) = 0$. It is the fastest but also most cost-efficient way of exiting the domain. For a fixed (feedback) control u , its cost is evaluated by

$$\kappa + u^2(x) - \gamma J(x) + \cos(u(x))J'(x) + \frac{1}{2}J''(x) = 0, \quad J(-1) = J(1) = 0. \quad (4.135)$$

On the other hand, once we obtain an approximation of cost function $Q \approx J^u$, the actor can be applied to minimize the integrated Hamiltonian via calculating

$$\arg \min_u \int_{[0,1]} u^2(x) + \cos(u(x))Q'(x)dx. \quad (4.136)$$

Iteratively solving (4.135) and (4.136) may leads to finding the optimal cost function which is given by

$$\min_u \{ \kappa + u^2(x) - \gamma J(x) + \cos(u(x)) J'(x) + \frac{1}{2} J''(x) \} = 0, \quad J(-1) = J(1) = 0. \quad (4.137)$$

4.4.2.1 Neural network training configuration

We apply the same offline training configuration as in last test case. Actor is initially set as $u = 2$. The critic network and the actor network are separate neural networks with the same structure: 64 units and single-layer.

4.4.2.2 Test results

We take parameters $\kappa = 1$ and $\gamma = 0.2$. The training of actor-critic pairs converges after 6 iterations. On the left-hand side of Figure 4.5, the critic gradually converges to its final state. The other half of the figure demonstrates how actor network converges from constant control $u = 2$ in different iterations. Intuitively, as a sanity check, we observe that u converges to 0 for $x \geq 0$.

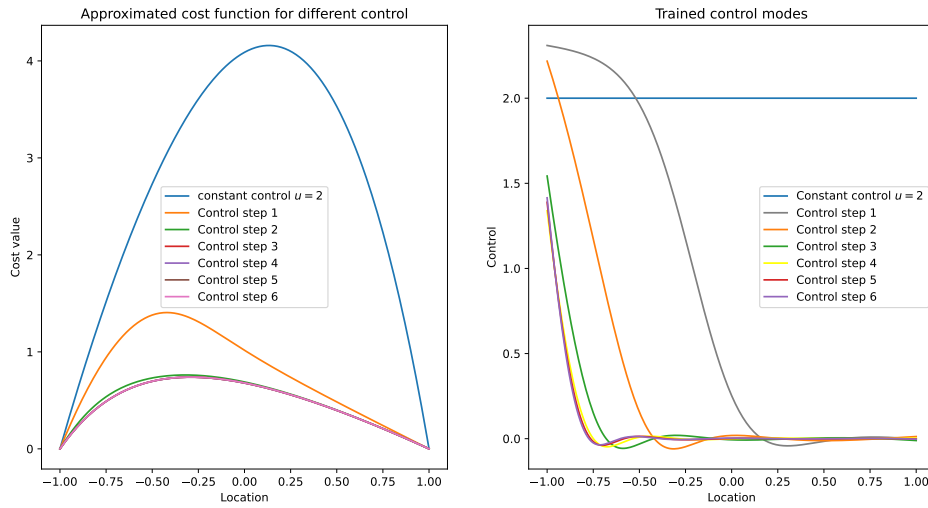


Figure 4.5: Trained critics and actors after each iteration

The optimal cost function is computed via applying FD approximation scheme to equation (4.137) on a grid of mesh size 0.05. Neural network solution is compared to this FD solution with results given in Figure 4.6. The maximum relative error is

at the scale of $\sim 0.1\%$. Finally, the FD solution also induces a sub-optimal control by computing pointwise minimum on those mesh points. The results are given in Figure (4.7). The sub-optimal control stays zero for $x \geq 0.8$ but changes sharply otherwise. Our neural network actor matches the control for $x < -0.9$ and $x > -0.8$, but a large gap exist for the interval in between. This suggest that, a single-layer neural network actor with smooth activation function may not be a good choice in this case for better precision. For future development of our model, applying multi-layer networks or taking ReLU activation function are suggested.

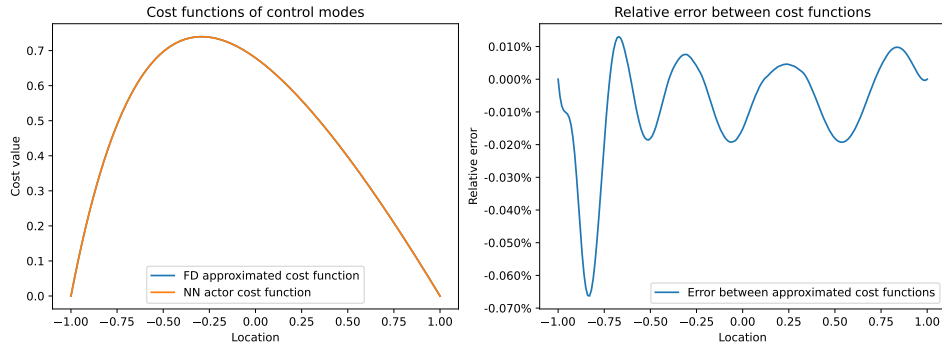


Figure 4.6: Cost function and its relative error

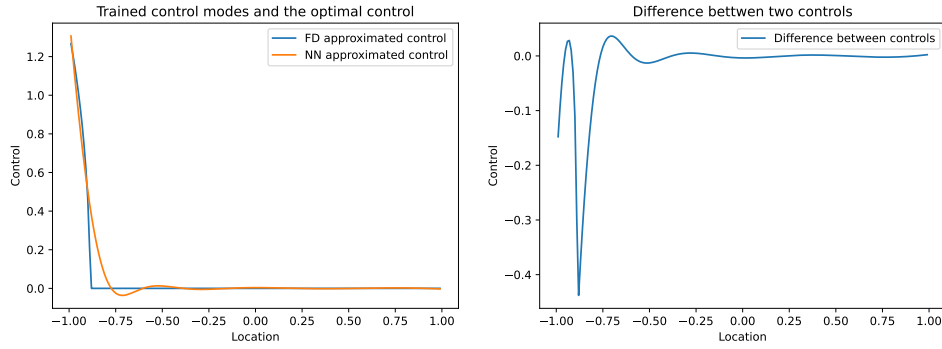


Figure 4.7: Neural network control and its error

Chapter 5

Conclusions & future works

In this thesis, we have studied neural-network-based PDE-solving algorithms in several aspects. In particular, we (i) propose the Q-PDE algorithm for solving a class of monotone PDEs, (ii) provide a convergence guarantee for deep Galerkin method (DGM) and Physics-informed Neural Networks (PINNs) solving linear PDEs, (iii) analyze the approach of solving Hamilton–Jacobi–Bellman PDEs via solving a series of linear PDEs with neural networks.

Solving PDEs numerically is not an easy task for neural networks. It requires learning the boundary condition and the PDE operator simultaneously. Unlike classic regression problems, these learning objectives involve differential operators which are typically unbounded in L^2 norm. In some cases, the non-linearity of PDE operators makes analysis even harder. To address these challenges, in Chapter 2, we show that certain boundary conditions can be embedded into the neural network representation by introducing an auxiliary function, and we can train the neural network approximator via a biased gradient flow for monotone PDEs. A limiting ODE characterizes the training process for the neural network approximator as the number of units goes to infinity, from which we show the convergence of the approximator.

In Chapter 3, we examine the training of neural networks under the deep Galerkin method (DGM) and its similar variants for solving linear PDEs via gradient descent. A novel neural tangent kernel is proposed, which induces the integral operator describing the dynamics of the approximation’s evolution over time. By the representation power of the neural network family, we prove that the optimization objective of DGM is achievable (though without an available convergence rate) and hence the PDE solution is evaluated.

Next, in Chapter 4, we study policy-iteration-based ideas of solving HJB PDEs via solving a series of linear PDEs associated with the controlled process. In the first half of the chapter, we examine whether neural networks can perform the (pointwise)

policy update steps via learning to minimize an alternative objective. In its second half, we model online actor-critic methods where the actor and critic networks are updated simultaneously.

A few interesting questions derived from the work in this thesis remain not answered, and we highlight some of them as follows:

1. *Multi-layer networks*: The analysis in this thesis may extend to the case where multi-layer neural networks are adopted rather than single-layer ones. Corresponding NTK and dynamics can be modeled.
2. *SGD*: We studied the gradient descent approach in this thesis. Convergence of those methods via stochastic gradient descent is of great practical interest.
3. *Eigenmodes*: In this thesis, we didn't numerically analyze the eigenfunctions and the eigenvalues of the novel NTK we proposed. Practically understanding the distribution of the eigenmodes may deepen our understanding of these neural-network-based training algorithms.
4. *Regularizer*: Practical application of these algorithms often suggests adding regularizing terms to the training objectives. We studied the vanilla version of these methods without such additional terms. Further studies can be extended to these cases where regularizers are applied.
5. *Broader ranges of PDEs*: By modifying our setting, we may analyze the process of solving parabolic PDEs with different kinds of (boundary-matching) auxiliary functions, or more general cases where auxiliary functions are not included. Also, solving specific families of nonlinear PDEs with these algorithms (e.g. Navier–Stokes equations) is significant but this thesis does not cover these cases. Our analysis can be further extended to other ranges of equations, for instance, stochastic PDEs (SPDEs).

Bibliography

- [1] Robert A. Adams and John J.F. Fournier. *Sobolev spaces*. Elsevier, 2003.
- [2] Ali Al-Aradi, Adolfo Correia, Gabriel Jardim, Danilo de Freitas Naiff, and Yuri Saporito. Extensions of the deep galerkin method. *Applied Mathematics and Computation*, 430:127287, 2022.
- [3] Anima Anandkumar, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Nikola Kovachki, Zongyi Li, Burigede Liu, and Andrew Stuart. Neural operator: Graph kernel network for partial differential equations. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.
- [4] Christian Beck, Martin Hutzenthaler, Arnulf Jentzen, and Benno Kuckuck. An overview on deep learning-based approximation methods for partial differential equations. *arXiv preprint arXiv:2012.12348*, 2020.
- [5] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.
- [6] Julius Berner, Markus Dablander, and Philipp Grohs. Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning. *Advances in Neural Information Processing Systems*, 33:16615–16627, 2020.
- [7] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE transactions on neural networks and learning systems*, 25(8):1553–1565, 2014.
- [8] Felix E. Browder. Existence and perturbation theorems for nonlinear maximal monotone operators in Banach spaces. *Bulletin of the American Mathematical Society*, 73(3):322–327, 1967.

- [9] Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*, 365(6456):885–890, 2019.
- [10] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.
- [11] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.
- [12] Zhiqiang Cai, Jingshuang Chen, Min Liu, and Xinyu Liu. Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic pdes. *Journal of Computational Physics*, 420:109707, 2020.
- [13] René Carmona and Mathieu Laurière. Deep learning for mean field games and mean field control with applications to finance. *arXiv preprint arXiv:2107.04568*, 7, 2021.
- [14] Fan Chen, Jianguo Huang, Chunmei Wang, and Haizhao Yang. Friedrichs learning: Weak solutions of partial differential equations via deep learning. *SIAM Journal on Scientific Computing*, 45(3):A1271–A1299, 2023.
- [15] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on neural networks*, 6(4):911–917, 1995.
- [16] Ziang Chen, Jianfeng Lu, and Yulong Lu. On the representation of solutions to elliptic pdes in barron spaces. *Advances in neural information processing systems*, 34:6454–6465, 2021.
- [17] Mengyu Chu, Lingjie Liu, Quan Zheng, Erik Franz, Hans-Peter Seidel, Christian Theobalt, and Rhaleb Zayer. Physics informed neural fields for smoke reconstruction with sparse data. *ACM Transactions on Graphics (TOG)*, 41(4):1–14, 2022.
- [18] Samuel N. Cohen, Deqing Jiang, and Justin Sirignano. Neural Q-learning for solving pdes. *Journal of Machine Learning Research*, 24(236):1–49, 2023.
- [19] John B Conway. *A course in functional analysis*, volume 96. Springer, 2019.

- [20] Francisco Sahli Costabal, Simone Pezzuto, and Paris Perdikaris. δ -PINNs: physics-informed neural networks on complex geometries. *Engineering Applications of Artificial Intelligence*, 127:107324, 2024.
- [21] Michael G Crandall and Pierre-Louis Lions. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American mathematical society*, 277(1):1–42, 1983.
- [22] John Crank and Phyllis Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. In *Mathematical proceedings of the Cambridge philosophical society*, volume 43, pages 50–67. Cambridge University Press, 1947.
- [23] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics-informed neural networks: Where we are and what’s next. *Journal of Scientific Computing*, 92(3):88, 2022.
- [24] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- [25] Beichuan Deng, Yeonjong Shin, Lu Lu, Zhongqiang Zhang, and George Em Karniadakis. Convergence rate of deepoanets for learning operators arising from advection-diffusion equations. *arXiv preprint arXiv:2102.10621*, 2021.
- [26] MWMG Dissanayake and Nhan Phan-Thien. Neural-network-based approximations for solving partial differential equations. *communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.
- [27] Nathan Doumèche, Gérard Biau, and Claire Boyer. Convergence and error analysis of pinns. *arXiv preprint arXiv:2305.01240*, 2023.
- [28] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.
- [29] Nicole El Karoui, Étienne Pardoux, and Marie Claire Quenez. Reflected backward sdes and american options. *Numerical methods in finance*, 13:215–231, 1997.

- [30] Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.
- [31] Olga Fuks and Hamdi A Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1), 2020.
- [32] Maximilien Germain, Huy  n Pham, Xavier Warin, et al. Neural networks-based algorithms for stochastic control and pdes in finance. *arXiv preprint arXiv:2101.08068*, 2021.
- [33] David Gilbarg, Neil S Trudinger, David Gilbarg, and NS Trudinger. *Elliptic partial differential equations of second order*, volume 224. Springer, 1977.
- [34] Desa Gobovic and Mona E Zaghloul. Analog cellular neural network with application to partial differential equations with variable mesh-size. In *Proceedings of IEEE International Symposium on Circuits and Systems-ISCAS'94*, volume 6, pages 359–362. IEEE, 1994.
- [35] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.
- [36] Lukas Gonon and Christoph Schwab. Deep relu neural networks overcome the curse of dimensionality for partial integrodifferential equations. *Analysis and Applications*, 21(01):1–47, 2023.
- [37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [38] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
- [39] Pierre Grisvard. *Elliptic problems in nonsmooth domains*. SIAM, 2011.
- [40] Philipp Grohs and Lukas Herrmann. Deep neural network approximation for high-dimensional elliptic pdes with boundary conditions. *IMA Journal of Numerical Analysis*, 42(3):2055–2082, 2022.

- [41] Philipp Grohs, Fabian Hornung, Arnulf Jentzen, and Philippe Von Wurstemberger. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of black-scholes partial differential equations. *arXiv preprint arXiv:1809.02362*, 2018.
- [42] Jiequn Han et al. Deep learning approximation for stochastic control problems. *arXiv preprint arXiv:1611.07422*, 2016.
- [43] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.
- [44] Jiequn Han, Arnulf Jentzen, et al. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in mathematics and statistics*, 5(4):349–380, 2017.
- [45] Jiequn Han and Jihao Long. Convergence of the deep bsde method for coupled fbsdes. *Probability, Uncertainty and Quantitative Risk*, 5:1–33, 2020.
- [46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [47] Alexander Henkes, Henning Wessels, and Rolf Mahnken. Physics informed neural networks for continuum micromechanics. *Computer Methods in Applied Mechanics and Engineering*, 393:114790, 2022.
- [48] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.
- [49] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
- [50] Ruimeng Hu, Qu Yuan Lin, Alan Raydan, and Sui Tang. Higher-order error estimates for physics-informed neural networks approximating the primitive equations. *arXiv preprint arXiv:2209.11929*, 2022.
- [51] Côme Huré, Huyên Pham, Achref Bachouch, and Nicolas Langrené. Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis. *SIAM Journal on Numerical Analysis*, 59(1):525–557, 2021.

- [52] Côme Huré, Huyên Pham, and Xavier Warin. Deep backward schemes for high-dimensional nonlinear pdes. *Mathematics of Computation*, 89(324):1547–1579, 2020.
- [53] Martin Hutzenthaler, Arnulf Jentzen, Thomas Kruse, and Tuan Anh Nguyen. A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations. *SN partial differential equations and applications*, 1:1–34, 2020.
- [54] Kazufumi Ito, Christoph Reisinger, and Yufei Zhang. A neural network-based policy iteration algorithm with global h^2 -superlinear convergence for stochastic games on domains. *Foundations of Computational Mathematics*, 21(2):331–374, 2021.
- [55] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.
- [56] Ameya D Jagtap and George Em Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5), 2020.
- [57] Ameya D Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020.
- [58] Ameya D Jagtap, Zhiping Mao, Nikolaus Adams, and George Em Karniadakis. Physics-informed neural networks for inverse problems in supersonic flows. *Journal of Computational Physics*, 466:111402, 2022.
- [59] Deqing Jiang, Justin Sirignano, and Samuel N Cohen. Global convergence of deep galerkin and pinns methods for solving partial differential equations. *arXiv preprint arXiv:2305.06000*, 2023.
- [60] Yifan Jiang and Jinfeng Li. Convergence of the deep bsde method for fbsdes with non-lipschitz coefficients. *arXiv preprint arXiv:2101.01869*, 2021.

- [61] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.
- [62] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.
- [63] Olav Kallenberg. *Foundations of Modern Probability*. Springer, 2nd edition, 2002.
- [64] Sadegh Karimpouli and Pejman Tahmasebi. Physics informed machine learning: Seismic wave equation. *Geoscience Frontiers*, 11(6):1993–2001, 2020.
- [65] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.
- [66] Arbaaz Khan and David A Lowther. Physics informed neural networks for electromagnetic analysis. *IEEE Transactions on Magnetics*, 58(9):1–4, 2022.
- [67] Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. hp-vpinns: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021.
- [68] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving for high-dimensional committor functions using artificial neural networks. *Research in the Mathematical Sciences*, 6:1–13, 2019.
- [69] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [70] Georgios Kissas, Yibo Yang, Eileen Hwuang, Walter R Witschey, John A Detre, and Paris Perdikaris. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358:112623, 2020.

- [71] Vassili Kolokoltsov. *Differential equations on measures and functional spaces*. Springer, 2019.
- [72] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.
- [73] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *The Journal of Machine Learning Research*, 22(1):13237–13312, 2021.
- [74] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.
- [75] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.
- [76] Thorsten Kurth, Shashank Subramanian, Peter Harrington, Jaideep Pathak, Morteza Mardani, David Hall, Andrea Miele, Karthik Kashinath, and Anima Anandkumar. Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–11, 2023.
- [77] Olga Aleksandrovna Ladyzhenskaia, Vsevolod Alekseevich Solonnikov, and Nina N Ural'tseva. *Linear and quasi-linear equations of parabolic type*, volume 23. American Mathematical Soc., 1968.
- [78] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.
- [79] Isaac E Lagaris, Aristidis C Likas, and Dimitris G Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, 2000.
- [80] Samuel Lanthaler, Siddhartha Mishra, and George E Karniadakis. Error estimates for deepnets: A deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 2022.

- [81] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- [82] Hyuk Lee and In Seok Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110–131, 1990.
- [83] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.
- [84] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32:8572–8583, 2019.
- [85] Junjie Li, Sotetsu Koyamada, Qiwei Ye, Guoqing Liu, Chao Wang, Ruihan Yang, Li Zhao, Tao Qin, Tie-Yan Liu, and Hsiao-Wuen Hon. Suphx: Mastering mahjong with deep reinforcement learning. *arXiv preprint arXiv:2003.13590*, 2020.
- [86] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021.
- [87] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
- [88] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.
- [89] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deepnet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

- [90] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.
- [91] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.
- [92] Yiping Lu, Jose Blanchet, and Lexing Ying. Sobolev acceleration and statistical optimality for learning elliptic equations via gradient descent. *Advances in Neural Information Processing Systems*, 35:33233–33247, 2022.
- [93] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30, 2017.
- [94] Günter Lumer and R.S. Phillips. Dissipative operators in a Banach space. *Pacific Journal of Mathematics*, 11:679–698, 1961.
- [95] Tao Luo and Haizhao Yang. Two-layer neural networks for partial differential equations: Optimization and generalization theory. *arXiv preprint arXiv:2006.15733*, 2020.
- [96] Mohammadamin Mahmoudabadbozchelou, George Em Karniadakis, and Safa Jamali. nn-pinns: Non-newtonian physics-informed neural networks for complex fluid modeling. *Soft Matter*, 18(1):172–185, 2022.
- [97] Alaeddin Malek and R Shekari Beidokhti. Numerical solution for high order differential equations using a hybrid neural network—optimization method. *Applied Mathematics and Computation*, 183(1):260–271, 2006.
- [98] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.
- [99] Carlo Marcati and Christoph Schwab. Exponential convergence of deep operator networks for elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 61(3):1513–1545, 2023.

- [100] Kevin Stanley McFall and James Robert Mahan. Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. *IEEE Transactions on Neural Networks*, 20(8):1221–1233, 2009.
- [101] Andrew J Meade Jr and Alvaro A Fernandez. The numerical solution of linear ordinary differential equations by feedforward neural networks. *Mathematical and Computer Modelling*, 19(12):1–25, 1994.
- [102] Andrew J Meade Jr and Alvaro A Fernandez. Solution of nonlinear ordinary differential equations by feedforward neural networks. *Mathematical and Computer Modelling*, 20(9):19–44, 1994.
- [103] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.
- [104] Pavan Pranjivan Mehta, Guofei Pang, Fangying Song, and George Em Karniadakis. Discovering a universal variable-order fractional model for turbulent couette flow using a physics-informed neural network. *Fractional calculus and applied analysis*, 22(6):1675–1688, 2019.
- [105] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.
- [106] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.
- [107] Siddhartha Mishra and Roberto Molinaro. Physics informed neural networks for simulating radiative transfer. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 270:107705, 2021.
- [108] George S Misyris, Andreas Venzke, and Spyros Chatzivasileiadis. Physics-informed neural networks for power systems. In *2020 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2020.
- [109] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

- [110] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [111] Johannes Müller and Marius Zeinhofer. Deep ritz revisited. *arXiv preprint arXiv:1912.03937*, 2019.
- [112] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [113] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.
- [114] Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.
- [115] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624, 2020.
- [116] Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.
- [117] Etienne Pardoux and Shige Peng. Adapted solution of a backward stochastic differential equation. *Systems & control letters*, 14(1):55–61, 1990.
- [118] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.
- [119] Huy  n Pham. *Continuous-time stochastic control and optimization with financial applications*, volume 61. Springer Science & Business Media, 2009.
- [120] Huy  n Pham and Xavier Warin. Mean-field neural networks-based algorithms for mckean-vlasov control problems. *arXiv preprint arXiv:2212.11518*, 2022.

- [121] Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.
- [122] Maziar Raissi, Paris Perdikaris, and George E 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.
- [123] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.
- [124] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.
- [125] Majid Rasht-Behesht, Christian Huber, Khemraj Shukla, and George Em Karniadakis. Physics-informed neural networks (pinns) for wave propagation and full waveform inversions. *Journal of Geophysical Research: Solid Earth*, 127(5):e2021JB023120, 2022.
- [126] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.
- [127] Christoph Reisinger and Yufei Zhang. Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems. *Analysis and Applications*, 18(06):951–999, 2020.
- [128] Keith Rudd. *Solving partial differential equations using artificial neural networks*. PhD thesis, Duke University, 2013.
- [129] Keith Rudd and Silvia Ferrari. A constrained integration (cint) approach to solving partial differential equations using artificial neural networks. *Neurocomputing*, 155:277–285, 2015.
- [130] Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris, Daniel E Hurtado, and Ellen Kuhl. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8:42, 2020.

- [131] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.
- [132] Yeonjong Shin, Jerome Darbon, and George Em Karniadakis. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes. *arXiv preprint arXiv:2004.01806*, 2020.
- [133] Yeonjong Shin, Zhongqiang Zhang, and George Em Karniadakis. Error estimates of residual minimization using neural networks for linear pdes. *Journal of Machine Learning for Modeling and Computing*, 4(4), 2023.
- [134] Ralph Edwin Showalter. *Monotone operators in Banach space and nonlinear partial differential equations*, volume 49. American Mathematical Soc., 2013.
- [135] Khemraj Shukla, Patricio Clark Di Leoni, James Blackshire, Daniel Sparkman, and George Em Karniadakis. Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks. *Journal of Nondestructive Evaluation*, 39:1–20, 2020.
- [136] Khemraj Shukla, Ameya D Jagtap, and George Em Karniadakis. Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics*, 447:110683, 2021.
- [137] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.
- [138] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.
- [139] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 130(3):1820–1852, 2020.
- [140] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A law of large numbers. *SIAM Journal on Applied Mathematics*, 80(2):725–752, 2020.

- [141] Justin Sirignano and Konstantinos Spiliopoulos. Asymptotics of reinforcement learning with neural networks. *Stochastic Systems*, 12(1):2–29, 2022.
- [142] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of deep neural networks. *Mathematics of Operations Research*, 47(1):120–152, 2022.
- [143] Iain Smears and Endre Süli. Discontinuous Galerkin finite element approximation of Hamilton–Jacobi–Bellman equations with Cordes coefficients. *SIAM Journal on Numerical Analysis*, 52(2):993–1016, 2014.
- [144] Endre Süli. Lecture notes on finite element methods for partial differential equations. *Mathematical Institute, University of Oxford*, 2012.
- [145] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [146] Vimal Thilak, Etai Littwin, Shuangfei Zhai, Omid Saremi, Roni Paiss, and Joshua Susskind. The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon. *arXiv preprint arXiv:2206.04817*, 2022.
- [147] Tapas Tripura and Souvik Chakraborty. Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 404:115783, 2023.
- [148] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.
- [149] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.
- [150] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deepnets. *Science advances*, 7(40):eabi8605, 2021.
- [151] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

- [152] Christopher J.C.H. Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
- [153] E Weinan, Jiequn Han, and Arnulf Jentzen. Algorithms for solving high dimensional pdes: from nonlinear monte carlo to machine learning. *Nonlinearity*, 35(1):278, 2021.
- [154] E Weinan, Chao Ma, and Lei Wu. Barron spaces and the compositional function spaces for neural network models. *arXiv preprint arXiv:1906.08039*, 2019.
- [155] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.
- [156] R Yentis and Mona E Zaghloul. Vlsi implementation of locally connected neural network for solving partial differential equations. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 43(8):687–690, 1996.
- [157] Bing Yu et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.
- [158] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.
- [159] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, 2020.
- [160] Eberhard Zeidler. *Nonlinear Functional Analysis and Its Applications: II/B: Nonlinear Monotone Operators*. Springer, 2013.
- [161] Enrui Zhang, Ming Dao, George Em Karniadakis, and Subra Suresh. Analyses of internal structures and defects in materials using physics-informed neural networks. *Science advances*, 8(7):eabk0644, 2022.
- [162] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*, 2019.

- [163] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. Deep reinforcement learning for search, recommendation, and online advertising: a survey. *ACM SIGWEB Newsletter*, (Spring):1–15, 2019.
- [164] Mo Zhou, Jiequn Han, and Jianfeng Lu. Actor-critic method for high dimensional static hamilton–jacobi–bellman partial differential equations based on neural networks. *SIAM Journal on Scientific Computing*, 43(6):A4043–A4066, 2021.