Acta Numerica (2024), pp. 633–713 Printed in the United Kingdom
doi:10.1017/S0962492923000089

Numerical analysis of physics-informed neural networks and related models in physics-informed machine learning

Tim De Ryck
Seminar for Applied Mathematics, ETH Zürich,
Rämistrasse 101, 8092 Zürich, Switzerland
E-mail: tim.deryck@math.ethz.ch

Siddhartha Mishra
Seminar for Applied Mathematics & ETH AI Center, ETH Zürich,
Rämistrasse 101, 8092 Zürich, Switzerland
E-mail: siddhartha.mishra@math.ethz.ch

Physics-informed neural networks (PINNs) and their variants have been very popular in recent years as algorithms for the numerical simulation of both forward and inverse problems for partial differential equations. This article aims to provide a comprehensive review of currently available results on the numerical analysis of PINNs and related models that constitute the backbone of physics-informed machine learning. We provide a unified framework in which analysis of the various components of the error incurred by PINNs in approximating PDEs can be effectively carried out. We present a detailed review of available results on approximation, generalization and training errors and their behaviour with respect to the type of the PDE and the dimension of the underlying domain. In particular, we elucidate the role of the regularity of the solutions and their stability to perturbations in the error analysis. Numerical results are also presented to illustrate the theory. We identify training errors as a key bottleneck which can adversely affect the overall performance of various models in physics-informed machine learning.

2020 Mathematics Subject Classification: Primary 65M15
Secondary 68T07, 35A35

CONTENTS

1. Introduction

Machine learning, particularly deep learning (Goodfellow, Bengio and Courville 2016), has permeated into every corner of modern science, technology and society, whether it is computer vision, natural language understanding, robotics and automation, image and text generation or protein folding and prediction.

The application of deep learning to computational science and engineering, in particular the numerical solution of (partial) differential equations, has gained enormous momentum in recent years. One notable avenue highlighting this application falls within the framework of *supervised learning*, i.e. approximating the solutions of PDEs by deep learning models from (large amounts of) data about the underlying PDE solutions. Examples include high-dimensional parabolic PDEs (Han, Jentzen and E 2018 and references therein), parametric elliptic (Schwab and Zech 2019, Kutyniok, Petersen, Raslan and Schneider 2022) or hyperbolic PDEs (De Ryck and Mishra 2023, Lye, Mishra and Ray 2020) and learning the solution operator directly from data (Chen and Chen 1995, Lu *et al.* 2021*b*, Li *et al.* 2021, Raonić *et al.* 2023 and references therein). However, generating and accessing large amounts of data on solutions of PDEs requires either numerical methods or experimental data, both of which can be (prohibitively) expensive. Consequently, there is a need for so-called *unsupervised* or *semi-supervised* learning methods which require only small amounts of data.

Given this context, it would be useful to solve PDEs using machine learning models and methods, directly from the underlying physics (governing equations) and without having to access data about the underlying solutions. Such methods can be loosely termed as constituting *physics-informed machine learning*.

The most prominent contemporary models in physics-informed machine learning are *physics-informed neural networks* or PINNs. The idea behind PINNs is very simple: as neural networks are universal approximators of a large variety of function classes (even measurable functions), we consider the strong form of the PDE residual within the *ansatz space* of neural networks, and minimize this residual

via (stochastic) gradient descent to obtain a neural network that approximates the solution of the underlying PDE. This framework was already considered in the 1990s, in Dissanayake and Phan-Thien (1994), Lagaris, Likas and Fotiadis (1998), Lagaris, Likas and Papageorgiou (2000) and references therein, and these papers should be considered as the progenitors of PINNs. However, the modern version of PINNs was introduced, and named more recently in Raissi and Karniadakis (2018) and Raissi, Perdikaris and Karniadakis (2019). Since then, there has been an explosive growth in the literature on PINNs and they have been applied in numerous settings, in the solution of both forward and inverse problems for PDEs and related equations (SDEs, SPDEs); see Karniadakis *et al.* (2021) and Cuomo *et al.* (2022) for extensive reviews of the available literature on PINNs.

However, PINNs are not the only models within the framework of physics-informed machine learning. One can consider other forms of the PDE residual such as the variational form resulting in VPINNs (Kharazmi, Zhang and Karniadakis 2019), the weak form resulting in wPINNs (De Ryck, Mishra and Molinaro 2024*c*), or minimizing the underlying energy resulting in the DeepRitz method (E and Yu 2018). Similarly, one can use Gaussian processes (Rasmussen 2003), DeepONets (Lu *et al.* 2021*b*) or neural operators (Kovachki *et al.* 2023) as ansatz spaces to realize alternative models for physics-informed machine learning.

Given the exponentially growing literature on PINNs and related models in physics-informed machine learning, it is essential to ask whether these methods possess rigorous mathematical guarantees on their performance, and whether one can analyse these methods in a manner that is analogous to the huge literature on the analysis of traditional numerical methods such as finite differences, finite elements, finite volumes and spectral methods. Although the overwhelming focus of research has been on the widespread applications of PINNs and their variants in different domains in science and engineering, a significant number of papers rigorously analysing PINNs have emerged in recent years. The aim of this article is to review the available literature on the *numerical analysis of PINNs and related models that constitute physics-informed machine learning*. Our goal is to critically analyse PINNs and its variants with a view to ascertaining when they can be applied and what are the limits to their applicability. To this end, we start in Section 2 by presenting the formulation for physics-informed machine-learning in terms of the underlying PDEs, their different forms of residuals and the approximating ansatz spaces. In Section 3 we outline the main components of the underlying errors with physics-informed machine learning; the resulting approximation, stability, generalization and training errors are analysed in Sections 4, 5, 6 and 7, respectively.

2. Physics-informed machine learning

In this section we set the stage for the rest of the article. In Section 2.1 we introduce an abstract PDE setting and consider a number of important examples of PDEs that will be used in the numerical analysis later on. Next, in Section 2.2, different model

classes are introduced. In Section 2.3 we introduce variations of physics-informed learning based on different formulations of PDEs, of which the resulting residuals will constitute the physics-informed loss function by being discretized (Section 2.4) and optimized (Section 2.5). Finally, a summary of the whole method is given in Section 2.6.

2.1. PDE setting

Throughout this article, we consider the following setting. Let X, Y, W be separable Banach spaces with norms $\|\cdot\|_X$, $\|\cdot\|_Y$ and $\|\cdot\|_W$, respectively, and analogously let $X^* \subset X$, $Y^* \subset Y$ and $W^* \subset W$ be closed subspaces with norms $\|\cdot\|_{X^*}$, $\|\cdot\|_{Y^*}$ and $\|\cdot\|_{W^*}$, respectively. Typical examples of such spaces include L^p and Sobolev spaces. We consider PDEs of the abstract form

$$\mathcal{L}[u] = f, \quad \mathcal{B}[u] = g, \tag{2.1}$$

where $\mathcal{L}\colon X^* \to Y^*$ is a *differential operator* and $f \in Y^*$ is an *input* or *source* function. The boundary conditions (including the initial condition for time-dependent PDEs) are prescribed by the *boundary operator* $\mathcal{B}\colon X^* \to W^*$ and the boundary data $g \in W^*$. We assume that for all $f \in Y^*$ there exists a unique $u \in X^*$ such that (2.1) holds. Finally, we assume that for all $u \in X^*$ and $f \in Y^*$, we have

$$\|\mathcal{L}[u]\|_{Y^*} < +\infty, \quad \|f\|_{Y^*} < +\infty, \quad \|\mathcal{B}[u]\|_{W^*} < +\infty. \tag{2.2}$$

We will also denote the domain of u as Ω, where either $\Omega = D \subset \mathbb{R}^d$ for time-independent PDEs or $\Omega = D \times [0, T] \subset \mathbb{R}^{d+1}$ for time-dependent PDEs.

2.1.1. Forward problems

The *forward problem* for the abstract PDE (2.1) amounts to the following: given a source f and boundary conditions g (and potentially another parameter function a such that $\mathcal{L} \coloneqq \mathcal{L}_a$), find the solution u of the PDE. This can be summarized as finding an *operator* that maps an input function $v \subset \{f, g, a\} \subset \mathcal{X}$ to the corresponding solution $u \in X^*$ of the PDE

$$\mathcal{G}\colon \mathcal{X} \to X^*\colon v \mapsto \mathcal{G}[v] = u. \tag{2.3}$$

Note that it can also be interesting to determine the whole operator $\mathcal{G}$ rather than just the function u. Finding a good approximation of $\mathcal{G}$ is referred to as *operator learning*. A first concrete example of an operator would be the mapping between the input or source function f and the corresponding solution u, and hence $\mathcal{X} = Y^*$ and $v = f$:

$$\mathcal{G}\colon Y^* \to X^*\colon f \mapsto \mathcal{G}[f] = u. \tag{2.4}$$

Another common example for a time-dependent PDE, where $\mathcal{B}[u] = g$ entails the prescription of the initial condition $u(\cdot, 0) = u_0$, is the *solution operator* that maps the initial condition u_0 to the solution of the PDE

$$\mathcal{G}\colon W^* \to X^*\colon u_0 \mapsto \mathcal{G}[u_0] = u, \tag{2.5}$$

where $\mathcal{X} = W^*$ and $v = u_0$. Alternatively, one can also consider the mapping $u_0 \mapsto u(\cdot, T)$.

A final example of operator learning can be found in *parametric PDEs*, where the differential operator, source function, initial condition or boundary condition depends on scalar parameters or a parameter function. Although solving parametric PDEs is usually not called operator learning, it is evident that it can be identified as such, the only difference being that the input space is not necessarily infinite-dimensional.

In what follows we give examples of PDEs (2.1) for which we will consider the forward problem later on.

Example 2.1 (semilinear heat equation). We first consider the set-up of the semilinear heat equation. Let $D \subset \mathbb{R}^d$ be an open connected bounded set with a continuously differentiable boundary ∂D. The semilinear parabolic equation is then given by

$$\begin{cases} u_t = \Delta u + f(u) & \text{for all } x \in D,\ t \in (0,T), \\ u(x,0) = u_0(x) & \text{for all } x \in D, \\ u(x,t) = 0 & \text{for all } x \in \partial D,\ t \in (0,T). \end{cases} \tag{2.6}$$

Here, $u_0 \in C^k(D)$, $k \geq 2$, is the initial data, $u \in C^k([0,T] \times D)$ is the classical solution and $f : \mathbb{R} \times \mathbb{R}$ is the nonlinear source (reaction) term. We assume that the nonlinearity is globally Lipschitz, that is, there exists a constant $C_f > 0$ such that

$$|f(v) - f(w)| \leq C_f |v - w| \quad \text{for all } v, w \in \mathbb{R}. \tag{2.7}$$

In particular, the homogeneous linear heat equation with $f(u) \equiv 0$ and the linear source term $f(u) = \alpha u$ are examples of (2.6). Semilinear heat equations with globally Lipschitz nonlinearities arise in several models in biology and finance (Beck *et al.* 2021). The existence, uniqueness and regularity of the semilinear parabolic equations with Lipschitz nonlinearities such as (2.6) can be found in classical textbooks such as that of Friedman (1964).

Example 2.2 (Navier–Stokes equations). We consider the well-known incompressible Navier–Stokes equations (Temam 2001 and references therein)

$$\begin{cases} u_t + u \cdot \nabla u + \nabla p = \nu \Delta u & \text{in } D \times [0,T], \\ \operatorname{div}(u) = 0 & \text{in } D \times [0,T], \\ u(t=0) = u_0 & \text{in } D. \end{cases} \tag{2.8}$$

Here, $u : D \times [0,T] \to \mathbb{R}^d$ is the fluid velocity, $p : D \times [0,T] \to \mathbb{R}$ is the pressure and $u_0 : D \to \mathbb{R}^d$ is the initial fluid velocity. The viscosity is denoted by $\nu \geq 0$. For the rest of the paper, we consider the Navier–Stokes equations (2.8) on the d-dimensional torus $D = \mathbb{T}^d = [0,1)^d$ with periodic boundary conditions.

Example 2.3 (viscous and inviscid scalar conservation laws). We consider the following one-dimensional version of *viscous scalar conservation laws* as a model problem for quasilinear, convection-dominated diffusion equations:

$$\begin{cases} u_t + f(u)_x = \nu u_{xx} & \text{for all } x \in (0,1),\ t \in [0,T], \\ u(x,0) = u_0(x) & \text{for all } x \in (0,1), \\ u(0,t) = u(1,t) \equiv 0 & \text{for all } t \in [0,T]. \end{cases} \tag{2.9}$$

Here, $u_0 \in C^k([0,1])$, for some $k \geq 1$, is the initial data and we consider zero Dirichlet boundary conditions. Note that $0 < \nu \ll 1$ is the viscosity coefficient. The flux function is denoted by $f \in C^k(\mathbb{R};\mathbb{R})$. One can follow standard textbooks such as that of Godlewski and Raviart (1991) to conclude that as long as $\nu > 0$, there exists a classical solution $u \in C^k([0,T) \times [0,1])$ of the viscous scalar conservation law (2.9).

It is often interesting to set $\nu = 0$ and consider the *inviscid scalar conservation law*

$$u_t + f(u)_x = 0. \tag{2.10}$$

In this case, the solution might no longer be continuous, as it can develop shocks, even for smooth initial data (Holden and Risebro 2015).

Example 2.4 (Poisson's equation). Finally, we consider a prototypical elliptic PDE. Let $\Omega \subset \mathbb{R}^d$ be a bounded open set that satisfies the exterior sphere condition, let $f \in C^1(\Omega) \cap L^\infty(\Omega)$ and let $g \in C(\partial\Omega)$. Then there exists $u \in C^2(\Omega) \cap C(\overline{\Omega})$ that satisfies Poisson's equation:

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega. \end{cases} \tag{2.11}$$

Moreover, if f is smooth then u is smooth in Ω.

2.1.2. Inverse problems

Secondly, we consider settings where we do not have complete information on the inputs to the PDE (2.1) (e.g. initial data, boundary conditions and parameter coefficients). As a result, the forward problem cannot be solved uniquely. However, when we have access to (noisy) data for (observables of) the underlying solution, we can attempt to determine the unknown inputs of the PDEs and consequently its solution. Problems with the aim of recovering the PDE input based on data are referred to as *inverse problems*.

We will mainly consider one kind of inverse problem, namely the *unique continuation problem* (for time-dependent problems also known as the *data assimilation problem*). Recall the abstract PDE $\mathcal{L}[u] = f$ in Ω (2.1). Moreover, we adopt the assumptions (2.2) of the introduction to this subsection. The difference from the forward problem is that we now have incomplete information on the boundary conditions. As a result, a unique solution to the PDE does not exist. Instead, we assume

that we have access to (possibly noisy) measurements of a certain observable

$$\Psi(u) = g \quad \text{in } \Omega', \tag{2.12}$$

where $\Omega' \subset \Omega$ is the observation domain, g is the measured data, and $\Psi\colon X^* \to Z^*$, where Z^* is some Banach space. We make the assumption that

$$\begin{aligned} &\|\Psi(u)\|_{Z^*} < +\infty \quad \text{for all } u \in X^*, \text{ with } \|u\|_{X^*} < +\infty, \\ &\|g\|_{Z^*} < +\infty. \end{aligned} \tag{2.13}$$

If Ω' includes the boundary of Ω, meaning that we have measurements on the boundary, then it is again feasible to just use the standard framework for forward problems. However, this is often not possible. In heat transfer problems, for example, the maximum temperature will be reached at the boundary. As a result, it might be too hot near the boundary to place a sensor there and we will only have measurements that are far enough away from the boundary. Fortunately, we can adapt the physics-informed framework to retrieve an approximation to u based on f and g.

In analogy with operator learning for forward problems, we can also attempt to entirely learn the *inverse operator*

$$\mathcal{G}^{-1}\colon Z^* \to X^*\colon \Psi[u] = g \mapsto \mathcal{G}^{-1}[g] = v, \tag{2.14}$$

where v could for instance be the boundary conditions, the parameter function a or the input/source function f.

Example 2.5 (heat equation). We revisit the heat equation $u_t - \Delta u = f$ with zero Dirichlet boundary conditions and $f \in L^2(\Omega)$, where $\Omega = D \times (0,T)$. We assume that $u \in H^1((0,T); H^{-1}(D)) \cap L^2((0,T); H^1(D))$ will solve the heat equation in a weak sense. The heat equation would have been well-posed if the initial conditions, i.e. $u_0 = u(x,0)$, had been specified. Recall that the aim of the data assimilation problem is to infer the initial conditions and the entire solution field at later times from some measurements of the solution in time. To model this, we consider the following *observables*:

$$\Psi[u] := u = g \quad \text{for all } (x,t) \in D' \times (0,T) =: \Omega', \tag{2.15}$$

for some open, simply connected observation domain $D' \subset D$ and for $g \in L^2(\Omega')$. Thus, solving the data assimilation problem for the heat equation amounts to finding the solution u of the heat equation in the whole space–time domain $\Omega = D \times (0,T)$, given data on the observation subdomain $\Omega' = D' \times (0,T)$.

Example 2.6 (Poisson equation). We revisit the Poisson equation (Example 2.4), but now with data on a subdomain of Ω:

$$-\Delta u = f \quad \text{in } \Omega \subset \mathbb{R}^d, \quad u|_{\Omega'} = g \quad \text{in } \Omega' \subset \Omega. \tag{2.16}$$

In this case the observable Ψ introduced in (2.12) is the identity $\Psi[u] = u$.

Example 2.7 (Stokes equation). Finally, we consider a simplified version of the Navier–Stokes equations, namely the stationary Stokes equation. Let $D \subset \mathbb{R}^d$ be an open, bounded, simply connected set with smooth boundary. We consider the Stokes equations as a model of stationary, highly viscous fluid:

$$\begin{cases} \Delta u + \nabla p = f & \text{for all } x \in D, \\ \operatorname{div}(u) = f_d & \text{for all } x \in D. \end{cases} \tag{2.17}$$

Here, $u\colon D \to \mathbb{R}^d$ is the velocity field, $p\colon D \to \mathbb{R}$ is the pressure and $f\colon D \to \mathbb{R}^d$, $f_d\colon D \to \mathbb{R}$ are source terms.

Note that the Stokes equation (2.17) is not well-posed as we are not providing any boundary conditions. In the corresponding data assimilation problem (Burman and Hansbo 2018 and references therein), we provide the following *data*:

$$\Psi[u] \coloneqq u = g \quad \text{for all } x \in D', \tag{2.18}$$

for some open, simply connected set $D' \subset D$. Thus the data assimilation inverse problem for the Stokes equation amounts to inferring the velocity field u (and the pressure p), given f, f_d and g.

2.2. *Ansatz spaces*

We give an overview of different classes of parametric functions $\{u_\theta\}_{\theta\in\Theta}$ that are often used to approximate the true solution u of the PDE (2.1). We let n be the number of (real) parameters, such that the parameter space Θ is a subset of $\mathbb{R}^n$.

2.2.1. *Linear models*

We first consider models that linearly depend on the parameter θ, i.e. models that are a linear combination of a fixed set of functions $\{\phi_i\colon \Omega \to \mathbb{R}\}_{1\le i\le n}$. For any $\theta \in \mathbb{R}^n$ the parametrized linear model u_θ is then defined as

$$u_\theta(x) = \sum_{i=1}^{n} \theta_i \phi_i(x). \tag{2.19}$$

This general model class constitutes the basis of many existing numerical methods for approximation solutions to PDEs, of which we give an overview below.

First, *spectral methods* use smooth functions that are globally defined, i.e. supported on almost all of Ω, and often form an orthogonal set; see e.g. Hesthaven, Gottlieb and Gottlieb (2007). The particular choice of functions generally depends on the geometry and boundary conditions of the considered problem: whereas trigonometric polynomials (Fourier basis) are a default choice on periodic domains, Chebyshev and Legendre polynomials might be more suitable choices on non-periodic domains. The optimal parameter vector θ^* is then determined by solving a linear system of equations. Assuming that $\mathcal{L}$ is a linear operator, this system is

given by

$$\sum_{i=1}^{n} \theta_i \int_\Omega \psi_k \mathcal{L}(\phi_i) = \int_\Omega \psi_k f \quad \text{for } 1 \le k \le K, \tag{2.20}$$

where either $\psi_k = \phi_k$, resulting in the *Galerkin method*, or $\psi_k = \delta_{x_k}$ with $x_k \in \Omega$, resulting in the *collocation method*, or the more general case where the ψ_k and ϕ_k are distinct (and also not Dirac deltas), which is referred to as the *Petrov–Galerkin method*. In the collocation method, the choice of so-called collocation points $\{x_k\}_k$ depends on the functions ϕ_i. Boundary conditions are implemented through the choice of the functions ϕ_i or by adding more equations to the above linear system.

In marked contrast to the global basis functions of the spectral method stands the *finite element method* (FEM), where we consider locally defined polynomials. More precisely, we first divide Ω into (generally overlapping) subdomains Ω_i (subintervals in one dimension) of diameter at most h and then consider on each Ω_i a piecewise defined polynomial ϕ_i of degree at most p that is zero outside Ω_i. Much like with the spectral Galerkin method, we can then define a linear system of equations such as (2.20) that needs to be solved to find the optimal parameter vector θ^*. As ϕ_i is now non-smooth on Ω, we need to rewrite $\int_\Omega \phi_k \mathcal{L}(\phi_i)$ using integration by parts. Another key difference from the spectral method is that, due to the local support of the functions ϕ_i, the linear system we have to solve will be sparse and have structural properties related to the choice of grid and polynomials. The accuracy of the approximation can be improved by reducing h (h-FEM), increasing p (p-FEM), or both (hp-FEM). Alternatively, the finite element method can also be used in combination with a squared loss, resulting in the *least-squares finite element method* (Bochev and Gunzburger 2009). The finite element method is ubiquitous in scientific computing and many variants can be found.

Another choice for the functions ϕ_i is that of *radial basis functions* (RBFs); see e.g. Buhmann (2000). The key ingredient of RBFs is a radial function $\varphi\colon [0,\infty) \to \mathbb{R}$, for which it must hold for any choice of nodes $\{x_i\}_{1\le i\le n} \subset \Omega$ that both (i) the functions $x \mapsto \phi_i(x) \coloneqq \varphi(\|x - x_i\|)$ are linearly independent and (ii) the matrix A with entries $A_{ij} = \varphi(\|x_i - x_j\|)$ is non-singular. Radial functions can be globally supported (e.g. Gaussians, polyharmonic splines) or compactly supported (e.g. bump functions). The optimal parameter θ^* can then be determined in a Galerkin, collocation or least-squares fashion. RBFs are particularly advantageous as they are mesh-free (as opposed to FEM), which is useful for complex geometries, and straightforward to use in high-dimensional settings, as we can use randomly generated nodes, rather than nodes on a grid.

Finally, RBFs with randomly generated nodes can be seen as an example of a more general *random feature model* (RFM), where the functions ϕ_i are randomly drawn from a chosen distribution over a function space (e.g. Rahimi and Recht 2008). In practice we often choose the function space to be parametric itself, such that generating ϕ_i reduces to randomly drawing finite vectors α_i and setting $\phi_i(x) \coloneqq \phi(x;\alpha_i)$. Random feature models where ϕ_i is a neural network with

randomly initialized weights and biases are closely connected to neural networks of which only the outer layer is trained. When the network is large then the latter model is often referred to as an *extreme learning machine* (ELM) (Huang, Zhu and Siew 2006).

2.2.2. Nonlinear models

Neural networks. A first, but very central, example of parametric models that depend nonlinearly on their parameters is that of *neural networks*. The simplest form of a neural network is a *multilayer perceptron* (MLP). MLPs get their name from their definition as a concatenation of affine maps and activation functions. As a consequence of this structure, the computational graph of an MLP is organized in layers. At the kth layer, the activation function is applied component-wise to obtain the value

$$z^k = \sigma(C_{k-1}(z^{k-1})) = \sigma(W_{k-1}z^{k-1} + b_{k-1}), \tag{2.21}$$

where z^0 is the input of the network, $W_k \in \mathbb{R}^{d_{k+1}\times d_k}$, $b_k \in \mathbb{R}^{d_{k+1}}$ and $C_k : \mathbb{R}^{d_k} \to \mathbb{R}^{d_{k+1}} : x \mapsto W_k x + b_k$ for $d_k \in \mathbb{N}$ is an affine map. A multilayer perceptron (MLP) with L layers, activation function σ, output function σ_o, input dimension d_{in} and output dimension d_{out} is then a function $f_\theta : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ of the form

$$f_\theta(x) = (\sigma_o \circ C_L \circ \sigma \circ \cdots \circ \sigma \circ C_0)(x) \quad \text{for all } x \in \mathbb{R}^{d_{\text{in}}}, \tag{2.22}$$

where σ is applied element-wise and $\theta = ((W_0, b_0), \ldots, (W_L, b_L))$. The dimension d_k of each layer is also called the *width* of that layer. The width of the MLP is defined as $\max_k d_k$. Furthermore, the matrices W_k are called the *weights* of the MLP and the vectors b_k are called the *biases* of the MLP. Together, they constitute the (trainable) parameters of the MLP, denoted by $\theta = ((W_0, b_0), \ldots, (W_L, b_L))$. Finally, an output function σ_o can be employed to increase the interpretability of the output, but is often equal to the identity in the setting of function approximation.

Remark 2.8. In some texts, the number of layers L corresponds to the number of affine maps in the definition of f_θ. Of these L layers, the first $L-1$ layers are called hidden layers. An L-layer network in this alternative definition hence corresponds to an $(L-1)$-layer network in our definition.

Many of the properties of a neural network depend on the choice of activation function. We discuss the most common activation functions below.

- The *Heaviside* function is the activation that was used in the McCulloch–Pitts neuron and is defined as

$$H(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0. \end{cases} \tag{2.23}$$

  Because the gradient is zero wherever it exists, it is not possible to train the network using a gradient-based approach. For this reason, the Heaviside function is no longer used.

- The *logistic* function is defined as

$$\rho(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{2.24}$$

and can be thought of as a smooth approximation of the Heaviside function as $\rho(\lambda x) \to H(x)$ for $\lambda \to \infty$ for almost every x. It is a so-called sigmoidal function, meaning that the function is smooth, monotonic and has horizontal asymptotes for $x \to \pm\infty$. Many early approximation-theoretical results were stated for neural networks with a sigmoidal activation function.

- The hyperbolic tangent or *tanh* function, defined by

$$\sigma(x) = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}}, \tag{2.25}$$

is another smooth, sigmoidal activation function that is symmetric, unlike the logistic function. One problem with the tanh (and all other sigmoidal functions) is that the gradient vanishes away from zero, which might be problematic when using a gradient-based optimization approach.

- The rectified linear unit, rectifier function or *ReLU* function, defined as

$$\mathrm{ReLU}(x) = \max\{x, 0\}, \tag{2.26}$$

is a very popular activation function. It is easy to compute, scale-invariant and reduces the vanishing gradient problem that sigmoidal functions have. One common issue with ReLU networks is that of 'dying neurons', caused by the fact that the gradient of ReLU is zero for $x < 0$. This issue can however be overcome by using the so-called *leaky ReLU* function

$$\mathrm{ReLU}_{\nu}(x) = \max\{x, -\nu x\}, \tag{2.27}$$

where $\nu > 0$ is small. Other (smooth) adaptations are the Gaussian error linear unit, the sigmoid linear unit, exponential linear unit and softplus function.

Neural operators. In order to approximate operators, we need to allow the input and output of the learning architecture to be infinite-dimensional. A possible approach is to use *deep operator networks* (DeepONets), as proposed in Chen and Chen (1995) and Lu, Jin, Pang, Zhang and Karniadakis (2021*a*). Given m fixed sensor locations $\{x_j\}_{j=1}^{m} \subset \Omega$ and the corresponding *sensor values* $\{v(x_j)\}_{j=1}^{m}$ as input, a DeepONet can be formulated in terms of two (deep) neural networks: a *branch net* $\boldsymbol{\beta}\colon \mathbb{R}^m \to \mathbb{R}^p$ and a *trunk net* $\boldsymbol{\tau}\colon \Omega \to \mathbb{R}^{p+1}$. The branch and trunk nets are then combined to approximate the underlying nonlinear operator as the following *DeepONet*:

$$\mathcal{G}_\theta\colon \mathcal{X} \to L^2(\Omega), \quad \text{with} \quad \mathcal{G}_\theta[v](y) = \tau_0(y) + \sum_{k=1}^{p} \beta_k(v)\tau_k(y). \tag{2.28}$$

One alternative to DeepONets that also allows us to learn maps between infinite-dimensional spaces is that of *neural operators* (Li *et al.* 2020). Neural operators

are inspired by the fact that under some general conditions the solution u to the PDE (2.1) can be represented in the form

$$u(x) = \int_D G(x, y) f(y) \, \mathrm{d}y, \tag{2.29}$$

where $G : \Omega \times \Omega \to \mathbb{R}$ is the Green's function. This kernel representation led Li *et al.* (2020) to propose replacing the formula for a hidden layer of a standard neural network with the following operator:

$$\mathcal{L}_\ell(v)(x) = \sigma\left(W_\ell v(x) + \int_D \kappa_\phi(x, y, a(x), a(y)) v(y) \, \mathrm{d}\mu_x(\mathrm{d}y) \right), \tag{2.30}$$

where W_ℓ and ϕ are to be learned from the data, κ_ϕ is a kernel and μ_x is a Borel measure for every $x \in D$. Li *et al.* (2020) modelled the kernel κ_ϕ as a neural network. More generally, neural operators are mappings of the form (Li *et al.* 2020, Kovachki, Lanthaler and Mishra 2021)

$$\mathcal{N} : \mathcal{X} \to \mathcal{Y} : u \mapsto \mathcal{Q} \circ \mathcal{L}_L \circ \mathcal{L}_{L-1} \circ \cdots \circ \mathcal{L}_1 \circ \mathcal{R}(u), \tag{2.31}$$

for a given depth $L \in \mathbb{N}$, where $\mathcal{R} : \mathcal{X}(D; \mathbb{R}^{d_\mathcal{X}}) \to \mathcal{U}(D; \mathbb{R}^{d_v})$, $d_v \geq d_u$, is a *lifting* operator (acting locally), of the form

$$\mathcal{R}(a)(x) = Ra(x), \quad R \in \mathbb{R}^{d_v \times d_\mathcal{X}}, \tag{2.32}$$

and $\mathcal{Q} : \mathcal{U}(D; \mathbb{R}^{d_v}) \to \mathcal{Y}(D; \mathbb{R}^{d_\mathcal{Y}})$ is a local *projection* operator, of the form

$$\mathcal{Q}(v)(x) = Qv(x), \quad Q \in \mathbb{R}^{d_\mathcal{Y} \times d_v}. \tag{2.33}$$

In analogy with canonical finite-dimensional neural networks, the layers $\mathcal{L}_1, \ldots, \mathcal{L}_L$ are nonlinear operator layers, $\mathcal{L}_\ell : \mathcal{U}(D; \mathbb{R}^{d_v}) \to \mathcal{U}(D; \mathbb{R}^{d_v})$, $v \mapsto \mathcal{L}_\ell(v)$, which we assume to be of the form

$$\mathcal{L}_\ell(v)(x) = \sigma(W_\ell v(x) + b_\ell(x) + (\mathcal{K}(a; \theta_\ell)v)(x)) \quad \text{for all } x \in D. \tag{2.34}$$

Here, the weight matrix $W_\ell \in \mathbb{R}^{d_v \times d_v}$ and bias $b_\ell(x) \in \mathcal{U}(D; \mathbb{R}^{d_v})$ define an affine pointwise mapping $W_\ell v(x) + b_\ell(x)$. We conclude the definition of a neural operator by stating that $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonlinear activation function that is applied component-wise, and $\mathcal{K}$ is a *non-local* linear operator,

$$\mathcal{K} : \mathcal{X} \times \Theta \to L(\mathcal{U}(D; \mathbb{R}^{d_v}), \mathcal{U}(D; \mathbb{R}^{d_v})), \tag{2.35}$$

that maps the input field a and a parameter $\theta \in \Theta$ in the parameter set Θ to a bounded linear operator $\mathcal{K}(a, \theta) : \mathcal{U}(D; \mathbb{R}^{d_v}) \to \mathcal{U}(D; \mathbb{R}^{d_v})$. When we define the linear operators $\mathcal{K}(a, \theta)$ through an integral kernel, then (2.34) reduces again to (2.30). *Fourier neural operators* (FNOs) (Li *et al.* 2021) are special cases of such general neural operators in which this integral kernel corresponds to a convolution kernel, which on the periodic domain $\mathbb{T}^d$ leads to nonlinear layers $\mathcal{L}_\ell$ of the form

$$\mathcal{L}_\ell(v)(x) = \sigma(W_\ell v(x) + b_\ell(x) + \mathcal{F}^{-1}(P_\ell(k) \cdot \mathcal{F}(v)(k))(x)). \tag{2.36}$$

In practice, however, this definition cannot be used, as evaluating the Fourier transform requires the exact computation of an integral. The practical implementation (i.e. discretization) of an FNO maps from and to the space of trigonometric polynomials of degree at most $N \in \mathbb{N}$, denoted by L^2_N, and can be identified with a finite-dimensional mapping that is a composition of affine maps and nonlinear layers of the form

$$\mathfrak{L}_\ell(z)_j = \sigma\big(W_\ell v_j + b_{\ell,j}\mathcal{F}_N^{-1}(P_\ell(k)\cdot\mathcal{F}_N(z)(k)_j)\big), \tag{2.37}$$

where the $P_\ell(k)$ are coefficients that define a non-local convolution operator via the discrete Fourier transform $\mathcal{F}_N$; see Kovachki *et al.* (2021).

It has been observed that the proposed operator learning models above may not behave as operators when implemented on a computer, questioning the very essence of what operator learning should be. Bartolucci *et al.* (2023) contend that in addition to defining the operator at the continuous level, some form of continuous–discrete equivalence is necessary for an architecture to genuinely learn the underlying operator rather than just discretizations of it. To this end, they introduce the unifying mathematical framework of *representation-equivalent neural operators* (ReNOs) to ensure that operations at the continuous and discrete level are equivalent. Lack of this equivalence is quantified in terms of aliasing errors.

Gaussian processes. Deep neural networks and their operator versions are only one possible nonlinear surrogate model for the true solution u. Another popular class of nonlinear models is *Gaussian process regression* (Rasmussen 2003), which belongs to a larger class of so-called Bayesian models. Gaussian process regression (GPR) relies on the assumption that u is drawn from a Gaussian measure on a suitable function space, parametrized by

$$u_\theta(x) \sim \mathrm{GP}(m_\theta(x), k_\theta(x,x')), \tag{2.38}$$

where $m_\theta(x) = \mathbb{E}[u_\theta(x)]$ is the mean, and the underlying covariance function is given by $k_\theta(x,x') = \mathbb{E}[(u_\theta(x) - m_\theta(x))(u_\theta(x') - m_\theta(x'))]$.

Popular choices for the covariance function in (2.38) are the squared exponential kernel (which is a radial function as for RBFs) and Matérn covariance functions:

$$k_{\mathrm{SE}}(x,x') = \exp\left(-\frac{\|x-x'\|^2}{2\ell^2}\right), \tag{2.39}$$

$$k_{\mathrm{Matern}}(x,x') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\frac{\|x-x'\|}{\ell}\right)^{\nu} K_\nu\left(\sqrt{2\nu}\frac{\|x-x'\|}{\ell}\right). \tag{2.40}$$

Here $\|\cdot\|$ denotes the standard Euclidean norm, K_ν is the modified Bessel function of the second kind and ℓ the *characteristic length*, describing the length scale of the correlations between the points x and x'. The parameter vector then consists of the hyperparameters of m_θ and k_θ, such as ν and ℓ.

2.3. Physics-informed residuals and loss functions

With a chosen model class in place, the key to physics-informed learning is to find a *physics-informed loss functional* $\mathcal{E}_{\text{PIL}}[\cdot]$, independent of u, such that when it is minimized over the class of models $\{u_\theta : \theta \in \Theta\}$, the minimizer is a good approximation of the PDE solution u. In other words, we must find a functional $\mathcal{E}_{\text{PIL}}[\cdot]$ such that for $\theta^* = \text{argmin}_\theta \mathcal{E}_{\text{PIL}}[u_\theta]$, the minimizer u_{θ^*} is a good approximation of u. We discuss three alternative formulations for a PDE, i.e. definitions of what it means to solve a PDE, all leading to different functionals and their corresponding *loss functions*:

- the classical formulation of the PDE, as in (2.1), in which a pointwise condition on a function and its derivatives is stated;
- the weak formulation of the PDE, which is a more general (weaker) condition than the classical formulation, often obtained by multiplying the classical formulation by a smooth test function and performing integration by parts;
- the variational formulation of the PDE, where the PDE solution can be defined as the minimizer of a functional.

Note that not all formulations are available for all PDEs.

2.3.1. Classical formulation

Physics-informed learning is based on the elementary observation that for the true solution of the PDE u it holds that $\mathcal{L}(u) = f$ and $\mathcal{B}(u) = g$, and the subsequent hope that if a model u_θ satisfies that $\mathcal{L}(u_\theta) - f \approx 0$ and $\mathcal{B}(u_\theta) - g \approx 0$, that then also $u \approx u_\theta$. To this end, a number of *residuals* are introduced. First and foremost, the *PDE residual* (or *interior residual*) measures how well the model u_θ satisfies the PDE

$$\mathcal{R}_{\text{PDE}}[u_\theta](z) = \mathcal{L}[u_\theta](z) - f(z) \quad \text{for all } z \in \Omega, \tag{2.41}$$

where we recall that $\Omega = D$ and $z = x$ for a time-independent PDE and $\Omega = D \times [0, T]$ and $z = (x, t)$ for a time-dependent PDE. In the first case, the boundary operator $\mathcal{B}$ in (2.1) uniquely refers to the spatial boundary conditions, such that $\mathcal{B}(u) = g$ corresponds to $D_x^k u = g$. The corresponding *spatial boundary residual* is then given by

$$\mathcal{R}_s[u_\theta](x) = D_x^k u_\theta(x) - g(x) \quad \text{for all } x \in \partial D. \tag{2.42}$$

For time-dependent PDEs the boundary operator $\mathcal{B}$ generally also prescribes the initial condition, i.e. $u(x, 0) = u_0(x)$ for all $x \in D$ for a given function u_0. This condition gives rise to the *temporal boundary residual*

$$\mathcal{R}_t[u_\theta](x) = u_\theta(x, 0) - u_0(x) \quad \text{for all } x \in D. \tag{2.43}$$

The physics-informed learning framework hence dictates that we should simultaneously minimize the residuals $\mathcal{R}_{\text{PDE}}[u_\theta]$ and $\mathcal{R}_s[u_\theta]$, and additionally $\mathcal{R}_t[u_\theta]$ for

a time-dependent PDE. This is usually translated as the minimization problem

$$\min_{\theta \in \Theta} \mathcal{E}_{\mathrm{PIL}}[u_\theta]^2, \tag{2.44}$$

where for a time-independent PDE the *physics-informed loss* $\mathcal{E}_{\mathrm{PIL}}$ is given by

$$\mathcal{E}_{\mathrm{PIL}}[u_\theta]^2 = \int_{D\times[0,T]} \mathcal{R}_{\mathrm{PDE}}[u_\theta]^2 + \lambda_s \int_{\partial D} \mathcal{R}_s[u_\theta]^2, \tag{2.45}$$

where $\lambda_s > 0$ is a weighting hyperparameter, and for a time-dependent PDE the quantity $\mathcal{E}_{\mathrm{PIL}}$ is given by

$$\mathcal{E}_{\mathrm{PIL}}[u_\theta]^2 = \int_{D\times[0,T]} \mathcal{R}_{\mathrm{PDE}}[u_\theta]^2 + \lambda_s \int_{\partial D\times[0,T]} \mathcal{R}_s[u_\theta]^2 + \lambda_t \int_0^T \mathcal{R}_t[u_\theta]^2, \tag{2.46}$$

where $\lambda_s, \lambda_t > 0$ are weighting hyperparameters. Since the boundary conditions (and initial condition) are enforced through an additional term in the loss function, we allow the possibility that the final model will only approximately satisfy the boundary conditions. This is generally referred to as having *soft boundary conditions*. On the other hand, if we choose the model class in such a way that all u_θ exactly satisfy the boundary conditions ($\mathcal{B}[u_\theta] = g$), we have implemented *hard boundary conditions*.

Remark 2.9. The residuals and loss functions defined above can readily be extended to systems of PDEs and more complicated boundary conditions. If the system of PDEs is given by $\mathcal{L}^{(i)}[u] = f^{(i)}$, then one can readily define multiple PDE residuals $\mathcal{R}^{(i)}_{\mathrm{PDE}}$ and consider the sum of integrals of the square PDE residuals $\sum_i \int (\mathcal{R}^{(i)}_{\mathrm{PDE}})^2$ in the physics-informed loss. The generalization to more complicated boundary conditions, including piecewise defined boundary conditions, proceeds in a completely analogous manner.

The minimization problem (2.44), solely based on the physics-informed loss functions defined in (2.45) and (2.46), is the most basic version of physics-informed learning and can be extended by adding additional terms to the objective in (2.44). For example, in some settings one might have access to measurements $\Psi(u) = g$ on a subdomain Ω', where Ψ is a so-called *observable*. This case is particularly relevant for inverse problems (Section 2.1.2). In this case it makes sense to add a *data term* to the optimization objective, defined in terms of a *data residual* $\mathcal{R}_d$:

$$\mathcal{R}_d[u_\theta](x) = \Psi[u](x) - g(x) \quad \text{for all } x \in \Omega'. \tag{2.47}$$

Note that the boundary residual can be seen as a special case of the data residual with $\Psi = \mathcal{B}$ and $\Omega' = \partial\Omega$. Additionally, it might happen in practice that we add a *parameter regularization term*, often the L^q-norm of the parameters θ, to the loss function to either aid the minimization process or improve the (generalization) properties of the final model. A generalization of the physics-informed

minimization problem (2.44) could therefore be

$$\min_{\theta\in\Theta}\left(\mathcal{E}_{\text{PIL}}[u_\theta]^2+\lambda_d\int_{\Omega'}\mathcal{R}_d[u_\theta]^2+\lambda_r\|\theta\|_q^q\right). \tag{2.48}$$

2.3.2. Weak formulation

In the previous subsection we considered PDE residuals and physics-informed loss functions that are based on the strong formulation of the PDE. However, for various classes of PDEs there might not be a smooth classical solution u for which it holds in every $x \in \Omega$ that $\mathcal{L}[u](x) = f(x)$. Fortunately, one can sometimes generalize the definition of 'a solution to a PDE' in a precise manner such that there are functions that satisfy this generalized definition after all. Such functions are called *weak solutions*. They are said to satisfy the *weak formulation* of the PDE, which can be obtained by multiplying the strong formulation of the PDE (2.1) with a smooth and compactly supported test function ψ and then performing integration by parts:

$$\int_\Omega u\mathcal{L}^*[\psi] = \int_\Omega f\psi, \tag{2.49}$$

where $\mathcal{L}^*$ is the (formal) adjoint of $\mathcal{L}$ (if it exists). Consequently, a function u is said to be a weak solution if it satisfies (2.49) for all (compactly supported) test functions ψ in some predefined set of functions. Note that u no longer needs to be differentiable but merely integrable.

Weak formulations such as (2.49) form the basis of many classical numerical methods, in particular the finite element method (FEM: Section 2.2.1). Often, however, there is no need to perform integration by parts until u is completely free of derivatives. In FEM, the weak formulation for second-order elliptic equations is generally obtained by performing integration by parts only once. For the Laplacian $\mathcal{L} = \Delta$ we then have

$$\int_\Omega \Delta u\psi = -\int_\Omega \nabla u\cdot\nabla\psi$$

(given suitable boundary conditions), which is a formulation with many useful properties in practice.

Other than using spectral methods (as in Section 2.2.1), there are now different avenues one can follow to obtain a *weak physics-informed loss function* based on (2.49). The first method draws inspiration from the Petrov–Galerkin method (Section 2.2.1) and considers a finite set of K test functions ψ_k for which (2.49) must hold. The corresponding loss function is then defined as

$$\mathcal{E}_{\text{VPIL}}[u_\theta]^2 = \sum_{k=1}^{K}\left(\int_\Omega u_\theta\mathcal{L}^*[\psi_k]-\int_\Omega f\psi_k\right)^2, \tag{2.50}$$

where additional terms can be added to take care of any boundary conditions. This method was first introduced for neural networks under the name *variational*

PINNs (VPINNs) (Kharazmi *et al.* 2019) and used sine and cosine functions or polynomials as test functions. In analogy with hp-FEM, the method was later adapted to use localized test functions, leading to hp-VPINNs (Kharazmi, Zhang and Karniadakis 2021). In this case the model u_θ is of limited regularity and might not be sufficiently many times differentiable for $\mathcal{L}[u_\theta]$ to be well-defined.

A second variant of a weak physics-informed loss function can be obtained by replacing ψ with a parametric model ψ_ϑ. In particular, we choose the ψ_ϑ to be the function that maximizes the squared weak residual, leading to the loss function

$$\mathcal{E}_{\text{wPIL}}[u_\theta]^2 = \max_\vartheta \left(\int_\Omega u_\theta \mathcal{L}^*[\psi_\vartheta] - \int_\Omega f\psi_\vartheta \right)^2, \tag{2.51}$$

where again additional terms can be added to take care of any boundary conditions. Using $\mathcal{E}_{\text{wPIL}}$ has the advantage over $\mathcal{E}_{\text{VPIL}}$ that we do not need a basis of test functions $\{\psi_k\}_k$, which might be inconvenient in high-dimensional settings, and settings with complex geometries. On the other hand, minimizing $\mathcal{E}_{\text{wPIL}}$ corresponds to solving a min-max optimization problem where the maximum and minimum are taken with respect to neural network training parameters, which is considerably more challenging than a minimization problem.

The weak physics-informed loss $\mathcal{E}_{\text{wPIL}}$ was originally proposed to approximate scalar conservation laws with neural networks under the name *weak PINNs* (wPINNs) (De Ryck *et al.* 2024*c*). As there might be infinitely many weak solutions for a given scalar conservation law, we need to impose further requirements other than (2.49) to guarantee that a scalar conservation law has a unique solution. This additional challenge is discussed in the next example.

Example 2.10 (wPINNs for scalar conservation laws). We revisit Example 2.3 and recall that weak solutions of scalar conservation laws need not be unique (Holden and Risebro 2015). To recover uniqueness, we need to impose additional admissibility criteria in the form of *entropy conditions*. To this end, we consider the so-called *Kruzkhov entropy functions*, given by $|u - c|$ for any $c \in \mathbb{R}$, and the resulting entropy flux functions

$$Q : \mathbb{R}^2 \to \mathbb{R} : (u, c) \mapsto \operatorname{sgn}(u - c)\,(f(u) - f(c)). \tag{2.52}$$

We then say that a function $u \in L^\infty(\mathbb{R} \times \mathbb{R}_+)$ is an *entropy solution* of (2.10) with initial data $u_0 \in L^\infty(\mathbb{R})$ if u is a weak solution of (2.10) and if u satisfies

$$\partial_t |u - c| + \partial_x Q[u; c] \leq 0, \tag{2.53}$$

in a distributional sense, or more precisely

$$\begin{aligned} &\int_0^T \int_{\mathbb{R}} (|u - c|\varphi_t + Q[u; c]\varphi_x)\,\mathrm{d}x\,\mathrm{d}t \\ &\quad - \int_{\mathbb{R}} (|u(x, T) - c|\phi(x, T) - |u_0(x) - c|\phi(x, 0))\,\mathrm{d}x \geq 0 \end{aligned} \tag{2.54}$$

for all $\phi \in C_c^1(\mathbb{R} \times \mathbb{R}_+)$, $c \in \mathbb{R}$ and $T > 0$. It holds that entropy solutions are unique and continuous in time. For more details of classical, weak and entropy solutions for hyperbolic PDEs, we refer the reader to LeVeque (2002) and Holden and Risebro (2015).

This definition inspired De Ryck *et al.* (2024*c*) to define the *Kruzkhov entropy residual*

$$\mathcal{R}(v, \phi, c) := -\int_D \int_{[0,T]} (|v(x,t) - c|\partial_t \phi(x,t) + Q[v(x,t); c]\partial_x \phi(x,t)) \,\mathrm{d}x \,\mathrm{d}t, \tag{2.55}$$

based on which the following weak physics-informed loss is defined:

$$\mathcal{E}_{\text{wPIL}}[u_\theta] = \max_{\vartheta, c} \mathcal{R}(u_\theta, \phi_\vartheta, c), \tag{2.56}$$

where additional terms need to be added to take care of initial and boundary conditions (De Ryck *et al.* 2024*c*, Section 2.4).

2.3.3. Variational formulation

Finally, we discuss a third physics-informed loss function that can be formulated for *variational problems*. Variational problems are PDEs (2.1) where the differential operator can (at least formally) be written as the derivative of an *energy functional* I, in the sense that

$$\mathcal{L}[u] - f = I'[u] = 0. \tag{2.57}$$

The *calculus of variations* thus allows us to replace the problem of solving the PDE (2.1) to finding the minimum of the functional I, which might be much easier (Evans 2022). A first instance of this observation is known as *Dirichlet's principle*.

Example 2.11 (Dirichlet's principle). Let $\Omega \subset \mathbb{R}^d$ be open and bounded, and define $\mathcal{A} = \{w \in C^2(\Omega) \colon w = g \text{ on } \partial\Omega\}$. If we consider the functional $I(w) = \frac{1}{2}\int_\Omega |\nabla w|^2$, then for $u \in \mathcal{A}$ we have

$$I[u] = \min_{w \in \mathcal{A}} I[w] \quad \Longleftrightarrow \quad \begin{cases} \Delta u = 0, & x \in \Omega, \\ u = g, & x \in \partial\Omega. \end{cases} \tag{2.58}$$

In other words, u minimizes the functional I over $\mathcal{A}$ if and only if u is a solution to Laplace's equation on Ω with Dirichlet boundary conditions g.

Dirichlet's principle can be generalized to functionals I of the form

$$I[w] = \int_\Omega L(\nabla w(x), w(x), x) \,\mathrm{d}x, \tag{2.59}$$

that are the integrals of a so-called *Lagrangian* L,

$$L \colon \mathbb{R}^d \times \mathbb{R} \times \Omega \to \mathbb{R} \colon (p, z, x) \mapsto L(p, z, x), \tag{2.60}$$

where $w \colon \overline{\Omega} \to \mathbb{R}$ is a smooth function that satisfies some prescribed boundary

conditions. One can then verify that any smooth minimizer u of $I[\cdot]$ must satisfy the nonlinear PDE

$$-\sum_{i=1}^{d} \partial_{x_i}(\partial_{p_i} L(\nabla u, u, x)) + \partial_z L(\nabla u, u, x) = 0, \quad x \in \Omega. \tag{2.61}$$

This equation is the *Euler–Lagrange equation* associated with the energy functional I from (2.59). Revisiting Example 2.11, one can verify that Laplace's equation is the Euler–Lagrange equation associated with the energy functional based on the Lagrangian $L(p, z, x) = \frac{1}{2}|p|^2$. Dirichlet's principle can also be generalized further.

Example 2.12 (generalized Dirichlet's principle). Let $a^{ij} : \Omega \to \mathbb{R}$ be functions with $a^{ij} = a^{ji}$ and consider the elliptic PDE

$$-\sum_{i,j=1}^{d} \partial_{x_i}(a^{ij} \partial_{x_j} u) = f, \quad x \in \Omega. \tag{2.62}$$

The above PDE is the Euler–Lagrange equation associated with the functional of the form (2.59) and the Lagrangian

$$L(p, z, x) = \frac{1}{2} \sum_{i,j=1}^{d} a^{ij}(x) p_i p_j - z f(x). \tag{2.63}$$

More examples and conditions under which $I[\cdot]$ has a (unique) minimizer can be found in Evans (2022), for example.

Solving a PDE within the class of variational problems by minimizing its associated energy functional is known in the literature as the *Ritz (–Galerkin) method*, or as the *Rayleigh–Ritz method* for eigenvalue problems. More recently, E and Yu (2018) have proposed the *deep Ritz method*, which aims to minimize the energy functional over the set of deep neural networks:

$$\min_{\theta} \mathcal{E}_{\mathrm{DRM}}[u_\theta]^2, \quad \mathcal{E}_{\mathrm{DRM}}[u_\theta]^2 := I[u_\theta] + \lambda \int_{\partial\Omega} (\mathcal{B}[u_\theta] - g)^2, \tag{2.64}$$

where the second term was added because the neural networks used do not automatically satisfy the boundary conditions of the PDE (2.1).

2.4. *Quadrature rules*

All the loss functions discussed in the previous subsections are formulated as integrals in terms of the model u_θ. Very often it will not be feasible or computationally desirable to evaluate these integrals in an exact way. Instead, one can discretize the integral and approximate it using a sum. We recall some basic results on numerical quadrature rules for integral approximation.

For a mapping $g\colon \Omega \to \mathbb{R}^m$, $\Omega \subset \mathbb{R}^{\overline{d}}$, such that $g \in L^1(\Omega)$, we are interested in approximating the integral

$$\overline{g} \coloneqq \int_\Omega g(y)\,\mathrm{d}y,$$

with $\mathrm{d}y$ denoting the Lebesgue measure on Ω. In order to approximate the above integral by a quadrature rule, we need the quadrature points $y_i \in \Omega$ for $1 \leq i \leq N$, for some $N \in \mathbb{N}$ as well as weights w_i, with $w_i \in \mathbb{R}_+$. Then a quadrature is defined by

$$\mathcal{Q}_N^\Omega[g] \coloneqq \sum_{i=1}^N w_i g(y_i), \tag{2.65}$$

for weights w_i and quadrature points y_i. Depending on the choice of weights and quadrature points, as well as the regularity of g, the quadrature error is bounded by

$$|\overline{g} - \mathcal{Q}_N^\Omega[g]| \leq C_{\text{quad}} N^{-\alpha}, \tag{2.66}$$

for some $\alpha > 0$ and where C_{quad} depends on g and its properties.

As an elementary example, we mention the *midpoint rule*. For $M \in \mathbb{N}$, we partition Ω into $N \sim M^d$ cubes of edge length $1/M$ and we let $\{y_i\}_{i=1}^N$ denote the midpoints of these cubes. The formula and accuracy of the midpoint rule $\mathcal{Q}_N^\Omega$ are then given by

$$\mathcal{Q}_N^\Omega[g] \coloneqq \frac{1}{N}\sum_{i=1}^N g(y_i), \quad |\overline{g} - \mathcal{Q}_M^\Omega[g]| \leq C_g N^{-2/d}, \tag{2.67}$$

where $C_g \lesssim \|g\|_{C^2}$.

As long as the domain Ω is in reasonably low dimensions, i.e. $\overline{d} \leq 4$, we can use the midpoint rule and more general standard (composite) Gauss quadrature rules on an underlying grid. In this case, the quadrature points and weights depend on the order of the quadrature rule (Stoer and Bulirsch 2002) and the rate α depends on the regularity of the underlying integrand. On the other hand, these grid-based quadrature rules are not suitable for domains in high dimensions. For moderately high dimensions, i.e. $4 \leq \overline{d} \leq 20$, we can use low-discrepancy sequences, such as the Sobol and Halton sequences, as quadrature points. As long as the integrand g is of bounded Hardy–Krause variation, the error in (2.66) converges at a rate $(\log(N))^{\overline{d}} N^{-1}$ (Longo, Mishra, Rusch and Schwab 2021). For problems in very high dimensions, $d \gg 20$, Monte Carlo quadrature is the numerical integration method of choice. In this case, the quadrature points are randomly chosen, independent and identically distributed (with respect to a scaled Lebesgue measure).

As an example, we discuss how the physics-informed loss based on the classical formulation for a time-dependent PDE (2.46) can be discretized. As the loss function (2.46) contains integrals over three different domains ($D \times [0,T]$, $\partial D \times [0,T]$ and D), we must consider three different quadratures. We consider quadratures

(2.65) for which the weights are the inverse of the number of grid points, such as for the midpoint or Monte Carlo quadrature, and we denote the quadrature points by $\mathcal{S}_i = \{(z_i)\}_{1\le i\le N_i} \subset D \times [0,T]$, $\mathcal{S}_s = \{y_i\}_{1\le i\le N_s} \subset \partial D \times [0,T]$ and $\mathcal{S}_t = \{x_i\}_{1\le i\le N_t} \subset D$. Following machine learning terminology, we refer to the set of all quadrature points $\mathcal{S} = (\mathcal{S}_i, \mathcal{S}_s, \mathcal{S}_t)$ as the *training set*. The resulting loss function $\mathcal{J}$ now depends not only on u_θ but also on $\mathcal{S}$, and is given by

$$\begin{aligned}\mathcal{J}(\theta,\mathcal{S}) &= \mathcal{Q}_{N_i}^{D\times[0,T]}[\mathcal{R}_{\mathrm{PDE}}^2] + \lambda_s \mathcal{Q}_{N_s}^{\partial D\times[0,T]}[\mathcal{R}_s^2] + \lambda_t \mathcal{Q}_{N_t}^{D}[\mathcal{R}_t^2] \\ &= \frac{1}{N_i}\sum_{i=1}^{N_i}(\mathcal{L}[u_\theta](z_i) - f(z_i))^2 + \frac{\lambda_s}{N_s}\sum_{i=1}^{N_s}(u_\theta(y_i) - u(y_i))^2 \\ &\quad + \frac{\lambda_t}{N_t}\sum_{i=1}^{N_t}(u_\theta(x_i,0) - u(x_i,0))^2. \end{aligned} \tag{2.68}$$

For any of the other loss functions defined in Section 2.3, a similar discretization can be defined. In practice, physics-informed learning thus comes down to solving the minimization problem given by

$$\theta^*(\mathcal{S}) \coloneqq \operatorname*{argmin}_{\theta\in\Theta} \mathcal{J}(\theta,\mathcal{S}). \tag{2.69}$$

The final optimized or *trained* model is then given by $u^* \coloneqq u_{\theta^*(\mathcal{S})}$.

2.5. *Optimization*

Since Θ is high-dimensional and the map $\theta \mapsto \mathcal{J}(\theta,\mathcal{S})$ is generally non-convex, solving the minimization problem (2.69) can be very challenging. Fortunately, the loss function $\mathcal{J}$ is almost everywhere differentiable, so gradient-based iterative optimization methods can be used.

The simplest example of such an algorithm is *gradient descent*. Starting from an initial guess θ_0, the idea is to take a small step in the parameter space Θ in the direction of the steepest descent of the loss function to obtain a new guess θ_1. Note that this comes down to taking a small step in the opposite direction to the gradient evaluated in θ_0. Repeating this procedure yields the iterative formula

$$\theta_{\ell+1} = \theta_\ell - \eta_\ell \nabla_\theta \mathcal{J}(\theta_\ell,\mathcal{S}) \quad \text{for all } \ell \in \mathbb{N}, \tag{2.70}$$

where the learning rate η_ℓ controls the size of the step and is generally quite small. The gradient descent formula yields a sequence of parameters $\{\theta_\ell\}_{\ell\in\mathbb{N}}$ that converge to a local minimum of the loss function under very general conditions. Because of the non-convexity of the loss function, convergence to a global minimum cannot be ensured. Another issue lies in the computation of the gradient $\nabla_\theta \mathcal{J}(\theta_\ell,\mathcal{S})$. A simple rewriting of the definition of this gradient (up to regularization term),

$$\nabla_\theta \mathcal{J}(\theta_\ell,\mathcal{S}) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \mathcal{J}_i(\theta_\ell), \quad \text{where } \mathcal{J}_i(\theta_\ell) = \mathcal{J}(\theta_\ell, \{(x_i, f(x_i))\}), \tag{2.71}$$

reveals that we actually need to evaluate $N = |\mathcal{S}|$ gradients. As a consequence, the computation of the full gradient will be very slow and memory intensive for large training data sets.

Stochastic gradient descent (SGD) overcomes this problem by only calculating the gradient in one data point instead of the full training data set. More precisely, for every ℓ a random index i_ℓ, with $1 \leq i_\ell \leq N$, is chosen, resulting in the following update formula:

$$\theta_{\ell+1} = \theta_\ell - \eta_\ell \nabla_\theta \mathcal{J}_{i_\ell}(\theta_\ell) \quad \text{for all } \ell \in \mathbb{N}. \tag{2.72}$$

Similarly to gradient descent, stochastic gradient descent provably converges to a local minimum of the loss function under rather general conditions, although the convergence will be slower. In particular, the convergence will be noisier and there is no guarantee that $\mathcal{J}(\theta_{\ell+1}) \leq \mathcal{J}(\theta_\ell)$ for every ℓ. Because the cost of each iteration is much lower than that of gradient descent, it might still make sense to use stochastic gradient descent.

Mini-batch gradient descent tries to combine the advantages of gradient descent and SGD by evaluating the gradient of the loss function in a subset of $\mathcal{S}$ in each iteration (as opposed to the full training set for gradient descent and one training sample for SGD). More precisely, for every $\ell \in \mathbb{N}$ a subset $\mathcal{S}_\ell \subset \mathcal{S}$ of size m is chosen. These subsets are referred to as *mini-batches* and their size m as the mini-batch size. In contrast to SGD, these subsets are not entirely randomly selected in practice. Instead, we randomly partition the training set into $\lceil N/m \rceil$ mini-batches, which are then all used as a mini-batch $\mathcal{S}_\ell$ for consecutive ℓ. Such a cycle of $\lceil N/m \rceil$ iterations is called an *epoch*. After every epoch, the mini-batches are reshuffled, meaning that a new partition of the training set is drawn. In this way, every training sample is used once during every epoch. The corresponding update formula reads as

$$\theta_{\ell+1} = \theta_\ell - \eta_\ell \nabla_\theta \mathcal{J}(\theta_\ell, \mathcal{S}_\ell) \quad \text{for all } \ell \in \mathbb{N}. \tag{2.73}$$

By setting $m = N$ (resp. $m = 1$), we retrieve gradient descent (resp. SGD). For this reason, gradient descent is also often referred to as *full batch* gradient descent.

The performance of mini-batch gradient descent can be improved in many ways. One particularly popular example of such an improvement is the *ADAM* optimizer (Kingma and Ba 2015). ADAM, whose name is derived from *adaptive moment estimation*, calculates at each ℓ (exponential) moving averages of the first and second moment of the mini-batch gradient g_ℓ,

$$m_\ell = \beta_1 m_{\ell-1} + (1-\beta_1) g_\ell, \quad v_\ell = \beta_2 v_{\ell-1} + (1-\beta_2) g_\ell^2, \quad g_\ell = \nabla_\theta \mathcal{J}(\theta_\ell, \mathcal{S}_\ell), \tag{2.74}$$

where β_1 and β_2 are close to 1. However, one can calculate that these moving averages are biased estimators. This bias can be removed using the following correction:

$$\widehat{m}_\ell = \frac{m_\ell}{1-\beta_1^\ell}, \quad \widehat{v}_\ell = \frac{v_\ell}{1-\beta_2^\ell} \quad \text{for all } \ell \in \mathbb{N}, \tag{2.75}$$

where $\widehat{m}_\ell, \widehat{v}_\ell$ are unbiased estimators. One can obtain ADAM from the basic mini-batch gradient descent update formula (2.73) by replacing $\nabla_\theta \mathcal{J}(\theta_\ell, \mathcal{S}_\ell)$ with the moving average m_ℓ and by setting $\eta_\ell = \alpha/\sqrt{\widehat{v}_\ell + \epsilon}$, where $\alpha > 0$ is small and $\epsilon > 0$ is close to machine precision. This leads to the update formula

$$\theta_{\ell+1} = \theta_\ell - \alpha \frac{\widehat{m}_\ell}{\sqrt{\widehat{v}_\ell + \epsilon}} \quad \text{for all } \ell \in \mathbb{N}. \tag{2.76}$$

Compared to mini-batch gradient descent, the convergence of ADAM will be less noisy for two reasons. First, a fraction of the noise will be removed since a moving average of the gradient is used. Second, the step size of ADAM decreases if the gradient g_ℓ varies a lot, i.e. when $\widehat{v}_\ell$ is large. These design choices mean that ADAM performs well on a large number of tasks, making it one of the most popular optimizers in deep learning.

Finally, the interest in *second-order* optimization algorithms has been steadily growing over the past years. Most second-order methods used in practice are adaptations of the well-known Newton method,

$$\theta_{\ell+1} = \theta_\ell - \eta_\ell (\nabla^2_\theta \mathcal{J}(\theta, \mathcal{S}))^{-1} \nabla_\theta \mathcal{J}(\theta, \mathcal{S}) \quad \text{for all } \ell \in \mathbb{N}, \tag{2.77}$$

where $\nabla^2_\theta \mathcal{J}(\theta, \mathcal{S})$ denotes the Hessian of the loss function. The generalization to a mini-batch version is immediate. As the computation of the Hessian and its inverse is quite costly, we often use a *quasi*-Newton method that computes an approximate inverse by solving the linear system $\nabla^2_\theta \mathcal{J}(\theta, \mathcal{S}) h_\ell = \nabla_\theta \mathcal{J}(\theta, \mathcal{S})$. An example of a popular quasi-Newton method in deep learning for scientific computing is *L-BFGS* (Liu and Nocedal 1989). In many other application areas, however, first-order optimizers remain the dominant choice, for a myriad of reasons.

2.5.1. *Parameter initialization*

All iterative optimization algorithms above require an initial guess θ_0. Making this initial guess is referred to as parameter initialization or weights initialization (for neural networks). Although it is often overlooked, a bad initialization can cause a lot of problems, as will be discussed in Section 7.

Many initialization schemes have already been considered, in particular for neural networks. When the initial weights are too large, the corresponding gradients calculated by the optimization algorithm might be very large, leading to the *exploding gradient problem*. Similarly, when the initial weights are too small, the gradients might also be very close to zero, leading to the *vanishing gradient problem*. Fortunately, it is possible to choose the variance of the initial weights in such a way that the output of the network has the same order of magnitude as the input of the network. One such possible choice is *Xavier initialization* (Glorot and

Bengio 2010), meaning that the weights are initialized as

$$(W_k)_{ij} \sim N\left(0, \frac{2g^2}{d_{k-1}+d_k}\right) \quad \text{or} \quad (W_k)_{ij} \sim U\left(-g\sqrt{\frac{6}{d_{k-1}+d_k}}, g\sqrt{\frac{6}{d_{k-1}+d_k}}\right), \tag{2.78}$$

where N denotes the normal distribution, U is the uniform distribution, d_k is the output dimension of the kth layer, and the value g depends on the activation function. For the ReLU activation we set $g = \sqrt{2}$, and for the tanh activation function $g = 1$ is a valid choice. Note that this initialization has primarily been proposed with a supervised learning setting in mind, e.g. when using a square loss such as $\int_\Omega (u_\theta - u)$. For the different loss functions of Section 2.3, different initialization schemes might be better suited. This will be discussed in Section 7.

Finally, it is customary to retrain neural network with different starting values θ_0 (drawn from the same distribution) as the gradient-based optimizer might converge to a different local minimum for each θ_0. One can then choose the 'best' neural network based on some suitable criterion, or combine the different neural networks if the setting allows. Also note that this task can be performed in parallel.

2.6. *Summary of algorithm*

We summarize the physics-informed learning algorithm as follows.

(1) Choose a model class $\{u_\theta : \theta \in \Theta\}$ (Section 2.2).

(2) Choose a loss function based on the classical, weak or variational formulation of the PDE (Section 2.3).

(3) Generate a training set $\mathcal{S}$ based on a suitable quadrature rule to obtain a feasibly computable loss function $\mathcal{J}(\theta, \mathcal{S})$ (Section 2.4).

(4) Initialize the model parameters and run an optimization algorithm from Section 2.5 until an approximate local minimum $\theta^*(\mathcal{S})$ is reached. The resulting function $u^* := u_{\theta^*(\mathcal{S})}$ is the desired model for approximating the solution u of the PDE (2.1).

3. Analysis

As intuitive as the physics-informed learning framework of Section 2.6 might seem, there is *a priori* little theoretical guarantee that it will actually work well. The aim of the next sections is therefore to theoretically analyse physics-informed machine learning and to give an overview of the available mathematical guarantees.

One central element will be the development of error estimates for physics-informed machine learning. The relevant concept of error is the error emanating from approximating the solution u of (2.1) by the model $u^* = u_{\theta^*(\mathcal{S})}$:

$$\mathcal{E}(\theta) := \|u - u_\theta\|_X, \quad \mathcal{E}^* := \mathcal{E}(\theta^*(\mathcal{S})). \tag{3.1}$$

In general, we will choose $\|\cdot\|_X = \|\cdot\|_{L^2(\Omega)}$. Note that it is not possible to compute $\mathcal{E}$ during the training process. On the other hand, we monitor the so-called *training error* given by

$$\mathcal{E}_T(\theta, \mathcal{S})^2 := \mathcal{J}(\theta, \mathcal{S}), \quad \mathcal{E}_T^* := \mathcal{E}_T(\theta^*(\mathcal{S}), \mathcal{S}), \tag{3.2}$$

with $\mathcal{J}$ being the loss function defined in Section 2.4 as the discretized version of the (integral-form) loss functions in Section 2.3. Hence, the training error $\mathcal{E}_T^*$ can be readily computed, after training has been completed, from the loss function. We will also refer to the integral form of the physics-informed loss function (see Section 2.3) as the *generalization error* $\mathcal{E}_G$,

$$\mathcal{E}_G(\theta) := \mathcal{E}_{\mathrm{PIL}}[u_\theta], \quad \mathcal{E}_G^* := \mathcal{E}_G(\theta^*(\mathcal{S}), \mathcal{S}), \tag{3.3}$$

as it measures how well the performance of a model transfers or *generalizes* from the training set $\mathcal{S}$ to the full domain Ω.

Given these definitions, some fundamental theoretical questions arise immediately.

Q1. *Existence.* Does there exist a model $\widehat{u}$ in the hypothesis class for which the generalization error $\mathcal{E}_G(\widehat{\theta})$ is small? More precisely, given a chosen error tolerance $\epsilon > 0$, does there exist a model in the hypothesis class $\widehat{u} = u_{\widehat{\theta}}$ for some $\widehat{\theta} \in \Theta$ such that for the corresponding generalization error $\mathcal{E}_G(\widehat{\theta})$ it holds that $\mathcal{E}_G(\widehat{\theta}) < \epsilon$? As minimizing the generalization error (i.e. the physics-informed loss) is the key pillar of physics-informed learning, obtaining a positive answer to this question is of the utmost importance. Moreover, it would be desirable to relate the size of the parameter space Θ (or hypothesis class) to the accuracy ϵ. For example, for linear models (Section 2.2.1) we would want to know how many functions ϕ_i are needed to ensure the existence of $\widehat{\theta} \in \Theta$ such that $\mathcal{E}_G(\widehat{\theta}) < \epsilon$. Similarly, for neural networks, we wish to find estimates of how large a neural network (Section 2.2.2) should be (in terms of depth, width and modulus of the weights) in order to ensure this. We will answer this question by considering the *approximation error* of a model class in Section 4.

Q2. *Stability.* Given that a model u_θ has a small generalization error $\mathcal{E}_G(\theta)$, will the corresponding total error $\mathcal{E}(\theta)$ be small as well? In other words, is there a function $\delta\colon \mathbb{R}_+ \to \mathbb{R}_+$ with $\lim_{\epsilon \to 0} \delta(\epsilon) = 0$ such that $\mathcal{E}_G(\theta) < \epsilon$ implies that $\mathcal{E}(\theta) < \delta(\epsilon)$ for any $\epsilon > 0$? In practice, we will be even more ambitious, as we hope to find constants $C, \alpha > 0$ (independent of θ) such that

$$\mathcal{E}(\theta) \leq C\mathcal{E}_G(\theta)^\alpha. \tag{3.4}$$

This is again an essential ingredient, as obtaining a small generalization error $\mathcal{E}_G^*$ is *a priori* meaningless, given that we only care about the total error $\mathcal{E}^*$. The answer to this question first and foremost depends on the properties of

the underlying PDE, and only to a lesser extent on the model class. We will discuss this question in Section 5.

Q3. *Generalization.* Given a small training error $\mathcal{E}_T^*$ and a sufficiently large training set $\mathcal{S}$, will the corresponding generalization error $\mathcal{E}_G^*$ also be small? This question is not uniquely tied to physics-informed machine learning and is closely tied to the accuracy of the chosen quadrature rule (Section 2.4). We will use standard arguments to answer this question in Section 6.

Q4. *Optimization.* Can $\mathcal{E}_T^*$ be made sufficiently close to $\min_\theta \mathcal{J}(\theta, \mathcal{S})$? This final question acknowledges the fact that it might be very hard to solve the minimization problem (2.69), even approximately. Especially for physics-informed neural networks there is a growing literature on the settings in which the optimization problem is hard to solve, or even infeasible, and how to potentially overcome these training issues. We theoretically analyse these phenomena in Section 7.

The above questions are of fundamental importance, as affirmative answers certify that the model that minimizes the optimization problem (2.69), denoted by u^*, is a good approximation of u in the sense that the error $\mathcal{E}^*$ is small. We show this by first proving an error decomposition for the generalization error $\mathcal{E}_G^*$, for which we need the following auxiliary result.

Lemma 3.1. For any $\theta_1, \theta_2 \in \Theta$ and training set $\mathcal{S}$, we have

$$\mathcal{E}_G(\theta_1) \leq \mathcal{E}_G(\theta_2) + 2 \sup_{\theta \in \Theta} |\mathcal{E}_T(\theta, \mathcal{S}) - \mathcal{E}_G(\theta)| + \mathcal{E}_T(\theta_1, \mathcal{S}) - \mathcal{E}_T(\theta_2, \mathcal{S}). \tag{3.5}$$

Proof. Fix $\theta_1, \theta_2 \in \Theta$ and generate a random training set $\mathcal{S}$. The proof consists of the repeated adding, subtracting and removing of terms. We have

$$\begin{aligned}
\mathcal{E}_G(\theta_1) &= \mathcal{E}_G(\theta_2) + \mathcal{E}_G(\theta_1) - \mathcal{E}_G(\theta_2) \\
&= \mathcal{E}_G(\theta_2) - (\mathcal{E}_T(\theta_1, \mathcal{S}) - \mathcal{E}_G(\theta_1)) + (\mathcal{E}_T(\theta_2, \mathcal{S}) - \mathcal{E}_G(\theta_2)) \\
&\quad + \mathcal{E}_T(\theta_1, \mathcal{S}) - \mathcal{E}_T(\theta_2, \mathcal{S}) \\
&\leq \mathcal{E}_G(\theta_2) + 2 \max_{\theta \in \{\theta_2, \theta_1\}} |\mathcal{E}_T(\theta, \mathcal{S}) - \mathcal{E}_G(\theta)| \\
&\quad + \mathcal{E}_T(\theta_1, \mathcal{S}) - \mathcal{E}_T(\theta_2, \mathcal{S}).
\end{aligned} \tag{3.6}$$

□

Assuming that we can answer Q1 in the affirmative, we can now take $\theta_1 = \theta^*(\mathcal{S})$ and $\theta_2 = \widehat{\theta}$ (as in Q1). Moreover, if we also assume that we can positively answer Q2, or more strongly we can find constants $C, \alpha > 0$ such that (3.4) holds, then we find the following error decomposition:

$$\mathcal{E}^* \leq C \Big(\underbrace{\mathcal{E}_G(\widehat{\theta})}_{\text{approximation error}} + \underbrace{2 \sup_{\theta \in \Theta} |\mathcal{E}_T(\theta, \mathcal{S}) - \mathcal{E}_G(\theta)|}_{\text{generalization gap}} + \underbrace{\mathcal{E}_T^* - \mathcal{E}_T(\widehat{\theta})}_{\text{optimization error}} \Big)^\alpha. \tag{3.7}$$

The total error $\mathcal{E}^*$ is now decomposed into three error sources. The *approximation error* $\mathcal{E}(\widehat{\theta})$ should be provably small, by answering Q1. The second term of the right-hand side, called the *generalization gap*, quantifies how well the training error approximates the generalization error, corresponding to Q3. Finally, an *optimization error* is incurred due to the inability of the optimization procedure to find $\widehat{\theta}$ based on a finite training data set. Indeed one can see that if $\theta^*(\mathcal{S}) = \widehat{\theta}$, the optimization error vanishes. This source of error is addressed by Q4. Hence, an affirmative answer to Q1–Q4 leads to a small error $\mathcal{E}^*$.

We will present general results to answer questions Q1–Q4 and then apply them to the following prototypical examples to further highlight interesting phenomena.

- The Navier–Stokes equation (Example 2.2) as an example of a challenging but low-dimensional PDE.
- The heat equation (Example 2.1) as a prototypical example of a potentially very high-dimensional PDE. We will investigate whether one can overcome the *curse of dimensionality* through physics-informed learning for this equation and related PDEs.
- Inviscid scalar conservation laws (Example 2.3 with $\nu = 0$) will serve as an example of a PDE where the solutions might not be smooth and even discontinuous. As a result, the weak residuals from Section 2.3.2 will be needed.
- Poisson's equation (Example 2.4) as an example for which the variational formulation of Section 2.3.3 can be used.

4. Approximation error

In this section we answer the first question, Q1.

Question 1 (Q1). Does there exist a model $\widehat{u} = u_{\widehat{\theta}}$ in the hypothesis class for which the generalization error $\mathcal{E}_G(\widehat{\theta})$ is small?

The difficulty in answering this question lies in the fact that the generalization error $\mathcal{E}_G(\theta)$ in physics-informed learning is given by the physics-informed loss of the model $\mathcal{E}_{\text{PIL}}(u_\theta)$ rather than just being $\|u - u_\theta\|_{L^2}^2$ as would be the case for regression. For neural networks, for example, the universal approximation theorems (e.g. Cybenko 1989) guarantee that any measurable function can be approximated by a neural network in supremum-norm. However, they do not imply an affirmative answer to Question 1: a neural network that approximates a function well in supremum-norm might be highly oscillatory, such that the derivatives of the network and that of the function are very different, giving rise to a large PDE residual. Hence, we require results on the existence of models that approximate functions in a *stronger norm* than the supremum-norm. In particular, the norm

should quantify how well the derivatives of the model approximate those of the function.

For solutions of PDEs, a very natural norm that satisfies this criterion is the *Sobolev norm*. For $s \in \mathbb{N}$ and $q \in [1, \infty]$, the Sobolev space $W^{s,q}(\Omega; \mathbb{R}^m)$ is the space of all functions $f : \Omega \to \mathbb{R}^m$ for which f, as well as the (weak) derivatives of f up to order s, belong to $L^q(\Omega; \mathbb{R}^m)$. When more smoothness is available, one could replace the space $W^{s,\infty}(\Omega; \mathbb{R}^m)$ with the space of s times continuously differentiable functions $C^s(\Omega; \mathbb{R}^m)$; for more information see the Appendix.

Given an approximation result in some Sobolev (semi-) norm, we can derive the physics-informed loss of the model $\mathcal{E}_{\text{PIL}}(u_\theta)$ if the following assumption holds true.

When the physics-informed loss is based on the strong (classical) formulation of the PDE (see Section 2.3.1), we assume that it can be bounded in terms of the errors related to all relevant partial derivatives, denoted by $D^{(k,\boldsymbol{\alpha})} := D_t^k D_x^{\boldsymbol{\alpha}} := \partial_t^k \partial_{x_1}^{\alpha_1} \dots \partial_{x_d}^{\alpha_d}$, for $(k, \boldsymbol{\alpha}) \in \mathbb{N}_0^{d+1}$ for time-dependent PDEs. This assumption is valid for many classical solutions of PDEs.

Assumption 4.1. Let $k, \ell \in \mathbb{N}$, $q \in [1, +\infty]$, $C > 0$ be independent from d and let $\mathcal{X} \subset \{u_\theta : \theta \in \Theta\}$ and $\Omega = [0, T] \times D$. It holds for every $u_\theta \in \mathcal{X}$ that

$$\|\mathcal{L}[u_\theta]\|_{L^q(\Omega)} \leq C \cdot \text{poly}(d) \cdot \sum_{\substack{(k',\boldsymbol{\alpha}) \in \mathbb{N}_0^{d+1} \\ k' \leq k, \|\boldsymbol{\alpha}\|_1 \leq \ell}} \|D^{(k',\boldsymbol{\alpha})}(u - u_\theta)\|_{L^q(\Omega)}. \tag{4.1}$$

Note that the corresponding assumption for operator learning can be obtained by replacing u with the operator $\mathcal{G}$ and u_θ with $\mathcal{G}_\theta$. We give a brief example to illustrate the assumption.

Example 4.2 (heat equation). For the heat equation we have

$$\begin{aligned} \|\mathcal{L}[u_\theta]\|_{L^q(\Omega)} &= \|\mathcal{L}[u_\theta] - \mathcal{L}[u]\|_{L^q(\Omega)} \\ &\leq \|\Delta(u_\theta - u)\|_{L^q(\Omega)} + \|\partial_t(u_\theta - u)\|_{L^q(\Omega)}. \end{aligned} \tag{4.2}$$

4.1. First general result

Note in particular that Assumption 4.1 is satisfied when

$$\|\mathcal{L}[u_\theta]\|_{L^q(\Omega)} \leq C \cdot \text{poly}(d) \cdot \|u - u_\theta\|_{W^{k,q}(\Omega)}. \tag{4.3}$$

Hence, it suffices to prove that there exists $\widehat{\theta} \in \Theta$ for which $\|u - u_{\widehat{\theta}}\|_{W^{k,q}(\Omega)}$ can be made small.

For many linear models, such approximation results are widely available. For example, when we choose the (default enumeration of the) d-dimensional Fourier basis as functions ϕ_i (see Section 2.2.1), then we can use the following result (Hesthaven *et al.* 2007).

Theorem 4.3. Let $s, k \in \mathbb{N}_0$ with $s > d/2$ and $s \geq k$, and $u \in C^s(\mathbb{T}^d)$. Then there exists $\widehat{\theta} \in \mathbb{R}^N$ such that

$$\left\| u - \sum_{i=1}^{N^d} \widehat{\theta}_i \phi_i \right\|_{H^k(\mathbb{T}^d)} \leq C(s,d) N^{-(s-k)/d} \|u\|_{H^s(\mathbb{T}^d)}, \tag{4.4}$$

for a constant $C(s,d) > 0$ that only depends on s and d.

Analogous results are also available for neural networks. We state a result that is tailored to tanh neural networks (De Ryck, Lanthaler and Mishra 2021, Theorem 5.1), but more general results are also available (Gühring and Raslan 2021).

Theorem 4.4. Let $\Omega = [0,1]^d$. For every $N \in \mathbb{N}$ and every $f \in W^{s,\infty}(\Omega)$, there exists a tanh neural network $\widehat{f}$ with two hidden layers of width N such that for every $0 \leq k < s$ we have

$$\|f - \widehat{f}\|_{W^{k,\infty}(\Omega)} \leq C(\ln(cN))^k N^{-(s-k)/d}, \tag{4.5}$$

where $c, C > 0$ are independent of N and explicitly known.

Proof. The main ingredients are a piecewise polynomial approximation, the existence of which is guaranteed by the Bramble–Hilbert lemma (Verfürth 1999), and the ability of tanh neural networks to efficiently approximate polynomials, the multiplication operator and an approximate partition of unity. The full proof can be found in De Ryck *et al.* (2021, Theorem 5.1). □

Remark 4.5. In this section we have mainly focused on bounding the (strong) PDE residual, as it is generally the most difficult residual to bound, containing the highest derivatives. Indeed, when using the classical formulation, the bounds on the spatial (and temporal) boundary residuals generally follow from a (Sobolev) trace inequality. Similarly, loss functions resulting from the weak or variational formulation contain less high derivatives of the neural network and are therefore easier to bound.

As an example, we apply the above result to the Navier–Stokes equations.

Example 4.6 (Navier–Stokes). One can use Theorem 4.4 to prove the existence of neural networks with a small generalization error for the Navier–Stokes equations (Example 2.2). The existence and regularity of the solution to (2.8) depends on the regularity of u_0, as is stated by the following well-known theorem (Majda and Bertozzi 2002, Theorem 3.4). Other regularity results with different boundary conditions can be found in Temam (2001), for example.

Theorem 4.7. If $u_0 \in H^r(\mathbb{T}^d)$ with $r > d/2 + 2$ and $\operatorname{div} u_0 = 0$, then there exist $T > 0$ and a classical solution u to the Navier–Stokes equation such that $u(t=0) = u_0$ and $u \in C([0,T]; H^r(\mathbb{T}^d)) \cap C^1([0,T]; H^{r-2}(\mathbb{T}^d))$.

Based on this result one can prove that u is Sobolev-regular, that is, $u \in H^k(D \times [0,T])$ for some $k \in \mathbb{N}$, provided that r is large enough (De Ryck, Jagtap and Mishra 2024*a*).

Corollary 4.8. If $k \in \mathbb{N}$ and $u_0 \in H^r(\mathbb{T}^d)$ with $r > d/2 + 2k$ and $\operatorname{div} u_0 = 0$, then there exist $T > 0$ and a classical solution u to the Navier–Stokes equation such that $u \in H^k(\mathbb{T}^d \times [0,T])$, $\nabla p \in H^{k-1}(\mathbb{T}^d \times [0,T])$ and $u(t=0) = u_0$.

Combining this regularity result with Theorem 4.4 then gives rise to the following approximation result for the Navier–Stokes equations (De Ryck *et al.* 2024*a*, Theorem 3.1).

Theorem 4.9. Let $n \geq 2$, $d, r, k \in \mathbb{N}$, with $k \geq 3$, and let $u_0 \in H^r(\mathbb{T}^d)$ with $r > d/2 + 2k$ and $\operatorname{div} u_0 = 0$. Let $T > 0$ be the time from Corollary 4.8 such that the classical solution of u exists on $\mathbb{T}^d \times [0,T]$. Then, for every $N > 5$, there exist tanh neural networks $\widehat{u}_j$, $1 \leq j \leq d$, and $\widehat{p}$, each with two hidden layers, of widths

$$3\left\lceil \frac{k+n-2}{2} \right\rceil \binom{d+k-1}{d} + \lceil TN \rceil + dN \quad \text{and} \quad 3\left\lceil \frac{d+n}{2} \right\rceil \binom{2d+1}{d} \lceil TN \rceil N^d,$$

such that

$$\|(\widehat{u}_j)_t + \widehat{u} \cdot \nabla \widehat{u}_j + (\nabla \widehat{p})_j - \nu \Delta \widehat{u}_j\|_{L^2(\Omega)} \leq C_1 \ln^2(\beta N) N^{-k+2}, \tag{4.6}$$

$$\|\operatorname{div} \widehat{u}\|_{L^2(\Omega)} \leq C_2 \ln(\beta N) N^{-k+1}, \tag{4.7}$$

$$\|(u_0)_j - \widehat{u}_j(t=0)\|_{L^2(\mathbb{T}^d)} \leq C_3 \ln(\beta N) N^{-k+1}, \tag{4.8}$$

for every $1 \leq j \leq d$, and where the constants β, C_1, C_2, C_3 are explicitly defined in the proof and can depend on k, d, T, u and p but not on N. The weights of the networks can be bounded by $O(N^\gamma \ln(N))$, where $\gamma = \max\{1, d(2+k^2+d)/n\}$.

4.2. Second general result

Combining results such as Theorem 4.3 or Theorem 4.4 with Assumption 4.1 thus allows us to prove that the generalization error $\mathcal{E}_G$ can be made arbitrarily small, thereby answering Question 1. There is, however, still room for improvement.

- For many model classes the main results available only state that the error $\|u - u_\theta\|_{L^q}$ can be made small. Is there a way to infer that if $\|u - u_\theta\|_{L^q}$ is small then $\|u - u_\theta\|_{W^{k,q}}$ is also small (under some assumptions)?
- Both Theorem 4.3 and Theorem 4.4 suffer from the *curse of dimensionality* (CoD), meaning that the number of parameters needed to reach an accuracy of ϵ scales exponentially with the input dimension d, namely as $\epsilon^{-(s-k)/d}$. Unless $s \approx k + d$, this will mean that an infeasibly large number of parameters is needed to guarantee a sufficiently small generalization error. Experimentally, however, we have observed for many PDEs that the CoD can be overcome. This raises the following question: Can we improve upon the approximation results stated in this section?

- Both Theorem 4.3 and Theorem 4.4 require the true solution u of the PDE to be sufficiently (Sobolev) regular. Is there a way to still prove that the approximation error is small if u is less regular, such as for inviscid scalar conservation laws?

As it turns out, the first two questions above can be answered in the same manner, which is the topic below.

We first show that, under some assumptions, we can indeed answer Question 1 even if we only have access to an approximation result for $\|u - u_\theta\|_{L^q}$ and not for $\|u - u_\theta\|_{W^{k,q}}$. We do this by using finite difference (FD) approximations. Depending on whether forward, backward or central differences are used, an FD operator might not be defined on the whole domain D; for example, for $f \in C([0,1])$ the (forward) operator $\Delta_h^+[f] := f(x+h) - f(x)$ is not well-defined for $x \in (1-h, 1]$. This can be solved by resorting to piecewise defined FD operators, e.g. a forward operator on $[0, 0.5]$ and a backward operator on $(0.5, 1]$. In a general domain Ω one can find a well-defined piecewise FD operator if Ω satisfies the following assumption, which is satisfied by many domains (e.g. rectangular or smooth domains).

Assumption 4.10. There exists a finite partition $\mathcal{P}$ of Ω such that for all $P \in \mathcal{P}$ there exists $\epsilon_P > 0$, and $v_P \in B_\infty^1 = \{x \in \mathbb{R}^{\dim(\Omega)} : \|x\|_\infty \leq 1\}$ such that for all $x \in P$ we have $x + \epsilon_P(v_P + B_\infty^1) \subset \Omega$.

Under this assumption we can prove an upper bound on the (strong) PDE residual of the model u_θ.

Theorem 4.11. Let $q \in [1, \infty]$, $r, \ell \in \mathbb{N}$ with $\ell \leq r$ and $u, u_\theta \in C^r(\Omega)$. If Assumptions 4.1 and 4.10 hold, then there exists a constant $C(r) > 0$ such that for any $\boldsymbol{\alpha} \in \mathbb{N}_0^d$ with $\ell := \|\boldsymbol{\alpha}\|_1$, it holds for all $h > 0$ that

$$\|\mathcal{L}(u_\theta)\|_{L^q} \leq C\big(\|u - u_\theta\|_{L^q} h^{-\ell} + (|u|_{C^r} + |u_\theta|_{C^r}) h^{r-\ell}\big). \tag{4.9}$$

Proof. The proof follows from approximating any $D^{\boldsymbol{\alpha}}(u_\theta - u)$ using an rth order accurate finite-difference formula (which is possible thanks to Assumption 4.10), and combining this with Assumption 4.1. See also De Ryck and Mishra (2022*b*, Lemma B.1). □

Theorem 4.11 shows that $\|\mathcal{L}(u_\theta)\|_{L^q}$ is small if $\|u - u_\theta\|_{L^q}$ is small and $|u_\theta|_{C^r}$ does not increase too much in terms of the model size. It must be stated that these assumptions are not necessarily trivially satisfied. Indeed, assume that $\Theta \subset \mathbb{R}^n$, that $\|u - u_\theta\|_{L^q} \sim n^{-\alpha}$ and $|u_\theta|_{C^r} \sim n^\beta$, and finally that $n \sim h^{-\gamma}$ for $\alpha, \beta, \gamma \geq 0$. In this case, the upper bound of Theorem 4.11 is equivalent to

$$\|\mathcal{L}(u_\theta)\|_{L^q} \lesssim h^{\alpha\gamma} h^{-\ell} + h^{-\beta\gamma} h^{r-\ell}. \tag{4.10}$$

To make sure that the right-hand side is small for $h \to 0$, the inequality $\beta\ell < \alpha(r-\ell)$ should hold. Hence, either $\|u - u_\theta\|_{L^q}$ should converge fast (large α), $|u_\theta|_{C^r}$

should diverge slowly (small β) or not at all ($\beta = 0$), or the true solution u should be very smooth (large r). By way of example, we investigate what kind of bound is produced by Theorem 4.11 if the Fourier basis is used.

Example 4.12 (Fourier basis). Theorem 4.3 tells us that $\alpha = r/d$ and that $\beta = 0$. Hence the condition $\beta\ell < \alpha(r - \ell)$ is already satisfied. We now choose $\gamma = d$ such that the optimal rate is obtained, for which $h^{\alpha\gamma}h^{-\ell} = h^{r-\ell}$. The final convergence rate is hence $N^{-(r-\ell)/d}$, in agreement with that of Theorem 4.3.

Remark 4.13. Unlike in the previous example, Theorem 4.11 is not expected to produce optimal convergence rates, particularly when loose upper bounds for $|u_\theta|_{C^r}$ are used. However, this is not a problem when trying to prove that a model class can overcome the curse of dimensionality in the approximation error, as is the topic of the next section.

Finally, as noted at the beginning of this section, all results in the section so far all assume that the solution u of the PDE is sufficiently (Sobolev) regular. In the next example we show that one can also prove approximation results, thereby answering Question 1, when u has discontinuities.

Example 4.14 (scalar conservation laws). In Example 2.10 a physics-informed loss based on the (weak) Kruzkhov entropy residual was introduced for scalar conservation laws (Example 2.3). It consists of the term $\max_{\vartheta,c} \mathcal{R}(u_\theta, \phi_\vartheta, c)$ together with the integral of the squared boundary residual $\mathcal{R}_s[u_\theta]$ and the integral of the squared initial condition residual $\mathcal{R}_t[u_\theta]$. Moreover, as solutions to scalar conservation laws might not be Sobolev-regular, an approximation result cannot be directly proved based on Theorem 4.4. Instead, De Ryck *et al.* (2024*c*) proved that the relevant counterpart to Assumption 4.1 is the following lemma.

Lemma 4.15. Let $p, q > 1$ be such that $1/p + 1/q = 1$, or let $p = \infty$ and $q = 1$. Let u be the entropy solution of (2.10) and let $v \in L^q(D \times [0,T])$. Assume that f has Lipschitz constant L_f. Then it holds for any $\varphi \in W_0^{1,\infty}(D \times [0,T])$ that

$$\mathcal{R}(v, \varphi, c) \leq (1 + 3L_f)|\varphi|_{W^{1,p}(D\times[0,T])}\|u - v\|_{L^q(D\times[0,T])}. \tag{4.11}$$

Hence we now need an approximation result for $\|u - u_\theta\|_{L^q(D\times[0,T])}$ rather than for a higher-order Sobolev norm of $u - u_\theta$. The following holds (De Ryck *et al.* 2024*c*, Lemma 3.3).

Lemma 4.16. For every $u \in BV([0,1] \times [0,T])$ and $\epsilon > 0$ there is a tanh neural network $\widehat{u}$ with two hidden layers and at most $O(\epsilon^{-2})$ neurons such that

$$\|u - \widehat{u}\|_{L^1([0,1]\times[0,T])} \leq \epsilon. \tag{4.12}$$

Proof. Write $\Omega = [0,1] \times [0,T]$. For every $u \in BV(\Omega)$ and $\epsilon > 0$ there exists a function $\widetilde{u} \in C^\infty(\Omega) \cap BV(\Omega)$ such that $\|u - \widetilde{u}\|_{L^1(\Omega)} \lesssim \epsilon$ and $\|\nabla\widetilde{u}\|_{L^1(\Omega)} \lesssim \|u\|_{BV(\Omega)} + \epsilon$ (Bartels 2012). Then use the approximation techniques of De Ryck *et al.* (2021, 2024*a*) and the fact that $\|\widetilde{u}\|_{W^{1,1}(\Omega)}$ can be uniformly bounded in ϵ to

find the existence of a tanh neural network $\widehat{u}$ with two hidden layers and at most $O(\epsilon^{-2})$ neurons that satisfies the mentioned error estimate. □

If we additionally know that u is piecewise smooth, for instance as in the solutions of convex scalar conservation laws (Holden and Risebro 2015), then one can use the results of Petersen and Voigtlaender (2018) to obtain the following result (De Ryck *et al.* 2024*c*, Lemma 3.4).

Lemma 4.17. Let $m, n \in \mathbb{N}$, $1 \leq q < \infty$ and let $u\colon [0,1] \times [0,T] \to \mathbb{R}$ be a function that is piecewise C^m-smooth and with a C^n-smooth discontinuity surface. Then there is a tanh neural network $\widehat{u}$ with at most three hidden layers and $O(\epsilon^{-2/m} + \epsilon^{-q/n})$ neurons such that

$$\|u - \widehat{u}\|_{L^q([0,1]\times[0,T])} \leq \epsilon. \tag{4.13}$$

Finally, De Ryck *et al.* (2024*c*) found that we have to consider test functions φ that might grow as $|\varphi|_{W^{1,p}} \sim \epsilon^{-3(1+2(p-1)/p)}$. Consequently, we will need to use Lemma 4.16 with $\epsilon \leftarrow \epsilon^{4+6(p-1)/p}$, leading to the following corollary.

Corollary 4.18. Assume the setting of Lemma 4.17, assume that $mq > 2n$, and let $p \in [1, \infty]$ be such that $1/p + 1/q = 1$. There is a fixed-depth tanh neural network $\widehat{u}$ with size $O(\epsilon^{-(4q+6)n})$ such that

$$\max_{c\in\mathcal{C}} \sup_{\varphi\in\overline{\Phi}_\epsilon} \mathcal{R}(\widehat{u}, \varphi, c) + \lambda_s \|\mathcal{R}_s[\widehat{u}]\|^2_{L^2(\partial D\times[0,T])} + \lambda_t \|\mathcal{R}_t[\widehat{u}]\|^2_{L^2(D)} \leq \epsilon, \tag{4.14}$$

where

$$\overline{\Phi}_\epsilon = \left\{\varphi\colon |\varphi|_{W^{1,p}} = O\left(\epsilon^{-3(1+2(p-1)/p)}\right)\right\}.$$

Hence we have answered Question 1 in the affirmative for scalar conservation laws.

Proof. We find that $\widehat{u}$ will need to have a size of $O(\epsilon^{-(4q+6)/\beta})$ for it to hold that $\max_{c\in\mathcal{C}} \sup_{\varphi\in\Phi_\epsilon} \mathcal{R}(\widehat{u}, \varphi, c) \leq \epsilon/3$, where we used that $q = p/(p-1)$. Since in the proof of Lemma 4.17 the network $\widehat{u}$ is constructed as an approximation of piecewise Taylor polynomials, the spatial and temporal boundary residuals ($\mathcal{R}_s$ and $\mathcal{R}_t$) are automatically minimized as well, given that Taylor polynomials provide approximations in the C^0-norm. □

4.3. Neural networks overcome curse of dimensionality in approximation error

We investigate whether (physics-informed) neural networks can overcome the curse of dimensionality (CoD) for the approximation error. Concretely, we want to prove the existence of a neural network with parameter $\widehat{\theta}$ for which $\mathcal{E}_G(\widehat{\theta}) < \epsilon$, without the size of the network growing exponentially in the input dimension d. We discuss two frameworks in which this can be done, both of which exploit the properties of the PDE.

- The first framework is tailored to time-dependent PDEs and considers initial conditions and PDE solutions that are Sobolev-regular (or continuously differentiable). The crucial assumption is that one must be able to approximate the initial condition u_0 with a neural network, without incurring the CoD, i.e. approximate it to accuracy ϵ with a network of size $O(d^\alpha \epsilon^{-\beta})$ with $\alpha, \beta > 0$ independent of d. This framework can also be used to prove results for (physics-informed) operator learning.
- The second framework circumvents this assumption by considering PDE solutions that belong to a smaller space, namely the so-called *Barron space*. It can be proved that functions in this space can be approximated without the curse of dimensionality. Hence, in this case the challenge lies in proving that the PDE solution is indeed a *Barron function*.

4.3.1. Results based on Sobolev- and C^k-regularity

We initially direct our focus onto the first framework. To start with, we give particular attention to the case where it is known that a neural network can efficiently approximate the solution to a time-dependent PDE at a fixed time. Such neural networks are usually obtained by emulating a classical numerical method. Examples include finite difference schemes, finite volume schemes, finite element methods, iterative methods and Monte Carlo methods (e.g. Jentzen, Salimova and Welti 2021, Opschoor, Petersen and Schwab 2020, Chen, Lu and Lu 2021, Marwah, Lipton and Risteski 2021). In these cases Theorem 4.11 cannot be directly applied, as there is no upper bound for $\|u - u_\theta\|_{L^q(D\times[0,T])}$. To allow for a further generalization to operator learning, we will write down the following assumption in terms of operators (see Section 2.1.1).

More precisely, for $\epsilon > 0$, we assume that we have access to an operator $\mathcal{U}^\epsilon : \mathcal{X} \times [0,T] \to \mathcal{H}$ that, for any $t \in [0,T]$, maps any initial condition/parameter function $v \in \mathcal{X}$ to a neural network $\mathcal{U}^\epsilon(v,t)$ that approximates the PDE solution $\mathcal{G}(v)(\cdot,t) = u(\cdot,t) \in L^q(D), q \in \{2,\infty\}$, at time t, as specified below. Moreover, we will assume that we know how its size depends on the accuracy ϵ.

Assumption 4.19. Let $q \in \{2,\infty\}$. For any $B, \epsilon > 0$, $\ell \in \mathbb{N}$, $t \in [0,T]$ and any $v \in \mathcal{X}$ with $\|v\|_{C^\ell} \leq B$, there exist a neural network $\mathcal{U}^\epsilon(v,t) \colon D \to \mathbb{R}$ and a constant $C^B_{\epsilon,\ell} > 0$ such that

$$\|\mathcal{U}^\epsilon(v,t) - \mathcal{G}(v)(\cdot,t)\|_{L^q(D)} \leq \epsilon \quad \text{and} \quad \max_{t\in[0,T]} \|\mathcal{U}^\epsilon(v,t)\|_{W^{\ell,q}(D)} \leq C^B_{\epsilon,\ell}. \tag{4.15}$$

For vanilla neural networks and PINNs, however, we can always set $\mathcal{X} := \{v\}$, with $v := u_0$ or $v := a$ and $\mathcal{G}(v) := u$ in Assumption 4.19 above.

Under Assumption 4.19, the existence of space–time neural networks that minimize the generalization error $\mathcal{E}_G$ (i.e. the physics-informed loss) can be proved (De Ryck and Mishra 2022*b*).

Theorem 4.20. Let $s, r \in \mathbb{N}$, let $u \in C^{(s,r)}([0,T] \times D)$ be the solution of the PDE (2.1) and let Assumption 4.19 be satisfied. There exists a constant $C(s,r) > 0$ such that, for every $M \in \mathbb{N}$ and $\epsilon, h > 0$, there exists a tanh neural network $u_\theta \colon [0,T] \times D \to \mathbb{R}$ for which

$$\|u_\theta - u\|_{L^q([0,T]\times D)} \leq C(\|u\|_{C^{(s,0)}} M^{-s} + \epsilon), \tag{4.16}$$

and if additionally Assumptions 4.1 and 4.10 hold, then

$$\begin{aligned} &\|\mathcal{L}(u_\theta)\|_{L^2([0,T]\times D)} + \|u_\theta - u\|_{L^2(\partial([0,T]\times D))} \\ &\quad \leq C \cdot \mathrm{poly}(d) \cdot \ln^k(M)\big(\|u\|_{C^{(s,\ell)}} M^{k-s} + M^{2k}\big(\epsilon h^{-\ell} + C^B_{\epsilon,\ell} h^{r-\ell}\big)\big). \end{aligned} \tag{4.17}$$

Moreover, the depth is given by $\mathrm{depth}(u_\theta) \leq C \cdot \sup_{t\in[0,T]} \mathrm{depth}(\mathcal{U}^\epsilon(u(t)))$ and the width by $\mathrm{width}(u_\theta) \leq CM \cdot \sup_{t\in[0,T]} \mathrm{width}(\mathcal{U}^\epsilon(u(t)))$.

Proof. We only provide a sketch of the full proof (De Ryck and Mishra 2022*b*, Theorem 3.5). The main idea is to divide $[0,T]$ into M uniform subintervals and construct a neural network that approximates a Taylor approximation in time of u in each subinterval. In the formula obtained, we approximate the monomials and multiplications by neural networks and approximate the derivatives of u by finite differences and use the accuracy of finite difference formulas to find an error estimate in the $C^k([0,T], L^q(D))$-norm. We again use finite difference operators to prove that spatial derivatives of u are accurately approximated as well. The neural network will also approximately satisfy the initial/boundary conditions as

$$\|u_\theta - u\|_{L^2(\partial([0,T]\times D))} \lesssim C\mathrm{poly}(d)\|u_\theta - u\|_{H^1([0,T]\times D)},$$

which follows from a Sobolev trace inequality. □

As a next step, we use Assumption 4.19 to prove estimates for deep operator learning. Given the connection between DeepONets and FNOs (Kovachki *et al.* 2021, Theorem 36), we focus on DeepONets in the following. In order to prove this error estimate, we need to assume that the operator $\mathcal{U}^\epsilon$ from Assumption 4.19 is stable with respect to its input function, as specified in Assumption 4.21 below. Moreover, we will take the d-dimensional torus as domain $D = \mathbb{T}^d = [0, 2\pi)^d$ and assume periodic boundary conditions for simplicity in what follows. This is not a restriction, as for every Lipschitz subset of $\mathbb{T}^d$ there exists a (linear and continuous) $\mathbb{T}^d$-periodic extension operator for which the derivatives are also $\mathbb{T}^d$-periodic (Kovachki *et al.* 2021, Lemma 41).

Assumption 4.21. Assumption 4.19 is satisfied and let $p \in \{2, \infty\}$. For every $\epsilon > 0$ there exists a constant $C^\epsilon_{\mathrm{stab}} > 0$ such that, for all $v, v' \in \mathcal{X}$,

$$\|\mathcal{U}^\epsilon(v,T) - \mathcal{U}^\epsilon(v',T)\|_{L^2} \leq C^\epsilon_{\mathrm{stab}} \|v - v'\|_{L^p}. \tag{4.18}$$

In this setting, we prove a generic approximation result for DeepONets (De Ryck and Mishra 2022*b*, Theorem 3.10).

Theorem 4.22. Let $s, r \in \mathbb{N}, T > 0, \mathcal{A} \subset C^r(\mathbb{T}^d)$ and let $\mathcal{G}\colon \mathcal{A} \to C^{(s,r)}([0,T] \times \mathbb{T}^d)$ be an operator that maps a function u_0 to the solution u of the PDE (2.1) with initial condition u_0, let Assumptions 4.1, 4.10 and 4.21 be satisfied and let $p^* \in \{2, \infty\} \setminus \{p\}$. There exists a constant $C > 0$ such that for every $Z, N, M \in \mathbb{N}$, $\epsilon, \rho > 0$ there is a DeepONet $\mathcal{G}_\theta\colon \mathcal{A} \to L^2([0,T] \times \mathbb{T}^d)$ with Z^d sensors with accuracy

$$\begin{aligned}\|\mathcal{L}(\mathcal{G}_\theta(v))\|_{L^2([0,T]\times\mathbb{T}^d)} \leq C M^{k+\rho} \big[\|u\|_{C^{(s,\ell)}} M^{-s} \\ + M^{s-1} N^\ell \left(\epsilon + C^\epsilon_{\text{stab}} Z^{-r+d/p^*} + C^{CB}_{\epsilon,r} N^{-r}\right)\big],\end{aligned} \tag{4.19}$$

for all v. Moreover,

$$\begin{aligned}&\text{width}(\beta) = O(M(Z^d + N^d \text{width}(\mathcal{U}^\epsilon))), \quad &&\text{depth}(\beta) = \text{depth}(\mathcal{U}^\epsilon), \\ &\text{width}(\tau) = O(MN^d(N + \ln(N))), &&\text{depth}(\tau) = 3,\end{aligned} \tag{4.20}$$

where $\text{width}(\mathcal{U}^\epsilon) = \sup_{u_0 \in \mathcal{A}} \text{width}(\mathcal{U}^\epsilon)(u_0))$ and similarly for $\text{depth}(\mathcal{U}^\epsilon)$.

Finally, we show how the results of this section can be applied to high-dimensional PDEs, for which it is not possible to obtain efficient approximation results using standard neural network approximation theory (see Theorem 4.4) as they will lead to convergence rates that suffer from the *curse of dimensionality* (CoD), meaning that the neural network size scales exponentially in the input dimension. In the literature it has been shown for some PDEs that their solution at a fixed time can be approximated to accuracy $\epsilon > 0$ with a network that has size $O(\text{poly}(d)\epsilon^{-\beta})$, with $\beta > 0$ independent of d, and therefore *overcomes the CoD*.

Example 4.23 (linear Kolmogorov PDEs). Linear Kolmogorov PDEs are a class of linear time-dependent PDEs, including the heat equation and the Black–Scholes equation, of the following form. Let $s, r \in \mathbb{N}$, $u_0 \in C^2_0(\mathbb{R}^d)$ and let $u \in C^{(s,r)}([0,T] \times \mathbb{R}^d)$ be the solution of

$$\mathcal{L}[u] = \partial_t u - \frac{1}{2}\text{Tr}(\sigma(x)\sigma(x)^\top \Delta_x[u]) - \mu^\top \nabla_x[u] = 0, \quad u(0,x) = u_0(x) \tag{4.21}$$

for all $(x,t) \in D \times [0,T]$, where $\sigma\colon \mathbb{R}^d \to \mathbb{R}^{d\times d}$ and $\mu\colon \mathbb{R}^d \to \mathbb{R}^d$ are affine functions. We make the assumption that $\|u\|_{C^{(s,2)}}$ grows at most polynomially in d, and that for every $\epsilon > 0$ there is a neural network $\widehat{u}_0$ of width $O(\text{poly}(d)\epsilon^{-\beta})$ such that $\|u_0 - \widehat{u}_0\|_{L^\infty(\mathbb{R}^d)} < \epsilon$.

In this setting, Grohs, Hornung, Jentzen and von Wurstemberger (2018), Berner, Grohs and Jentzen (2020) and Jentzen *et al.* (2021) construct a neural network that approximates $u(T)$ and overcomes the CoD by emulating Monte Carlo methods based on the Feynman–Kac formula. De Ryck and Mishra (2022*a*) have proved that PINNs overcome the CoD as well, in the sense that the network size grows as $O(\text{poly}(\text{d}\rho_d)\epsilon^{-\beta})$, where ρ_d is a PDE-dependent constant that for a subclass of Kolmogorov PDEs scales as $\rho_d = \text{poly}(d)$, such that the CoD is fully overcome. We demonstrate that the generic bounds of this section (Theorem 4.20) can be

used to provide a (much shorter) proof of this result (De Ryck and Mishra 2022*b*, Theorem 4.2).

Theorem 4.24. For every $\sigma, \epsilon > 0$ and $d \in \mathbb{N}$, there is a tanh neural network u_θ of depth $O(\text{depth}(\widehat{u}_0))$ and width

$$O\left(\text{poly}(\text{d}\rho_d)\epsilon^{-(2+\beta)\frac{r+\sigma}{r-2}\frac{s+1}{s-1}-\frac{1+\sigma}{s-1}}\right)$$

such that

$$\|\mathcal{L}(u_\theta)\|_{L^2([0,T]\times[0,1]^d)} + \|u_\theta - u\|_{L^2(\partial([0,T]\times[0,1]^d))} \leq \epsilon. \tag{4.22}$$

Example 4.25 (nonlinear parabolic PDEs). Next, we consider nonlinear parabolic PDEs as in (4.23), which typically arise in the context of nonlinear diffusion–reaction equations that describe the change in space and time of some quantities, such as in the well-known *Allen–Cahn equation* (Allen and Cahn 1979). Let $s, r \in \mathbb{N}$, and for $u_0 \in \mathcal{X} \subset C^r(\mathbb{T}^d)$ let $u \in C^{(s,r)}([0,T] \times \mathbb{T}^d)$ be the solution of

$$\mathcal{L}(u)(x,t) = \partial_t u(t,x) - \Delta_x u(t,x) - F(u(t,x)) = 0, \quad u(0,x) = u_0(x), \tag{4.23}$$

for all $(t,x) \in [0,T] \times D$, with period boundary conditions, where $F\colon \mathbb{R} \to \mathbb{R}$ is a polynomial. As in Example 4.23, we assume that $\|u\|_{C^{(s,2)}}$ grows at most polynomially in d, and that for every $\epsilon > 0$ there is a neural network $\widehat{u}_0$ of width $O(\text{poly}(d)\epsilon^{-\beta})$ such that $\|u_0 - \widehat{u}_0\|_{L^\infty(\mathbb{T}^d)} < \epsilon$.

Hutzenthaler, Jentzen, Kruse and Nguyen (2020) have proved that ReLU neural networks overcome the CoD in the approximation of $u(T)$. Using Theorem 4.20 we can now prove that PINNs overcome the CoD for nonlinear parabolic PDEs (De Ryck and Mishra 2022*b*, Theorem 4.4).

Theorem 4.26. For every $\sigma, \epsilon > 0$ and $d \in \mathbb{N}$, there is a tanh neural network u_θ of depth $O(\text{depth}(\widehat{u}_0) + \text{poly}(d)\ln(1/\epsilon))$ and width

$$O\left(\text{poly}(d)\epsilon^{-(2+\beta)\frac{r+\sigma}{r-2}\frac{s+1}{s-1}-\frac{1+\sigma}{s-1}}\right)$$

such that

$$\|\mathcal{L}(u_\theta)\|_{L^2([0,T]\times\mathbb{T}^d)} + \|u - u_\theta\|_{L^2(\partial([0,T]\times\mathbb{T}^d))} \leq \epsilon. \tag{4.24}$$

Similarly, one can use the results of this section to obtain estimates for (physics-informed) DeepONets for nonlinear parabolic PDEs (4.23) such as the Allen–Cahn equation. In particular, a dimension-independent convergence rate can be obtained if the solution is smooth enough, improving upon the result of Lanthaler, Mishra and Karniadakis (2022), which incurred the CoD. For simplicity, we present results for $C^{(2,r)}$ functions, rather than $C^{(s,r)}$ functions, as we found that assuming more regularity did not necessarily improve the convergence rate further (De Ryck and Mishra 2022*b*, Theorem 4.5).

Theorem 4.27. Let $\mathcal{G}\colon \mathcal{X} \to C^r(\mathbb{T}^d)\colon u_0 \mapsto u(T)$ and $\mathcal{G}^*\colon \mathcal{X} \to C^{(2,r)}([0,T] \times \mathbb{T}^d)\colon u_0 \mapsto u$. For every $\sigma, \epsilon > 0$, there exists a DeepONet $\mathcal{G}^*_\theta$ such that

$$\|\mathcal{L}(\mathcal{G}^*_\theta)\|_{L^2([0,T]\times\mathbb{T}^d\times\mathcal{X})} \leq \epsilon. \tag{4.25}$$

Moreover, for $\mathcal{G}_\theta^*$ we have $O\left(\epsilon^{-\frac{(3+\sigma)d}{r-2}}\right)$ sensors and

$$\begin{aligned} \text{width}(\beta) &= O\left(\epsilon^{-1-\frac{(3+\sigma)(d+r(2+\beta))}{r-2}}\right), \quad \text{depth}(\beta) = O(\ln(1/\epsilon)), \\ \text{width}(\tau) &= O\left(\epsilon^{-1-\frac{(3+\sigma)(d+1)}{r-2}}\right), \qquad \text{depth}(\tau) = 3. \end{aligned} \tag{4.26}$$

4.3.2. Results based on Barron regularity

A second framework that can be used to prove that neural networks can overcome the curse of dimensionality is that of Barron spaces. These spaces are named after the seminal work of Barron (1993), who showed that a function f with Fourier transform $\widehat{f}$ can be approximated to accuracy $1/\sqrt{m}$ by a shallow neural network with m neurons, as long as

$$\int_{\mathbb{R}^d} |\widehat{f}(\xi)| \cdot |\xi| \, \mathrm{d}\xi < \infty. \tag{4.27}$$

Later, the space of functions satisfying condition (4.27) was named a *Barron space*, and its definition was generalized in many different ways. One notable generalization is that of *spectral Barron spaces*.

Definition 4.28. The spectral Barron space with index s, denoted $\mathcal{B}^s(\mathbb{R}^d)$, is defined as the collection of functions $f\colon \mathbb{R}^d \to \mathbb{R}$ for which the spectral Barron norm is finite:

$$\|f\|_{\mathcal{B}^s(\mathbb{R}^d)} := \int_{\mathbb{R}^d} |\widehat{f}(\xi)| \cdot (1 + |\xi|^2)^{s/2} \, \mathrm{d}\xi < \infty. \tag{4.28}$$

First, notice how the initial condition (4.27) of Barron (1993) corresponds to the spectral Barron space $\mathcal{B}^1(\mathbb{R}^d)$. Second, from this definition the difference between Barron spaces and Sobolev spaces becomes apparent: whereas the spectral Barron norm is the L^1-norm of $|\widehat{f}(\xi)| \cdot (1 + |\xi|^2)^{s/2}$, the Sobolev $H^s(\mathbb{R}^d)$-norm is defined as the L^2-norm of that same quantity. Finally, it follows that $\mathcal{B}^r(\mathbb{R}^d) \subset \mathcal{B}^s(\mathbb{R}^d)$ for $r \geq s$ and that $\mathcal{B}^0(\mathbb{R}^d) \subset L^\infty(\mathbb{R}^d)$; see e.g. Chen, Lu, Lu and Zhou (2023).

An example of an approximation result for functions in spectral Barron spaces can be found in Lu, Lu and Wang (2021*c*) for functions with the softplus activation function.

Theorem 4.29. Let $\Omega = [0,1]^d$. For every $u \in \mathcal{B}^2(\Omega)$, there exists a shallow neural network $\widehat{u}$ with softplus activation function ($\sigma(x) = \ln(1 + \exp(x))$, with width at most m, such that

$$\|u - \widehat{u}\|_{H^1(\Omega)} \leq \frac{\|u\|_{\mathcal{B}^2(\Omega)}(6\log(m) + 30)}{\sqrt{m}}. \tag{4.29}$$

Moreover, an exact bound on the weights is given in Lu *et al.* (2021*c*, Theorem 2.2).

There are two key questions that must be answered before this theorem can be used to argue that neural networks can overcome the CoD in the approximation of PDE solutions.

- Under which conditions does it hold that $u \in \mathcal{B}^2(\Omega)$?
- How does $\|u\|_{\mathcal{B}^2(\Omega)}$ depend on d?

Answering these questions requires us to build a regularity theory for Barron spaces, which can be challenging as they are not Hilbert spaces such as the Sobolev spaces H^s. An important contribution for high-dimensional elliptic equations has been made in Lu *et al.* (2021*c*).

Example 4.30 (Poisson's equation). Let $s \geq 0$, let $f \in \mathcal{B}^{s+2}(\Omega)$ satisfy

$$\int_\Omega f(x)\,\mathrm{d}x = 0$$

and let u be the unique solution to Poisson's equation $-\Delta u = f$ with zero Neumann boundary conditions. Then we have $u \in \mathcal{B}^s(\Omega)$ and $\|u\|_{\mathcal{B}^s(\Omega)} \leq d\|f\|_{\mathcal{B}^{s+2}(\Omega)}$ (Lu *et al.* 2021*c*, Theorem 2.6).

Example 4.31 (static Schrödinger equation). Let $s \geq 0$, let $V \in \mathcal{B}^{s+2}(\Omega)$ satisfy $V(x) \geq V_{\min} > 0$ for all $x \in \Omega$ and let u be the unique solution to the static Schrödinger's equation $-\Delta u + Vu = f$ with zero Neumann boundary conditions. Then we have $u \in \mathcal{B}^s(\Omega)$ and $\|u\|_{\mathcal{B}^s(\Omega)} \leq C\|f\|_{\mathcal{B}^{s+2}(\Omega)}$ (Chen *et al.* 2023, Theorem 2.3).

Similar results for a slightly different definition of Barron spaces (E, Ma and Wu 2022) can be found in E and Wojtowytsch (2022*b*), for example. E *et al.* also showed that the Barron space is the *right* space for two-layer neural network models in the sense that optimal direct and inverse approximation theorems hold. Moreover, E and Wojtowytsch (2020, 2022*a*) explored the connection between Barron and Sobolev spaces and provided examples of functions that are and functions that (sometimes surprisingly) are not Barron functions.

Another slightly different definition can be found in Chen *et al.* (2021), where a Barron function g is defined as a function that can be written as an infinite-width neural network,

$$g = \int a\sigma(w^\top x + b)\rho(\mathrm{d}a, \mathrm{d}w, \mathrm{d}b), \tag{4.30}$$

where ρ is a probability distribution on the (now infinite) parameter space Θ. In particular, the neural network

$$\frac{1}{k}\sum_{i=1}^{k} a_i\sigma\left(w_i^\top x + b_i\right)$$

is a Barron function.

Definition 4.32. Fix $\Omega \subset \mathbb{R}^d$ and $R \in [0, +\infty]$. For a function g as in (4.30), we define its Barron norm on Ω with index $p \in [1, +\infty]$ and support radius R by

$$\|g\|_{\mathcal{B}^p_R(\Omega)} = \inf_{\rho \in \mathcal{A}_g} \left\{ \left(\int |a|^p \rho(\mathrm{d}a, \mathrm{d}w, \mathrm{d}b) \right)^{1/p} \right\}, \tag{4.31}$$

where $\mathcal{A}_g$ is the set of probability measures ρ supported on $\mathbb{R} \times \overline{B}^d_R \times \mathbb{R}$, where

$$\overline{B}^d_R = \{x \in \mathbb{R}^d : \|x\| \leq R\},$$

such that

$$g = \int a\sigma(w^\top x + b)\rho(\mathrm{d}a, \mathrm{d}w, \mathrm{d}b).$$

The corresponding Barron space is then defined as

$$\mathcal{B}^p_R(\Omega) = \left\{ g : \|g\|_{\mathcal{B}^p_R(\Omega)} < \infty \right\}. \tag{4.32}$$

A number of interesting facts have been proved about this space, including the following.

- The Barron norms and spaces introduced in Definition 4.32 are independent of p, that is,

  $$\|g\|_{\mathcal{B}^p_R(\Omega)} = \|g\|_{\mathcal{B}^q_R(\Omega)},$$

  and hence

  $$\mathcal{B}^p_R(\Omega) = \mathcal{B}^q_R(\Omega) \quad \text{for any } p, q \in [1, +\infty].$$

  See Chen *et al.* (2021, Proposition 2.4) and also E *et al.* (2022, Proposition 1).
- Under some conditions on the activation function σ, the Barron space $\mathcal{B}^p_R(\Omega)$ is an algebra (Chen *et al.* 2021, Lemma 3.3).
- Neural networks can approximate Barron functions in Sobolev norm without incurring the curse of dimensionality.

We demonstrate the final point by slightly adapting Theorem 2.5 of Chen *et al.* (2021).

Theorem 4.33. Let σ be a smooth activation function, $\ell \in \mathbb{N}$, $R \geq 1$, let $\Omega \subset \mathbb{R}^d$ be open and let $f \in B^1_R(\Omega)$. Let μ be a probability measure on Ω and set $C_1 := \max_{m \leq \ell} \|\sigma^{(m)}\|_\infty < \infty$. For any $k \in N$ there exist $\{(a_i, w_i, b_i)\}_{i=1}^k$ such that

$$\left\| \frac{1}{k} \sum_{i=1}^k a_i \sigma\left(w_i^T x + b_i\right) - f(x) \right\|_{H^\ell_\mu(\Omega)} \leq \frac{2C_1(R\sqrt{ed})^\ell \|f\|_{B^1_R(\Omega)}}{\sqrt{k}}. \tag{4.33}$$

Proof. Since $f \in B^1_R(\Omega)$, there must exist a probability measure ρ supported on $\mathbb{R} \times \overline{B}^d_R \times \mathbb{R}$ such that

$$f(x) = \int a\sigma(w^T x + b)\rho(\mathrm{d}a, \mathrm{d}w, \mathrm{d}b), \qquad \int |a|^2 \,\mathrm{d}\rho \leq \frac{4}{\sqrt{\pi}} \|f\|_{B^2(\Omega)}, \tag{4.34}$$

where $4/\sqrt{\pi}$ is a constant that is strictly larger than 1, which will be convenient later on. Define the error

$$\mathcal{E}_k = \frac{1}{k} \sum_{i=1}^{k} a_i \sigma\left(w_i^T x + b_i\right) - f(x). \tag{4.35}$$

We calculate that, for $\ell = |\alpha|_1$,

$$\begin{aligned}
&\mathbb{E}\left[\|D^\alpha \mathcal{E}_k\|^2_{L^2(\Omega_0)}\right] \\
&= \iint_{\Omega_0} \left(\frac{1}{k} \sum_{i=1}^{k} a_i \sigma^l(w_i^T x + b_i) \prod_{j=1}^{d} w_{ij}^{\alpha_j} - D^\alpha f(x) \right)^2 \mathrm{d}\mu \,\mathrm{d}\rho^k \\
&= \frac{1}{k} \iint_{\Omega_0} \left(a\sigma^l(w^T x + b) \prod_{j=1}^{d} w_j^{\alpha_j} - D^\alpha f(x) \right)^2 \mathrm{d}\mu \,\mathrm{d}\rho \\
&= \frac{1}{k} \int_{\Omega_0} \mathrm{Var}\left(a\sigma^l(w^T x + b) \prod_{j=1}^{d} w_j^{\alpha_j} \right) \mathrm{d}\mu \\
&\leq \frac{1}{k} \int_{\Omega_0} \mathbb{E}\left[\left(a\sigma^l(w^T x + b) \prod_{j=1}^{d} w_j^{\alpha_j} \right)^2 \right] \mathrm{d}\mu \\
&\leq \frac{(C_1 R^l)^2}{k} \mathbb{E}[|a|^2] \leq \frac{4(C_1 R^l)^2 \|f\|^2_{B^2(\Omega)}}{k\sqrt{\pi}}.
\end{aligned} \tag{4.36}$$

Then, using Lemma 2.1 of De Ryck *et al.* (2021) and the previous inequality, we find that

$$\mathbb{E}\left[\|\mathcal{E}_k\|^2_{H^\ell(\Omega)}\right] \leq \sqrt{\pi}(ed)^\ell \max_{\substack{\alpha \in \mathbb{N}_0^d, \\ |\alpha|_1 \leq \ell}} \mathbb{E}[\|D^\alpha \mathcal{E}_k\|^2] \leq \frac{4C_1^2 (R^2 ed)^\ell \|f\|^2_{B^2(\Omega)}}{k}. \tag{4.37}$$

We can then conclude by using the fact that for a random variable Y that satisfies $\mathbb{E}[|Y|] \leq \epsilon$ for some $\epsilon > 0$, it must hold that $\mathbb{P}(|Y| \leq \epsilon) > 0$ (Grohs *et al.* 2018, Proposition 3.3). □

If we know how $\|f\|_{B^2(\Omega)}$ scales with d, then we can prove the existence of neural networks that overcome the CoD for physics-informed loss functions, under Assumption 4.1 and by following the approach of Section 4.1. More related works can be found in Bach (2017), Hong, Siegel and Xu (2021) and references therein.

5. Stability

Next, we investigate whether a small physics-informed loss implies a small total error, often the L^2-error, as formulated in the following question.

Question 2 (Q2). Given that a model u_θ has a small generalization error $\mathcal{E}_G(\theta)$, will the corresponding total error $\mathcal{E}(\theta)$ be small as well?

In the general results we will focus on physics-informed loss functions based on the strong (classical) formulation of the PDE, but we also give some examples when the loss is based on the weak solution (Example 5.8) or the variational solution (Example 5.11). We first look at stability results for forward problems (Section 5.1) and afterwards for inverse problems (Section 5.2).

5.1. *Stability for forward problems*

We investigate whether a small PDE residual implies that the total error (3.1) will be small as well (Question 2). Such a stability bound can be formulated as the requirement that for any $u, v \in X^*$, the differential operator $\mathcal{L}$ satisfies

$$\|u - v\|_X \leq C_{\text{PDE}} \|\mathcal{L}(u) - \mathcal{L}(v)\|_Y, \tag{5.1}$$

where the constant $C_{\text{PDE}} > 0$ is allowed to depend on $\|u\|_{X^*}$ and $\|v\|_{X^*}$.

As a first example of a PDE with solutions satisfying (5.1), we consider a linear differential operator $\mathcal{L}\colon X \to Y$, i.e. $\mathcal{L}(\alpha u + \beta v) = \alpha\mathcal{L}(u) + \beta\mathcal{L}(v)$, for any $\alpha, \beta \in \mathbb{R}$. For simplicity, let $X^* = X$ and $Y^* = Y$. By the assumptions on the existence and uniqueness of the underlying linear PDE (2.1), there exists an *inverse* operator $\mathcal{L}^{-1}\colon Y \to X$. Note that the assumption (5.1) is satisfied if the inverse is bounded i.e. $\|\mathcal{L}^{-1}\|_{\text{op}} \leq C < +\infty$, with respect to the natural norm on linear operators from Y to X. Thus the assumption (5.1) on stability boils down to the boundedness of the inverse operator for linear PDEs. Many well-known linear PDEs possess such bounded inverses (Dautray and Lions 1992).

As a second example, we will consider a nonlinear PDE (2.1), but with a well-defined linearization, that is, there exists an operator $\overline{\mathcal{L}}\colon X^* \mapsto Y^*$, such that

$$\mathcal{L}(u) - \mathcal{L}(v) = \overline{\mathcal{L}}_{(u,v)}(u - v) \quad \text{for all } u, v \in X^*. \tag{5.2}$$

Again for simplicity, we will assume that $X^* = X$ and $Y^* = Y$. We further assume that the inverse of $\overline{\mathcal{L}}$ exists and is bounded in the following manner:

$$\|\overline{\mathcal{L}}_{(u,v)}^{-1}\|_{\text{op}} \leq C(\|u\|_X, \|v\|_X) < +\infty \quad \text{for all } u, v \in X, \tag{5.3}$$

with the norm of $\overline{\mathcal{L}}^{-1}$ being an operator norm, induced by linear operators from Y to X. Then a straightforward calculation shows that (5.3) suffices to establish the stability bound (5.1).

We summarize our findings with the following informal theorem.

Theorem 5.1. Let $\mathcal{L}$ be a linear operator or a nonlinear operator with linearization as in (5.3). If $\mathcal{L}$ has a bounded inverse operator, then equation (5.1) will hold, that is, a small physics-informed loss will imply a small total error.

We give examples of the above theorem, with explicit stability constants, for the semilinear heat equation (strong formulation), Navier–Stokes equation (strong formulation), scalar conservation laws (strong and weak formulation) and the Poisson equation (variational formulation). Further examples can be found in Mishra and Molinaro (2023) and Lu *et al.* (2021*c*).

Example 5.2 (semilinear heat equation). We address the stability (Question 2) of the semilinear heat equation (see Example 2.1). The following theorem (Mishra and Molinaro 2023, Theorem 3.1) ensures that one can indeed bound the total error $\mathcal{E}(\theta)$ in terms of the generalization error $\mathcal{E}_G(\theta)$. A generalization to linear Kolmogorov equations (including the Black–Scholes model) can be found in De Ryck and Mishra (2022*a*, Theorem 3.7).

Theorem 5.3. Let $u \in C^k(\overline{D} \times [0,T])$ be the unique classical solution of the semilinear parabolic equation (2.6) with the source f satisfying (2.7), and let $v \in C^2(\overline{D} \times [0,T])$. Then the total error (3.1) can be estimated as

$$\begin{aligned} \|u - v\|^2_{L^2(D\times[0,T])} \leq C_1 \big[&\|\mathcal{R}_{\mathrm{PDE}}[v]\|^2_{L^2(D\times[0,T])} + \|\mathcal{R}_t[v]\|^2_{L^2(D)} \\ &+ C_2 \|\mathcal{R}_s[v]\|_{L^2(\partial D\times[0,T])} \big], \end{aligned} \tag{5.4}$$

with constants given by

$$\begin{aligned} C_1 &= \sqrt{T + (1+2C_f)T^2 \mathrm{e}^{(1+2C_f)T}}, \\ C_2 &= \sqrt{T^{1/2}|\partial D|^{1/2}\big(\|u\|_{C^1([0,T]\times\partial D)} + \|v\|_{C^1([0,T]\times\partial D)}\big)}. \end{aligned} \tag{5.5}$$

Proof. First, note that for $w = v - u$ we have

$$\partial_t w = \Delta w + f(v) - f(u) + \mathcal{R}_{\mathrm{PDE}}[v], \quad w(x,0) = \mathcal{R}_t[v](x), \quad v = \mathcal{R}_s[v]. \tag{5.6}$$

Multiplying this PDE by w, integrating over D and performing integration by parts then gives rise to the inequality

$$\begin{aligned} \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\int_D |w|^2 &\leq \int_{\partial D} \mathcal{R}_s[v]\nabla w \cdot \widehat{n}_D + \int_D w(f(v) - f(u) + \mathcal{R}_{\mathrm{PDE}}[v]) \\ &\lesssim \left(\int_{\partial D} |\mathcal{R}_s[v]|^2\right)^{1/2} + \int_D |w|^2 + \int_D |\mathcal{R}_{\mathrm{PDE}}[v]|^2. \end{aligned} \tag{5.7}$$

Integrating the above inequality over $[0,\tau] \subset [0,T]$, applying Grönwall's inequality and then integrating once again over time yields the claimed result. The full version can be found in Mishra and Molinaro (2023, Theorem 3.1). □

Example 5.4 (radiative transfer equation). The study of radiative transfer is of vital importance in many fields of science and engineering including astrophysics,

climate dynamics, meteorology, nuclear engineering and medical imaging (Modest 2003). The fundamental equation describing radiative transfer is a *linear partial integro-differential equation*, referred to as the *radiative transfer equation*. Under the assumption of a static underlying medium, it has the form (Modest 2003)

$$\frac{1}{c}u_t + \omega \cdot \nabla_x u + ku + \sigma\left(u - \frac{1}{s_d}\int_\Lambda \int_S \Phi(\omega, \omega', \nu, \nu')u(t, x, \omega', \nu')\,\mathrm{d}\omega'\,\mathrm{d}\nu'\right) = f, \tag{5.8}$$

with time variable $t \in [0, T]$, space variable $x \in D \subset \mathbb{R}^d$ (and $D_T = [0, T] \times D$), *angle* $\omega \in S = \mathbb{S}^{d-1}$, i.e. the d-dimensional sphere, and *frequency* (or group energy) $\nu \in \Lambda \subset \mathbb{R}$. The constants in (5.8) are the speed of light c and the surface area s_d of the d-dimensional unit sphere. The unknown of interest in (5.8) is the so-called *radiative intensity* $u\colon D_T \times S \times \Lambda \mapsto \mathbb{R}$, while $k = k(x, \nu)\colon D \times \Lambda \mapsto \mathbb{R}_+$ is the *absorption coefficient* and $\sigma = \sigma(x, \nu)\colon D \times \Lambda \mapsto \mathbb{R}_+$ is the *scattering coefficient*. The integral term in (5.8) involves the so-called *scattering kernel* $\Phi\colon S \times S \times \Lambda \times \Lambda \mapsto \mathbb{R}$, which is normalized as $\int_{S\times\Lambda} \Phi(\cdot, \omega', \cdot, \nu')\,\mathrm{d}\omega'\,\mathrm{d}\nu' = 1$, in order to account for the conservation of photons during scattering. The dynamics of radiative transfer are driven by a source (emission) term $f = f(x, \nu)\colon D \times \Lambda \mapsto \mathbb{R}$.

Although the radiative transfer equation (5.8) is linear, explicit solution formulas are only available in very special cases (Modest 2003). Hence, numerical methods are essential for the simulation of the radiative intensity. Fortunately, Mishra and Molinaro (2021) have provided an affirmative answer to Question 2 for the radiative transfer equations, so that one can use physics-informed techniques (based on the strong PDE residual) to retrieve an approximation of the solution of (5.8).

Theorem 5.5. Let $u \in L^2(\Omega)$ be the unique weak solution of the radiative transfer equation (5.8), with absorption coefficient $0 \leq k \in L^\infty(D \times \Lambda)$, scattering coefficient $0 \leq \sigma \in L^\infty(D \times \Lambda)$ and a symmetric scattering kernel $\Phi \in C^\ell(S \times \Lambda \times S \times \Lambda)$, for some $\ell \geq 1$, such that the function Ψ, given by

$$\Psi(\omega, \nu) = \int_{S\times\Lambda} \Phi(\omega, \omega', \nu, \nu')\,\mathrm{d}\omega'\,\mathrm{d}\nu', \tag{5.9}$$

is in $L^\infty(S \times \Lambda)$. For any sufficiently smooth model u_θ we have

$$\|u - u_\theta\|_{L^2}^2 \leq C\left(\|\mathcal{R}_{\text{PDE}}[u_\theta]\|_{L^2}^2 + \|\mathcal{R}_s[u_\theta]\|_{L^2}^2 + \|\mathcal{R}_t\|_{L^2}^2\right), \tag{5.10}$$

where $C > 0$ is a constant that only depends (in a monotonously increasing way) on T and the quantity $s_d^{-1}(\|\sigma\|_{L^\infty} + \|\Psi\|_{L^\infty})$, where s_d is the surface area of the d-dimensional unit sphere.

Example 5.6 (Navier–Stokes equations). Next we revisit Example 2.2 to show that neural networks with small PINN residuals will provide a good L^2-approximation of the true solution $u\colon \Omega = D \times [0, T] \to \mathbb{R}^d$, $p\colon \Omega \to \mathbb{R}$ of the Navier–Stokes equation (2.8) on the torus $D = \mathbb{T}^d = [0, 1)^d$ with periodic boundary conditions. The analysis can be readily extended to other boundary conditions, such as no-slip

boundary conditions, i.e. $u(x,t) = 0$ for all $(x,t) \in \partial D \times [0,T]$, and no-penetration boundary conditions, i.e. $u(x,t) \cdot \widehat{n}_D = 0$ for all $(x,t) \in \partial D \times [0,T]$.

For neural networks (u_θ, p_θ), we define the following PINN-related residuals:

$$\begin{aligned} \mathcal{R}_{\text{PDE}} &= \partial_t u_\theta + (u_\theta \cdot \nabla) u_\theta + \nabla p_\theta - \nu \Delta u_\theta, \quad \mathcal{R}_{\text{div}} = \operatorname{div} u_\theta, \\ \mathcal{R}_{s,u}(x) &= u_\theta(x) - u_\theta(x+1), \quad \mathcal{R}_{s,p}(x) = p_\theta(x) - p_\theta(x+1), \\ \mathcal{R}_{s,\nabla u}(x) &= \nabla u_\theta(x) - \nabla u_\theta(x+1), \quad \mathcal{R}_s = (\mathcal{R}_{s,u}, \mathcal{R}_{s,p}, \mathcal{R}_{s,\nabla u}), \\ \mathcal{R}_t &= u_\theta(t=0) - u(t=0), \end{aligned} \tag{5.11}$$

where we drop the θ-dependence in the definition of the residuals for notational convenience.

The following theorem (De Ryck *et al.* 2024*a*) then bounds the L^2-error of the PINN in terms of the residuals defined above. We write $|\partial D|$ for the $(d-1)$-dimensional Lebesgue measure of ∂D and $|D|$ for the d-dimensional Lebesgue measure of D.

Theorem 5.7. Let $d \in \mathbb{N}$, $D = \mathbb{T}^d$ and $u \in C^1(D \times [0,T])$ be the classical solution of the Navier–Stokes equation (2.8). Let (u_θ, p_θ) be a PINN with parameters θ. Then the resulting L^2-error is bounded as follows:

$$\int_\Omega \|u(x,t) - u_\theta(x,t)\|_2^2 \, \mathrm{d}x \, \mathrm{d}t \leq \mathcal{C} T \exp(T(2d^2 \|\nabla u\|_{L^\infty(\Omega)} + 1)), \tag{5.12}$$

where the constant $\mathcal{C}$ is defined as

$$\begin{aligned} \mathcal{C} = \|\mathcal{R}_t\|_{L^2(D)}^2 + \|\mathcal{R}_{\text{PDE}}\|_{L^2(\Omega)}^2 + C_1 \sqrt{T} \Big[\sqrt{|D|} \|\mathcal{R}_{\text{div}}\|_{L^2(\Omega)} \\ + (1+\nu) \sqrt{|\partial D|} \|\mathcal{R}_s\|_{L^2(\partial D \times [0,T])} \Big], \end{aligned} \tag{5.13}$$

and

$$C_1 = C_1\big(\|u\|_{C^1}, \|u_\theta\|_{C^1}, \|p\|_{C^0}, \|p_\theta\|_{C^0}\big) < \infty.$$

Proof. The proof is in a similar spirit to that of Theorem 5.3 and can be found in De Ryck *et al.* (2024*a*). □

Example 5.8 (scalar conservation law). In this example we consider viscous scalar conservation laws, as introduced in Example 2.3. We first assume that the solution to it is sufficiently smooth, so that we can use the strong PDE residual as in Section 2.3.1. We can then prove the following stability bound (Mishra and Molinaro 2023, Theorem 4.1).

Theorem 5.9. Let $\Omega = (0,T) \times (0,1)$, $\nu > 0$ and let $u \in C^k(\Omega)$ be the unique classical solution of the viscous scalar conservation law (2.9). Let $u^* = u_{\theta^*}$ be the PINN generated according to Section 2.6. Then the total error is bounded by

$$\begin{aligned} \|u - u_\theta\|_{L^2(\Omega)}^2 \leq (T + C_1 T^2 \mathrm{e}^{C_1 T}) \big[\|\mathcal{R}_{\text{PDE}}[u_\theta]\|_{L^2(\Omega)}^2 \\ + \|\mathcal{R}_t[u_\theta]\|_{L^2(D)}^2 + 2C_2 \|\mathcal{R}_s[u_\theta]\|_{L^2(\partial D \times [0,T])}^2 \\ + 2\nu \sqrt{T} C_3 \|\mathcal{R}_s[u_\theta]\|_{L^2(\partial D \times [0,T])} \big]. \end{aligned} \tag{5.14}$$

Here the constants are defined as

$$C_1 = 1 + 2|f''(\max\{\|u\|_{L^\infty}, \|u_\theta\|_{L^\infty}\})|\|u_x\|_{L^\infty}, \tag{5.15}$$

$$C_2 = \|\partial_x u\|_{C(\overline{\Omega})} + \|\partial_x u_\theta\|_{C(\overline{\Omega})}, \tag{5.16}$$

$$C_3 = C_3(\|f'\|_\infty, \|u_\theta\|_{C^0(\Omega)}). \tag{5.17}$$

The proof and further details can be found in Mishra and Molinaro (2023). A close inspection of the estimate (5.14) reveals that at the very least, the classical solution u of the PDE (2.9) needs to be in $L^\infty((0,T); W^{1,\infty}((0,1)))$ for the right-hand side of (5.14) to be bounded. This indeed holds as long as $\nu > 0$. However, it is well known (see Godlewski and Raviart (1991) and references therein) that if u^ν is the solution of (2.9) for viscosity ν, then, for some initial data,

$$\|u^\nu\|_{L^\infty((0,T);W^{1,\infty}((0,1)))} \sim \frac{1}{\sqrt{\nu}}. \tag{5.18}$$

Thus, in the limit $\nu \to 0$, the constant C_1 can blow up (exponentially in time) and the bound (5.14) no longer controls the generalization error. This is not unexpected as the whole strategy of this paper relies on pointwise realization of residuals. However, the zero-viscosity limit of (2.9) leads to a scalar conservation law with discontinuous solutions (shocks), and the residuals are measures that do not make sense pointwise. Thus the estimate (5.14) also points out the limitations of a PINN for approximating discontinuous solutions.

The above discussion is also the perfect motivation to consider the weak residual, rather than the strong residual, for scalar conservation laws. De Ryck *et al.* (2024*c*) therefore introduce a weak PINN (wPINN) formulation for scalar conservation laws. The wPINN loss function also reflects the fact that *physically admissible* weak solutions should also satisfy an entropy condition, giving rise to an entropy residual. The details of this weak residual and the corresponding loss function were already discussed in Example 2.10. Using the famous doubling of variables argument of Kruzkhov, one can prove the following stability bound on the L^1-error with *wPINNs* (De Ryck *et al.* 2024*c*, Theorem 3.7).

Theorem 5.10. Assume that u is the piecewise smooth entropy solution of (2.9) with $\nu = 0$, essential range $\mathcal{C}$ and $u(0,t) = u(1,t)$ for all $t \in [0,T]$. There is a constant $C > 0$ such that, for every $\epsilon > 0$ and $u_\theta \in C^1(D \times [0,T])$, we have

$$\begin{aligned}
&\int_0^1 |u_\theta(x,T) - u(x,T)|\,\mathrm{d}x \\
&\quad \le C\left(\int_0^1 |u_\theta(x,0) - u(x,0)|\,\mathrm{d}x + \max_{c\in\mathcal{C},\varphi\in\Phi_\epsilon} \mathcal{R}(u_\theta,\varphi,c)\right. \\
&\qquad \left. + (1 + \|u_\theta\|_{C^1})\ln(1/\epsilon)^3\epsilon + \int_0^T |u_\theta(1,t) - u_\theta(0,t)|\,\mathrm{d}t\right).
\end{aligned} \tag{5.19}$$

Whereas the bound of Theorem 5.9 becomes unusable in the low viscosity regime ($\nu \ll 1$), the bound of Theorem 5.10 is still valid if the solution contains shocks.

Hence we can give an affirmative answer to Question 2 for scalar conservation laws using both the PDE residual (classical formulation, only for $\nu > 0$) and the weak residual (for $\nu = 0$).

Example 5.11 (Poisson's equation). We revisit Poisson's equation (Example 2.4) for $\Omega = [0,1]^d$, but now with zero Neumann conditions, i.e. $\partial u/\partial \nu = 0$ on $\partial\Omega$, and with $f \in L^2(\Omega)$ with $\int_\Omega f \,\mathrm{d}x = 0$. Following Section 2.3.3, the solution of Poisson's equation is a minimizer of the following loss function, which is equal to the energy functional or variational formulation of the PDE

$$I[w] := \frac{1}{2}\int_\Omega |\nabla w|^2 \,\mathrm{d}x + \frac{1}{2}\left(\int_\Omega w \,\mathrm{d}x\right)^2 - \int_\Omega f w \,\mathrm{d}x. \tag{5.20}$$

Now define $\widetilde{u} = \operatorname{argmin}_{w \in H^1(\Omega)} I[w]$ and define

$$\mathcal{E}_G(\theta) := I[u_\theta] - I[\widetilde{u}], \tag{5.21}$$

where we have slightly adapted the definition of the generalization error to reflect the fact that $I[\cdot]$ merely needs to be minimized and does not need to vanish to produce a good approximation. In this setting, one can prove the following stability result (Lu *et al.* 2021*c*).

Proposition 5.12. For any $w \in H^1(\Omega)$,

$$2(I[w] - I[\widetilde{u}]) \le \|w - \widetilde{u}\|^2_{H^1(\Omega)} \le 2\max\{2C_P + 1, 2\}(I[w] - I[\widetilde{u}]), \tag{5.22}$$

where C_P is the Poincaré constant on the domain Ω, that is, for any $v \in H^1(\Omega)$,

$$\left\| v - \int_\Omega v \,\mathrm{d}x \right\|^2_{L^2(\Omega)} \le C_P \|\nabla v\|^2_{L^2(\Omega)}. \tag{5.23}$$

In particular, by setting $w = u_\theta$ this implies that

$$\mathcal{E}(\theta)^2 := \|u_\theta - \widetilde{u}\|^2_{H^1(\Omega)} \le 2\max\{2C_P + 1, 2\}\mathcal{E}_G(\theta). \tag{5.24}$$

The examples and theorems above might give the impression that stability holds for all PDEs and loss functions. It is important to highlight that this is not the case. The next example discusses a negative answer to Question 2 for the Hamilton–Jacobi–Bellman (HJB) equation when using an L^2-based loss function (Wang, Li, He and Wang 2022*a*).

Example 5.13 (Hamilton–Jacobi–Bellman equation). Wang *et al.* (2022*a*) consider the class of HJB equations in which the cost rate function is formulated as $r(x,m) = a_1|m_1|^{\alpha_1} + \cdots + a_n|m_n|^{\alpha_n} - f(x,t)$, giving rise to HJB equations of the

form

$$\begin{cases} \mathcal{L}[u] := \partial_t u(x,t) + \frac{1}{2}\sigma^2 \Delta u(x,t) - \sum_{i=1}^{n} A_i |\partial_{x_i} u|^{c_i} = f(x,t), \\ \mathcal{B}[u] := u(x,T) = g(x), \end{cases} \tag{5.25}$$

for $x \in \mathbb{R}^d$ and $t \in [0,T]$ and where $A_i = (a_i\alpha_i)^{-1/(\alpha_i-1)} - a_i(a_i\alpha_i)^{-\alpha_i/(\alpha_i-1)} \in (0,+\infty)$ and $c_i = \alpha_i/(\alpha_i - 1) \in (1,\infty)$. Their chosen form of the cost function is relevant in optimal control, e.g. in optimal execution problems in finance. One of the main results in Wang *et al.* (2022*a*) is that stability can only be achieved when the physics-informed loss is based on the L^p-norm of the PDE residual with $p \sim d$. They show that this linear dependence on d cannot be relaxed in the following theorem (Wang *et al.* 2022*a*, Theorem 4.3).

Theorem 5.14. There exists an instance of the HJB equation (5.25), with exact solution u, such that for any $\varepsilon > 0$, $A > 0$, $r \geq 1$, $m \in \mathbb{N} \cup \{0\}$ and $p \in [1, d/4]$, there exists a function $w \in C^\infty(\mathbb{R}^n \times (0,T])$ such that $\text{supp}(w-u)$ is compact and

$$\|\mathcal{L}[w] - f\|_{L^p(\mathbb{R}^d \times [0,T])} < \varepsilon, \quad \mathcal{B}[w] = \mathcal{B}[u], \tag{5.26}$$

and yet simultaneously

$$\|w - u\|_{W^{m,r}(\mathbb{R}^d \times [0,T])} > A. \tag{5.27}$$

This shows that for high-dimensional HJB equations, the learned solution may be arbitrarily distant from the true solution u if an L^2-based physics-inspired loss based on the classical residual is used.

Another example showing that stability of physics-informed learning is not guaranteed can be found in the following two variants on physics-informed neural networks.

Example 5.15 (XPINNs and cPINNs). Jagtap and Karniadakis (2020) and Jagtap, Kharazmi and Karniadakis (2020) proposed two variants on PINNs, inspired by domain decomposition techniques, in which separate neural networks are trained on different subdomains. In order to ensure continuity of the different subnetworks over the boundaries of the subdomains, both methods add a first term to the loss function to enforce that the function values of the subnetworks are (approximately) equal on the boundaries. *Extended PINNs* (XPINNs) (Jagtap and Karniadakis 2020) then propose adding another additional term to make sure the strong PDE residual is (approximately) continuous over the boundaries of the subdomains. In contrast, *conservative PINNs* (cPINNs) (Jagtap *et al.* 2020) focus on PDEs of the form $u_t + \nabla_x f(u) = 0$ (i.e. conservation laws) and add an additional term to make sure that the fluxes based on f are (approximately) equal over the boundaries. De Ryck, Jagtap and Mishra (2024*b*) have given an affirmative answer to Question 2 for cPINNs for the Navier–Stokes equations, advection–diffusion equations and scalar conservation laws. On the other hand, XPINNs were not found to be

stable (in the sense of Question 2) for prototypical PDEs such as Poisson's equation and the heat equation.

Finally, we highlight some other works that have proved stability results for physics-informed machine learning. Shin (2020) proved a consistency result for PINNs, for linear elliptic and parabolic PDEs, where they show that if $\mathcal{E}_G(\theta_m) \to 0$ for a sequence of neural networks $\{u_{\theta_m}\}_{m\in\mathbb{N}}$, then $\|u_{\theta_m} - u\|_{L^\infty} \to 0$, under the assumption that we add a specific $C^{k,\alpha}$-regularization term to the loss function, thus partially addressing Question 2 for these PDEs. However, this result does not provide quantitative estimates on the underlying errors. A similar result, with more quantitative estimates for advection equations is provided in Shin, Zhang and Karniadakis (2023).

5.2. *Stability for inverse problems*

Next, we extend our analysis to inverse problems, as introduced in Section 2.1.2. A crucial ingredient to answer Question 2 for inverse problems will be the assumption that solutions to the inverse problem, defined by (2.1) and (2.12), satisfy the following *conditional stability estimate*. Let $\widehat{X} \subset X^* \subset X = L^p(\Omega)$ be a Banach space. For any $u, v \in \widehat{X}$, the differential operator $\mathcal{L}$ and restriction operator Ψ satisfy

$$\|u - v\|_{L^p(E)} \leq C_{\mathrm{pd}} \left(\|\mathcal{L}(u) - \mathcal{L}(v)\|_Y^{\tau_p} + \|\Psi(u) - \Psi(v)\|_Z^{\tau_d} \right) \tag{5.28}$$

for some $0 < \tau_p, \tau_d \leq 1$, for any subset $\Omega' \subset E \subset \Omega$ and where C_{pd} can depend on $\|u\|_{\widehat{X}}$ and $\|v\|_{\widehat{X}}$. This bound (5.28) is termed a *conditional stability estimate* as it presupposes that the underlying solutions have sufficiently regularity as $\widehat{X} \subset X^* \subset X$.

Remark 5.16. We can extend the hypothesis for the inverse problem as follows.

- Allow the measurement set Ω' to intersect the boundary, i.e. $\partial\Omega' \cap \Omega \neq \emptyset$.
- Replace the bound (5.28) with the weaker bound

$$\|u - v\|_{L^p(E)} \leq C_{\mathrm{pd}} \omega(\|\mathcal{L}(u) - \mathcal{L}(v)\|_Y + \|\Psi(u) - \Psi(v)\|_Z), \tag{5.29}$$

  with $\omega \colon \mathbb{R} \mapsto \mathbb{R}_+$ being a modulus of continuity.

We will prove a general estimate on the error due to a model u_θ in approximating the solution u of the inverse problem for PDE (2.1) with data (2.12). Following Mishra and Molinaro (2023) we set $\Omega' \subset E \subset \Omega$ and define the total error (3.1) as

$$\mathcal{E}(E; \theta) := \|u - u_\theta\|_{L^p(E)}. \tag{5.30}$$

We will bound the error in terms of the PDE residual (2.41) and the data residual (2.47) (Mishra and Molinaro 2022, Theorem 2.4).

Theorem 5.17. Let $u \in \widehat{X}$ be the solution of the inverse problem associated with PDE (2.1) and data (2.12). Assume that the stability hypothesis (5.28) holds for

any $\Omega' \subset E \subset \Omega$. Let u_θ be any sufficiently smooth function. Assume that the residuals $\mathcal{R}_{\text{PDE}}$ and $\mathcal{R}_d$ are square-integrable. Then the following estimate on the generalization error (5.30) holds:

$$\|u - u_\theta\|_{L^2(E)} \leq C_{\text{pd}}\left(\|\mathcal{R}_{\text{PDE}}[u_\theta]\|^{\tau_p}_{L^2(\Omega)} + \|\mathcal{R}_d[u_\theta]\|^{\tau_d}_{L^2(\Omega')}\right), \tag{5.31}$$

with constants $C_{\text{pd}} = C_{\text{pd}}\left(\|u\|_{\widehat{X}}, \|u^*\|_{\widehat{X}}\right)$ as from (5.28).

Many concrete examples of PDEs where PINNs are used to find a unique continuation can be found in Mishra and Molinaro (2022). The following examples summarize their results for the Poisson equation and heat equation.

Example 5.18 (Poisson equation). We consider the inverse problem for the Poisson equation, as introduced in Example 2.6. In this case, the conditional stability (see (5.28)) is guaranteed by the three balls inequality (Alessandrini, Rondi, Rosset and Vessella 2009). Therefore a result like Theorem 5.17 holds. Note that this theorem only calculates the generalization error on $E \subset \Omega$. However, it can be guaranteed that the generalization error is small on the whole domain Ω and even in Sobolev norm, as follows from the next lemma (Mishra and Molinaro 2022, Lemma 3.3).

Lemma 5.19. For $f \in C^{k-2}(\Omega)$ and $g \in C^k(\Omega')$, with continuous extensions of the functions and derivatives up to the boundaries of the underlying sets and with $k \geq 2$, let $u \in H^1(\Omega)$ be the weak solution of the inverse problem corresponding to the Poisson's equation (2.16) and let u_θ be any sufficiently smooth model. Then the total error is bounded by

$$\|u - u_\theta\|_{H^1(D)} \leq C\left|\log\left(\|\mathcal{R}_{\text{PDE}}[u_\theta]\|_{L^2(\Omega)} + \|\mathcal{R}_d[u_\theta]\|_{L^2(\Omega')}\right)\right|^{-\tau} \tag{5.32}$$

for some $\tau \in (0, 1)$ and a constant $C > 0$ depending on u, u_θ and τ.

Example 5.20 (heat equation). We consider the data assimilation problem for the heat equation, as introduced in Example 2.5, which amounts to finding the solution u of the heat equation in the whole space–time domain $\Omega = D \times (0, T)$, given data on the observation subdomain $\Omega' = D' \times (0, T)$. For any $0 \leq \tau < T$, we define the error of interest for the model u_θ as

$$\mathcal{E}_\tau(\theta) = \|u - u_\theta\|_{C([\tau,T];L^2(D))} + \|u - u_\theta\|_{L^2((0,T);H^1(D))}. \tag{5.33}$$

The theory for this data assimilation inverse problem for the heat equation is classical, and several well-posedness and stability results are available. Our subsequent error estimates for physics-informed models rely on a classical result of Imanuvilov (1995), based on the well-known Carleman estimates. Using these results, one can state the following theorem (Mishra and Molinaro 2022, Lemma 4.3).

Theorem 5.21. For $f \in C^{k-2}(\Omega)$ and $g \in C^k(\Omega')$, with continuous extensions of the functions and derivatives up to the boundaries of the underlying sets and with $k \geq 2$, let $u \in H^1((0, T); H^{-1}(D)) \cap L^2((0, T); H^1(D))$ be the solution of the

inverse problem corresponding to the heat equation and that satisfies (2.15). Then, for any $0 \leq \tau < T$, the error (5.33) corresponding to the sufficiently smooth model u_θ is bounded by

$$\mathcal{E}_\tau(\theta) \leq C\big(\|\mathcal{R}_{\mathrm{PDE}}[u_\theta]\|_{L^2(\Omega)} + \|\mathcal{R}_d[u_\theta]\|_{L^2(\Omega')} + \|\mathcal{R}_s[u_\theta]\|_{L^2(\partial D\times(0,T))}\big) \quad (5.34)$$

for some constant C depending on τ, u and u_θ.

Example 5.22 (Stokes equation). The effectiveness of PINNs in approximating inverse problems was recently showcased by Raissi, Yazdani and Karniadakis (2018), who proposed PINNs for the data assimilation problem with the Navier–Stokes equations (Example 2.2). As a first step towards rigorously analysing this, we follow Mishra and Molinaro (2022) and focus on the stationary Stokes equation as introduced in Example 2.7.

Recall that the data assimilation inverse problem for the Stokes equation amounts to inferring the velocity field u (and the pressure p)), given f, f_d and g. In particular, we wish to find solutions $u \in H^1(D;\mathbb{R}^d)$ and $p \in L^2_0(D)$ (i.e. square-integrable functions with zero mean), such that the following holds:

$$\begin{aligned} \int_D \nabla u \cdot \nabla v \,\mathrm{d}x + \int_D p \,\mathrm{div}(v)\,\mathrm{d}x &= \int_D f v \,\mathrm{d}x, \\ \int_D \mathrm{div}(u) w \,\mathrm{d}x &= \int_D f_d w \,\mathrm{d}x, \end{aligned} \quad (5.35)$$

for all test functions $v \in H^1_0(D;\mathbb{R}^d)$ and $w \in L^2(D)$.

Let $B_{R_1}(x_0)$ be the largest ball inside the observation domain $D' \subset D$. We will consider balls $B_R(x_0) \in D$ such that $R > R_1$ and estimate the following error for the model u_θ:

$$\mathcal{E}_R(\theta) := \|u - u_\theta\|_{L^2(B_R(x_0))}. \quad (5.36)$$

The well-posedness and conditional stability estimates for the data assimilation problem for the Stokes equation (2.17)–(2.18) have been extensively investigated in Lin, Uhlmann and Wang (2010) and references therein. Using these results, we can state the following estimate on $\mathcal{E}_R(\theta)$ (Mishra and Molinaro 2022, Lemma 6.2). The physics-informed residuals $\mathcal{R}_{\mathrm{PDE}}$ and $\mathcal{R}_{\mathrm{div}}$ are defined analogously to those in Example 5.6.

Theorem 5.23. Let $f \in C^{k-2}(D;\mathbb{R}^d)$, $f_d \in C^{k-1}(D)$ and $g \in C^k(D')$, with $k \geq 2$. Further, let $u \in H^1(D;\mathbb{R}^d)$ and $p \in H^1(D)$ be the solution of the inverse problem corresponding to the Stokes equations (2.17), that is, they satisfy (5.35) for all test functions $v \in H^1_0(D;\mathbb{R}^d)$, $w \in L^2(D)$ and satisfy the data (2.18). Let u_θ be a sufficiently smooth model and let $B_{R_1}(x_0)$ be the largest ball inside $D' \subset D$. Then there exists $\tau \in (0,1)$ such that the generalization error (5.36) for balls $B_R(x_0) \subset D$

with $R > R_1$ is bounded by

$$\mathcal{E}_R(\theta)^2 \leq C \cdot \max_{\gamma \in \{1,\tau\}} \left(\|\mathcal{R}_{\text{PDE}}[u_\theta]\|^2_{L^2(D)} + \|\mathcal{R}_{\text{div}}[u_\theta]\|^2_{L^2(D)} \right)^\gamma \left(1 + \|\mathcal{R}_d[u_\theta]\|^{2\tau}_{L^2(D)}\right), \tag{5.37}$$

where C depends on u and u_θ.

Further examples can be found in Mishra and Molinaro (2022).

6. Generalization

Now we turn our attention to Question 3.

Question 3 (Q3). Given a small training error $\mathcal{E}_T^*$ and a sufficiently large training set $\mathcal{S}$, will the corresponding generalization error $\mathcal{E}_G^*$ also be small?

We will answer this question by proving that for any model u_θ,

$$\mathcal{E}_G(\theta) \leq \mathcal{E}_T(\theta) + \epsilon(\theta), \tag{6.1}$$

for some value $\epsilon(\theta) > 0$ that depends on the model class and the size of the training set. Using the terminology of the error decomposition (3.7), this will then imply that the *generalization gap* is small:

$$\sup_{\theta \in \Theta} |\mathcal{E}_T(\theta, \mathcal{S}) - \mathcal{E}_G(\theta)| \leq \sup_{\theta \in \Theta} \epsilon(\theta). \tag{6.2}$$

As $\epsilon(\theta)$ can depend on θ, we must choose the model (parameter) space Θ in a suitable way to avoid $\sup_{\theta \in \Theta} \epsilon(\theta)$ diverging.

In view of Section 2.4, one can see that the training error is nothing more than applying a suitable quadrature rule to each term in the generalization error. It is clear that the error bound for a quadrature rule immediately proves that the generalization error can be bounded in terms of the training error and the training set size, thereby answering Question 3, at least for deterministic quadrature rules. We can now combine this generalization result with the stability bounds from Section 5 to prove an upper bound on the total error of the optimized model $\mathcal{E}^*$ (3.1) (Mishra and Molinaro 2023, Theorem 2.6).

Theorem 6.1. Let $u \in X^*$ be the unique solution of the PDE (2.1) and assume that the stability hypothesis (5.1) holds. Let $u_\theta \in X^*$ be a model and let $\mathcal{S}$ be a training set of quadrature points corresponding to the quadrature rule (2.65) with order α, as in (2.66). Then, for any L^2-based generalization error $\mathcal{E}_G$, such as (2.45) or (2.46),

$$\mathcal{E}_G(\theta)^2 \leq \mathcal{E}_T(\theta)^2 + C_{\text{quad}} N^{-\alpha}, \tag{6.3}$$

with constant C_{quad} stemming from (2.66), and which might depend on u_θ.

The approach described above only makes sense for low to moderately high dimensions ($d < 20$), since the convergence rate of numerical quadrature rules with

deterministic quadrature decreases with increasing d. For problems in very high dimensions, $d \gg 20$, Monte Carlo quadrature is the numerical integration method of choice. In this case, the quadrature points are randomly chosen, independent and identically distributed (i.i.d.) (with respect to a scaled Lebesgue measure). Since in this case the optimized model is correlated with all training points, we must be careful when calculating the convergence rate. We can, however, prove error bounds that hold with a certain probability or, alternatively, that hold averaged over all possible training sets of the same size.

In this setting, we prove a general *a posteriori* upper bound on the generalization error. Consider $\mathcal{F}\colon \mathcal{D} \to \mathbb{R}$ (an operator or function) and a corresponding model $\mathcal{F}_\theta\colon \mathcal{D} \to \mathbb{R}$, $\theta \in \Theta$. Given a training set $\mathcal{S} = \{X_1, \dots X_N\}$, where $\{X_i\}_{i=1}^N$ are i.i.d. random variables on $\mathcal{D}$ (according to a measure μ), the training error $\mathcal{E}_T$ and generalization error $\mathcal{E}$ are

$$\mathcal{E}_T(\theta, \mathcal{S})^2 = \frac{1}{N}\sum_{i=1}^{N} |\mathcal{F}(z_i) - \mathcal{F}_\theta(z_i)|^2, \quad \mathcal{E}_G(\theta)^2 = \int_{\mathcal{D}} |\mathcal{F}_\theta(z) - \mathcal{F}(z)|^2 \,\mathrm{d}\mu(z), \tag{6.4}$$

where μ is a probability measure on $\mathcal{D}$. This setting allows us to bound all possible terms and residuals that were mentioned in Section 2.3.

- For the term resulting from the PDE residual (2.41), we can set $\mathcal{D} = \Omega$, $\mathcal{F} = 0$ and $\mathcal{F}_\theta = \mathcal{L}[u_\theta] - f$.
- For the data term (2.47), we can set $\mathcal{D} = \Omega'$, $\mathcal{F} = u$ and $\mathcal{F}_\theta = u_\theta$. Similarly, for the term arising from the spatial boundary conditions, we set $\mathcal{D} = \partial D$ (or $\mathcal{D} = \partial D \times [0, T]$), and for the term arising from the initial condition, we set $\mathcal{D} = D$.
- For operator learning with an input function space $\mathcal{X}$, we can set $\mathcal{D} = \Omega \times \mathcal{X}$, $\mathcal{F} = \mathcal{G}$ and $\mathcal{F}_\theta = \mathcal{G}_\theta$.
- Finally, for physics-informed operator learning, we can set $\mathcal{D} = \Omega \times \mathcal{X}$, $\mathcal{F} = 0$ and $\mathcal{F}_\theta = \mathcal{L}(\mathcal{G}_\theta)$.

With the above definitions in place, we can state the following theorem (De Ryck and Mishra 2022*b*, Theorem 3.11), which provides a computable *a posteriori* error bound on the expectation of the generalization error for a general class of approximators. We refer to Beck, Jentzen and Kuckuck (2022) and De Ryck and Mishra (2022*a*), for example, for bounds on n, c and $\mathfrak{L}$.

Theorem 6.2. For $R > 0$ and $N, n \in \mathbb{N}$, let $\Theta = [-R, R]^n$ be the parameter space, and for every training set $\mathcal{S}$, let $\theta^*(\mathcal{S}) \in \Theta$ be an (approximate) minimizer of $\theta \mapsto \mathcal{E}_T(\theta, \mathcal{S})^2$, assume that $\theta \mapsto \mathcal{E}(\theta, \mathcal{S})^2$ and $\theta \mapsto \mathcal{E}_T(\theta, \mathcal{S})^2$ are bounded by $c > 0$ and Lipschitz-continuous with Lipschitz constant $\mathfrak{L} > 0$. If $N \geq 2c^2 \mathrm{e}^8 / (2R\mathfrak{L})^{n/2}$,

then

$$\mathbb{E}[\mathcal{E}_G(\theta^*(\mathcal{S}))^2] \leq \mathbb{E}[\mathcal{E}_T(\theta^*(\mathcal{S}), \mathcal{S})^2] + \sqrt{\frac{2c^2(n+1)}{N} \ln(R\mathfrak{L}\sqrt{N})}. \tag{6.5}$$

Proof. The proof combines standard techniques, based on covering numbers and Hoeffding's inequality, with an error composition from De Ryck and Mishra (2022*a*). □

Due to its generality, Theorem 6.2 provides a satisfactory answer to Question 3 when a randomly generated training set is used. However, the two central assumptions, the existence of the constants $c > 0$ and $\mathfrak{L} > 0$, should not be swept under the rug. In particular, for neural networks it might initially be unclear how large these constants are.

For any type of neural network architecture of depth L, width W and weights bounded by R, we find that $N \sim LW(W+d)$. For tanh neural networks and operator learning architectures, we have $\ln(\mathfrak{L}) \sim L \ln(dRW)$, whereas for physics-informed neural networks and DeepONets we find that $\ln(\mathfrak{L}) \sim (k+\ell)L \ln(dRW)$, with k and ℓ as in Assumption 4.1 (Lanthaler *et al.* 2022, De Ryck and Mishra 2022*a*). Taking this into account, we also find that the imposed lower bound on n is not very restrictive. Moreover, the right-hand side of (6.5) depends at most polynomially on L, W, R, d, k, ℓ and c. For physics-informed architectures, however, upper bounds on c often depend exponentially on L (De Ryck and Mishra 2022*a*, De Ryck *et al.* 2024*a*).

Remark 6.3. As Theorem 6.2 is an *a posteriori* error estimate, one can use the network sizes of the trained networks for L, W and R. The sizes stemming from the approximation error estimates of the previous sections can be disregarded for this result. Moreover, instead of considering the expected values of $\mathcal{E}$ and $\mathcal{E}_T$ in (6.5), one can also prove that such an inequality holds with a certain probability (De Ryck and Mishra 2022*b*).

Next, we show how Theorems 6.1 and 6.2 can be used in practice as *a posteriori* error estimates.

Example 6.4 (Navier–Stokes). We first state the *a posteriori* estimate for the Navier–Stokes equation (Example 2.2). By using the midpoint rule (2.67) on the sets $D \times [0,T]$ with N_{int} quadrature points, D with N_t quadrature points and $\partial D \times [0,T]$ with N_s quadrature points, one can prove the following estimate (De Ryck *et al.* 2024*a*, Theorem 3.10).

Theorem 6.5. Let $T > 0$, $d \in \mathbb{N}$, let $(u,p) \in C^4(\mathbb{T}^d \times [0,T])$ be the classical solution of the Navier–Stokes equation (2.8) and let $(u_\theta, p_\theta) \in C^4(\mathbb{T}^d \times [0,T])$ be

a model. Then the following error bound holds:

$$\mathcal{E}(\theta)^2 = \int_\Omega \|u(x,t) - u_\theta(x,t)\|_2^2 \,\mathrm{d}x\,\mathrm{d}t \\ \leq O\big(\mathcal{E}_T(\theta,\mathcal{S}) + N_t^{-2/d} + N_{\text{int}}^{-1/(d+1)} + N_s^{-1/d}\big). \tag{6.6}$$

The exact constant implied in the O-notation depends on (u_θ, p_θ) and (u,p) and their derivatives (De Ryck *et al.* 2024*a*, Theorem 3.10). There are two interesting observations to be made from equation (6.6).

- When the training set is large enough, the total error scales with the *square root* of the training error, $\mathcal{E}(\theta) \sim \sqrt{\mathcal{E}_T(\theta,\mathcal{S})}$. This sublinear relation is also observed in practice; see Figure 6.1.
- The bound (6.6) reveals different convergence rates in terms of the training set sizes N_{int}, N_t and N_s. In particular, we need relatively many training points in the interior of the domain compared to its boundary. We will see that these ratios will be different for other PDEs.

De Ryck *et al.* (2024*a*) verified the above result by training a PINN to approximate the Taylor–Green vortex in two space dimensions, the exact solution of which is given by

$$u(t,x,y) = -\cos(\pi x)\sin(\pi y)\exp(-2\pi^2\nu t), \tag{6.7}$$

$$v(t,x,y) = \sin(\pi x)\cos(\pi y)\exp(-2\pi^2\nu t), \tag{6.8}$$

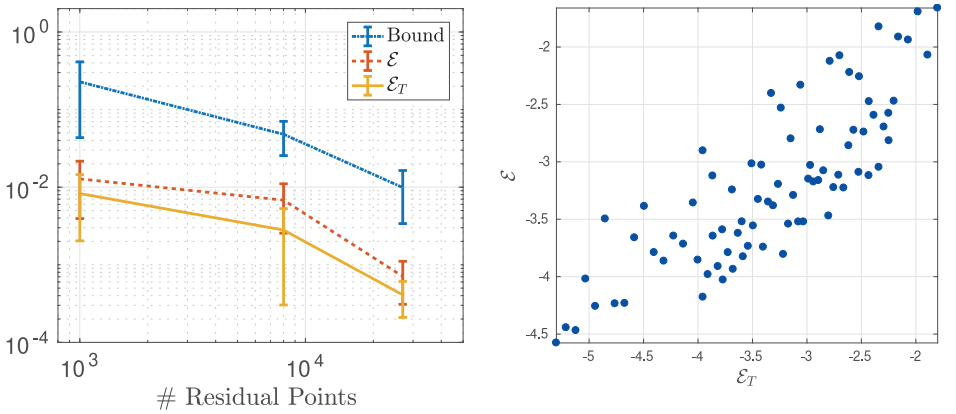
$$p(t,x,y) = -\frac{\rho}{4}[\cos(2\pi x) + \cos(2\pi y)]\exp(-4\pi^2\nu t). \tag{6.9}$$

The spatio-temporal domain is $x, y \in [0.5, 4.5]^2$ and $t \in [0,1]$. The results are reported in Figure 6.1. As one can see, all error types decrease with increasing number of quadrature points. In particular, the relationship $\mathcal{E}(\theta) \sim \sqrt{\mathcal{E}_T(\theta,\mathcal{S})}$ is (approximately) observed as well.

In the next example we show that Theorem 6.2 can be used to show that the curse of dimensionality can also be overcome in the training set size.

Example 6.6 (high-dimensional heat equation). We consider the high-dimensional (linear) heat equation (Example 2.1), which is an example of a linear Kolmogorov equation, for which we have proved that the CoD can be overcome in the approximation error in Example 4.23. In this example, we show that the training set size only needs to grow at most polynomially in the input dimension d to achieve a certain accuracy. Applying Theorem 6.2 to every term of the right-hand side of the stability result for the heat equation (Theorem 5.3) then gives us the following result.

Theorem 6.7. Under the assumptions of Theorem 6.2 and given a training set of size N_{int} on $D \times [0,T]$, size N_t on D and size N_s on $\partial D \times [0,T]$, we find that there



(a) (b)

Figure 6.1. Experimental results for the Navier–Stokes equation with $\nu = 0.01$. The total error $\mathcal{E}$ and the training error $\mathcal{E}_T$ are shown in terms of the number of residual points N_{int} ((a), and compared with the bound from Theorem 6.5) and also in terms of each other (b). Reproduced from De Ryck *et al.* (2024*a*), with permission of the IMA.

exist constants $C, \alpha > 0$ that are independent of d such that

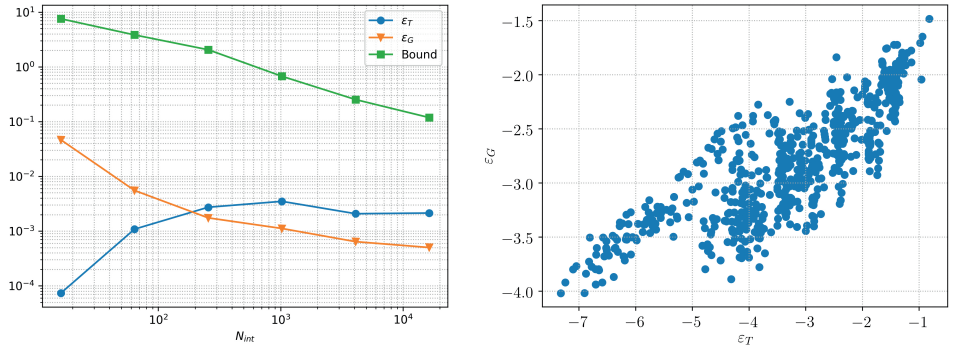
$$\mathbb{E}[\mathcal{E}(\theta^*(\mathcal{S}))^2] \leq C d^\alpha \left(\mathbb{E}[\mathcal{E}_T(\theta^*(\mathcal{S}), \mathcal{S})] + \frac{\ln(N_{\text{int}})}{N_{\text{int}}^2} + \frac{\ln(N_t)}{N_t^2} + \frac{\ln(N_s)}{N_s^4} \right). \quad (6.10)$$

A more general result for linear Kolmogorov equations, including a more detailed expression for C and α, can be found in De Ryck and Mishra (2022*a*). A comparison with the corresponding result for the Navier–Stokes equations (Theorem 6.5) reveals the following.

- Just as for the Navier–Stokes equations, the total error scales with the *square root* of the training error, $\mathcal{E}(\theta) \sim \sqrt{\mathcal{E}_T(\theta, \mathcal{S})}$ when the training set is large enough.
- For the Navier–Stokes equations (with $d = 2$ or $d = 3$) we found that we need to choose $N_{\text{int}} \gg N_s \gg N_t$. However, in this case we find that far fewer training points in the interior are needed, as $N_s \gg N_t \approx N_{\text{int}}$ is sufficient.

As a first example, Mishra and Molinaro (2022) consider the one-dimensional heat equation on the domain $[-1, 1]$ with initial condition $\sin(\pi x)$ and final time $T = 1$. The results can be seen in Figure 6.2. As one can see, both the generalization and training error decrease with increasing number of quadrature points. In particular, the relationship $\mathcal{E}(\theta) \sim \sqrt{\mathcal{E}_T(\theta, \mathcal{S})}$ is (approximately) observed as well.

As a second example, we consider the one-dimensional heat equation with parametric initial condition, and where the parameter space is very high-dimensional.



(a) (b)

Figure 6.2. Experimental results for the heat equation. The generalization error $\mathcal{E}_G$ and the training error $\mathcal{E}_T$ are shown in terms of the number of residual points N_{int} ((a), and compared with the bound from Theorem 6.7) and also in terms of each other (b). Figure from Molinaro (2023).

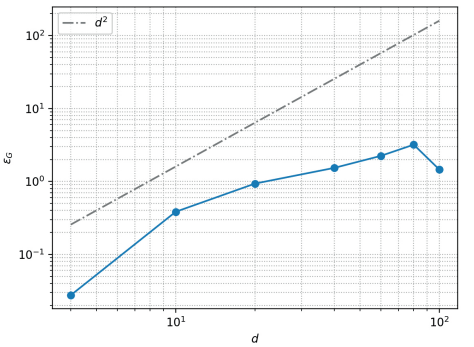


Figure 6.3. Relative generalization percentage error $\mathcal{E}_G$ in terms of the parameter dimension d for the heat equation with a parametrized initial condition. Figure from Molinaro (2023).

In more detail, the initial condition considered is parametrized by μ and given by

$$u_0(x;\mu) = \sum_{i=1}^{d} \frac{-1}{dm^2} \sin(m\pi(x-\mu_i)). \tag{6.11}$$

Figure 6.3 demonstrates that the generalization error $\mathcal{E}_G$ does not grow exponentially (but rather sub-quadratically) in the parameter dimension d, and that therefore the curse of dimensionality is mitigated. Even for very large d the relative generalization error is less than 2%.

Example 6.8 (scalar conservation laws). We again consider weak PINNs for the scalar conservation laws. Recall from the stability result (Theorem 5.8) that the generalization error is given by

$$\begin{aligned}\mathcal{E}(\theta,\varphi,c)&\\ &= -\int_0^1\int_0^T (|u_{\theta^*(\mathcal{S})}(x,t)-c|\partial_t\varphi(x,t)+Q[u_{\theta^*(\mathcal{S})}(x,t);c]\partial_x\varphi(x,t))\,\mathrm{d}x\,\mathrm{d}t\\ &\quad+\int_0^1 |u_\theta(x,0)-u(x,0)|\,\mathrm{d}x+\int_0^T |u_\theta(0,t)-u_\theta(1,t)|\,\mathrm{d}t. \end{aligned}\tag{6.12}$$

To this end, we consider the simplest case of random (Monte Carlo) quadrature and generate a set of collocation points,

$$\mathcal{S}=\{(x_i,t_i)\}_{i=1}^N\subset D\times[0,T],$$

where all (x_i,t_i) are i.i.d. drawn from the uniform distribution on $D\times[0,T]$. For a fixed $\theta\in\Theta$, $\varphi\in\Phi_\epsilon$, $c\in\mathcal{C}$ and for this data set $\mathcal{S}$, we can then define the *training error*

$$\begin{aligned}\mathcal{E}_T(\theta,\mathcal{S},\varphi,c)&\\ &= -\frac{T}{N}\sum_{i=1}^N(|u_\theta(x_i,t_i)-c|\partial_t\varphi(x_i,t_i)+Q[u_\theta(x_i,t_i);c]\partial_x\varphi(x_i,t_i))\\ &\quad+\frac{T}{N}\sum_{i=1}^N|u_\theta(x_i,0)-u(x_i,0)|+\frac{T}{N}\sum_{i=1}^N|u_\theta(0,t_i)-u_\theta(1,t_i)|. \end{aligned}\tag{6.13}$$

During training, we then aim to obtain neural network parameters $\theta^*_{\mathcal{S}}$, a test function $\varphi^*_{\mathcal{S}}$ and a scalar $c^*_{\mathcal{S}}$ such that

$$\mathcal{E}_T\left(\theta^*_{\mathcal{S}},\mathcal{S},\varphi^*_{\mathcal{S}},c^*_{\mathcal{S}}\right)\approx\min_{\theta\in\Theta}\max_{\varphi\in\Phi_\epsilon}\max_{c\in\mathcal{C}}\mathcal{E}_T(\theta,\mathcal{S},\varphi,c)\tag{6.14}$$

for some $\epsilon>0$. We call the resulting neural network $u^*:=u_{\theta^*_{\mathcal{S}}}$ a *weak PINN* (wPINN). If the network has width W, depth L and its weights are bounded by R, and the parameter $\epsilon>0$ is as in Theorem 5.10, then we obtain the following theorem (De Ryck *et al.* 2024*c*, Theorem 3.9 and Corollary 3.10).

Theorem 6.9. Let $\mathcal{C}$ be the essential range of u and let $N,L,W\in\mathbb{N}$, $R\geq\max\{1,T,|\mathcal{C}|\}$ with $L\geq 2$ and $N\geq 3$. Moreover, let $u_\theta\colon D\times[0,T]\to\mathbb{R}$, $\theta\in\Theta$, be tanh neural networks with at most $L-1$ hidden layers, width at most W and weights and biases bounded by R. Assume that $\mathcal{E}_G$ and $\mathcal{E}_T$ are bounded by $B\geq 1$. It holds with a probability of at least $1-\delta$ that

$$\mathcal{E}_G\left(\theta^*_{\mathcal{S}},\varphi^*_{\mathcal{S}},c^*_{\mathcal{S}}\right)\leq\mathcal{E}_T\left(\theta^*_{\mathcal{S}},\mathcal{S},\varphi^*_{\mathcal{S}},c^*_{\mathcal{S}}\right)+\frac{3BLW}{\sqrt{N}}\sqrt{\ln\left(\frac{C\ln(1/\epsilon)WRN}{\epsilon^3\delta B}\right)}.\tag{6.15}$$

7. Training

Despite the generally affirmative answers to our first three questions (Q1–Q3) in the previous sections, significant problems have been identified with physics-informed machine learning. Arguably, the foremost problem lies in *training* these frameworks with (variants of) gradient descent methods. It has been increasingly observed that PINNs and their variants are *slow, even infeasible*, to train even on certain model problems, with the training process either not converging, or converging to unacceptably large loss values (Krishnapriyan *et al.* 2021, Moseley, Markham and Nissen-Meyer 2023, Wang, Teng and Perdikaris 2021*a*, Wang, Yu and Perdikaris 2022*b*). These observations highlight the relevance of our fourth question.

Question 4 (Q4). Can the training error $\mathcal{E}_T^*$ be made sufficiently close to the true minimum of the loss function $\min_\theta \mathcal{J}(\theta, \mathcal{S})$?

To answer this question, we must find out the reason behind the issues observed with training physics-informed machine learning algorithms. Empirical studies such as that of Krishnapriyan *et al.* (2021) attribute failure modes to the non-convex loss landscape, which is much more complex when compared to the loss landscape of supervised learning. Others such as Moseley *et al.* (2023) and Dolean, Heinlein, Mishra and Moseley (2023) have implicated the well-known spectral bias (Rahaman *et al.* 2019) of neural networks as being a cause of poor training, whereas Wang *et al.* (2021*a*) and Wang, Wang and Perdikaris (2021*b*) used infinite-width NTK theory to propose that the subtle balance between the PDE residual and supervised components of the loss function could explain and possibly ameliorate training issues. Nevertheless, it is fair to say that there is still a paucity of principled analysis of the training process for gradient descent algorithms in the context of physics-informed machine learning.

In what follows, we first demonstrate that when an affirmative answer to Question 4 is available (or assumed), it becomes possible to provide *a priori error estimates* on the error of models learned by minimizing a physics-informed loss function (Section 7.1). Next, we derive precise conditions under which gradient descent for a physics-informed loss function can be approximated by a *simplified gradient descent* algorithm, which amounts to the gradient descent update for a *linearized* form of the training dynamics (Section 7.2). It will then turn out that the speed of convergence of the gradient descent is related to the *condition number* of an operator, which in turn is composed of the *Hermitian square* ($\mathcal{L}^*\mathcal{L}$) of the differential operator ($\mathcal{L}$) of the underlying PDE and a *kernel integral operator*, associated to the tangent kernel for the underlying model (Section 7.3). This analysis automatically suggests that *preconditioning* the resulting operator is necessary to alleviate training issues for physics-informed machine learning (Section 7.4). Finally, in Section 7.5, we discuss how different preconditioning strategies can overcome training bottlenecks, and also how existing techniques, proposed in the

literature for improving training, can be viewed from this operator preconditioning perspective.

7.1. Global minimum of the loss is a good approximation to the PDE solution

We demonstrate that the results of the previous sections can be used to prove *a priori error estimates* for physics-informed learning, provided that one can find a global minimum of the physics-informed loss function (2.69).

We revisit the error decomposition (3.7) proposed in Section 3. There it was shown that if one can answer Question 2 in the affirmative, then, for any $\theta^*, \widehat{\theta} \in \Theta$,

$$\begin{aligned}\mathcal{E}(\theta^*) &\leq C\mathcal{E}_G(\theta^*)^\alpha \\ &\leq C\left(\mathcal{E}_G(\widehat{\theta}) + 2\sup_{\theta\in\Theta}|\mathcal{E}_T(\theta,\mathcal{S}) - \mathcal{E}_G(\theta)| + \mathcal{E}_T(\theta^*,\mathcal{S}) - \mathcal{E}_T(\widehat{\theta},\mathcal{S})\right)^\alpha. \end{aligned} \tag{7.1}$$

For any $\epsilon > 0$, we can now let $\widehat{\theta}$ be the parameter corresponding to the model from Question 1, and hence we find that $\mathcal{E}_G(\widehat{\theta}) < \epsilon$ if the model class is expressive enough. Next we can use the results of Section 6 to deduce a lower limit on the size of the training set $\mathcal{S}$ (in terms of ϵ) such that also $\sup_{\theta\in\Theta}|\mathcal{E}_T(\theta,\mathcal{S}) - \mathcal{E}_G(\theta)| < \epsilon$. Finally, if we assume that $\theta^* = \theta^*(\mathcal{S})$ is the global minimizer of the loss function $\theta \mapsto \mathcal{J}(\theta,\mathcal{S}) = \mathcal{E}_T(\theta,\mathcal{S})$ (2.69), then it must hold that $\mathcal{E}_T(\theta^*,\mathcal{S}) \leq \mathcal{E}_T(\widehat{\theta},\mathcal{S})$ and hence $\mathcal{E}_T(\theta^*,\mathcal{S}) - \mathcal{E}_T(\widehat{\theta},\mathcal{S}) \leq 0$. As a result, we can infer from (7.1) that

$$\mathcal{E}^* \leq C(3\epsilon)^\alpha. \tag{7.2}$$

Alternatively, we note that many of the *a posteriori* error bounds in Section 6 are of the form

$$\mathcal{E}(\theta^*) \leq C(\mathcal{E}_T(\theta^*,\mathcal{S}) + \epsilon)^\alpha \tag{7.3}$$

if the training set is large enough (depending on ϵ). As before,

$$\mathcal{E}_T(\theta^*,\mathcal{S}) \leq \mathcal{E}_T(\widehat{\theta},\mathcal{S}) \leq \mathcal{E}_G(\widehat{\theta}) + \sup_{\theta\in\Theta}|\mathcal{E}_T(\theta,\mathcal{S}) - \mathcal{E}_G(\theta)| \leq 2\epsilon, \tag{7.4}$$

such that in total we again find that $\mathcal{E}^* \leq C(3\epsilon)^\alpha$, in agreement with (7.2).

We first demonstrate this argument for the Navier–Stokes equations, and then show a similar result for the Poisson equation in which the curse of dimensionality is overcome.

Example 7.1 (Navier–Stokes equations). In Sections 4, 5 and 6 we found affirmative answers to questions Q1–Q3, and hence we should be able to apply the above argument under the assumptions that an exact global minimizer to the loss function can be found.

An *a posteriori* estimate as in (7.3) is provided by Theorem 6.5, but it is initially unclear whether this can also be used for an *a priori* estimate, given the dependence of the bound on the derivatives of the model. It turns out that this is possible for PINNs if the true solution is sufficiently smooth and the neural network and training set sufficiently large (De Ryck *et al.* 2024*a*, Corollary 3.9).

Corollary 7.2. Let $\epsilon > 0$, $T > 0$, $d \in \mathbb{N}$, $k > 6(3d + 8) =: \gamma$, let $(u, p) \in H^k(\mathbb{T}^d \times [0, T])$ be the classical solution of the Navier–Stokes equation (2.8), let the hypothesis space consist of PINNs for which the upper bound on the weights is at least $R \geq \epsilon^{-1/(k-\gamma)} \ln(1/\epsilon)$, the width is at least $W \geq \epsilon^{-(d+1)/(k-\gamma)}$ and the depth is at least $L \geq 3$, let $(u_{\theta^*(\mathcal{S})}, p_{\theta^*(\mathcal{S})})$ be the PINN that solves (2.69) and let the training set $\mathcal{S}$ satisfy $N_t \geq \epsilon^{-d(1+\gamma/(k-\gamma))}$, $N_{\text{int}} \geq \epsilon^{-2(d+1)(1+\gamma/(k-\gamma))}$ and $N_s \geq \epsilon^{-2d(1+\gamma/(k-\gamma))}$. It holds that

$$\|u - u_{\theta^*(\mathcal{S})}\|_{L^2(D\times[0,T])} = O(\epsilon). \tag{7.5}$$

Proof. The results follow from combining results on the approximation error (Theorem 4.9), the stability (Theorem 5.7) and the generalization error (Theorem 6.5). □

Example 7.3 (Poisson's equation). We revisit Poisson's equation with the variation residual, as previously considered in Example 4.30 (Question 1) and Example 5.11 (Question 2). Recall that if $f \in \mathcal{B}^0(\Omega)$ (see Definition 4.28) satisfies $\int_\Omega f(x)\,\mathrm{d}x = 0$, then the corresponding unique solution u to Poisson's equation $-\Delta u = f$ with zero Neumann boundary conditions is $u \in \mathcal{B}^2(\Omega)$.

Now, assume that the training set $\mathcal{S}$ consists of N i.i.d. randomly generated points in Ω, and we optimize over a subset of shallow softplus neural networks of width m, corresponding to parameter space $\Theta^* \subset \Theta$; a more precise definition can be found in Lu *et al.* (2021*c*, eq. 2.13). As usual, we then define $u_{\theta^*(\mathcal{S})}$ as the minimizer of the discretized energy residual $\mathcal{J}(\theta, \mathcal{S})$. In this setting, one can prove the following *a priori* generalization result (Lu *et al.* 2021*c*, Remark 2.1).

Theorem 7.4. Assume the setting defined above. If we set $m = N^{1/3}$, then there exists a constant $C > 0$ (independent of m and N) such that

$$\mathbb{E}\left[\|u - u_{\theta^*(\mathcal{S})}\|^2_{H^1(\Omega)}\right] \leq C\frac{(\log(N))^2}{N^{1/3}}. \tag{7.6}$$

In the above theorem, we can see that the convergence rate is independent of d, such that the curse of dimensionality is mitigated. The constant C might depend on the Barron norm of u, and its dependence on d might therefore not be exactly clear.

Example 7.5 (scalar conservation law). Using Theorem 5.10 and the above bound on the generalization gap, we can prove the following rigorous upper bound (De Ryck *et al.* 2024*c*, Corollary 3.10) on the total L^1-error of the weak PINN, which we denote as

$$\mathcal{E}(\theta) = \int_D |u_\theta(x, T) - u(x, T)|\,\mathrm{d}x. \tag{7.7}$$

Corollary 7.6. Assume the setting of Theorem 6.9. It holds with a probability of at least $1-\delta$ that

$$\mathcal{E}^* \leq C\left[\mathcal{E}_T^* + \frac{3BdLW}{\sqrt{N}}\sqrt{\ln\left(\frac{C\ln(1/\epsilon)WRN}{\epsilon^3\delta B}\right)} + (1+\|u^*\|_{C^1})\ln(1/\epsilon)^3\epsilon\right], \quad (7.8)$$

where $\mathcal{E}^* := \mathcal{E}(\theta_{\mathcal{S}}^*)$ and $\mathcal{E}_T^* := \mathcal{E}_T\left(\theta_{\mathcal{S}}^*, \mathcal{S}, \varphi_{\mathcal{S}}^*, c_{\mathcal{S}}^*\right)$.

Unfortunately, this result in its current form is not strong enough to lead to an *a priori* generalization error. Using (7.4), one can easily ensure that $\mathcal{E}_T^*$ is small, and verify that B, R, N and L depend (at most) polynomially on ϵ^{-1} such that the second term in (7.8) can be made small for large N. The problem lies in the third term, as most available upper bounds on $\|u^*\|_{C^1}$ will be overestimates that grow with ϵ^{-1}.

7.2. *Characterization of gradient descent*

As mentioned at the beginning of Section 7, for some types of PDEs significant problems during the training of the model have been observed. To investigate this issue, we first consider a linearized version of the gradient descent (GD) update formula.

First, recall that physics-informed machine learning boils down to minimizing the physics-informed loss, as constructed in Section 2.4, that is, to find

$$\theta^*(\mathcal{S}) = \operatorname*{argmin}_{\theta\in\Theta} L(\theta), \quad (7.9)$$

where in this section we focus on loss functions of the form

$$L(\theta) = \underbrace{\frac{1}{2}\int_\Omega |\mathcal{L}u(x) - f(x)|^2\,\mathrm{d}x}_{R(\theta)} + \underbrace{\frac{\lambda}{2}\int_{\partial\Omega} |u(x)-g(x)|^2\,\mathrm{d}\sigma(x)}_{B(\theta)}. \quad (7.10)$$

As mentioned in Section 2.5, it is customary in machine learning (Goodfellow *et al.* 2016) that the non-convex optimization problem (7.9) is solved with (variants of) a gradient descent algorithm which takes the generic form

$$\theta_{k+1} = \theta_k - \eta\nabla_\theta L(\theta_k), \quad (7.11)$$

with descent steps $k > 0$, learning rate $\eta > 0$, loss L and the initialization θ_0 chosen randomly, following Section 2.5.1.

We want to investigate the *rate of convergence* to ascertain the computational cost of training. As the loss L (7.11) is non-convex, it is hard to rigorously analyse the training process in complete generality. We need to make certain assumptions on (7.11) to make the problem tractable. To this end, we fix the step k in (7.11) and start with the following Taylor expansion:

$$u(x;\theta_k) = u(x;\theta_0) + \nabla_\theta u(x;\theta_0)^\top(\theta_k-\theta_0) + \frac{1}{2}(\theta_k-\theta_0)^\top H_k(x)(\theta_k-\theta_0). \quad (7.12)$$

Here, $H_k(x) \coloneqq \mathrm{Hess}_\theta(u(x;\tau_k\theta_0 + (1-\tau_k)\theta_k)$ is the Hessian of $u(\cdot\,,\theta)$ evaluated at intermediate values, with $0 \le \tau_k \le 1$. Now introducing the notation $\phi_i(x) = \partial_{\theta_i} u(x;\theta_0)$, we define the matrix $\mathbb{A} \in \mathbb{R}^{n\times n}$ and the vector $\mathbb{C}, \in \mathbb{R}^n$ as

$$\begin{aligned} \mathbb{A}_{i,j} &= \langle \mathcal{L}\phi_i, \mathcal{L}\phi_j\rangle_{L^2(\Omega)} + \lambda\langle \phi_i, \phi_j\rangle_{L^2(\partial\Omega)}, \\ \mathbb{C}_i &= \langle \mathcal{L}u_{\theta_0} - f, \mathcal{L}\phi_i\rangle_{L^2(\Omega)} + \lambda\langle u_{\theta_0} - u, \phi_i\rangle_{L^2(\partial\Omega)}. \end{aligned} \tag{7.13}$$

Substituting the above formulas in the GD algorithm (7.11), we can rewrite it identically as

$$\theta_{k+1} = \theta_k - \eta\nabla_\theta L(\theta_k) = (I - \eta\mathbb{A})\theta_k + \eta(\mathbb{A}\theta_0 + \mathbb{C}) + \eta\epsilon_k, \tag{7.14}$$

where ϵ_k is an error term that collects all terms that depend on the Hessians H_k and $\mathcal{L}H_k$, and is explicitly given by

$$\begin{aligned} 2\epsilon_k &= \langle \mathcal{L}u_{\theta_k} - f, \mathcal{L}H_k(\theta_k - \theta_0)\rangle_{L^2(\Omega)} + \lambda\langle u_{\theta_k} - u, H_k(\theta_k - \theta_0)\rangle_{L^2(\partial\Omega)} \\ &\quad + \langle (\theta_k - \theta_0)^\top \mathcal{L}H_k(\theta_k - \theta_0), \mathcal{L}\nabla_\theta u_{\theta_0}\rangle_{L^2(\Omega)} \\ &\quad + \lambda\langle (\theta_k - \theta_0)^\top H_k(\theta_k - \theta_0), \nabla_\theta u_{\theta_0}\rangle_{L^2(\partial\Omega)}. \end{aligned} \tag{7.15}$$

From this characterization of gradient descent, we clearly see that (7.11) is related to a *simplified* version of gradient descent given by

$$\widetilde{\theta}_{k+1} = (I - \eta\mathbb{A})\widetilde{\theta}_k + \eta(\mathbb{A}\widetilde{\theta}_0 + \mathbb{C}), \quad \widetilde{\theta}_0 = \theta_0, \tag{7.16}$$

modulo the *error* term ϵ_k defined in (7.15).

The following lemma (De Ryck, Bonnet, Mishra and de Bézenac 2023) shows that this simplified GD dynamics (7.16) approximates the full GD dynamics (7.11) to desired accuracy as long as the error term ϵ_k is small.

Lemma 7.7. Let $\delta > 0$ be such that $\max_k \|\epsilon_k\|_2 \le \delta$. If $\mathbb{A}$ is invertible and $\eta = c/\lambda_{\max}(\mathbb{A})$ for some $0 < c < 1$, then it holds for any $k \in \mathbb{N}$ that

$$\|\theta_k - \widetilde{\theta}_k\|_2 \le \kappa(\mathbb{A})\delta/c, \tag{7.17}$$

with $\lambda_{\min} \coloneqq \min_j |\lambda_j(\mathbb{A})|$, $\lambda_{\max} \coloneqq \max_j |\lambda_j(\mathbb{A})|$ and the *condition number*

$$\kappa(\mathbb{A}) = \lambda_{\max}(\mathbb{A})/\lambda_{\min}(\mathbb{A}). \tag{7.18}$$

The key assumption in Lemma 7.7 is the smallness of the error term ϵ_k (7.14) for all k. This is trivially satisfied for linear models $u_\theta(x) = \sum_k \theta_k\phi_k$ as $\epsilon_k = 0$ for all k in this case. From the definition of ϵ_k (7.15), we see that a more general sufficient condition for ensuring this smallness is to ensure that the Hessians of u_θ and $\mathcal{L}u_\theta$ (resp. H_k and $\mathcal{L}H_k$ in (7.12)) are small during training. This amounts to requiring *approximate linearity* of the parametric function $u(\cdot\,;\theta)$ near the initial value θ_0 of the parameter θ. For any differentiable parametrized function f_θ, its linearity is equivalent to the constancy of the associated *tangent kernel* (Liu, Zhu and Belkin 2020)

$$\Theta[f_\theta](x,y) \coloneqq \nabla_\theta f_\theta(x)^\top \nabla_\theta f_\theta(y). \tag{7.19}$$

Hence, it follows that if the tangent kernel associated to u_θ and $\mathcal{L}u_\theta$ is (approximately) constant along the optimization path, then the error term ϵ_k will be small.

For neural networks this entails that the *neural tangent kernels* (NTK) $\Theta[u_\theta]$ and $\Theta[\mathcal{L}u_\theta]$ stay approximately constant along the optimization path. The following informal lemma from De Ryck *et al.* (2023), based on Wang *et al.* (2022*b*), confirms that this is indeed the case for wide enough neural networks.

Lemma 7.8. For a neural network u_θ with one hidden layer of width m and a linear differential operator $\mathcal{L}$, we have $\lim_{m\to\infty} \Theta[u_{\theta_k}] = \lim_{m\to\infty} \Theta[u_{\theta_0}]$ and $\lim_{m\to\infty} \Theta[\mathcal{L}u_{\theta_k}] = \lim_{m\to\infty} \Theta[\mathcal{L}u_{\theta_0}]$ for all k. Consequently, the error term ϵ_k (7.14) is small for wide neural networks, $\lim_{m\to\infty} \max_k \|\epsilon_k\|_2 = 0$.

Now, given the much simpler structure of (7.16), when compared to (7.11), one can study the corresponding gradient descent dynamics explicitly and obtain the following convergence theorem (De Ryck *et al.* 2023).

Theorem 7.9. Let $\mathbb{A}$ in (7.16) be invertible with condition number $\kappa(\mathbb{A})$ (7.18) and let $0 < c < 1$. Set $\eta = c/\lambda_{\max}(\mathbb{A})$ and $\vartheta = \theta_0 + \mathbb{A}^{-1}\mathbb{C}$. It holds for any $k \in \mathbb{N}$ that

$$\|\widetilde{\theta}_k - \vartheta\|_2 \leq (1 - c/\kappa(\mathbb{A}))^k \|\theta_0 - \vartheta\|_2. \tag{7.20}$$

An immediate consequence of the quantitative convergence rate (7.20) is as follows: to obtain an error of size ε, i.e. $\|\widetilde{\theta}_k - \vartheta\|_2 \leq \varepsilon$, we can readily calculate the number of GD steps $N(\varepsilon)$ as

$$N(\varepsilon) = \ln(\varepsilon/\|\theta_0 - \vartheta\|_2)/\ln(1 - c/\kappa(\mathbb{A})) = O\left(\kappa(\mathbb{A}) \ln \frac{1}{\epsilon}\right). \tag{7.21}$$

Hence, for a fixed value c, large values of the condition number $\kappa(\mathbb{A})$ will severely impede convergence of the simplified gradient descent (7.16) by requiring a much larger number of steps.

7.3. Training and operator preconditioning

So far, we have established that, under suitable assumptions, the rate of convergence of the gradient descent algorithm for physics-informed machine learning boils down to the *conditioning* of the matrix $\mathbb{A}$ (7.13). However, at first sight, this matrix is not very intuitive and we want to relate it to the differential operator $\mathcal{L}$ from the underlying PDE (2.1). To this end, we first introduce the so-called *Hermitian square* $\mathcal{A}$, given by

$$\mathcal{A} = \mathcal{L}^*\mathcal{L}, \tag{7.22}$$

in the sense of operators, where $\mathcal{L}^*$ is the *adjoint operator* for the differential operator $\mathcal{L}$. Note that this definition implicitly assumes that the adjoint $\mathcal{L}^*$ exists and the Hermitian square operator $\mathcal{A}$ is defined on an appropriate function space. As an example, consider as differential operator the Laplacian, i.e. $\mathcal{L}u = -\Delta u$,

defined for instance on $u \in H^1(\Omega)$; then the corresponding Hermitian square is $\mathcal{A}u = \Delta^2 u$, identified as the *bi-Laplacian* that is well-defined on $u \in H^2(\Omega)$.

Next, for notational simplicity, we consider a loss function that only consists of the integral of the squared PDE residual (2.41) and omit boundary terms in the following. Let $\mathcal{H}$ be the span of the functions $\phi_k := \partial_{\theta_k} u(\cdot; \theta_0)$. Define the maps $T\colon \mathbb{R}^n \to \mathcal{H}, v \to \sum_{k=1}^n v_k \phi_k$ and $T^*\colon L^2(\Omega) \to \mathbb{R}^n; f \to \{\langle \phi_k, f\rangle\}_{k=1,\dots,n}$. We define the following scalar product on $L^2(\Omega)$:

$$\langle f, g\rangle_{\mathcal{H}} := \langle f, TT^* g\rangle_{L^2(\Omega)} = \langle T^* f, T^* g\rangle_{\mathbb{R}^n}. \tag{7.23}$$

Note that the maps T, T^* provide a correspondence between the continuous space (L^2) and discrete space ($\mathcal{H}$) spanned by the functions ϕ_k. This continuous–discrete correspondence allows us to relate the conditioning of the matrix $\mathbb{A}$ in (7.13) to the conditioning of the Hermitian square operator $\mathcal{A} = \mathcal{L}^*\mathcal{L}$ via the following theorem (De Ryck *et al.* 2023).

Theorem 7.10. It holds for the operator $\mathcal{A} \circ TT^*\colon L^2(\Omega) \to L^2(\Omega)$ that $\kappa(\mathbb{A}) \geq \kappa(\mathcal{A} \circ TT^*)$. Moreover, if the Gram matrix $\langle \phi, \phi\rangle_{\mathcal{H}}$ is invertible then equality holds, i.e. $\kappa(\mathbb{A}) = \kappa(\mathcal{A} \circ TT^*)$.

Thus we show that the conditioning of the matrix $\mathbb{A}$ that determines the speed of convergence of the simplified gradient descent algorithm (7.16) for physics-informed machine learning is intimately tied to the conditioning of the operator $\mathcal{A} \circ TT^*$. This operator, in turn, composes the Hermitian square of the underlying differential operator of the PDE (2.1), with the so-called *kernel integral operator* TT^*, associated with the (neural) tangent kernel $\Theta[u_\theta]$. Theorem 7.10 implies in particular that if the operator $\mathcal{A} \circ TT^*$ is ill-conditioned, then the matrix $\mathbb{A}$ is ill-conditioned and the gradient descent algorithm (7.16) for physics-informed machine learning will converge very slowly.

Remark 7.11. One can readily generalize Theorem 7.10 to the setting with boundary conditions, i.e. with $\lambda > 0$ in the loss (7.10). In this case one can prove for the operator $\mathcal{A} = \mathbb{1}_{\mathring{\Omega}} \cdot \mathcal{L}^*\mathcal{L} + \lambda \mathbb{1}_{\partial\Omega} \cdot \mathrm{Id}$, and its corresponding matrix $\mathbb{A}$ (as in (7.13)) that $\kappa(\mathbb{A}) \geq \kappa(\mathcal{A} \circ TT^*)$, where equality holds if the relevant Gram matrix is invertible. More details can be found in De Ryck *et al.* (2023, Appendix A.6).

It is instructive to compare physics-informed machine learning with standard supervised learning through the prism of the analysis presented here. It is straightforward to see that for supervised learning, i.e. when the physics-informed loss is replaced by the supervised loss $\frac{1}{2}\|u - u_\theta\|^2_{L^2(\Omega}$ by simply setting $\mathcal{L} = \mathrm{Id}$, the corresponding operator in Theorem 7.10 is simply the kernel integral operator TT^*, associated with the tangent kernel as $\mathcal{A} = \mathrm{Id}$. Thus the complexity in training physics-informed machine learning models is entirely due to the spectral properties of the Hermitian square $\mathcal{A}$ of the underlying differential operator $\mathcal{L}$.

7.4. Preconditioning to improve training in physics-informed machine learning

Having established in the previous section that, under suitable assumptions, the speed of training physics-informed machine learning models depends on the condition number of the operator $\mathcal{A} \circ TT^*$, or equivalently the matrix $\mathbb{A}$ (7.13), we now investigate whether this operator is ill-conditioned and, if so, how we can better condition it by reducing the condition number. The fact that $\mathcal{A} \circ TT^*$ (equivalently $\mathbb{A}$) is very poorly conditioned for most PDEs of practical interest will be demonstrated both theoretically and empirically below. This makes *preconditioning*, i.e. strategies to improve (reduce) the conditioning of the underlying operator (matrix), a key component in improving training for physics-informed machine learning models.

7.4.1. General framework for preconditioning

Intuitively, reducing the condition number of the underlying operator $\mathcal{A} \circ TT^*$ amounts to finding new maps $\widetilde{T}, \widetilde{T}^*$ for which the kernel integral operator $\widetilde{T}\widetilde{T}^*$ is approximately equal to $\mathcal{A}^{-1}$, i.e. choosing the architecture and initialization of the parametrized model u_θ such that the associated kernel integral operator $\widetilde{T}\widetilde{T}^*$ is an (approximate) Green's function for the Hermitian square $\mathcal{A}$ of the differential operator $\mathcal{L}$. For an operator $\mathcal{A}$ with well-defined eigenvectors ψ_k and eigenvalues ω_k, the ideal case $\widetilde{T}\widetilde{T}^* = \mathcal{A}^{-1}$ is realized when

$$\widetilde{T}\widetilde{T}^* \phi_k = \frac{1}{\omega_k} \psi_k.$$

This can be achieved by transforming ϕ linearly with a (positive definite) matrix $\mathbb{P}$ such that

$$(\mathbb{P}^\top \phi)_k = \frac{1}{\omega_k} \psi_k,$$

which corresponds to the change of variables $\mathcal{P} u_\theta := u_{\mathbb{P}\theta}$. Assuming the invertibility of $\langle \phi, \phi \rangle_{\mathcal{H}}$, Theorem 7.10 then shows that $\kappa(\mathcal{A} \circ \widetilde{T}\widetilde{T}^*) = \kappa(\widetilde{\mathbb{A}})$ for a new matrix $\widetilde{\mathbb{A}}$, which can be computed as

$$\widetilde{\mathbb{A}} := \langle \mathcal{L}\nabla_\theta u_{\mathbb{P}\theta_0}, \mathcal{L}\nabla_\theta u_{\mathbb{P}\theta_0} \rangle_{L^2(\Omega)} = \langle \mathcal{L}\mathbb{P}^\top \nabla_\theta u_{\theta_0}, \mathcal{L}\mathbb{P}^\top \nabla_\theta u_{\theta_0} \rangle_{L^2(\Omega)} = \mathbb{P}^\top \mathbb{A} \mathbb{P}. \tag{7.24}$$

This implies a general approach for preconditioning, namely linearly transforming the parameters of the model, i.e. considering $\mathcal{P} u_\theta := u_{\mathbb{P}\theta}$ instead of u_θ, which corresponds to replacing the matrix $\mathbb{A}$ with its preconditioned variant $\widetilde{\mathbb{A}} = \mathbb{P}^\top \mathbb{A} \mathbb{P}$. The new simplified GD update rule is then

$$\theta_{k+1} = \theta_k - \eta \widetilde{\mathbb{A}}(\theta_k - \theta_0) + \mathbb{C}. \tag{7.25}$$

Hence, finding $\widetilde{T}\widetilde{T}^* \approx \mathcal{A}^{-1}$, which is the aim of preconditioning, reduces to constructing a matrix $\mathbb{P}$ such that $1 \approx \kappa(\widetilde{\mathbb{A}}) \ll \kappa(\mathbb{A})$. We emphasize that $\widetilde{T}\widetilde{T}^*$ need not serve as the exact inverse of $\mathcal{A}$: even an approximate inverse can lead to significant performance improvements. This is the foundational principle of preconditioning.

Moreover, performing gradient descent using the transformed parameters $\widehat{\theta}_k := \mathbb{P}\theta_k$ yields

$$\widehat{\theta}_{k+1} = \mathbb{P}\theta_{k+1} = \mathbb{P}\theta_k - \eta\mathbb{P}\mathbb{P}^\top \nabla_\theta L(\mathbb{P}\theta_k) = \widehat{\theta}_k - \eta\mathbb{P}\mathbb{P}^\top \nabla_\theta L(\widehat{\theta}_k). \tag{7.26}$$

Given that any positive definite matrix can be written as $\mathbb{P}\mathbb{P}^\top$, this shows that linearly transforming the parameters is equivalent to preconditioning the gradient of the loss by multiplying by a positive definite matrix. Hence, parameter transformations are all we need in this context.

7.4.2. Preconditioning linear physics-informed machine learning models

We now rigorously analyse the effect of preconditioning linear parametrized models of the form $u_\theta(x) = \sum_k \theta_k \phi_k(x)$, where $\phi_1, \dots, \phi_n$ are any smooth functions, as introduced in Section 2.2.1. A corresponding preconditioned model, as explained above, would have the form $\widetilde{u}_\theta(x) = \sum_k (\mathbb{P}\theta)_k \phi_k(x)$, where $\mathbb{P} \in \mathbb{R}^{n\times n}$ is the preconditioner. We motivate the choice of this preconditioner with a simple, yet widely used example.

Our differential operator is the one-dimensional Laplacian $\mathcal{L} = \mathrm{d}^2/\mathrm{d}x^2$, defined on the domain $(-\pi, \pi)$, for simplicity with periodic zero boundary conditions. Consequently, the corresponding PDE is the Poisson equation (Example 2.4). As the machine learning model, we choose $u_\theta(x) = \sum_{k=-K}^{K} \theta_k \phi_k(x)$ with

$$\phi_0(x) = \frac{1}{\sqrt{2\pi}}, \quad \phi_{-k}(x) = \frac{1}{\sqrt{\pi}}\cos(kx), \quad \phi_k(x) = \frac{1}{\sqrt{\pi}}\sin(kx) \tag{7.27}$$

for $1 \le k \le K$. This model corresponds to the widely used learnable *Fourier features* in the machine learning literature (Tancik *et al.* 2020) or *spectral methods* in numerical analysis (Hesthaven *et al.* 2007). We can readily verify that the resulting matrix $\mathbb{A}$ (7.13) is given by

$$\mathbb{A} = \mathbb{D} + \lambda v v^\top, \tag{7.28}$$

where $\mathbb{D}$ is a diagonal matrix with $\mathbb{D}_{kk} = k^4$ and $v := \phi(\pi)$, that is, v is a vector with $v_{-k} = (-1)^k/\sqrt{\pi}$, $v_0 = 1/\sqrt{2\pi}$, $v_k = 0$ for $1 \le k \le K$. Preconditioning solely based on $\mathcal{L}^*\mathcal{L}$ would correspond to finding a matrix $\mathbb{P}$ such that $\mathbb{P}\mathbb{D}\mathbb{P}^\top = \mathrm{Id}$. However, given that $\mathbb{D}_{00} = 0$, this is not possible. We therefore set $\mathbb{P}_{kk} = 1/k^2$ for $k \neq 0$ and $\mathbb{P}_{00} = \gamma \in \mathbb{R}$. The preconditioned matrix is therefore

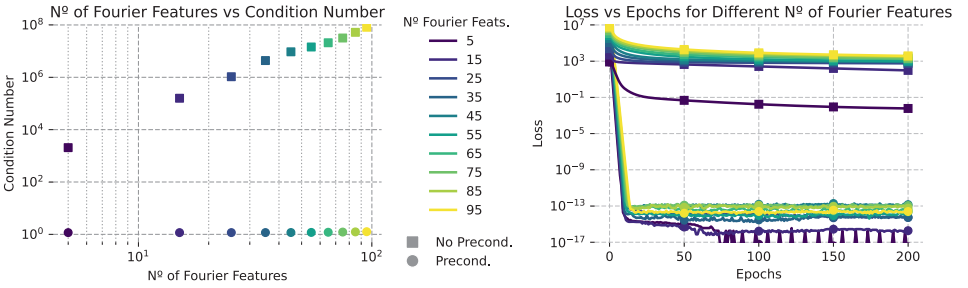
$$\widetilde{\mathbb{A}}(\lambda, \gamma) = \mathbb{P}\mathbb{D}\mathbb{P}^\top + \lambda \mathbb{P}v(\mathbb{P}v)^\top. \tag{7.29}$$

The conditioning of the unpreconditioned and preconditioned matrices considered above are summarized in the following theorem (De Ryck *et al.* 2023).

Theorem 7.12. The following statements hold for all $K \in \mathbb{N}$.

(1) The condition number of the unpreconditioned matrix above satisfies

$$\kappa(\mathbb{A}(\lambda)) \ge K^4. \tag{7.30}$$



(a) (b)

Figure 7.1. Poisson equation with Fourier features. (a) Optimal condition number vs. number of Fourier features. (b) Training for the unpreconditioned and preconditioned Fourier features. Figure from De Ryck *et al.* (2023).

(2) There exists a constant $C(\lambda, \gamma) > 0$ that is independent of K such that

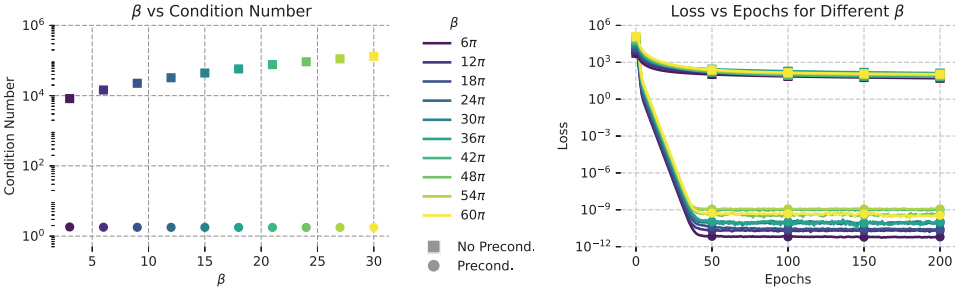
$$\kappa(\widetilde{\mathbb{A}}(\lambda, \gamma) \leq C. \tag{7.31}$$

(3) It holds that $\kappa(\widetilde{\mathbb{A}}(2\pi/\gamma^2, \gamma)) = 1 + O(1/\gamma)$ and hence

$$\lim_{\gamma \to +\infty} \kappa(\widetilde{\mathbb{A}}(2\pi/\gamma^2, \gamma)) = 1. \tag{7.32}$$

We observe from Theorem 7.12 that (i) the matrix $\mathbb{A}$, which governs gradient descent dynamics for approximating the Poisson equation with learnable Fourier features, is very poorly conditioned, and (ii) we can (optimally) precondition it by *rescaling* the Fourier features based on the eigenvalues of the underlying differential operator (or its Hermitian square).

These conclusions are also observed empirically. In Figure 7.1(a) we plot the condition number of the matrix $\mathbb{A}$, minimized over λ, as a function of maximum frequency K and verify that this condition number increases as K^4, as predicted by Theorem 7.12. Consequently, as shown in Figure 7.1(b), where we plot the loss function in terms of increasing training epochs, the model is very hard to train with large losses (particularly for higher values of K), showing a very slow decay of the loss function as the number of frequencies is increased. On the other hand, in Figure 7.1(a) we also show that the condition number (minimized over λ) of the preconditioned matrix (7.29) remains constant with increasing frequency and is very close to the optimal value of 1, verifying Theorem 7.12. As a result, we observe from Figure 7.1(b) that the loss in the preconditioned case decays exponentially fast as the number of epochs is increased. This decay is independent of the maximum frequency of the model. The results demonstrate that the preconditioned version of the Fourier features model can learn the solution of the Poisson equation efficiently, in contrast to the failure of the unpreconditioned model to do so. Entirely analogous results are obtained with the Helmholtz equation (De Ryck *et al.* 2023).



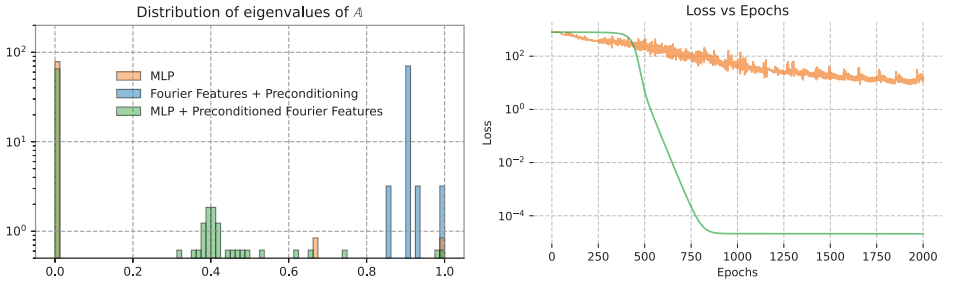
(a) (b)

Figure 7.2. Linear advection equation with Fourier features. (a) Optimal condition number vs. β. (b) Training for the unpreconditioned and preconditioned Fourier features. Figure from De Ryck *et al.* (2023).

As a different example, we consider the linear advection equation $u_t + \beta u_x = 0$ on the one-dimensional spatial domain $x \in [0, 2\pi]$ and with 2π-periodic solutions in time with $t \in [0, 1]$. As in Krishnapriyan *et al.* (2021), our focus in this case is to study how physics-informed machine learning models train when the advection speed $\beta > 0$ is increased. To empirically evaluate this example, we again choose learnable time-dependent Fourier features as the model and precondition the resulting matrix $\mathbb{A}$ (7.13). In Figure 7.2(a) we see that the condition number of $\mathbb{A}(\beta) \sim \beta^2$ grows quadratically with advection speed. On the other hand, the condition number of the preconditioned model remains constant. Consequently, as shown in Figure 7.2(b), the unpreconditioned model trains very slowly (particularly for increasing values of the advection speed β), with losses remaining high despite being trained for a large number of epochs. In complete contrast, the preconditioned model trains very fast, irrespective of the values of the advection speed β.

7.4.3. Preconditioning nonlinear physics-informed machine learning models

Next, De Ryck *et al.* (2023) have investigated the conditioning of nonlinear models (Section 2.2.2) by considering the Poisson equation on $(-\pi, \pi)$ and learning its solution with neural networks of the form $u_\theta(x) = \Phi_\theta(x)$. It was observed that most of the eigenvalues of the resulting matrix $\mathbb{A}$ (7.13) are clustered near zero. Consequently, it is difficult to analyse the condition number *per se.* However, they also observed that there are only a couple of large non-zero eigenvalues of $\mathbb{A}$, indicating a very uneven spread of the spectrum. It is well known in classical numerical analysis (Trefethen and Embree 2005) that such spread-out spectra are very poorly conditioned and that this will impede training with gradient descent. This is corroborated in Figure 7.3(b), where the physics-informed MLP trains very slowly. Moreover, it turns out that preconditioning also localizes the spectrum (Trefethen and Embree 2005). This is attested in Figure 7.3(a), where we see



(a) (b)

Figure 7.3. Poisson equation with MLPs. (a) Histogram of normalized spectrum (eigenvalues multiplied with learning rate). (b) Loss vs. number of epochs. Figure from De Ryck *et al.* (2023).

the localized spectrum of the preconditioned Fourier features model considered previously, which is also correlated with its fast training (Figure 7.1(b)).

However, preconditioning $\mathbb{A}$ for nonlinear models such as neural networks can, in general, be hard. Here we consider a simple strategy by coupling the MLP with Fourier features ϕ_k defined above, i.e. setting

$$u_\theta = \Phi_\theta\left(\sum_k \alpha_k \phi_k\right).$$

Intuitively, we must choose α_k carefully to control $(\mathrm{d}^{2n}/\mathrm{d}x^{2n})u_\theta(x)$, as it will include terms such as

$$\left(\sum_{k\neq 0} \alpha_k (-k)^{2n} \phi_k\right) \Phi_\theta^{(2n)}\left(\sum_k \alpha_k \phi_k\right).$$

Hence, such rescaling could better condition the Hermitian square of the differential operator. To test this for Poisson's equation, we choose $\alpha_k = 1/k^2$ (for $k \neq 0$) in this FF-MLP model and present the eigenvalues in Figure 7.3(a), which shows that although there are still quite a few (near-) zero eigenvalues, the number of non-zero eigenvalues is significantly increased, leading to a much more even spread in the spectrum when compared to the unpreconditioned MLP case. This possibly accounts for the fact that the resulting loss function decays much more rapidly, as shown in Figure 7.3(b), when compared to unpreconditioned MLP.

7.5. Strategies for improving training in physics-informed machine learning

Given the difficulties encountered in training physics-informed machine learning models, several *ad hoc* strategies have been proposed in the recent literature to improve training. It turns out that many of these strategies can also be interpreted in terms of preconditioning.

7.5.1. Choice of λ

The parameter λ in a physics-informed loss such as (2.46) plays a crucial role, as it balances the relative contributions of the physics-informed loss R and the supervised loss at the boundary B. Given the analysis of the previous sections, it is natural to suggest that this parameter should be chosen as

$$\lambda^* := \min_{\lambda} \kappa(\mathbb{A}(\lambda)) \tag{7.33}$$

in order to obtain the smallest condition number of $\mathbb{A}$ and accelerate convergence. Another strategy is suggested by Wang *et al.* (2021*a*), who suggest a learning rate annealing algorithm to iteratively update λ throughout training as follows:

$$\lambda_a^* := \frac{\max_k |(\nabla_\theta R(\theta))_k|}{(2K+1)^{-1} \sum_k |(\nabla_\theta B(\theta))_k|}. \tag{7.34}$$

Similarly, Wang *et al.* (2022*b*) propose setting

$$\lambda_b^* := \frac{\mathrm{Tr}(K_{rr}(n))}{\mathrm{Tr}(K_{uu}(n))} \approx \frac{\int_\Omega \nabla_\theta \mathcal{L} u_\theta^\top \nabla_\theta \mathcal{L} u_\theta}{\int_{\partial\Omega} \nabla_\theta u_\theta^\top \nabla_\theta u_\theta}; \tag{7.35}$$

we refer to Wang *et al.* (2022*b*) for the exact definition of K_{rr} and K_{uu}.

In the below example, we compare λ^* (7.33), λ_a^* (7.34) and λ_b^* (7.35) for the Poisson equation.

Example 7.13 (Poisson's equation). We compare the three proposed strategies above for the one-dimensional Poisson equation and a linear model using Fourier features up to frequency K. First of all, De Ryck *et al.* (2023) computed that $\lambda^* \sim K^2$. Next, to calculate λ_a^* we observe that $(\nabla_\theta R(\theta))_k = k^4 \theta_k$ for $k \neq \ell$ and $(\nabla_\theta R(\theta))_\ell = \ell^4(\theta_k - 1)$. We find that $\max_k |(\nabla_\theta R(\theta))_k| \sim K^4$ at initialization. Next we calculate that (given that at initialization $\theta_m \sim \mathcal{N}(0,1)$ i.i.d.)

$$\mathbb{E} \sum_k |(\nabla_\theta B(\theta))_k| = \sum_{k=0}^K \mathbb{E} \left| \sum_{m=0}^K \theta_m (-1)^{k+m} \right| = \sum_{k=0}^K \mathbb{E} \left| \sum_{m=0}^K \theta_m \right| = (K+1)^{3/2} \tag{7.36}$$

This brings us to the rough estimate that

$$(2K+1)^{-1} \sum_k |(\nabla_\theta B(\theta))_k| \sim \sqrt{K}.$$

So in total we find that $\lambda_a^* \sim K^{3.5}$. Finally, for λ_b^* one can make the estimate that

$$\lambda_b^* \approx \frac{2\pi \sum_{k=1}^K k^4}{K+1} \sim K^4, \tag{7.37}$$

Hence it turns out that applying these different strategies leads to different scalings of λ with respect to increasing K for the Fourier features model. Despite these differences, it was observed that the resulting condition numbers $\kappa(\mathbb{A}(\lambda^*))$, $\kappa(\mathbb{A}(\lambda_a^*))$ and $\kappa(\mathbb{A}(\lambda_b^*))$ are actually very similar.

7.5.2. Hard boundary conditions

From the very advent of PINNs (Lagaris *et al.* 1998, 2000), several authors have advocated modifying machine learning models such that the boundary conditions in PDE (2.1) can be imposed exactly and the boundary loss in (2.46) is zero. Such *hard* imposition of boundary conditions (BCs) has been empirically shown to aid training; see e.g. Dong and Ni (2021), Moseley *et al.* (2023), Dolean *et al.* (2023) and references therein. This can be explained by saying that implementing hard boundary conditions leads to a different matrix $\mathbb{A}$, by changing the operator TT^*.

We first consider a toy model to demonstrate these phenomena in a straightforward way. We again consider the Poisson equation $-\Delta u = -\sin$ with zero Dirichlet boundary conditions (see Example 2.4). We choose as our model

$$u_\theta(x) = \theta_{-1}\cos(x) + \theta_0 + \theta_1 \sin(x), \tag{7.38}$$

and report the comparison of the condition number between various variants of soft and hard boundary conditions from De Ryck *et al.* (2023).

- *Soft boundary conditions.* If we set the optimal λ using (7.33), we find that the condition number for the optimal λ^* is given by $3 + 2\sqrt{2} \approx 5.83$.
- *Hard boundary conditions: variant 1.* A first common method to implement hard boundary conditions is to multiply the model by a function $\eta(x)$ so that the product exactly satisfies the boundary conditions, regardless of u_θ. In our setting, we could consider $\eta(x)u_\theta(x)$ with $\eta(\pm\pi) = 0$. For $\eta = \sin$ the total model is given by

$$\eta(x)u_\theta(x) = -\frac{\theta_{-1}}{2}\cos(2x) + \frac{\theta_{-1}}{2} + \theta_0 \sin(x) + \frac{\theta_1}{2}\sin(2x), \tag{7.39}$$

  and gives rise to a condition number of 4. Different choices of η will inevitably lead to different condition numbers.
- *Hard boundary conditions: variant 2.* Another option would be to subtract $u_\theta(\pi)$ from the model so that the boundary conditions are exactly satisfied. This corresponds to the model

$$u_\theta(x) - u_\theta(\pi) = \theta_{-1}(\cos(x) + 1) + \theta_1 \sin(x). \tag{7.40}$$

  Note that this implies that one can discard θ_0 as parameter, leaving only two trainable parameters. The corresponding condition number is 1.

Hence, in this example the condition number for hard boundary conditions is strictly smaller than for soft boundary conditions, that is,

$$\kappa(\mathbb{A}_{\text{hard BC}}) < \min_\lambda \kappa(\mathbb{A}_{\text{soft BC}}(\lambda)). \tag{7.41}$$

This phenomenon can also be observed for other PDEs such as the linear advection equation (De Ryck *et al.* 2023).

7.5.3. Second-order optimizers

There are many empirical studies which demonstrate that first-order optimizers such as (stochastic) gradient descent or ADAM are not suitable for physics-informed machine learning, and we need to use second-order (quasi-) Newton-type optimizers such as L-BGFS in order to make training of physics-informed machine learning models feasible. As it turns out, one can prove that for linear physics-informed models the Hessian of the loss is identical to the matrix $\mathbb{A}$ (7.13). Given any loss $\mathcal{J}(\theta)$, the gradient flow of Newton's method is given by

$$\frac{\mathrm{d}\theta(t)}{\mathrm{d}t} = -\gamma H[\mathcal{J}(\theta(t))]^{-1}\nabla_\theta \mathcal{J}(\theta(t)), \tag{7.42}$$

where H is the Hessian. In our case, we consider the physics-informed loss $L(\theta) = R(\theta) + \lambda B(\theta)$ and consider the model $u_\theta(x) = \sum_\ell \theta_\ell \phi_\ell(x)$, which corresponds to a linear model or a neural network in the NTK regime (e.g. Lemma 7.8). Using $\partial_{\theta_i}\partial_{\theta_j} u_\theta = 0$, we calculate

$$\begin{aligned}\partial_{\theta_i}\partial_{\theta_j} L(\theta) &= \int_\Omega (\mathcal{L}\partial_{\theta_i} u_\theta(x)) \cdot \mathcal{L}\partial_{\theta_j} u_\theta(x)\,\mathrm{d}x + \lambda \int_{\partial\Omega} \partial_{\theta_i} u_{\theta(t)}(x) \cdot \partial_{\theta_j} u_\theta(x)\,\mathrm{d}x \\ &= \int_\Omega \mathcal{L}\phi_i(x) \cdot \mathcal{L}\phi_j(x)\,\mathrm{d}x + \lambda \int_{\partial\Omega} \phi_i(x) \cdot \phi_j(x)\,\mathrm{d}x \\ &= \mathbb{A}_{ij}. \end{aligned} \tag{7.43}$$

Therefore, in this case (quasi-) Newton methods automatically compute an (approximate) inverse of the Hessian and hence precondition the matrix $\mathbb{A}$, relating the use of (quasi-) Newton-type optimizers to preconditioning operators in this context. See Müller and Zeinhofer (2023) for further analysis of this connection.

7.5.4. Domain decomposition

Domain decomposition (DD) is a widely used technique in numerical analysis to precondition linear systems that arise out of classical methods such as finite elements (Dolean, Jolivet and Nataf 2015). Recently there have been attempts to use DD-inspired methods within physics-informed machine learning (see Moseley *et al.* 2023, Dolean *et al.* 2023 and references therein), although no explicit link with preconditioning the models was established. This link was established in De Ryck *et al.* (2023) for the linear advection equation, where they computed that the condition number of $\mathbb{A}$ depends quadratically on the advection speed β. They argued that considering N identical submodels on subintervals of $[0, T]$ essentially comes down to rescaling β. Since the condition number scales as β^2, splitting the time domain in N parts will then reduce the condition number of each individual submodel by a factor of N^2. This is also intrinsically connected to what happens in *causal learning* based training of PINNs (Wang, Sankaran and Perdikaris 2024), which therefore can also be viewed as a strategy for improving the condition number.

8. Conclusion

In this article we have provided a detailed review of available theoretical results on the numerical analysis of physics-informed machine learning models, such as PINNs and its variants, for the solution of forward and inverse problems for large classes of PDEs. We formulated physics-informed machine learning in terms of minimizing suitable forms (strong, weak, variational) of the *residual* of the underlying PDE within suitable subspaces of Ansatz spaces of *parametric functions*. PINNs are a special case of this general formulation corresponding to the strong form of the PDE residual and neural networks as Ansatz spaces. The residual is minimized with variants of the (stochastic) gradient descent algorithm.

Our analysis revealed that as long as the solutions to the PDE are *stable* in a suitable sense, the overall (total) error, which is the mismatch between the exact PDE solution and the approximation produced (after training) of the physics-informed machine learning model, can be estimated in terms of the following constituents: (i) the approximation error, which measures the smallness of the PDE residual within the Ansatz space (or hypothesis class), (ii) the generalization gap, which measures the *quadrature* error, and (iii) the optimization error, which quantifies how well the optimization problem is solved. Detailed analysis of each component of the error is provided, both in general terms as well as exemplified for many prototypical PDEs. Although a lot of the analysis is PDE-specific, we can discern many features that are shared across PDEs and models, which we summarize below.

- The (weak) stability of the underlying PDE plays a key role in the analysis, and this can be PDE-specific. In general, error estimates can depend on how the solution of the underlying PDEs changes with respect to perturbations. This is also consistent with the analysis of traditional numerical methods.
- Sufficient regularity of the underlying PDE solution is often required in the analysis, although this regularity can be lessened by changing the form of the PDE.
- The key bottleneck in physics-informed machine learning is due to the inability of the models to train properly. Generically, training PINNs and their variants can be difficult, as it depends on the spectral properties of the Hermitian square of the underlying differential operator, although suitable preconditioning might ameliorate training issues.

Finally, theory does provide some guidance to practitioners about how and when PINNs and their variants can be used. Succinctly, regularity dictates the form (strong vs. weak), and physics-informed machine learning models are expected to outperform traditional methods on high-dimensional problems (see Mishra and Molinaro 2021 as an example) or problems with complex geometries, where grid generation is prohibitively expensive, or inverse problems, where data supplements the physics. Physics-informed machine learning will be particularly attractive when coupled with operator learning models such as neural operators, as the underlying

space is infinite-dimensional. More research on physics-informed operator learning models is highly desirable.

Appendix: Sobolev spaces

Let $d \in \mathbb{N}$, $k \in \mathbb{N}_0$, $1 \le p \le \infty$ and let $\Omega \subseteq \mathbb{R}^d$ be open. For a function $f\colon \Omega \to \mathbb{R}$ and a (multi-) index $\alpha \in \mathbb{N}_0^d$, we let

$$D^\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}} \tag{A.1}$$

denote the classical or distributional (i.e. weak) derivative of f. We let $L^p(\Omega)$ denote the usual Lebesgue space, and we define the Sobolev space $W^{k,p}(\Omega)$ as

$$W^{k,p}(\Omega) = \{f \in L^p(\Omega)\colon D^\alpha f \in L^p(\Omega) \text{ for all } \alpha \in \mathbb{N}_0^d \text{ with } |\alpha| \le k\}. \tag{A.2}$$

We define the following seminorms on $W^{k,p}(\Omega)$: for $p < \infty$,

$$|f|_{W^{m,p}(\Omega)} = \left(\sum_{|\alpha|=m} \|D^\alpha f\|_{L^p(\Omega)}^p \right)^{1/p} \quad \text{for } m = 0, \dots, k, \tag{A.3}$$

and for $p = \infty$,

$$|f|_{W^{m,\infty}(\Omega)} = \max_{|\alpha|=m} \|D^\alpha f\|_{L^\infty(\Omega)} \quad \text{for } m = 0, \dots, k. \tag{A.4}$$

Based on these seminorms, we can define the following norms: for $p < \infty$,

$$\|f\|_{W^{k,p}(\Omega)} = \left(\sum_{m=0}^{k} |f|_{W^{m,p}(\Omega)}^p \right)^{1/p}, \tag{A.5}$$

and for $p = \infty$,

$$\|f\|_{W^{k,\infty}(\Omega)} = \max_{0 \le m \le k} |f|_{W^{m,\infty}(\Omega)}. \tag{A.6}$$

The space $W^{k,p}(\Omega)$ equipped with the norm $\|\cdot\|_{W^{k,p}(\Omega)}$ is a Banach space.

We let $C^k(\Omega)$ denote the space of functions that are k times continuously differentiable, and equip this space with the norm $\|f\|_{C^k(\Omega)} = \|f\|_{W^{k,\infty}(\Omega)}$.

References

G. Alessandrini, L. Rondi, E. Rosset and S. Vessella (2009), The stability of the Cauchy problem for elliptic equations, *Inverse Problems* **25**, art. 123004.

S. M. Allen and J. W. Cahn (1979), A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening, *Acta Metallurg.* **27**, 1085–1095.

F. Bach (2017), Breaking the curse of dimensionality with convex neural networks, *J. Mach. Learn. Res.* **18**, 629–681.

A. R. Barron (1993), Universal approximation bounds for superpositions of a sigmoidal function, *IEEE Trans. Inform. Theory* **39**, 930–945.

S. Bartels (2012), Total variation minimization with finite elements: Convergence and iterative solution, *SIAM J. Numer. Anal.* **50**, 1162–1180.
F. Bartolucci, E. de Bézenac, B. Raonić, R. Molinaro, S. Mishra and R. Alaifari (2023), Representation equivalent neural operators: A framework for alias-free operator learning, in *Advances in Neural Information Processing Systems 36* (A. Oh *et al.*, eds), Curran Associates, pp. 69661–69672.
C. Beck, S. Becker, P. Grohs, N. Jaafari and A. Jentzen (2021), Solving the Kolmogorov PDE by means of deep learning, *J. Sci. Comput.* **88**, 1–28.
C. Beck, A. Jentzen and B. Kuckuck (2022), Full error analysis for the training of deep neural networks, *Infin. Dimens. Anal. Quantum Probab. Relat. Top.* **25**, art. 2150020.
J. Berner, P. Grohs and A. Jentzen (2020), Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations, *SIAM J. Math. Data Sci.* **2**, 631–657.
P. B. Bochev and M. D. Gunzburger (2009), *Least-Squares Finite Element Methods*, Vol. 166 of Applied Mathematical Sciences, Springer.
M. D. Buhmann (2000), Radial basis functions, *Acta Numer.* **9**, 1–38.
E. Burman and P. Hansbo (2018), Stabilized nonconfirming finite element methods for data assimilation in incompressible flows, *Math. Comp.* **87**, 1029–1050.
T. Chen and H. Chen (1995), Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, *IEEE Trans. Neural Netw.* **6**, 911–917.
Z. Chen, J. Lu and Y. Lu (2021), On the representation of solutions to elliptic PDEs in Barron spaces, in *Advances in Neural Information Processing Systems 34* (M. Ranzato *et al.*, eds), Curran Associates, pp. 6454–6465.
Z. Chen, J. Lu, Y. Lu and S. Zhou (2023), A regularity theory for static Schrödinger equations on $\mathbb{R}^d$ in spectral Barron spaces, *SIAM J. Math. Anal.* **55**, 557–570.
S. Cuomo, V. S. di Cola, F. Giampaolo, G. Rozza, M. Raissi and F. Piccialli (2022), Scientific machine learning through physics-informed neural networks: Where we are and what's next, *J. Sci. Comput.* **92**, art. 88.
G. Cybenko (1989), Approximation by superpositions of a sigmoidal function, *Math. Control Signals Systems* **2**, 303–314.
R. Dautray and J. L. Lions (1992), *Mathematical Analysis and Numerical Methods for Science and Technology*, Vol. 5, *Evolution Equations I*, Springer.
T. De Ryck and S. Mishra (2022*a*), Error analysis for physics-informed neural networks (PINNs) approximating Kolmogorov PDEs, *Adv. Comput. Math.* **48**, art. 79.
T. De Ryck and S. Mishra (2022*b*), Generic bounds on the approximation error for physics-informed (and) operator learning, in *Advances in Neural Information Processing Systems 35* (S. Koyejo *et al.*, eds), Curran Associates, pp. 10945–10958.
T. De Ryck and S. Mishra (2023), Error analysis for deep neural network approximations of parametric hyperbolic conservation laws, *Math. Comp.* Available at https://doi.org/10.1090/mcom/3934.
T. De Ryck, F. Bonnet, S. Mishra and E. de Bézenac (2023), An operator preconditioning perspective on training in physics-informed machine learning. Available at arXiv:2310.05801.
T. De Ryck, A. D. Jagtap and S. Mishra (2024*a*), Error estimates for physics-informed neural networks approximating the Navier–Stokes equations, *IMA J. Numer. Anal.* **44**, 83–119.

T. De Ryck, A. D. Jagtap and S. Mishra (2024*b*), On the stability of XPINNs and cPINNs. In preparation.
T. De Ryck, S. Lanthaler and S. Mishra (2021), On the approximation of functions by tanh neural networks, *Neural Networks* **143**, 732–750.
T. De Ryck, S. Mishra and R. Molinaro (2024*c*), wPINNs: Weak physics informed neural networks for approximating entropy solutions of hyperbolic conservation laws, *SIAM J. Numer. Anal.* **62**, 811–841.
M. W. M. G. Dissanayake and N. Phan-Thien (1994), Neural-network-based approximations for solving partial differential equations, *Commun. Numer. Methods Engrg* **10**, 195–201.
V. Dolean, A. Heinlein, S. Mishra and B. Moseley (2023), Multilevel domain decomposition-based architectures for physics-informed neural networks. Available at arXiv:2306.05486.
V. Dolean, P. Jolivet and F. Nataf (2015), *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*, SIAM.
S. Dong and N. Ni (2021), A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks, *J. Comput. Phys.* **435**, art. 110242.
W. E and S. Wojtowytsch (2020), On the Banach spaces associated with multi-layer ReLU networks: Function representation, approximation theory and gradient descent dynamics. Available at arXiv:2007.15623.
W. E and S. Wojtowytsch (2022*a*), Representation formulas and pointwise properties for Barron functions, *Calc. Var. Partial Differential Equations* **61**, 1–37.
W. E and S. Wojtowytsch (2022*b*), Some observations on high-dimensional partial differential equations with Barron data, in *Proceedings of the Second Conference on Mathematical and Scientific Machine Learning*, Vol. 145 of Proceedings of Machine Learning Research, PMLR, pp. 253–269.
W. E and B. Yu (2018), The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems, *Commun. Math. Statist.* **6**, 1–12.
W. E, C. Ma and L. Wu (2022), The Barron space and the flow-induced function spaces for neural network models, *Constr. Approx.* **55**, 369–406.
L. C. Evans (2022), *Partial Differential Equations*, Vol. 19 of Graduate Studies in Mathematics, American Mathematical Society.
A. Friedman (1964), *Partial Differential Equations of the Parabolic Type*, Prentice Hall.
X. Glorot and Y. Bengio (2010), Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Vol. 9 of Proceedings of Machine Learning Research, PMLR, pp. 249–256.
E. Godlewski and P. A. Raviart (1991), *Hyperbolic Systems of Conservation Laws*, Ellipsis.
I. Goodfellow, Y. Bengio and A. Courville (2016), *Deep Learning*, MIT Press.
P. Grohs, F. Hornung, A. Jentzen and P. von Wurstemberger (2018), A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations. Available at arXiv:1809.02362.
I. Gühring and M. Raslan (2021), Approximation rates for neural networks with encodable weights in smoothness spaces, *Neural Networks* **134**, 107–130.
J. Han, A. Jentzen and W. E (2018), Solving high-dimensional partial differential equations using deep learning, *Proc. Nat. Acad. Sci.* **115**, 8505–8510.

J. S. Hesthaven, S. Gottlieb and D. Gottlieb (2007), *Spectral Methods for Time-Dependent Problems*, Vol. 21 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press.

H. Holden and N. H. Risebro (2015), *Front Tracking for Hyperbolic Conservation Laws*, Vol. 152 of Applied Mathematical Sciences, Springer.

Q. Hong, J. W. Siegel and J. Xu (2021), *A priori* analysis of stable neural network solutions to numerical PDEs. Available at arXiv:2104.02903.

G.-B. Huang, Q.-Y. Zhu and C.-K. Siew (2006), Extreme learning machine: Theory and applications, *Neurocomput.* **70**, 489–501.

M. Hutzenthaler, A. Jentzen, T. Kruse and T. A. Nguyen (2020), A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations, *SN Partial Differ. Equ. Appl.* **1**, 1–34.

O. Y. Imanuvilov (1995), Controllability of parabolic equations, *Sb. Math.* **186**, 109–132.

A. D. Jagtap and G. E. Karniadakis (2020), Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, *Commun. Comput. Phys.* **28**, 2002–2041.

A. D. Jagtap, E. Kharazmi and G. E. Karniadakis (2020), Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, *Comput. Methods Appl. Mech. Engrg* **365**, art. 113028.

A. Jentzen, D. Salimova and T. Welti (2021), A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients, *Commun. Math. Sci.* **19**, 1167–1205.

G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang and L. Yang (2021), Physics informed machine learning, *Nat. Rev. Phys.* **3**, 422–440.

E. Kharazmi, Z. Zhang and G. E. Karniadakis (2019), Variational physics-informed neural networks for solving partial differential equations. Available at arXiv:1912.00873.

E. Kharazmi, Z. Zhang and G. E. Karniadakis (2021), hp-VPINNs: Variational physics-informed neural networks with domain decomposition, *Comput. Methods Appl. Mech. Engrg* **374**, art. 113547.

D. P. Kingma and J. Ba (2015), Adam: A method for stochastic optimization, in *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

N. B. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. M. Stuart and A. Anandkumar (2023), Neural operator: Learning maps between function spaces with applications to PDEs, *J. Mach. Learn. Res.* **24**, 1–97.

N. Kovachki, S. Lanthaler and S. Mishra (2021), On universal approximation and error bounds for Fourier neural operators, *J. Mach. Learn. Res.* **22**, 13237–13312.

A. S. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby and M. W. Mahoney (2021), Characterizing possible failure modes in physics-informed neural networks, in *Advances in Neural Information Processing Systems 34* (M. Ranzato *et al.*, eds), Curran Associates, pp. 26548–26560.

G. Kutyniok, P. Petersen, M. Raslan and R. Schneider (2022), A theoretical analysis of deep neural networks and parametric PDEs, *Constr. Approx.* **55**, 73–125.

I. E. Lagaris, A. Likas and D. I. Fotiadis (1998), Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Netw.* **9**, 987–1000.

I. E. Lagaris, A. Likas and G. D. Papageorgiou (2000), Neural-network methods for boundary value problems with irregular boundaries, *IEEE Trans. Neural Netw.* **11**, 1041–1049.
S. Lanthaler, S. Mishra and G. E. Karniadakis (2022), Error estimates for DeepONets: A deep learning framework in infinite dimensions, *Trans. Math. Appl.* **6**, tnac001.
R. J. LeVeque (2002), *Finite Volume Methods for Hyperbolic Problems*, Vol. 31 of Cambridge Texts in Applied Mathematics, Cambridge University Press.
Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart and A. Anandkumar (2020), Neural operator: Graph kernel network for partial differential equations. Available at arXiv:2003.03485.
Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart and A. Anandkumar (2021), Fourier neural operator for parametric partial differential equations, in *International Conference on Learning Representations (ICLR 2021)*.
C.-L. Lin, G. Uhlmann and J.-N. Wang (2010), Optimal three-ball inequalities and quantitative uniqueness for the Stokes system, *Discrete Contin. Dyn. Syst.* **28**, 1273–1290.
C. Liu, L. Zhu and M. Belkin (2020), On the linearity of large non-linear models: When and why the tangent kernel is constant, in *Advances in Neural Information Processing Systems 33* (H. Larochelle *et al.*, eds), Curran Associates, pp. 15954–15964.
D. C. Liu and J. Nocedal (1989), On the limited memory BFGS method for large scale optimization, *Math. Program.* **45**, 503–528.
M. Longo, S. Mishra, T. K. Rusch and C. Schwab (2021), Higher-order quasi-Monte Carlo training of deep neural networks, *SIAM J. Sci. Comput.* **43**, A3938–A3966.
L. Lu, P. Jin, G. Pang, Z. Zhang and G. E. Karniadakis (2021*a*), Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nat. Mach. Intell.* **3**, 218–229.
Y. Lu, H. Chen, J. Lu, L. Ying and J. Blanchet (2021*b*), Machine learning for elliptic PDEs: Fast rate generalization bound, neural scaling law and minimax optimality, in *International Conference on Learning Representations (ICLR 2022)*.
Y. Lu, J. Lu and M. Wang (2021*c*), *A priori* generalization analysis of the deep Ritz method for solving high dimensional elliptic partial differential equations, in *Proceedings of Thirty Fourth Conference on Learning Theory*, Vol. 134 of Proceedings of Machine Learning Research, PMLR, pp. 3196–3241.
K. O. Lye, S. Mishra and D. Ray (2020), Deep learning observables in computational fluid dynamics, *J. Comput. Phys.* **410**, art. 109339.
A. J. Majda and A. L. Bertozzi (2002), *Vorticity and Incompressible Flow*, Vol. 55 of Cambridge Texts in Applied Mathematics, Cambridge University Press.
T. Marwah, Z. Lipton and A. Risteski (2021), Parametric complexity bounds for approximating PDEs with neural networks, in *Advances in Neural Information Processing Systems 34* (M. Ranzato *et al.*, eds), Curran Associates, pp. 15044–15055.
S. Mishra and R. Molinaro (2021), Physics informed neural networks for simulating radiative transfer, *J. Quant. Spectrosc. Radiative Transfer* **270**, art. 107705.
S. Mishra and R. Molinaro (2022), Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs, *IMA J. Numer. Anal.* **42**, 981–1022.
S. Mishra and R. Molinaro (2023), Estimates on the generalization error of physics-informed neural networks for approximating PDEs, *IMA J. Numer. Anal.* **43**, 1–43.
M. F. Modest (2003), *Radiative Heat Transfer*, Elsevier.

R. Molinaro (2023), Applications of deep learning to scientific computing. PhD thesis, ETH Zürich.
B. Moseley, A. Markham and T. Nissen-Meyer (2023), Finite basis physics-informed neural networks (FBPINNs): A scalable domain decomposition approach for solving differential equations, *Adv. Comput. Math.* **49**, art. 62.
J. Müller and M. Zeinhofer (2023), Achieving high accuracy with PINNs via energy natural gradient descent, in *Proceedings of the 40th International Conference on Machine Learning* (A. Krause *et al.*, eds), Vol. 202 of Proceedings of Machine Learning Research, PMLR, pp. 25471–25485.
J. A. A. Opschoor, P. C. Petersen and C. Schwab (2020), Deep ReLU networks and high-order finite element methods, *Anal. Appl.* **18**, 715–770.
P. Petersen and F. Voigtlaender (2018), Optimal approximation of piecewise smooth functions using deep ReLU neural networks, *Neural Networks* **108**, 296–330.
N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio and A. Courville (2019), On the spectral bias of neural networks, in *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, Vol. 97 of Proceedings of Machine Learning Research, PMLR, pp. 5301–5310.
A. Rahimi and B. Recht (2008), Uniform approximation of functions with random bases, in *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, IEEE, pp. 555–561.
M. Raissi and G. E. Karniadakis (2018), Hidden physics models: Machine learning of nonlinear partial differential equations, *J. Comput. Phys.* **357**, 125–141.
M. Raissi, P. Perdikaris and G. E. Karniadakis (2019), Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* **378**, 686–707.
M. Raissi, A. Yazdani and G. E. Karniadakis (2018), Hidden fluid mechanics: A Navier–Stokes informed deep learning framework for assimilating flow visualization data. Available at arXiv:1808.04327.
B. Raonić, R. Molinaro, T. D. Ryck, T. Rohner, F. Bartolucci, R. Alaifari, S. Mishra and E. de Bézenac (2023), Convolutional neural operators for robust and accurate learning of PDEs, in *Advances in Neural Information Processing Systems 36* (A. Oh *et al.*, eds), Curran Associates, pp. 77187–77200.
C. E. Rasmussen (2003), Gaussian processes in machine learning, in *Summer School on Machine Learning*, Springer, pp. 63–71.
C. Schwab and J. Zech (2019), Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in uq, *Anal. Appl.* **17**, 19–55.
Y. Shin (2020), On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs, *Commun. Comput. Phys.* **28**, 2042–2074.
Y. Shin, Z. Zhang and G. E. Karniadakis (2023), Error estimates of residual minimization using neural networks for linear PDEs, *J. Mach. Learn. Model. Comput.* **4**, 73–101.
J. Stoer and R. Bulirsch (2002), *Introduction to Numerical Analysis*, Springer.
M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron and R. Ng (2020), Fourier features let networks learn high frequency functions in low dimensional domains, in *Advances in Neural Information Processing Systems 33* (H. Larochelle *et al.*, eds), Curran Associates, pp. 7537–7547.
R. Temam (2001), *Navier–Stokes Equations: Theory and Numerical Analysis*, American Mathematical Society.

N. Trefethen and M. Embree (2005), *Spectra and Pseudo-Spectra*, Princeton University Press.
R. Verfürth (1999), A note on polynomial approximation in Sobolev spaces, *ESAIM Math. Model. Numer. Anal.* **33**, 715–719.
C. Wang, S. Li, D. He and L. Wang (2022*a*), Is L^2 physics informed loss always suitable for training physics informed neural network?, in *Advances in Neural Information Processing Systems 35* (S. Koyejo *et al.*, eds), Curran Associates, pp. 8278–8290.
S. Wang, S. Sankaran and P. Perdikaris (2024), Respecting causality for training physics-informed neural networks, *Comput. Methods Appl. Mech. Engrg* **421**, art. 116813.
S. Wang, Y. Teng and P. Perdikaris (2021*a*), Understanding and mitigating gradient flow pathologies in physics-informed neural networks, *SIAM J. Sci. Comput.* **43**, A3055–A3081.
S. Wang, H. Wang and P. Perdikaris (2021*b*), Learning the solution operator of parametric partial differential equations with physics-informed DeepONets, *Sci. Adv.* **7**, eabi8605.
S. Wang, X. Yu and P. Perdikaris (2022*b*), When and why PINNs fail to train: A neural tangent kernel perspective, *J. Comput. Phys.* **449**, art. 110768.