ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics SAM

Error estimates for physics informed neural networks approximating the Navier-Stokes equations

T. De Ryck and A. D. Jagtap and S. Mishra

Research Report No. 2022-08

March 2022
Latest revision: February 2023

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

ERROR ESTIMATES FOR PHYSICS-INFORMED NEURAL NETWORKS APPROXIMATING THE NAVIER-STOKES EQUATIONS

TIM DE RYCK, AMEYA D. JAGTAP, AND SIDDHARTHA MISHRA

Abstract. We prove rigorous bounds on the errors resulting from the approximation of the incompressible Navier-Stokes equations with (extended) physics-informed neural networks. We show that the underlying PDE residual can be made arbitrarily small for tanh neural networks with two hidden layers. Moreover, the total error can be estimated in terms of the training error, network size and number of quadrature points. The theory is illustrated with numerical experiments.

1. Introduction

Deep learning has been very successfully deployed in a variety of fields including computer vision, natural language processing, game intelligence, robotics, augmented reality and autonomous systems (LeCun et al. 2015) and references therein. In recent years, deep learning is being increasingly used in various contexts in scientific computing such as protein folding and controlled nuclear fusion.

As deep neural networks are universal function approximators, it is also natural to use them as *ansatz spaces* for the solutions of (partial) differential equations (PDEs). In fact, the literature on the use of deep learning for numerical approximation of PDEs has witnessed exponential growth in the last 2-3 years. Prominent examples for the use of deep learning in PDEs include the deep neural network approximation of high-dimensional semi-linear parabolic PDEs (E et al. 2017), linear elliptic PDEs (Schwab & Zech 2019, Kutyniok et al. 2021) and nonlinear hyperbolic PDEs (Lye et al. 2020, 2021) and references therein. More recently, DNN-inspired architectures such as DeepONets (Chen & Chen 1995, Lu et al. 2019, Lanthaler et al. 2022) and Fourier neural operators (Li et al. 2020, Kovachki et al. 2021) have been shown to even learn infinite-dimensional *operators*, associated with underlying PDEs, efficiently.

Another extremely popular avenue for the use of machine learning in numerical approximation of PDEs is in the area of *physics-informed neural networks* (PINNs). First proposed in slightly different forms in the 90s (Dissanayake & Phan-Thien 1994, Lagaris, Likas & Fotiadis 2000, Lagaris, Likas & D. 2000), PINNs were resurrected recently in (Raissi & Karniadakis 2018, Raissi et al. 2019) as a practical and computationally efficient paradigm for solving both forward and inverse problems for PDEs. Since then, there has been an explosive growth in designing and applying PINNs for a variety of applications involving PDEs. A very incomplete list

(T. De Ryck) Seminar for Applied Mathematics, D-MATH, ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland.

(S. Mishra) Seminar for Applied Mathematics, D- MATH, and ETH AI Center, ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland.

(A.D. Jagtap) Division of Applied Mathematics, Brown University, 182 George street, Providence, RI 02912, USA.

E-mail addresses: tim.deryck@sam.math.ethz.ch, ameya_jagtap@brown.edu, siddhartha.mishra@sam.math.ethz.ch.

of references includes (Raissi et al. 2018, Mao et al. 2020, Pang et al. 2019, Yang et al. 2021, Jagtap & Karniadakis 2020, Jagtap et al. 2020, Jagtap, Mao, Adams & Karniadakis 2022, Jin et al. 2021, Mishra & Molinaro 2020, 2021*a*,*b*, Bai et al. 2021, Shukla, Jagtap, Blackshire, Sparkman & Karniadakis 2021, Jagtap, Mitsotakis & Karniadakis 2022, Hu et al. 2021, Shukla, Jagtap & Karniadakis 2021) and references therein.

On the other hand and in stark contrast to the widespread applications of PINNs, there has been a pronounced scarcity of papers that rigorously justify why PINNs work. Notable exceptions include (Shin, Darbon & Karniadakis 2020) where the authors show consistency of PINNs with the underlying linear elliptic and parabolic PDE under stringent assumptions and in (Shin, Zhang & Karniadakis 2020) where similar estimates are derived for linear advection equations. In (Mishra & Molinaro 2020, 2021*a*), the authors proposed a strategy for deriving error estimates for PINNs. To describe this strategy and highlight the underlying theoretical issues, it is imperative to introduce PINNs and we do so in an informal manner here (see section 2 for the formal definitions). To this end, we consider the following very general form of an abstract PDE,

$$\begin{aligned} \mathcal{D}[u](x,t) = 0, \quad \mathcal{B}u(y,t) = \psi(y,t), \\ u(x,0) = \varphi(x), \quad \text{for } x \in D, y \in \partial D, t \in [0,T]. \end{aligned} \tag{1.1}$$

Here, $D \subset \mathbb{R}^d$ is compact and $\mathcal{D}, \mathcal{B}$ are the differential and boundary operators, $u : D \times [0,T] \to \mathbb{R}^m$ is the solution of the PDE, $\psi : \partial D \times [0,T] \to \mathbb{R}^m$ specifies the (spatial) boundary condition and $\varphi : D \to \mathbb{R}^m$ is the initial condition.

We seek deep neural networks $u_\theta : D \times [0,T] \to \mathbb{R}^m$ (see (2.5) for a definition), parameterized by $\theta \in \Theta$, constituting the weights and biases, that approximate the solution u of (1.1). The key idea behind PINNs is to consider pointwise *residuals*, defined for any sufficiently smooth function $f : D \times [0,T] \to \mathbb{R}^m$ as,

$$\begin{aligned} \mathcal{R}_i[f](x,t) = \mathcal{D}[f](x,t), \quad \mathcal{R}_s[f](y,t) = \mathcal{B}f(y,t) - \psi(y,t), \\ \mathcal{R}_t[f](x) = f(x,0) - \varphi(x), \quad x \in D, y \in \partial D, t \in [0,T] \end{aligned} \tag{1.2}$$

for $x \in D$, $y \in \partial D$, $t \in [0,T]$. Using these residuals, one measures how well a function f satisfies resp. the PDE, the boundary condition and the initial condition of (1.1). Note that for the exact solution $\mathcal{R}_i[u] = \mathcal{R}_s[u] = \mathcal{R}_t[u] = 0$.

Hence, within the PINNs algorithm, one seeks to find a neural network u_θ, for which all residuals are simultaneously minimized, e.g. by minimizing the quantity,

$$\begin{aligned} \mathcal{E}_G(\theta)^2 = \int_{D\times[0,T]} \left|\mathcal{R}_i[u_\theta](x,t)\right|^2 dxdt + \int_{\partial D\times[0,T]} \left|\mathcal{R}_s[u_\theta](x,t)\right|^2 ds(x)dt \\ + \int_D \left|\mathcal{R}_t[u_\theta](x)\right|^2 dx. \end{aligned} \tag{1.3}$$

However, the quantity $\mathcal{E}_G(\theta)$, often referred to as the *population risk* or *generalization error* (Mishra & Molinaro 2020) of the neural network u_θ involves integrals and can therefore not be directly minimized in practice. Instead, the integrals in (1.3) are approximated by a suitable numerical quadrature (see section 2.3 for details), resulting in,

$$\begin{aligned} \mathcal{E}_T^i(\theta,\mathcal{S}_i)^2 = \sum_{n=1}^{N_i} w_i^n \left|\mathcal{R}_i[u_\theta](x_i^n,t_i^n)\right|^2, \\ \mathcal{E}_T^s(\theta,\mathcal{S}_s)^2 = \sum_{n=1}^{N_s} w_s^n \left|\mathcal{R}_s[u_\theta](x_s^n,t_s^n)\right|^2, \quad \mathcal{E}_T^t(\theta,\mathcal{S}_t)^2 = \sum_{n=1}^{N_t} w_t^n \left|\mathcal{R}_t[u_\theta](x_i^t)\right|^2, \\ \mathcal{E}_T(\theta,\mathcal{S})^2 = \mathcal{E}_T^i(\theta,\mathcal{S}_i)^2 + \mathcal{E}_T^s(\theta,\mathcal{S}_s)^2 + \mathcal{E}_T^t(\theta,\mathcal{S}_t)^2, \end{aligned} \tag{1.4}$$

with quadrature points in space-time constituting data sets $\mathcal{S}_i = \{(x_i^n, t_i^n)\}_{n=1}^{N_i}$, $\mathcal{S}_s = \{(x_s^n, t_s^n)\}_{n=1}^{N_s}$ and $\mathcal{S}_t = \{x_t^n\}_{n=1}^{N_t}$, and w_q^n are suitable quadrature weights for $q = i, t, s$.

Thus, the underlying essence of PINNs is to minimize the training error $\mathcal{E}_T(\theta, \mathcal{S})^2$ over the neural network parameters θ. This procedure immediately raises the following key theoretical questions (see also (De Ryck & Mishra 2021)) starting with

[Q1.] Given a tolerance $\varepsilon > 0$, do there exist neural networks $\hat{u} = u_{\hat{\theta}}$, $\widetilde{u} = u_{\widetilde{\theta}}$, parametrized by $\hat{\theta}, \widetilde{\theta} \in \Theta$ such that the corresponding generalization $\mathcal{E}_G(\hat{\theta})$ (1.3) and training $\mathcal{E}_T(\widetilde{\theta}, \mathcal{S}_s)$(1.4) errors are small i.e., $\mathcal{E}_G(\hat{\theta}), \mathcal{E}_T(\widetilde{\theta}, \mathcal{S}_s) < \varepsilon$?

As the aim in the PINNs algorithm is to minimize the training error (and indirectly the generalization error), an affirmative answer to this question is of vital importance as it ensures that the loss (PDE residual) being minimized can be made small. However, minimizing the PDE residual does not necessarily imply that the overall error (difference between the exact solution of the PDE (1.1) and its PINN approximation) is small. This leads to the second key question,

[Q2.] Given a PINN $\hat{u}$ with small generalization error, is the corresponding *total error* $\|u - \hat{u}\|$ small, i.e., is $\|u - \hat{u}\| < \delta(\varepsilon)$, for some $\delta(\varepsilon) \sim \mathcal{O}(\varepsilon)$, for some suitable norm $\|\cdot\|$, and with u being the solution of the PDE (1.1)?

An affirmative answer to Q2 (and Q1) certifies that, *in principle*, there exists a (physics-informed) neural network, corresponding to the parameter $\hat{\theta}$, such that the PDE residual and consequently, the overall error in approximating the solution of the PDE (1.1), are small. However, in practice, we minimize the training error $\mathcal{E}_T$ (1.4) and this leads to another key question,

[Q3.] Given a small training error $\mathcal{E}_T(\theta^*)$ and a sufficiently large training set $\mathcal{S}$, is the corresponding generalization error $\mathcal{E}_G(\theta^*)$ also proportionately small?

An affirmative answer to question Q3, together with question Q2, will imply that the trained PINN u_{θ^*} is an accurate approximation of the solution u of the underlying PDE (1.1). Thus, answering the above three questions affirmatively will constitute a comprehensive theoretical investigation of PINNs and provide a rationale for their very successful empirical performance.

Given this context, we examine how far the literature has come in answering these key questions on the theory for PINNs. In (Mishra & Molinaro 2020, 2021*a*), the authors leverage the *stability* of solutions of the underlying PDE (1.1) to bound the total error in terms of the generalization error (question Q2). Similarly, they use the accuracy of quadrature rules to bound the generalization error in terms of the training error (question Q3). This approach is implemented for forward problems corresponding to a variety of PDEs such as the semi-linear and quasi-linear parabolic equations and the incompressible Euler and the Navier-Stokes equations (Mishra & Molinaro 2020), radiative transfer equations (Mishra & Molinaro 2021*b*), nonlinear dispersive PDEs such as the KdV equations (Bai et al. 2021) and for the unique continuation (data assimilation) inverse problem for many linear elliptic, parabolic and hyperbolic PDEs (Mishra & Molinaro 2021*a*). However, Q1 was not answered in these papers. Moreover, the authors imposed rather stringent assumptions on the weights and biases of the trained PINN, which may not hold in practice.

In (De Ryck & Mishra 2021), the authors answered the key questions Q1, Q2 and Q3 in the case of a large class of *linear parabolic PDEs*, namely the Kolmogorov PDEs, which include the heat equation and the Black-Scholes equation of option pricing as special examples. Thus, they provided a rigorous and comprehensive error analysis of PINNs for these PDEs. Moreover, they also showed that PINNs

overcome the *curse of dimensionality* in the context of very high-dimensional Kolmogorov equations.

The authors of (De Ryck & Mishra 2021) utilized the linearity of the underlying Kolmogorov heavily in their analysis. It is natural to ask if analogous error estimates can be shown for PINN approximations of nonlinear PDEs. This consideration sets the stage for the current paper where we carry out a thorough error analysis for PINNs approximating a prototypical nonlinear PDE and answer Q1, Q2 and Q3 affirmatively. The nonlinear PDE that we consider is the incompressible Navier-Stokes equation, which is the fundamental mathematical model governing the flow of incompressible Newtonian fluids (Temam 2001).

We are going to show the following results on the PINN approximation of the incompressible Navier-Stokes equations,

- We show that there exist neural networks that approximate the classical solutions of Navier-Stokes equations such that the PINN generalization error (1.3) and the PINN training error (1.4) can be made arbitrarily small. Moreover, we provide explicit bounds on the number of neurons as well as the weights of the network in terms of error tolerance and Sobolev norms of the underlying Navier-Stokes equations. This analysis is also extended for the XPINN approximation (Jagtap & Karniadakis 2020) of the Navier-Stokes equations, answering Q1 affirmatively for both PINNs and XPINNs.
- We bound the total error of the PINN (and XPINN) approximation of the Navier-Stokes equations in terms of the PDE residual (generalization error (1.3)). Consequently, a small PDE residual implies a small total error, answering Q2 affirmatively.
- We bound the generalization error (1.3) in terms of the training error (1.4) and the number of quadrature points using a midpoint quadrature rule. This affirmatively answers question Q3 and establishes the fact that a small training error and sufficient number of quadrature points suffice to yield a small total error for the PINN (and XPINN) approximation of the Navier-Stokes equations, under a mild assumption on the growth of the network weights (see discussion in Section 3.3).
- We present numerical experiments to illustrate our theoretical results.

The rest of our paper is organized as follows: In section 2, we collect preliminary information on the Navier-Stokes equations and neural networks and present the PINN and XPINN algorithms. The error analysis is carried out in section 3 and numerical experiments are presented in section 4.

2. Preliminaries

In this section, we collect preliminary information on concepts used in rest of the paper. We start with the form of the Navier-Stokes equations.

2.1. **The incompressible Navier-Stokes equations.**

We consider the well-known incompressible Navier-Stokes equations (Temam 2001) and references therein,

$$\begin{cases} u_t + u \cdot \nabla u + \nabla p = \nu \Delta u & \text{in } D \times [0,T], \\ \operatorname{div}(u) = 0 & \text{in } D \times [0,T], \\ u(t=0) = u_0 & \text{in } D. \end{cases} \tag{2.1}$$

Here, $u : D \times [0,T] \to \mathbb{R}^d$ is the fluid velocity, $p : D \to \mathbb{R}$ 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 (2.1) on the d-dimensional torus $D = \mathbb{T}^d = [0,1)^d$ with periodic boundary conditions.

The existence and regularity of the solution to (2.1) depends on the regularity of u_0, as is stated by the following well-known theorem (Majda et al. 2002, Theorem 3.4). Other regularity results with different boundary conditions can be found in e.g. (Temam 2001).

Theorem 2.1. *If $u_0 \in H^r(\mathbb{T}^d)$ with $r > \frac{d}{2} + 2$ 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, we prove that u is Sobolev regular i.e., that $u \in H^k(D \times [0, T])$ for some $k \in \mathbb{N}$, provided that r is large enough.

Corollary 2.2. *If $k \in \mathbb{N}$ and $u_0 \in H^r(\mathbb{T}^d)$ with $r > \frac{d}{2} + 2k$ 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 $u(t = 0) = u_0$.*

Proof. The corollary follows directly from Theorem 2.1 for $k = 1$. Therefore, let $k \geq 2$ be arbitary and assume that $r > \frac{d}{2} + 2k$ and $u_0 \in H^r(\mathbb{T}^d)$ and $\operatorname{div}(u_0) = 0$. By Theorem 2.1 there exists $T > 0$ and a classical solution u to the Navier-Stokes equation such that $u(t = 0) = u_0$ and $u \in C([0, T]; H^r(\mathbb{T}^d)) \cap C^1([0, T]; H^{r-2}(\mathbb{T}^d))$. Following (Majda et al. 2002, Section 1.8), we find that the pressure p satisfies the equation

$$-\Delta p = \operatorname{Trace}((\nabla u)^2) = \sum_{i,j} u^i_{x_j} u^j_{x_i}. \tag{2.2}$$

As $r > \frac{d}{2} + 1$, $H^{r-1}(\mathbb{T}^d)$ is a Banach algebra (see Lemma A.1), it holds that $\Delta p \in C([0, T]; H^{r-1}(\mathbb{T}^d))$ and accordingly $\nabla p \in C([0, T]; H^r(\mathbb{T}^d))$. Since $u \in C^1([0, T]; H^{r-2})$, we can take the time derivative of equation (2.2) to find that $\Delta p_t \in C([0, T]; H^{r-3})$, since the conditions for $H^{r-3}(\mathbb{T}^d)$ to be a Banach algebra are met. As a result we find that $\nabla p_t \in C([0, T]; H^{r-2})$. Taking the time derivative of the Navier-Stokes equations (2.1), we find that $u_{tt} \in C([0, T]; H^{r-4})$ and therefore $u \in C^2([0, T]; H^{r-4})$. Repeating these steps, one can prove that $u \in \cap_{\ell=0}^{k} C^\ell([0, T]; H^{r-2\ell}(\mathbb{T}^d))$. The statement of the corollary then follows from this observation since $\ell + r - 2\ell \geq k$ for all $0 \leq \ell \leq k$ if $r > \frac{d}{2} + 2k$. Similarly, one can prove that $\nabla p \in \cap_{\ell=0}^{k-1} C^\ell([0, T]; H^{r-2\ell}(\mathbb{T}^d))$. □

2.2. **Neural networks.** As our objective is to approximate the solution of the incompressible Navier-Stokes equations (2.1) with neural networks, here we formally introduce our definition of a neural network and the related terminology.

Definition 2.3. *Let $R \in (0, \infty]$, $L, W \in \mathbb{N}$ and $l_0, \ldots, l_L \in \mathbb{N}$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a twice differentiable* activation function *and define*

$$\Theta = \Theta_{L,W,R} := \bigcup_{L' \in \mathbb{N}, L' \leq L} \bigcup_{l_0, \ldots, l_L \in \{1, \ldots, W\}} \bigtimes_{k=1}^{L'} \left([-R, R]^{l_k \times l_{k-1}} \times [-R, R]^{l_k} \right). \tag{2.3}$$

For $\theta \in \Theta_{L,W,R}$, we define $\theta_k := (\mathcal{W}_k, b_k)$ and $\mathcal{A}_k : \mathbb{R}^{l_{k-1}} \to \mathbb{R}^{l_k} : x \mapsto \mathcal{W}_k x + b_k$ for $1 \leq k \leq L$ and and we define $f^\theta_k : \mathbb{R}^{l_{k-1}} \to \mathbb{R}^{l_k}$ by

$$f^\theta_k(z) = \begin{cases} \mathcal{A}^\theta_L(z) & k = L, \\ (\sigma \circ \mathcal{A}^\theta_k)(z) & 1 \leq k < L. \end{cases} \tag{2.4}$$

We denote by $u_\theta : \mathbb{R}^{l_0} \to \mathbb{R}^{l_L}$ the function that satisfies for all $z \in \mathbb{R}^{l_0}$ that

$$u_\theta(z) = \left(f^\theta_L \circ f^\theta_{L-1} \circ \cdots \circ f^\theta_1 \right)(z), \tag{2.5}$$

where in the setting of approximating the Navier-Stokes equation (2.1) we set $l_0 = d + 1$ and $z = (x, t)$. We refer to u_θ as the realization of the neural network

associated to the parameter θ with L layers and widths $(l_0, l_1, \dots, l_L)$. We refer to the first $L-1$ layers as hidden layers. *For $1 \leq k \leq L$, we say that layer k has width l_k and we refer to $\mathcal{W}_k$ and b_k as the* weights and biases *corresponding to layer k. The width of u_θ is defined as $\max(l_0, \dots, l_L)$. If $L = 2$, we say that u_θ is a* shallow neural network*; if $L \geq 3$, we say that u_θ is a* deep neural network.

2.3. **Quadrature rules.** In the following sections, we will need to approximate integrals of functions. For this reason, we introduce some notation and recall well-known results on numerical quadrature rules.

Given $\Lambda \subset \mathbb{R}^d$ and $f \in L^1(\Lambda)$, we will be interested in approximating $\int_\Lambda f(y)dy$, with dy denoting the d-dimensional Lebesgue measure. A numerical quadrature rule provides such an approximation by choosing some quadrature points $y_m \in \Lambda$ for $1 \leq m \leq M$, and quadrature weights $w_m > 0$ for $1 \leq m \leq M$, and considers the approximation

$$\frac{1}{M} \sum_{m=1}^{M} w_m f(y_m) \approx \int_\Lambda f(y)dy. \tag{2.6}$$

The accuracy of this approximation depends on the chosen quadrature rule, the number of quadrature points M and the regularity of f. Whereas in very high dimensions, random training points or low-discrepancy training points (Mishra & Rusch 2021) are needed, the relatively low-dimensional setting of the Navier-Stokes equations i.e., $d \leq 4$, allows the use of standard deterministic numerical quadrature points. In order to obtain explicit rates, we will focus on the *midpoint rule*, but our analysis will also hold for general deterministic numerical quadrature rules.

We briefly recall the midpoint rule. For $N \in \mathbb{N}$, we partition Λ into $M \sim N^d$ cubes of edge length $1/N$ and we denote by $\{y_m\}_{m=1}^M$ the midpoints of these cubes. The formula and accuracy of the midpoint rule $\mathcal{Q}_M^\Lambda$ are then given by,

$$\mathcal{Q}_M^\Lambda[f] := \frac{1}{M} \sum_{m=1}^{M} f(y_m), \qquad \left| \int_\Lambda f(y)dy - \mathcal{Q}_M^\Lambda[f] \right| \leq C_f M^{-2/d}, \tag{2.7}$$

where $C_f \lesssim \|f\|_{C^2}$.

2.4. **Physics-informed neural networks (PINNs).** We seek deep neural networks $u_\theta : D \times [0,T] \to \mathbb{R}^d$ and $p_\theta : D \times [0,T] \to \mathbb{R}$ (cf. Definition 2.3), parameterized by $\theta \in \Theta$, constituting the weights and biases, that approximate the solution u of (2.1). To this end, the key idea behind PINNs is to consider pointwise *residuals*, defined in the setting of the Navier-Stokes equations (2.1) for any sufficiently smooth $v : D \times [0,T] \to \mathbb{R}^d$ and $q : D \times [0,T] \to \mathbb{R}$ as,

$$\begin{aligned} \mathcal{R}_{\text{PDE}}[(v,q)](x,t) &= (v_t + v \cdot \nabla v + \nabla q - \nu \Delta v)(x,t), \\ \mathcal{R}_{\text{div}}[v](x,t) &= \text{div}\,(v)\,(x,t) \\ \mathcal{R}_s[v](y,t) &= \mathcal{B}v(y,t) - \psi(y,t), \\ \mathcal{R}_t[v](x) &= v(x,0) - \varphi(x) \end{aligned} \tag{2.8}$$

for $x \in D$, $y \in \partial D$, $t \in [0,T]$. In the above, $\mathcal{B}$ is the boundary operator, $\psi : \partial D \times [0,T] \to \mathbb{R}^d$ specifies the (spatial) boundary condition and $\varphi : D \to \mathbb{R}^d$ is the initial condition. Using these residuals, one measures how well a function f satisfies resp. the PDE, the boundary condition and the initial condition of (2.1). Note that for the exact solution to the Navier-Stokes equations (2.1) it holds that $\mathcal{R}_{\text{PDE}}[(u,p)] = \mathcal{R}_{\text{div}}[u] = \mathcal{R}_s[u] = \mathcal{R}_t[u] = 0$.

Hence, within the PINNs algorithm, one seeks to find a neural network (u_θ, p_θ), for which all residuals are simultaneously minimized, e.g. by minimizing the quantity,

$$\begin{aligned}\mathcal{E}_G(\theta)^2 = &\int_{D\times[0,T]} \left\|\mathcal{R}_{\mathrm{PDE}}[(u_\theta,p_\theta)](x,t)\right\|_{\mathbb{R}^d}^2 dxdt + \int_{D\times[0,T]} \left|\mathcal{R}_{\mathrm{div}}[u_\theta](x,t)\right|^2 dxdt \\ &+ \int_{\partial D\times[0,T]} \left\|\mathcal{R}_s[u_\theta](x,t)\right\|_{\mathbb{R}^d}^2 ds(x)dt + \int_D \left\|\mathcal{R}_t[u_\theta](x)\right\|_{\mathbb{R}^d}^2 dx.\end{aligned} \tag{2.9}$$

The different terms of (2.9) are often rescaled using some weights. For simplicity, we set all these weights to one. The quantity $\mathcal{E}_G(\theta)$, often referred to as the *population risk* or *generalization error* of the neural network u_θ, involves integrals and can therefore not be directly minimized in practice. Instead, the integrals in (2.9) are approximated by a numerical quadrature, as introduced in Section 2.3. As a result, we define the (squared) training loss for PINNs $\theta \mapsto \mathcal{E}_T(\theta,\mathcal{S})^2$ as follows,

$$\begin{aligned}\mathcal{E}_T(\theta,\mathcal{S})^2 &= \mathcal{E}_T^{\mathrm{PDE}}(\theta,\mathcal{S}_{\mathrm{int}})^2 + \mathcal{E}_T^{\mathrm{div}}(\theta,\mathcal{S}_{\mathrm{int}})^2 + \mathcal{E}_T^{s}(\theta,\mathcal{S}_s)^2 + \mathcal{E}_T^{t}(\theta,\mathcal{S}_t)^2 \\ &= \sum_{n=1}^{N_{\mathrm{int}}} w_{\mathrm{int}}^n \left\|\mathcal{R}_{\mathrm{PDE}}[(u_\theta,p_\theta)](t_{\mathrm{int}}^n,x_{\mathrm{int}}^n)\right\|_{\mathbb{R}^d}^2 + \sum_{n=1}^{N_{\mathrm{int}}} w_{\mathrm{int}}^n \left|\mathcal{R}_{\mathrm{div}}[u_\theta](t_{\mathrm{int}}^n,x_{\mathrm{int}}^n)\right|^2 \\ &\quad + \sum_{n=1}^{N_s} w_s^n \left\|\mathcal{R}_s[u_\theta](t_s^n,x_s^n)\right\|_{\mathbb{R}^d}^2 + \sum_{n=1}^{N_t} w_t^n \left\|\mathcal{R}_t[u_\theta](x_t^n)\right\|_{\mathbb{R}^d}^2,\end{aligned} \tag{2.10}$$

where the training data set $\mathcal{S} = (\mathcal{S}_{\mathrm{int}}, \mathcal{S}_s, \mathcal{S}_t)$ is chosen as quadrature points with respect to the relevant domain (resp. $D\times[0,T]$, $\partial D\times[0,T]$ and D) and where the w_*^n are corresponding quadrature weights.

A *trained PINN* $u^* = u_{\theta^*}$ is then defined as a (local) minimum of the optimization problem,

$$\theta^*(\mathcal{S}) = \arg\min_{\theta\in\Theta} \mathcal{E}_T(\theta,\mathcal{S})^2, \tag{2.11}$$

with loss function (2.10) (possibly with additional data and weight regularization terms), found by a (stochastic) gradient descent algorithm such as ADAM or L-BFGS.

2.5. **Extended physics-informed neural networks (XPINNs).** In many applications, it happens that the computational domain has a very complicated shape or that the PDE solution shows localized features. In such cases, it is beneficial to decompose the computational domain into non-overlapping regions and deploy different neural networks to approximate the PDE solution in different sub-regions. This idea was first presented in (Jagtap & Karniadakis 2020), where the authors proposed to decompose the domain in $\mathcal{N}$ closed subdomains with non-overlapping interior and deploy PINNs u_{θ_q} to approximate the exact solution u in each of those subdomains Ω_q. Patching together the PINNs for all the subnetworks yields the final approximation u_θ, termed *extended physics-informed neural network (XPINN)*, defined as,

$$u_\theta(z) = \sum_{q=1}^{\mathcal{N}} \chi_q(z) u_{\theta_q}(z), \tag{2.12}$$

for $z=(x,t)$ and where the weight function χ_q is given by,

$$\chi_q(z) = \begin{cases} 0 & z\notin\Omega_q, \\ \frac{1}{\#\{n\,:\,z\in\Omega_n\}} & z\in\Omega_q, \end{cases} \tag{2.13}$$

where $\#\{n \,:\, z \in \Omega_n\}$ represents the number of subdomains z belongs to. Hence $\sum_q \chi_q(z) = 1$ for all z. One can define neural networks p_θ and p_{θ_q} in an analogous way. It is clear that mimimizing the standard PINN loss (2.10) for an XPINN (2.12) would not be a suitable approach. It is necessary that additional terms in the form of *interface conditions* should be added to the loss function. For this purpose, we define for every q the following residuals in addition to the standard PINN residuals (2.8),

$$\mathcal{R}_u[f](y,t) = f(y,t) - u_\theta(y,t), \tag{2.14}$$

where $y \in \partial\Omega_q \setminus \partial D$ and $t \in [0,T]$. The squared generalization error of an XPINN u_θ is then given by

$$\begin{aligned}
\mathcal{E}_G(\theta)^2 = &\int_{D\times[0,T]} \left\|\mathcal{R}_{\mathrm{PDE}}[(u_\theta,p_\theta)](x,t)\right\|_{\mathbb{R}^d}^2 dxdt + \int_{D\times[0,T]} \left|\mathcal{R}_{\mathrm{div}}[u_\theta](x,t)\right|^2 dxdt \\
&+ \int_{\partial D\times[0,T]} \left\|\mathcal{R}_s[u_\theta](x,t)\right\|_{\mathbb{R}^d}^2 ds(x)dt + \int_D \left\|\mathcal{R}_t[u_\theta](x)\right\|_{\mathbb{R}^d}^2 dx \\
&+ \sum_{q=1}^{\mathcal{N}} \int_{(\partial\Omega_q\setminus\partial D)\times[0,T]} \left\|\mathcal{R}_u[u_{\theta_q}](x,t)\right\|_{\mathbb{R}^d}^2 ds(x)dt \\
&+ \sum_{q=1}^{\mathcal{N}} \int_{(\partial\Omega_q\setminus\partial D)\times[0,T]} \left\|\mathcal{R}_{\mathrm{PDE}}[(u_{\theta_q},p_{\theta_q})](x,t) - \mathcal{R}_{\mathrm{PDE}}[u_\theta](x,t)\right\|_{\mathbb{R}^d}^2 ds(x)dt.
\end{aligned} \tag{2.15}$$

The interface conditions on the two last lines of (2.15) enforce the continuity and possibly even higher regularity of the XPINN at the interface of neighbouring subdomains.

The XPINN training loss can then be defined by replacing the integrals in (2.15) by numerical quadratures, in the same way the standard PINN training loss (2.10) was derived from (2.9).

3. Error analysis

In this section, we will obtain rigorous on the PINN and XPINN approximations of the solutions of the incompressible Navier-Stokes equations. We start with bounds on PINN residuals below.

3.1. Bound on the PINN residuals.

From the definition of the interior PINN residuals (2.8), it is clear that if we can find a neural network $\hat{u}$ such that $\|u-\hat{u}\|_{H^2(D\times[0,T])}$ is small, then the interior PINN residual will be small as well. The approximation (in Sobolev norm) of Sobolev regular functions by tanh neural networks is discussed in Appendix B. The main ingredients are a piecewise polynomial approximation, the existence of which is guaranteed by the Bramble-Hilbert lemma, and the ability of tanh neural networks to efficiently approximate polynomials, the multiplication operator and an approximate partition of unity. The main result of Appendix B is Theorem B.7, which is a variant of (De Ryck et al. 2021, Theorem 5.1). It proves that a tanh neural network with two hidden layers suffices to make $\|u-\hat{u}\|_{H^2(D\times[0,T])}$ arbitrarily small and provides explicit bounds on the needed network width. Using this theorem, we can prove the following upper bound on the PINN residual.

Theorem 3.1. *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 $r > \frac{d}{2} + 2k$ and $\operatorname{div}(u_0) = 0$. It holds that:*

- *there exist $T > 0$ and a classical solution u to the Navier-Stokes equations such that $u \in H^k(\Omega)$, $\nabla p \in H^{k-1}(\Omega)$, $\Omega = \mathbb{T}^d \times [0,T]$, and $u(t=0) = u_0$,*

- *for every $N > 5$, there exist tanh neural networks $\hat{u}_j$, $1 \le j \le d$, and $\widehat{p}$, each with two hidden layers, of widths $3\left\lceil \frac{k+n-2}{2} \right\rceil \binom{d+k-1}{d} + \lceil TN \rceil + dN$ and $3\left\lceil \frac{d+n}{2} \right\rceil \binom{2d+1}{d} \lceil TN \rceil N^d$, such that for every $1 \le j \le d$,*

$$\left\|(\hat{u}_j)_t + \hat{u} \cdot \nabla \hat{u}_j + (\nabla \widehat{p})_j - \nu \Delta \hat{u}_j\right\|_{L^2(\Omega)} \le C_1 \ln^2(\beta N) N^{-k+2}, \tag{3.1}$$

$$\left\|\operatorname{div}(\hat{u})\right\|_{L^2(\Omega)} \le C_2 \ln(\beta N) N^{-k+1}, \tag{3.2}$$

$$\left\|(u_0)_j - \hat{u}_j(t=0)\right\|_{L^2(\mathbb{T}^d)} \le C_3 \ln(\beta N) N^{-k+1}, \tag{3.3}$$

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 $\mathcal{O}(N^\gamma \ln(N))$ where $\gamma = \max\{1, d(2+k^2+d)/n\}$.

Proof. Let $N > 5$. By Corollary 2.2 it holds that $u \in H^k(\mathbb{T}^d \times [0,T])$ and $\nabla p \in H^{k-1}(\mathbb{T}^d \times [0,T])$, hence also $p \in H^{k-1}(\mathbb{T}^d \times [0,T])$. As a result of Theorem B.7, there then exists for every $1 \le j \le d$ a tanh neural network $\hat{u}_j := \hat{u}_j^N$ with two hidden layers and widths $3\left\lceil \frac{k+n-2}{2} \right\rceil \binom{d+k-1}{d} + \lceil TN \rceil + dN$ and $3\left\lceil \frac{d+n}{2} \right\rceil \binom{2d+1}{d} \lceil TN \rceil N^d$ such that for every $0 \le \ell \le 2$,

$$\left\|u_j - \hat{u}_j\right\|_{H^\ell(\Omega)} \le C_{\ell,k,d+1,u_j} \lambda_\ell(N) N^{-k+\ell}, \tag{3.4}$$

where $\lambda_\ell(N) = 2^{\ell+1} 3^d (1+\delta) \ln^\ell\left(\beta_{\ell,d+1,u_j} N^{d+k+2}\right)$, $\delta = \frac{1}{100}$, and the definition of the other constants can be found in Theorem B.7. The weights can be bounded by $\mathcal{O}(N^\gamma \ln(N))$ where $\gamma = \max\{1, d(2+k^2+d)/n\}$. We write $\hat{u} = (\hat{u}_1, \dots, \hat{u}_d)$. Moreover, by Theorem B.7, there also exists a tanh neural network $\widehat{p} := \widehat{p}^N$ with two hidden layers and the same widths as before such that

$$\left\|(\nabla p)_j - (\nabla \widehat{p})_j\right\|_{L^2(\Omega)} \le \|p - \widehat{p}\|_{H^1(\Omega)} \le C_{1,k-1,d+1,p} \lambda_1(N) N^{-k+2}. \tag{3.5}$$

It is now straightforward to bound the PINN residual.

$$\left\|(u_j)_t - (\hat{u}_j)_t\right\|_{L^2(\Omega)} \le \left|u_j - \hat{u}_j\right|_{H^1(\Omega)}. \tag{3.6}$$

By the Sobolev embedding theorem (Lemma A.2) it follows from $u \in C^1([0,T], H^{r-2}(\mathbb{T}^d))$ that $u \in C^1(\Omega)$, and hence

$$\begin{aligned}\left\|u \cdot \nabla u_j - \hat{u} \cdot \nabla \hat{u}_j\right\|_{L^2(\Omega)} &\le \left\|u \cdot \nabla u_j - \hat{u} \cdot \nabla u_j\right\|_{L^2(\Omega)} + \left\|\hat{u} \cdot \nabla u_j - \hat{u} \cdot \nabla \hat{u}_j\right\|_{L^2(\Omega)} \\ &\le \sqrt{d}\left\|u_j\right\|_{C^1} \max_i \|u_i - \hat{u}_i\|_{L^2(\Omega)} + \sqrt{d} \max_i \|\hat{u}_i\|_{C^0} \left|u_j - \hat{u}_j\right|_{H^1(\Omega)}\end{aligned} \tag{3.7}$$

and finally also

$$\left\|\Delta u_j - \Delta \hat{u}_j\right\|_{L^2(\Omega)} \le \sqrt{d}\left\|u_j - \hat{u}_j\right\|_{H^2(\Omega)} \tag{3.8}$$

$$\left\|\operatorname{div}(u) - \operatorname{div}(\hat{u})\right\|_{L^2(\Omega)} \le \sqrt{d} \max_i |u_i - \hat{u}_i|_{H^1(\Omega)}. \tag{3.9}$$

Hence, we find that for $1 \le j \le d$,

$$\begin{aligned}\left\|(\hat{u}_j)_t + \hat{u} \cdot \nabla \hat{u}_j + (\nabla \widehat{p})_j - \nu \Delta \hat{u}_j\right\|_{L^2(\Omega)} &\le C_{1,k-1,d+1,p} \lambda_1(N) N^{-k+2} \\ &+ C_{1,k,d+1,u_j} \lambda_1(N)(1 + \sqrt{d} \max_i \|\hat{u}_i\|_{C^0}) N^{-k+1} \\ &+ \sqrt{d} \lambda_0(N) \left\|u_j\right\|_{C^1} C_{0,k,d+1,u_j} N^{-k} + \nu \sqrt{d} C_{2,k,d+1,u_j} \lambda_2(N) N^{-k+2}\end{aligned} \tag{3.10}$$

and also

$$\left\|\operatorname{div}(\hat{u})\right\|_{L^2(\Omega)} \le \sqrt{d} C_{1,k,d+1,u_1} \lambda_1(N) N^{-k+1}. \tag{3.11}$$

Finally, we find from the multiplicative trace theorem (Lemma A.3) that

$$\begin{aligned}\left\|(u_0)_j - \hat{u}_j(t=0)\right\|_{L^2(\mathbb{T}^d)} &\leq \left\|u_j - \hat{u}_j\right\|_{L^2(\partial\Omega)} \\ &\leq \sqrt{\frac{2\max\{2h_\Omega, d+1\}}{\rho_\Omega}}\left\|u_j - \hat{u}_j\right\|_{H^1(\Omega)} \\ &\leq \sqrt{\frac{2\max\{2h_\Omega, d+1\}}{\rho_\Omega}} C_{1,k,d+1,u_1}\lambda_1(N)N^{-k+1},\end{aligned} \tag{3.12}$$

where h_Ω is the diameter of Ω and ρ_Ω is the radius of the largest $(d+1)$-dimensional ball that can be inscribed into Ω. This concludes the proof. □

Thus, the bounds (3.1), (3.2) and (3.3) clearly show that by choosing N sufficiently large, we can make the PINN residuals (2.8) and consequently the generalization error arbitrarily small. This affirmatively answers Q1 in the introduction.

To further illustrate the bounds of Theorem 3.1, we look for a suitable neural network such that (3.1), (3.2) and (3.3) are all smaller than 1%. Using the notation of the proof, we set $n = 2$, $T = 1$ and $\nu = \frac{1}{1000}$ and make the simplification that $\|u\|_{H^k(\Omega)} = 1$ for all k. The results are shown for $d = 2, 3$ in Figure 1 for varying regularity r of the initial condition i.e., $u_0 \in H^r(\mathbb{T}^d)$. In particular, for $d = 2$ we find that the minimal network size of every sub-network $\hat{u}_j$ is $54 \cdot 10^3$ neurons. Although it is certainly possible to reach this level of accuracy with smaller networks, see e.g. (Jin et al. 2021), the networks that follow from Theorem 3.1 are not unreasonably large, even in three space dimensions.

Remark 3.2. *One can easily prove that the XPINN loss of the network constructed in the proof of Theorem 3.1 will be small as well.*

Remark 3.3. *In Theorem 3.1 the parameter $n \geq 2$ can be chosen arbitrarily and is independent of u. It controls the trade-off between the network width (which grows linearly in n) and the network weights (which grow as $\mathcal{O}(N^\gamma \ln(N))$ where $\gamma = \max\{1, d(2 + k^2 + d)/n\}$). Note that it only makes sense to choose $2 \leq n \leq d(2+k^2+d)$ as the bound on the weights can not be made smaller than $\mathcal{O}(N \ln(N))$. Hence, this bound proves the existence of a neural network architecture for which the weights will only grow very moderately with increasing accuracy.*

3.2. **Bound on the total error.** Next, we will show that neural networks for which the (X)PINN residuals are small, will provide a good L^2-approximation of the true solution $u : \Omega = D \times [0, T] \to \mathbb{R}^d$, $p : \Omega \to \mathbb{R}$ of the Navier-Stokes equation (2.1) on the torus $D = \mathbb{T}^d = [0, 1)^d$ with periodic boundary conditions. Our analysis can be readily extended to other boundary conditions, such as no-slip boundary condition i.e., $u(x, t) = 0$ for all $(x, t) \in \partial D \times [0, T]$, and no-penetration boundary conditions i.e., $u(x, t) \cdot \hat{n}_D = 0$ for all $(x, t) \in \partial D \times [0, T]$.

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

$$\begin{aligned}&\mathcal{R}_{\text{PDE}} = \partial_t u_\theta + (u_\theta \cdot \nabla)u_\theta + \nabla p_\theta - \nu\Delta u_\theta, \qquad \mathcal{R}_{\text{div}} = \operatorname{div}(u_\theta), \\ &\mathcal{R}_{s,u}(x) = u_\theta(x) - u_\theta(x+1), \qquad \mathcal{R}_{s,p}(x) = p_\theta(x) - p_\theta(x+1), \\ &\mathcal{R}_{s,\nabla u}(x) = \nabla u_\theta(x) - \nabla u_\theta(x+1), \qquad \mathcal{R}_s = (\mathcal{R}_{s,u}, \mathcal{R}_{s,p}, \mathcal{R}_{s,\nabla u}), \\ &\mathcal{R}_t = u_\theta(t=0) - u(t=0),\end{aligned} \tag{3.13}$$

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

We will also extend our analysis to the XPINN framework for two subdomains (the extension to more subdomains is straightforward). For this reason, we assume

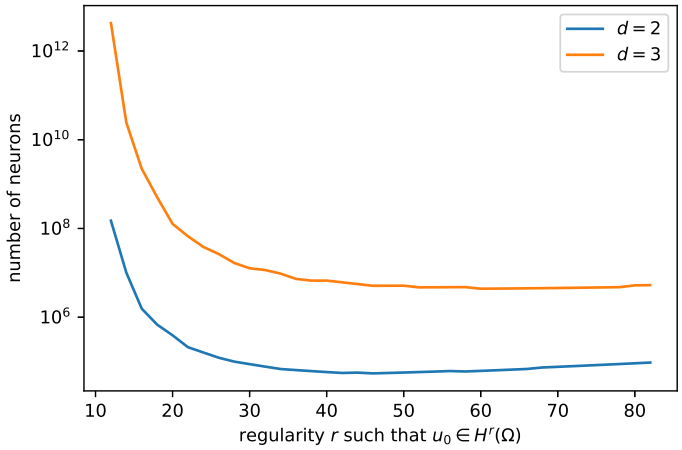

FIGURE 1. Needed neural network size according to Theorem 3.1 such that (3.1), (3.2) and (3.3) are all smaller than 1% for varying regularity r of the initial condition i.e., $u_0 \in H^r(\mathbb{T}^d)$.

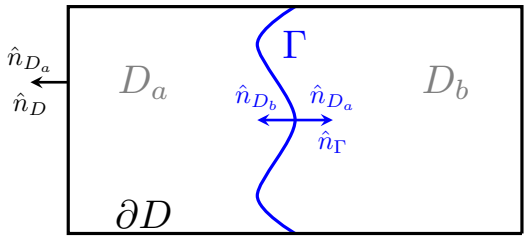

FIGURE 2. Visualization of the set-up for the XPINN framework with two subdomains.

that $D = D_a \cup D_b$, where D_a and D_b are closed with non-overlapping interior $\mathring{D}_a \cap \mathring{D}_b = \emptyset$ and common boundary $\Gamma = D_a \cap D_b$, which we assume to be suitably smooth. We define $\hat{n}_\Gamma$ to point outwards of D_a. Figure 2 provides a visualization of this set-up.

Following (2.12), the XPINN solution is then defined as

$$u_\theta = \begin{cases} u_\theta^a & \text{in } D_a \setminus \Gamma, \\ u_\theta^b & \text{in } D_b \setminus \Gamma, \\ \frac{1}{2}(u_\theta^a + u_\theta^b) & \text{in } \Gamma, \end{cases} \qquad p_\theta = \begin{cases} p_\theta^a & \text{in } D_a \setminus \Gamma, \\ p_\theta^b & \text{in } D_b \setminus \Gamma, \\ \frac{1}{2}(p_\theta^a + p_\theta^b) & \text{in } \Gamma, \end{cases} \tag{3.14}$$

where $u_\theta^a, u_\theta^b, p_\theta^a, p_\theta^b$ are neural networks. In addition to the PINN-related residuals, the following XPINN-related residuals need to be defined,

$$\begin{aligned} \mathcal{R}_u &= \max_j \left| (u_\theta^a)_j - (u_\theta^b)_j \right|, \qquad \mathcal{R}_{\nabla u} = \max_{i,j} \left| \partial_i (u_\theta^a)_j - \partial_i (u_\theta^b)_j \right|, \\ \mathcal{R}_p &= \max_{i,j} \left| p_\theta^a - p_\theta^b \right|. \end{aligned} \tag{3.15}$$

The following theorem then bounds the L^2-error of the (X)PINN in terms of the residuals defined above, see also (Mishra & Molinaro 2020, Biswas et al. 2020) for versions of the stability argument used below. We write $|\partial D|$, $|\Gamma|$ for the $(d -$

1)-dimensional Lebesgue measure of ∂D and Γ, respectively, and $|D|$ for the d-dimensional Lebesgue measure of D.

Theorem 3.4. *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 (2.1). Let (u_θ, p_θ) be a PINN/XPINN with parameters θ, then the resulting L^2-error is bounded as follows,*

$$\int_\Omega \big\|u(x,t) - u_\theta(x,t)\big\|_2^2 dxdt \leq \mathcal{C} T \exp\Big(T(2d^2 \|\nabla u\|_{L^\infty(\Omega)} + 1)\Big), \tag{3.16}$$

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

$$\begin{aligned}\mathcal{C} = \|\mathcal{R}_t\|^2_{L^2(D)} + \|\mathcal{R}_{\text{PDE}}\|^2_{L^2(\Omega)} + C_1\sqrt{T}\Bigg[\sqrt{|D|}\|\mathcal{R}_{\text{div}}\|_{L^2(\Omega)} + (1+\nu)\sqrt{|\partial D|}\|\mathcal{R}_s\|_{L^2(\partial D\times[0,T])} \\ + \sqrt{|\Gamma|}\Big((1+\nu)\|\mathcal{R}_u\|_{L^2(\Gamma\times[0,T])} + \nu\|\mathcal{R}_{\nabla u}\|_{L^2(\Gamma\times[0,T])} + \big\|\mathcal{R}_p\big\|_{L^2(\Gamma\times[0,T])}\Big)\Bigg],\end{aligned} \tag{3.17}$$

and $C_1 = C_1\big(\|u\|_{C^1}, \|\hat{u}\|_{C^1}, \|p\|_{C^0}, \|\hat{p}\|_{C^0}\big) < \infty$. For PINNs, it holds that $\mathcal{R}_u = \mathcal{R}_{\nabla u} = \mathcal{R}_p = 0$.

Proof. Let $\hat{u} = u_\theta - u$ and $\hat{p} = p_\theta - p$ denote the difference between the solution of the Navier-Stokes equations and a PINN with parameter vector θ. Using the Navier-Stokes equations (2.1) and the definitions of the different residuals, we find after a straightforward calculation that,

$$\begin{aligned}&\mathcal{R}_{\text{PDE}} = \hat{u}_t + (\hat{u}\cdot\nabla)\hat{u} + (u\cdot\nabla)\hat{u} + (\hat{u}\cdot\nabla)u + \nabla\hat{p} - \nu\Delta\hat{u},\\
&\mathcal{R}_{\text{div}} = \operatorname{div}(\hat{u}), \qquad \mathcal{R}_s(x) = u_\theta(x) - u_\theta(x+1), \qquad \mathcal{R}_t = \hat{u}(t=0),\\
&\mathcal{R}_u = \max_j \Big|(u_\theta^a)_j - (u_\theta^b)_j\Big|, \quad \mathcal{R}_{\nabla u} = \max_{i,j}\Big|\partial_i(u_\theta^a)_j - \partial_i(u_\theta^b)_j\Big|, \quad \mathcal{R}_p = \max_{i,j}\Big|p_\theta^a - p_\theta^b\Big|.\end{aligned} \tag{3.18}$$

Next, we recall the following vector equalities,

$$\hat{u}\cdot\hat{u}_t = \frac{1}{2}\partial_t\|u\|_2^2, \quad \hat{u}\cdot((\hat{u}\cdot\nabla)\hat{u}) = \frac{1}{2}(\hat{u}\cdot\nabla)\|\hat{u}\|_2^2, \quad \hat{u}\cdot((u\cdot\nabla)\hat{u}) = \frac{1}{2}(u\cdot\nabla)\|\hat{u}\|_2^2. \tag{3.19}$$

We take the inner product of the first equation in (2.1) and $\hat{u}$, and use the previous vector inequalities to obtain,

$$\frac{1}{2}\partial_t\|\hat{u}\|_2^2 + \frac{1}{2}(\hat{u}\cdot\nabla)\|\hat{u}\|_2^2 + \frac{1}{2}(u\cdot\nabla)\|\hat{u}\|_2^2 + \hat{u}\cdot((\hat{u}\cdot\nabla)u) + (\hat{u}\cdot\nabla)\hat{p} - \nu\hat{u}\cdot\Delta\hat{u} = \hat{u}\cdot\mathcal{R}_{\text{PDE}} \tag{3.20}$$

Now let $\Lambda \subset D$ be such that $\partial\Lambda$ is piecewise smooth with outward normal vector $\hat{n}_\Lambda$. Denote by T_Λ the corresponding trace operator. Integrating (3.20) over Λ and integrating by parts yields,

$$\begin{aligned}\frac{d}{dt}\int_\Lambda \|\hat{u}\|_2^2 dx = &\int_\Lambda \mathcal{R}_{\text{div}}(\|\hat{u}\|_2^2 + 2\hat{p})dx - \int_{\partial\Lambda} T_\Lambda(\hat{u})\cdot\hat{n}_\Lambda(\|\hat{u}\|_2^2 + 2\hat{p})ds(x)\\
&- 2\int_\Lambda \hat{u}\cdot((\hat{u}\cdot\nabla)u)dx - 2\nu\sum_{j=1}^d\int_\Lambda \big\|\nabla\hat{u}_j\big\|_2^2 dx\\
&+ 2\nu\sum_{j=1}^d\int_{\partial\Lambda} T_\Lambda(\hat{u}_j)(\hat{n}_\Lambda\cdot T_\Lambda(\nabla\hat{u}_j))ds(x) + 2\int_\Lambda \hat{u}\cdot\mathcal{R}_{\text{PDE}}\,dx.\end{aligned} \tag{3.21}$$

The use of the trace operator T_Λ is necessary since the trace of $\hat{u}$ on $\partial\Lambda$ might not agree with the actual definition of $\hat{u}$ as in (3.14). We then find

$$\begin{aligned}
&\sum_{i=a}^{b}\int_{\partial D_i} T_{D_i}(\hat{u}^i)\cdot\hat{n}_{D_i}(\left\|\hat{u}^i\right\|_2^2+2\hat{p}^i)ds(x)-\int_{\partial D}\hat{u}\cdot\hat{n}_D(\|\hat{u}\|_2^2+2\hat{p})ds(x)\\
&=\int_\Gamma \hat{u}^a\cdot\hat{n}_\Gamma(\|\hat{u}^a\|_2^2+2\hat{p}^a)ds(x)-\int_\Gamma \hat{u}^b\cdot\hat{n}_\Gamma(\left\|\hat{u}^b\right\|_2^2+2\hat{p}^b)ds(x)\\
&=\int_\Gamma (u_\theta^a-u_\theta^b)\cdot\hat{n}_\Gamma(\|\hat{u}^a\|_2^2+2\hat{p}^a)ds(x)+\int_\Gamma \hat{u}^b\cdot\hat{n}_\Gamma(\|\hat{u}^a\|_2^2-\left\|\hat{u}^b\right\|_2^2+2(p_\theta^a-p_\theta^b))ds(x)
\end{aligned}\tag{3.22}$$

And similarly,

$$\begin{aligned}
&\sum_{i=a}^{b}\int_{\partial D_i} T_{D_i}(\hat{u}_j^i)(\hat{n}_{D_i}\cdot T_{D_i}(\nabla\hat{u}_j^i))ds(x)-\int_{\partial D}\hat{u}_j(\hat{n}_D\cdot\nabla\hat{u}_j)ds(x)\\
&=\int_\Gamma \hat{u}_j^a(\hat{n}_\Lambda\cdot\nabla\hat{u}_j^a)ds(x)-\int_\Gamma \hat{u}_j^b(\hat{n}_\Lambda\cdot\nabla\hat{u}_j^b)ds(x)\\
&=\int_\Gamma ((u_\theta^a)_j-(u_\theta^b)_j)(\hat{n}_\Lambda\cdot\nabla\hat{u}_j^a)ds(x)-\int_\Gamma \hat{u}_j^b(\hat{n}_\Lambda\cdot\nabla((u_\theta^a)_j-(u_\theta^b)_j))ds(x)
\end{aligned}\tag{3.23}$$

Moreover, we calculate that for a constant $C_1\big(\|u\|_{C^1},\|\hat{u}\|_{C^1},\|p\|_{C^0},\|\hat{p}\|_{C^0}\big)$ it holds that,

$$\begin{aligned}
&-\int_D \hat{u}\cdot((\hat{u}\cdot\nabla)u)dx\le d^2\|\nabla u\|_{L^\infty(\Omega)}\int_D\|\hat{u}\|_2^2dx,\\
&\left|\int_{\partial D}\hat{u}\cdot\hat{n}_D(\|\hat{u}\|_2^2+2\hat{p})ds(x)\right|\le C_1\left(\left\|\mathcal{R}_{s,u}\right\|_{L^1(\partial D)}+\left\|\mathcal{R}_{s,p}\right\|_{L^1(\partial D)}\right),\\
&\int_{\partial D}\hat{u}_j(\hat{n}_D\cdot\nabla\hat{u}_j)ds(x)\le C_1\left(\left\|\mathcal{R}_{s,u}\right\|_{L^1(\partial D)}+\left\|\mathcal{R}_{s,\nabla u}\right\|_{L^1(\partial D)}\right),
\end{aligned}\tag{3.24}$$

where $\Omega=D\times[0,T]$. Now, summing (3.20) over the different $\Lambda=D_i$, integrating over the interval $[0,\tau]\subset[0,T]$ and using (3.21), (3.22), (3.23) we find that,

$$\begin{aligned}
\int_D\left\|\hat{u}(x,\tau)\right\|_2^2dx\le{}&\|\mathcal{R}_t\|_{L^2(D)}^2+C_1\sqrt{T|D|}\|\mathcal{R}_{\mathrm{div}}\|_{L^2(\Omega)}+C_1(1+\nu)\sqrt{T|\partial D|}\|\mathcal{R}_s\|_{L^2(\partial D\times[0,T])}\\
&+C_1(1+\nu)\sqrt{T|\Gamma|}\max_j\left\|(u_\theta^a)_j-(u_\theta^b)_j\right\|_{L^2(\Gamma\times[0,T])}\\
&+C_1\sqrt{T|\Gamma|}\left\|p_\theta^a-p_\theta^b\right\|_{L^2(\Gamma\times[0,T])}+2d^2\|\nabla u\|_{L^\infty(\Omega)}\int_{D\times[0,\tau]}\left\|\hat{u}(x,t)\right\|_2^2dxdt\\
&+C_1\nu\sqrt{T|\Gamma|}\max_{i,j}\left\|\partial_i(u_\theta^a)_j-\partial_i(u_\theta^b)_j\right\|_{L^2(\Gamma\times[0,T])}\\
&+\|\mathcal{R}_{\mathrm{PDE}}\|_{L^2(\Omega)}^2+\int_{D\times[0,\tau]}\left\|\hat{u}(x,t)\right\|_2^2dxdt,
\end{aligned}\tag{3.25}$$

where $\mathcal{R}_s=(\mathcal{R}_{s,u},\mathcal{R}_{s,p},\mathcal{R}_{s,\nabla u})$ as in (3.13). Using Grönwall's inequality and integrating over $[0,T]$, we find that,

$$\int_\Omega\left\|\hat{u}(x,t)\right\|_2^2dxdt\le\mathcal{C}T\exp\Big(T(2d^2\|\nabla u\|_{L^\infty(\Omega)}+1)\Big),\tag{3.26}$$

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

$$\mathcal{C} = \|\mathcal{R}_t\|^2_{L^2(D)} + \|\mathcal{R}_{\text{PDE}}\|^2_{L^2(\Omega)} + C_1\sqrt{T}\Bigg[\sqrt{|D|}\|\mathcal{R}_{\text{div}}\|_{L^2(\Omega)} + (1+\nu)\sqrt{|\partial D|}\|\mathcal{R}_s\|_{L^2(\partial D\times[0,T])} + \sqrt{|\Gamma|}\Bigg((1+\nu)\|\mathcal{R}_u\|_{L^2(\Gamma\times[0,T])} + \nu\|\mathcal{R}_{\nabla u}\|_{L^2(\Gamma\times[0,T])} + \big\|\mathcal{R}_p\big\|_{L^2(\Gamma\times[0,T])}\Bigg)\Bigg]. \tag{3.27}$$

□

Remark 3.5. *Although the existence of a C^1 solution of the Navier-Stokes solution is guaranteed by Theorem 2.1, it is still possible that $\|\nabla u\|_{L^\infty(\Omega)}$ becomes very large, e.g. for complicated solutions characterized by strong vorticity (Mishra & Molinaro 2020). In such a case, Theorem 3.4 indicates that the generalization error might be large.*

Remark 3.6. *For PINNs, the L^2-error is bounded uniquely in terms of residuals that are a part of the PINN generalization error* (2.9)*. This implies that for neural networks with a small PINN loss the corresponding L^2-error will be small as well, provided that the C^1-norm of the network does not blow up. This affirmatively answers question Q2. For XPINNs, we can see that the XPINN-specific residuals $\mathcal{R}_u$ and $\mathcal{R}_p$ (as defined in* (3.15)*) are equivalent with the $\mathcal{R}_u$ residual in the XPINN generalization error* (2.15)*. The residual $\mathcal{R}_{\nabla u}$ however does not show up in the original XPINN framework, and should therefore be added to the XPINN loss function* (2.15) *to theoretically guarantee a small L^2-error.*

Remark 3.7. *Variants of Theorem 3.4 for different kinds of boundary conditions can be proven in the same way as above. For example, the statement from Theorem 3.4 still holds for no-slip boundary conditions i.e., $u(x,t) = 0$ for all $(x,t) \in \partial D \times [0,T]$, if one defined the spatial boundary residual as $\mathcal{R}_s = u_\theta$.*

Remark 3.8. *Although the focus in this paper lies on solving the Navier-Stokes equations for the velocity, we want to note that one can also prove a stability result for $\|p - p_\theta\|_{L^2(\Omega)}$ in a similar spirit to Theorem 3.4. The main steps consists of taking the divergence of the Navier-Stokes equations, using the identity* (2.2) *and rewriting the result in terms of the different residuals.*

The existence of a PINN (XPINN) with an arbitrarily small L^2-error is a simple byproduct of the proof of Theorem 3.1. For completeness, we show that one can also use Theorem 3.4 to obtain a quantitative convergence result on the L^2-error of the PINN approximation of the solution of the Navier-Stokes equation in terms of the number of neurons of the neural network.

Corollary 3.9. *Let $n \geq 2$, $d, r, k \in \mathbb{N}$, where $k \geq 3$ and let $u_0 \in H^r(\mathbb{T}^d)$ with $r > \frac{d}{2} + 2k$ and $\operatorname{div}(u_0) = 0$. It holds that:*

- *there exist $T > 0$ and a classical solution u to the Navier-Stokes equations such that $u \in H^k(\Omega)$, $\nabla p \in H^{k-1}(\Omega)$, $\Omega = \mathbb{T}^d \times [0,T]$, and $u(t=0) = u_0$,*
- *there exist constants $C, \beta > 0$ such that for every $N \in \mathbb{N}$, there exist tanh neural networks $\hat{u}_j$, $1 \leq j \leq d$, and $\widehat{p}$, each with two hidden layers, of widths $3\left\lceil \frac{k+n-2}{2} \right\rceil \binom{d+k-1}{d} + \lceil TN \rceil + dN$ and $3\left\lceil \frac{d+n}{2} \right\rceil \binom{2d+1}{d} \lceil TN \rceil N^d$, such that for every $1 \leq j \leq d$,*

$$\|u - \hat{u}\|_{L^2(\Omega)} \leq C \ln^\kappa(\beta N) N^{\frac{-k+1}{2}}. \tag{3.28}$$

The value of $C > 0$ follows from the proof, $\beta > 0$ and the network weight growth are as in Theorem 3.1, and $\kappa = 2$ for $k = 3$ and $\kappa = \frac{1}{2}$ for $k \geq 4$.

Proof. The corollary is a direct consequence of Theorem 3.4 and Theorem 3.1 and its proof. □

3.3. **Bounds on the total error in terms of training error.** Next, we answer the question Q3, raised in the introduction, by providing a bound of the generalization error in terms of the training error and the size of the training set $\mathcal{S}$, where $u_{\theta^*(\mathcal{S})}$ is the PINN that minimizes the training loss. Combined with Theorem 3.4, it will enable us to bound the total error (the L^2-mismatch between the exact solution of (2.1) and the trained PINN) in terms of the training error and size of the training set.

As already announced in Section 2.3, we will focus on training sets obtained using the *midpoint rule* $\mathcal{Q}_M$ for simplicity. For $f \in \{\mathcal{R}^2_{\text{PDE}}, \mathcal{R}^2_{\text{div}}\}$ and $\Lambda = \Omega = D \times [0,T]$ we obtain the quadrature $\mathcal{Q}^{\text{int}}_M$, for $f = \mathcal{R}^2_t$ and $\Lambda = D$ we obtain the quadrature $\mathcal{Q}^t_M$ and for $f = \mathcal{R}^2_s$ and $\Lambda = \partial D \times [0,T]$ we obtain the quadrature $\mathcal{Q}^s_M$. For XPINNs, one additionally needs to consider the quadrature $\mathcal{Q}^\Gamma_M$ obtained for $f \in \{\mathcal{R}^2_u, \mathcal{R}^2_{\nabla u}, \mathcal{R}^2_p\}$ and $\Lambda = \Gamma$.

This notation allows us to write the PINN loss (2.10) in a compact manner,

$$\begin{aligned}\mathcal{E}_T(\theta,\mathcal{S})^2 &= \mathcal{E}^{\text{PDE}}_T(\theta,\mathcal{S}_{\text{int}})^2 + \mathcal{E}^{\text{div}}_T(\theta,\mathcal{S}_{\text{int}})^2 + \mathcal{E}^s_T(\mathcal{S}_s)^2 + \mathcal{E}^t_T(\theta,\mathcal{S}_t)^2,\\ &= \mathcal{Q}^{\text{int}}_{M_{\text{int}}}[\mathcal{R}^2_{\text{PDE}}] + \mathcal{Q}^{\text{int}}_{M_{\text{int}}}[\mathcal{R}^2_{\text{div}}] + \mathcal{Q}^s_{M_s}[\mathcal{R}^2_s] + \mathcal{Q}^t_{M_t}[\mathcal{R}^2_t],\end{aligned} \tag{3.29}$$

Using this notation and Theorem 3.4 from the previous section, we obtain the following theorem that bounds the L^2-error of a neural network in terms of the training loss and the number of training points. In particular, it applies to the trained PINN $u_{\theta^*(\mathcal{S})}$.

Theorem 3.10. *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 (2.1) and let (u_θ, p_θ) be a PINN with parameters $\theta \in \Theta_{L,W,R}$ (cf. Definition 2.3). Then the following error bound holds,*

$$\begin{aligned}\int_\Omega \left\|u(x,t) - u_\theta(x,t)\right\|_2^2 dxdt &\le \mathcal{C}(M) T \exp\Big(T(2d^2\|\nabla u\|_{L^\infty(\Omega)} + 1)\Big)\\ &= \mathcal{O}\left(\mathcal{E}_T(\theta,\mathcal{S})^2 + M_t^{-\frac{2}{d}} + M_{\text{int}}^{-\frac{1}{d+1}} + M_s^{-\frac{1}{d}}\right).\end{aligned} \tag{3.30}$$

In the above formula, the constant $\mathcal{C}(M)$ is defined as,

$$\begin{aligned}\mathcal{C}(M) = \mathcal{E}^t_T(\theta,\mathcal{S}_t)^2 &+ C_t M_t^{-\frac{2}{d}} + \mathcal{E}^{\text{PDE}}_T(\theta,\mathcal{S}_{\text{int}})^2 + C_{\text{PDE}} M_{\text{int}}^{-\frac{2}{d+1}}\\ &+ C_1 T^{\frac{1}{2}}\left[\mathcal{E}^{\text{div}}_T(\theta,\mathcal{S}_{\text{int}}) + C_{\text{div}} M_{\text{int}}^{-\frac{1}{d+1}} + (1+\nu)\big(\mathcal{E}^s_T(\theta,\mathcal{S}_s) + C_s M_s^{-\frac{1}{d}}\big)\right],\end{aligned} \tag{3.31}$$

and where,

$$\begin{aligned}C_1 &\lesssim \|u\|_{C^1} + \|p\|_{C^0} + \|\hat{u}\|_{C^1} + \|\hat{p}\|_{C^0}\\ &\lesssim \|u\|_{C^1} + \|p\|_{C^0} + (d+1)^2\left(16e^2W^3R\|\sigma\|_{C^1}\right)^L,\\ C_t &\lesssim \|u\|^2_{C^2} + \|\hat{u}\|^2_{C^2} \lesssim \|u\|^2_{C^2} + \left(e^2 2^6 W^3 R^2 \|\sigma\|_{C^2}\right)^{2L},\\ C_{\text{PDE}} &\lesssim \left\|\hat{u}_j\right\|^2_{C^4} \lesssim \left(2e^2 4^4 W^3 R^4 \|\sigma\|_{C^4}\right)^{4L},\\ C_{\text{div}}, C_s &\lesssim \left\|\hat{u}_j\right\|_{C^3} \lesssim \left(4e^2 3^4 W^3 R^3 \|\sigma\|_{C^3}\right)^{3L/2}.\end{aligned} \tag{3.32}$$

Proof. The main error estimate of the theorem follows directly from combining Theorem 3.4 with the quadrature error formula (2.7). The complexity of C_1 follows

from Theorem 3.4 and Lemma C.1, which states that

$$\left\|\hat{u}_j\right\|_{C^n} \leq 16^L (d+1)^{2n} \left(e^2 n^4 W^3 R^n \|\sigma\|_{C^n}\right)^{nL} \tag{3.33}$$

for $n \in \mathbb{N}$, all j and similarly for $\hat{p}$. The complexities of the other constants then follow from this formula and the observation that for every residual $\mathcal{R}_q$ it holds that $\left\|\mathcal{R}_q^2\right\|_{C^n} \leq 2^n \left\|\mathcal{R}_q\right\|_{C^n}^2$ (from the general Leibniz rule). For instance, we obtain in this way that

$$C_t \lesssim \left\|\mathcal{R}_t^2\right\|_{C^2} \lesssim \|u\|_{C^2}^2 + \|\hat{u}\|_{C^2}^2 \lesssim \|u\|_{C^2}^2 + \left(e^2 2^6 W^3 R^2 \|\sigma\|_{C^2}\right)^{2L}. \tag{3.34}$$

In a similar way, one can calculate that

$$C_{\text{PDE}} \lesssim \left\|\mathcal{R}_{\text{PDE}}^2\right\|_{C^2} \lesssim \|\hat{u}\|_{C^4}^2 \lesssim \left(2e^2 4^4 W^3 R^4 \|\sigma\|_{C^4}\right)^{4L}. \tag{3.35}$$

Finally, we find that

$$C_s^2, C_{\text{div}}^2 \lesssim \left\|\mathcal{R}_s^2\right\|_{C^2}, \left\|\mathcal{R}_{\text{div}}^2\right\|_{C^2} \lesssim \|\hat{u}\|_{C^3}^2 \lesssim \left(4e^2 3^4 W^3 R^3 \|\sigma\|_{C^3}\right)^{3L}. \tag{3.36}$$

□

Remark 3.11. *For XPINNs, an entirely analogous result can be proven using the same approach.*

Remark 3.12. *The upper bounds on the constants in* (3.32) *depend polynomially on the network width W but exponentially on the network depth L. These bounds seem to suggest that one might expect a smaller L^2-error for a rather shallow (but wide) network than for a very deep network. In Theorem 3.1, we have already proven explicit error bounds for a neural network with only two hidden layers. It is important to note that the constants in* (3.32) *can be estimated from the network used in practice, and that there is no need to use the (over)estimates from Theorem 3.1 in this case.*

Remark 3.13. *If we assume that the optimization algorithm used to minimize the training loss i.e., to solve* (2.11)*, finds a global minimum, then one can prove that the training error in Theorem 3.10 is small if the training set and hypothesis space is large enough. To see this, first fix an error tolerance ϵ and observe that for the network $\hat{u}$ that was constructed in Theorem 3.1 it holds that all relevant PINN residuals and therefore also the generalization error $\mathcal{E}_G(\theta_{\hat{u}})$* (2.9) *are of order $\mathcal{O}(\varepsilon)$. If one then constructs the training set such that $M_{\text{int}} \sim \epsilon^{-\frac{d+1}{2}}$ and $M_t \sim M_s \sim \epsilon^{-\frac{d}{2}}$ then it holds that $\mathcal{E}_T^q(\theta_\Psi) \leq \epsilon + \mathcal{E}_G^q(\theta_\Psi)$ for $q \in \{s, t, \text{div}, \text{PDE}\}$ and as a consequence that $\mathcal{E}_T(\theta_{\hat{u}}, \mathcal{S}) = \mathcal{O}(\epsilon)$. If the optimization algorithm reaches a global minimum, the training loss of $u_{\theta^*(\mathcal{S})}$ will be upper bounded by that of $\hat{u}$. Therefore it also holds that $\mathcal{E}_T(\mathcal{S}) = \mathcal{O}(\epsilon)$.*

With these remarks in place, we now discuss how Theorem 3.10 answers question Q3, raised in the introduction, which asked whether the generalization error $\mathcal{E}_G(\theta^*)$ will be small if the training error is small and the training set is sufficiently large.

First of all, Theorem 3.10 can be used as an *a posteriori* error estimate for the trained PINN as (3.30) shows that the generalization error can be upper bounded in terms of the various PINN-related residuals, as well as the C^k-norms of the trained neural network and the training set sizes. The power of (3.30) as an a posteriori error estimate will be demonstrated with numerical experiments in Section 4.

Next, we combine Theorem 3.1 with Theorem 3.10 to prove an *a priori* error estimate. The following result states that the total L^2-error of the trained PINN $\|u - u_{\theta^*(\mathcal{S})}\|_{L^2(D\times[0,T])}$ can be made arbitrarily small if the optimization procedure

for (2.11) leads to a global minimum, the training set and the underlying spaces of neural networks are sufficiently large, and the solution u is sufficiently smooth.

Corollary 3.14. *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 (2.1), let the hypothesis space Θ satisfy $R \geq \epsilon^{-1/(k-\gamma)} \ln(1/\epsilon)$, $W \geq \epsilon^{-(d+1)/(k-\gamma)}$ and $L \geq 3$, and let $(u_{\theta^*(\mathcal{S})}, p_{\theta^*(\mathcal{S})})$ be the PINN that solves (2.11) where the training set $\mathcal{S}$ satisfies $M_t \geq \epsilon^{-d(1+\gamma/(k-\gamma))}$, $M_{\text{int}} \geq \epsilon^{-2(d+1)(1+\gamma/(k-\gamma))}$ and $M_s \geq \epsilon^{-2d(1+\gamma/(k-\gamma))}$. It holds that*

$$\|u - u_{\theta^*(\mathcal{S})}\|_{L^2(D\times[0,T])} = \mathcal{O}(\epsilon). \tag{3.37}$$

Proof. For arbitrary $N \in \mathbb{N}$, we set $R = \mathcal{O}(N \ln(N))$, $W = \mathcal{O}(N^{d+1})$ and $L = 3$ and we let $\Theta := \Theta_{L,W,R}$ as in Definition 2.3. From Theorem 3.1 we know that there exists $\hat{\theta} \in \Theta$ for which $\mathcal{E}_G(\hat{\theta}) = \mathcal{O}(\ln^2(N) N^{-k+2})$.

Now let $\theta^*(\mathcal{S})$ be the parameter that minimizes $\min_{\theta\in\Theta} \mathcal{E}_T(\theta, \mathcal{S})$ as in (2.11). We note that Theorem 3.10 implies that

$$\|u - u_{\theta^*(\mathcal{S})}\|^2_{L^2(D\times[0,T])} \lesssim (W^3 R^4)^{4L} \left(\mathcal{E}_T(\theta^*(\mathcal{S}), \mathcal{S})^2 + M_t^{-\frac{2}{d}} + M_{\text{int}}^{-\frac{1}{d+1}} + M_s^{-\frac{1}{d}} \right). \tag{3.38}$$

By definition, it must hold that $\mathcal{E}_T(\theta^*(\mathcal{S}), \mathcal{S}) \leq \mathcal{E}_T(\hat{\theta}, \mathcal{S})$. We use the same quadrature rules as in Theorem 3.10 to bound $\mathcal{E}_T(\hat{\theta}, \mathcal{S})$ by $\mathcal{E}_G(\hat{\theta})$ and rewrite the previous bound as,

$$\|u - u_{\theta^*(\mathcal{S})}\|^2_{L^2(D\times[0,T])} \lesssim (W^3 R^4)^{4L} \left(\mathcal{E}_G(\hat{\theta})^2 + M_t^{-\frac{2}{d}} + M_{\text{int}}^{-\frac{1}{d+1}} + M_s^{-\frac{1}{d}} \right). \tag{3.39}$$

Rewriting everything in terms of N, we find that

$$\begin{aligned} \|u - u_{\theta^*(\mathcal{S})}\|^2_{L^2(D\times[0,T])} &\lesssim \left(N^{3(d+1)} N^4 \ln^4(N) \right)^{12} \left(\ln^4(N) N^{-2k+4} + M_t^{-\frac{2}{d}} + M_{\text{int}}^{-\frac{1}{d+1}} + M_s^{-\frac{1}{d}} \right) \\ &\lesssim N^{12(3d+8)} \left(N^{-2k} + M_t^{-\frac{2}{d}} + M_{\text{int}}^{-\frac{1}{d+1}} + M_s^{-\frac{1}{d}} \right). \end{aligned} \tag{3.40}$$

We now choose N and the training set sizes in such a way that the RHS of the above inequality is $\mathcal{O}(\epsilon^2)$. Concretely, this means setting $N = \epsilon^{-1/(k-\gamma)}$, $M_t = \epsilon^{-d(1+\gamma/(k-\gamma))}$, $M_{\text{int}} = \epsilon^{-2(d+1)(1+\gamma/(k-\gamma))}$ and $M_s = \epsilon^{-2d(1+\gamma/(k-\gamma))}$. This concludes the proof of the corollary. □

4. Numerical experiments

In this section, we seek to illustrate the bounds on error of the PINN and XPINN approximations of the Navier-Stokes equations (2.1), empirically with a numerical experiment.

To this end, we consider the Navier-Stokes equations in two space dimensions and initial data that corresponds to the Taylor-Green vortex test case, which is an unsteady flow of decaying vortices. The exact closed form solutions of Taylor-Green vortex problem are given by

$$\begin{aligned} u(t,x,y) &= -\cos(\pi x)\sin(\pi y)\exp(-2\pi^2\nu t) \\ v(t,x,y) &= \sin(\pi x)\cos(\pi y)\exp(-2\pi^2\nu t) \\ p(t,x,y) &= -\frac{\rho}{4}\left[\cos(2\pi x) + \cos(2\pi y)\right]\exp(-4\pi^2\nu t) \end{aligned}$$

The spatio-temporal domain is $x, y \in [0.5, 4.5]^2$ and $t \in [0, 1]$.

The Taylor-Green vortex serves two key requirements in our context. First, it provides an analytical solution of the Navier-Stokes equations and enables us to evaluate L^2-errors with respect to this exact solution and without having to consider further (numerical) approximations. Second, the underlying solution is clearly smooth enough to fit the regularity criteria of all our error estimates, presented in the previous section.

We will approximate the Taylor-Green vortex with PINNs and XPINNs. In case of XPINNs, we decompose the domain into two subdomains along x-axis ($x \geq 2.5$ and $x < 2.5$) where separate neural networks are employed. On the common interface we used 300 points for stitching these two subdomains together. The value of density is set at $\rho = 1$. An ensemble training procedure is performed to find the correlation between the total error ($\mathcal{E}$) and the training error ($\mathcal{E}_T$) for different values of ν. For the neural network training we used full batch with Adam optimizer for the first 20000 number of iterations, followed by L-BFGS optimizer (Byrd et al. 1995) for another 60000 iterations or till convergence. The number of layers in both PINN and XPINN are 2 (as suggested by the theory) with 80 neurons in each layer, and the quadrature points are 27K, which are obtained using mid-point rule. The learning rate is 8e-4, and the activation function is hyperbolic tangent in both cases. We train the networks 80 times with different set of initialization to weights and biases. Figure 3 shows the log of training error vs. log of total error for each parameter configuration during ensemble training with three different values of viscosity ν. As seen from this figure, total error $\mathcal{E} = \|u - u^*\|_{L^2}$ and the training error $\mathcal{E}_T$ (3.29) are very tightly correlated (along the diagonal in Figure 3). In particular and consistent with the estimates in Theorem 3.10, a small training error implies a small total error. Moreover, we see from Figure 3 that the total error $\mathcal{E}$ approximately scales as the square root of training error i.e., $\mathcal{E} \lesssim \sqrt{\mathcal{E}_T}$, which is also consistent with the bounds in Theorem 3.10.

Next, we investigate the behavior of the total and training errors by varying the number of quadrature points. To this end, we train both PINNs and XPINNs 20 times with different parameter initializations and plot the mean and standard deviation of the errors as shown in Figure 4. All results are of a neural network architecture with 2 hidden layers, with 80 neurons in each layer and the hyperbolic tangent activation function. Moreover, the learning rate is the same as before.

We see from Figure 4 that both the training as well as total errors decay with respect to the number of quadrature points till they are saturated around 27K quadrature points and do not decay any further. To further illustrate the error estimates derived in the previous section, we revisit the error estimate (3.30). Given the elaboration of the appearing constants in (3.32) and the fact that we have access to the exact solution for the Taylor-Green vortex as well as to the (derivatives of) the PINN, we can *explicitly compute* a theoretical bound on the total error in (3.30). This error depends on the number of quadrature points as well as on the particular weights of the trained PINN. This theoretical bound is also depicted in Figure 4. We see from this figure that the computed theoretical bound closely tracks the qualitative as well as quantitative behavior of the total error for all cases considered here. The rates of decay of both the error and the bound are very similar. However the bound is not quantitatively sharp as there is an approximately one order of magnitude difference in its amplitude vis a vis the total error. Such non-sharp bounds on the error are common in theoretical machine learning, see (Arora et al. 2018) for instance. Even in the case of PINNs, they were already seen in (Mishra & Molinaro 2020) where the authors observed at least two to three orders of magnitude discrepancy between their theoretical bounds and the realized total error. Given

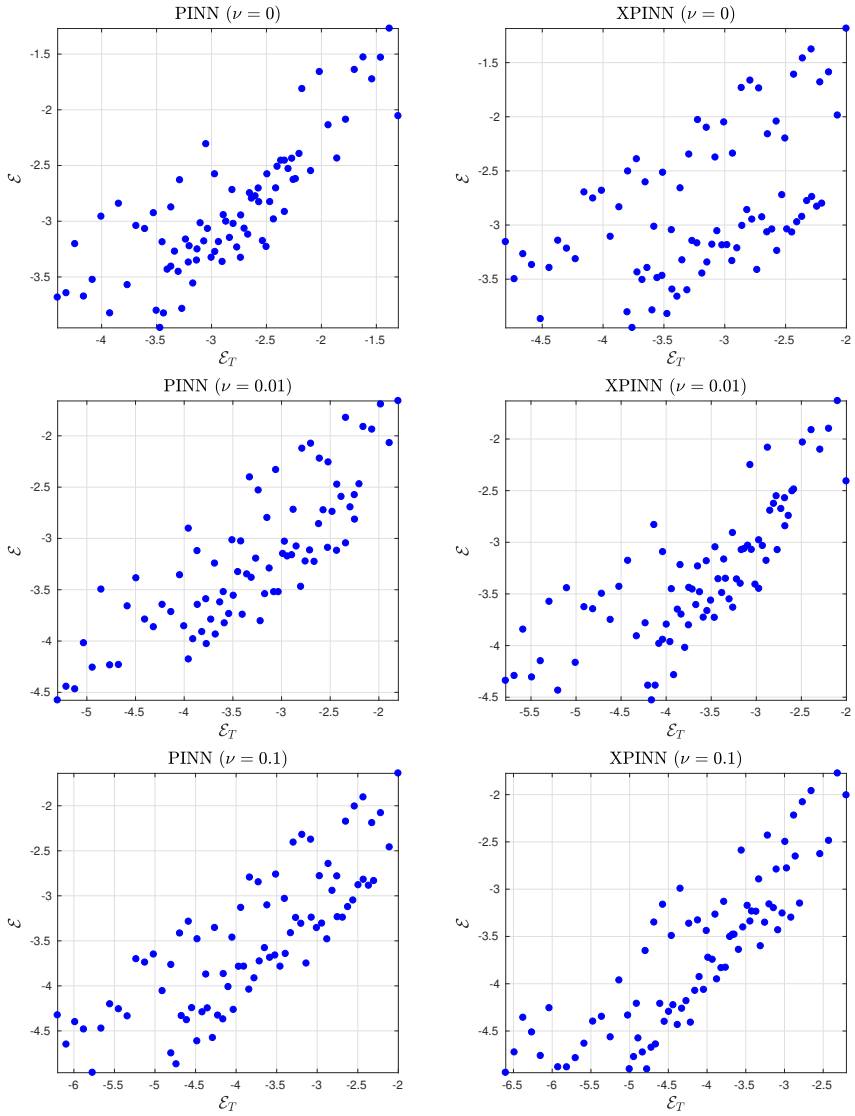

FIGURE 3. Log of training error vs. log of total error for each parameter (weights and biases) configuration during ensemble training. We used $\nu = 0, 0.01$ and 0.1.

this context, an order of magnitude discrepancy between the bound in (3.30) and the observed total error is quite satisfactory.

Finally, we study the behavior of the error as the number of neurons is increased. Given our theoretical considerations, where the relevant error estimates where shown for tanh neural networks with two hidden layers, we restrict ourselves to this setting by only varying the network width and keep the number of hidden layers fixed at two. We again train the PINN and XPINNs networks 20 times with different parameter initializations. The learning rate has the same value as in the previous numerical experiments and the number of quadrature points is fixed at 64 K. The resulting training and total errors are presented in Figure 5 and show that the total error decreases with the number of neurons in each layer till it gets saturated. Moreover, the computable upper bound (3.30) is also depicted and we

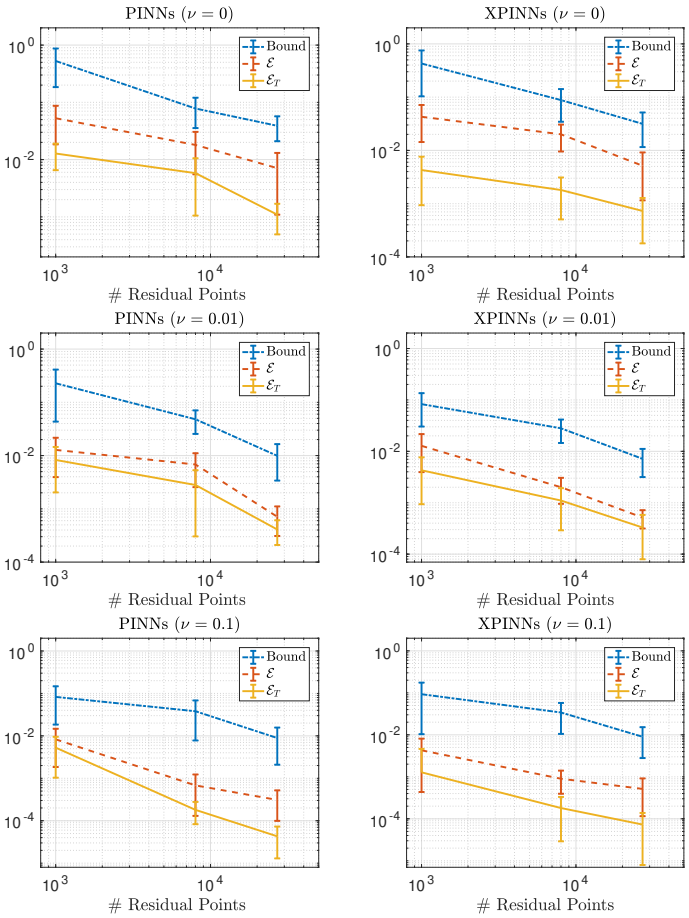

FIGURE 4. Training and total errors for different number of quadrature points (residual points).

see from this figure, that the bound (3.30) follows the same decaying trend, till saturation, as the total as well as training errors in this particular example.

5. Discussion

Physics-informed neural networks have been very successful in the numerical approximation of the solutions of forward as well as inverse problems for various classes of PDEs. However, there is a significant paucity of theoretical results on the resulting errors. Following the framework of a recent paper (De Ryck & Mishra 2021), we revisit the key theoretical questions Q1 (on the smallness of the PDE residual in the class of neural networks), Q2 (a small residual implying a small total error) and Q3 (small training errors imply small total errors for sufficient number of quadrature points), raised in the introduction. We have answered these questions affirmatively for the incompressible Navier-Stokes equations in this paper. The incompressible Navier-Stokes equations constitute a very important example for nonlinear PDEs and PINNs have already been used in approximating them before (Jin et al. 2021) but without much theoretical justification.

Summarizing our theoretical results, we have shown in this paper that

- For sufficiently smooth (Sobolev regular) initial data, there exists neural networks, with the tanh activation function and with two hidden layers,

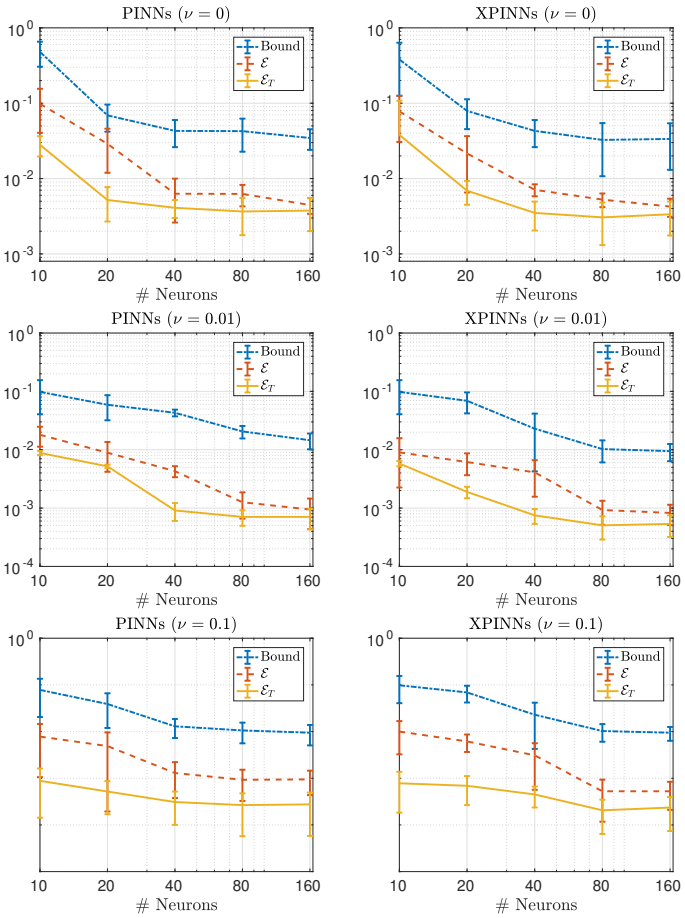

FIGURE 5. Training and total errors for different number of neurons.

such that the resulting PDE residuals can be arbitrarily small. Moreover in Theorem 3.1, we obtain very precise quantitative estimates on the sizes of resulting neural networks, in terms of regularity of the underlying classical solution. The proof of this approximation result relies heavily on the smoothness of the solutions of Navier-Stokes and on the approximation of smooth functions by neural networks in sufficiently high Sobolev norms.

- In Theorem 3.4, we show that the total L^2 error of the PINN (and XPINN) approximations is bounded by the PDE residuals for the incompressible Navier-Stokes equations. Moreover, the underlying constants in the bound are clearly quantified in terms of the underlying classical solution as well as the approximating neural networks. This result leverages the stability (or rather coercivity) of classical solutions of the Navier-Stokes equations. Thus, we answer question Q2 affirmatively by showing a small PDE residual implies a small total error.
- In Theorem 3.10, we answer question Q3 by proving a bound (3.30) on the total error in terms of the training error and the number of quadrature points. Thus, if one reaches a global minimum of the underlying optimization problem (2.11), one can show that the training error, and consequently the total error, can be made as small as possible if sufficient number of quadrature points are considered.

Taken together, the above theorems constitute the first comprehensive theoretical analysis of PINNs (and XPINNs) for a prototypical nonlinear PDE, the Navier-Stokes equations. We also illustrate the bounds in a simple numerical experiment demonstrating a qualitative as well as quantitative agreement between the rigorous bounds and the empirical results.

Given this account of the strengths of our results, it is also fair to point out possible limitations and highlight avenues for future investigation. These include

- Our estimates do not estimate the training error $\mathcal{E}_T$, except under the assumption that one finds a global minimum for the optimization problem (2.11). In practice, it is well known that (stochastic) gradient descent algorithms converge to local minima. In such cases, there is no guarantee on the smallness of the training error. Thus, one needs to find new techniques to estimate training errors for PINNs. On the other hand, our and other numerical results, see (Jin et al. 2021) for instance, indicate that the training error can be made small. Then a bound like (3.30) clearly indicates that the overall error will be small. This is indeed borne out in numerical experiments (see Figure 5).
- Our estimates rely heavily on the regularity of the underlying solutions of Navier-Stokes equations (2.1). There are two caveats in this context. First, in two space dimensions, one knows that the underlying solution will be sufficiently regular if the corresponding initial data is regular enough (Temam 2001). However, in three space dimensions, such results are a part of the millennium prize problems and are incredibly hard to obtain. On the more practical level, it is clear from Theorems 3.4 and 3.10 that the errors will grow if the C^1 norms of the underlying exact solutions are large. This is clearly the case, particularly in three space dimensions, where solution gradients grow as vortices are stretched. Hence, one can expect the PINN errors to grow too and this is indeed seen in practice (see section 5 of (Mishra & Molinaro 2020) for instance). However, traditional numerical methods such as finite element methods and spectral viscosity methods also suffer from the same issue and it is not expected to be different for PINNs. In this context, it would be interesting to investigate if the approaches which are based on weak formulations of PDE residuals might lead to better estimates and numerical results.

Finally, only the forward problem is considered here. It would be interesting to extend the theoretical tools and bounds in the paper to inverse problems for the Navier-Stokes equations (see (Raissi et al. 2018)) as well as physics-informed operator learning (Wang & Perdikaris 2021).

References

Arora, S., Ge, R., Neyshabur, B. & Zhang, Y. (2018), Stronger generalization bounds for deep nets via a compression approach, *in* 'Proceedings of the 35th International Conference on Machine Learning, ICML', Vol. 80 of *Proceedings of Machine Learning Research*, pp. 254–263.

Bai, G., Koley, U., Mishra, S. & Molinaro, R. (2021), 'Physics informed neural networks (PINNs) for approximating nonlinear dispersive PDEs', *arXiv preprint arXiv:2104.05584* .

Biswas, A., Tian, J. & Ulusoy, S. (2020), 'Error estimates for deep learning methods in fluid dynamics', *arXiv preprint arXiv:2008.02844v1* .

Byrd, R. H., Lu, P., Nocedal, J. & Zhu, C. (1995), 'A limited memory algorithm for bound constrained optimization', *SIAM Journal on scientific computing* **16**(5), 1190–1208.

Chen, T. & Chen, H. (1995), 'Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems', *IEEE Transactions on Neural Networks* **6**(4), 911–917.

De Ryck, T., Lanthaler, S. & Mishra, S. (2021), 'On the approximation of functions by tanh neural networks', *Neural Networks* **143**, 732–750.

De Ryck, T. & Mishra, S. (2021), Error analysis for physics informed neural networks (PINNs) approximating Kolmogorov PDEs. Preprint, available from arXiv:2106:14473.

Dissanayake, M. & Phan-Thien, N. (1994), 'Neural-network-based approximations for solving partial differential equations', *Communications in Numerical Methods in Engineering* .

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

Fornberg, B. (1988), 'Generation of finite difference formulas on arbitrarily spaced grids', *Mathematics of computation* **51**(184), 699–706.

Gühring, I. & Raslan, M. (2021), 'Approximation rates for neural networks with encodable weights in smoothness spaces', *Neural Networks* **134**, 107–130.

Hiptmair, R. & Schwab, C. (2008), *Numerical Methods for Elliptic and Parabolic Boundary Value Problems*, ETH Zürich.

Hu, Z., Jagtap, A. D., Karniadakis, G. E. & Kawaguchi, K. (2021), 'When do extended physics-informed neural networks (XPINNs) improve generalization?', *arXiv preprint arXiv:2109.09444* .

Jagtap, A. D. & Karniadakis, G. E. (2020), 'Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations', *Communications in Computational Physics* **28**(5), 2002–2041.

Jagtap, A. D., Kharazmi, E. & Karniadakis, G. E. (2020), 'Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems', *Computer Methods in Applied Mechanics and Engineering* **365**, 113028.

Jagtap, A. D., Mao, Z., Adams, N. & Karniadakis, G. E. (2022), 'Physics-informed neural networks for inverse problems in supersonic flows', *arXiv preprint arXiv:2202.11821* .

Jagtap, A. D., Mitsotakis, D. & Karniadakis, G. E. (2022), 'Deep learning of inverse water waves problems using multi-fidelity data: Application to Serre–Green–Naghdi equations', *Ocean Engineering* **248**, 110775.

Jin, X., Cai, S., Li, H. & Karniadakis, G. E. (2021), 'NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations', *Journal of Computational Physics* **426**, 109951.

Kovachki, N., Lanthaler, S. & Mishra, S. (2021), 'On universal approximation and error bounds for Fourier Neural Operators', *Journal of Machine Learning Research* **22**, 1–76.

Kutyniok, G., Petersen, P., Raslan, M. & Schneider, R. (2021), 'A theoretical analysis of deep neural networks and parametric PDEs', *Constructive Approximation* pp. 1–53.

Lagaris, I. E., Likas, A. & D., P. G. (2000), 'Neural-network methods for boundary value problems with irregular boundaries', *IEEE Transactions on Neural Networks* **11**, 1041–1049.

Lagaris, I. E., Likas, A. & Fotiadis, D. I. (2000), 'Artificial neural networks for solving ordinary and partial differential equations', *IEEE Transactions on Neural*

Networks **9(5)**, 987–1000.

Lanthaler, S., Mishra, S. & Karniadakis, G. E. (2022), 'Error estimates for deeponets: A deep learning framework in infinite dimensions', *Transactions of Mathematics and Its Applications* **6**(1).

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *Nature* **521**(7553), 436–444.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. & Anandkumar, A. (2020), 'Fourier neural operator for parametric partial differential equations'.

Lu, L., Jin, P. & Karniadakis, G. E. (2019), 'DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators', *arXiv preprint arXiv:1910.03193* .

Lye, K. O., Mishra, S. & Ray, D. (2020), 'Deep learning observables in computational fluid dynamics', *Journal of Computational Physics* p. 109339.

Lye, K. O., Mishra, S., Ray, D. & Chandrashekar, P. (2021), 'Iterative surrogate model optimization (ISMO): An active learning algorithm for PDE constrained optimization with deep neural networks', *Computer Methods in Applied Mechanics and Engineering* **374**, 113575.

Majda, A. J., Bertozzi, A. L. & Ogawa, A. (2002), 'Vorticity and incompressible flow. cambridge texts in applied mathematics', *Appl. Mech. Rev.* **55**(4), B77–B78.

Mao, Z., Jagtap, A. D. & Karniadakis, G. E. (2020), 'Physics-informed neural networks for high-speed flows.', *Computer Methods in Applied Mechanics and Engineering* **360**, 112789.

Mishra, S. & Molinaro, R. (2020), 'Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs', *arXiv preprint arXiv:2006.16144* .

Mishra, S. & Molinaro, R. (2021*a*), 'Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs', *IMA Journal of Numerical Analysis* .

Mishra, S. & Molinaro, R. (2021*b*), 'Physics informed neural networks for simulating radiative transfer', *Journal of Quantitative Spectroscopy and Radiative Transfer* **270**, 107705.

Mishra, S. & Rusch, T. K. (2021), 'Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences', *SIAM Journal on Numerical Analysis* **59**(3), 1811–1834.

Pang, G., Lu, L. & Karniadakis, G. E. (2019), 'fPINNs: Fractional physics-informed neural networks', *SIAM journal of Scientific computing* **41**, A2603–A2626.

Raissi, M. & Karniadakis, G. E. (2018), 'Hidden physics models: Machine learning of nonlinear partial differential equations', *Journal of Computational Physics* **357**, 125–141.

Raissi, M., Perdikaris, P. & Karniadakis, G. E. (2019), 'Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations', *Journal of Computational Physics* **378**, 686–707.

Raissi, M., Yazdani, A. & Karniadakis, G. E. (2018), 'Hidden fluid mechanics: A Navier-Stokes informed deep learning framework for assimilating flow visualization data', *arXiv preprint arXiv:1808.04327* .

Schwab, C. & Zech, J. (2019), 'Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ', *Analysis and Applications* **17**(01), 19–55.

Shin, Y., Darbon, J. & Karniadakis, G. E. (2020), 'On the convergence and generalization of physics informed neural networks', *arXiv preprint arXiv:2004.01806*

.
Shin, Y., Zhang, Z. & Karniadakis, G. E. (2020), 'Error estimates of residual minimization using neural networks for linear equations', *arXiv preprint arXiv:2010.08019* .
Shukla, K., Jagtap, A. D., Blackshire, J. L., Sparkman, D. & Karniadakis, G. E. (2021), 'A physics-informed neural network for quantifying the microstructural properties of polycrystalline nickel using ultrasound data: A promising approach for solving inverse problems', *IEEE Signal Processing Magazine* **39**(1), 68–77.
Shukla, K., Jagtap, A. D. & Karniadakis, G. E. (2021), 'Parallel physics-informed neural networks via domain decomposition', *Journal of Computational Physics* **447**, 110683.
Temam, R. (2001), *Navier-Stokes equations: theory and numerical analysis*, Vol. 343, American Mathematical Soc.
Verfürth, R. (1999), 'A note on polynomial approximation in Sobolev spaces', *ESAIM: Mathematical Modelling and Numerical Analysis* **33**(4), 715–719.
Wang, S. & Perdikaris, P. (2021), Long-time integration of parametric evolution equations with physics-informed deeponets. Preprint, available from arXiv:2106:05384.
Yang, L., Meng, X. & Karniadakis, G. E. (2021), 'B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data', *Journal of Computational Physics* **425**, 109913.

Appendix A. Notation and auxiliary results

This section provides an overview of the notation used in the paper and recalls some basic results on Sobolev spaces.

A.1. **Multi-index notation.** For $d \in \mathbb{N}$, we call a d-tuple of non-negative integers $\alpha \in \mathbb{N}_0^d$ a multi-index. We write $|\alpha| = \sum_{i=1}^d \alpha_i$, $\alpha! = \prod_{i=1}^d \alpha_i!$ and, for $x \in \mathbb{R}^d$, we denote by $x^\alpha = \prod_{i=1}^d x_i^{\alpha_i}$ the corresponding multinomial. Given two multi-indices $\alpha, \beta \in \mathbb{N}_0^d$, we say that $\alpha \leq \beta$ if, and only if, $\alpha_i \leq \beta_i$ for all $i = 1, \dots, d$. For a multi-index α, we define the following multinomial coefficient

$$\binom{|\alpha|}{\alpha} = \frac{|\alpha|!}{\alpha!}, \tag{A.1}$$

and, given $\alpha \leq \beta$, we define a corresponding multinomial coefficient by

$$\binom{\beta}{\alpha} = \prod_{i=1}^d \binom{\beta_i}{\alpha_i} = \frac{\beta!}{\alpha!(\beta-\alpha)!}. \tag{A.2}$$

For $\Omega \subseteq \mathbb{R}^d$ and a function $f : \Omega \to \mathbb{R}$ we denote by

$$D^\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}} \tag{A.3}$$

the classical or distributional (i.e. weak) derivative of f.

We will also encounter the set $P_{n,d} = \{\alpha \in \mathbb{N}_0^d : |\alpha| = n\}$, for which it holds that $\left|P_{n,d}\right| = \binom{n+d-1}{n}$.

A.2. **Sobolev spaces.** Let $d \in \mathbb{N}$, $k \in \mathbb{N}_0$, $1 \leq p \leq \infty$ and let $\Omega \subseteq \mathbb{R}^d$ be open. We denote by $L^p(\Omega)$ the usual Lebesgue space and for we define the Sobolev space $W^{k,p}(\Omega)$ as

$$W^{k,p}(\Omega) = \{f \in L^p(\Omega) : D^\alpha f \in L^p(\Omega) \text{ for all } \alpha \in \mathbb{N}_0^d \text{ with } |\alpha| \leq k\}. \tag{A.4}$$

For $p < \infty$, we define the following seminorms on $W^{k,p}(\Omega)$,

$$|f|_{W^{m,p}(\Omega)} = \left(\sum_{|\alpha|=m} \|D^\alpha f\|^p_{L^p(\Omega)} \right)^{1/p} \qquad \text{for } m = 0, \dots, k, \tag{A.5}$$

and for $p = \infty$ we define

$$|f|_{W^{m,\infty}(\Omega)} = \max_{|\alpha|=m} \|D^\alpha f\|_{L^\infty(\Omega)} \qquad \text{for } m = 0, \dots, k. \tag{A.6}$$

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

$$\|f\|_{W^{k,p}(\Omega)} = \left(\sum_{m=0}^{k} |f|^p_{W^{m,p}(\Omega)} \right)^{1/p}, \tag{A.7}$$

and for $p = \infty$ we define the norm

$$\|f\|_{W^{k,\infty}(\Omega)} = \max_{0 \le m \le k} |f|_{W^{m,\infty}(\Omega)}. \tag{A.8}$$

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

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

We define the Hilbertian Sobolev spaces for $k \in \mathbb{N}_0$ as $H^k(\Omega) = W^{k,2}(\Omega)$ with corresponding norms $\|\cdot\|_{H^k(\Omega)} = \|\cdot\|_{W^{k,2}(\Omega)}$ and seminorms $|\cdot|_{H^m(\Omega)} = |\cdot|_{W^{m,2}(\Omega)}$ for integers m with $0 \le m \le k$. If k is large enough, the space $H^k(\Omega)$ is a Banach algebra. We also recall a version of the Sobolev embedding theorem and a multiplicative trace inequality.

Lemma A.1. *For $d, k \in \mathbb{N}$ with $k > \frac{d}{2}$, $H^k(\Omega)$ is a Banach algebra i.e., there exists $c_k > 0$ such that*

$$\forall u, v \in H^k(\Omega) : \ \|uv\|_{H^k(\Omega)} \le c_k \|u\|_{H^k(\Omega)} \|v\|_{H^k(\Omega)}. \tag{A.9}$$

Lemma A.2. *Let $d \in \mathbb{N}$, $k, \ell \in \mathbb{N}_0$ with $k > \ell + \frac{d}{2}$ and $\Omega \subset \mathbb{R}^d$ an open set. Every function $f \in H^k(\Omega)$ has a continuous representative belonging to $C^\ell(\Omega)$.*

Lemma A.3 (Multiplicative trace inequality, e.g. Theorem 3.10.1 in (Hiptmair & Schwab 2008))**.** *Let $d \ge 2$, $\Omega \subset \mathbb{R}^d$ be a Lipschitz domain and let $\gamma_0 : H^1(\Omega) \to L^2(\partial\Omega) : u \mapsto u|_{\partial\Omega}$ be the trace operator. Denote by h_Ω the diameter of Ω and by ρ_Ω the radius of the largest d-dimensional ball that can be inscribed into Ω. Then it holds that*

$$\|\gamma_0 u\|_{L^2(\partial\Omega)} \le \sqrt{\frac{2 \max\{2h_\Omega, d\}}{\rho_\Omega}} \|u\|_{H^1(\Omega)} \tag{A.10}$$

Next, we recall the Bramble-Hilbert lemma, which quantifies the accuracy of polynomial approximations of functions in Sobolev spaces. We present a variant of the Bramble-Hilbert lemma for Hilbertian Sobolev spaces proven in (Verfürth 1999).

Lemma A.4. *Let Ω be a bounded convex open domain $\mathbb{R}^d$, $d \ge 2$, with diameter h. For every $f \in H^m(\Omega)$ there exists a polynomial p of degree at most $m-1$ such that for all $0 \le j \le m-1$ it holds that*

$$|f - p|_{H^j(\Omega)} \le c_{m,j} h^{m-j} |f|_{H^m(\Omega)} \tag{A.11}$$

where

$$c_{m,j} = \pi^{j-m} \binom{d+j-1}{j}^{1/2} \frac{((m-j)!)^{1/2}}{\left(\left\lceil \frac{m-j}{d} \right\rceil ! \right)^{d/2}}. \tag{A.12}$$

We proceed by stating a corollary of the general Leibniz rule for Sobolev regular functions.

Lemma A.5. *Let $d \in \mathbb{N}$, $k \in \mathbb{N}_0$, $\Omega \subset \mathbb{R}^d$ and $f \in H^k(\Omega)$ and $g \in W^{k,\infty}(\Omega)$. Then it holds that*

$$\|fg\|_{H^k} \leq 2^k \|f\|_{H^k} \|g\|_{W^{k,\infty}}. \tag{A.13}$$

Finally, we present a result on the Sobolev norm of the composition of two n times continuously differentiable functions (De Ryck et al. 2021, Lemma A.7).

Lemma A.6. *Let $d, m, n \in \mathbb{N}$, $\Omega_1 \subset \mathbb{R}^d$, $\Omega_2 \subset \mathbb{R}^m$, $f \in C^n(\Omega_1; \Omega_2)$ and $g \in C^n(\Omega_2; \mathbb{R})$. Then it holds that*

$$\|g \circ f\|_{W^{n,\infty}(\Omega_1)} \leq 16(e^2 n^4 m d^2)^n \|g\|_{W^{n,\infty}(\Omega_2)} \max_{1 \leq i \leq m} \left\|(f)_i\right\|^n_{W^{n,\infty}(\Omega_1)}. \tag{A.14}$$

Appendix B. Function approximation by tanh neural networks

In this section, we show how one can prove that for every $f \in H^m(\Omega)$, $m \geq 3$, there exists a tanh neural network $\hat{f}$ with two hidden layers such that $\|f - \hat{f}\|_{H^2(\Omega)} \leq \epsilon$ for some $\epsilon > 0$. Results of this type can be found in (Gühring & Raslan 2021) for very general activation functions and in (De Ryck et al. 2021) for the tanh activation function. Both references prove such a result as follows: first, one divides the domain Ω into cubes of edge length $1/N$, with $N \in \mathbb{N}$ large enough. On each of these cubes, f can be approximated in Sobolev norm by a polynomial, by virtue of the Bramble-Hilbert lemma. A global approximation can then be constructed by multiplying each polynomial with the indicator function of the corresponding cubes and summing over all cubes. Replacing these polynomials, multiplications and indicator functions with suitable neural networks results in a new approximation that has approximately the same accuracy.

In the following, we choose (De Ryck et al. 2021) as a guideline, as it provides explicit upper bounds on the neural network size, which is something we aim to provide for the Navier-Stokes equations. The preceding reference Gühring & Raslan (2021) does not give such explicit bounds, but one can use their proofs to obtain similar explicit bounds for more general activation functions.[1] We improve upon (De Ryck et al. 2021) by adapting their proof of the neural network approximation of polynomials such that the bound on the network weights grows less fast. This is accomplished by using an n-th order accurate finite difference formula in the proof, rather than a second order accurate one. Below, we provide an overview of the improved versions of the results of (De Ryck et al. 2021).

The following lemma treats the neural network approximation of multivariate monomials and is the main source of change compared to the original results in (De Ryck et al. 2021). All updates in the other results are mainly consequences of the following lemma.

Lemma B.1 (Approximation of multivariate monomials)**.** *Let $d, s, n \in \mathbb{N}$, $k \in \mathbb{N}_0$ and $M > 0$. Then for every $\epsilon > 0$, there exists a shallow tanh neural network $\Phi_{s,d} : [-M, M]^d \to \mathbb{R}^{\left|P_{s,d+1}\right|}$ of width $3 \left\lceil \frac{s+n-1}{2} \right\rceil \left|P_{s,d+1}\right|$ such that*

$$\max_{\beta \in P_{s,d+1}} \left\| x^\beta - (\Phi_{s,d}(x))_{\iota(\beta)} \right\|_{W^{2,\infty}([-M,M]^d)} \leq \epsilon, \tag{B.1}$$

where $\iota : P_{s,d+1} \to \{1, \ldots \left|P_{s,d+1}\right|\}$ is a bijection. Furthermore, the weights of the network scale as $O\left(\epsilon^{-s/n}\right)$ for small ϵ.

[1] A more complete discussion about the differences between Gühring & Raslan (2021) and De Ryck et al. (2021) can be found in De Ryck et al. (2021).

Proof. We start by constructing a neural network $\hat{f}_{p,h,n}$ that approximates the univariate monomial $f_p : [-M, M] \to \mathbb{R} : x \mapsto x^p$ in $W^{k,\infty}$-norm. In (De Ryck et al. 2021, Lemma 3.1) this has been done by using a second-order accurate finite difference formula. We will generalize this result by using an n-th-order accurate finite difference scheme. In particular we define $\hat{f}_{p,h,n}$ by,

$$\hat{f}_{p,h,n}(x) = \frac{1}{\sigma^{(p)}(0)h^p} \sum_{i=-\ell}^{\ell} a_i \sigma(ihx), \tag{B.2}$$

where $\ell = \frac{p+n-1}{2}$ and where the a_i are the solution to the system of equations

$$\sum_{i=-\ell}^{\ell} a_i \ell^j = p!\,\delta(l-j) = \begin{cases} p! & (j=p), \\ 0 & (j \neq p), \end{cases} \quad \text{for } 0 \leq j \leq p+n-1. \tag{B.3}$$

A solution to this system exists and can even be efficiently constructed (Fornberg 1988). Following the exact steps of (De Ryck et al. 2021, Lemma 3.1), but now using the above equation instead of equation (18) in (De Ryck et al. 2021) we find that that for all $1 \leq p \leq s$, p odd, we can find neural networks $\hat{f}_{p,h,n}$ such that for arbitrary $k \in \mathbb{N}$ it holds,

$$\left\| f_p - \hat{f}_{p,h} \right\|_{W^{k,\infty}} \leq C(\sigma, M, s, n, k) h^n =: \epsilon. \tag{B.4}$$

Note that $\{\hat{f}_{p,h,n} \ : \ 1 \leq p \leq s, \ p \text{ odd}\}$ is a shallow tanh neural network with $\ell = \frac{p+n-1}{2}$ neurons (where we used the symmetry of σ) and of which the weights grow as $\mathcal{O}(h^{-s}) = \mathcal{O}(\epsilon^{-s/n})$ for $\epsilon \to 0$. One can then follow the exact same steps of (De Ryck et al. 2021, Lemma 3.2 and Section 3.2) to generalize this result to multivariate polynomials of arbitrary degree, which leads to the statement of this lemma. □

Lemma B.2 (Shallow approximation of multiplication of d numbers)**.** *Let $d, n \in \mathbb{N}$, $k \in \mathbb{N}_0$ and $M > 0$. Then for every $\epsilon > 0$, there exists a shallow tanh neural network $\widehat{\times}_d^\epsilon : [-M, M]^d \to \mathbb{R}$ of width $3 \left\lceil \frac{d+n-1}{2} \right\rceil \left| P_{d,d} \right|$ such that*

$$\left\| \widehat{\times}_d^\epsilon(x) - \prod_{i=1}^{d} x_i \right\|_{W^{k,\infty}} \leq \epsilon. \tag{B.5}$$

Furthermore, the weights of the network scale as $O(\epsilon^{-d/n})$.

Proof. This is the counterpart of (De Ryck et al. 2021, Corollary 3.7), with the only difference that now the construction of Lemma B.1 is used. □

Lemma B.3. *It holds that* $\max\{\left|\sigma(x)\right|, \left|\sigma'(x)\right|, \left|\sigma''(x)\right|\} \leq 1$ *for all* $x \in \mathbb{R}$.

Next, we summarize the construction of an approximate partition of unity of a domain $\Omega = \prod_{i=1}^{d}[0, b_i]$, as in (De Ryck et al. 2021, Section 4). We divide the domain into cubes of edge length $1/N$ and denote the corresponding index set by

$$\mathcal{N}^N = \{j \in \mathbb{N}^d \ : \ j_i \leq N b_i \text{ for all } 1 \leq i \leq d\}. \tag{B.6}$$

We can then define the cubes for every $j \in \mathcal{N}^N$ as,

$$I_j^N = \bigtimes_{i=1}^{d} \left((j_i - 1)/N, j_i/N \right). \tag{B.7}$$

Observe that $\left|\sigma'\right|$ and $\left|\sigma''\right|$ are monotonously decreasing on $[1, \infty)$. Given $\epsilon > 0$, we first find an $\alpha = \alpha(N, \epsilon)$ large enough such that

$$\alpha/N \geq 1, \quad 1 - \sigma(\alpha/N) \leq \epsilon, \quad \alpha^m \left|\sigma^{(m)}(\alpha/N)\right| \leq \epsilon \text{ for } m = 1, 2. \tag{B.8}$$

A suitable choice of α is given by the following lemma.

Lemma B.4. *The conditions stated in* (B.8) *for* $0 < \epsilon < 1$ *are satisfied if*

$$\alpha = N \ln\left(\frac{4N^2}{e^2\epsilon}\right). \tag{B.9}$$

Proof. This is an adaptation of Lemma A.5 in (De Ryck et al. 2021) for $k = 2$. The proof is as in (De Ryck et al. 2021), except that one can use Lemma B.3 instead of (De Ryck et al. 2021, Lemma A.4). □

For $y \in \mathbb{R}$, we then define

$$\begin{aligned}
\rho_1^N(y) &= \frac{1}{2} - \frac{1}{2}\sigma\left(\alpha\left(y - \frac{1}{N}\right)\right), \\
\rho_j^N(y) &= \frac{1}{2}\sigma\left(\alpha\left(y - \frac{j-1}{N}\right)\right) - \frac{1}{2}\sigma\left(\alpha\left(y - \frac{j}{N}\right)\right) \quad \text{for } 2 \le j \le N-1, \\
\rho_N^N(y) &= \frac{1}{2}\sigma\left(\alpha\left(y - \frac{N-1}{N}\right)\right) + \frac{1}{2}.
\end{aligned} \tag{B.10}$$

Finally, we define for $D \le d$ the functions

$$\Phi_j^{N,D}(x) = \prod_{i=1}^{D} \rho_{j_i}^{N_i}(x_i) \tag{B.11}$$

and the sets $\mathcal{V}_D = \{v \in \mathbb{Z}^d : \max_{1\le i\le D} |v_i| \le 1 \text{ and } v_{D+1} = \cdots = v_d = 0\}$. The functions $\Phi_j^{N,d}$ approximate a partition of unity in the sense that for every j it holds on I_j^N that,

$$\sum_{v\in\mathcal{V}_d} \Phi_{j+v}^{N,d} \approx 1 \quad \text{and} \quad \sum_{\substack{v\notin\mathcal{V}_d,\\ j+v\in\{1,\dots,N\}^d}} \Phi_{j+v}^{N,d} \approx 0. \tag{B.12}$$

This is made exact in the following lemmas.

Lemma B.5 (Lemma 4.1 in (De Ryck et al. 2021))**.** *If* $k \in \mathbb{N}_0$ *and* $0 < \epsilon < 1/4$*, then*

$$\left\| \sum_{v\in\mathcal{V}_d} \Phi_{j+v}^{N,d} - 1 \right\|_{W^{k,\infty}(I_j^N)} \le 2^{kd} d\epsilon. \tag{B.13}$$

Lemma B.6. *Let* $k \in \{0, 1, 2\}$ *and* $v \in \mathbb{Z}^d$ *with* $\|v\|_\infty \ge 2$*. Then it holds that*

$$\left\| \Phi_{j+v}^{N,d} \right\|_{W^{k,\infty}(I_j^N)} \le \alpha^k \epsilon. \tag{B.14}$$

Proof. This is an adaptation of Lemma 4.2 in (De Ryck et al. 2021) for $k \le 2$. The proof is as in (De Ryck et al. 2021), except that one can use Lemma B.3 instead of (De Ryck et al. 2021, Lemma A.4). □

We can now present a generalization of (De Ryck et al. 2021, Theorem 5.1) where a parameter n can be freely chosen in order to control the network width and weights. For $n = 2$ one recovers (De Ryck et al. 2021, Theorem 5.1) exactly.

Theorem B.7. *Let* $d, n \ge 2$*,* $m \ge 3$*,* $\delta > 0$*,* $a_i, b_i \in \mathbb{Z}$ *with* $a_i < b_i$ *for* $1 \le i \le d$*,* $\Omega = \prod_{i=1}^d [a_i, b_i]$ *and* $f \in H^m(\Omega)$*. Then for every* $N \in \mathbb{N}$ *with* $N > 5$ *there exists a tanh neural network* $\widehat{f}^N$ *with two hidden layers, one of width at*

most $3\left\lceil\frac{m+n-2}{2}\right\rceil\left|P_{m-1,d+1}\right|+\sum_{i=1}^{d}(b_i-a_i)(N-1)$ *and another of width at most* $3\left\lceil\frac{d+n}{2}\right\rceil\left|P_{d+1,d+1}\right|N^d\prod_{i=1}^{d}(b_i-a_i)$*, such that for* $k\in\{0,1,2\}$ *it holds that,*

$$\left\|f-\widehat{f}^N\right\|_{H^k(\Omega)}\leq 2^k 3^d C_{k,m,d,f}\,(1+\delta)\ln^k\left(\beta_{k,\delta,d,f}N^{d+m+2}\right)N^{-m+k}, \tag{B.15}$$

and where we define

$$\beta_{k,\delta,d,f}=\frac{5\cdot 2^{kd}\max\{\prod_{i=1}^{d}(b_i-a_i),d\}\max\{\|f\|_{W^{k,\infty}(\Omega)},1\}}{3^d\delta\min\{1,C_{k,m,d,f}\}}, \tag{B.16}$$

$$C_{k,m,d,f}=\max_{0\leq\ell\leq k}\binom{d+\ell-1}{\ell}^{1/2}\frac{((m-\ell)!)^{1/2}}{\left(\left\lceil\frac{m-\ell}{d}\right\rceil!\right)^{d/2}}\left(\frac{3\sqrt{d}}{\pi}\right)^{m-\ell}|f|_{H^m}. \tag{B.17}$$

Moreover, the weights of $\widehat{f}^N$ *scale as* $O(N\ln(N)+N^\gamma)$ *with* $\gamma=\max\{m^2,d(2+m+d)\}/n$*.*

Proof. **Step 1: construction of the approximation.** We divide the domain Ω into cubes of edge length $1/N$ and denote the corresponding index set by

$$\mathcal{N}^N=\{j\in\mathbb{N}^d\,:\,j_i\leq N(b_i-a_i)\text{ for all }1\leq i\leq d\}. \tag{B.18}$$

Furthermore we write $T=\prod_{i=1}^{d}(b_i-a_i)$. As a result, $\left|\mathcal{N}^N\right|=TN^d$. Let us denote $J_j^N=\times_{i=1}^{d}\big((j_i-2)/N,(j_i+1)/N\big)$. We calculate that $\operatorname{diam}(J_j^N)=\frac{3\sqrt{d}}{N}$. As a consequence, the Bramble-Hilbert lemma (Lemma A.4) ensures the existence of a polynomial p_j^N of degree at most $m-1$ such that for all $0\leq\ell\leq m-1$ it holds that

$$\left|f-p_j^N\right|_{H^\ell(J_j^N)}\leq\binom{d+\ell-1}{\ell}^{1/2}\frac{((m-\ell)!)^{1/2}}{\left(\left\lceil\frac{m-\ell}{d}\right\rceil!\right)^{d/2}}\left(\frac{3\sqrt{d}}{\pi N}\right)^{m-\ell}|f|_{H^m}=:\frac{\mathcal{C}_\ell^*}{N^{m-\ell}}. \tag{B.19}$$

To simplify notation, we also define $\mathcal{C}_k:=\max_{0\leq\ell\leq k}\mathcal{C}_\ell^*$ and $p^N=\sum_j p_j^N\chi_j$, where χ_j denotes the indicator function on I_j^N. Next, let q_j^N be a tanh neural network as in Lemma B.1 (where we still leave $n\in\mathbb{N}$ undefined for the moment) such that

$$\left\|q_j^N-p_j^N\right\|_{W^{k,\infty}(\Omega)}\leq\eta\quad\text{and}\quad\left\|q_j^N-p_j^N\right\|_{H^k(\Omega)}\leq\eta. \tag{B.20}$$

In addition, we define

$$q_j^N(x)\widehat{\times}\Phi_j^{N,d}(x):=\widehat{\times}_{d+1}^h(q_j^N(x),\phi_{j_1}^{N,d}(x_1),\ldots,\phi_{j_d}^{N,d}(x_d)), \tag{B.21}$$

where $\widehat{\times}:=\widehat{\times}_{d+1}^h$ is the network from Corollary B.2 and $h=h(N)$ will be defined in the remainder of the proof. We then define our approximation as

$$\widehat{f}^N(x)=\sum_{j\in\mathcal{N}^N}q_j^N(x)\widehat{\times}\Phi_j^{N,d}(x). \tag{B.22}$$

Step 2: estimating the error of the approximation. The triangle inequality gives us

$$\begin{aligned}\left\|f-\widehat{f}^N\right\|_{H^k(\Omega)} \leq &\left\|f-\sum_{j\in\mathcal{N}^N} f\cdot\Phi_j^{N,d}\right\|_{H^k(\Omega)} + \left\|\sum_{j\in\mathcal{N}^N}(f-q_j^N)\cdot\Phi_j^{N,d}\right\|_{H^k(\Omega)}\\ &+\left\|\sum_{j\in\mathcal{N}^N}(q_j^N\cdot\Phi_j^{N,d}-q_j^N\widehat{\times}\Phi_j^{N,d})\right\|_{H^k(\Omega)}\end{aligned} \tag{B.23}$$

We proceed by bounding each term of the right hand side separately.

Step 2a: First term of (B.23). Let $i\in\mathcal{N}^N$ be arbitrary. Recalling that $\mathcal{V}_d=\{v\in\mathbb{Z}^d:\|v\|_\infty\leq 1\}$, we observe that for $k\in\{0,1,2\}$,

$$\begin{aligned}\left\|f-\sum_{j\in\mathcal{N}^N} f\cdot\Phi_j^{N,d}\right\|_{H^k(I_i^N)} &\leq 2^k\|f\|_{H^k(I_i^N)}\left\|1-\sum_{v\in\mathcal{V}_d}\Phi_{i+v}^{N,d}\right\|_{W^{k,\infty}(I_i^N)}\\ &\quad+2^k\|f\|_{H^k(I_i^N)}\left\|\sum_{\substack{j\in\mathcal{N}^N\\ j-i\notin\mathcal{V}_d}}\Phi_j^{N,d}\right\|_{W^{k,\infty}(I_i^N)}\\ &\leq 2^k\|f\|_{H^k(I_i^N)}(2^{kd}d\epsilon+\left|\mathcal{N}^N\right|\alpha^k\epsilon)\\ &\leq 2^{k(1+d)}\|f\|_{H^k(I_i^N)}d\epsilon\\ &\quad+2^k\|f\|_{H^k(I_i^N)}\left|\mathcal{N}^N\right|N^k\ln^k\left(\frac{4N^2}{e^2\epsilon}\right)\epsilon\\ &\leq 2^k3^d\frac{\delta}{4}\ln^k\left(\frac{4N^2}{e^2\epsilon}\right)\frac{\mathcal{C}_k}{N^{m-k}},\end{aligned} \tag{B.24}$$

where we used Lemma A.5 with $k=2$, Lemma B.5, Lemma B.6 and Lemma B.4, as well as a suitable definition of ϵ, e.g. satisfying

$$\epsilon\leq\frac{3^d\delta\mathcal{C}_k}{2^{3+k+kd}\max\{T,d\}N^{d+m}\|f\|_{H^k(\Omega)}}, \tag{B.25}$$

where we used that $N>5$.

Step 2b: Second term of (B.23). Let $\beta\in\mathbb{N}_0^d$ be such that $|\beta|\leq k$. Then as a consequence of the general Leibniz rule we find that

$$\left\|D^\beta\left(\sum_{v\in\mathcal{V}_d}(f-q_{i+v}^N)\Phi_{i+v}^{N,d}\right)\right\|_{L^2(I_i^N)}\leq\sum_{\beta'\leq\beta}\binom{\beta}{\beta'}\sum_{v\in\mathcal{V}_d}\left\|D^{\beta'}(f-q_{i+v}^N)\right\|_{L^2(I_i^N)}\left\|D^{\beta-\beta'}\Phi_{i+v}^{N,d}\right\|_{L^\infty(I_i^N)}. \tag{B.26}$$

For every $v\in\mathcal{V}_d$ and $\beta'\leq\beta$ with $\ell:=\left|\beta-\beta'\right|$, we can then use the bounds

$$\left\|D^{\beta'}(f-q_{i+v}^N)\right\|_{L^2(I_i^N)}\leq\left\|f-q_{i+v}^N\right\|_{H^{k-\ell}(I_i^N)}\leq\frac{\mathcal{C}_k}{N^{m-k+\ell}}+\eta, \tag{B.27}$$

which follows from (B.19) and (B.20), and,

$$\left\|D^{\beta-\beta'}\Phi_{i+v}^{N,d}\right\|_{L^\infty(I_i^N)}\leq N^\ell\ln^\ell\left(\frac{4N^2}{e^2\epsilon}\right), \tag{B.28}$$

which follows from Lemma B.3 and Lemma B.4. As $\sum_{\beta' \leq \beta} \binom{\beta}{\beta'} \leq 2^k$ (as a consequence of the multi-binomial theorem), we find that

$$\left\| \sum_{v \in \mathcal{V}_d} (f - q_{i+v}^N) \Phi_{i+v}^{N,d} \right\|_{W^{k,\infty}(I_i^N)} \leq 2^k 3^d \left(\frac{\mathcal{C}_k}{N^{m-k}} + \eta N^k \right) \ln^k \left(\frac{4N^2}{e^2 \epsilon} \right). \tag{B.29}$$

Combining this result with the triangle inequality, Lemma B.4, Lemma A.5, (B.19), (B.20), Lemma B.6 and the fact that $\ln(x) \leq \sqrt{x}$ for $x > 0$, we find that

$$\begin{aligned}
&\left\| \sum_{j \in \mathcal{N}^N} (f - q_j^N) \cdot \Phi_j^{N,d} \right\|_{H^k(I_i^N)} \\
&\leq \left\| \sum_{v \in \mathcal{V}_d} (f - q_{i+v}^N) \Phi_{i+v}^{N,d} \right\|_{H^k(I_i^N)} + \sum_{\substack{j \in \mathcal{N}^N \\ j - i \notin \mathcal{V}_d}} \left\| (f - q_j^N) \Phi_j^{N,d} \right\|_{H^k(I_i^N)} \\
&\leq \left\| \sum_{v \in \mathcal{V}_d} (f - q_{i+v}^N) \Phi_{i+v}^{N,d} \right\|_{H^k(I_i^N)} + \sum_{\substack{j \in \mathcal{N}^N \\ j - i \notin \mathcal{V}_d}} 2^k \left\| (f - q_j^N) \right\|_{H^k(I_i^N)} \left\| \Phi_j^{N,d} \right\|_{W^{k,\infty}(I_i^N)} \\
&\leq 2^k 3^d \left(\frac{\mathcal{C}_k}{N^{m-2}} + \eta N^k \right) \ln^k \left(\frac{4N^2}{e^2 \epsilon} \right) + 2^k \left| \mathcal{N}^N \right| (\mathcal{C}_k + \eta) N^k \ln^k \left(\frac{4N^2}{e^2 \epsilon} \right) \epsilon \\
&\leq 2^k 3^d \left(1 + \frac{\delta}{4} \right) \ln^k \left(\frac{4N^2}{e^2 \epsilon} \right) \frac{\mathcal{C}_k}{N^{m-k}},
\end{aligned} \tag{B.30}$$

where we obtain the last inequality by making a suitable choice of η and ϵ, satisfying

$$\eta \leq \frac{\delta \mathcal{C}_k}{8N^m} \quad \text{and} \quad \epsilon \leq \frac{3^d \delta}{4TN^{d+m}}. \tag{B.31}$$

Step 2d: Third term of (B.23). Finally, using the triangle inequality, Lemma A.6, Lemma B.1 and Lemma B.3 we obtain that for some $C > 0$ depending only on k and d,

$$\begin{aligned}
&\left\| \sum_{j \in \mathcal{N}^N} (q_j^N \cdot \Phi_j^{N,d} - q_j^N \widehat{\times} \Phi_j^{N,d}) \right\|_{H^k(I_i^N)} \leq \sqrt{\mu(\Omega)} \left\| \sum_{j \in \mathcal{N}^N} (q_j^N \cdot \Phi_j^{N,d} - q_j^N \widehat{\times} \Phi_j^{N,d}) \right\|_{W^{k,\infty}(I_i^N)} \\
&\leq \sqrt{\mu(\Omega)} \left| \mathcal{N}^N \right| C \cdot \left\| \widehat{\times}_{d+1}^h - \prod_{i=1}^{d+1} x_i \right\|_{W^{k,\infty}} \left(\left\| q_j^N \right\|_{W^{k,\infty}(\Omega)} + \left\| \rho_i^N \right\|_{W^{k,\infty}(\Omega)} \right)^k \\
&\leq \sqrt{\mu(\Omega)} \left| \mathcal{N}^N \right| C \cdot h \left(\left\| q_j^N \right\|_{W^{k,\infty}(\Omega)} + \alpha^k \right)^k \\
&\leq 2^k 3^d \frac{\delta}{4} \ln^k \left(\frac{4N^2}{e^2 \epsilon} \right) \frac{\mathcal{C}_k}{N^{m-k}},
\end{aligned} \tag{B.32}$$

where we obtain the last inequality by making a suitable choice of h, satisfying

$$h \leq \frac{3^d \delta \mathcal{C}_k}{4\sqrt{\mu(\Omega)} T N^{d+m-k} C \left(\left\| q_j^N \right\|_{W^{k,\infty}(\Omega)} + \alpha^k \right)^k}. \tag{B.33}$$

Step 2e: Final error bound. From (B.25) and (B.31) we find that a suitable definition of ϵ is given by

$$\epsilon = \frac{3^d \delta \min\{1, \mathcal{C}_k\}}{2^{3+kd} N^{m+d} \max\{T, d\} \max\{\|f\|_{W^{k,\infty}(\Omega)}, 1\}}. \tag{B.34}$$

Combining this observation with all previous steps of the proof then leads to the error bound

$$\left\| f - \widehat{f}^N \right\|_{W^{k,\infty}(\Omega)} \leq 2^k 3^d \, (1+\delta) \ln^k \left(\beta N^{d+m+2} \right) \frac{\mathcal{C}_k}{N^{m-k}}, \tag{B.35}$$

where $k = 2$ and where we define

$$\beta = \frac{5 \cdot 2^{kd} \max\{T, d\} \max\{\|f\|_{W^{k,\infty}(\Omega)}, 1\}}{3^d \delta \min\{1, \mathcal{C}_k\}}, \tag{B.36}$$

where we used that $2^5/e^2 \leq 5$.

Step 3: Estimating the network and weights sizes. The first hidden layer requires $3 \left\lceil \frac{s+n-2}{2} \right\rceil \left| P_{s-1,d+1} \right|$ neurons for the computation of all multivariate monomials (cf. Lemma B.1). For the computation of all $\rho_j^N(x_i)$ another $\sum_{i=1}^d (b_i - a_i)(N-1)$ neurons are needed in the first hidden layer. The second hidden layer needs at most $3 \left\lceil \frac{d+n}{2} \right\rceil \left| P_{d+1,d+1} \right|$ neurons for realizing $\widehat{\times}_{d+1}^h$, which needs to be performed $N^d \prod_{i=1}^d (b_i - a_i)$ times.

In the proof we achieved the wanted accuracy by making suitable choices of η, ϵ, h. From equation (B.34) and Lemma B.4, it follows that $\alpha = O\left(N \ln(N)\right)$.

For the approximate multiplication, (B.33) requires that $h^{-1} = O(N^{d+m+2})$. Corollary B.2 then proves that the weights of $\widehat{\times}_{d+1}^h$ grow as $O(N^{d(d+m+2)/n})$. Finally, the condition $\eta^{-1} = O(N^m)$ from (B.31) corresponds to weights growing as $O\left(N^{m^2/n}\right)$ as a consequence of Corollary B.1. This concludes the proof. □

Appendix C. Bounds on the derivative of a neural network

Lemma C.1. *Let $d, n, L, W \in \mathbb{N}$ and let $u_\theta : \mathbb{R}^{d+1} \to \mathbb{R}^{d+1}$ be a neural network with $\theta \in \Theta_{L,W,R}$ for $L \geq 2$, $R, W \geq 1$, cf. Definition 2.3. Assume that $\|\sigma\|_{C^n} \geq 1$. Then it holds for $1 \leq j \leq d+1$ that*

$$\left\| (u_\theta)_j \right\|_{C^n} \leq 16^L (d+1)^{2n} \left(e^2 n^4 W^3 R^n \|\sigma\|_{C^n} \right)^{nL} \tag{C.1}$$

Proof. Using the notation of Definition 2.3, we define the functions $F_k = \mathbb{R}^{l_{k-1}} \to \mathbb{R}$ for every $1 \leq k \leq L$ as,

$$F_k = f_L^\theta \circ f_{L-1}^\theta \circ \cdots \circ f_k^\theta, \tag{C.2}$$

and note that $F_1 = u_\theta$ and $F_L = f_L^\theta$. An application of (De Ryck et al. 2021, Lemma A.7) then brings us that

$$\|F_k\|_{C^n} \leq 16 (e^2 n^4 l_k l_{k-1}^2)^n \max_{1 \leq i \leq l_k} \left\| (f_k^\theta)_i \right\|_{C^n}^n \|F_{k+1}\|_{C^n}. \tag{C.3}$$

For $R \geq 1$ and $1 \leq k < L$ we find that $\left\| (f_k^\theta)_i \right\|_{C^n} \leq R^n \|\sigma\|_{C^n}$ for every i and for $k = L$ we find that $\left\| (f_L^\theta)_i \right\|_{C^n} \leq R(W \|\sigma\|_{C^0} + 1)$. Combining these inequalities

recursively gives us

$$\begin{aligned}\|F_1\|_{C^n} &\leq \|F_L\|_{C^n} \prod_{k=1}^{L-1} \left[16(e^2n^4l_kl_{k-1}^2)^n \max_{1\leq i\leq l_k} \left\|(f_k^\theta)_i\right\|_{C^n}^n\right] \\ &\leq R(W\|\sigma\|_{C^0}+1)\left[16(e^2n^4W^3R^n\|\sigma\|_{C^n})^n\right]^{L-1}(d+1)^{2n} \\ &\leq 16^L(d+1)^{2n}\left(e^2n^4W^3R^n\|\sigma\|_{C^n}\right)^{nL}.\end{aligned} \tag{C.4}$$

This concludes the proof of the lemma as $F_1 = u_\theta$. □