Physics informed neural networks for elliptic equations with oscillatory differential operators

Anonymous authors
Paper under double-blind review

Abstract

Physics informed neural network (PINN) based solution methods for differential equations have recently shown success in a variety of scientific computing applications. Several authors have reported difficulties, however, when using PINNs to solve equations with multiscale features. The objective of the present work is to illustrate and explain the difficulty of using standard PINNs for the particular case of elliptic partial differential equations (PDEs) with oscillatory coefficients. We show that if the coefficient in the elliptic operator contains frequencies on the order of $1/\epsilon$, then the Frobenius norm of the neural tangent kernel matrix associated to the loss function grows as $1/\epsilon^2$. This implies that as the separation of scales in the problem increases, training the neural network with gradient descent based methods to achieve an accurate approximation of the solution to the PDE becomes increasingly difficult. Numerical examples illustrate the stiffness of the optimization problem.

1 Introduction

Recent developments in deep learning have shown great promise for advancing computational and applied mathematics (Weinan, 2021). Physics informed neural networks (PINNs) have recently emerged as a popular method for scientific computation. Building off earlier work Lagaris et al. (1998); Psichogios & Ungar (1992), PINNs were introduced in Raissi et al. (2019) and seek to approximate the true solution of a differential equation by a neural network $u(x;\theta)$ parameterized by weights and biases θ. Consider the following general partial differential equation (PDE) defined on some $\Omega \subset \mathbb{R}^n$ by a differential operator $\mathcal{N}$

$$\begin{aligned} \mathcal{N}[u](x) &= f(x), \qquad x \in \Omega \\ u(x) &= g(x), \qquad x \in \partial\Omega, \end{aligned} \tag{1}$$

where x and Ω represent some general space-time coordinate and domain, respectively. Here and throughout this paper, the boundary conditions are taken to be of Dirichlet type. By the uniform approximation theorem (Hornik et al., 1989; Leshno et al., 1993), given sufficient data, a neural network can uniformly approximate classical smooth solutions to (1) whenever they exist.

In practice, the network parameters θ are determined by minimizing a loss function $\mathcal{L}$ that enforces (1) to hold for a set of N_c collocation and N_b boundary points, $\{x_i\}_{i=1}^{N_c} \in \Omega$ and $\{s_i\}_{i=1}^{N_b} \in \partial\Omega$, respectively. The loss is

$$\mathcal{L}(\theta) = \frac{1}{N_c}\sum_{i=1}^{N_c}\frac{1}{2}\Big|\mathcal{N}[u](x_i;\theta) - f(x_i)\Big|^2 + \frac{\lambda}{N_b}\sum_{i=1}^{N_b}\frac{1}{2}\Big|u(s_i;\theta) - g(s_i)\Big|^2, \tag{2}$$

where λ is a tunable parameter that weighs the relative importance of the boundary conditions. We refer to Karniadakis et al. (2021) for a review of this methodology applied in a wide range of contexts in scientific computing.

Despite the success of PINNs in a wide variety of applications, numerous authors have reported difficulties applying the technique to problems with multiscale features. Some examples include a scalar, nonlinear hyperbolic equation from a model of two-phase immiscible fluid flow in porous media (Fuks & Tchelepi, 2020), which can support shock waves, and systems of ordinary differential equations governing chemical kinetics (Ji et al., 2021), which exhibit stiff dynamics that evolve over a wide range of time scales. Difficulties have also been reported for the Helmholtz equation (Wang et al., 2021a), as well as linear hyperbolic problems, for example for example the one-dimensional advection equation (Krishnapriyan et al., 2021) and the wave equation (Wang et al., 2021b).

The focus of the current work is to illustrate and explain the difficulty of using standard PINN solution methods for linear elliptic boundary value problems (BVPs) of the form

$$\begin{aligned} -\nabla \cdot \big(a^\epsilon(x)\nabla u^\epsilon(x)\big) &= f(x), \qquad x \in \Omega \\ u^\epsilon(x) &= g(x), \qquad x \in \partial\Omega, \end{aligned} \tag{3}$$

where the bounded and coercive coefficient tensor $a^\epsilon : \mathbb{R}^n \to \mathbb{R}^n$ is assumed to consist of entries that contain frequencies on the order of ϵ^{-1} for $0 < \epsilon \ll 1$. The measure of the domain $\Omega \subset \mathbb{R}^n$ is assumed to be $\mathcal{O}(1)$, and hence, (3) is a multiscale problem that models, for example, steady-state heat conduction in a composite material or porous media flow governed by Darcy's law.

Theoretical analysis and numerical experiments presented below illustrate that, whenever standard PINN architectures are used in conjunction with gradient descent based training for the multiscale problem (3), the resulting optimization problem becomes increasingly difficult as the scale separation in the BVP increases, i.e. as ϵ vanishes. After motivating the present study in Section 2, we show in Section 3 that the neural tangent kernel matrix associated with the PINN approximation to (3) has a Frobenius norm that becomes unbounded as $\epsilon \downarrow 0$. Numerical examples in Section 4 illustrate that during training, the ordinary differential equation that governs the evolution of the BVP residuals indeed becomes increasingly stiff as $\epsilon \downarrow 0$, translating to poor PINN performance for problems with a large separation of scales.

2 Motivation

The motivation for attempting to use physics informed neural network solutions to the oscillatory problem (3) is to investigate whether a connection can be established between asymptotic homogenization theory and the so-called "frequency principle" in deep learning.

Recall from mathematical homogenization theory (Bensoussan et al., 2011) that, under suitable conditions, the solution to (3) is well approximated as $\epsilon \downarrow 0$ by the solution to a homogenized equation

$$\begin{aligned} -\nabla \cdot \big(\overline{a}(x)\nabla \overline{u}(x)\big) &= f(x), \qquad x \in \Omega \\ \overline{u}(x) &= g(x), \qquad x \in \partial\Omega. \end{aligned}$$

Both the homogenized coefficients $\overline{a}$ and the solution $\overline{u}$ do not contain ϵ-scale oscillations; the latter approximates the large-scale, low-frequency features of the oscillatory function u^ϵ.

Additionally, when neural networks learn a target function, they are known to learn the low frequency components more rapidly than the large frequencies (Rahaman et al., 2019; Xu et al., 2020). This "frequency principle" was shown to hold for gradient descent training in Luo et al. (2021); Markidis (2021); empirically the result can be observed for PINNs applied to relatively simple problems (Wang et al., 2021b), even when more widely used optimizers are used for training, e.g. Adam (Kingma & Ba, 2014). Consider as a brief representative example the PINN solution to the one-dimensional Poisson BVP

$$-\frac{d^2}{dx^2}u(x) = \sin(x) + \sin(5x) + \sin(15x) + \sin(55x) =: f(x) \tag{4}$$

for $x \in (-\pi, \pi)$ with homogeneous Dirichlet boundary conditions $u(-\pi) = u(\pi) = 0$. For collocation points $\{x_i\}_{i=1}^{N_c} \in (-\pi, \pi)$ the loss function becomes

$$\mathcal{L}(\theta) = \frac{1}{N_c}\sum_{i=1}^{N_c}\frac{1}{2}\left|\frac{d^2}{dx^2}u(x_i;\theta) + f(x_i)\right|^2 + \frac{1}{4}\lambda\left(\left|u(-\pi;\theta)\right|^2 + \left|u(\pi;\theta)\right|^2\right). \tag{5}$$

Figure 1 shows the evolution (as a function of training iteration) of the complex modulus of

$$\widehat{e}_k = \widehat{u}_{\text{true}} - \widehat{u}_{\text{NN}} \tag{6}$$

where u_{true} is the true solution to the BVP, u_{NN} is the neural network approximation, and $\widehat{\cdot}$ denotes the discrete Fourier transform. A rolling average is used to make the trajectories more legible. The PINN solution is computed with a fully connected neural network with four hidden layers of sixty nodes each, $N_c = 512$ equispaced collocation points, and $\lambda = 80$; see Appendix A for a complete description of the training process that generates the final neural network output. One can clearly observe the frequency principle in this simple example; the low frequencies of the target BVP solution are more rapidly learned than the high frequencies. See also Markidis (2021) for another example in a two-dimensional Poisson problem, as well as Wang et al. (2021b) for mathematical analysis of this phenomenon for Poisson equations.

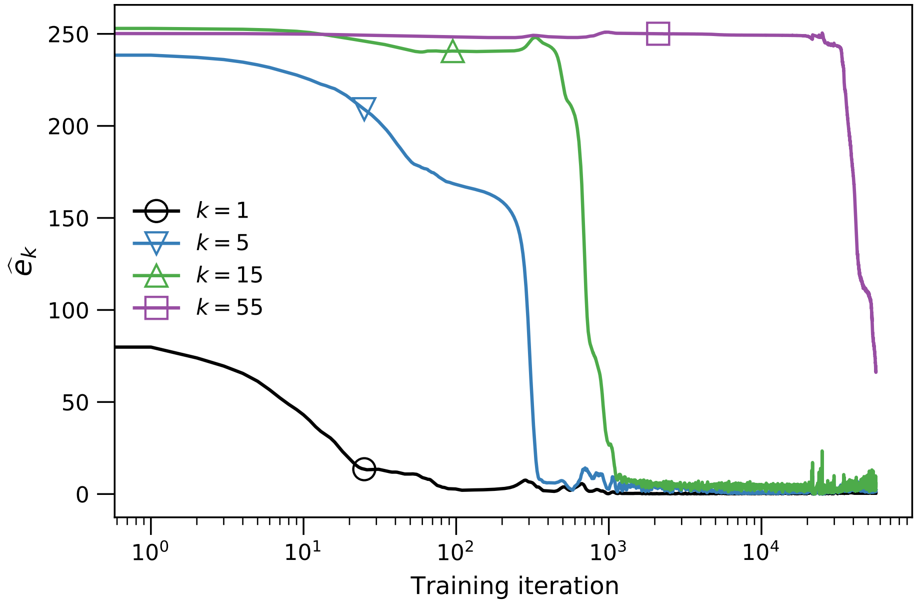

Figure 1: A simple illustration of the frequency principle: evolution as a function of training iteration of the magnitude of the discrete Fourier transform of the error (6) between the true solution to the BVP (4) and the neural network approximation.

Given some evidence that PINNs can learn low frequency features of PDE solutions, it is reasonable to ask if the homogenized solution 'naturally' arises when trying to learn solutions to the multiscale problem (3), and, if not, for what reason? The homogenized solution itself is of course useful in a variety of applications; it additionally could be used to construct coarse grid solutions for a multigrid solver of the full, oscillatory problem as in Engquist & Luo (1997). The natural smoothing properties of standard iterative solvers, e.g. the method of Gauss-Seidel or successive over-relaxation (Briggs et al., 2000), would then rapidly reduce the error of high-frequency solution components that are not easily approximated by the neural networks; such a 'hybrid' multigrid strategy was explored in Markidis (2021) for Poisson equations.

Recent theoretical results in Shin et al. (2020) guarantee that as the number of collocation N_c and boundary N_b points tend to infinity, the sequence of minimizers to a "Hölder regularized" version of the loss function (2) will converge to classical solutions of elliptic and parabolic PDEs (whenever they exist) in the limit of infinite learning data. Nevertheless, the authors in both Han & Lee (2023) and Leung et al. (2022) reported difficulties training neural networks to achieve such minimizers of the loss function associated to the multiscale problem (3). The purpose of the current work is to characterize the reason why optimization is difficult with both theory and numerical examples.

The theoretical results presented below describe the neural tangent kernel matrix of the PINN solution to the BVP (3); this matrix appears in the ordinary differential equation that, under gradient descent training dynamics, governs the evolution of the PDE and boundary residuals that define the PINN loss function. Under a few technical assumptions, we show that the matrix Frobenius norm becomes unbounded as $\epsilon \downarrow 0$. This result, along with numerical evidence presented in Section 4, suggests that as that as the scale separation in the differential operator in (3) increases, the optimization problem that determines the neural network approximation to the PDE solution becomes increasingly stiff.

3 Neural tangent kernel matrix theory

We now describe the neural tangent kernel (NTK) matrix for the physics informed neural network approximation to the multiscale elliptic equation (3) before showing that it can grow arbitrarily large in norm as $\epsilon \downarrow 0$. See Jacot et al. (2018) for the original development of the NTK theory for least-squares regression problems, as well as Wang et al. (2021b) and Wang et al. (2022) for an extension of the theory to PINNs.

For the presentation below it will be useful to define the linear multiscale differential operator

$$L^\epsilon \varphi := -\nabla \cdot \big(a^\epsilon \nabla \varphi\big),$$

where φ is some suitably regular function. The multiscale elliptic boundary value problem (3) then becomes

$$\begin{aligned} L^\epsilon u^\epsilon &= f \qquad \Omega \\ u^\epsilon &= g \qquad \partial\Omega. \end{aligned} \tag{7}$$

For simplicity, we assume throughout that the entries of the coefficient tensor a^ϵ are at least once continuously differentiable, and hence bounded on any compact domain Ω.

Given some neural network $u(x;\theta)$ parameterized by N_p weights and biases θ, as well as collocation $\{x_i\}_{i=1}^{N_c} \in \Omega$ and boundary points $\{s_i\}_{i=1}^{N_b} \in \partial\Omega$, the loss function associated to (7) becomes

$$\mathcal{L}^\epsilon(\theta) := \frac{1}{N_c}\sum_{i=1}^{N_c} \frac{1}{2}\Big|L^\epsilon u(x_i;\theta) - f(x_i)\Big|^2 + \frac{\lambda}{N_b}\sum_{i=1}^{N_b}\frac{1}{2}\Big|u(s_i;\theta) - g(s_i)\Big|^2. \tag{8}$$

If the residual values

$$r_{\text{pde}}(x_i;\theta) := L^\epsilon u(x_i;\theta) - f(x_i), \qquad i = 1,\dots,N_c \tag{9}$$

and

$$r_{\text{b}}(s_i;\theta) := u(s_i;\theta) - g(s_i), \qquad i = 1,\dots,N_b \tag{10}$$

are grouped together into a single vector $y(\theta)$, and if the parameters θ evolve according to the gradient flow

$$\frac{d\theta}{dt} = -\nabla_\theta \mathcal{L}^\epsilon, \tag{11}$$

then $y(\theta(t))$ will evolve according to an initial value problem of the form

$$\frac{d}{dt}y(t) = -K^\epsilon(y(t))\,y(t), \tag{12}$$

where K^ϵ is called the neural tangent kernel matrix. Note here that both θ and y implicitly depend on ϵ; however, the dependence is not explicitly marked.

We next derive (12), noting again that similar results can be found in Wang et al. (2021b) and Wang et al. (2022).

Lemma 3.1. *For neural network parameters $\theta \in \mathbb{R}^{N_p}$, let $y(\theta) \in \mathbb{R}^{N_c+N_b}$ be the vector of residual values*

$$y(\theta) = \big(r_{\text{pde}}(x_1;\theta),\dots,r_{\text{pde}}(x_{N_c};\theta), r_{\text{b}}(s_1;\theta),\dots,r_{\text{b}}(s_{N_b};\theta)\big)^T$$

where the entries r_{pde} *and* r_{b} *are defined by* (9) *and* (10)*. Suppose that the parameters* θ *evolve from some initial value* θ_0 *according to the gradient flow* (11)*. Then* $y(\theta(t))$ *evolves from the initial condition* $y(\theta_0)$ *according to*

$$\frac{d}{dt}y(t) = -K^{\epsilon}(y(t))\, y(t), \tag{13}$$

where the explicit dependence on θ *in* (13) *is dropped for convenience. The neural tangent kernel matrix is given by*

$$K^{\epsilon}(t) = \begin{pmatrix} K^{\epsilon}_{uu}(t) & K^{\epsilon}_{ub}(t) \\ K^{\epsilon}_{bu}(t) & K^{\epsilon}_{bb}(t) \end{pmatrix} \in \mathbb{R}^{(N_c+N_b)\times(N_c+N_b)}, \tag{14}$$

where the subblocks have entries

$$(K^{\epsilon}_{uu})_{ij}(t) = \frac{1}{N_c}\sum_{l=1}^{N_p} \frac{\partial}{\partial\theta_l} L^{\epsilon}u(x_i;\theta(t)) \frac{\partial}{\partial\theta_l} L^{\epsilon}u(x_j;\theta(t)), \qquad 1 \le i,j \le N_c,$$

$$(K^{\epsilon}_{bb})_{ij}(t) = \frac{\lambda}{N_b}\sum_{l=1}^{N_p} \frac{\partial}{\partial\theta_l} u(s_i;\theta(t)) \frac{\partial}{\partial\theta_l} u(s_j;\theta(t)), \qquad 1 \le i,j \le N_b,$$

and

$$(K^{\epsilon}_{ub})_{ij}(t) = \frac{\lambda}{N_b}\sum_{l=1}^{N_p} \frac{\partial}{\partial\theta_l} L^{\epsilon}u(x_i;\theta(t)) \frac{\partial}{\partial\theta_l} u(s_j;\theta(t)), \qquad 1 \le i \le N_c,\ 1 \le j \le N_b,$$

while K^{ϵ}_{bu} *is simply the scaled transpose of* K^{ϵ}_{ub}*:*

$$(K^{\epsilon}_{bu})_{ij} = \frac{N_b}{\lambda}\frac{1}{N_c}\,(K^{\epsilon}_{ub})_{ji}, \qquad 1 \le i \le N_c,\ 1 \le j \le N_b.$$

The proof is a computation that follows from the chain rule from differential calculus; it can be found in Appendix B.

Next we show that the NTK matrix (14) associated to the multiscale elliptic problem–in particular the K^{ϵ}_{uu}, K^{ϵ}_{ub} and K^{ϵ}_{bu} subblocks–can become arbitrarily large for vanishing ϵ. For simplicity we assume the problem is one-dimensional ($n = 1$) and that the multiscale coefficient $a^{\epsilon}(x) = a(x/\epsilon)$ is ϵ-periodic, but the result ought to hold for more general classes of coefficients so long as a^{ϵ} contains $\mathcal{O}(1/\epsilon)$ frequencies. After showing the desired result for $n = 1$, we briefly sketch below an additional proof for the higher dimension case $n > 1$ under additional technical assumptions.

In dimension $n = 1$, (7) becomes

$$\begin{aligned} -\frac{d}{dx}\left(a(x/\epsilon)\frac{d}{dx}\,u^{\epsilon}(x)\right) &= f(x), \qquad x \in (a,b) \\ u^{\epsilon}(a) = u_a, \qquad u^{\epsilon}(b) &= u_b. \end{aligned} \tag{15}$$

The loss function associated to (15) is equation (8) with $N_b = 2$ and elliptic operator L^{ϵ} given by

$$L^{\epsilon}u(x;\theta) = -\Big(a(x/\epsilon)\frac{d^2}{dx^2}\,u(x;\theta) + \frac{1}{\epsilon}a'(x/\epsilon)\frac{d}{dx}u(x;\theta)\Big), \tag{16}$$

where it is assumed the oscillatory function a is differentiable. The $1/\epsilon$ factor in the second term will be the source of the divergent behavior of the NTK as $\epsilon \downarrow 0$.

We first show the result for a neural network with one hidden layer before extending it to more general fully connected networks.

Theorem 3.2. *Let* $u : \mathbb{R} \to \mathbb{R}$ *be a neural network with one hidden layer of width* d *and smooth activation function* σ*, so that*

$$u(x;\theta) = \sum_{k=1}^{d} W^{(1)}_k \sigma\Big(W^{(0)}_k x + b^{(0)}_k\Big) + b^{(1)}, \tag{17}$$

where $b^{(1)} \in \mathbb{R}$, $W^{(1)} \in \mathbb{R}^{1\times d}$, *and* $W^{(0)}, b^{(0)} \in \mathbb{R}^{d\times 1}$, *and assume that* $\forall x \in \mathbb{R}$

$$0 < \sigma'(x) < \infty. \tag{18}$$

Let $a(y)$ *be a one-periodic, non-constant* C^1 *function, and let* $a^\epsilon(x) = a(x/\epsilon)$*. For a fixed number of collocation points* N_c*, let* K^ϵ *be the neural tangent kernel matrix* (14) *associated to the loss function* (8) *of the boundary value problem* (15)*, and suppose the network parameters* θ *evolve according to the gradient flow* (11) *for* $0 \le t \le T$ *for some* $T > 0$*.*

Suppose also that the parameters θ *evolving according to* (11) *are bounded uniformly in both* t *and* ϵ*:*

$$\sup_{0<\epsilon<\epsilon_0} \sup_{0\le t\le T} \|\theta(t)\|_\infty < C \tag{19}$$

for some $\epsilon_0 > 0$*. Finally, let* Υ *be the set of* $t \in [0,T]$ *for which there exists at least one entry of* $W^{(0)}$ *that is asymptotically larger than* ϵ*; that is,* $t \in \Upsilon$ *if and only if*

$$\lim_{\epsilon\downarrow 0} \epsilon / W_l^{(0)}(t) = 0$$

for some $1 \le l \le d$*. Then* $\forall t \in \Upsilon$

$$\lim_{\epsilon\downarrow 0} \|K^\epsilon(t)\|_F = \infty$$

where $\|\cdot\|_F$ *is the Frobenius norm.*

Before proving the result, we briefly remark on the theorem assumptions. First note that the number of collocation points N_c is assumed to be fixed. Since we will show that the entries of the NTK matrix are proportional to

$$\frac{1}{N_c}\frac{1}{\epsilon^2},$$

the singular nature of the evolution of the residuals (12) could in principle be prevented by taking $N_c \sim \epsilon^{-2}$. However, this can make the computational cost of gradient descent based optimization prohibitively expensive whenever the problem scale separation is sufficiently large.

Next, note the assumption (18) on $d\sigma/dx$ holds for sigmoidal-type activation functions commonly used for PINNs (Lu et al., 2021b), such as the hyberbolic tangent and logistic functions. Although it does not hold for the ReLU function $\sigma(x) = \max(0, x)$, these are of course not suitable for NN based solutions to PDEs with second order (or higher) differential operators.

Regarding the set $\Upsilon \subseteq [0,T]$, note that if $t \notin \Upsilon$, then each entry of $W^{(0)}$ will converge to 0 in the limit $\epsilon \downarrow 0$; in this scenario the neural network (17) would limit to a constant function and hence not be a suitable solution to (15) in general. Finally, if the uniform bound (19) on the network parameters θ did not hold, then an unstable, divergent training process could result (Wang et al., 2022); the bound helps prevent the so-called "vanishing gradient" problem that can occur when components of $\nabla_\theta \mathcal{L}^\epsilon$ vanish, preventing descent in that direction.

Proof of Theorem 3.2. The proof, like that of Lemma 3.1, is a consequence of direct computation; note that it is sufficient for a single entry of a matrix to diverge to ensure the Frobenius norm diverges. In particular we show that entries in the K^ϵ_{uu} subblock of the full NTK matrix K^ϵ scale as $1/(N_c\,\epsilon^2)$.

Indeed, recall from Lemma 3.1 that

$$(K^\epsilon_{uu})_{\alpha\alpha}(t) = \frac{1}{N_c}\sum_{\gamma=1}^{N_p}\Big(\frac{\partial}{\partial\theta_\gamma} L^\epsilon u(x_\alpha;\theta(t))\Big)^2 \tag{20}$$

for any $1 \le \alpha \le N_c$. Since (20) is a sum of squares, if just one entry in the sum blows up as $\epsilon \downarrow 0$, then the entire sum of course will as well. Below we drop the explicit dependence of the parameters θ on t for notational convenience.

First, let $g(x;\theta)$ be the $\mathbb{R}^d$ valued function whose k-th component equals

$$g_k(x;\theta) = \sigma\Big(W_k^{(0)} x + b_k^{(0)}\Big),$$

so that

$$u(x;\theta) = \sum_{k=1}^{d} W_k^{(1)} g_k(x;\theta)) + b^{(1)}.$$

By linearity,

$$\frac{\partial}{\partial W_\gamma^{(1)}} L^\epsilon u(x;\theta) = L^\epsilon g_\gamma(x;\theta)$$

for any $1 \le \gamma \le d$. So, to prove the desired result, it suffices to show that

$$\frac{1}{N_c}\big[L^\epsilon g_\gamma(x_\alpha;\theta)\big]^2 \to \infty \tag{21}$$

as $\epsilon \downarrow 0$. Using (16), the left-hand side of (21) equals

$$\frac{1}{N_c}\Big[\Big(a(x_\alpha/\epsilon)\frac{d^2}{dx^2} g_\gamma(x_\alpha;\theta)\Big)^2 + \frac{2}{\epsilon}\Big(a(x_\alpha/\epsilon)\frac{d^2}{dx^2} g_\gamma(x_\alpha;\theta)\, a'(x_\alpha/\epsilon)\frac{d}{dx} g_\gamma(x_\alpha;\theta)\Big) \\ + \frac{1}{\epsilon^2}\Big(a'(x_\alpha/\epsilon)\frac{d}{dx} g_\gamma(x_\alpha;\theta)\Big)^2\Big]. \tag{22}$$

Since a is a periodic C^1 function, both $a(x_\alpha/\epsilon)$ and $a'(x_\alpha/\epsilon)$ are bounded independent of $x_\alpha/\epsilon \in \mathbb{R}$. The derivatives of g_γ are

$$\frac{d}{dx}\, g_\gamma(x;\theta) = \sigma'\Big(W_\gamma^{(0)} x + b_\gamma^{(0)}\Big) W_\gamma^{(0)} \tag{23}$$

and

$$\frac{d^2}{dx^2}\, g_\gamma(x;\theta) = \sigma''\Big(W_\gamma^{(0)} x + b_\gamma^{(0)}\Big)\big(W_\gamma^{(0)}\big)^2. \tag{24}$$

Since the activation function is smooth and the network parameters are uniformly bounded in ϵ and t, both (23) and (24) are also bounded for all $x_\alpha \in [a,b]$. Consequently, (22) is dominated by the $1/\epsilon^2$ term for ϵ vanishingly small. The positivity assumption (18) on $d\sigma/dx$ and the uniform boundedness of the network parameters imply also that the (absolute value of the) first derivative (23) is bounded below by a constant independent of ϵ; this implies that for any $t \in \Upsilon$

$$\lim_{\epsilon\downarrow 0} \frac{1}{\epsilon}\left|\frac{d}{dx}\, g_l(x;\theta)\right| = \infty$$

holds for at least one $l \in \{1,\dots d\}$. Take $\gamma = l$, and let ϵ vanish monotonically to zero in such a way that $a'(x_\alpha/\epsilon) \neq 0$ for any ϵ (such a sequence exists since a is periodic and non-constant). Then (22) indeed limits to positive infinity, giving the desired result.

□

Without giving a formal proof, we now briefly describe sufficient conditions under which Theorem 3.2 could be extended to the more general case $n > 1$ (assuming one retains the positivity assumption (18) on $d\sigma/dx$ and the uniform boundedness assumption (19) on the network parameters). A neural network with one hidden layer of width d would be

$$u(x;\theta) = \sum_{k=1}^{d} W_k^{(1)} \sigma\Big(\sum_{l=1}^{n} W_{kl}^{(0)} x_l + b^{(0)}\Big) + b^{(1)};$$

here x_l denotes the l-th component of the function input $x \in \mathbb{R}^n$. Following the same argument just given, define

$$g_k(x;\theta) = \sigma\Big(\sum_{l=1}^{n} W_{kl}^{(0)} x_l + b^{(0)}\Big),$$

and assume that there exists some $1 \leq \mu \leq d$ and $1 \leq \nu \leq n$ such that $W^{(0)}_{\mu\nu}$ is asymptotically larger than ϵ (as before, if no such entries exist, then $W^{(0)}$ limits to the zero matrix as $\epsilon \downarrow 0$, and hence $u(x;\theta)$ limits to a constant function). The Frobenius norm of the PINN's neural tangent kernel matrix will blow-up if $[L^\epsilon g_\mu(x;\theta)]^2$ blows-up at some collocation point x. This can occur if the entries for which $\nabla \cdot a$ are nonzero coincide with the entries where $W^{(0)}$ is larger than $\mathcal{O}(\epsilon)$; more precisely, if there is a sequence $\epsilon \downarrow 0$ such that

$$\sum_{i=1}^{n} \frac{\partial a_{i\nu}}{\partial x_i}\Big(\frac{x}{\epsilon}\Big)\sigma'\Big(\sum_{l=1}^{n} W^{(0)}_{\mu l} x_l + b^{(0)}_\mu\Big) W^{(0)}_{\mu\nu} \neq 0$$

for any collocation point x, then the main result will generalize.

Returning to the case of one-dimension ($n = 1$), we now extend Theorem 3.2 to more general fully connected neural networks under an additional assumption on the network parameter's asymptotic behavior.

Theorem 3.3. *Let $u : \mathbb{R} \to \mathbb{R}$ be a fully connected neural network with Λ hidden layers of widths $d_1, d_2, \ldots, d_\Lambda$, so that*

$$\begin{aligned} u^{(0)}(x) &= x, \\ u^{(l)}(x) &= \sigma\big(W^{(l)} u^{(l-1)}(x) + b^{(l)}\big), \qquad 1 \leq l \leq \Lambda, \\ u(x;\theta) &= W^{(\Lambda+1)} \cdot u^{(\Lambda)}(x) + b^{(\Lambda+1)}. \end{aligned} \tag{25}$$

If all of the assumptions from Theorem 3.2 are retained, but Υ is now defined to be the set of $t \in [0,T]$ such that the magnitude of every entry of each matrix $W^{(l)}$, $1 \leq l \leq \Lambda$, are asymptotically larger than $\epsilon^{1/\Lambda}$, then $\forall t \in \Upsilon$

$$\lim_{\epsilon \downarrow 0} \|K^\epsilon(t)\|_F = \infty,$$

where K^ϵ is the NTK matrix associated to the loss function (8) *of the boundary value problem* (15).

The proof proceeds nearly identically to that of Theorem 3.2, and hence it is in Appendix C.

4 Numerical results

4.1 Scaling of the neural tangent kernel matrix

Theorems 3.2 and 3.3 state that the Frobenius norm of the neural tangent kernel (NTK) matrix K^ϵ associated to the PINN approximation of the multiscale BVP (15) should scale inversely proportional to $N_c\,\epsilon^2$ for gradient descent training with infinitesimal time steps. Here we numerically reproduce this scaling for a neural network trained in a more practical setting, namely with the Adam optimizer (Kingma & Ba, 2014).

As in Theorems 3.2 and 3.3, we focus on the K^ϵ_{uu} subblock of the NTK matrix. For a neural network with one hidden layer d, the ij-th entry of K^ϵ_{uu} is given by

$$\begin{aligned} \big(K^\epsilon_{uu}\big)_{ij}(t) = \sum_{k=1}^{d} \Big(&\frac{\partial}{\partial W^{(1)}_k} L^\epsilon u(x_i;\theta(t)) \frac{\partial}{\partial W^{(1)}_k} L^\epsilon u(x_j;\theta(t)) + \frac{\partial}{\partial W^{(0)}_k} L^\epsilon u(x_i;\theta(t)) \frac{\partial}{\partial W^{(0)}_k} L^\epsilon u(x_j;\theta(t)) \\ &+ \frac{\partial}{\partial b^{(0)}_k} L^\epsilon u(x_i;\theta(t)) \frac{\partial}{\partial b^{(0)}_k} L^\epsilon u(x_j;\theta(t)) \Big) \end{aligned}$$

(note $\partial L^\epsilon u(x;\theta)/\partial b^{(1)} = 0$). For network width $d = 50$ and $N_c = 256$ collocation points, Figure 2(a) shows the Frobenius norm $\|K^\epsilon_{uu}\|_F$ associated to (15) for the particular case that $[a,b] = [-\pi,\pi]$, $u_a = u_b = 0$, and $a^\epsilon(x) = 1/(2.1 + 2\sin(2\pi x/\epsilon))$. The norm is computed at time $t = 0$, i.e. at initialization, for a sequence of ϵ values between $1/10$ and $1/100$. For every ϵ value, the neural network parameters are initialized from the normal distribution with mean zero and unit variance. Figure 2(b) shows the Frobenius norms after initializing with the Glorot distribution (Glorot & Bengio, 2010) and then training for ten thousand iterations with the Adam optimizer at a learning rate of $\eta = 10^{-5}$. In both cases, the norm increases as $1/\epsilon^2$, consistent with the theory developed above.

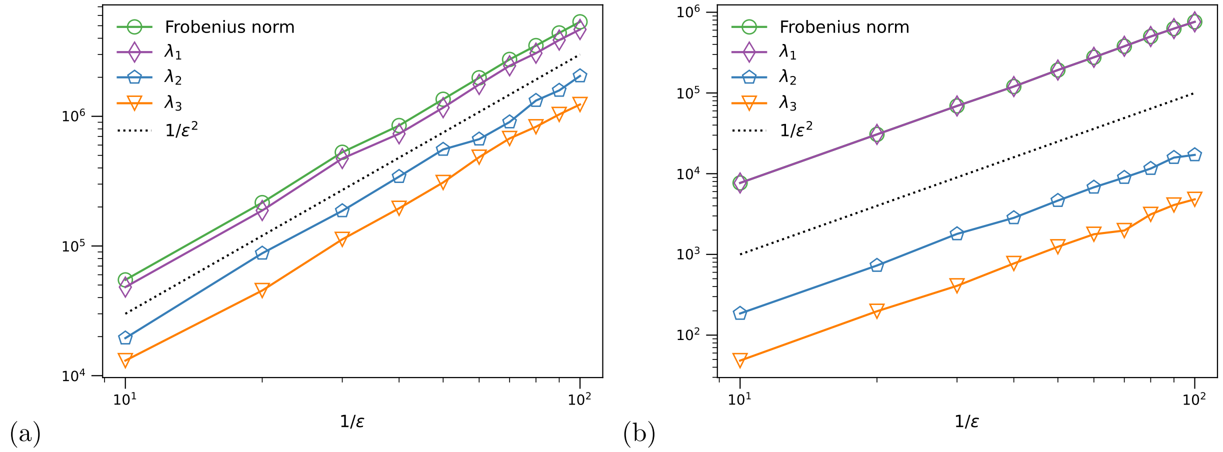

Figure 2: Scaling of the Frobenius norm of the NTK matrix and the first three principal eigenvalues (a) at initialization and (b) after training with Adam optimizer.

Recall that the Frobenius norm of a square matrix is equal to the Euclidean norm of its singular values. Since K^{ϵ}_{uu} is a symmetric matrix, its singular values equal the absolute value of its (real valued) eigenvalues. Hence, Theorems 3.2 and 3.3 imply that $\sum_{i=1}^{N_c}(\lambda_i^{\epsilon})^2 \to \infty$ as ϵ vanishes, where λ_i^{ϵ} are the eigenvalues of K^{ϵ}_{uu}. Figures 2(a) and 2(b) further suggest that, for the BVP just described, the first few principal eigenvalues (denoted λ_1, λ_2, and λ_3) also increase at a rate proportional to $1/\epsilon^2$, both at initialization and after training with the Adam optimizer. In contrast, the spectral radius of the K^{ϵ}_{bb} subblock (not shown) is essentially independent of ϵ.

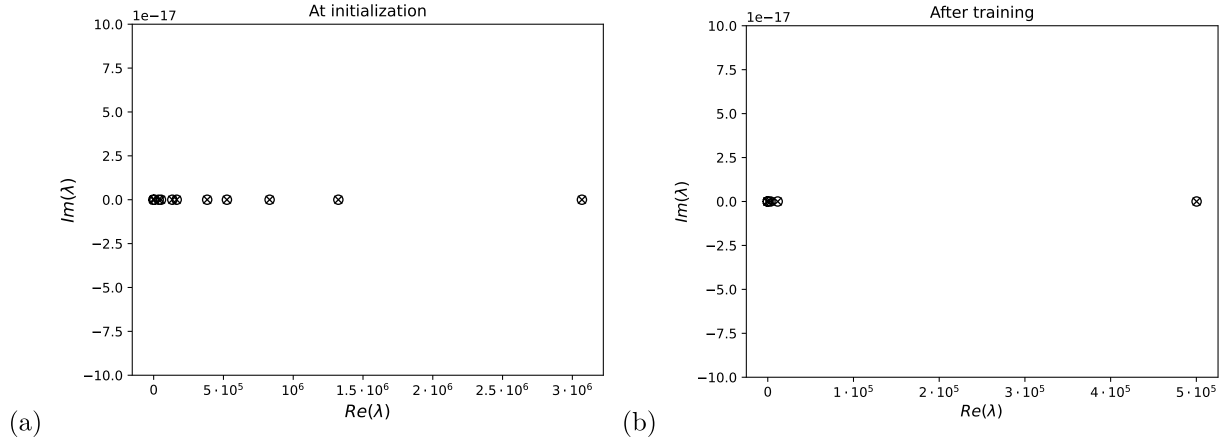

Figure 3: The eigenspectrum of the NTK (a) at initialization and (b) after training.

For the particular case $\epsilon = 1/80$, Figure 3 shows the full spectra of the NTK matrix both at initialization and after training. At initialization, there are a handful of large eigenvalues, but the rest are clustered around the origin; after training, there is a single large eigenvalue well-separated from the cluster near the origin. In both cases the ratio $\lambda^{\epsilon}_{\max}/\lambda^{\epsilon}_{\min}$ is quite large, indicating that (12) is a stiff initial value problem.

4.2 Learning a two-scale function

Consider next the two-scale function

$$u^{\epsilon}(x) = \sin(2x) + \epsilon\sin(x/\epsilon) - \frac{\epsilon}{\pi}\sin(\pi/\epsilon)\,x. \tag{26}$$

This function satisfies the Poisson BVP

$$-\frac{d^2}{dx^2}u^\epsilon(x) = 4\sin(2x) + \frac{1}{\epsilon}\sin(x/\epsilon), \qquad -\pi < x < \pi \tag{27}$$

with homogeneous Dirichlet boundary conditions $u^\epsilon(-\pi) = u^\epsilon(\pi) = 0$; it additionally satisfies the elliptic, Darcy BVP given by

$$-\frac{d}{dx}\Big(a(x/\epsilon)\frac{d}{dx}u^\epsilon(x)\Big) = f^\epsilon(x), \tag{28}$$

for $x \in (-\pi, \pi)$, again with homogeneous Dirichlet conditions, where

$$a(x/\epsilon) = \big(2.1 + 2\sin(x/\epsilon)\big)^{-1}$$

and the expression for the right-hand side forcing $f^\epsilon(x)$ is listed in Appendix D.

We describe three different neural network approximations to the two-scale function $u^\epsilon(x)$ by:

(1) regression (i.e. supervised learning),

(2) a PINN solver for the Poisson problem (27), and

(3) a PINN solver for the Darcy problem (28).

While the first two methods generate satisfactory approximations to (26), indicating that (a good approximation to) the target function lives in the "span" of the neural network used, the third method does not. All of the numerical tests are implemented in the open-source DeepXDE package (Lu et al., 2021b), and in each case, the neural network weights and biases are randomly initialized with the Glorot distribution (Glorot & Bengio, 2010).

Consider first approximating $u^\epsilon(x)$ with a neural network by regression, which we denote as $u_R(x;\theta)$. For $\{x_i\}_{i=1}^N \in [-\pi, \pi]$, the network parameters are determined by minimizing

$$\mathcal{L}^\epsilon(\theta) = \frac{1}{N}\sum_{i=1}^{N}\big(u_{\mathrm{R}}(x_i;\theta) - u^\epsilon(x_i)\big)^2.$$

We set $\epsilon = 1/32$ and use a neural network with hyperbolic tangent activation functions and four hidden layers, each of width $d = 40$. Using $N = \lceil 2\pi/(\epsilon/2)\rceil = 403$ equispaced training points, the network is trained with the Adam (Kingma & Ba, 2014) and L-BFGS (Byrd et al., 1995) optimizers for ten independent trials. In each test, Adam is used for $20,000$ and $10,000$ iterations with learning rates of $\eta = 10^{-2}$ and $\eta = 10^{-3}$, respectively, and L-BFGS is then used, on average, for approximately 3600 steps.

Let $\mathbb{E}\,u_{\mathrm{R}}(x;\theta)$ denote the neural network averaged over all ten trials. Taking $\{x_i\}_{i=1}^{1000}$ to be a set of equispaced points from $-\pi$ to π, Figures 4(a) and (b) show the resulting approximation and error, respectively. The maximum error is approximately

$$\max_{x_i}|e_{\mathrm{R}}| := \max_{x_i}\big|u^\epsilon(x_i) - \mathbb{E}\,u_{\mathrm{R}}(x_i;\theta)\big| \approx 2.65\cdot 10^{-3},$$

and the average variance in the error over the interval $[-\pi, \pi]$ is about $3\cdot 10^{-7}$.

Next, consider approximating the solution to the Poisson problem (27) with a PINN $u_{\mathrm{P}}(x;\theta)$. The loss function is

$$\mathcal{L}^\epsilon(\theta) = \frac{1}{N_c}\sum_{i=1}^{N_c}\frac{1}{2}\Big(\frac{d^2}{dx^2}u_{\mathrm{P}}(x_i;\theta) + (4\sin(2x_i) + \frac{1}{\epsilon}\sin(x_i/\epsilon))\Big)^2 + \frac{1}{2}\Big([u_{\mathrm{P}}(-\pi;\theta)]^2 + [u_{\mathrm{P}}(\pi;\theta)]^2\Big).$$

For $\epsilon = 1/32$ and same neural network as before, we train with $N_c = 1024$ equispaced collocation points for ten independent trials. The training schedule consists of $10,000$, $10,000$, $20,000$, and $20,000$ iterations of

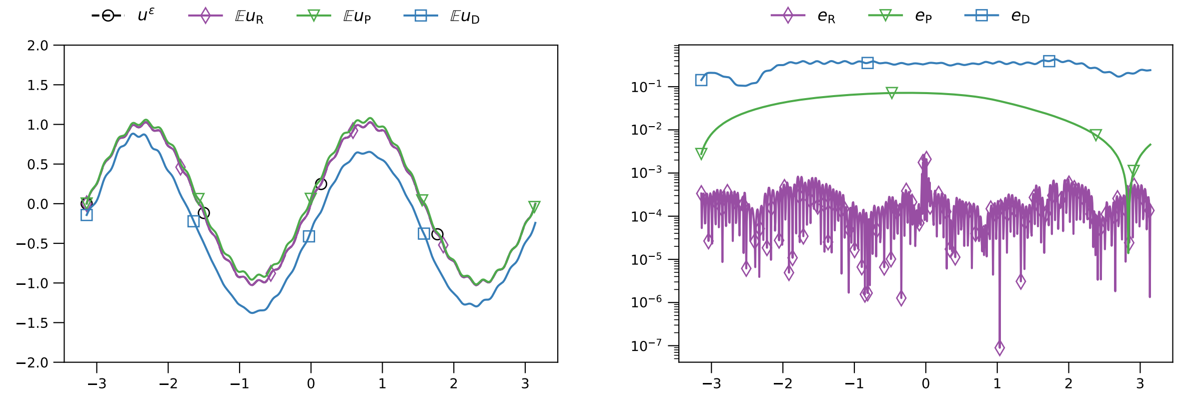

Figure 4: (a) For $\epsilon = 1/32$, the two-scale function $u^\epsilon(x)$ given by (26) and its approximations by regression ($\mathbb{E}\,u_\mathrm{R}$), a Poisson PINN ($\mathbb{E}\,u_\mathrm{P}$) and a Darcy PINN ($\mathbb{E}\,u_\mathrm{D}$); in each case the neural network contains four hidden layers of width $d = 40$. (b) The absolute value of corresponding generalization errors e_R, e_P and e_D at 1000 equispaced points from $-\pi$ to π.

Adam with learning rates of $\eta = 10^{-4}$, $\eta = 10^{-5}$, $\eta = 10^{-6}$, and $\eta = 10^{-7}$, respectively, and then L-BFGS for a few thousand iterations, on average.

Figures 4(a) and (b) show the resulting approximations and the generalization errors of the neural networks after training. Taking again $\mathbb{E}\,u_\mathrm{P}(x;\theta)$ to be the network averaged over all the trials, the maximum generalization error is

$$\max_{x_i} |e_\mathrm{P}| := \max_{x_i} |u^\epsilon(x_i) - \mathbb{E}\,u_\mathrm{P}(x_i;\theta)| \approx 7.20 \cdot 10^{-2},$$

while the average variance across the interval is about $1.12 \cdot 10^{-1}$.

Finally, consider the PINN approximation $u_\mathrm{D}(x;\theta)$ of the Darcy problem (28), where the loss function is

$$\mathcal{L}^\epsilon(\theta) = \frac{1}{N_c}\sum_{i=1}^{N_c} \frac{1}{2}\Big[\frac{d}{dx}\Big(a(x_i/\epsilon)\frac{d}{dx}u_\mathrm{D}(x_i;\theta)\Big) + f^\epsilon(x_i)\Big]^2 + \frac{1}{2}\Big([u_\mathrm{D}(-\pi;\theta)]^2 + [u_\mathrm{D}(\pi;\theta)]^2\Big).$$

Once again, $\epsilon = 1/32$, and the network architecture and number of collocation points is the same as the Poisson case. The training schedule consists of 10,000, 10,000, 20,000, and 20,000 iterations of Adam with learning rates of $\eta = 10^{-3}$, $\eta = 10^{-4}$, $\eta = 10^{-5}$, and $\eta = 10^{-6}$, respectively, and then L-BFGS for a few thousand iterations, on average. The maximum generalization error

$$\max_{x_i} |e_\mathrm{D}| := \max_{x_i} |u^\epsilon(x_i) - \mathbb{E}\,u_\mathrm{D}(x_i;\theta)| \approx 4.31 \cdot 10^{-1},$$

which is about 40%, and the average variance across the interval is nearly as large–about $2.83 \cdot 10^{-1}$. From Figures 4(a) and (b) it can be seen that the error contains both low and high frequency components.

5 Conclusions

Physics informed neural networks have demonstrated success in a wide variety of problems in scientific computing (Karniadakis et al., 2021), however, they can sometimes struggle to approximate solutions to differential equations with multiscale features. Athough PINN convergence theory guarantees that minimizers of (regularized) loss functions can well approximate solutions to elliptic problems, in practice, achieving those minimizers is difficult for boundary value problems with highly oscillatory coefficients present in the differential operator.

We show that for a class of linear, multiscale elliptic equations, the Frobenius norm of the neural tangent kernel matrix associated to the PINN becomes unbounded as the characteristic wavelength of the oscillations vanishes. Numerical examples illustrate that during training, the ordinary differential equation that governs the evolution of the PDE residuals becomes increasingly stiff as $\epsilon \downarrow 0$, translating to poor PINN training behavior.

The present work considers standard, fully connected neural network architectures; it may be possible in future research to develop alternative PINN architectures that are specifically adapted to the problem. Future work will likely also benefit from more sophisticated learning techniques beyond (stochastic) gradient descent, for example genetic algorithms (Mirjalili, 2019). Another possibility is to employ continuation methods (Allgower & Georg, 2003), i.e. "curriculum regularization" (Krishnapriyan et al., 2021).

Recent results establishing a "frequency principle" in machine learning motivated this study; during training, neural networks tend to learn their target functions from low to high frequency (Luo et al., 2021). See also the work by Wang et al. (2021b) for spectral analysis of physics informed neural networks applied to Poisson problems with multiscale forcing. An interesting avenue for future study is to characterize the spectral behavior during training of deep learning methods for learning operators; see for example (Li et al., 2020; Khoo et al., 2021; Bhattacharya et al., 2021; Lu et al., 2021a; Zhang et al., 2022). In the context of the present work, this would be the nonlinear map from the problem data a^ϵ and f in (3) to the solution u^ϵ. Finally, we note many other novel neural network approaches to multiscale elliptic problems not discussed here have recently been proposed, for example (Han & Lee, 2023; Leung et al., 2022; Fabian et al., 2022). We hope some of the results presented here might inspire insight into these exciting approaches.

References

Eugene L Allgower and Kurt Georg. *Introduction to Numerical Continuation Methods.* SIAM, 2003.

Alain Bensoussan, Jacques-Louis Lions, and George Papanicolaou. *Asymptotic analysis for periodic structures*, volume 374. American Mathematical Soc., 2011.

Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model Reduction And Neural Networks For Parametric PDEs. *The SMAI journal of computational mathematics*, 7:121–157, 2021.

William L Briggs, Van Emden Henson, and Steve F McCormick. *A multigrid tutorial.* SIAM, 2000.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

Bjorn Engquist and Erding Luo. Convergence of a multigrid method for elliptic equations with highly oscillatory coefficients. *SIAM Journal on Numerical Analysis*, 34(6):2254–2273, 1997.

Kröpfl Fabian, Roland Maier, and Daniel Peterseim. Operator compression with deep neural networks. *Advances in Difference Equations*, 2022(1), 2022.

Olga Fuks and Hamdi A Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1), 2020.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Jihun Han and Yoonsang Lee. A neural network approach for homogenization of multiscale problems. *Multiscale Modeling & Simulation*, 21(2):716–734, 2023.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Weiqi Ji, Weilun Qiu, Zhiyu Shi, Shaowu Pan, and Sili Deng. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *The Journal of Physical Chemistry A*, 125(36):8098–8106, 2021.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. *European Journal of Applied Mathematics*, 32(3):421–435, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

Wing Tat Leung, Guang Lin, and Zecheng Zhang. NH-PINN: Neural homogenization-based physics-informed neural network for multiscale problems. *Journal of Computational Physics*, 470:111539, 2022.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2020.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021a.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021b.

Tao Luo, Zheng Ma, Zhi-Qin John Xu, and Yaoyu Zhang. Theory of the frequency principle for general deep neural networks. *CSIAM Transactions on Applied Mathematics*, 2(3):484–507, 2021.

Stefano Markidis. The old and the new: Can physics-informed deep-learning replace traditional linear solvers? *Frontiers in big Data*, pp. 92, 2021.

Seyedali Mirjalili. Genetic algorithm. In *Evolutionary algorithms and neural networks*, pp. 43–55. Springer, 2019.

Dimitris C Psichogios and Lyle H Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5301–5310. PMLR, 09–15 Jun 2019.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Yeonjong Shin, Jerome Darbon, and George Em Karniadakis. On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs. *Communications in Computational Physics*, 28(5):2042–2074, 2020.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021a.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021b.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

E Weinan. The dawning of a new era in applied mathematics. *Notices of the American Mathematical Society*, 68(4):565–571, 2021.

Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 28(5):1746–1767, 2020.

Lulu Zhang, Tao Luo, Yaoyu Zhang, Weinan E, Zhi-Qin John Xu, and Zheng Ma. MOD-Net: A Machine Learning Approach via Model-Operator-Data Network for Solving PDEs. *Communications in Computational Physics*, 32(2):299–335, 2022.

A Description of PINN approximation for multiscale Poisson problem

Here we give a more complete description of the example from Section 2 of a PINN approximation to the Poisson boundary value problem (4) with homogeneous Dirichlet boundary conditions.

The PINN solution is computed with a fully connected neural network with four hidden layers of sixty nodes each and hyperbolic tangent activation functions. The training consisted of 100, $1,000$, $15,000$, $30,000$ and $10,000$ iterations with the Adam optimizer with learning rates of $\eta = 10^{-2}$, $\eta = 10^{-3}$, $\eta = 10^{-4}$, $\eta = 10^{-5}$, and $\eta = 10^{-6}$, respectively. We note, however, using a constant value of $\eta = 10^{-3}$ throughout training produced results qualitatively similar to those in Figure 1; i.e. the observed frequency principle was fairly robust to changes in the learning rate. As mentioned in the description in Section 2, for legibility, the results shown in Figure 1 are that of the rolling average (of 50 values) to the trajectories of the error in the Fourier coefficients.

Finally, we mention that at the end of the training process, the ∞-norm error between the true solution and the neural network approximation is about $1.26 \cdot 10^{-2}$.

B Proof of Lemma 3.1

Proof. Computing first the entries of the gradient of the loss function (8) gives

$$\frac{\partial \mathcal{L}^\epsilon}{\partial \theta_l} = \frac{1}{N_c} \sum_{j=1}^{N_c} r_{\text{pde}}(x_j;\theta) \frac{\partial}{\partial \theta_l} L^\epsilon u(x_j;\theta) + \frac{\lambda}{N_b} \sum_{j=1}^{N_b} r_{\text{b}}(s_j;\theta) \frac{\partial}{\partial \theta_l} u(s_j;\theta). \tag{29}$$

Next, compute

$$\begin{aligned} \frac{d}{dt} r_{\text{pde}}(x_i;\theta(t)) = \frac{d}{dt} \left(L^\epsilon u(x_i;\theta(t)) - f(x_i) \right) &= \sum_{l=1}^{N_p} \frac{\partial}{\partial \theta_l} L^\epsilon u(x_i;\theta) \frac{d\theta_l}{dt} \\ &= -\sum_{l=1}^{N_p} \frac{\partial}{\partial \theta_l} L^\epsilon u(x_i;\theta) \frac{\partial \mathcal{L}^\epsilon}{\partial \theta_l}, \end{aligned} \tag{30}$$

where the final equality comes from the assumption of gradient flow dynamics (11). A similar computation gives

$$\begin{aligned}\frac{d}{dt}r_{\mathrm{b}}(s_i;\theta(t)) &= \frac{d}{dt}\left(u(s_i;\theta(t)) - g(s_i)\right) = \sum_{l=1}^{N_p}\frac{\partial}{\partial\theta_l}u(s_i;\theta)\frac{d\theta_l}{dt}\\ &= -\sum_{l=1}^{N_p}\frac{\partial}{\partial\theta_l}u(s_i;\theta)\frac{\partial\mathcal{L}^\epsilon}{\partial\theta_l},\end{aligned} \tag{31}$$

Inserting (29) into (30) and rearranging the sums gives

$$\begin{aligned}\frac{d}{dt}r_{\mathrm{pde}}(x_i;\theta(t)) = -\Big(&\sum_{j=1}^{N_c}\Big[\frac{1}{N_c}\sum_{l=1}^{N_p}\frac{\partial}{\partial\theta_l}L^\epsilon u(x_i;\theta)\frac{\partial}{\partial\theta_l}L^\epsilon u(x_j;\theta)\Big]r_{\mathrm{pde}}(x_j;\theta)\\ &+\sum_{j=1}^{N_b}\Big[\frac{\lambda}{N_b}\sum_{l=1}^{N_p}\frac{\partial}{\partial\theta_l}L^\epsilon u(x_i;\theta)\frac{\partial}{\partial\theta_l}u(s_j;\theta)\Big]r_{\mathrm{b}}(s_j;\theta)\Big),\end{aligned}$$

while inserting (29) into (31) similarly results in

$$\begin{aligned}\frac{d}{dt}r_{\mathrm{b}}(s_i;\theta(t)) = -\Big(&\sum_{j=1}^{N_c}\Big[\frac{1}{N_c}\sum_{l=1}^{N_p}\frac{\partial}{\partial\theta_l}u(s_i;\theta)\frac{\partial}{\partial\theta_l}L^\epsilon u(x_j;\theta)\Big]r_{\mathrm{pde}}(x_j;\theta)\\ &+\sum_{j=1}^{N_b}\Big[\frac{\lambda}{N_b}\sum_{l=1}^{N_p}\frac{\partial}{\partial\theta_l}u(s_i;\theta)\frac{\partial}{\partial\theta_l}u(s_j;\theta)\Big]r_{\mathrm{b}}(s_j;\theta)\Big)\end{aligned}$$

as desired.

□

C Proof of Theorem 3.3

Proof. From (25) we note that $W^{(1)} \in \mathbb{R}^{d_1\times 1}$, $W^{(l)} \in \mathbb{R}^{d_l\times d_{l-1}}$ for $l = 2,\dots,\Lambda$, and $W^{(\Lambda+1)} \in \mathbb{R}^{1\times d_\Lambda}$, while $b^{(l)} \in \mathbb{R}^{d_l}$ for $l = 1,\dots,\Lambda$ and $b^{(\Lambda+1)} \in \mathbb{R}$. The fully connected neural network (25) can then be written as

$$\begin{aligned}u(x;\theta) &= \sum_{k_\Lambda=1}^{d_\Lambda}W^{(\Lambda+1)}_{k_\Lambda}u^{(\Lambda)}_{k_\Lambda}(x) + b^{(\Lambda+1)},\\ u^{(\Lambda)}_{k_\Lambda}(x) &= \sigma\Big(\sum_{k_{\Lambda-1}=1}^{d_{\Lambda-1}}W^{(\Lambda)}_{k_\Lambda k_{\Lambda-1}}u^{(\Lambda-1)}_{k_{\Lambda-1}}(x) + b^{(\Lambda)}_{k_\Lambda}\Big), \qquad 1\le k_\Lambda\le d_\Lambda,\\ u^{(\Lambda-1)}_{k_{\Lambda-1}}(x) &= \sigma\Big(\sum_{k_{\Lambda-2}=1}^{d_{\Lambda-2}}W^{(\Lambda-1)}_{k_{\Lambda-1}k_{\Lambda-2}}u^{(\Lambda-2)}_{k_{\Lambda-2}}(x) + b^{(\Lambda-1)}_{k_{\Lambda-1}}\Big), \qquad 1\le k_{\Lambda-1}\le d_{\Lambda-1},\\ &\vdots\\ u^{(2)}_{k_2}(x) &= \sigma\Big(\sum_{k_1=1}^{d_1}W^{(2)}_{k_2k_1}u^{(1)}_{k_1}(x) + b^{(2)}_{k_2}\Big), \qquad 1\le k_2\le d_2\\ u^{(1)}_{k_1}(x) &= \sigma\Big(W^{(1)}_{k_1}x + b^{(1)}_{k_1}\Big), \qquad 1\le k_1\le d_1.\end{aligned}$$

By linearity, note that

$$\frac{\partial}{\partial W^{(\Lambda+1)}_\gamma}L^\epsilon u(x;\theta) = L^\epsilon u^{(\Lambda)}_\gamma(x) \tag{32}$$

for any $1 \leq \gamma \leq d_\Lambda$.

As before, consider a diagonal entry in the K^ϵ_{uu} subblock of the NTK matrix (14)

$$(K^\epsilon_{uu})_{\alpha\alpha}(t) = \frac{1}{N_c} \sum_{\gamma=1}^{N_p} \frac{\partial}{\partial \theta_\gamma} L^\epsilon u(x_\alpha; \theta(t)) \frac{\partial}{\partial \theta_\gamma} L^\epsilon u(x_\alpha; \theta(t)).$$

By (32), a sufficient condition for the Frobenius norm of K^ϵ to blow-up, then, is for

$$\lim_{\epsilon \downarrow 0} \left(L^\epsilon u^{(\Lambda)}_\gamma (x_\alpha) \right)^2 = \infty$$

for some collocation point x_α, some $1 \leq \gamma \leq d_\Lambda$, and $t \in \Upsilon$ (here the dependence of $u^{(\Lambda)}_\gamma$ on t and θ is implied). Since

$$\begin{aligned}\left(L^\epsilon u^{(\Lambda)}_\gamma (x_\alpha) \right)^2 = \Big[\Big(a(x_\alpha/\epsilon) \frac{d^2}{dx^2}\, u^{(\Lambda)}_\gamma (x_\alpha) \Big)^2 &+ \frac{2}{\epsilon} \Big(a(x_\alpha/\epsilon) \frac{d^2}{dx^2}\, u^{(\Lambda)}_\gamma (x_\alpha)\, a'(x_\alpha/\epsilon) \frac{d}{dx}\, u^{(\Lambda)}_\gamma (x_\alpha) \Big) \\ &+ \frac{1}{\epsilon^2} \Big(a'(x_\alpha/\epsilon) \frac{d}{dx}\, u^{(\Lambda)}_\gamma (x_\alpha) \Big)^2 \Big], \end{aligned} \tag{33}$$

and since a is a periodic C^1 function, the desired result will follow if (i) the second derivative of $u^{(\Lambda)}_\gamma$ is uniformly bounded in ϵ and (ii) the first derivative is asymptotically larger than ϵ, so that

$$\lim_{\epsilon \downarrow 0} \frac{1}{\epsilon} \left| \frac{d}{dx}\, u^{(\Lambda)}_\gamma (x) \right| = \infty \tag{34}$$

for any $t \in \Upsilon$.

The first derivative is

$$\frac{d}{dx} u^{(\Lambda)}_\gamma (x) = \sigma' \Big(\sum_{k_{\Lambda-1}=1}^{d_{\Lambda-1}} W^{(\Lambda)}_{\gamma k_{\Lambda-1}} u^{(\Lambda-1)}_{k_{\Lambda-1}}(x) + b^{(\Lambda)}_{k_\Lambda} \Big) \sum_{k_{\Lambda-1}=1}^{d_{\Lambda-1}} \Big[W^{(\Lambda)}_{\gamma k_{\Lambda-1}} \frac{d}{dx} u^{(\Lambda-1)}_{k_{\Lambda-1}}(x) \Big] \tag{35}$$

after applying the chain rule once. Using Einstein summation notation for brevity, (35) simplifies to

$$\frac{d}{dx} u^{(\Lambda)}_\gamma (x) = \sigma' \Big(W^{(\Lambda)}_{\gamma k_{\Lambda-1}} u^{(\Lambda-1)}_{k_{\Lambda-1}}(x) + b^{(\Lambda)}_{k_\Lambda} \Big) \Big[W^{(\Lambda)}_{\gamma k_{\Lambda-1}} \frac{d}{dx} u^{(\Lambda-1)}_{k_{\Lambda-1}}(x) \Big].$$

Continuing with the chain rule (and the Einstein notation), we get

$$\begin{aligned}\frac{d}{dx} u^{(\Lambda)}_\gamma (x) = \sigma' \Big(W^{(\Lambda)}_{\gamma k_{\Lambda-1}} u^{(\Lambda-1)}_{k_{\Lambda-1}}(x) + b^{(\Lambda)}_{k_\Lambda} \Big) \times& \\ \Big[W^{(\Lambda)}_{\gamma k_{\Lambda-1}} \sigma' \Big(W^{(\Lambda-1)}_{k_{\Lambda-1} k_{\Lambda-2}} u^{(\Lambda-2)}_{k_{\Lambda-2}}(x) + b^{(\Lambda-1)}_{k_{\Lambda-1}} \Big) \Big[W^{(\Lambda-1)}_{k_{\Lambda-1} k_{\Lambda-2}} \frac{d}{dx} u^{(\Lambda-2)}_{k_{\Lambda-2}}(x) \Big] \Big]&.\end{aligned}$$

From here we can recurse downwards to compute the full derivative as

$$\begin{aligned}\frac{d}{dx} u^{(\Lambda)}_\gamma (x) = \sigma' \Big(W^{(\Lambda)}_{\gamma k_{\Lambda-1}} u^{(\Lambda-1)}_{k_{\Lambda-1}}(x) + b^{(\Lambda)}_{k_\Lambda} \Big) \times& \\ \Big[W^{(\Lambda)}_{\gamma k_{\Lambda-1}} \sigma' \Big(W^{(\Lambda-1)}_{k_{\Lambda-1} k_{\Lambda-2}} u^{(\Lambda-2)}_{k_{\Lambda-2}}(x) + b^{(\Lambda-1)}_{k_{\Lambda-1}} \Big) \Big[W^{(\Lambda-1)}_{k_{\Lambda-1} k_{\Lambda-2}} \sigma' \Big(W^{(\Lambda-2)}_{k_{\Lambda-2} k_{\Lambda-3}} u^{(\Lambda-3)}_{k_{\Lambda-3}}(x) + b^{(\Lambda-2)}_{k_{\Lambda-2}} \Big) \times& \\ \Big[\dots \Big[W^{(3)}_{k_3 k_2} \sigma' \Big(W^{(2)}_{k_2 k_1} u^{(1)}_{k_1}(x) + b^{(2)}_{k_2} \Big) \Big[W^{(2)}_{k_2 k_1} \sigma' \Big(W^{(1)}_{k_1} x + b^{(1)}_{k_1} \Big) W^{(1)}_{k_1} \Big] \Big] \dots \Big] \Big] \Big]&. \end{aligned} \tag{36}$$

By the uniform boundedness assumption (19) on the network parameters θ, (36) is uniformly bounded in ϵ from above for any $x \in [a, b]$. As a consequence of the positivity assumption (18), (19) further implies that each instance of $d\sigma/dx$ in (36) is bounded below by a positive number that is independent of ϵ. Note that

(36) generally contains sums and products of $d\sigma/dx$ and network weights (entries in the $W^{(l)}$ matrices). For any $t \in \Upsilon$, the weights are asymptotically larger than $\epsilon^{1/\Lambda}$, so that

$$\lim_{\epsilon \downarrow 0} \left(\frac{\epsilon}{W^{(1)}_{k_1} W^{(2)}_{k_2 k_1} \dots W^{(\Lambda-1)}_{k_{\Lambda-1} k_{\Lambda-2}} W^{(\Lambda)}_{k_\Lambda k_{\Lambda-1}}} \right) = 0$$

for any combination of indices $1 \leq k_1 \leq d_1$, $1 \leq k_2 \leq d_2$, and so on, meaning that (34) indeed holds.

Ergo, as in the proof of Theorem 3.2, as long as ϵ vanishes monotonically to zero in such a way that $a'(x_\alpha/\epsilon) \neq 0$ any ϵ, (33) limits to positive infinity as desired.

□

D Boundary value problem forcing functions

The right-hand side forcing function $f^\epsilon(x)$ for the Darcy BVP (28) with homogenenous Dirichlet boundary conditions is given by $g^\epsilon(x)/h^\epsilon(x)$, where

$$\begin{aligned} g^\epsilon(x) = 10 \Big[& 20\,\pi + 168\,\pi\epsilon\cos(x)\sin(x) - 20\Big(2\pi - 4\pi\cos^2(x) + \epsilon\sin(\pi/\epsilon)\Big)\cos(x/\epsilon) \\ & + \Big(21\pi + 160\,\pi\epsilon\cos(x)\sin(x)\Big)\sin(x/\epsilon)\Big], \end{aligned}$$

and

$$h^\epsilon(x) = 400\,\pi\epsilon\sin^2(x/\epsilon) + 840\,\pi\epsilon\sin(x/\epsilon) + 441\,\pi\epsilon.$$