

A COMPUTATIONAL ASSESSMENT OF DISCRETE VARIATIONAL PHYSICS-INFORMED NEURAL NETWORKS: PERFORMANCE ACROSS SYMMETRIC ELLIPTIC PROBLEMS

D. COTRIM, J. A. FERREIRA, L. MARQUES, AND G. PENA

ABSTRACT. In this paper we introduce discrete variational physics-informed neural networks (dvPINNs), a discretise-then-optimize approach where a neural network (NN) is trained to solve a fully discrete piecewise linear finite element (FE) formulation on nonuniform meshes, equivalent to a finite difference (FD) scheme. For symmetric elliptic problems, this strategy minimizes the naturally arising loss functional associated with the same finite element method (FEM), eliminating spatial automatic differentiation (AD). We analyse three fundamental error sources: optimisation, network approximation and discretisation, and establish empirical convergence orders with respect to epochs, architectural hyperparameters and sampling density, recovering the classical convergence orders of the underlying numerical methods for Poisson, biharmonic and steady linear elasticity problems. Finally, we benchmark dvPINNs against strong-form PINNs, demonstrating their improved behaviour under reduced solution regularity, mitigation of spectral bias on highly oscillatory problems and accelerated convergence using a multilevel mesh-refinement training strategy.

1. INTRODUCTION

Neural networks (NN) have shown success across a variety of applications, from computer vision to scientific computing [28]. Incorporating the governing PDE directly into the loss functional has led to physics-informed neural networks (PINNs). Unlike standard supervised learning [35], PINNs can be trained without labelled solution data, using the governing PDE as a constraint that guides the optimisation. This approach was popularised by [34], building on foundational concepts from [13, 27].

Specifically, PINNs approximate the solution of a (initial) boundary value problem offering a meshless alternative to classical numerical methods, such as the finite difference method (FDM) or finite element method (FEM). To illustrate the standard formulation, consider a continuous boundary value problem of the form

$$\begin{cases} \mathcal{L}u = f & \text{in } \Omega, \\ \mathcal{B}u = g & \text{on } \partial\Omega. \end{cases} \quad (1)$$

where $\mathcal{L}$ is a differential operator, f is a given source function and $\mathcal{B}$ encodes boundary conditions (BCs) on $\partial\Omega$. Letting u_θ denote a neural network parametrised by θ , an approximation of the solution of (1) is sought by minimizing a suitable objective function. Strong-form PINNs use (1) directly to define an *ideal* continuous loss functional, typically defined as the sum of the weighted L^2 -norms of the PDE residual and boundary residual,

$$\mathcal{J}^S(\theta) = \lambda_r \int_{\Omega} |\mathcal{L}u_\theta(x) - f(x)|^2 dx + \lambda_b \int_{\partial\Omega} |\mathcal{B}u_\theta(x) - g(x)|^2 dx. \quad (2)$$

In practice, these integrals are approximated using numerical quadrature formulas over collocation points $\{x_i^r\}_{i=1}^{N_r} \subset \Omega$ and $\{x_j^b\}_{j=1}^{N_b} \subset \partial\Omega$, which define the discrete strong-form loss functional,

$$\mathcal{J}_H^S(\theta) = \lambda_r \sum_{i=1}^{N_r} w_i^r |\mathcal{L}u_\theta(x_i^r) - f(x_i^r)|^2 + \lambda_b \sum_{j=1}^{N_b} w_j^b |\mathcal{B}u_\theta(x_j^b) - g(x_j^b)|^2, \quad (3)$$

where w_i^r and w_j^b are quadrature weights. Derivatives in equation (3) as well as the parameter gradient $\nabla_\theta \mathcal{J}_H^S$, are computed with AD.

Despite its success, this classical strong-form formulation faces two significant bottlenecks. First, it imposes restrictive regularity assumptions on the solution u : the PDE must hold in a strong pointwise sense, requiring higher-order differentiability ($u \in C^m$) that fails for problems with rough data or

Date: October 20, 2026.

2020 Mathematics Subject Classification. 65N30, 68T07, 35J20, 74B05.

Key words and phrases. physics-informed neural networks, discrete variational PINNs, discretise-then-optimize, finite element methods, empirical convergence orders, spectral bias, multilevel mesh refinement, symmetric elliptic problems, linear elasticity.

lower Sobolev regularity. Second, evaluating high-order derivatives with AD becomes computationally expensive, with alternatives explored in [41].

To relax regularity requirements, energy-based PINNs have been proposed in the literature. These approaches use the equivalence between (1) and a variational problem: find $u \in X$ such that

$$a(u, v) = \ell(v), \quad \forall v \in X, \quad (4)$$

where $a : X \times X \rightarrow \mathbb{R}$ is a continuous bilinear form and $\ell : X \rightarrow \mathbb{R}$ is a linear functional. Notable examples include the Deep-Ritz Method [14], which minimises the continuous energy functional, and variational PINNs (vPINNs and *hp*-vPINNs) [3, 25], which enforce weak residuals against piecewise polynomial test functions. In parallel, domain decomposition and discrete alternatives have been proposed. Extended PINNs (XPINNs, in [12]) and conservative PINNs (cPINNs, in [22]) decompose the computational domain into subdomains trained with separate local sub-networks coupled via interface loss penalties. Meanwhile, CAN-PINN [10], Discretization-Informed Neural Networks (DINN, in [45]) and Finite-Difference Time-stepping Optimization (FDTO, in [44]) incorporate finite differences to circumvent high-order AD in strong-form residuals. However, existing variational and conservative methods predominantly follow an *optimise-then-discretise* philosophy. They rely on continuous integral forms or penalty-based interface conditions that are subsequently approximated via Monte Carlo or numerical quadrature formulas. These, however, can introduce integration errors while still requiring spatial derivatives of the network.

In this paper, we propose *discrete variational physics-informed neural networks*, adopting a *discretise-then-optimise* approach that combines classical finite element or finite difference discretisations and scientific machine learning [1, 42]. Recent studies comparing standard PINNs against Optimizing a DIscrete Loss (ODIL, [23]) have highlighted the improved stability and accuracy of this paradigm [9]. Given a continuous variational problem, we first discretise it using a fully discrete piecewise linear FEM on nonuniform meshes. For a finite-dimensional space of grid functions X_H , we formulate the discrete weak problem: find $u_H \in X_H$ such that

$$a_H(u_H, v_H) = \ell_H(v_H), \quad \forall v_H \in X_H. \quad (5)$$

In the present framework, the resulting discrete scheme is algebraically equivalent to a FDM and represents an instance of the Galerkin method with numerical integration (GNI). In this scenario, X_H can be the piecewise continuous finite element space associated with polynomials of a fixed degree or simply the space of grid functions which is isomorphic to the former one. When the bilinear form $a_H(\cdot, \cdot)$ is symmetric, continuous, and coercive, the unique solution of (5) is the minimiser of the discrete functional

$$\mathcal{J}_H^W(u_H) = \frac{1}{2}a_H(u_H, u_H) - \ell_H(u_H), \quad u_H \in X_H. \quad (6)$$

This functional, already described in [8], is a natural loss function to train dvPINNs since we know that this minimisation problem has a unique solution in X_H . If we take this functional as the loss function to minimise, we are implicitly assuming that the neural network is parametrizing the values of u_H at the mesh vertices. This construction yields three major advantages: (i) lower regularity requirements directly inherited from the weak formulation; (ii) quadrature formulas inherently embedded within the discrete inner products; and (iii) complete elimination of spatial automatic differentiation, as spatial derivatives are evaluated through discrete difference stencils rather than backpropagation through the network architecture.

The main goal of this paper is to investigate the computational and convergence properties of dvPINNs. In particular, we analyse three potential sources of error: the optimisation error arising from nonconvex training, the neural network approximation error dictated by the architecture's width and depth, and the discretisation error determined by the mesh size and sampling density [5, 46]. We perform standard convergence tests and establish estimated convergence orders with respect to mesh spacing $H_{\max}$. Across all test cases, dvPINNs recover the classical $\mathcal{O}(H_{\max}^2)$ asymptotic rates predicted by FDM/FEM numerical analysis [2, 16, 17].

Additionally, we explore several important computational properties of dvPINNs. We show that dvPINNs naturally mitigate the well-known *spectral bias* [33] when learning high-frequency oscillatory solutions. We confirm that dvPINNs generalise accurately to unseen points outside the training mesh, showing that the network learns the underlying continuous solution rather than memorizing nodal values. We also test the applicability of an alternative training routine for dvPINNs, based on a mini-batch routine [29]. Finally, we show the robustness of dvPINNs over strong-form PINNs on problems with reduced Sobolev regularity.

The paper is organised as follows. In Section 2, we provide a brief overview of feed-forward neural networks as well as the discrete spaces, difference operators and discrete inner products and norms for dvPINNs. Section 3 gathers the majority of the results. In Section 3.1 we derive empirical convergence orders with respect to epochs. Section 3.2 is dedicated to obtaining empirical convergence orders with respect to the sampling density and neural network's architecture, for the Poisson, biharmonic and elasticity problems. In Section 4 we compare dvPINNs against classical PINNs on oscillatory problems and low-regularity solutions. Finally, in Section 5, we benchmark generalisation and introduce the multi-level mesh-refinement training algorithm. Concluding remarks are given in Section 6, and optimisation parameter details are provided in Appendix A.

2. PRELIMINARIES AND THE DVPINN FRAMEWORK

2.1. Feed-forward neural networks. We briefly introduce the notion of a feed-forward neural network and its underlying architecture, establishing the notation used throughout the paper. Let $d_{\text{in}}, d_{\text{out}} \in \mathbb{N}$ be the dimension of the input and output spaces, respectively. A generic fully connected feed-forward neural network architecture $\mathcal{F}_{\Theta} \subseteq C(\mathbb{R}^{d_{\text{in}}}; \mathbb{R}^{d_{\text{out}}})$ is the set of functions, denoted $u_{\theta} : \mathbb{R}^{d_{\text{in}}} \rightarrow \mathbb{R}^{d_{\text{out}}}$, consisting of $L \in \mathbb{N}$ layers (also known as the *depth*) of $\mathcal{M} \in \mathbb{N}$ neurons in each hidden layer (the *width*), resulting from the alternating composition of $L + 1$ affine transformations parametrised by θ (comprised of weights W_i and biases b_i) with a component-wise nonlinear activation function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$. Although layer widths may vary in practice, we set all hidden layers to width $\mathcal{M}$ without loss of generality. Figure 1 illustrates the described architecture.

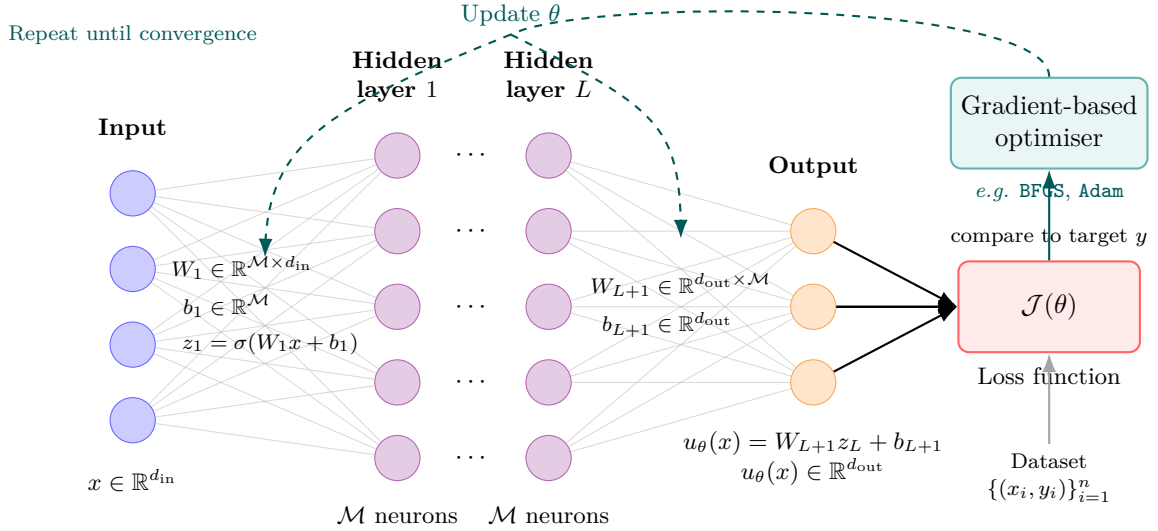


FIGURE 1. Architecture of a fully connected feed-forward neural network with L hidden layers of width $\mathcal{M}$.

In particular, one defines a NN as $u_{\theta} \in \mathcal{F}_{\Theta}$ such that

$$u_{\theta}(x) = W_{L+1} z_L + b_{L+1},$$

where

$$z_0 = x, \quad z_{\ell} = \sigma(W_{\ell} z_{\ell-1} + b_{\ell}), \quad \ell = 1, \dots, L.$$

Here $x \in \mathbb{R}^{d_{\text{in}}}$, $W_1 \in \mathbb{R}^{\mathcal{M} \times d_{\text{in}}}$, $b_1 \in \mathbb{R}^{\mathcal{M}}$; $W_{\ell} \in \mathbb{R}^{\mathcal{M} \times \mathcal{M}}$ and $b_{\ell} \in \mathbb{R}^{\mathcal{M}}$ for $\ell = 2, \dots, L$, while $W_{L+1} \in \mathbb{R}^{d_{\text{out}} \times \mathcal{M}}$ and $b_{L+1} \in \mathbb{R}^{d_{\text{out}}}$. The parameter space is therefore given by

$$\Theta = \{(W_{\ell}, b_{\ell})_{\ell=1}^{L+1}\}.$$

One refers to training a neural network as calculating $\theta \in \Theta$ that approximately minimises a given loss function $\mathcal{J} : \Theta \rightarrow \mathbb{R}$, typically through a gradient-based optimisation method. We denote by θ^* a minimiser (or approximate minimiser) of $\mathcal{J}$,

$$\theta^* := \underset{\theta \in \Theta}{\operatorname{argmin}} \mathcal{J}(\theta),$$

and by u_{θ^*} the corresponding trained network. In practice, θ^* is only known up to the accuracy of the optimisation algorithm. Therefore, we can summarise the core components into: (i) neural network's

architecture (specifically, the *width*, *depth* and activation function); (ii) the loss function for the problem; (iii) an optimiser algorithm to update the NN parameters.

2.2. Discrete spaces, grid functions and operators. While the discrete strong loss function $\mathcal{J}_H^S$ can work with any choice of points (e.g., completely random or defined by a well behaved grid), the dvPINN approach proposed here requires a structured choice. In the context of one-dimensional grids, we denote by

$$\begin{aligned}\bar{\Omega}_H &= \{x_i, i = 0, \dots, N : x_i = x_{i-1} + h_i, i = 1, \dots, N, x_0 = a, x_N = b\}, \\ \bar{\Omega}_H^* &= \bar{\Omega}_H \cup \{x_{-1} := a - h_1, x_{N+1} := b + h_N\},\end{aligned}\tag{7}$$

two possible sets of points which will be useful depending on the numerical method and differential equation being studied. In this case, we denote by V_H the space of grid functions defined on $\bar{\Omega}_H$ and, similarly, V_H^* the space of grid functions defined on $\bar{\Omega}_H^*$. For $v_H \in V_H^*$, we define the backward, forward and centred finite difference operators D_{-x} , D_x^* and $D_{c,x}$ as

$$\begin{aligned}D_{-x}v_H(x_i) &= \frac{v_H(x_i) - v_H(x_{i-1}))}{h_i}, \\ D_x^*v_H(x_i) &= 2\frac{v_H(x_{i+1}) - v_H(x_i)}{h_i + h_{i+1}}, \\ D_{c,x}v_H(x_i) &= \frac{v_H(x_{i+1}) - v_H(x_{i-1}))}{h_i + h_{i+1}},\end{aligned}$$

respectively, wherever the indices make sense. We also define the discrete Laplacian operator as $\Delta_H v_H(x_i) = D_x^* D_{-x} v_H(x_i)$, wherever it makes sense. With the previous notation, we introduce the following discrete subspaces

$$V_{H,0} = \{v_H \in V_H : v_H(x_0) = v_H(x_N) = 0\}$$

and

$$V_{H,ss}^* = \{u_H \in V_H^* : u_H(x_i) = \Delta_H u_H(x_i) = 0, i = 0, N\}.$$

As discussed before, the naturally occurring inner products in the variational formulation are discretised with suitable quadrature formulas, leading to discrete finite-sum inner products. First, we consider a discrete L^2 inner product and norm on V_H (or V_H^*) defined by

$$\begin{aligned}(u_H, v_H)_H &= \sum_{i=1}^{N-1} h_{i+1/2} u_H(x_i) v_H(x_i), \quad \|u_H\|_H = \sqrt{(u_H, u_H)_H}, \\ (u_H, v_H)_{H,*} &= \sum_{i=0}^N h_{i+1/2} u_H(x_i) v_H(x_i), \quad \|u_H\|_{H,*} = \sqrt{(u_H, u_H)_{H,*}},\end{aligned}\tag{8}$$

where $h_{i+1/2} = (h_i + h_{i+1})/2$, $h_{1/2} = h_1/2$ and $h_{N+1/2} = h_N/2$. We also introduce the notation

$$(u_H, v_H)_+ = \sum_{i=1}^N h_i u_H(x_i) v_H(x_i), \quad \|u_H\|_+ = \sqrt{(u_H, u_H)_+}.\tag{9}$$

To mimic the standard $H^1(\Omega)$ norm in V_H , and $H^2(\Omega)$ in $V_{H,ss}^*$, we define the discrete H^1 and H^2 norms respectively by

$$\begin{aligned}\|v_H\|_{1,H} &= (\|D_{-x}v_H\|_+^2 + \|v_H\|_H^2)^{1/2}, \\ \|w_H\|_{2,H} &= (\|w_H\|_{H,*}^2 + \|D_{-x}w_H\|_+^2 + \|\Delta_H w_H\|_{H,*}^2)^{1/2},\end{aligned}$$

where $v_H \in V_H$ and $w_H \in V_{H,ss}^*$. Finally, we consider the standard restriction operator $R_H : C(\bar{\Omega}) \rightarrow V_H$ and the averaging operator $\hat{R}_H : L^1(\Omega) \rightarrow V_H$ as

$$\hat{R}_H g(x_0) = \frac{2}{h_1} \int_{x_0}^{x_{1/2}} g(x) dx, \quad \hat{R}_H g(x_N) = \frac{2}{h_N} \int_{x_{N-1/2}}^{x_N} g(x) dx, \quad \hat{R}_H g(x_i) = \frac{1}{h_{i+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} g(x) dx.$$

For two-dimensional problems (see [15, 16] for more theoretical details on these operators and inner products), we extend (7), which now requires a nonuniform grid structure to exist. Therefore, if $1 \leq N, M \in \mathbb{N}$, we introduce

$$\bar{\Omega}_H = \{(x_i, y_j) : i = 0, \dots, N; j = 0, \dots, M\},$$

where $h_i = x_i - x_{i-1}$ and $k_j = y_j - y_{j-1}$. We define $k_{j+1/2}$ analogously to $h_{i+1/2}$, and

$$H_{\max} = \max(\max_i h_i, \max_j k_j), \quad \square_{i,j} = (x_{i-1/2}, x_{i+1/2}) \times (y_{j-1/2}, y_{j+1/2}) \cap \Omega,$$

where $x_{i+1/2} = x_i + h_{i+1}/2$, $x_{i-1/2} = x_i - h_i/2$ and $y_{j\pm 1/2}$ is defined analogously.

The one-dimensional difference operators extend naturally to two dimensions by attaching y_j for the x -direction derivatives and similarly defining D_{-y}, D_y^* and $D_{c,y}$. We introduce the discrete gradient for scalar functions as

$$\nabla_H u_H = (D_{-x} u_H, D_{-y} u_H)$$

as well as the gradient for vector-valued functions as the componentwise scalar gradient, denoted by $\nabla_H \mathbf{u}_H$, where $\mathbf{u}_H = (u_{1,H}, u_{2,H})$.

Finally, we define the spaces V_H and $V_{H,0}$ similarly to the one-dimensional case and extend (8) to two dimensions as

$$(u_H, v_H)_H = \sum_{i=1}^{N-1} \sum_{j=1}^{M-1} |\square_{i,j}| u_H(x_i, y_j) v_H(x_i, y_j), \quad \|u_H\|_H = \sqrt{(u_H, u_H)_H}, \quad (10)$$

where $|\square_{i,j}|$ denotes the area of $\square_{i,j}$. For (9), the extension is defined as $(u_H, v_H)_+ = (u_{1,H}, v_{1,H})_{+,x} + (u_{2,H}, v_{2,H})_{+,y}$ where

$$\begin{aligned} (u_{1,H}, v_{1,H})_{+,x} &= \sum_{i=1}^N \sum_{j=1}^{M-1} h_i k_{j+1/2} u_{1,H}(x_i, y_j) v_{1,H}(x_i, y_j), \quad \|u_H\|_{+,x} = \sqrt{(u_H, u_H)_{+,x}}, \\ (u_{2,H}, v_{2,H})_{+,y} &= \sum_{i=1}^{N-1} \sum_{j=1}^M h_{i+1/2} k_j u_{2,H}(x_i, y_j) v_{2,H}(x_i, y_j), \quad \|u_H\|_{+,y} = \sqrt{(u_H, u_H)_{+,y}}. \end{aligned}$$

The discrete version of the H^1 seminorm and norm are, respectively

$$\begin{aligned} \|\nabla_H u_H\|_+ &= (\|D_{-x} u_H\|_{+,x}^2 + \|D_{-y} u_H\|_{+,y}^2)^{1/2}, \\ \|u_H\|_{1,H} &= (\|u_H\|_H^2 + \|\nabla_H u_H\|_+^2)^{1/2}. \end{aligned}$$

The restriction operator R_H and averaging operator $\hat{R}_H$ are also extended, where the latter one integrates over $\square_{i,j}$ and divides by its area.

3. COMPUTATIONAL ERROR ANALYSIS AND EMPIRICAL CONVERGENCE

In this section, we shall do a full assessment of the computational properties of dvPINNs. In particular, as discussed before, we assume that three potential sources of error exist:

- **Optimisation error.** This is the classical and most expected source of error. Whenever a highly nonconvex optimisation problem is tackled, one cannot guarantee that the optimiser always converges to the true minimum. Numerically, this can be evaluated as

$$\mathcal{J}_H(u_\theta) - \inf_{\beta \in \Theta} \mathcal{J}_H(u_\beta), \quad (11)$$

which measures how far the loss function value (computed by the trained NN) is from the true minimum.

- **Sampling density.** As already stated, the ideal PINN loss function is approximated by numerical quadrature rules, relying on *collocation points*. It is expected that the number of points and the structure of the underlying grid contribute to a better loss function approximation associated with the dvPINN. This error can be measured as

$$\|u - u_H\|_{X_H},$$

which is viewed as a discretisation error. From the point of view of FDM/FEM, this is the natural discretisation error which describes the quality of the approximation with respect to the grid resolution.

- **Neural network capacity.** Finally, since we are training a NN to approximate the solution of a PDE, several questions arise regarding the neural network's architecture. In particular, it is safe to assume that a NN with more parameters should provide a better approximation. This will be evaluated as

$$\inf_{\beta \in \Theta} \|u - u_\beta\|_{X_H},$$

which characterises how well a given NN architecture approximates the exact solution.

Hence, this section is organised as follows. First, we examine the choice of optimiser. With this settled, we then analyse the remaining sources of error for a selection of problems.

Remark 1. During preliminary experiments, we tested **tanh**, **ReLU** and **ReLU²** as activation functions. While **ReLU** performed very poorly, the remaining achieved comparable results. Consequently, we opted to use **tanh** throughout this work. More details will be provided in Section 4.1.

3.1. Optimisation algorithm. Consider the one-dimensional Poisson problem with Dirichlet boundary conditions

$$\begin{cases} -u'' = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \quad (12)$$

where $\Omega = (-1, 1)$, with source term $f(x) = \pi^2 \sin(\pi x)$ and solution $u(x) = \sin(\pi x)$ for all $x \in \overline{\Omega}$. From a FEM standpoint, see [2] for more theoretical details, we look to approximate the solution u with $u_H \in V_{H,0}$, such that

$$a_H(u_H, v_H) = \ell_H(v_H), \forall v_H \in V_{H,0},$$

where

$$\begin{aligned} a_H(u_H, v_H) &= (D_{-x}u_H, D_{-x}v_H)_+, \\ \ell_H(v_H) &= (\hat{R}_H f, v_H)_H. \end{aligned}$$

In the context of the dvPINN, this translates in training u_θ to minimise (6) assuming the previous forms. We use this benchmark problem to evaluate dvPINNs across multiple configurations. Since the boundary conditions are imposed by the function space, we need to impose them on the NN exactly via an approximate distance function (ADF), which adds a final layer to solely treat the Dirichlet boundary conditions.

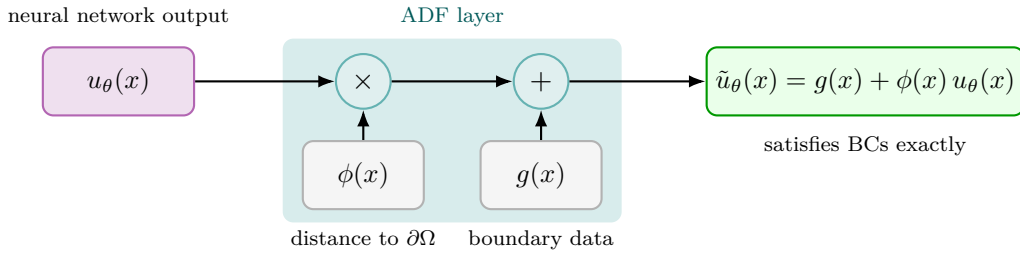


FIGURE 2. Approximate distance function (ADF), to impose the boundary Conditions.

For a general $\Omega = (a, b)$, the typical choice for the ADF is $\phi(x) = (x - a)(b - x)$ which, multiplied by the neural network's output, nullifies the output from the NN at the boundary. If nonhomogeneous Dirichlet BCs are present, after this step, one just has to add the boundary data.

In what follows of this section, we generate uniform meshes with $N \in \{16, 32, 64, 128, 256, 512\}$. The use of uniform meshes in this experiment is to isolate the optimisation errors since ill-conditioned grids might contaminate the results. To compute the error, we fully train the dvPINN associated with each N and restrict the exact solution to the underlying grid.

Given that we will address the influence of the neural network's capacity on the error separately, for these experiments we employ a sufficiently large NN to avoid capacity constraints (in this case, a feed-forward NN with 3-hidden layers and 50 neurons per layer).

We start by plotting the evolution of the error with respect to epochs, for the ADF-Dirichlet problem (12), using several standard optimisation algorithms. The parameters and stopping conditions are described in Appendix A.

As expected, the error decreases as the number of points N increases, throughout all the optimisers. This is a property of the sampling density, which will be tackled separately.

As per the optimisers, while ADAM [26] (Figure 3a) is the standard optimiser for classical PINNs, we clearly see it struggle to train dvPINNs to an acceptable threshold. This is expected, as ADAM is known to suffer when faced with stiff, highly nonconvex problems associated with complex PDE systems [40]. While ADAM can work on the simpler pointwise loss functional of classical PINNs, the variational nature of dvPINNs is inherently trickier.

Regarding BFGS [6, 18, 20, 36] and L-BFGS [32] (Figures 3b and 3c, respectively), the results are very similar. They are a clear improvement over ADAM. Their behaviour is also textbook: the full BFGS achieves slightly lower errors with fewer epochs, because it better approximates the inverse Hessian, while the limited counterpart dramatically lowers memory usage, sacrificing some curvature information thus taking more epochs with slightly higher errors.

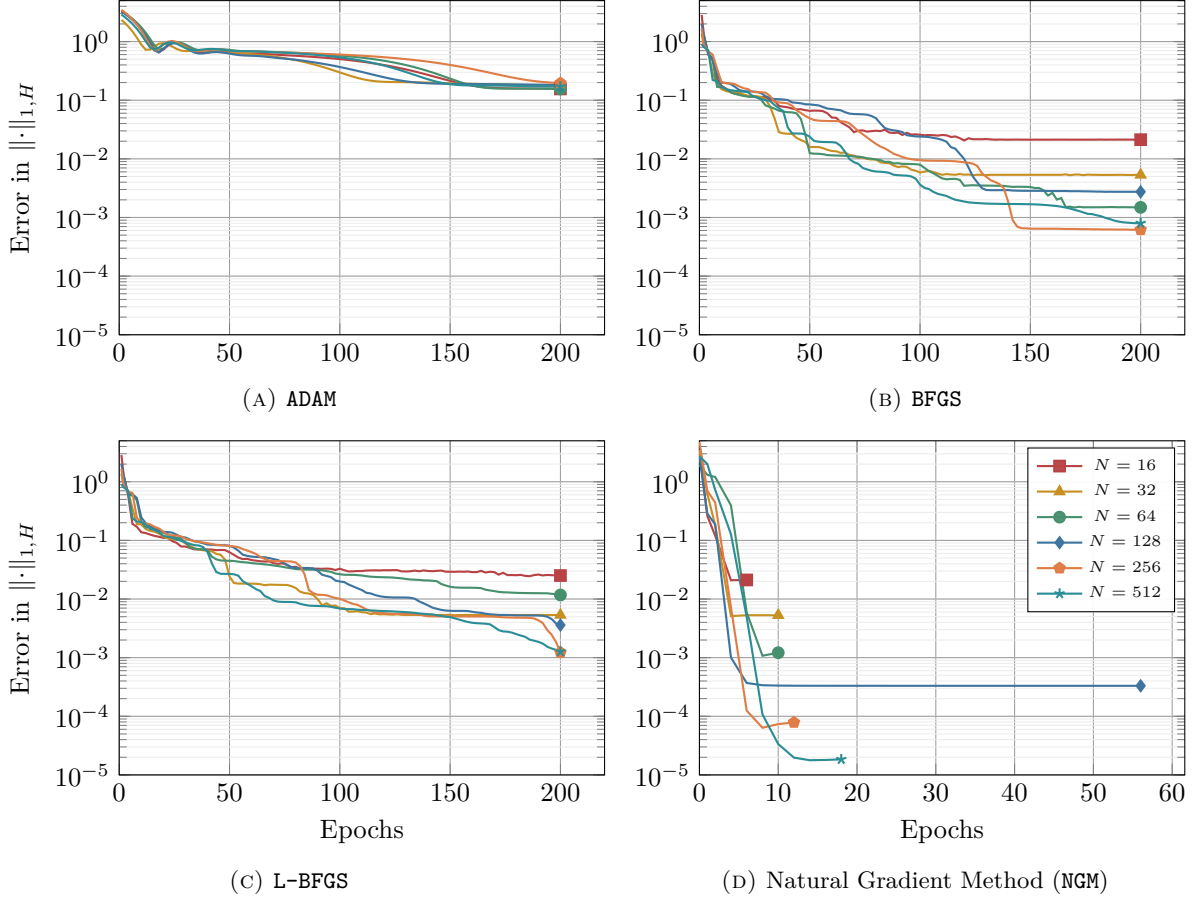


FIGURE 3. Error vs. Epochs plots, for Dirichlet ADF-dvPINN, on several optimisers.

Finally, the Natural Gradient Method [31], NGM, is the best option, as can be seen from Figure 3d. It achieves smaller errors than the other methods in far fewer epochs. This result is expected knowing that NGM preconditions the problem using a metric that is intrinsic to the variational problem itself. Therefore, NGM will be the optimiser of choice and used throughout the work unless otherwise specified.

Nonetheless, we must acknowledge that these results could be improved with further tuning of the parameters, in particular for ADAM, BFGS and L-BFGS. While this tuning phase was not thoroughly probed, we expect the NGM to remain the better optimiser for this type of problems.

Recall that we assumed one of the potential sources of error to be the optimisation error, described in equation (11). Notice that, while the optimum value for classical PINNs is greater than or equal to zero (and thus the training error is solely the value of the loss function at the trained NN), for dvPINNs, the optimal value is always nonpositive. For the problem (12), if we consider

$$\inf_{\beta \in \Theta} \mathcal{J}_H^W(u_\beta) \approx \mathcal{J}^W(u) = -\frac{\pi^2}{2},$$

we can perform an empirical convergence analysis of the optimisation with respect to epochs, i.e. whether or not

$$\mathcal{E}_\theta^\mathcal{N} := \mathcal{J}_H(u_\theta) - \left(-\frac{\pi^2}{2}\right) \approx \mathcal{O}(\mathcal{N}^{-k}), \quad (13)$$

where $\mathcal{N}$ is the number of epochs and k the empirical order. We present the empirical convergence plots of optimisation with respect to the number of epochs $\mathcal{N}$ associated with Figure 3, reusing the colour-based legend entries from it.

Although reference slopes are included with several potential convergence orders, the curves do not exhibit a sufficiently long linear region, on the log-log scale, to assign a global empirical order of convergence. Instead, we must analyse the results in terms of objective value decrease power and consistency.

Figure 4a (ADAM) exhibits the slowest convergence of the presented algorithms. While different runs exhibit similar behaviour, indicating stable but slow convergence, the existence of long plateaus during training makes this algorithm unsuitable for these problems.

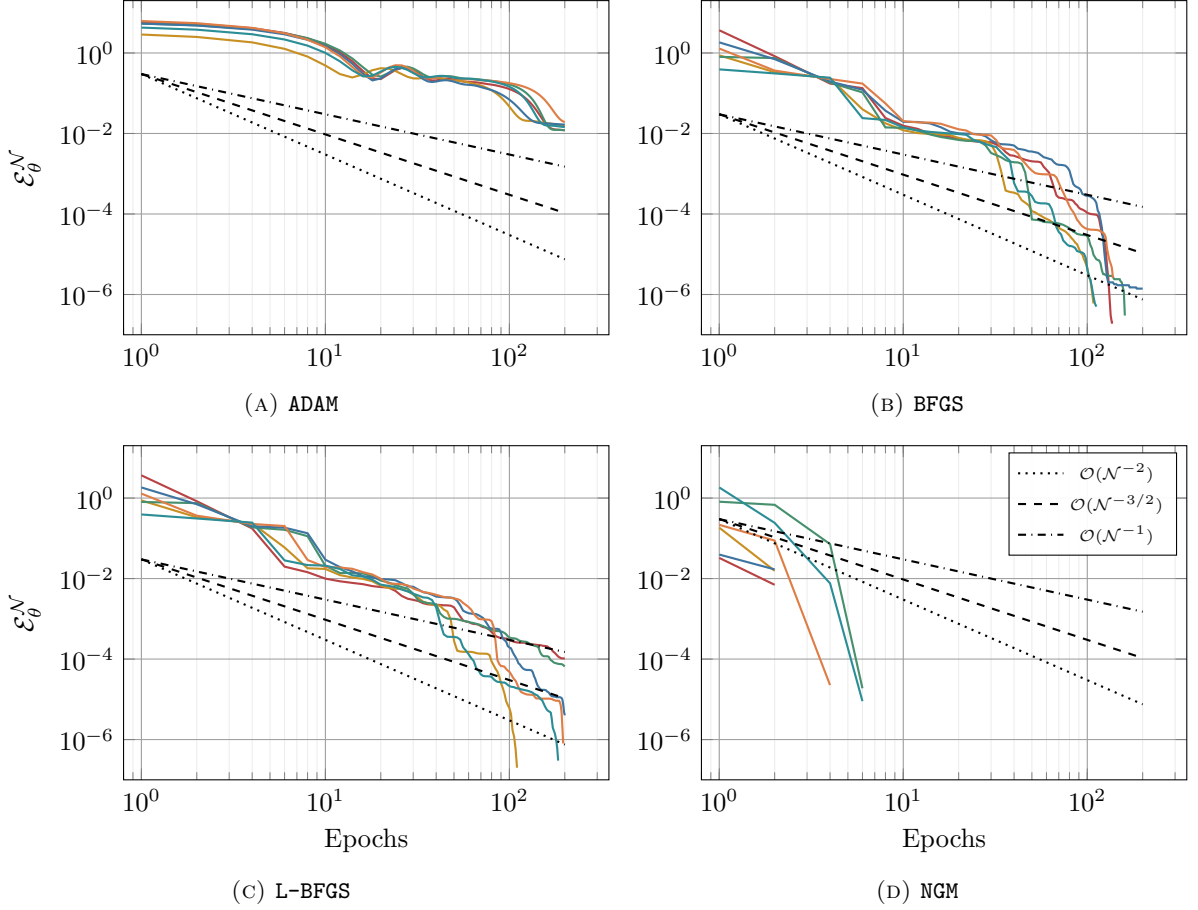


FIGURE 4. Empirical convergence analysis of optimisation with respect to epochs, for Dirichlet ADF-dvPINN, related to Figure 3.

Regarding Figure 4b and Figure 4c, BFGS and L-BFGS respectively, there are clear advantages over ADAM. The optimisation error decreases substantially in the first epochs, with a general empirical second-order convergence until 10^2 epochs, with some variability on the final stages near the optimum value.

Finally, Figure 4d shows NGM to exhibit a distinct behaviour: since the log-log representation omits epochs with *negative* optimisation error, the missing points from the plot are caused by epochs where the optimiser approached the true minimiser from below. It is clearly faster than all other optimisers, reducing the optimisation error by several orders of magnitude within fewer than 10 epochs.

Both experiments indicate that NGM is the clear choice of an optimiser for dvPINNs since it achieves smaller errors in far fewer iterations. Consequently, we shall employ NGM in the settings described in Appendix A, unless otherwise specified.

3.2. Density sampling and neural network capacity. We shall now address the remaining two sources of error simultaneously: density sampling and neural network capacity. For that, we shall consider several problems, namely the one-dimensional Poisson's problem, for all the standard boundary conditions; the two-dimensional Poisson's problem with strongly imposed Dirichlet boundary conditions; the one-dimensional biharmonic problem and, finally, a two-dimensional steady linear elasticity problem. The appropriate loss functions will be introduced as we present each problem.

Regarding the density sampling source of error, we will follow ideas very similar to those used in the standard convergence order estimation employed in FDM/FEM analysis. We consider a collection of randomly generated grids with increasing values of N and, therefore, decreasing values of $H_{\max} = \max_{i=1,\dots,N} h_i$. Next, we plot the error, in the discrete norms, for the associated value of $H_{\max}$ of the underlying grid. It is expected that finer meshes produce smaller errors, with an empirical convergence order obtained via comparison with reference slopes. It is desired that the dvPINNs inherit the underlying numerical method's convergence order. Since we want to isolate the density sampling error, we shall, as before, employ a sufficiently large NN with 3-hidden layers and 50 neurons per layer, adding to a

parameter count of approximately 5×10^3 . We also present the corresponding iteration log, for all the problems, to compare with the behaviour obtained in Section 3.1.

As per the neural network’s capacity (also called parameter space dimension), we will be interested in checking the influence of the parameter count in the error. It is expected that a NN with more parameters can express harder solutions more easily. Therefore, we consider the *ansatz* of

$$|\Theta| \approx \frac{1}{H_{\max}^d} \approx \left(\frac{N}{b-a} \right)^d, \quad (14)$$

matching the number of degrees of freedom of a d -dimensional tensor product FDM/FEM grid with N points per axis. For $d \geq 2$, the quantity $b-a$ should be interpreted as the domain length in the same direction as $H_{\max}$ is found.

Hence, freezing the chosen depth as a 2-hidden-layer NN, the necessary width of each layer to match the *ansatz* (14) depends directly on the number of grid points. In Table 1 we present the necessary widths for each layer to match *ansatz* (14). They were calculated using uniform grids since the high variability could cause an ill-conditioned problem with an extraordinarily large $H_{\max}$ value, consequently capping the parameter count too harshly.

The objective of this experiment is thus to check if a NN with just enough parameters to express the solution for the associated N value can also express the solution for a finer mesh. If the *ansatz* is correct, we should see that every NN (with the associated parameter count) should be able to express its associated solution and the coarser ones relative to itself, but not the relative finer ones. If the NN can outperform itself, *i.e.*, express a finer solution with a parameter count below par, then we can state that the *ansatz* can be relaxed.

$\Omega \backslash N$	16	32	64	128	256	512
$(-1, 1)$	2	3	4	7	10	15
$(-3/4, 3/4)$	2	3	5	8	12	17
$(-1, 1)^2$	6	14	30	62	-	-

TABLE 1. Widths, to check the dependence of the parameter space dimension with respect to $H_{\max}$.

Notice the omission of hypothetical necessary widths for the two-dimensional case on the finest meshes. This is because our implementation was not memory-conservatively optimised, raising storage problems particularly regarding the Gram matrix associated with the NGM. While an optimisation is most likely possible, the presented results are enough to derive substantiated conclusions.

3.2.1. One-dimensional Poisson with strongly imposed Dirichlet BCs. We start with the Poisson’s problem, with strongly imposed Dirichlet boundary conditions via an ADF, as already stated in equation (12). Consider, in Figure 5, the density sampling plots with the associated epoch log.

Notice, in Figure 5, that dvPINNs exhibits second-order convergence with respect to $H_{\max}$. This matches the convergence order of the underlying numerical method, see [2]. We expect this case to be the easiest to train, since the ADF layer addresses the boundary conditions, leaving the optimiser to deal solely with the differential portion.

We shall now test the *ansatz* (14). Denote N -dvPINN the dvPINN trained with a NN of 2 hidden layers and width corresponding to N in Table 1.

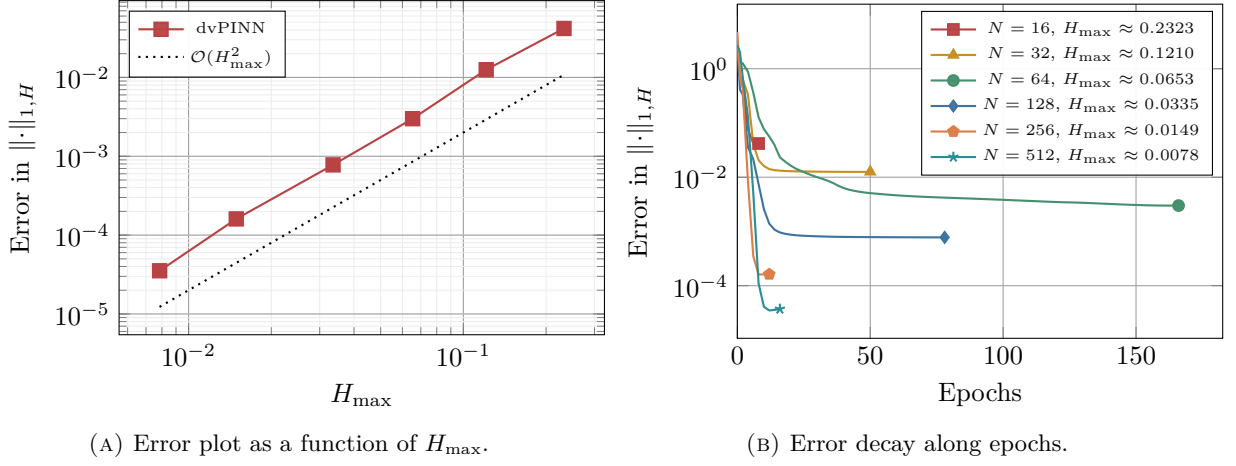
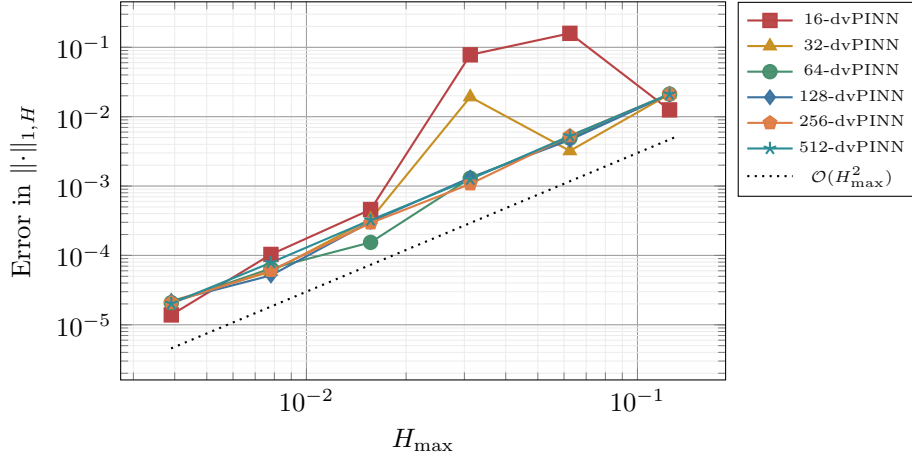


FIGURE 5. Error and Epochs log plots for Poisson Dirichlet ADF-dvPINN.

FIGURE 6. Parameter space dimension influence on the error in $\|\cdot\|_{1,H}$, for the Dirichlet ADF-dvPINN.

We can see in Figure 6 that the error decays are cleaner when the width of the NN is increased according to Table 1. This validates our hypothesis of the influence of the neural network capacity. We note that for $N = 16$, the dvPINN's width was calculated to match the degrees of freedom of the approximate solution. It is expected that, for finer meshes, this parameter count reveals itself to be insufficient. In fact, for $N = 32$ or $N = 64$, the 16-dvPINN causes an increase in the error, thus showing its insufficient capacity. However, while the 64-dvPINN should only have the capacity to express the approximate solution for the three coarsest grids, note that it can express the solution for all meshes. The same argument can be made for all subsequent N -dvPINNs, as they all are able to express all the solutions. This means some dvPINNs exceed the expressive capacity predicted by the *ansatz*.

3.2.2. One-dimensional Poisson with Neumann BCs. Let $\Omega = (-1, 1)$. Consider the problem with Neumann boundary conditions,

$$\begin{cases} -u'' = f & \text{in } \Omega, \\ u' = -\pi & \text{on } \partial\Omega, \\ \int_{-1}^1 u(x) dx = 0, \end{cases} \quad (15)$$

with the same solution and source term as before. We now seek to approximate the solution u in V_H^* , where the ghost points in $\bar{\Omega}_H^*$ help enforcing Neumann or Robin boundary conditions. Variationally, this

leads to a numerical method with the forms

$$\begin{aligned} a_{H,N}(u_H, v_H) &= (D_{-x}u_H, D_{-x}v_H)_+ + \frac{\gamma_m}{H_{\max}} \left(\sum_{i=0}^N h_{i+1/2} u_H(x_i) \right) \left(\sum_{i=0}^N h_{i+1/2} v_H(x_i) \right), \\ \ell_{H,N}(v_H) &= (\hat{R}_H f, v_H)_{H,*} + \sum_{i=0,N} \frac{\pi}{\beta_i} v_H(x_i), \end{aligned} \quad (16)$$

where $\beta_0 = 1$, $\beta_N = -1$ and γ_m is a positive parameter. We design, as before, the loss function following (6). Consider in Figure 7, the density sampling and associated epoch log.

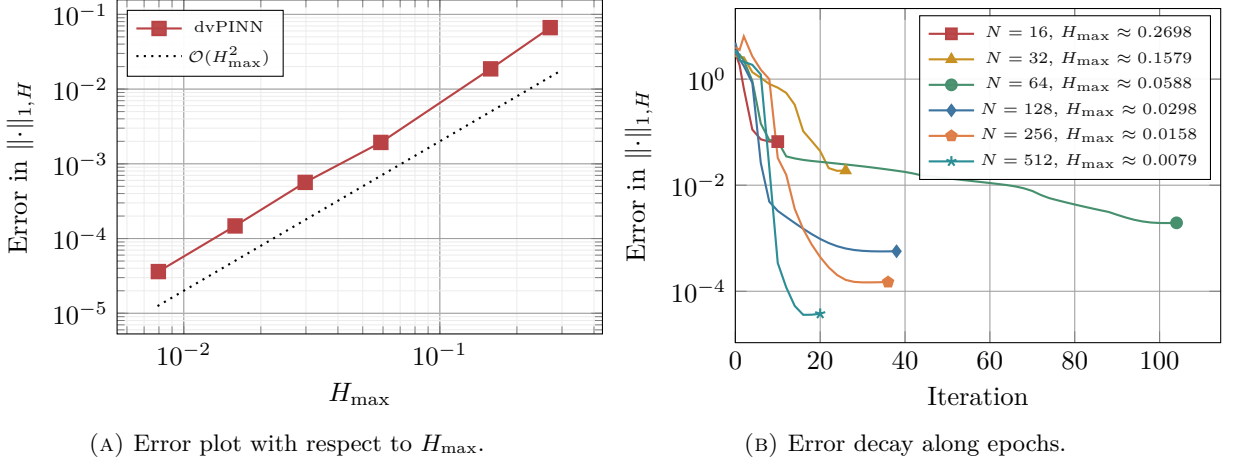


FIGURE 7. Error and Epochs log plots for the case of Neumann dvPINN.

Once again, the dvPINN inherits the convergence order of the underlying numerical method. The epoch log reveals a very similar behaviour to Figure 5, with clear convergence along epochs. This is due to the fact that the Neumann boundary conditions training are naturally embedded in the bilinear form when one derives the variational formulation.

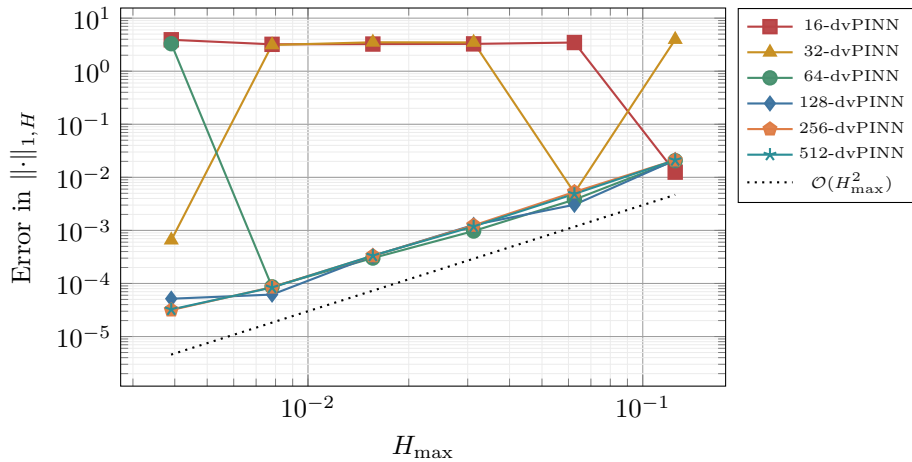


FIGURE 8. Parameter space dimension influence on the error in the norm $\|\cdot\|_{1,H}$, for the Neumann dvPINN.

Similar to the previous case, Figure 8 confirms the influence of the parameter space dimension on the convergence error, as well as the empirical observation regarding a potential relaxation of *ansatz* (14).

3.2.3. *One-dimensional Poisson with Robin BCs.* Consider $\Omega = (-3/4, 3/4)$. We now tackle the Poisson problem with Robin boundary conditions,

$$\begin{cases} -u'' = f & \text{in } \Omega, \\ u(-3/4) - u'(-3/4) = \frac{\sqrt{2}}{2}(\pi - 1), \\ u(3/4) + u'(3/4) = \frac{\sqrt{2}}{2}(1 - \pi), \end{cases} \quad (17)$$

with the same solution u and source term f as before. For the same reasons as the Neumann case, we seek to approximate a solution in V_H^* . As before, we can obtain a discrete variational problem like (5), where, in this case, we have

$$\begin{aligned} a_{H,R}(u_H, v_H) &= (D_{-x}u_H, D_{-x}v_H)_+ + \sum_{k=0,N} u_H(x_k)v_H(x_k), \\ \ell_{H,R}(v_H) &= (\hat{R}_H f, v_H)_{H,*} + \sum_{i=0,N} \frac{\frac{\sqrt{2}}{2}(\pi - 1)}{\beta_i} v_H(x_i), \end{aligned} \quad (18)$$

where, as before, $\beta_0 = 1, \beta_N = -1$. This leads to the now-classic design for a loss function (6), replacing the bilinear and linear forms.

We plot in Figure 9 the same tests as before: density sampling convergence analysis and the associated epoch log.

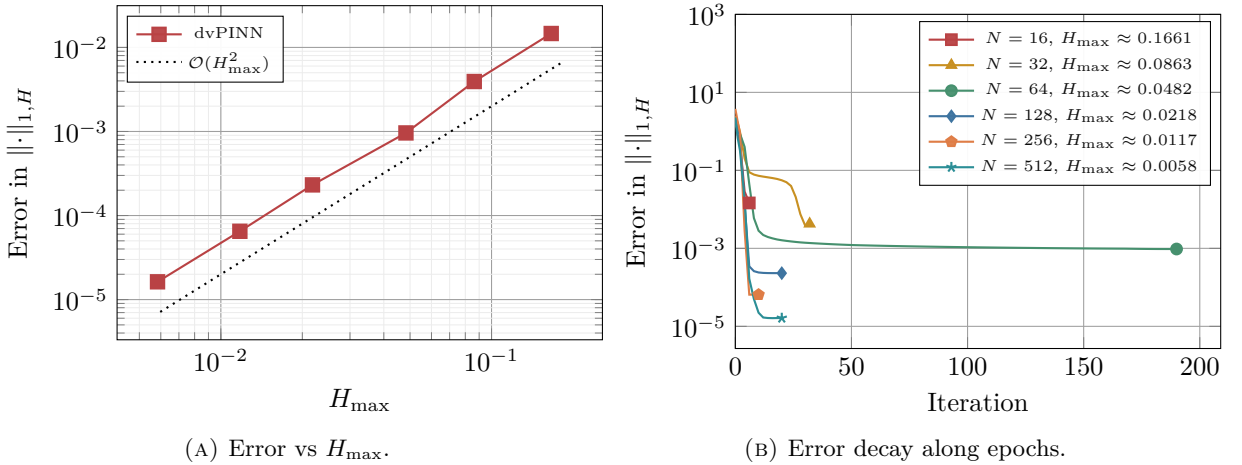


FIGURE 9. Convergence and Epochs log plots for Robin dvPINN.

Very similar conclusions to the previous cases can be drawn from Figure 9. We get second order estimated convergence with respect to $H_{\max}$. In Figure 10 we observe the *ansatz* to be correct, but could be slightly relaxed.

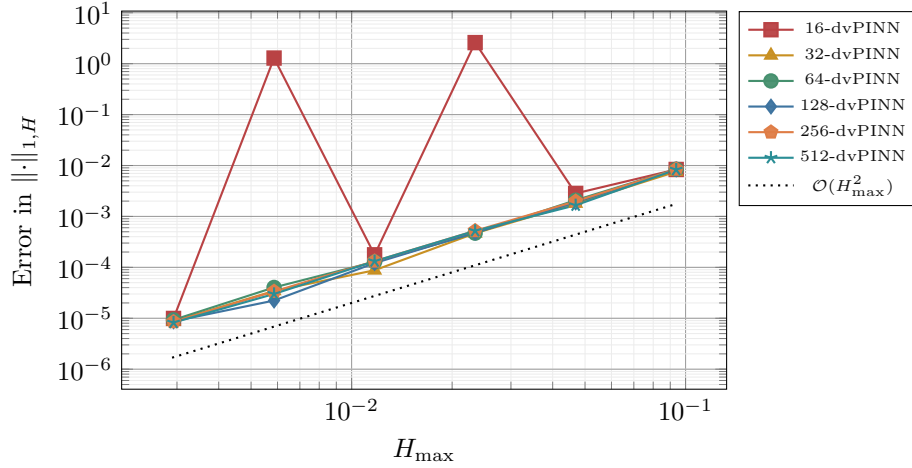


FIGURE 10. Parameter space dimension influence on the error in $\|\cdot\|_{1,H}$, for the Robin dvPINN.

3.2.4. *Biharmonic problem with simply supported BCs.* Regarding the Biharmonic problem, we shall test the following problem with simply supported boundary conditions,

$$\begin{cases} u^{(4)} = f & \text{in } \Omega, \\ u = u'' = 0 & \text{on } \partial\Omega, \end{cases} \quad (19)$$

where $\Omega = (-1, 1)$ and source term $f(x) = \pi^4 \sin(\pi x)$, $x \in \overline{\Omega}$, with the same solution u as before. We now seek to approximate a function in $V_{H,ss}^*$ which, upon application of the definitions of the discrete operators, implies

$$u_H(a) = u_H(b) = 0, \quad u_H(x_{-1}) = -u_H(x_1), \quad u_H(x_{N+1}) = -u_H(x_{N-1}).$$

To strongly impose the boundary conditions, we use the following function as a final layer,

$$\Phi((x_{-1}, x_0, x_1, \dots, x_{N-1}, x_N, x_{N+1})) = (-u_1, 0, u_1, \dots, u_{N-1}, 0, -u_{N-1}). \quad (20)$$

The corresponding discrete variational problem is obtained by choosing

$$\begin{aligned} a_H(u_H, v_H) &= (\Delta_H u_H, \Delta_H v_H)_{H,*}, \\ \ell_H(v_H) &= (\widehat{R}_H f, v_H)_{H,*}. \end{aligned}$$

where the loss function derives directly from (6).

As expected, the results in Figure 11 are very similar to the Poisson problem with Dirichlet ADF-dvPINN. The dvPINN method inherits the quadratic convergence order with respect to $H_{\max}$ from the underlying FEM method, when a Chebyshev-Gauss-Lobatto mesh is employed. When a completely random mesh is applied, the convergence order drops to linear. Since the boundary conditions are imposed exactly, a remarkably fast convergence with respect to epochs was expected and confirmed.

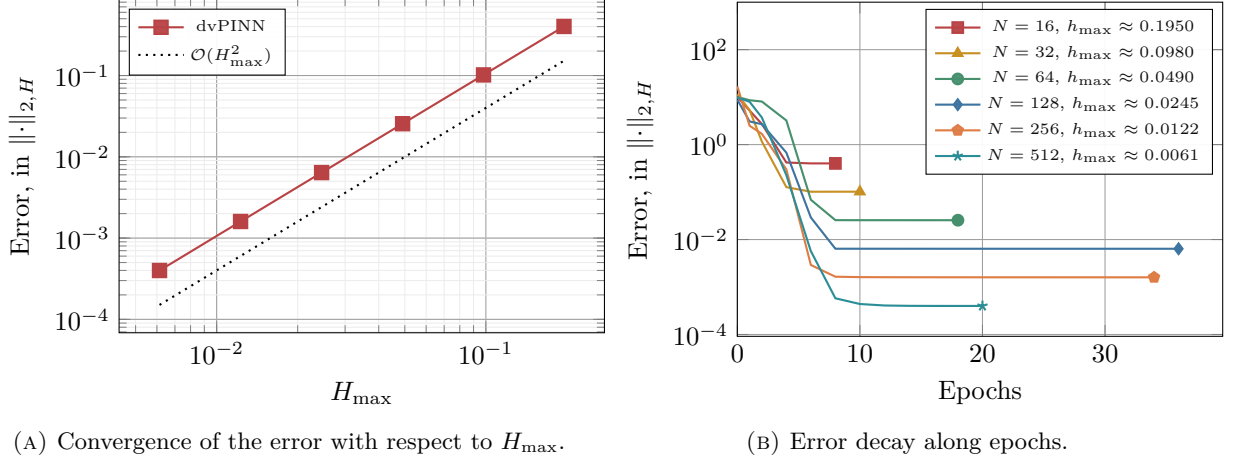


FIGURE 11. Error and Epochs log plots for the Bilaplacian dvPINN.

As before, we observe yet again the possible relaxation of *ansatz* (14).

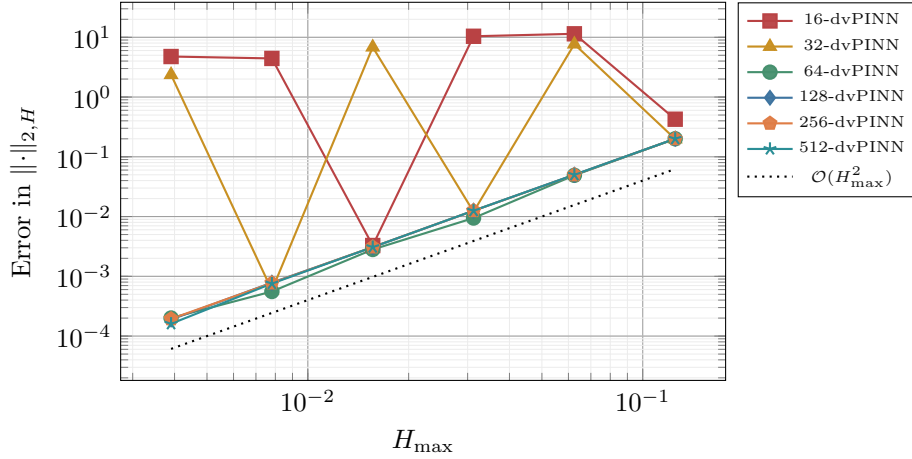


FIGURE 12. Parameter space dimension influence on the error convergence, for the Bilaplacian dvPINN.

3.2.5. *Two-dimensional Poisson with strongly imposed Dirichlet BCs.* Let $\Omega = (-1, 1)^2$. Consider now the two-dimensional Poisson problem with strongly imposed boundary conditions,

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega \end{cases} \quad (21)$$

where the source term is given by $f(x, y) = 5\pi^2 \sin(\pi x) \sin(2\pi y)$ and the solution by $u(x, y) = \sin(\pi x) \sin(2\pi y)$, for $(x, y) \in \bar{\Omega}$. This is a direct extension of the problem (12) for two spatial dimensions. As we seek a function in the two-dimensional generalisation of $V_{H,0}$, we consider the natural extension to the ADF as

$$\phi(x, y) = (x - a_x)(x - b_x)(y - a_y)(y - b_y),$$

with $(x, y) \in [a_x, b_x] \times [a_y, b_y]$, which simultaneously imposes homogeneous boundary conditions on the entire domain square. Following (5), we can rewrite problem (21) in a discrete variational formulation where

$$\begin{aligned} a_H(u_H, v_H) &= (\nabla_H u_H, \nabla_H v_H)_+, \\ \ell_H(v_H) &= (\hat{R}_H f, v_H)_H. \end{aligned}$$

Consider, in Figure 13, the convergence plot with respect to $H_{\max}$ and the associated epoch log, calculated using the discrete H^1 -norm adapted for the two-dimensional setting.

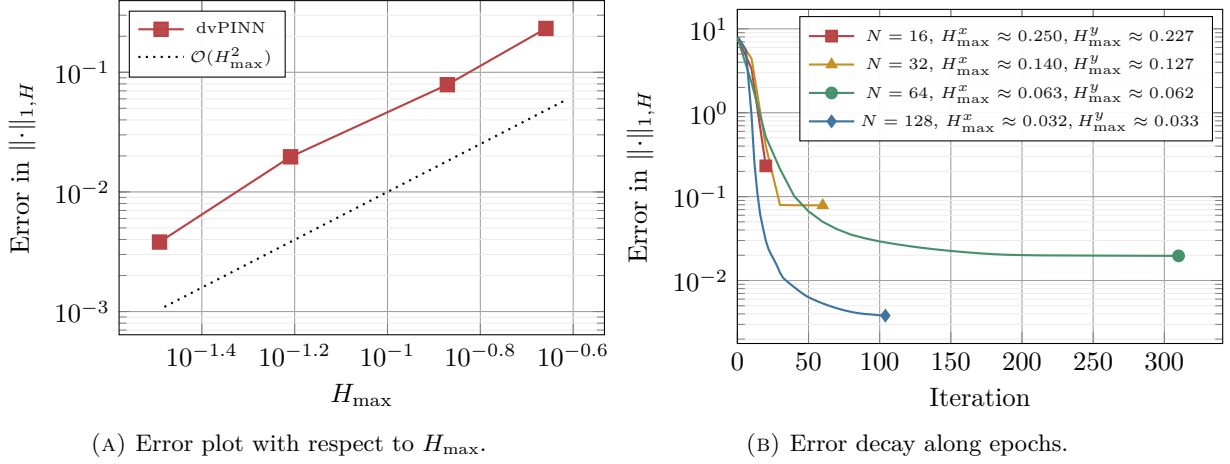
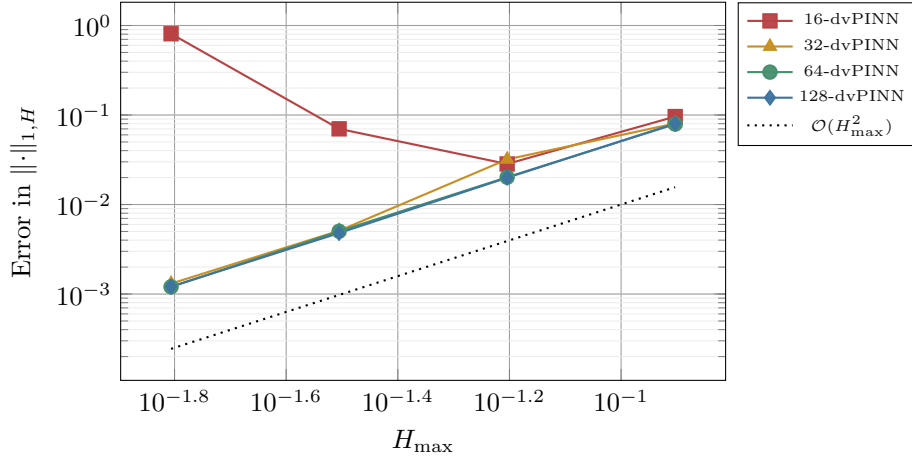


FIGURE 13. Error and Epochs log plots for 2D strongly imposed Dirichlet dvPINN.

Yet again, we observe a quadratic convergence order with respect to the maximum spacing. The epoch log shows clean convergence, with obviously slightly more epochs used than the one-dimensional. Regarding the influence of the parameter space dimension, Figure 14 confirms our *ansatz* (14) for $d = 2$.

FIGURE 14. Parameter space dimension influence on the error in $\|\cdot\|_{1,H}$, for the strongly imposed two-dimensional Dirichlet dvPINN.

3.2.6. Elasticity problem. Finally, we consider a steady linear elasticity problem. Let $\Omega = (-1, 1)^2$ and let us denote $\mathbf{u} = (u_1, u_2)$ the displacement in the x and y directions. The problem reads: find $\mathbf{u}$ such that

$$\begin{cases} -\nabla \cdot \epsilon(\mathbf{u}) = \mathbf{f} & \text{in } \Omega, \\ \mathbf{u} = \mathbf{0} & \text{on } \partial\Omega, \end{cases} \quad (22)$$

where $\epsilon(\mathbf{u}) = (\nabla \mathbf{u} + \nabla \mathbf{u}^T)/2$ and the divergence of $\epsilon(\mathbf{u})$ is to be interpreted row-wise.

Notice that, contrary to all previous examples, the dvPINN now has two outputs, $u_{\theta,1}$ and $u_{\theta,2}$ respectively. Compressing into a single variational formulation leads to

$$\begin{aligned} a_{H,E}(\mathbf{u}_H, \mathbf{v}_H) &= (\nabla_H \mathbf{u}_H, \nabla_H \mathbf{v}_H)_+ + \frac{1}{2}(D_{-y}u_{H,1}, D_{-y}v_{H,1})_{y,+} + \frac{1}{2}(D_{-x}u_{H,2}, D_{-x}v_{H,2})_{x,+} \\ &\quad + \frac{1}{2}(D_{c,y}u_{H,2}, D_{c,x}v_{H,1})_H + \frac{1}{2}(D_{c,x}u_{H,1}, D_{c,y}v_{H,2})_H \\ \ell_{H,E}(\mathbf{v}_H) &= (R_H \mathbf{f}, \mathbf{v}_H)_H, \end{aligned} \quad (23)$$

where we have extended the inner products to $\mathbf{u}_H = (u_{H,1}, u_{H,2})$, $\mathbf{v}_H = (v_{H,1}, v_{H,2})$ as $(\mathbf{u}_H, \mathbf{v}_H)_H = (u_{H,1}, v_{H,1})_H + (u_{H,2}, v_{H,2})_H$ and $(\cdot, \cdot)_+$ similarly, and the restriction operator is applied componentwise. This formulation trains both displacement components simultaneously, via (6). The force term $\mathbf{f}$ is determined such that $\mathbf{u}(x, y) = \pi(\sin(2\pi x) \sin^2(\pi y), \sin^2(\pi x) \sin(2\pi y))$, $(x, y) \in \bar{\Omega}$ is the solution of (22).

In Figure 15 we observe yet again second order convergence with respect to $H_{\max}$, for both components of the displacement.

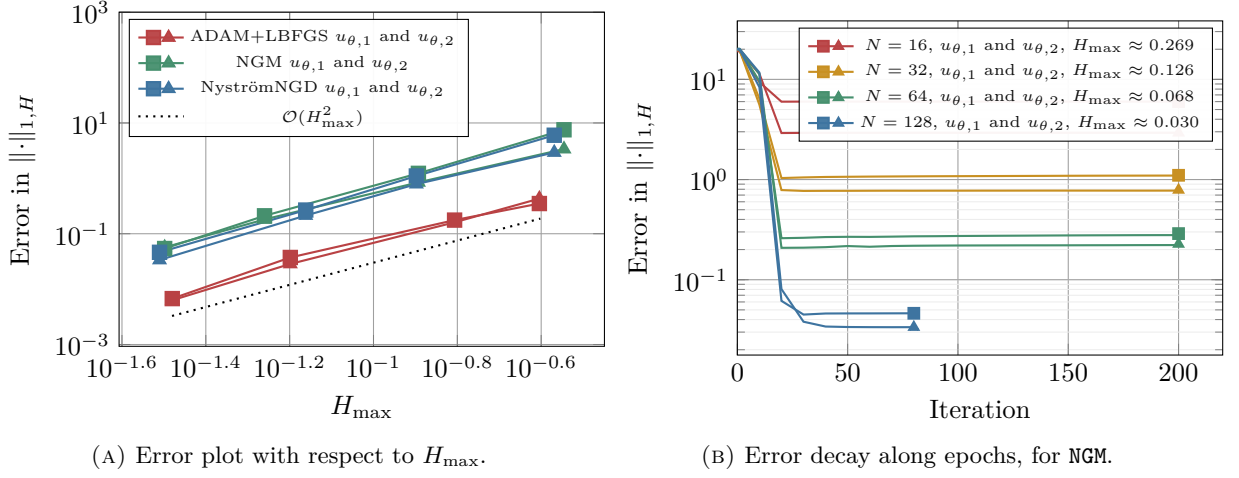


FIGURE 15. Error and Epochs log plots for steady linear elasticity problem.

Since this is the hardest problem tackled, it is expected to be the one where the optimiser has more difficulties in finding the solution. Although Figure 15b indicates rapid epoch-wise convergence, the per-epoch cost of NGM is substantial due to the dense Gram matrix. This was already documented in [4, 19, 31], where the authors propose a matrix-free NGM, which we refer to as NyströmNGD. In fact, our results corroborate said premise: we obtain absolute errors very similar to NGM with much less computational time. Since we only go up to $N = 128$, the traditional ADAM+L-BFGS routine still produces great convergence results, without plateauing, as observed.

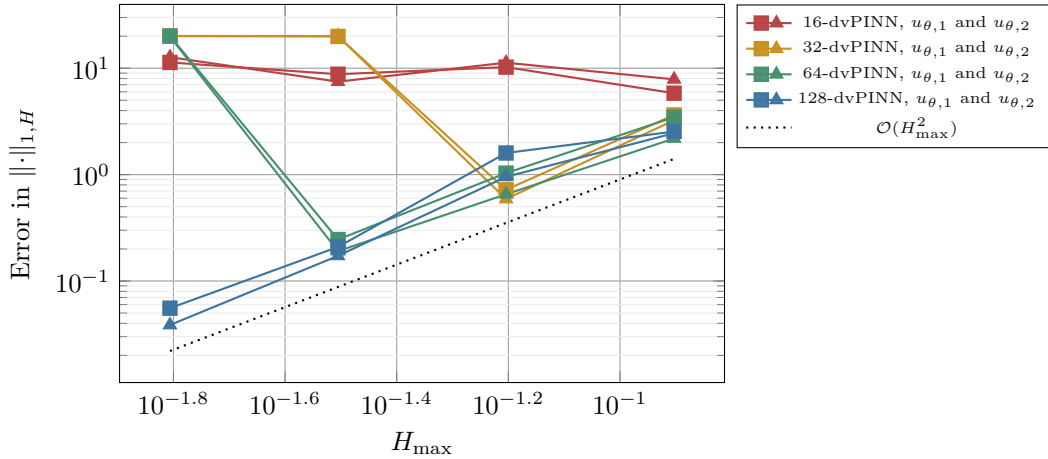


FIGURE 16. Parameter space dimension influence on the error in $\|\cdot\|_{1,H}$, the steady linear elasticity problem.

Regarding the parameter space dimension influence, Figure 16 exhibits the exact expected behaviour: a given dvPINN can only express its solution or a coarser one, but never a finer one. This means that, for this problem, the *ansatz* (14) is accurate and cannot be relaxed.

Remark 2. While not exhaustive, the presented analysis allows us to conclude that the three potential sources of error we have hypothesised can be controlled, in the sense that: (i) the training error can be made small enough, if the optimiser is well suited and considering enough epochs; (ii) the discretisation error decreases quadratically with $H_{\max}$; (iii) the neural network's capacity is not a bottleneck if the parameter space dimension checks our *ansatz* (14). In these conditions, it is possible to repeatedly train a dvPINN successfully when the framework is applicable.

4. COMPARATIVE STUDY: ROBUSTNESS AND CLASSICAL PINN PATHOLOGIES

Much more work has been developed for the classical PINN formulation compared to variational PINNs, from theoretical results [30, 37] to several computational applications [7, 11, 24]. Hence, it is also well-known that classical PINNs are not a universal solution. In this section, we shall make some comparative tests, as well as evaluate dvPINNs in two well-known limitations of classical PINNs.

4.1. Influence of activation functions and piecewise linear FEM emulation. We expected a dvPINN with activation function ReLU, see [21], to closely mimic classical linear FEM, since it emulates piecewise linear functions [21]. Similarly, one using ReLU² (the activation function obtained by squaring ReLU) should mimic FEM using piecewise second-order elements. To put this hypothesis to the test, we trained several PINNs and dvPINNs, for several activation functions, under several optimisers. The combinations of *method-activation function-optimiser* were applied to the one-dimensional Poisson, problem (12), with the corresponding convergence results condensed in Figure 17.

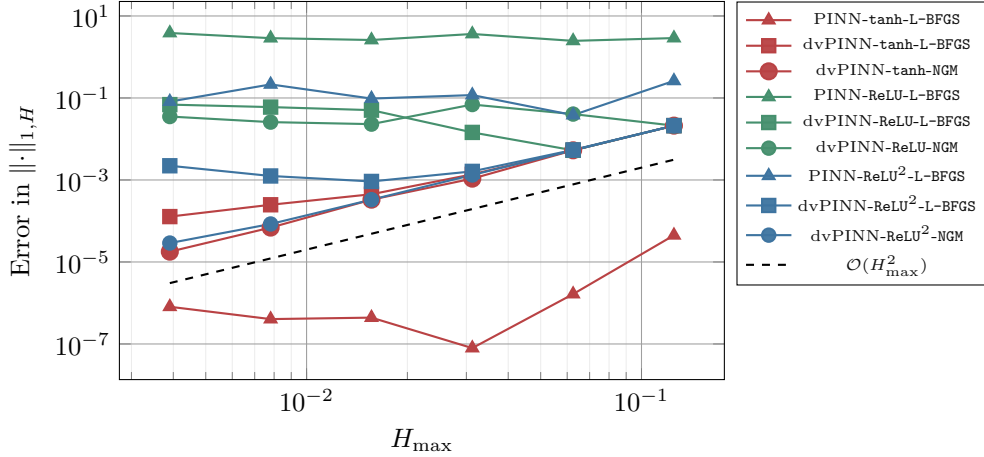


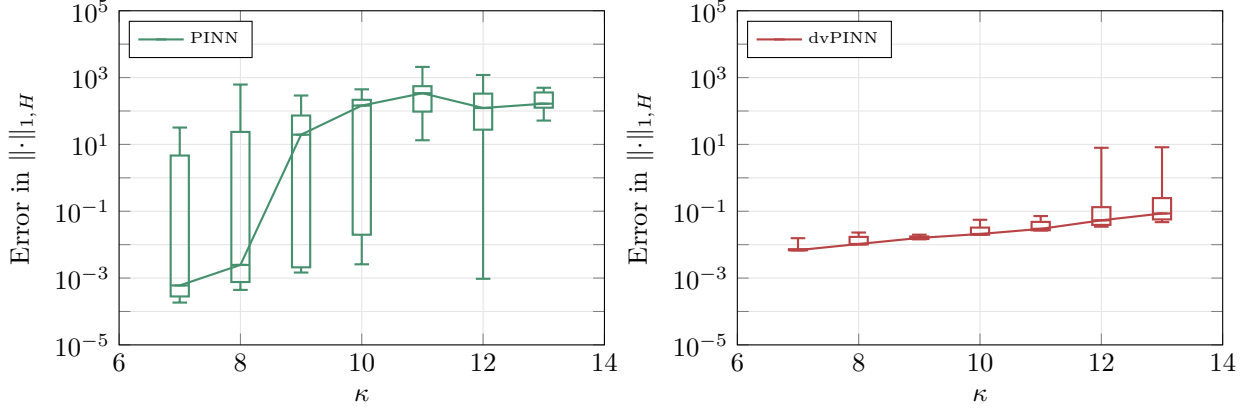
FIGURE 17. Comparative test between several activation functions.

On one hand, our tests showed that using ReLU as an activation function does not produce good results overall. It kept the error relatively constant and at large values even when refining the mesh. On the other hand, we see that the combination of PINNs+tanh+L-BFGS performed best. Nonetheless, we note the close agreement between dvPINNs+ReLU² and dvPINNs+tanh under NGM, as well as their convergence with respect to $H_{\max}$. Even using L-BFGS, both combinations performed relatively well until plateauing, which was expected given our previous tests.

4.2. Mitigation of spectral bias in highly oscillatory regimes. It is well-known that highly oscillatory solutions are more difficult to learn for classical PINNs. Indeed, PINNs learn better lower frequencies leading to a phenomenon called *spectral bias* [33, 43]. While there are NN specifically designed to go around this issue for the Stokes flow [39], we want to put the dvPINN framework to the test by considering a sequence of problems with increasingly oscillatory solutions. To compare the results, we shall train the classical PINNs approach (using the corresponding grid from the numerical method associated with the dvPINN as a training set) as well as the dvPINN with the more standard ADAM+BFGS optimisation scheme, described in Appendix A (even though NGM would be the best optimiser for dvPINNs, we deploy the standard optimisation routines for classical PINNs to ensure they had the best chance). Considering $\Omega = (-1, 1)$ and the wavenumber $\kappa \in \mathbb{N}$, both methods will be trained to approximate $u(x) = \sin(\kappa\pi x)$, $x \in \bar{\Omega}$, solution of the problem

$$\begin{cases} -u'' = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \quad (24)$$

where f is calculated enforcing u as solution. Both methods were trained in the finest mesh corresponding to $N = 512$, with 3 hidden layers having 50 neurons each. In Figure 18, we show the comparison between the classical PINN and the dvPINN, for each wavenumber κ that represents the number of full oscillations in the domain. To overcome hypothetical bad initialisation, we reinitialised the problem 10 times and built standard box-and-whiskers plots.



(A) Box-and-whiskers error plot, for classical PINNs, for highly oscillatory regime.

(B) Box-and-Whiskers error plot for dvPINNs, for highly oscillatory regime.

FIGURE 18. Error, on strongly imposed Dirichlet boundary conditions, with respect to full oscillations κ .

It is clear that for $\kappa = 9$ onwards, the median of classical PINNs errors is far above what the error should be if the method converged. Even for smaller κ values, classical PINN can reach lower errors (compared to dvPINN) but have a much higher interquartile range, which implies a nonconsistent convergence. Regarding dvPINN, we observe a small interquartile range, confirming the reliability of the method. While there are a few known tricks that can be used to improve classical PINN convergence in highly oscillatory regimes (e.g., using a periodic activation function [38]), we highlight the ability of dvPINNs to surpass these difficulties without further tuning.

4.3. Robustness under reduced regularity. Another possible drawback of classical PINNs is the need for higher solution regularity than variational methods require. Consider the one-dimensional Poisson problem with Dirichlet boundary conditions (12), where $\Omega = (-1, 1)$ and

$$f(x) = \begin{cases} -(4\alpha + 2)x^\alpha - \alpha(\alpha - 1)(x^2 - 1)x^{\alpha-2}, & x > 0 \\ 0, & x = 0, \\ -2(-x)^\alpha + 4\alpha x(-x)^{\alpha-1} - \alpha(\alpha - 1)(x^2 - 1)(-x)^{\alpha-2}, & x < 0 \end{cases},$$

where α is a positive parameter. While this might seem a peculiar problem, it was built such that the solution of (12) is

$$u(x) = (x^2 - 1)|x|^\alpha, \quad x \in \overline{\Omega} \quad (25)$$

where α dictates the regularity of the solution. In fact, if $\alpha \in (3/2, 5/2]$, then $u \in H^2$ while $u \notin H^3$. In Figure 19, we show the results from training a sequence of dvPINNs with underlying increasingly finer meshes, and plotting the error by comparing the dvPINNs output with (25), in the discrete H^1 -norm.

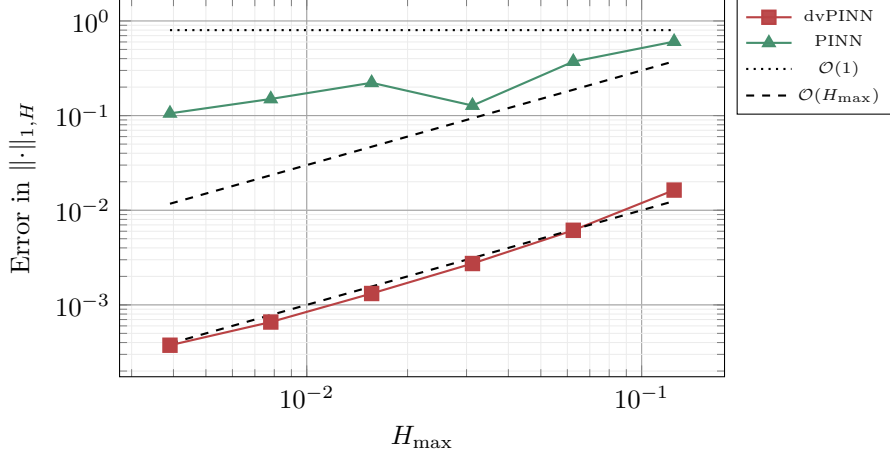


FIGURE 19. Error convergence with respect to $H_{\max}$, for PINNs and dvPINNs, approximating (25) with $\alpha = 1.51$.

It is clear that dvPINN performs better. It reaches smaller errors and follows a first-order convergence rate. The poor behaviour of PINNs is explained by the regularity requirements: we need $u''_{\theta} \in C^0$, thus implying that $u_{\theta} \in C^2$. Since gradient-based optimisers always require an additional derivative, we need at least $u \in C^3$ which, by construction, is not available.

5. ALGORITHMIC SCALING AND GENERALISATION

We are now in a position to control the three sources of error to successfully train dvPINNs. Hence, the real applicability of the dvPINN method is evaluated through standard neural network's tests. In particular, while Section 3.1 shows that with enough epochs, the training error can be controlled, one might be interested in reducing the computational load and severely reducing the number of epochs needed. One possible approach is described in Section 5.1.

While, in general, the error of a dvPINN with respect to the training data decreases, since it repeatedly optimises to fit this data, a natural question is whether the dvPINN can generalise the solution for new data points, unseen during training process. This will be tackled in Section 5.2

5.1. Multilevel training via mesh refinement. Since the dvPINN's loss function requires the entire discretisation to be well-defined, a standard mini-batch training routine is not directly applicable. One possible alternative is a multilevel training via mesh refinement, where the parameters obtained on a coarse grid training are used to initialise the optimisation on a finer grid.

To test this idea, we shall train two ADF-dvPINNs, in the same setting as Section 4.2, with $\kappa = 6$, for the usual sequence of decreasing $H_{\max}$ meshes. The chosen targeted problem is arbitrary, but should be one hard enough to expose the slow convergence for the finer meshes. Therefore, we will train the ADF-dvPINNs using three different approaches: (i) the classical direct training from a random initialisation; (ii) we fully train sequentially from $N = 16$ to $N = 512$, using each calibrated parameter as the initialisation for the next finer mesh; (iii) we also train sequentially, but only for 500 epochs before mandatorily skipping to the next level.

Figure 20 shows the error log for the three training routines, clipped at 2000 epochs (due to optimiser stagnation). The data for $N = 16$ and $N = 32$ is not displayed because, for such coarse meshes, the methodology does not make sense, though they are coarse enough to train from random initialisations. Nonetheless, they were used to sequentially initialise the parameters for $N = 64$, and consequently, the subsequent values of N , for the second and third proposed routines.

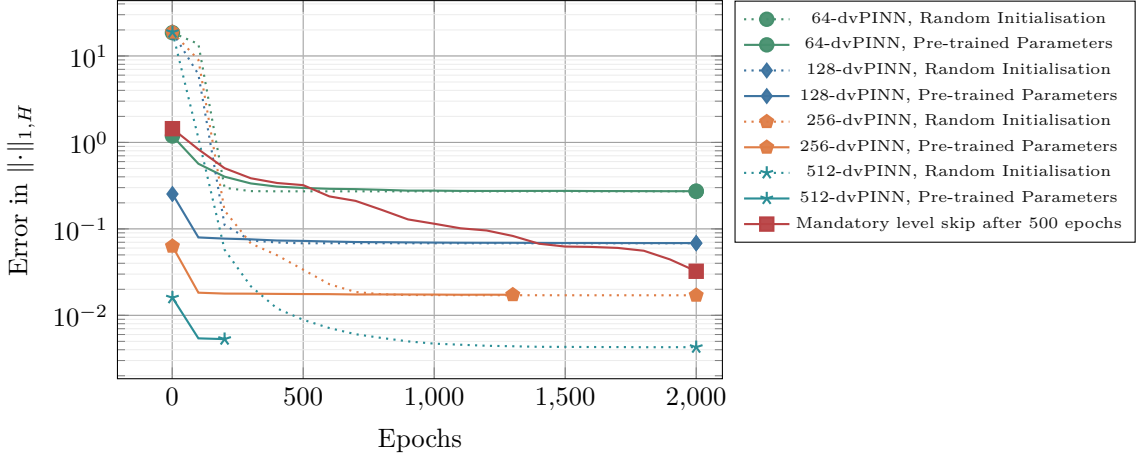


FIGURE 20. Error decay along epochs, for ADF-dvPINN, comparing random initialisation versus pre-trained parameters.

There are clear advantages of using the Multilevel Training routine, since we observe that the pre-trained dvPINNs reach the optimal error with very few iterations. This is particularly noticeable for the finest meshes, where the convergence with respect to epochs is very slow. This shows that the benefits from using this multilevel approach get better as N increases.

Regarding the mandatory level skip after 500 iterations, we observe that it is worse than a fresh random initialisation, for all N presented. Perhaps this could be tuned, where the epoch skip interval is dynamic with respect to the level (making more epochs on the coarser meshes and conversely less epochs in the finer meshes). The advantage of the second training routine is that, since we only skip levels when the training stops, we should be a lot closer to the true minimum. When skipping too soon, the optimiser might need time (epochs) to readjust the weights for a favourable direction. These results indicate that multilevel warmup-starting can further accelerate dvPINN training.

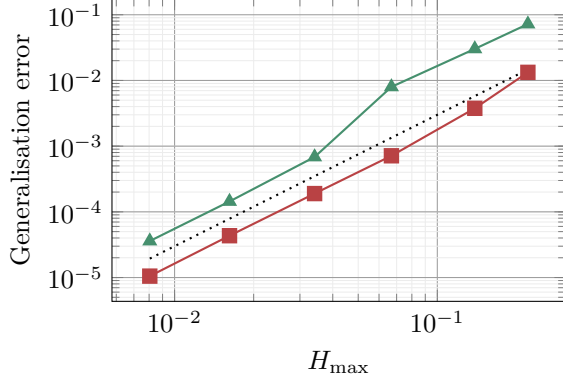
5.2. Generalisation capability on unseen evaluation points. Another typical numerical experiment to evaluate any sort of NN is its ability to express the solution on a new data set not seen during training. This is commonly addressed as the generalisation capability, or out-of-sample error, of a NN. In particular, we evaluate the generalisation error with respect to $H_{\max}$ of the training mesh. To that end, similarly to before, we train several dvPINNs for a sequence of uniform meshes with decreasing $H_{\max}$, with the finest being $N = 512$. After training, we calculate the generalisation error in the L^2 norm and the H^1 seminorm for a random sample of points (4096 points in 1D, 512×512 grid for 2D). The metrics are calculated using the composite trapezoidal rule, and the gradient for the seminorm is computed via automatic differentiation.

The results in Figure 21 illustrate the experiment for all the tackled problems. Both metrics exhibit second-order convergence with respect to $H_{\max}$. While this was expected for the training set, the fact that, for a completely new set of points, the error remains comparable in magnitude to the error already introduced by the training mesh supports the heuristic that a dvPINN trained on a finer mesh will better generalise to unseen points.

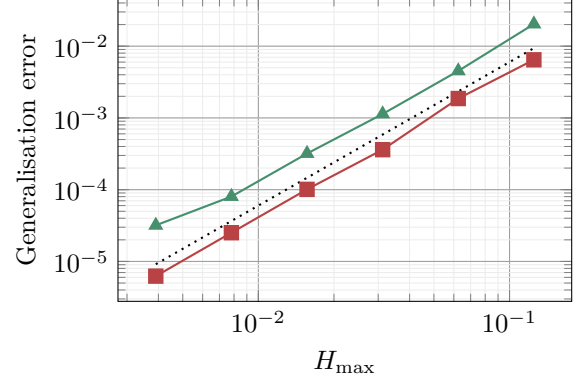
6. CONCLUDING REMARKS AND FUTURE WORK

In this work, we presented a computational assessment and empirical analysis of discrete variational physics-informed neural networks. By shifting the learning process from the strong (pointwise) form to a discrete variational formulation, we established a loss function with theoretical guarantees of a unique minimiser in the context of FEMs. We identified potential sources of error as optimisation error, sampling density, and neural network capacity, with a per error computational analysis.

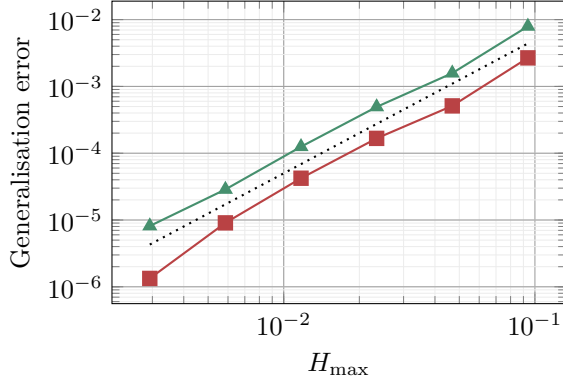
Our numerical experiments on the one-dimensional and two-dimensional Poisson problems, the bi-harmonic problem, and the steady linear elasticity problem showed that the NGM outperforms standard optimisers like ADAM and L-BFGS for dvPINNs, avoiding stagnation and converging in a fraction of the epochs. Also, isolating the density sampling error revealed that dvPINNs inherit the convergence order from the underlying numerical method, even when measuring the error using continuous norms. This is surprising since the numerical methods do not, in general, possess this property. The conducted analysis



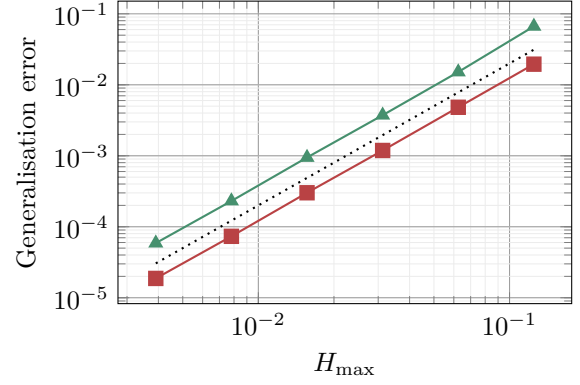
(A) 1D Dirichlet Poisson, problem (12).



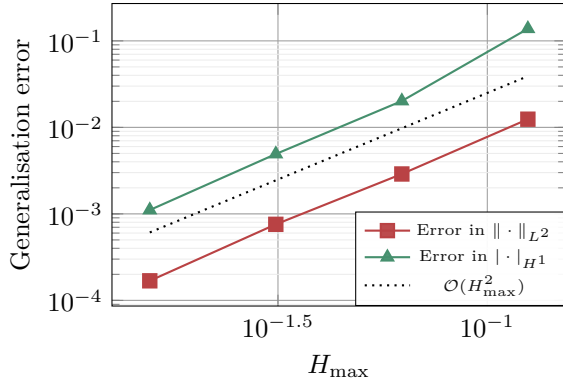
(B) 1D Poisson with Neumann BC, problem (15).



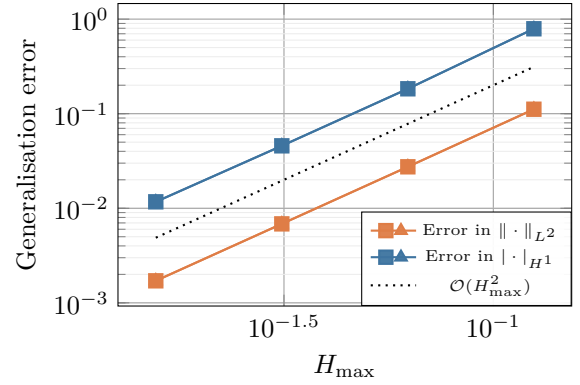
(C) 1D Poisson with Robin BC, problem (17).



(D) 1D Biharmonic with BC, problem (19).



(E) 2D Dirichlet Poisson, problem (21).



(F) Steady linear elasticity, problem (22).

FIGURE 21. Generalisation errors.

validated the proposed *ansatz* relating the parameter space dimension to the grid's resolution, showing it to be a good rule of thumb, though not sharp.

Our benchmarking tests confirmed that dvPINNs possess advantages over classical PINNs. They mitigate the *spectral bias* phenomenon, learning highly oscillatory solutions without additional tuning. Additionally, dvPINNs show a robust behaviour with reduced-regularity solutions, where classical PINNs completely fail, while also displaying optimal generalisation capability to learn the solution on out-of-sample data. Although a standard mini-batch routine is not directly applicable, the computational costs can be significantly reduced using a multilevel mesh-refinement training routine.

Several theoretical questions remain open. In particular, establishing rigorous error bounds for the empirical error components would allow us to clearly confirm the inheritance of the underlying numerical method's order onto dvPINNs. We expect the *ansatz* (or a similar version) to be theoretically derived as a balancing condition between the numerical method's order and the parameter space dimension.

ACKNOWLEDGMENTS

The authors acknowledge financial support by the Centre for Mathematics of the University of Coimbra (CMUC, <https://doi.org/10.54499/UID/00324/2025>) under the Portuguese Foundation for Science and Technology (FCT), Grants UID/00324/2025 and UID/PRR/00324/2025. Diogo Cotrim was also supported by FCT - Fundação para a Ciência e Tecnologia, I.P., under project reference 2025.03926.BD and DOI identifier <https://doi.org/10.54499/2025.03926.BD>.

APPENDIX A. OPTIMISATION ALGORITHM PARAMETERS

We present below the parameters used for all the employed algorithms.

ADAM warmup-stage (via <code>adamupdate</code>)		
Parameter	Value	Notes
Learning rate	10^{-3}	Decayed by 0.9 every 100 epochs
$\beta_1, \beta_2, \epsilon$	Matlab defaults	(0.9, 0.999, 10^{-8}) Not set explicitly
Maximum iterations	1000	Stopping condition: small warmup-stage

TABLE 2. Parameters for ADAM optimisation algorithm.

BFGS/L-BFGS (via <code>fminunc</code>)		
Parameter	Value	Notes
Algorithm	Quasi-Newton	
Hessian update	BFGS/L-BFGS	One or the other, accordingly
Gradient supplied	<code>true</code>	Analytic gradients via <code>SpecifyObjectiveGradient</code>
Maximum iterations	10000	Stopping condition
Optimality tolerance	10^{-10}	Stopping condition (gradient norm)
Step tolerance	10^{-12}	Stopping condition (minimum step size)
Function tolerance	10^{-14}	Stopping condition (change in objective)

TABLE 3. Parameters for BFGS/L-BFGS optimisation algorithm.

Natural Gradient Method		
Parameter	Value	Notes
Learning rate	1.0	
Backtracking factor	0.5	Geometric shrink factor per line-search step
Damping factor	10^{-7}	Tikhonov regularisation added to Gram matrix diagonal
Direction solver	Direct	Computes the full Gram Matrix
Metric update freq.	1	Recompute the Gram metric every iteration
Step candidates	0 : 15	16 candidate steps per iteration
Maximum iterations	5000	Stopping condition
Error roll back	5	Stopping condition: If the error increases for 5 consecutive iterations, roll back the parameters to the last “good” state
Line search stall	2	Stopping condition: No candidate step in the 16 candidates improved the loss even after a fresh metric

TABLE 4. Parameters for NGM optimisation algorithm.

REFERENCES

- [1] L. Amoudruz, S. Litvinov, C. Papadimitriou, and P. Koumoutsakos, *Bayesian inference for PDE-based inverse problems using the optimization of a discrete loss*, Computer Methods in Applied Mechanics and Engineering **455** (2026), 118903.
- [2] S. Barbeiro, J. A. Ferreira, and R. D. Grigorieff, *Supraconvergence of a finite difference scheme for solutions in $H^s(0, L)$* , IMA Journal of Numerical Analysis **25** (2005), no. 4, 797–811.
- [3] S. Berrone, C. Canuto, and M. Pintore, *Variational Physics Informed Neural Networks: the Role of Quadratures and Test Functions*, Journal of Scientific Computing **92** (2022), no. 3, 100.
- [4] I. Bioli, C. Marcati, and G. Sangalli, *Accelerating Natural Gradient Descent for PINNs with Randomized Numerical Linear Algebra* (2026), available at [2505.11638](https://arxiv.org/abs/2505.11638). <https://arxiv.org/abs/2505.11638>.
- [5] A. Bonito, R. DeVore, G. Petrova, and J. W. Siegel, *Convergence and error control of consistent PINNs for elliptic PDEs*, IMA Journal of Numerical Analysis **46** (2026), no. 1, 90–148.
- [6] C. G. Broyden, *The convergence of a class of double-rank minimization algorithms 1. General considerations*, IMA Journal of Applied Mathematics **6** (1970), no. 1, 76–90.
- [7] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. Karniadakis, *Physics-informed neural networks (PINNs) for fluid mechanics: a review*, Acta Mechanica Sinica **37** (2022), 1727–1738.
- [8] W. Cao and W. Zhang, *Overcoming the loss conditioning bottleneck in optimization-based PDE solvers: a well-conditioned loss function*, Communications in Nonlinear Science and Numerical Simulation **160** (2026), 109952.
- [9] V. L. Carvalho and J. C. F. Pereira, *Comparative Study of Physics-Informed Neural Networks (PINNs) and Optimizing a Discrete Loss (ODIL) for Forward and Inverse Problems in Unsteady Non-Linear Partial Differential Equations*, Computation **14** (2026), no. 8.
- [10] P.-H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, and Y.-S. Ong, *CAN-PINN: A fast physics-informed neural network based on coupled-automatic-numerical differentiation method*, Computer Methods in Applied Mechanics and Engineering **395** (2022), 114909.
- [11] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, *Scientific Machine Learning Through Physics-Informed Neural Networks: Where we are and What's Next*, Journal of Scientific Computing **92** (2022), no. 3, 88.
- [12] A. D. Jagtap and G. Karniadakis, *Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations*, Communications in Computational Physics **28** (2020), no. 5, 2002–2041.
- [13] M. W. M. G. Dissanayake and N. Phan-Thien, *Neural-network-based approximations for solving partial differential equations*, Communications in Numerical Methods in Engineering **10** (1994), no. 3, 195–201.
- [14] W. E and B. Yu, *The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems*, Communications in Mathematics and Statistics **6** (2018), no. 1, 1–12.
- [15] A. Fernandes, J. A. Ferreira, and G. Pena, *A parabolic–elliptic–hyperbolic system of Keller–Segel type: Numerical analysis and simulation*, Mathematics and Computers in Simulation **252** (2027), 189–210.
- [16] J. A. Ferreira and R. D. Grigorieff, *Supraconvergence and Supercloseness of a Scheme for Elliptic Equations on Nonuniform Grids*, Numerical Functional Analysis and Optimization **27** (2006), no. 5–6, 539–564.
- [17] J. A. Ferreira and G. Pena, *FDM/FEM for nonlinear convection-diffusion-reaction equations with Neumann boundary conditions – Convergence analysis for smooth and nonsmooth solutions*, Journal of Computational and Applied Mathematics **446** (2024), 115866.
- [18] R. Fletcher, *A new approach to variable metric algorithms*, The Computer Journal **13** (1970), no. 3, 317–322.
- [19] R. Frangella, J. A. Tropp, and M. Udell, *Randomized Nyström Preconditioned Conjugate Gradients for Solving Large Scale Linear Systems*, SIAM Journal on Matrix Analysis and Applications **44** (2023), no. 4, 1565–1592.
- [20] D. Goldfarb, *A family of variable-metric methods derived by variational means*, Mathematics of Computation **24** (1970), no. 109, 23–26.
- [21] J. He, L. Li, J. Xu, and C. Zheng, *ReLU Deep Neural Networks and Linear Finite Elements*, Journal of Computational Mathematics **38** (2020), no. 3, 502–527.
- [22] A. D. Jagtap, E. Kharazmi, and G. Karniadakis, *Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems*, Computer Methods in Applied Mechanics and Engineering **365** (2020), 113028.
- [23] P. Karnakov, S. Litvinov, and P. Koumoutsakos, *Solving inverse problems in physics by optimizing a discrete loss: Fast and accurate learning without neural networks*, PNAS Nexus **3** (2024), no. 1, pgae005.
- [24] G. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, *Physics-informed machine learning*, Nature Reviews Physics **3** (2021), no. 6, 422–440.
- [25] E. Kharazmi, Z. Zhang, and G. Karniadakis, *hp-VPINNs: Variational physics-informed neural networks with domain decomposition*, Computer Methods in Applied Mechanics and Engineering **374** (2021), 113547.
- [26] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, 3rd International Conference on Learning Representations (ICLR), 2015.
- [27] I. E. Lagaris, A. C. Likas, and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE transactions on neural networks **9** 5 (1997), 987–1000.
- [28] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature **521** (2015), no. 7553, 436–444.
- [29] M. Li, T. Zhang, Y. Chen, and A. J. Smola, *Efficient mini-batch training for stochastic optimization*, Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 661–670.
- [30] S. Mishra and R. Molinaro, *Estimates on the generalization error of physics-informed neural networks for approximating PDEs*, IMA Journal of Numerical Analysis **43** (2023), no. 1, 1–43.
- [31] J. Müller and M. Zeinhofer, *Achieving High Accuracy with PINNs via Energy Natural Gradient Descent*, Proceedings of the 40th International Conference on Machine Learning, 2023, pp. 25471–25485.

- [32] J. Nocedal, *Updating Quasi-Newton Matrices with Limited Storage*, Mathematics of Computation **35** (1980), no. 151, 773–782.
- [33] N. Rahaman, A. Baratin, D. Arpit, F. Dräxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. C. Courville, *On the Spectral Bias of Neural Networks*, International Conference on Machine Learning, 2018.
- [34] M. Raissi, P. Perdikaris, and G. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics **378** (2019), 686–707.
- [35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and Li F.-F., *ImageNet Large Scale Visual Recognition Challenge*, International Journal of Computer Vision **115** (2015), no. 3, 211–252.
- [36] D. F. Shanno, *Conditioning of Quasi-Newton Methods for Function Minimization*, Mathematics of Computation **24** (1970), no. 111, 647–656.
- [37] Y. Shin, J. Darbon, and G. Karniadakis, *On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs*, Communications in Computational Physics **28** (2020), no. 5, 2042–2074.
- [38] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, *Implicit neural representations with periodic activation functions*, Proceedings of the 34th International Conference on Neural Information Processing Systems, 2020.
- [39] B. Wang, W. Zhang, and W. Cai, *Multi-Scale Deep Neural Network (MscaleDNN) Methods for Oscillatory Stokes Flows in Complex Domains*, Communications in Computational Physics **28** (2020), no. 5, 2139–2157.
- [40] S. Wang, Y. Teng, and P. Perdikaris, *Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks*, SIAM Journal on Scientific Computing **43** (2021), no. 5, A3055–A3081.
- [41] X. Wang and Z.-Y. Yin, *Physics-encoded convolutional attention network for forward and inverse analysis of spatial-temporal parabolic dynamics considering discontinuous heterogeneity*, Computer Methods in Applied Mechanics and Engineering **442** (2025), 118025.
- [42] X. Wang, Z.-Y. Yin, W. Wu, and H.-H. Zhu, *Differentiable finite element method with Galerkin discretization for fast and accurate inverse analysis of multidimensional heterogeneous engineering structures*, Computer Methods in Applied Mechanics and Engineering **437** (2025), 117755.
- [43] Z.-Q. J. Xu, L. Zhang, and W. Cai, *On understanding and overcoming spectral biases of deep neural network learning methods for solving PDEs*, Journal of Computational Physics **530** (2025), 113905.
- [44] L. Yali, Z. Yiye, Z. Heng, Z. Mingjie, W. Gang, W. Jingyu, and D. Xiaogang, *Optimizing a discrete loss with finite-difference-based physical constraints and time-stepping for solving PDEs*, Acta Mechanica Sinica (2026).
- [45] J. Yuan, L. Zeng, Y. Shi, Y. Gui, B. Xia, and D. Wei, *Discretization-informed neural networks for solving inverse heat conduction problems*, International Communications in Heat and Mass Transfer **176** (2026), 111359.
- [46] M. Zeinhofer, R. Masri, and K.-A. Mardal, *A unified framework for the error analysis of physics-informed neural networks*, IMA Journal of Numerical Analysis **45** (202411), no. 5, 2988–3025.

UNIVERSITY OF COIMBRA, CMUC, DEPARTMENT OF MATHEMATICS, FCTUC, LARGO D. DINIS, 3000–143 COIMBRA, PORTUGAL

Email address: cotrimnunes@uc.com, ferreira@mat.uc.pt, lino@deec.uc.pt, goncalo.pena@uc.pt