Physics-Informed Neural Networks: Minimizing Residual Loss with Wide Networks and Effective Activations

Nima Hosseini Dashtbayaz1, Ghazal Farhani2,*, Boyu Wang1,3,* and Charles X. Ling1

1Department of Computer Science, University of Western Ontario
2National Research Council of Canada
3Vector Institute

Corresponding Author

nhosse5@uwo.ca, ghazal.farhani@nrc-cnrc.gc.ca, bwang@csd.uwo.ca, charles.ling@uwo.ca

Abstract

The residual loss in Physics-Informed Neural Networks (PINNs) alters the simple recursive relation of layers in a feed-forward neural network by applying a differential operator, resulting in a loss landscape that is inherently different from those of common supervised problems. Therefore, relying on the existing theory leads to unjustified design choices and suboptimal performance. In this work, we analyze the residual loss by studying its characteristics at critical points to find the conditions that result in effective training of PINNs. Specifically, we first show that under certain conditions, the residual loss of PINNs can be globally minimized by a wide neural network. Furthermore, our analysis also reveals that an activation function with well-behaved high-order derivatives plays a crucial role in minimizing the residual loss. In particular, to solve a $k$-th order PDE, the $k$-th derivative of the activation function should be bijective. The established theory paves the way for designing and choosing effective activation functions for PINNs and explains why periodic activations have shown promising performance in certain cases. Finally, we verify our findings by conducting a set of experiments on several PDEs. Our code is publicly available at https://github.com/nimahsn/pinns_tf2.

1 Introduction

The success of deep learning in a wide variety of tasks has motivated its application in scientific domains as well [Sirignano and Spiliopoulos, 2018; Reiser et al., 2022; Li et al., 2020]. PINNs [Raissi et al., 2017] in particular are designed to solve differential equations as an alternative to traditional solvers, benefiting from discretization-free construction and the vast availability of machine learning tools and techniques. As a result, PINNs have been deployed in various physics and engineering problems, such as solving inverse scattering problems in photonics [Chen et al., 2020a], flow problems in fluid dynamics [Cai et al., 2021], and computational musculoskeletal models raised in biomedical and rehabilitation sciences [Zhang et al., 2022].

Consider a general-form PDE with a Dirichlet boundary condition such as

$$
\mathcal{D}\left[ u \right] (x) = f(x) \quad x \in \Omega
$$

$$
\quad u(x) = g(x) \quad x \in \partial\Omega,
$$

where $u$ is the solution of the PDE on a bounded domain $\Omega \subset \mathbb{R}^d$ of $d$ independent variables with boundaries $\partial\Omega$, $f$ and $g$ are known functions, and $\mathcal{D}$ is a differential operator. Here, the operator $\mathcal{D}$ expresses the physical rules governing $u$ through a differential expression. PINNs are then trained to respect the underlying physical dynamics given in $\mathcal{D}$ by minimizing the residual loss

$$
L_r = \sum_{x \in \mathbb{X}} l(\mathcal{D}[\hat{u}](x) - f(x)),
$$

where $\hat{u}$ is a neural network approximation of $u$, $l$ is an error function such as squared error, and $\mathbb{X}$ is a set of training collocation points in $\Omega$. To guarantee a unique solution, boundary (and initial) conditions are also imposed by adding other supervised loss terms, referred to as boundary loss, trained with boundary data sampled from $\partial\Omega$. The resulting loss function can then be treated as a multi-objective optimization task [Raissi et al., 2017].

While proven effective, training PINNs is often a challenging task. These challenges usually originate from either the discrepancy between the residual loss and the boundary loss [Wang et al., 2020b; Farhani et al., 2022; Wang et al., 2020a] or the nature of the residual loss [Krishnapriyan et al., 2021; Wang et al., 2022]. Notably, as Eq. 2 involves differentiation over a neural network with respect to (w.r.t.) its inputs, the outputs of the network undergo a significant structural transformation. To better understand the aforementioned process and its implications, let us consider a simple differential operator $\frac{d}{dx}$ of a single independent variable and an $L$-layer feed-forward network with an activation function $\sigma$. One can find that the application of this differential operator on the neural network, $\mathcal{D}[\hat{u}]$, is given by

$$
\mathcal{D}[\hat{u}] = W_L^\top \times (\sigma'(G_{L-1}) \circ W_{L-1})^\top \times \cdots \times (\sigma'(G_1) \circ W_1)^\top,
$$

where $W_i$ and $G_i$ are weights and linear outputs of layer $i$, and $\circ$ and $\times$ denote element-wise (Hadamard) and matrix

*Corresponding Author
products. In contrast, the original neural network $\hat{u}$ can be defined recursively as

$$\hat{u}(x) = G_L(x), \quad G_i(x) = \sigma(G_{i-1}(x)) \times W_i + b_i.$$ 

Eq. 3 shows how the differentiation transforms the outputs of a neural network. Firstly, note that the simple recursive relation between the layers of a feed-forward network is disrupted by applying $D$, and additional element-wise products with weights emerge as well. More significantly, we observe that the derivative $\sigma'$ of the activation function appears in the outputs. This presence of $\sigma'$ in PINNs highlights the importance of an activation function with well-behaved derivatives in the model’s expressive power in learning $D$ and likewise in the optimization process as it involves higher-order derivatives of $\sigma$.

Altogether, the distinct characteristics of $D[u]$ and the resulting residual loss, contribute to a problem that is quite different from common supervised training tasks. Consequently, the existing theory around loss functions and their characteristics cannot readily be applied to PINNs, and the lack of understanding about PINNs and their optimization dynamics leads to uninformed design choices and suboptimal performance even for seemingly easy PDEs.

In this work, we focus on the residual loss and its landscape. Specifically, we are interested in finding what neural networks and design choices enable PINNs to globally minimize the residual loss. To this end, we study the residual loss at a critical point of the network parameter space and search for distinctive characteristics of a global minimum compared to other critical points. Once these characteristics are identified, our investigation shifts towards determining sufficient conditions within the network design, in particular, width and activation function, that guarantee the existence of global minima in the parameter space. Our findings underscore the importance of the width and activation functions with well-behaved high-order derivatives in acquiring a high expressive power in learning the differential operator. Finally, we verify our findings by conducting extensive experiments on several PDEs.

We summarize our contributions as follows. (1) We theoretically show that the residual loss of PINNs can be globally minimized, given a two-layer neural network with a width equal to or greater than the number of collocation points. (2) Through our analysis, we establish that the residual loss for a $k$-th order differential operator is optimally minimized when using an activation function with a bijective $k$-th order derivative. We leverage this theoretical foundation as a guideline for selecting activation functions, justifying the choice of sinusoidal activations, and subsequently validating their effectiveness through empirical demonstrations and experiments.

2 Related Works

2.1 Wide Neural Networks

Wide neural networks have historically been of significant interest in machine learning. With classical results such as Universal Approximation and Gaussian processes, and more recently, NTK theory [Jacot et al., 2018], wide networks have been studied to understand neural networks in certain regimes [Chen et al., 2020b; Lee et al., 2019]. The optimal width of a neural network is also studied for convergence guarantees [Oymak and Soltanolkotabi, 2020; Du et al., 2019; Allen-Zhu et al., 2019; Nguyen and Mondelli, 2020] and loss geometry [Safran and Shamir, 2016; Nguyen and Hein, 2017] with certain classes of neural networks and optimizers. The convergence guarantees are often provided for a width polynomial in the number of training samples and the number of layers [Allen-Zhu et al., 2019]. Safran and Shamir [2016] studied the basins of the loss function for wide two-layer ReLU networks, showing that wider networks are initialized at a good basin with higher probability. [Nguyen and Hein, 2017] also showed that most of the critical points in a wide neural network are also global minima. The developed theory in most of the aforementioned works cannot be directly applied to PINNs, as they either rely on specific neural network formulations [Nguyen and Hein, 2017; Nguyen and Mondelli, 2020], which are disrupted by differentiation, or certain hyper-parameters that are not effective for PINNs, such as ReLU activation function [Du and Hu, 2019; Safran and Shamir, 2016; Allen-Zhu et al., 2019].

2.2 Periodic Activation Functions

Sitzmann et al. [2020] proposed using sinusoidal activation functions in neural networks with low-dimensional inputs for learning differentiable signals. Notably, they also showed the capability of Sine networks in solving Wave and Helmholtz PDEs with PINNs. Since then, few works have explored the behaviour of neural networks with periodic activation functions at initialization [Belbute-Peres and Kolter, 2022] and their expressive power as function approximators [Meronen et al., 2021]. Meronen et al. [Meronen et al., 2021] studied the inductive bias introduced by periodic activation functions on the neural network functional space, and showed that such networks are less sensitive to input shifts.

2.3 Physics-Informed Neural Networks

Besides the applications of PINNs in solving various PDEs, there has been a surge in analyzing the behaviour and pitfalls of PINNs in recent years, especially from the optimization perspective [Wang et al., 2020a; Liu et al., 2020; Farhani et al., 2022]. Using Neural Tangent Kernel (NTK) theory from infinitely wide neural networks, [Wang et al., 2020b] showed that high-frequency terms in a PDE result in discrepancies in the convergence rate between the loss objectives when trained with Gradient Descent, leading the model to exhibit behaviours similar to spectral bias [Rahaman et al., 2018]. Wang et al. [2020a] also showed similar results by studying the magnitude of the loss gradients at different layers. Later on, the momentum term was shown to address the discrepancy in optimization in the infinite-width regime [Farhani et al., 2022].

Many recent works alleviate the optimization challenges in PINNs and improve their performance by assigning weights to each loss term [Wang et al., 2020b; McClenny and Braganeto, 2020; Wight and Zhao, 2020], designing new architectures and embeddings [Wong et al., 2022; Wang et al., 2020a; Wang et al., 2021; Dong and Ni, 2021], and using sophisticated training strategies such as curriculum learning [Kr-
ishnapriyan et al., 2021; Wang et al., 2022]. Among them, [Wang et al., 2020b] and [Wong et al., 2022], suggested mapping the inputs to random or trainable Fourier features and the use of sinusoidal activation functions to overcome the spectral bias and the convergence discrepancy.

3 Global Minima of the Residual Loss

In this section, we study the residual loss at its critical points to obtain sufficient conditions for the existence of global minima. We present the lemmas and theorems for a simple differential operator with a single independent variable and generalize in the Supplementary Material. First, we introduce the notation and the setup used throughout this section.

3.1 Notation and Setup

We use \( \tilde{u}_W : \mathbb{R}^d \times \mathbb{R}^{\mid W \mid} \to \mathbb{R}^n \) to denote an \( L \)-layer feedforward neural network parameterized by \( W = \{ W_i, b_i \mid 1 \leq i \leq L, W_i \in \mathbb{R}^{n_i \times n_{i+1}}, b_i \in \mathbb{R}^{n_i} \} \), where \( n_i \) is the number of neurons in layer \( i \), \( n_0 = d \), and \( n_L = 1 \). We drop \( W \) from \( \tilde{u}_W \) for simplicity if there is no ambiguity. The neural network \( \tilde{u} \) for an input \( x = (x_1, \ldots, x_d) \) is formulated as

\[
\tilde{u}(x) = G_L(x), \\
G_i(x) = F_{i-1}(x) \times W_i + b_i \quad \forall i \in \{1, \ldots, L\}, \\
F_i(x) = \sigma(G_i(x)) \quad \forall i \in \{1, \ldots, L-1\},
\]

where \( \sigma \) is an activation function, and \( F_0(x) = x \). We further define \( F_i^{(k)}(x) \) as

\[
F_i^{(k)}(x) = \sigma^{(k)}(G_i(x)),
\]

where \( \sigma^{(k)} \) is the \( k \)-th derivative of \( \sigma \). In the case of \( k = 1 \), we simply use \( F_i \) and \( \sigma \) instead. For a batch of \( x \) of \( N \) samples, \( F_i(x), G_i(x), \) and \( F_i^{(k)}(x) \) are \( N \times n_i \) matrices. Also, the matrix power \( W^k \) represents an element-wise power.

In a PINN, the neural network \( \tilde{u} \) is trained to approximate the solution \( u \) of a differential equation denoted as in Eq. 1. In this work, we assume that \( \mathcal{D} \) is a linear differential operator, i.e., the PDE is linear in the derivatives of \( u \) and \( u \) itself. We reformulate the residual loss in Eq. 2 to be a function of weights \( \mathcal{W} \) and choose \( l(r) = r^2 \).

\[
\phi_r(x; \mathcal{W}) = \sum_{x \in \mathcal{X}} l(\mathcal{D}[\tilde{u}](x) - f(x)).
\]

Throughout the rest of this section, we consider a two-layer neural network and a simple \( k \)-th order differential operator \( \mathcal{D}[u] = \frac{\partial^k u}{\partial x^k} \) for a single independent variable \( x \) (i.e., \( d = 1 \)).

3.2 Residual Loss of a Two-layer PINN

To study the residual loss and its critical points, we first need to derive the analytic formula for the residual loss and its gradients. The next two lemmas, provide us with these tools by finding the differentiation \( \mathcal{D}[\tilde{u}] \) and then deriving the gradients of the resulting residual loss.

Lemma 1. For a two-layer neural network \( \tilde{u} \) defined in Eq. 4, and a \( k \)-th order differential operator \( \mathcal{D}[u] = \frac{\partial^k u}{\partial x^k} \) of a single independent variable \( x \), \( \mathcal{D}[\tilde{u}] \) is

\[
\mathcal{D}[\tilde{u}](x) = W_2^T \times (F_1^{(k)}(x) \circ W_1^k)^T.
\]

With the analytic formula for \( \mathcal{D}[\tilde{u}] \) in hand, it is easy to plug it into Eq. 5 to get the residual loss. The next lemma derives the gradients \( \nabla_{W_2} \phi_r(x; \mathcal{W}) \) of the residual loss w.r.t. the weights of the last layer.

Lemma 2. For \( \tilde{u} \) and \( \mathcal{D}[u] \) given in Lemma 1, gradients of the residual loss w.r.t. the weights of the second layer over the training collocation data \( x \) of \( N \) samples are given by

\[
\nabla_{W_2} \phi_r(x; \mathcal{W}) = W_1^k \circ [l'(\mathcal{D}[\tilde{u}](x) - f(x))^T \times F_1^{(k)}(x)].
\]

Remark 1. Lemmas 1 and 2 generalize the appearance of derivatives in the outputs of the neural network as in Eq. 3, showing that a \( k \)-th order differential term similarly contains the \( k \)-th derivative of the activation function. Thus, activation functions with vanishing high-order derivatives, such as ReLU, significantly reduce the network representation power in approximating the residuals. Note that the gradients w.r.t. \( W_1 \) contain the \((k+1)\)-th derivative of the activation function, further highlighting the importance of well-behaved derivatives in optimization.

In the following section, the gradients given in Lemma 2 are studied at a critical point to find the characteristics of global minima of the residual loss. Note that global minimum in this context refers to the parameters that make the loss zero.

3.3 Critical Points of Wide PINNs

We are eventually interested in finding sufficient conditions for the existence of a global minimum of the residual loss, i.e., \( \phi_r(x; \mathcal{W}) = 0 \). The following theorem takes the first step by providing a necessary condition for globally minimizing the residual loss. We then turn this requirement into a sufficient condition by establishing a set of assumptions. Note that the squared error \( l(r) \) is a non-negative convex function of the residuals \( r \), and \( l'(r) = 0 \) results in \( l(r) = 0 \). Thus, a critical point \( \mathcal{W} \) of \( \phi_r(x; \mathcal{W}) \) in the parameter space globally minimizes the residual loss if \( l'(\mathcal{D}[\tilde{u}](x) - f(x)) = 0 \) for every training sample in \( x \).

Theorem 1. For \( \tilde{u} \) and \( \mathcal{D}[u] \) in Eq. 1, a critical point \( \mathcal{W} \) of the residual loss \( \phi_r(x; \mathcal{W}) \) is a global minimum if the following conditions are satisfied:

1. Weights \( W_{(i)} \) of the first layer are strictly non-zero,
2. \( F_1^{(k)}(x) \) has full row rank, i.e., \( \text{rank}(F_1^{(k)}(x)) = N \).

Theorem 1 distinguishes the global minima from other critical points of the residual loss. However, there is no guarantee that an arbitrary neural network can satisfy the conditions in this theorem. In other words, a critical point that makes \( F_1^{(k)}(x) \) full row rank may not exist in the parameter space of a neural network at all. Still, this theorem does give out a necessary condition for such a neural network. Since \( F_1^{(k)}(x) \) is an \( N \times n_1 \) matrix, the width \( n_1 \) of the first layer must be at
least $N$ for it to be full row rank. In fact, given other assumptions, the next theorem shows that $n_{1} \geq N$ is also a sufficient condition for the existence of a global minimum. Note that the first condition on $W_{1}$ is satisfied with a high probability in a continuous high-dimensional parameter space.

We first define the non-degenerate critical points used in the next theorem and establish a set of assumptions that connect the two theorems together.

Definition 1 (Non-degenerate Critical Point [Nguyen and Hein, 2017]). For a function $f \in C^{2} : U \subset \mathbb{R}^{n} \rightarrow \mathbb{R}$ (i.e., $f$ has continuous second-order derivatives), a critical point $x = (x_{1}, \ldots, x_{n}) \in U$ of $f$ is non-degenerate if its Hessian matrix at $x$ is non-singular. Furthermore, $x$ is non-degenerate on a subset of variables $s \subset \{x_{1}, \ldots, x_{n}\}$ if the Hessian w.r.t. only the variables in $s$ is non-singular at $x$.

Assumptions 1. For the collocation training data $x$ of $N$ points, the activation function $\sigma$ in $\bar{u}$, and the $k$-th order differential operator defined in Lemma 1, we assume that
1. samples in $x$ are distinct,
2. $\sigma^{(k)}$ is a continuous and strictly monotonically increasing function, and
3. $\sigma^{(k)}$ is a bounded function with an infimum of zero.

Theorem 2. With Assumptions 1 holding and for $\mathcal{D}[\bar{u}]$ as in Lemma 1, if $n_{1} \geq N$, then every critical point $\bar{W}$ of $\phi_{r}(x; \bar{W})$ that is non-degenerate on $\{W_{2}, b_{2}\}$ is a global minimum of $\phi_{r}$.

The following remark allows us to make our final conclusion from Theorem 2.

Remark 2. As explained in [Milnor et al., 1965] and [Nguyen and Hein, 2017], for a function $f \in C^{2}$ that maps an open subset $U \subset \mathbb{R}^{n}$ to $\mathbb{R}$, the degenerate critical points in $U$ are rare as the set of all degenerate points has Lebesgue measure zero.

Theorem 2 provides sufficient conditions for a global minimum of the residual loss with a wide network of width $N$ or higher. Based on this theorem, if a PINN with a width of at least $N$ has a non-degenerate critical point, then it also has a global minimum for the residual loss. Since the residual loss $\phi_{r}(x; \cdot)$ is a function from $\mathbb{R}^{|\mathcal{W}|}$ to $\mathbb{R}$ and has continuous second derivatives, the degenerate critical points are rare, and the wide PINN in Theorem 2 has a global minimum.

Note that the residual loss is a strong regularizer that results in a data-efficient training process. Thus, PINNs are often trained with $O(1000)$ collocation points and even fewer boundary data [Raissi et al., 2017; Krishnapriyan et al., 2021]. Therefore, the constraint on the width is well within the practical settings of neural networks. Furthermore, as we observe in the experiments in Section 5, while satisfying the constraint on the width improves the performance, one can expect relatively good results with smaller width as long as the other conditions in Assumptions 1 are almost satisfied.

4 On the Choice of Activation Function

The conditions outlined in Theorem 2 and Assumptions 1 collectively establish an important set of necessities for achieving global minimization of the residual loss. Notably, the requirement of strictly monotonically increasing $\sigma^{(k)}$ implies that it should be a bijection, providing an important guideline in choosing effective activation functions for PINNs. It is noteworthy that bijective activation functions are widely prevalent in deep learning, and extending this characteristic to their derivatives for improved expressiveness in representing differential operators is a plausible goal.

However, the activation functions frequently used in deep learning do not satisfy the bijection property even for the first-order derivatives. As depicted in Figure 1, only Softplus has a bijective first-order derivative. As shown in Figure 2, we observe that the linear outputs of the layers in a neural network with the Sine activation function at initialization are centred at zero with low variance when initialized with normal Xavier initialization. Sitzmann et al. [2020] also proposed a uniform initialization scheme for Sine networks that produces normal linear outputs at all layers with a desired variance. Consequently, in both cases, most of the linear outputs of the layers lie in the $[-\pi/2, \pi/2]$ interval where Sine is bijective. Furthermore, as we train the PINNs with the Sine activation function, we observe that layers still exhibit the same behaviour, i.e., most of the linear outputs of the layers are between $-\pi/2$ and $\pi/2$ after convergence, especially as the width grows larger. Figure 3 illustrates the output distributions for each layer of the trained Wave and Klein-Gordon PINNs (We later define these equations in Section 5).

The observations above suggest that the sinusoidal func-
tions can be utilized to almost satisfy the bijective condition of the activation function. Specifically, we use Cosine and Sine activation functions to train PINNs with first- and second-order terms, respectively. As a result, as long as the width is adequately large to produce low-variance pre-activations within $[-\pi/2, \pi/2]$, the first-order terms in Cosine networks and the second-order terms in Sine networks are determined with the bijective interval of Sine. The same approach can be taken when solving PDEs with higher odd or even terms.

The experiments in the next section show that sinusoidal non-linearity greatly improves the performance of PINNs compared to the common Tanh activation, and the gains are often greater as the width grows. We note that while the Assumption 1.3 facilitates the proof of the Theorem 2, the crucial property is the bijective $\sigma^{(k)}$, and we relax the assumption on the infimum of the derivatives.

5 Experiments

In this section, we provide numerical results for several PDEs, revealing the impact of the activation functions and the width. We first experiment with the first-order Transport equation, comparing the Softplus and Cosine activation functions with Tanh. Next, we study three second-order PDEs using Sine and Tanh activation functions. We empirically show that Sine significantly improves the performance of PDEs with second-order terms with a noticeable decrease in error as the width exceeds the number of training samples.

In all of the experiments, we use a three-layer feed-forward network with a width varying from 64 neurons up to 1024 neurons and initialized with Normal Xavier initialization. All the models are trained with normalized inputs for 80K epochs using the Adam optimizer and an exponential learning rate decay scheme. The only exception is the Wave equation, for which the models are trained for 120K epochs for better convergence. We repeat each experiment three times with a different random initialization and report the average and the best results.

5.1 Transport Equation

The transport equation is a first-order linear PDE that describes a quantity as it moves through time and space. We experiment with the one-dimensional equation with the following formulation:

$$\frac{\partial u}{\partial t} + 30 \frac{\partial u}{\partial x} = 0 \quad x \in [0, 2\pi], t \in [0, 1]$$

We also impose a periodic boundary condition $u(t, 0) = u(t, 2\pi)$ and a Dirichlet initial condition $u(0, x)$ consistent with the solution in [Krishnapriyan et al., 2021]. The PINN is then trained with 256 collocation training points and 200 boundary samples. To verify our results in Section 3.3, we choose Softplus as the activation function as it is a smooth version of ReLU with a bijective first derivative equal to the Sigmoid function. We also use Cosine, as it results in a Sine network for the first-order terms.

Table 1 reports the mean absolute errors for the transport equation. In all cases, Softplus and Cosine perform significantly better than Tanh, as shown in Figure 4. Furthermore, as the width becomes equal to the number of collocation samples (256), we observe a noticeable decrease in the absolute error. The same improvement is also evident in the training curve of the residual loss as shown in Figure 5, where the wide models follow a steep curve. For the PINNs with a width of 256 or wider with both Cosine and Softplus activation functions, the absolute errors are between $10^{-2}$ and $10^{-3}$, outperforming the reported $1.1 \times 10^{-2}$ absolute error achieved with curriculum learning in [Krishnapriyan et al., 2021].
Figure 5: Average residual loss curve for the Transport PINNs with the Softplus activation function and trained with 256 collocation samples.

<table>
<thead>
<tr>
<th>Width</th>
<th>Tanh Avg</th>
<th>Best</th>
<th>Softplus Avg</th>
<th>Best</th>
<th>Cosine Avg</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>0.6314</td>
<td>0.0207</td>
<td>0.6212</td>
<td>0.0111</td>
<td><strong>0.0095</strong></td>
<td>0.0086</td>
</tr>
<tr>
<td>128</td>
<td>0.6118</td>
<td>0.0229</td>
<td>0.6083</td>
<td>0.0156</td>
<td><strong>0.0042</strong></td>
<td>0.0039</td>
</tr>
<tr>
<td>256</td>
<td>0.5673</td>
<td>0.0092</td>
<td>0.5588</td>
<td>0.0040</td>
<td><strong>0.0029</strong></td>
<td>0.0013</td>
</tr>
<tr>
<td>512</td>
<td>0.5168</td>
<td>0.0093</td>
<td>0.4158</td>
<td>0.0062</td>
<td><strong>0.0019</strong></td>
<td>0.0004</td>
</tr>
<tr>
<td>1024</td>
<td>0.3632</td>
<td>0.0103</td>
<td>0.0011</td>
<td>0.0062</td>
<td><strong>0.0014</strong></td>
<td>0.0011</td>
</tr>
</tbody>
</table>

Table 1: Average and best mean absolute error for the Transport equation over three random initializations trained with 256 collocation points.

5.2 Wave Equation

The wave equation describes mechanical and electromagnetic waves and has the following form in 1-D:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

Here $c$ is the velocity of the wave. For $c = 1$ and the solution

$$u(t, x) = \sin(5\pi x) \cos(5\pi t) + 2\sin(7\pi x) \cos(7\pi t),$$

we train the PINN with 512 collocation points for $t, x \in [0, 1]$ and impose the initial and boundary conditions below with 256 boundary data points:

$$u(0, x) = \sin(5\pi x) + 2\sin(7\pi x) \quad x \in [0, 1]$$

$$\frac{\partial u}{\partial t}(0, x) = 0 \quad x \in [0, 1]$$

$$u(t, 1) = u(t, 0) = 0 \quad t \in [0, 1]$$

The residual loss training curves in Figure 5 highlight the impact of the width in training PINNs, with wider models consistently achieving smaller loss values, and the 512- and 1024-neuron models following almost the same path. This behaviour is similar to the curves in Figure 5 for the transport equation, where all the models with a width of at least $N$ achieve very close loss values. As evident in Table 2, PINNs with both Tanh and Sine activation functions perform notably better than narrow PINNs, with a mean absolute error of $3.11 \times 10^{-2}$ and $5.62 \times 10^{-2}$ for Tanh and Sine respectively. Figure 6 illustrates the inability of the narrow Tanh network compared to Sine and wide Tanh models in representing the solution. Also, while narrow Sine PINNs are able to find good solutions, the training is more unstable and the performance is worse on average compared to wider models.

5.3 Helmholtz Equation

We consider a 2D Helmholtz equation of the following form with $t, x \in [-1, 1]$ as in [Wong et al., 2022]:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + u = (1 - \pi^2 - (6\pi)^2) \sin(\pi x) \cos(6\pi y)$$

With zero boundary conditions, the solution is given by

$$u(x, y) = \sin(\pi x) \sin(6\pi y).$$

The PINN is then trained with 512 collocation points and 256 boundary data, using Sine and Tanh as the activation function. As reported in Table 2, Sine performs remarkably better than Tanh across all widths. Similar to the Wave and Transport equations, there is also a slight decrease in the errors as the width exceeds the number of collocation points. As in the Wave equation, PINNs with the Tanh activation function start to perform better as the width grows larger. However, as illustrated in Figure 8, even the 1024 neurons-wide Tanh network is still unable to capture the solution, while the Sine...
Table 2: Average and best mean absolute errors for second-order PDEs over three random initializations. Underlined values show where the width is equal to $N$.

<table>
<thead>
<tr>
<th>Width</th>
<th>Tanh Avg</th>
<th>Sine Avg</th>
<th>Tanh Best</th>
<th>Sine Best</th>
<th>Tanh Avg</th>
<th>Sine Avg</th>
<th>Tanh Best</th>
<th>Sine Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>4.7235</td>
<td>12.3456</td>
<td>0.0125</td>
<td>0.0087</td>
<td>0.0275</td>
<td>0.0116</td>
<td>0.0097</td>
<td>0.0059</td>
</tr>
<tr>
<td>128</td>
<td>2.8161</td>
<td>2.2007</td>
<td>0.0105</td>
<td>0.0064</td>
<td>0.0134</td>
<td>0.0096</td>
<td>0.0056</td>
<td>0.0035</td>
</tr>
<tr>
<td>256</td>
<td>1.8516</td>
<td>1.4016</td>
<td>0.0273</td>
<td>0.0212</td>
<td>0.0606</td>
<td>0.0191</td>
<td>0.0052</td>
<td>0.0031</td>
</tr>
<tr>
<td>512</td>
<td>0.7854</td>
<td>0.4909</td>
<td>0.0044</td>
<td>0.0028</td>
<td>0.0875</td>
<td>0.0289</td>
<td>0.0096</td>
<td>0.0049</td>
</tr>
<tr>
<td>1024</td>
<td>0.5946</td>
<td>0.2221</td>
<td>0.0067</td>
<td>0.0056</td>
<td>0.1939</td>
<td>0.0928</td>
<td>0.0090</td>
<td>0.0046</td>
</tr>
</tbody>
</table>

Figure 8: Helmholtz Equation. Top panels: Exact solution, Middle panels: Predicted solution, Bottom panels: Absolute Error.

Figure 9: Absolute error of the Klein-Gordon PINNs.

Tanh also performs reasonably well with the 128 neurons-wide model, although it still performs slightly worse than the Sine models, as shown in Figure 9.

6 Conclusion

The differentiation process in the residual loss of PINNs transforms the structure of the neural networks and their outputs, rendering the existing theory around loss functions and common supervised tasks ineffective in analyzing PINNs. In this work, we aim to fill the gap in our understanding of the residual loss and derive the requirements in network design that lead to achieving global minimization of this loss function. To this end, we study the residual loss at a critical point in the parameter space of the neural network and look for distinct characteristics of a global minimum that sets it apart from other critical points. We then use those characteristics to derive the requirements in the neural network design that ensure the existence of a global minimum. Additionally, we reveal that activation functions with well-behaved high-order derivatives are crucial in the optimal minimization of the residual loss. We then use the established theory and empirical observations to choose activation functions and verify their effectiveness by conducting a set of experiments. The theory developed in this work paves the way for further development of better activation functions and provides a guideline for designing effective PINNs.

Acknowledgements

This research is supported by Natural Sciences and Engineering Research Council of Canada (NSERC), Discovery Grants program, and the Vector Scholarship in Artificial Intelligence, provided through the Vector Institute.
References


[Sirignano and Spiliopoulos, 2018] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algo-


