Numerical Heat Transfer, Part B: Fundamentals
An International Journal of Computation and Methodology

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/unhb20

Hyperparameter selection for physics-informed neural networks (PINNs) – Application to discontinuous heat conduction problems

Prakhar Sharma, Llion Evans, Michelle Tindall & Perumal Nithiarasu

To cite this article: Prakhar Sharma, Llion Evans, Michelle Tindall & Perumal Nithiarasu (09 Oct 2023): Hyperparameter selection for physics-informed neural networks (PINNs) – Application to discontinuous heat conduction problems, Numerical Heat Transfer, Part B: Fundamentals, DOI: 10.1080/10407790.2023.2264489

To link to this article: https://doi.org/10.1080/10407790.2023.2264489

NUMERICAL HEAT TRANSFER, PART B: FUNDAMENTALS
https://doi.org/10.1080/10407790.2023.2264489

Hyperparameter selection for physics-informed neural networks (PINNs) – Application to discontinuous heat conduction problems

Prakhar Sharma[a], Llion Evans[a,b], Michelle Tindall[b], and Perumal Nithiarasu[a]

[a]Zienkiewicz Institute for Modelling, Data and AI, Swansea University, UK; [b]Culham Science Center, United Kingdom Atomic Energy Authority, Abingdon, UK

ABSTRACT

In recent years, physics-informed neural networks (PINNs) have emerged as an alternative to conventional numerical techniques to solve forward and inverse problems involving partial differential equations (PDEs). Despite its success in problems with smooth solutions, implementing PINNs for problems with discontinuous boundary conditions (BCs) or discontinuous PDE coefficients is a challenge. The accuracy of the predicted solution is contingent upon the selection of appropriate hyperparameters. In this work, we performed hyperparameter optimization of PINNs to find the optimal neural network architecture, number of hidden layers, learning rate, and activation function for heat conduction problems with a discontinuous solution. Our aim was to obtain all the settings that achieve a relative L2 error of 10% or less across all the test cases. Results from five different heat conduction problems show that the optimized hyperparameters produce a mean relative L2 error of 5.60%.

ARTICLE HISTORY
Received 5 May 2023
Revised 13 September 2023
Accepted 18 September 2023

KEYWORDS
Discontinuous boundaries conditions; heat conduction; hyperparameter tuning; physics-informed neural networks; stiff partial differential equation

1. Introduction

Conventional numerical techniques have their limitations when it comes to solving problems involving partial differential equations (PDEs). They include incorporating noisy experimental data into existing algorithms, solving high-dimensional parametric PDEs in a reasonable time frame, solution dependency on mesh quality, and the requirement of complicated algorithms for inverse problems.

Recently, physics-informed neural networks (PINNs) [1] have gained popularity as an approach to solve forward [2–6] and inverse problems [7–10] involving PDEs. Unlike conventional machine learning techniques, PINNs are physics-based supervised machine learning techniques and do not require true solutions or pre-trained models to solve PDEs [11]. Instead, PINNs satisfy a well-posed PDE to obtain a unique solution in forward problems.

Despite its success in problems with smooth solutions [8,12,13], PINN and their variations [14–17] still encounter fundamental issues. For instance, PINN is still lacking guaranteed convergence criteria [18], and the correlation between the network hyperparameters and performance is elusive [19]. Moreover, PINNs encounter significant issues solving stiff-PDEs, especially, problems with discontinuous solutions [12,20]. We refer the reader to our survey on solving problems with

CONTACT Perumal Nithiarasu p.nithiarasu@swansea.ac.uk Zienkiewicz Institute for Modelling, Data and AI, Swansea University, Bay Campus, Swansea SA1 8EN, UK.

discontinuous BCs using PINNs [20]. Hence, there exists a necessity for studies that specifically address convergence of PINNs for problems with discontinuous solutions.

Generally, PINNs will not converge unless we mitigate the effect of the discontinuities. A technique to accomplish this involves the use of a signed distance function (SDF) and requires a prior understanding of the region of the discontinuity [21]. The SDF ensures that the points sampled close to the region of discontinuity are assigned with minuscule weights, effectively reducing the contribution of these regions toward the training loss. As a result, the use of SDF results in a high relative L2 error near the discontinuity.

Similar to any neural network (NN), PINNs have a number of hyperparameters such as, the learning rate, the activation function, the optimizer, the weights in the loss function, the width and depth of the NN. The high-dimensional hyperparameter search space makes it difficult to obtain optimized hyperparameters that are suitable for a broad range of problems. Therefore, it is necessary to select fewer hyperparameters and focus on a specific set of problems.

In some early attempts, researchers treated the coefficients of loss terms as hyperparameters to balance the contribution of each loss term [12,13,22–24]. The approach was computationally efficient as the hyperparameter optimization did not involve a complicated algorithm instead it was updated through the same optimizer used to train the PINN. However, this approach had limitations regarding the range of hyperparameters that could be chosen. It did not allow choosing hyperparameters such as the number of hidden layers or the NN architecture.

Some researchers used grid search [25,26] and Bayesian optimization [27–29] to gain greater flexibility in selecting hyperparameters. However, with the increase of number of hyperparameters, the approach became computationally expensive due to the high-dimensional search space.

In this study, we have chosen to employ manual tuning. It offers several advantages, including, full control over the hyperparameter values, better interpretability and computational efficiency in comparison to grid search or Bayesian optimization methods, when only a few hyperparameters are studied.

Several studies conducted manual tuning with a wide range of hyperparameters [30,31]. The outcomes were transferable to similar problems with minor variations, making them ideal starting points for problems in the same domain of physics.

The present work aims to identify optimal hyperparameters for PINNs in order to improve their accuracy and convergence when solving heat transfer problems with discontinuous BCs. We performed hyperparameter optimization to identify all possible settings of NN architectures, number of hidden layers, activation functions, and learning rates that could achieve a relative L2 error of 10% or less across all test cases. These optimal hyperparameters enable more accurate and efficient solutions to similar problems.

We investigated 2-D and 3-D steady-state heat conduction problems with a discontinuous solution. Additionally, we investigated the 2-D steady-state heat conduction problem with parametric conductivity and parametric geometry as separate test cases. The remainder of this paper is organized as follows. In Section 2, we provide a brief overview of PINNs. In Section 3, we discuss additional tools that we used to enhance the accuracy of the predicted solution. In Section 4, we discuss the network parameters and the hyperparameters used in this study. Sections 5 and 6 present the test cases and results of the hyperparameter optimization. In Section 7, we discuss the outcome of our investigation.

2. Physics-informed neural networks

PINNs are deep neural networks (DNNs) that leverage the known physics, that is, PDEs or constitutive equations to solve forward and inverse problems. In the case of forward problems, it enforces the well-posedness of the problem to obtain a unique solution. This section provides a brief overview of PINNs.

In a PINN, the independent variables serve as input, and the dependent variables or the solution to the governing PDE are the outputs. A well-posed PDE can be defined as follows:

$$\begin{aligned} \boldsymbol{u}_t + \mathcal{N}_x[\boldsymbol{u}] &= 0, \quad \boldsymbol{x} \in \boldsymbol{\Omega}, \ t \in [0,T] \\ \boldsymbol{u}(\boldsymbol{x},0) &= h(\boldsymbol{x}), \quad \boldsymbol{x} \in \boldsymbol{\Omega} \\ \boldsymbol{u}(\boldsymbol{x},t) &= g(\boldsymbol{x},t), \quad \boldsymbol{x} \in \partial\boldsymbol{\Omega}, \ t \in [0,T] \end{aligned} \tag{1}$$

where $\mathcal{N}_x[\boldsymbol{u}]$ is a differential operator, $\boldsymbol{x}$ and t are the independent variables of the PDE, $\boldsymbol{\Omega}$ and $\partial\boldsymbol{\Omega}$ denote the spatial domain and the boundary of the problem, $h(\boldsymbol{x})$ denotes the prescribed BC which is the solution to the PDE at all spatial points ($\boldsymbol{\Omega}$) at the initial time ($t = 0$), $g(\boldsymbol{x},t)$ is the prescribed BC at the boundary of the domain ($\partial\boldsymbol{\Omega}$).

2.1. Discrete loss terms

A well-posed PDE problem must have the PDE, BCs, and IC (if transient). Thus, a PINN must satisfy these, resulting in a multitask loss function. The total loss ($\mathcal{L}$) and the individual loss terms ($\mathcal{L}_{PDE}$, $\mathcal{L}_{BC}$, $\mathcal{L}_{IC}$) are defined as follows:

$$\mathcal{L} = \lambda_{PDE}\mathcal{L}_{PDE} + \lambda_{BC}\mathcal{L}_{BC} + \lambda_{IC}\mathcal{L}_{IC} \tag{2}$$

$$\begin{aligned} \mathcal{L}_{PDE} &= \frac{1}{N_r}\sum_{i=1}^{N_r} |\hat{u}_t(\boldsymbol{x}_i,t_i) + \mathcal{N}_x[\hat{u}(\boldsymbol{x}_i,t_i)]|^2 \\ \mathcal{L}_{BC} &= \frac{1}{N_b}\sum_{i=1}^{N_b} |\hat{u}(\boldsymbol{x}_i,t_i) - g(\boldsymbol{x}_i,t_i)|^2 \\ \mathcal{L}_{IC} &= \frac{1}{N_0}\sum_{i=1}^{N_0} |\hat{u}(\boldsymbol{x}_i,0) - h(\boldsymbol{x}_i,0)|^2 \end{aligned} \tag{3}$$

where $\hat{u}$ is the predicted solution, N_r, N_b, and N_0 are the number of data points that are sampled to satisfy the PDE, the BCs, and the IC in Eq. (2). The coefficients λ_{PDE}, λ_{BC}, and λ_{IC} in Eq. (2), help in achieving convergence and better accuracy, and are an active field of research.

2.2. Integral formulation of loss

In recent work by Nabian et al. [32] and Hennigh et al. [33], an alternative approach to calculate the loss was proposed. In this approach, a specific loss term is scaled proportional to its length, area, or volume. This prevents the large number of interior points from dominating the training loss, and ensures that both the $\mathcal{L}_{PDE}$ and $\mathcal{L}_{BC}$ contribute proportionally. This particular formulation employs an integral form of the loss for each term as follows:

$$\begin{aligned} \mathcal{L}_{PDE} &= \frac{1}{N_r}\sum_{i=1}^{N_r} |\hat{u}_t(\boldsymbol{x}_i,t_i) + \mathcal{N}_x[\hat{u}(\boldsymbol{x}_i,t_i)]|^2 \int_{\boldsymbol{\Omega}} d\boldsymbol{\Omega} \\ \mathcal{L}_{BC} &= \frac{1}{N_b}\sum_{i=1}^{N_b} \left|\hat{u}(\boldsymbol{x}_i,t_i) - g(\boldsymbol{x}_i,t_i)\right|^2 \int_{\partial\boldsymbol{\Omega}} d(\partial\boldsymbol{\Omega}) \end{aligned} \tag{4}$$

3. Additional tools

In this section, we provide a brief description of the methods that enhance the accuracy of the predicted solution. In addition to the feedforward fully-connected neural network (FCNN), we also included the modified Fourier neural network (MFNN) [34] and the deep Galerkin method (DGM) architecture [35] to address the spectral bias in FCNN [13,36,37]. These architectures

project the low-dimensional training data to higher dimensions which enables learning the high-frequency functions.

We also employed adaptive activation [38], SDF [21], importance sampling [39], and quasi-random sampling [40]. A brief description of these tools has been given in this section and a detailed survey of these tools is presented in [20].

The SDF plays an important role in solving PDEs with discontinuous BCs. It requires the prior knowledge of the region of discontinuity. These functions are designed such that the points sampled in these regions are given minuscule weights [21,41,42]. While the SDF can help converging the training loss for problems with discontinuity, it is important to note that they may not be able to capture the discontinuous BC, which can lead to a high relative L2 error, this error is usually much lower than without the use of SDFs, providing a significant improvement in the predicted solution.

Activation functions enable NNs to predict complex outputs by performing a nonlinear transformation of the feature space. In adaptive activation, a trainable coefficient is multiplied with each neuron in the NN. These trainable coefficients are updated iteratively, based on the total loss $\mathcal{L}$. The introduction of additional parameters facilitates faster convergence [38].

Importance sampling is used to re-sample the interior points iteratively. The sample is drawn from an alternative sampling distribution proportional to the pointwise loss. This approach is similar to mesh refinement and helps to capture the sharp changes in the gradient [32].

A common practice in PINNs is to use a uniform random sampling strategy to sample the boundary and interior points. Low-discrepancy quasi-random sequences are a suitable replacement for uniformly distributed random points as they require less points [43]. This results in faster convergence of the training loss. In this work, we used the Halton sequence to sample the boundary and interior points in the domain [44].

4. Summary of methodology

In machine learning, hyperparameters are parameters that influence the overall performance of the model. Hyperparameter optimization is a process of selecting the optimal combination of hyperparameters that minimize or maximize an objective function such as the training loss or the model accuracy.

In this work, we investigated the NN architecture, the number of hidden layers, the learning rate, and the activation function over a wide range of settings, as detailed in Table 1.

Typically, hyperparameter optimization is implemented using automated optimization techniques to obtain the optimal settings. However, in this work, we employed a manual tuning approach [45] to obtain multiple settings that results in a relative L2 error (Eq. (5)) of 10% or less across all five test cases.

$$\text{Relative L2 Error} = \frac{\|\hat{u} - u\|_2}{\|u\|_2} \tag{5}$$

The weights were initialized with the Xavier initialization [46] and the losses were calculated using the integral formulation (Eq. (4)). Initially, we experimented with activation functions such as rectified linear unit (RELU), scaled exponential linear unit (SELU), Sigmoid linear unit (SiLU),

Table 1. Range of hyperparameters explored.

Hyperparameters	Range
Architecture	FCNN, MFNN, DGM
No. of hidden layers	1, 5, 10, 15, 20
Activations	ReLU, SELU, SiLU, GELU, Tanh, Sin
Learning rates	1e − 2, 7.5e − 3, 5e − 3, 2.5e − 3, 1e − 3, 7.5e − 4, 5e − 4, 2.5e − 4, 1e − 4, 7.5e − 5, 5e − 5, 2.5e − 5, 1e − 5

Sine function (Sin), Gaussian error linear units (GELU), and hyperbolic tangent (Tanh) over a range of learning rates starting from $1e-5$ to $1e-2$. The mathematical form of the activation functions is listed below.

$$\begin{aligned}
\mathrm{RELU}(x) &= \max(0,x)\\
\mathrm{SELU}(x) &= 1.0507\begin{cases} x & \text{if } x>0\\ 1.6733(e^x-1) & \text{if } x\le 0\end{cases}\\
\mathrm{SiLU}(x) &= x\cdot\frac{1}{1+e^{-x}}\\
\mathrm{GELU}(x) &= 0.5x\left(1+\mathrm{Tanh}\left(\sqrt{\frac{2}{\pi}}(x+0.044715x^3)\right)\right)\\
\mathrm{Tanh}(x) &= \frac{e^x-e^{-x}}{e^x+e^{-x}}
\end{aligned} \tag{6}$$

where x is the input value. It is important to note that the RELU and SELU are non-differentiable. Most machine learning libraries handle the non-differentiable activation functions using sub-gradient optimization. Bertoin et al. [47] emphasized that the derivative of non-differentiable activation functions can impact the backpropagation and the Adam optimizer can help minimize this impact. Therefore, we used the Adam optimizer in all test cases.

The batch training was performed on a high-performance computing (HPC) cluster, Sunbird at the Swansea University. The test cases were trained on 2 x AMD EPYC Rome 7452 processor with 32-cores and a NVIDIA A100 tensor core with 40 GB of available GPU memory.

5. Test cases

In this section, we present the details of the five different test cases. The first two cases are 2-D and 3-D steady-state heat conduction problems with discontinuous BCs. The next two cases are simple extension of the 2-D problem, incorporating parametric conductivity and parametric geometry. The final test case is a steady-state heat conduction problem with discontinuous conductivity.

5.1. Problem 1: 2-D steady-state heat conduction

In Problem 1, we chose a 2-D steady-state heat conduction problem with discontinuous BCs at the corners. The problem can be stated as follows:

$$\begin{aligned}
&u_{xx}+u_{yy}=0, \qquad x\in[-0.5,0.5], \quad y\in[0,1]\\
&u(x,0)=u(x,1)=0 \quad u(-0.5,y)=u(0.5,y)=1
\end{aligned} \tag{7}$$

where u is the temperature, x and y are the independent variables of the PDE.

5.2. Problem 2: 3-D steady-state heat conduction

Our next focus is on extending Problem 1 into 3-D space. Problem 2 is constrained by discontinuous BCs at the edges and the corners. The problem is outlined as follows:

$$\begin{aligned}
&u_{xx}+u_{yy}+u_{zz}=0, \qquad x\in[-0.5,0.5], \qquad y\in[-0.5,0.5], \quad z\in[0,1]\\
&u(x,y,0)=u(x,y,1)=u(0.5,y,z)=0\\
&u(-0.5,y,z)=u(x,-0.5,z) \quad =u(x,0.5,z)=1
\end{aligned} \tag{8}$$

The reference solution for Problems 1 and 2 was generated through FEM implemented in MATLAB Partial Differential Equation Toolbox, and is shown in Figure 1. Details of the mesh statistics of the FEM solutions are shown in Table 2.

Raissi et al. [1] proposed the first PINN framework which is often referred as the baseline PINN. It comprises a FCNN architecture with Tanh activation, trained using a learning rate of $1e-3$. It was known to have convergence issues with discontinuous BCs, as demonstrated in the loss curve (Figure 2) of Problem 1, trained with a FCNN of 8 hidden layers and 20 units in each layer. Since, the baseline PINNs are continuous functions they cannot represent discontinuities. This is evident in Problem 1, where the pointwise BC loss ($\mathcal{L}_{BC}$) around the corners is relatively high after training for 14k iterations (Figure 3).

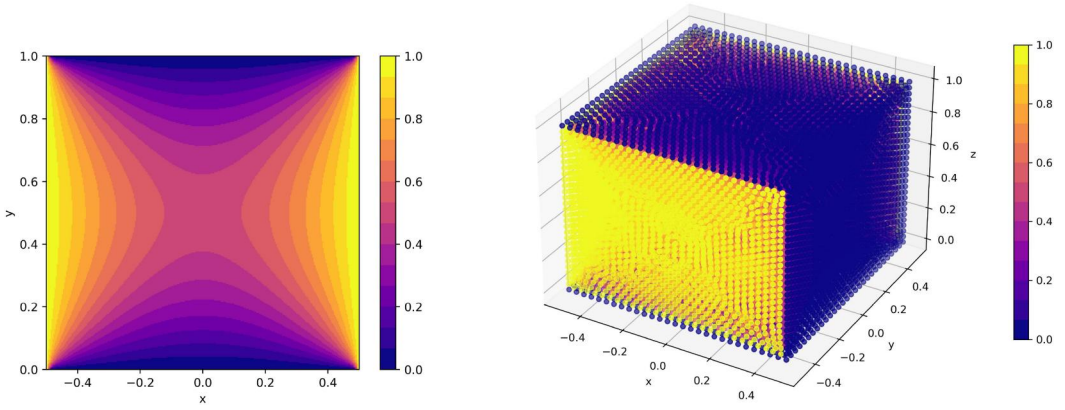

Figure 1. FEM solution of Problem 1 (left) and Problem 2 (right).

Table 2. Mesh statistics of the FEM solution.

	Problem 1	Problem 2
Element type	Triangle	Tetrahedral
Element order	Quadratic	Quadratic
No. of elements	2841	23424
Max. element size	0.0566	0.0693
Min. element size	0.0283	0.0346
Mesh growth rate	1.5	1.5

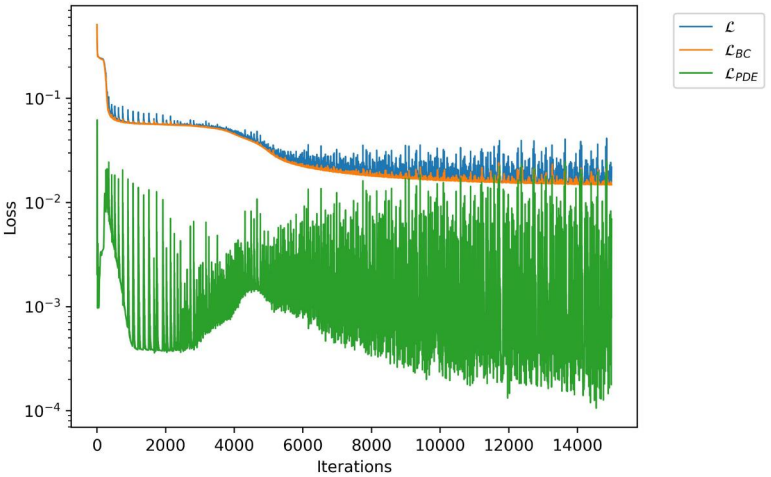

Figure 2. The loss curve (training loss) of Problem 1 trained with Tanh activation and baseline PINN for 14k iterations.

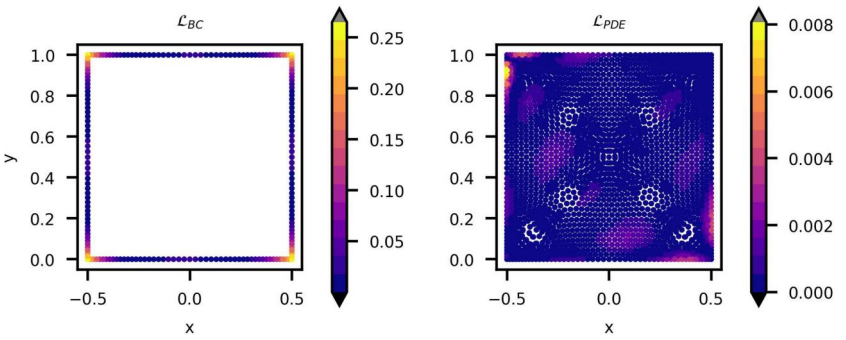

Figure 3. The pointwise BC loss (left) and pointwise PDE loss (right) of Problem 1 trained with Tanh activation and baseline PINN for 14k iterations.

5.3. Problem 3: Problem 1 with parametric geometry

In Problem 3, the upper boundary of the domain is varying from 1 to 1.05.

$$\begin{aligned} &u_{xx} + u_{yy} = 0, \qquad x \in [-0.5, 0.5], \quad y \in [0, L], \quad L \in [1, 1.05] \\ &u(x, 0) = u(x, L) = 0 \quad u(-0.5, y) = u(0.5, y) = 1 \end{aligned} \tag{9}$$

Parametric PDEs can be solved with a simple extension of the PINN by adding the parametric terms as additional features in the training dataset. Since the upper boundary is not fixed, care should be taken to ensure the y-coordinate for each sample in the training dataset resides inside the domain.

5.4. Problem 4: Problem 1 with parametric conductivity

Similar to Problem 3, we can parameterize the conductivity. We formulate Problem 1 with parametric conductivity κ as follows:

$$\begin{aligned} &u_{xx} + \kappa\ u_{yy} = 0, \qquad x \in [-0.5, 0.5], \quad y \in [0, 1], \quad \kappa \in [0, 1] \\ &u(x, 0) = u(x, 1) = 0 \quad u(-0.5, y) = u(0.5, y) = 1 \end{aligned} \tag{10}$$

5.5. Problem 5: 2-D steady-state heat conduction with discontinuous conductivity

In Problem 5, we have a 2-D steady-state heat conduction problem with discontinuous conductivity κ and Robin BCs. The problem can be stated as follows:

$$\begin{aligned} &\frac{\partial}{\partial x}(\kappa(x) \cdot u_x) + \frac{\partial}{\partial y}(\kappa(x) \cdot u_y) = 0, \quad x \in [0, 1], \quad y \in [0, 1] \\ &u_x = \frac{u}{2\kappa(x)} \quad \text{when} \quad x = 0, \qquad u_x = \frac{1 - u}{2\kappa(x)} \quad \text{when} \quad x = 1 \\ &\kappa(x) = \begin{cases} \kappa_1 & \text{if } x \leq 0.5 \\ \kappa_2 & \text{if } x > 0.5 \end{cases} \end{aligned} \tag{11}$$

where $\kappa_1 = 1/30$ and $\kappa_2 = 1/15$. Here, the pointwise conductivity varies as a piecewise function of the x-coordinates. The problem has an analytical solution given as a piecewise function of κ_1 and κ_2 (Eq. (12)) [48].

$$u = \begin{cases} \dfrac{\kappa_2 x + 2\kappa_1\kappa_2}{0.5(\kappa_1 + \kappa_2) + 4\kappa_1\kappa_2} & \text{if } x \leq 0.5 \\ \dfrac{\kappa_1 x + 2\kappa_1\kappa_2 + 0.5(\kappa_2 - \kappa_1)}{0.5(\kappa_1 + \kappa_2) + 4\kappa_1\kappa_2} & \text{if } x > 0.5 \end{cases} \tag{12}$$

6. Results and discussion

In this section, we present the results of hyperparameter optimization across all the test cases. We leverage bar charts to discuss the relationship between the hyperparameters and the mean relative L2 error for all possible settings. These bar charts illustrate the mean relative L2 error for each hyperparameter to identify the problematic hyperparameter values.

SELU is known for fast and stable training, however, based on the initial investigation, we observed that the SELU activation function is causing convergence issues in all the test cases. This is due to the fact that SELU is prone to exploding gradients in deep networks [49].

Unlike SELU, RELU suffers from vanishing gradients. So even though the training loss is converged, it exhibits a relative L2 error that is several magnitudes higher compared to other activation functions across most test cases. This is due to the fact that, for negative inputs, the gradient of RELU is zero. During backpropagation, a large negative gradient multiplied with zero effectively vanishes the gradient. So, the corresponding neurons will not be updated properly, resulting in a high relative L2 error. Thus, we decided to exclude SELU and RELU activation from further studies and continued further investigation with GELU, SiLU, Sin, and Tanh.

Figure 4 shows the mean relative L2 error for each activation across all the test cases. We observed some anomalies, for example, in Problem 3 with Tanh activation, the mean relative L2 error reaches to the order of 1e5. Again, in Problem 3, with Sin activation the mean relative L2

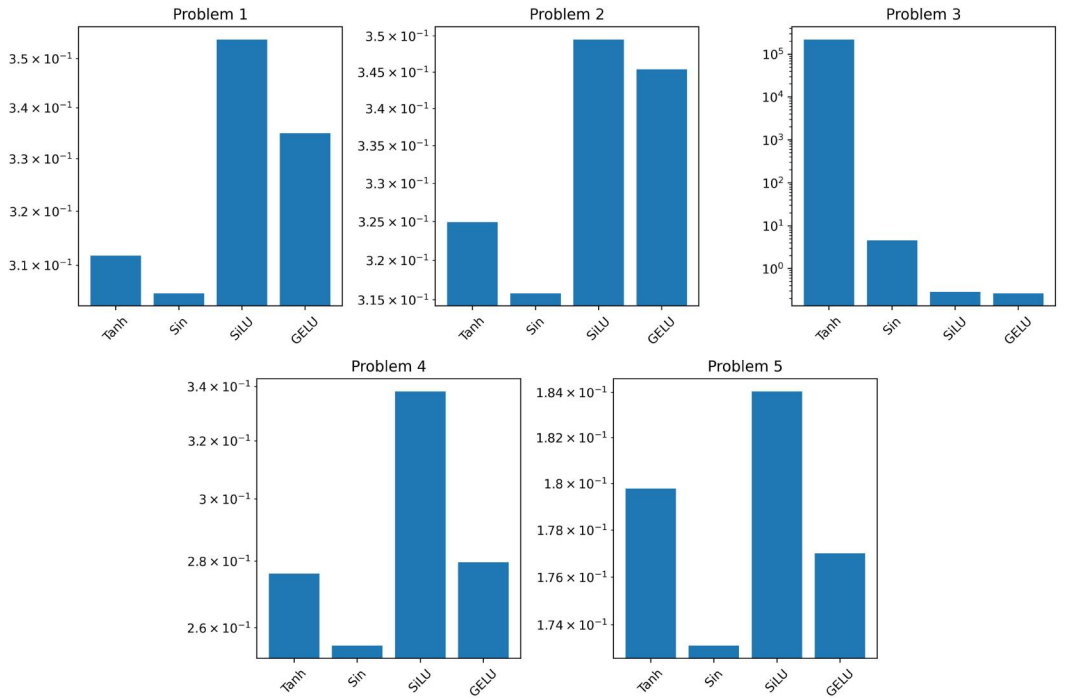

Figure 4. Comparison of activation function for five different test cases. The horizontal axis shows the activation function while the vertical axis shows the mean relative L2 error.

error is more than 10^0 or 100%. These anomalies are attributed to all the settings with a large learning rate (1e − 2).

Figure 5 illustrates the relationship between the mean relative L2 error with number of hidden layers. As anticipated, the mean relative L2 error decreases with the increase in the number of hidden layers. With the addition of layers, the network gains the ability to learn more complex patterns. Similar to activations, in Problem 3, with five hidden layers, the mean relative L2 error reaches to the order of 1e5. This anomaly is attributed to all the settings with large learning rate (1e − 2).

In Figure 6, we present the variation of mean relative L2 error with NN architectures. Similar to the activations and the number of hidden layers, in Problem 3, the mean relative L2 error has been skewed by an anomaly attributed to the DGM architecture and the use of a large learning rate. It becomes evident that large learning rates may lead to anomalies or unexpected behavior and must be avoided while solving stiff-PDEs.

Although, we used SDF in all the settings, it is worth noting that FCNN has a lower mean relative L2 error than MFNN in all the test cases. The SDF excludes the regions with high-frequency function in the domain, thus FCNN is less susceptible to spectral bias. On the other hand, MFNN projects the low-dimensional input dataset to higher dimensions thus increasing the complexity of the model. Thus, MFNN may need more iterations to converge, potentially affecting the performance compared to FCNN.

Figure 7 shows the variation of mean relative L2 error with learning rate. In Problem 3, all the anomalies discussed earlier were caused by the learning rate of 1e − 2. This is due to the parametric nature of Problem 3, where the moving discontinuity around the upper boundary results in a highly complex loss landscape. Consequently, the optimizer fails to find the global minima, resulting in a mean relative L2 error of the order of 1e6.

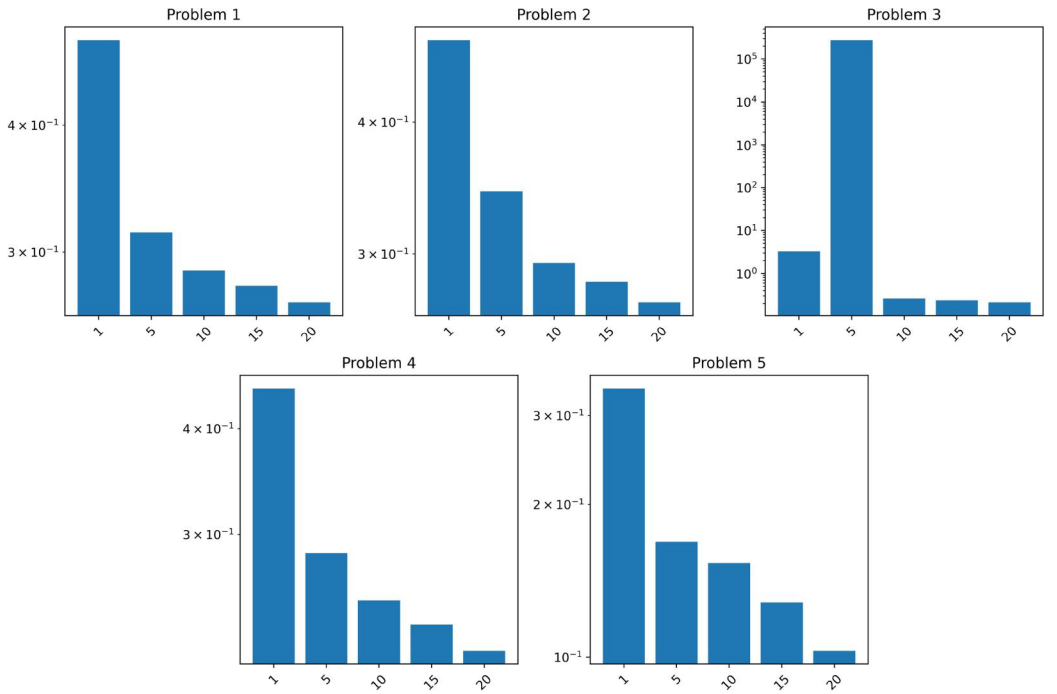

Figure 5. Comparison of number of hidden layers for five different test cases. The horizontal axis shows the number of hidden layers while the vertical axis shows the mean relative L2 error. For each specific number of hidden layers, the mean relative L2 error includes data from all architectures with that specific number of hidden layers.

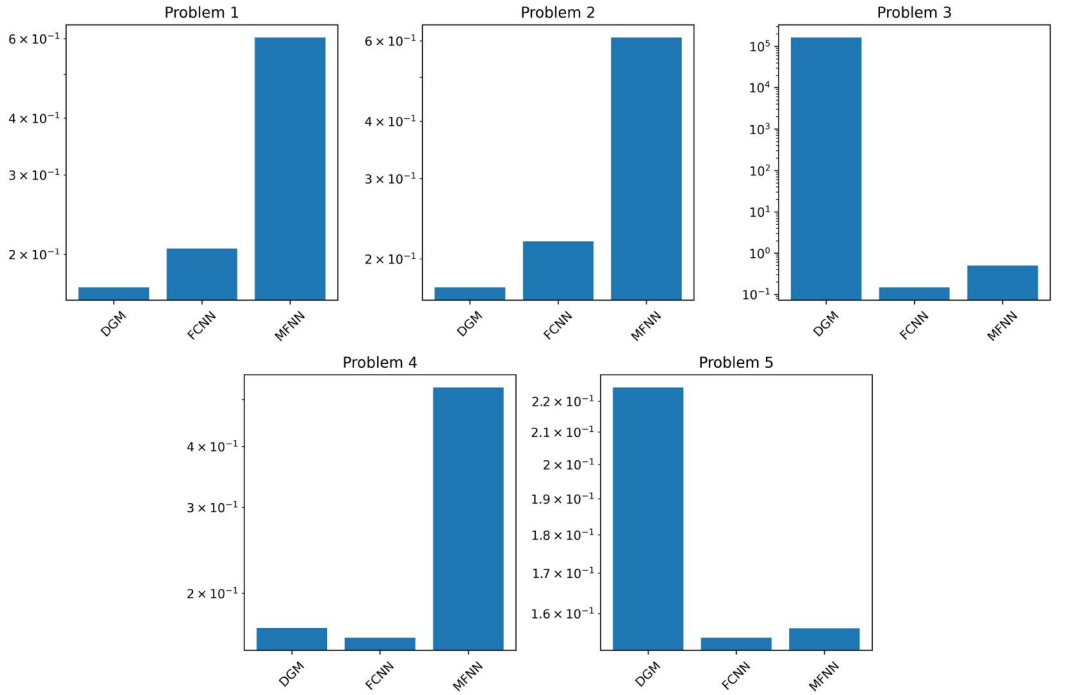

Figure 6. Comparison of neural network architectures for five different test cases. The horizontal axis shows the individual neural network architectures while the vertical axis shows the mean relative L2 error.

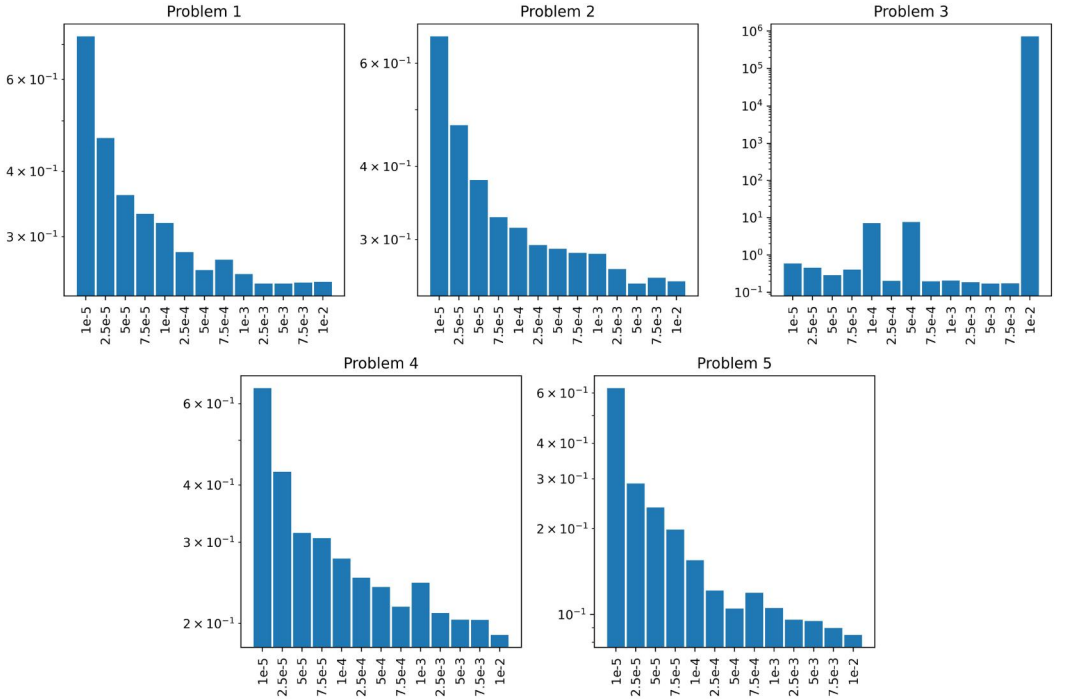

Figure 7. Comparison of learning rates for five different test cases. The horizontal axis shows the learning rates while the vertical axis shows the mean relative L2 error.

Table 3. Optimal hyperparameter settings.

Hyperparameters	Range
Architecture	FCNN, DGM
No. of hidden layers	20
Activations	SiLU, GELU, Tanh, Sin
Learning rates	7.5e − 3, 5e − 3, 2.5e − 3, 1e − 3, 7.5e − 4, 5e − 4

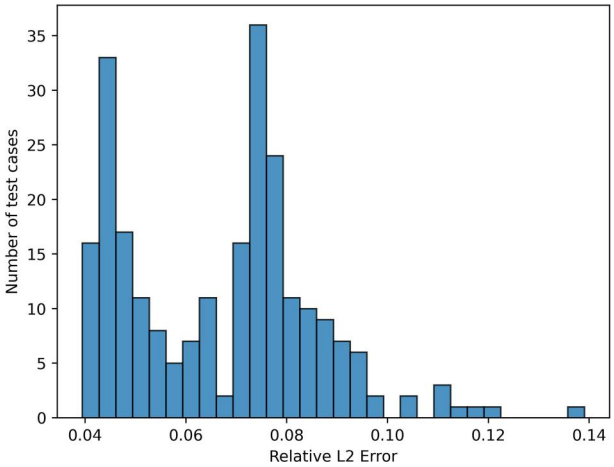

Figure 8. The distribution of relative L2 error with optimized settings across all the test cases.

In contrast, for all other test cases, we observed an improvement in the mean relative L2 error with an increase in learning rate. This is a well-known trend with the Adam optimizer where the learning rate determines the step size taken in each iteration. However, due to inherent highly non-convex nature of PINNs, the optimizer may get trapped in a local minima when using small learning rates.

After analyzing the data, we have identified optimal hyperparameter settings, these are shown in Table 3. With these settings, we obtained a mean relative L2 error of 5.60%, a maximum relative L2 error of 13.9% with a standard deviation of 1.08% across all the test cases. While our optimal number of hidden layers was 20, it should be noted that more hidden layers can be used. However, in such cases, the number of iterations should be enough to allow for convergence. Furthermore, Figure 8 shows the distribution of relative L2 error across these test cases. It demonstrates that in majority of the test cases the relative L2 error remains below 10%.

Alternatively, if we restrict the list of optimal activations to SiLU, we achieved a mean relative L2 error of 5.32%, a maximum relative L2 error of 9.55% with a standard deviation of 0.7048% across all the test cases. This is in agreement with the objective of the work, that is, the optimal settings should achieve a relative L2 error of 10% or less across all the test cases. However, we observed that, on case-by-case basis, other activations performed better than SiLU. Therefore, we include other activations in the list of optimal activations, allowing the reader to choose the most suitable activation function for specific scenarios.

7. Conclusions

In this study, we conducted a comprehensive hyperparameter optimization of PINNs to determine the optimal NN architectures, number of hidden layers, activation functions, and learning rates for heat conduction problems with discontinuous solutions. We aimed to identify all

possible settings of these hyperparameters that could achieve a relative L2 error of 10% or less across all test cases.

These test cases include a 2-D and a 3-D steady-state heat conduction problem with discontinuous BCs. Additionally, we extended the 2-D steady-state heat conduction problem with parametric conductivity and parametric geometry. Finally, the last test case involved a 2-D steady-state heat conduction problem with discontinuous conductivity and Robin BCs.

Throughout the investigation, we employed a range of tools, such as, adaptive activation, SDF, importance sampling, and quasi-random sampling. These methods accelerate the convergence of the training loss, resulting in much lower relative L2 error compared to not using them. Specifically, the SDFs have limited ability to accurately capture discontinuities in the solution. As a result, models with a relative L2 error within 10% are generally considered to be good models.

The range of hyperparameters investigated has been summarized in Table 1. In Section 6, we discussed the relationship between each hyperparameter and the mean relative L2 error. Finally, we presented a list of optimal hyperparameter settings in Table 3.

In general, hyperparameters that are found to be effective for a specific class of problems can often be used for similar problems. Therefore, these optimal hyperparameters can be used directly to obtain a satisfactory solution, that is, a relative L2 error less than 10% and as a starting point for obtaining optimal hyperparameters for other heat conduction problems that exhibit similar discontinuities. Examples of variations may include changes in the geometry, different boundary temperatures, and the introduction of source terms in the PDE. However, the effectiveness of these optimal hyperparameters must be validated on a case-by-case basis.

The hyperparameter optimization of PINNs is an effective approach to obtain accurate solutions for heat transfer problems with discontinuous solution. This work also demonstrates the potential of hyperparameter optimization to enhance the predictive capability of PINNs for solving stiff-PDEs.

Disclosure statement

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Funding

This work is part-funded by the United Kingdom Atomic Energy Authority (UKAEA) and the Engineering and Physical Sciences Research Council (EPSRC) under the Grant Agreement Numbers EP/W006839/1, EP/T517987/1, and EP/R012091/1. We acknowledge the support of Supercomputing Wales and AccelerateAI projects, which is part-funded by the European Regional Development Fund (ERDF) *via* the Welsh Government for giving us access to NVIDIA A100 40 GB GPUs for batch training. We also acknowledge the support of NVIDIA academic hardware grant for donating us NVIDIA RTX A5000 24 GB for local testing.

ORCID

Prakhar Sharma http://orcid.org/0000-0002-7635-1857
Llion Evans http://orcid.org/0000-0002-4964-4187
Michelle Tindall http://orcid.org/0000-0003-3034-9636
Perumal Nithiarasu http://orcid.org/0000-0002-4901-2980

References

[1] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019. Available: https://www.sciencedirect.com/science/article/pii/S0021999118307125. DOI: 10.1016/j.jcp.2018.10.045.

[2] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nat. Rev. Phys.*, vol. 3, no. 6, pp. 422–440, Jun. 2021. Available: https://www.nature.com/articles/s42254-021-00314-5. DOI: 10.1038/s42254-021-00314-5.

[3] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: a review," *Acta Mech. Sin.*, vol. 37, no. 12, pp. 1727–1738, Jan. 2021. DOI: 10.1007/s10409-021-01148-1.

[4] J. Yu, L. Lu, X. Meng, and G. E. Karniadakis, "Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems," *arXiv:2111.02801* [*physics*], Nov. 2021. arXiv: 2111.02801. [Online]. Available: http://arxiv.org/abs/2111.02801

[5] X. Meng and G. E. Karniadakis, "A composite neural network that learns from multi-fidelity data: application to function approximation and inverse PDE problems," *J. Comput. Phys.*, vol. 401, pp. 109020, Jan. 2020. Available: https://www.sciencedirect.com/science/article/pii/S0021999119307260. DOI: 10.1016/j.jcp.2019.109020.

[6] Z. Mao, L. Lu, O. Marxen, T. A. Zaki, and G. E. Karniadakis, "DeepM&Mnet for hypersonics: predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators," *J. Comput. Phys.*, vol. 447, pp. 110698, Dec. 2021. Available: https://www.sciencedirect.com/science/article/pii/S0021999121005933. DOI: 10.1016/j.jcp.2021.110698.

[7] L. Lu, M. Dao, P. Kumar, U. Ramamurty, G. E. Karniadakis, and S. Suresh, "Extraction of mechanical properties of materials through deep learning from instrumented indentation," *Proc. Natl. Acad. Sci. USA*, vol. 117, no. 13, pp. 7052–7062, Mar. 2020. Available: DOI: 10.1073/pnas.1922210117.

[8] S. Wang, H. Wang, and P. Perdikaris, "On the eigenvector bias of Fourier feature networks: from regression to solving multi-scale PDEs with physics-informed neural networks," *Comput. Methods Appl. Mech. Eng.*, vol. 384, pp. 113938, Oct. 2021. Available: https://www.sciencedirect.com/science/article/pii/S0045782521002759. DOI: 10.1016/j.cma.2021.113938.

[9] C. Bard and J. Dorelli, "Neural network reconstruction of plasma space–time," *Front. Astron. Space Sci.*, vol. 8, 2021. Available: DOI: 10.3389/fspas.2021.732275.

[10] C. Xu, B. T. Cao, Y. Yuan, and G. Meschke, "Transfer learning based physics-informed neural networks for solving inverse problems in tunneling," *arXiv, Tech. Rep. arXiv:2205.07731* [*cs*] *type: article*, May 2022. arXiv:2205.07731. [Online]. Available: http://arxiv.org/abs/2205.07731.

[11] V. Gopakumar, S. Pamela, and D. Samaddar, "Loss landscape engineering via data regulation on PINNs," *arXiv, Tech. Rep. arXiv:2205.07843* [*physics*] *type: article*, May 2022. arXiv:2205.07843. [Online]. Available: http://arxiv.org/abs/2205.07843.

[12] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM J. Sci. Comput.*, vol. 43, no. 5, pp. A3055–A3081, Jan. 2021. Available: DOI: 10.1137/20M1318043.

[13] S. Wang, X. Yu, and P. Perdikaris, "When and why PINNs fail to train: a neural tangent kernel perspective," *J. Comput. Phys.*, vol. 449, pp. 110768, Jan. 2022. Available: DOI: 10.1016/j.jcp.2021.110768.

[14] Z. Hu, A. D. Jagtap, G. E. Karniadakis, and K. Kawaguchi, "When do extended physics-informed neural networks (XPINNs) improve generalization?" *arXiv:2109.09444* [*cs, math, stat*], Dec. 2021. arXiv: 2109.09444. [Online]. Available: http://arxiv.org/abs/2109.09444.

[15] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, "Physics-informed neural networks with hard constraints for inverse design," *SIAM J. Sci. Comput.*, vol. 43, no. 6, pp. B1105–B1132, Jan. 2021. Available: DOI: 10.1137/21M1397908.

[16] H. Gao, M. J. Zahr, and J.-X. Wang, "Physics-informed graph neural Galerkin networks: a unified framework for solving PDE-governed forward and inverse problems," *Comput. Methods Appl. Mech. Eng.*, vol. 390, pp. 114502, Feb. 2022. Available: https://www.sciencedirect.com/science/article/pii/S0045782521007076. DOI: 10.1016/j.cma.2021.114502.

[17] E. Kharazmi, Z. Zhang, and G. E. M. Karniadakis, "hp-VPINNs: variational physics-informed neural networks with domain decomposition," *Comput. Methods Appl. Mech. Eng.*, vol. 374, pp. 113547, Feb. 2021. Available: https://www.sciencedirect.com/science/article/pii/S0045782520307325. DOI: 10.1016/j.cma.2020.113547.

[18] Y. Shin, "On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs," *CiCP*, vol. 28, no. 5, pp. 2042–2074, Jun. 2020. Available: http://global-sci.org/intro/article_detail/cicp/18404.html. DOI: 10.4208/cicp.OA-2020-0193.

[19] 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," *arXiv:2201.05624* [*physics*], Feb. 2022. arXiv: 2201.05624. [Online]. Available: http://arxiv.org/abs/2201.05624.

[20] P. Sharma, L. Evans, M. Tindall, and P. Nithiarasu, "Stiff-PDEs and physics-informed neural networks," *Arch. Comput. Methods Eng.*, vol. 30, no. 5, pp. 2929–2958, Feb. 2023. DOI: 10.1007/s11831-023-09890-4.

[21] Z. Xiang, W. Peng, W. Zhou, and W. Yao, "Hybrid finite difference with the physics-informed neural network for solving PDE in complex geometries," *arXiv:2202.07926 [physics]*, Feb. 2022. arXiv: 2202.07926. [Online]. Available: http://arxiv.org/abs/2202.07926.

[22] L. McClenny and U. Braga-Neto, "Self-adaptive physics-informed neural networks using a soft attention mechanism," *AAAI-MLPS, Tech. Rep.* 68, Feb. 2019. [Online]. Available: http://ceur-ws.org/Vol-2964/article_68.pdf.

[23] S. Shi, D. Liu, and Z. Zhao, "Non-Fourier heat conduction based on self-adaptive weight physics-informed neural networks," in *2021 40th Chin. Control Conf. (CCC)*, Jul. 2021, pp. 8451–8456. iSSN: 1934–1768.

[24] X.-K. Wen, G.-Z. Wu, W. Liu, and C.-Q. Dai, "Dynamics of diverse data-driven solitons for the three-component coupled nonlinear Schrödinger model by the MPS-PINN method," *Nonlinear Dyn.*, vol. 109, no. 4, pp. 3041–3050, Sep. 2022. Available: DOI: 10.1007/s11071-022-07583-4.

[25] C. Wei and R. Ooka, "Indoor airflow field reconstruction using physics-informed neural network," *Build. Environ.*, vol. 242, pp. 110563, Aug. 2023. Available: https://linkinghub.elsevier.com/retrieve/pii/S0360132323005905. DOI: 10.1016/j.buildenv.2023.110563.

[26] P. Ren, C. Rao, Y. Liu, J.-X. Wang, and H. Sun, "PhyCRNet: physics-informed convolutional-recurrent network for solving spatiotemporal PDEs," *Comput. Methods Appl. Mech. Eng.*, vol. 389, pp. 114399, Feb. 2022. Available: https://www.sciencedirect.com/science/article/pii/S0045782521006514. DOI: 10.1016/j.cma.2021.114399.

[27] S. Markidis, "The old and the new: can physics-informed deep-learning replace traditional linear solvers?," *Front. Big Data*, vol. 4, pp. 669097, Nov. 2021. Available: 10.3389/fdata.2021.669097/full.

[28] P. Escapil-Inchauspé and G. A. Ruz, "Hyper-parameter tuning of physics-informed neural networks: application to helmholtz problems," *ArXiv preprint* 2022. publisher: arXiv Version Number: 2. [Online]. Available: https://arxiv.org/abs/2205.06704.

[29] Y. Wang, X. Han, C.-Y. Chang, D. Zha, U. Braga-Neto, and X. Hu, "Auto-PINN: understanding and optimizing physics-informed neural architecture," *ArXiv preprint* 2022. publisher: arXiv Version Number: 1. [Online]. Available: https://arxiv.org/abs/2205.13748.

[30] P. Pantidis, H. Eldababy, C. M. Tagle, and M. E. Mobasher, "Error convergence and engineering-guided hyperparameter search of PINNs: towards optimized I-FENN performance," *Comput. Methods Appl. Mech. Eng.*, vol. 414, pp. 116160, Sep. 2023. Available: https://linkinghub.elsevier.com/retrieve/pii/S0045782523002840. DOI: 10.1016/j.cma.2023.116160.

[31] G. Cho, D. Zhu, J. J. Campbell, and M. Wang, "An LSTM–PINN hybrid method to estimate lithium-ion battery pack temperature," *IEEE Access*, vol. 10, pp. 100594–100604, 2022. Available: https://ieeexplore.ieee.org/document/9895422/. DOI: 10.1109/ACCESS.2022.3208103.

[32] M. A. Nabian, R. J. Gladstone, and H. Meidani, "Efficient training of physics-informed neural networks via importance sampling," *Comput. Aid. Civil Eng.*, vol. 36, no. 8, pp. 962–977, 2021. Available: DOI: 10.1111/mice.12685.

[33] O. Hennigh *et al.*, "NVIDIA SimNet^{TM}: an AI-accelerated multi-physics simulation framework," in *ICCS, Ser. Lecture Notes in Computer Science*, M. Paszynski, D. Kranzlmüller, V. V. Krzhizhanovskaya, J. J. Dongarra, and P. M. Sloot, Eds. Cham: Springer Int. Pub., 2021, pp. 447–461.

[34] "Modulus user guide," Nov. 2021. release v21.06. [Online]. Available: https://developer.nvidia.com/modulus-user-guide-v2106.

[35] J. Sirignano and K. Spiliopoulos, "DGM: a deep learning algorithm for solving partial differential equations," *J. Comput. Phys.*, vol. 375, pp. 1339–1364, Dec. 2018. Available: https://www.sciencedirect.com/science/article/pii/S0021999118305527. DOI: 10.1016/j.jcp.2018.08.029.

[36] M. Tancik *et al.*, "Fourier features let networks learn high frequency functions in low dimensional domains," in *Adv. Neural Inf. Process. Syst.*, vol. 33. Curran Assoc., Inc. 2020, pp. 7537–7547. Available: https://proceedings.neurips.cc/paper/2020/hash/55053683268957697aa39fba6f231c68-Abstract.html.

[37] N. Rahaman, *et al.*, "On the spectral bias of neural networks," in *Proc. 36th Int. Conf. Mach. Learn, PMLR*. May 2019, pp. 5301–5310, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v97/rahaman19a.html.

[38] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," *J. Comput. Phys.*, vol. 404, pp. 109136, Mar. 2020. Available: https://www.sciencedirect.com/science/article/pii/S0021999119308411. DOI: 10.1016/j.jcp.2019.109136.

[39] L. Martino, V. Elvira, and F. Louzada, "Effective sample size for importance sampling based on discrepancy measures," *Signal Process.*, vol. 131, pp. 386–401, Feb. 2017. Available: https://www.sciencedirect.com/science/article/pii/S0165168416302110. DOI: 10.1016/j.sigpro.2016.08.025.

[40] W. J. Morokoff and R. E. Caflisch, "Quasi-Monte Carlo integration," *J. Comput. Phys.*, vol. 122, no. 2, pp. 218–230, Dec. 1995. Available: https://www.sciencedirect.com/science/article/pii/S0021999185712090. DOI: 10.1006/jcph.1995.1209.

[41] N. Sukumar and A. Srivastava, "Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks," *Comput. Methods Appl. Mech. Eng.*, vol. 389, pp. 114333, Feb. 2022. Available: https://www.sciencedirect.com/science/article/pii/S0045782521006186. DOI: 10.1016/j.cma.2021.114333.

[42] T. Chan and W. Zhu, "Level set based shape prior segmentation," in *CVPR*, vol. 2, Jun. 2005, pp. 1164–1170, iSSN: 1063–6919.

[43] J. E. H. Shaw, "A quasirandom approach to integration in Bayesian statistics," *Ann. Stat.*, vol. 16, no. 2, pp. 895–914, 1988. Available: https://www.jstor.org/stable/2241763.

[44] J. H. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numer. Math.*, vol. 2, no. 1, pp. 84–90, Dec. 1960. Available: . DOI: 10.1007/BF01386213.

[45] N. Hasebrook, F. Morsbach, N. Kannengießer, J. Franke, F. Hutter, and A. Sunyaev, "Why do machine learning practitioners still use manual tuning? A qualitative study," Mar. 2022. arXiv:2203.01717 [cs]. [Online]. Available: http://arxiv.org/abs/2203.01717.

[46] R.-Y. Sun, "Optimization for deep learning: an overview," *J. Oper. Res. Soc. China*, vol. 8, no. 2, pp. 249–294, Jun. 2020. [Online]. Available DOI: 10.1007/s40305-020-00309-6.

[47] D. Bertoin, J. Bolte, S. Gerchinovitz, and E. Pauwels, "Numerical influence of ReLU'(0) on backpropagation," in *Adv. Neural Inf. Process. Syst.*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, Eds., vol. 34. New York: Curran Assoc., Inc., 2021, pp. 468–479. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/043ab21fc5a1607b381ac3896176dac6-Paper.pdf.

[48] H. Nishikawa, "On hyperbolic method for diffusion with discontinuous coefficients," *J. Comput. Phys.*, vol. 367, pp. 102–108, Aug. 2018. Available: https://linkinghub.elsevier.com/retrieve/pii/S0021999118302444. DOI: 10.1016/j.jcp.2018.04.027.

[49] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," Sep. 2017. arXiv:1706.02515 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1706.02515