Physics-informed neural networks for solving partial differential equations.

Prakhar Sharma[1]*, Michelle Tindall[1,2]† and Perumal Nithiarasu[1]§
[1] Zienkiewicz Institute for Modelling, Data and AI,
Faculty of Science and Engineering, Swansea University, Swansea, UK.
[2] Culham Science Centre, United Kingdom Atomic Energy Authority,
Abingdon, OX14 3DB, UK.

Contents

*https://orcid.org/0000-0002-7635-1857
†https://orcid.org/0000-0003-3034-9636
‡https://orcid.org/0000-0002-4901-2980
§Corresponding Author

Abstract

In recent years, Physics-Informed Neural Networks (PINNs) have gained popularity, across different engineering disciplines, as an alternative to conventional numerical techniques for solving partial differential equations (PDEs). PINNs are physics-based deep learning frameworks that seamlessly integrate the measurements and the PDE in a multitask loss function. In forward problems, these measurements are initial (IC) and boundary conditions (BCs), whereas in the inverse problems they are sparse measurements such as temperature recorded by thermocouples. The scope of PDEs applicable in PINNs could include integer-order PDEs, integro-differential equations, fractional PDEs or even stochastic PDEs. This chapter presents a brief state-of-the-art overview of PINNs for solving PDEs. Our discussion primarily focuses on solution to parametric problems, approaches to tackle stiff-PDEs and problems involving complex geometries. The advantages and disadvantages of several PINNs frameworks are also discussed.

Keywords : physics-informed neural networks, partial differential equation

1 Introduction

Conventional numerical techniques for solving PDEs have been a cornerstone in engineering design for years. However, their implementation poses challenges for a variety of problems. Some of the common issues encountered include: mesh dependency of solutions, computational expense in high-dimensional parametric solutions, stress concentration at sharp corners, challenges in achieving convergence and stability, and difficulties in generating adaptive meshes.

Conventional numerical techniques such as finite elements and finite volumes were primarily developed for forward problems. They encounter significant challenges when solving inverse problems. The ill-conditioned nature of inverse problems, need to integrate noisy experimental data and the existence of non-unique solutions makes these problems intractable for conventional numerical techniques. This survey does not cover these challenges associated with inverse problems, instead focuses on the advancements and applications of PINNs in forward problems.

Although early efforts were made to solve differential equations with neural networks, these were constrained by the limitations of smaller-scale neural networks (NNs) and less efficient optimisers [1]−[4]. The recent advancement in existing algorithms and computing power have led to significant achievements in the field [5]−[7].

Recently, Reference [8] introduced the groundbreaking concept of PINNs, a method to seamlessly integrate both data and PDE within a deep learning framework. They demonstrated its

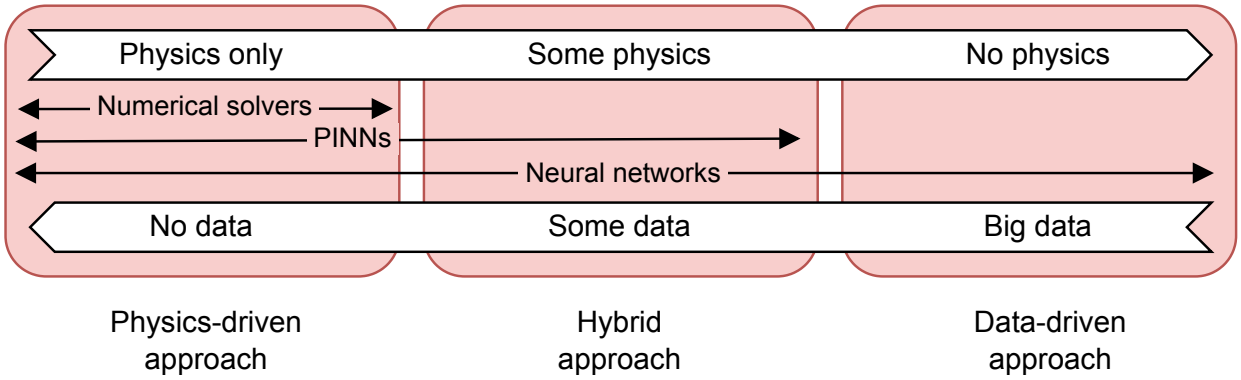

Figure 1: A broad classification of different approaches for solving physics problems.

capability to effectively solve both forward and inverse problems. Unlike conventional numerical techniques, PINNs are inherently meshless, effectively addressing several aforementioned issues such as mesh dependency and adaptive mesh generation. Additionally, by utilising information from previously solved problems, one can accelerate the solution to newer problems through the so called transfer learning method.

The PINNs are particularly noteworthy due to their ability to solve PDEs without the need of additional ground truth, unlike standard training based approaches. This represents a significant advantage over traditional NN-based PDE solvers, which typically requires a large number of ground truth, such as simulation results, for model training.

2 Physics-driven vs. Data-driven approaches

There is a plethora of methodologies exists for tackling physics-based problems, leveraging both conventional numerical solvers and machine learning (ML) models. Before the advent of PINNs, the predominant approaches were either physics-driven or data-driven. As depicted in Figure 1, physics-driven techniques such as conventional numerical solvers, are on the left, whereas purely data-driven methods, including traditional supervised ML models, are on the right.

There are two distinct data-driven methods for enforcing the underlying physics: physics-guided NN (PGNN) and physics-encoded NN (PENN). PGNNs follow a classical supervised learning framework, constructing surrogate models from data derived from experiments and simulations. These models typically necessitate extensive datasets to achieve generalisation [9, 10].

PENN-based models aim to directly encode underlying physical principles within the neural network's architecture, offering an advantage when the explicit form of the differential equations are not well-defined. Among these, two notable approaches stand out: physics-encoded recurrent convolutional NN (PERCNN) [11] and neural ordinary differential equations (NeuralODE) [12].

The PERCNN integrates the non-linear systems directly into the NN by replacing the traditional activation function with a novel element-wise product operation. This modification allows the NN to directly encode the dynamics of physical systems into its computational process, enabling it to handle complex, non-linear behaviours more effectively than traditional NNs.

Table 1: Conventional numerical techniques vs PINNs

	Conventional numerical techniques	PINNs
Basis function	Piecewise polynomial	Neural network
Solution methodology	Numerical approximation & iterative methods	Optimisation problem
PDE embedding	Discretised equations	Loss function
Geometric representation	Mesh	Point cloud

In contrast, NeuralODE reimagines the structure of NNs to parallel the behaviour of ordinary differential equations (ODEs) that can be solved using Euler's method. By designing the NN's layers to represent discrete steps in solving an ODE, NeuralODE allows for the direct application of numerical methods within the NN. This architecture creates a bridge between numerical analysis and machine learning.

PINNs belong to the region between physics-driven and hybrid approach. This signifies that PINNs are capable of solving PDEs without needing additional ground truth, but they can seamlessly integrate noisy experimental data which is a significant advantage over conventional numerical solvers.

3 How PINNs are different from conventional numerical techniques?

Conventional numerical techniques such as finite elements (FEM) and finite volumes (FVM) were primarily developed for forward problems[13, 14, 15]. In these methods, the domain is discretised into a mesh consisting of elements (in FEM) or control volumes (in FVM), with corners or boundaries defined by nodes. In FEM, each element uses a basis function, often described by a piecewise polynomial function, to interpolate the solution within the element based on the nodal values of the field variable. In contrast, FVM divides the domain into control volumes, and the solution is directly approximated within these volumes.

The PINNs are NNs that model the forward problem as an optimisation problem. Instead of discretising the PDE, PINNs formulate them as a part of the loss function. Traditional mesh-based discretisation is replaced with a point cloud throughout the domain. The solution is then inferred at these points once the network has been fitted to minimise the loss function. The key distinctions between conventional numerical techniques and the PINNs are summarised in Table 1.

4 The baseline PINN

Consider a well-posed PDE problem as follows:

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

where the first equation represents the PDE with a temporal derivative u_t and a spatial derivative operator $\mathscr{N}_x[u]$. Here, $u(x,t)$ is the dependent variable, where x and t denote the independent spatial and temporal variables, respectively. The Ω and $\partial\Omega$ denotes the spatial domain and the boundary of the problem. The function $g(x)$ specifies the BCs and $h(x,0)$ denotes the initial condition (IC) at $t = 0$.

The inputs to a PINN for a 2D time dependent problem, are the spatio-temporal coordinates denoted by x, y, z and t . Unlike the mesh generation in conventional numerical methods like FEM, where the structure of the grid can significantly influence solution, the sampling of coordinates for a PINN can be conducted in a more arbitrary manner. There are certain techniques to effectively reduce the number of random points needed while still ensuring comprehensive domain coverage as discussed in Section 5.1. These random points are then fed into the NN, as illustrated in Figure 2. The baseline PINN employs a feed-forward NN (FNN), which comprises several fully connected layers leading to the predicted output, represented by $\hat{u}$ [16].

Similar to traditional supervised ML techniques, in PINNs, the predicted output $\hat{u}$ plays a crucial role in formulating the constraints, which are represented through a loss function [17]. Unlike simpler models, PINNs require multiple loss functions to simultaneously satisfy the PDE, the BCs and the IC, if the problem is transient. This approach results in a multitask loss function, comprising the total loss ($\mathscr{L}$) and the individual loss terms ($\mathscr{L}_{PDE}$, $\mathscr{L}_{BC}$, $\mathscr{L}_{IC}$), which are defined as follows:

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

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

In these equations, N_r, N_b and N_0 represent the number of data points sampled to satisfy the PDE, BCs, and IC, respectively, as mentioned in Equation 2. The coefficients λ_{PDE}, λ_{BC} and λ_{IC}

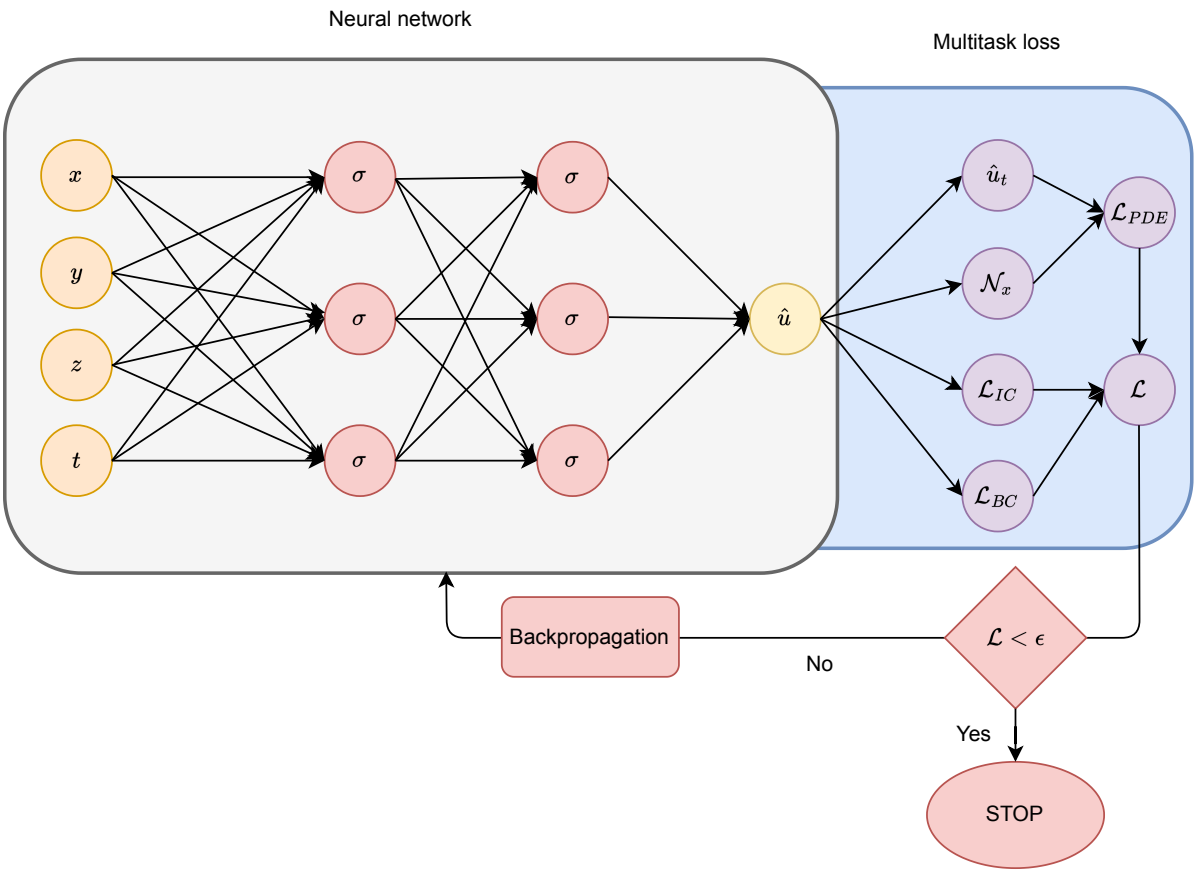

Figure 2: The architecture of a PINN with two hidden layers for a 3D spatio-temporal forward problem. Here the inputs are x, y, z and t, σ is the activated neuron and $\hat{u}$ is the predicted output or the solution to the PDE.

are weighting factors in Equation 2, that help in achieving better convergence and accuracy in the model. The PDE loss $\mathscr{L}_{PDE}$ is the mean squared error (MSE) of the residual of the PDE. Similarly, the BC loss $\mathscr{L}_{BC}$ and the IC loss $\mathscr{L}_{IC}$ are MSEs of the difference between $\hat{u}(x, y, z, t)$ and the known BC and IC at their respective locations. Figure 2 illustrates the overall architecture of a PINN for a 3D spatio-temporal problem.

In the baseline PINN framework, gradient-based optimisers are employed to minimise the total loss $\mathscr{L}$, by adjusting the weights of the NN during the training process [18]. Among these optimisers, Adam (Adaptive Moment Estimation) and L-BFGS (Limited memory Broyden- Fletcher-Goldfarb- Shanno) are frequently utilised due to their efficiency in handling large-scale optimisation problems.

Both Adam and L-BFGS bring distinct advantages to the training process of PINNs. Adam's adaptive learning mechanism can lead to faster convergence, especially in the early stages of training. Whereas, L-BFGS is often preferred in the later stages of training when fine-tuning around minima is required, as it can provide more accurate updates by approximating second-order curva-

ture information.

Given that the problem is well-posed, there exists a unique solution [19]. The enforcement of the loss terms $\mathscr{L}_{PDE}$, $\mathscr{L}_{BC}$, $\mathscr{L}_{IC}$ within the PINN framework contributes to maintaining the well-posedness of the problem, thereby facilitating the convergence towards a unique solution.

4.1 Nature of solutions in PINNs

The PINNs were originally developed as solvers for PDEs. In conventional applications, once a PDE is solved within the specified domain with prescribed BCs and IC, further inference on new spatio-temporal locations is typically unnecessary. However, by incorporating validation dataset, PINNs can be generalised to interpolate or extrapolate the field variables at new spatio-temporal locations. This predictive capacity aligns with the conventional ML techniques, where the availability of ground truth enables the model to learn and make accurate predictions on new spatio-temporal locations. It is important to note that this generalisation approach deviates from the traditional use-case of PDE solvers, which don't rely on ground truth data.

5 Developments in the PINN frameworks

The PINNs have rapidly evolved, with significant advancements in each of their core components such as sampling strategy, network architecture, activation function etc. These enhancements not only improved the accuracy and efficiency of PINNs but also contribute to the broader field of ML. This section will briefly discuss these developments, highlighting how they address previous limitations of the PINNs.

5.1 Sampling

Sampling plays a crucial role in the training of PINNs, just as it does in other ML techniques. For the PINNs, this involves generating a point cloud within the domain of interest, which serves as the training data. Various strategies can be employed to sample these points effectively.

A common practice in PINNs is to employ low-discrepancy quasi-random sequences. These sequences are advantageous as they require fewer points than uniformly distributed random points to achieve a comparable level of domain coverage. Essentially, these sequence "spread out" the points in such a way that they are evenly distributed across all the dimensions. Figure 3 demonstrates the distribution of 10 points within a unit square for various sampling methods, including random, grid, Latin hypercube sampling (LHS), Sobol, Halton, and Hammersley sequence. The random sampling shows no pattern, which can lead to clustering and gaps. The grid pattern, does not randomise the locations, which may not capture the local variations in the solution. In contrast, quasi-random sequences like the Sobol, Halton, and Hammersley methods provide a more uniform distribution without clustering, which is beneficial for capturing the local variations in the solution [20, 21].

Importance sampling can be seen as substitute of adaptive mesh refinement in PINNs. Rather than using the same sampled points in each training iteration, the points are drawn from a distribu-

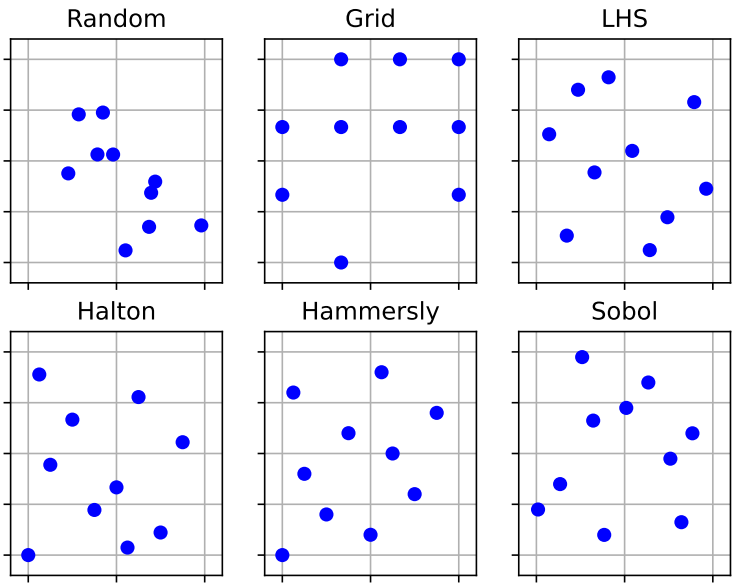

Figure 3: The comparison of various sampling methods.

tion that is proportional to the total loss, $\mathscr{L}$. Consequently, regions with higher pointwise total loss are sampled more densely, thereby focusing areas where the model needs the most improvement [22, 23, 24].

Figure 4 illustrates the evolution of sampling strategies in a 1D feature space, x, ranging from 0 to 1. Initially, 1000 sample points are distributed uniformly across the feature space using LHS, as depicted by the evenly spaced histogram in blue in the top plot. As the training progresses, samples are drawn from a distribution that aligns with the total loss $\mathscr{L}$. This distribution, represented by the histogram in orange, is concentrated around regions where the pointwise total loss $\mathscr{L}$, shown in the bottom plot, is higher. By dynamically adjusting the sampling density in accordance with the pointwise total loss $\mathscr{L}$, the PINN effectively focuses on learning complex dynamics within the feature space.

5.2 Imbalanced loss terms

The individual loss terms in a PINN can exhibit significant differences in magnitude, leading to imbalanced contributions to the total loss. For instance, the $\mathscr{L}_{PDE}$, which often includes higher-order derivatives, might be substantially lower than the $\mathscr{L}_{BC}$. This disparity can result in the PINN predominantly learning to satisfy the BCs while ignoring the PDE. Such an imbalance can yield erroneous behaviour, as the problem effectively becomes ill-posed.

A predominant approach to address this issue is the introduction of balancing coefficients for each loss term. These coefficients denoted as λ_{PDE}, λ_{BC} and λ_{IC} are multiplied with the respective terms in the total loss function (Equation 2). By adjusting these coefficients, the relative magnitude of respective loss terms can be balanced [25].

Efforts to automatically adjust these coefficients have led to notable developments, such as self-

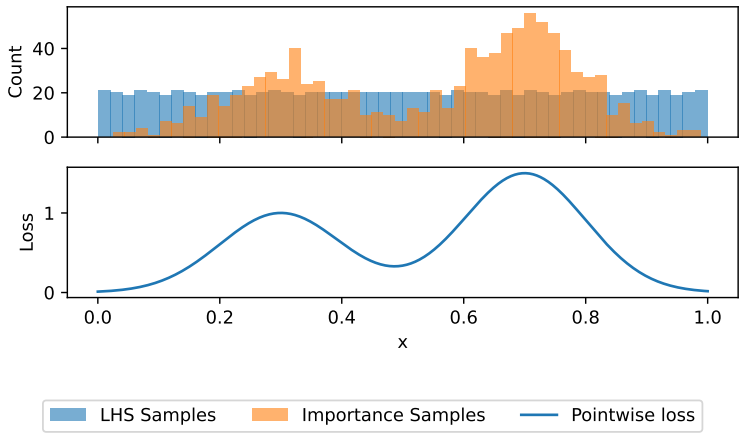

Figure 4: The comparison of initial and importance sampling strategies in a 1D feature space x. The top histogram shows initial samples obtained via LHS, and the target distribution of samples derived from importance sampling. The bottom plot shows the pointwise total loss $\mathscr{L}$ across the 1D feature space x, highlighting regions of higher loss where more points are sampled.

adaptive PINNs [26] and the self-adaptive weight PINN [27] and the implementation of algorithms like learning rate annealing [28] and neural tangent kernel [29].

5.3 Activation function

An activation function, denoted as σ, imparts nonlinearity to a NN enabling it to learn complex input-output relationships. Selection of a suitable activation function can affect the convergence. The choice of an appropriate activation function is crucial for convergence in PINNs, as they require smooth activation functions to compute higher-order derivatives present in PDEs. Thus, activation functions with discontinuities, such as the rectified linear unit (ReLU), exponential linear unit (ELU), or scaled exponential linear units (SELU), should generally be avoided [30].

Mathematically, a NN is a function, where the linear combination of network's weights w and previous layer's input is passed through the activation function σ which serves as the input to the next layer (Equation 4). Reference [31] proposed the concept of a global adaptive activation function (GAAF), where a trainable parameter A is also passed through the activation function (Equation 5). This parameter, acting as the slope of the activation function, allows for more sophisticated feature transformations between the hidden layers. Later, Reference [32], developed the layer-wise locally adaptive activation functions (L-LAAF), incorporating a distinct trainable parameter, denoted as $A^{(2)}$, in each hidden layer (Equation 6). This layer-specific adaptability further enhances the NN's capacity to capture complex behaviours. The standard, GAAF and L-LAAF structures are given as,

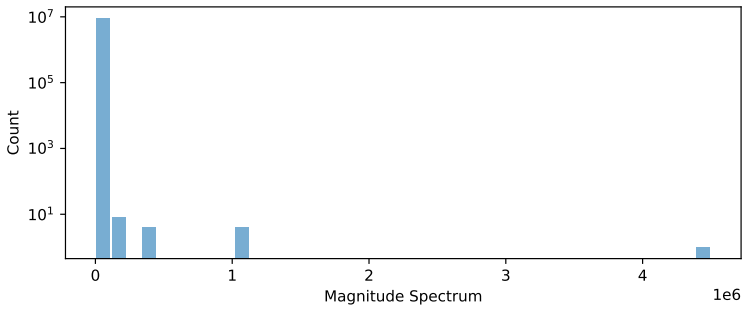

Figure 5: Histogram of the magnitude spectrum obtained from the Fourier transform of 2D spatial data, indicating a the presence of high-frequency components due to discontinuities.

$$\hat{u} = w^{(3)}\sigma\left(w^{(2)}\sigma\left(w^{(1)}X\right)\right) \tag{4}$$

$$\hat{u} = w^{(3)}\sigma\left(Aw^{(2)}\sigma\left(Aw^{(1)}X\right)\right) \tag{5}$$

$$\text{and} \quad \hat{u} = w^{(3)}\sigma\left(A^{(2)}w^{(2)}\sigma\left(A^{(1)}w^{(1)}X\right)\right). \tag{6}$$

5.4 Spectral bias in the NN

Spectral bias is a learning bias of NNs towards low-frequency functions. This is a challenge when dealing with high-frequency functions that represent sharp variations, especially in solutions within low-dimensional domains. In Figure 5, we present a histogram of the magnitude spectrum derived from a Fourier transform of 2D spatial data exhibiting discontinuities. While the distribution mostly consists of low-frequency components, there are a few high-frequency attributable to the discontinuities. These high-frequency components pose a challenge for traditional FNN architectures, potentially leading to non-convergence issues during training.

Reference [33] proposed the Fourier NN, an approach that employs input encoding to project data from low-dimensional domains into a higher-dimensional Fourier space using a frequency matrix. Equation 7 shows the high-dimensional training dataset, mitigating the effects of spectral bias.

$$\begin{bmatrix}\sin(2\pi fX)\\ \cos(2\pi fX)\end{bmatrix}^T X \tag{7}$$

where f is trainable frequency matrix and X is the the data in low dimensional domains. Similar input encodings have been utilised in modified Fourier network [29], sinusoidal representation networks (SiReNs)[34] and the deep Galerkin method (DGM) network [35]. A comprehensive survey by Sharma et. al. [36] discusses solutions to discontinuous problems with PINNs, detailing these architectures among others.

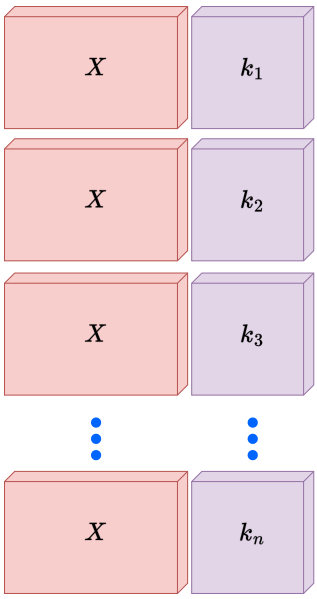

Figure 6: Schematic representation of the training dataset with varying parameter k.

6 Applications of PINNs to Forward Problems

In this section, we will briefly discuss the application of PINNs to forward problems. We highlight scenarios where PINNs offer solutions to challenges commonly faced by conventional numerical methods, such as handling parametric problems, complex geometries, and transfer learning.

6.1 Parametric problems

The PINNs can be easily extended to solve the problem over a range of parameters, by including them as additional features in the training dataset. These parameters can encompass BCs, IC, coefficients of the PDE and even the geometry of the domain. Consider a training dataset where X represents the input features $[x, y, t]$, for a 2D time-dependent problem, alongside a range of parameters k_i, where i ranges from 1 to n. To learn the parametric solutions, the PINN's training dataset is constructed by concatenating the X with each instance of k_i as shown in Figure 6[37].

Recently, Reference [38] proposed physics-informed deep operator network (PIDeepONets), an operator learning architecture to solve parametric problems. Similar to PINNs, PIDeepONets only require the PDE, IC and BCs. While a discussion on PIDeepONets is beyond the scope of this chapter, those interested can refer to a survey in Reference [39].

6.2 Complex geometry

Reference [40] proposed a so called conservative PINN, a space decomposition for the PINNs. This is similar to the concept of elements in FEM, where each element has its own trail function. However, the domain can be decomposed in any arbitrary way without needing any special algo-

rithm as opposed to FEM. Specifically, two additional loss terms were introduced to account for the mismatch in the $\mathscr{L}_{PDE}$ and $\hat{u}$ at the interface of two neighbouring sub-domains. XPINNs further advanced this by handling space-time domain decomposition (DD) for any irregular geometry [41]. The XPINNs were able to handle problems with sharp gradient over complicated geometry, at the cost of longer training time. Parallel PINNs addressed this by introducing efficient parallel algorithms [42]. The work in Reference [43] developed theoretical insights on the convergence and generalisation properties of PINNs, enabling accurate modelling of discontinuities, like shockwaves, with prior knowledge of their locations.

6.3 Transfer learning

Transfer learning stands out as a key advantage of PINNs when compared to conventional numerical methods. It allows the utilisation of a model trained on one problem, referred to as the base task, to solve similar problem, known as target task. The base task is generally a simpler problem, which may differ from the target task in terms of the geometry, BC or PDE. By utilising a PINN trained on the base task, we can approach more complicated target task, leveraging the pre-trained model, and avoid the lengthy and computationally intensive training from scratch [44, 45].

Figure 7, illustrates transfer learning of a pre-trained model to solve the target task with a different geometry and BCs denoted by "*". We refer the reader to comprehensive overview of transfer learning with PINNs presented in Reference [46].

7 Examples

We present three test cases: 1D burgers equation, Allen-Cahn equation and Lid driven cavity to showcase the capability of various tools that we have discussed so far. In both the test cases, we used Adam optimiser, with Xavier normal weight initialisation and hyperbolic tangent activation function. We sampled the initial set of collocation points with Sobol sequence in both the test cases.

7.1 1D Burgers equation

The 1D Burgers equation is a time-dependent problem with details given in Equation 8.

$$\begin{aligned} u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1,1], \quad t \in [0,1], \\ u(0,x) = -\sin(\pi x), \\ u(t,-1) = u(t,1) = 0 \end{aligned} \tag{8}$$

The Burgers equation is a second-order non-linear convection-diffusion problem with an analytical solution available in Reference [47]. The presence of a non-linear convection term uu_x exhibits a discontinuity over time.

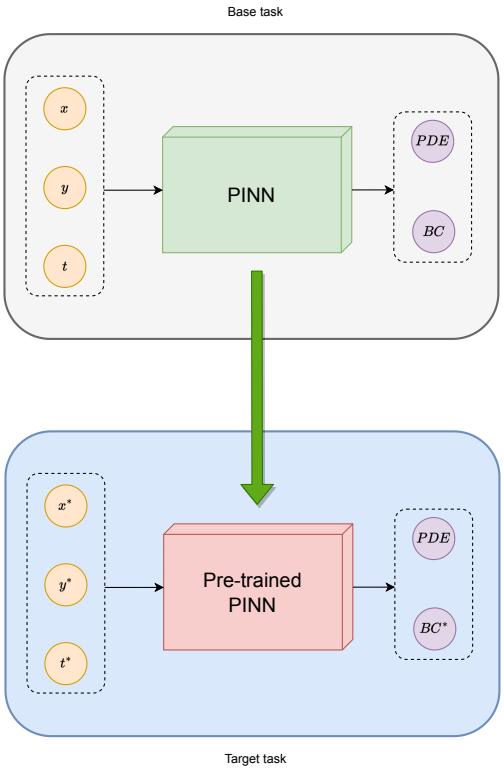

Figure 7: Transfer learning from a base task to a target task with different geometry and BC, indicated by '*'.

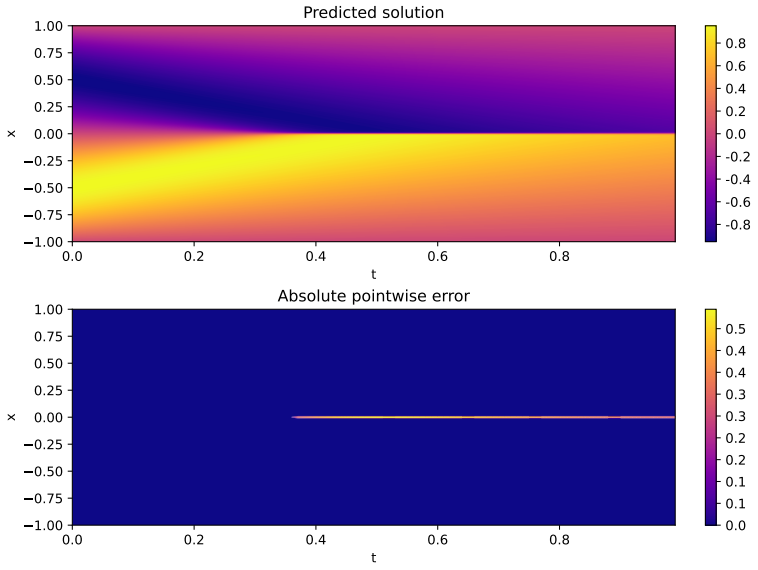

Figure 8: Baseline PINN predicted solution of 1D Burgers equation is shown on the top and absolute pointwise error between the analytical solution and PINN predicted solution is shown on the bottom.

We employed a FNN architecture, i.e., a baseline PINN, and trained the model for 10k iterations. Figure 8, shows the PINN predicted solution and the absolute pointwise error between the analytical solution and PINN predicted solution.

The observed discontinuity in the solution may be attributed to the spectral bias inherent to FNNs, which we have previously discussed. Thus, the network is not able to capture the discontinuity as shown in Figure 8. This limitation is reflected in the computed relative L2 error of 5.17%.

7.2 1D Allen-Cahn equation

The Allen-Cahn equation is used to model the process of phase separation and is characterised by a diffusion term and a non-linear reaction term that drives the system towards minimising its free energy, often resulting in the creation of interfaces between phases over time. Consider the Allen–Cahn equation along with periodic BC (Equation 9).

$$\begin{aligned} u_t - 0.0001u_{xx} + 5u^3 - 5u = 0, x \in [-1,1], t \in [0,1], \\ u(0,x) = x^2\cos(\pi x), \\ u(t,-1) = u(t,1), \\ u_x(t,-1) = u_x(t,1) \end{aligned} \tag{9}$$

The ground truth was generated using spectral Fourier discretisation and fourth-order explicit Runge–Kutta time integrator. The solution $u(x,t)$ evolves over time due to the combined effects of

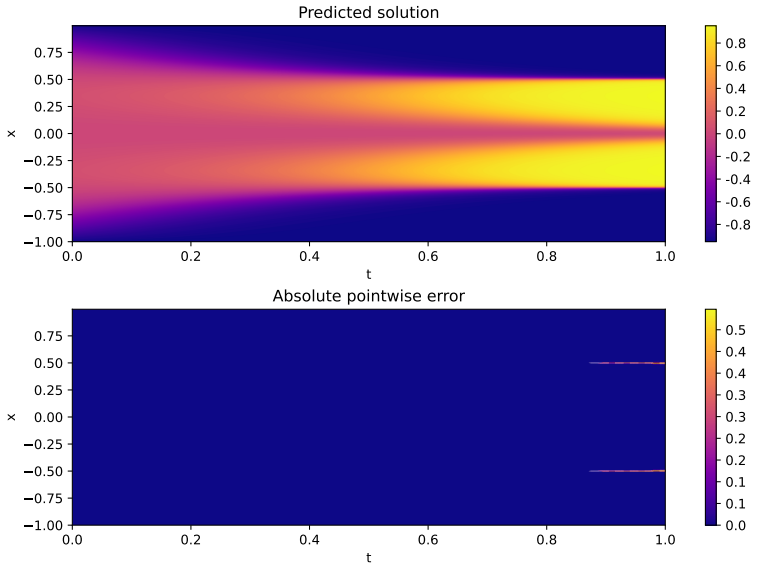

Figure 9: Baseline PINN predicted solution of 1D Allen-Cahn equation is shown on the top and absolute pointwise error between the analytical solution and PINN predicted solution is shown on the bottom.

diffusion and reaction. The diffusion term $-0.0001u_{xx}$ tends to smooth out variations in u, while the cubic non-linear reaction term $u^3 - 5u$ can create multiple stable states over time.

The initial condition $u(0,x) = x^2\cos(\pi x)$ is smooth and contains no discontinuities. However, as time progresses, the solution develops sharp transitions between the phases due to the non-linear dynamics, which resembles a stiffness in the numerical solution.

Similar to the 1D Burgers equation the baseline PINN couldn't capture the regions with stiff solution, as shown in Figure 9, leading to a relative L2 error of 1.32%. However, with the application of Fourier NN the PINN was able to accurately capture the stiff regions in the solution, as shown in Figure 10, thus reducing the relative L2 error to 0.06%. This example underscores how employing a high-dimensional training dataset can effectively mitigate the spectral bias.

7.3 Lid driven cavity

The lid-driven cavity is a well-known benchmark problem in computational fluid dynamics. It consists of a square cavity with of three rigid walls having no-slip conditions and the top lid moving with a tangential unit velocity $u = 1$. The lower left corner of the domain has a reference pressure p of 0 as shown in Figure 11.

The Lid driven cavity uses the 2D steady-state incompressible Navier-Stokes equations to model fluid flow, as detailed in Equation 10.

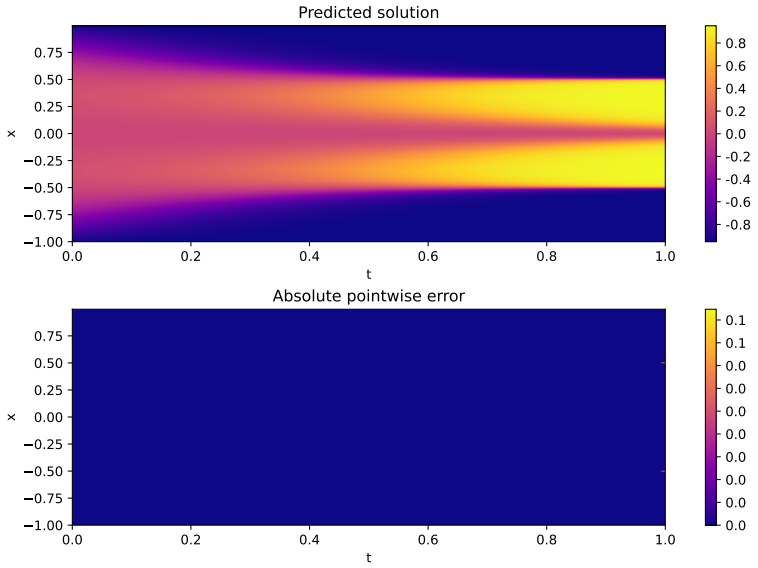

Figure 10: Fourier NN predicted solution of 1D Allen-Cahn equation is shown on the top and absolute pointwise error between the analytical solution and PINN predicted solution is shown on the bottom side.

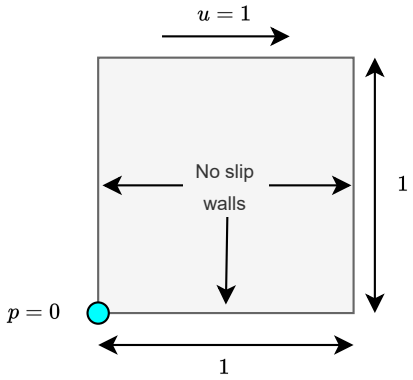

Figure 11: Geometry of the lid-driven cavity problem.

$$
\begin{aligned}
\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \\
u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= -\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\
u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= -\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)
\end{aligned}
\tag{10}
$$

where u and v are velocities in x and y direction, p is the pressure, ν is the kinematic viscosity

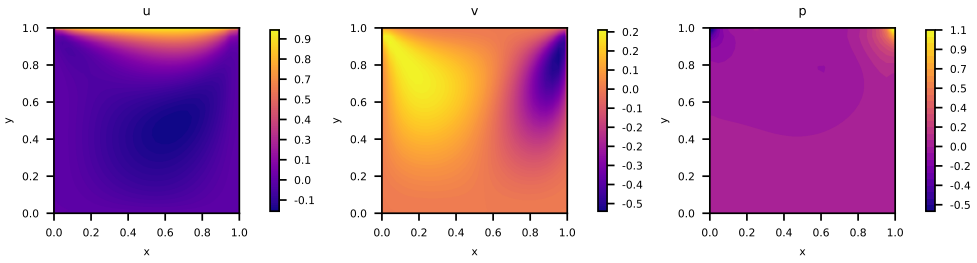

Figure 12: Numerical solution for the lid-driven cavity problem, obtained using the SIMPLE algorithm, showcasing the velocity field and pressure distribution.

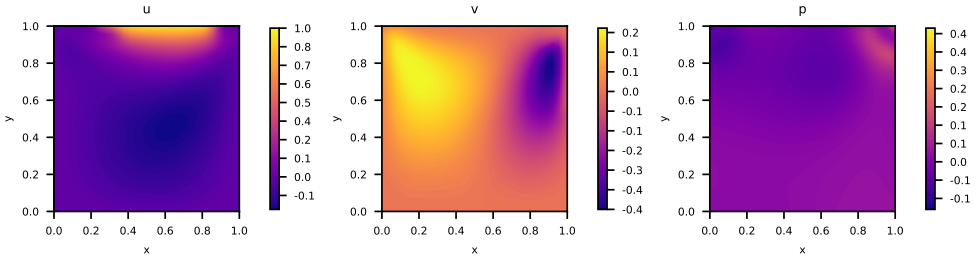

Figure 13: Baseline PINN predicted solution of the lid-driven cavity.

and ρ is the density of the fluid. Focusing on a Reynolds number of 100 to simulate laminar flow conditions, we set $\rho = 1$ and $\nu = 0.01$. The numerical solution was acquired through the SIMPLE (Semi-Implicit Method for Pressure-Linked Equation) algorithm [48], and Figure 12 shows this solution.

The conflicting BCs on top left and top right corners result in sharp discontinuities. Similar to the Burgers equation, the baseline PINN struggles to accurately capture these discontinuities due to spectral bias. This limitation is shown in Figure 13.

To mitigate the issue of spectral bias, we employed the Deep Galerkin Method (DGM) architecture [35], coupled with self-adaptive PINN's weight balancing algorithm, L-LAAF and importance sampling. This approach resulted in a reduction of the relative L2 error by nearly 50% compared to the baseline PINN, as shown in Table 2. Figure 14, showcases the solution predicted by DGM, highlighting the improved accuracy.

Following the DGM-based model, we integrated it with the XPINN framework, dividing the domain into three equally spaced sub-domains along the x-direction. This division resulted in three times reduction in the relative L2 error compared to the DGM-based model, as detailed in Table 2. This shows the effectiveness of domain decomposition technique while solving problem involving discontinuities.

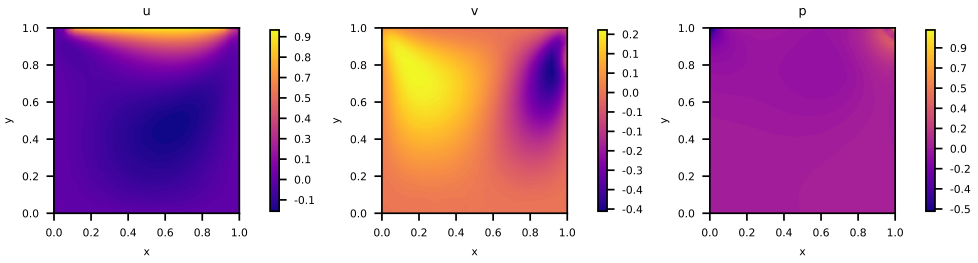

Figure 14: DGM predicted solution of the lid-driven cavity.

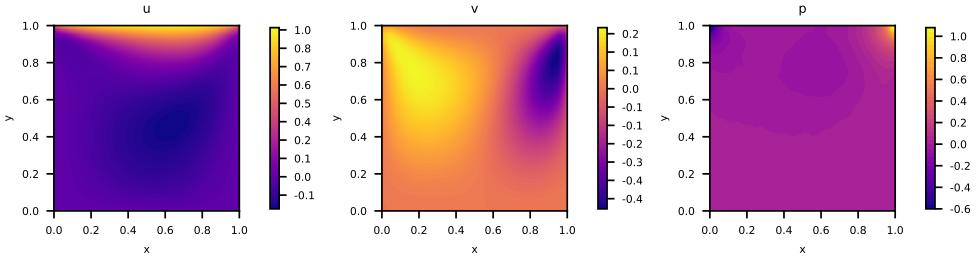

Figure 15: DGM predicted solution of the lid-driven cavity with three sub-domains equally spaced along the x-direction.

8 Conclusions

Physics-informed neural networks have emerged as a powerful framework for solving problems involving PDEs. By seamlessly integrating physical laws into deep learning model, they allow exploring a wide range of scientific and engineering challenges. This chapter has provided a comprehensive overview of PINNs, covering the core components, advancements in sampling strategies, multitask loss function challenges, adaptive activation functions, and the issues with spectral bias, enhanced accuracy and efficiency of PINNs. Notably, PINNs offer significant advantages over traditional numerical techniques, enabling:

- The application of transfer learning to leverage insights from solved problems on new and similar challenges.
- Efficiently addressing high-dimensional parametric problems.

Table 2: Relative L2 error (in %) for Lid driven cavity

	u	v	p
Baseline PINN	47.7	26.3	80.4
DGM architecture	26.8	18.3	41.3
DGM with 3 sub-domain	8.6	9.7	12.6

- Simplified domain decomposition for stiff problems, avoiding complex algorithms required for mesh generation.

We solved the 1D Burgers equation, 1D Allen-Cahn equation and the lid-driven cavity problem to showcase the various improvements over the baseline PINN that allowed for effective handling of discontinuities. These improvements mark PINNs as a powerful tool for efficiently solving complex problems. As we enhance PINNs further, we believe their full potential is yet to be realised, promising more innovative solutions in the future.

Acknowledgements

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 40GB GPUs for batch training. We also acknowledge the support of NVIDIA academic hardware grant for donating us NVIDIA RTX A5000 24GB for local testing.

Statements and Declarations

References

[1] Hyuk Lee and In Seok Kang. Neural algorithm for solving differential equations. *J. Comput. Phys.*, 91(1):110–131, November 1990.

[2] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.*, 9(5):987–1000, September 1998. Conference Name: IEEE Transactions on Neural Networks.

[3] I.E. Lagaris, A.C. Likas, and D.G. Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Trans. Neural Netw.*, 11(5):1041–1049, September 2000. Conference Name: IEEE Transactions on Neural Networks.

[4] A. Malek and R. Shekari Beidokhti. Numerical solution for high order differential equations using a hybrid neural network—Optimization method. *Appl. Math. Comput.*, 183(1):260–271, December 2006.

[5] Keith Rudd and Silvia Ferrari. A constrained integration (CINT) approach to solving partial differential equations using artificial neural networks. *Neurocomputing*, 155:277–285, May 2015.

[6] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations. *SIAM J. Sci. Comput.*, January 2018. Publisher: Society for Industrial and Applied Mathematics.

[7] Maziar Raissi, Zhicheng Wang, Michael S. Triantafyllou, and George Em Karniadakis. Deep learning of vortex-induced vibrations. *J. Fluid Mech.*, 861:119–137, February 2019. Publisher: Cambridge University Press.

[8] 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.*, 378:686–707, February 2019.

[9] Haakon Robinson, Suraj Pawar, Adil Rasheed, and Omer San. Physics guided neural networks for modelling of non-linear dynamics. *Neural Networks*, 154:333–345, 2022.

[10] Kristinn Andersen, George E Cook, Gabor Karsai, and Kumar Ramaswamy. Artificial neural networks applied to arc welding process modeling and control. *IEEE Transactions on industry applications*, 26(5):824–830, 1990.

[11] Chengping Rao, Hao Sun, and Yang Liu. Hard encoding of physics for learning spatiotemporal dynamics. *arXiv preprint arXiv:2105.00557*, 2021.

[12] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[13] O.C. Zienkiewicz, R.L. Taylor, and P. Nithiarasu. *The Finite Element Method for Fluid Dynamics*. Elsevier, Oxford, 7th edition, 2013.

[14] P. Nithiarasu, R.W. Lewis, and K.N. Seetharamu. *Fundamentals of the finite element method for heat and mass transfer*. Wiley, 2015.

[15] C. Hirsch. *Numerical Computation of Internal and External Flows*, volume 1. John Wiley & Sons, New York, 1992.

[16] Yuntian Chen, Dou Huang, Dongxiao Zhang, Junsheng Zeng, Nanzhe Wang, Haoran Zhang, and Jinyue Yan. Theory-guided hard constraint projection (HCP): A knowledge-based data-driven scientific machine learning method. *J. Comput. Phys.*, 445:110624, November 2021.

[17] Giorgio Gnecco, Marco Gori, Stefano Melacci, and Marcello Sanguineti. Learning with Hard Constraints. In Valeri Mladenov, Petia Koprinkova-Hristova, Günther Palm, Alessandro E. P. Villa, Bruno Appollini, and Nikola Kasabov, editors, *Artificial Neural Networks and Machine Learning – ICANN 2013*, Lecture Notes in Computer Science, pages 146–153, Berlin, Heidelberg, 2013. springer.

[18] Ange Tato and Roger Nkambou. IMPROVING ADAM OPTIMIZER. page 4, 2018.

[19] Persi Diaconis and Mehrdad Shahshahani. On Nonlinear Functions of Linear Combinations. *SIAM J. Sci. Stat. Comput.*, 5(1):175–191, March 1984. Publisher: Society for Industrial and Applied Mathematics.

[20] William J. Morokoff and Russel E. Caflisch. Quasi-Monte Carlo Integration. *J. Comput. Phys.*, 122(2):218–230, December 1995.

[21] C. Bard and J.C. Dorelli. Neural Network Reconstruction of Plasma Space-Time. *Front. Astron. Space Sci.*, 8, 2021.

[22] Christian P. Robert and George Casella. Monte Carlo Integration. In Christian P. Robert and George Casella, editors, *Monte Carlo Statistical Methods*, Springer Texts in Statistics, pages 71–138. springer, New York, NY, 1999.

[23] Luca Martino, Víctor Elvira, and Francisco Louzada. Effective sample size for importance sampling based on discrepancy measures. *Signal Process.*, 131:386–401, February 2017.

[24] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Comput.-Aided Civ. Infrastruct. Eng.*, 36(8):962–977, 2021. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12685.

[25] Colby L. Zhao. Solving Allen-Cahn and Cahn-Hilliard Equations using the Adaptive Physics Informed Neural Networks. *Comm. Comput. Phys.*, 29(3), July 2020.

[26] Levi McClenny and Ulisses Braga-Neto. Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism. Technical Report 68, AAAI-MLPS, February 2019.

[27] Shuyan Shi, Ding Liu, and Zhongdan Zhao. Non-Fourier Heat Conduction based on Self-Adaptive Weight Physics-Informed Neural Networks. In *2021 40th Chin. Control Conf. (CCC)*, pages 8451–8456, July 2021. ISSN: 1934-1768.

[28] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM J. Sci. Comput.*, 43(5):A3055–A3081, January 2021. Publisher: Society for Industrial and Applied Mathematics.

[29] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *J. Comput. Phys.*, 449:110768, January 2022.

[30] Prakhar Sharma, Llion Evans, Michelle Tindall, and Perumal Nithiarasu. Hyperparameter selection for physics-informed neural networks (pinns)–application to discontinuous heat conduction problems. *Numerical Heat Transfer, Part B: Fundamentals*, pages 1–15, 2023.

[31] Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *J. Comput. Phys.*, 404:109136, March 2020.

[32] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society A*, 476(2239):20200334, 2020.

[33] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Adv. Neural Inf. Process. Syst.*, volume 33, pages 7537–7547. Curran Assoc., Inc., 2020.

[34] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Adv. Neural Inf. Process. Syst.*, volume 33, pages 7462–7473. Curran Assoc., Inc., 2020.

[35] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.*, 375:1339–1364, December 2018.

[36] Prakhar Sharma, Llion Evans, Michelle Tindall, and Perumal Nithiarasu. Stiff-PDEs and Physics-Informed Neural Networks. *Arch. Comput. Methods Eng.*, February 2023.

[37] Enrico Schiassi, Carl Leake, Mario De Florio, Hunter Johnston, Roberto Furfaro, and Daniele Mortari. Extreme Theory of Functional Connections: A Physics-Informed Neural Network Method for Solving Parametric Differential Equations. Technical Report arXiv:2005.10632, arXiv, May 2020. arXiv:2005.10632 [physics, stat] type: article.

[38] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Sci. Adv.*, 7(40):eabi8605, September 2021. Publisher: American Association for the Advancement of Science.

[39] Seid Koric and Diab W Abueidda. Data-driven and physics-informed deep learning operators for solution of heat conduction equation with parametric heat source. *International Journal of Heat and Mass Transfer*, 203:123809, 2023.

[40] Ameya D. Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Comput. Methods Appl. Mech. Eng.*, 365:113028, June 2020.

[41] Ameya D. Jagtap and George Em 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(5):2002–2041, 2020.

[42] Khemraj Shukla, Ameya D Jagtap, and George Em Karniadakis. Parallel physics-informed neural networks via domain decomposition. *Journal of Computational Physics*, 447:110683, 2021.

[43] Zheyuan Hu, Ameya D. Jagtap, George Em Karniadakis, and Kenji Kawaguchi. When Do Extended Physics-Informed Neural Networks (XPINNs) Improve Generalization? *arXiv:2109.09444 [cs, math, stat]*, December 2021. arXiv: 2109.09444.

[44] Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Comput. Methods Appl. Mech. Eng.*, 379:113741, June 2021.

[45] Chen Xu, Ba Trung Cao, Yong Yuan, and Günther Meschke. Transfer learning based physics-informed neural networks for solving inverse problems in tunneling. Technical Report arXiv:2205.07731, arXiv, May 2022. arXiv:2205.07731 [cs] type: article.

[46] Konstantinos Prantikos, Stylianos Chatzidakis, Lefteri H Tsoukalas, and Alexander Heifetz. Physics-informed neural network with transfer learning (tl-pinn) based on domain similarity measure for prediction of nuclear reactor transients. *Scientific Reports*, 13(1):16840, 2023.

[47] Mayur P. Bonkile, Ashish Awasthi, C. Lakshmi, Vijitha Mukundan, and V. S. Aswin. A systematic literature review of Burgers' equation with recent advances. *Pramana - J Phys*, 90(6):69, April 2018.

[48] Timothy Francis Miller. *Application of multilevel methods to a semi-implicit pressure-linked algorithm*. The Pennsylvania State University, 1987.