Adaptive Physics-informed Neural Networks: A Survey

Anonymous authors
Paper under double-blind review

Abstract

Physics-informed neural networks (PINNs) have emerged as a promising approach for solving partial differential equations (PDEs) using neural networks, particularly in data-scarce scenarios due to their unsupervised training capability. However, a key limitation is the need for re-optimization with each change in PDE parameters, similar to the challenge in traditional numerical methods where each system of equations corresponds to a specific PDE instance. This characteristic poses a barrier to the widespread adoption of PINNs across scientific and engineering applications. This survey explores research addressing this limitation through transfer learning and meta-learning, synthesizing insights to establish a foundation for efficient data generation strategies tailored to PINNs. These methods can potentially improve PINNs' training efficiency, enabling quicker adaptation to new PDEs with fewer data and computational demands. While numerical methods directly solve systems of equations to derive solutions, neural networks implicitly learn solutions by adjusting their parameters. One notable advantage of neural networks lies in their capacity to abstract away from specific problem domains, enabling them to retain, discard, or adapt learned representations to efficiently address similar problems. By understanding how these techniques can be applied to PINNs, this survey seeks to identify promising directions for future research to enable the widespread adoption of PINNs across a wide range of scientific and engineering applications.

1 Introduction

Advances in machine learning have led to important applications in various fields, such as computer vision (enabling technologies like self-driving cars), natural language processing (powering intelligent agents and chat-bots), and image generation (facilitating media creation). With such success, there has been growing interest in developing Machine Learning (ML) solutions to solve problems in science and engineering. However, unlike many other fields where data is abundant or easily acquired, these fields often face data limitations due to the high cost of experiments and simulations needed to generate data. Therefore, to ease the development of ML approaches in these disciplines, intelligent methods that are data and computationally efficient need to be created. To this end, other domains have tackled similar problems with techniques such as transfer learning, meta-learning, and few-shot learning, indicating a significant potential for applying these techniques in science and engineering.

One specific application in science and engineering where these efficient ML models can be particularly beneficial is determining the approximate solutions of PDEs. PDEs are fundamental in modeling and describing natural phenomena across various scientific and engineering domains. Traditionally, these equations are solved numerically, which (in some cases) can become prohibitively expensive, especially when dealing with nonlinear and high-dimensional problems (Han et al., 2018). This challenge limits their application in areas where fast evaluation of a PDE is required. Recognizing this challenge, neural networks have been explored as a potential solution, offering advantages in effectively modeling complex nonlinearities (Raissi et al., 2019; Khoo et al., 2021; Sirignano & Spiliopoulos, 2018; Cuomo et al., 2022), presenting the potential for faster evaluation compared to classical iterative solvers, as well as offering mesh-free solutions not constrained to computational grids (Jiang et al., 2023; Li et al., 2020; Raissi et al., 2019; Cuomo et al., 2022). Moreover, machine learning techniques provide an approach to solving *inverse problems*, where the goal is to infer

unknown parameters or initial/boundary conditions from observed data, a task challenging for numerical methods (Arridge et al., 2019; Cai et al., 2021). In addition, machine learning implementations are simpler than numerical methods allowing for faster development and easy maintenance (Cai et al., 2021).

ML methods for solving PDEs can broadly be categorized into two types: neural surrogates and neural PDE solvers[1][2]. Neural surrogates, including physics-guided neural networks (Faroughi et al., 2023) and neural operators, function by training neural networks in a regression manner using data generated from numerical solvers. The most popular among these are neural operators, which approximate nonlinear mappings between infinite-dimensional function spaces using datasets of input-output pairs from solvers or observations. Examples include Fourier Neural Operators (Li et al., 2020) and DeepONet (Lu et al., 2019).

On the other hand, neural PDE solvers directly incorporate physical laws by embedding the governing equations into the learning process. A key example is PINNs (Raissi et al., 2019), which approximate solutions by minimizing the residuals of governing equations, initial conditions, and boundary conditions. Figure 1 illustrates the relationship between data requirements and scientific knowledge across different methods.

While both neural operators and PINNs have strengths and limitations, this study suggests that PINNs are more suitable for data generation tasks in scientific and engineering domains, especially in applications with limited data resources. Neural operators typically require large datasets, often derived from costly simulations, and do not explicitly incorporate governing physics equations, which can lead to generalization issues and physical inconsistencies outside the training data distributions(Négiar et al., 2022). In contrast, PINNs integrate governing equations directly into the training process, ensuring that the solutions adhere to the underlying physics while reducing the reliance on pre-existing datasets, making them particularly effective for data-scarce applications (Négiar et al., 2022).

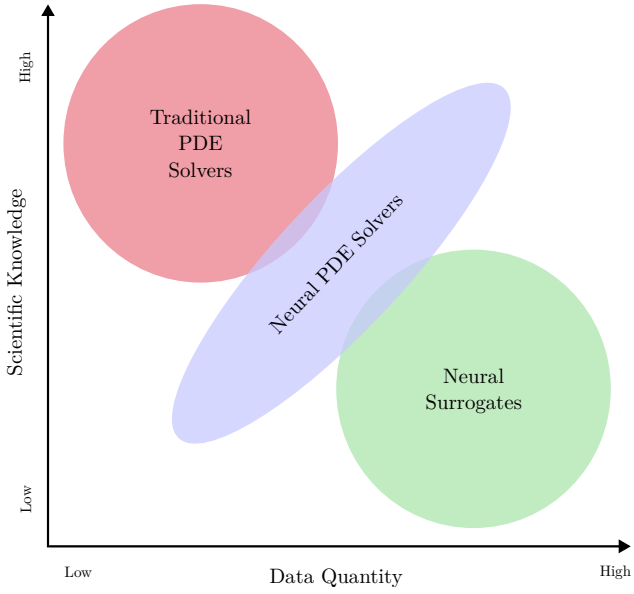

Figure 1: Data and scientific knowledge requirements for different modeling approaches.

Nevertheless, PINNs have some known limitations. They can struggle with convergence, particularly for high-dimensional or complex physics problems Négiar et al. (2022), which results in long training times. The computation of residuals, which requires derivative evaluation through automatic differentiation, becomes computationally expensive for PDEs with higher-order derivatives. Additionally, their convergence is sensitive to hyperparameters. Moreover, PINNs are typically trained on a per-PDE instance basis, meaning they can only solve one specific problem at a time and must be retrained from scratch for each parameter change. These drawbacks limit their adoption in diverse data generation tasks involving different PDEs.

To address these limitations, this survey explores the integration of advanced ML techniques, such as transfer learning and meta-learning into PINNs to enhance model adaptivity by maximizing knowledge reuse, reducing adaptation time and minimizing data resources. Additionally, these methods can help overcome the convergence challenges typically associated with PINNs. This advancement aims for "efficient model adap-

[1]A surrogate model can be thought of as a "regression" to a set of data, where the data is a set of input-output parings obtained by evaluating a black-box model of a complex system Eason & Cremaschi (2014); Caballero & Grossmann (2008). Conversely, a solver is an algorithm or method used to find a solution to a mathematical model.

[2]While some authors use the terms "Neural Surrogates" and "Neural PDE Solvers" interchangeably, this work makes a distinction to highlight the specific requirements for obtaining the solution to a PDE.

tivity," enabling rapid learning in data-limited environments and facilitating their adoption in real-world applications where data is scarce and fast evaluation is essential.

The key contributions of this work include:

- An introductory overview of PINNs, highlighting their connections to traditional numerical methods for solving PDEs and how certain techniques, such as Reduced Order Modeling (ROM), reuse solution data across similar PDE problem instances.
- A review of recent advancements in PINNs, focusing on techniques like transfer learning and meta-learning that enhance model adaptivity and performance.
- Identification of potential metrics and benchmarks for assessing model adaptivity.
- Insights into future research directions and potential applications of adaptive PINNs across various domains.

To the best of our knowledge, there has not been a review paper addressing model adaptivity in PINNs.

The structure of this paper is as follows. First, key concepts and terminology are introduced. Section 3 offers an overview of how transfer learning and meta-learning have been utilized to enhance PINNs' adaptivity. Section 4 discusses various metrics and benchmarking methodologies used to assess the efficient adaptation of the model. Section 5 highlights works that have been applied to real-world applications. Finally, Section 6 outlines future directions and presents conclusions.

2 Background

2.1 Initial Boundary Value Problem

In science and engineering, problems are often framed as Initial Boundary Value Problems (IBVPs), which encompass a wide range of phenomena. An IBVP is typically represented as:

$$\begin{aligned} \mathcal{N}[u(x,t;\mu)] &= f(x) && \forall\ x \in \Omega\ ,\ t \in [t_0, T], \\ \mathcal{B}[u(x,t)] &= g(x) && \forall\ x \in \delta\Omega\ ,\ t \in [t_0, T], \\ \mathcal{I}[u(x,t)] &= h(x) && \forall\ x \in \Omega,\ t = t_0, \end{aligned} \tag{1}$$

where, $\mathcal{N}$ represents the differential operator acting on the function u, which depends on parameters denoted by μ. $f(x)$ represents the source term defined in the domain Ω. The operator $\mathcal{B}$ imposes the boundary conditions $g(x)$ on u at the boundary $\partial\Omega$ of the domain. Lastly, $\mathcal{I}$ sets the initial conditions $h(x)$ for the function $u(x, t_0)$, representing the initial state of u within Ω at the initial time t_0. The differential operator $\mathcal{N}$ can take the form $\mathcal{N}[u(x,t;\mu)] = F(x, u, \frac{\partial u}{\partial t}, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \ldots)$, where F is some given function that describes the dynamics of the system.

The objective in solving an IBVP is to find the function $u(x,t;\mu)$ that satisfies the differential equation, the boundary conditions, and the initial conditions simultaneously.

To solve such IBVP problems numerically, three methods are often used - the Finite Element Method, the Finite Difference Method, and the Finite Volume Method. These methods differ in their mathematical formulation and approach, but all work by discretizing the domain into smaller subdomains (elements, cells, or grid points) and performing local approximations to obtain the global solution.

2.2 Physics-Informed Neural Networks

Physics-informed Neural Networks approximate the solution u using a neural network u_θ and incorporating information from the IBVP directly into the training process. While various methodologies exist, this explanation focuses on one of the most common approaches, which involves utilizing the residual of the IBVP in the loss function Raissi et al. (2019).

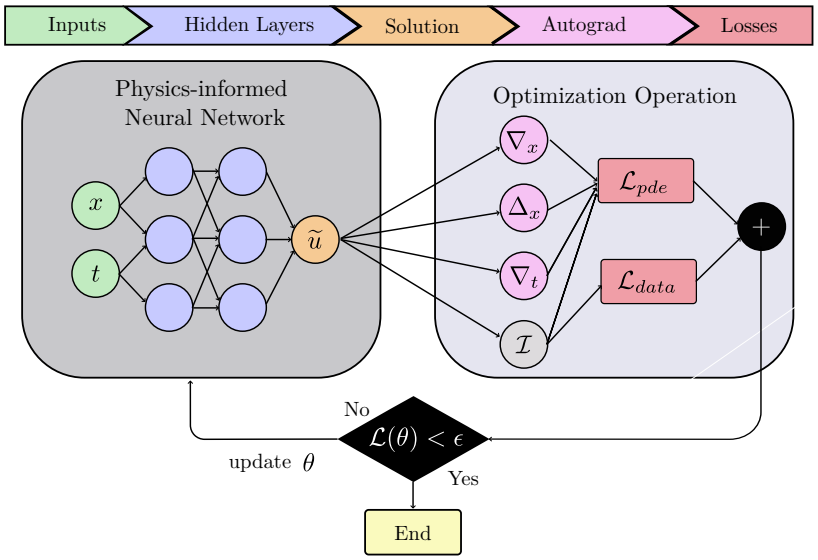

Figure 2: Schematic representation of PINNs. The network combines data and physical laws by minimizing a loss function based on the PDE residual, boundary and initial conditions, and data points. Gradients of the PDE are obtained via automatic differentiation.

Considering the definition of the IBVP (1), the equation can be reformulated in terms of its residuals. Where these residuals are estimated at collocation points which are sampled within the corresponding domain (Ω, $\delta\Omega$ or $[t_0, T]$) in $N_{(\cdot)}$ quantities:

$$\begin{aligned} \mathcal{N}[u_\theta(x,t;\mu)] - f(x) &= r_{pde} \qquad \forall\ x \in \{x_i\}_{i=1}^{N_{pde}},\ t \in \{t_i\}_{i=1}^{N_{pde}}, \\ \mathcal{B}[u_\theta(x,t)] - g(x) &= r_{bc} \qquad \forall\ x \in \{x_i\}_{i=1}^{N_{bc}},\ t \in \{t_i\}_{i=1}^{N_{bc}}, \\ \mathcal{I}[u_\theta(x,t)] - h(x) &= r_{ic} \qquad \forall\ x \in \{x_i\}_{i=1}^{N_{ic}},\ t = t_0. \end{aligned} \tag{2}$$

Here, the derivatives of the differential operator $\mathcal{N}[\cdot]$ are obtained by automatic differentiation.

The loss function is then described as a weighted sum of the corresponding residuals, where the weights act as a regularization term:

$$\mathcal{L}(u_\theta) = w_{pde}\boldsymbol{r_{pde}} + w_{bc}\boldsymbol{r_{bc}} + w_{ic}\boldsymbol{r_{ic}}. \tag{3}$$

Optimizing the weights $\boldsymbol{\theta}$ w.r.t to this loss function leads the solution u_θ to converge toward the true solution u. If partial observation data are available, for example from experiments or sensors, an additional regression loss term can be incorporated into the equation (3). Figure 2 illustrates the key elements involved in the PINN training process.

2.3 Weighted Residuals: Collocation Method

To highlight the similarities between PINNs and numerical methods, this paper compares the method of collocation, a numerical approach to solving PDEs, with the PINN approach. For simplicity, a steady 1-D case will be considered. The residual of an IBVP is formulated as:

$$\mathcal{N}[u(x;\mu)] - f(x) = R(x) \qquad \forall\ x \in \Omega \tag{4}$$

An approximate solution $\widetilde{u}$ is devised in such a form that it is possible to approximate a wide range of functions:

$$\widetilde{u}(x) = \sum_{i=1}^{N} a_i \cdot \phi_i(x), \tag{5}$$

where, given a good choice of basis $\boldsymbol{\phi}$, the task is to find the expansion coefficients $\boldsymbol{a}$ such that the residual is minimized. Since it is a continuous domain and the goal is to minimize the residual throughout this whole domain, the residual is integrated over the domain. Furthermore, to account for the variation of the residual throughout the entire domain, the residual is weighted with $\omega(x)$. Therefore, the weighted residual formulation is as follows:

$$\int_{x_0}^{x_f} \omega(x) \cdot R(x) \; dx = 0 \tag{6}$$

This formulation allows for the minimization of the weighted sum of residuals in the domain, leading to an approximation of the true solution to the PDE.

The collocation method is a special case of the weighted residual (6) formulation, where the weighting function is expressed as a Dirac delta function: $\omega(x) = \delta(x - x_i)$. Using the Dirac delta function enables the direct evaluation of the residual at specific locations. Consequently, by selectively minimizing the residual at these specified locations, a system of equations can be constructed to derive the approximated solution.

The basis functions are chosen according to the characteristics of the problem; these can be either continuous or piecewise. The choice of basis functions $\phi_i(x)$ play a crucial role in determining the accuracy and effectiveness of the approximation. These basis functions should have certain properties that enable them to capture the behavior of the solution within the problem domain accurately.

2.4 Reduced Order Modeling: A numerical approach for reusing information

In the pursuit of efficient model adaptation, it is valuable to explore numerical methods that leverage previously obtained solutions to infer new, similar solutions, thereby reducing computational demands. ROM encompasses a class of numerical techniques that aim to construct a simplified version of the original model by reducing its computational complexity. This is achieved by constructing a low-dimensional approximation that captures the essential behavior of the high-fidelity model or simulation, but with significantly fewer degrees of freedom. The key objective of ROM is to enable efficient exploration and adaptation to new scenarios by leveraging known information from existing simulations, experimental data, or solutions to similar problems.

One such approach is the Galerkin-POD method, which leverages known information from existing solutions (snapshots) to construct reduced bases (POD modes) that capture the essential dynamics of the system across various scenarios. This diverse set of snapshots, obtained from multiple tasks or parameter configurations, encapsulates the shared knowledge and dominant features of the system's behavior. By performing Proper Orthogonal Decomposition (POD) on these snapshots, the method extracts the dominant POD modes, which serve as a compact representation of the solution manifold.

Considering the IBVP problem defined in section 1, the goal of the Galerkin-POD method is to find an approximate solution $\widetilde{u}(x, t; \mu)$ expressed as a linear combination of the extracted POD basis functions $\phi_i(x)$:

$$\widetilde{u}(x, t; \mu) = \sum_{i=0}^{N} a_i(t; \mu) \cdot \phi_i(x), \tag{7}$$

where $a_i(t; \mu)$ are the time-dependent modal coefficients, and N is the number of POD modes retained.

The reduced basis is then used to project the governing equations onto a reduced subspace, yielding a reduced system of equations for the modal coefficients. Consequently, when faced with a new task or scenario, the Galerkin-POD method can efficiently adapt the solution by solving this reduced system. By reducing the dimensionality, the resulting reduced order model becomes computationally less expensive to solve or simulate, while maintaining an acceptable level of accuracy.

2.5 Relationship with PINNs

Both PINNs and the collocation method leverage residual information to guide the approximated solution towards the ground truth solution of the governing equations. However, they differ in their approach to

constructing the solution ansatz. PINNs exploit the universal approximation theorem, using neural networks as the ansatz solution, with the ability to capture local behavior and sharp gradients dependent on the choice of activation function. In contrast, the collocation method builds the solution as a linear combination of linearly independent basis functions and expansion coefficients. Where selection of basis functions, whether global or piece-wise, involves a trade-off between accuracy and computational efficiency . Piece-wise basis functions, with their compact support and ability to capture local behavior, can provide higher accuracy for problems with localized features or complex geometries, but at the cost of increased computational complexity and larger systems of equations. Global basis functions, on the other hand, are generally smoother and require fewer basis functions, leading to smaller systems and simpler implementation, but may struggle to accurately represent sharp gradients or complex geometries.

The POD-Galerkin method takes a different approach by building a global basis that generalizes to several PDE instances, resulting in fewer equations to solve. By projecting the governing equations onto the reduced space spanned by the global basis, the POD-Galerkin method transmits information from other solutions through the basis functions, enabling efficient and accurate approximations for a range of PDE instances.

It is worth noting that the linear combination of basis functions in the collocation and POD-Galerkin methods can potentially increase the expressability of the solution ansatz compared to the universal approximation theorem employed in PINNs. This has led to recent efforts in framing neural networks in a similar linear combination form to enhance their representational capacity (Chen & Koohy, 2024; Desai et al., 2021; Peng et al., 2020; Penwarden et al., 2023; Bischof & Kraus, 2022).

2.6 Efficient Model Adaptivity

Efficient model adaptivity refers to the ability of a machine learning model to quickly and effectively adjust to new, previously unseen tasks using knowledge gained from previous tasks. Given a model pre-trained on one or multiple source tasks $T_s \subset T$, the goal is to adapt this model to an arbitrary novel target task sampled from $t \sim T_t$ or several target tasks $T_t \subset T$ where $T_t \cap T_s = \emptyset$. It is assumed that all tasks from T share some common characteristics. In the context of PINNs, a model is typically trained per task, where each task corresponds to an instance of the IBVP subject to different parameters μ, which can be a material property, boundary condition, or initial condition. Figure 3 illustrates two examples of IBVP. The first example corresponds to the 2D heat equation, where the task-defining parameter is the diffusivity coefficient. The second example corresponds to the Burgers equation, where the task is defined by the initial condition.

Key aspects that affect efficient model adaptivity include computational efficiency and data efficiency. Computational efficiency is influenced by several factors, primarily the number of model parameters, the model complexity and optimization steps required for adaptation. These elements directly impact the overall training time and computational resources needed. Data efficiency, on the other hand, refers to the effectiveness with which a model can learn from limited data samples. This data can encompass various types: collocation points (evaluation points), partial observations, or the pre-training tasks needed for generalization.

To achieve efficient model adaptation, this work suggests the application of transfer learning and meta-learning to physics-informed neural networks.

2.7 Transfer Learning and Parameter-Efficient Fine-Tuning

Transfer learning is a machine learning approach that transfers knowledge gained from a source domain to a different but related target domain. The fundamental principle is to leverage a model pre-trained on a source task or dataset and adapt its learned representations to a new target task. By fine-tuning the pre-trained model on the target data, transfer learning aims to accelerate the learning process and improve generalization performance compared to training from scratch. This technique is particularly beneficial when the target task has limited labeled data.

Parameter-Efficient Fine-Tuning (PEFT) is an advancement in transfer learning that aims to make the fine-tuning process even more efficient and scalable. Unlike traditional fine-tuning, which updates all the parameters of the pre-trained model, PEFT introduces a small number of trainable parameters that modulate

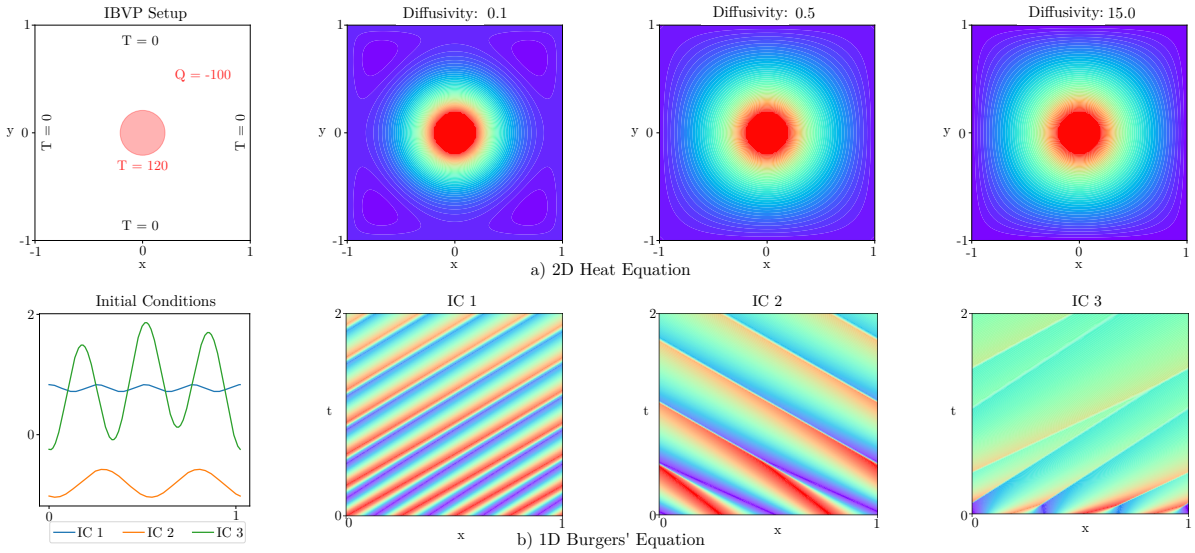

Figure 3: Illustration of IBVP as Tasks: a) Heat equation tasks with varying material properties. b) Burgers' equation tasks with different initial conditions (adapted from Takamoto et al. (2022)).

the pre-trained model's behavior. These additional parameters are trained to adapt the pre-trained model to the target task, while the vast majority of the original model parameters remain frozen.

A Brief History on Transfer Learning

The initial research in transfer learning dates back to the 1970s and 1980s, as described in the work of Bozinovski (2020).
Some years later, Pratt et al. (1991) conducted pioneering studies that explored how a neural network can be recycled, coining this the "transfer problem". Sharkey & Sharkey (1993) also investigated transfer learning, focusing on using prior knowledge to improve the performance of new tasks and understanding when knowledge can be transferred between networks. The work of Sharkley suggests that the concept of transfer learning has strong roots in psychology.
In 1995, the fundamental motivation for transfer learning was discussed in the NIPS-95 workshop on "Learning to Learn" by Baxter et al. (1995), as referenced by Pan & Yang (2009).
The first comprehensive survey on transfer learning was published by Pan. More recently, (Zhuang et al., 2020) provided a comprehensive survey covering more than 40 representative transfer learning approaches from data and model perspectives.

2.8 Meta-learning

Meta-learning is a field in machine learning that encompasses the notion of "learning to learn". The core idea is to leverage an external architecture or algorithm beyond single-task learning models, which can capture the relationships and shared knowledge across different tasks. By learning the correlations between various tasks during a meta-training phase, this external meta-learner can then modify, e.g., the structure of the base learning algorithms, their hyperparameters, and/or the model architectures. This allows the meta-learner to adapt and transfer the acquired meta-knowledge to new, unseen tasks more efficiently, facilitating rapid learning and generalization. This distinguishes itself from traditional transfer learning, which focuses on adapting between a source and target domain. Meta-learning aims to extract task-general knowledge from a distribution of tasks, enabling systematic adaptation to any task within that distribution.

A Brief History on Meta-learning

Some works that have laid the foundation of meta-learning as a concept in machine learning include the work of Schmidhuber (1987), work in which he introduced the term "self-referential learning", a technique where a network receives its own weights as inputs and predicts new updates for such weights[a]. Bengio et al. (1990) hypothesized that it is possible to learn algorithms for synaptic learning rules, and that these rules could be constrained such that the resulting neural networks are capable of solving difficult AI tasks. Hochreiter et al. (2001) then demonstrated how to leverage gradient-based optimizations to automatically discover effective algorithms tailored to specific tasks, such as time-series forecasting.
As meta-learning research progressed, the need for efficient training methods became apparent. Addressing this, Finn et al. (2017) introduced Model-Agnostic Meta-Learning (MAML), a general approach that aimed to provide a good weight initialization, thereby reducing the fine-tuning effort required for new tasks. Taking inspiration from MAML, Nichol et al. (2018) proposed Reptile, a first-order approximation of MAML for weight initialization, avoiding the computationally expensive second-order derivatives required in MAML's bi-level optimization.
These works have set the ground to the current state of meta-learning.

[a](Hospedales et al., 2021)

3 Methods

3.1 Transfer Learning in PINNs

This section reviews advancements in PINNs enhanced by transfer learning techniques, which address the computational cost and convergence issues of training from scratch. The literature discussed is summarized in Table 1.

3.1.1 Full Fine-tuning

Full model fine-tuning (FFT) involves training a model on a source task and then fine-tuning it to a target task by updating all its parameters. This approach uses the weights of the source task as the starting point for the target task, facilitating efficient adaptation.

An example of full model fine-tuning in PINNs is the TL-PINN method introduced by Prantikos et al. (2023), which addresses the Point Kinetic Equation[3]. In this application, real-time performance is crucial. TL-PINN leverages transfer learning by pre-training a PINN on a source task and fine-tuning it on a related target task, thereby accelerating the prediction of reactor transients. During fine-tuning, the entire model is adapted to the target task. The authors found a correlation between the performance improvements of TL-PINN and a similarity measure of the reactor transients, providing guidance on when to apply TL-PINN. If the source and target transients are sufficiently similar, fine-tuning leads to faster convergence and better accuracy compared to training a PINN from scratch. The study underscores the importance of task similarity in determining the effectiveness of transfer learning.

Focusing on the inverse problem, Lin & Chen (2024) introduced TL-gPINNs, an extension of gPINNs, which incorporate the gradient of the residual into the loss function to improve accuracy. While gPINNs provide better predictions, they come with increased computational costs due to the need for gradient calculations. To mitigate this, the authors proposed pre-training a standard PINN and using its weights to initialize the gPINN, resulting in both lower error and reduced computational overhead compared to training gPINNs from scratch.

Zhou & Mei (2023) combined the smoothed finite element method (S-FEM) with PINNs to improve the efficiency of solving inverse problems with limited data. Their approach involves pre-training a PINN using

[3]The Point Kinetic Equation is a simplified model used to analyze nuclear reactor transients.

S-FEM-generated data, then fine-tuning it on a new problem with additional S-FEM data. This transfer learning strategy significantly improves computational efficiency and accuracy, proving at least twice as efficient as directly coupling S-FEM with PINNs, making it a practical solution for inverse problems in engineering with limited data availability.

Table 1: Transfer Learning in Physics-informed Neural Networks.

Transfer Learning Strategies in PINNs					
FS.	**Literature**	**Task**	**PT**	**PType**	**Benchmark Equations**
FFT	Prantikos et al. (2023)	ODE	ST	Fwd.	*Point Kinetic (PKEs)
	Lin & Chen (2024)	PDE	MT	Inv.	Schrödinger^{1D}
	†Zhou & Mei (2023)	PDE	ST	Inv.	*Elastoplastic^{2D}
PEFT	Desai et al. (2021)	PDE/ODE	MT	Fwd.	Pois.^{2D}, Schr.^{1D}, 1st/2nd-order ODEs
	Goswami et al. (2020)	PDE	ST	Fwd.	*Fracture Mechanics^{2D}
	Gao et al. (2022)	PDE	ST	Fwd.	Linear Parabolic^{10D}, Allen Cahn^{10D}
	†Xu et al. (2022)	PDE	MT	Inv.	*Elastic$^{2\text{-}3D}$, Hyperelastic^{2D}
	Pellegrin et al. (2022)	ODE	MT	Fwd.	Stochastic Branched Flow^{2D}
	†Chakraborty (2021)	PDE/ODE	ST	Fwd.	Stochastic ODE, Burgers^{1D}
CTL	†Mustajab et al. (2024)	ODE/PDE	ST	Fwd.	Harmonic Oscillator, Wave Equation^{1D}

Note: Fine-tune Strategy (FS), Pre-train type (PT), Problem Type (PType), Full Fine-tune (FFT), Parameter-efficient Fine-tuning (PEFT), Curriculum Transfer Learning (CTL), Single-task Learning (ST), Multi-task Learning (MT). Equations with (*) are domain-specific problems. References marked with (†) indicate the use of few-shot learning techniques. Abbreviations: Poisson = Pois., Schrödinger = Schr.

3.1.2 Parameter-Efficient Fine-tuning

PEFT reduces computational and memory costs by selectively fine-tuning only a small number of parameters. This approach is examined in relevant studies presented here.

In the context of PINNs, Desai et al. (2021) employs a pre-training and fine-tuning strategy for solving ODEs and PDEs using a PEFT-like approach. During the pre-training phase, the method learns a shared basis, represented by the hidden layers $\boldsymbol{H}$, from multiple source tasks involving ODEs or PDEs. During the fine-tuning phase, the approach focuses on determining the expansion coefficients, represented by the output layers $\boldsymbol{W}_{\text{out}}$, for a new problem instance. If these coefficients can be obtained analytically in closed form, they are calculated by solving a linear system of equations. Otherwise, they are determined through a gradient descent optimization approach, where only the output layers $\boldsymbol{W}_{\text{out}}$ are updated, while the shared basis $\boldsymbol{H}$ remains frozen. The final solution $\boldsymbol{\Psi}_\theta$ is defined as:

$$\boldsymbol{\Psi}_\theta = \boldsymbol{H}\boldsymbol{W}_{\text{out}}. \tag{8}$$

This method allows the shared basis $\boldsymbol{H}$ to be transferred across tasks. The accuracy of the final solution depends on how well the basis spans the solution space for the target problem. By calculating the weight updates $\boldsymbol{W}_{\text{out}}$ in closed form, this approach achieves one-step adaptation. This means that the shared basis $\boldsymbol{H}$ can be effectively adapted to a new problem instance with a single fine-tuning step without the need for iterative training or gradient updates to the shared basis itself.

Goswami et al. (2020) proposed applying transfer learning to PINNs for phase-field fracture modeling. Unlike conventional residual-based PINNs that minimize the PDE residual, the approach minimizes the variational energy of the system. The neural network is modified to exactly satisfy the boundary conditions as hard constraints. This formulation offers two key advantages: 1) imposing boundary conditions is simpler and more robust, and 2) the resulting equations involve lower-order derivatives, making training faster. One major bottleneck associated with applying the proposed PINN for phase-field-based crack propagation

problems is that the model needs to be trained for each displacement step, which can make the training phase computationally expensive. To address this issue, transfer learning is applied. The framework follows two simple steps: first, from the second displacement step onward, only the weights and biases of the last layer are retrained, while the weights and biases of all other layers remain fixed at the previously learned values; second, the last layer from the previous displacement step is used as weight initialization for the current displacement step. This transfer learning approach significantly improves computational efficiency by requiring fewer iterations to achieve convergence and substantially reducing the time required for each iteration.

Chakraborty (2021) proposes a transfer learning approach to approximate high-fidelity models using PINNs. The method begins by training a PINN on a low-fidelity model of a given IBVP task. Then, a transfer learning technique is applied, where only the last one or two layers of the network remain trainable. This pre-trained model is subsequently fine-tuned to approximate a higher-fidelity model using limited high-fidelity observations of the same task. The approach is particularly useful in scenarios where the exact high-fidelity model is not known a priori, allowing for efficient adaptation to different boundary conditions or initial conditions.

Gao et al. (2022) introduces SVD-PINNs, a transfer learning method designed to solve high-dimensional PDEs efficiently. The core idea involves pre-training a PINN, which consists of a three-layer MLP, on an arbitrary PDE task. During fine-tuning for a new PDE, the weight matrix of the middle layer is decomposed using SVD (Singular Value Decomposition). The singular vectors, which capture intrinsic patterns from the source task, are kept frozen, while the singular values and the weights of the initial and final layers are adapted. The authors demonstrate the effectiveness of SVD-PINNs on 10-dimensional linear parabolic equations and Allen-Cahn equations, showing superior performance in terms of relative error and convergence speed compared to vanilla PINNs and other transfer learning methods. A notable advantage of SVD-PINNs is their efficiency when solving multiple related PDEs with identical differential operators but different right-hand side functions. However, the main challenge lies in optimizing the singular values during training. Successful optimization of these values leads to better performance than methods that freeze the first layer, as seen in previous transfer learning approaches for PINNs. Conversely, biased or inaccurate singular values can result in worse outcomes than prior methods.

Pellegrin et al. (2022) introduced a multi-task learning strategy aimed at enhancing training efficiency and performance by leveraging on knowledge from related tasks. Their approach employs a multi-head architecture with two primary phases: pre-training and fine-tuning. Initially, during the pre-training phase, the multi-head model, comprising a shared base neural network along with multiple task-specific heads, is trained concurrently on various related PDE tasks. Through training on these interrelated tasks concurrently, the shared base network learns to extract relevant features and representations that encapsulate the underlying dynamics common across the tasks. In the subsequent fine-tuning phase, the weights of the pre-trained base network are fixed, and a new task-specific head is introduced and fine-tuned for the target transfer learning task. This fine-tuning phase enables the model to adjust the learned shared representations to the specific requirements of the new task, while leveraging the knowledge gained from related tasks during pre-training. By leveraging the shared base network learned from multiple related tasks, the model can potentially converge faster and achieve better performance on the target task than training a PINN from scratch.

Xu et al. (2022) addresses the challenge of inverse analysis in engineering structures, where acquiring data for structural components is often expensive. To address this, the authors sought to improve the training efficiency and accuracy of PINNs for inverse problems through a multi-task transfer learning approach. Their proposed solution involves a two-stage learning process. Initially, in the offline stage, a simplified loading scenario is used to pre-train the PINN model, including both the model weights and task-independent loss balancing weights. Subsequently, in the online stage, the pre-trained model is partially fine-tuned with data from real engineering problems, with only the last two layers are updated. The method was applied to 2D linear and hyperelasticity problems in solid mechanics. Combining layer freezing with inherited multi-task weights from pre-trained models significantly accelerated training convergence.

3.1.3 Curriculum Transfer Learning

Curriculum Transfer Learning involves gradually increasing the complexity of tasks during the transfer process, starting with simpler tasks and progressively moving to more complex ones. An example of this approach applied to PINNs is the work of Mustajab et al. (2024), which employed a curriculum transfer learning strategy for solving high-frequency and multi-scale problems. The proposed methods begins training with low-frequency problems and gradually increasing the frequency while transferring knowledge from lower-frequency tasks. This approach improved the convergence and robustness of PINNs for high-frequency and multi-scale PDEs.

3.2 Meta-learning in PINNs

The integration of meta-learning techniques with physics-informed neural networks has found a promising path towards model adaptivity and generalization. Table 2, shows the taxonomy employed for the study of this techniques in the context with PINNs, which focus on the meta-learning representation ("what is being meta-learned")(Hospedales et al., 2021). In the following section, different studies are introduced following this taxonomy.

Table 2: Meta-learning Strategies in Physics-informed Neural Networks.

Meta-learning Strategies in PINNs				
Type	**Approach**	**Literature**	**PType**	**Equation**
Weight Init.	FFT	Liu et al. (2022)	Both	Pois.$^{1\text{-}2D}$, [Burg., Schr.]^{1D}
		Zhong et al. (2023)	Fwd.	*Plasma Sim.^{1D}
		Penwarden et al. (2023)	Fwd.	[Burg., Heat]^{1D}, [A-C, D-R]^{2D}
		Cheng & Alkhalifah (2024)	Fwd.	Wavefield^{2D}
		Qin et al. (2022)	Fwd.	Burg.^{1D}, [Pois., Hyp.-elast.]^{2D}
	PEFT	Cho et al. (2024)	Fwd.	C-D-R^{1D}, Helm.^{2D}
Net. Struct.	Lay./Neur.	†Chen et al. (2021)	Inv.	A-D-R^{1D}
	Activations	Bischof & Kraus (2022)	Fwd.	Pois.^{2D}
		†Chen & Koohy (2024)	Fwd.	[Burg., K-G, A-C]^{1D}
Input	Sampling Points/Params	†Toloubidokhti et al. (2023)	Fwd.	[Burg., Conv., R-D]^{1D}, Helm.^{2D}
		Tang et al. (2023)	Fwd.	Ellip.$^{2\text{-}10D}$, Nonlinear PDE^{10D}
	Latent Rep.	Huang et al. (2022)	Fwd.	Burg.^{1D}, [Max., Laplace.]^{2D}
		Iwata et al. (2023)	Fwd.	Arbitrary Param. PDE^{1D}
Loss	Param. Loss	Psaros et al. (2022)	Both	[Adv., Burg.]^{1D}, SS R-D^{2D}
	Loss Attention	Song et al. (2024)	Fwd.	Burg.^{1D}, [LDC Flow, Pois.]^{2D}

Note: Problem Type (PType), Forward Problem (Fwd.), Inverse Problem (Inv). Equations marked with (*) represent domain-specific problems. References marked with (†) indicate the use of few-shot learning techniques. Abbreviations: Poisson = Pois., Burgers = Burg., Schrödinger = Schr., Simulation = Sim., Allen-Cahn = A-C, Diffusion-Reaction = D-R, Convection-Diffusion-Reaction = C-D-R, Helmholtz = Helm., Advection-Diffusion-Reaction = A-D-R, Klein-Gordon = K-G, Reaction-Diffusion = R-D, Elliptic = Ellip., Hyper-elasticity = Hyp.-elast., Maxwell = Max., Parametric = Param., Advection = Adv., Steady State = SS, Lid-driven Cavity = LDC.

3.2.1 Learning the Weight Initialization

Weight initialization is crucial for PINNs to achieve fast convergence and high accuracy during training, thereby reducing the large computational costs that limit their adoptation in real-time applications. A majority of the works surveyed here focus on this strategy.

Liu et al. (2022) employs the Reptile algorithm (Nichol et al., 2018) to find a good parameter initialization and compares against other initialization techniques such as Xavier in a unsupervised, supervised, and semi-supervised settings. With the reptile weight initialization the training efficiency and prediction accuracy improved in comparison to the vanilla PINN.

Zhong et al. (2023) and Cheng & Alkhalifah (2024) both adapted the Model-agnostic Meta-learning (MAML) approach by Finn et al. (2017) to PINNs. Their goal was to use a meta-network for weight initialization to reduce the training steps needed for optimizing a new task. However, their implementations and benchmark problems differ. Zhong et al. (2023) applied this technique to plasma simulation, while Cheng & Alkhalifah (2024) focused on seismic simulation, particularly the seismic wave equation. Zhong et al. (2023) found that training performance depends on task difficulty; when the target task significantly differs from the source task, generalization becomes uncertain. Conversely, Cheng & Alkhalifah (2024) showed that their approach achieved superior convergence speed and prediction accuracy compared to vanilla PINNs. However, this improvement relies on an initial meta-training phase to establish robust network parameters. The MAML algorithm, which is used in this phase, requires two gradient computations—one for the inner task and one for meta initialization—making it both memory-intensive and computationally costly.

Qin et al. (2022) compared MAML and LEAP (Flennerhag et al., 2018) weight initialization with PINNs, extending tasks to different geometries and boundary conditions. In their work, they found that MAML outperforms LEAP in accuracy for a given runtime and requires less hyperparameter tuning. However, LEAP's meta-training is faster and less memory-intensive.

Penwarden et al. (2023) introduced a two-step weight prediction method for PINNs. First, optimized weights are collected from pre-trained PINNs from multiple tasks. Then, a secondary model is used to approximate the mapping from task parameters to weights. Several models, including Gaussian Processes, Cubic Spline Interpolation, and Radial Basis Functions, were examined. Despite exploring these prediction models, additional research is needed to tackle high-dimensional parametric domains.

Cho et al. (2024) developed the Hyper-Low-Rank PINN, which combines meta-learning with PEFT to address parametric PDEs more efficiently. The method features a two-phase training process. In the pre-training phase, the weights of the hidden layers of the base model are constructed using a SVD approach, $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$, where $\boldsymbol{\Sigma}$ (the singular values) are provided by the meta-network and the singular vectors $\boldsymbol{U}$ and $\boldsymbol{V}$ are part of the base model. The first and last layers of the base model are kept as standard linear layers. During the fine-tuning phase, the meta-network generates adaptive weights for new tasks and trims less significant weights to maintain a compact, hyper-low-rank structure. Additionally, the first and last layers are optimized along with the adaptive weights.

3.2.2 Learning the Network Structure

Another aspect that has been tackled with meta-learning is the adaptation of the network structure according to a new given task. This category includes features such as the number of layers, hidden dimensions, and activation functions.

Chen et al. (2021) developed a method to solve the mixed problem (forward/inverse) of the advection-diffusion-reaction (ADR) system using sparse measurements. Given the stochastic nature of the problem, they utilized a PINN architecture called sPINN (Zhang et al., 2019). This composite network comprises several sub-networks, each corresponding to different fluid flow frequencies (modes). Due to the complexity of determining the optimal number of layers and neurons for each sub-network, Meta-bayesian optimization was employed to automate the selection process. The study focuses on realistic scenarios with limited or sparse data, addressing both forward and inverse problems common in engineering, where some material properties and sensor measurements are known. Future work will address uncertainty quantification and explore other optimization methods, such as genetic algorithms, greedy methods, or reinforcement learning, to improve NN architecture.

Bischof & Kraus (2022) proposed a method that combines Mixture-of-Experts (MoE) and PINNs to solve a single task. In this approach, multiple expert PINNs are trained on different partitions of the input space, and a gating network learns the optimal weighting of their predictions. This allows the experts to specialize

in different regions of the input space, potentially improving overall accuracy. The meta-learning aspect lies in modulating the gating mechanism to balance the contribution of each expert PINN, thereby enhancing the accuracy and convergence of the overall model. Some key remarks of this work are that when having different experts with different architectures, the gating mechanism discarded the networks with tanh activation and favored sine activation. Another takeaway is that the regularization that weights the importance of each expert is crucial, in this case the optimal solution was achieved with three learners. With this method, the focus was on improving convergence and accuracy. Future research is concerned with increasing performance, efficiency, robustness, and scalability.

Chen & Koohy (2024) introduced GPT-PINN, which combines meta-learning with a task sampling strategy that dynamically expands a shared basis dictionary. This method aims to solve new parameter instances $u(x, t; \mu)$ by approximating them as a weighted sum of pre-trained PINNs:

$$u(\boldsymbol{x}, \boldsymbol{t}; \mu) \approx \sum_{i=1}^{n} c_i(\mu) \Psi_{NN}^{\theta^i}(x, t)$$

Here, $\Psi_{NN}^{\theta^i}$ represents pre-trained PINNs at different parameter configurations, and $c_i(\mu)$ are parameter-dependent coefficients learned by a meta-network. The meta-network modulates the influence of each pre-trained PINN basis $\Psi_{NN}^{\theta^i}$ through the coefficients $c_i(\mu)$ for a given PDE instance. If the approximation fails to meet the accuracy criterion, a new full PINN is trained individually for that parameter instance, and its solution is added to the set of basis functions $\Psi_{NN}^{\theta^i}$, thereby continuously expanding the generalization range of the overall structure.

3.2.3 Learning the Loss Function

Meta-learning techniques are also employed to optimize loss functions for PINNs, further enhancing their performance. Psaros et al. (2022) proposed a gradient-based meta-learning algorithm that discovers effective loss functions for diverse PDE problems. The algorithm operates in two phases: meta-training and meta-testing. During the meta-training phase, a parametric loss function is optimized over a distribution of PDE tasks, learning the optimal parameter values that generalize well across the task distribution. In the subsequent meta-testing phase, the meta-learned loss function is employed to train PINNs on unseen PDE tasks that may differ from the original task distribution and PINN architectures used during meta-training. This meta-learning approach significantly improves the generalization performance of PINNs, enabling them to achieve high accuracy even on out-of-distribution scenarios and PINN architectures that were not encountered during meta-training.

Another recent work that utilizes meta-learning for the loss function is the Loss-Attentional PINN (LA-PINN) by Song et al. (2024). LA-PINN treats the loss function as a learnable component by employing multiple loss-attentional networks (LANs) that are adversarially trained alongside the main PINN model. While the main network minimizes the loss via gradient descent, the LANs use gradient ascent to meta-learn point-wise weights for the loss terms, effectively discovering an "attentional function" to distribute different weights to each collocation point error. This loss-attentional meta-learning framework allows tailoring the loss function per problem by leveraging experience from related tasks, potentially enhancing PINN performance over fixed hand-crafted losses. The adversarial training process, inspired by Generative Adversarial Networks (GANs), enables the LANs to assign higher weights to stiff or hard-to-fit regions, aiding convergence by emphasizing the challenging areas during optimization.

3.2.4 Learning the input

Learning the input involves adapting the input data provided to neural networks. In the context of Physics-Informed Neural Networks, this can refer to selecting and adjusting the number and locations of collocation points. Additionally, during a two-phase training process, another strategy is to adapt the task sampling strategy in the pre-training phase. A third approach involves self-referential learning, where the meta-learner dynamically adapts parts of the input based on the specific task requirements.

Toloubidokhti et al. (2023) indicates that many existing meta-learning strategies neglect the varying difficulty levels across different tasks, and propose that depending on the difficulty, different collocation point positions

and densities should be employed accordingly. To address this, they develop a Difficulty-Aware Task Sampler (DATS) for meta-learning of PINNs, which aims to optimize the task sampling probabilities during meta-training to minimize the average performance across all tasks during meta-validation. DATS employs two strategies: adaptively weighting PINN tasks based on their difficulty, allowing more challenging tasks to contribute more to the meta-learning process; and dynamically allocating the optimal number of residual points (collocation points) across tasks, ensuring that more difficult tasks receive a higher collocation point budget. The evaluation of DATS against uniform and self-paced task-sampling baselines on two meta-PINN models, across four benchmark PDEs, demonstrates that it improves the accuracy of meta-learned PINN solutions and reduces performance disparity among different tasks, while using only a fraction of the residual sampling budget required by the baseline methods.

Another significant contribution is made by Tang et al. (2023), who propose a Deep Adaptive Sampling (DAS) method for solving high-dimensional PDEs using PINNs. The key innovation is treating the residual of the PINN as a probability density function, approximated by a deep generative model called KRnet. The DAS method involves two main components: solving the PDE by minimizing the residual loss on the current set of collocation points, and adaptive sampling where new collocation points are sampled from the KRnet distribution. This method places more points in regions with high residuals, similar to classical adaptive methods like adaptive finite elements. By iteratively refining the training set, the PINN retrains on an updated set with a focus on high-error regions. The KRnet model is meta-learned to approximate the residual distribution, enabling adaptive sampling tailored to the current PINN solution. The approach is particularly effective for low regularity and high-dimensional PDEs, where uniform sampling is inefficient. The authors provide theoretical analysis showing the DAS method can reduce error bounds, with numerical experiments demonstrating significant accuracy improvements compared to uniform sampling.

Self-referential meta-learning approaches have also been combined with PINNs: The Meta-Auto-Decoder (MAD) introduced by Huang et al. (2022) utilizates a self-referential approach to learn parametric PDEs. This approach involves pre-training a neural network $u_\theta(x, z)$ to approximate parametric PDE solutions using a physics-informed loss, where z is a tunable input latent vector that implicitly encodes PDE parameters μ. After pre-training, the network is fine-tuned for new PDEs by either fixing weights θ and tuning z, or tuning both z and θ, allowing it to search for solutions on or near the learned solution manifold.

Instead of learning the latent representation implicitly within the same PINN network, Iwata et al. (2023) proposes to leverage multiple meta-networks to encode the governing equations and boundary conditions into a latent vector z. The fine-tuning strategy involves initializing the latent vector z with the help of the meta-networks, while keeping these meta-networks frozen. The initialized latent vector z is then set to be tunable, and both the PINN weights θ and the latent vector z are fine-tuned for the new PDE task. In both MAD and Iwata et al. (2023), the PINN weights are initialized according to the final pre-training phase.

3.3 Few-shot Learning in PINNs

Few-shot learning aims to minimize the number of training examples required to train a model. In the context of PINNs, this can be achieved through several strategies: minimizing the number of sampling tasks needed for pre-training, reducing collocation points during fine-tuning, and leveraging limited real-world observational data. This section highlights works that focus on achieving few-shot learning through meta-learning and transfer learning, which are indicated with the '†' symbol in Tables 1 and 2.

For example, Chen & Koohy (2024) focuses on reducing pre-training samples by gradually selecting them according to the tasks with the highest residual. Similarly, Toloubidokhti et al. (2023) develops a technique that optimizes the sampling probability of tasks based on their difficulty and dynamically allocates collocation points according to task complexity. This method aims to improve the efficiency and effectiveness of meta-learning for PINNs.

Mustajab et al. (2024) finds that their transfer learning method reduces the number of collocation points needed during the fine-tuning phase, thereby enhancing the efficiency of the training process.

Finally, Chen et al. (2021); Xu et al. (2022); Zhou & Mei (2023) explore leveraging limited real-world observational data for fine-tuning, which is especially valuable for practical applications.

4 Metrics & Benchmarks

4.1 PDE Problems

Table 1 and table 2 summarize the benchmark equations used in each work. In addition, the full description of the most common equations are summarized in Appendix A.1.

4.2 Error Measures

Chen & Koohy (2024) introduces a summary of the choices of error measures, which are common throughout most of the works. These are presented as a reference in Table 3.

Table 3: Evaluation Metrics Commonly Used. Here the worst-case losses and errors are evaluated over a set of tasks, providing insight into the performance of the worst-performing tasks. The term "Terminal" refers to metric evaluated at the last iteration step.

Evaluation Metrics		
Metric	**Description**	**Equation**
Largest Loss	Worst-case	$\max_{\mu} \mathcal{L}(\mathbf{u}_\theta(\cdot))$
Terminal Loss	Task-wise	$\mathcal{L}(\mathbf{u}_\theta(\cdot))$
Largest Rel. L_2 Err.	Worst-case	$\max_{\mu} \|\|\mathbf{u}_\theta - \mathbf{u}_{\text{gt}}\|\|_2 / \|\|\mathbf{u}_{\text{gt}}\|\|_2$
Terminal Rel. L_2 Err.	Task-wise	$\|\|\mathbf{u}_\theta - \mathbf{u}_{\text{gt}}\|\|_2 / \|\|\mathbf{u}_{\text{gt}}\|\|_2$
Terminal Abs. Err.	Task-wise	$\|\|\mathbf{u}_\theta - \mathbf{u}_{\text{gt}}\|\|_1$

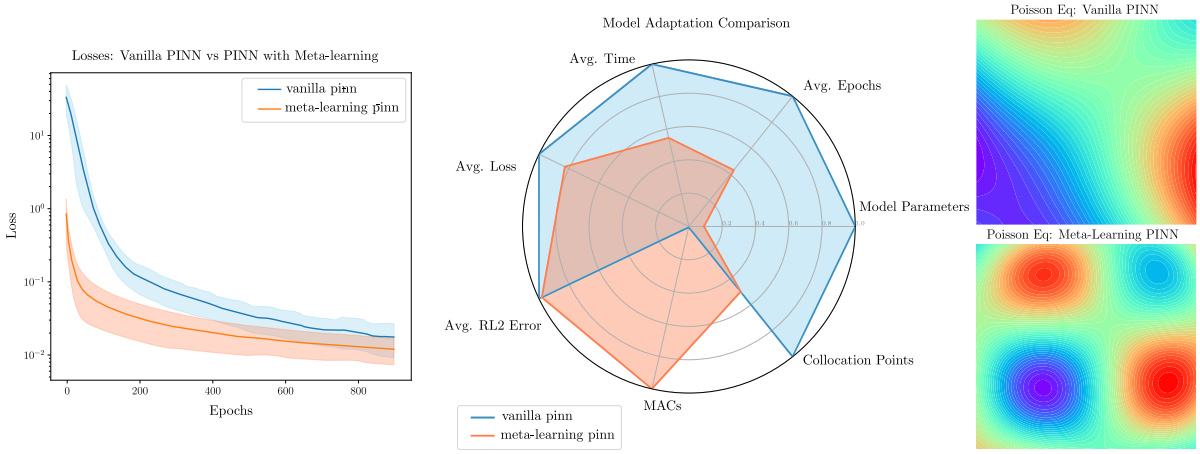

Figure 4: Example of efficient model adaptation through meta-learning.

4.3 Efficient Adaptivity Metrics

Evaluating the adaptivity and efficiency of PINNs is crucial for practical applications. Key metrics assess data requirements and computational efficiency, focusing on minimal data usage and reduced training time. Figure 4 illustrates the key factors affecting efficient adaptation in solving the Poisson equation with varying forcing parameters. It compares a vanilla PINN with a meta-learning approach, emphasizing the reductions in training time, epochs, and collocation points. The meta-learning strategy is based on the method presented by Cho et al. (2024). The solutions shown in the figure correspond to 100 epochs of training for both the vanilla PINN and the meta-learning PINN. The values presented in the radar chart are the normalized averaged values required to reach a loss of 0.05 under a budget of 1200 epochs; if the budget is exceeded without reaching the target loss, then the final values are reported.

4.3.1 Data Efficiency

Collocation Point Budget. The number of collocation points significantly impacts the convergence speed and accuracy of PINNs. Some studies use a fixed number of collocation points, while others employ adaptive sampling strategies, allocating more points to regions with higher PDE losses (Toloubidokhti et al., 2023). A useful evaluation strategy is to measure the accuracy of different sampling techniques given varying collocation point budgets (Wu et al., 2023). By examining the trade-off between the number of collocation points and the resulting accuracy, it is possible to optimize the sampling strategy to suit specific problems and computational constraints, enhancing the training performance and accuracy of PINNs.

Regression Point Budget. In many applications, only a limited number of observation points are available for analysis. Rather than relying solely on unsupervised loss, it is important to make the most of these scarce observations. To address this, a metric is designed to assess the relationship between the number of observation points and the accuracy of the analysis. Specifically, it evaluates the accuracy that can be achieved given a fixed budget of N observation points. This is particularly crucial for real-world scenarios where data availability is constrained.

Task Sampling. Several PINN techniques employ a two-phase training process consisting of a pre-training phase and a fine-tuning phase. The pre-training phase can be conducted using either a single instance or multiple instances of a PDEs. To achieve better generalization across a distribution of tasks, it is crucial to pre-train on multiple tasks distributed along a given parameter range. However, determining the optimal sampling strategy for selecting pre-training tasks is a challenging task. The goal is to identify the fewest number of pre-training tasks that yield the best results, as the number of pre-training tasks influences both the pre-training resources and the final fine-tuning accuracy. Recent works, such as in the work of Toloubidokhti et al. (2023), have attempted to address this problem. One approach is to measure the "performance disparity" within a given range of tasks, defined as the performance difference between the worst-performing and the best-performing PINN. If a network architecture generalizes well across the range of tasks, both the accuracy and the performance disparity should be low. This analysis can also serve as a tool to assess which PDE parameters an architecture struggles with, providing valuable insights for further improvements.

4.3.2 Computational Efficiency

To evaluate computational efficiency, four key metrics are commonly reported. First, the parameter count provides a measure of the model size, which impacts memory usage. Second, the number of MACs (Multiply-Accumulate Operations) directly reflects the computational complexity, influencing processing speed. Third, the epoch count assesses convergence by either reporting the final accuracy within a set epoch budget or the number of epochs required to reach a target error threshold. Fourth, training time offers a direct quantification of computational cost by measuring the duration needed to achieve the desired accuracy. These metrics collectively provide a comprehensive view of the computational demands associated with different PINN architectures and training strategies, facilitating informed decisions for their deployment in resource-constrained environments.

5 Applications

Meta-learning and transfer learning techniques have extended the application of PINNs from traditional benchmarks to practical and domain-specific problems, as indicated by the studies marked with (*) in Tables 1 and 2. These approaches are further discussed in the following sections.

5.0.1 Forward Problems

Prantikos et al. (2023) applied transfer-learning techniques to accelerate PINN retraining time for nuclear reactor safety applications. This approach enables near real-time prediction of reactor transient states by solving PDEs. In brittle fracture mechanics, Goswami et al. (2020) employed transfer learning to predict crack paths in structures. For each displacement increment Δu, transfer learning is used to enhance computational efficiency, avoiding the need to retrain a PINN for each step. Zhong et al. (2023) applied meta-learning for

weight initialization in plasma physics. While traditional methods require complex meshing, PINNs offer a meshless alternative. This approach addresses challenges in classical plasma modeling, such as irregular geometries and complex multiphysics, which often lead to poor convergence or divergence in simulations.

5.0.2 Inverse Problems

Zhou & Mei (2023) focused on using PINNs and transfer learning to solve inverse problems in engineering applications, particularly for elasto-plastic problems. Their work aimed to find new material parameters through inverse solutions, addressing the high computational cost associated with traditional numerical methods. Xu et al. (2022) employed transfer learning to predict external loads on diverse engineering structures based on limited displacement monitoring points. Their approach pre-trains PINNs on simplified loading scenarios, then fine-tunes them using limited displacement observations to predict loads on various structures, including thick cylinders, plates, 3D beams, and tunnels.

These applications demonstrate the versatility and potential of meta-learning and transfer learning techniques in enhancing PINNs for both forward and inverse problems across various engineering and scientific domains.

6 Discussion and Conclusion

The application of transfer learning and meta-learning techniques in PINNs has demonstrated significant potential for achieving efficient adaptivity. These techniques offer promising solutions for both forward and inverse problems, extending beyond traditional benchmarks to practical and domain-specific applications.

However, comparing these techniques directly remains challenging due to varying benchmarking methods. For instance, Qin et al. (2022) and Penwarden et al. (2023) reported conflicting results regarding the effectiveness of MAML compared to random initialization. To better compare different methods and understand their suitability, establishing consistent evaluation criteria and clarifying terms such as "in-distribution" versus "out-of-distribution" tasks is crucial.

Future research should focus on adapting models to out-of-distribution conditions, potentially through few-shot learning and theoretical convergence analysis. Additionally, methods like those proposed by Song et al. (2024), which enhance residual utilization, show promise for improving model adaptivity. Furthermore, Exploring faster approaches for gradient calculation in residuals could further enhance efficiency.

In conclusion, the effort to make PINNs more adaptive by reusing learned information could offer significant advantage over traditional numerical solvers. Learning to exploit this characteristic could lead to the development of more efficient methods for solving PDEs or enhancing traditional numerical approaches. This study suggests incorporating meta-learning, and transfer learning to facilitate knowledge reuse within PINNs and achieve efficient model adaptivity. While PINNs may incur high initial training costs, their adaptivity becomes particularly beneficial when solving similar PDEs repeatedly, such as in parameter identification and design optimization. This approach may prove more efficient compared to solving each PDE independently.

References

Simon Arridge, Peter Maass, Ozan Öktem, and Carola-Bibiane Schönlieb. Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174, 2019.

Hubert Baty. A hands-on introduction to physics-informed neural networks for solving partial differential equations with benchmark tests taken from astrophysics and plasma physics. *arXiv preprint arXiv:2403.00599*, 2024.

Jonathan Baxter, Rich Caruana, Tom Mitchell, Lorien Y Pratt, Daniel L Silver, and Sebastian Thrun. Learning to learn: Knowledge consolidation and transfer in inductive systems. In *NIPS Workshop, http://plato. acadiau. ca/courses/comp/dsilver/NIPS95_LTL/transfer. workshop*, 1995.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Citeseer, 1990.

Rafael Bischof and Michael A Kraus. Mixture-of-experts-ensemble meta-learning for physics-informed neural networks. In *Proceedings of 33. Forum Bauinformatik*, 2022.

Stevo Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica (Slovenia)*, 44(3), 2020. doi: 10.31449/INF.V44I3.2828. URL https://doi.org/10.31449/inf.v44i3.2828.

José A Caballero and Ignacio E Grossmann. An algorithm for the use of surrogate models in modular flowsheet optimization. *AIChE journal*, 54(10):2633–2650, 2008.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

Souvik Chakraborty. Transfer learning based multi-fidelity physics informed deep neural network. *Journal of Computational Physics*, 426:109942, 2021.

Xiaoli Chen, Jinqiao Duan, and George Em Karniadakis. Learning and meta-learning of stochastic advection–diffusion–reaction systems from sparse measurements. *European Journal of Applied Mathematics*, 32(3): 397–420, 2021.

Yanlai Chen and Shawn Koohy. Gpt-pinn: Generative pre-trained physics-informed neural networks toward non-intrusive meta-learning of parametric pdes. *Finite Elements in Analysis and Design*, 228:104047, 2024.

Shijun Cheng and Tariq Alkhalifah. Meta-pinn: Meta learning for improved neural network wavefield solutions. *arXiv preprint arXiv:2401.11502*, 2024.

Woojin Cho, Kookjin Lee, Donsub Rim, and Noseong Park. Hypernetwork-based meta-learning for low-rank physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

Shaan Desai, Marios Mattheakis, Hayden Joy, Pavlos Protopapas, and Stephen Roberts. One-shot transfer learning of physics-informed neural networks. *arXiv preprint arXiv:2110.11286*, 2021.

John Eason and Selen Cremaschi. Adaptive sequential sampling for surrogate model generation with artificial neural networks. *Computers & Chemical Engineering*, 68:220–232, 2014.

Salah A Faroughi, Nikhil Pawar, Celio Fernandes, Maziar Raissi, Subasish Das, Nima K. Kalantari, and Seyed Kourosh Mahjour. Physics-guided, physics-informed, and physics-encoded neural networks in scientific computing, 2023. URL https://arxiv.org/abs/2211.07377.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Sebastian Flennerhag, Pablo G Moreno, Neil D Lawrence, and Andreas Damianou. Transferring knowledge across learning processes. *arXiv preprint arXiv:1812.01054*, 2018.

Yihang Gao, Ka Chun Cheung, and Michael K Ng. Svd-pinns: Transfer learning of physics-informed neural networks via singular value decomposition. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1443–1450. IEEE, 2022.

Somdatta Goswami, Cosmin Anitescu, Souvik Chakraborty, and Timon Rabczuk. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, 106:102447, 2020.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *Artificial Neural Networks—ICANN 2001: International Conference Vienna, Austria, August 21–25, 2001 Proceedings 11*, pp. 87–94. Springer, 2001.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

Xiang Huang, Zhanhong Ye, Hongsheng Liu, Shi Ji, Zidong Wang, Kang Yang, Yang Li, Min Wang, Haotian Chu, Fan Yu, et al. Meta-auto-decoder for solving parametric partial differential equations. *Advances in Neural Information Processing Systems*, 35:23426–23438, 2022.

Tomoharu Iwata, Yusuke Tanaka, and Naonori Ueda. Meta-learning of physics-informed neural networks for efficiently solving newly given pdes. *arXiv preprint arXiv:2310.13270*, 2023.

Zichao Jiang, Junyang Jiang, Qinghe Yao, and Gengchao Yang. A neural network-based pde solving algorithm with high precision. *Scientific Reports*, 13(1):4479, 2023.

Yang Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric partial differential equations using the neural convolution. *SIAM Journal on Scientific Computing*, 43(3):A1697–A1719, 2021.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Shuning Lin and Yong Chen. Gradient-enhanced physics-informed neural networks based on transfer learning for inverse problems of the variable coefficient differential equations. *Physica D: Nonlinear Phenomena*, 459:134023, 2024.

Xu Liu, Xiaoya Zhang, Wei Peng, Weien Zhou, and Wen Yao. A novel meta-learning initialization method for physics-informed neural networks. *Neural Computing and Applications*, 34(17):14511–14534, 2022.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Abdul Hannan Mustajab, Hao Lyu, Zarghaam Rizvi, and Frank Wuttke. Physics-informed neural networks for high-frequency and multi-scale problems using transfer learning. *Applied Sciences*, 14(8):3204, 2024.

Geoffrey Négiar, Michael W Mahoney, and Aditi S Krishnapriyan. Learning differentiable solvers for systems with hard constraints. *arXiv preprint arXiv:2207.08675*, 2022.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

Raphaël Pellegrin, Blake Bullwinkel, Marios Mattheakis, and Pavlos Protopapas. Transfer learning with physics-informed neural networks for efficient simulation of branched flows. *arXiv preprint arXiv:2211.00214*, 2022.

Wei Peng, Weien Zhou, Jun Zhang, and Wen Yao. Accelerating physics-informed neural network training with prior dictionaries. 2020.

Michael Penwarden, Shandian Zhe, Akil Narayan, and Robert M Kirby. A metalearning approach for physics-informed neural networks (pinns): Application to parameterized pdes. *Journal of Computational Physics*, 477:111912, 2023.

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.

Lorien Y Pratt, Jack Mostow, and Candace A Kamm. Direct transfer of learned information among neural networks. In *Proceedings of the ninth National conference on Artificial intelligence-Volume 2*, pp. 584–589, 1991.

Apostolos F Psaros, Kenji Kawaguchi, and George Em Karniadakis. Meta-learning pinn loss functions. *Journal of computational physics*, 458:111121, 2022.

Tian Qin, Alex Beatson, Deniz Oktay, Nick McGreivy, and Ryan P Adams. Meta-pde: Learning to solve pdes quickly without a mesh. *arXiv preprint arXiv:2211.01604*, 2022.

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

Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Noel E Sharkey and Amanda JC Sharkey. Adaptive generalisation. *Artificial Intelligence Review*, 7(5): 313–328, 1993.

Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

Yanjie Song, He Wang, He Yang, Maria Luisa Taccari, and Xiaohui Chen. Loss-attentional physics-informed neural networks. *Journal of Computational Physics*, 501:112781, 2024.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

Kejun Tang, Xiaoliang Wan, and Chao Yang. Das-pinns: A deep adaptive sampling method for solving high-dimensional partial differential equations. *Journal of Computational Physics*, 476:111868, 2023.

Maryam Toloubidokhti, Yubo Ye, Ryan Missel, Xiajun Jiang, Nilesh Kumar, Ruby Shrestha, and Linwei Wang. Dats: Difficulty-aware task sampler for meta-learning physics-informed neural networks. In *The Twelfth International Conference on Learning Representations*, 2023.

Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.

Chen Xu, Ba Trung Cao, Yong Yuan, and Günther Meschke. Transfer learning based physics-informed neural networks for solving inverse problems in tunneling. *arXiv e-prints*, pp. arXiv–2205, 2022.

Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, November 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2019.07.048. URL `http://dx.doi.org/10.1016/j.jcp.2019.07.048`.

Linlin Zhong, Bingyu Wu, and Yifan Wang. Accelerating physics-informed neural network based 1d arc simulation by meta learning. *Journal of Physics D: Applied Physics*, 56(7):074006, 2023.

Meijun Zhou and Gang Mei. Transfer learning-based coupling of smoothed finite element method and physics-informed neural network for solving elastoplastic inverse problems. *Mathematics*, 11(11):2529, 2023.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

A Appendix

A.1 Benchmark Equations

A.1.1 Poisson equation

The Poisson equation is a second-order elliptic PDE appearing in many fields, such as electrostatics, steady heat transfer, and many others. This equation has the following form:

$$\begin{aligned} -\Delta u(\boldsymbol{x}) &= f(\boldsymbol{x}), \quad && \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) &= 0, && \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{9}$$

where $\Delta(\cdot)$ is the Laplace operator. A common feature among all works is that the domain is 2D, specifically $\Omega \subseteq [-1,1] \times [-1,1]$, except for Bischof & Kraus (2022), which uses an L-shaped domain. The forcing or source term $f(\boldsymbol{x})$ changes among works:

Table 4: Different forms of $f(x,y)$ used in various studies. Desai et al. (2021) employs a different forcing term during testing. In the work of Liu et al. (2022), n represents the number of heat sources, and $\mathcal{U}$ denotes uniform sampling.

Literature	$f(x,y)$	**Parameters**
Desai et al. (2021)	$\sin(k\pi x)\sin(k\pi y)$	$k \in \{1,2,3,4\}$
Liu et al. (2022)	$\sum_{i=1}^{n} c_i \cdot \exp\left(-\frac{(x-a_i)^2+(y+b_i)^2}{0.01}\right)$	$a_i, b_i \sim \mathcal{U}(0.1, 0.9)$, $c_i \sim \mathcal{U}(0.8, 1.2)$
Bischof & Kraus (2022)	1	-
Song et al. (2024)	$2\pi^2 \sin(\pi x)\sin(\pi y)$	-

A.1.2 Burgers' Equation

Burgers' equation is a time-dependent PDE that models a system consisting of a moving viscous fluid. The 1D form of the equation models the fluid flow through an ideal thin pipe. The Burgers' equation is given by:

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} &= 0, && x \in \Omega, \quad t \in [0,T], \\ u(x,t) &= 0, && x \in \partial\Omega, \quad t \in [0,T], \\ u(x,0) &= u_0(x), && x \in \Omega, \end{aligned} \tag{10}$$

The unknown $u(x,t)$ is the speed of the fluid, and ν the fluid viscosity. When the viscosity is low, then the fluid flow develops a shock wave.

Most of the works presented here treat ν as the parameter that defines a task, the initial condition as $u_0(x) = -\sin(\pi x)$, and the computational domain as $\Omega \in [-1,1]; t \in [0,1]$. The table below shows the different choices of ν across various works.

A.1.3 Allen-Cahn Equation

The Allen-Cahn equation is given by:

$$\begin{aligned} \frac{\partial u}{\partial t} - \lambda\Delta u + \epsilon(u^3 - u) &= f(x,t), && x \in \Omega, \quad t \in [0,T], \\ u(x,t) &= 1, && x \in \partial\Omega, \quad t \in [0,T], \\ u(x,0) &= x^2\cos(\pi x), && x \in \Omega, \end{aligned} \tag{11}$$

Table 5: Different choice of $\nu for Burgers' Equation 1D$ used in various studies.

Literature	Parameters
Liu et al. (2022)	$\nu \in [0, 0.1/\pi]$
Penwarden et al. (2023)	$\nu \in [0.005, 0.05]$
Chen & Koohy (2024)	$\nu \in [0.005, 1/\pi]$
Toloubidokhti et al. (2023)	$\nu \in [0.001, 0.1]$
Chen & Koohy (2024)	$\nu \in [0.005, 1/\pi]$
Psaros et al. (2022)	$\nu \in [0.001, 0.002]$ & $\nu \in [0.01, 1.0]$
Song et al. (2024)	$\nu = 0.01/\pi$

where in $\Omega = [-1, 1]$ represents the spatial domain, and $T = 1$ denotes the final time. The coefficient λ is chosen from the interval $[0.0001, 0.001]$, while the parameter ϵ, which controls the strength of the nonlinear term, is selected from the range $[1, 5]$. The forcing term f(x,t) is set to zero for this study, focusing on the intrinsic dynamics of the Allen-Cahn equation. This represents an initial-boundary value problem (IBVP) as per Chen & Koohy (2024). An alternative formulation of the IBVP exists in other works that derive a forcing term based on an exact solution, such as Penwarden et al. (2023) and Xu et al. (2022).

A.1.4 Wave equation

The wave equation for a scalar wave function $u(x, t)$ is given by:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u, \tag{12}$$

where c is the wave speed and ∇^2 is the Laplacian operator in three dimensions.

A.1.5 Helmholtz Equation

The Helmholtz equation for a scalar field $u(\mathbf{r})$ is given by:

$$\Delta u + k^2 u = 0, \tag{13}$$

where k is the wave number related to the wavelength λ.

A.1.6 Schrödinger Equation

The time-dependent Schrödinger equation for a single particle in three-dimensional space is given by:

$$i\hbar \frac{\partial \Psi(\mathbf{r}, t)}{\partial t} = -\frac{\hbar^2}{2m} \nabla^2 \Psi(\mathbf{r}, t) + V(\mathbf{r}) \Psi(\mathbf{r}, t), \tag{14}$$

where $\Psi(\mathbf{r}, t)$ is the wave function, $\mathbf{r} = (x, y, z)$ are the spatial coordinates, t is time, $\hbar$ is the reduced Planck's constant, m is the mass of the particle, ∇^2 is the Laplacian operator, and $V(\mathbf{r})$ is the potential energy function.

A.1.7 Advection-Reaction-Diffusion

Advection-reaction-diffusion equations, as considered in this section, are known to be stiff problems when the advection term dominates over the diffusion one. In such cases, sharp transition layers appear in the solution, which are difficult to capture by traditional numerical schemes." Baty (2024) The advection-reaction-diffusion equation is given by:

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u = D \nabla^2 u + R(u), \tag{15}$$

where $u = u(\mathbf{r}, t)$ is the dependent variable (scalar field), t is time, $\mathbf{r} = (x, y, z)$ represents spatial coordinates, $\mathbf{v} = (v_x, v_y, v_z)$ is the velocity field (advection term), D is the diffusion coefficient, ∇^2 is the Laplacian operator, and $R(u)$ is the reaction term.

A.1.8 Lid-Cavity Driven Flow

The lid-driven cavity flow equations are:

$$\begin{aligned} &\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \\ &u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\nabla^2 u + F_x, \\ &u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\nabla^2 v + F_y, \end{aligned} \tag{16}$$

with boundary conditions:

$$\begin{aligned} &u(x, 0) = 0, \quad u(x, 1) = 1 \quad \text{(lid)}, \\ &v(0, y) = v(1, y) = 0 \quad \text{(walls)}, \\ &u(x, y) = v(x, 0) = v(x, 1) = 0 \quad \text{(other boundaries)}. \end{aligned}$$

Here, $u(x, y)$ and $v(x, y)$ are the velocity components, $p(x, y)$ is the pressure, ρ is the fluid density, ν is the kinematic viscosity, and F_x, F_y are additional body forces.