Survey: Adaptive Physics-informed Neural Networks

Edgar Torres[1] **Jonathan Schiefer**[2] **Mathias Niepert**[1]

[1]Institute for Artificial Intelligence, University of Stuttgart
[2]Robert Bosch GmbH
Emails: `{edgar.torres, mathias.niepert}@ki.uni-stuttgart.de`

Abstract

Physics-informed neural networks (PINNs) present a promising solution for solving partial differential equations (PDEs) using neural networks, especially in data-scarce scenarios due to their unsupervised learning abilities. However, a key limitation is the need for re-optimization with every change in PDE parameters, similar to the constraints in traditional numerical methods, which limits the broader use of PINNs. This survey explores research addressing this limitation through transfer learning and meta-learning. These methods can potentially improve PINNs' training efficiency, enabling quicker adaptation to new PDEs with fewer data and computational demands. Instead of relying on extensive data to build general models, typical for existing foundation model approaches, efficient adaptation in PINNs focuses on smaller information domains that quickly adjust to similar problems by leveraging previously learned knowledge. By synthesizing insights from these advanced learning techniques, this survey identifies strategies to facilitate the broader adoption of PINNs across scientific and engineering fields.

1 Introduction

Advances in machine learning (ML) have led to foundation models with applications in fields like computer vision (e.g., self-driving cars), natural language processing (e.g., intelligent agents, chat-bots), and image generation. Building on this success, there is growing interest in the developing foundation models to solve problems in science and engineering. However, unlike domains with abundant data, these fields often face data limitations due to the high cost of experiments and simulations needed to generate data. To address this, data- and computationally-efficient methods such as transfer learning, meta-learning, and few-shot learning—successful in other domains—show promise for advancing ML in scientific disciplines.

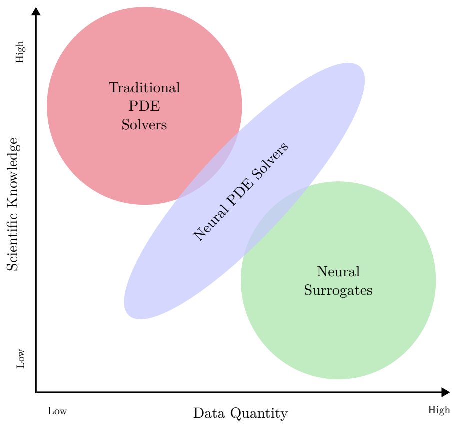

Figure 1: Data and scientific knowledge requirements for different modeling approaches.

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 [1]. 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 [2, 3, 4, 5], presenting the potential for faster evaluation compared to classical iterative solvers, as well as offering mesh-free solutions not constrained to computational grids [6, 7, 2, 5]. Moreover, ML 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 [8, 9]. In addition, ML implementations are simpler than numerical methods allowing for faster development and easy maintenance [9].

ML methods for solving PDEs can be broadly categorized into neural surrogates and neural PDE solvers [1][2]. Neural surrogates, including physics-guided neural networks and neural operators, train networks 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 [7] and DeepONet [12].

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 [2], which approximate solutions by minimizing the residuals of governing equations, initial conditions, and boundary conditions.

While both neural operators and PINNs have strengths and limitations, this study suggests that PINNs are more suitable for data generation in scientific and engineering domains with limited data. 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 [13]. 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 preexisting datasets, making them particularly effective for data-scarce applications [13]. Figure 1 shows how different methods balance data requirements with scientific knowledge. When the underlying governing equations are not incorporated, more data is needed to infer the physical behavior. However, when the governing equations are utilized, only minimal data is required, such as boundary conditions or material properties.

However, PINNs also face certain challenges, including convergence issues in high-dimensional problems, long training times due to the computational expense of derivative evaluations, and sensitivity to hyperparameters. Additionally, PINNs are typically trained on a per-PDE instance basis, requiring retraining for each parameter change [13].

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. Moreover, this integration offers a promising alternative to traditional foundation models, providing a more data-efficient approach to training models. This advancement aims to achieve "efficient model adaptivity," enabling rapid learning and adaptation in data-limited environments. Ultimately, this can help overcome some of PINNs' current challenges and facilitate their adoption in real-world applications where data is limited and fast evaluation are crucial.

The key contributions of this work include:

- Reviewing recent advancements in PINNs, with a focus on adaptive techniques such as transfer learning, meta-learning, and few-shot learning.
- Identifying potential benchmarks and real-world applications where these adaptive techniques can significantly improve PINNs efficiency, especially in data-limited scenarios.
- Providing insights into future research directions and use cases for adaptive PINNs across various domains.

[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 [10, 11]. 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.

To the best of our knowledge, no prior review has focused specifically on model adaptivity in PINNs.

The structure of this paper is as follows: First, essential concepts are introduced. Section 3 reviews the application of transfer learning and meta-learning to PINNs. Section 4 covers benchmarks and metrics for assessing model adaptation. In Section 5, works applied to real-world scenarios are discussed. Finally, Section 6 suggests future research directions and a concluding remark.

2 Background

Physics-Informed Neural Networks (PINNs). PINNs approximate the solution of PDEs by using a neural network u_θ that incorporates information from the initial-boundary value problem (IBVP)[3] directly into the optimization process. A common approach involves including the residuals of the IBVP equations in the loss function, where these residuals are calculated at collocation points sampled within the problem's domain. Specifically, the residuals correspond to the discrepancies in the PDE, boundary conditions, and initial conditions at these sampled points. Automatic differentiation is employed to compute the necessary derivatives in the differential operator. This method allows the network to learn solutions that satisfy the IBVP constraints accurately.

Efficient Model Adaptivity. Efficient model adaptivity in ML is the ability to quickly adjust to new, unseen tasks using prior knowledge. Given a model pre-trained on source tasks $T_s \subset T$, the goal is to adapt it to novel target tasks $T_t \subset T$, where $T_t \cap T_s = \emptyset$, assuming shared characteristics across tasks. In PINNs, each task typically corresponds to an IBVP instance with different parameters (e.g., material properties, boundary/initial conditions). Figure 2 illustrates two examples of IBVPs.

Efficient adaptivity is influenced by two main factors: training performance and data efficiency. Training performance is affected by the number of model parameters, the complexity of the model, and the optimization steps required for adaptation. Data efficiency, on the other hand, seeks to reduce the number of collocation points or make the best use of limited observational data. To achieve efficient model adaptation, this work suggests the application of transfer learning, meta-learning to physics-informed neural networks.

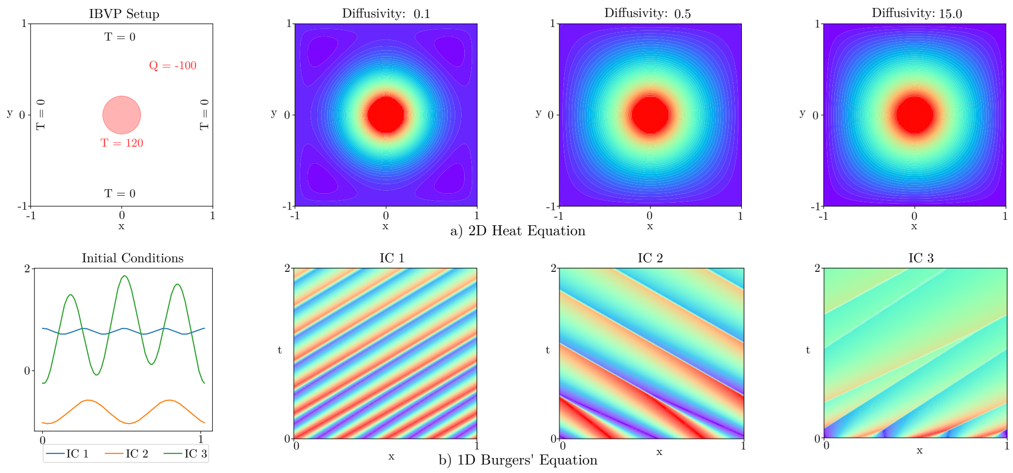

Figure 2: a) Heat equation with different material properties as a task. b) Burgers' equation with different initial conditions as a task (adapted from [14]).

Transfer Learning. Transfer learning involves adapting a model pre-trained on a source task to a related target task. By fine-tuning this model on new, often limited data, transfer learning accelerates learning and enhances performance compared to training from scratch. This approach is especially useful when the target domain has scarce labeled data.

[3]An IBVP is a mathematical framework that combines a partial differential equation with initial and boundary conditions to yield a unique solution that describes the system's behavior.

Meta-learning. Meta-learning, or 'learning to learn', aims to enhance both model training efficiency and overall performance by leveraging shared knowledge across multiple tasks. During a meta-training phase, a meta-learner captures task relationships and adjusts base learning algorithms, hyperparameters, or architectures. This enables faster and more efficient adaptation to new tasks.

Few-shot Learning. Few-shot learning is a technique designed to enable models to learn from few examples. It involves transferring knowledge from previously learned tasks to new, similar tasks with minimal data. Few-shot learning often leverages meta-learning to generalize from limited examples, making it valuable for scenarios with constrained data availability. By efficiently adapting to new tasks with minimal training data, few-shot learning enhances model performance and practical applicability.

3 Methodology

3.1 Transfer Learning Techniques 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.

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. [15]	ODE	ST	Fwd.	*Point Kinetic (PKEs)
	Lin and Chen [16]	PDE	MT	Inv.	Schrödinger^{1D}
	Zhou and Mei [17]	PDE	ST	Inv.	*Elastoplastic^{2D}
PEFT	Desai et al. [18]	PDE/ODE	MT	Fwd.	Pois.^{2D}, Schr.^{1D}, 1st/2nd-order ODEs
	Goswami et al. [19]	PDE	ST	Fwd.	*Fracture Mechanics^{2D}
	Gao et al. [20]	PDE	ST	Fwd.	Linear Parabolic^{10D}, Allen Cahn^{10D}
	Xu et al. [21]	PDE	MT	Inv.	*Elastic$^{2\text{-}3D}$, Hyperelastic^{2D}
	Pellegrin et al. [22]	ODE	MT	Fwd.	Stochastic Branched Flow^{2D}
	†Chakraborty [23]	PDE/ODE	ST	Fwd.	Stochastic ODE, Burgers^{1D}
CTL	†Mustajab et al. [24]	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.

3.1.1 Full Fine-tuning

Full model fine-tuning (FFT) updates all parameters of a pre-trained model for a new task. For example, Prantikos et al. [15] introduced TL-PINN for solving the Point Kinetic Equation[4], where a pre-trained PINN is fine-tuned to accelerate predictions by leveraging task similarity. They demonstrated that performance improvement correlates with task similarity, leading to faster convergence and better accuracy. A key takeaway is the importance of measuring task similarity to assess when transfer learning is beneficial. Similarly, Lin and Chen [16] enhanced gPINNs[5] by using transfer learning to initialize gPINN models with pre-trained weights from a standard PINN, enabling efficient training.

[4]Point Kinetics Equations are a simplified model for analyzing nuclear reactor dynamics.

[5]Gradient-enhanced PINNs [25] include the residual's gradient in the loss function, improving accuracy but increasing training costs.

Zhou and Mei [17] 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 S-FEM-generated data, then using this pre-trained model to initialize weights for a new problem and fine-tuning it with additional S-FEM data. This method demonstrated enhanced accuracy and efficiency compared to directly coupling S-FEM with PINNs.

3.1.2 Parameter-Efficient Fine-tuning (PEFT)

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.

Desai et al. [18] applied a PEFT approach for solving ordinary differential equations (ODEs) and PDEs within PINNs. In this method, the hidden layers represent a shared basis learned during pre-training, while the output layers are task-specific. For new tasks, only the output weights are updated, while the shared basis remains fixed. In some cases, these output weights can be calculated analytically by solving a linear system, enabling a one-step adaptation. Otherwise, they are optimized through gradient descent. This approach achieves efficient task transfer by leveraging the pre-trained shared basis across tasks.

Goswami et al. [19] applied a similar strategy for phase-field fracture modeling, a problem where displacements are calculated in small steps. A full PINN is trained for the first step, and in subsequent steps, only the last layer is retrained while sharing pre-trained weights, improving efficiency.

Chakraborty [23] fine-tuned a low-fidelity PINN model to approximate a high-fidelity model by adjusting only the last one or two layers. A data-driven loss was employed to guide this fine-tuning process, effectively achieving a high-fidelity approximation with a limited amount of high-fidelity data.

Gao et al. [20] introduced SVD-PINNs, which apply singular value decomposition (SVD) to the hidden layers, fine-tuning only the singular values and the weights of the initial and last layers. This approach improves efficiency in solving high-dimensional PDEs, especially when dealing with multiple related PDEs that share differential operators but differ in right-hand side functions. Optimizing singular values is crucial for performance; proper optimization stabilizes training and generally outperforms training PINNs from scratch, while inaccurate values can worsen outcomes.

Pellegrin et al. [22] employed a multi-task learning strategy, training a shared base network on multiple related tasks and fine-tuning task-specific heads to improve convergence and performance.

Xu et al. [21] addressed inverse analysis in engineering structures with a two-stage transfer learning process. First, a PINN is pre-trained on simplified tasks using multi-task learning. Then, it is fine-tuned on real data, updating only specific layers. Applied to 2D elasticity problems, this method improved accuracy and accelerated convergence, even with simplified pre-training.

3.1.3 Curriculum Transfer Learning (CTL)

Curriculum Transfer Learning (CTL) gradually increases task complexity. Mustajab et al. [24] applied this strategy to PINNs, starting with simpler problems and progressively introducing more complex ones, which improved convergence for high-frequency and multi-scale PDEs.

3.2 Meta-Learning Techniques in PINNs

The integrating of meta-learning with PINNs enhances model adaptivity and generalization. Table 2 outlines the taxonomy of these techniques in the context of PINNs, focusing on what is meta-learned. The next section presents studies based on this taxonomy.

3.2.1 Learning the Weight Initialization

Effective weight initialization plays a critical role in accelerating PINN convergence and accuracy while reducing computational costs. Several meta-learning approaches have been developed to address this challenge. For instance, Liu et al. [26] applied the Reptile Algorithm [27] for weight initialization, demonstrating improved training efficiency and accuracy compared to methods like Xavier across various settings. Another approach, Model-Agnostic Meta-Learning (MAML) [28], was adapted by Zhong et al. [29] and Cheng and Alkhalifah [30] for PINNs in plasma and seismic simulations. Although MAML significantly enhances convergence and accuracy, it requires substantial computational resources due to its two-gradient computation process.

In comparing MAML with LEAP [31], Qin et al. [32] found that while MAML outperforms LEAP in accuracy for a given runtime, LEAP offers faster meta-training and reduced memory requirements.

Penwarden et al. [33] 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. [34] 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 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

Meta-learning also plays a significant role in adapting the network structure to improve performance in various applications. For example, Meta-Bayesian Optimization, as applied by Chen et al. [35], was used to select the optimal network architecture for solving Advection-Diffusion-Reaction (ADR) systems with sparse data.

Similarly, Mixture-of-Experts, utilized by Bischof and Kraus [36], combined with PINNs, leverages a gating network to balance contributions from different experts across various input regions. This method improved both accuracy and convergence, with ongoing research aimed at refining performance and scalability.

In another advancement, Chen and Koohy [37] introduced GPT-PINN, which integrates meta-learning with dynamic task sampling to expand a shared basis dictionary. It approximates new parameter instances $u(x, t; \mu)$ as a weighted sum of pre-trained PINNs:

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

where $\Psi_{NN}^{\theta^i}$ are pre-trained PINNs for various parameters, and $c_i(\mu)$ are coefficients from a meta-network. If the approximation is inaccurate, a new PINN is trained for that parameter and added to the basis, enhancing the model's generalization.

3.2.3 Learning the Loss Function

Meta-learning techniques are also employed to optimize loss functions for PINNs, further enhancing their performance. For instance, Psaros et al. [38] developed a gradient-based meta-learning approach that discovers optimal loss functions across various PDE tasks by designing a parametric loss function optimized during the pre-training phase. This approach improves generalization, allowing the model to perform better on new, unseen tasks.

Another significant development is the Loss-Attentional PINN (LA-PINN), introduced by Song et al. [39]. This approach treats the loss function as a learnable component, using multiple loss-attentional networks (LANs) trained adversarially with the main PINN. While the PINN minimizes the loss via gradient descent, the LANs employ gradient ascent to learn point-wise weights, effectively creating an "attentional function" that dynamically adjusts weights based on the difficulty of fitting each collocation point. Inspired by GANs, this method emphasizes challenging regions of the problem, improving convergence and overall accuracy.

3.2.4 Learning the Input

Adapting input data, including collocation points, plays a key role in reducing computational demands and improving the performance of PINNs. One such method is the Difficulty-Aware Task Sampler (DATS), introduced by Toloubidokhti et al. [40], which optimizes task sampling based on the difficulty of the tasks, leading to improved accuracy and reduced performance disparity.

Similarly, Deep Adaptive Sampling (DAS), proposed by Tang et al. [41], employs a generative model to guide adaptive sampling, particularly in regions with high residuals. This approach has shown significant improvements in accuracy, especially for high-dimensional PDEs.

Finally, Self-Referential Learning approaches have been explored by Huang et al. [42] and Iwata et al. [43]. Huang's method, MAD, approximates PDE solutions using a latent vector, while Iwata's approach leverages meta-networks to encode PDE parameters into a latent vector, refining it for new tasks, further enhancing the adaptability and efficiency of PINNs.

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. [26]	Both	Pois.$^{1\text{-}2D}$, [Burg., Schr.]^{1D}
		Zhong et al. [29]	Fwd.	*Plasma Sim.^{1D}
		Penwarden et al. [33]	Fwd.	[Burg., Heat]^{1D}, [A-C, D-R]^{2D}
		Cheng and Alkhalifah [30]	Fwd.	Wavefield^{2D}
		Qin et al. [32]	Fwd.	Burg.^{1D}, [Pois., Hyp.-elast.]^{2D}
	PEFT	Cho et al. [34]	Fwd.	C-D-R^{1D}, Helm.^{2D}
Net. Struct.	Lay./Neur.	†Chen et al. [35]	Inv.	A-D-R^{1D}
	Activations	Bischof and Kraus [36]	Fwd.	Pois.^{2D}
		†Chen and Koohy [37]	Fwd.	[Burg., K-G, A-C]^{1D}
Input	Sampling Points/Params	†Toloubidokhti et al. [40]	Fwd.	[Burg., Conv., R-D]^{1D}, Helm.^{2D}
		†Tang et al. [41]	Fwd.	Ellip.$^{2\text{-}10D}$, Nonlinear PDE^{10D}
	Latent Rep.	Huang et al. [42]	Fwd.	Burg.^{1D}, [Max., Laplace.]^{2D}
		Iwata et al. [43]	Fwd.	Arbitrary Param. PDE^{1D}
Loss	Param. Loss	Psaros et al. [38]	Both	[Adv., Burg.]^{1D}, SS R-D^{2D}
	Loss Attention	Song et al. [39]	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.

3.3 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 and Koohy [37] and Toloubidokhti et al. [40] focus on reducing sampling tasks, while Mustajab et al. [24], Toloubidokhti et al. [40], and Tang et al. [41] address reducing collocation points. Additionally, Chen et al. [35] explores the use of limited real-world observational data for fine-tuning, which is particularly valuable for practical applications.

4 Benchmarks and Metrics

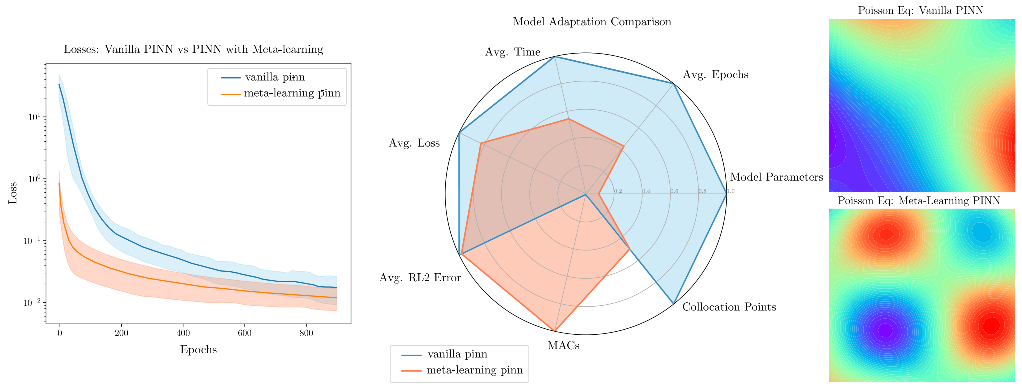

Figure 3: Example of efficient model adaptation through meta-learning.

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 and Koohy [37] 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: Common Evaluation Metrics for error measurement. Here the worst-case losses and errors are evaluated over a set of tasks $\boldsymbol{\mu}$, 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	$\max_{\mu} \|\|\mathbf{u}_\theta - \mathbf{u}_{\text{gt}}\|\|_{\mathbf{2}} / \|\|\mathbf{u}_{\text{gt}}\|\|_{\mathbf{2}}$
Terminal Rel. L_2 Err.	Task-wise	$\|\|\mathbf{u}_\theta - \mathbf{u}_{\text{gt}}\|\|_{\mathbf{2}} / \|\|\mathbf{u}_{\text{gt}}\|\|_{\mathbf{2}}$
Terminal Abs. Err.	Task-wise	$\|\|\mathbf{u}_\theta - \mathbf{u}_{\text{gt}}\|\|_{\mathbf{1}}$

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 3 illustrates the key factors influencing efficient adaptation when solving the Poisson equation with varying forcing parameters. It compares a standard PINN with a meta-learning approach, highlighting reductions in training time, epochs, and collocation points. The meta-learning strategy follows the method proposed by Cho et al. [34]. The solutions on the right side of the figure correspond to 100 epochs of training for both the vanilla PINN and the meta-learning PINN. The radar chart compares the two models, with all values normalized relative to the highest value for each metric. Two sets of metrics are displayed: one describing the model and the other describing training performance. The training performance metrics are averaged across tasks and reflect the values needed to reach a loss of 0.05 within a budget of 1200 epochs. If the target loss is not achieved within this budget, the final values are reported.

4.3.1 Data Efficiency (Few-shot)

Collocation Point Budget. This metric evaluates the trade-off between the number of collocation points and accuracy. It compares fixed and adaptive sampling strategies to determine the optimal balance for training performance and accuracy.

Regression Point Budget. Measures accuracy based on the number of available observation points, crucial for applications with limited data.

Task Sampling. Involves pre-training on multiple PDE instances to improve generalization. The goal is to determine the optimal number of pre-training tasks for best results, balancing resource use and final accuracy. Performance disparity, or the difference between the best and worst results, is used to gauge generalization and identify areas for improvement.

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 expanded PINN applications beyond traditional benchmarks to practical and domain-specific problems, as noted in Tables 1 and 2. These are discussed in the following sections.

5.0.1 Forward Problems

In various applications, transfer learning and meta-learning have been employed to enhance the performance of PINNs. For example, in nuclear reactor safety, Prantikos et al. [15] used transfer learning to accelerate PINN retraining, enabling real-time reactor state prediction. Similarly, in brittle fracture mechanics, Goswami et al. [19] applied transfer learning to efficiently predict crack paths in structures, eliminating the need to retrain the model for each displacement step. In plasma physics, Zhong et al. [29] utilized meta-learning for weight initialization, significantly improving convergence in complex plasma simulations.

5.0.2 Inverse Problems

In elasto-plastic problems, Zhou and Mei [17] employed PINNs and transfer learning to solve inverse problems and determine material parameters, significantly reducing the computational cost of traditional methods. Similarly, in load prediction, Xu et al. [21] used transfer learning to predict external loads on structures based on limited displacement observations, enhancing efficiency in diverse engineering scenarios. These examples highlight the potential of meta-learning and transfer learning to improve PINNs across various scientific and engineering fields.

6 Discussion & Conclusions

6.1 Future Direction

While transfer and meta-learning techniques have the potential to enhance PINN training efficiency and data reusability, the lack of consistent benchmarking methods makes direct comparisons of these techniques challenging. For instance, Qin et al. [32] and Penwarden et al. [33] reported conflicting findings regarding MAML's effectiveness compared to random initialization. To facilitate better comparisons and improve understanding of these methods' suitability, it is essential to establish consistent evaluation criteria. Additionally, clarifying terms such as "in-distribution" versus "out-of-distribution" tasks will help standardize how future research evaluates transfer learning and meta-learning approaches in PINNs.

A key priority for future research is to enhance the adaptability of PINNs to out-of-distribution conditions. One major challenge in this area is aligning the learned bases with these out-of-distribution tasks. Currently, many methods derive their bases specifically from the training tasks, which limits their ability to generalize to distant tasks [34, 20, 18]. Developing strategies to extend and adapt these bases for tasks outside the original training distribution will significantly improve the generalization capabilities of PINNs, making them more adaptable across a wider range of problems.

Another challenge lies in the convergence behavior when using different loss functions. Training with regression losses often leads to smoother, faster convergence, whereas gradient-based losses (such as PDE residual losses) can result in slower or less stable convergence, despite better enforcing physical constraints. This highlights a potential pitfall in how gradient information is utilized. Further research is needed to address these convergence difficulties and better integrate gradient information, building on techniques like those proposed by Song et al. [39] and Yu et al. [25] (gPINNs).

Finally, investigating how concepts from neural fields can be applied to transfer learning and meta-learning in PINNs presents a promising avenue for improving efficient model adaptation to new tasks or domains. Exploring more complex problems—such as those with varying boundary conditions or higher-dimensional domains beyond 2D—is crucial. Establishing strong baseline problems in these areas will also be a valuable direction for impactful future research.

6.2 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, transfer learning to facilitate knowledge reuse within PINNs. 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.

Acknowledgments and Disclosure of Funding

The authors acknowledge the support of the Stuttgart Center for Simulation Science (SimTech).

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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)*, pages 1443–1450. IEEE, 2022.

[21] 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*, pages arXiv–2205, 2022.

[22] 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.

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

[24] 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.

[25] Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

[26] 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.

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

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

[29] 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.

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

[31] Sebastian Flennerhag, Pablo G. Moreno, Neil D. Lawrence, and Andreas Damianou. Transferring knowledge across learning processes, 2019. URL `https://arxiv.org/abs/1812.01054`.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

[37] 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.

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

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:

$$\begin{aligned} -\Delta u(\boldsymbol{x}) &= f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) &= 0, \qquad\quad \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{1}$$

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 [36], 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. [18] employs a different forcing term during testing. In the work of Liu et al. [26], n represents the number of heat sources, and $\mathcal{U}$ denotes uniform sampling.

Literature	$f(x,y)$	**Parameters**
Desai et al. [18]	$\sin(k\pi x)\sin(k\pi y)$	$k \in \{1,2,3,4\}$
Liu et al. [26]	$\sum_{i=1}^{n} c_i \cdot \exp\left(-\frac{(x-a_i)^2+(y+b_i)^2}{0.01}\right)$	$a_i, b_i \sim \mathcal{U}(0.1, 0.9)$, $c_i \sim \mathcal{U}(0.8, 1.2)$
Bischof and Kraus [36]	1	-
Song et al. [39]	$2\pi^2 \sin(\pi x)\sin(\pi y)$	-

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:

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

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.

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

Literature	Parameters
Liu et al. [26]	$\nu \in [0, 0.1/\pi]$
Penwarden et al. [33]	$\nu \in [0.005, 0.05]$
Chen and Koohy [37]	$\nu \in [0.005, 1/\pi]$
Toloubidokhti et al. [40]	$\nu \in [0.001, 0.1]$
Chen and Koohy [37]	$\nu \in [0.005, 1/\pi]$
Psaros et al. [38]	$\nu \in [0.001, 0.002]$ & $\nu \in [0.01, 1.0]$
Song et al. [39]	$\nu = 0.01/\pi$

A.1.3 Allen-Cahn Equation

The Allen-Cahn equation is given by:

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

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 IBVP as per Chen and Koohy [37]. An alternative formulation of the IBVP exists in other works that derive a forcing term based on an exact solution, such as [33] and [21].

A.1.4 Wave equation

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

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u, \tag{4}$$

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:

$$\Delta u + k^2 u = 0, \tag{5}$$

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:

$$i\hbar \frac{\partial \Psi(\mathbf{r}, t)}{\partial t} = -\frac{\hbar^2}{2m} \nabla^2 \Psi(\mathbf{r}, t) + V(\mathbf{r})\Psi(\mathbf{r}, t), \tag{6}$$

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." [44] The advection-reaction-diffusion equation is given by:

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u = D\nabla^2 u + R(u), \tag{7}$$

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:

$$\begin{aligned}
&\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \\
&u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\nabla^2 u + F_x, \\
&u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\nabla^2 v + F_y,
\end{aligned} \tag{8}$$

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.