SCIENCE CHINA
Physics, Mechanics & Astronomy

• Invited Review •
Special Topic: AI for Mechanics

January 2026 Vol. 69 No. 1: 214602
https://doi.org/10.1007/s11433-024-2665-5

Physics-informed neural networks (PINNs) as intelligent computing technique for solving partial differential equations: Limitation and future prospects

Weiwei Zhang[1,2,3*], Wei Suo[1,2,3], Jiahao Song[1,2,3], and Wenbo Cao[1,2,3]

[1] *School of Aeronautic, Northwestern Polytechnical University, Xi'an 710072, China*
[2] *International Joint Institute of Artificial Intelligence on Fluid Mechanics, Northwestern Polytechnical University, Xi'an 710072, China*
[3] *National Key Laboratory of Aircraft Configuration Design, Xi'an 710072, China*

Received November 28, 2024; accepted April 11, 2025; published online September 1, 2025

In recent years, physics-informed neural networks (PINNs) have become a representative method for solving partial differential equations (PDEs) with neural networks. PINNs provide a novel approach to solving PDEs through optimization algorithms, offering a unified framework for solving both forward and inverse problems. However, some limitations in terms of solution accuracy and generality have also been revealed. This paper systematically summarizes the limitations of PINNs and identifies three root causes for their failure in solving PDEs: (1) poor multiscale approximation ability and ill-conditioning caused by PDE losses; (2) insufficient exploration of convergence and error analysis, resulting in weak mathematical rigor; (3) inadequate integration of physical information, causing mismatch between residuals and iteration errors. By focusing on addressing these limitations in PINNs, we outline the future directions and prospects for the intelligent computing of PDEs: (1) analysis of ill-conditioning in PINNs and mitigation strategies; (2) improvements to PINNs by enforcing temporal causality; (3) empowering PINNs with classical numerical methods.

physics-informed neural networks, partial differential equations, limitations, intelligent computing, deep learning

Citation: W. Zhang, W. Suo, J. Song, and W. Cao, Physics-informed neural networks (PINNs) as intelligent computing technique for solving partial differential equations: Limitation and future prospects, Sci. China-Phys. Mech. Astron. **69**, 214602 (2026), https://doi.org/10.1007/s11433-024-2665-5

1 Introduction

Partial differential equations (PDEs) play an increasingly important role in modern science, revealing the mathematical laws underlying physical systems. They are crucial in analysis, prediction, and control across many fields [1]. Due to the difficulty in analytically solving most PDEs, researchers have proposed various numerical methods for solving PDEs in the past few decades, including finite difference method [2], finite volume method [3], finite element method (FEM) [4], and spectral method [5]. These methods have been widely applied in many physical and engineering fields. However, these numerical methods also have some limitations, including relying on meshes [6,7], facing the curse of dimensionality when solving high-dimensional problems [8,9], design difficulties, and weak robustness of high-order schemes [10,11].

In recent years, the rapid development of deep learning has made it possible to solve PDEs by artificial intelligence. In 2019, Raissi et al. [12] proposed physics-informed neural networks (PINNs) for solving forward and inverse problems involving PDEs. The central idea of this method is to encode PDEs into the loss function of the neural network using automatic differentiation, minimizing the PDEs residuals

*Corresponding author (email: aeroelastic@nwpu.edu.cn)

© Science China Press 2025

while the network approximates the initial and boundary conditions (IBCs) and observed data. In fact, the concept of PINNs can be traced back to some research [13-15] on solving PDEs based on neural networks in the 1990s. In addition to PINNs, some frameworks have also adopted the idea of incorporating PDE constraints into the loss function of neural networks, such as deep Galerkin method [16], Deep Ritz method [17], and theory-guided neural networks [18]. Since the introduction of PINNs, researchers have made significant efforts in algorithm improvements and practical applications. The growth of the number of PINNs-related papers with the topic "PINNs" in the Web of Science database is shown in Figure 1. PINNs have been demonstrated for various phenomena, including fluid mechanics [19-23], heat transfer [24-27], fluid-structure interaction [28-30], electromagnetic propagation [31,32], and geosciences [33,34].

PINNs approximate solutions to PDEs by optimizing neural network parameters, which is notably different from classical numerical methods that solve PDEs through numerical discretization and iterative methods. The fundamental idea behind classical numerical methods is to approximate PDE solutions over continuous space by piecewise continuous polynomials in discretized space. This typically involves domain discretization, equation discretization, and solving systems of algebraic equations. In contrast, vanilla PINNs use differentiable continuous neural networks to approximate the solution over continuous space. The general procedure for solving PDEs with PINNs includes neural network architecture design, weighted loss function design, and parameter optimization of the neural networks. A comparison of PINNs and classical numerical methods is presented in Table 1 [35-37], which highlights the main advantage of PINNs: they leverage the flexibility and adaptability of neural networks in handling optimization problems, making it easier to integrate data (e.g., experimental observations or optimization objectives) and IBCs.

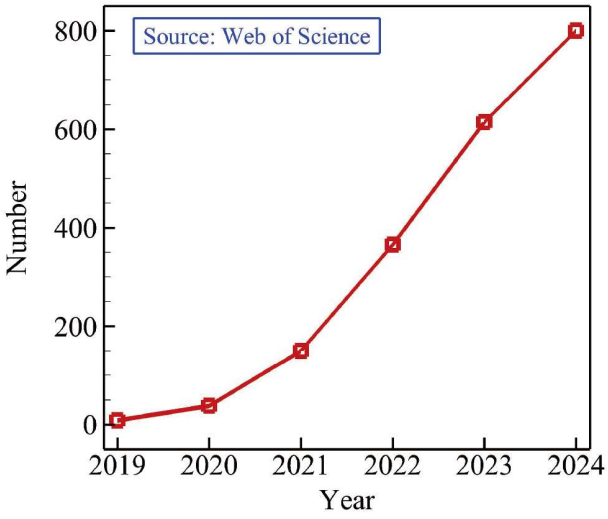


Figure 1 (Color online) Number of PINNs-related papers published from 2019 to 2024.

This facilitates a unified framework for solving forward and inverse problems. Additionally, the well-developed code ecosystem for neural networks simplifies the implementation and maintenance of PINNs. By using scattered points to discretize the solution domain, PINNs also reduce the high discretization cost associated with complex geometries. A promising application based on the advantages of PINNs is flow visualization technology [19,38,39], also known as data assimilation [40-43], where the flow field or other unknown parameters can be inferred from sparse observations (e.g., concentration fields and images). This type of inverse problem is challenging for classical PDE numerical solvers. Recent studies [44-47] have also demonstrated the advantages of PINNs in inverse design and optimal control, where solving PDEs and obtaining optimal designs are carried out simultaneously. In contrast, the direct adjoint looping approach requires hundreds of repeated solutions to the forward problem, resulting in prohibitively high computation

Table 1 Comparison of PINNs and classical numerical methods

Characteristics	PINNs	Classical numerical methods
Combination of data and algorithms	Easy to implement, requiring only additional constraint terms in the loss function	Difficult to implement
Inverse problem-computation cost	Unified framework for forward and inverse problems, no additional code required	Framework requires additional extensions, resulting in high computation cost
Difficulty in discretizing complex geometries	Only requires generating scattered points, no explicit mesh generation needed	Complex geometries have high mesh requirements, relying on mesh generation techniques
Code implementation and maintenance	Easy to implement	Difficult to implement
Forward problem-solving approach	Optimization-based, theory still developing	Iteration-based, well-established theory
Forward problem-computation accuracy	Can only guarantee local optimum, with accuracy limitations	Accuracy progressively improves with increased grid numbers or polynomial order within elements, supported by well-established approximation theory
Generality and robustness	IBCs as soft constraints, highly empirical, weak generality	Robust method with strong generality

costs for PDE-constrained optimization problems. PINNs-like methods have also shown significant results in addressing parametric problems [16,48,49].

Although the neural network-based optimization approach for solving PDEs gives PINNs high flexibility, it also brings certain challenges and limitations. As shown in Table 1, compared to the well-established iterative approach in classical numerical methods, PINNs solve PDEs under given IBCs through optimization. Due to the incomplete development of optimization theory, PINNs can only guarantee locally optimal solutions, which limit their accuracy. Studies indicate that the loss functions in PINNs are high-dimensional non-convex functions, making it difficult to solve with existing optimization methods. As a result, PINNs are less accurate and efficient than high-order computational fluid dynamics (CFD) schemes in forward problems, and cannot replace current CFD methods [36]. Additionally, the local optimum issue also means that vanilla PINNs cannot strictly enforce IBCs. During the weighted design of the loss function for IBCs and PDEs, PINNs are highly empirical and lack strong generality.

Accurately identifying the limitations of PINNs and proposing feasible improvements are crucial issues for the advancement of PINNs. Based on a review of relevant literature summarizing the shortcomings of PINNs, this paper further explores the root causes of these limitations through detailed mathematical analysis and case studies. Building on this, several recommendations are proposed to improve PINNs. Finally, the paper discusses future directions and visions for intelligent methods in solving PDEs.

2 Discussion on the limitations of PINNs

Currently, with the growing popularity and in-depth research on PINNs, the existing shortcomings of PINNs have been summarized in related studies, such as difficulties in loss function weight design and low solving efficiency. Based on these reviews, this paper summarizes three main limitations of PINNs: (1) low solution accuracy and high computation cost; (2) poor generality; (3) numerous empirical parameters and strong parameter sensitivity. The following sections provide a detailed discussion of these three limitations.

2.1 Low solution accuracy and high computation cost

Jagtap et al. [50] identified two main limitations in solving PDEs with PINNs. The first is solution accuracy: due to the local optimum resulting from solving high-dimensional non-convex optimization problems, the absolute error of PINNs solutions is difficult to reduce below the 10^{-5} level. The second is computation cost: optimizing the parameters of deep neural networks with calculating the loss of PDEs through automatic differentiation incurs high training costs. As mentioned by McGreivy and Hakim [51], it is widely recognized that PINN-based solvers are less efficient than standard numerical methods for solving forward problems. Obtaining an accurate PDE solution with PINNs typically requires hours to days in most cases, whereas standard numerical methods take only seconds to minutes. Furthermore, following ref. [51], we randomly selected 59 articles on solving fluid-related PDEs using the PINN method. Our goal was to analyze these articles to determine whether the accuracy of PINNs was mentioned and whether it was compared to numerical methods, as well as whether the computational cost of PINNs was mentioned and whether it was compared to numerical methods. Our statistical analysis indicates that although most of the selected articles mentioned the accuracy and computational cost of PINNs (100% of the selected articles mentioned accuracy, and nearly 50% mentioned computational cost), only a small number of articles provided comparative data on the accuracy and computational cost of PINNs versus numerical methods (1.7% compared accuracy, and 8.5% compared computational cost). Additionally, 18.6% of the articles included qualitative assessments of the accuracy and computational cost of PINNs relative to numerical methods. Overall, these comparisons consistently presented a negative evaluation of PINNs compared to numerical methods. More detailed comparisons can be found in Appendix.

Grossmann et al. [52] compared the computation time and accuracy of the FEM and vanilla PINNs in different types of PDEs, including elliptic, parabolic, and hyperbolic equations. The computation time refers to the time it takes for FEM and PINNs to solve the PDEs on given grids and points, while the accuracy is measured by the L_2 relative error. For example, in the case of elliptic equations, Figure 2 shows the curve of computation time (s) versus error, using one-dimensional, two-dimensional, and three-dimensional Poisson equations for comparison. It can be observed that in terms of computation time, FEM has an advantage of 1-3 orders of magnitude over PINNs, with this advantage being more significant in lower-dimensional PDEs. As the dimensionality of the PDEs increases, the computation time advantage diminishes somewhat because the number of discrete elements required for FEM grows exponentially with the dimension of the PDEs, facing the curse of dimensionality. In contrast, PINNs can represent functions in higher-dimensional spaces without a significant increase in the number of parameters. However, overall, due to the extensive parameter updates required during neural network training, the computation time of PINNs is still much higher than that of FEM. In terms of computation accuracy, for the given maximum grid size, the highest accuracy of FEM in the one-dimensional Poisson equation is two orders of magnitude higher than that of PINNs. Moreover, similar to computation time,

as the dimensionality of the PDEs increases, the difference in maximum accuracy between FEM and PINNs decreases. It is also noticeable that for smaller grid sizes, the computation accuracy of PINNs can outperform that of FEM. Additionally, for numerical methods, an increase in grid number leads to improved accuracy, while in PINNs, accuracy improvement is not directly related to the increase in the number of network parameters.

Dong and Li [53] compared the computation accuracy and time of the deep Galerkin method (DGM), PINNs, and FEM, where DGM is another neural network-based method for solving PDEs. Table 2 presents the comparison of the maximum error, root mean square error, and computation time between PINNs and FEM for the one-dimensional nonlinear Helmholtz equation. In this case, PINNs used Adam and L-BFGS as optimizers, while FEM used different numbers of discrete elements. It can be seen that the maximum error of PINNs is six orders of magnitude higher than that of FEM, and in terms of computation time, PINNs require over 31.9 times more time than FEM (578.2/18.1 ≈ 31.9), which is attributed to the tens of thousands of epochs required for training PINNs. Table 3 compares the maximum error, root mean square error, and computation time between DGM and FEM for the one-dimensional viscous Burgers' equation. Similar to PINNs, FEM shows a significant advantage over DGM in both computation accuracy and cost.

In addition, PINNs rely on automatic differentiation to compute PDE residuals, and the computational cost increases, especially when higher-order derivatives (such as second-order or higher) are involved. Therefore, the combination of PINNs with numerical differentiation has also attracted researchers' attention. Specifically, automatic differentiation used to compute partial derivatives in PDEs is replaced or supplemented by numerical differentiation [54-58]. Karnakov et al. [58] highlighted two advantages of numerical differentiation. First, it can reduce computation cost, as the cost of automatic differentiation increases exponentially with the order of partial derivatives, while numerical differentiation increases linearly. Second, discretizing PDEs through numerical differentiation allows

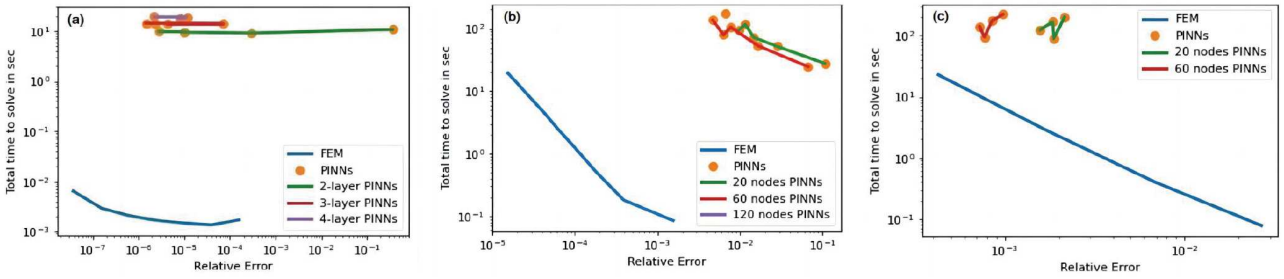


Figure 2 (Color online) Curve of computation time (s) versus error for (a) 1D Poisson equation, (b) 2D Poisson equation, and (c) 3D Poisson equation [52].

Table 2 Comparison of error and time for solving one-dimensional nonlinear Helmholtz equation [53]

Method	Elements	Maximum error	RMS error	Epochs	Training time/wall-time (s)
PINN (Adam)	–	4.56×10^{-3}	5.04×10^{-4}	45000	578.2
PINN (L-BFGS)	–	1.69×10^{-3}	1.69×10^{-4}	22000	806.4
FEM	200000	5.26×10^{-9}	1.37×10^{-9}	–	4.7
	400000	1.31×10^{-9}	3.43×10^{-10}	–	8.8
	800000	3.29×10^{-10}	8.57×10^{-11}	–	18.1

Table 3 Comparison of error and time for solving one-dimensional viscous Burgers' equation [53]

Method	Elements	Δt	Maximum error	RMS error	Epochs	Training time/wall-time (s)
DGM (Adam)	–	–	4.57×10^{-2}	5.76×10^{-3}	128000	1797.8
DGM (L-BFGS)	–	–	7.50×10^{-3}	1.55×10^{-3}	28000	1813.5
FEM	2000	0.001	2.64×10^{-5}	5.15×10^{-6}	–	12.5
	2000	0.0005	3.07×10^{-5}	5.76×10^{-6}	–	25.4
	5000	0.001	1.89×10^{-5}	1.74×10^{-6}	–	26.0
	5000	0.0005	4.13×10^{-6}	7.90×10^{-7}	–	50.8
	10000	0.001	2.22×10^{-5}	1.99×10^{-6}	–	47.7
	10000	0.0005	4.74×10^{-6}	4.36×10^{-7}	–	92.6

for the convenient integration of various well-established numerical methods, such as upwind schemes. Chiu et al. [57] verified the potential of numerical differentiation in reducing the number of collocation points. They proposed the coupled automatic-numerical differentiation PINNs (CAN-PINNs) by combining automatic differentiation and numerical differentiation, which can achieve satisfactory results with fewer collocation points. Raynaud et al. [56] improved the reconstruction accuracy of flow around a cylinder by discretizing the Navier-Stokes equations in the time dimension using Fourier transform. However, the limitations of numerical differentiation should also be considered, such as the introduction of discretization errors and dependence on the mesh in some cases.

2.2 Poor generality

Poor generality is also a major limitation of PINNs. On the one hand, it is difficult to handle complex engineering problems. On the other hand, even when faced with simple model equations, PINNs may fail to solve for various reasons. Some typical examples include: for flow problems in fluid mechanics, the PDEs loss function in PINNs cannot guarantee a decrease in the total PDE residual of the entire computational domain under nonuniform collocation points, resulting in non-physical results [59] (as shown in Figure 3). For unsteady cylinder flow, due to the lack of disturbance in the solution process, PINNs are unable to obtain periodic vortex shedding (Karman vortex street) [60]. When the Reynolds number in the Navier-Stokes equation is high, PINNs cannot obtain satisfactory results [61]. For turbulence simulation, the training of PINNs with turbulence models becomes more complex, and the prediction results for the backward-facing step flow show significant differences compared to the reference solution [62]. When the solution of PDEs has multi-scale or high-frequency features, PINNs that follow the frequency principle [63] have insufficient approximation ability for small-scale/high-frequency, making it difficult to obtain accurate results [64-66]. Wang et al. [64] used PINNs to solve the one-dimensional Poisson equation $\Delta u(x) = f(x)$, and its analytical solution was designed to have significant frequency discrepancy $u(x) = \sin(2\pi x) + 0.1\sin(50\pi x)$. The comparison between the NN-predicted solution and the analytical solution is shown in Figure 4. We observe that the results of PINNs have significant errors. For time-dependent PDEs, PINNs that minimize the equation residual over the entire time interval violate the time evolution laws of dynamic systems, leading to failure in solving problems that are sensitive to initial conditions [67]. For problems where the external excitation is a point source (represented mathematically by a Dirac function), PINNs fail to solve correctly due to the singularity of the Dirac function [68].

2.3 Numerous empirical parameters and strong parameter sensitivity

The conflict of multiple losses has always been widely present in machine learning. To address this problem, researchers have introduced empirical parameters as loss weights to ensure a reasonable training process [69]. In PINNs, this issue is particularly prominent because there are not only conflicts between the IBCs, equation residuals, and observed data mismatch (inverse problem), but also conflicts between different equations (for equation systems) and equation residuals at different collocation points, all of which can undermine the correct solution of PINNs. For example, the conflict between PDE loss and boundary condition loss resulted in a significant deviation in the solution of the Helmholtz equation at the boundary [70]. Conflicts in equation residuals at different times lead to a solution process that violates the time evolution laws of dynamical sys-

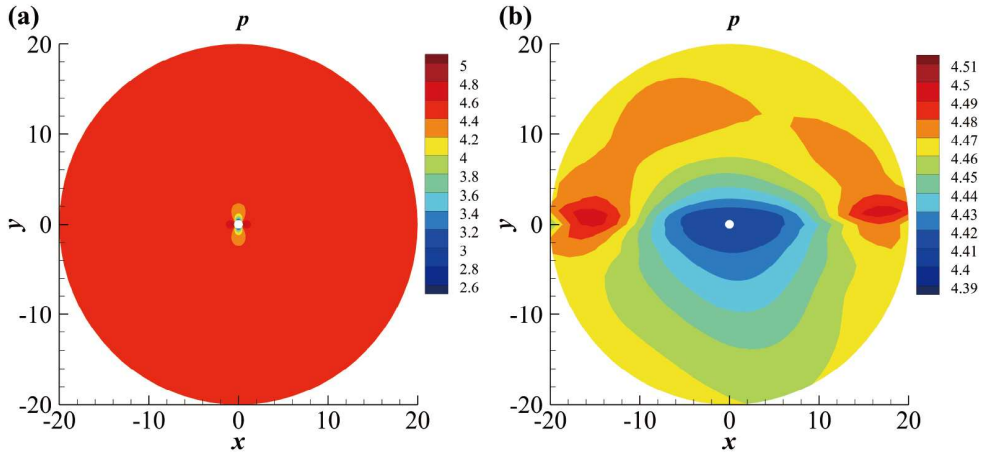


Figure 3 (Color online) Comparison of pressure field between (a) the reference solution from the finite volume method and (b) the solution given by PINNs for solving inviscid compressible flow over a circular cylinder [59].

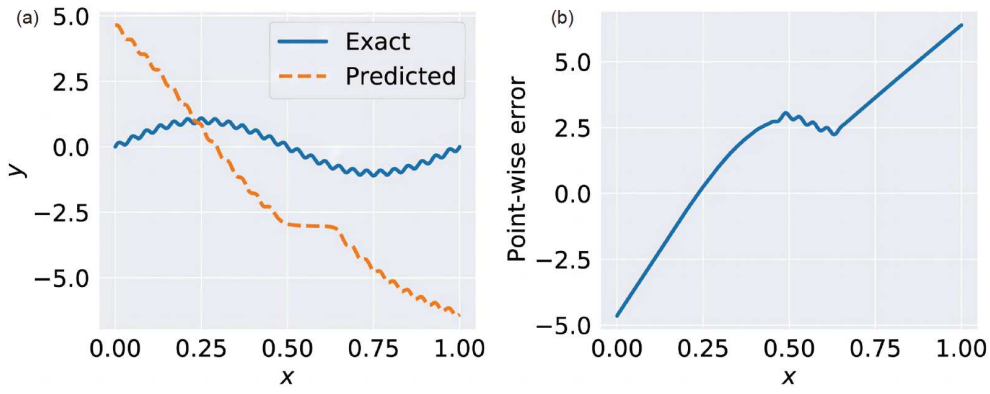


Figure 4 (Color online) 1D Poisson equation: results obtained by training a PINN. (a) Comparison of predicted and exact solutions. (b) Point-wise error between the predicted and the exact solution [64].

tems. Conflicts in equation residuals at collocation points with different sampling densities result in ill-conditioning [59]. Additionally, there are magnitude differences between the equation residuals involving differential terms and the losses from IBCs/observed data, which further exacerbate the difficulty of balancing the losses [65,70-72].

To alleviate the loss conflict problem of PINNs, many researchers have conducted targeted research, and the common strategy is to weight the losses and equation residuals at each collocation point through empirical parameters. The weight design for various losses is mainly carried out through adaptive methods: Wang et al. [64] have designed a weight calculation method by statistically analyzing the gradient of the loss during the network training process. Yu et al. [73] have alleviated potential conflicts by projecting gradient descent directions for different losses. Xiang et al. [74] have constructed loss weights by incorporating maximum likelihood estimation. Wang et al. [65] have also designed a weight adaptive method by conducting neural tangent kernel (NTK) analysis on PINNs. Wang et al. [75] have proposed a loss-balancing framework based on the concept of power-law scaling. For the weighting of equation residuals at each collocation point, Song et al. [59] have proposed volume weighting PINNs (VW-PINNs), which weights equation residuals through the volume occupied by collocation points in the computational domain, effectively improving the performance of PINNs under nonuniform collocation points. In addition, some adaptive point weighting methods [76-78] have also been proposed to enhance vanilla PINNs frameworks. Undoubtedly, these methods have to some extent alleviated the conflict issues during the network training process. However, they have also introduced numerous empirical parameters with high algorithmic sensitivity, increasing the complexity of PINNs and potentially compromising robustness in solving process. In other words, these methods have not fundamentally eliminated conflicts of various losses.

Recently, researchers have proposed some methods of enforcing IBCs through hard constraints to eliminate the losses of IBCs in PINNs. These frameworks are mainly categorized into implementations involving fully connected networks [47,79-82] and convolutional networks [55,83,84]. For the former, the researchers constructed solution functions that a priori satisfy the IBCs by introducing boundary functions and distance functions. For the latter, the imposition of hard constraints is achieved through padding the graph, which is similar to the handling of classical numerical methods. These methods avoid the conflict between the loss of IBCs and other losses, but they still have limitations. On the one hand, conflicts still exist between different equations at different collocation points. On the other hand, the boundary function and distance function introduced in fully connected networks are still empirical. In convolutional networks, PINNs cannot use automatic differentiation and rely on mesh, which results in PINNs losing their unique advantages over classical numerical methods, including solving high-dimensional problems [16,17] and parametric problems [48,85].

3 Exploration of the root causes of PINNs' failure in solving PDEs

The three main limitations of PINNs summarized above are based on the case results from relevant literature. In fact, this approach of summarizing is more from the perspective of PINNs users' operations. However, to improve PINNs, it is necessary to analyze the underlying mechanisms behind these limitations. Building on the three main limitations mentioned above, this section discusses the root causes of PINNs' failure in solving PDEs, analyzing the following three points: (1) poor multiscale approximation ability and ill-conditioning caused by PDE losses; (2) insufficient exploration of convergence and error analysis, resulting in weak mathematical rigor; (3) inadequate integration of physical information, causing mismatch between residuals

and iteration errors.

3.1 Poor multiscale approximation ability and ill-conditioning caused by PDE losses

As is well known, searching for neural network parameters is an optimization problem aimed at minimizing the loss functions. According to the current research, whether it is data-driven modeling or solving PDEs, researchers commonly use deep neural networks with a large number of parameters to achieve sufficient nonlinear approximation capability. The optimization problem becomes high-dimensional and non-convex, making it extremely difficult to search for the global minimum. Compared to data-driven modeling, the optimization problem constructed by PINNs is more complex and challenging [71,72].

The optimization difficulties of PINNs mainly come from two aspects. On the one hand, there are significant scale discrepancies such as multiple frequencies and strong anisotropy in the solution, which makes it difficult to be accurately approximated by the global basis functions constructed by vanilla PINNs. On the other hand, PDE losses lead to ill-conditioning in optimization. Compared to the L_2 regularization constraints in the data-driven modeling, the regularization constraints corresponding to PDE residuals composed of differential terms are nontrivial. To handle the former difficulty, Liu et al. [86] and Wang et al. [64] both pointed out the challenges that PINNs face in the multiscale problems, and they designed improved frameworks by constructing multiscale inputs a priori and by scaling spatio-temporal coordinates in the sine space, respectively. In fact, these studies can be traced back to the work of Rahaman et al. [87] and Xu et al. [63], who explained the fitting preferences of neural networks for low-frequency features from the perspectives of spectral bias and frequency domain. Cao et al. [88] have pointed out that the large gradient characteristics of inviscid airfoil flow near the wall in physical space are unfavorable for flow simulation, and have achieved satisfactory results in computational space with very few meshes through the mesh transformation method. In contrast, the second-order finite volume method would require nearly 20 times more meshes to achieve the same accuracy, as shown in Figure 5. Wu et al. [89] have designed transformation parameters based on prior knowledge to perform linear transformations on PDEs, enhancing the learning ability of PINNs in the boundary layer region of the flow field. The author verified the effectiveness of the proposed method through a large number of examples.

For the latter difficulty, recent preprocessing studies on PINNs [90-92] have theoretically analyzed in detail the ill-conditioning in optimization caused by PDE loss. Rathore et al. [90] analyzed the condition numbers of the Hessian matrix corresponding to each loss term in PINNs (Figure 6), and pointed out the ill-conditioning in optimization caused by equation residuals from the perspective of the loss landscape. They further explained the advantages and limitations of the L-BFGS optimization algorithm in PINNs, and finally proposed the NysNewton CG optimization method to improve it. By introducing appropriate assumptions, de Ryck et al. [91] proved that the convergence rate of gradient descent in PINNs is related to the condition number of the composite operator composed of the Hermitian square of the differential operator and the kernel integral operator related to the tangent kernel of the underlying model, and proposed improvement strategies through operator preprocessing. Liu et al. [92] combined the ideas of traditional paradigms, transforming PDEs into a system of linear equations and analyzing the reasons for poor convergence and accuracy of PINNs from the perspective of coefficient matrix condition numbers, and proposed a method to preprocess equation residuals to improve algorithm performance. Cao and Zhang [93] established a relationship between the ill-conditioning of PINNs and the Jacobian matrix of the PDE system, demonstrating that the degree of ill-conditioning of PINNs is proportional to the condition number of the Jacobian matrix. They proposed an improved strategy based on the time-stepping method, successfully solving the forward problem of three-dimensional flow around a wing. Overall, despite

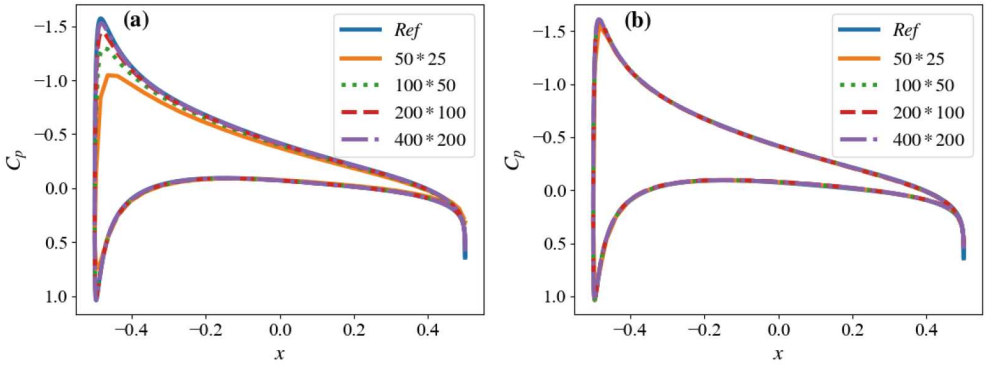


Figure 5 (Color online) Pressure coefficient distributions of the flow around the NACA0012 airfoil obtained by (a) FVM and (b) NNfoil under different meshes [88].

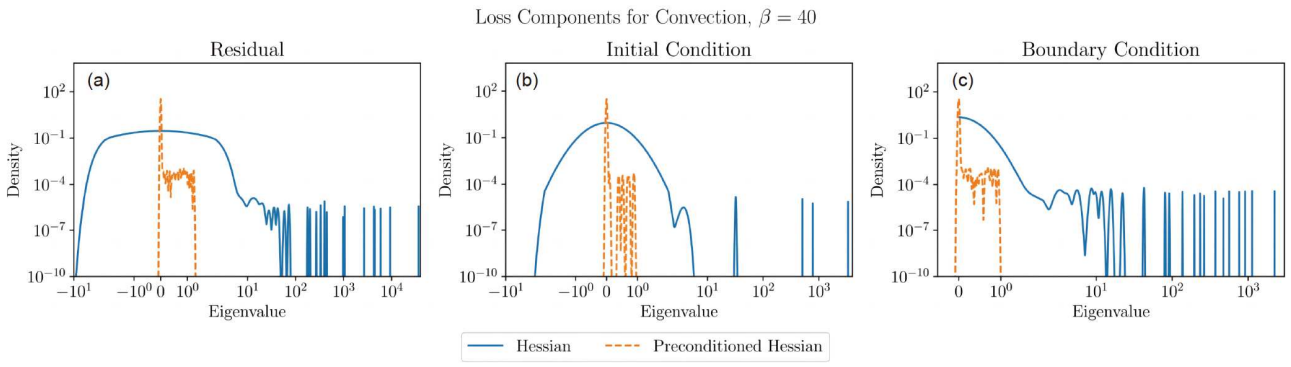


Figure 6 (Color online) Spectral density of the Hessian and the preconditioned Hessian of each loss term for the convection equation [90]. (a) Residual components of the PINN loss. (b) Initial condition components of the PINN loss. (c) Boundary condition components of the PINN loss.

PINNs embedding PDEs to ensure that network outputs better adhere to physics, satisfactory results can be achieved even with no data or only a small amount of data. However, the PDE loss also leads to poor trainability and even falling into ill-conditioning in optimization, which is a fundamental reason for the low accuracy, efficiency, and poor generality of PINNs.

3.2 Insufficient exploration of convergence and error analysis, resulting in weak mathematical rigor

Convergence is an important property considered in error analysis for classical numerical methods, reflecting whether the solution of the discretized equation approaches the solution of the differential equation as the spatial and temporal grid steps tend toward zero. In classical numerical methods, extensive research has focused on theoretical proofs of convergence and strategies for improving convergence [94-100], thereby enhancing the robustness of these methods in solving complex problems and engineering applications.

However, for PINNs, research on convergence remains to be deepened. Referring to the concept of classical numerical methods, the primary challenge in studying the convergence of PINNs is how to define the error of PINNs and the degrees of freedom that affect error magnitude. Regarding PINNs' error, Shin et al. [101] decomposed the total error of PINNs into approximation error, estimation error, and optimization error from the perspective of statistical learning error. The approximation error refers to the theoretical error between the neural network of a given architecture and the true model, the estimation error arises when evaluating the model at unknown points, and the optimization error is the difference between the network's achieved accuracy and its theoretical accuracy after training. The approximation error depends solely on the complexity of the model architecture, while the estimation error and approximation error together form the generalization error. Optimization error originates from the current difficulty of finding the global minimum of a non-convex function. As for the degrees of freedom that affect error magnitude, they need to be defined according to the type of PINNs' error. The approximation error is related only to model architecture complexity, determined by the number of neurons. The universal approximation theorem indicates that a neural network with at least one hidden layer can approximate any function to arbitrary precision [102]. The number of neurons sets the precision the network can achieve, making it a degree of freedom that affects the approximation error. For estimation and optimization errors, the optimization process is influenced by multiple factors, such as model width and depth, dataset size, optimizers, training epochs, learning rate, and activation functions. A relatively effective research approach is to study the convergence characteristics of the error with changes in model width, model depth, and dataset size while keeping optimizers, training epochs, learning rate, and activation function constant. Then, by varying the optimizers, training epochs, learning rate, and activation functions, the consistency of the convergence behavior can be investigated [103].

Building on the error definitions for PINNs by Shin et al. [101], researchers have conducted theoretical studies on the convergence of PINNs [101,104-107]. However, as noted in ref. [103], current theoretical research on PINNs' convergence mainly focuses on analyzing generalization error bounds for different types of PDEs. By establishing a link between training error and generalization error, it aims to ensure that, under certain conditions, sufficiently low training error can imply a sufficiently low generalization error. Although this research has advanced the theoretical understanding of PINNs' convergence, these theories are based on the assumption that PINNs' optimization can reach a global minimum, which exceeds the capabilities of current optimizers. The NTK theory [65] considers the impact of the optimizers, exploring gradient descent convergence for fully connected neural networks in the infinite-width limit. By monitoring PINNs' convergence behavior through NTK and the relative change of the parameter vectors, results show that, for shallow networks in the infinite-width limit, gradient descent training with an infinitesimally small learning

rate can guarantee convergence of PINNs training. However, the NTK theory's assumptions are idealized, requiring an infinitely wide single-layer network, low learning rates, and a gradient descent optimizer, which limits its practical applicability. Since current theoretical studies on PINNs' convergence are restricted to idealized cases and lack practical guidance, Pantidis et al. [103] investigated PINNs' convergence concerning network architecture complexity and dataset size through numerical studies. Considering randomness from initialization, Pantidis et al. [103] found that both training error and global error of PINNs tend to minimize with increasing network complexity and dataset size, as shown in Figure 7. Consistent convergence patterns were observed across different optimizers, training epochs, and learning rates. In summary, further progress in studying PINNs' convergence is needed through both theoretical and numerical approaches. On the one hand, theoretical research should enhance generality and applicability to provide practical guidance; on the other hand, numerical research should expand, verifying and generalizing findings across different types of problems.

3.3 Inadequate integration of physical information, causing mismatch between residuals and iteration errors

Residuals and iterative errors are two important concepts in the CFD simulation. Residuals refer to the degree to which the governing equation is satisfied at each discrete point during a numerical iteration process. In other words, residuals measure the imbalance between the current solution and the governing equation. While iterative errors refer to the difference between the current numerical solution and the exact solution (theoretical solution of the discrete equations). Iterative errors reflect the process of the computed solution gradually approaching the theoretical solution. The relationship between the two is that small iterative errors usually require small residuals as a basis, but conversely, small residuals do not always lead to small iterative errors [108].

In PINNs, the mismatch between residuals and iterative errors is further exacerbated due to the inherent characteristics of the method. The vanilla PINNs have two typical characteristics: one is the use of soft constraints on boundary conditions, and the other is the discretization of the solution space using scattered points. These features give PINNs flexibility in handling various boundary condition types and geometric shapes. From the perspective of the loss function, the solving process in PINNs involves progressively minimizing the equation loss on scattered points and the boundary condition loss on boundary points, with the solution deemed complete once the residuals are below a certain threshold. However, from a physical perspective, the soft constraints on boundaries can lead to deviations in boundary conditions, potentially affecting the well-posedness of the PDE solutions. Additionally, an improper distribution of scattered points may result in mismatches between the characteristics of the PDE in continuous and discrete spaces.

Let us start by analyzing the treatment of boundary conditions. Classical numerical methods handle boundary conditions by explicitly applying values, gradients, or a combination of both at the boundaries [109-111]. These imposed boundary conditions are incorporated into the algebraic equation system formed by the numerical discretization of the PDEs, directly influencing values within the solution domain through iterative solving. In classical methods, boundary handling is a strict physical constraint; though there may be variations in accuracy across different boundary handling methods, the physical characteristics of the problem are directly transferred across the entire solution domain via the boundary. However, in PINNs, boundary conditions are addressed as part of the loss function, which approximates boundary conditions by minimizing the difference between model predictions and actual boundary conditions. This means boundary conditions are applied in a

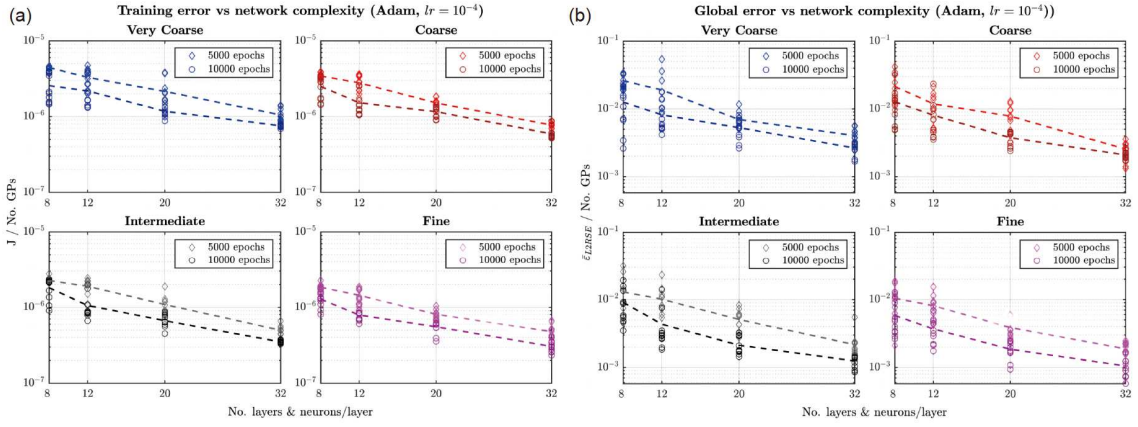


Figure 7 (Color online) (a) Training error and (b) global error of PINNs tend to minimize as network complexity increases (using the Adam optimizer with a learning rate of 10^{-4}, with results recorded on four different grid sets) [103].

statistical rather than a strictly physical sense. Soft boundary constraints make it challenging for complex boundary conditions to be strictly satisfied in PINNs. Although the numerical boundary error magnitude may be small, such an error can lead to significant deviations in boundary conditions from a physical perspective. For instance, unlike flow around bluff bodies, flow around airfoils may be sensitive to boundary condition errors, which implicitly alter the airfoil shape distribution; even slight deviations in airfoil shape can have a substantial impact on flow behavior [112,113]. Designing appropriate loss function weight coefficients based on the physical characteristics of the problem to ensure boundary condition errors meet accuracy requirements is key to improving the consistency of PINNs' results with expected physical field behavior.

Next, let us analyze from the perspective of domain discretization. Classical numerical methods generally discretize the solution domain using structured or unstructured grids. The griddistribution is often pre-adjusted according to the physical characteristics of the problem, such as locally refining around boundary layers or shock regions in fluid flow. This targeted handling of grids not only reduces computation cost but also incorporates the physical scale features of the problem into the grid distribution, enhancing the capability of classical numerical methods to handle complex problems [114,115]. In contrast, PINNs represent the solution domain using scattered points for discretization. Although scattered points can better handle complex geometries without needing to consider grid topology, a scattered point distribution that does not align with the physical spatial scale features increases the difficulty for vanilla PINNs to learn the physical characteristics of the solution domain. Ideally, the distribution of points in PINNs should be associated with the physical features of the domain. Adaptive approaches, for example, could be used to adjust the point distribution in PINNs [78,116-118]. Moreover, grids have the advantage of containing volume information, a physical attribute that, when introduced into the loss function of PINNs, can positively influence the approximation of the optimization objective [59], as shown in Figure 8.

In summary, PINNs require the integration of domain-specific expert knowledge to effectively design and optimize the model, enhancing the physical relevance of the model and increasing its applicability in complex scenarios.

4 Summary and outlook on solving PDEs with neural networks

PINNs introduce PDEs into the loss function of neural networks, offering a novel approach to solving PDEs through optimization, which has driven their widespread application across various fields and suggests a promising future for this research direction. Compared with traditional numerical

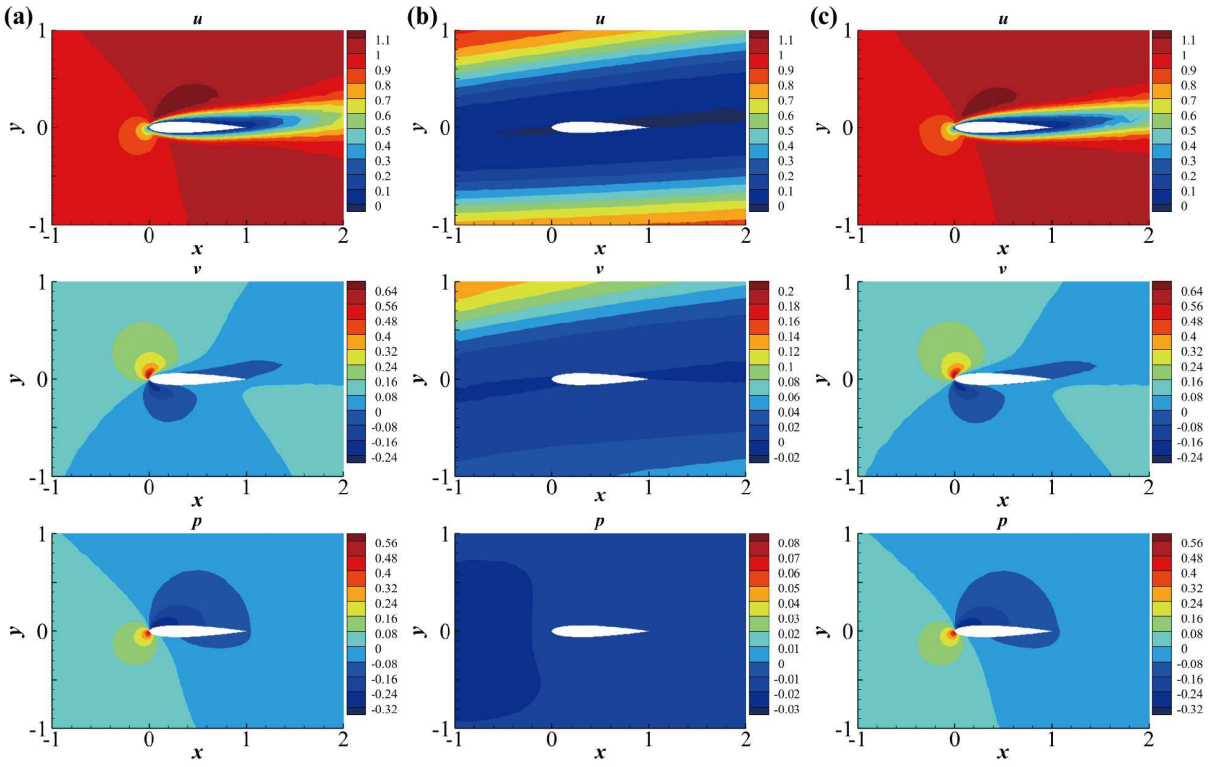


Figure 8 (Color online) Comparison of the near-wall flow field around the viscous incompressible NACA0012 airfoil: (a) reference solution; (b) PINNs; (c) VW-PINNs [59].

methods that rely on domain discretization and iterative solvers, PINNs offer a more flexible and efficient computational framework. By leveraging neural networks' ability to approximate functions continuously over a given parameter space, PINNs can efficiently solve for the entire solution space in one pass, making them a promising and cost-effective surrogate modeling approach [48,119,120]. This capability enables real-time solution predictions and provides strong support for optimization and design [49,121]. Furthermore, PINNs combine observational data with physics-consistent regularization, allowing for accurate reconstruction of flow fields even under incomplete or noisy data conditions [38,39]. This presents an innovative method for integrating experimental data into computational models, bridging the gap between simulations and real-world measurements.

However, achieving stable training and obtaining correct results remain a challenge in many complex scenarios. This paper analyzes the case results of PINNs from relevant literature, summarizing the failure phenomena and limitations of PINNs: (1) low solution accuracy and high computation cost; (2) poor generality; (3) numerous empirical parameters and strong parameter sensitivity. An in-depth analysis of these limitations is then conducted, revealing the underlying mechanisms that contribute to these constraints. Three root causes for the failure of PINNs in solving complex problems are identified: (1) poor multiscale approximation ability and ill-conditioning caused by PDE losses; (2) insufficient exploration of convergence and error analysis, resulting in weak mathematical rigor; (3) inadequate integration of physical information, causing mismatch between residuals and iteration errors. The above analysis provides a basis for further improvements in PINNs. We believe that subsequent improvements in using neural networks to solve PDEs can focus on the following aspects.

4.1 Analysis of ill-conditioning in PINNs and mitigation strategies

The ill-conditioning during optimization in PINNs is a major obstacle to their application in complex engineering problems. Different perspectives on the ill-conditioning of PINNs have inspired various mitigation strategies. One perspective is to relate the ill-conditioning of PINNs to the Hessian matrix of the loss function. Rathore et al. [90] explained the optimization stagnation caused by residual loss in terms of the spectral density of the Hessian matrix of the loss function and improved PINNs performance by enhancing the gradient descent algorithm. De Ryck et al. [91] argued that the convergence rate of gradient descent in PINNs is related to the condition number of a composite operator, which consists of the Hermitian square of differential operators and the kernel integral operator associated with the tangent kernel for the underlying model. They proposed an improvement strategy from the perspective of operator preconditioning. Another perspective is to relate the ill-conditioning of PINNs to the Jacobian matrix of the PDE system. Based on an understanding of ill-conditioning in classical numerical methods, Cao and Zhang [93] associated the ill-conditioning of PINNs with the Jacobian matrix of the PDE system rather than the Hessian matrix of the loss function. They constructed a controlled system based on control theory for any dynamic system, allowing adjustment of the gain to alter the Jacobian matrix's condition number. Numerical examples then verified the correlation between the ill-conditioning of PINNs and that of the PDE system's Jacobian matrix. This evolving understanding not only sheds light on the intrinsic limitations of PINNs but also opens avenues for their broader application in solving complex physical problems.

4.2 Improvements to PINNs by enforcing temporal causality

When solving time-dependent PDE problems, PINNs typically select residual sampling points across the entire spacetime domain and use all these points simultaneously for training the neural network parameters. This solving process indicates that PINNs treat the spacetime dimensions of PDEs in a uniform manner. Clearly, this disrupts the inherent temporal causality of the PDE system and weakens the physical validity of the PINNs solution. Enforcing temporal causality is an improvement direction for PINNs, ensuring the correct representation of the intrinsic physical properties of unsteady systems as they evolve over time.

To adhere to the physical evolution of flow over time, Wang et al. [122] assigned weights to the collocation points along the time dimension, only solving for the next time step when the loss at the previous time step is sufficiently low, indicating that the solution has been adequately obtained. Krishnapriyan et al. [71] performed time segmentation with equal intervals and solved each time interval sequentially. Cao and Zhang [61,93] proposed the time-stepping-oriented neural network (TSONN), which replaces the original equation with a series of pseudo-time stepping equations. This approach aligns the convergence history of the neural network with the pseudo-time evolution of the physical system, making it applicable to both time-dependent and time-independent problems. These studies demonstrate that adhering to temporal causality can significantly improve the performance of PINNs in solving problems.

4.3 Empowering PINNs with classical numerical methods

Classical numerical methods have made significant progress

over several decades, accumulating a profound understanding of physics in areas such as mesh generation strategies and numerical scheme design. How to leverage various techniques from classical numerical methods and their deep insights into physics to empower the solution of PINNs, while not losing the core advantages that PINNs offer, is a direction worth considering for improving PINNs. He et al. [123], inspired by the entropy viscosity method [124] developed in CFD, proposed the artificial viscosity (AV)-based PINN method to stabilize the simulation of high Reynolds number flows, successfully solving the two-dimensional lid-driven cavity flow at $Re = 1000$. Similarly, the TSONN method proposed by Cao and Zhang [61,93] was inspired by the pseudo-time stepping method in CFD and achieved simulated lid-driven cavity flow and three-dimensional flow around a wing at $Re = 5000$. In classical numerical methods, the grid distribution also plays a crucial role. For instance, in high Reynolds number wall-bounded turbulence with multiscale phenomenon, anisotropic grids within the boundary layer are often indispensable. Drawing from grid transformation strategies in CFD, Cao et al. [88] and Zhang et al. [125] used neural networks to learn flows in the computational space rather than the physical space. By partially incorporating the physical scale characteristics into the grid distribution, they significantly alleviated challenges in using PINNs to solve inviscid airfoil flow and high Reynolds number wall-bounded turbulence problems. In the future, the integration of more numerical methods with PINNs holds significant potential for exploration and development. This combination could leverage the strengths of traditional numerical techniques and neural networks, addressing limitations in current methods while enhancing the efficiency and accuracy of solving complex physical problems.

5 Conclusion

Since its formal introduction in 2019, PINNs have been widely applied across various fields due to their ability to unify the solution of forward and inverse problems, strong flexibility, and mesh-free nature. PINNs and their variants have also seen significant advancements, becoming an important complement to the numerical methods for CFD. However, as PINNs are increasingly applied, certain failure phenomena have also been gradually revealed and discussed. To clarify the underlying causes of these failures, this paper first summarizes the failure phenomena of PINNs mentioned in the literature and categorizes them into three limitations of PINNs: (1) low solution accuracy and high computation cost; (2) poor generality; (3) numerous empirical parameters and strong parameter sensitivity. An in-depth analysis of these limitations is then conducted, revealing the underlying mechanisms that contribute to these constraints. Three root causes for the failure of PINNs in solving complex problems are identified: (1) poor multiscale approximation ability and ill-conditioning caused by PDE losses; (2) insufficient exploration of convergence and error analysis, resulting in weak mathematical rigor; (3) inadequate integration of physical information, causing mismatch between residuals and iteration errors. By focusing on addressing these limitations in PINNs, we outline the future directions and prospects for the PINNs: (1) analysis of ill-conditioning in PINNs and mitigation strategies; (2) improvements to PINNs by enforcing temporal causality; (3) empowering PINNs with classical numerical methods. We believe that further research aimed at addressing the limitations of PINNs will unlock PINNs' full potential and establish them as a crucial pillar of intelligent computing.

This work was supported by the National Natural Science Foundation of China (Grant No. 92152301).

Conflict of interest The authors declare that they have no conflict of interest.

1 G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Nat. Rev. Phys. **3**, 422 (2021).
2 R. Courant, K. Friedrichs, and H. Lewy, Math. Ann. **100**, 32 (1928).
3 R. Eymard, G. Thierry, and R. Herbin, *Finite Volume Methods* (Elsevier, Amsterdam, 2000).
4 G. Dhatt, E. Lefrançois, and G. Touzot, *Finite Element Method* (John Wiley & Sons, Hoboken, 2012).
5 A. T. Patera, J. Comput. Phys. **54**, 468 (1984).
6 E. Divo, and A. J. Kassab, Eng. Anal. Bound. Elem. **29**, 136 (2005).
7 M. Zerroukat, K. Djidjeli, and A. Charafi, Int. J. Numer. Meth. Eng. **48**, 19 (2000).
8 W. M. McEneaney, SIAM J. Control Opt. **46**, 1239 (2007).
9 A. Doostan, and G. Iaccarino, J. Comput. Phys. **228**, 4332 (2009).
10 J. P. Boyd, and N. Flyer, Comput. Methods Appl. Mech. Eng. **175**, 281 (1999).
11 D. Charrier, *Communication-Avoiding Algorithms for a High-Performance Hyperbolic Pde Engine* (Dissertation for Master Degree, Durham University, 2020).
12 M. Raissi, P. Perdikaris, and G. E. Karniadakis, J. Comput. Phys. **378**, 686 (2019).
13 M. W. M. G. Dissanayake, and N. Phan-Thien, Commun. Numer. Meth. Eng. **10**, 195 (1994).
14 I. E. Lagaris, A. Likas, and D. I. Fotiadis, IEEE Trans. Neural Netw. **9**, 987 (1998).
15 B. P. van Milligen, V. Tribaldos, and J. A. Jiménez, Phys. Rev. Lett. **75**, 3594 (1995).
16 J. Sirignano, and K. Spiliopoulos, J. Comput. Phys. **375**, 1339 (2018).
17 W. E, and B. Yu, Commun. Math. Stat. **6**, 1 (2018).
18 N. Wang, D. Zhang, H. Chang, and H. Li, J. Hydrol. **584**, 124700 (2020).
19 M. Raissi, A. Yazdani, and G. E. Karniadakis, Science **367**, 1026 (2020).
20 X. Jin, S. Cai, H. Li, and G. E. Karniadakis, J. Comput. Phys. **426**, 109951 (2021).
21 H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa, Phys. Fluids **34**, 075117 (2022).
22 Z. Mao, A. D. Jagtap, and G. E. Karniadakis, Comput. Methods Appl. Mech. Eng. **360**, 112789 (2020).
23 C. Rao, H. Sun, and Y. Liu, Theor. Appl. Mech. Lett. **10**, 207 (2020).

24 S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, J. Heat Transfer **143**, 060801 (2021).
25 N. Zobeiry, and K. D. Humfeld, Eng. Appl. Artif. Intell. **101**, 104232 (2021).
26 S. A. Niaki, E. Haghighat, T. Campbell, A. Poursartip, and R. Vaziri, Comput. Methods Appl. Mech. Eng. **384**, 113959 (2021).
27 X. Zhao, Z. Gong, Y. Zhang, W. Yao, and X. Chen, Eng. Appl. Artif. Intell. **117**, 105516 (2023).
28 M. Raissi, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, J. Fluid Mech. **861**, 119 (2019).
29 C. Cheng, H. Meng, Y. Z. Li, and G. T. Zhang, Ocean Eng. **240**, 109932 (2021).
30 E. Kharazmi, D. Fan, Z. Wang, and M. S. Triantafyllou, J. Fluids Struct. **107**, 103367 (2021).
31 Z. Fang, and J. Zhan, IEEE Access **8**, 24506 (2019).
32 A. J. G. Inda, S. Y. Huang, N. İmamoğlu, R. Qin, T. Yang, T. Chen, Z. Yuan, and W. Yu, Diagnostics **12**, 2627 (2022).
33 C. Song, and Y. Wang, Geophys. J. Int. **232**, 1503 (2023).
34 J. D. Smith, Z. E. Ross, K. Azizzadenesheli, and J. B. Muir, Geophys. J. Int. **228**, 698 (2022).
35 B. Huang, and J. Wang, IEEE Trans. Power Syst. **38**, 572 (2022).
36 S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, Acta Mech. Sin. **37**, 1727 (2021).
37 S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, J. Sci. Comput. **92**, 88 (2022).
38 S. Cai, Z. Wang, F. Fuest, Y. J. Jeon, C. Gray, and G. E. Karniadakis, J. Fluid Mech. **915**, A102 (2021).
39 M. Raissi, A. Yazdani, and G. E. Karniadakis, arXiv: 1808.04327.
40 H. Wang, Y. Liu, and S. Wang, Phys. Fluids **34**, 017116 (2022).
41 J. G. R. von Saldern, J. M. Reumschüssel, T. L. Kaiser, M. Sieber, and K. Oberleithner, Phys. Fluids **34**, 115129 (2022).
42 M. Delcey, Y. Cheny, and S. Kiesgen de Richter, Phys. Fluids **35**, 027124 (2023).
43 A. Arzani, J. X. Wang, and R. M. D'Souza, Phys. Fluids **33**, 071905 (2021).
44 P. Yin, G. Xiao, K. Tang, and C. Yang, SIAM J. Sci. Comput. **46**, C127 (2024).
45 N. Demo, M. Strazzullo, and G. Rozza, Comput. Math. Appl. **143**, 383 (2023).
46 S. Mowlavi, and S. Nabi, J. Comput. Phys. **473**, 111731 (2023).
47 L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, SIAM J. Sci. Comput. **43**, B1105 (2021).
48 L. Sun, H. Gao, S. Pan, and J. X. Wang, Comput. Methods Appl. Mech. Eng. **361**, 112732 (2020).
49 Y. Sun, U. Sengupta, and M. Juniper, Comput. Methods Appl. Mech. Eng. **411**, 116042 (2023).
50 A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, Comput. Methods Appl. Mech. Eng. **365**, 113028 (2020).
51 N. McGreivy, and A. Hakim, Nat. Mach. Intell. **6**, 1256 (2024).
52 T. G. Grossmann, U. J. Komorowska, J. Latz, and C. B. Schönlieb, IMA J. Appl. Math. **89**, 143 (2024).
53 S. Dong, and Z. Li, Comput. Methods Appl. Mech. Eng. **387**, 114129 (2021).
54 Y. Zou, T. Li, L. Lu, J. Wang, S. Zou, L. Zhang, and X. Deng, Phys. Fluids **36**, 103608 (2024).
55 H. Gao, L. Sun, and J. X. Wang, J. Comput. Phys. **428**, 110079 (2021).
56 G. Raynaud, S. Houde, and F. P. Gosselin, J. Comput. Phys. **464**, 111271 (2022).
57 P. H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, and Y. S. Ong, Comput. Methods Appl. Mech. Eng. **395**, 114909 (2022).
58 P. Karnakov, S. Litvinov, P. Koumoutsakos, and D. Abbott, Proc. Natl. Acad. Sci. USA Nexus **3**, pgae005 (2024).
59 J. Song, W. Cao, F. Liao, and W. Zhang, Acta Mech. Sin. **41**, 324140 (2025).
60 P.-Y. Chuang, and L. A. Barba, arXiv: 2205.14249.
61 W. Cao, and W. Zhang, arXiv: 2310.16491.
62 F. Pioch, J. H. Harmening, A. M. Müller, F. J. Peitzmann, D. Schramm, and O. el Moctar, Fluids **8**, 43 (2023).
63 Z.-Q.J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma, arXiv: 1901.06523.
64 S. Wang, H. Wang, and P. Perdikaris, Comput. Methods Appl. Mech. Eng. **384**, 113938 (2021).
65 S. Wang, X. Yu, and P. Perdikaris, J. Comput. Phys. **449**, 110768 (2022).
66 A. A. Ramabathiran, and P. Ramachandran, J. Comput. Phys. **445**, 110600 (2021).
67 S. Wang, S. Sankaran, and P. Perdikaris, Comput. Methods Appl. Mech. Eng. **421**, 116813 (2024).
68 X. Huang, H. Liu, B. Shi, Z. Wang, K. Yang, Y. Li, M. Wang, H. Chu, J. Zhou, F. Yu, et al., in *A universal pinns method for solving partial differential equations with a point source: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, Vienna, 2022, pp. 3839-3846.
69 R. Bischof, and M. Kraus, arXiv: 2110.09813.
70 S. Wang, Y. Teng, and P. Perdikaris, SIAM J. Sci. Comput. **43**, A3055 (2021).
71 A. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby, and M. W. Mahoney, arXiv: 2109.01050.
72 S. Basir, and I. Senocak, arXiv: 2206.09961.
73 T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, arXiv: 2001.06782.
74 Z. Xiang, W. Peng, X. Liu, and W. Yao, Neurocomputing **496**, 11 (2022).
75 Y. Wang, Y. Yao, J. Guo, and Z. Gao, J. Comput. Phys. **510**, 113112 (2024).
76 L. D. McClenny, and U. M. Braga-Neto, J. Comput. Phys. **474**, 111722 (2023).
77 Y. Gu, H. Yang, and C. Zhou, J. Comput. Phys. **441**, 110444 (2021).
78 W. Li, C. Zhang, C. Wang, H. Guan, and D. Tao, arXiv: 2205.08754.
79 J. Berg, and K. Nyström, Neurocomputing **317**, 28 (2018).
80 H. Sheng, and C. Yang, J. Comput. Phys. **428**, 110085 (2021).
81 H. Sheng, and C. Yang, arXiv: 2205.00593.
82 S. Liu, Z. Hao, C. Ying, H. Su, J. Zhu, and Z. Cheng, arXiv: 2210.03526.
83 C. Rao, P. Ren, Q. Wang, O. Buyukozturk, H. Sun, and Y. Liu, Nat. Mach. Intell. **5**, 765 (2023).
84 H. Gao, M. J. Zahr, and J. X. Wang, Comput. Methods Appl. Mech. Eng. **390**, 114502 (2022).
85 W. Cao, J. Song, and W. Zhang, arXiv: 2401.07203.
86 Z. Liu, W. Cai, and Z.-Q.J. Xu, arXiv: 2007.11207.
87 N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, in *On the spectral bias of neural networks: Proceedings of the 36th International Conference on Machine Learning*, Long Beach, 2019, pp. 5301-5310.
88 W. Cao, J. Song, and W. Zhang, Phys. Fluids **36**, 027134 (2024).
89 J. Wu, Y. Wu, G. Zhang, and Y. Zhang, J. Comput. Phys. **500**, 112761 (2024).
90 P. Rathore, W. Lei, Z. Frangella, L. Lu, and M. Udell, arXiv: 2402.01868.
91 T. de Ryck, F. Bonnet, S. Mishra, and E. de Bézenac, arXiv: 2310.05801.
92 S. Liu, C. Su, J. Yao, Z. Hao, H. Su, Y. Wu, and J. Zhu, arXiv: 2402.00531.
93 W. Cao, and W. Zhang, J. Comput. Phys. **520**, 113494 (2025).
94 B. Chartres, and R. Stepleman, SIAM J. Numer. Anal. **9**, 476 (1972).
95 A. M. Rogerson, and E. Meiburg, J. Sci. Comput. **5**, 151 (1990).
96 P. Morin, R. H. Nochetto, and K. G. Siebert, SIAM Rev. **44**, 631 (2002).
97 F. Coquel, and P. Le Floch, SIAM J. Numer. Anal. **30**, 675 (1993).
98 S. Zhang, J. Zhu, and C. W. Shu, Adv. Aerodyn. **1**, 16 (2019).
99 B. Cockburn, F. Coquel, and P. G. LeFloch, SIAM J. Numer. Anal. **32**, 687 (1995).
100 E. Tadmor, SIAM J. Numer. Anal. **26**, 30 (1989).

101 Y. Shin, J. Darbon, and G. E. Karniadakis, arXiv: 2004.01806.
102 K. Hornik, M. Stinchcombe, and H. White, Neural Networks **2**, 359 (1989).
103 P. Pantidis, H. Eldababy, C. M. Tagle, and M. E. Mobasher, Comput. Methods Appl. Mech. Eng. **414**, 116160 (2023).
104 Y. Shin, Z. Zhang, and G. E. Karniadakis, J. Mach. Learn Model. Comput. **4**, 73 (2023).
105 Y. Jiao, Y. Lai, D. Li, X. Lu, F. Wang, Y. Wang, and J. Z. Yang, arXiv: 2109.01780.
106 S. Mishra, and R. Molinaro, IMA J. Numer. Anal. **42**, 981 (2022).
107 T. De Ryck, A. D. Jagtap, and S. Mishra, IMA J. Numer. Anal. **44**, 83 (2024).
108 F. Stern, R. V. Wilson, H. W. Coleman, and E. G. Paterson, in *Verification and validation of CFD simulations: Proceedings of the 1999 3rd ASME/JSME Joint Fluids Engineering Conference, FEDSM'99*, San Francisco, 1999, pp. 18-23.
109 J. C. Wu, AIAA J. **14**, 1042 (1976).
110 A. Bayliss, and E. Turkel, J. Comput. Phys. **48**, 182 (1982).
111 F. Moukalled, L. Mangani, and M. Darwish, *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with Openfoam and Matlab* (Springer, Cham, 2016).
112 D. Ma, Y. Zhao, Y. Qiao, and G. Li, Chin. J. Aeronaut. **28**, 1003 (2015).
113 M. A. Ashraf, J. Young, and J. C. S. Lai, J. Fluids Struct. **27**, 145 (2011).
114 M. J. Berger, and P. Colella, J. Comput. Phys. **82**, 64 (1989).
115 R. V. Garimella, and M. S. Shephard, Int. J. Numer. Meth. Eng. **49**, 193 (2000).
116 C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu, Comput. Methods Appl. Mech. Eng. **403**, 115671 (2023).
117 L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, SIAM Rev. **63**, 208 (2021).
118 S. Zeng, Z. Zhang, and Q. Zou, J. Comput. Phys. **463**, 111232 (2022).
119 S. Wang, H. Wang, and P. Perdikaris, Sci. Adv. **7**, eabi8605 (2021).
120 W. Cao, S. Tang, Q. Ma, W. Ouyang, and W. Zhang, arXiv: 2501.01165.
121 W. Cao, J. Song, and W. Zhang, J. Comput. Phys. **516**, 113285 (2024).
122 S. Wang, S. Sankaran, and P. Perdikaris, arXiv: 2203.07404.
123 Y. He, Z. Wang, H. Xiang, X. Jiang, and D. Tang, Appl. Math. Mech.-Eng. Ed. **44**, 1101 (2023).
124 Z. Wang, M. S. Triantafyllou, Y. Constantinides, and G. Em Karniadakis, J. Fluid Mech. **859**, 691 (2019).
125 Z. Zhang, X. Su, and X. Yuan, Physics-informed neural networks with coordinate transformation to solve high Reynolds number boundary layer flows, 2024, https://ssrn.com/abstract=4866658 or http://dx.doi.org/10.2139/ssrn.4866658.
126 L. Zhong, B. Wu, and Y. Wang, J. Phys. D-Appl. Phys. **56**, 074006 (2023).
127 C. Song, T. Alkhalifah, and U. B. Waheed, Geophys. J. Int. **228**, 1750 (2022).
128 M. H. Saadat, B. Gjorgiev, L. Das, and G. Sansavini, arXiv: 2211.11716.
129 L. Zhong, B. Wu, and Y. Wang, Phys. Fluids **34**, 087116 (2022).
130 M. Aliakbari, M. Mahmoudi, P. Vadasz, and A. Arzani, Int. J. Heat Fluid Flow **96**, 109002 (2022).
131 C. Davi, and U. Braga-Neto, arXiv: 2202.01943.
132 X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, Comput. Methods Appl. Mech. Eng. **370**, 113250 (2020).
133 C. Amalinadhi, P. S. Palar, R. Stevenson, and L. Zuhal, in *On physics-informed deep learning for solving navier-stokes equations: Proceedings of the 2022 AIAA Science and Technology Forum and Exposition (AIAA SciTech Forum)*, San Diego, 2022, p. 1436.
134 E. H. W. Ang, G. Wang, and B. F. Ng, Energies **16**, 4558 (2023).
135 Q. Zhang, X. Guo, X. Chen, C. Xu, and J. Liu, Int. J. Mod. Phys. C **33**, 2250166 (2022).
136 X. Chen, C. Gong, Q. Wan, L. Deng, Y. Wan, Y. Liu, B. Chen, and J. Liu, Adv. Aerodyn. **3**, 36 (2021).
137 J. C. Wong, C. C. Ooi, A. Gupta, and Y. S. Ong, IEEE Trans. Artif. Intell. **5**, 985 (2022).
138 V. Dwivedi, N. Parashar, and B. Srinivasan, arXiv: 1907.08967.
139 Y. Huang, Z. Zhang, and X. Zhang, Fluids **7**, 56 (2022).
140 Y. Wang, and L. Zhong, J. Comput. Phys. **496**, 112603 (2024).
141 R. Laubscher, Phys. Fluids **33**, 087101 (2021).
142 K. L. Lim, R. Dutta, and M. Rotaru, in *Physics informed neural network using finite difference method: Proceedings of the 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Prague, 2022, pp. 1828-1833.
143 S. Zhang, C. Zhang, X. Han, and B. Wang, Comput. Mech. **75**, 1137 (2025).
144 M. Penwarden, A. D. Jagtap, S. Zhe, G. E. Karniadakis, and R. M. Kirby, J. Comput. Phys. **493**, 112464 (2023).
145 A. F. Ihsan, and P. H. Gunawan, in *Performance analysis of the neural network solution of advection-diffusion-reaction problem: Proceedings of the 2022 5th International Conference on Information and Communications Technology (ICOIACT)*, Yogyakarta, 2022, pp. 238-243.
146 L. Liu, S. Liu, H. Xie, F. Xiong, T. Yu, M. Xiao, L. Liu, and H. Yong, J. Sci. Comput. **98**, 22 (2024).
147 R. Mojgani, M. Balajewicz, and P. Hassanzadeh, Comput. Methods Appl. Mech. Eng. **404**, 115810 (2023).
148 M. Mahmoudabadbozchelou, G. E. Karniadakis, and S. Jamali, Soft Matter **18**, 172 (2022).
149 Y. Chen, and S. Koohy, Finite Elem. Anal. Des. **228**, 104047 (2024).
150 J. Cho, S. Nam, H. Yang, S.-B. Yun, Y. Hong, and E. Park, arXiv: 2211.08761.
151 Y. Liu, L. Chen, and J. Ding, in *Grad-rar: An adaptive sampling method based on residual gradient for physical-informed neural networks: Proceedings of the 2022 International Conference on Automation, Robotics and Computer Engineering (ICARCE)*, Wuhan, 2022, pp. 1-5.
152 X. Huang, and T. Alkhalifah, J. Comput. Phys. **501**, 112760 (2024).
153 A. Sagar, Aa-pinn: Attention augmented physics informed neural networks, 2022, https://openreview.net/forum?id=Aot3sKdraW.
154 J. C. Wong, P.-H. Chiu, C. Ooi, M. Ha Dao, and Y.-S. Ong, in *Lsa-pinn: Linear boundary connectivity loss for solving PDEs on complex geometry: Proceedings of the 2023 International Joint Conference on Neural Networks (IJCNN)*, Gold Coast, 2023, pp. 1-10.
155 R. Laubscher, P. Rousseau, and C. Meyer, in *Modeling of inviscid flow shock formation in a wedge-shaped domain using a physics-informed neural network-based partial differential equation solver: Proceedings of the ASME Turbo Expo 2022: Turbomachinery Technical Conference and Exposition. Volume 10C: Turbomachinery—Design Methods and CFD Modeling for Turbomachinery; Ducts, Noise, and Component Interactions*, Rotterdam, 2022.
156 J. Kim, K. Lee, D. Lee, S. Y. Jhin, and N. Park, AAAI **35**, 8146 (2021).
157 R. Laubscher, and P. Rousseau, Appl. Soft Comput. **114**, 108050 (2022).
158 Z. Xiang, W. Peng, W. Zhou, and W. Yao, arXiv: 2202.07926.
159 R. J. Gladstone, M. A. Nabian, N. Sukumar, A. Srivastava, and H. Meidani, arXiv: 2210.14320.
160 G. M. Gie, Y. Hong, and C. Y. Jung, Appl. Anal. **103**, 2554 (2024).
161 Y. Guo, X. Cao, B. Liu, and M. Gao, Appl. Sci. **10**, 5917 (2020).
162 U. B. Waheed, IEEE Geosci. Remote Sens. Lett. **19**, 1 (2022).
163 X. Huang, T. Alkhalifah, and C. Song, in *A modified physics-informed neural network with positional encoding: Proceedings of the SEG Technical Program Expanded Abstracts*, Denver, 2021, pp. 2480-2484.
164 R. Zhang, P. Hu, Q. Meng, Y. Wang, R. Zhu, B. Chen, Z.-M. Ma, and T.-Y. Liu, arXiv: 2206.09571.
165 J. Li, L. Zhou, J. Sun, and G. Sun, in *Physically plausible and conservative solutions to navier-stokes equations using physics-*

informed CNNs: ICLR 2023 Conference, Kigali, 2023, p. 1.
166 X. Li, Y. Liu, and Z. Liu, Phys. Fluids **35**, 063608 (2023).
167 A. Arzani, K. W. Cassel, and R. M. D′Souza, J. Comput. Phys. **473**, 111768 (2023).
168 C. L. Pettit, and D. K. Wilson, Proc. Mtgs. Acoust. **42**, 022002 (2020).
169 J. C. Wong, P.-H. Chiu, C. C. Ooi, M. H. Dao, and Y.-S. Ong, in *Fast-PINN for complex geometry: Solving PDEs with boundary connectivity loss: Proceedings of the ICLR 2023 Conference*, Kigali, 2023, p. 1.
170 N. Navaneeth, and S. Chakraborty, Comput. Methods Appl. Mech. Eng. **406**, 115842 (2023).
171 Q. Zeng, Y. Kothari, S.r H. Bryngelson, and F. Schäfer, arXiv: 2204.11144.
172 L. Liu, Enhancing the shock-capturing ability of physics-informed neural network by weakening its expression near shocks, 2023, https://doi.org/10.36227/techrxiv.19391279.v2.
173 P. Ren, C. Rao, Y. Liu, J. X. Wang, and H. Sun, Comput. Methods Appl. Mech. Eng. **389**, 114399 (2022).
174 A. Farkane, M. Ghogho, M. Oudani, and M. Boutayeb, arXiv: 2304.03689.
175 N. Sung, J. C. Wong, P.-H. Chiu, A. Gupta, and C. Ooi, in *Yew-Soon Ong Neuroevolution of physics-informed neural nets: Benchmark problems and comparative results*: *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, Lisbon, 2023, pp. 2144-2151.
176 H. Huang, Y. Liu, and V. Yang, arXiv: 2109.09316.
177 H. Wang, X. Qian, Y. Sun, and S. Song, A modified physics informed neural networks for solving the partial differential equation with conservation laws, 2022, https://ssrn.com/abstract=4274376.

Appendix

Referring to ref. [51], we conducted our analysis based on the 59 articles randomly selected in that study on solving fluid-related PDEs using the PINN method (originally 60 articles, but our screening revealed one duplicate, appearing as both a preprint and a postprint). Our investigation focused on whether these articles mentioned the accuracy of PINNs and whether they provided a comparison with numerical methods, as well as whether they discussed the computational cost of PINNs and whether they compared it to numerical methods. Our method for identifying mentions of accuracy and computational cost in the articles involved first checking the abstract for relevant descriptions, then reviewing the conclusion, and finally examining all figures and tables in the article. Referring to ref. [51], we introduce the criterion for determining whether a comparison between PINNs and numerical methods is provided: the difference in computational cost required to achieve the same solution accuracy or the difference in solution accuracy under the same computational cost. Although some studies provide comparisons between PINNs and numerical methods, in these cases, the numerical methods just serve as reference solutions to evaluate the accuracy differences between PINNs and their improved versions (e.g., ref. [126]). Therefore, we do not consider such comparisons to meet our criterion. Additionally, in some articles, no quantitative comparison of accuracy and computational cost between PINNs and numerical solutions is provided; instead, only qualitative assessments are given. For example, ref. [127] states, “Currently, the training cost of PINN in solving the 2-D Helmholtz equation is higher than the numerical method.” We have also included such evaluations as supplementary information in our analysis. Our statistical results are shown in Tables a1 and a2, from which it can be inferred that:

(1) Although most of the selected articles mentioned the accuracy and computational cost of PINNs (100% of the selected articles mentioned accuracy, and nearly 50% mentioned computational cost), only a small number of articles provided comparative data on the accuracy and computational cost of PINNs versus numerical methods (1.7% compared accuracy, and 8.5% compared computational cost). Additionally, 18.6% of the articles included qualitative assessments of the accuracy and computational cost of PINNs relative to numerical methods.

(2) Based on the quantitative and qualitative comparisons between PINNs and numerical methods, at the current stage, PINNs are still inferior to numerical methods in terms of solution accuracy and cost for forward problems.

(3) In the selected articles, many studies use numerical solutions as reference solutions, which indirectly indicate that numerical methods have a greater ability to achieve higher accuracy compared to PINNs.

(4) As stated in the first inference, only a small portion of the selected articles reported comparisons between PINNs and numerical methods, which also demonstrate the reporting biases mentioned in ref. [51]. Improvements to PINNs over the vanilla version are more likely to yield positive results and are therefore widely reported, whereas the persistent shortcomings of PINNs in accuracy and computational cost compared to numerical methods are negative results that are reported rarely.

Table a1 Statistical results of the accuracy and computational cost comparison between PINNs and numerical methods

Article	Reports accuracy	Compares computational accuracy of PINNs with numerical methods	Reports computational cost	Compares computational cost of PINNS with numerical methods	Comments on computational accuracy or cost of PINNS with numerical methods
1. Saadat et al. [128]	√	×	×	×	×
2. Zhong et al. [126]	√	×	√	×	√
3. Rao et al. [23]	√	×	×	×	×
4. Zhong et al. [129]	√	×	×	×	×
5. Chiu et al. [57]	√	×	√	×	×
6. Aliakbari et al. [130]	√	×	√	√	×
7. Chuang and Barba [60]	√	√	√	√	×
8. Song et al. [127]	√	×	√	×	√
9. Davi and Braga-Neto [131]	√	×	×	×	×
10. Meng et al. [132]	√	×	√	×	×
11. Amalinadhi et al. [133]	√	×	√	×	×
12. Ang et al. [134]	√	×	√	√	×
13. Zhang et al. [135]	√	×	×	×	×
14. Chen et al. [136]	√	×	×	×	×
15. Wong et al. [137]	√	×	×	×	√
16. Krishnapriyan et al. [71]	√	×	×	×	×
17. Dwivedi et al. [138]	√	×	×	×	×
18. Huang et al. [139]	√	×	×	×	×
19. Wang and Zhong [140]	√	×	√	×	√
20. Laubscher [141]	√	×	×	×	√
21. Lim et al. [142]	√	×	√	×	×
22. Zhang et al. [143]	√	×	√	×	×
23. Penwarden et al. [144]	√	×	√	×	×
24. Ihsan and Gunawan [145]	√	×	√	√	×
25. Liu et al. [146]	√	×	×	×	×
26. Mojgani et al. [147]	√	×	×	×	×
27.Mahmoudabadbozchelou et al. [148]	√	×	√	×	×
28. Chen and Koohy [149]	√	×	√	×	√
29. Cho et al. [150]	√	×	√	×	√
30. Liu et al. [151]	√	×	×	×	×
31. Huang and Alkhalifah [152]	√	×	√	×	√
32. Sagar [153]	√	×	√	×	×
33. Wong et al. [154]	√	×	√	×	×
34. Laubscher et al. [155]	√	×	×	×	×
35. Kim et al. [156]	√	×	×	×	×
36. Laubscher and Rousseau [157]	√	×	√	√	×
37. Xiang et al. [158]	√	×	×	×	×
38. Gladstone et al. [159]	√	×	×	×	×
39. Gie et al. [160]	√	×	×	×	×
40. Sun et al. [49]	√	×	√	×	√
41. Guo et al. [161]	√	×	√	×	×
42. Waheed [162]	√	×	×	×	√
43. Huang et al. [163]	√	×	×	×	×
44. Zhang et al. [164]	√	×	√	×	×
45. Li et al. [165]	√	×	×	×	×
46. Li et al. [166]	√	×	√	×	×

(To be continued on the next page)

(Continued)

Article	Reports accuracy	Compares computational accuracy of PINNs with numerical methods	Reports computational cost	Compares computational cost of PINNS with numerical methods	Comments on computational accuracy or cost of PINNS with numerical methods
47. Arzani et al. [167]	√	×	×	×	×
48. Pettit and Wilson [168]	√	×	×	×	×
49. Wong et al. [169]	√	×	×	×	×
50. Navaneeth and Chakraborty [170]	√	×	√	×	×
51. Zeng et al. [171]	√	×	×	×	√
52. McClenny and Braga-Neto [76]	√	×	√	×	×
53. Liu et al. [172]	√	×	×	×	×
54. Ren et al. [173]	√	×	√	×	×
55. Farkane et al. [174]	√	×	√	×	×
56. Gao et al. [55]	√	×	√	×	×
57. Sung et al. [175]	√	×	×	×	×
58. Huang et al. [176]	√	×	×	×	×
59. Wang et al. [177]	√	×	×	×	×
	59/59 (100%)	1/59 (1.69%)	29/59 (49.2%)	5/59 (8.47%)	11/59 (18.6%)

Table a2 Statistical results of the numerical methods used, comparison outcomes, and comments about accuracy and cost

Article	Numerical methods	Comparison outcomes about accuracy and cost	Comments about accuracy and cost
Zhong et al. [126]	–	–	Training of PINNs is sometimes very time-consuming.
Aliakbari et al. [130]	FEniCS (finite element solver) and Fluent (finite volume solver)	In solving 2D steady-state problems with simple geometries, PINN training is still significantly slower than CFD modeling (approximately 1-2 d *vs*. 20 min for the results in the current study).	–
Chuang and Barba [60]	PetIBM (a traditional solver using staggered-grid finite difference methods with MPI parallelization and GPU computing)	For the 2D Taylor-Green vortex problem, about 32 h of training were required for the PINN method's accuracy (used 18432 points in each training iteration) to match the accuracy of a 16 × 16 finite difference simulation, which took less than 20 s. While in the 2D cylinder flow, The PINN method behaved like a steady-flow solver and did not capture the vortex-shedding phenomenon.	–
Song et al. [127]	–	–	Currently, the training cost of PINN in solving the 2-D Helmholtz equation is higher than the numerical method.
Ang et al. [134]	simpleFoam (Semi-Implicit Method for Pressure Linked Equations (SIMPLE) algorithm)	PINN is more efficient in memory usage than CFD (requiring 5-10 times less memory) but not necessary for computation time (requiring approximately 3 times more than that of CFD). In simple cases where the number of cells in the CFD mesh is low, PINN takes longer than CFD to compute the results. However, in more complex cases that involve a higher number of cells in the CFD mesh, PINN has the potential to be more time-efficient.	–

(To be continued on the next page)

(Continued)

Article	Numerical methods	Comparison outcomes about accuracy and cost	Comments about accuracy and cost
Wong et al. [137]	–	–	PINN is yet to become feasible for broad science and engineering applications due to expensive computational costs and training challenges.
Wang and Zhong [140]	–	–	The design of the PINNs structure basically relies on prior knowledge and experience, which has caused great trouble and high computational overhead.
Laubscher [141]	–	–	Both the single- and segregated-network PINN models produced accurate results for temperature and velocity profiles, with an average percentage difference relative to the CFD results of approximately 7.5% for velocity and 8% for temperature. The mean error percentages for the species mass fractions are 9% for the single-network model and 1.5% for the segregated-network approach.
Ihsan and Gunawan [145]	standard finite difference (FD) scheme	Compared to numerical simulation which only needs 6 s to compute around 1000 × 2000 grids of the domain, PINN is much slower than the existing numerical method.	–
Chen and Koohy [149]	–	–	PINN's training is still time-consuming.
Cho et al. [150]	–	–	For PINNs, the expensive computational costs to obtain solutions often restrict their broader applicability.
Huang and Alkhalifah [152]	–	–	PINN's training cost has limited their practical use as a real alternative to classic numerical methods.
Laubscher and Rousseau [157]	ANSYS Fluent (using the finite volume CFD approach)	PINN takes significantly longer to solve than the traditional finite volume numerical methods utilized in commercial CFD software. For example, the steady-state CFD model solved within 5 min using an i7-7700K quad-core processor whereas the PINN method took approximately an hour to train using a Nvidia RTX 2080 GPU card.	–
Sun et al. [49]	–	–	By comparison with conventional CFD, the velocity and pressure fields solved by PINNs have small point-wise errors.
Waheed [162]	–	–	For the PINN-based Helmholtz solvers, their large training cost has been the main impediment in their widespread adoption for wavefield computations.
Zeng et al. [171]	–	–	PINNs currently cannot produce high-accuracy solutions, typically attaining about 0.1% relative error.