State-of-the-art implementations of PINNs for the lid-driven cavity problem – a critical review with future perspectives

Results in Engineering 32 (2026) 112399

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering

Review article

State-of-the-art implementations of PINNs for the lid-driven cavity problem – a critical review with future perspectives

Mohammad Sheikholeslami [a,*], Saeed Salehi [b], Wengang Mao [a], Arash Eslamdoost [a], Håkan Nilsson [a]

[a] Department of Mechanical Engineering, Chalmers University of Technology, Gothenburg, Sweden
[b] Department of Management and Engineering, Linköping University, Linköping, Sweden

ARTICLE INFO

Keywords:
Physics-informed neural networks (PINNs)
Lid-driven cavity (LDC)
Incompressible flow
Fluid dynamics

ABSTRACT

The lid-driven cavity (LDC) problem is a canonical benchmark for incompressible flow. Owing to the corner boundary discontinuities, multiscale vortical structures, possible multiplicity of steady solutions, and the change of flow properties at different Reynolds numbers, LDC provides a stringent test for numerical methods. This review surveys the application of physics-informed neural networks (PINNs) to LDC. The literature is organized around four main classes of design choices, namely LDC formulation for PINNs, PINNs architecture, optimization and training strategy, and loss function, evaluation, and some selected results. Within each class, the review examines the literature, then presents insights drawn from the reviewed studies and introduces the open questions and suggestions for future research associated with each design aspect. The review indicates that transfer learning, domain decomposition, special training strategies, discretization-based residual evaluation, and optimization modifications can be interpreted as a spectrum of approaches that differ in implementation difficulty and resource requirements when improving PINNs performance on LDC. Furthermore, the review shows that comparisons across studies remain difficult because architectures, boundary treatments, and evaluation metrics vary widely, while ablation studies are often lacking.

1. Introduction

Modeling incompressible flow is essential for understanding a wide range of natural and engineered systems. Benchmark problems in this field serve not only as validation tools for numerical solvers but also as stepping stones for developing new computational methods. Among these benchmarks, the lid-driven cavity problem (LDC) is widely recognized due to its simple geometry and rich flow features, including boundary layers, secondary vortices, turbulence and pressure gradients, which cover almost all the features that incompressible flow can represent [1]. Further, LDC has a closed domain, which facilitates stringent verification of mass conservation [2]. This problem also has significant practical importance, as it exists in short-dwell and flexible blade coaters and mixing technologies [3,4]. LDC has been solved in both 2D [5] and 3D [6] configurations. This problem has also been explored experimentally by involving a small amount of throughflow to the system [3]. The problem can be defined as single-LDC, which has only one moving wall [7], and double-LDC (also called two-sided LDC) [8] where two walls move in opposite directions.

Despite its simple geometry, LDC exhibits nontrivial dynamics and remains challenging for numerical modeling. The governing equations are nonlinear and coupled through the incompressibility constraint, so any numerical method must satisfy momentum balance and mass conservation simultaneously. A further difficulty is the discontinuity in the boundary condition at the top corners of the domain [9], where the horizontal velocity component decreases abruptly from a nonzero value along the moving wall to zero on the adjacent stationary walls. At higher Reynolds numbers, the flow may also develop thin boundary layers, secondary vortices, and multiple steady solutions [10,11], which increase sensitivity to discretization, initialization, and solution strategy. Together, these features make LDC a stringent and informative benchmark for numerical methods.

Physics-informed neural networks (PINNs) have emerged as a powerful framework for solving partial differential equations [12], to identify parameters [13], and to assimilate data [14]. This development builds on decades of research on neural network approaches to differential equations dating back to the 1990s [15]. By embedding the governing equations, boundary conditions, and initial conditions into the

* Corresponding author.
E-mail addresses: mohshe@chalmers.se (M. Sheikholeslami), saeed.salehi@liu.se (S. Salehi), wengang.mao@chalmers.se (W. Mao), arash.eslamdoost@chalmers.se (A. Eslamdoost), hakan.nilsson@chalmers.se (H. Nilsson).

https://doi.org/10.1016/j.rineng.2026.112399
Received 3 June 2026; Received in revised form 5 August 2026; Accepted 8 August 2026
Available online 11 August 2026

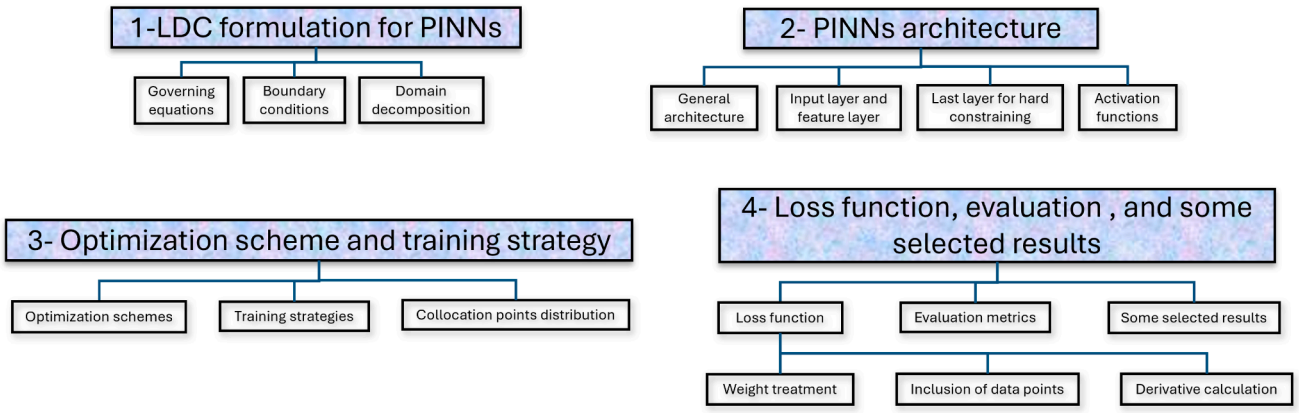


Fig. 1. Overview of the topics covered in this review on PINNs for LDC.

loss function of neural networks, PINNs enable mesh-free computations [16].

LDC is particularly valuable for assessing PINN methods because it is simple to formulate but challenging to solve accurately with neural networks. This is evident from the growing number of studies summarized in Table 1. However, despite the large number of individual studies, there is still no dedicated review of PINN methods for LDC. The aim of the present study is therefore to identify the current state of the art in this area. As illustrated in Fig. 1, we organize the main design choices in PINN-based LDC modeling into four groups, namely LDC formulation for PINNs, neural network design, optimization and training strategy, and loss function design. We then compare the existing literature through this framework in order to clarify the main trends, strengths, and unresolved challenges. The term LDC in the paper refers to 2D steady LDC, unless otherwise stated.

2. Physics-informed neural network and the review methodology

PINNs can approximate unknown fields using neural networks and enforce governing equations by penalizing residuals computed via automatic differentiation (AD). A schematic workflow of a basic PINN is shown in Fig. 2. In this framework, the solution map is represented as

$$f_\theta : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}, \qquad \mathbf{x} \mapsto \mathbf{u}(\mathbf{x}), \tag{1}$$

where θ denotes trainable parameters, $\mathbf{x}$ represents the input coordinates (e.g., spatial and temporal variables), and $\mathbf{u}$ represents the solution fields. For a set of N interior training points $\{\mathbf{x}_i\}_{i=1}^{N}$, the network is evaluated in batch as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} \in \mathbb{R}^{N\times d_{\text{in}}}, \qquad F_\theta(\mathbf{X}) = \begin{bmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_N^T \end{bmatrix} \in \mathbb{R}^{N\times d_{\text{out}}}, \tag{2}$$

where $\mathbf{u}_i = f_\theta(\mathbf{x}_i)$. Derivatives required by the governing equations (e.g., $\partial\mathbf{u}/\partial\mathbf{x}, \partial^2\mathbf{u}/\partial\mathbf{x}^2$) are computed by differentiating network outputs with respect to the input coordinates. Unlike numerical differentiation on a discrete grid, AD utilizes the computational graph of the neural network to apply the chain rule iteratively.

For a general non-linear partial differential equation defined on the domain Ω, the PDE residual is defined via a general differential operator $\mathcal{N}[\cdot]$ on Ω. The residual $r_f(\mathbf{x})$ is given by

$$r_f(\mathbf{x}) = \mathcal{N}[\mathbf{u}](\mathbf{x}) \quad \mathbf{x} \in \Omega. \tag{3}$$

Also, a boundary operator $\mathcal{B}$ with prescribed data g defines the boundary residual r_b on the boundaries $\partial\Omega$. This residual is given by

$$r_b(\mathbf{x}) = \mathcal{N}[\mathbf{u}](\mathbf{x}) \quad \mathbf{x} \in \partial\Omega. \tag{4}$$

Ideally, the exact solution satisfies $r_f(\mathbf{x}) = 0$ in Ω and $r_b(\mathbf{x}) = 0$ on the boundaries. For time-dependent problems, it also satisfies the initial-condition residual $r_i(\mathbf{x}) = 0$ at the initial time. The PINN framework enforces these constraints by minimizing the mean squared error (MSE) of the residuals over the corresponding collocation points. The PDE loss is defined as

$$\mathcal{L}_{\text{PDE}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(r_f(\mathbf{x}_i)\right)^2, \tag{5}$$

where N is the number of collocation points in the domain. The boundary loss $\mathcal{L}_{\text{BC}}$ is evaluated at a set of N_b boundary points $\{\mathbf{x}_j^b\}_{j=1}^{N_b}$ where Dirichlet or Neumann conditions are known, as

$$\mathcal{L}_{\text{BC}}(\theta) = \frac{1}{N_b}\sum_{j=1}^{N_b}\left(r_b(\mathbf{x}_j^b)\right)^2. \tag{6}$$

For unsteady problems, the initial-condition loss is evaluated at N_i points $\{\mathbf{x}_j^i\}_{j=1}^{N_i}$ at the initial time, as

$$\mathcal{L}_{\text{IC}}(\theta) = \frac{1}{N_i}\sum_{j=1}^{N_i}\left(r_i(\mathbf{x}_j^i)\right)^2. \tag{7}$$

If experimental data, numerical reference data, or sparse measurements are available, a data loss, $\mathcal{L}_{\text{data}}(\theta)$, can also be included.

The total loss function, $\mathcal{L}_{\text{total}}$, is then formulated as a weighted sum of these components, as

$$\mathcal{L}_{\text{total}}(\theta) = w_{\text{PDE}}\mathcal{L}_{\text{PDE}}(\theta) + w_{\text{BC}}\mathcal{L}_{\text{BC}}(\theta) + w_{\text{IC}}\mathcal{L}_{\text{IC}}(\theta) + w_{\text{data}}\mathcal{L}_{\text{data}}(\theta), \tag{8}$$

where w_{PDE}, w_{BC}, w_{IC}, and w_{data} are non-negative hyperparameters that balance the contributions of the governing equations, boundary conditions, initial conditions, and data constraints. Terms that are not relevant to a given problem, such as $\mathcal{L}_{\text{IC}}$ for a steady-state formulation or $\mathcal{L}_{\text{data}}$ when no data are available, are omitted. Finally, the optimal network parameters are obtained by minimizing the total loss, as

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}_{\text{total}}(\theta). \tag{9}$$

The optimization is typically performed using gradient-based optimizers. Further details on PINNs can be found in reviews [91,92] and references within them.

2.1. Review methodology

The literature considered in this review was identified through structured searches of Scopus, Web of Science, and Google Scholar, supplemented by backward and forward citation tracking. The search covered relevant studies published up to May 2026 and used broad combinations of terms related to physics-informed neural networks and lid-driven cavity flow. Studies were included when they applied PINNs to the lid-driven cavity problem. Studies were excluded when they addressed conventional CFD without a direct connection to physics-informed learning

Table 1
Application of PINN for LDC in the literature.

Study	Re	Data	Optimizer	Epochs	Activation function	Collocation points	Network
Lid velocity treatment (soft and hard)							
Służalec et al. [17]	1, 10, 100, 1000	No	–	up to 1.0×10^5	–	–	MLP
McDevitt et al. [18]	100, 400, 1000, 2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000, 21000	Yes/No	Adam + L-BFGS-B	2.15×10^5- 8.0×10^5	tanh	5×10^4 2.5×10^5	MLP
Pal et al.[P] [19]	100, 400, 1000	No	Adam	3.0×10^5	swish	1.0×10^4	MLP modified
Chen et al. [20]	–	No	Adam	8.0×10^4	tanh	5.12×10^2– 4.096×10^3	MLP modified
Lin et al. [21]	1000, 2000, 3000, 5000	No	Adam	8.0×10^4- 2.5×10^5	tanh, ReLU	$961 - 10201$	MLP + DeepONet
Sababha et al. [22]	25, 500, 5000	No	Adam + L-BFGS-B	–	tanh	4.0×10^4	MLP
Li et al. [23]	100	No	Adam	2.0×10^4	tanh	$\approx 5.2 \times 10^3$	MLP modified
Lou et al. [24]	100	No	Adam + L-BFGS-B	$> 1.0 \times 10^5$	tanh	46058	MLP
Hao et al. [25]	100	No	Adam, L-BFGS, MultiAdam	2.0×10^4		1.024×10^4	MLP
Ding et al. [26]	100	No	Adam + L-BFGS	1.2×10^4		10^3	MLP
Urbán et al. [27]	1000	No	BFGS, self-scaled Broyden	2.0×10^4	tanh	2.5×10^4	MLP
Shukla et al. [28]	400, 2000	No	Adam	4.0×10^5– 9.0×10^5	tanh	1.0×10^4– 2.0×10^4	MLP, KAN
Inclusion of data points							
Geng et al. [29]	100	Yes	Adam	4.0×10^5	–	–	MLP modified
Sun et al. [30]	100, 400, 1000	Yes	Adam	1.0×10^4	–	2.601×10^3	MLP
Satyadharma et al. [31]	100, 200, 500, 1000, 2000, 5000	Yes	Adam + L-BFGS-B	$> 4.0 \times 10^4$	tanh	$< 2.5 \times 10^5$ random sampling with Latin hypercube	MLP modified
Bai et al. [14]	100	Yes/No	–	–	sine		PINN
Feng et al. [32]	100, 400, 1000, 5000	Yes	Adam + L-BFGS	$> 3.0 \times 10^5$		1.0×10^4	MLP modified
Liu et al. [33]	100	Yes	–	3.0×10^5	–	–	MLP modified
Pandey et al. [34]	100, 400	Yes	Adam	1.0×10^4	tanh	4.2×10^4	MLP
Jangir et al.[P] [35]	10, 30, 50, 80, 100, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000, 1200, 1500, 2000	Yes/No	Adam + L-BFGS	5.0×10^4- 5.0×10^5	tanh	3.3×10^3, 5.2×10^3, 1.17×10^4	MLP modified
Ba et al. [36]	400, 1000, 3200	Yes/No	–	5.0×10^4	tanh	1.1×10^4	MLP modified
Zou et al. [37]	–	Yes	Adam	2.0×10^5	tanh	9.801×10^3	MLP
Domain decomposition							
Jagtap et al. [38]	100	No	Adam	4.5×10^6	adaptive tanh	$\approx 4.17 \times 10^4$	MLP
Shukla et al. [39]	100	No	Adam	–	tanh	various distributions	MLP modified
Gu et al. [40]	100, 400	No	Adam + L-BFGS-B	3.0×10^4	tanh	1.268×10^4	MLP
Khademi and Dufour [41]	100	No	Adam	$\lesssim 1.2 \times 10^4$ - $\gtrsim 3.5 \times 10^4$	tanh	$\approx 5.4 \times 10^3$ 2.08×10^4	modified MLP modified
Qian et al.[P] [42]	100	Yes	Adam	1.2×10^4	tanh	5.0×10^3	MLP
Roy et al.[P] [1]	400, 1000	No	L-BFGS	up to 2×10^5	tanh	6.25×10^2- 9×10^4	MLP
Pan and Xiao [43]	100–300	Yes	Adam	5.0×10^3- 1.5×10^4	tanh	1.0201×10^4	MLP

Note: [P] denotes a preprint. Unmarked entries are non-preprint sources, including journal articles, conference papers, and theses.

Table 1
Continued.

Study	Re	Data	Optimizer	Epochs	Activation function	Collocation points	Network
Less common activation functions							
Xiong et al. [44]	1000, 2500, 4000	No	L-BFGS	9.0×10^3- 1.0×10^4	learnable AF tanh,	–	MLP, KAN
Khademi and Dufour [45]	100, 3200	Yes/No	Adam	8×10^3	adaptive sine	4.5×10^3 – 8.0×10^3	MLP modified
Zhou et al.[P] [46]	–	Yes	Adam	5.0×10^4	Mish, tanh	–	MLP
Zhang et al. [47]	100	No	Adam	2.0×10^4	swish	1.6641×10^4	MLP
Chiu et al.[P] [48]	400, 3200, 7500, 10000, 20000	No	Adam	5.0×10^4 to 1.0×10^5	swish	1.0×10^4- 6.5536×10^4 441 velocity	modified MLP
Gao et al. [49]	100	No	Adam	3.3×10^4	ReLU	+121 pressure	PI-GGN/ GCN
Chen et al. [50]	100, 300, 600	No	Adam	2.0×10^4	swish	–	MLP
Yang et al. [51]	100, 500	No	Adam	2.0×10^4 – 1.0×10^5	swish	1.15×10^3 – 4.8×10^4	MLP
Loss calculation novelties							
Wei et al. [52]	100, 400, 1000	No	L-BFGS	1.5×10^6 up to	tanh	1.7184×10^4 – 6.6336×10^4	MLP modified
Wei et al. [53]	1000, 3200	No	L-BFGS	1.5×10^6	tanh	4.74×10^4	MLP
Wei et al. [54]	100, 400, 1000, 3200, 5000, 7500, 10000	No	Adam	5.0×10^4	swish	2.601×10^3 – 6.5536×10^4	modified MLP
Jiang et al. [55]	100, 1000, 5000, 10000	Yes	Adam + L-BFGS-B	1.0×10^5 2.0×10^5	tanh	2.601×10^3 1.0201×10^4	MLP
Vasheghani Farahani et al. [56]	1000, 3200	No	Adam, L-BFGS	5.0×10^5	adaptive tanh	1.0201×10^4	modified MLP
Służalec et al.[P] [57]	–	No	Adam	2.5×10^4	–	$\approx 2.5 \times 10^3$	MLP
Abda et al. [58]	100	No	Adam +L-BFGS	8.9×10^5	tanh	121×121	MLP
Ranade et al. [59]	1000, 2500, 5000	No	Adam	3×10^4	tanh	64^3	CNN-based encoder and decoder
Wei et al.[P] [60]	10000, 20,000	No	Adam	1.0×10^5	swish	258×258	modified MLP
Gradient-flow pathology							
Wang et al. [61]	100	No	Adam	4.0×10^4	tanh	–	modified MLP
Wang et al.[P] [62]	100, 400, 1000, 3200	No	Adam	5.0×10^4- 5.0×10^5	tanh	8.192×10^3 (mini-batch)	modified MLP
Wang et al. [63]	100, 400, 1000, 1600, 3200	No	Adam	1.0×10^4, 5.0×10^5	tanh	4.096×10^3	PirateNet
Wang et al. [64]	5000	No	Adam, Kron, Muon, SOAP	2.0×10^5	tanh	8192	PirateNet
Cohen Tillberg and Ingerskog [65]	100	No	Adam	2.0×10^5	tanh	2.2×10^4	MLP
Li and Feng [66]	100	No	Adam	3.2×10^4	tanh	1.3×10^4	MLP
Wang et al.[P] [67]	5000	No	Adam, SOAP	10^5	tanh	4.096×10^3	PirateNet modified
Niu et al. [68]	100	No	Adam	4.0×10^4	tanh	6.4×10^2	MLP
Feature layer embedding							
Wong et al. [69]	400	Yes	Adam	2.0×10^4	tanh, sine, sigmoid	2.704×10^3	MLP
Chiu et al. [70]	400 (forward & inverse), 1000 (inverse)	No	Adam	2.0×10^5 – 2.0×10^6	sine	2.601×10^3- 4.0401×10^4	MLP
Biswas and Anand [71]	10, 50, 100	No	Adam + L-BFGS	3.0×10^4 + 600	tanh	1.55×10^5	modified MLP
Tsai et al. [72]	400, 1000, 2000, 3200, 5000	No	Adam	1.11297×10^5- 1.207382×10^6	swish	8.695×10^3- 1.9335×10^4	modified MLP
Hu and Senocak[P] [2]	2500, 5000, 7500	No	L-BFGS	10^5	tanh	2.1024×10^4	modified MLP

Note: [P] denotes a preprint. Unmarked entries are non-preprint sources, including journal articles, conference papers, and theses.

Table 1
Continued.

Study	Re	Data	Optimizer	Epochs	Activation function	Collocation points	Network
Unsteady LDC							
Zhang et al. [73]	–	No	Adam	10^5	tanh, swish	$2^{12} - 2^{24}$	DFS-PINN
Khadijeh et al. [74]	–	No	Adam	3.0×10^5	tanh	1.0×10^3	MLP
Farea et al. [75]	100	No	Adam	6.0×10^4	tanh (FNN), learnable swish + B-spline (KAN)	–	MLP, KAN
Tan et al. [76]	100, 400, 1000	Yes	Adam + L-BFGS-B	$> 3.0 \times 10^4$	tanh	4.0×10^3- 1.4×10^5	MLP
Kumar et al.[P] [77]	User-specified / configurable framework						
Not-square geometry							
Takao and Ii [78]	50, 100, 160, 200	Yes	Adam	5.0×10^5	tanh	1.681×10^3	modified MLP
Kiyani et al. [79]	–	No	Adam, Adam + SSBFGS, Lion, AdaBelief, Adam + Nonlinear CG	$\lesssim 5.0 \times 10^5$	tanh	1.2×10^5	MLP
Jnini et al.[P] [80]	–	No	Adam, Adam + SSBFGS, SOAP, SOAP + SSBFGS, SOAP + SSBroyden	$\lesssim 5.2 \times 10^5$	tanh	1.2×10^5	MLP
Solution multiplicity							
Wang et al.[P] [7]	2000, 3000, 5000	Yes	Adam	2.5×10^5- 3.0×10^6	tanh	4.0×10^4- 8.0×10^4	MLP
Zou et al. [81]	–	No	Adam	6.0×10^4	tanh	6.0×10^4	MLP
Zheng et al. [82]	100, 1000 – 50000, 100000	Yes/No	Adam	2.0×10^4	tanh	250×250 500×500	modified MLP
He et al. [83]	1000	No	Adam	6.0×10^5	–	4.126×10^4	MLP
Miscellaneous							
Feng et al. [84]	100, 400, 1000	No	Adam	8.0×10^4	tanh	2.5×10^4	modified MLP
Kumar and Ranjan [85]	100	No	Adam + L-BFGS	$> 3.0 \times 10^4$	tanh	1.04×10^4	MLP
Srinivasan et al.[P] [86]	–	No	Adam	–	–	4.922×10^3	MLP
Cao and Zhang[P] [87]	100, 250, 500, 1000, 2500, 5000	No	L-BFGS	9.0×10^4- 1.5×10^6	tanh	2.2×10^4	MLP
Karnakov et al. [88]	100, 1000	No	L-BFGS-B	5.0×10^5	–	1.04×10^4	MLP
Cao and Zhang [89]	2500	No	Adam	$\approx 5.0 \times 10^5$	tanh	4.0×10^4	MLP
Barman et al.[P] [90]	100	No	L-BFGS	2.0×10^3	learnable sine	–	PhysicsFormer

Note: [P] denotes a preprint. Unmarked entries are non-preprint sources, including journal articles, conference papers, and theses.

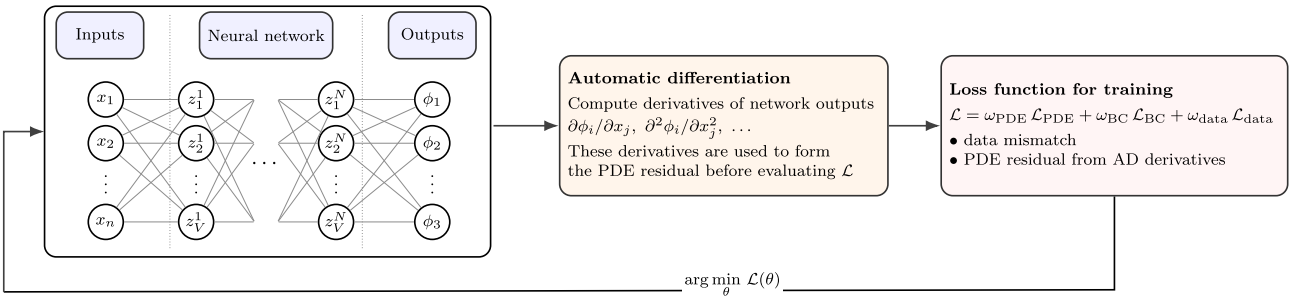


Fig. 2. Basic PINN workflow.

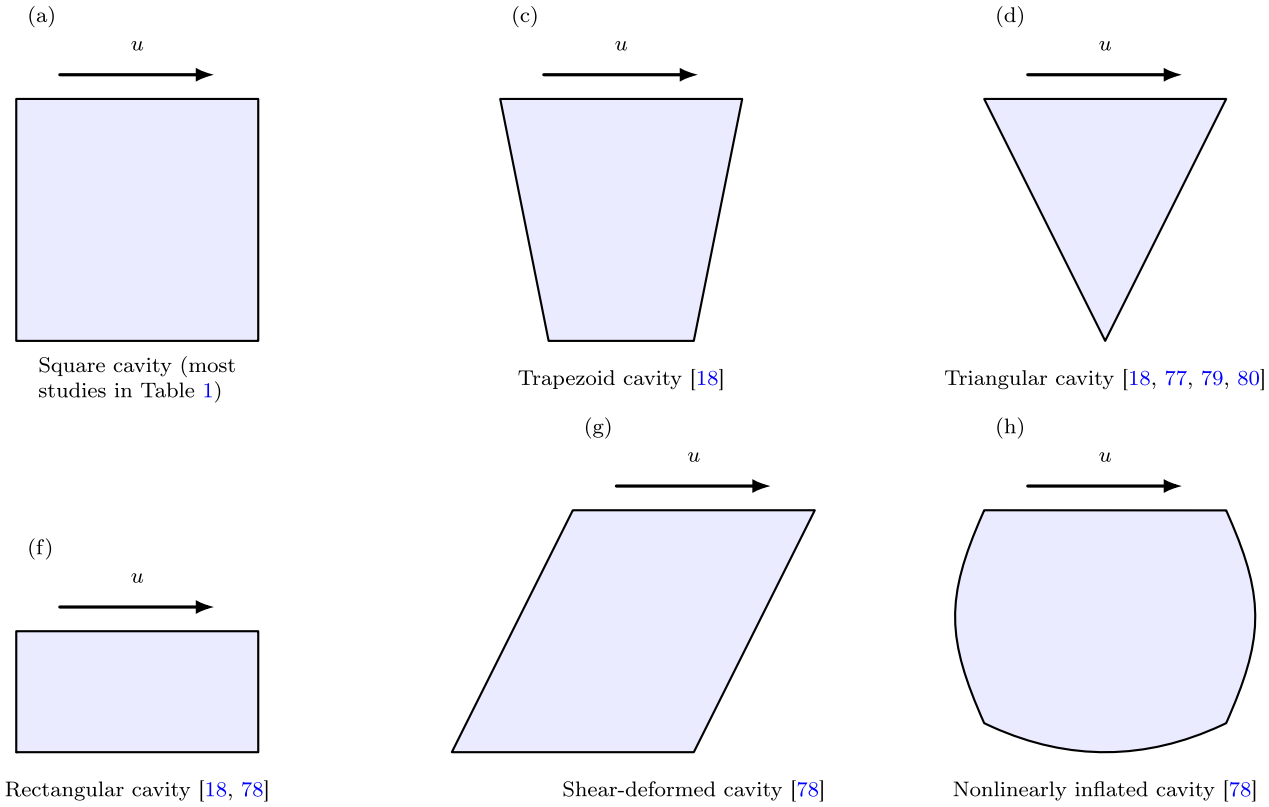


Fig. 3. Different geometries used for studying LDC via PINNs.

or considered cavity configurations outside the scope of the review, duplicated another version of the same work, or were unavailable in full text. When both preprint and peer-reviewed versions were available, the published version was used. The selected studies were then compared to identify methodological trends, reported performance, limitations, and research gaps.

Since physics-informed machine learning is a rapidly developing field, some recent and directly relevant studies were available only as preprints at the time of the review. These sources are explicitly identified in the reference list and in Table 1. Findings based on preprint studies should be regarded as preliminary and interpreted with appropriate caution when considering the insights, comparisons, and research directions presented in this review.

3. LDC formulation for PINNs

LDC is a canonical benchmark for incompressible viscous flow in a bounded domain. As shown in Fig. 3, the geometry of the 2D computational domain can vary. However, in most cases, the computational domain is a square cavity enclosed by solid walls on all sides. The vertical and bottom walls remain stationary, while the top wall (the lid) moves tangentially at a constant velocity of u. The motion of the top wall induces a strong shear layer near the top boundary, which propagates into the interior. As shown in Fig. 4, a large primary vortex develops in the center, accompanied by secondary vortices in the lower corners, whose size and structure depend on the Reynolds (Re) number. The *Re* number is usually defined as $\mathrm{Re} = uH/\nu$, where H is the cavity height and ν is the kinematic viscosity. Fig. 4 also shows how the size and location of vortices vary with the change of *Re* number. As *Re* number increases, the secondary vortices become stronger, and new vortices also appear. The location of minimum and maximum pressure throughout *Re* number variation is also shown in Fig. 5.

A 2D LDC flow remains steady at $\mathrm{Re} < 7000$ [22] or $\mathrm{Re} < 8000$ [5], and for 3D LDC this threshold is $\mathrm{Re} < 3000$ [59]. Also, by the increase of *Re* number, the convective transport dominates the diffusive terms of the Navier-Stokes equation, and sharp velocity and vorticity gradients emerge, which affects the accuracy of PINN prediction [35]. Despite this, Wei et al. [54] showed that their PINN solved LDC up to $\mathrm{Re} = 10000$ with a steady 2D PINN model. Table 1 shows that LDC has been modeled with PINN up to $\mathrm{Re} = 50000$. The distribution of *Re* studied in the literature is shown in Fig. 6.

Although most papers on LDC with PINNs have solved the problem as a steady 2D problem, some studies have solved it as a 3D steady problem [54,69] or 2D unsteady problem [54,73–76]. For example, Wei et al. [54] solved it as an unsteady 2D problem at $\mathrm{Re} = 400$ and a steady 3D problem at $\mathrm{Re} = 1000$, and in both cases they obtained reasonable accuracy, although extending the problem to three spatial dimensions led to a reduction in accuracy.

3.1. Governing equations

An overview of the different governing equations that have been used to model LDC with PINN is presented in Table 2, and the references that employed them are shown in Fig. 7. The graphical convention used in these schematic comparison diagrams in this paper is that citations inside boxes denote studies that used the corresponding method, while citations on arrows denote studies that compared the connected methods. Green arrows point toward the better-performing method, whereas red arrows point toward the underperforming method in the cited comparison. As can be seen, not all governing equations have been compared, which highlights the need for more studies in this regard.

The Navier-Stokes equations are usually used as governing equations for LDC. The common formulation of these equations relates velocity and pressure. Jin et al. [93] call this formulation the velocity–pressure (VP) formulation. An alternative formulation is vorticity-velocity (VV) formulation for modeling incompressible fluid flows [93]. The general formulation of Navier-Stokes equations and also the steady-state 2D formulation of these equations are presented in Table 2. The majority of

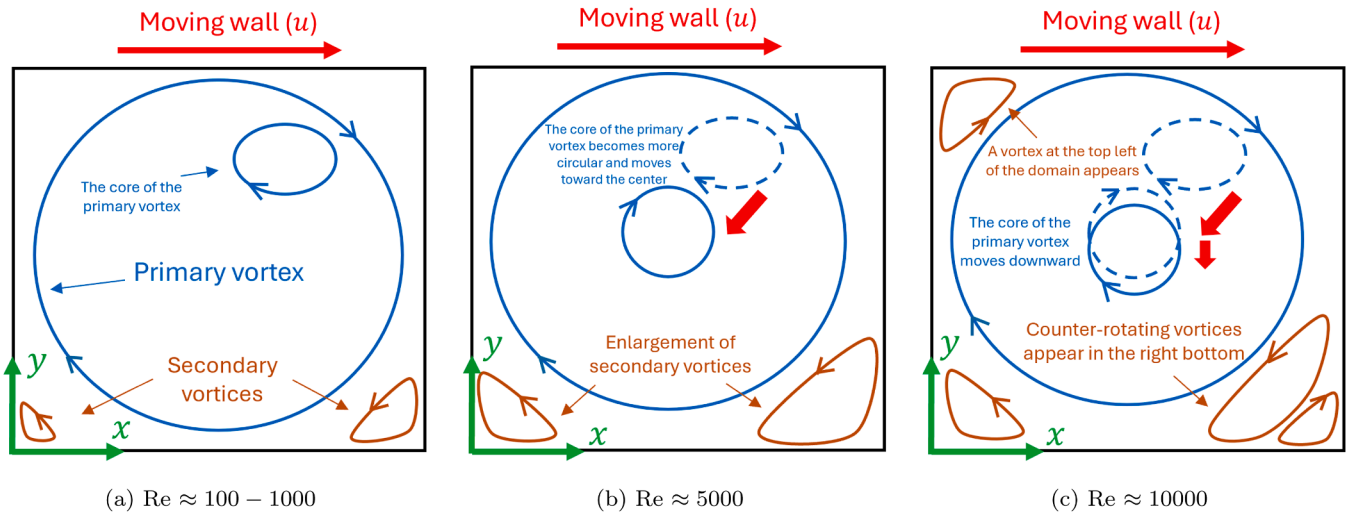


Fig. 4. Schematic illustration of vortices at different *Re* numbers, inspired by the streamlines presented in the literature [54,55].

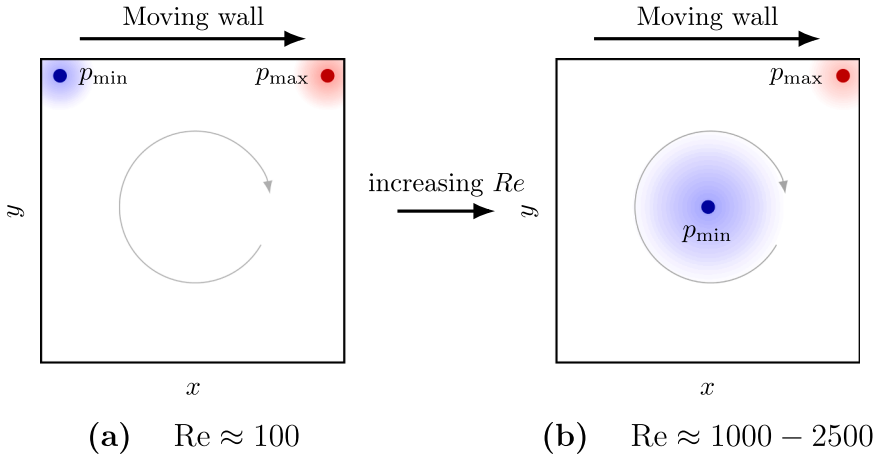


Fig. 5. Schematic illustration of the effect of increasing *Re* number on the pressure field in LDC. The pressure maximum remains anchored near the top-right corner, whereas the pressure minimum shifts from the upper-left region at low *Re* number to the core of the primary vortex at high *Re* number, based on the study of McDevitt et al [18] and Hu and Senocak [2].

studies on LDC with PINN have been done with the VP formulation of the Navier-Stokes equations. After presenting promising results by using the VV formulation for incompressible flows [93], some studies on LDC adopted VV instead of VP, including Chen et al. [50] in *Re* number up to 600 and Tsai et al. [72] up to $Re = 5000$. To the best of our knowledge, there is no comparison between the VP and VV formulations for modeling LDC with PINN.

The VP formulation of the Navier-Stokes equation has been further explored to improve the results of LDC with PINN. Hu and Senocak [2] explained that the VP formulation is fine for high *Re* numbers given supplemental data be used for the training. Khademi and Dufour [41] and Gu et al. [40] employed a scalar streamfunction, ψ, defined through $u = \partial\psi/\partial y$ and $v = -\partial\psi/\partial x$. These relations satisfy the continuity equation identically and, together with pressure, form the streamfunction–pressure (ψP) formulation. Gu et al. [40] compared the regular VP and the ψP formulations at $Re = 100$ (with domain decomposition) and indicated that the ψP formulation had better performance in terms of L_2 error. However, still they stated that the results of the ψP formulation also were not satisfactory, which necessitates the adjustment of the network's architecture and using transfer learning for higher Re number. McDevitt et al. [18] showed that both formulations predicted the flow adequately at $Re = 100$ and 400, whereas the ψP formulation achieved substantially better accuracy at $Re = 1000$. In their formulation, substituting the streamfunction-based velocity definitions into the steady two-dimensional momentum equations while retaining pressure produces the third-order derivatives of ψ shown in Table 2. McDevitt et al. attributed the improved performance to the exact enforcement of incompressibility through the ψP formulation. It can be concluded that the ψP formulation outperforms the VP formulation for LDC. However, it must be noted that the ψP formulation is limited to 2D flows [2]. According to Lin et al. [21], modeling LDC with PINN with the ψP formulation demands fewer collocation points and training epochs.

As shown in the viscosity-enhanced steady-state 2D formulation row of Table 2, the basic VP form of the Navier–Stokes equations can be modified by adding an eddy viscosity, ν_E, to the momentum equations. The auxiliary relations in the last two lines of viscosity-enhanced steady-state 2D formulation row in Table 2 define how this additional viscosity is parameterized in the cited approaches. This improves PINN accuracy for LDC by helping capture high-frequency flow features [83]. However, as mentioned in Table 2, in that case additional equations must also be satisfied. Wang et al. [7] and He et al. [83] stated this equation in two different forms as presented in Table 2. Adding eddy viscosity to the formula is claimed to facilitate the use of PINNs for modeling LDC at higher *Re* number [2]. With this method, He et al. [83] modeled LDC up to $Re = 1000$, and Wang et al. [7] reached up to $Re = 5000$.

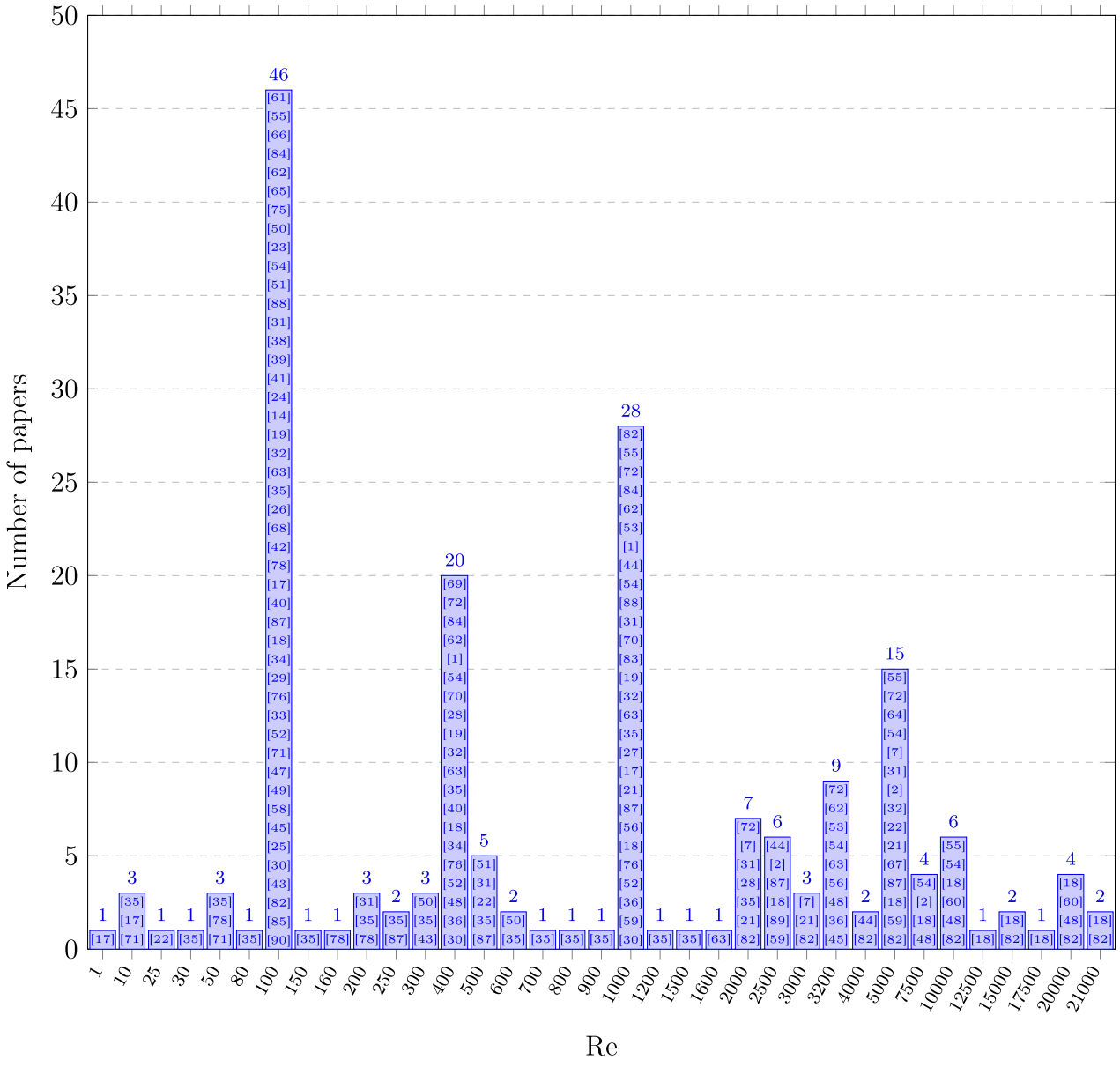


Fig. 6. Number of papers for each *Re* based on Table 1. Numbers in brackets denote the corresponding references.

As summarized in Table 2, the Navier–Stokes equations and their derived formulations are not the only governing-equation constraints used for LDC problems in PINNs. Lou et al. [24] and Bai et al. [14] instead used a Boltzmann-BGK formulation, in which the primary variable is the particle distribution function f_i associated with discrete lattice velocities, rather than the macroscopic variables (u, v, p) directly. The velocity, density, and pressure fields are then recovered from moments of f_i. Both Lou et al. [24] and Bai et al. [14] applied the Boltzmann-BGK formulation to model the LDC problem at $Re = 100$.

The steady incompressible Stokes system has also been used to model LDC [57,79,80,86], given by

$$-\nabla^2\mathbf{u} + \nabla p = \mathbf{0}, \qquad \nabla \cdot \mathbf{u} = 0. \tag{10}$$

Taking the curl of the momentum equation removes the pressure since $\nabla \times (\nabla p) = \mathbf{0}$, giving $\nabla^2\omega = 0$ for the scalar vorticity $\omega = \partial_x v - \partial_y u$. Introducing a streamfunction ψ via $u = \psi_y$ and $v = -\psi_x$ satisfies incompressibility automatically, and with $\omega = -\nabla^2\psi$ we obtain the biharmonic equation, shown in Table 2.

Another consideration concerning governing equations in modeling LDC is the non-dimensionality. Jiang et al. [55] stated that dimensionless governing equations are preferred, and some others also have used these non-dimensional formulations [18–21,25,30,32–35,39,54,58,61,63,68,70,82,85,87,89].

Insights from the literature. The literature indicates that, within Navier-Stokes-based PINN models for LDC, the ψP formulation has been reported to improve accuracy relative to the standard VP form, which highlights the importance of satisfying the divergence-free constraint in LDC. Using the ψP formulation, despite its benefits, imposes more computational cost per training epoch, as a higher-order derivative must be computed. Viscosity-enhanced formulations also appear to improve robustness and accuracy, especially at higher *Re* number. In addition, non-dimensional governing equations are generally recommended.

Room for further research. An important direction for future research is to draw on the extensive literature on incompressible-flow treatment in conventional numerical methods. One example is the addition of eddy viscosity to the momentum equations. These ideas could be adapted to the PINN framework for LDC. More broadly, they could also benefit PINN formulations for incompressible flows. In addition, the limited

Table 2
Representative governing-equation formulations for LDC.

Case	VP formulation	VV formulation
General formulation	$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u}$ $\nabla\cdot\mathbf{u} = 0$	$\frac{\partial \boldsymbol{\omega}}{\partial t} + \nabla\times(\boldsymbol{\omega}\times\mathbf{u}) = -\frac{1}{Re}\nabla\times\nabla\times\boldsymbol{\omega}$ $\nabla^2\mathbf{u} = -\nabla\times\boldsymbol{\omega}$
Steady-state 2D formulation	$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$ $u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$ $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$	
	$u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right)$ $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\frac{\partial \omega}{\partial y}$ $\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = \frac{\partial \omega}{\partial x}$ $\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$	
ψP formulation	$\frac{\partial \psi}{\partial y}\frac{\partial^2 \psi}{\partial x\,\partial y} - \frac{\partial \psi}{\partial x}\frac{\partial^2 \psi}{\partial y^2} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^3 \psi}{\partial x^2\,\partial y} + \frac{\partial^3 \psi}{\partial y^3}\right)$ $-\frac{\partial \psi}{\partial y}\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial \psi}{\partial x}\frac{\partial^2 \psi}{\partial x\,\partial y} = -\frac{\partial p}{\partial y} - \frac{1}{Re}\left(\frac{\partial^3 \psi}{\partial x^3} + \frac{\partial^3 \psi}{\partial x\,\partial y^2}\right)$	*N/A*
Viscosity-enhanced steady-state 2D formulation	$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \left(\frac{1}{Re} + \nu_E\right)\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$ $u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \left(\frac{1}{Re} + \nu_E\right)\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$ $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$ $(u-u_m)e_1 + (v-v_m)e_2 - r = 0$[7,28,81] $\alpha(e_1 u + e_2 v) - \nu_E = 0$[83]	*N/A*
Case	**Boltzmann-BGK**	
Boltzmann-BGK equation	$\frac{\partial f}{\partial t} + \boldsymbol{\xi}\cdot\nabla_{\mathbf{x}} f = -\frac{1}{\tau}(f - f^{eq})$	
Case	**Biharmonic**	
Biharmonic equation (2D)	$\frac{\partial^4 \psi}{\partial x^4} + 2\frac{\partial^4 \psi}{\partial x^2\,\partial y^2} + \frac{\partial^4 \psi}{\partial y^4} = 0$	

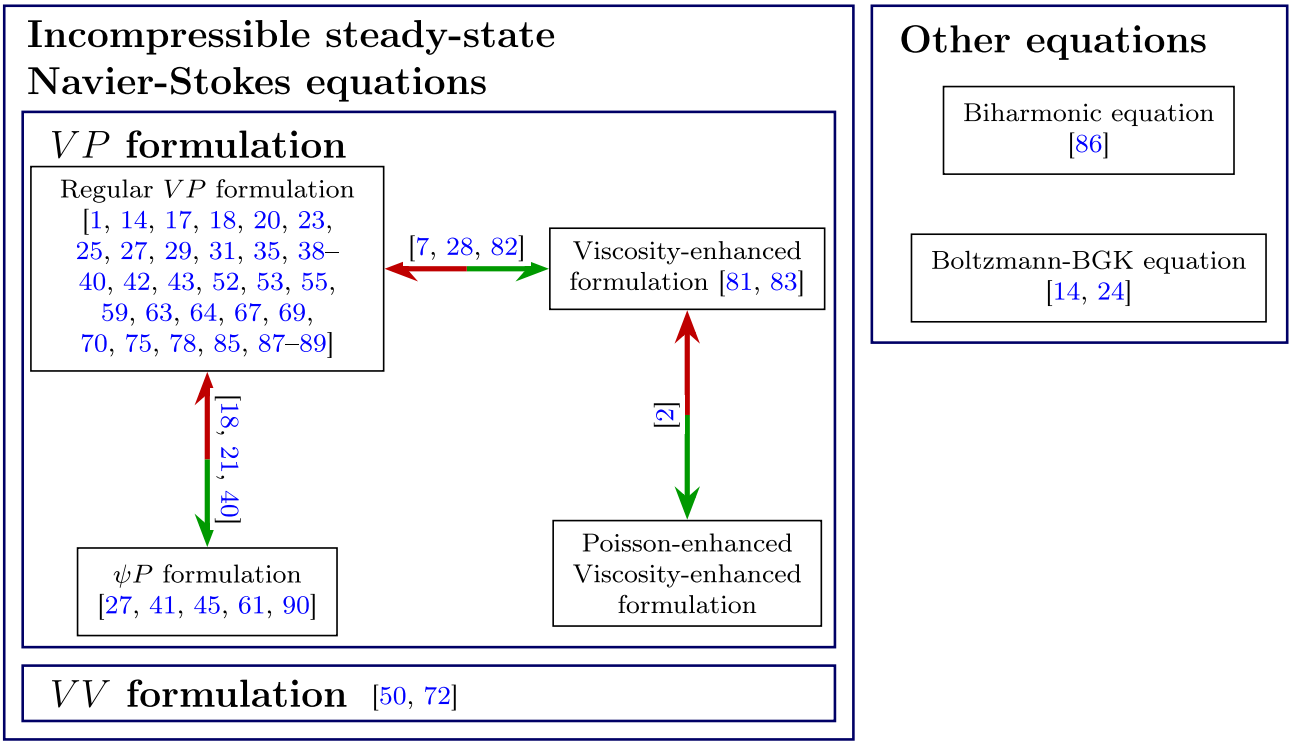


Fig. 7. Comparison between different formulations and equations for LDC in PINN literature. Citations inside boxes denote studies that used the corresponding formulation, while citations on arrows denote studies that compared the connected formulations. Green arrows point toward the better-performing formulation, whereas red arrows point toward the underperforming formulation in the cited comparison. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

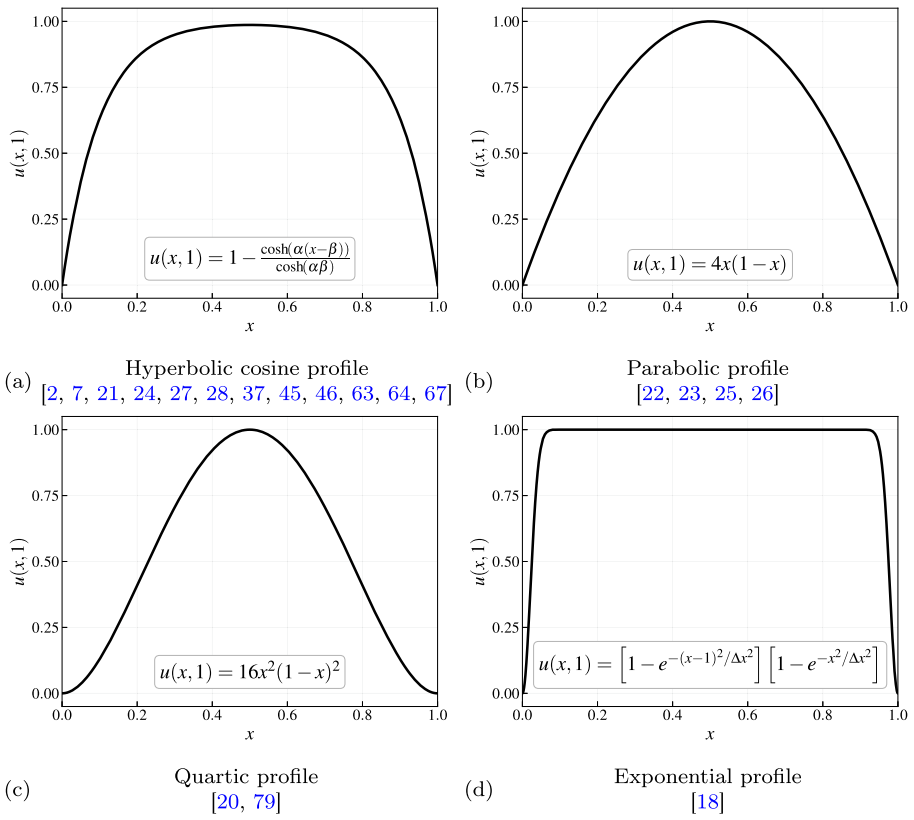


Fig. 8. Lid velocity profiles used for LDC in PINN.

number of direct comparisons shown in Fig. 7 indicates that further systematic comparisons are needed to clarify the relative merits of the available formulations.

3.2. Boundary conditions

In 2D LDC, the three variables u, v, and p are usually under investigation. For u and v, no-slip and no-penetration boundary conditions are imposed on all walls. For p, however, no independent physical boundary condition is prescribed at the walls, since pressure is determined only up to an arbitrary constant. The boundary condition imposed on v along all walls, together with that imposed on u along the side and bottom walls, is generally straightforward and is handled in a similar manner in most studies. However, there are different approaches to enforce the u velocity on the top wall and also enforcing the p boundary conditions. Having a constant u velocity on the top wall matches with the physical definition that a moving lid is driving the flow. This constant horizontal velocity boundary condition is chosen in most studies.

Despite the simplicity of imposing a constant horizontal velocity on the top lid, this condition introduces a discontinuity in the horizontal velocity at the top corners, where the moving wall meets the stationary side walls. As a consequence, sharp velocity gradients and strong localized vorticity arise near the corners. These features make the near-corner flow particularly difficult to resolve accurately, even when an additional data-loss term is included [78]. For this reason, some researchers have adopted a smoothed velocity profile for the lid along the top boundary to remove or reduce the corner discontinuity, although this modification departs from the physical definition of a uniformly moving lid. As shown in Fig. 8, the most common smoothed lid-velocity profiles used for the square LDC problem with PINNs include hyperbolic cosine, parabolic, quartic, and exponential profiles.

It must be mentioned that although the smoothed boundary profiles diminish the problems caused by the discontinuity, they can introduce their own problems. Lin et al. [21] argued that the hyperbolic cosine velocity profile over the lid causes the problem to be ill-posed, ending up in instability. The uncommon flow pattern at $Re = 2000$-2500, found by some studies [2,7,28] that used this velocity profile, supports this argument. However, observing this issue led to discover another merit of PINN, which is the capability of finding different possible solutions of a problem, where they are hard to find by some other methods. Similarly, Zou et al. [81] used a sinusoidal velocity profile for the lid and reached different solutions via their PINN. Then they fed these solutions as initial guesses to a conventional numerical solver, and that solver kept the initial solution in some cases, which proves those solutions could be found by those solvers, but demanded specialized initial guesses, while PINNs by only a few random initialization can converge to the different possible solutions. It is also noteworthy that Shukla et al. [28] related the discovery of the new pattern to getting stuck in local minima.

Regarding the pressure boundary condition, Jiang et al. [55] applied zero gradient on the four walls, and it was found that it did not affect the results considerably. They argued that for PINN simulations, the zero gradient pressure condition on walls is not needed, as the pressure Poisson equation is not solved to update pressure values. Roy et al. [1] also followed Jiang et al. [55] and did not enforce boundary condition for the pressure. Farea et al. [75] also stated that pressure boundary conditions are not needed when solving LDC and that pressure is a latent variable. Gu et al. [40] also did not use a pressure loss term. Instead, they did a shift on the pressure predictions. Takao and Ii [78] set the average value of the predicted pressure to zero. Hu and Senocak [2] used one anchor point in the center ($p(0.5, 0.5) = 0$) to prevent unbounded development. In contrast, Tan et al. [76] did not leave the pressure completely latent for the steady LDC case, and used pressure values obtained from CFD simulation for the four boundaries in a new boundary condition loss term. However, for the unsteady LDC case, they left pressure at boundaries unconstrained.

Insights from the literature. The pressure boundary condition does not have to be enforced, but in order to make the results comparable with the reference data, an anchor point or shifting the average suffices. Regarding the moving wall u boundary condition, although the discontinuity caused by a constant velocity may lead to optimization difficulties, smoothing out the boundary ends may lead to unknown issues, such as the emergence of unexpected flow patterns, which makes the results incomparable to the literature.

Room for further research. The ability of PINNs to discover multiple solutions of LDC has so far been demonstrated only for the hyperbolic cosine and sinusoidal lid velocity profiles. Further research is needed to clarify theoretically which profiles admit multiple solutions, how the number and structure of these solutions vary with the lid profile and *Re* number, and what initializations are required for PINNs to recover as many admissible branches as possible. It is also necessary to distinguish genuinely admissible flow states from patterns that may arise because the optimization is trapped in local minima. Such investigations could also generalize to other nonlinear fluid dynamics problems where multiple solutions may coexist.

3.3. Domain decomposition

Deciding whether the whole domain should be solved via a single network or if a system of networks should collaborate to find the solution is one of the first choices to make. In LDC, large velocity gradients develop near the top corners, where the moving lid meets the stationary side walls. These localized sharp features are very different from the flow in the remainder of the domain. Sharp gradients correspond to higher spatial-frequency content that can be attenuated by the spectral bias of MLP-based PINNs [94]. Consequently, a single global model may struggle to represent both the smooth interior solution and the corner layers simultaneously [95]. By splitting the computational domain into subdomains, separate subnetworks are assigned, each responsible for a subset of the flow with less diverse features. Domain decomposition also facilitates parallelization [39,40,42].

Table 3 presents the main design choices of some studies using domain decomposition for solving LDC with PINNs. The degrees of freedom in domain decomposition design can be classified into three main choices, the optimization method across subdomains, the interaction losses, and the number of subdomains. The first domain decomposition framework for PINNs, called conservative PINN (CPINN), was introduced by Jagtap et al. [38] for systems of conservation laws. As seen in the schematic view in Fig. 9a, CPINN performs non-overlapping domain decomposition. They claimed that domain decomposition reduces all three forms of error, i.e., generalization, approximation, and optimization errors. Building on CPINN, Jagtap and Karniadakis [96] later proposed XPINN, an extended PINN framework that generalizes non-overlapping domain decomposition to arbitrary PDEs, not just conservation laws. XPINN maintains a non-overlapping partitioning of the domain, but it allows for flexible spatial and spatio-temporal decomposition on irregular geometries. Shukla et al. [39] compared CPINN and XPINN for LDC at $Re = 100$, demonstrating the effect of different decomposition strategies on solution accuracy. Some other domain decomposition techniques on LDC are compared in Table 3.

As shown in Fig. 9b, overlapping interfaces have been used by Qian et al. [42] for LDC. They also addressed the pressure indeterminacy across different subnetworks and reported that by increasing the number of subdomains to two and four, the L2 errors of both velocity and pressure decrease.

Figure. 9c denotes that the subdomains can be treated completely independent from each other with no interaction losses. Roy et al. [1] used this type of domain decomposition and showed that decomposing the domain noticeably improves capturing the secondary vortices in the bottom corners. However, there are also cases where domain decomposition worsened the results. Lou et al. [24] also mentioned that they split the large domain into 9 subdomains. However, they essentially used a non-uniform mesh in different areas of the domain, without using different neural networks for the subdomains. They used no interaction function between the subdomains, and the whole domain was being optimized altogether.

The interaction losses that couple neighboring subdomains in all reviewed studies include a term enforcing solution consistency across the interface. Some works directly penalize the mismatch between the interface predictions of adjacent subdomains [40–42], whereas others penalize the mismatch between each subdomain prediction and the average of the two neighboring interface predictions [38,39,43]. For two neighboring subdomains, these two formulations are mathematically equivalent up to a constant scaling factor and therefore differ mainly in their effective weighting rather than in the continuity condition they impose. In the case of Qian et al. [42], this consistency is enforced through ghost-layer losses, where each subdomain is penalized against the predictions received from its neighboring subnetworks.

No comparable consensus exists for the remaining interaction losses. Among them, flux continuity is the most commonly adopted coupling mechanism [38–40]. It should be noted, however, that the mass-flux continuity condition becomes redundant when continuity of the velocity components is already enforced at the interface, since matching u and v across neighboring subdomains also implies continuity of the normal mass flux. In contrast, the momentum-flux continuity terms provide a genuinely additional coupling effect because they involve pressure and spatial derivatives, and therefore cannot be recovered from solution matching alone. Other interface treatments reported in the literature include PDE-residual continuity in XPINN [39], first-derivative matching aimed at enforcing differentiability across interfaces [41], and interface reduced-PDE consistency, in which the interface governing equation is evaluated using information from both neighboring subdomains [43]. Notably, Qian et al. [42] only used solution matching interface loss. They argued that additional constraints require the evaluation of higher-order derivatives on ghost-layer points, substantially increasing the computational cost, while offering only marginal accuracy improvements.

Insights from the literature. Although domain decomposition is often argued to localize non-convex features and prevent local errors from contaminating the entire solution [38], its application to LDC is not straightforward. In LDC, the flow is driven entirely by the moving lid, so the dominant dynamics originate at the boundary and must propagate coherently throughout the whole domain. If the interfaces between subdomains do not communicate accurately, isolating this region can instead degrade the global solution. This makes domain decomposition a non-trivial strategy for LDC and adds substantial implementation complexity. The literature reviewed here suggests that, although domain decomposition can improve overall accuracy and is particularly helpful in resolving secondary vortices, its benefits should not be overstated. None of the reported studies reached *Re* numbers beyond 1000, and the method can significantly increase computational cost. Therefore, for LDC, domain decomposition appears to be a relatively expensive strategy that provides moderate improvements in PINN performance.

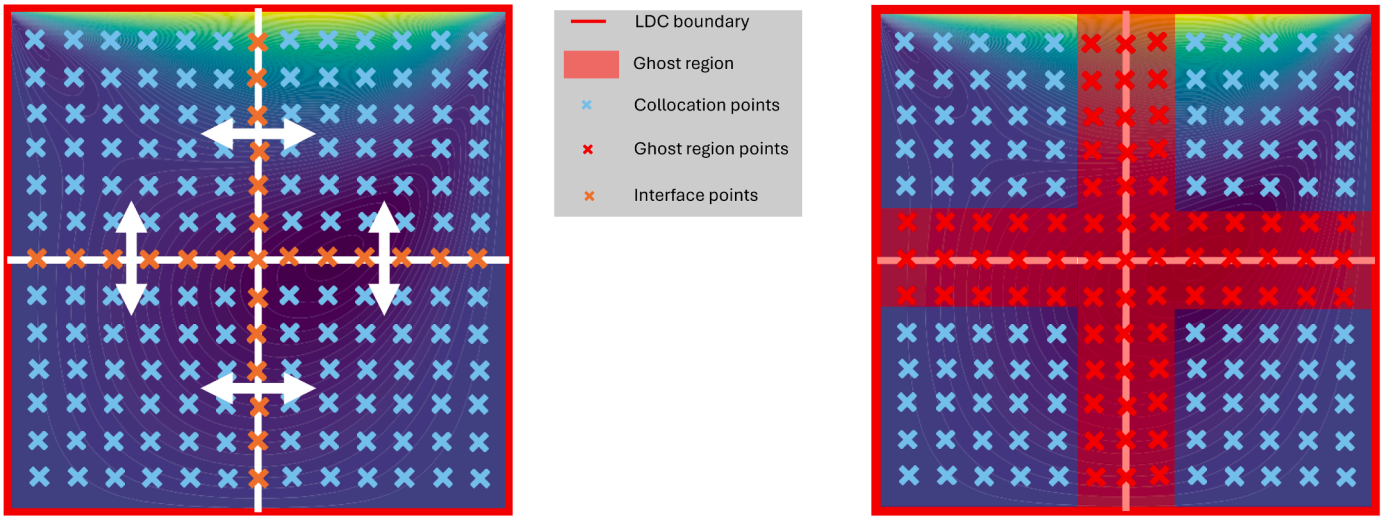
Room for further research. The lack of consensus on interface-loss design highlights a clear direction for future research. A systematic comparison of the existing coupling strategies would help clarify their relative merits and limitations.

4. PINNs architecture

This section reviews the neural network architectures used for PINN-based modeling of the LDC problem. It first summarizes the types of networks reported in the literature. Then, with a focus on multi-layer perceptron (MLP)-based formulations, it discusses four key architectural design choices, namely the input representation, the feature mapping

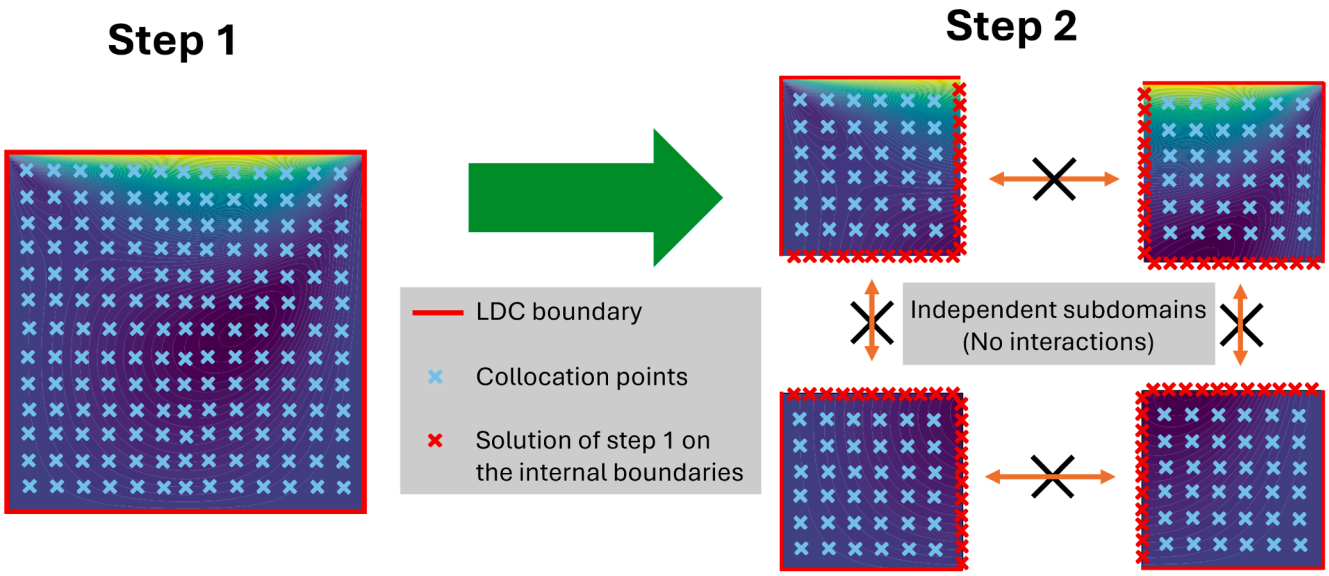
Table 3
Summary of studies applying domain decomposition for solving LDC.

Reference	Optimization method	Interaction losses	Number of subdomains
Jagtap et al. [38]	Subdomain-wise optimization of separate subnetworks with neighbor-coupled local loss functions.	1. Average-solution matching 2. Flux continuity	12
Roy et al. [1]	Subdomain-wise optimization of separate PINNs, while initialized by a pre-trained single-domain model.	No interaction function	16
Gu et al. [40]	Similar to Jagtap et al. [38].	1. Solution matching 2. Flux continuity	4
Qian et al. [42]	Similar to Jagtap et al. [38].	Solution matching in overlapping regions.	2, 4
Shukla et al. [39]	Similar to Jagtap et al. [38].	cPINN: Similar to Jagtap et al. [38] XPINN: 1. Average-solution matching 2. Residual continuity	4
Khademi and Durfour [41]	Similar to Jagtap et al. [38].	1. Solution matching 2. First derivative matching	4
Pan and Xiao [43]	Stage 1: Subnetworks are trained independently using only subdomain losses and no interface coupling. Stage 2: all subnetworks are jointly fine-tuned with interaction losses.	1. Average-solution matching 2. Interface PDE consistency	3



(a) Interfaces on subdomain boundaries [38–41, 43]

(b) Overlapping interfaces [42]



(c) No interface and compeletly independent subdomains [1]

Fig. 9. Different domain decomposition techniques used for studying LDC via PINNs.

layer, the trial functions used in the output layer to enforce hard constraints, and the activation functions.

4.1. General architecture

As seen in Table 1, MLP is the most commonly used network architecture for PINN-based solution of the LDC problem. In a basic MLP, the main architectural choices are the activation function, the number of hidden layers, and the number of neurons in each layer. Since there is still no well-established theory for selecting the optimal depth and width of an MLP for this problem, these parameters are often chosen based on experience, empirical tuning, or trial-and-error procedures [1,54]. Chen et al. [50], for example, increased the number of neurons in the hidden layers as *Re* number increased. However, increasing the size of an MLP does not necessarily improve its performance [82]. Several studies have therefore regarded plain MLPs as insufficient and proposed modified MLP-based architectures for the LDC problem. Fig. 10 compares two such modified MLPs with a plain MLP. Fig. 10b shows a coordinate-wise PINN architecture, where the input coordinates are processed by separate subnetworks before being combined through a fusion module. This allows the model to extract coordinate-specific features before predicting the velocity and pressure fields. Fig. 10c illustrates a multi-task PINN architecture. A shared feature extractor first learns common representations from the spatial inputs, while separate task-specific output heads are then used to predict u, v, and p. This structure allows the model to share useful flow features while retaining flexibility for each physical variable. Another outstanding example is the Physics-Informed Residual Adaptive Network, known as PirateNet, developed by Wang et al. [63]. PirateNet introduces residual adaptive connections to improve the expressiveness and trainability of PINNs, and has been specifically evaluated for the LDC problem in several studies [63,64,67].

MLPs and their variants are not the only network architectures used for PINN-based modeling of LDC. Kolmogorov-Arnold Networks (KAN) have also recently been applied to this problem [28,44,75]. Shukla et al. [28] compared MLPs with different KAN variants for the LDC problem and found that the KAN models did not considerably improve the accuracy, while requiring longer training times. The following subsections focus mainly on MLP-based networks.

4.2. Input layer and feature layer

Most PINN studies of LDC have focused on the 2D steady-state case. In this setting, the input layer is typically minimal, consisting of only two neurons, x and y, which represent the spatial coordinates. Accordingly, as shown in Fig. 11, these two inputs appear in all studies. For 3D configurations [54,59,71], an additional spatial input, usually z, is included. For unsteady LDC problems, time must also be incorporated, requiring an additional input t, as in unsteady 2D studies [54,73–76].

Additional neurons may be included in the input layer when either the governing equations or the computational domain is parameterized, with each parameter represented by a separate input. For example, McDevitt et al. [18] and Jangir et al. [35] treated *Re* number as an input parameter, enabling the PINN to be trained over multiple *Re* numbers and then evaluated across a continuous range of *Re* numbers. Zheng et al. [82] instead used a normalized logarithmic form of *Re* number mapped to the interval $[-1, 1]$. As illustrated in Fig. 11, the neuron labeled *Re* represents this type of parametric input. Geometric descriptors can also be appended to the input layer when parameterized domains are considered, as demonstrated by McDevitt et al. [18] and indicated schematically in Fig. 11.

It is noteworthy that the time dimension can also be included in other ways, as done by Sababha et al. [22], without adding a specific neuron to the input layer. In the study of Cao and Zhang [87], the concept of pseudo-time has been used, which does not need adding a new neuron to the input layer. Discussing these methods is out of the scope of the current review study.

As shown in Fig. 11, some PINN studies on the LDC problem use a first hidden layer (feature layer) that differs from the subsequent hidden layers in its design or formulation. Sinusoidal feature mapping was introduced by Wong et al. [69], as a way to avoid getting stuck in local minima. They dedicated the first hidden layer right after the input layer to carry out the sinusoidal mapping imposing $\gamma(\mathbf{x}) = \sin\left(2\pi(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)\right)$, where $\mathbf{W}$ and b were trainable. Their results showed that using this layer increases the accuracy of modeling LDC, regardless of the initialization algorithm or the activation function used in the following hidden layers. This idea has been followed in other studies. Chiu et al. [70] followed Wong et al. [69] and used the same mapping in the first hidden layer. Tsai et al. [72], also inspired by the work of Wong et al. [69], used $\sigma(x) = \sin 2\pi x$ as the activation function of the first hidden layer. Biswas and Anand [71], in their study of the 3D LDC, used a sine activation function in the first hidden layer, while employing the tanh in the subsequent hidden layers. Hu and Senocak [2] used a Fourier feature mapping in the first hidden layer to enhance the expressive capacity of the PINN, defined as

$$\gamma(\mathbf{x}) = \begin{bmatrix} \cos(2\pi\mathbf{B}\mathbf{x}) \\ \sin(2\pi\mathbf{B}\mathbf{x}) \end{bmatrix},$$

where $\mathbf{B}$ is sampled from a Gaussian distribution and kept fixed during training. Wei et al. [54] also used a frequency mapping layer in their FFV-PINN to project the input coordinates into a high-dimensional space for capturing high-frequency information. In their subsequent SIMPLE-PINN study [60], they employed a first-layer frequency annealing mapping for the same purpose, enhancing the representation of high-frequency flow features.

Insights from the literature. Introducing additional parameters into the problem offers both benefits and drawbacks. For example, when *Re* number is treated as an input parameter, data available at only a limited set of *Re* number can be leveraged to improve predictions at other *Re* numbers. However, increasing the number of input dimensions also exacerbates the curse of dimensionality [73].

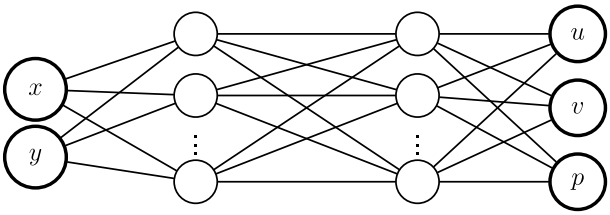
Room for further research. Although sinusoidal mapping in the first hidden layer has proved beneficial to solve LDC with PINN in the literature, at least four different formula have been used, from $\sin x$ [71] to the rather complicated formula used by wong et al. [69] accompanied by trainable variables. An ablation study between these candidates would be insightful.

Another point is that parameterizing the governing equations increases the generality of the model. However, such parameterization may also push PINNs toward tasks that are more suited to operator-learning frameworks. Although the relative simplicity of PINNs, compared with operator-learning methods, makes them attractive in practice, it remains unclear whether they are the most efficient choice in parameterized settings. This raises an important research question. When problem parameters must be incorporated, is it preferable to remain within the PINN framework or to transition to operator learning?

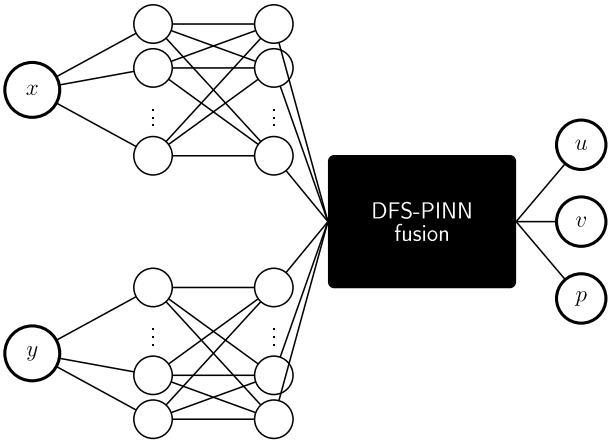
4.3. Last layer for hard constraining

The network architecture can be constructed in a way to satisfy the constraints automatically [97], which is called setting hard constraints [98]. By hard-imposing a constraint like boundary conditions, the perfect satisfaction of that constraint is guaranteed and the optimization process does not need to focus on that. The use of hard constraints has been considered in the literature as a helpful tool for modeling LDC with PINN [53].

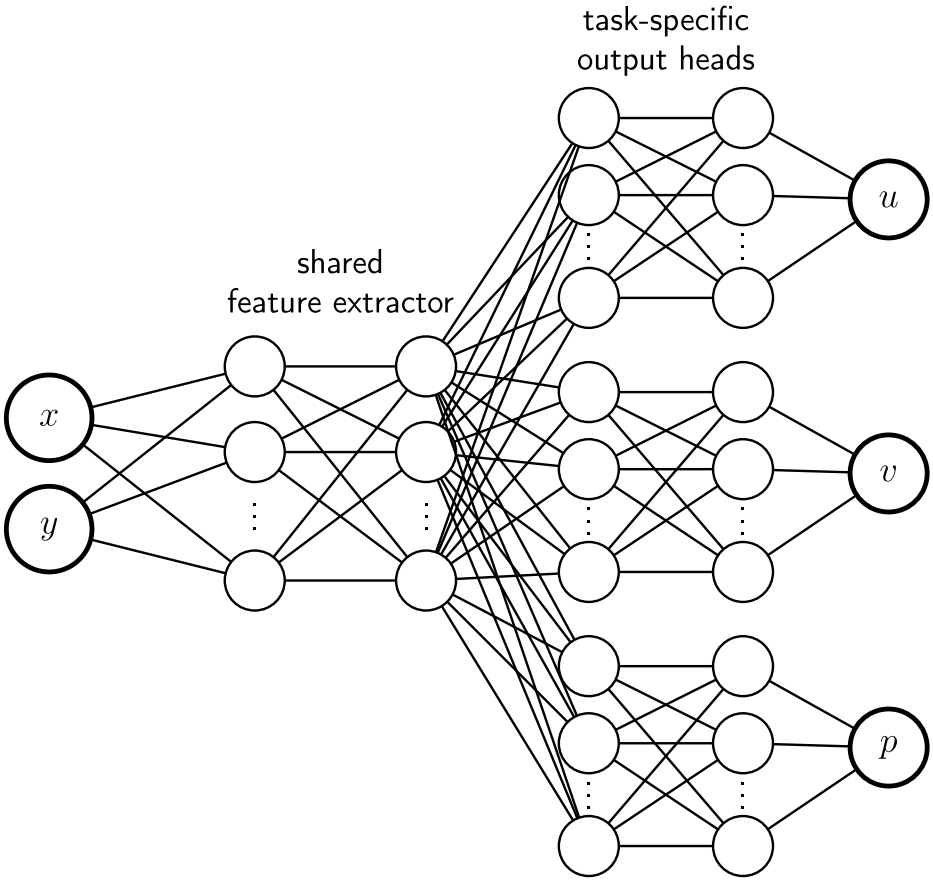
A well-known hard constraining technique that has been used to model LDC with PINN [17–19] is using trial functions. This idea was introduced by Lagaris et al. [15], who expressed the solution as a trial function that satisfies the boundary conditions by construction. When



(a) Plain MLP architecture (used in many studies e.g. [1]).



(b) Coordinate-wise neural network with a fusion module [73].



(c) Multi-task PINN with a shared trunk and task-specific output heads [54, 72].

Fig. 10. Plain MLP (a) and its modified versions (b and c) used to model LDC.

boundary conditions are imposed through trial functions, the neural network is left to learn only the governing partial differential equations, which can help alleviate gradient imbalance arising from multiple competing loss terms [99]. In general, a hard-constrained trial function can be written as

$$\phi(x, y) = \phi_{\text{boundary}}(x, y) + \phi_{\text{zero}}(x, y)\hat{\phi}(x, y), \tag{11}$$

where ϕ_{boundary} satisfies the prescribed nonhomogeneous boundary conditions, ϕ_{zero} is constructed so that it vanishes on the boundaries where hard constraints are imposed, $\hat{\phi}(x, y)$ is the neural network output. For homogeneous boundary conditions, $\phi_{\text{boundary}} = 0$, and the ansatz reduces to a purely multiplicative form.

For LDC, homogeneous Dirichlet conditions of u and v can be enforced through multiplicative factors that vanish on the relevant boundaries. The main difficulty arises for the horizontal velocity u, because the moving-lid condition is discontinuous at the two top corners. As a result, a single smooth trial function cannot simultaneously impose $u = 1$ on the entire top wall and $u = 0$ on both side walls pointwise at the corners. As listed in Table 4, McDevitt et al. [18] proposed fully hard-constrained formulations in both (u, v) and ψ formulations. As shown in Fig. 8d, their choice of u_{boundary} regularizes the top-corner discontinu-

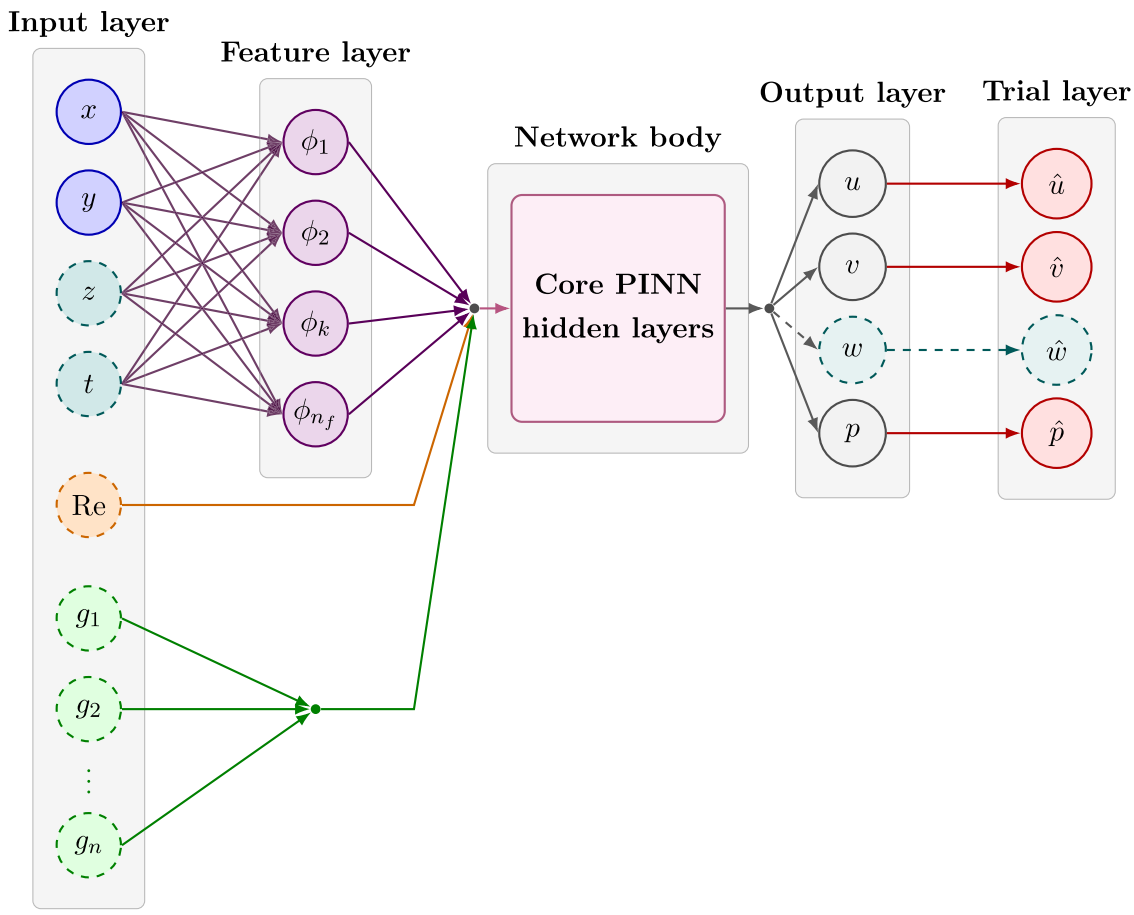


Fig. 11. Generalized PINN architecture for modeling LDC. The feature layer acts only on the spatio-temporal inputs, while auxiliary inputs such as *Re* number and geometry descriptors bypass that layer and are concatenated later. The network body produces the output variables (u, v, p), with an optional w component shown in dashed style for three-dimensional cases, which are then passed through a hard-constraining layer to obtain the constrained fields $(\hat{u}, \hat{v}, \hat{p})$, with optional $\hat{w}$.

ity by replacing the abrupt lid-velocity jump with a smooth transition over a small length scale set by Δx. Consequently, the lid speed is not uniform near the two top corners and exhibits small localized deviations from the ideal $u = 1$ boundary condition. In the ψ formulation, the same regularization is embedded in the boundary construction for ψ, after which the velocity field is recovered from the ψ via $u = \partial\psi/\partial y$ and $v = -\partial\psi/\partial x$, as discussed in Section 3.1. As shown in Table 4, Pal et al. [19] also presented trial functions to satisfy the boundary conditions u, v, and p. In their structure, by approaching the top boundary, the u velocity nears 1, but the limitation is that no collocation points can be put exactly on the boundaries, as the denominators in the trial function of u become zero. With this method, they could reach $Re = 400$ successfully, but the results at $Re = 1000$ did not match the reference results well. As shown in Table 4, Pal et al. [19] used also a trial function to impose the reference pressure point $p(0, 1) = 0$ as a hard constraint. Pal et al. [19] compared their hard-constrained setting with a setting having boundary conditions as soft constraints and reported that hard-constrained boundary conditions reduced the relative L_2 error of u up to 50%. Kiyani et al. [79] also solved LDC for a wedge geometry via hard constrained boundary conditions.

Table 4 also includes two partially hard-constrained alternatives introduced in the present review rather than adopted from the existing literature. In partially hard constrained 1 (u, v), the trial function for u enforces $u = 0$ on the left, right, and bottom boundaries, while the top boundary is imposed softly. In partially hard constrained 2 (u, v), the trial function for u enforces $u = 0$ at the bottom wall and $u = 1$ at the top wall, while the side-wall conditions are imposed softly. In both alternatives, the trial function for v enforces $v = 0$ on all boundaries by construction.

Other hard constraining methods. In addition to trial functions, some other hard constraining methods have also been used for LDC with PINN. Hu and Senocak [2] used the augmented Lagrangian method for LDC at $Re = 7500$. For PINNs using discretization-based methods, such as finite difference for the calculation of the loss value, boundary conditions are directly imposed and excluded from the training process. Jiang et al. [55] and Feng et al. [32] have used this kind of method and modeled LDC up to respectively $Re = 10{,}000$ and 5,000. Another approach that can be seen as hard constraining is using the $\psi - p$ formulation of the Navier–Stokes equation as applied on LDC in the literature [18]. By using this formula, the incompressibility constraint and the continuity equation are automatically enforced. Therefore, there is no need to use a continuity loss term, although one more derivative must be taken to calculate u and v from ψ. Ba et al. [36] developed a PINN framework that provides an alternative to using trial functions for hard constraining boundary conditions. Their method introduces a classification neural network and a boundary condition network in a composite PINN architecture.

Insights from the literature. A key advantage of hard boundary enforcement is that it removes the need for an additional boundary condition objective in the loss function, thereby simplifying the optimization problem, leading to lower error at least at $Re = 100$ [19]. However, this benefit may come with a trade-off. If the trial function used to impose the

Table 4
Representative hard-constraint trial functions for LDC.

Method	ϕ	Trial function
Fully hard constrained (u, v) [18]	u	$u_{\text{boundary}}(x,y) = \sin\left[\frac{3\pi}{2}\left(y-\frac{2}{3}\right)\right] \times \left(1-\exp\left[-\frac{(x-1)^2}{\Delta x^2}\right]\right)\left(1-\exp\left[-\frac{x^2}{\Delta x^2}\right]\right)$,
		$u_{\text{zero}}(x,y) = 16x(x-1)y(y-1)$
	v	$v_{\text{boundary}}(x,y) = 0$,
		$v_{\text{zero}}(x,y) = 16x(x-1)y(y-1)$
Fully hard constrained (ψ) [18]	ψ	$\psi_{\text{boundary}}(x,y) = (y-1)y^2 \times \left(1-\exp\left[-\frac{(x-1)^2}{\Delta x^2}\right]\right)\left(1-\exp\left[-\frac{x^2}{\Delta x^2}\right]\right)$,
		$\psi_{\text{zero}}(x,y) = 256x^2(x-1)^2y^2(y-1)^2$
Fully hard constrained (u, v, p) [19]	u	$u_{\text{boundary}}(x,y) = \frac{4}{\pi^2}\theta_1\theta_2\, y$,
		$\theta_1 = \tan^{-1}\left(\frac{x}{1-y}\right), \quad \theta_2 = \tan^{-1}\left(\frac{1-x}{1-y}\right)$,
		$u_{\text{zero}}(x,y) = x(1-x)y(1-y)$
	v	$v_{\text{boundary}}(x,y) = 0$,
		$v_{\text{zero}}(x,y) = x(1-x)y(1-y)$
	p	$p_{\text{boundary}}(x,y) = 0$,
		$p_{\text{zero}}(x,y) = r_1$,
		$r_1 = \sqrt{x^2+(1-y)^2}$
Partially hard constrained 1 (u, v)	u	$u_{\text{boundary}}(x,y) = 0$,
		$u_{\text{zero}}(x,y) = x(1-x)y$
	v	$v_{\text{boundary}}(x,y) = 0$,
		$v_{\text{zero}}(x,y) = x(1-x)y(1-y)$
Partially hard constrained 2 (u, v)	u	$u_{\text{boundary}}(x,y) = y$,
		$u_{\text{zero}}(x,y) = y(1-y)$
	v	$v_{\text{boundary}}(x,y) = 0$,
		$v_{\text{zero}}(x,y) = x(1-x)y(1-y)$

top lid boundary condition alters the prescribed lid velocity profile, as in McDevitt et al. [18], then the resulting solution is not strictly comparable with the standard LDC benchmark, for which the top lid is usually taken to move at a constant velocity. This is essentially the same comparability issue that arises when the lid velocity is smoothed under a soft constraint treatment.

Room for further study. Trial function design is not unique, and many alternative constructions can satisfy the boundary conditions exactly. Although the trial functions proposed by Pal et al. [19] showed clear improvement over the soft-constrained formulation at $Re = 100$, their performance deteriorated at higher Re numbers and did not yield satisfactory results at $Re = 1000$. This suggests that a more systematic investigation may find trial functions leading to better results.

4.4. Activation functions

Activation functions supply the nonlinearity needed for neural networks to approximate complex solution fields. Their choice is important because their derivatives affect the gradients of the PINN loss and therefore the training dynamics [100]. Despite extensive research, there is still no consensus on an optimal activation function for modeling fluid flows with PINNs. This section reviews the suitability of some of the most common smooth and non-smooth activation functions used to model LDC, with the discussion focusing on MLP-based PINNs unless another architecture, such as KAN, is stated explicitly.

Hyperbolic tangent (tanh). According to Table 1, tanh is overwhelmingly the most common activation function for solving LDC with PINNs. Tanh has been suggested for solving LDC with PINN, because of its smoothness, bounded output, and symmetric output range [35]. Wong et al. [69] also showed that it has better performance than sigmoid. However, despite its popularity, tanh has been criticized for not handling high-frequency gradients and causing the PINN to suffer from spectral bias [101], which can be a concern for LDC modeling at high Re numbers. Also, tanh has been blamed for causing a slow training process [102]. In a comparison with the sine function, tanh showed inferior performance [69]. Jagtap et al. [38] and Vasheghani et al. [56] used adaptive tanh to solve LDC.

Swish or Sigmoid Linear Unit (SiLU). According to Table 1, swish function ranks second in frequency of use. It has been adopted in a small number of PINN studies on LDC [19,47,50,51,54,60,72,73]. Tsai et al. [72] and Chen et al. [50] cited its reduced susceptibility to vanishing and exploding gradients. In most works, however, the choice of swish is generally not explicitly justified. Zhang et al. [73] modeled LDC with PINNs using swish and tanh, but they did not report a direct tanh vs swish ablation. Nevertheless, the results reported by Wei et al. [54,60] suggest that swish can be compatible with successful simulations at relatively high Re numbers, reaching $Re = 10{,}000$ and $Re = 20{,}000$. Farea et al. [75], who employed swish within a KAN architecture, argued that swish is an activation function with a smooth global profile, making it suitable for approximating complex nonlinear relationships. Its smoothness may also promote more stable gradient propagation during backpropagation, which can, in turn, improve optimization.

Sine. Bai et al. [14] employed the sine activation function to model LDC, although they did not report a comparison with other activation functions. Wong et al. [69] conducted a systematic investigation and demonstrated that the sine activation function outperforms both tanh and sigmoid activation functions. Motivated by this observation, Chiu et al. [70] adopted the sine activation function instead of tanh. Khademi and Dufour [45] proposed an adaptive sine activation function, which yielded improved results compared with tanh. However, they did not compare it against the standard sine activation function. Barman et al. [90] used learnable sine activation function. It is also worth noting that several other studies have used the sine function specifically in the first hidden layer [71,72]. In some studies, this layer is referred to as a feature-mapping layer, as discussed in Section 4.2.

Sigmoid. It was compared with tanh and sine activation functions for LDC at $Re = 400$ by Wong et al. [69], and was found to perform worse than both. More broadly, evaluations of the sigmoid function remain

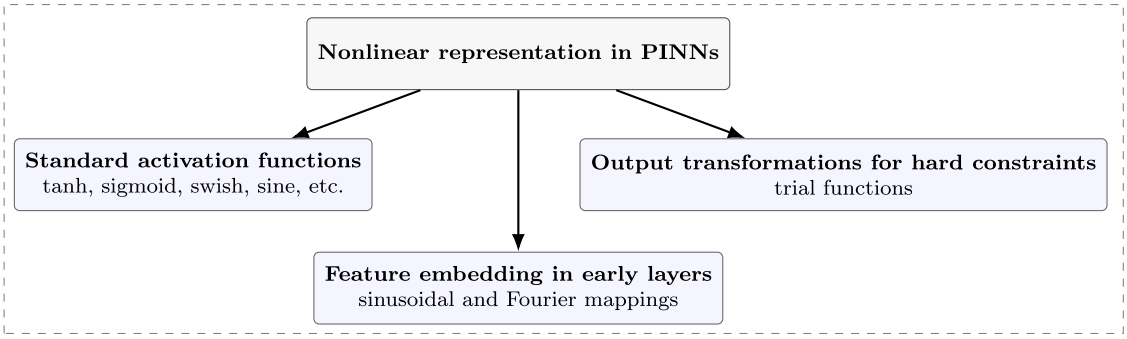


Fig. 12. Conceptual relation between standard activation functions and other nonlinear transformations used in PINNs.

mixed. Although Jin et al. [93] successfully used it to model incompressible flow, Glorot and Bengio [103] argued that it should generally be avoided in randomly initialized deep neural networks. Similarly, Kurukulasuriya et al. [102] noted that its boundedness can be advantageous, but identified its non-zero-centered output and susceptibility to vanishing gradients as major limitations.

Rectified Linear Unit (ReLU). ReLU is one of the most widely used activation functions in neural network training [102]. However, ReLU is generally not well-suited for PINNs with MLPs in which the PDE residuals are evaluated by automatic differentiation. ReLU does not possess continuous second-order derivatives [93], and PINNs based on ReLU are known to suffer from spectral bias [101]. Moreover, Leng and Thiyagalingam [104] showed that MLPs with ReLU yield a vanishing Hessian, which obstructs convergence when solving PDEs of second order or higher. Since the Navier–Stokes equations are second-order PDEs, PINNs based on MLPs that compute second derivatives via automatic differentiation are not compatible with ReLU as the activation function. However, ReLU can still be used where the neural network does not directly serve as a twice-differentiated continuous trial function. For example, Gao et al. [49] successfully used ReLU in a physics-informed graph neural Galerkin network for LDC. In their work, the required differential operators are not obtained by taking second-order automatic derivatives of an MLP.

Insights from the literature. The literature suggests that tanh remains the most established activation function for MLP-based PINNs applied to LDC. ReLU appears unsuitable for MLPs because its lack of smooth higher-order derivatives conflicts with the derivative requirements of LDC. Sigmoid is generally a weaker candidate because of optimization difficulties and inferior reported performance, whereas its descendant swish, shows more promising results and is being adopted more often in recent studies, including at relatively high *Re* numbers. Sine also appears to be a credible alternative, with comparative studies indicating that it can outperform both tanh and sigmoid in some LDC settings.

At first glance, activation function design may appear to have only a limited role in LDC literature. However, as illustrated in Fig. 12, the sections on feature embedding in the first hidden layer (Section 4.2) and on trial functions for hard constraining (Section 4.3) suggest a broader perspective. In both cases, the methods are not presented explicitly as branches of activation function design, yet they rely on special transformations whose underlying role is closely related to what activation functions do in the hidden layers, namely shaping how the network represents the solution. This indicates that careful attention to activation functions and related nonlinear transformations can be effective for improving performance and even opening new directions in the design space of PINNs.

Room for further research. Generally, there are not many comparative studies on the impact of different activation functions on solving LDC with PINNs. Particularly, regarding the popularity of tanh and emerging studies with swish with promising results, a direct comparison between these two activation functions is needed.

5. Optimization scheme and training strategy

In this section, we discuss three important choices for modeling LDC with PINNs. These are the optimization scheme, the training strategies, and the distribution of collocation points. Since the inclusion of data points is primarily a design choice in the formulation of the loss terms, methods for distributing data points are discussed in the section on loss function design.

5.1. Optimization schemes

The choice of optimization scheme plays a central role in the performance of PINNs. This is mainly because PINN training involves minimizing a highly non-convex loss function, often leading to slow convergence, sensitivity to initialization, and possible trapping in local minima [27,54]. These difficulties become more pronounced for the Navier–Stokes equations, where the nonlinear convective terms and the coupling between velocity and pressure further complicate the optimization process [75]. In the specific case of LDC, Ding et al. [26] reported that conventional PINNs may converge to local minima, highlighting the need for robust optimization strategies.

According to Table 1 Adam and L-BFGS are the most common optimizers used to model LDC with PINN. Many studies chose to use Adam at the beginning of the optimization and then turn to L-BFGS. Rathore et al. [105] in a study focusing on the difficulties in training PINN argued that Adam + L-BFGS is usually better than Adam or L-BFGS individually. Jangir et al. [35] reported in their study on LDC that training begins with the Adam optimizer, whose stochastic updates promote fast exploration of the parameter space and help avoid shallow local minima. Once this initial stage yields a smoother loss landscape, the L-BFGS optimizer is used for deterministic refinement and final fine-tuning. A similar Adam + L-BFGS-B strategy was also used by Tan et al. [76] for the steady LDC case. However, they reported that applying L-BFGS-B after Adam increased the loss in the unsteady LDC case, and therefore omitted this fine-tuning step for that problem. Hao et al. [25] tried MultiAdam, a derivative of Adam developed by Yao et al. [106] on LDC and reported more than 18x increase in relative L_2 error, compared to the case optimized by Adam. A summary of comparisons made by Wang et al. [64] and Hao et al. [25] between different optimizers is presented in Fig. 13.

Given this, there is significant evidence that Adam + L-BFGS can be seen as the most reliable optimization scheme for modeling steady LDC with PINN. However, the transition point from Adam to L-BFGS is a questionable point. Most studies choose a constant number of Adam epochs and then continue L-BFGS until convergence [71]. This is while we know the whole point of using Adam before L-BFGS is to let L-BFGS to start from a better starting point. When the loss history of Adam can

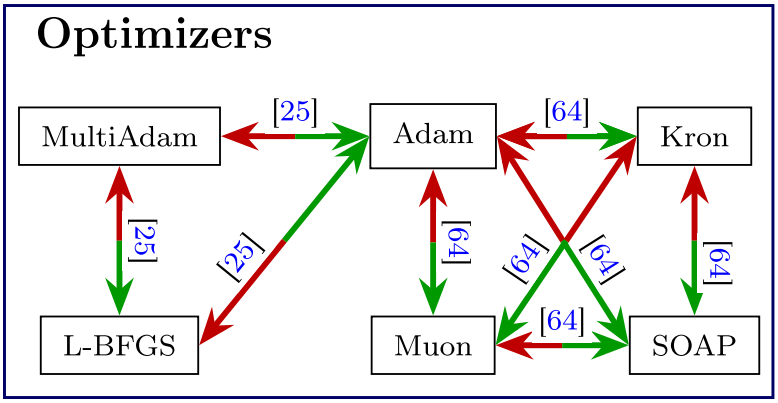


Fig. 13. Comparison between different optimizers for LDC in PINN literature. Citations inside boxes denote studies that used the corresponding optimizer, while citations on arrows denote studies that compared the connected optimizers. Green arrows point toward the better-performing optimizer, whereas red arrows point toward the underperforming optimizer in the cited comparison. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

be quite oscillatory, choosing a predetermined epoch number for Adam may lead to give a not appropriate starting point for L-BFGS.

To address this, some studies adopt convergence-based stopping rules rather than a hard epoch budget. For example, Tsai et al. [72] halted training when the average change of the loss components over the preceding 1000 epochs fell below a chosen threshold; otherwise, training continued up to a specified maximum. While such rules can reduce computational cost, their effectiveness depends on the selection of the threshold. On the other hand, it must be taken into account that the least amount of loss in the loss-history diagram is not necessarily equivalent to the least error value [17,35,55,87]. Therefore, as a suggestion, after allocating a sufficient training budget with Adam, it is recommended to examine the training trajectory and pick the model from the epoch that attains the lowest total error. Then, training with L-BFGS starts from that point. In addition, since the ultimate goal of training is to minimize the error rather than loss value, it is suggested to consider error rather than loss in methods like the one used by Tsai et al. [72] to find the best starting point of L-BFGS.

Despite the benefits and popularity of the Adam + L-BFGS method, it faces limitations. One of the most important limitations is the inability of L-BFGS to work with mini-batch training. Because of it, some efforts have been made to come up with either new optimizers or new optimization schemes based on the existing optimizers to solve LDC with PINN. Wang et al. [64] showed that optimizers of SOAP, Muon, and Kron, in order, lead to the least L_2 error for LDC at $\mathrm{Re} = 5 \times 10^3$, and all of them outperformed Adam in terms of error.

Beyond changing the optimizer itself, the optimization process has been adjusted in other ways to improve LDC modeling with PINN. Hu and Senocak [2] defined LDC in PINN as an equality-constrained optimization problem and solved it with an augmented Lagrangian method, rather than treating all physics and boundary terms as soft penalties in a loss function. Using this approach, they modeled LDC up to $Re = 7500$. Wei et al. [54] continued with Adam, but they added the residual correction loss, which can be interpreted as an optimization-informed loss function. They argued that their method reaches more accurate results than those found by Wang et al. [64]. It must be noted that the overall methods used in these two studies have other differences, other than the optimization procedure.

Insights from the literature. Adam, L-BFGS, and the sequence of these two have proved to be well compatible with many different setups of PINN, and the latter was shown to outperform each individual optimizer. Despite this, there is at least one study [64] reporting that a few less popular optimizers, including SOAP, Muon, and Kron optimizers, can lead to significantly more accurate results, and facilitate reaching $Re = 5000$. Also, other optimization-related adjustments, such as using the augmented Lagrangian method, enable PINNs to model LDC up to at least $Re = 7500$ [2]. Such potential to improve the results has not been seen in other design features of PINNs, such as governing equations selection and domain decomposition.

Room for further research. The substantial improvement obtained by tweaking the optimization scheme, which allows PINNs to reach higher *Re* numbers without requiring additional data points, together with the repeated reliance on Adam and L-BFGS across many studies, lends support to the statement by Urbán et al. [27] that, because loss function modifications are easier to implement, they have received more attention in PINN research than optimization methods. This suggests the need for a more systematic study of the optimization process from several perspectives, including the use of new optimizers, as explored by Wang et al. [64], the tuning of Adam and L-BFGS hyperparameters, and the incorporation of optimization-induced components into the loss function, introduced by Wei et al. [54].

5.2. Training strategies

Training strategy refers to the overall way in which a PINN is deployed to achieve a given objective. The choices made in problem definition, neural network configuration, and optimization can be viewed as the design of a machine, while the training strategy determines how that machine is used to perform the task. As shown in Fig. 14, the training strategy for applying PINNs to LDC can be described, in broad terms, by two questions of where to start the training process and how to proceed thereafter. The most common starting point is random initialization of the model parameters. However, some studies report that initializing the network with the weights and biases of a pretrained model on a related problem can be highly beneficial. Such pretrained models may have been trained at different *Re* numbers or on different geometries. Once the starting point has been selected, the next step is to determine how training should proceed. In most studies, the hyperparameters h, including the collocation points and the weights assigned to the loss terms, are chosen before training and kept fixed throughout the optimization process. However, these hyperparameters may also be adjusted gradually or modified at specific epochs, an approach that can be described as multi-phase training. They may also be treated as trainable quantities, denoted by $h(\theta)$, that evolve continuously during training. The approaches adopted by different studies on LDC are discussed in the following.

Where to start. Most studies on LDC have been initialized using random initializers such as Xavier method [2,18,19,22,55,64,69,75,78] or He

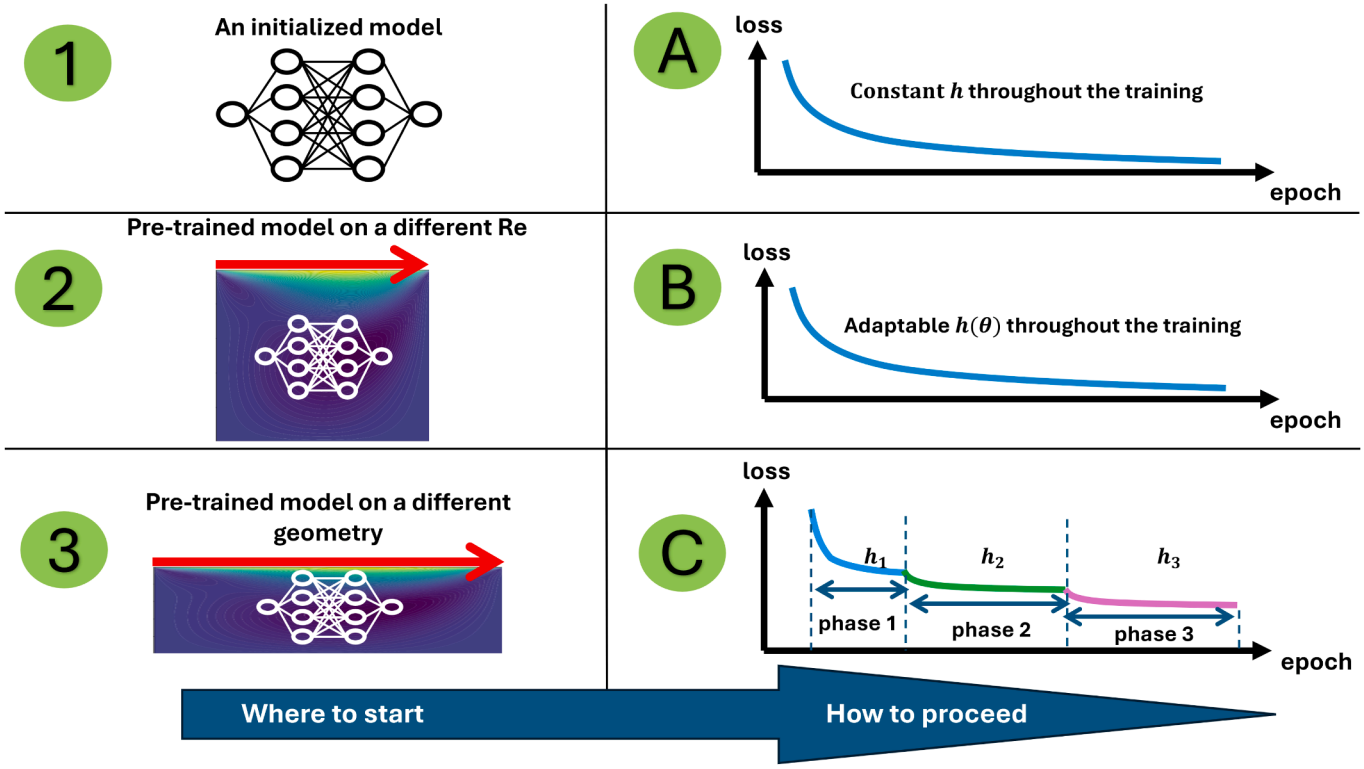


Fig. 14. Different training strategies from two distinct perspectives: the initialization of model parameters (1–3) and tuning hyperparameter during the training (A–C).

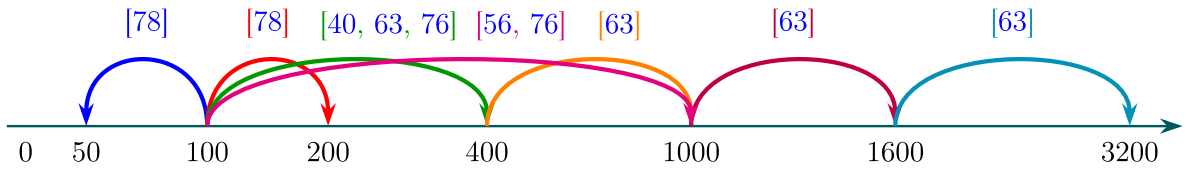


Fig. 15. Overview of transfer learning across different *Re* numbers in the literature on using PINNs for LDC.

method [69,70]. Starting from PINNs pre-trained on LDC at lower *Re* numbers has been suggested by Wang et al. [62] as one of their concluding remarks. As shown in Fig. 15, this type of transfer learning has been used across different *Re* numbers. Gu et al. [40] used a pretrained model at $Re = 100$ to model LDC at $Re = 400$. Takao and Li [78] also used a model trained at $Re = 100$ to model LDC at $Re = 50$ and 200. Wang et al. [63] have used transfer learning (they called curriculum learning) to use models pretrained on lower *Re* numbers for modeling LDC at higher *Re* numbers. Jangir et al. [35] stated that to train PINN at $Re = 500$-1000, they used a pretrained model at a lower *Re* number. Another approach that can be seen as something between transfer learning and using data points is what Lin et al. [21] introduced, which is connecting data from a pre-trained DeepONet through an intermediate data-driven neural network for training PINNs.

Transfer learning across different geometries has been explored by Takao and Ii [78] and Roy et al. [1]. Takao and Ii [78] considered pre-trained PINN models for LDC in several geometries, including the configurations shown in Fig. 3a, d, f, and g. Their study used a model pre-trained on the square cavity to initialize models for rectangular and shear-deformed cavities, and also examined transfer learning among different nonlinearly inflated cavity geometries. Roy et al. [1] used transfer learning for their domain decomposition technique. They first trained a model on the entire domain. Then, they used that pretrained model as the base for training models for the subdomains.

How to proceed. Changing hyperparameters h, including loss term weights, optimizer learning rates, and collocation point distributions during training, has been found to be an effective strategy for training PINNs in general and for modeling LDC in particular. Khadijeh et al. [74] employed adaptive weights for the loss terms and showed that, compared with fixed weights, this approach leads to smoother convergence. Adaptive loss weighting has also been reported to be beneficial in several other studies on LDC [20,36,50,61,62,66,68]. Lin et al. [21] reported that, for modeling LDC at relatively high *Re* numbers of $Re = 2000$, 3000, and 5000, training was performed in three stages, each lasting several thousand epochs and using different numbers of collocation points and loss term weights. Learning-rate scheduling has likewise been shown to improve optimization in modeling LDC [19,33,45,46,54,60,70,73,81,82]. Mini-batch training is another strategy that has been used for solving LDC with PINNs [54,70]. However, as noted by Wei et al. [54], this choice was primarily motivated by computational limitations and is unlikely to provide an intrinsic advantage in enhancing PINN performance for LDC modeling.

The design of the training process is not limited to specifying whether hyperparameters change during training. Other novel approaches have also been used for modeling LDC with PINN. Khadijeh et al. [74] introduced a multistage PINN that focuses on learning u, v, and p through three consecutive stages. Tsai et al. [72] leveraged transfer learning in the sense that they used a tailored sequence of colloca-

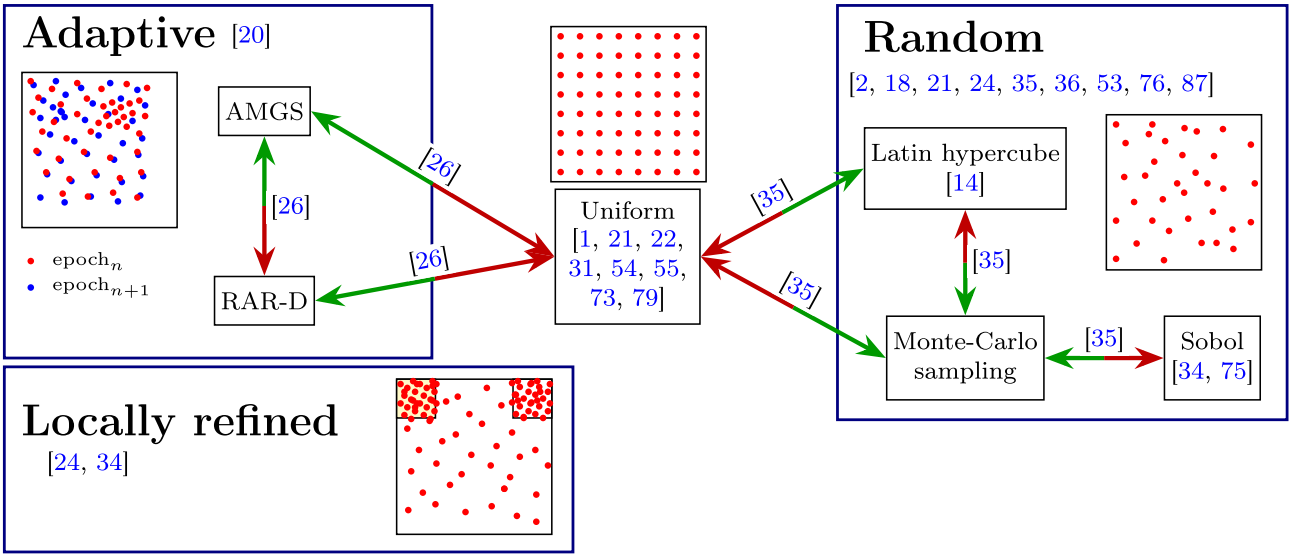


Fig. 16. Comparison between different collocation-point sampling methods for LDC in PINN literature. Citations inside boxes denote studies that used the corresponding sampling method, while citations on arrows denote studies that compared the connected sampling methods. Green arrows point toward the better-performing sampling method, whereas red arrows point toward the underperforming sampling method in the cited comparison. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

tion points to train PINN up to $Re = 5000$ and reported 60% accuracy improvement.

Insights from the literature. There is broad agreement that initializing a PINN with a model pretrained on a related problem can improve training. However, this strategy is most practical when several related cases must be solved. When only a single case is required, transfer learning may demand additional simulations that are not otherwise needed. Its improved convergence therefore comes at the cost of additional preliminary training.

5.3. Collocation points distribution

The impact of the distribution of collocation points on modeling PINN with LDC has been studied extensively in the literature. As depicted in Fig. 16, different sampling methods can be categorized into four different groups of uniform, random, adaptive, and locally refined distributions. These four types of grids are being reviewed in the case of LDC in the following.

Uniform grid. A uniform distribution makes the training process unbiased w.r.t. different parts of the domain, and many studies have chosen that [1,19,21,22,31,54,55]. It also makes the results reproducible, as reproducing non-uniform clouds of collocation points can be very hard. In addition, for some implementations, like implementing the additive trial function [107], uniform distribution reduces the computational cost. However, generally a uniform distribution puts focus on some unnecessary areas that increase the computational cost, and reaching a good mesh density around important regions demands a very high number of collocation points [31]. Although this approach is not perfect, it has not been a hurdle to reach high *Re* numbers like $Re = 10000$ with PINN for LDC [54,55].

Non-uniform grid. Non-uniform grids encompass a few types of grids, including random grids, semi-random grids, and uniform grids with varied densities in different locations. Using a random grid with no specific concentration point has been used in some papers [2,21,24,87]. If these schemes manage to provide a conclusive cover, they provide an unbiased grid as a uniform grid. Farea et al. [75] used the Sobol sequence, which is a quasi-random method, and is more space-filling than random sampling. Jangir et al. [35] compared the Sobol method with two random methods of Latin hypercube sampling and Monte Carlo sampling, and a uniform grid. Sobol took the fewest epochs to reach a specific loss value, but had the highest error at that point, which makes the judgment difficult. However, Monte Carlo sampling had fairly the best balance between accuracy, robustness, and computational efficiency among all competitors. The Latin hypercube sampling method also had a better performance than the uniform grid.

As shown in Fig. 16, another type of grid used for LDC in PINN is a locally refined distribution, where collocation points are concentrated in challenging regions of the domain. The most challenging regions in LDC are where secondary vortices appear, top corners, and also where the first and second order derivatives of velocity components rapidly change. Pandey et al. [34] put extra points near the top corners at Re = 400, where they reported improvement.

Adaptive grid. Ding et al. [26] compared three adaptive-sampling-based strategies: (1) the residual-based adaptive refinement with distribution (RAR-D), which redistributes collocation points according to the PDE residual so that regions with larger residuals receive denser sampling; (2) the combined gradient-enhanced PINN (gPINN) & RAR-D, which supplements this residual-driven refinement with gradient-based loss terms [108] to enforce smoother and more stable physics learning; and (3) the Adaptive Multi-Scale Gradient-Enhanced Sampling PINN (AMGS-PINN), which further integrates residual-based adaptive sampling with adaptive multi-scale gradient regularization, using fine-scale residual-driven points together with coarse-scale points selected by K-means clustering to improve both local accuracy and global training stability. In the paper, AMGS-PINN is presented as the most effective of these strategies because it couples adaptive sampling with multi-scale regularization rather than relying on residual refinement alone.

The number of collocation points. To the best of our knowledge, there is no theory that determines what the optimal number of collocation points in modeling with PINN is. This problem also persists in the case of modeling LDC. The problem is that the finer grid sometimes has a negative impact on the accuracy. Jangir et al. [35] reported this controversy with regard to the PINN with automatic differentiation (AD-PINN) for LDC, and Roy et al. [1] reported almost the same with their PINN that used

the finite difference discretization for the calculation of residuals (FD-PINN), no matter LDC was solved via a single domain or in a domain decomposition scenario. However, compared with CFD, the computational cost of AD-PINN and, to an even greater extent, FD-PINN are less sensitive to increases in grid resolution [55].

Insights from the literature. There are inconsistent conclusions from different studies. The uniform method generally performs poorly compared to other methods in different studies (Fig. 16). However, some PINNs using it have successfully reached high *Re* numbers up to 10000 [54]. Given this uncertainty, the grid design process currently is more of a trial-and-error process, in both realms of the grid type and resolution.

Room for further research. A promising direction for future PINN studies is the development of a grid that combines the comparability, reproducibility, and space-filling properties of uniform meshes, the compatibility of structured layouts with approaches such as separable PINN [73] and additive trial functions [107], and the point-efficiency of locally refined meshes. Since most LDC are defined on square or rectangular domains, a structured yet non-uniform grid with local refinement appears to be a particularly suitable candidate. If such a grid is made adaptive during training, it could further unify the advantages of uniform, locally refined, and adaptive meshing strategies. To the best of the authors' knowledge, no LDC study has yet investigated this type of grid.

6. Loss function, evaluation metrics, and some selected results

6.1. Loss function

The loss function of a PINN, in essence, is composed of a summation of some loss terms enforcing governing equations, initial or boundary conditions, and consistency with some field data. Whether each of these constraints must exist in the loss function, a new constraint must be added, and the relative contribution of each term are matters of design choice. By default, the loss function of a PINN designed to model the steady 2D LDC via VP formulation of the Navier-Stokes equation had three loss terms for the continuity and the momentum conservation equation in x and y directions. The Dirichlet boundary conditions of the top moving wall and the stationary walls are also usually set as loss terms.

6.1.1. Weight treatment

Weight treatment determines the relative importance of the loss terms and their distribution across the spatio-temporal domain, Eq. (8). The default fixed-weight loss formulation has been identified as a limiting factor [68]. However, manual manipulation of the weights also should be taken with caution, as Hao et al. [25] showed that increasing the weight of the boundary losses of LDC from 1 to 100, when weights of the governing equations are 1, increases the relative L_2 error by more than 3x. One of the most effective strategies is the improved adaptive weighting method proposed by Niu et al. [68]. This approach updates the loss weights adaptively while imposing an upper bound to prevent any weight from exceeding a prescribed limit. Using this method, they reduced the relative L_2 error for LDC at $Re=100$.

Another weight treatment method that was designed specifically for LDC is to apply spatially varying weight functions to the loss terms, developed by Satyadharma et al. [31]. This technique aims to address corner discontinuities. They simultaneously employed weights on both the PDE residual and boundary loss components to suppress the influence of regions near the top corners. These weights are used in the loss terms for the governing equations and the moving wall boundary condition, respectively, as in

$$\mathcal{L}_{\mathcal{F}} = \frac{1}{N_C}\sum_{i=1}^{N_C} \omega_{\mathcal{F}} \left|\mathcal{F}(\hat{\mathcal{P}}_\theta(z_i))\right|^2, \tag{12}$$

$$\mathcal{L}_{\mathcal{B}} = \frac{1}{N_{\mathcal{B},\mathrm{mw}}}\sum_{i=1}^{N_{\mathcal{B},\mathrm{mw}}} \omega_{\mathrm{mw}} \left|\mathcal{B}_{\mathrm{mw}}(\hat{\mathcal{P}}_\theta(z_i)) - \mathcal{B}_{\mathrm{Ref}}(z_i)\right|^2 + \frac{1}{N_{\mathcal{B},\mathrm{nmw}}}\sum_{i=1}^{N_{\mathcal{B},\mathrm{nmw}}} \left|\mathcal{B}_{\mathrm{nmw}}(\hat{\mathcal{P}}_\theta(z_i)) - \mathcal{B}_{\mathrm{Ref}}(z_i)\right|^2, \tag{13}$$

where

$$\omega_{\mathcal{F}} = 1 - 0.99\exp\left(-20x^2 - 20(y-1.0)^2\right) - 0.99\exp\left(-20(x-1.0)^2 - 20(y-1.0)^2\right), \tag{14}$$

$$\omega_{\mathrm{mw}} = 1 - 4(x-0.5)^2. \tag{15}$$

N_C, $N_{B,\mathrm{mw}}$, and $N_{B,\mathrm{nmw}}$ denote the number of collocation points in the entire domain, on the moving wall, and on the non-moving walls, respectively. Here, $\hat{\mathcal{P}}_\theta(z_i)$ represents the physics-informed neural network's prediction of the solution at the collocation point $z_i = (x_i, y_i)$, with θ denoting the network parameters. The effects of these weights are visualized in Fig. 17. The left subplot shows the contour of $\omega_{\mathcal{F}}(z)$, with the darkest regions near the top corners indicating the highest reduction in loss weighting. The right subplot illustrates $\omega_{\mathrm{mw}}(z)$ along the top wall, where it reaches its maximum at the center and tapers to zero at the side edges, thereby reducing the boundary loss' influence where the velocity gradients are steepest. This method was successful up to $Re=200$. However, at $Re>500$, the simulation needed data points.

Insights from the literature. The spatially varying weight functions do not enable PINNs to resolve high-gradient regions in the top corners. Instead, they make the PINN put less attention to those regions. However, this method can be seen as a safer option compared to using smoothed boundary conditions that were suspicious to dramatically change the flow pattern.

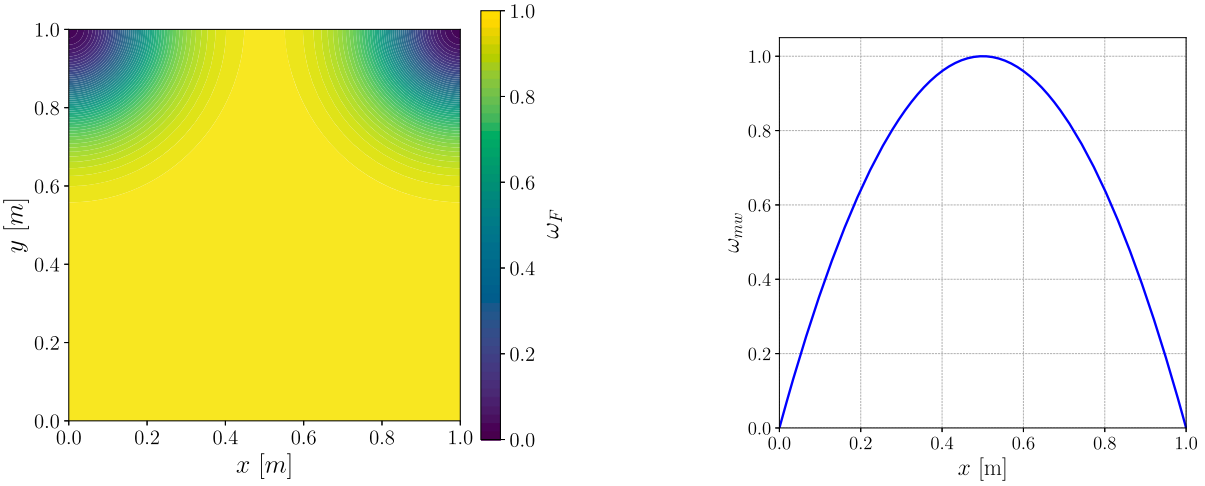
Room for further research. As a direction for future research, the weighting functions employed in this work may be further refined or optimized. Modifying their concentration zones would introduce additional hyperparameters, whose effects on the training dynamics and prediction accuracy remain unexplored. Also, despite this mixed-weight strategy demonstrated improvements in stability and convergence, the authors [31] did not investigate the individual effects of each weight nor compare their weighted formulation against an unweighted baseline.

6.1.2. Inclusion of data points

Interior data points, i.e., field measurements that are not associated with boundary conditions, can be incorporated into the training process for several purposes. First of all, they provide an effective means of improving predictive accuracy and can facilitate the modeling of LDC at higher *Re* numbers. In addition, data points can be used in inverse problems, for example to infer unknown physical parameters or reconstruct missing flow quantities [21,24,32,37,69]. Such inverse applications, however, are beyond the scope of the present study.

Fig. 18 summarizes some of the *Re*-number ranges reported in the literature for LDC. Green bars indicate *Re* numbers at which LDC was modeled without using data points, whereas the ranges beyond which data were introduced are marked with a red arrow. In several studies, there is a gap between the highest *Re* number reported for zero-data PINN modeling and the first *Re* number at which data points were used. Because it is unclear whether the PINN could successfully model LDC without data within those intermediate ranges, these intervals are shown in gray. This figure does not include all papers presented in Table 1. For example, many studies modeled LDC at $Re=100$ with no data, and representing them here will not add any new information.

In Fig. 18, only the best-reported result from each paper is shown. As the figure shows, the *Re* numbers at which data points were used span a wide range, from 400 to 5000. This is because the literature reveals substantial variability in the performance of data-free PINNs, where some



(a) Distribution of $\omega_{\mathcal{F}}$ (b) Distribution of ω_{mw}

Fig. 17. Distributions of ω_F and ω_{mw} over the entire domain and the top wall, respectively. Both weights diminish the loss influence at the top corners.

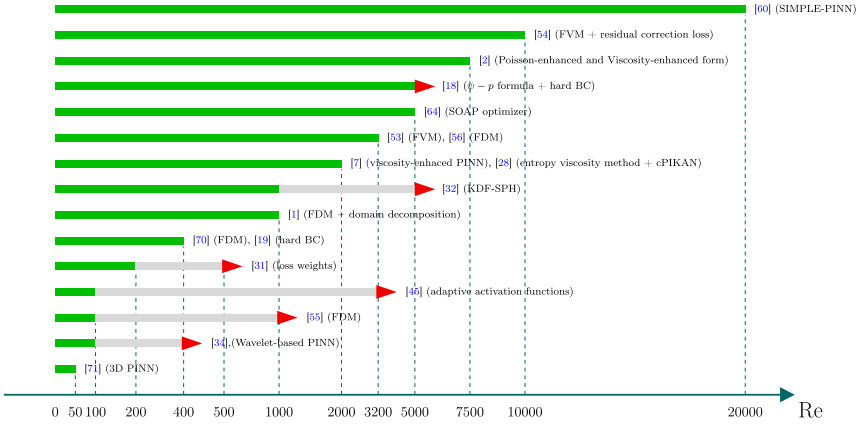


Fig. 18. The zero-data *Re* number limits reported for modeling LDC with PINNs.

studies report unsatisfactory results even at much lower *Re* [17] numbers, whereas others achieve accurate solutions at considerably higher *Re* numbers without using data points [54].

Data points distribution. In the same way that distributing collocation points was a design choice, as discussed in Section 5.3, data points have also been distributed in the domain of LDC differently. Fig. 19 shows four main ways of distributing data points in the LDC domain.

Placing data points uniformly was the choice of Geng et al. [29], where they used 81 data points of (u, v) uniformly distributed and covering boundaries, considering they did not define separate boundary loss terms. They did it at $Re = 100$. Satyadharma [31] compared the uniform distribution, distribution along high second-derivative locations, and the mixed scheme. They concluded that the locations with higher second derivative values are most effective for placing data points. Satyadharma [31] investigated different numbers of data points for the velocity components (u, v), all below 100. They found that the best position to put data points is where the second derivative of the outputs is the highest, in contrast to the belief that it should be put where the gradient (first derivative) is the highest.

Unlike collocation point distribution design, random distribution for the data points is not always a choice, rather it is sometimes the only option, as available data might be sparse. Feng et al. [32] used 50 data points randomly distributed throughout the domain. Jiang et al. [55] used 10, 25, 50, 100 data points (u, v, p). They used randomly located data points, with more density near the boundaries and corners. They reported that with these data points, both AD-PINN and FD-PINNs were successful in capturing the main vortex, but FD-PINN was significantly more successful in capturing secondary vortices. Bai et al. [14] also used scattered data points at $Re = 100$, but the details about the number of points have not been reported. Wang et al. [7] also used 100 randomly selected points to model LDC at $Re = 2000$.

McDevitt et al. [18] used only u data at 15 positions along the center vertical line, and it helped them to capture the flow up to $Re = 21000$. Pandey et al. [34] modeled LDC at $Re = 100$ without data, but for $Re = 400$ they used four points located almost on the horizontal and vertical centerlines. Wang et al. [7] used two scenarios of using 1 and 5 data points, almost on the horizontal and vertical centerlines. Wang et al. [7] stated that even one data point can diminish the roughness of the loss landscape. Sun et al. [30] developed an optimization-based method to find the best position for data points. They also showed that adding more data points does not necessarily improve the results.

Effects of noise. Several studies have investigated the robustness of PINNs to noisy training data in modeling LDC. Feng et al. [32] compared PINN performance using clean data with cases containing 1% and 4% noise, showing that prediction accuracy generally deteriorates as the noise level increases. Jiang et al. [55] and Satyadharma et al. [31] also examined the effect of noisy data. Jiang et al. [55] reported that FD-

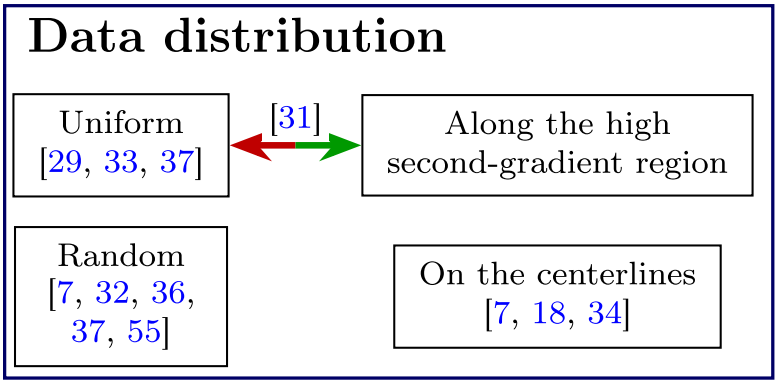


Fig. 19. Comparison between different data distribution methods for LDC in PINN literature. Citations inside boxes denote studies that used the corresponding methods, while citations on arrows denote studies that compared the connected methods. Green arrows point toward the better-performing method, whereas red arrows point toward the underperforming method in the cited comparison. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

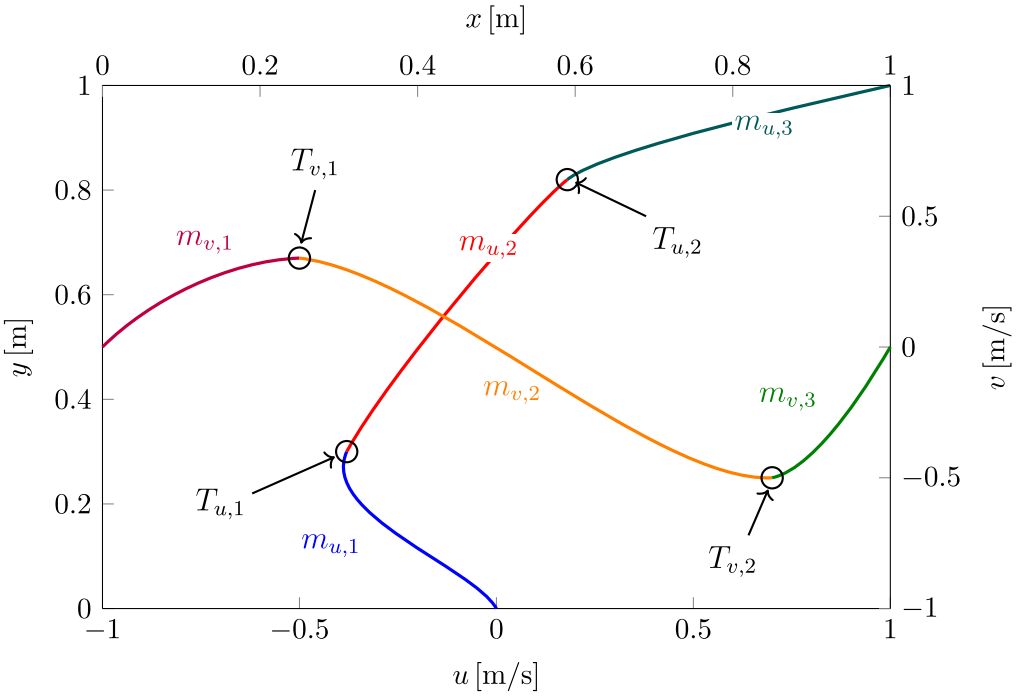


Fig. 20. Schematic representation of the centerline velocity profiles in LDC. Horizontal velocity u along the vertical centerline, $u(y)$, and vertical velocity v along the horizontal centerline, $v(x)$. Each profile is divided qualitatively into three approximately constant-slope regions, separated by two turning points where the slope changes. Colored segments highlight the piecewise-linear interpretation of the profiles.

PINN is generally more robust than AD-PINN under noise, although AD-PINN exhibited somewhat inconsistent behavior and, in some cases, performed competitively. Similarly, Geng et al. [29] and Liu et al. [33] assessed measurement uncertainty by adding Gaussian noise to the training data.

Insights from the literature. The conclusion drawn by Satyadharma et al. [31] suggests that PINNs struggle more at high second-derivative regions, rather than high first-derivative regions. Although the verification of this statement demands a systematic study, a quick verification is possible by focusing on the velocity components profile over centerlines. If we consider Fig. 20 as the schematics of velocity profiles over the centerlines, at most *Re* numbers studied in the literature, each profile can be made of three segments, shown by $m_{u/v,1/2/3}$ having small slope changes, and two turning points, $T_{u/v,1/2}$, where the slopes drastically change. It should be mentioned that $T_{u,1}$ starts to appear as *Re* number goes up. According to Satyadharma et al. [31], PINNs struggle at these turning points.

Now, Fig. 21, reprinted from the study of Jiang et al. [55], shows at $Re=100$, the u profile of the less capable PINN, i.e., the AD-PINN starts to deviate from the correct profile near $T_{u,2}$, although it was very successful in capturing the high-gradient region before this turning point. For the v profile also, $T_{v,1}$ and $T_{v,2}$ are discrepancy regions. This issue better presents itself in $Re=1000$, where at high y values, the steep u change is captured well with AD-PINN, but it misses the slope change at $T_{u,2}$. This failure disconnects the excitation caused by the top moving wall and the rest of the domain, which makes the v component zero along the centerline. This behavior is not limited to AD-PINN, and the small discrepancies of the FD-PINN at $Re=1000$ also follow the same pattern.

Room for further research. Many developments and innovations in PINNs are based on this assumption that PINNs struggle with the high-

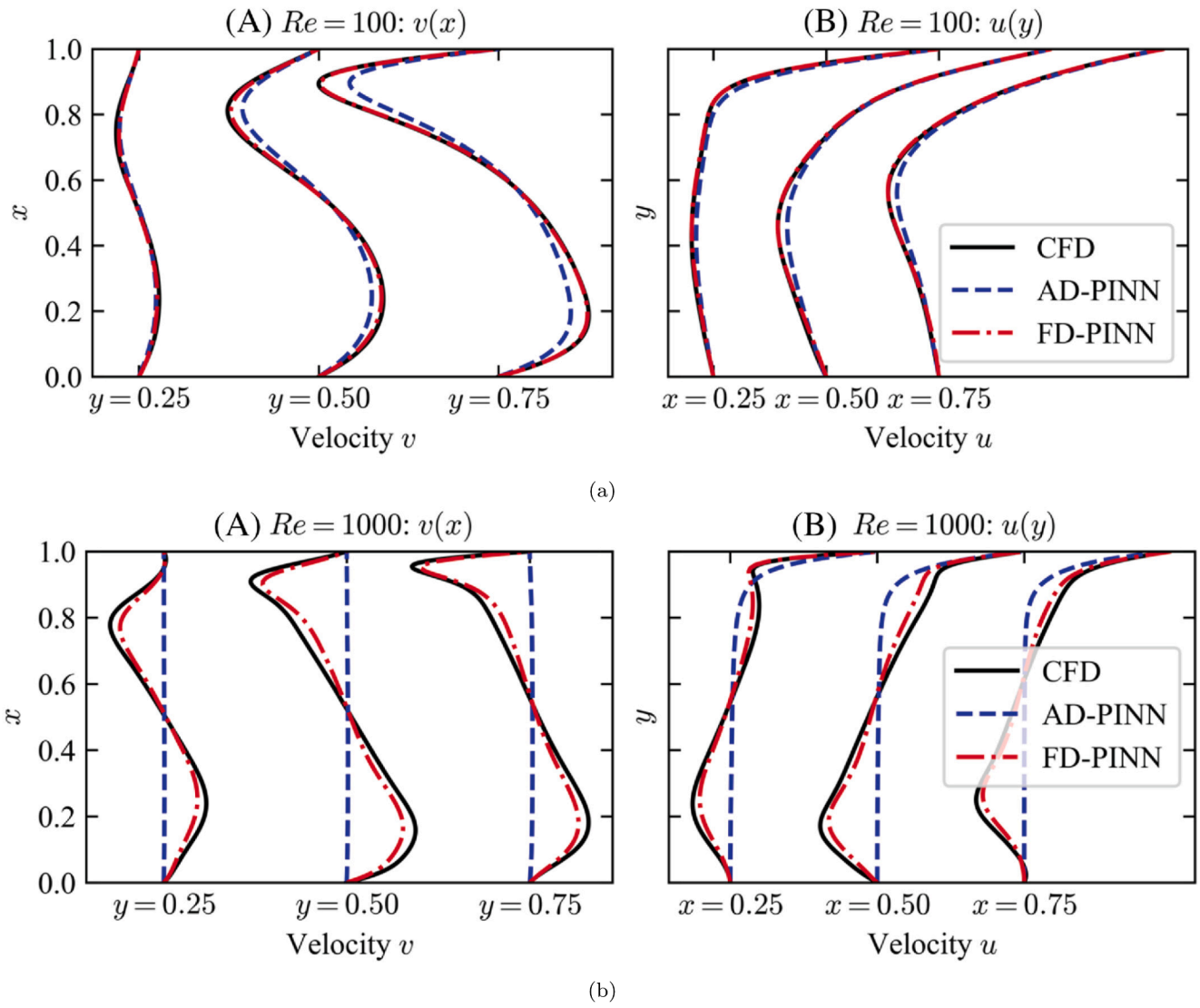


Fig. 21. Velocity components along horizontal and vertical profiles. Reprinted from Jiang et al. [55], used under the Creative Commons CC BY-NC-ND 4.0 license (https://creativecommons.org/licenses/by-nc-nd/4.0/).

gradient (first derivative) regions. If the regions with high second derivative values are also hard to deal with in PINNs, or maybe more difficult than high-gradient regions, all those methods developed based on these assumptions need to be revisited. Design of subdomains in domain decomposed PINNs and the distribution of collocation points are two examples.

6.1.3. Derivative calculation in the loss function

Vanilla PINNs (AD-PINN) rely on AD to compute the derivatives of the network outputs w.r.t. the inputs when forming the PDE residuals and other loss terms. However, AD is not the only viable choice, and the recursive application of the chain rule becomes increasingly expensive and cumbersome for high-order derivatives [32]. Several studies have also reported that conventional AD-PINNs struggle to accurately model LDC at $Re \geq 100$ [32,55], which has motivated the development of discretization-based alternatives for evaluating the derivatives appearing in the loss function. Once this route is taken, the PINN formulation is no longer strictly mesh-free, and the required derivatives can instead be approximated using standard numerical schemes such as the finite difference method (FDM) [1,55] and the finite volume method (FVM) [53,54]. One key motivation for incorporating discretization into PINNs is that vanilla PINNs typically do not explicitly exploit information from neighboring points, which can weaken the enforcement of physical constraints and reduce solution accuracy [54]. In contrast, discretization introduces explicit algebraic coupling between each collocation point and its neighboring nodes.

Finite-difference method (FDM). Jiang et al. [55] used FDM for the calculation of the residual values for LDC up to $Re = 10000$. They showed that by using FDM for the calculation of residuals, the secondary vortices can be predicted much better, but they also stated that for $Re = 1000$ and higher, data points should be used to improve accuracy. Jiang et al. [55] showed that PINNs using FDM (FD-PINNs) required approximately one-eleventh of the training time of those using AD (AD-PINNs) for the same number of Adam epochs. FD-PINN also proved faster convergence and less loss oscillation. However, in terms of RMSE accuracy, FD-PINN has not always been more accurate than AD-PINN. Another positive feature of the FD-PINN is that the boundary conditions can be perfectly satisfied [55]. Roy et al. [1] proved that the benefit of FD-PINN can be further improved by domain decomposition, and modeled LDC up to $Re = 1000$. However, since they have not compared the results with AD-PINN, the impact of the discretization cannot be learned from this study. Chiu et al. [70] developed a combined method that uses FDM and AD to find the residual values for solving LDC with PINN at $Re = 400$, while AD-

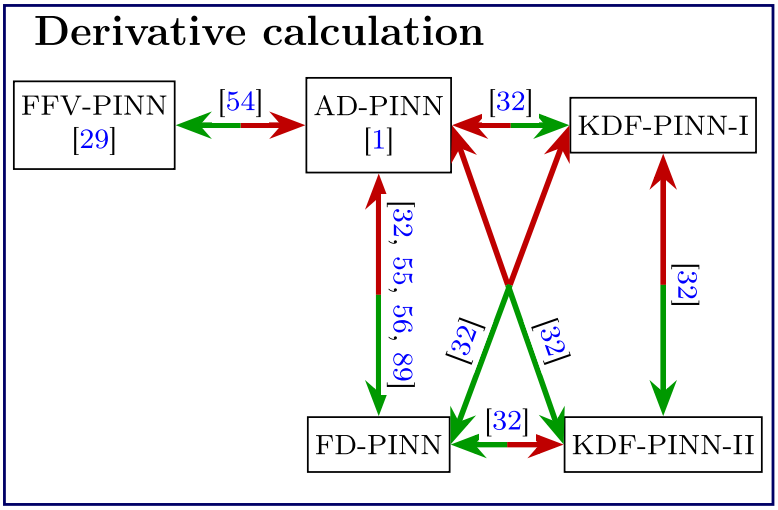


Fig. 22. Comparison between different derivative calculation methods for LDC in PINN literature. Citations inside boxes denote studies that used the corresponding methods, while citations on arrows denote studies that compared the connected methods. Green arrows point toward the better-performing method, whereas red arrows point toward the underperforming method in the cited comparison. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

PINN could not reach that *Re* number. Although their method showed improvement in accuracy, the computational time was even higher than AD-PINN (but AD-PINN effectively failed to reach the solution, which makes the computational time irrelevant).

Finite-volume method (FVM). Wei et al. [53] used FVM and trained a PINN for modeling LDC up to $Re = 3200$ without data points. They showed that using the FVM method enables the PINN to capture secondary vortices, while the AD-PINN was not able to do that even at $Re = 1000$. They used the central difference scheme for discretizing both diffusion and convection terms. The boundary conditions have been treated as soft constraints, having a special term in the loss function. In the same way that using the FDM method [55] decreased the computation time, Wei et al. [53] also reported up to one thirteenth time to the same number of L-BFGS under identical settings. Wei et al. [54] in another study used a simplified form of the FVM and reached $Re = 10{,}000$ without data points, capturing the secondary vortices with good accuracy. It must be noted that using this FVM method was not the only novelty their model had over the vanilla PINN. The architecture of their neural network was equipped with a frequency mapping layer, a combination of shared and variable-specific hidden layers, and a new loss term, called the residual correction loss term. Geng et al. [29] addressed one of the main limitations of earlier FVM-based PINNs, namely their reliance on predefined meshes, by introducing a formulation that generates local control volumes around sampled points and thus avoids time-consuming mesh generation.

Other methods. Feng et al. [32] proposed a hybrid derivative-evaluation framework for PINNs in which kernel derivative-free smoothed particle hydrodynamics (KDF-SPH) is used to approximate spatial derivatives in the interior of the domain, while finite differences are employed near the boundaries. Within this framework, they introduced two variants, KDF-PINN-I and KDF-PINN-II, where the former enforces boundary conditions through soft constraints in the loss function, whereas the latter imposes them as hard constraints. The corresponding comparison for LDC is presented in Fig. 22.

Insights from the literature. Studies indicate that, in most cases, discretization-based residual evaluation improves LDC predictions, particularly in capturing complex flow features such as secondary vortices, while also significantly reducing computational time. These advantages make discretization-based methods strong candidates for PINN loss function design. Their main drawback, however, is that they sacrifice the meshless nature of PINNs and require additional meshing. Still, given the computational savings they offer during training, this added discretization effort may be justified.

Room for further research. Both FDM and FVM are well-established methods with many variants. Given the substantial gains in both accuracy and computational efficiency achieved through these approaches, there is strong motivation to adapt advances from classical discretization methods to the PINN framework. This potential is illustrated by Wei et al. [54], who simplified the FVM procedure by leveraging PINN-specific capabilities and extended the achievable range to $Re = 10000$, compared with the highest reported *Re* of $Re = 3200$ in their earlier study [53].

Another promising direction is a systematic comparison between discretization-based PINNs and transformer-based architectures. Both approaches are motivated by the idea of incorporating information from neighboring points during training, yet they do so through fundamentally different mechanisms. A careful comparison could clarify their respective strengths, limitations, and computational trade-offs for challenging flow problems such as LDC.

6.2. Evaluation metrics

Table 5 summarizes the main evaluation metrics used in the literature to study LDC with PINN. The central point is that different metrics test different aspects of solution quality. As a result, two studies may reach different conclusions about the same method. A PINN can match centerline velocity profiles well, yet still miss the correct secondary vortices or retain large local errors elsewhere in the domain. Performance claims should therefore be read together with the evaluation protocol.

Profile-based comparisons along horizontal and vertical lines, especially the cavity centerlines, are widely used because they connect directly to established benchmark data in the literature [18,34,55]. This makes them attractive when the goal is to compare against published results without running a new CFD simulation. Their limitation is that they probe only a few cuts of the flow and may miss errors in the rest of the domain, especially in the secondary vortices. This issue is important at moderate and high *Re* numbers, where local recirculation structures are part of the main physics [55].

Field-wise contours provide broader spatial information. Velocity contours have been used to visualize the predicted flow field itself [2],

Table 5
Comparison of evaluation metrics used for LDC in PINN literature.

Metric	Error concentration recognition	Quantitative comparison	Flow pattern recognition
Average error	No	Yes	No
Error contours	Yes	Partially	No
Velocity contours	Partially	Partially	Partially
Streamlines	Partially	Partially	Yes
Profiles along horizontal and vertical lines	Partially	Partially	No
Loss history	No	Yes	No
Reference-free metrics	Partially	Yes	No

while error contours have been used to show where the PINN fails relative to a reference solution [1,26,54]. These plots are useful for identifying error concentration near the moving lid, near the corners, or in thin shear layers. Still, they do not describe vortex topology as clearly as streamline plots, and quantitative error contours require a reliable reference field.

Streamline plots are one of the most direct tools for assessing the flow topology, as they reveal the presence, position, and morphology of the primary and secondary vortices [1,2,18,21,53,54]. This is particularly useful for LDC problem, where vortex formation and evolution are central features of the flow. However, streamline plots are often interpreted qualitatively. Unless quantities such as vortex-center locations, vortex sizes, and secondary-vortex positions are extracted, streamlines alone do not provide a complete quantitative metric. This limitation is consistent with the observation of Lin et al. [21], who noted that comparing different cases solely from streamline plots can be difficult.

Scalar error measures such as mean squared error (MSE) [35,54], root mean squared error (RMSE) [55], mean absolute error (MAE) [72], relative L_2 error [21,40,50,54,68,72], and the coefficient of determination (R^2) [35] are useful for compact comparison across methods and Re numbers. However, they reduce the whole field to one number and may hide localized failure in physically important regions. Loss history is also commonly reported [2,18,21,74,78], but it should be treated as an optimization diagnostic rather than an accuracy metric because lower loss does not always imply lower solution error [17,35,55,70,87].

A useful additional option is the reference-free residual-based metric proposed by Urbán et al. [27], which evaluates the normalized L_2 norm of the discretized governing-equation residual on a test grid. This is especially relevant when a trusted reference solution is unavailable. Apart from point-wise error metrics, Zhang et al. [73] have used some metrics common for image quality assessment for their study on LDC, which is out of the scope of the current study.

The benchmark study by Ghia et al. [109] is among the most used references to evaluate the PINN results for LDC [1,2,34,54,55,72]. Other references used for this purpose are Botella and Peyret [110], Gupta and Kalita [111], Li et al. [112], Tian and Yu [113], Bruneau and Jouron [114], Erturk et al. [115], and Cortes and Miller [116].

Insights from the literature. The most comprehensive assessment of PINN results for LDC, both in terms of evaluating solution quality and enabling comparison across studies, is obtained by combining scalar error measures, error contours, velocity contours, and streamlines. This strategy is preferred when reliable reference solutions are available and the computational cost of generating them is acceptable. When evaluation must rely on benchmark data reported in the literature rather than full reference fields, comparisons along horizontal and vertical centerlines provide a practical alternative. In the absence of any reference solution, reference-free residual-based metrics become particularly valuable [27]. Loss history may still be reported as a supplementary indicator of optimization behavior, but it should be interpreted cautiously, since a lower loss does not necessarily correspond to a more accurate solution.

6.3. Some selected results

Figs. 23–26 summarize selected quantitative results reported for $Re = 100$, 2000, 3200, and 5000, respectively. In cases where a study reported several methods or configurations, only the lowest reported error from that paper is included. The bars are ordered to facilitate comparison, and a logarithmic axis is used to accommodate the wide range of reported error values. These figures provide a more direct comparison than a study-by-study description, but several differences among the original investigations prevent them from constituting a fully controlled benchmark. The studies use different error metrics, including relative L_2 error, RMSE, and MSE, and they do not always evaluate the same physical quantity. Some report the error of the velocity magnitude, V, whereas others report separate errors for u, v, and p. The spatial support of the error also varies, as in some studies it is evaluated over the entire computational domain, while in others the comparison is restricted to one or more centerlines. In addition, the reference solution may be drawn from established benchmark studies, such as Ghia et al. [109], or generated independently by the authors using a conventional numerical solver. Consequently, the lowest reported error should be interpreted as the best reported value under the conditions of the corresponding study, rather than as proof that the underlying method is universally superior.

At $Re = 100$, Fig. 23(a) presents the relative L_2 errors of the velocity magnitude. However, even at the same Re number and for the same nominal metric, the comparison is only partially aligned because the studies use different reference fields, numerical solvers, grids, and evaluation points. Among the studies included in this panel, Chen et al. [50] report the lowest error and therefore provide the strongest reported result for this particular comparison. However, their method was demonstrated only up to $Re = 600$, and its performance at substantially higher Re numbers remains unknown. By comparison, Cao and Zhang [87] and Wei et al. [54] applied their methods up to $Re = 5000$ and $Re = 10000$, respectively. Thus, the lowest reported error at $Re = 100$ does not necessarily identify the method with the greatest robustness over a wider range of flow regimes. Fig. 23(b) and (c) also show that the errors in u, v, and p can differ substantially within the same simulation. Ranking a method solely by the error of the velocity magnitude may therefore conceal weak prediction of an individual velocity component or pressure.

Fig. 24 shows the variable-specific relative L_2 errors reported by Wang et al. [7] and the MSE values reported by Jangir et al. [35] at $Re = 2000$. The two studies also use different reference solutions. No study was found that reported the relative L_2 error of V. This illustrates a limitation of the literature, as differences in reported metrics and reference solutions can prevent a meaningful head-to-head comparison even when the same Re number is considered. Fig. 25 compares results at $Re = 3200$. An additional source of inconsistency at this Re number is the treatment of the moving lid. Wang et al. [62] and Wei et al. [54] used the constant lid velocity, whereas Khademi and Dufour [45] and Wang et al. [63] used the hyperbolic-cosine profile shown in Fig. 8(a). The comparison between Wang et al. [62] and Wei et al. [54] is comparatively well aligned because both report the relative L_2 error of the velocity magnitude V using the same benchmark dataset. Under these conditions, Wei et al. [54] report a better quantitative performance at

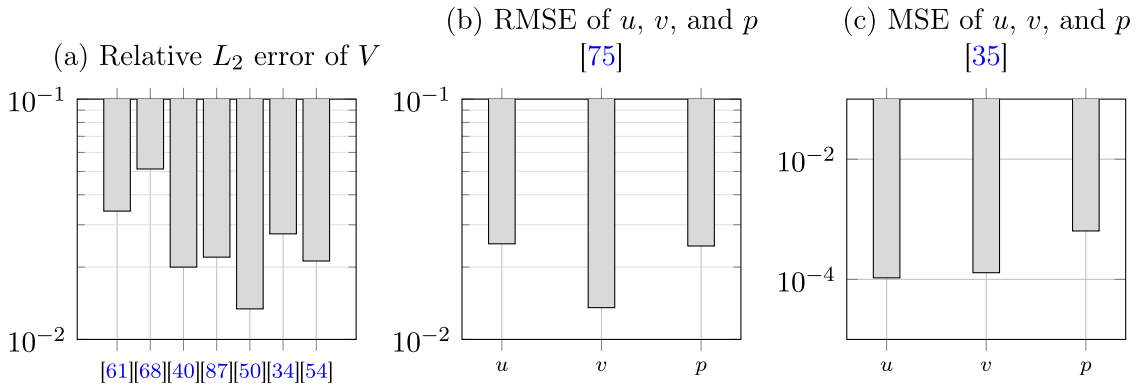


Fig. 23. Reported errors for PINN solutions of LDC at $Re = 100$. Panel (a) shows relative L_2 errors of the velocity magnitude V. Panel (b) shows RMSE values for u, v, and p. Panel (c) shows MSE values of u, v, and p.

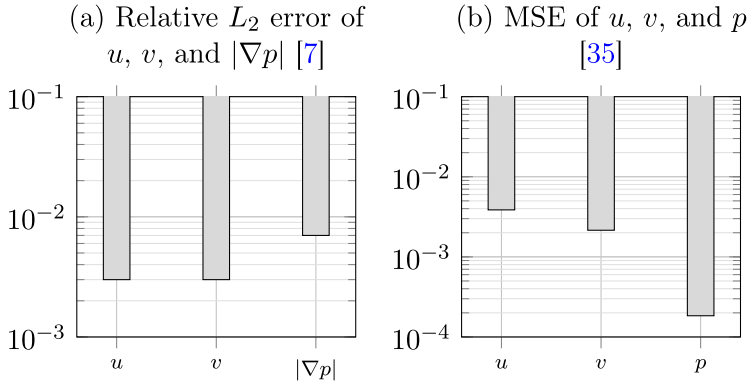


Fig. 24. Reported errors for PINN solutions of LDC at $Re = 2000$. Panel (a) shows relative L_2 error values of u, v, and $|\nabla p|$. Panel (b) shows MSE values of u, v, and p.

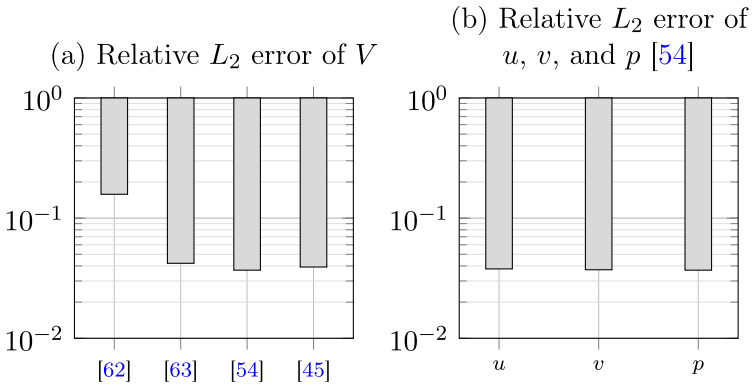


Fig. 25. Reported errors for PINN solutions of LDC at $Re = 3200$. Panel (a) shows relative L_2 error values of the velocity magnitude V. Panel (b) shows relative L_2 error values of u, v, and p.

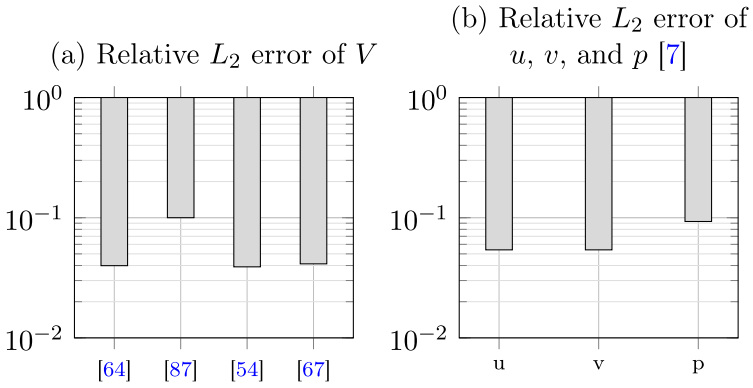


Fig. 26. Reported errors for PINN solutions of LDC at $Re = 5000$. Panel (a) shows relative L_2 error values of the velocity magnitude V. Panel (b) shows relative L_2 error values of u, v, and p.

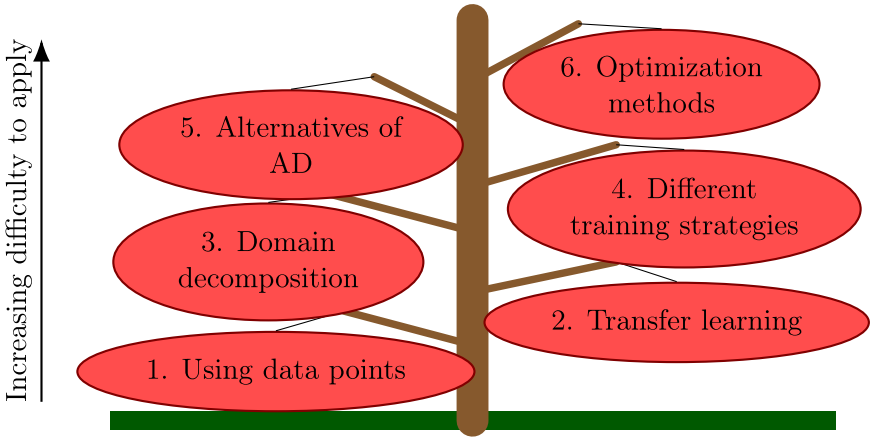


Fig. 27. Tree-style illustration of design choices for improving the accuracy of PINNs for LDC. Lower fruits correspond to methods that are easier to adopt, while higher fruits represent methods that are harder to implement.

Table 6
General lessons for PINNs suggested by studies on LDC.

No.	Observation from LDC studies	General implication for PINNs	Suggested research direction
1	Many PINN developments are motivated by the assumption that failure mainly occurs in high-gradient regions [20,21,26,35]. However, evidence from modeling LDC [31] suggests that the more severe difficulty may arise in regions with large second derivatives (Section 6.1.2).	The main bottleneck in PINN approximation may be related more strongly to higher-order variation than to first-order variation alone.	Methods designed to improve learning in high-gradient regions can be reformulated under a second-derivative hypothesis and compared directly with their original versions.
2	In standard optimization, the comparison between the current epoch and previous epochs is usually handled by the optimizer. Some PINN studies [54,75,87] on LDC indicate that this comparison can instead be incorporated into the loss function (Section 6).	Loss design in PINNs can do more than enforce physics and boundary conditions. It can also encode training-history information and influence optimization dynamics directly.	Future PINN losses can be designed to include history-aware or epoch-comparative terms and then benchmarked against purely optimizer-driven formulations.
3	Reported conclusions about PINN variants may conflict when methods are implemented under different architectures, collocation strategies, training schedules, or boundary treatments. For example, one study [32] may conclude that FD-PINN fails for LDC at $Re=1000$, whereas another [55] may show that FD-PINN succeeds at the same Re (Section 6.1.3).	Fair assessment of PINN methodologies requires a common implementation and evaluation framework. Otherwise, differences in outcome may reflect the framework rather than the method itself.	A clear research need is the systematic comparison of several PINN methods within a common framework, so that differences in performance can be attributed to the methods themselves rather than to inconsistencies in architecture, sampling, optimization, or evaluation.
4	For LDC, PINNs recover flow structures that differ from those typically reported by conventional simulations, while still satisfying the governing equations and boundary conditions [2,7,28] (Section 3.1).	PINNs may reveal multiple admissible solution branches or physically consistent states that conventional CFD workflows do not readily expose.	This possibility should be investigated systematically, especially in complex flows where multiple solution branches or sensitivity to initialization may exist.

$Re=3200$. At $Re=5000$, Fig. 26 shows that Wei et al. [54] report the lowest explicitly identified relative L_2 error of V among the selected studies. However, the comparison is only partially aligned because the original studies use different reference solutions and do not clearly establish identical evaluation points. Wang et al. [64] and Wang et al. [67] also used the hyperbolic-cosine profile shown in Fig. 8(a), whereas Wei et al. [54] retained the constant lid velocity.

Considering the selected results, Wei et al. [54] report competitive accuracy at $Re=100$, the lowest velocity-magnitude errors in the selected comparisons at $Re=3200$ and $Re=5000$, and simulations up to $Re=10000$ without field data while retaining the constant lid velocity. Their strong data-free performance is also evident in Fig. 18, where their method reaches one of the highest Re-number limits reported. The method also combines several design choices identified in this review as favorable. These include the classical constant lid velocity, which preserves consistency with the standard benchmark (Section 3.2), dimensionless governing equations (Section 3.1), finite-volume residual evaluation (Section 6.1.3), and a residual-correction loss that incorporates optimization-history information into the loss formulation (Section 5.1). The frequency-mapping layer (Section 4.2) and the combination of shared and variable-specific hidden layers (Section 4.1) further improve the representation of complex flow features. The use of swish is also consistent with its promising performance in recent high-Re studies (Section 4.4). In addition, Wei et al. [54] evaluated their predictions using the established benchmark of Ghia et al. [109], together with scalar errors, error contours, and streamlines, thereby considering several complementary aspects of solution quality (Section 6.2).

This work [54] is part of a continuing research line by Wei et al. [52–54,60], in which the authors progressively developed their PINN formulation for LDC across wider Re-number ranges. In their most recent contribution, Wei et al. [60] extended the reported range to $Re=20000$. However, this study was available only as a preprint at the time of the review and does not report lower-Re results suitable for direct comparison in Fig. 23–26. Therefore, the approach proposed by Wei et al. [54] appears to be the strongest overall candidate in this selected quantitative assessment. However, this conclusion remains conditional. This subsection does not include every study listed in Table 1, and the available results were not obtained under fully identical evaluation conditions.

7. Concluding remarks

The insights and open research questions identified in this review are not repeated in detail here because each section already closes with dedicated subsections on the main findings from the literature and the remaining gaps. Instead, this concluding section summarizes the principal overall assessment, offers several broader recommendations for future PINN studies on LDC and draws attention to what studies of LDC

reveal for fluid modeling with PINNs in a wider sense. These broader lessons are presented in Table 6.

No universally best PINN method for LDC can currently be identified because the available studies differ in problem formulation, boundary treatment, use of training data, and evaluation protocols. Nevertheless, as discussed in Section 6.3, the approach of Wei et al. [54] appears to be the strongest overall candidate among the studies included in the selected quantitative assessment. This conclusion remains conditional and may change as new methods and more directly comparable results become available. This uncertainty underscores the value of benchmark problems such as LDC for assessing new PINN developments, provided that the methods are formulated and reported under comparable conditions. Without such comparability, methodological progress is difficult to evaluate because results obtained under different conditions cannot be contrasted directly. For LDC, several aspects of the problem definition should therefore be standardized to facilitate comparison. Based on the practice followed by most studies, the most suitable reference configuration is a square cavity in the steady two-dimensional regime. Likewise, smoothing the moving-wall velocity instead of imposing a constant lid speed may improve optimization, but it reduces comparability with the classical benchmark adopted in much of the literature. For example, Hao et al. [25] provided valuable ablation results. However, because they employed a parabolic velocity profile at the moving wall, their results are not directly comparable with those of most published LDC studies.

The methods reviewed in this study can be interpreted as a trade-off between ease of implementation and the resources required for their use. As illustrated in Fig. 27, these improvement strategies are arranged according to their relative difficulty of application, much like the branches of a fruit tree. Methods placed near the bottom of the tree are generally easier to adopt, while those placed higher require more substantial modifications to the PINN framework, training procedure, or computational workflow. The simplest strategy is the use of additional data points, which can improve accuracy with minimal changes to the model. However, this approach is not always feasible, since it depends on the availability of reliable high-fidelity reference data. Transfer learning can also be placed in the lower part of the tree. It does not usually require major changes in the design of the PINN itself, yet it still demands additional resources because some preliminary PINN simulations or prior training effort are needed. Moving higher up the tree leads to domain decomposition and different training strategies. These approaches are more difficult to apply and introduce additional design choices, while also increasing the computational effort required during training. Even so, they usually do not alter the core structure of the PINN. At the highest level of the tree are alternatives to automatic differentiation and improvements in optimization methods. These directions require deeper methodological intervention and greater implementation effort. Only a limited number of studies have reached this upper part of the tree, which suggests that some ripe fruits still remain untouched.

LDC should be viewed as more than a standard benchmark. It is a compact and informative testbed for understanding where PINNs succeed, where they struggle, and which methodological modifications are most worth pursuing. Future progress will depend not only on proposing new PINN variants, but also on evaluating them under sufficiently consistent conditions so that genuine advances can be distinguished from improvements caused mainly by differences in implementation details or benchmark definition.

CRediT authorship contribution statement

Mohammad Sheikholeslami: Writing – original draft, Visualization, Methodology, Investigation, Formal analysis, Conceptualization, Data curation; **Saeed Salehi:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization, Data curation; **Wengang Mao:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization, Data curation; **Arash Eslamdoost:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization, Data curation; **Håkan Nilsson:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization, Data curation.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work, the authors used ChatGPT (OpenAI) to assist with language editing and the refinement of LaTeX code. The authors reviewed, verified, and edited all outputs as needed and take full responsibility for the content of the published article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The work was financed by the Department of Mechanical Engineering of Chalmers University of Technology.

Data availability

No data was used for the research described in the article.

References

[1] N. Roy, R. Dürr, A. Bück, S. Sundar, Finite difference physics-informed neural networks enable improved solution accuracy of the Navier-Stokes equations, 2024. arXiv Preprint:2501.00014 [physics.comp-ph]
[2] Q. Hu, I. Senocak, Unsupervised simulation of incompressible flows with physics- and equality- constrained artificial neural networks, 2026. arXiv Preprint:2511.18820 [physics.flu-dyn]
[3] C.K. Aidun, N.G. Triantafillopoulos, J.D. Benson, Global stability of a lid–driven cavity with throughflow: flow visualization studies, Phys. Fluids A Fluid Dyn. 3 (9) (1991) 2081–2091. https://doi.org/10.1063/1.857891
[4] A.M. Grillet, B. Yang, B. Khomami, E.S.G. Shaqfeh, Modeling of viscoelastic lid driven cavity flow using finite element simulations, J. Non-Newton. Fluid Mech. 88 (1) (1999) 99–131. https://doi.org/10.1016/S0377-0257(99)00015-4
[5] F. Auteri, N. Parolini, L. Quartapelle, Numerical investigation on the stability of singular driven cavity flow, J. Comput. Phys. 183 (1) (2002) 1–25. https://doi.org/10.1006/jcph.2002.7145
[6] V. THEOFILIS, P.W. DUCK, J. OWEN, Viscous linear stability analysis of rectangular duct and cavity flows, J. Fluid Mech. 505 (2004) 249–286. https://doi.org/10.1017/S002211200400850X
[7] Z. Wang, X. Meng, X. Jiang, H. Xiang, G.E. Karniadakis, Solution multiplicity and effects of data and eddy viscosity on Navier-Stokes solutions inferred by physics-informed neural networks, 2023. arXiv Preprint:2309.06010 [physics.flu-dyn]
[8] H.C. KUHLMANN, M. WANSCHURA, H.J. RATH, Flow in two-sided lid-driven cavities: non-uniqueness, instabilities, and cellular structures, J. Fluid Mech. 336 (1997) 267–299. https://doi.org/10.1017/S0022112096004727
[9] M.A. Ferrari, A.T. Franco, Exploring the periodic behavior of the lid-driven cavity flow filled with a Bingham fluid, J. Non-Newton. Fluid Mech. 316 (2023) 105030. https://doi.org/10.1016/j.jnnfm.2023.105030
[10] P.N. Shankar, M.D. Deshpande, Fluid mechanics in the driven cavity, Annu. Rev. Fluid Mech. 32 (2000) 93–136. Review Article. https://doi.org/10.1146/annurev.fluid.32.1.93
[11] H. Kuhlmann, F. Romanò, The lid-driven cavity, in: A. Gelfgat (Ed.), Computational Modelling of Bifurcations and Instabilities in Fluid Dynamics, Springer, 2019, pp. 233–309. https://doi.org/10.1007/978-3-319-91494-7
[12] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707. https://doi.org/10.1016/j.jcp.2018.10.045
[13] J. Lin, S.-s. Chen, H. Yang, Q.-f. Jiang, J. Liu, A physics-informed neural network-based aerodynamic parameter identification method for aircraft, Phys. Fluids 37 (2) (2025) 027200. https://doi.org/10.1063/5.0249130
[14] X.-d. Bai, Y. Wang, W. Zhang, Applying physics informed neural network for flow data assimilation, J. Hydrodyn. 32 (6) (2020) 1050–1058. https://doi.org/10.1007/s42241-020-0077-2

[15] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000. https://doi.org/10.1109/72.712178

[16] J.D. Toscano, V. Oommen, A.J. Varghese, Z. Zou, N.A. Daryakenari, C. Wu, G.E. Karniadakis, From PINNs to PIKANs: recent advances in physics-informed machine learning, Mach. Learn. Comput. Sci. Eng. 1 (2025) 15. https://doi.org/10.1007/s44379-025-00015-1

[17] T. Służalec, D. Wójcik, C. Uriarte, M. Łoś, A. Paszyńska, M. Paszyński, Reliable physics-informed neural networks for Navier–Stokes simulations. Can we trust AI-generated numerical simulations?, J. Comput. Sci. 95 (2026) 102817. https://doi.org/10.1016/j.jocs.2026.102817

[18] C. McDevitt, E. Fowler, S. Roy, Physics-constrained deep learning of incompressible cavity flows, https://doi.org/10.2514/6.2024-1692

[19] R. Pal, S. Mukherjee, U. Dutta, A. Choudhury, Solving Navier-Stokes equations using data-free physics-informed neural networks with hard boundary conditions, 2025. arXiv Preprint:2511.14497 [physics.flu-dyn]

[20] W. Chen, A. Howard, P. Stinis, Self-adaptive weighting and sampling for physics-informed neural networks, Mach. Learn. Sci. Technol. 7 (2) (2026) 025050. https://doi.org/10.1088/2632-2153/ae556e

[21] B. Lin, Z. Mao, Z. Wang, Operator learning augmented physics-informed neural networks for partial differential equations exhibiting sharp features, Phys. Rev. E 113 (2026) 035306. https://doi.org/10.1103/xyjy-gx6f

[22] H. Sababha, A. Elmaradny, H. Taha, M. Daqaq, A variational computational-based framework for unsteady incompressible flows, Comput. Methods Appl. Mech. Eng. 444 (2025) 118091. https://doi.org/10.1016/j.cma.2025.118091

[23] W. Li, H. Wang, H. Guan, R. Zhou, C. Zhang, D. Tao, AdaPW: an adaptive point-weighting method for training physics-informed neural networks, Comput. Math. Appl. 198 (2025) 255–273. https://doi.org/10.1016/j.camwa.2025.08.026

[24] Q. Lou, X. Meng, G.E. Karniadakis, Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-BGK formulation, J. Comput. Phys. 447 (2021) 110676. https://doi.org/10.1016/j.jcp.2021.110676

[25] Z. Hao, J. Yao, C. Su, H. Su, Z. Wang, F. Lu, Z. Xia, Y. Zhang, S. Liu, L. Lu, J. Zhu, PINNacle: a comprehensive benchmark of physics-informed neural networks for solving PDEs, in: Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS '24, Curran Associates Inc., Red Hook, NY, USA, 2024. https://doi.org/10.52202/079017-2442

[26] S. Ding, J. Lei, L. Xu, B. Zhang, G. Sun, J. Guo, Y. Zhang, Adaptive multi-scale gradient-enhanced sampling physics-informed neural network for incompressible flow, Phys. Fluids 38 (2) (2026) 027106. https://doi.org/10.1063/5.0311686

[27] J.F. Urbán, P. Stefanou, J.A. Pons, Unveiling the optimization process of physics informed neural networks: how accurate and competitive can PINNs be?, J. Comput. Phys. 523 (2025) 113656. https://doi.org/10.1016/j.jcp.2024.113656

[28] K. Shukla, J.D. Toscano, Z. Wang, Z. Zou, G.E. Karniadakis, A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks, Comput. Methods Appl. Mech. Eng. 431 (2024) 117290. https://doi.org/10.1016/j.cma.2024.117290

[29] Z. Geng, H. Liu, R. Deng, X. Yu, D. Jiang, S. Cai, Flow field reconstruction from sparse and noisy data based on finite volume physics-informed neural networks, Phys. Fluids 38 (3) (2026) 037120. https://doi.org/10.1063/5.0307754

[30] B. Sun, S. Cai, Q. Du, Z. Wang, Y. Chen, Z. Tian, J. Zhu, Reconstruction of fields based on physics-informed neural networks with sensor placement optimization, Acta Mech. Sin. 42 (7) (2026) 725195. https://doi.org/10.1007/s10409-025-25195-x

[31] A. Satyadharma, M.-J. Chern, H.-C. Kan, Harinaldi, J. Julian, Assessing physics-informed neural network performance with sparse noisy velocity data, Phys. Fluids 36 (10) (2024) 103619. https://doi.org/10.1063/5.0213522

[32] J. Feng, Y. Xiao, R. Imin, A. Rozi, Kernel derivative free-based physics-informed neural network for steady incompressible flows, Eng. Anal. Bound. Elem. 181 (2025) 106523. https://doi.org/10.1016/j.enganabound.2025.106523

[33] H. Liu, Z. Wang, R. Deng, S. Wang, X. Meng, C. Xu, S. Cai, Flow reconstruction with uncertainty quantification from noisy measurements based on Bayesian physics-informed neural networks, Phys. Fluids 36 (11) (2024) 117104. https://doi.org/10.1063/5.0231684

[34] H. Pandey, A. Singh, R. Behera, An efficient wavelet-based physics-informed neural network for multiscale problems, Neural Netw. (2026) 108860. https://doi.org/10.1016/j.neunet.2026.108860

[35] A. Jangir, R. Clements, R. Goyal, G. Tabor, Sparse-supervised hybrid parameterized physics-informed neural networks for incompressible flows across reynolds numbers, 2026. arXiv Preprint:2602.04670 [physics.flu-dyn]

[36] J. Ba, S. Fan, J. Liu, Y. Ji, H. Wang, A hard-constrained physics-informed neural network framework for solving steady incompressible flows, Ocean Eng. 357 (2026) 125341. https://doi.org/10.1016/j.oceaneng.2026.125341

[37] Z. Zou, X. Meng, G.E. Karniadakis, Correcting model misspecification in physics-informed neural networks (PINNs), J. Comput. Phys. 505 (2024) 112918. https://doi.org/10.1016/j.jcp.2024.112918

[38] A.D. Jagtap, E. Kharazmi, G.E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems, Comput. Methods Appl. Mech. Eng. 365 (2020) 113028. https://doi.org/10.1016/j.cma.2020.113028

[39] K. Shukla, A.D. Jagtap, G.E. Karniadakis, Parallel physics-informed neural networks via domain decomposition, J. Comput. Phys. 447 (2021) 110683. https://doi.org/10.1016/j.jcp.2021.110683

[40] L. Gu, S. Qin, L. Xu, R. Chen, Physics-informed neural networks with domain decomposition for the incompressible Navier–Stokes equations, Phys. Fluids 36 (2) (2024) 021914. https://doi.org/10.1063/5.0188830

[41] A. Khademi, S. Dufour, A novel discretized physics-informed neural network model applied to the Navier–Stokes equations, Phys. Scr. 99 (7) (2024) 076016. https://doi.org/10.1088/1402-4896/ad5592

[42] Y. Qian, J. Liu, Z. Xia, S. Chen, C. Xu, S. Cai, Distributed physics-informed neural networks via domain decomposition for fast flow reconstruction, 2026. arXiv Preprint:2602.15883 [cs.LG]

[43] X. Pan, D. Xiao, Domain decomposition for physics-data combined neural network based parametric reduced order modelling, J. Comput. Phys. 519 (2024) 113452. https://doi.org/10.1016/j.jcp.2024.113452

[44] X. Xiong, K. Lu, Z. Zhang, Z. Zeng, S. Zhou, Z. Deng, R. Hu, J-PIKAN: a physics-informed KAN network based on Jacobi orthogonal polynomials for solving fluid dynamics, Commun. Nonlinear Sci. Numer. Simul. (2025) 109414. https://doi.org/10.1016/j.cnsns.2025.109414

[45] A. Khademi, S. Dufour, Physics-informed neural networks with trainable sinusoidal activation functions for approximating the solutions of the Navier-Stokes equations, Comput. Phys. Commun. 314 (2025) 109672. https://doi.org/10.1016/j.cpc.2025.109672

[46] W. Zhou, X. Feng, L. Guo, H. Wu, Latent representation learning based model correction and uncertainty quantification for PDEs, 2026. arXiv Preprint:2603.24948 [math.NA]

[47] Z. Zhang, X. Liu, S. Chen, LSMPS-PINN and LSFVP-PINN: complementary meshless operator learning strategies for solving partial differential equations, Eng. Anal. Bound. Elem. 187 (2026) 106747. https://doi.org/10.1016/j.enganabound.2026.106747

[48] P.-H. Chiu, J.C. Wong, C.C. Ooi, C. Wei, Y. Fan, Y.-S. Ong, Scale-PINN: learning efficient physics-informed neural networks through sequential correction, 2026. arXiv Preprint:2602.19475 [cs.CE]

[49] H. Gao, M.J. Zahr, J.-X. Wang, Physics-informed graph neural Galerkin networks: a unified framework for solving PDE-governed forward and inverse problems, Comput. Methods Appl. Mech. Eng. 390 (2022) 114502. https://doi.org/10.1016/j.cma.2021.114502

[50] X. Chen, R. Chen, Q. Wan, R. Xu, J. Liu, An improved data-free surrogate model for solving partial differential equations using deep neural networks, Sci. Rep. 11 (1) (2021) 19507. https://doi.org/10.1038/s41598-021-99037-x

[51] T. Yang, Z. Qian, N. Hang, M. Liu, S-PINN: Stabilized physics-informed neural networks for alleviating barriers between multi-level co-optimization, Comput. Methods Appl. Mech. Eng. 447 (2025) 118348. https://doi.org/10.1016/j.cma.2025.118348

[52] C. Wei, Y. Zhou, Y. Fan, X. Liu, C. Li, X. Li, H. Wang, Physics-informed neural network based on the finite volume method for solving forward and inverse problems, Phys. Fluids 37 (8) (2025) 087190. https://doi.org/10.1063/5.0284425

[53] C. Wei, Y. Fan, Y. Zhou, X. Liu, C. Li, X. Li, H. Wang, Physics-informed neural network based on control volumes for solving time-independent problems, Phys. Fluids 37 (3) (2025) 037128. https://doi.org/10.1063/5.0256470

[54] C. Wei, Y. Fan, J.C. Wong, C.C. Ooi, H. Wang, P.-H. Chiu, FFV-PINN: a fast physics-informed neural network with simplified finite volume discretization and residual correction, Comput. Methods Appl. Mech. Eng. 444 (2025) 118139. https://doi.org/10.1016/j.cma.2025.118139

[55] Q. Jiang, C. Shu, L. Zhu, L. Yang, Y. Liu, Z. Zhang, Applications of finite difference-based physics-informed neural networks to steady incompressible isothermal and thermal flows, Int. J. Numer. Methods Fluids 95 (10) (2023) 1565–1597. https://doi.org/10.1002/fld.5217

[56] H.V. Farahani, A.M. Tahsini, A. Nematollahi, M.G. Afzal, A novel physics-informed neural network architecture for solving nonlinear PDEs in fluid mechanics without labeled data, Eng. Comput. 42 (2026). https://doi.org/10.1007/s00366-026-02277-6

[57] T. Służalec, M. Łoś, A. Vilkha, M. Paszyński, Python library supporting discrete variational formulations and training solutions with collocation-based robust variational physics informed neural networks (DVF-CRVPINN), 2026. arXiv Preprint:2604.15398 [cs.LG]

[58] M. Abda, L. Berthet, M. Hamedi, D. Hibatullah, E. Piollet, C. Blake, B. Blais, F.P. Gosselin, The finite element neural network method to simulate two dimensional partial differential equations and perform parameter identification, Sci. Rep. (2026). https://doi.org/10.1038/s41598-026-46707-3

[59] R. Ranade, C. Hill, J. Pathak, DiscretizationNet: a machine-learning based solver for Navier–Stokes equations using finite volume discretization, Comput. Methods Appl. Mech. Eng. 378 (2021) 113722. https://doi.org/10.1016/j.cma.2021.113722

[60] C. Wei, Y. Fan, C.C. Ooi, J.C. Wong, H. Wang, P.-H. Chiu, Bridging Computational Fluid Dynamics algorithm and physics-informed learning: simple-pinn for incompressible Navier-Stokes equations, 2026. arXiv Preprint:2603.24013 [cs.CE]

[61] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (5) (2021) A3055–A3081. https://doi.org/10.1137/20M1318043

[62] S. Wang, S. Sankaran, H. Wang, P. Perdikaris, An Expert's Guide to Training Physics-informed Neural Networks, 2023. arXiv Preprint:2308.08468 [cs.LG]

[63] S. Wang, B. Li, Y. Chen, P. Perdikaris, PirateNets: physics-informed deep learning with residual adaptive networks, J. Mach. Learn. Res. 25 (402) (2024) 1–51. https://www.jmlr.org/papers/v25/24-0313.html.

[64] S. Wang, A. bhartari, B. Li, P. Perdikaris, Gradient alignment in physics-informed neural networks: a second-order optimization perspective, in: D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, N. Chen (Eds.), Advances in Neural Information Processing Systems, vol. 38, Curran Associates, Inc., 2025, pp. 168482–168532. https://proceedings.neurips.cc/paper_files/paper/2025/file/f655706547885b8e32ef46f1c067ece2-Paper-Conference.pdf.

[65] M. Cohen Tillberg, I. Ingerskog, Modelling flow in hydrocyclones using physics-informed neural networks, Msc thesis, Department of Automatic Control, Lund University, Lund, Sweden, 2025. https://lup.lub.lu.se/student-papers/record/9186762.
[66] S. Li, X. Feng, Dynamic weight strategy of physics-informed neural networks for the 2D Navier–Stokes equations, Entropy 24 (9) (2022). https://doi.org/10.3390/e24091254
[67] S. Wang, S. Koohy, Y. Lu, P. Perdikaris, When PINNs go wrong: pseudo-time stepping against spurious solutions, 2026. arXiv Preprint:2604.23528 [cs.LG]
[68] P. Niu, J. Guo, Y. Chen, Y. Zhou, M. Feng, Y. Shi, Improved physics-informed neural network in mitigating gradient-related failures, Neurocomputing 638 (2025) 130167. https://doi.org/10.1016/j.neucom.2025.130167
[69] J.C. Wong, C.C. Ooi, A. Gupta, Y.-S. Ong, Learning in sinusoidal spaces with physics-informed neural networks, IEEE Trans. Artif. Intell. 5 (3) (2024) 985–1000. https://doi.org/10.1109/tai.2022.3192362
[70] P.-H. Chiu, J.C. Wong, C. Ooi, M.H. Dao, Y.-S. Ong, CAN-PINN: a fast physics-informed neural network based on coupled-automatic–numerical differentiation method, Comput. Methods Appl. Mech. Eng. 395 (2022) 114909. https://doi.org/10.1016/j.cma.2022.114909
[71] S.K. Biswas, N.K. Anand, Three-dimensional laminar flow using physics informed deep neural networks, Phys. Fluids 35 (12) (2023) 121703. https://doi.org/10.1063/5.0180834
[72] Y.-H. Tsai, H.-T. Juan, P.-H. Chiu, C.-A. Lin, MLD-PINN: a multi-level datasets training method in physics-informed neural networks, Comput. Fluids 303 (2025) 106849. https://doi.org/10.1016/j.compfluid.2025.106849
[73] Z. Zhang, S. Zhang, Y. Zhao, W. Wang, H. Wu, X. Yang, C. Yang, DFS-PINN: a dynamic feature separation physics-informed neural network, Comput.-Aided Des. 191 (2026) 103992. https://doi.org/10.1016/j.cad.2025.103992
[74] M. Khadijeh, V. Cerqueglini, C. Kasbergen, S. Erkens, A. Varveri, Multistage physics informed neural network for solving coupled multiphysics problems in material degradation and fluid dynamics, Eng. Comput. (2025). https://doi.org/10.1007/s00366-025-02174-4
[75] A. Farea, S. Khan, M.S. Celebi, Multi-objective loss balancing in physics-informed neural networks for fluid flow applications, in: 2025 IEEE 32nd International Conference on High Performance Computing, Data, and Analytics (HiPC), IEEE, Hyderabad, India, 2025, pp. 108–118. https://doi.org/10.1109/HiPC66333.2025.00020
[76] Q. Tan, X. Wu, Y. Qin, et al., NS-PINN for solving the incompressible navier-Stokes equations, Res. Sq. (2026). Preprint, Version 1. https://doi.org/10.21203/rs.3.rs-9160993/v1
[77] V. Kumar, L. Gleyzer, A. Kahana, K. Shukla, G.E. Karniadakis, MyCrunchGPT: a chatGPT assisted framework for scientific machine learning, 2023. arXiv Preprint:2306.15551 [cs.LG]
[78] R. Takao, S. Ii, Ii, Fine-tuning physics-informed neural networks for cavity flows using coordinate transformation, Comput. Fluids 306 (2026) 106957. https://doi.org/10.1016/j.compfluid.2025.106957
[79] E. Kiyani, K. Shukla, J.F. Urbán, J. Darbon, G.E. Karniadakis, Optimizing the optimizer for physics-informed neural networks and Kolmogorov-Arnold networks, Comput. Methods Appl. Mech. Eng. 446 (2025) 118308. https://doi.org/10.1016/j.cma.2025.118308
[80] A. Jnini, E. Kiyani, K. Shukla, J.F. Urban, N.A. Daryakenari, J. Muller, M. Zeinhofer, G.E. Karniadakis, Curvature-aware optimization for high-accuracy physics-informed neural networks, 2026. arXiv Preprint:2604.05230 [cs.LG]
[81] Z. Zou, Z. Wang, G. Em Karniadakis, Learning and discovering multiple solutions using physics-informed neural networks with random initialization and deep ensemble, Proc. R. Soc. A Math. Phys. Eng. Sci. 481 (2325) (2025) 20250205. https://doi.org/10.1098/rspa.2025.0205
[82] Y. Zheng, F. Peng, Z. Wang, J. Lei, S. Pian, A vorticity-enhanced physics-informed neural network with logarithmic reynolds embedding, Fluids 11 (4) (2026). https://doi.org/10.3390/fluids11040093
[83] Y. He, Z. Wang, H. Xiang, X. Jiang, D. Tang, An artificial viscosity augmented physics-informed neural network for incompressible flow, Appl. Math. Mech. 44 (7) (2023) 1101–1110. https://doi.org/10.1007/s10483-023-2993-9
[84] M. Feng, M. Shan, L. Kuai, C. Yang, Y. Yang, C. Yin, Q. Han, Hybrid physics-informed neural networks integrating multi-relaxation-time lattice boltzmann method for forward and inverse flow problems, (2025). https://doi.org/10.20944/preprints202510.1309.v1
[85] P. Kumar, R. Ranjan, Evaluation of physics-informed machine learning approach for computation of fluid flows, in: H. Kothadia, K.R. Arun, G. Rajesh, J.H. Arakeri (Eds.), Proceedings of Fluid Mechanics and Fluid Power (FMFP) 2023, Vol. 2, Springer Nature Singapore, Singapore, 2025, pp. 429–440.
[86] A.G. Srinivasan, V. Dwivedi, B. Srinivasan, Deep vs. Shallow: benchmarking physics-informed neural architectures on the biharmonic equation, 2025. arXiv Preprint:2510.04490 [cs.CE]
[87] W. Cao, W. Zhang, TSONN: Time-stepping-oriented neural network for solving partial differential equations, 2023. arXiv Preprint:2310.16491 [cs.LG]
[88] P. Karnakov, S. Litvinov, P. Koumoutsakos, Solving inverse problems in physics by optimizing a discrete loss: fast and accurate learning without neural networks, PNAS Nexus 3 (1) (2024) pgae005. https://doi.org/10.1093/pnasnexus/pgae005
[89] W. Cao, W. Zhang, Overcoming the loss conditioning bottleneck in optimization-based PDE solvers: a well-conditioned loss function, Commun. Nonlinear Sci. Numer. Simul. (2026) 109952. https://doi.org/10.1016/j.cnsns.2026.109952
[90] B. Barman, D. Chatterjee, R.K. Ray, A simple but efficient transformer-based physics-informed neural network for incompressible Navier–Stokes equations, 2026. arXiv Preprint:2601.03613 [physics.flu-dyn]
[91] S. Cuomo, V. Schiano Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics-Informed neural networks: where we are and what's next, J. Sci. Comput. 92 (3) (2022) 1–62. Published online: 26 July 2022. https://doi.org/10.1007/s10915-022-01939-z
[92] Ahmad, H. Zafar, A. Zafar, M.N. Sadiq, A.K. Awasthi, H. Emadifar, K.K. Ahmed, Evolution of physics-informed neural networks: recent architectural variants and optimization strategies, Array 29 (2026) 100688. https://doi.org/10.1016/j.array.2026.100688
[93] X. Jin, S. Cai, H. Li, G.E. Karniadakis, NSFnets (Navier-Stokes flow nets): physics-informed neural networks for the incompressible Navier-Stokes equations, J. Comput. Phys. 426 (2021) 109951. https://doi.org/10.1016/j.jcp.2020.109951
[94] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, A. Courville, On the spectral bias of neural networks, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, vol. 97 of *Proceedings of Machine Learning Research*, PMLR, 2019, pp. 5301–5310. https://proceedings.mlr.press/v97/rahaman19a.html.
[95] J. Abbasi, A.D. Jagtap, B. Moseley, A. Hiorth, P.øo. Andersen, Challenges and advancements in modeling shock fronts with physics-informed neural networks: a review and benchmarking study, Neurocomputing 657 (2025) 131440. https://doi.org/10.1016/j.neucom.2025.131440
[96] A.D. Jagtap, G.E. Karniadakis, Extended physics-informed neural networks (xpinns): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. 28 (5) (2020) 2002–2041. https://doi.org/10.4208/cicp.OA-2020-0164
[97] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41. https://doi.org/10.1016/j.neucom.2018.06.056
[98] N. Sukumar, A. Srivastava, Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks, Comput. Methods Appl. Mech. Eng. 389 (2022) 114333. https://doi.org/10.1016/j.cma.2021.114333
[99] K. Du, Z. Huang, J. Li, D. Tao, Z. Chen, A label-free physics informed neural network with hard constraints and Fourier features spectrally-enhanced for multi-frequency seismic structural dynamic response, Eng. Appl. Artif. Intell. 166 (2026) 113640. https://doi.org/10.1016/j.engappai.2025.113640
[100] A.D. Jagtap, K. Kawaguchi, G.E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404 (2020) 109136. https://doi.org/10.1016/j.jcp.2019.109136
[101] J.E. Chrisnanto, S.R. Alia, N. Fadillah, Y.H. Chrisnanto, High-fidelity modeling of stochastic chemical dynamics on complex manifolds: a multiscale SIREN-PINN framework for the curvature-perturbed Ginzburg–Landau equation, J. Nonlinear Sci. 36 (4) (2026) 80. https://doi.org/10.1007/s00332-026-10300-9
[102] M. Kurukulasuriya, C. Batuwatta-Gamage, C. Karunasena, H. Jeong, K. Ranathunga, C. Rathnayaka, Y. Gu, A novel sigmexd activation function to enhance physics-informed neural networks (PINNs) for differential equations with a perspective on the role of activation functions in PINNs, Neurocomputing 666 (2026) 132364. https://doi.org/10.1016/j.neucom.2025.132364
[103] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Y.W. Teh, M. Titterington (Eds.), Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, vol. 9 of *Proceedings of Machine Learning Research*, PMLR, Chia Laguna Resort, Sardinia, Italy, 2010, pp. 249–256. https://proceedings.mlr.press/v9/glorot10a.html.
[104] K. Leng, J. Thiyagalingam, On the compatibility between neural networks and partial differential equations for physics-informed learning, 2023. arXiv Preprint:2212.00270 [physics.comp-ph]
[105] P. Rathore, W. Lei, Z. Frangella, L. Lu, M. Udell, Challenges in training PINNs: a loss landscape perspective, in: R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, F. Berkenkamp (Eds.), Proceedings of the 41st International Conference on Machine Learning, vol. 235 of *Proceedings of Machine Learning Research*, PMLR, Vienna, Austria, 2024, pp. 42159–42191. https://proceedings.mlr.press/v235/rathore24a.html.
[106] J. Yao, C. Su, Z. Hao, S. Liu, H. Su, J. Zhu, MultiAdam: parameter-wise scale-invariant optimizer for multiscale training of physics-informed neural networks, in: A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, J. Scarlett (Eds.), Proceedings of the 40th International Conference on Machine Learning, 202 of *Proceedings of Machine Learning Research*, 2023, pp. 39702–39721. https://proceedings.mlr.press/v202/yao23c.html.
[107] N. Sukumar, R. Roy, A wachspress-based transfinite formulation for exactly enforcing dirichlet boundary conditions on convex polygonal domains in physics-informed neural networks, Comput. Mech. (2026). https://doi.org/10.1007/s00466-026-02789-4
[108] J. Yu, L. Lu, X. Meng, G.E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Comput. Methods Appl. Mech. Eng. 393 (2022) 114823. https://doi.org/10.1016/j.cma.2022.114823
[109] U. Ghia, K.N. Ghia, C.T. Shin, High-Re solutions for incompressible flow using the navier-Stokes equations and a multigrid method, J. Comput. Phys. 48 (3) (1982) 387–411. https://doi.org/10.1016/0021-9991(82)90058-4
[110] O. Botella, R. Peyret, Benchmark spectral results on the lid-driven cavity flow, Comput. Fluids 27 (4) (1998) 421–433. https://doi.org/10.1016/S0045-7930(98)00002-4
[111] M.M. Gupta, J.C. Kalita, A new paradigm for solving Navier–Stokes equations: streamfunction–velocity formulation, J. Comput. Phys. 207 (1) (2005) 52–68. https://doi.org/10.1016/j.jcp.2005.01.002
[112] D. Li, F. Li, B. Xu, Non-orthogonal multiple-relaxation-time lattice Boltzmann method for vorticity-streamfunction formulation, Comput. Math. Appl. 152 (2023) 308–316. https://doi.org/10.1016/j.camwa.2023.10.025
[113] Z.F. Tian, P.X. Yu, An efficient compact difference scheme for solving the stream-function formulation of the incompressible Navier–Stokes equations, J. Comput. Phys. 230 (17) (2011) 6404–6419. https://doi.org/10.1016/j.jcp.2010.12.031

[114] C.-H. Bruneau, C. Jouron, An efficient scheme for solving steady incompressible Navier-Stokes equations, J. Comput. Phys. 89 (2) (1990) 389–413. https://doi.org/10.1016/0021-9991(90)90149-U

[115] E. Erturk, T.C. Corke, C. Gökçöl, Numerical solutions of 2-D steady incompressible driven cavity flow at high Reynolds numbers, Int. J. Numer. Methods Fluids 48 (7) (2005) 747–774. https://doi.org/10.1002/fld.953

[116] A.B. Cortes, J.D. Miller, Numerical experiments with the lid driven cavity flow problem, Comput. Fluids 23 (8) (1994) 1005–1027. https://doi.org/10.1016/0045-7930(94)90002-7