Nonlinear Dyn (2024) 112:16113–16134
https://doi.org/10.1007/s11071-024-09856-6

RESEARCH

An improved physics-informed neural network with adaptive weighting and mixed differentiation for solving the incompressible Navier–Stokes equations

Jie Wang · Xufeng Xiao · Xinlong Feng · Hui Xu

Received: 5 February 2024 / Accepted: 6 June 2024 / Published online: 19 June 2024

Abstract Physics-informed neural networks (PINNs) have been an effective tool for approximating the mapping between points in the spatio-temporal domain and solutions of partial differential equations (PDEs). However, there are still some challenges in dealing with the nonlinear characteristics and complexity of the Navier–Stokes (N–S) equations. In this paper, the improved adaptive weighting PINNs based on the Gaussian likelihood estimation are applied to solve the N–S equations. The weights of the different loss items are allocated adaptively by the maximum likelihood estimation. The improved network structure has been designed with considering both the global and local information, making it easier to capture the part of PDEs solution with drastic changes. A combinational method of the numerical differentiation (ND) and the automatic differentiation (AD) is proposed to compute the differential operators, with the improved computational efficiency. The derivative operation of the convection and pressure-gradient terms was carried out using the combined method in solving the incompressible N–S equations. The results show that the effectiveness and training efficiency of this method are better than PINNs.

J. Wang · X. Xiao (✉) · X. Feng · H. Xu
College of Mathematics and System Science, Xinjiang University, Urumqi 830046, People's Republic of China
e-mail: xiaoxufeng111@sina.com

J. Wang
e-mail: wangjie06152023@163.com

X. Feng
e-mail: fxlmath@xju.edu.cn

H. Xu
e-mail: dr.hxu@sjtu.edu.cn

H. Xu
School of Aeronautics and Astronautics, Shanghai Jiao Tong University, Shanghai 200240, People's Republic of China

Keywords Physics-informed neural networks · Adaptive weighting · Expressive capacity of neural networks · Mixed differentiation · Navier–Stokes equations

1 Introduction

The incompressible Navier–Stokes (N–S) equations are fundamental for fluid flow. They are widely applied in fluid simulation, such as the applications in meteorology, aviation, and engineering. The classical numerical methods, e.g., the finite element [1], the finite volume [2], and the finite difference [3] methods are widely used in solving the N–S equations. These discretization-based methods utilize the values of grid points distributed in the space and time domains to approximate the solution of the partial differential equations (PDEs). However, the classical numerical methods present some challenges, e.g., the curse of dimensionality, the reliance on grid structures, and the numerical instabilities. Compared to traditional numerical methods, the machine learning demonstrates remarkable efficiency and accuracy in solving the incompressible Navier–Stokes equations, particularly when dealing with complex geometries, high-dimensional problems, and inverse problems. And the method without discretization can successfully address the challenges of the traditional methods in dealing with the complex problems [4–9].

The neural networks have been applied to solve PDEs, which is an important application of the machine learning. The neural networks in solving PDEs were proposed at in 1990s [10,11]. In recent years, various types of neural networks have emerged so far, in the fields of scientific computation and deep learning. With the increase in computational power and availability of data, neural networks, show a significant impact on solving PDEs [12–18]. In particular, the frameworks of the physics-informed neural networks (PINNs) were developed [19–21], and the automatic differentiation (AD) techniques were used to compute the derivative terms in time and space. The loss function is designed based on the physical principles underlying the governing equations. The back-propagation algorithm iteratively adjusts the weights of the neural networks. Subsequently, the PINNs framework has been further developed, with various improvements and applications. Chiu et al. [22] proposed a method that combines AD and the finite difference methods, which utilizes the neighboring points to calculate the derivatives. Patel et al. [23] proposed cvPINNs, combining the finite volume method with PINNs and utilizes the integral methods to the offset for derivative operations in equations. Other methods based on domain decomposition are also available, which divide the computational domains into a multiple of smaller subdomains [24–26]. Researchers have also studied some other variants of PINNs [27–32].

The continuous development of PINNs has led to widespread applications fluid mechanics [33–36]. There are two notable methods for solving the N–S equations, which deserves special attention. Jin et al. [33] proposed the NSFnets model for simulating the incompressible laminar and turbulent flows. This method directly encodes the governing equations in the deep neural networks to address the time-consuming issues of integrating multiple datasets and generating grids. Dwivedi et al. [35] developed a distributed PINNs framework for the incompressible N–S equations. It decomposes the computational domain into multiple subdomains with simple learning machines on each subdomain to solve problems, thereby reducing the burden. These methods show high accuracy and stability in solving the incompressible N–S equations.

The methods mentioned above have shown promising advantages, but they are only applicable to regular domains. Therefore, a new deep learning method is proposed in this study, which can solve the incompressible N–S equations with irregular domains. This study will focus on three aspects to reduce the errors as much as possible and improve the computational efficiency of the neural networks. Firstly, the loss function for PINNs is a combination of multiple weighted loss terms. Minimizing the loss function can be viewed as an optimization problem with multiple objectives. The issues of imbalanced each loss term can be addressed by adaptively assigning the corresponding weights to each loss term based on the maximum likelihood estimation using the Gaussian distributions. Secondly, an improved network is proposed with the global and local information. This design can transfer the input information to the hidden layers of the networks and capture the drastic parts of the PDEs solution. This enhances the expressive power of the deep neural networks with reducing the approximation errors. Thirdly, combining AD with ND can deal with the issues of computational efficiency and stability encountered in AD, thereby improving the performance of PINNs.

The paper is organized as follows. Section 2 details the incompressible N–S equations and the basic structure of the PINNs. In Sect. 3, the methods proposed in this study are presented, including the adaptive weighting, the improved neural networks, and the mixed differentiation. And the effectiveness of our method is validated through the numerical experiments in Sect. 4. Section 5 summarizes the paper and the future research prospects.

2 Preliminaries

2.1 Model equation

The non-dimensional incompressible N–S equations are considered in this study, which is expressed in a three-dimensional domain $\Omega \times (0, T]$ as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}}\nabla^2 \mathbf{u} \quad \text{in } \Omega \times (0, T], \tag{2.1a}$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega, \tag{2.1b}$$

$$\mathbf{u} = \mathbf{u}_{\Upsilon} \quad \text{on } \Upsilon_D, \tag{2.1c}$$

$$\frac{\partial \mathbf{u}}{\partial n} = 0 \quad \text{on } \Upsilon_N, \tag{2.1d}$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{s}(x) \quad \text{in } \Omega, \tag{2.1e}$$

where $\mathbf{u} = [u(\mathbf{x}, t), v(\mathbf{x}, t), w(\mathbf{x}, t)]$ is the non-dimensional velocity vector, $p(\mathbf{x}, t)$ is the non-dimensional pressure, and Re is the Reynolds number. The Dirichlet and Neumann boundary conditions are denoted with Υ_D and Υ_N, respectively. $\mathbf{s}(x)$ represents the initial velocity field.

2.2 PINNs

The standard PINNs (s-PINNs) approximate the mapping between the spatio–temporal domain points and the solution of PDEs using the training neural networks. In the s-PINNs framework, the core of the training model is the fully-connected neural networks. This type of neural networks typically consist of an input layer, an output layer, and n hidden layers. Figure 1 depicts the structure of the s-PINNs.

We will use the first component $\mathbf{u}(\mathbf{x}, t)$ as an example to outline the general approach, and similar methods can be applied to the other variables. Ultimately, the construction of the network will involve all three variables simultaneously. To effectively approximate the solution $\mathbf{u}(\mathbf{x}, t)$ of the N–S equations, the fully-connected neural networks represented as $\hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)$ are applied. The input of the neural networks consists of the independent variables $\mathbf{x}$ and t, while the output corresponds to the predicted solution $\hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)$ at that specific points. The connections between the layers are represented as follows:

$$\begin{aligned}
&Input \quad layer: \quad \hat{u}^i_N(\mathbf{x}, t) = [\mathbf{x}, t]^{\mathrm{T}},\\
&Hidden \quad layers: \quad \hat{u}^i_N(\mathbf{x}, t) = \Phi\left(\boldsymbol{W}^i \mathcal{N}^{i-1}(\mathbf{x}) + \boldsymbol{b}^i\right)\\
&\quad for \quad 2 \le i \le L-1,\\
&Output \quad layer: \quad \hat{u}^L_N(\mathbf{x}, t) = \boldsymbol{W}^L \mathcal{N}^{L-1}(\mathbf{x}) + \boldsymbol{b}^L,
\end{aligned} \tag{2.2}$$

where $\hat{u}^i_N(\mathbf{x}, t)$ and $\hat{u}^L_N(\mathbf{x}, t)$ are the inputs and outputs of the model. Φ is the activation function, usually taken as sigmoid or tanh, sin, etc [37]. The activation function realizes the nonlinear approximation of the model by nonlinear transformation of the output values. $\theta = (\mathbf{W}^i, \mathbf{b}^i)$ represents a set of weight matrices and the bias vectors of the i-th layer.

The loss functions of the s-PINNs consist of three components, including the initial and boundary conditions reflecting the spatio-temporal domains, and the residuals of the PDEs at the selected points in the domain (called collocation points). They are defined as the initial loss, the boundary loss and the residual loss, respectively. The output of the neural networks is denoted as $\hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)$, the three error parts of the s-PINNs loss functions are given below [38]:

The loss of the mean squared error of the initial condition is given as:

$$\mathcal{L}_{ini} = \mathrm{MSE}_{ini} = \frac{1}{N_{ini}} \sum_{i=1}^{N_{ini}} |\hat{\mathbf{u}}_N\left(\mathbf{x}^{ini}_i, 0\right) - \mathbf{s}^{ini}_i|^2, \quad \mathbf{x}^{ini}_i \in \Omega, \tag{2.3}$$

where $\hat{\mathbf{u}}_N\left(\mathbf{x}^{ini}_i, 0\right)$ is the output of the neural networks and $\mathbf{s}^{ini}_i$ is the given initial condition at $\left(\mathbf{x}^{ini}_i, 0\right)$.

The loss of the mean squared error on the boundary condition:

$$\mathcal{L}_{bcs} = \mathrm{MSE}_{bcs} = \frac{1}{N_{bcs}} \sum_{j=1}^{N_{bcs}} \left|\hat{\mathbf{u}}_N\left(\mathbf{x}^{bcs}_j, t^{bcs}_j\right) - \mathbf{h}^{bcs}_i\right|^2, \left(\mathbf{x}^{bcs}_j, t^{bcs}_j\right) \in \Upsilon \times (0, T], \tag{2.4}$$

$\mathbf{h}^{bcs}_j$ stands for the boundary conditions. The point set $\left(\mathbf{x}^{bcs}_j, t^{bcs}_j\right)$ represents the boundary points.

The mean squared error resulting from the residuals of the incompressible N–S equations:

$$\begin{aligned}
R_1(\mathbf{x}, t) &:= \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p - \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u},\\
R_2(\mathbf{x}, t) &:= \nabla \cdot \mathbf{u}.
\end{aligned} \tag{2.5}$$

$$\mathcal{L}_{res} = \mathrm{MSE}_{res} = \frac{1}{N_{res}} \sum_{k=1}^{N_{res}} \left|R_1\left(\mathbf{x}^{res}_k, t^{res}_k\right)\right|^2 + \left|R_2\left(\mathbf{x}^{res}_k, t^{res}_k\right)\right|^2, \quad \left(\mathbf{x}^{res}_k, t^{res}_k\right) \in \Upsilon \times (0, T], \tag{2.6}$$

where $\left(\mathbf{x}^{res}_k, t^{res}_k\right)$ are the internal residual points that are randomly sampled and uniformly distributed. The Latin hypercube sampling method [39] is used to obtain the points within the interior domain and boundaries. Therefore, the expression for the total loss function can be written as follows:

$$\mathcal{L}_{sum} = \mathcal{L}_{ini} + \mathcal{L}_{bcs} + \mathcal{L}_{res}. \tag{2.7}$$

The loss function $\mathcal{L}_{sum}$ was then minimized using the optimization techniques like Adam, LBFGS, and SGD until it is close to zero [40]. The ideal parameter is

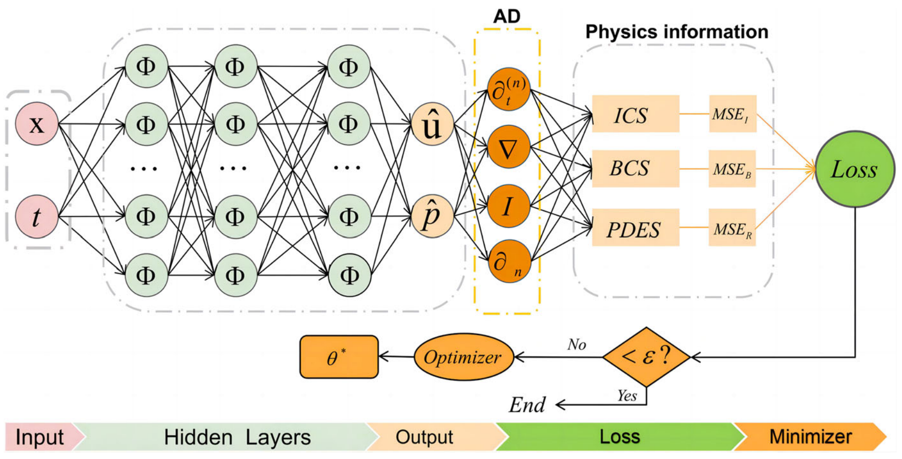


Fig. 1 The architecture of the standard physics-informed neural networks

finally obtained after a specific number of iterative training.

The relative discrete l^2-error is introduced to evaluate the deviation between the approximate solution $\hat{\mathbf{u}}_N\left(\mathbf{x}_i^*, t_i^*\right)$ of the neural networks and the exact solution $\mathbf{u}\left(\mathbf{x}_i^*, t_i^*\right)$ of the PDEs.

$$\mathcal{E}_{\text{error}} := \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{\mathbf{u}}_N\left(\mathbf{x}_i^*, t_i^*\right) - \mathbf{u}\left(\mathbf{x}_i^*, t_i^*\right)\right)^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{u}\left(\mathbf{x}_i^*, t_i^*\right)\right)^2}}, \tag{2.8}$$

where $\hat{\mathbf{u}}_N\left(\mathbf{x}_i^*, t_i^*\right)$ is the output for a series of test point networks $\left\{\left(\mathbf{x}_i^*, t_i^*\right)\right\}_{i=1}^{N}$, and $\mathbf{u}\left(\mathbf{x}_i^*, t_i^*\right)$ denotes the exact value.

3 Solution methodology

In this section, the maximum likelihood estimation of the Gaussian distribution is utilized to adaptively balance the weights of each loss term. According to the attention mechanism in computer vision, the s-PINNs are improved by introducing the additional spatio-temporal variable sets. Furthermore, the improved neural networks utilize the mixed differentiation to compute the differential operators, thereby enhancing the computational efficiency of the training neural networks.

3.1 Adaptive weighting method for PINNs

The schematic diagram of the adaptive weighting PINNs (aw-PINNs) is shown in Fig. 2. Following the methods in Ref [41], there are effective methods for determining the weightings of the multiple loss functions in scene geometry and semantic multi-task deep learning problems. The principle is to assign the different weights to the various losses based on their contribution to the overall performance of the model.

When solving the N–S equations, we model the output $\hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)$ of PINNs as a Gaussian probability distribution,

$$P(u \mid \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)) = N\left(\hat{\mathbf{u}}_N(\mathbf{x}, t; \theta), \xi^2\right), \tag{3.1}$$

with an uncertainty parameter ξ.

The uncertainty parameter for the decay of the weights is adjusted using the maximum likelihood inference. According to the probability density function of the Gaussian distribution:

$$P(\zeta \mid \theta) = \frac{1}{\sqrt{2\pi}\,\xi}\exp\left(-\frac{(\zeta - \omega)^2}{2\xi^2}\right), \tag{3.2}$$

where ω represents the mathematic expectation, and ξ is the standard deviation. By minimizing the objective and preventing data underflow, the negative log-likelihood of the model is minimized following Eq. (3.2),

$$\begin{aligned}
&-\log P(u \mid \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)) \\
&\quad= -\log N\left(\hat{\mathbf{u}}_N(\mathbf{x}, t; \theta), \xi^2\right) \\
&\quad\propto \frac{1}{2\xi^2}\left|u - \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)\right|^2 + \log\xi \\
&\quad= \frac{1}{2\xi^2}\mathcal{L}(\theta) + \log\xi,
\end{aligned} \tag{3.3}$$

where the weighted loss functions and the uncertainty regularization terms for each task are considered. The parameter ξ of each epoch is updated iteratively using the maximum likelihood estimation to adapt the weight of each loss term.

The loss of the initial and boundary conditions can be represented by the Gaussian probability models of the outputs s and h.

$$\begin{aligned} & P(u, s, h \mid \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)) \\ & \quad = P(u \mid \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)) \cdot P(s \mid \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)) \\ & \quad \cdot P(h \mid \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)) \\ & \quad = N\left(\hat{\mathbf{u}}_N(\mathbf{x}, t; \theta), \xi_r^2\right) \cdot N\left(\hat{\mathbf{u}}_N(\mathbf{x}, t; \theta), \xi_i^2\right) \\ & \quad \cdot N\left(\hat{\mathbf{u}}_N(\mathbf{x}, t; \theta), \xi_b^2\right). \end{aligned} \tag{3.4}$$

We aim to minimize the joint probability distribution of the multi-output model.

$$\begin{aligned} & -\log P(u, s, h \mid \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)) \propto \frac{1}{2\xi_r^2}|u - \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)|^2 \\ & \quad + \frac{1}{2\xi_i^2}|s - \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)|^2 \\ & \quad + \frac{1}{2\xi_b^2}|h - \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)|^2 + \log \xi_r \xi_i \xi_b \\ & = \frac{1}{2\xi_r^2}\mathcal{L}_{\text{res}}(\theta) + \frac{1}{2\xi_i^2}\mathcal{L}_{ini}(\theta) \\ & \quad + \frac{1}{2\xi_b^2}\mathcal{L}_{bcs}(\theta) + \log \xi_r \xi_i \xi_b. \end{aligned} \tag{3.5}$$

In a word, the loss functions are defined for the PINNs using a multi-output model with four vectors. The loss function of the adaptive weighting PINNs can be expressed as follows:

$$\begin{aligned} \mathcal{L}(\theta; \xi) = & \frac{1}{2\xi_r^2}\mathcal{L}_{res}(\theta) + \frac{1}{2\xi_i^2}\mathcal{L}_{ini}(\theta) \\ & + \frac{1}{2\xi_b^2}\mathcal{L}_{bcs}(\theta) + \log \xi_r \xi_i \xi_b, \end{aligned} \tag{3.6}$$

where $\xi = \{\xi_r, \xi_i, \xi_b\}$ represents the adaptive weighting coefficients assigned to each loss term. $\lambda = \{\lambda_{res}, \lambda_{ini}, \lambda_{bcs}\}$, $\lambda := \frac{1}{2\xi^2}$ denotes the overall weight of each loss term. Thus, the weight of each loss term can be adjusted automatically and systematically. During the training process, we introduce a trainable and adaptive parameter $Q = \{Q_r, Q_i, Q_b\}$ to prevent the denominator from being zero, where $Q := \log \xi^2$ improves the robustness of the model. The final adaptive weighting loss function can be represented as:

$$\begin{aligned} \mathcal{L}(\theta; Q) = & \frac{1}{2}\exp(-Q_r)\mathcal{L}_{res}(\theta) \\ & + \frac{1}{2}\exp(-Q_i)\mathcal{L}_{ini}(\theta) \\ & + \frac{1}{2}\exp(-Q_b)\mathcal{L}_{bcs}(\theta) \\ & + \frac{1}{2}(Q_r + Q_i + Q_b) \\ & \propto \exp(-Q_r)\mathcal{L}_{res}(\theta) + \exp(-Q_i)\mathcal{L}_{ini}(\theta) \\ & + \exp(-Q_b)\mathcal{L}_{bcs}(\theta) + Q_r + Q_i + Q_b. \end{aligned} \tag{3.7}$$

We use Eq. (3.7) as the final loss function for aw-PINNs. The loss function can be minimized through the exponential mapping without constraints. The adaptive weights converge slowly to zero because the exponential function produces only positive values, which contributes to improve numerical stability during training.

3.2 Improved network structure for PINNs

To enhance the approximation capability of adaptive the weighting neural networks, we have devised a novel network architecture, drawing inspiration from the attention mechanisms that are prominent in transformers [42]. The neural networks with attention mechanisms are currently a hot research topic in deep learning. They enable the neural networks to selectively focus on the specific parts of the input data when processing information, rather than processing all input data equally. This selective focus can greatly improve the accuracy and efficiency of the neural networks in various tasks. The attention technique used by the neural networks is to merge the encoder–decoder framework, as described in [43]. The encoder–decoder framework encodes variable-length input sequences into fixed-length vectors, which are then processed by neural network layers to decode them into corresponding variable-length output sequences. Applying the encoder–decoder framework to PINNs, the model adds three additional transformer networks γ_1, γ_2 and γ_3 to enhance the performance and accuracy of the model. Finally, the neural networks will generate the output $\hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)$ by decoding. The designed network framework connects three transformer networks to two fully connected trunk networks. The improved neural network introduces additional spatial and temporal variables as inputs, which makes the input information

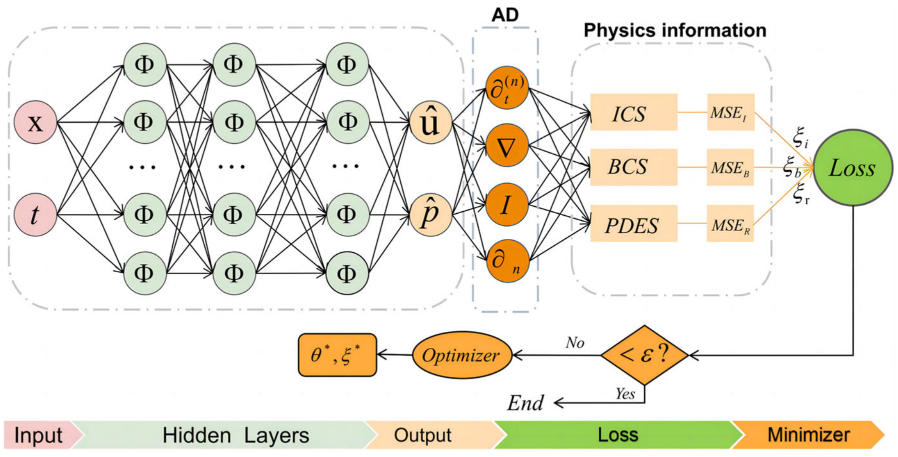


Fig. 2 The schematic diagram of the adaptive weighting physics-informed neural networks

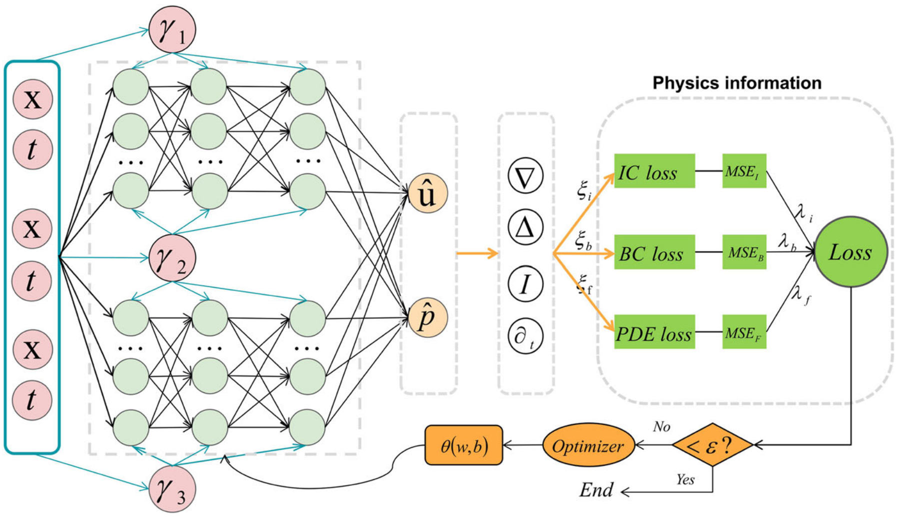


Fig. 3 Improved neural networks: attention mechanisms are combined with neural networks, with only additional weights and biases for the transformer networks

easier to transfer the hidden layers. Each transformer network structure is shown in Fig. 4. This design can deal with the global and local information at the same time, with maintaining the integrity of the input and enhancing the expressive capacity of the neural networks. We call the improved adaptive weighting PINNs (iaw-PINNs). The model framework of the iaw-PINNs is shown in Fig. 3. The improved adaptive weighting method is summarized as the algorithm1.

The hidden layers of the iaw-PINNs is updated based on the forward propagation rules:

$$
\begin{aligned}
\gamma_1 &= \Phi\left(\mathrm{W}_0^1 X + \mathrm{b}_0^1\right), \\
\gamma_2 &= \Phi\left(\mathrm{W}_0^2 X + \mathrm{b}_0^2\right), \\
\gamma_3 &= \Phi\left(\mathrm{W}_0^3 X + \mathrm{b}_0^3\right),
\end{aligned}
$$

$$
\begin{aligned}
& Z^1 = \Phi\left(\mathrm{W}^1 X + \mathrm{b}^1\right), \\
& Y_1^{k+1} = \Phi\left(\mathrm{W}_0^{k+1} Z^k + \mathrm{b}_0^{k+1}\right), \\
& k = 1, 2, \cdots, L-2, \\
& Y_2^{k+1} = \Phi\left(\mathrm{W}_1^{k+1} Z^k + \mathrm{b}_1^{k+1}\right), \\
& k = 1, 2, \cdots, L-2, \\
& H^{k+1} = \left(I - Y_1^{k+1} - Y_2^{k+1}\right) \odot \gamma_1 + Y_1^{k+1} \odot \gamma_2 \\
& \quad + Y_2^{k+1} \odot \gamma_3, \quad k = 1, 2, \cdots, L-2, \\
& \hat{\mathbf{u}}_N(\mathbf{x}, t; \theta) = H^L\, \mathrm{W}^L + b^L.
\end{aligned}
\tag{3.8}
$$

where $X = [\mathbf{x}, t]^{\mathrm{T}}$ denotes the input training datapoints, Φ is the activation function and $\odot$ denotes the element-wise multiplication. $\left\{\left(\mathrm{W}_0^1, \mathrm{b}_0^1\right), \left(\mathrm{W}_0^1, \mathrm{b}_0^1\right), \left(\mathrm{W}_0^1, \mathrm{b}_0^1\right)\right\}$ are the weights and the biases used in the addition of three transformer networks. γ_1, γ_2 and γ_3 represent the transformer networks that incorporate the

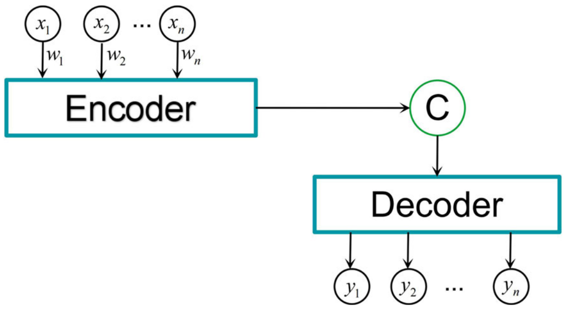
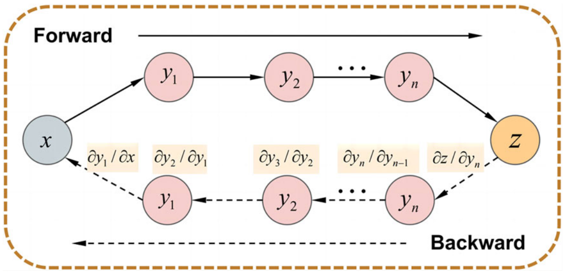


Fig. 4 The architecture of the transformer network



Forward
Backward
x
y_1
y_2
y_n
z
$\partial y_1/\partial x$
$\partial y_2/\partial y_1$
$\partial y_3/\partial y_2$
$\partial y_n/\partial y_{n-1}$
$\partial z/\partial y_n$

Fig. 5 Schematic diagram of the backward propagation for derivative computation in fully connected feed-forward neural networks

extra primitive input X. Z^1 is the input to the first hidden layer of an ordinary fully-connected neural network, while Y_1 and Y_2 are the two fully connected trunk networks. For example, when $k = 1$, if there are no additional γ_1, γ_2 and γ_3, then Y_1 and Y_2 represent the outputs produced by the first hidden layer of the two trunk fully connected networks. By introducing the extra γ_1, γ_2 and γ_3, the input to the second hidden layer is no longer Y_1 and Y_2, but rather H^2 that is transformed from γ_1, γ_2 and γ_3. I is a matrix whose elements are all equal to one.

3.3 Mixed differentiation for PINNs

There are significant differences between ND and AD when it comes to computing the differential operators. ND approximates the derivative term by utilizing the local support points, whereas AD precisely computes the derivatives at any given point. The AD in Tensorflow [44] is a built-in function that can be called directly. For derivative operations in the neural networks, the x-derivative of the output z concerning the input can be calculated according to the reverse chain shown in Fig. 5. The advantages of AD and ND are

Algorithm1 : Adaptive weighting method for improved the network structure.

```
Step1: Consider an improved fully connected neural network to define the PINNs
       $\hat{\mathbf{u}}_N(\mathbf{x}, t; \theta)$.
Step2: Compute the residuals of the PDEs with automatic differentiation and establish
       terms for loss functions.(Eq.(2.3),Eq.(2.4),Eq.(2.6)).
Step3: Initialize the adaptive weights collection $\xi = \{\xi_r, \xi_i, \xi_b\}$.
Step4: Build Gaussian probability models using the mean generated by PINNs and
       the adapted collection of weights $\xi$.
Step5: Using the definition in Eq.(3.8),the hidden layers are updated.
Step6: Use $L$ steps gradient descent algorithm iterations to update the parameters
       $\xi$ and $\theta$ as:
for $k = 1$ to $L$ do
$Z^1 = \Phi\left(X\,\mathrm{W}^1 + \mathrm{b}^1\right)$,
$Y_1^{k+1} = \Phi\left(\mathrm{W}^{k+1} Z^k + \mathrm{b}^{k+1}\right), \quad k = 1, 2, \cdots, L-2$,
$Y_2^{k+1} = \Phi\left(\mathrm{W}^{k+1} Z^k + \mathrm{b}^{k+1}\right), \quad k = 1, 2, \cdots, L-2$,
$H^{k+1} = \left(I - Y_1^{k+1} - Y_2^{k+1}\right) \odot \gamma_1 + Y_1^{k+1} \odot \gamma_2 + Y_2^{k+1} \odot \gamma_3, \quad k = 1, 2, \cdots, L-2$,
(1) Create the adaptive weighting loss function $\mathcal{L}(\theta_k; \xi_k)$ (Eq.(3.6)) using the maximum
    likelihood estimation as the basis.
(2) Adjust the adaptive weights collection $\xi$ using the Adam+L-BFGS optimizer to maximize
    the likelihood of meeting the constraints.
    $\xi_{k+1} \leftarrow \text{Adam} + \text{L-BFGS}\left(\mathcal{L}(\theta_k; \xi_k)\right)$
(3) Optimize the network parameters $\theta$ using Adam+L-BFGS optimizer.
    $\theta_{k+1} \leftarrow \text{Adam} + \text{L-BFGS}\left(\mathcal{L}(\xi_k; \theta_k)\right)$
end for
Return
The optimal values of the model parameters $\theta^*$ and the updated adaptive weights collection
$\xi^*$are obtained at the end.
```

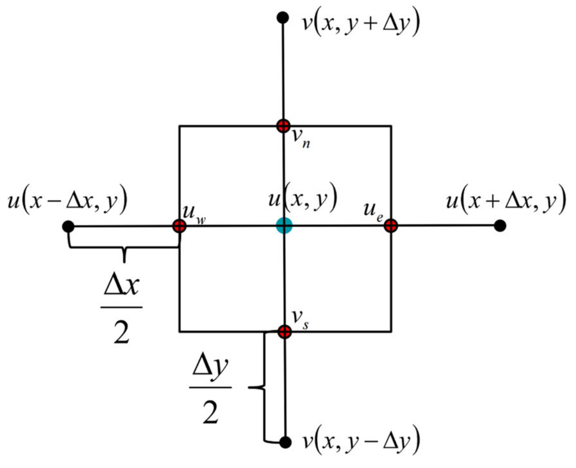


Fig. 6 The schematic of the approximate derivative of the mixed differentiation for PINNs, the red and black points are the additional support points

combined to construct a loss function coupling the neighboring support points and their derivative terms. The AD and ND are combined as the mixed differentiation in this work, which captures the physical features more accurately with greater training efficiency as compared to that of AD [22].

To estimate the first-order derivative $\frac{\partial \mathbf{u}(x)}{\partial x}$, we employ the following approach.

$$\frac{\partial \mathbf{u}(x)}{\partial x} = \frac{\hat{\mathbf{u}}_{Ne} - \hat{\mathbf{u}}_{Nw}}{\Delta x} = \frac{\hat{\mathbf{u}}_N(x+\Delta x/2) - \hat{\mathbf{u}}_N(x-\Delta x/2)}{\Delta x}, \tag{3.9}$$

The distance between the two adjacent points is represented by Δx in Eq. (3.9). For the mixed differentiation method, Δx is a hyper-parameter, and the $\hat{\mathbf{u}}_N$ value at $(x+\Delta x)$ and $(x-\Delta x)$ is obtained using $\hat{\mathbf{u}}_N(x+\Delta x;\theta)$ and $\hat{\mathbf{u}}_N(x-\Delta x;\theta)$, as illustrated in Fig. 6.

3.3.1 The second-order upwind scheme for PINNs

Take the velocity component $u(\mathbf{x}, t)$ as an example to illustrate the scheme. As for the first-order derivative term $\hat{u}_{Nx}|_m$ of the proposed mixed differentiation, according to the multi-moment method mentioned in [45], we use $\hat{u}_N$ and $\hat{u}_{Nx}$ to approximate the first-order derivative term $\hat{u}_{Nx}|_m$, where $\hat{u}_{Nx}$ is obtained using AD. To couple $\hat{u}_N$ and $\hat{u}_{Nx}$, $\hat{u}_{Ne}$ is approximated as $u_e|_{\mathrm{m}(us)}$ [22]:

$$\hat{u}_{Ne} \cong u_e\big|_{\mathrm{m}(us)} = a\hat{u}_N(x,t;\theta) + b\hat{u}_{Nx}(x,t;\theta). \tag{3.10}$$

We perform Taylor series expansions on $\hat{u}_N(x,t;\theta)$ and $\hat{u}_{Nx}(x,t;\theta)$ with respect to $\hat{u}_{Ne}$ and $\frac{\partial \hat{u}_{Ne}}{\partial x}$.

$$\begin{aligned}
\hat{u}_N(x,t;\theta) &= \hat{u}_{Ne} - \frac{\Delta x}{2}\frac{\partial \hat{u}_{Ne}}{\partial x} + \left(\frac{\Delta x}{2}\right)^2 \frac{\partial^2 \hat{u}_{Ne}}{\partial x^2} \\
&\quad - \left(\frac{\Delta x}{2}\right)^3 \frac{\partial^3 \hat{u}_{Ne}}{\partial x^3} + \left(\frac{\Delta x}{2}\right)^4 \frac{\partial^4 \hat{u}_{Ne}}{\partial x^4} + \cdots, \\
\hat{u}_{Nx}(x,t;\theta) &= \frac{\partial \hat{u}_{Ne}}{\partial x} - \frac{\Delta x}{2}\frac{\partial^2 \hat{u}_{Ne}}{\partial x^2} + \left(\frac{\Delta x}{2}\right)^2 \frac{\partial^3 \hat{u}_{Ne}}{\partial x^3} \\
&\quad - \left(\frac{\Delta x}{2}\right)^3 \frac{\partial^4 \hat{u}_{Ne}}{\partial x^4} + \left(\frac{\Delta x}{2}\right)^4 \frac{\partial^5 \hat{u}_{Ne}}{\partial x^5} + \cdots,
\end{aligned} \tag{3.11}$$

By eliminating the primary error terms for the above two equations, we can obtain the values of a and b:

$$\begin{aligned} a &= 1, \\ b &= \frac{a\Delta x}{2} = \frac{\Delta x}{2}. \end{aligned} \tag{3.12}$$

Substituting a and b in Eq. (3.10) we get:

$$u_e|_{\mathrm{m}(us)} = \hat{u}_N(x,t;\theta) + \frac{\Delta x}{2}\hat{u}_{Nx}(x,t;\theta). \tag{3.13}$$

$u_w|_\mathrm{m}$ can be obtained in the same way as:

$$u_w|_{\mathrm{m}(us)} = \hat{u}_N(x-\Delta x,t;\theta) + \frac{\Delta x}{2}\hat{u}_{Nx}(x-\Delta x,t;\theta). \tag{3.14}$$

The first-order derivative can be approximately expressed as follows:

$$\begin{aligned}
\frac{\partial u(x)}{\partial x} &\cong \frac{\partial u(x)}{\partial x}\bigg|_{\mathrm{m}(us)} \\
&= \frac{u_e|_{\mathrm{m}(us)} - u_w|_{\mathrm{m}(us)}}{\Delta x} \\
&= \frac{\hat{u}_N(x,t;\theta) - \hat{u}_N(x-\Delta x,t;\theta)}{\Delta x} \\
&\quad + \frac{1}{2}\left(\hat{u}_{Nx}(x,t;\theta) - \hat{u}_{Nx}(x-\Delta x,t;\theta)\right).
\end{aligned} \tag{3.15}$$

The above equations have been streamlined as follows:

$$\begin{aligned}
&\frac{\partial u(x)}{\partial x}\bigg|_{\mathrm{m}(us)} \\
&= \hat{u}_{Nx}(x,t;\theta) \\
&\quad + \left(\frac{\hat{u}_N(x,t;\theta) - \hat{u}_N(x-\Delta x,t;\theta)}{\Delta x} \right. \\
&\quad \left. - \frac{1}{2}\left(\hat{u}_{Nx}(x,t;\theta) + \hat{u}_{Nx}(x-\Delta x,t;\theta)\right)\right).
\end{aligned} \tag{3.16}$$

Following the Taylor series, by expanding $\hat{u}_N(x-\Delta x, t;\theta)$ and $\hat{u}_{Nx}(x-\Delta x, t;\theta)$ at $\hat{u}_N(x,t;\theta)$ and $\hat{u}_{Nx}(x,t;\theta)$, it can then be further simplified as:

$$\left.\frac{\partial u(x)}{\partial x}\right|_{\mathrm{m(us)}} = \hat{u}_{Nx}(x,t;\theta) - \left(\frac{\Delta x^2}{12}\right)\hat{u}_N^{(3)}(x,t;\theta) + \left(\frac{\Delta x^3}{24}\right)\hat{u}_N^{(4)}(x,t;\theta) + \cdots . \tag{3.17}$$

The second-order upwind scheme introduces the additional stabilization terms with the second-order accuracy. When Δx approaches to zero, $\hat{u}_{Nx}$ is the derivative of AD.

3.3.2 The central difference scheme for PINNs

When solving the incompressible N–S equations, the traditional central difference methods for the pressure-gradient terms may lead to decoupling between the velocity and pressure variables. To address the decoupling issue, we adopt the central difference scheme of the mixed differentiation method to approximate the pressure-gradient terms [22]. Following the same approach, we can derive the expressions for $\hat{p}_e$ and $\hat{p}_w$.

$$\hat{p}_e \cong p_e\big|_{m(cd)} = \frac{\hat{p}_N(x+\Delta x,t;\theta)+\hat{p}_N(x,t;\theta)}{2} - \frac{\Delta x}{8}\left(\hat{p}_{Nx}(x+\Delta x,t;\theta) - \hat{p}_{Nx}(x,t;\theta)\right), \tag{3.18}$$

$$\hat{p}_w \cong p_w\big|_{m(cd)} = \frac{\hat{p}_N(x,t;\theta)+\hat{p}_N(x-\Delta x,t;\theta)}{2} - \frac{\Delta x}{8}\left(\hat{p}_{Nx}(x,t;\theta) - \hat{p}_{Nx}(x-\Delta x,t;\theta)\right), \tag{3.19}$$

The first-order derivative can be approximately expressed as follows:

$$\frac{\partial p(x)}{\partial x} \cong \left.\frac{\partial p(x)}{\partial x}\right|_{\mathrm{m}(cd)} = \frac{p_e|_{\mathrm{m}(cd)} - p_w|_{\mathrm{m}(cd)}}{\Delta x} = \frac{\hat{p}_N(x+\Delta x,t;\theta)-\hat{p}_N(x-\Delta x,t;\theta)}{2\Delta x} - \frac{1}{8}\left(\hat{p}_{Nx}(x+\Delta x,t;\theta) - 2\hat{p}_{Nx}(x,t;\theta) + \hat{p}_{Nx}(x-\Delta x,t;\theta)\right). \tag{3.20}$$

Equation (3.20) is simplified as follows:

$$\left.\frac{\partial p(x)}{\partial x}\right|_{\mathrm{m}(cd)} = \hat{p}_{Nx}(x,t;\theta) + \left(\frac{\hat{p}_N(x+\Delta x,t;\theta)-\hat{p}_N(x-\Delta x,t;\theta)}{2\Delta x} - \frac{1}{8}\left(\hat{p}_{Nx}(x+\Delta x,t;\theta)+6\hat{p}_{Nx}(x,t;\theta) + \hat{p}_{Nx}(x-\Delta x,t;\theta)\right)\right), \tag{3.21}$$

To further simplify into the following format:

$$\left.\frac{\partial p(x)}{\partial x}\right|_{\mathrm{m}(cd)} = \hat{p}_{Nx}(x,t;\theta) + \left(\frac{\Delta x^2}{24}\right)\hat{p}_N^{(3)}(x,t;\theta) - \frac{\Delta x^4}{480}\hat{p}_N^{(5)}(x,t;\theta) + \cdots . \tag{3.22}$$

From the above Eq. (3.20), it is evident that the equation involves both the contribution of the adjacent points and the collocated contribution $\hat{p}_{Nx}(x;t;\theta)$. Equation (3.22) shows that the theoretical accuracy of the central difference scheme is also second-order. The fundamental analysis in [22] shows that the proposed schemes have better dispersion and dissipation performance, when compared with the baseline scheme. Furthermore, the inclusion of AD leads to more precise solutions than those of the central difference scheme.

4 Numerical experiments

In this section, the proposed method is used to simulate the two-dimensional unsteady Taylor vortex problem, the two-dimensional steady Kovasznay flow, the three-dimensional unsteady Beltrami flow, the Flow in a Lid-driven cavity and the forward and inverse problem of Cylinder wake. The convection terms are approximated using the second-order upwind scheme, whereas the pressure-gradient terms are approximated using the central difference scheme to account for the physical properties of the different derivative terms. We assessed the efficacy of the proposed approach using the relative l^2-error and compared it with that of s-PINNs.

4.1 Taylor vortex problem

The Taylor vortex flow [46] is simulated to verify the effectiveness of our method for solving the incompressible fluids. There is an analytical solution expressed as:

$$\begin{aligned} u(\mathbf{x},t) &= -\cos(\pi x)\sin(\pi y)\exp\left(-\frac{2\pi^2 t}{\mathrm{Re}}\right), \\ v(\mathbf{x},t) &= \sin(\pi x)\cos(\pi y)\exp\left(-\frac{2\pi^2 t}{\mathrm{Re}}\right), \\ p(\mathbf{x},t) &= -\frac{1}{4}[\cos(2\pi x)+\cos(2\pi y)]\exp\left(-\frac{4\pi^2 t}{\mathrm{Re}}\right). \end{aligned} \tag{4.1}$$

In this case, Re=1000, and the initial and boundary conditions can be derived from Eq. (4.1). The simulation was conducted within a square domain $[-1, 1]^2$ with a circular boundary, and the radius is $r = 0.5$. For this problem, we utilized a network structure consists of 4 hidden layers, and each layer contains 50 neurons. We randomly generate a subset of samples in a size of 1000 for the initial boundary value training data. The boundary points are equidistantly spaced collocation points, as shown in Fig. 7a. The difference from the s-PINNs sampling is shown in Fig. 7b. Taking 10000 equidistantly spaced collocation points are used in the solution domain to apply to Eq. (2.1), the sampling distribution is displayed in Fig. 7c. The space-filling Latin hypercube sampling strategy is applied in the s-PINNs to generate the randomly sampled points as shown in Fig. 7d. The equidistantly spaced collocation points can cover the entire solution region, accurately capturing the variation characteristics of the function within the region. Therefore, the evenly spaced sampling points were selected as the interpolation basis points to simplify the calculation process and improve the accuracy of the model. Figure 8 shows the downward trend of the loss functions of the s-PINNs and the improved PINNs, while the improved PINNs display a more rapid decline particularly at the initial stages. Figure 9 presents the prediction solution of velocity $u(\mathbf{x}, t)$, $v(\mathbf{x}, t)$, and pressure $p(\mathbf{x}, t)$ and the comparison with the exact solutions. The experimental results show that the error between the predicted and the real values is very small, and the prediction performance is excellent. The performance decline of the loss functions is compared in Fig. 10 throughout the entire iteration epoch via boxplot. The red horizontal lines refer to the usual medians, and the minimum value is marked in the form of a small circle. Observations indicate that the loss function associated with our method diminishes at a more gradual rate, which leads to the enhanced training performance. This method can accurately predict the flow behavior of Taylor vortex. It can be seen that the proposed method also achieves better prediction results than the s-PINNs. For the Taylor-vortex problem, the method adopted in this paper improves the accuracy of velocity and pressure by two orders of magnitude under the same conditions, as shown in Table 1. Table 2 presents the relative discrete l^2-error between the predicted solution and the exact solution for different numbers of sampling points.

Table 1 Relative discrete l^2-error comparison between the predicted and the exact solution obtained using different methods for the Taylor vortex problem

Method	Layers	Neurons	ε_u	ε_v	ε_p	Total loss
Standard PINNs	4	50	5.9899e−03	5.9514e−03	1.9402e−02	2.8398e−04
Adaptive weighting (aw)	4	50	1.9397e−04	2.1026e−04	3.8006e−03	5.5799e−07
Improved network (in)	4	50	7.6382e−04	8.7836e−04	5.0853e−03	7.3092e−07
Mixed differentiation (md)	4	50	5.8693e−04	6.2351e−04	2.9563e−03	8.9512e−07
aw-in-PINNs	4	50	8.6721e−05	9.6401e−05	6.6609e−04	9.2089e−08
aw-in-md-PINNs	4	50	6.5724e−05	7.6401e−05	1.3098e−04	7.6544e−08

Table 2 Relative discrete l^2-errors between the predicted and the exact solutions for the Taylor vortex problem at different numbers of residual points

Method	Layers	Neurons	Residual points	ε_u	ε_v	ε_p
Standard PINNs	4	50	1000	7.6956e−03	6.0901e−02	3.0400e−02
aw-in-md-PINNs	4	50	1000	8.0391e−05	8.0510e−05	3.1242e−04
Standard PINNs	4	50	10000	5.9899e−03	5.9514e−03	1.9402e−02
aw-in-md-PINNs	4	50	10000	6.5724e−05	7.6401e−05	1.3098e−04
Standard PINNs	4	50	50000	7.9782e−03	7.0939e−03	3.6201e−02
aw-in-md-PINNs	4	50	50000	8.6921e−05	9.0058e−05	4.5628e−04

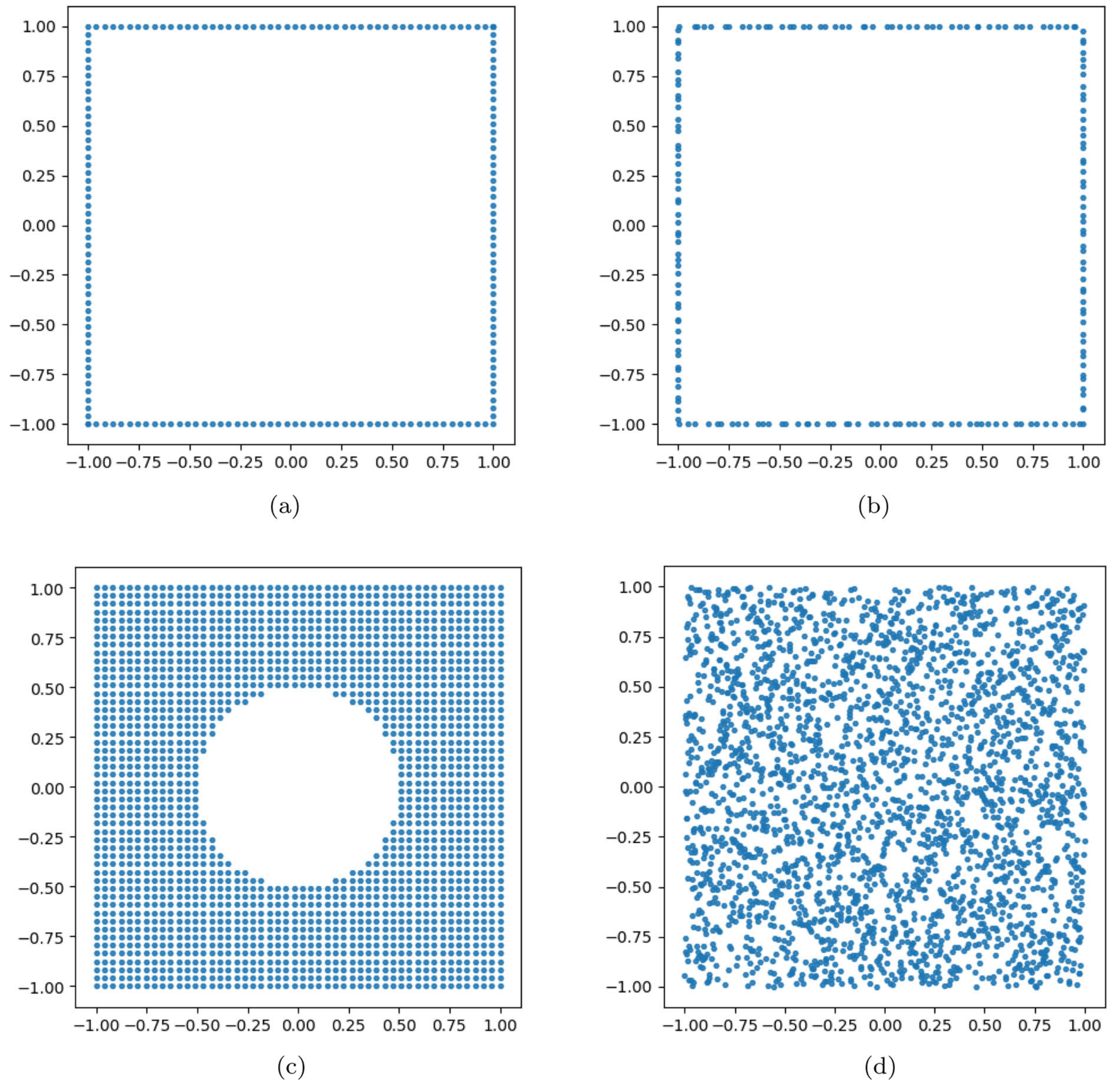


Fig. 7 **a** The sampling points of the improved PINNs on the boundary. **b** The sampling points of s-PINNs on the boundary. **c** The sampling points of the improved PINNs within the region. **d** The sampling points of the s-PINNs within the region

4.2 The Kovasznay flow

The two-dimensional steady Kovasznay flow is further studied. We utilize the Kovasznays analytical solution as the benchmark for comparison. For a given viscosity of $\nu = 0.1$, this solution is given by

$$\begin{aligned} u^*(x, y) &= 1 - e^{\lambda x}\cos(2\pi y), \\ v^*(x, y) &= \frac{\lambda e^{\lambda x}}{2\pi}\sin(2\pi y), \\ p^*(x, y) &= -\frac{1}{2}e^{\lambda x} + 1/2. \end{aligned} \tag{4.2}$$

where

$$\lambda = \frac{-8\pi^2}{\nu^{-1} + \sqrt{\nu^{-2} + 64\pi^2}}, \tag{4.3}$$

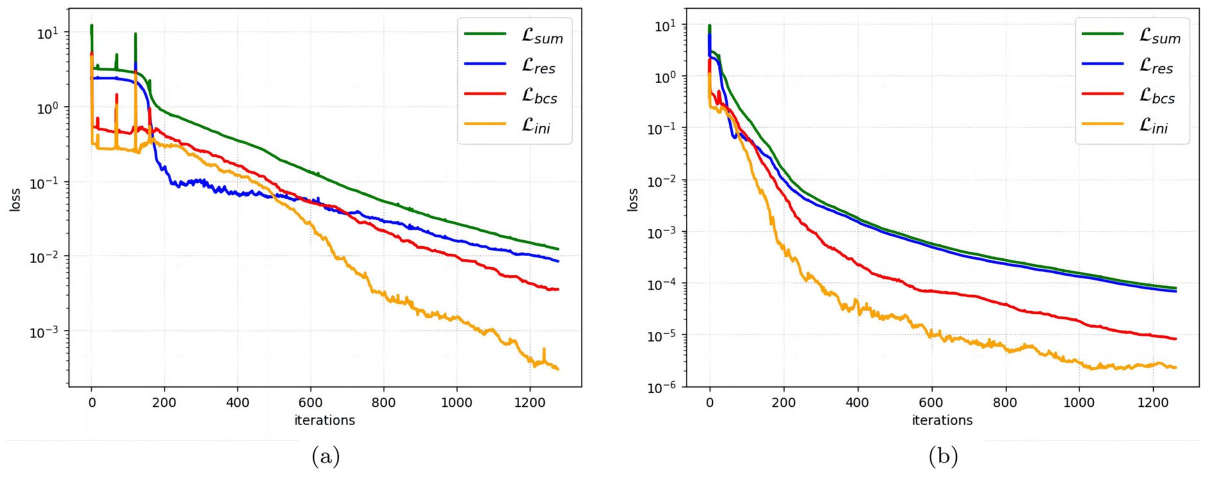


Fig. 8 **a** The graph depicts the decreasing trend of the loss function obtained from training with s-PINNs. **b** The graph depicts the decreasing trend of the loss function obtained from training with improved PINNs

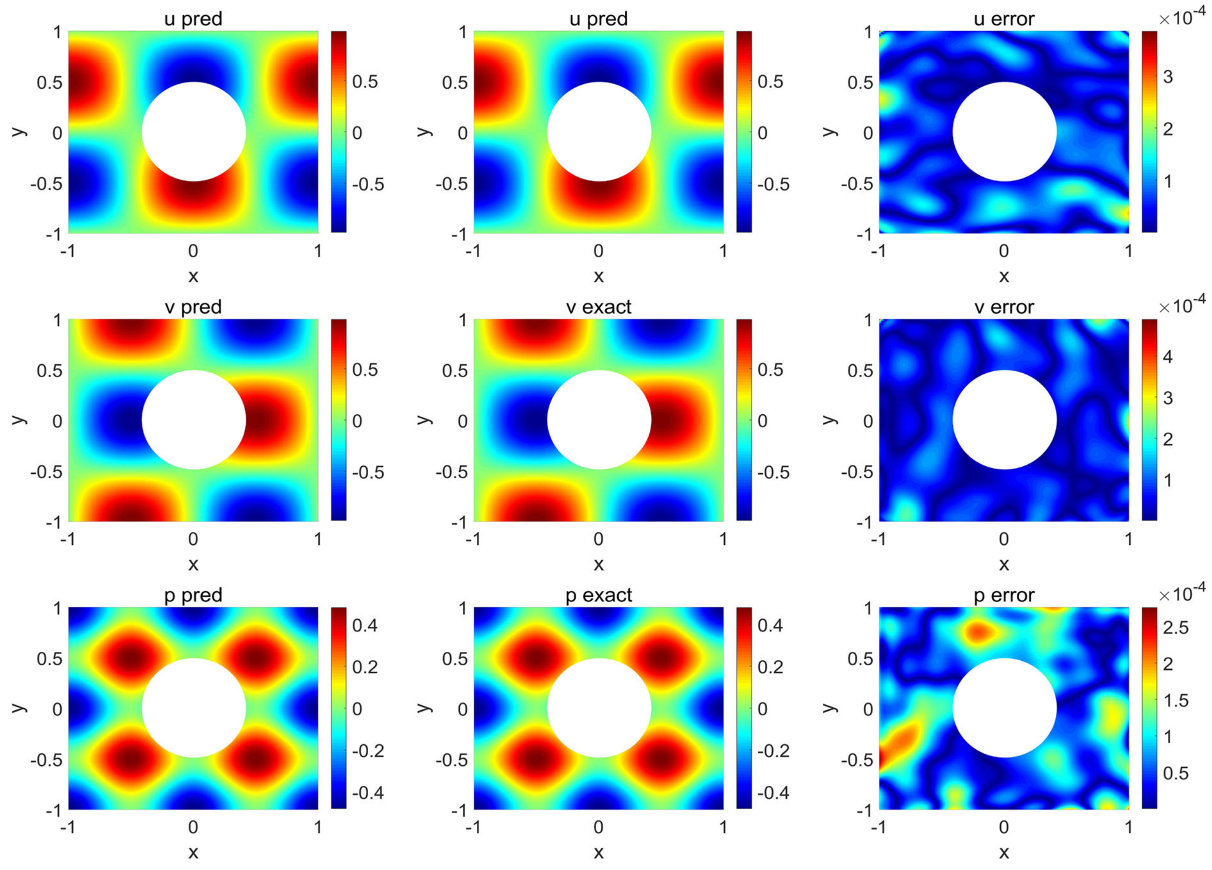
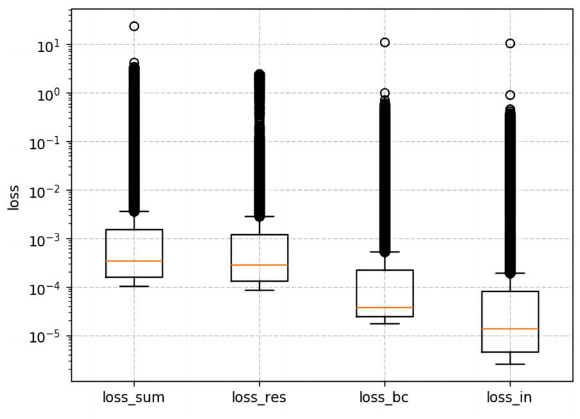
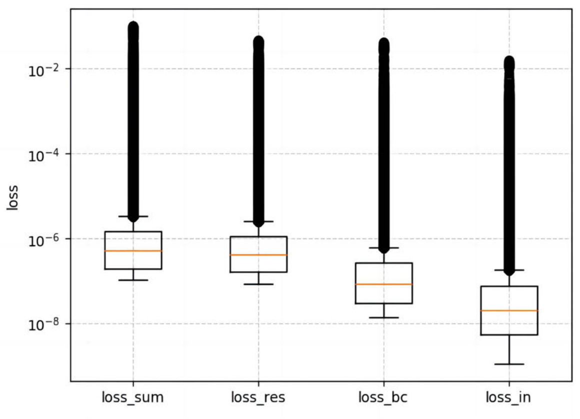


Fig. 9 Distribution of velocity and pressure of prediction solution, exact solution, and absolute pointwise error for Taylor vortex problem a snapshot in time $t = 1$, $Re = 1000$



(a)



(b)

Fig. 10 **a** Training loss distribution of s-PINNs for the Taylor vortex problem. **b** Training loss distribution of the improved PINNs for the Taylor vortex problem

Table 3 Relative discrete l^2-error comparison between the predicted and the exact solution obtained using different methods for the Kovasznay flow

Method	Layers	Neurons	ε_u	ε_v	ε_p	Total loss
Standard PINNs	4	50	1.3605e−03	1.3789e−03	3.4027e−03	2.3092e−05
Adaptive weighting (aw)	4	50	1.9397e−04	2.1026e−04	3.8006e−04	5.5799e−07
Improved network (in)	4	50	7.3521e−04	7.6805e−04	6.5432e−04	6.5799e−07
Mixed differentiation (md)	4	50	3.9876e−04	4.3938e−04	4.6573e−04	5.5799e−07
aw-in-PINNs	4	50	6.5724e−05	7.6401e−05	1.3098e−04	8.6544e−08
aw-in-md-PINNs	4	50	1.7379e−05	1.3922e−05	3.1526e−05	7.0582e−08

the L-shaped domain of $\Omega = \left(-\frac{1}{2}, \frac{3}{2}\right) \times (0, 2) \backslash \left(-\frac{1}{2}, \frac{1}{2}\right) \times (0, 1)$ is considered. A deep neural network architecture with 4 hidden layers of 50 neurons each was used. The 500 equidistantly spaced collocation points of the boundaries, and the 2601 equidistantly spaced collocation points are distributed the region. In Table 3, the relative discrete l^2-error and the comparison between the s-PINNs and the results of this paper are shown. The l^2-error of the s-PINNs can only reach the order of 10^{-3}, while our method can reach the order of 10^{-5} and predict a more accurate solution. Figure 11 shows a boxplot of the loss drops for the s-PINNs and the improved PINNs over the entire domain. It can be seen that the loss function of this method decreases fast and converges easily. Figure 12 presents a comprehensive comparison among the prediction solution, the exact solution, and the absolute pointwise error of Kovasznay flow across the entire domain. It can be seen that our method has greatly improved the accuracy of velocity and pressure, and the convergence is faster than s-PINNs.

4.3 Three-dimensional Beltrami flow

The unsteady Beltrami flow [47] is also considered, and the analytic solution is given as follows:

$$\begin{aligned}
u(\mathbf{x}, t) &= -a\left[e^{ax}\sin(ay+dz) + e^{az}\cos(ax+dy)\right]e^{-d^2t},\\
v(\mathbf{x}, t) &= -a\left[e^{ay}\sin(az+dx) + e^{ax}\cos(ay+dz)\right]e^{-d^2t},\\
w(\mathbf{x}, t) &= -a\left[e^{az}\sin(ax+dy) + e^{ay}\cos(az+dx)\right]e^{-d^2t},\\
p(\mathbf{x}, t) &= -\frac{1}{2}a^2\Big[e^{2ax} + e^{2ay} + e^{2az}\\
&\quad +2\sin(ax+dy)\cos(az+dx)e^{a(y+z)}\\
&\quad +2\sin(ay+dz)\cos(ax+dy)e^{a(z+x)}\\
&\quad +2\sin(az+dx)\cos(ay+dz)e^{a(x+y)}\Big]e^{-2d^2t}.
\end{aligned} \tag{4.4}$$

where the parameters $a = d = 1$. The computational domain is $[-1, 1] \times [-1, 1] \times [-1, 1]$, $t \in [0, 1]$. The

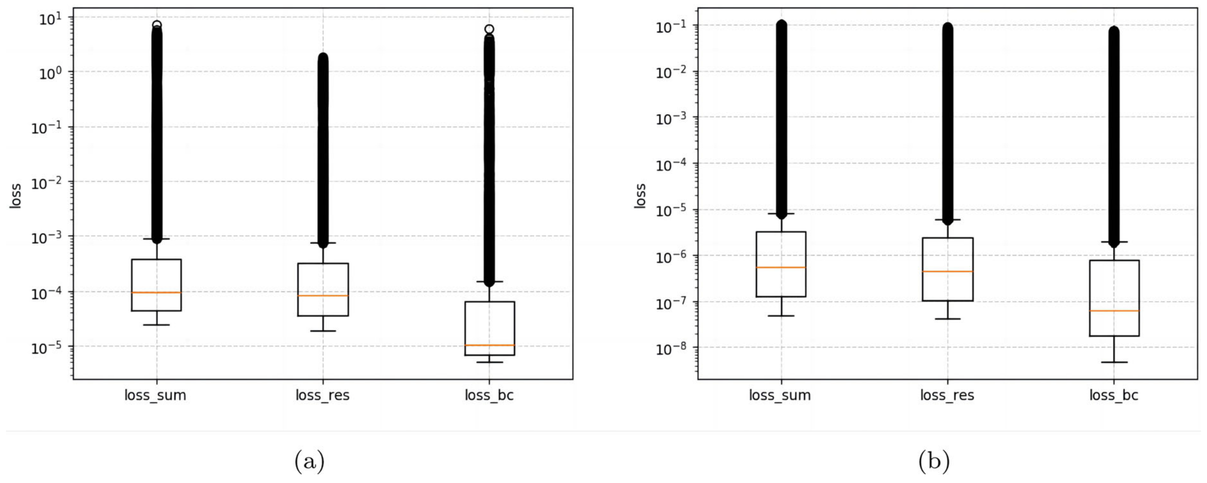


Fig. 11 **a** Training loss distribution of s-PINNs for the Kovasznay flow. **b** Training loss distribution of the improved PINNs for the Kovasznay flow

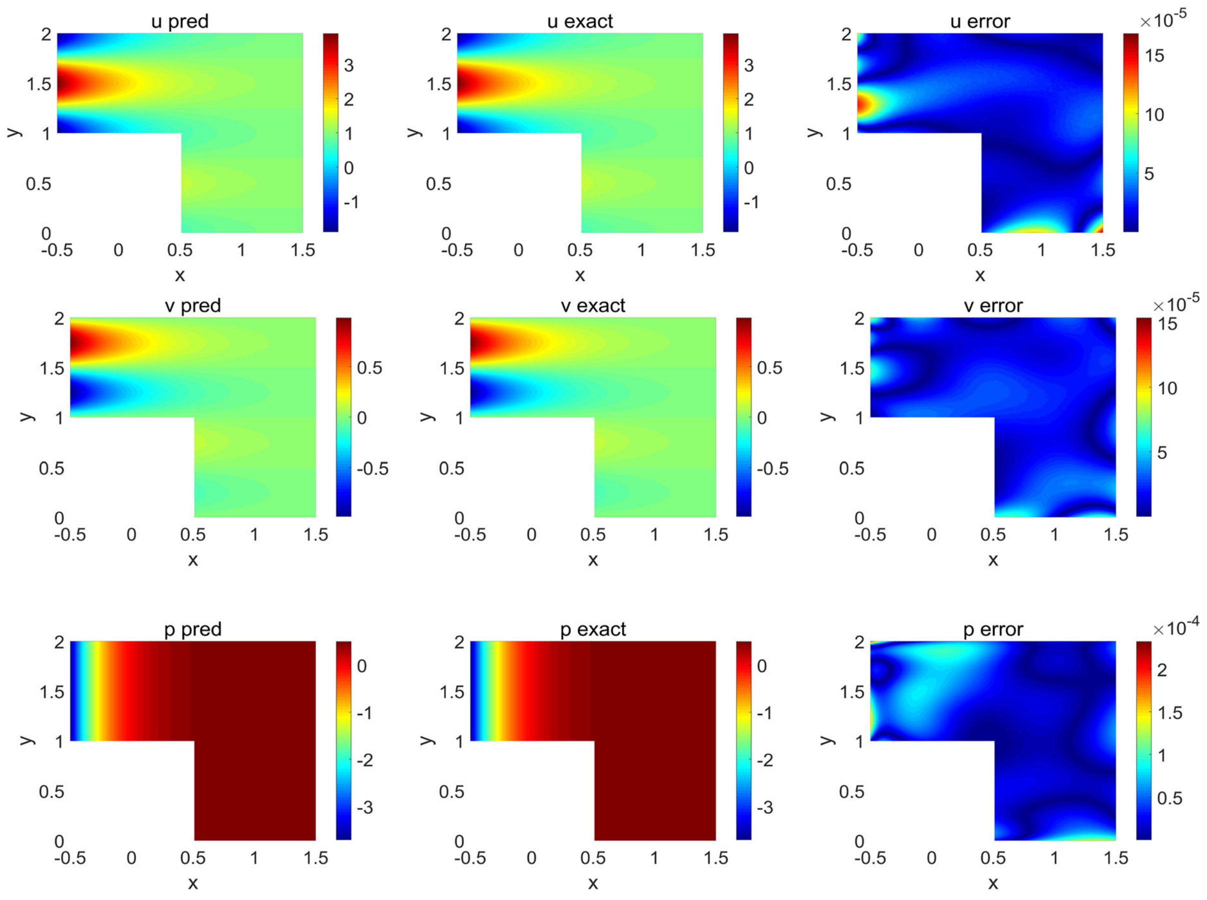
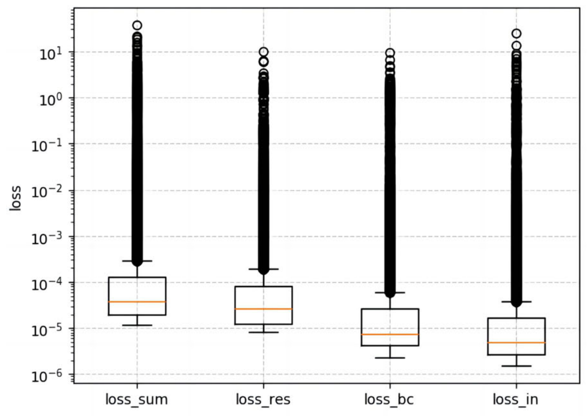
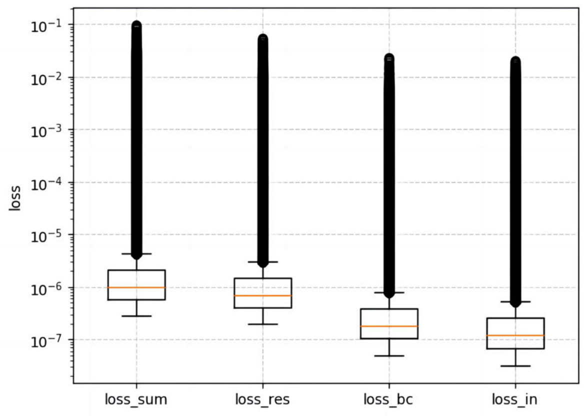


Fig. 12 Distribution of velocity and pressure prediction solution, exact solution, and absolute pointwise error for the Kovasznay flow



(a)



(b)

Fig. 13 **a** Training loss distribution of s-PINNs for the Beltrami flow. **b** Training loss distribution of the improved PINNs for the Beltrami flow

initial condition for PINNs training is the flow field at the initial moment. The unsteady Beltrami flow field is solved at a time step set of 0.05 s. In this case, we used a neural network with 5 hidden layers 50 neurons per layer to simulate the dynamic behavior of this problem. The 10000 residual training points are distributed in the spatio-temporal domain for the equation, and the 800 boundary training points, and the 1000 initial training points are also adopted. In Table 4, the relative discrete l^2-errors of the velocity and pressure predictions obtained by the different methods are shown. It is clear that the relative discrete l^2-errors of our method can reach the order of 10^{-5}. The final adaptive error means and standard deviations are shown in Fig. 13. Figure 14 illustrates the comparison of the predicted velocity components u, v, and w on the $z = 0$ plane at $t = 1$ and their exact solutions, as well as the absolute pointwise error. The absolute pointwise error of the velocity given in [33] reaches the maximum of order of 10^{-3}, while our method is in the order of 10^{-5}. According to the results shown in Fig. 15, our method shows a high accuracy for the prediction solution of the three-dimensional Beltrami flow pressure.

4.4 Flow in a Lid-driven cavity

The classical steady flow in a two-dimensional lid-driven cavity flow in computational fluid dynamics [48] is also simulated with our method. In this case, $Re = 100$. Our goal is to train a neural network in the domain $[0, 1] \times [0, 1]$, and apply the no-slip boundary conditions at the left, lower, and right boundaries. The top boundary moves at a constant speed in the positive x-direction. We have constructed a neural network with 5 hidden layers, each contains 50 neurons, for predicting the potential velocity and pressure fields. We use 5000 residual points and 1000 upper bound training points to simulate fluid flow. The average relative discrete relative l^2-error for five independent runs $|\mathbf{u}(\mathbf{x})| = \sqrt{u_1^2(x) + u_2^2(x)}$ are summarized in Table 5. In Figs. 16, 17 and 18, the experimental results of our neural network prediction are shown. The predictions of our neural network model are consistent with the results in [48]. The predicted velocity field is in good agreement with the reference solution, whereas the s-PINNs fail to produce a reasonable prediction.

4.5 Cylinder wake

Cylinder wake is a classical fluid dynamics problem involving the flow and turbulence patterns of a fluid after it flows through a cylinder. This phenomenon has aroused great interest in many engineering applications, such as wind energy, ocean engineering, and aerospace. We employ the method proposed in this article to simulate the Cylinder wake, Re = 3900 [49]. The size of the simulation domain is $[0, 4] \times [-1.5, 1.5]$. In this test case, a neural network with 7 hidden layers is utilized to predict Cylinder wake. Each hidden

Table 4 Relative discrete l^2-error comparison between the predicted and the exact solution obtained using different methods for the Beltrami flow

Method	ε_u	ε_v	ε_w	ε_p	Total loss
Standard PINNs	1.3697e−03	1.1152e−03	1.0262e−03	1.3925e−03	9.6812e−05
Adaptive weighting (aw)	4.5839e−04	5.9397e−04	3.5867e−04	3.8006e−04	5.5799e−06
Improved network (in)	6.2583e−04	7.1523e−04	5.6923e−04	6.3104e−04	7.2798e−06
Mixed differentiation (md)	5.4327e−04	4.5069e−04	4.6672e−04	4.1526e−04	6.9213e−06
aw-in-PINNs	6.2396e−05	8.6721e−05	9.6401e−05	6.6609e−05	9.2089e−07
aw-in-md-PINNs	2.6366e−05	4.3535e−05	2.7418e−05	3.1624e−05	3.1481e−07

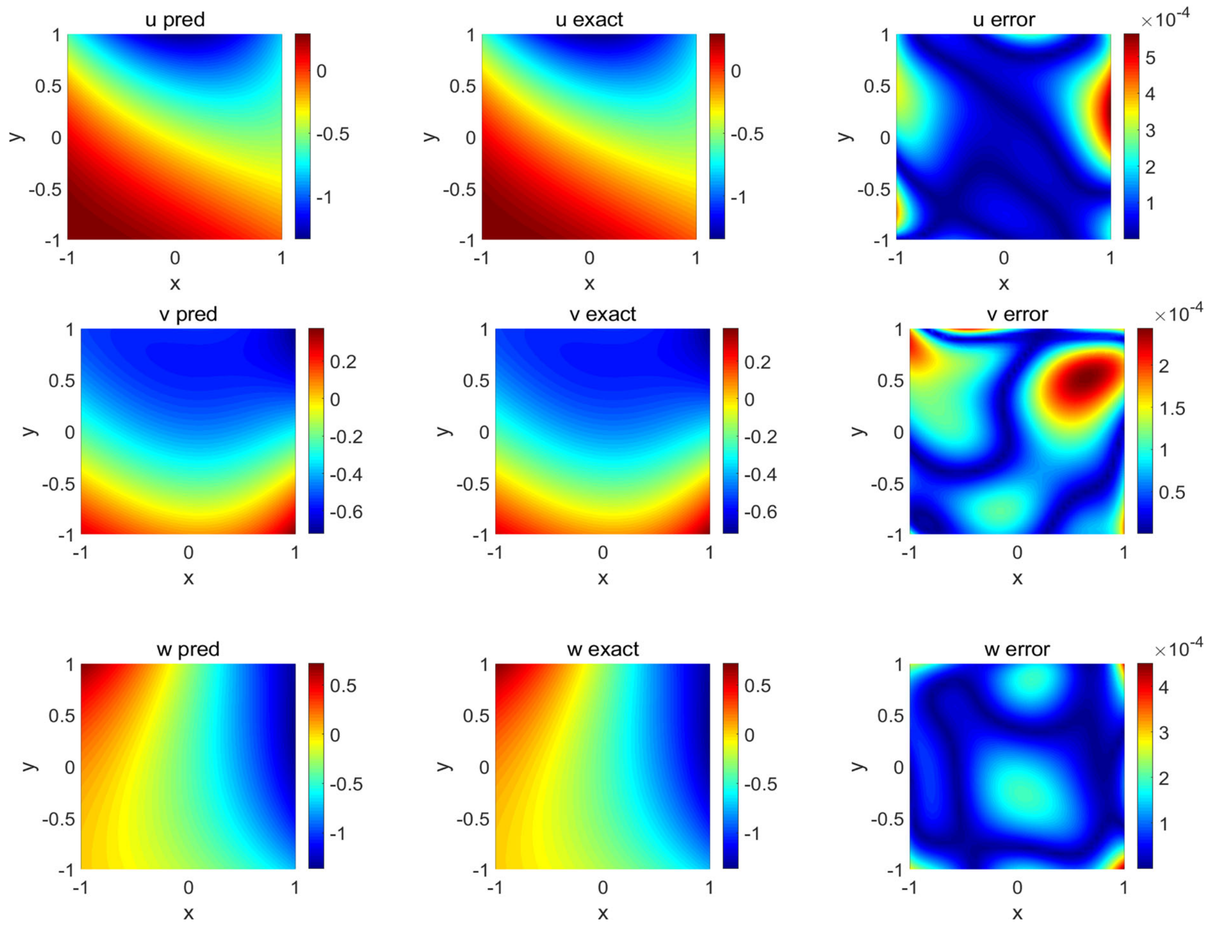


Fig. 14 The distribution of exact velocities u, v, and w, as well as prediction solution for a three-dimensional Beltrami flow in the $z = 0$ plane at a snapshot in time $t = 1$

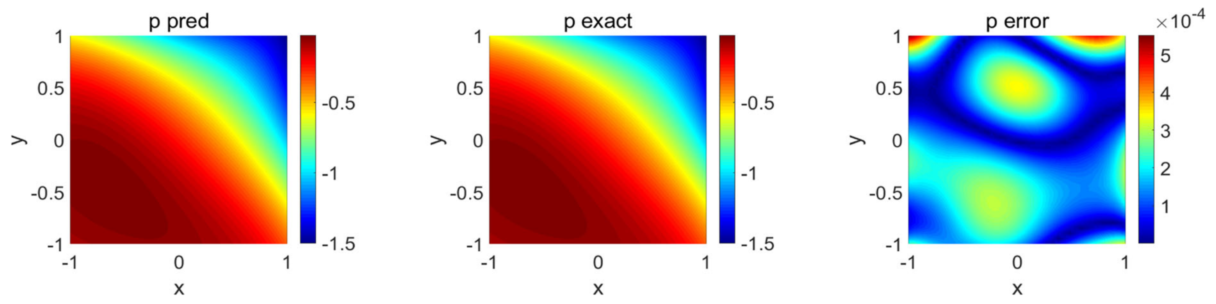


Fig. 15 The distribution of three-dimensional Beltrami flow pressure of the prediction solution, exact solution, and absolute pointwise error

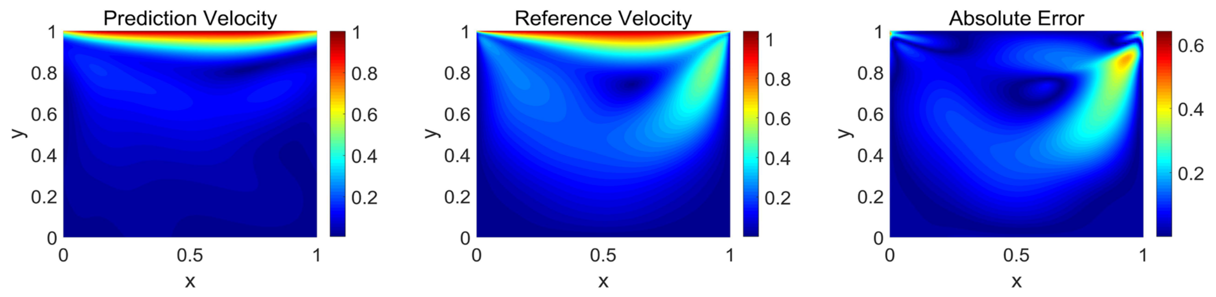


Fig. 16 s-PINNs shows the velocity prediction solution, reference solution, and absolute point error of Flow in a Lid-driven cavity

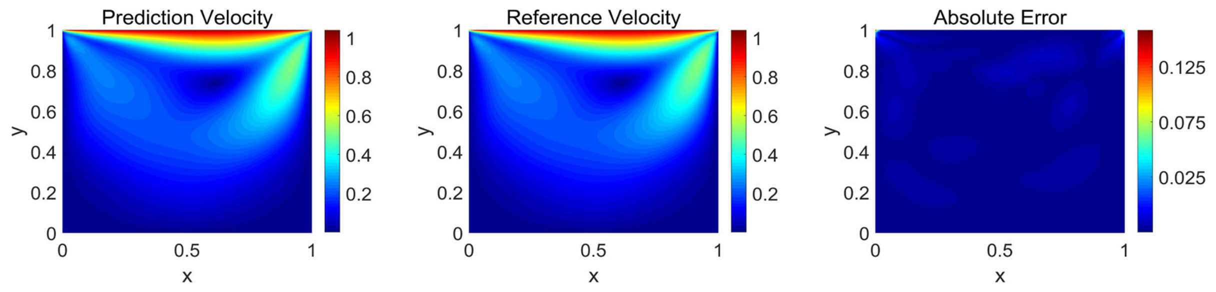


Fig. 17 Our method shows the velocity prediction solution, reference solution, and absolute point error of flow in a Lid-driven cavity

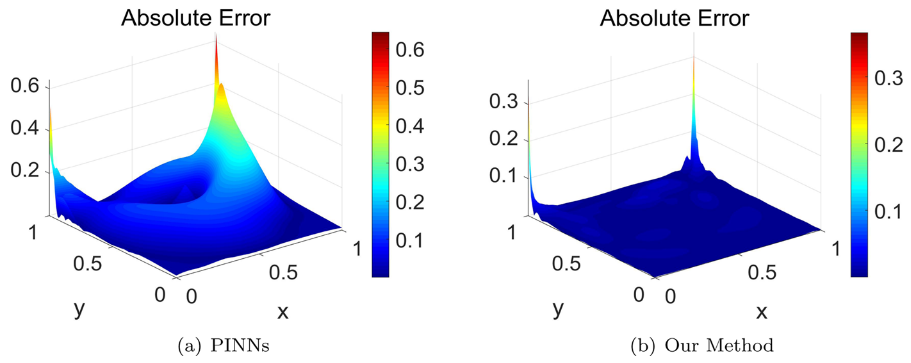


Fig. 18 Absolute pointwise error between the prediction solution and reference solution in 3D of the s-PINNs and our method for flow in a Lid-driven cavity

Table 5 Relative discrete l^2-error comparison between the predicted and the reference solution obtained using different methods for flow in a Lid-driven cavity

Method	Layers	Neurons	Activation function	l^2-error
Standard PINNs	5	50	*tanh*	5.3651e−01
Adaptive weighting (aw)	5	50	*tanh*	2.5031e−01
Improved network (in)	5	50	*tanh*	1.6305e−01
Mixed differentiation (md)	5	50	*tanh*	5.6895e−02
aw-in-md-PINNs	5	50	*tanh*	2.3267e−02

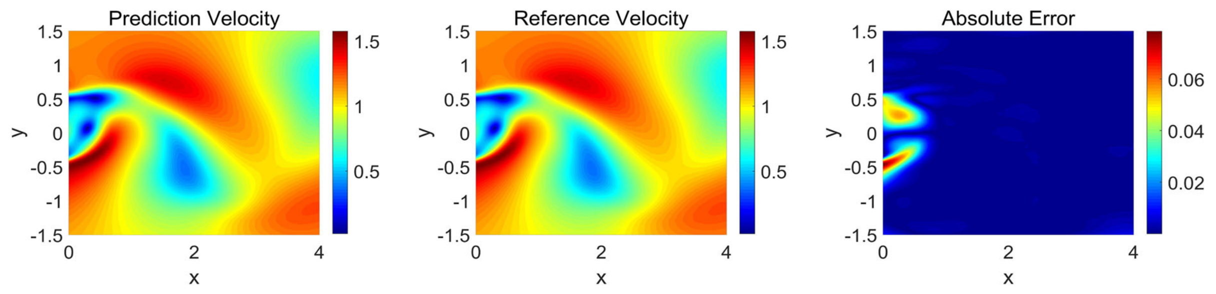


Fig. 19 Distribution of velocity prediction solution, reference solution, and absolute pointwise error for cylinder wake

Table 6 Relative discrete l^2-error comparison between the predicted and the reference solution obtained using different methods for cylinder wake

Method	Layers	Neurons	Activation function	l^2-error
Standard PINNs	7	100	*tanh*	7.8631e−01
Adaptive weighting (aw)	7	100	*tanh*	3.5781e−01
Improved network (in)	7	100	*tanh*	2.9854e−01
Mixed differentiation (md)	7	100	*tanh*	6.9056e−02
aw-in-md-PINNs	7	100	*tanh*	3.7267e−02

layer consists of 100 neurons. The training dataset for this problem contains 10000 boundary training points, 5000 initial training points, and 40000 residual training points. Table 6 summarizes the results of the average relative discrete l^2 -error for $|\mathbf{u}| = \sqrt{u_1^2 + u_2^2}$ across five independent runs. It is evident that our method significantly enhances the accuracy of the prediction. Figure 19 shows the distribution of the prediction solution, reference solution, and absolute pointwise error in the computational domain. It can be seen that our method produces accurate predictions with small errors, which reflects the excellent performance of the method in predicting fluid mechanics problems.

4.6 Inverse problem of the wake flow of a circular cylinder

We simulated the two-dimensional vortex shedding behind a cylinder of diameter 1 under Re = 100, $\Omega = [0, 8] \times [-2, -2]$ [19]. In this case, $\lambda_1 = 1.0, \lambda_2 = 0.01$. We take 10000 residual points and 3000 initial boundary points as the training dataset. Our goal is to determine the values of the unknown parameters λ_1, λ_2, and obtain a reasonably accurate reconstruction of the velocity $u(\mathbf{x}, t)$ and $v(\mathbf{x}, t)$ in the wake flow of a circular cylinder. We consider the unsteady incompressible N–S equations in two dimensions as

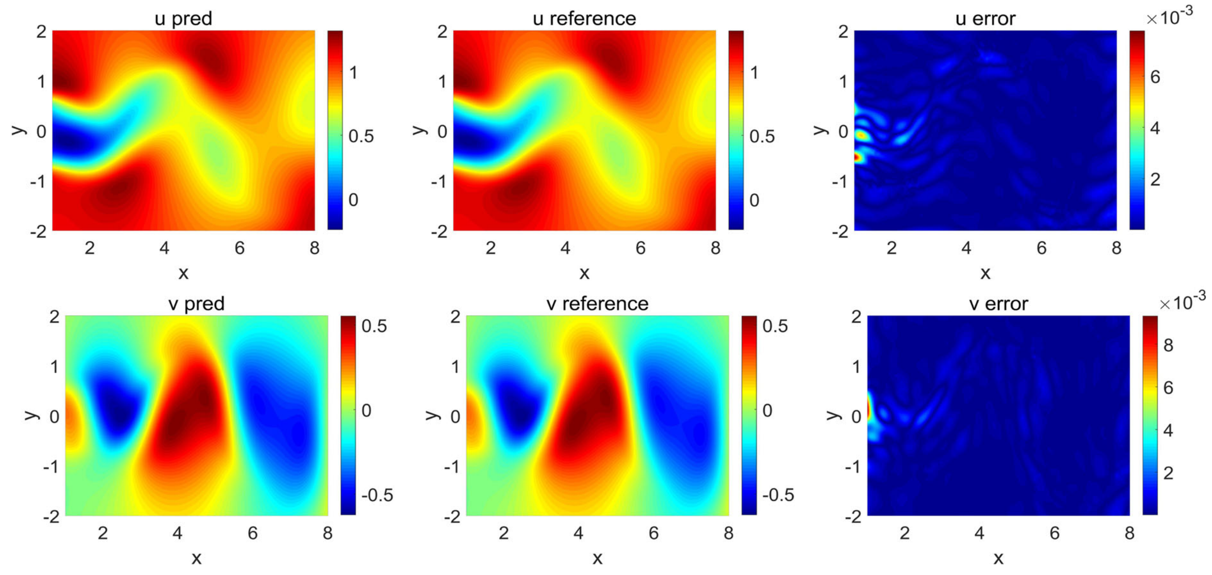


Fig. 20 Distribution of velocity prediction solution and reference solution of the wake flow of a circular cylinder for a snapshot in time t = 10

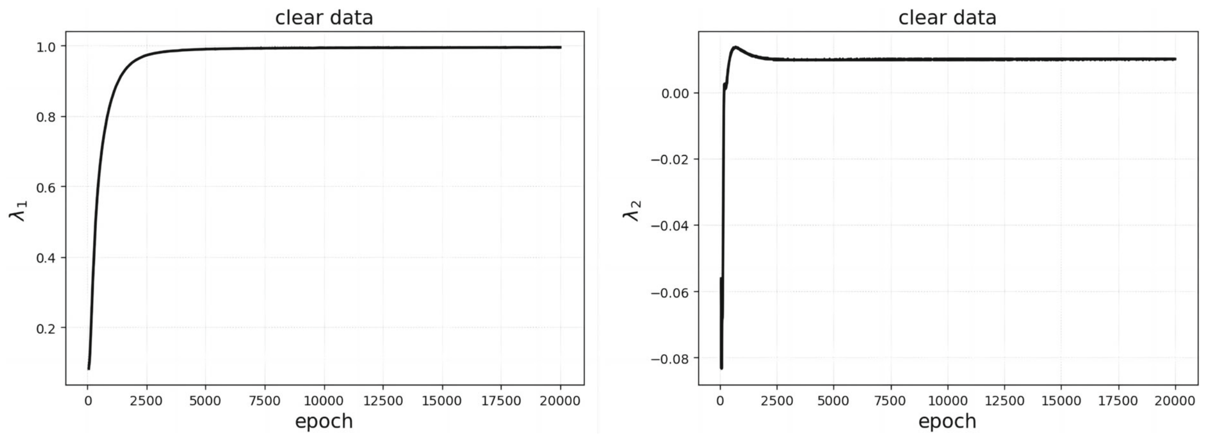


Fig. 21 Iterative training curves for the parameters λ_1 and λ_2

Table 7 Relative discrete l^2-error comparison between the predicted and the reference solution obtained using different methods for the wake flow of a circular cylinder

Method	l^2-error u	l^2-error v	λ_1(%)	λ_2(%)
Standard PINNs	8.103e−03	6.378e−03	0.99	2.30
Adaptive weighting (aw)	1.158e−03	2.563e−03	0.73	1.46
Improved network (in)	2.591e−03	3.156e−03	0.69	1.53
Mixed differentiation (md)	1.301e−03	1.905e−03	0.58	1.29
aw-in-md-PINNs (clean)	3.285e−04	4.624e−04	0.05	0.85
aw-in-md-PINNs (1%noise)	5.534e−04	6.102e−04	0.20	1.90

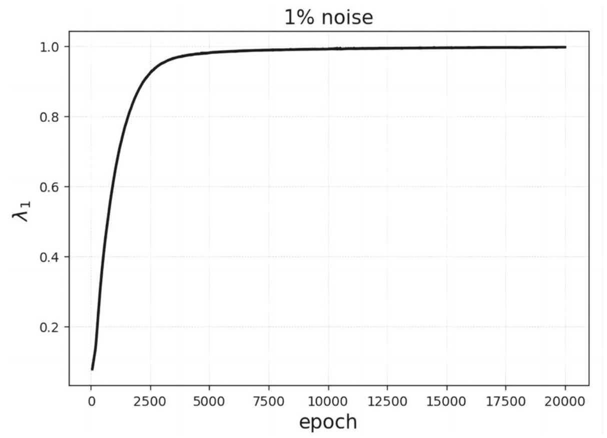


Fig. 22 Iterative training curves for the parameters λ_1 and λ_2

follows.

$$\begin{aligned} u_t + \lambda_1 \left(uu_x + vu_y\right) &= \lambda_2 \left(u_{xx} + u_{yy}\right) - p_x, \\ v_t + \lambda_1 \left(uv_x + vv_y\right) &= \lambda_2 \left(v_{xx} + v_{yy}\right) - p_y, \\ u_x + v_y &= 0. \end{aligned} \tag{4.5}$$

The architecture of the neural network is constructed using 4 hidden layers, each consists of 50 neurons. The numerical results of the wake flow of a circular cylinder are presented in Table 7. Figure 20 shows the representative snapshots of the velocity components $u(\mathbf{x}, t)$, $v(\mathbf{x}, t)$ predicted by the training model. The results show that the error between the predicted and the true values is very small in the whole calculation domain. This method can accurately simulate the complex flow phenomena in the wake flow of a circular cylinder. The predicted $\lambda_1 = 1.00043$, $\lambda_2 = 0.00996$. Figure 21 shows the curves of the predicted parameters λ_1 and λ_2, and the neural network can accurately infer the unknown parameters in the equation. As shown in Fig. 22, the method can still accurately identify the unknown parameters λ_1 and λ_2 after the training data is corrupted by noise of 1%. This shows that the model performs well in dealing with complex problems, with a high degree of accuracy and reliability. Our adopted method accurately solves both the forward and inverse problems of circular tail traces. This method can also be applied to address corresponding challenges in other fields.

5 Conclusions

In this paper, we proposed a novel adaptive PINNs method to solve the incompressible N–S equations. The significant impact of the initial and boundary conditions on the accuracy of the problem has been studied. Secondly, the loss functions of PINNs are composed of a weighted combination of the PDEs and the initial boundary values. The weighted combination of the loss functions can easily affect the performance and convergence of the networks. Therefore, we proposed an adaptive weighting PINNs, which adaptively assigns the weights of the loss functions based on the maximum likelihood estimation of Gaussian distribution. he performance and effectiveness of the network are further enhanced, utilizing the physical equation constraints and initial boundary information to improve the accuracy of the solution. In addition, an improved neural network architecture has been designed simultaneously using the global and local information, which is good for passing the input information to the hidden layers while maintaining the integrity. As shown in the numerical experiments, this network can grasp the part of the N–S equations solution with drastic changes. The derivative term in the loss function is then approximated using the mixed differentiation, i.e., AD and the local support points. We adopt the upwind and central difference schemes to compute the derivatives of the convective and pressure-gradient terms in the N–S equations. By combining the advantages of both AD and ND, the sampling efficiency, the convergence speed, and the accuracy are improved. In future work, we will further explore how to apply the proposed model to higher dimensionality and complex geometries.

Author contributions Jie Wang: investigation, software, validation, writing—original draft. Xufeng Xiao: conceptualization, methodology, formal analysis, software, writing—review and editing. Xinlong Feng: formal analysis, writing—review and editing. Hui Xu: formal analysis, writing—review and editing.

Funding This work is supported by the National Natural Science Foundation of China (Nos. 12361090, 12071406 and 12271465), the Natural Science Foundation of Xinjiang Province (Nos. 2023D01C164, 6142A05230203, 2022D01D32), and the Tianshan Talent Training Program (No. 2023TSYCQNTJ0015 and No. 2022TSYCTD0019).

Data availability The code can be found at https://github.com/upwaj/aw-in-md-PINNs.

Declarations

Conflict of interest No conflict of interest exists in the submission of this manuscript, and the manuscript is approved by all authors for publication.

References

1. Girault, V., Raviart, P.-A.: Finite element methods for Navier–Stokes equations: theory and algorithms, vol. 5. Springer Science & Business Media, Cham (2012)
2. Coelho, P.J., Pereira, J.C.F.: Finite volume computation of the turbulent flow over a hill employing 2D or 3D non-orthogonal collocated grid systems. Int. J. Numer. Methods Fluids **14**(4), 423–441 (1992)
3. Weinan, E., Liu, J.-G.: Finite difference methods for 3D viscous incompressible flows in the vorticity-vector potential formulation on nonstaggered grids. J. Comput. Phys. **138**(1), 57–82 (1997)
4. Ranade, R., Hill, C., Pathak, J.: DiscretizationNet: a machine-learning based solver for Navier–Stokes equations using finite volume discretization. Comput. Methods Appl. Mech. Eng. **378**, 113722 (2021)
5. Wang, Y., Lai, C.-Y.: Multi-stage neural networks: function approximator of machine precision. J. Comput. Phys. **504**, 112865 (2024)
6. Li, X., Liu, Y., Liu, Z.: Physics-informed neural network based on a new adaptive gradient descent algorithm for solving partial differential equations of flow problems. Phys. Fluids **35**(6), 063608 (2023)
7. Eivazi, H., Tahani, M., Schlatter, P., Vinuesa, R.: Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations. Phys. Fluids **34**(7), 075117 (2022)
8. Xiao, M.-J., Teng-Chao, Y., Zhang, Y.-S., Yong, H.: Physics-informed neural networks for the Reynolds-averaged Navier–Stokes modeling of Rayleigh–Taylor turbulent mixing. Comput. Fluids **266**, 106025 (2023)
9. Azizzadenesheli, K., Kovachki, N., Li, Z., Liu-Schiaffini, M., Kossaifi, J., Anandkumar, A.: Neural operators for accelerating scientific simulations and design. Nat. Rev. Phys. **6**, 1–9 (2024)
10. Lee, H., Kang, I.S.: Neural algorithm for solving differential equations. J. Comput. Phys. **91**(1), 110–131 (1990)
11. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neural Netw. **9**(5), 987–1000 (1998)
12. Weinan, E., Bing, Y.: The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. **6**, 1–12 (2018)
13. Chen, M., Niu, R., Zheng, W.: Adaptive multi-scale neural network with resnet blocks for solving partial differential equations. Nonlinear Dyn. **111**, 6499–6518 (2022)
14. Gao, R., Wei, H., Fei, J., Hongyu, W.: Boussinesq equation solved by the physics-informed neural networks. Nonlinear Dyn. **111**, 15279–15291 (2023)
15. Zhang, T., Hui, X., Guo, L., Feng, X.: A non-intrusive neural network model order reduction algorithm for parameterized parabolic PDEs. Comput. Math. Appl. **119**, 59–67 (2022)
16. Hou, J., Li, Y., Ying, S.: Enhancing PINNs for solving PDEs via adaptive collocation point movement and adaptive loss weighting. Nonlinear Dyn. **111**, 15233–15261 (2023)
17. Wang, H., Zou, B., Jian, S., Wang, D.: Variational methods and deep Ritz method for active elastic solids. Soft Matter **18**(7), 6015–6031 (2022)
18. Gao, Z., Yan, L., Zhou, T.: Failure-informed adaptive sampling for PINNs. SIAM J. Sci. Comput. **45**, A1971–A1994 (2023)
19. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
20. Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics-informed machine learning. Nat. Rev. Phys. **3**(6), 422–440 (2021)
21. Lu, L., Meng, X., Mao, Z., Karniadakis, G.E.: DeepXDE: a deep learning library for solving differential equations. SIAM Rev. **63**(1), 208–228 (2021)
22. Chiu, P.-H., Wong, J.C., Ooi, C., Dao, M.H., Ong, Y.-S.: CAN-PINN: a fast physics-informed neural network based on coupled-automatic-numerical differentiation method. Comput. Methods Appl. Mech. Eng. **395**, 114909 (2022)
23. Patel, R.G., Manickam, I., Trask, N.A., Wood, M.A., Lee, M., Tomas, Ignacio, Cyr, Eric C.: Thermodynamically consistent physics-informed neural networks for hyperbolic systems. J. Comput. Phys. **449**, 110754 (2022)
24. Jagtap, A.D., Kharazmi, E., Karniadakis, G.E.: Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. Comput. Methods Appl. Mech. Eng. **365**, 113028 (2020)
25. Jagtap, A.D., Karniadakis, G.E.: 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**, 2002–2041 (2020)

26. Wei, W., Feng, X., Hui, X.: Improved deep neural networks with domain decomposition in solving partial differential equations. J. Sci. Comput. **93**, 1–34 (2022)
27. Wang, S., Sankaran, S., Wang, H., Perdikaris, P.: An expert's guide to training physics-informed neural networks (2023). arXiv preprint: arXiv: 2308.08468
28. McClenny, L.D., Braga-Neto, U.M.: Self-adaptive physics-informed neural networks. J. Comput. Phys. **474**, 111722 (2023)
29. Tang, S., Feng, X., Wei, W., Hui, X.: Physics-informed neural networks combined with polynomial interpolation to solve nonlinear partial differential equations. Comput. Math. Appl. **132**, 48–62 (2023)
30. Peng, P., Pan, J., Hui, X., Feng, X.: RPINNs: rectified-physics informed neural networks for solving stationary partial differential equations. Comput. Fluids **245**, 105583 (2022)
31. Chenxi, W., Zhu, M., Tan, Q., Kartha, Y., Lu, L.: A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. Comput. Methods Appl. Mech. Eng. **403**, 115671 (2023)
32. Bai, Y., Chaolu, T., Bilige, S.: Solving Huxley equation using an improved PINN method. Nonlinear Dyn. **105**, 3439–3450 (2021)
33. Jin, X., Cai, S., Li, H., Karniadakis, G.E.: NSFnets (Navier–Stokes flow nets): physics-informed neural networks for the incompressible Navier–Stokes equations. J. Comput. Phys (2021). https://doi.org/10.1016/j.jcp.2020.109951
34. Rao, C., Sun, H., Liu, Y.: Physics-informed deep learning for incompressible laminar flows. Theor. Appl. Mech. Lett. **10**(3), 207–212 (2020)
35. Dwivedi, B.S.V., Parashar, N.: Distributed learning machines for solving forward and inverse problems in partial differential equations. Neurocomputing **420**, 299–316 (2021)
36. Gao, H., Sun, L., Wang, J.-X.: PhyGeoNet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. J. Comput. Phys. **428**, 110079 (2021)
37. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Adv. Neural Inf. Process. Syst. **33**, 7462–7473 (2020)
38. Mattey, R., Ghosh, S.: A novel sequential method to train physics informed neural networks for Allen–Cahn and Cahn–Hilliard equations. Comput. Methods Appl. Mech. Eng. **390**, 114474 (2022)
39. Stein, M.: Large sample properties of simulations using Latin hypercube sampling. Technometrics **29**(2), 143–151 (1987)
40. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Program. **45**(1–3), 503–528 (1989)
41. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7482–7491 (2018)
42. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł, Polosukhin, I.: Attention is all you need. Adv. Neural Inf. Process. Syst. **30**, 1 (2017)
43. Cho, K., Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Conference on Empirical Methods in Natural Language Processing (EMNLP 2014) (2014)
44. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M.: Tensorflow: a system for large-scale machine learning on heterogeneous distributed systems (2016). arXiv: 1603.04467
45. Xiao, F., Akoh, R., Ii, S.: Unified formulation for compressible and incompressible flows by using multi-integrated moments ii: multi-dimensional version for compressible and incompressible flows. J. Comput. Phys. **213**(1), 31–56 (2006)
46. Sheu, T.W.H., Chiu, P.H.: A divergence-free-condition compensated method for incompressible Navier–Stokes equations. Comput. Methods Appl. Mech. Eng. **196**(45–48), 4479–4494 (2007)
47. Ethier, C.R., Steinman, D.A.: Exact fully 3D Navier–Stokes solutions for benchmarking. Int. J. Numer. Methods Fluids **19**(5), 369–375 (1994)
48. Ghia, U.K.N.G., Ghia, K.N., Shin, C.T.: High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method. J. Comput. Phys. **48**(3), 387–411 (1982)
49. Williamson, C.H.K.: Vortex dynamics in the cylinder wake. Ann. Rev. Fluid Mech. **28**(1), 477–539 (1996)