Physics of Fluids

RESEARCH ARTICLE | JUNE 28 2023

Spatiotemporal parallel physics-informed neural networks: A framework to solve inverse problems in fluid mechanics

Xu Shengfeng (许盛峰) ; Yan Chang (闫畅) ; Zhang Guangtao (张广涛) ; Sun Zhenxu (孙振旭) ; Huang Renfang (黄仁芳) ; Ju Shengjun (鞠胜军) ; Guo Dilong (郭迪龙) ; Yang Guowei (杨国伟)

Physics of Fluids 35, 065141 (2023)
https://doi.org/10.1063/5.0155087

Physics of Fluids ARTICLE pubs.aip.org/aip/pof

Spatiotemporal parallel physics-informed neural networks: A framework to solve inverse problems in fluid mechanics

Cite as: Phys. Fluids **35**, 065141 (2023); doi: 10.1063/5.0155087
Submitted: 17 April 2023 · Accepted: 9 June 2023 ·
Published Online: 28 June 2023

View Online Export Citation CrossMark

Shengfeng Xu (许盛峰),[1,2] Chang Yan (闫畅),[1,3] Guangtao Zhang (张广涛),[4,5] Zhenxu Sun (孙振旭),[1,a)] Renfang Huang (黄仁芳),[1] Shengjun Ju (鞠胜军),[1] Dilong Guo (郭迪龙),[1,2] and Guowei Yang (杨国伟)[1,2]

AFFILIATIONS

[1]Key Laboratory for Mechanics in Fluid Solid Coupling Systems, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China
[2]School of Engineering Science, University of Chinese Academy of Sciences, Beijing 100049, China
[3]School of Future Technology, University of Chinese Academy of Sciences, Beijing 100049, China
[4]SandGold AI Research, Guangzhou 510642, China
[5]Department of Mathematics, Faculty of Science and Technology, University of Macau, Macau 519000, China

[a)]**Author to whom correspondence should be addressed:** sunzhenxu@imech.ac.cn

ABSTRACT

Physics-informed neural networks (PINNs) are widely used to solve forward and inverse problems in fluid mechanics. However, the current PINNs framework faces notable challenges when presented with problems that involve large spatiotemporal domains or high Reynolds numbers, leading to hyper-parameter tuning difficulties and excessively long training times. To overcome these issues and enhance PINNs' efficacy in solving inverse problems, this paper proposes a spatiotemporal parallel physics-informed neural networks (STPINNs) framework that can be deployed simultaneously to multi-central processing units. The STPINNs framework is specially designed for the inverse problems of fluid mechanics by utilizing an overlapping domain decomposition strategy and incorporating Reynolds-averaged Navier–Stokes equations, with eddy viscosity in the output layer of neural networks. The performance of the proposed STPINNs is evaluated on three turbulent cases: the wake flow of a two-dimensional cylinder, homogeneous isotropic decaying turbulence, and the average wake flow of a three-dimensional cylinder. All three turbulent flow cases are successfully reconstructed with sparse observations. The quantitative results along with strong and weak scaling analyses demonstrate that STPINNs can accurately and efficiently solve turbulent flows with comparatively high Reynolds numbers.

I. INTRODUCTION

Physics-informed neural networks (PINNs) have gained significant attention in scientific literature and research as a promising method for solving partial differential equations (PDEs).[1,2] Introduced by Raissi in 2019,[3] this method provides a framework to solve forward and inverse problems involving nonlinear PDEs via the popular machine learning framework named TensorFlow. PINNs offer a distinct approach to scientific machine learning by incorporating both physical principles and data-driven information through automatic differentiation.[4] PINNs have been used in various fields such as heat transfer,[5,6] climate science,[7,8] quantum physics,[9,10] solid mechanics,[11,12] and fluid mechanics.[13,14]

In the field of fluid mechanics, extensive research has been conducted in both forward and inverse problems.[15–21] Forward problems seek to solve fluid flows by providing correct initial and boundary conditions, while inverse problems aim to derive unknown flow features or undetermined parameters through abundant data or sparse data with ill-posed condition. Despite various efforts to enhance the trainability of PINNs, such as adaptive weighting,[22,23] adaptive sampling,[24,25] and adaptive activation functions,[26,27] the prohibitively high computational cost and the rather unpredictable solution precision compared to traditional computational fluid dynamics (CFD) methods make PINNs no match for finite difference and finite element methods in terms of forward problems. Therefore, the primary strength of

applying PINNs in fluid mechanics lies in the simplicity and flexibility in solving inverse problems, such as identifying unknown parameters,[28] inferring flow field from scalar transportation,[20] super-resolution,[17,29] and flow field reconstruction.[16,30,31]

Focusing on the inverse problem of fluid mechanics, normal PINNs might still suffer from the extremely long time spent on calculation and the inability to solve long-duration and large domain problems. To address this problem, the classic "divide and conquer" idea originated from traditional numerical simulation was adopted by the literature[32,33] and developed PINNs with domain decomposition, aiming to assign separate neural networks in different sub-domains and provide parallelization capacity. Two popular decomposition frameworks that resulted from this idea include conservative physics-informed neural networks (cPINNs),[34] which imposed flux and data continuity conditions along sub-domain interfaces to decompose the problem in space, and extended physics-informed neural networks (XPINNs),[35] which enable space–time domain decomposition for any irregular, non-convex geometry through residual and data continuity along interfaces. Parallel physics-informed neural networks[36] were also proposed, where a long-duration problem can be decomposed into independent short-duration problems monitored by an inexpensive coarse-grained (CG) solver. In addition to the normal form of PINNs, domain decomposition was also seen in other variations of solving PDEs. For instance, hp-variational physics-informed neural networks (hp-VPINNs)[37] combined deep neural networks and the sub-domain Petrov–Galerkin method for solving partial differential equations with a focus on domain decomposition. Local extreme learning machines (locELM)[38] also incorporated domain decomposition and demonstrated comparable or superior computational performance when compared to the finite element method. However, the fitting capacity of extreme learning machines is limited, and they are not as effective as PINNs in solving Navier–Stokes (NS) equations.[39] The above-mentioned frameworks all employed a non-overlapping decomposition strategy, which could be improved by adopting an overlapping strategy since the upper and lower boundaries usually present a noticeably higher residual error.[20]

Apart from domain decomposition, other endeavors have been undertaken to modify the loss function to enable physics-informed neural networks (PINNs) to solve problems with higher Reynolds numbers. For instance, Eivazi *et al.*[40] inserted Reynolds-averaged Navier–Stokes (RANS) equations into the loss function and successfully solved incompressible turbulent flows with PINNs. Xu *et al.*[41] introduced an artificial viscosity term in the loss function, which allows the neural network to learn the spatially distributed parameter on its own. Pioch *et al.*[42] compared the performance of different RANS models on a backward-facing step flow problem. These studies all suggest that better alternatives exist in selecting equations for solving fluid mechanics problems with PINNs, particularly for relatively high Reynolds numbers.

In this paper, a spatiotemporal parallel PINNs framework, referred to as STPINNs, is proposed to efficiently solve the problem of flow field reconstruction from sparse observation. Unlike existing frameworks that impose extra constraint on the sub-domain interfaces, the proposed STPINNs only require data continuity condition, making it less computationally demanding. By introducing overlapping domain decomposition strategy and incorporating RANS equations, STPINNs have demonstrated robustness and efficiency in solving two-dimensional (2D) turbulent flow and average three-dimensional (3D) flow. The performance of the proposed framework is evaluated quantitatively. In addition, strong and weak scaling analyses are conducted to investigate the parallel capability, which is similar to traditional parallel numerical algorithms[43,44] and pioneering work in PINNs.[22]

This article is organized as follows: Sec. II provides a detailed description of the proposed STPINNs framework. In Sec. III, three turbulent datasets used in this study are introduced. Section IV demonstrates the application of STPINNs on three distinct problems, and Sec. V concludes the paper with a brief summary and discussion.

II. METHODOLOGY

A. Physics-informed neural network

Physics-informed neural networks (PINNs) were first proposed in the literature[28,45] as a new alternative to traditional mesh-based methods in solving PDEs. As shown in Fig. 1, by rewriting the NS equations into residual form,

$$\mathcal{R}_{continuity} = u_x + v_y, \tag{1}$$

$$\mathcal{R}_x = u_t + uu_x + vu_y + p_x - 1/Re(u_{xx} + u_{yy}), \tag{2}$$

$$\mathcal{R}_y = v_t + uv_x + vv_y + p_y - 1/Re(v_{xx} + v_{yy}), \tag{3}$$

the conservation laws can be imposed in a soft manner. Meanwhile, PINNs reserve the traditional function to fit labeled data in the training set, making it able to switch swiftly and flexibly between forward and inverse problems.

Given a neural network function $\tilde{u}(\mathbf{x}, t; w)$, where w means the undetermined weights and biases in the neural network, the optimization goal of PINNs is to find w that minimizes the loss function

$$MSE = MSE_d + \lambda_r \times MSE_r, \tag{4}$$

where the MSE stands for mean squared error and is given for each term by

$$MSE_d = \frac{1}{N_d}\sum_{i=1}^{N_d} ||\tilde{u}(\mathbf{x}_d^i, t_d^i; w) - u_d^i||^2, \tag{5}$$

$$MSE_r = \frac{1}{N_r}\sum_{i=1}^{Nr} ||\mathbf{R}(\tilde{u}(\mathbf{x}_r^i, t_r^i; w))||^2, \tag{6}$$

where λ_r is a tunable hyper-parameter that controls the weights of prediction deviation and residual error. In recent research,[23] λ_r can also be a dynamic value that evolves with iterations to accelerate convergence. In this paper, λ_r is set to 1 for simplicity. $\{\mathbf{x}_d^i, t_d^i, u_d^i\}_{i=1}^{N_d}$ is the training dataset with N_d data points, and $\{\mathbf{x}_r^i, t_r^i\}_{i=1}^{N_r}$ is the residual set with N_r collocation points. The subscripts d and r represent the data points set and residual (collocation) points set, respectively. $\mathbf{R}(\tilde{u}(\mathbf{x}, t))$ is a vector consisting of $\mathcal{R}_{continuity}$, $\mathcal{R}_x$, and $\mathcal{R}_y$. By minimizing both the value of data derivation and residual error, PINNs can be made possible not only fitting the real data in the training set but also predicting unknown information of the flow field under the guidance of physical principles.

B. NS equations with eddy viscosity

PINNs have already demonstrated relatively stable performance in solving laminar flow in recent literature.[28,46,47] Nevertheless, a

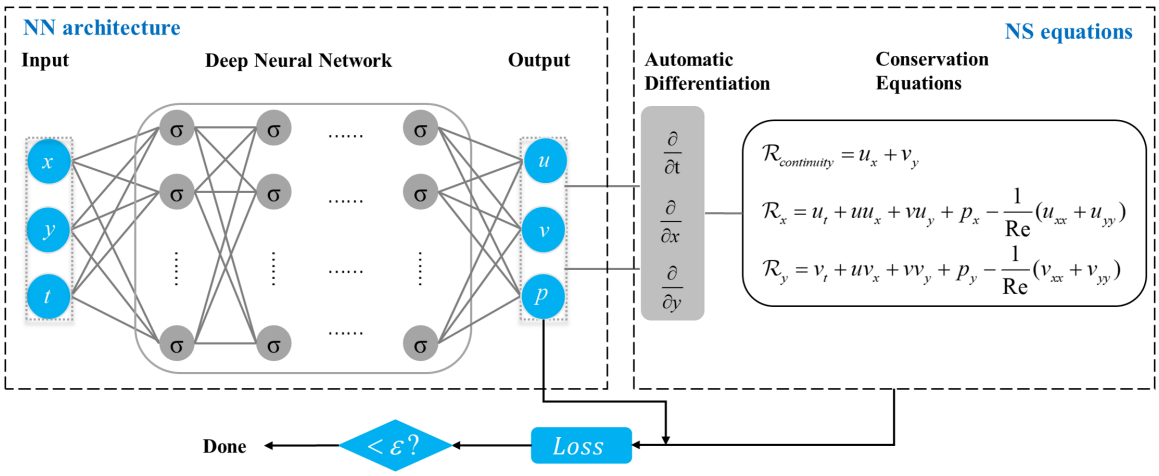


FIG. 1. Physics-informed neural networks for solving incompressible NS equations. The loss function consists of both the prediction deviation and the residual error of the conservation equations. σ denotes the activation function.

common question among researchers is whether PINNs can solve fluid flow with higher Reynolds numbers. In the pioneering works,[40–42] various methods similar to engineering applications have been explored in PINNs to train RANS equations instead of NS equations. The main issue in RANS is to model the Reynolds stress $\tau_{ij} = -\rho\overline{u_i' u_j'}$, where $\overline{(\cdot)}$ denotes the Reynolds average or the spatial filtering and u_i' denotes the velocity fluctuation. While it is possible to train PINNs with the original RANS equations without any specific model or assumption for turbulence[40] due to the special nature of automatic differentiation, it is more promising to model Reynolds stress using eddy viscosity ν_t [as shown in Eq. (7)] given a PINNs framework, since it introduces only one spatial–temporal parameter while preserving the ability to lower the level of difficulty for solving the original NS equations. This approach is originally proposed by Xu *et al.*[41] in exploring missing flow dynamics and successfully validated in both steady and unsteady cases

$$-\overline{u_i' u_j'} = \nu_t \left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i} \right). \tag{7}$$

The neural network architecture for NS equations with eddy viscosity is depicted in Fig. 2. By introducing the non-dimensional parameter ν_t^* in the output layer of the neural network, the optimal distribution of eddy viscosity is automatically fitted during backward propagation of the training process. While there are discernible differences in performance when using different RANS equations as the loss functions in PINNs, depending on whether and how the Reynolds stress is modeled, NS equations with eddy viscosity are simply referred to as RANS equations in the following content for the sake of clarity and reference, since only the original NS equations and NS equations with eddy viscosity are compared in this paper.

C. Overlapping domain decomposition

To mitigate the use of extremely large neural networks for problems with a vast spatiotemporal domain, a practical approach is to partition the overall domain Ω into several smaller sub-domains $\Omega_q (\Omega = \cup_{q=1}^{N_{sub}} \Omega_q)$. Each sub-domain corresponds to a separate sub neural network that represents the local solution $\tilde{u}_q(\mathbf{x}, t; w_q) | (\mathbf{x}, t) \in \Omega_q$ of the NS equations. To ensure connectivity between adjacent

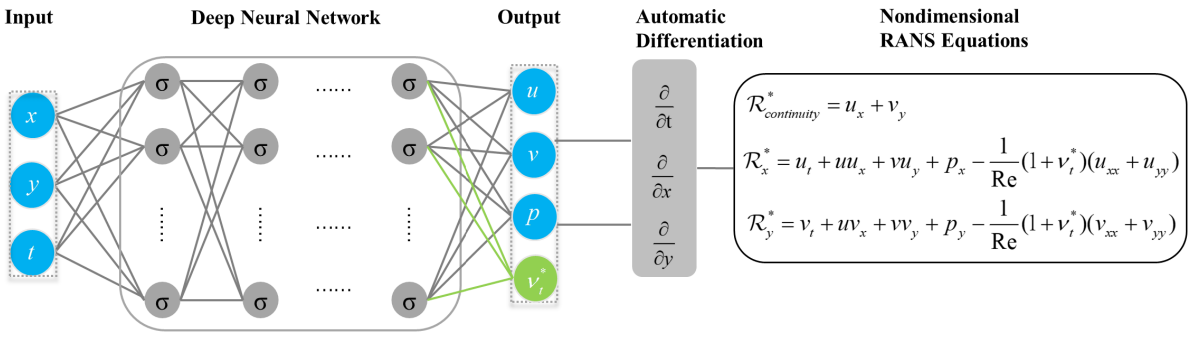


FIG. 2. Physics-informed neural networks for solving incompressible NS equations with artificial viscosity term. ν_t^* denotes the non-dimensional spatiotemporal viscosity parameter. σ denotes the activation function.

sub neural networks, additional constraints are required along the interfaces. Determining the number of constraints is a trade-off; while imposing more constraints can result in smoother, more continuous solutions across neighboring domains, it can also increase the computational burden and potentially lead to the neural networks' failure to solve the local field. Therefore, in this study, only data continuity is enforced on the interfaces. The loss function for the qth separate sub neural network is given by

$$MSE_q = MSE_{d_q} + \lambda_{r_q} \times MSE_{r_q} + \lambda_{I_q} \times MSE_{I_q}, \tag{8}$$

where

$$MSE_{d_q} = \frac{1}{N_{d_q}} \sum_{i=1}^{N_{d_q}} ||\tilde{u}_q(\mathbf{x}^i_{d_q}, t^i_{d_q}; w_q) - {u_{d_q}}^i||^2, \tag{9}$$

$$MSE_{r_q} = \frac{1}{N_{r_q}} \sum_{i=1}^{N_{r_q}} ||\mathbf{R}(\tilde{u}(\mathbf{x}^i_{r_q}, t^i_{r_q}; w_q))||^2, \tag{10}$$

$$MSE_{I_q} = \frac{1}{N_{I_q}} \sum_{i=1}^{N_{I_q}} ||\tilde{u}_q(\mathbf{x}^i_{I_q}, t^i_{I_q}; w_q) - \tilde{u}_{q^+}(\mathbf{x}^i_{I_q}, t^i_{I_q}; w_{q^+})||^2, \tag{11}$$

where N_{d_q}, N_{r_q}, and N_{I_q} denote the number of data points, collocation points, and interface points on the qth sub-domain, respectively. q^+ means the neighboring sub neural networks of the qth neural network. The subscripts d_q, r_q, and I_q represent data points set, residual (collocation) points set, and interface points set in the qth sub-domain, respectively. Equation (11) indicates that only data continuity is required on the interfaces.

For non-overlapping domain decomposition strategy, the data exchange boundary (where interface conditions are imposed) is the sub-domain boundary, as depicted in the left figure of Fig. 3. However, PINNs have been shown to perform poorly on the domain boundary, making it more sensible for sub-domains to exchange information within their boundaries.[48] Overlapping domain decomposition can be implemented based on non-overlapping strategy by enlarging the original sub-domain boundary while keeping the data exchange boundary unchanged. Therefore, more reliable data can be exchanged between sub neural networks without introducing additional communication cost, as illustrated in the middle and right figures of Fig. 3.

To train PINNs with overlapping domain decomposition, N_{sub} sub neural networks are deployed separately and simultaneously to minimize the loss function in Eq. (8). After every epoch (all the batches iterated for once), each sub neural network predicts the output $\tilde{u}_q(\mathbf{x}^i_{I_q}, t^i_{I_q}; w_q)$ on the N_{I_q} interface points and exchange the information with neighboring sub neural networks to generate Eq. (11). The specific parallel algorithm for training an overlapping decomposed problem is presented in Algorithm 1. After proper training, the overall solution of the flow field can be written as

$$\tilde{u}(\mathbf{x}, t) = \sum_{q=1}^{N_{sub}} \tilde{u}_q(\mathbf{x}, t; w_q) \cdot \Pi(\mathbf{x}, t), \tag{12}$$

where

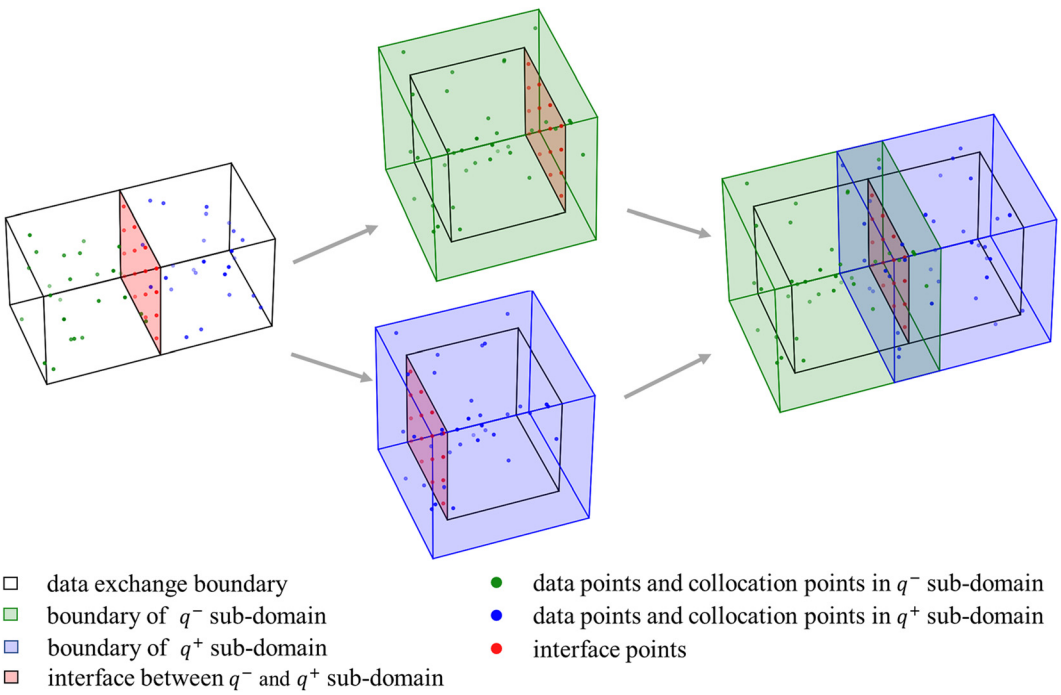


FIG. 3. Schematic diagram of overlapping domain decomposition. *Left*: typical non-overlapping domain decomposition, where data exchange boundary is exactly the sub-domain boundary. *Middle*: extension of non-overlapping domain decomposition, where sub-domain boundary is larger than data exchange boundary. *Right*: overlapping domain decomposition, where data exchange interface is within the boundary of each sub-domain. Data exchange boundary is where interface conditions are imposed and information on different sub neural networks is communicated. Sub-domain boundary is the boundary of data points and collocation points for a certain sub neural network.

ALGORITHM 1: Parallel algorithm for STPINNs.

Step 0: Prepare overall data points and collocation points

Data points: $\{u_d^i, \mathbf{x}_d^i, t_d^i\}_{i=1}^{N_d}$

Collocation points: $\{\mathbf{x}_r^i, t_r^i\}_{i=1}^{N_r}$

Step 1: Initialize MPI

$comm = MPI.COMM\_WORLD$

$size = comm.Get\_size()$

$rank = comm.Get\_rank()$

Step 2: Specify local data points, collocation points, and interface points in each process

Divide the domain into $N_{sub} = size$ number of non-overlapping sub-domains $\Omega_{q\_exchange}$

Enlarge the sub-domains $\Omega_{q\_exchange}$ by $overlapping\_rate$% and derive N_{sub} number of overlapping sub-domains Ω_{q_extend}

Assign the process to specific sub-domain $q = rank$

Local data points: $\{u_{d_q}^i, \mathbf{x}_{d_q}^i, t_{d_q}^i\}_{i=1}^{N_{d_q}}$, where $\{\mathbf{x}_{d_q}^i, t_{d_q}^i\}_{i=1}^{N_{d_q}} = \{\{\mathbf{x}_d^i, t_d^i\}_{i=1}^{N_d} \cap \Omega_{q\_extend}\}$

Local collocation points: $\{\mathbf{x}_{r_q}^i, t_{r_q}^i\}_{i=1}^{N_{r_q}}$, where $\{\mathbf{x}_{r_q}^i, t_{r_q}^i\}_{i=1}^{N_{r_q}} = \{\{\mathbf{x}_r^i, t_r^i\}_{i=1}^{N_{r_q}} \cap \Omega_{q\_extend}\}$

Local interface points: $\{\mathbf{x}_{I_q}^i, t_{I_q}^i\}_{i=1}^{N_{I_q}}$, where $\{\mathbf{x}_{I_q}^i, t_{I_q}^i\}_{i=1}^{N_{I_q}} \in \partial\Omega_{q\_exchange}$

for *epoch in Nepochs* **do**

- predict $\tilde{u}_q(\mathbf{x}_{I_q}^i, t_{I_q}^i; w_q)_{i=1}^{N_{I_q}}$
- communicate: upward, downward, rightward, leftward, forward, and backward, derive $\tilde{u}_{q^+}(\mathbf{x}_{I_q}^i, t_{I_q}^i; w_{q^+})_{i=1}^{N_{I_q}}$ from neighboring processes
- predict $\tilde{u}_q(\mathbf{x}_{d_q}^i, t_{d_q}^i; w_q)_{i=1}^{N_{d_q}}$
- predict residual $\mathbf{R}(\tilde{u}(\mathbf{x}_{r_q}^i, t_{r_q}^i; w_q))_{i=1}^{N_{r_q}}$
- compute $MSE_{I_q} = \frac{1}{N_{I_q}} \sum_{i=1}^{N_{I_q}} ||\tilde{u}_q(\mathbf{x}_{I_q}^i, t_{I_q}^i; w_q) - \tilde{u}_{q^+}(\mathbf{x}_{I_q}^i, t_{I_q}^i; w_{q^+})||^2$
- compute $MSE_{d_q} = \frac{1}{N_{d_q}} \sum_{i=1}^{N_{d_q}} ||\tilde{u}(\mathbf{x}_{d_q}^i, t_{d_q}^i; w_q) - {u_{d_q}}^i||^2$
- compute $MSE_{r_q} = \frac{1}{N_{r_q}} \sum_{i=1}^{N_{r_q}} ||\mathbf{R}(\tilde{u}(\mathbf{x}_{r_q}^i, t_{r_q}^i, w_q))||^2$
- compute $MSE_q = MSE_{d_q} + \lambda_{r_q} \times MSE_{r_q} + \lambda_{I_q} \times MSE_{I_q}$
- optimize independently on each process
- $w_q^* = \text{argmin}_{w_q}(MSE_{d_q} + \lambda_{r_q} \times MSE_{r_q} + \lambda_{I_q} \times MSE_{I_q})$

end

$$\Pi(x,t) = \begin{cases} 0 & \text{if } (\mathbf{x},t) \notin \Omega_q, \\ 1/n & \text{if } (\mathbf{x},t) \in \bigcap_{k=1}^{n} \Omega_k. \end{cases} \tag{13}$$

In this paper, unsteady 2D flows are considered, and the overall dimension is three (x, y, t). Therefore, the possible values of n are $1, 2, 4, 8$, meaning there are at most eight sub-domains overlapping each other.

III. NUMERICAL DATASET USED FOR TRAINING AND VALIDATION

In this paper, three turbulent flow cases were examined to evaluate the performance of STPINNs in reconstructing flow fields, i.e., the wake flow past a 2D circular cylinder, 2D decaying turbulence, and the wake flow past a 3D circular cylinder. The data used in flow field reconstruction are typically sparse in spatial domain due to the limitations of existing measurement methods and instrument accuracy. However, for high sampling frequency, the observed data can be dense in temporal domain. Therefore, in this study, the flow data for all three cases are sparsely distributed in space but densely distributed in time.

The wake flow past a 2D circular cylinder at Reynolds number $Re = 3900$ is simulated using the $k-\epsilon$ model.[49] The flow data u, v, p from 25 sparsely distributed points covering a period of 42.9 s are used as labeled training data, as shown in Fig. 4. The total number of data points N_d is 2500, with 25 data points in each of the 100 snapshots. The overall number of collocation points N_r is 480 000, which are sampled using the Latin hypercube sampling (LHS) method in the entire spatiotemporal domain.

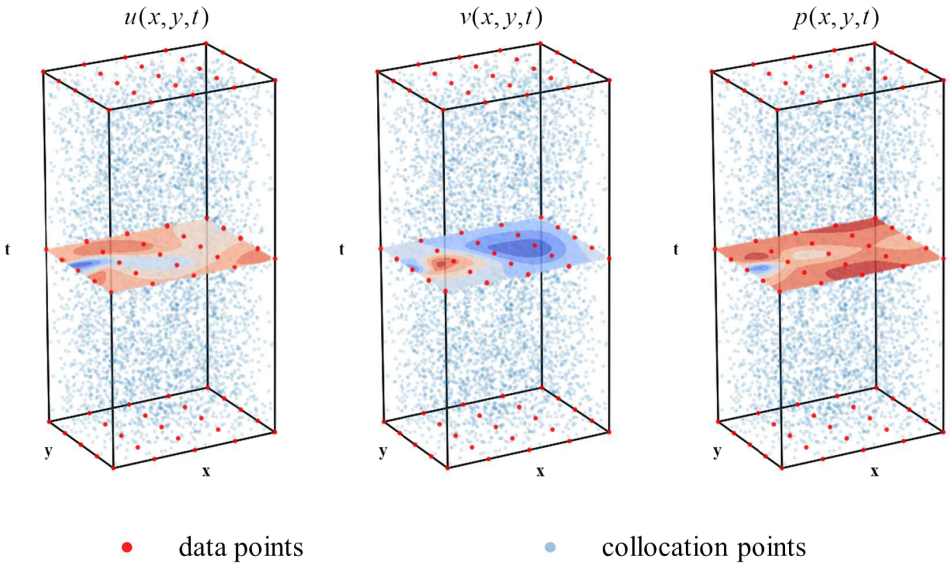


FIG. 4. Training set for wake flow of 2D cylinder. *Left*: stream-wise velocity *u*. *Middle*: transverse velocity *v*. *Right*: pressure *p*. 25 sparsely distributed data points from 100 snapshots (only three snapshots are plotted) covering a period of 43 s, and 480 000 collocation points sampled through Latin Hypercube Sampling are taken as a training set.

The 2D decaying turbulence at $Re = 2000$ is calculated using the pseudo-spectral method[50] with 512×512 grids on a periodic domain of $x \in [0, 2\pi], y \in [0, 2\pi]$. As shown in Fig. 5, the central part of the decaying turbulence is selected as the training and validation domain in this paper. Within the range of $x \in [0.50\pi, 1.50\pi], y \in [0.50\pi, 1.50\pi]$, a total of 25 points are chosen to provide sparse observation of u, v, p. Similar to Fig. 4, 2500 pieces of data information ($N_d = 2500$) are gathered from 100 snapshots, covering a period of 1.3 s. The overall number of collocation points N_r for this case is 320 000, which is also sampled using the LHS method.

The original 3D flow past a circular cylinder at $Re = 3900$ is calculated using large eddy simulation (LES),[49,51] as shown in Fig. 6(a). By averaging the wake flow along the span-wise direction, a 2D unsteady flow can be obtained, as shown in Fig. 6(b). For this 3D average turbulent flow, 49 points [shown in Fig. 6(b)] from 100 snapshots covering a period of 14.3 s are selected to provide sparse observation of $u_{ave}, v_{ave}, p_{ave}$. The overall number of data points N_d and collocation points N_r are 4900 and 1 000 000, respectively.

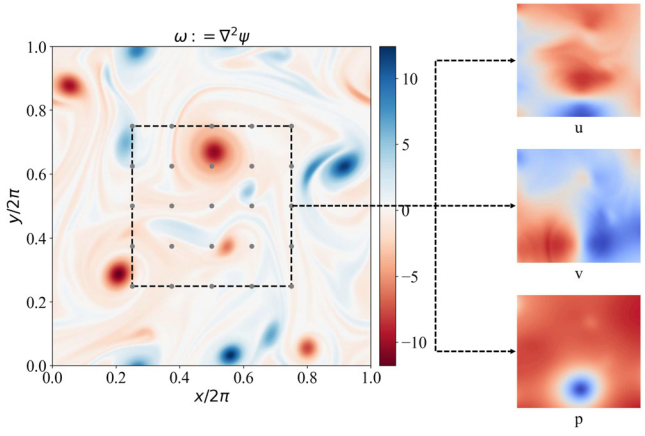
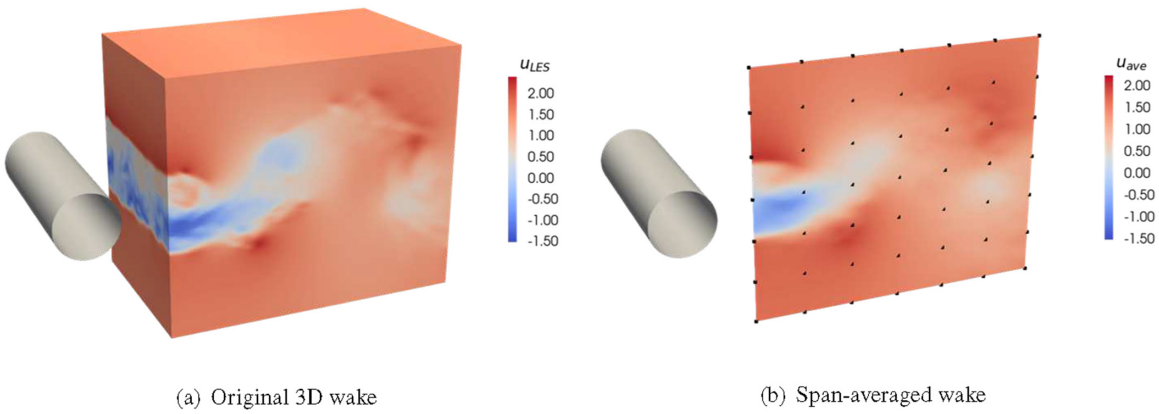


FIG. 5. 2D decaying turbulence. *Left*: vorticity. *Right*: stream-wise velocity *u*, transverse velocity *v*, and pressure *p*. The rectangular area is the selected training and validation domain in this paper. The gray points are where data of u, v, p are sampled.

To verify the accuracy of the numerical simulations, Table I summarizes the results for the flow around a cylinder observed in experiment and given by numerical simulations at $Re = 3900$. Figure 7 presents the energy spectrum of the 2D decaying turbulence used in this paper.

IV. RESULTS

In this section, the proposed STPINNs are used to solve the three turbulent flow cases discussed earlier. Domain decomposition is carried out along *x*, *y*, and *t* directions, as shown in Fig. 8. Research has previously demonstrated that spatial domain decomposition has a considerable impact on the precision of the final prediction,[55] where too many sub neural networks might result in insufficient labeled data allocation to each neural network, making them prone to overfitting and unable to identify the appropriate local solution. In the context of reconstructing flow fields from sparse observations, the observed data are usually too spatially few to be assigned to too many sub neural networks. Therefore, in this paper, only four sub-domains (2×2, as shown in Fig. 8) were spatially divided for all three cases. Nevertheless, the observed data can be dense in time due to the high sampling frequency of current measurement methods, thus domain decomposition along time axis can also be relatively dense. The commonly used tanh (x) function was chosen as the activation function for this study. To efficiently train the sub neural networks and minimize hyperparameter tuning, the Adam optimizer[56] was used in conjunction with cosine annealing learning rate schedule.[57] The specific learning rate schedule adopted in this paper is shown in Fig. 9, which demonstrates excellent performance in accelerating convergence[47,49] by regularly decaying and warm restarting learning rate. The STPINNs were trained using multi-central processing units (CPUs): X86 7285.



(a) Original 3D wake (b) Span-averaged wake

FIG. 6. 3D flow past a circular cylinder at $Re = 3900$. (a) Stream-wise velocity u_{LES}; (b) span-averaged value u_{ave} obtained from LES of the target wake region, and the scattered black points are where data of span-averaged $u_{ave}, v_{ave}, p_{ave}$ are sampled. (a) Original 3D wake and (b) span-averaged wake.

TABLE I. Drag coefficient and Strouhal number of flow around a cylinder.

Re	case	Cd	St
3900	Exp. (Norberg, 1994)[52]	···	0.210
	CFD 3D LES (Lysenko *et al.*, 2012)[53]	0.970	0.209
	CFD 2D k–ε (Rahman *et al.*, 2007)[54]	0.997	0.200
	Present CFD case 2D k–ε	**0.922**	**0.208**
	Present CFD case 3D LES	**0.954**	**0.208**

A. Wake flow of 2D cylinder

For the wake flow past a 2D circular cylinder, 40 CPUs were deployed to train 40 sub neural networks within the overall non-dimensional spatiotemporal domain $x \in [1.000, 5.000], y \in [-2.000, 2.000]$, and $t \in [0.000, 42.937]$. Each sub neural network consisted of ten hidden layers, each with 50 neurons. The domain was divided into four spatial (2×2) and ten temporal divisions. The overlapping rate for sub-domains was 10%, meaning that the sub-domain boundary was 10% larger than data exchange boundary along x, y, t directions. The flow was reconstructed using the training set described in Sec. III. The interface points were $\{N_x = 25, N_y = 25, N_t = 10\}$, meaning that for each sub neural network, 625 pieces of information were exchanged on the $x - y$ interfaces, and 250 pieces of information were exchanged on the $x - t$ and $y - t$ interfaces. For each sub neural network, the localized numbers of data points N_{d_q}, interface points N_{I_q}, and collocation points N_{r_q} were 99, 1125, and 12 000, respectively.

A comparison was conducted between using NS equations and RANS equations, namely, STPINNs-NS and STPINNs-RANS. As can

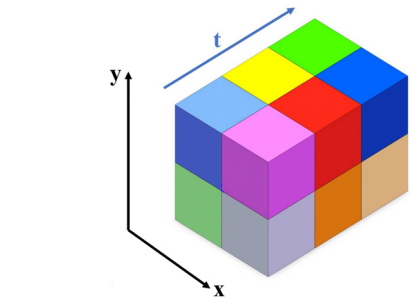


FIG. 8. Domain decomposition along x, y, and t directions. Only the data exchange boundaries are shown for visualization. The graph demonstrates a spatiotemporal domain divided into four spatial (2×2) and three temporal divisions.

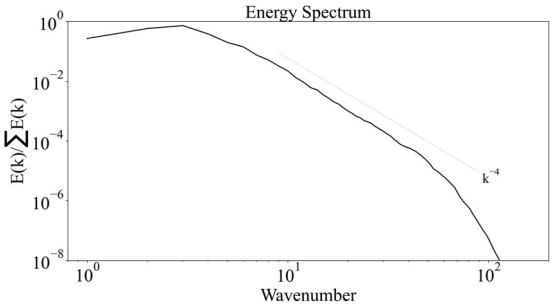


FIG. 7. Energy spectrum of 2D decaying turbulence.

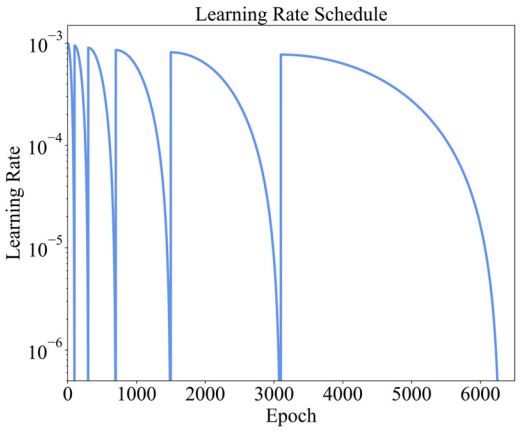


FIG. 9. Cosine annealing learning rate schedule, with warm restart.

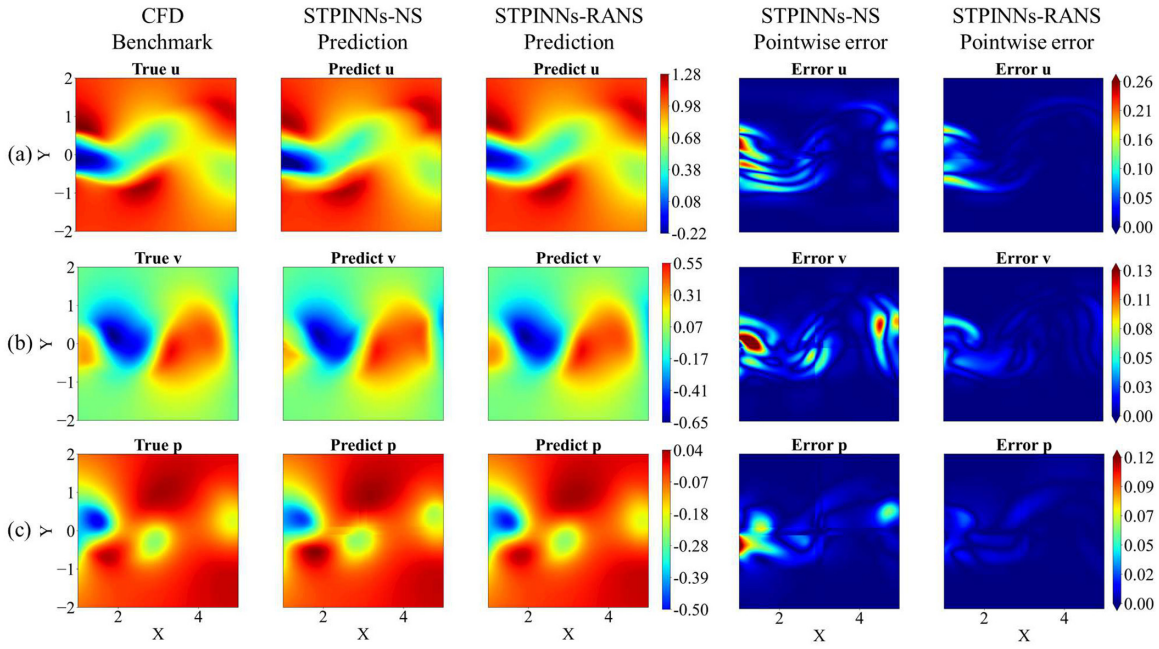


FIG. 10. CFD benchmark flow field (first column, flow past 2D circular cylinder), predicted flow field using NS equations (second column) and RANS equations (third column), and the pointwise error between the benchmark field and the predicted field using NS equations (fourth column) and RANS equations (fifth column) at a temporal snapshot $t = 17.78$. (a) Stream-wise velocity u; (b) transverse velocity v; (c) pressure p.

be seen in Fig. 10, the proposed STPINNs framework reconstructed the original flow field and captured the vortex shedding pattern from sparse observation of only 25 points. Figure 11 compares the time-averaged velocity profiles sampled at x/D = 1.06, 1.54, and 2.02. In addition, using RANS equations in the loss function resulted in better outcomes than using the commonly used NS equations. Relative L_2 norm was also introduced here to quantitatively compare the deviation of the predicted flow field and the original flow field in the entire temporal domain, as defined in the following equation:

$$R_{L_2} = \frac{||\hat{U} - U||}{||U||}, \tag{14}$$

where $||\hat{U} - U||$ is the L_2 norm of the prediction deviation of interested quantity $\{u, v, p\}$ at a certain time and $||U||$ denotes the L_2 norm

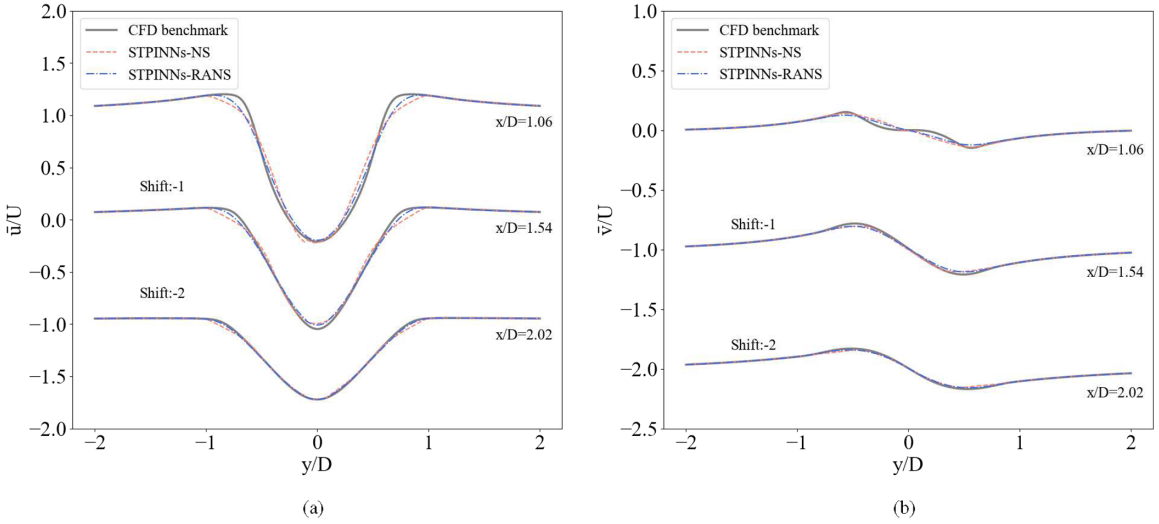


FIG. 11. Time-averaged velocity profiles sampled at x/D = 1.06, 1.54, and 2.02 (flow past 2D circular cylinder). (a) Stream-wise velocity; (b) transverse velocity. For ease of presentation, the velocity profiles at x/D = 1.54 and 2.02 have been shifted by certain values.

of the original quantity at that time. As shown in Fig. 12, the predicted flow field using RANS equations was more accurate than that of NS equations. Through adding artificial viscosity term in the NS equations, the obtained average R_{L_2} of u, v, and p was 0.018, 0.033, and 0.040, respectively, which were lower than 0.043, 0.096, and 0.119 obtained using NS equations.

A study on the strong and weak scaling capacity was also conducted for STPINNs-RANS. For the strong scaling study, the problem size was fixed, which was to reconstruct the flow field based on the available 2500 pieces of labeled data over the period of 42.9 s. Due to the extensive consumption of computational resources when calculating the residual of conservation equations, to fix the problem size, the total number of collocation points $N_r = \sum_{q=1}^{N_{sub}} N_{r_q}$ should be fixed as well when exploring strong scaling capacity. Therefore, for strong scaling study, the 2500 data points and 480 000 collocation points were split across multi-CPUs. Furthermore, due to the random nature of neural networks, it was also necessary to evaluate prediction accuracy while conducting strong and weak scaling study, as the accuracy may vary as the number of deployed sub neural networks changes, which is notably different from traditional numerical methods. The results of strong scaling capacity for this case are presented in Figs. 13(a)–13(c). As shown in Fig. 13(a), the speedup line of strong scaling is close to the ideal line whose slope is 1, and in some area, it even exceeds the ideal line. This is due to the training procedure used in this paper, where the labeled data for flow field reconstruction are so precious that after each iteration\batch training, all the data points assigned to a sub neural network would be trained once, making the training procedure more demanding for serial training (or when the number of CPUs is small). However, the speedup deteriorates as the number of CPUs is increased beyond 32, when the percentage of communication time keeps rising, as illustrated in Fig. 13(b). The prediction accuracy remains stable when the number of CPUs is increased beyond 8, as shown in Fig. 13(c).

In terms of weak scaling, the problem size was enlarged as the number of CPUs increased. Specifically, when four CPUs were deployed, only 250 pieces of data information from ten snapshots were used as labeled data to reconstruct the original flow field covering a period of 4.3 s. Whereas when 40 CPUs were deployed, 2500 pieces of data information from 100 snapshots were used to reconstruct the 43-s long unsteady flow field. For weak scaling, the problem size on each separate sub neural network was the same, with $N_{d_q} = 99$, $N_{I_q} = 1125$, and $N_{r_q} = 12\,000$. As shown in Fig. 13(d), the parallel efficiency decreases as the number of devices increases, since the communication time increases with the problem size, as shown in Fig. 13(e). The prediction accuracy remains almost the same as the number of devices increases, as illustrated in Fig. 13(f).

B. 2D decaying turbulence

2D homogeneous isotropic decaying turbulence was also tested here to further validate the performance of the proposed STPINNs. Similar to the wake flow past a 2D circular cylinder, 40 CPUs were deployed to train 40 sub neural networks within the range of $x \in [0.50\pi, 1.50\pi], y \in [0.50\pi, 1.50\pi]$, and $t \in [19.637, 20.937]$. The spatial division was 4 and temporal division was 10, with an overlapping rate of 10%. Each sub neural network had a size of ten hidden layers and 50 neurons per hidden layer. The interface points were $\{N_x = 25, N_y = 25, N_t = 10\}$, with 625 interface points on the x–y interfaces and 250 interface points on the x–t and y–t interfaces. The localized numbers of data points N_{d_q}, interface points N_{I_q}, and collocation points N_{r_q} was 99, 1125, and 8000, respectively.

Figures 14 and 15 compare the performance between using NS equations (STPINNs-NS) and RANS equations (STPINNs-RANS) and demonstrate that by adding eddy viscosity in the output of the neural network, STPINNs can get a better result using essentially the same amount of computational resources. Figure 16 further gives a detailed comparison of R_{L_2} at each time snapshot. The results demonstrate that the average R_{L_2} values obtained using RANS equations were lower than those obtained using NS equations. Specifically, the average R_{L_2} values for u, v, and p obtained using RANS equations were 0.120, 0.090, and 0.052, respectively, while those obtained using NS equations were 0.150, 0.101, and 0.053.

Strong scaling and weak scaling capacity for this 2D fully turbulent flow using RANS equations is demonstrated in Fig. 17. For strong scaling, the problem size was fixed, and the overall 2500 pieces of labeled data and 320 000 collocation points were spilt across multi-CPUs. As can be seen in Fig. 17(a), the speedup achieved in this case is even higher than that presented in Fig. 13(a). This is caused by the more demanding feature of the training procedure for serial training (or when the number of devices is small) when solving flow field reconstruction problems, where labeled data are so precious that needs

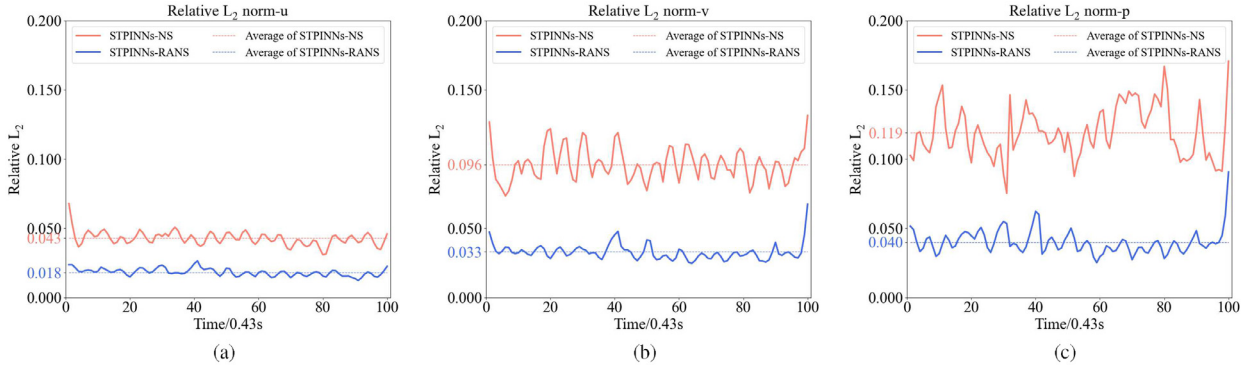


FIG. 12. Relative L_2 norm between the benchmark flow field (flow past 2D circular cylinder) and the predicted flow field using NS equations and RANS equations. (a) Streamwise velocity u; (b) transverse velocity v; (c) pressure p.

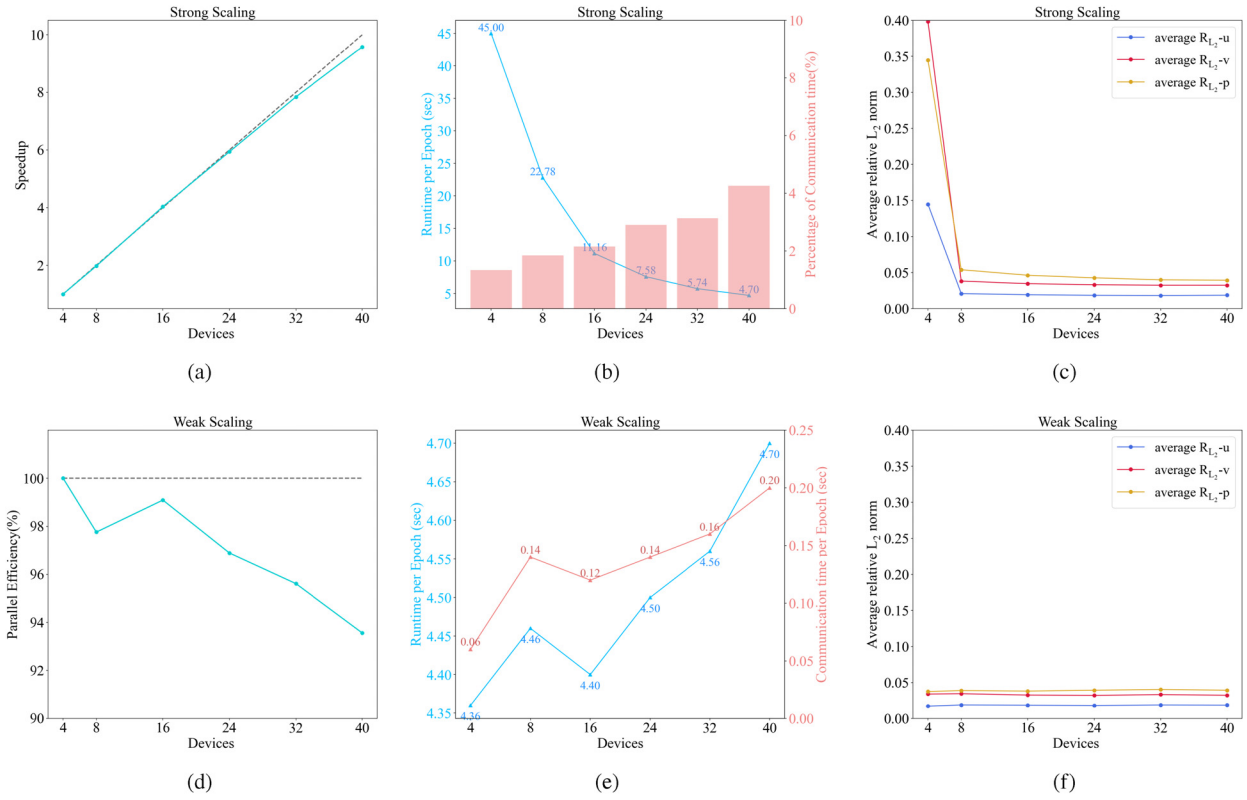


FIG. 13. Strong and weak scaling capacity of wake flow of the 2D cylinder. (a) Speedup of strong scaling; (b) runtime and communication time of strong scaling; (c) prediction accuracy of strong scaling, represented by average R_{L_2}; (d) parallel efficiency of weak scaling; (e) runtime and communication time of weak scaling (f) prediction accuracy of weak scaling, represented by average R_{L_2}. The gray lines in (a) and (d) are the ideal speedup and parallel efficiency lines. *Strong scaling*: keep the problem size fixed, and data points and collocation points are split across multi devices. *Weak scaling*: keep the problem size on each separate sub neural network fixed, and the overall problem size is enlarged as the number of devices increases.

to be trained after every iteration. Since the number of collocation points in this case is lower than that in Sec. IV A, thus, the speedup line is even higher, and the communication time represents a higher percentage in this case, as shown in Fig. 17(b). However, the speedup deteriorates as the number of CPUs reaches 40. Meanwhile, as shown in Fig. 17(c), the prediction accuracy for different number of devices is relatively stable, indicating that this 2D turbulent case is insensitive to the number of collocation points assigned in each sub neural network given the training setup used in this paper.

For weak scaling, the problem size on each separate sub neural network was fixed, with $N_{d_q} = 99$, $N_{I_q} = 1125$, and $N_{r_q} = 8000$. Similar to Figs. 13(d) and 17(e), the parallel efficiency decreases, and the communication scale increases as the number of devices increases, as shown in Figs. 17(d) and 17(e). As for the prediction accuracy, a noticeable downward trend is evident in Fig. 17(f) as the number of devices increases, which is significantly different from the pattern demonstrated in Fig. 13(f). This difference can be attributed to the distinctive flow characteristics of the wake flow past a circular cylinder and decaying turbulence. The periodic vortex shedding pattern of wake flow past a circular cylinder at $Re = 3900$ results in near-constant prediction accuracy for different lengths of time (different numbers of devices) given the reconstruction task, whereas for decaying turbulence, the flow originates from an initial vorticity distribution and developed due to viscous dissipation, resulting in flow characteristics that are not entirely consistent across different time frames. Therefore, as the number of devices increases, the problem size is increased, and more information along the time axis is added, making the training set more abundant and the prediction error lower.

C. Wake flow of 3D cylinder

To explore STPINNs' potential for solving 3D turbulent flow, the wake flow of 3D cylinder calculated using the LES method was studied here. However, the current framework of STPINNs proposed in this paper was designed to resolve 2D unsteady flow by deploying three-dimensional (x, y, t) domain decomposition strategy. To efficiently solve the inverse problem of 3D unsteady flow, a four-dimensional (x, y, z, t) domain decomposition strategy will be needed, which is part of our research goal for future work and beyond the scope of this paper. However, by span-averaging the 3D unsteady flow, an average 2D flow can be derived and used to test the performance of STPINNs, as introduced in Sec. III. Similar to previous cases, 40 CPUs were deployed to train 40 sub neural networks within the range of $x \in [1.000, 5.000], y \in [-2.000, 2.000], t \in [0.000, 13.313]$, with a spatiotemporal subdivision of 4 and 10. The overlapping rate was 10%, and the size of each sub neural network was ten hidden layers

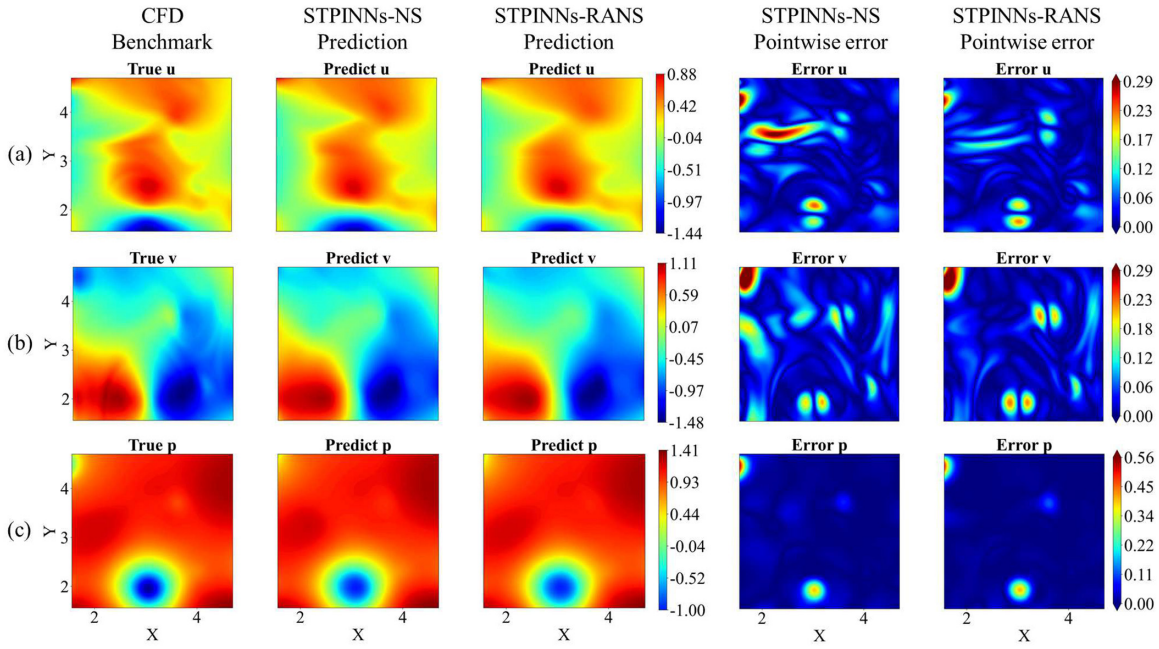


FIG. 14. CFD benchmark flow field (first column, 2D decaying turbulence flow), predicted flow field using NS equations (second column) and RANS equations (third column), and the pointwise error between the benchmark field and the predicted field using NS equations (fourth column) and RANS equations (fifth column) at a temporal snapshot $t = 20.157$. (a) Stream-wise velocity u; (b) transverse velocity v; (c) pressure p.

with 50 neurons per hidden layer. The interface points were $\{N_x = 25, N_y = 25, N_t = 10\}$, with 625 interface points on the x–y interfaces and 250 interface points on the x–t and y–t interfaces. The localize numbers of data points N_{d_q}, interface points N_{I_q}, and collocation points N_{r_q} were 176, 1125, and 25 000, respectively.

A comparative study was also conducted to evaluate the effectiveness of STPINNs-NS and STPINNs-RANS. As can be seen in Figs. 18 and 19, utilizing three-dimensional domain decomposition strategy, the vortex shedding pattern was accurately captured, and the average flow field can be reconstructed with relatively high level of accuracy, with RANS equations exhibiting marginally better performance than NS equations. Specifically, the average R_{L_2} values for u, v, and p obtained using RANS equations were 0.049, 0.131, and 0.197, respectively, which were slightly higher than those obtained using NS

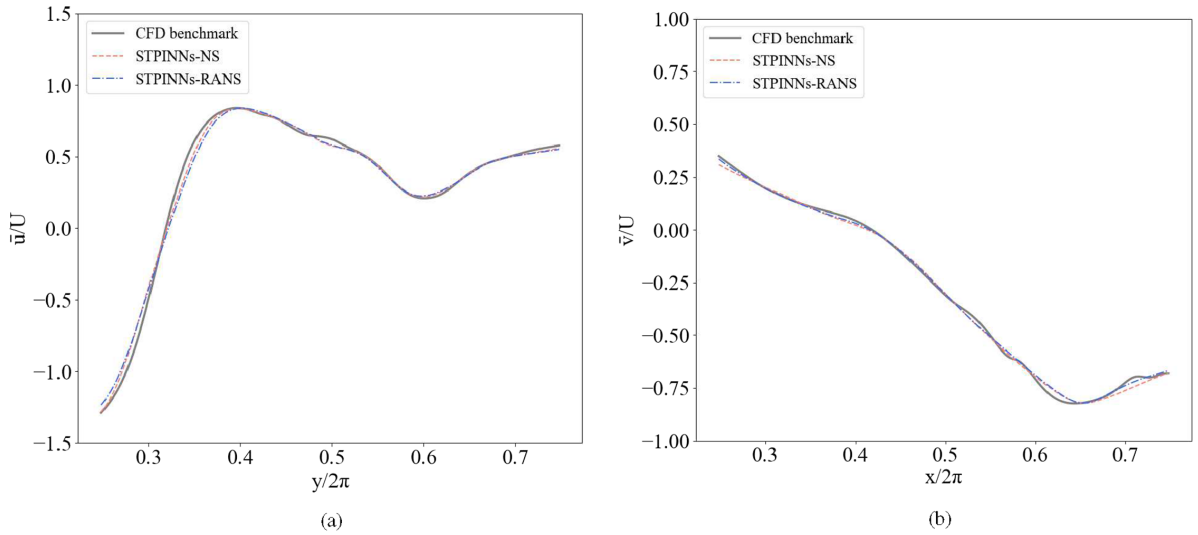


FIG. 15. Time-averaged velocity profiles. (a) Stream-wise velocity sampled at $x/2\pi = 0.5$; (b) transverse velocity sampled at $y/2\pi = 0.5$.

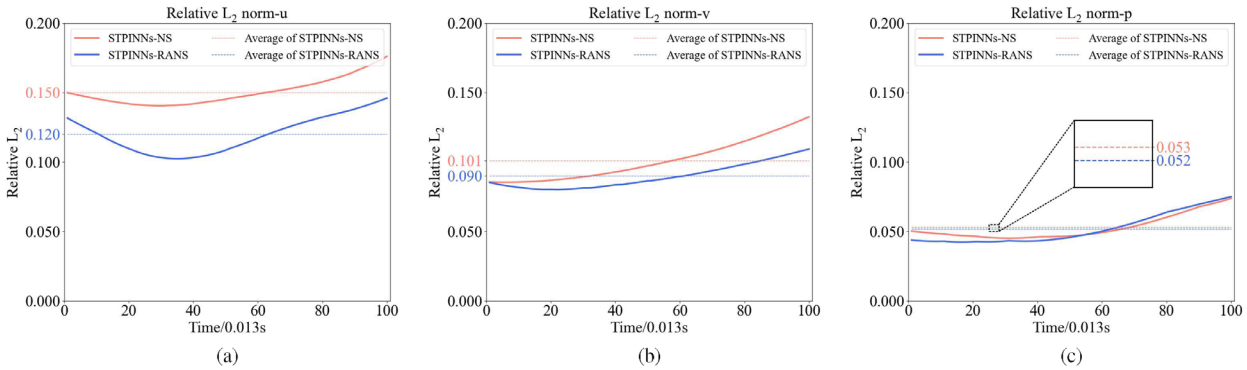


FIG. 16. Relative L_2 norm between benchmark flow field (2D decaying turbulence flow) and predicted flow field using NS equations and RANS equations. (a) Stream-wise velocity u; (b) transverse velocity v; (c) pressure p.

equations, which were 0.055, 0.145, and 0.236, as shown in Fig. 20. While RANS equations do not exhibit overwhelming advantage compared to commonly used NS equations in this case, the results still illustrate the robust ability of the proposed STPINNs for solving pseudo-3D turbulent flow.

V. CONCLUSION AND DISCUSSION

To summarize, we proposed a spatiotemporal parallel framework of physics-informed neural networks (PINNs) that aimed to accurately and efficiently solve inverse problems of fluid mechanics. The specific inverse problem studied in this paper is the flow field reconstruction

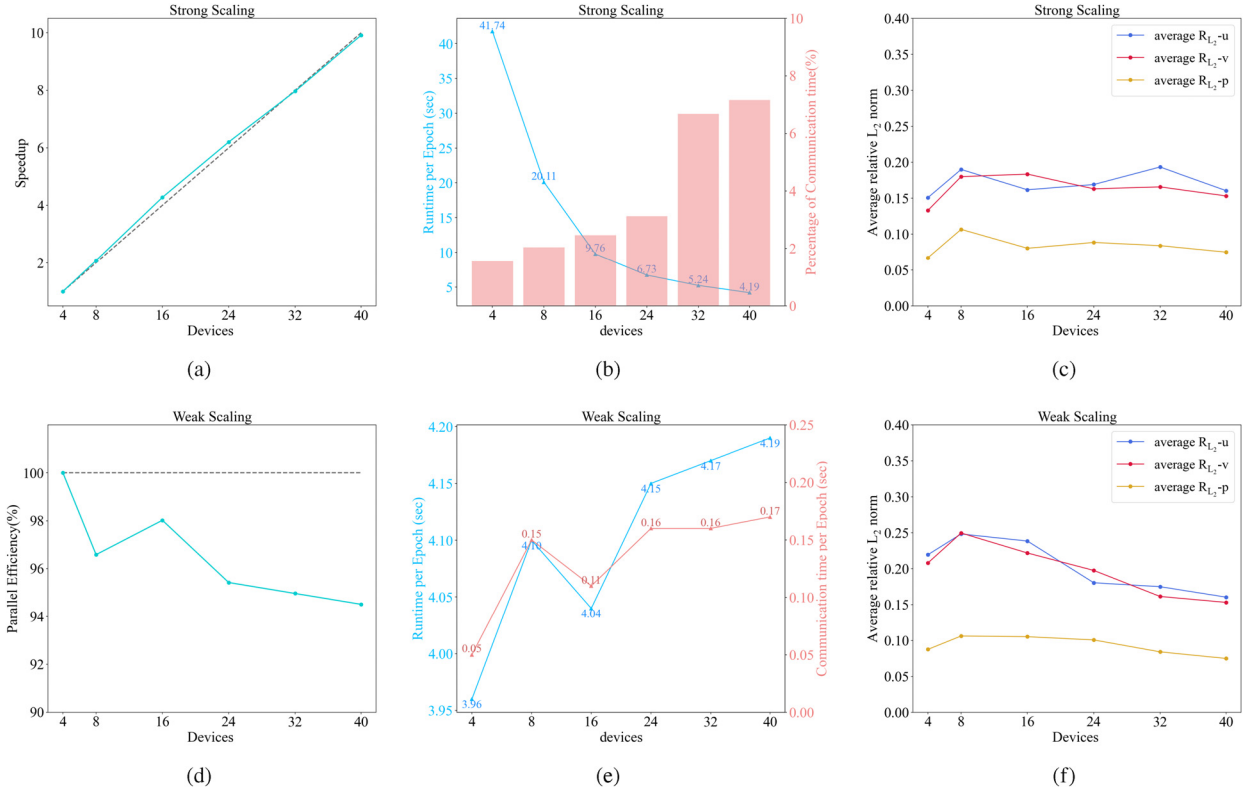


FIG. 17. Strong and weak scaling capacity of 2D decaying turbulence. (a) Speedup of strong scaling; (b) runtime and communication time of strong scaling; (c) prediction accuracy of strong scaling, represented by average R_{L_2}; (d) parallel efficiency of weak scaling; (e) runtime and communication time of weak scaling (f) prediction accuracy of weak scaling, represented by average R_{L_2}. The gray lines in (a) and (d) are the ideal speedup and parallel efficiency lines. *Strong scaling*: keep the problem size fixed, and data points and collocation points are split across multi devices. *Weak scaling*: keep the problem size on each separate sub neural network fixed, and the overall problem size is enlarged as the number of devices increases.

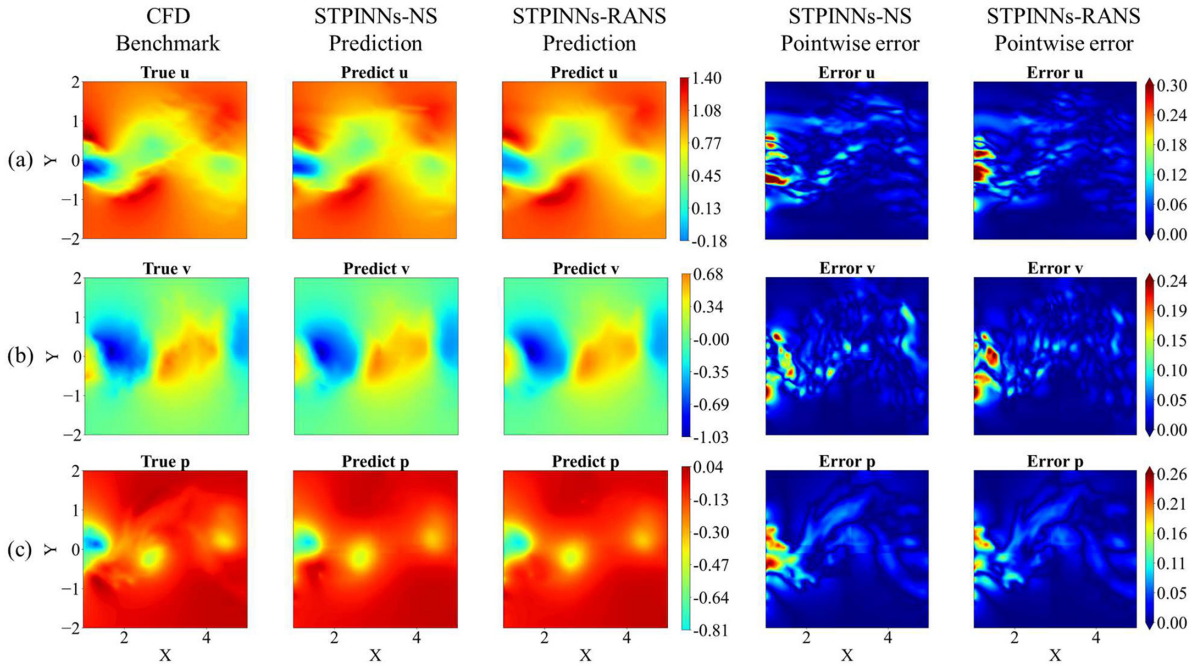


FIG. 18. CFD benchmark flow field (first column, span-average wake flow past a 3D circular cylinder), predicted flow field using NS equations (second column) and RANS equations (third column), and the pointwise error between the benchmark field and the predicted field using NS equations (fourth column) and RANS equations (fifth column) at a temporal snapshot $t = 20.157$. (a) Stream-wise velocity u; (b) transverse velocity v; (c) pressure p.

from sparse observation. We successfully reconstructed the unsteady flow of wake past a 2D cylinder and 2D decaying turbulence using only 25 sparsely distributed points and captured the flow characteristics of average wake flow past a 3D cylinder with relatively high accuracy using 49 points through the implementation of a 3D overlapping domain decomposition strategy that only adopted the data continuity interface condition. To suit the need for solving turbulent flow with relatively high Reynolds number, eddy viscosity was incorporated in the output of neural networks to model Reynolds stress of RANS equations. Utilizing STPINNs framework, the average R_{L_2} derived from RANS equations was lower than those derived from NS equations in all three turbulent cases, which indicated that incorporating

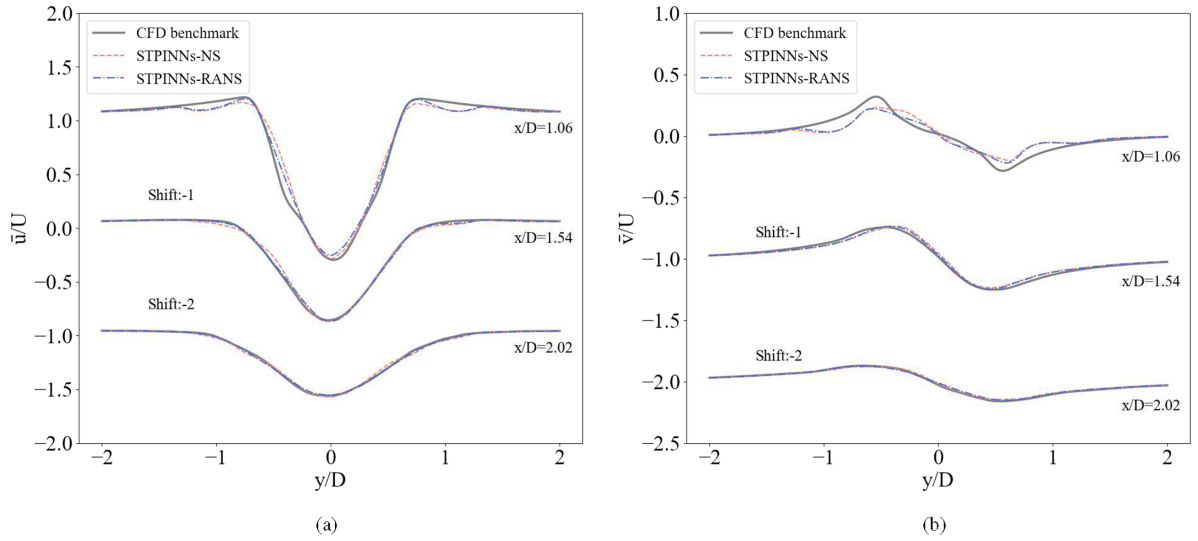


FIG. 19. Time-averaged velocity profiles sampled at x/D = 1.06, 1.54, and 2.02 (span-average wake flow past a 3D circular cylinder). (a) Stream-wise velocity; (b) transverse velocity. For ease of presentation, the velocity profiles at x/D = 1.54 and 2.02 have been shifted by certain values.

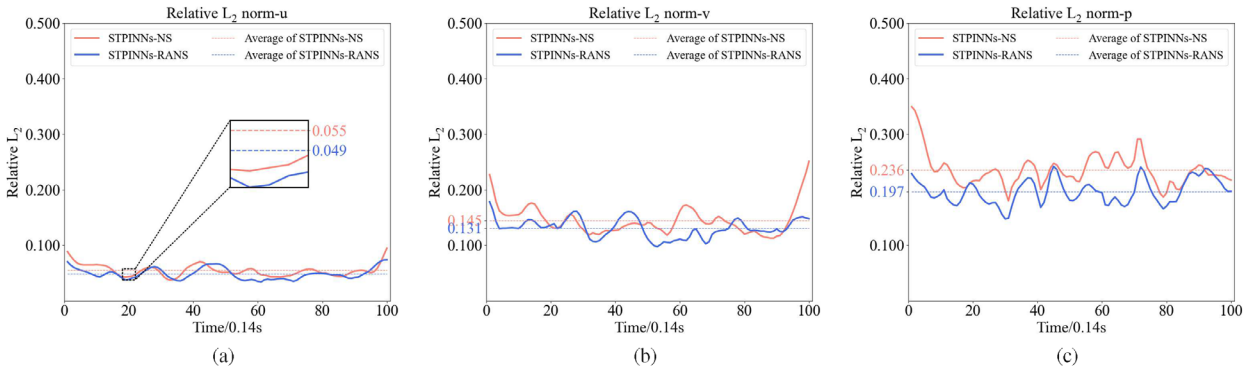


FIG. 20. Relative L_2 norm between benchmark flow field (span-average wake flow past a 3D circular cylinder) and predicted flow field using NS equations and RANS equations. (a) Stream-wise velocity u; (b) transverse velocity v; (c) pressure p.

RANS equations in the loss function can provide better performance while using essentially the same amount of computational resources. Moreover, strong scaling and weak scaling study of wake flow past 2D cylinder and 2D decaying turbulence illustrated the parallel capacity of the STPINNs framework. For strong scaling, STPINNs yielded satisfactory speedup, which was close to the ideal line, and exhibited stable prediction accuracy. For weak scaling, STPINNs presented parallel efficiency of over 90% when the number of devices was less than 40 and exhibited relatively stable and reasonable prediction accuracy. The results of the proposed STPINNs presented in this paper show encouraging insights that PINNs can be further adjusted and utilized to solve turbulent flow by adopting traditional domain decomposition strategy and Reynolds averaging strategy.

However, the current STPINNs framework is limited to providing three-dimensional overlapping domain decomposition solely for 2D unsteady flows. To efficiently solve the inverse problem of 3D unsteady flows, a four-dimensional parallel architecture will be needed, which is the next stage of our research goal and is distinctively different from traditional domain decomposition architecture. Meanwhile, inconsistent and uncontrollable prediction accuracy among the sub neural networks is another major challenge that needs to be tackled for the STPINNs framework to be applied to more turbulent flows.

ACKNOWLEDGMENTS

This work was supported by National Key Research and Development Project (Grant No. 2022YFB2603400), International Partnership Program of Chinese Academy of Sciences (Grant No. 025GJHZ2022118FN), and the China National Railway Group Science and Technology Program (Grant No. K2023J047).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Shengfeng Xu: Conceptualization (lead); Data curation (lead); Formal analysis (equal); Methodology (lead); Software (lead); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Chang Yan:** Conceptualization (equal); Data curation (equal); Methodology (lead); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Guangtao Zhang:** Conceptualization (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Validation (equal). **Zhenxu Sun:** Conceptualization (equal); Funding acquisition (lead); Supervision (lead); Writing – original draft (equal); Writing – review & editing (equal). **Renfang Huang:** Methodology (equal); Supervision (equal); Validation (equal); Visualization (equal). **Shengjun Ju:** Investigation (equal); Methodology (equal); Validation (equal); Visualization (equal). **Dilong Guo:** Project administration (equal); Supervision (equal); Validation (equal); Visualization (equal). **Guowei Yang:** Project administration (equal); Resources (equal); Supervision (equal); Validation (equal).

DATA AVAILABILITY

The data and code that support the findings of this study are openly available at https://github.com/Shengfeng233/PINN-MPI.

REFERENCES

[1]G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," Nat. Rev. Phys. **3**, 422–440 (2021).
[2]S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics–informed neural networks: Where we are and what's next," J. Sci. Comput. **92**, 88 (2022).
[3]M. Raissi, P. Perdikaris, and 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**, 686–707 (2019).
[4]A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," J. Mach. Learn. Res. **18**, 1–43 (2018).
[5]S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks for heat transfer problems," J. Heat Transfer **143**, 060801 (2021).
[6]R. Laubscher, "Simulation of multi-species flow and heat transfer using physics-informed neural networks," Phys. Fluids **33**, 087101 (2021).
[7]E. De Bézenac, A. Pajot, and P. Gallinari, "Deep learning for physical processes: Incorporating prior scientific knowledge," J. Stat. Mech.: Theory Exp. **2019**, 124009.

[8]Y. Zhu, R.-H. Zhang, J. N. Moum, F. Wang, X. Li, and D. Li, "Physics-informed deep-learning parameterization of ocean vertical mixing improves climate simulations," Nat. Sci. Rev. **9**, nwac044 (2022).
[9]Z. Zhou and Z. Yan, "Solving forward and inverse problems of the logarithmic nonlinear Schrödinger equation with PT-symmetric harmonic potential via deep learning," Phys. Lett. A **387**, 127010 (2021).
[10]H. Jin, M. Mattheakis, and P. Protopapas, "Physics-informed neural networks for quantum eigenvalue problems," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2022), pp. 1–8.
[11]J. Bai, T. Rabczuk, A. Gupta, L. Alzubaidi, and Y. Gu, "A physics-informed neural network technique based on a modified loss function for computational 2d and 3d solid mechanics," Comput. Mech. **71**, 543–562 (2023).
[12]E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, "A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics," Comput. Methods Appl. Mech. Eng. **379**, 113741 (2021).
[13]S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNS) for fluid mechanics: A review," Acta Mech. Sin. **37**, 1727–1738 (2021).
[14]V. Kag, K. Seshasayanan, and V. Gopinath, "Physics-informed data based neural networks for two-dimensional turbulence," Phys. Fluids **34**, 055130 (2022).
[15]X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations," J. Comput. Phys. **426**, 109951 (2021).
[16]H. Wang, Y. Liu, and S. Wang, "Dense velocity reconstruction from particle image velocimetry/particle tracking velocimetry using a physics-informed neural network," Phys. Fluids **34**, 017116 (2022).
[17]H. Gao, L. Sun, and J.-X. Wang, "Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels," Phys. Fluids **33**, 073603 (2021).
[18]Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," Comput. Methods Appl. Mech. Eng. **360**, 112789 (2020).
[19]S. Cai, Z. Wang, F. Fuest, Y. J. Jeon, C. Gray, and G. E. Karniadakis, "Flow over an espresso cup: Inferring 3-d velocity and pressure fields from tomographic background oriented schlieren via physics-informed neural networks," J. Fluid Mech. **915**, A102 (2021).
[20]M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," Science **367**, 1026–1030 (2020).
[21]G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, and P. Perdikaris, "Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow MRI data using physics-informed neural networks," Comput. Methods Appl. Mech. Eng. **358**, 112623 (2020).
[22]S. Wang, S. Sankaran, and P. Perdikaris, "Respecting causality is all you need for training physics-informed neural networks," arXiv:2203.07404 (2022).
[23]S. Maddu, D. Sturm, C. L. Müller, and I. F. Sbalzarini, "Inverse dirichlet weighting enables reliable training of physics informed neural networks," Mach. Learn.: Sci. Technol. **3**, 015026 (2022).
[24]A. Daw, J. Bu, S. Wang, P. Perdikaris, and A. Karpatne, "Rethinking the importance of sampling in physics-informed neural networks," arXiv:2207.02338 (2022).
[25]C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu, "A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks," Comput. Methods Appl. Mech. Eng. **403**, 115671 (2023).
[26]A. D. Jagtap, K. Kawaguchi, and G. Em Karniadakis, "Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks," Proc. R. Soc. A **476**, 20200334 (2020).
[27]A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," J. Comput. Phys. **404**, 109136 (2020).
[28]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning. II. Data-driven discovery of nonlinear partial differential equations," arXiv:1711.10566 (2017).
[29]M. F. Fathi, I. Perez-Raya, A. Baghaie, P. Berg, G. Janiga, A. Arzani, and R. M. D'Souza, "Super-resolution and denoising of 4d-flow MRI using physics-informed deep neural nets," Comput. Methods Programs Biomed. **197**, 105729 (2020).
[30]N. Wang, Q. Chen, and Z. Chen, "Reconstruction of nearshore wave fields based on physics-informed neural networks," Coastal Eng. **176**, 104167 (2022).
[31]M. P. Sitte and N. A. K. Doan, "Velocity reconstruction in puffing pool fires with physics-informed neural networks," Phys. Fluids **34**, 087124 (2022).
[32]K. Shukla, A. D. Jagtap, and G. E. Karniadakis, "Parallel physics-informed neural networks via domain decomposition," J. Comput. Phys. **447**, 110683 (2021).
[33]A. Heinlein, A. Klawonn, M. Lanser, and J. Weber, "Combining machine learning and domain decomposition methods for the solution of partial differential equations—A review," GAMM-Mitteilungen **44**, e202100001 (2021).
[34]A. D. Jagtap, E. Kharazmi, and 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**, 113028 (2020).
[35]A. D. J. Karniadakis and G. Em, "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).
[36]X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, "PPINN: Parareal physics-informed neural network for time-dependent pdes," Comput. Methods Appl. Mech. Eng. **370**, 113250 (2020).
[37]E. Kharazmi, Z. Zhang, and G. E. Karniadakis, "hp-VPINNS: Variational physics-informed neural networks with domain decomposition," Comput. Methods Appl. Mech. Eng. **374**, 113547 (2021).
[38]S. Dong and Z. Li, "Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations," Comput. Methods Appl. Mech. Eng. **387**, 114129 (2021).
[39]E. Schiassi, R. Furfaro, C. Leake, M. De Florio, H. Johnston, and D. Mortari, "Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations," Neurocomputing **457**, 334–356 (2021).
[40]H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa, "Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations," Phys. Fluids **34**, 075117 (2022).
[41]H. Xu, W. Zhang, and Y. Wang, "Explore missing flow dynamics by physics-informed deep learning: The parameterized governing systems," Phys. Fluids **33**, 095116 (2021).
[42]F. Pioch, J. H. Harmening, A. M. Müller, F.-J. Peitzmann, D. Schramm, and O. E. Moctar, "Turbulence modeling for physics-informed neural networks: Comparison of different RANS models for the backward-facing step flow," Fluids **8**, 43 (2023).
[43]G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the Spring Joint Computer Conference*, 18–20 April (Association for Computing Machinery, New York, 1967), pp. 483–485.
[44]J. L. Gustafson, "Reevaluating Amdahl's law," Commun. ACM **31**, 532–533 (1988).
[45]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning. I. Data-driven solutions of nonlinear partial differential equations," arXiv:1711.10561 (2017).
[46]C. Rao, H. Sun, and Y. Liu, "Physics-informed deep learning for incompressible laminar flows," Theor. Appl. Mech. Lett. **10**, 207–212 (2020).
[47]S. Xu, Z. Sun, R. Huang, D. Guo, G. Yang, and S. Ju, "A practical approach to flow field reconstruction with sparse or incomplete data through physics informed neural network," Acta Mech. Sin. **39**, 322302 (2023).
[48]B. Moseley, A. Markham, and T. Nissen-Meyer, "Finite basis physics-informed neural networks (FBPINNS): A scalable domain decomposition approach for solving differential equations," arXiv:2107.07871 (2021).
[49]C. Yan, S. Xu, Z. Sun, D. Guo, S. Ju, R. Huang, and G. Yang, "Exploring hidden flow structures from sparse data through deep-learning-strengthened proper orthogonal decomposition," Phys. Fluids **35**, 037119 (2023).

[50]M. Lauber, see https://github.com/marinlauber/2D-Turbulence-Python for "2d-turbulence-python, 2021;" accessed 15 February 2023.
[51]H. Jiang and L. Cheng, "Large-eddy simulation of flow past a circular cylinder for Reynolds numbers 400 to 3900," Phys. Fluids **33**, 034119 (2021).
[52]C. Norberg, "An experimental investigation of the flow around a circular cylinder: Influence of aspect ratio," J. Fluid Mech. **258**, 287 (1994).
[53]D. A. Lysenko, I. S. Ertesvåg, and K. E. Rian, "Large-eddy simulation of the flow over a circular cylinder at Reynolds number 3900 using the OpenFOAM toolbox," Flow Turbul. Combust. **89**, 491 (2012).
[54]M. M. Rahman, M. M. Karim, and M. A. Alim, "Numerical investigation of unsteady flow past a circular cylinder using 2-d finite volume method," J. Nav. Archit. Mar. Eng. **4**, 27–42 (2007).
[55]Z. Hu, A. D. Jagtap, G. E. Karniadakis, and K. Kawaguchi, "When do extended physics-informed neural networks (XPINNS) improve generalization?," arXiv:2109.09444 (2021).
[56]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980 (2014).
[57]I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," arXiv:1608.03983 (2016).