Acta Mechanica Sinica (2021) 37(12):1727–1738
https://doi.org/10.1007/s10409-021-01148-1

INVITED REVIEW

Physics-informed neural networks (PINNs) for fluid mechanics: a review

Shengze Cai[1] · Zhiping Mao[2] · Zhicheng Wang[3] · Minglang Yin[4,5] · George Em Karniadakis[1,4]

Received: 21 May 2021 / Accepted: 10 September 2021 / Published online: 23 January 2022

Abstract
Despite the significant progress over the last 50 years in simulating flow problems using numerical discretization of the Navier–Stokes equations (NSE), we still cannot incorporate seamlessly noisy data into existing algorithms, mesh-generation is complex, and we cannot tackle high-dimensional problems governed by parametrized NSE. Moreover, solving *inverse flow problems* is often prohibitively expensive and requires complex and expensive formulations and new computer codes. Here, we review *flow physics-informed learning*, integrating seamlessly data and mathematical models, and implement them using physics-informed neural networks (PINNs). We demonstrate the effectiveness of PINNs for inverse problems related to three-dimensional wake flows, supersonic flows, and biomedical flows.

Keywords Physics-informed learning · PINNs · Inverse problems · Supersonic flows · Biomedical flows

1 Introduction

In the last 50 years there has been a tremendous progress in computational fluid dynamics (CFD) in solving numerically the incompressible and compressible Navier–Stokes equations (NSE) using finite elements, spectral, and even meshless methods [1–4]. Yet, for real-world applications, we still cannot incorporate seamlessly (multi-fidelity) data into existing algorithms, and for industrial-complexity problems the mesh generation is time consuming and still an art. Moreover, solving *inverse problems*, e.g., for unknown boundary conditions or conductivities [5], etc., is often prohibitively expensive and requires different formulations and new computer codes. Finally, computer programs such as OpenFOAM [6] have more than 100,000 lines of code, making it almost impossible to maintain and update them from one generation to the next.

Physics-informed learning [7], introduced in a series of papers by Karniadakis's group both for Gaussian-process regression [8,9] and physics-informed neural networks (PINNs) [10], can seamlessly integrate multifidelity/multimodality experimental data with the various Navier–Stokes formulations for incompressible flows [11,12] as well as compressible flows [13] and biomedical flows [14]. PINNs use automatic differentiation to represent all the differential operators and hence there is no explicit need for a mesh generation. Instead, the Navier–Stokes equations and any other kinematic or thermodynamic constraints can be directly incorporated in the loss function of the neural network (NN) by penalizing deviations from the target values (e.g., zero residuals for the conservation laws) and are properly weighted with any given data, e.g., partial measurements of the surface pressure. PINNs are not meant to be a replacement of the existing CFD codes, and in fact the current generation of PINNs is not as accurate or as efficient as high-order CFD codes [2] for solving the standard forward

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin and George Em Karniadakis contributed equally to the paper

Executive Editor: Bernd Noack

✉ George Em Karniadakis
george_karniadakis@brown.edu

1 Division of Applied Mathematics, Brown University, Providence, RI 02912, USA

2 School of Mathematical Sciences Xiamen University, Xiamen 361005, China

3 Laboratory of Ocean Energy Utilization of Ministry of Education, Dalian University of Technology, Dalian 116024, China

4 School of Engineering, Brown University, Providence, RI 02912, USA

5 Center for Biomedical Engineering, Brown University, Providence, RI 02912, USA

problems. This limitation is associated with the minimization of the loss function, which is a high-dimensional non-convex function, a limitation which is a grand challenge of all neural networks for even commercial machine learning. However, PINNs perform much more accurately and more efficiently than any CFD solver if any scattered partial spatio-temporal data are available for the flow problem under consideration. Moreover, the forward and inverse PINN formulations are identical so there is no need for expensive data assimilation schemes that have stalled progress especially for optimization and design applications of flow problems in the past.

In this paper we first review the basic principles of PINNs and recent extensions using domain decomposition for multiphysics and multiscale flow problems. We then present new results for a three-dimensional (3D) wake formed in incompressible flow behind a circular cylinder. We also show results for a two-dimensional (2D) supersonic flow past a blunt body, and finally we infer material parameters in simulating thrombus deformation in a biomedical flow.

2 PINN

In this section we first review the basic PINN concept and subsequently discuss more recent advancements in incompressible, compressible and biomedical flows.

2.1 PINNs: basic concepts

We consider a parametrized partial differential equation (PDE) system given by:

$$\begin{aligned} &f(\mathbf{x}, t, \hat{u}, \partial_{\mathbf{x}}\hat{u}, \partial_t\hat{u}, \dots, \boldsymbol{\lambda}) = 0, \quad \mathbf{x} \in \Omega, \ t \in [0, T], \\ &\hat{u}(\mathbf{x}, t_0) = g_0(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ &\hat{u}(\mathbf{x}, t) = g_\Gamma(t), \quad \mathbf{x} \in \partial\Omega, \ t \in [0, T], \end{aligned} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the spatial coordinate and t is the time; f denotes the residual of the PDE, containing the differential operators (i.e., $[\partial_{\mathbf{x}}\hat{u}, \partial_t\hat{u}, \dots]$); $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \dots]$ are the PDE parameters; $\hat{u}(\mathbf{x}, t)$ is the solution of the PDE with initial condition $g_0(\mathbf{x})$ and boundary condition $g_\Gamma(t)$ (which can be Dirichlet, Neumann or mixed boundary condition); Ω and $\partial\Omega$ represent the spatial domain and the boundary, respectively.

In the context of the vanilla PINNs [15], a fully-connected feed-forward neural network, which is composed of multiple hidden layers, is used to approximate the solution of the PDE $\hat{u}$ by taking the space and time coordinates $(\mathbf{x}, t)$ as inputs, as shown in the blue panel in Fig. 1. Let the hidden variable of the k^{th} hidden layer be denoted by $\mathbf{z}^k$, then the neural network can be expressed as

$$\begin{aligned} &\mathbf{z}^0 = (\mathbf{x}, t), \\ &\mathbf{z}^k = \sigma(\mathbf{W}^k\mathbf{z}^{k-1} + \mathbf{b}^k), \quad 1 \le k \le L-1, \\ &\mathbf{z}^k = \mathbf{W}^k\mathbf{z}^{k-1} + \mathbf{b}^k, \quad k = L, \end{aligned} \tag{2}$$

where the output of the last layer is used to approximate the true solution, namely $\hat{u} \approx \mathbf{z}^L$. $\mathbf{W}^k$ and $\mathbf{b}^k$ denote the weight matrix and bias vector of the k^{th} layer; $\sigma(\cdot)$ is a nonlinear activation function. All the trainable model parameters, i.e., weights and biases, are denoted by θ in this paper.

In PINNs, solving a PDE system (denoted by Eq. (1)) is converted into an optimization problem by iteratively updating θ with the goal to minimize the loss function L:

$$L = \omega_1 L_{\text{PDE}} + \omega_2 L_{\text{data}} + \omega_3 L_{\text{IC}} + \omega_4 L_{\text{BC}}, \tag{3}$$

where ω_{1-4} are the weighting coefficients for different loss terms. The first term L_{PDE} in Eq. 3 penalizes the residual of the governing equations. The other terms are imposed to satisfy the model predictions for the measurements L_{data}, the initial condition L_{IC}, and the boundary condition L_{BC}, respectively. In general, the mean square error (MSE), taking the L_2-norm of the sampling points, is employed to compute the losses in Eq. 3. The sampling points are defined as a data set $\{\mathbf{x}^i, t^i\}_{i=1}^N$, where the number of points (denoted by N) for different loss terms can be different. Generally, we use the ADAM optimizer [16], an adaptive algorithm for gradient-based first-order optimization, to optimize the model parameters θ.

Remark 1 We note that the definition of the loss function shown in Eq. (3) is problem-dependent, hence some terms may disappear for different types of the problem. For example, when we solve a forward problem in fluid mechanics with the known parameters ($\breve{}$) and the initial/boundary conditions of the PDEs, the data loss L_{data} is not necessarily required. However, in the cases where the model parameters or the initial/boundary conditions are unknown (namely, inverse problems), the data measurements should be taken into account in order to make the optimization problem solvable. We also note that the PINN framework can be employed to solve an "over-determined" system, e.g., well-posed in a classical sense with initial and boundary conditions known and additionally some measurements inside the domain or at boundaries (e.g., pressure measurements).

One of the key procedures to construct the PDE loss in Eq. (3) is the computation of partial derivatives, which is addressed by using automatic differentiation (AD). Relying on the combination of the derivatives for a sequence of operations by using the chain rule, AD calculates the derivatives of the outputs with respect to the network inputs directly in the computational graph. The computation of partial derivatives can be calculated with an explicit expression, hence avoiding introducing discretization and truncation errors as

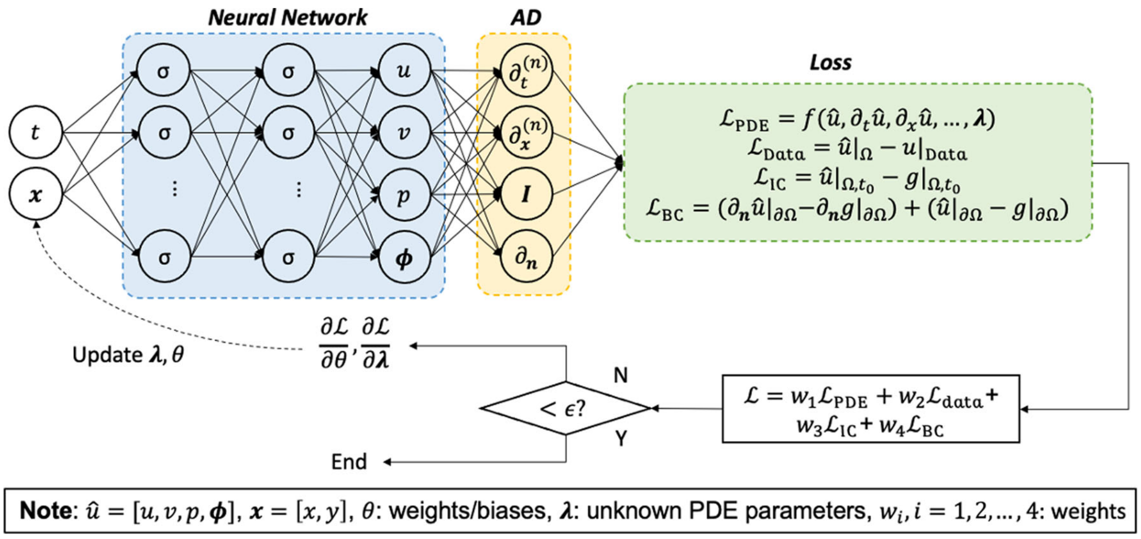
Neural Network
AD
Loss
t
x
σ
u
v
p
ϕ
$\partial_t^{(n)}$
$\partial_x^{(n)}$
I
∂_n
$\mathcal{L}_{\mathrm{PDE}} = f(\hat{u}, \partial_t\hat{u}, \partial_x\hat{u}, \dots, \boldsymbol{\lambda})$
$\mathcal{L}_{\mathrm{Data}} = \hat{u}|_{\Omega} - u|_{\mathrm{Data}}$
$\mathcal{L}_{\mathrm{IC}} = \hat{u}|_{\Omega,t_0} - g|_{\Omega,t_0}$
$\mathcal{L}_{\mathrm{BC}} = (\partial_n\hat{u}|_{\partial\Omega} - \partial_n g|_{\partial\Omega}) + (\hat{u}|_{\partial\Omega} - g|_{\partial\Omega})$
Update $\boldsymbol{\lambda}, \theta$
$\frac{\partial\mathcal{L}}{\partial\theta}, \frac{\partial\mathcal{L}}{\partial\boldsymbol{\lambda}}$
N
$< \epsilon?$
Y
End
$\mathcal{L} = w_1\mathcal{L}_{\mathrm{PDE}} + w_2\mathcal{L}_{\mathrm{data}} + w_3\mathcal{L}_{\mathrm{IC}} + w_4\mathcal{L}_{\mathrm{BC}}$
Note: $\hat{u} = [u, v, p, \boldsymbol{\phi}]$, $\boldsymbol{x} = [x, y]$, θ: weights/biases, $\boldsymbol{\lambda}$: unknown PDE parameters, $w_i, i = 1, 2, \dots, 4$: weights

Fig. 1 Schematic of a physics-informed neural network (PINN). A fully-connected neural network, with time and space coordinates $(t, \mathbf{x})$ as inputs, is used to approximate the multi-physics solutions $\hat{u} = [u, v, p, \phi]$. The derivatives of $\hat{u}$ with respect to the inputs are calculated using automatic differentiation (AD) and then used to formulate the residuals of the governing equations in the loss function, that is generally composed of multiple terms weighted by different coefficients. The parameters of the neural network θ and the unknown PDE parameters λ can be learned simultaneously by minimizing the loss function

in conventional numerical methods. However, as the PDE and its derivatives are parameterized by neural networks in PINNs, we note that there may exist generalization error and optimization error depending on the training data and the optimizer, respectively [17]. At the present time, AD has been implemented in various deep learning frameworks [18,19], which makes it convenient for the development of PINNs.

A schematic of PINNs is shown in Fig. 1, where the key elements (e.g., neural network, AD, loss function) are indicated in different colors. Here, we consider a multi-physics problem, where the solutions include the velocity (u, v), pressure p and a scalar field ϕ, which are coupled in a PDE system f. The schematic in Fig. 1 represents most of the typical problems in fluid mechanics. For instance, the PDEs considered here can be the Boussinesq approximation of the Navier–Stokes equations, where ϕ is the temperature. Following the paradigm in Fig. 1, we will describe the governing equations, the loss function and the neural network configurations of PINNs case-by-case in the rest of this paper.

2.2 Recent advances of PINNs

First proposed in Refs. [20,21], see also Ref. [15], PINNs have attracted a lot of attention in the scientific computing community as well as the fluid mechanics community. Here, we review some related works regarding the methodology and the application to fluid mechanics.

Beneficial due to the high flexibility and the expressive ability in function approximation, PINNs have been extended to solve various classes of PDEs, e.g., integro-differential equations [22], fractional equations [22], surfaces PDEs [23] and stochastic differential equations [24]. A variational formulation of PINNs based on the Galerkin method (hp-VPINN) was proposed to deal with PDEs with non-smooth solutions [25]. In addition, the variational hp-VPINN considered domain decomposition, and similar pointwise versions were also studied in CPINN [26] and XPINN [27]. A general parallel implementation of PINNs with domain decomposition for flow problems is presented in [28]; the NVIDIA library SimNet [29] is also a very efficient implementation of PINNs. Another important extension is the uncertainty quantification for the PDE solutions inferred by neural networks [30–34]. This has been studied by using the Bayesian framework [34]. Moreover, some other researches on PINNs focused on the development of the neural network architecture and the training, e.g., using multi-fidelity framework [35], adaptive activation functions [36] and dynamic weights of the loss function [37], hard constraints [38] and CNN-based network architectures [39], which can improve the performance of PINNs on different problems. On the theoretical side, some recent works [17,40–42] have provided more guarantees and insights into the convergence of PINNs. In particular, the authors in Refs. [17] and [40] both state that the approximation to the PDE solution by PINNs converges as the number of training data increases. It is also reported in Refs. [40,41] that the upper bound of the generalization error of PINNs can be quantified, which is dependent on the training error, the number of training samples and some constants from the stability estimate. These analyses have covered a wide range of linear PDEs with concrete examples in both forward and inverse problems.

The development of the methodology has inspired a number of applications in other fields, especially in fluid mechanics where the flow phenomena can be described by the NSE. In Ref. [15], the vanilla PINN was proposed to infer the unknown parameters (e.g., the coefficient of the convection term) in the NS equations based on velocity measurements for the 2D flow over a cylinder. Following this work, PINNs were then applied to various flows [10–14,43–50], covering the applications on compressible flows [13], biomedical flows [14,43,51], turbulent convection flows [49], free boundary and Stefan problems [48], etc. The main attractive advantage of PINNs in solving fluid mechanics problems is that a unified framework (shown in Fig. 1) can be used for both forward and inverse problems. Compared to the traditional CFD solvers, PINNs are superior at integrating the data (observations of the flow quantities) and physics (governing equations). A promising application is on the flow visualization technology [12,52], where the flow fields can be easily inferred from the observations such as concentration fields and images. On the contrary, such inverse problems are difficult for conventional CFD solvers. More relevant works on mechanics in general can be found in references: [53] for turbulent flow, [54] for phase-field fracture model, and [55] for inferring modulus in a nonhomogeneous material.

3 Case study for 3D incompressible flows

In this section, we demonstrate the effectiveness of PINNs for solving inverse problems in incompressible flows. In particular, we apply PINNs to reconstruct the 3D flow fields based on a few two-dimensional and two-component (2D2C) velocity observations. The proposed algorithm is able to infer the full velocity and pressure fields very accurately with limited data, which is promising for diagnosis of complex flows when only 2D measurements (e.g., planar particle image velocimetry) are available.

3.1 Problem setup

We consider the 3D wake flow past a stationary cylinder at Reynolds number $Re = 200$ in this section. In order to evaluate the performance of PINNs, we generate the reference solution numerically by using the spectral/*hp* element method [2]. The computational domain is defined as Ω: $[-7.5, 28.5] \times [-20, 20] \times [0, 12.5]$, where the coordinates are non-dimensionalized by the diameter of the cylinder. The center of the cylinder is located at $(x, y) = (0, 0)$. We assume that the velocity ($u = 1$) is uniform at the inflow boundary where $x = -7.5$. A periodic boundary condition is used at the lateral boundaries where $y = \pm 20$, and the zero-pressure is prescribed at the outlet where $x = 28.5$. Moreover, the no-slip boundary condition is imposed on the cylinder surface. The governing equations in this case study are the dimensionless incompressible Navier–Stokes equations with $Re = 200$. Under this configuration, the wake flow over the cylinder is 3D and unsteady. The simulation is performed until the vortex shedding flow becomes stable. At the end, the time-dependent data are collected for PINN training and evaluation.

The simulation results of the 3D flow are shown in Fig. 2, where Fig. 2a shows the iso-surface of streamwise vorticity ($\omega_x = -0.3$) color-coded with the streamwise velocity u. In this section, we are interested in the 3D flow reconstruction problem from limited data, and we only focus on a subdomain in the wake flow, namely Ω_s: $[1.5, 7.5] \times [-3, 3] \times [4, 9]$, which is represented by a cube with blue edges in Fig. 2a. The contours of the three velocity components and pressure field are shown in Fig. 2b. An Eulerian mesh with $61 \times 61 \times 26$ grid points is used for plotting. To demonstrate the unsteadiness of the motion, we consider 50 snapshots with $\Delta t = 0.2$, which cover about two periods of the vortex shedding cycle.

Here, we aim to apply PINNs for reconstructing the 3D flow field from the velocity observations of a few 2D planes. As illustrated in Fig. 3, three different "experimental" setups are considered in this paper:

- Case 1: two x-planes ($x = 1.5, 7.5$), one y-plane ($y = 0$) and two z-planes ($z = 4.0, 9.0$) are observed.
- Case 2: two x-planes ($x = 1.5, 7.5$), one y-plane ($y = 0$) and one z-plane ($z = 6.4$) are observed.
- Case 3: one x-plane ($x = 1.5$), one y-plane ($y = 0$) and one z-plane ($z = 6.4$) are observed.

We note that for these cross-planes, only the projected vectors (two components) are considered known. For example, the velocities (u, v) can be observed on the z-plane, while the orthogonal component (w) is unknown. The purpose of doing this is to mimic the planar particle image velocimetry in real experiments. Moreover, the resolutions of these 2D2C observations are different, which can be found in Table 1.

3.2 Implementation of PINNs

Given the observation data on the cross-planes, we train a PINNs model to approximate the flow fields over the space-time domain. The PINN in this section take $(\mathbf{x}, t) = (x, y, z, t)$ as inputs and outputs the velocity and pressure (u, v, w, p). The loss function in PINN can be defined as:

$$L = L_{\text{data}} + L_{\text{PDE}}, \tag{4}$$

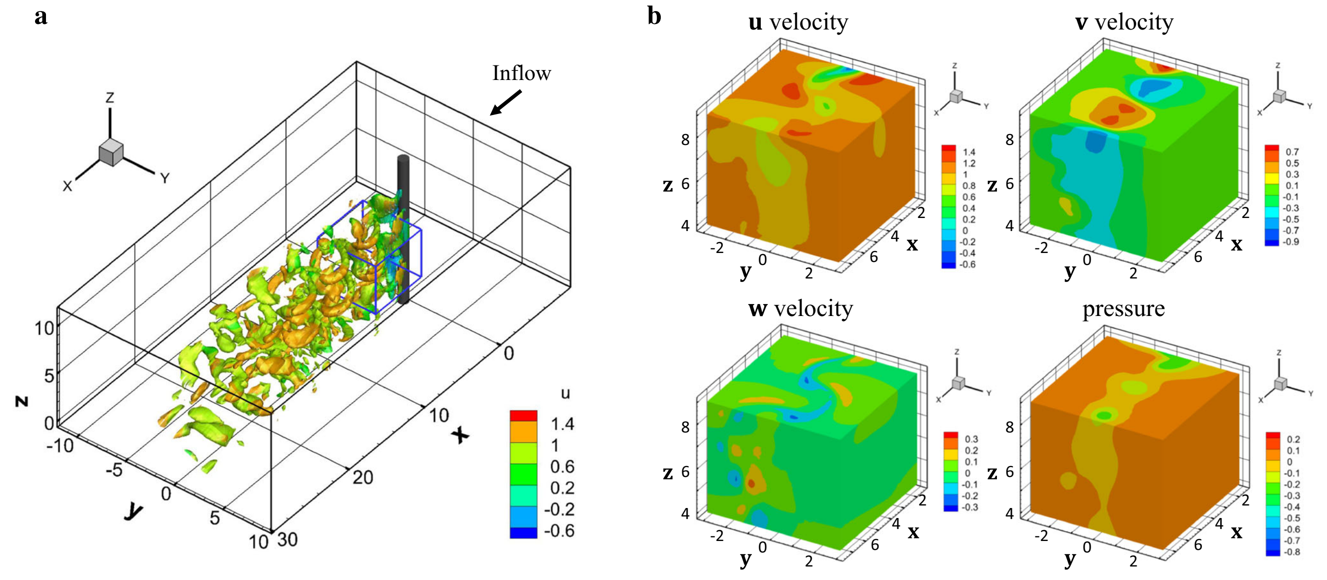

Fig. 2 Case study of PINNs for incompressible flows: illustration of simulating the 3D wake flow over a circular cylinder. **a** Iso-surface of the vorticity (x-component) in the whole domain color-coded by the streamwise velocity. The cube with blue edges represents the computational domain in this case. **b** Velocity and pressure fields in the domain. The simulation was performed by the CFD solver Nektar, which is based on the spectral/*hp* element method [2]

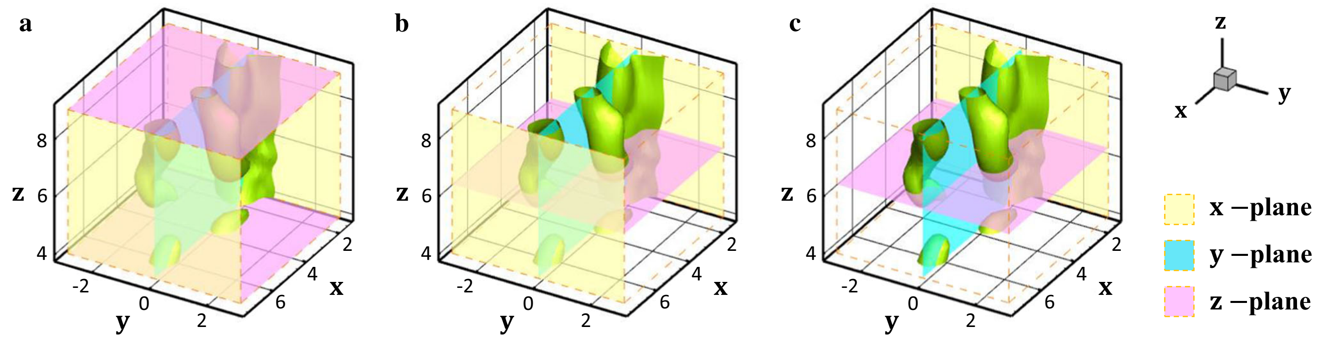

Fig. 3 Case study of PINNs for incompressible flows: problem setup for 3D flow reconstruction from 2D2C observations. **a** Case 1: two x-planes ($x = 1.5, 7.5$), one y-plane ($y = 0$) and two z-planes ($z = 4.0, 9.0$) are observed. **b** Case 2: two x-planes ($x = 1.5, 7.5$), one y-plane ($y = 0$) and one z-plane ($z = 6.4$) are observed. **c** Case 3: one x-plane ($x = 1.5$), one y-plane ($y = 0$) and one z-plane ($z = 6.4$) are observed. Note that for the cross-planes, only the projected vectors are measured. The goal is to infer the 3D flow in the investigated domain using PINNs from these 2D2C observations

where

$$\begin{aligned} L_{\text{data}} =& \frac{1}{N_u}\sum_i^{N_u} \| u(\mathbf{x}^i_{\text{data}}, t^i_{\text{data}}) - u^i_{\text{data}} \|_2^2 \\ &+ \frac{1}{N_v}\sum_i^{N_v} \| v(\mathbf{x}^i_{\text{data}}, t^i_{\text{data}}) - v^i_{\text{data}} \|_2^2 \\ &+ \frac{1}{N_w}\sum_i^{N_w} \| w(\mathbf{x}^i_{\text{data}}, t^i_{\text{data}}) - w^i_{\text{data}} \|_2^2, \end{aligned} \tag{5}$$

and

$$L_{\text{PDE}} = \frac{1}{N_f}\sum_i^{N_f}\sum_j^{4} \| f_j(\mathbf{x}^i_f, t^i_f) \|_2^2, \tag{6}$$

$$\begin{aligned} f_{1,2,3} &= \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \frac{1}{Re}\nabla^2\mathbf{u}, \\ f_4 &= \nabla\cdot\mathbf{u}. \end{aligned}$$

Table 1 Case study of PINNs for incompressible flows: details of the 2D2C observations

Cross-section	Observed velocity components	Observed spatial resolution
x-plane	(v, w)	61×26
y-plane	(u, w)	61×26
z-plane	(u, v)	61×61

The data loss L_{data} is composed of three components, and the number of training data (namely N_u, N_v and N_w) depends on the number of observed planes, the data resolution of each plane as well as the number of snapshots. On the other hand, the residual points for L_{PDE} can be randomly selected, and here we sample $N_f = 3 \times 10^6$ points over the investigated space and time domain Ω_s. Note that in this study, the boundary and initial conditions are not required unlike the classical setting. Moreover, no information about the pressure is given. The weighting coefficients for the loss terms are all equal to 1.

A fully-connected neural network with 8 hidden layers and 200 neurons per layer is employed. The activation function of each neuron is $\sigma = \sin(\cdot)$. We apply the ADAM optimizer with mini-batch for network training, where a batch size of $N = 10^4$ is used for both data and residual points. The network is trained for 150 epochs with learning rates 1×10^{-3}, 5×10^{-4} and 1×10^{-4} for every 50 epochs. After training, the velocity and pressure fields are evaluated on the Eulerian grid for comparison and visualization.

3.3 Inference results

For a quantitative assessment, we first define the relative L_2-norm error as the evaluation metric, which is expressed as:

$$\epsilon_V = \frac{\| V_{\text{CFD}} - \hat{V} \|_2}{\| V_{\text{CFD}} \|_2} \times 100\%, \tag{7}$$

where $V \in \{u, v, w, p\}$; V_{CFD} and $\hat{V}$ account for the CFD data and the output of PINNs, respectively, $\| \cdot \|_2$ denotes the L_2-norm. We compute the errors in the investigated time domain, which are shown in Fig. 4. It can be seen from the plots that PINNs can infer the 3D flow very accurately for Case 1 and Case 2; using 5 planes (Case 1) is slightly better than using 4 planes (Case 2). When only 3 cross-planes are available (Case 3), the errors become much larger. However, the result of Case 3 is still acceptable as we are able to infer the main flow features with high accuracy (the error of the streamwise velocity is mostly less than 2%). We note that the errors of w-velocity are larger than the other components since the w-velocity magnitude is relatively small. Moreover, we observe larger discrepancy for the initial and final time instants, which can be attributed to the lack of training data for computing derivatives at $t < 0$ and $t > 10$. This generally happens in the cases when the initial condition is not provided in the unsteady case [12].

To visualize more details, we emphasize the result of Case 2 and demonstrate the iso-surface of vorticity magnitude and iso-surfaces of pressure at $t = 8.0$ in Fig. 5, where Fig. 5a shows the reference CFD data and Fig. 5b shows the result inferred by PINNs. The vorticity value is $|\omega| = 1.2$ and the color represents the streamwise velocity component. It can be seen that the PINNs inference result (inferred from a few 2D2C observations) is very consistent with the CFD simulation. In addition, the velocities (u, v) at a single point $(x = 3, y = 0, z = 6.4)$ against time are plotted in Fig. 5c, where we can find that PINNs can capture the unsteadiness of vortex shedding flow very accurately.

4 Case study for compressible flows

PINNs have also been used in simulating high-speed flows [13]. In this section, we consider the following 2D steady compressible Euler equations:

$$\nabla \cdot f(U) = 0, \;\; x \in \Omega \subset \mathbb{R}^2, \tag{8}$$

where

$$U = [\rho, \rho u, \rho v, \rho E]^{\text{T}}, \;\; f = (G_1, G_2),$$

with $G_1(U) = [\rho u, p + \rho u^2, \rho uv, pu + \rho uE]$, $G_2(U) = [\rho v, \rho uv, p + \rho v^2, pv + \rho vE]$. Here, ρ is the density, p is the pressure, $[u, v]$ are the velocity components, and E is the total energy. We use the additional equation of state, which describes the relation of the pressure and energy, to close the above Euler equations. For instance, we consider the equation of state for a polytropic gas given by

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho \|\boldsymbol{u}\|^2\right), \tag{9}$$

where γ is the adiabatic index and $\boldsymbol{u} = (u, v)$.

We shall employ the PINNs to solve the inverse problem of the compressible Euler Eq. (8). In particular, we shall infer the density, pressure and velocity fields by using PINNs based on the information of density gradients, limited data of pressure (pressure on the surface of the body), inflow conditions and global physical constrains.

4.1 Problem setup

We consider a 2D bow shock wave problem. For traditional CFD simulations, it is crucial that the boundary conditions, which play an important role, are properly implemented.

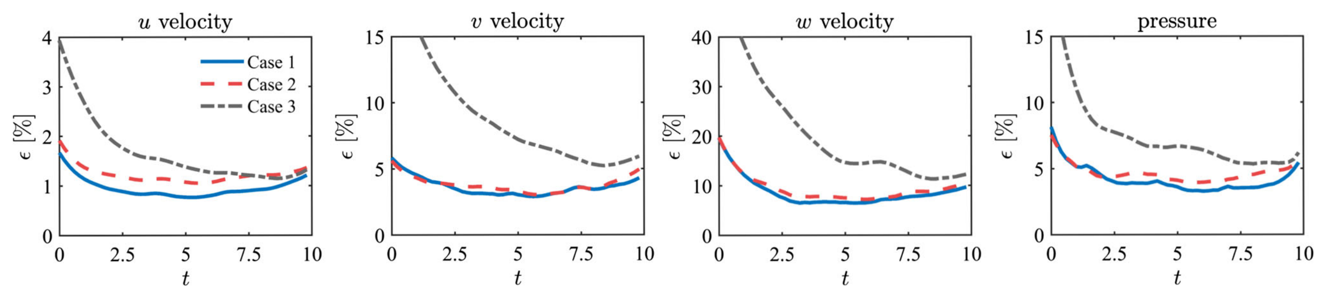

Fig. 4 Case study of PINNs for incompressible flows: relative L_2-norm errors of velocities and pressure for different flow reconstruction setups. These three cases correspond to those shown in Fig. 3. The errors are computed over the entire investigated domain

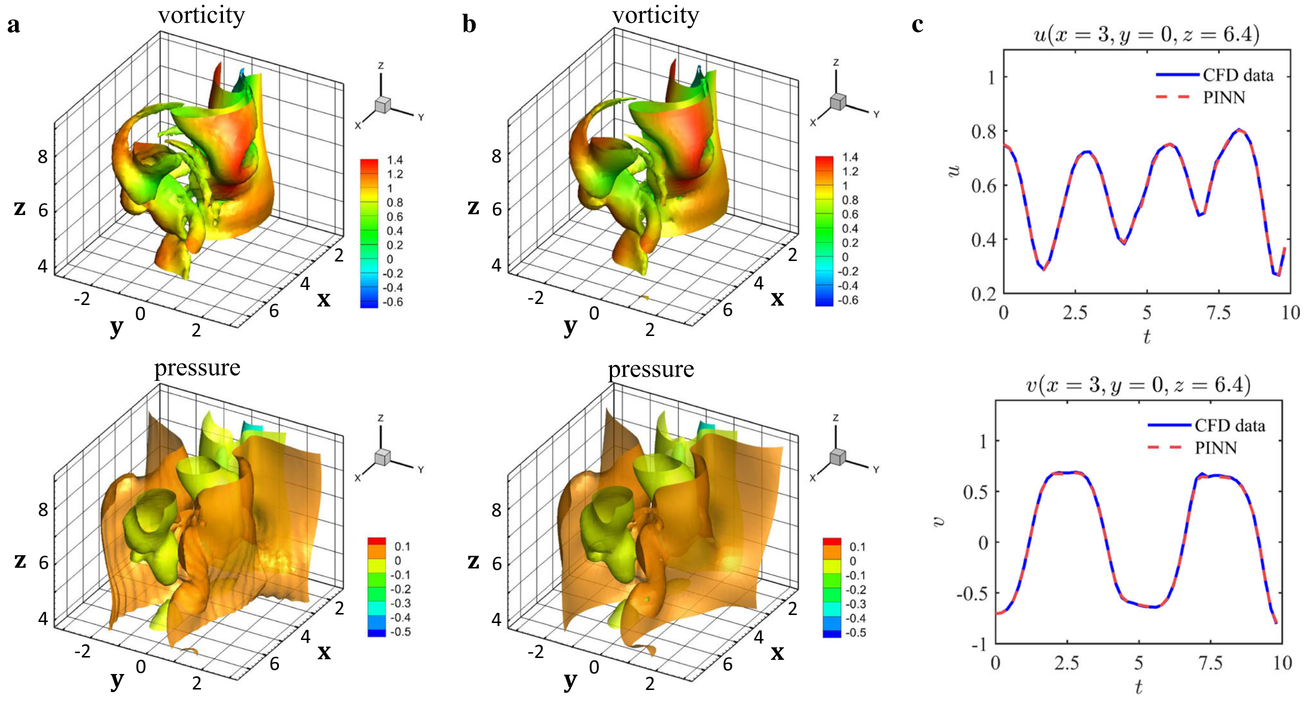

Fig. 5 Case study of PINNs for incompressible flows: inference result of PINNs for Case 2. **a** Iso-surfaces of vorticity magnitude (top) and pressure (bottom) at $t = 8.0$ from CFD data. **b** Iso-surfaces of vorticity magnitude (top) and pressure (bottom) at $t = 8.0$ inferred by PINNs. **c** Point measurement $(x = 3, y = 0, z = 6.4)$ of velocity (u, v) against time. In this case, the 3D flow is inferred by PINNs from four cross-planes

However, for the shock wave problem in high-speed flows, the boundary conditions in real experiments are usually not known and can only be estimated approximately. In the present work, instead of using most of the boundary conditions required by the traditional CFD simulation, we solve the Euler Eq. (8) using PINNs based on the data of density gradients $\nabla\rho$ motivated by the Schlieren photography available experimentally; additionally, we use limited data of the surface pressure obtained by pressure sensors as well as the global constrains (mass, momentum and energy). We also use the inflow conditions here. Note that unlike the one-dimensional case, where the density gradient in the whole computational domain is used in Ref. [13], here we only use the density gradients in a sub-domain D of the computational domain Ω, i.e., $D \subset \Omega$. By combining the mathematical model and the given data, we have the weighted loss function of PINN given by

$$Loss = Loss_{\mathrm{PDE}} + Loss_{\mathrm{Constrain}} + Loss_{\mathrm{data}}, \tag{10}$$

where

$$\begin{aligned} Loss_{\mathrm{PDE}} &= \omega_1 Loss_{\nabla\cdot f(U)}, \\ Loss_{\mathrm{Constrain}} &= \omega_2 \left(Loss_{\mathrm{Mass}} + Loss_{\mathrm{Momentum}} \right. \\ &\quad \left. + Loss_{\mathrm{Energy}}\right), \\ Loss_{\mathrm{data}} &= \omega_3 Loss_{\nabla\rho|_D} + \omega_4 Loss_{\mathrm{inflow}} + \omega_5 Loss_{p^*} \\ &\quad + \omega_6 Loss_{n\cdot\boldsymbol{u}}, \end{aligned}$$

here $Loss_{\nabla\cdot f(U)}$ corresponds to the Eq. (8), $Loss_{\mathrm{Mass}}$, $Loss_{\mathrm{Momentum}}$ and $Loss_{\mathrm{Energy}}$ correspond to the total mass,

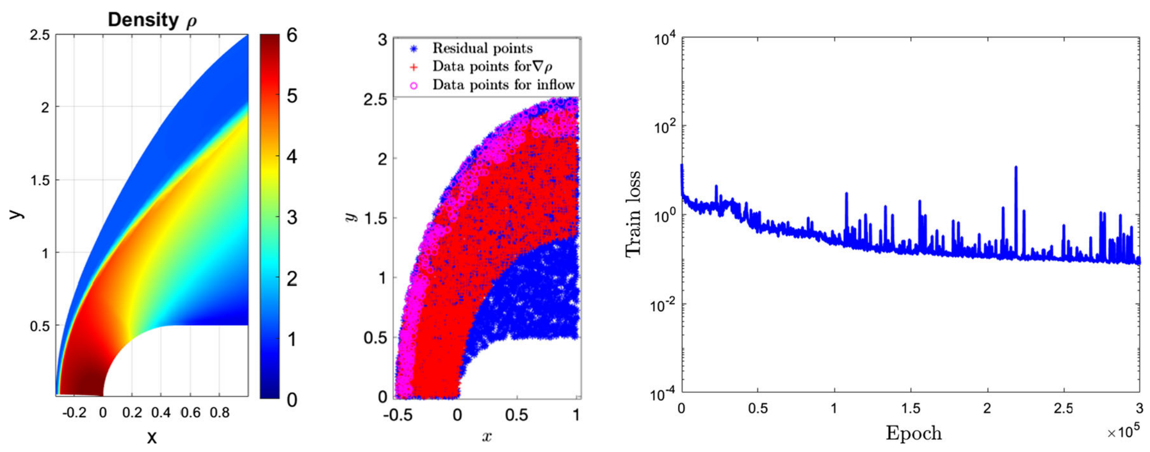

Fig. 6 Case study of PINNs for compressible flows. Left: the density obtained by using CFD simulation with the inlet flow condition (11). Middle: Distributions of the residual points (blue $*$ points), the data points for the density gradient (red $+$ points) and the data points for the inflow conditions (magenta $\circ$ points). Right: training loss vs. number of epochs

momentum and energy, $Loss_{\nabla\rho|_D}$ corresponds to the data of density gradients in the sub-domain $D \subset \Omega$, $Loss_{\text{inflow}}$ corresponds to the inlet flow conditions, $Loss_{p^*}$ corresponds to the data of pressure on the body surface, $Loss_{n\cdot\boldsymbol{u}}$ corresponds to the velocity condition on the body surface. The number of residual points for the equation is about 7000, the number of data points for the density gradients is about 3000, the number of data points for the pressure on the surface is 40, the number of data points for the inlet flow conditions is about 800. The number of points for the last term is 31.

4.2 Inference results

To demonstrate the effectiveness of PINNs for compressible flows, we consider the bow shock problem with the following inlet flow conditions

$$M_\infty = 4,\ \ p_\infty = 101253.6\,\text{Pa},\ \ \rho_\infty = 1.225\,\text{kg/m}^3,\\ u_\infty = 1360.6963\,\text{m/s},\ \ v_\infty = 0,\ \ T_\infty = 288\,\text{K}. \tag{11}$$

The data points for the pressure are located on the surface of the body. By using the above inflow conditions and CFD code, we can obtain the steady state flow. We show the density computed by CFD in the left plot of Fig. 6. We employ a 6×60 (6 hidden layers) neural network and train it by using layer-wise adaptive tanh activate function [36] and the Adam optimizer with the learning rate being 6×10^{-4} for 3×10^5 epochs. Here, we also use the technique of dynamic weights [11,37]. The history of the training loss is shown in the right plot of Fig. 6. The results of the PINN solutions for the pressure and velocity (u) are shown in Fig. 7. Observe that the PINN solutions are in good agreement with the CFD data. This indicates that we can reconstruct the flow fields for high-speed flows using some other available knowledge

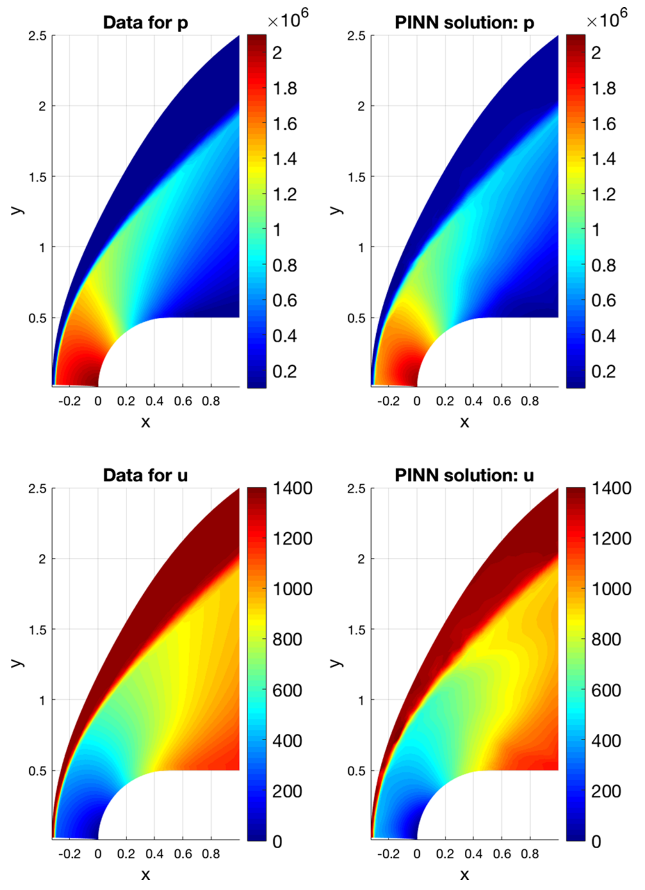

Fig. 7 Case study of PINNs for compressible flows. Comparison between the PINN solutions and the CFD solutions. Top: pressure p, Bottom: velocity component u

different from the boundary conditions required by the traditional CFD simulation.

5 Case study for biomedical flows

In addition to the aforementioned flow examples, PINNs have also been used in biomedical flows [14]. In this section, we consider inferring material properties of a thrombus in arterial flow described by the Navier–Stokes and Cahn–Hilliard equations, which characterize the mechanical interaction between thrombus and the surrounding flow. The PDE system is written as:

$$\rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}) + \nabla p = \nabla \cdot (\boldsymbol{\sigma}_{\mathbf{vis}} + \boldsymbol{\sigma}_{\mathbf{coh}}) - \mu \frac{(1-\phi)\mathbf{u}}{2\kappa(\phi)}, \tag{12}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{13}$$

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = \tau \Delta \omega, \tag{14}$$

$$\omega = \Delta \phi + \gamma g(\phi), \tag{15}$$

$g(\phi)$ is the derivative of the double-well potential $(\phi^2 - 1)^2/4h^2$. Variables, i.e., $\mathbf{u}(\mathbf{x}, t)$, $p(\mathbf{x}, t)$ $\boldsymbol{\sigma}(\mathbf{x}, t)$, and $\phi(\mathbf{x}, t)$, represent the velocity, pressure, stress, and phase field. Viscous and cohesive stresses are defined as:

$$\boldsymbol{\sigma}_{\mathbf{vis}} = \mu \nabla u, \tag{16}$$

$$\boldsymbol{\sigma}_{\mathbf{coh}} = \lambda \nabla \cdot (\nabla \phi \otimes \phi). \tag{17}$$

In particular, we set the density $\rho = 1$, viscosity $\mu = 0.1$, $\lambda = 4.2428 \times 10^{-5}$, $\tau = 10^{-6}$, and the interface length $h = 0.05$. We impose a Dirichlet boundary condition for the velocity at inlet Γ_i as $\mathbf{u} = g$, $(\mathbf{x}, t) \in \Gamma_i \times (0, T)$. A Neumann-type boundary condition, $\frac{\partial}{\partial \mathbf{n}} = 0$ for $\mathbf{x} \in \Gamma_w \cup \Gamma_i \cup \Gamma_o$ is set for ϕ and ω at all boundaries. A more detailed description of the governing equations can be found in Refs. [14,56].

5.1 PINNs

We construct two fully-connected neural networks, *Net U* and *Net W*, where the former predicts *u, v, p*, and ϕ and the latter predicts the intermediate variable, ω. It is preferable to write the governing equation as a system of PDEs since computing the fourth-order partial derivative on ϕ is decomposed into computing second-order partial derivations on ϕ and ω. Both networks have 20 neurons per layer and 9 hidden layers.

The total loss L is a linear combination of losses from data, initial and boundary condition, and PDE residuals:

$$L = \omega_1 L_{\text{PDE}} + \omega_2 L_{\text{IC}} + \omega_3 L_{\text{BC}} + \omega_4 L_{\text{data}}, \tag{18}$$

where each term is written as:

$$L_{\text{PDE}}(\theta, \boldsymbol{\lambda}; X_{\text{PDE}}) = \frac{1}{|X_{\text{PDE}}|} \sum_{\mathbf{x} \in X_{\text{PDE}}} \left\| f(\mathbf{x}, \partial_t \hat{\mathbf{u}}, \partial_x \hat{\mathbf{u}}, ...; \boldsymbol{\lambda}) \right\|_2^2, \tag{19}$$

$$L_{\text{BC}}(\theta, \boldsymbol{\lambda}; X_{\text{BC}}) = \frac{1}{|X_{\text{BC}}|} \sum_{\mathbf{x} \in X_{\text{BC}}} B(\hat{\mathbf{u}}, \mathbf{x})_2^2, \tag{20}$$

$$L_{\text{IC}}(\theta, \boldsymbol{\lambda}; X_{\text{IC}}) = \frac{1}{|X_{\text{IC}}|} \sum_{\mathbf{x} \in X_{\text{IC}}} \left\| \hat{\mathbf{u}} - \mathbf{u}_{t_0} \right\|_2^2, \tag{21}$$

$$L_{\text{data}}(\theta, \boldsymbol{\lambda}; X_{\text{data}}) = \frac{1}{|X_{\text{data}}|} \sum_{\mathbf{x} \in X_{\text{data}}} \left\| \hat{\mathbf{u}} - \mathbf{u}_{\text{data}} \right\|_2^2, \tag{22}$$

We set ω_i as the weights of each term and randomly sample points as the training sets X_{PDE}, X_{BC}, and X_{IC} from the inner spatio-temporal domain, boundaries, and initial snapshot, respectively. In particular, we have point measurements, X_{data}, as the known data for minimizing the data loss; $|\cdot|$ indicates the size of the set. Finally, the optimal $\boldsymbol{\theta}$ and $\boldsymbol{\lambda} = [\kappa]$ are obtained by minimizing the total loss $L(\boldsymbol{\theta}, \boldsymbol{\lambda})$ iteratively until the loss satisfies the stopping criteria.

5.2 Problem setup

Herein, we consider a 2D permeable thrombus in a channel with a flow coming from the left (Fig. 8). The thrombus has two layers, representing an idealized scenario where a fibrin clot is coated by a permeable shell (consisting of loosely-packed and partially-activated platelets) [56–58]. We aim at inferring the unknown permeability for each constituent and

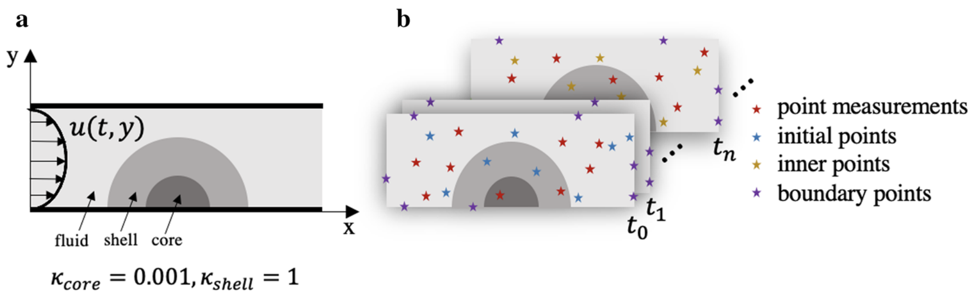

Fig. 8 Case study for 2D flow past a thrombus with phase-dependent permeability. **a** the inlet flow $u(t, y)$ (denoted by $\phi = 1$) enters the channel from the left side. A two-layer thrombus is present at the bottom wall with a fibrin-clotted core $\phi = -1$ and permeable shell $\phi = 0$. Their permeabilities are set as 0.001 and 1. **b** Four types of sampling points. Initial points (★), inner points (★), boundary points (★), and point measurements (★) are sampled accordingly in the spatio-temporal domain. (Figure adapted from [14])

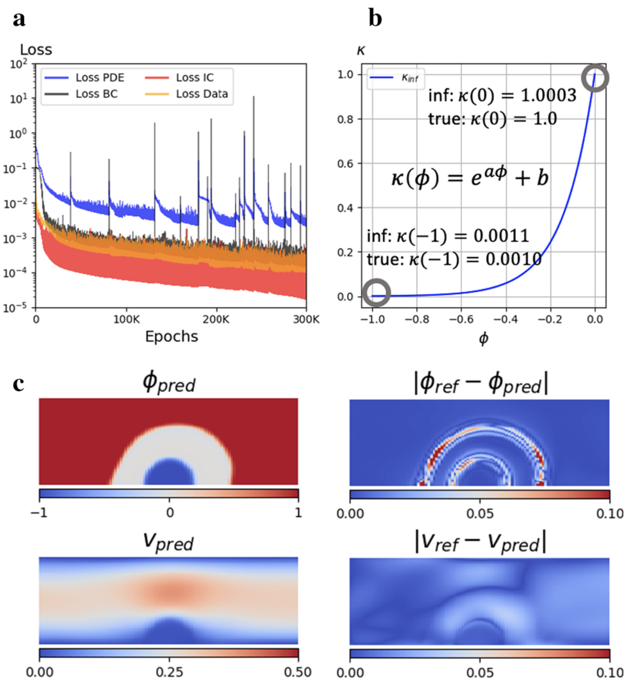

Fig. 9 Inference results for 2D flow past a thrombus with phase-dependent permeability. **a** History of training losses (Loss PDE, Loss IC, Loss BC, and Loss Data) and **b** inferred relation function $\kappa = f(\phi)$. **c** ϕ_{pred} and v_{pred} are plotted against their absolute error at $t = 0.78$. We draw 10,000 data points from 30 snapshots ($t \in [0.03, 0.93]$) as the training data to infer the permeability. Inferred permeability values are 0.0011 and 1.0003 compared to the true values of 0.001 and 1.0. (Figure adapted from [14])

velocity field based on the measurable phase field data, X_{data}. We draw 1,000 points from an initial snapshot (t_0 (⋆), 10,000 inner points from the inner spatio-temporal domain (⋆), and 1,000 points at boundaries (⋆) to estimate the total loss at each training epoch (shown in Fig. 8b).

In our setup shown in Fig. 8a, the core permeability is $\kappa = 0.001$ ($\phi = -1$) with a permeable outer shell as $\kappa = 1$ ($\phi = 0$). The phase field variable ϕ is expressed as a function of κ, namely $\kappa(\phi) = e^{a\phi} + b$. a and b are model parameters to be optimized in the PINN model with their true value as 6.90 and 0.0. The form of the relation function is not unique as long as the permeability value matches the true value at $\phi = 1$ and 0.

5.3 Inference results

We present the history of different losses over the total 300,000 epochs in Fig. 9a. Although the PDE residual has the largest error, each term has a value lower than 10^{-2}, indicating a convergence at the final epoch. Fig. 9b plots the predicted relation function; the optimized parameters, a and b, have a value at 7.1 and 0.0003, resulting in a well-matched prediction for permeability at the core and shell. In Fig. 9c, we qualitatively compare the predicted field ϕ_{pred} and v_{pred} along with their absolute error at $t = 0.78$; both fields exhibit an agreement with the reference data whereas the errors for the phase field are mainly distributed in and around the outlet layer and its interface. Overall, the results indicate that our model can accurately infer the hidden velocity field and the unknown PDE parameters only based on the sampled data in phase field.

6 Summary

PINNs offer a new approach to simulating realistic fluid flows, where some data are available from multimodality measurements whereas the boundary conditions or initial conditions may be unknown. While this is perhaps the prevailing scenario in practice, existing CFD solvers cannot handle such ill-posed problems and hence one can think of PINNs for fluid problems as a complementary approach to the plethora of existing numerical methods for CFD for idealized problems.

There are several opportunities for further research, e.g., using PINNs for active flow control to replace expensive experiments and time-consuming large-scale simulations as in Ref. [59], or predict fast the flow at a new high Reynolds number using transfer learning techniques assuming we have available solutions at lower Reynolds numbers. Moreover, a new area for exploration could be the development of closure models for unresolved flow dynamics at very high Reynolds number using the automatic data assimilation method provided by PINNs. Computing flow problems at scale requires efficient multi-GPU implementations in the spirit of data parallel [29] or a hybrid data parallel and model parallel paradigms as in Ref. [28]. The parallel speed up obtained for flow simulations so far is very good, suggesting that PINNs can be used in the near future for industrial complexity problems at scale that CFD methods cannot tackle.

Acknowledgements The research of the second author (ZM) was supported by the National Natural Science Foundation of China (Grant 12171404). The last author (GEK) would like to acknowledge support by the Alexander von Humboldt fellowship.

References

1. Brooks, A.N., Hughes, T.J.: Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. Comput. Methods Appl. Mech. Eng. **32**, 199–259 (1982)
2. Karniadakis, G.E., Sherwin, S.: Spectral-hp Element Methods for Computational Fluid Dynamics, 2nd edn. Oxford University Press, Oxford (2005)
3. Katz, A.J.: Meshless Methods for Computational Fluid Dynamics. Stanford University Stanford, Stanford (2009)

4. Liu, M., Liu, G.: Smoothed particle hydrodynamics (SPH): an overview and recent developments. Arch. Comput. Methods Eng. **17**, 25–76 (2010)
5. Beck, J.V., Blackwell, B., Clair Jr, C.R.S.: Inverse heat conduction: Ill-posed problems. James Beck (1985)
6. Jasak, H., Jemcov, A., Tukovic, Z., et al.: OpenFOAM: A C++ library for complex physics simulations. In: International Workshop on Coupled Methods in Numerical Dynamics, vol. 1000, pp. 1–20. IUC Dubrovnik Croatia (2007)
7. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed learning machine (2021). US Patent 10,963,540
8. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Machine learning of linear differential equations using Gaussian processes. J. Comput. Phys. **348**, 683–693 (2017)
9. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Numerical Gaussian processes for time-dependent and nonlinear partial differential equations. SIAM J. Sci. Comput. **40**, A172–A198 (2018)
10. Raissi, M., Wang, Z., Triantafyllou, M.S., et al.: Deep learning of vortex-induced vibrations. J. Fluid Mech. **861**, 119–137 (2019)
11. Jin, X., Cai, S., Li, H., et al.: NSFnets (Navier-Stokes flow nets): physics-informed neural networks for the incompressible Navier-Stokes equations. J. Comput. Phys. **426**, 109951 (2021)
12. Raissi, M., Yazdani, A., Karniadakis, G.E.: Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations. Science **367**, 1026–1030 (2020)
13. Mao, Z., Jagtap, A.D., Karniadakis, G.E.: Physics-informed neural networks for high-speed flows. Comput. Methods Appl. Mech. Eng. **360**, 112789 (2020)
14. Yin, M., Zheng, X., Humphrey, J.D., et al.: Non-invasive inference of thrombus material properties with physics-informed neural networks. Comput. Methods Appl. Mech. Eng. **375**, 113603 (2021)
15. 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)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
17. Shin, Y., Darbon, J., Karniadakis, G.E.: On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. Commun. Comput. Phys. **28**, 2042–2074 (2020)
18. Abadi, M., Barham, P., Chen, J., et al.: Tensorflow: a system for large-scale machine learning. In: 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), pp. 265–283 (2016)
19. Paszke, A., Gross, S., Massa, F., et al.: Pytorch: an imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703 (2019)
20. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part i): data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:1711.10561 (2017)
21. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part ii): data-driven discovery of nonlinear partial differential equations. arxiv. arXiv preprint arXiv:1711.10561 (2017)
22. Pang, G., Lu, L., Karniadakis, G.E.: fPINNs: fractional physics-informed neural networks. SIAM J. Sci. Comput. **41**, A2603–A2626 (2019)
23. Fang, Z., Zhan, J.: A physics-informed neural network framework for PDEs on 3D surfaces: time independent problems. IEEE Access **8**, 26328–26335 (2019)
24. Zhang, D., Guo, L., Karniadakis, G.E.: Learning in modal space: solving time-dependent stochastic PDEs using physics-informed neural networks. SIAM J. Sci. Comput. **42**, A639–A665 (2020)
25. Kharazmi, E., Zhang, Z., Karniadakis, G.E.: hp-VPINNs: variational physics-informed neural networks with domain decomposition. Comput. Methods Appl. Mech. Eng. **374**, 113547 (2021)
26. 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)
27. 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)
28. Shukla, K., Jagtap, A.D., Karniadakis, G.E.: Parallel physics-informed neural networks via domain decomposition. arXiv preprint arXiv:2104.10013 (2021)
29. Hennigh, O., Narasimhan, S., Nabian, M.A., et al.: NVIDIASimNet^{TM}: an AI-accelerated multi-physics simulation-frameworkarxivhttp://arxiv.org/abs/2012.07938arXiv:2012.07938
30. Yang, Y., Perdikaris, P.: Adversarial uncertainty quantification in physics-informed neural networks. J. Comput. Phys. **394**, 136–152 (2019)
31. Zhang, D., Lu, L., Guo, L., et al.: Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. J. Comput. Phys. **397**, 2019 (2019)
32. Zhu, Y., Zabaras, N., Koutsourelakis, P.S., et al.: Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. J. Comput. Phys. **394**, 56–81 (2019)
33. Sun, L., Wang, J.X.: Physics-constrained Bayesian neural network for fluid flow reconstruction with sparse and noisy data. Theor. Appl. Mech. Lett. **10**, 161–169 (2020)
34. Yang, L., Meng, X., Karniadakis, G.E.: B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. J. Comput. Phys. **425**, 109913 (2021)
35. Meng, X., Karniadakis, G.E.: A composite neural network that learns from multi-fidelity data: application to function approximation and inverse PDE problems. J. Comput. Phys. **401**, 109020 (2020)
36. Jagtap, A.D., Kawaguchi, K., Karniadakis, G.E.: Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. J. Comput. Phys. **404**, 109136 (2020)
37. Wang, S., Teng, Y., Perdikaris, P.: Understanding and mitigating gradient pathologies in physics-informed neural networks. arXiv preprint arXiv:2001.04536 (2020)
38. Lu, L., Pestourie, R., Yao, W., et al.: Physics-informed neural networks with hard constraints for inverse design. arXiv preprint arXiv:2102.04626 (2021)
39. 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)
40. Mishra, S., Molinaro, R.: Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs. arXiv preprint arXiv:2006.16144 (2020)
41. Mishra, S., Molinaro, R.: Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating a class of inverse problems for PDEs. arXiv preprint arXiv:2007.01138 (2020)
42. Wang, S., Yu, X., Perdikaris, P.: When and why PINNs fail to train: A neural tangent kernel perspective. arXiv preprint arXiv:2007.14527 (2020)
43. Kissas, G., Yang, Y., Hwuang, E., et al.: 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)
44. Yang, X., Zafar, S., Wang, J.X., et al.: Predictive large-eddy-simulation wall modeling via physics-informed neural networks. Phys. Rev. Fluids **4**, 034602 (2019)

45. Lou, Q., Meng, X., Karniadakis, G.E.: Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-BGK formulation. arXiv preprint arXiv:2010.09147 (2020)
46. Cai, S., Wang, Z., Wang, S., et al.: Physics-informed neural networks for heat transfer problems. J. Heat Transf. **143**, 060801 (2021)
47. Cai, S., Wang, Z., Fuest, F., et al.: 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** (2021)
48. Wang, S., Perdikaris, P.: Deep learning of free boundary and Stefan problems. J. Comput. Phys. **428**, 109914 (2021)
49. Lucor, D., Agrawal, A., Sergent, A.: Physics-aware deep neural networks for surrogate modeling of turbulent natural convection. arXiv preprint arXiv:2103.03565 (2021)
50. Mahmoudabadbozchelou, M., Caggioni, M., Shahsavari, S., et al.: Data-driven physics-informed constitutive metamodeling of complex fluids: a multifidelity neural network (mfnn) framework. J. Rheol. **65**, 179–198 (2021)
51. Arzani, A., Wang, J.X., D'Souza, R.M.: Uncovering near-wall blood flow from sparse data with physics-informed neural networks. arXiv preprint arXiv:2104.08249 (2021)
52. Cai, S., Li, H., Zheng, F., et al.: Artificial intelligence velocimetry and microaneurysm-on-a-chip for three-dimensional analysis of blood flow in physiology and disease. Proc. Natl. Acad. Sci. **118**(13) (2021)
53. Wang, R., Kashinath, K., Mustafa, M., et al.: Towards physics-informed deep learning for turbulent flow prediction. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1457–1466 (2020)
54. Goswami, S., Anitescu, C., Chakraborty, S., et al.: Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. Theor. Appl. Fract. Mech. **106**, 102447 (2020)
55. Zhang, E., Yin, M., Karniadakis, G.E.: Physics-informed neural networks for nonhomogeneous material identification in elasticity imaging. arXiv preprint arXiv:2009.04525 (2020)
56. Zheng, X., Yazdani, A., Li, H., et al.: A three-dimensional phase-field model for multiscale modeling of thrombus biomechanics in blood vessels. PLoS Comput. Biol. **16**, e1007709 (2020)
57. Xu, Z., Chen, N., Kamocka, M.M., et al.: A multiscale model of thrombus development. J. R. Soc. Interface **5**, 705–722 (2008)
58. Yazdani, A., Li, H., Humphrey, J.D., et al.: A general shear-dependent model for thrombus formation. PLoS Comput. Biol. **13**, e1005291 (2017)
59. Fan, D., Yang, L., Wang, Z., et al.: Reinforcement learning for bluff body active flow control in experiments and simulations. Proc. Natl. Acad. Sci. **117**, 26091–26098 (2020)