

A physics-informed neural network for fluid–structure coupled simulation of a Euler–Bernoulli beam under steady flow

Xuanhan Xia¹, Xiaofan Li¹ , Jinfeng Zhang², Shunxiang Cao³, and Guangyao Wang^{4,5,6} 

¹Department of the Mechanical Engineering, Faculty of Engineering, The University of Hong Kong, Hong Kong 999077, China

²State Key Laboratory of Hydraulic Engineering Intelligent Construction and Operation, Tianjin University, Tianjin 300072, China

³Institute for Ocean Engineering, Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China

⁴Centre for Regional Oceans & Department of Ocean Science and Technology, Faculty of Science and Technology, University of Macau, Macau 999078, China

⁵State Key Laboratory of Internet of Things for Smart City, University of Macau, Macau 999078, China

⁶Zhuhai UM Science and Technology Research Institute, Zhuhai 519031, China



Cite This: *Ocean*, 2026, 2, 9470017



Read Online

ABSTRACT: Despite the significant progress in recent years in numerical simulations using physics-informed neural networks (PINNs), it remains challenging to apply PINNs to solve coupled fluid–structure interaction problems. This work employs PINNs to approximate two-dimensional fluid–structure interaction in both forward and inverse problem settings. For the forward problem, data are randomly sampled from the fluid domain and combined with governing physical laws, including the Robin boundary condition and the Euler–Bernoulli equation, to infer the beam displacement. For the inverse problem, in which a parameter in the Euler–Bernoulli equation is unknown, the problem is solved using both data and physical constraints to infer the fluid pressure and pressure gradient at the interface, from which the beam pressure and acceleration are computed via the Robin boundary condition. Finally, using the data obtained in the previous step, both the beam displacement and the unknown parameter are identified. The results demonstrate that PINNs can accurately solve fluid–structure coupled problems by enforcing conservation laws at randomly distributed collocation points, achieving relative L2 errors below 1%, particularly in inverse problems, for which conventional techniques are often ineffective.

KEYWORDS: physics-informed neural networks, fluid–structure interaction, forward problem, inverse problem, sparse data

1 Introduction

Fluid–solid interaction (FSI) problems are widespread across many fields, including aerospace engineering [1,2], hydraulic engineering [3,4], and biomechanics [5], and research on FSI phenomena enables a deeper understanding of the underlying mechanisms, thereby facilitating the solution of practical engineering problems. The study of FSI problems can be traced back to the 1950s, when research was primarily focused on structural responses in simplified flow fields, such as vibrations induced by hydrodynamic forces. By the 1970s, with the development of the finite element method (FEM) and computational fluid dynamics (CFD) [6,7], researchers began to couple fluid and structural computations. In the 21st century, FSI research has advanced toward a multiphysics coupling stage. The immersed boundary method (IBM) [8] and open-source computational libraries, such as OpenFOAM [9], have enabled the simulation of complex motions and extreme deformations, including biological swimming and impact loading.

In recent years, deep learning has emerged as a powerful tool for modeling complex fluid and structural phenomena, such as

turbulence and multiphase flows [10]. For instance, Fukami et al. [11] utilized convolutional neural networks (CNN) and generative adversarial networks for the super-resolution reconstruction of turbulent flow fields, thereby recovering high-fidelity flow details from low-resolution data. Meanwhile, Vlachas et al. [12] combined long short-term memory (LSTM) networks with dynamic mode decomposition (DMD) to efficiently predict the transient behavior of unsteady flows. Additionally, reinforcement learning (RL) facilitates the optimization of control strategies through interactions between intelligent agents and fluid environments. For example, Verma et al. [13] employed RL to simulate the collective behavior of swimming fish, while Reddy et al. [14] applied RL to the design and optimization of aircraft aerodynamic shapes.

Although extensive research has verified the effectiveness and robustness of machine learning methods, one of the greatest challenges in training these data-driven models lies in acquiring large amount of data in real-world scenarios, especially under harsh conditions, such as marine environments [15]. Therefore, physics-driven approaches, which incorporate known physical laws into the deep learning training process, are becoming a promising direction for solving high-dimensional nonlinear problems in data-scarce regimes. Lagaris et al. [16] proposed an early framework for solving partial differential equations (PDEs) using neural networks. They leveraged the nonlinear approximation capability of neural networks by embedding PDE residuals, boundary conditions, and initial conditions into the loss function. By optimizing the loss

Received: November 4, 2025; **Revised:** December 12, 2025

Accepted: December 15, 2025

 Address correspondence to Xiaofan Li, lixf@hku.hk; Guangyao Wang, wanggy@um.edu.mo

function, the neural network was trained to approximate the solution of the PDE [17,18]. Despite some progress during this period, the application of neural networks to PDE solving encountered numerous challenges, largely due to the immaturity of deep learning techniques, including training instability, slow convergence, and limited generalization. Raissi et al. [19] introduced the physics-informed neural network (PINN), a neural network framework designed to solve supervised learning tasks while respecting prescribed physical laws governed by general nonlinear PDEs. The PINN approach involves incorporating known physical equations as regularization terms within the network loss function, thereby allowing physical laws to explicitly constrain the training process. Through iterative optimization of the total loss function, the network output progressively conforms to the prescribed physical laws [20–22].

Studies over the past few years have demonstrated the potential of PINNs in various fields [23–25], thereby motivating extensions of the classical framework [26–28]. From a hydrodynamics perspective, PINNs, compared with traditional methods, not only reduce computational time but also maintain high accuracy across a wide range of scenarios [29,30]. For example, Mao et al. [31] investigated the potential of using PINNs to approximate the Euler equations for the simulation of high-speed aerodynamic flows. By exploring one-dimensional and two-dimensional problems, they demonstrated the effectiveness of PINNs in various hypersonic flow scenarios, including shock-wave capture, heat conduction, and viscous-effect simulations. Sun et al. [32] utilized a physics-constrained deep learning framework, embedding fundamental fluid dynamics laws, such as the Navier-Stokes equations, into the loss function, thereby enabling the model to maintain physical consistency even in the absence of training data. By directly learning the spatiotemporal distribution of flow-field variables (e.g., velocity and pressure) using neural networks, they achieved high accuracy in low-*Re* flows and demonstrated the capability to handle parameterized problems. In the field of solid and materials mechanics, PINNs have been widely applied to damage identification [33,34]. Shukla et al. [35] explored the use of PINNs for ultrasonic non-destructive testing to quantify the depth and geometry of surface cracks. By embedding sparse data and the governing wave equation into the neural network loss function, they reconstructed the overall wave field, in which anomalies indicated the locations of cracks. Additionally, a substantial body of research has demonstrated the effectiveness of PINNs in ocean engineering [36,37], aeronautics [38,39], and related engineering disciplines.

While PINNs have demonstrated remarkable success in single-physics domains, their application to tightly coupled multiphysics problems, such as FSI, remains an emerging and challenging area of research. Therefore, investigating the application of PINNs to FSI problems can play a significant role in addressing increasingly complex practical scenarios, particularly when available data are sparse. In this paper, we systematically develop a novel PINN-based framework, termed FSI-PINNs, to solve both forward and inverse FSI problems. For the forward problem, traditional CFD methods, such as FEM and IBM, typically require well-defined physical conditions, including governing equations, initial conditions, and boundary conditions, to produce accurate results. Any inappropriate specification can easily lead to computational failure, making the simulation process time-consuming and inefficient. In contrast, PINNs, as a physics-informed learning approach, can

perform effectively by embedding limited observational data together with known physical laws, even when the available data are sparse and noisy. The proposed FSI-PINNs framework is designed by representing the fluid domain and structure with two separate neural networks, which are then coupled through interface conditions. This design enables accurate prediction of structural displacement using only a small number of samples from the fluid domain. One key objective of the proposed FSI-PINNs framework is to address forward problems in harsh environments, such as marine settings, where data acquisition is challenging due to waves, winds, currents, and extreme weather conditions, and where traditional simulation methods are notoriously difficult to apply. With respect to the inverse problem, PINNs, as artificial intelligence-based methods, are conceptually rooted in traditional optimization-based approaches. The proposed FSI-PINNs framework is further utilized to infer unknown structural parameters, which can be regarded as a novel approach to fault detection. While most previous PINN-based inverse studies focus on single structural components, such as beams or plates, the proposed FSI-PINNs framework incorporates sparse data from both the fluid domain and the structure for parameter identification. Data sampled from the fluid domain are used to reconstruct the entire fluid field and subsequently generate additional structural data through interface conditions. The success of this sampling strategy indicates that even when insufficient structural data are available for training, additional information can be obtained from the fluid domain to ensure accurate parameter identification.

In general, the main contribution of this research is the proposal of a novel FSI-PINNs framework and the validation of its feasibility for both forward and inverse problems under sparse data conditions. First, the FSI-PINNs framework is designed using two separate neural networks, each with distinct input, output, and network structures, representing the fluid domain and structural domain, respectively. This separation allows independent hyperparameter tuning, enhancing the model's stability and convergence. Second, the model demonstrates high data efficiency, producing accurate results even when only sparse or noisy data are available, which is common in practical applications. Third, this study successfully extends the PINNs methodology from single-physics domains to fluid–structure interaction problems. It demonstrates considerable potential as a robust tool for simulation and parameter identification, capabilities that are challenging for conventional methods.

The structure of this paper is as follows: In Section 2, we describe the FSI problem model and the standard PINNs procedures, as well as the FSI-PINNs framework for both forward and inverse problems. In Section 3, we present the performance of the proposed FSI-PINNs framework. For the forward problem, the model is tested with varying numbers of sampling points and subjected to noise analysis. For the inverse problem, the model is validated by inferring the unknown densities of uniform and non-uniform beams. In Section 4, we discuss the conclusions and future research directions.

2 Numerical framework

2.1 Physical model of FSI

We consider the FSI between a Euler–Bernoulli beam and a square-shaped fluid domain. Figure 1 presents a schematic of the problem

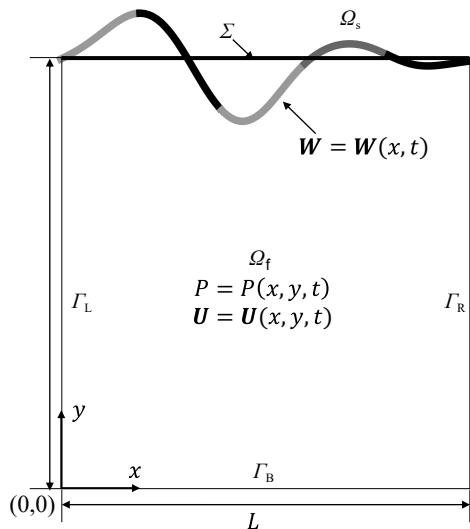


Figure 1 A simplified fluid-structure interaction model: Ω_f denote the 2D fluid domain, with width L and height H . Ω_s is the Euler-Bernoulli beam above the fluid domain with constant cross section. Σ represents the interface of fluid domain and beam. Adapted with permission from Ref. [40] © 2020, John Wiley & Sons, Ltd.

setup. The computational domain has a height of H and a length of L . We assume that the beam Ω_s is initially stationary and own acceleration, and then vibrates under the action of the fluid flow Ω_f . The beam is simply supported at both ends, meaning that its displacement and acceleration at the boundaries are zero. We assume the beam's cross-section has a rectangular shape, with its width b , density ρ_s , thickness h , and flexural rigidity EI being a constant along the beam length. For the flow field, the left and right boundaries are assumed to be periodic, while the bottom boundary is a no-slip wall. On the fluid-beam interface Σ , we combine kinematic and dynamic interface conditions into a Robin-type interface condition [40], which directly relates the fluid pressure gradient to the structural acceleration. Further details of the parameters are provided in Section 3.

The governing equations of this FSI problem are given by

$$EI \frac{\partial^4 W}{\partial x^4} + \rho_s b h \frac{\partial^2 W}{\partial t^2} = f(x, t), \text{ in } (0, T) \times \Omega_s \quad (1)$$

$$W = 0 \text{ and } \frac{\partial^2 W}{\partial x^2} = 0, \text{ at } x = 0 \text{ and } x = L \quad (2)$$

$$\nabla^2 P = 0, \text{ in } (0, T) \times \Omega_f \quad (3)$$

$$\frac{\partial P}{\partial y} = 0, \text{ on } \Gamma_B \quad (4)$$

$$\left. \frac{\partial P}{\partial y} \right|_{\Gamma_L} = \left. \frac{\partial P}{\partial y} \right|_{\Gamma_R}, P|_{\Gamma_L} = P|_{\Gamma_R} \quad (5)$$

$$\frac{\partial P}{\partial y} = -\rho_f \frac{\partial^2 W}{\partial t^2}, \text{ on } \Sigma \quad (6)$$

$$f = Pb, \text{ on } \Sigma \quad (7)$$

where W denotes the displacement of the beam. ρ_f and P represent fluid density and pressure, respectively. Γ_L , Γ_R , and Γ_B denote the left,

right, and bottom boundaries of Ω_f , respectively. f is the flow-induced force on the beam.

This physical model is based on simplified assumptions, including potential flow (inviscid and incompressible), governed by the Laplace equation, and an Euler-Bernoulli beam under small deformations. However, despite these simplifications, the core contribution of this study lies in the novel coupled PINNs framework for solving FSI problems. In this work, we demonstrate the robustness and effectiveness of the proposed model. Future research will focus on extending the framework to more complex physical models, including three-dimensional, transient, or large-deformation scenarios.

2.2 Framework of PINNs

The physical quantities involved in PINNs often span multiple scales and may display high-frequency or abrupt variations. For neural networks, capturing such variations presents a significant challenge. Therefore, before training, it is necessary to perform feature engineering on both the input features and target labels. Feature engineering in PINNs typically involves rescaling techniques, including non-dimensionalization and standardization. In this study, we apply standardization to the input coordinates and the training data P to constrain their values within the range $[0, 1]$. The formulas for standardization are as follows:

$$\bar{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} = \frac{x - x_{\min}}{X_A} \quad (8)$$

$$\bar{y} = \frac{y - y_{\min}}{y_{\max} - y_{\min}} = \frac{y - y_{\min}}{Y_A} \quad (9)$$

$$\bar{t} = \frac{t - t_{\min}}{t_{\max} - t_{\min}} = \frac{t - t_{\min}}{T_A} \quad (10)$$

$$\bar{P} = \frac{P - P_{\min}}{P_{\max} - P_{\min}} = \frac{P - P_{\min}}{P_A} \quad (11)$$

Equations 8–11 constrain the range of features and data to $[0, 1]$, which not only accelerates the convergence of the network but also enhances the parameter sensitivity of the model. Additionally, since the mapping relationship in PINNs includes PDEs, changes in the input features also lead to variations in the governing equations. Finally, Eqs. 1–7 are re-formulated as follows:

$$\frac{1}{X_A^4} \frac{\partial^4 W}{\partial \bar{x}^4} + \alpha \frac{1}{T_A^2} \frac{\partial^2 W}{\partial \bar{t}^2} = \frac{f(\bar{x}, \bar{t})}{EI}, \text{ in } (0, 1) \times \Omega_s \quad (12)$$

$$W = 0 \text{ and } \frac{\partial^2 W}{\partial \bar{x}^2} = 0, \text{ at } \bar{x} = 0 \text{ and } \bar{x} = 1 \quad (13)$$

$$\frac{P_A}{X_A^2} \frac{\partial^2 \bar{P}}{\partial \bar{x}^2} + \frac{P_A}{Y_A^2} \frac{\partial^2 \bar{P}}{\partial \bar{y}^2} = 0, \text{ in } (0, 1) \times \Omega_f \quad (14)$$

$$\frac{\partial \bar{P}}{\partial \bar{y}} = 0, \text{ on } \Gamma_B \quad (15)$$

$$\left. \frac{\partial \bar{P}}{\partial \bar{y}} \right|_{\Gamma_L} = \left. \frac{\partial \bar{P}}{\partial \bar{y}} \right|_{\Gamma_R}, \bar{P}|_{\Gamma_L} = \bar{P}|_{\Gamma_R} \quad (16)$$

$$\frac{P_A}{Y_A} \frac{\partial \bar{P}}{\partial \bar{y}} = -\rho_f \frac{1}{T_A^2} \frac{\partial^2 W}{\partial \bar{t}^2}, \text{ on } \Sigma \quad (17)$$

$$f = P_A \bar{P} \cdot b, \text{ on } \Sigma \quad (18)$$

After feature engineering, the next step is to train the neural network.

We consider a fully-connected neural network in PINNs [41]. A fully connected neural network is composed of $L+1$ layers, where the 0th layer is the input layer, which takes time and spatial coordinates, $\mathbf{x} = (x, y, z) \in \mathbb{R}^M$, as input. The L th layer is the output layer, producing the target features $\mathbf{u} \in \mathbb{R}^N$, where M is the dimension of the input features and N is the dimension of the output features. The layers between the output layer and the input layer are referred to as hidden layers $l_i, i = 1, 2, \dots, L$. Specifically, the input layer is connected to the first hidden layer, and the L th hidden layer is connected to the output layer. The term “fully connected neural network” arises because each neuron in a hidden layer is connected to every neuron in both the preceding and the following layers. The training process can also be expressed by the following formulas (Eq. 19),

$$\begin{aligned} \mathbf{u} &= \sigma(\mathbf{w}^L \mathbf{z}^{L-1} + \mathbf{b}^L) \\ \mathbf{z}^{L-1} &= \sigma(\mathbf{w}^{L-1} \mathbf{z}^{L-2} + \mathbf{b}^{L-1}) \\ \mathbf{z}^{L-2} &= \sigma(\mathbf{w}^{L-2} \mathbf{z}^{L-3} + \mathbf{b}^{L-2}) \\ &\vdots \\ \mathbf{z}^1 &= \sigma(\mathbf{w}^1 \mathbf{x} + \mathbf{b}^1) \end{aligned} \quad (19)$$

Here, $\mathbf{w}^i$ and $\mathbf{b}^i$ represent the weight matrix and bias vector of the i th hidden layer, which are parameters that the neural network updates over thousands of iterations. σ is the activation function, which is commonly chosen as tanh in PINNs.

During training, loss functions play a critical role in directing the network output to converge toward both the data and the governing equations. They quantify the discrepancy between the network predictions and the reference values, which include both training data and physical constraints. Data loss is typically evaluated using the mean squared error, which computes the average of the squared differences between predicted and true values. This formulation emphasizes larger errors, effectively penalizing predictions with significant deviations. Let $\hat{u}$ denote the predicted network values, u_{data} represent the known data values, and N_{data} indicate the size of the dataset; the data loss L_{data} can be expressed as

$$L_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_i^{N_{\text{data}}} (\hat{u} - u_{\text{data}})^2 \quad (20)$$

Residuals are generally composed of residuals from PDEs, boundary conditions, and initial conditions. Unlike data loss, equation residuals lack known labels for training and are instead incorporated into the loss function by constructing residual equations

$$\begin{aligned} L_{\text{PDE}} &= \frac{1}{N_{\text{PDE}}} \sum_i^{N_{\text{PDE}}} [f(\hat{W})]^2 \\ &= \frac{1}{N_{\text{PDE}}} \sum_i^{N_{\text{PDE}}} \left[\frac{1}{X_A^4} \frac{\partial^4 \hat{W}}{\partial \bar{x}^4} + \alpha \frac{1}{T_A^2} \frac{\partial^2 \hat{W}}{\partial \bar{t}^2} - \frac{f(x, t)}{\text{EI}} \right]^2 \end{aligned} \quad (21)$$

where N_{PDE} represents the number of residual sampling points. By minimizing the residual loss function L_{PDE} , the network's predictions can be aligned with known physical laws. Similarly,

residuals for boundary conditions L_{BC} and initial conditions L_{IC} can be constructed following the same approach shown in Eq. 21. The total loss function can be described as

$$L_{\text{total}} = \lambda_1 L_{\text{PDE}} + \lambda_2 L_{\text{BC}} + \lambda_3 L_{\text{IC}} + \lambda_4 L_{\text{data}} \quad (22)$$

The parameters $\lambda_1 - \lambda_4$ are weights that balance the contributions of different loss components, facilitating faster and more stable convergence. The procedure of incorporating residuals from boundary or initial conditions into the loss function, as described above, is referred to as soft constraints. Soft constraints ensure that the approximate solution satisfies boundary and initial conditions by iteratively optimizing the loss function during training. However, this approach does not guarantee exact satisfaction of boundary or initial conditions and can sometimes result in training difficulties, particularly when the loss function contains a large number of terms.

Lu et al. [42] proposed a method based on a transformation of the network output, enabling the network to automatically satisfy constraint conditions. This approach, known as hard constraints, has been shown to significantly enhance the training performance of PINNs [43,44]. In this context, the network's output is defined as $\mathbf{h}(\mathbf{x}; \Theta)$, and a new network output after hard constraint $\mathbf{h}'(\mathbf{x}; \Theta)$ is introduced by

$$\mathbf{h}'(\mathbf{x}; \Theta) = g(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{h}(\mathbf{x}; \Theta) \quad (23)$$

where $g(\mathbf{x})$ is a function that simultaneously satisfies the required boundary conditions or initial conditions, $\mathbf{G}(\mathbf{x})$ referred to as a distance function, equaling 0 both at the initial time and on the boundaries, $\mathbf{x}$ and Θ represent the input and hyperparameters of the network, respectively. Through hard constraints, the network's output can automatically satisfy part of the boundary and initial conditions, reducing the number of terms in the total loss function. This algorithm effectively mitigates the convergence difficulties in training process.

Apart from optimization design choices, the training strategy of PINNs also plays a key role in the training process and is commonly implemented using Adam and limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimizers. Adam is particularly effective during the initial training phase due to its adaptive learning rate and momentum, enabling robust convergence even in complex, high-dimensional loss landscapes typical of PINNs. It efficiently navigates non-convex optimization problems and is relatively insensitive to hyperparameters, making it well-suited for early-stage training. L-BFGS, a quasi-Newton method, is applied later for precision refinement. By approximating second-order curvature (Hessian information), it achieves faster and more accurate convergence near local minima. However, L-BFGS is sensitive to initial conditions and can be computationally expensive for large-scale problems. Using it after Adam provides a stable starting point and reduces the risk of convergence failure.

In this study, all cases are trained sequentially, first using the Adam optimizer, followed by L-BFGS.

2.3 FSI-PINNs framework

2.3.1 Forward problem

As the physical model consists of two distinct domains, we employ two separate neural networks to represent them. This decoupled training strategy is derived from domain decomposition, in which the computational domain is divided into several subdomains, each

represented by a single neural network. The networks are then coupled through boundary conditions [45–47]. The first neural network is designed to simulate the fluid field. Pressure data are sampled from the fluid domain, and Network 1 is trained to reconstruct the entire fluid domain by incorporating both the governing equations and observed data into its loss function. The two sub-loss functions and the total loss function are defined as follows:

$$L_{f,PDE} = \frac{1}{N_{f,PDE}} \sum_i \left(\frac{P_A}{X_A^2} \frac{\partial^2 \hat{P}}{\partial \bar{x}^2} + \frac{P_A}{Y_A^2} \frac{\partial^2 \hat{P}}{\partial \bar{y}^2} \right)^2 \quad (24)$$

$$L_{f,data} = \frac{1}{N_{f,data}} \sum_i (\hat{P} - P_{data})^2 \quad (25)$$

$$L_{f,total} = \lambda_{PDE} L_{f,PDE} + \lambda_{data} L_{f,data} \quad (26)$$

where λ_{PDE} and λ_{data} are the weights of the residual loss and data loss, respectively, and we set them to 1 and 100. The weights in the loss function were determined empirically to balance the magnitude of back-propagated gradients from the data and PDE residuals [19,22]. While advanced self-adaptive weighting algorithms (e.g., neural tangent kernel-based method) have been proposed to address gradient pathologies in complicated scenarios [20,21], our proposed decoupled training strategy inherently reduces the complexity of the loss functions. Therefore, manual adjustment, aimed at normalizing the scale of different loss terms, was found to be sufficient for achieving robust convergence in this study.

The next step is to convert the fluid data from Network 1 into structure data based on the interface conditions. In this problem,

we can compute the external force acting on the beam and the acceleration of the beam according to Eqs. 17 and 18:

$$f_{data} = P_A b \hat{P}|_{\Sigma} \quad (27)$$

$$\left. \frac{\partial^2 W}{\partial t^2} \right|_{data} = - \frac{P_A T_A^2}{Y_A \rho_f} \left. \frac{\partial \hat{P}}{\partial \bar{y}} \right|_{\Sigma} \quad (28)$$

It is noted that the output of Network 1 is the pressure of fluid field P and the neural network can automatically compute its derivatives to conduct computation of interface conditions. In the end, we utilize the force and acceleration data we get from Eqs. 27 and 28 to train Network 2 and the output of Network 2 is the displacement of the beam. Compared to PINNs applied to a single Euler-Bernoulli beam, where the network is typically trained using sparse displacement data obtained from sensors, our FSI-PINNs approach uses acceleration instead of displacement to train Network 2. This allows to avoid sampling data directly from the structure, instead relying on the fluid domain, making the approach more cost-efficient and feasible, particularly for scenarios involving minor deformations. The entire workflow is demonstrated in Fig. 2.

Additionally, hard constraint is imposed on the output of Network 2 to enhance the accuracy of the results. For Eq. 23,

$$g(\bar{x}, \bar{t}) = 0 \quad (29)$$

$$G(\bar{x}, \bar{t}) = \sin(2\pi\bar{x}) \quad (30)$$

the sinusoidal function is chosen to enforce the Dirichlet boundary conditions exactly, ensuring that the displacement and acceleration are strictly zero at the boundaries. Moreover, the sine function is

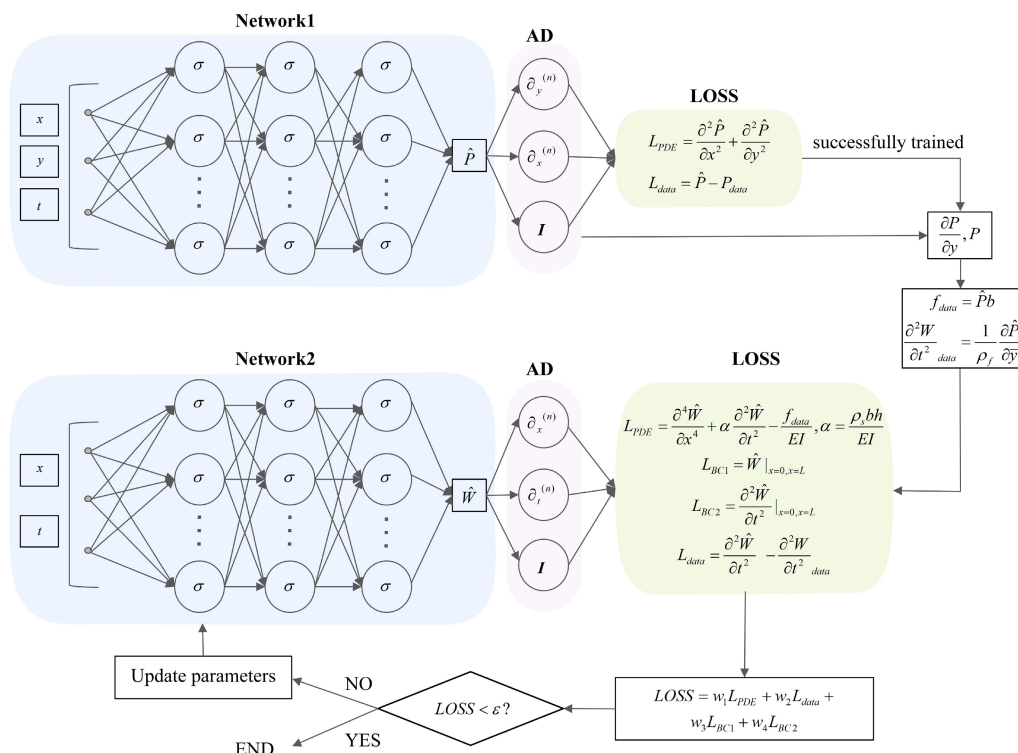


Figure 2 Forward problem framework of FSI-PINNs. Network 1, with time and space coordinates (x, y, t) as inputs, is used to approximate the pressure of the fluid field $\hat{P}$; Network 2, with time and space coordinates (x, t) as inputs, is used to approximate the displacement of the beam $\hat{W}$. Automatic differentiation (AD) is used to calculate the derivatives of outputs and I , identity operator, pass the output to the loss functions directly without differentiation.

infinitely differentiable, which guarantees that the hard constraint does not introduce singularities or numerical instabilities during the AD process for high-order derivatives.

Using this approach, a new output for Network 2 can be obtained:

$$\hat{W}'(\bar{x}, \bar{t}, \Theta) = \hat{W}(\bar{x}, \bar{t}, \Theta) \sin(2\pi\bar{x}) \quad (31)$$

In this context, the boundary conditions of the beam are satisfied automatically, allowing the loss functions of Network 2 to be simplified as follows:

$$L_{b,PDE} = \frac{1}{N_{b,PDE}} \sum_i \left(\frac{1}{X_A^4} \frac{\partial^4 \hat{W}}{\partial \bar{x}^4} + \alpha \frac{1}{T_A^2} \frac{\partial^2 \hat{W}}{\partial \bar{t}^2} - \frac{f_{data}}{EI} \right)^2 \quad (32)$$

$$L_{b,data} = \frac{1}{N_{b,data}} \sum_i \left(\frac{\partial^2 \hat{W}}{\partial \bar{t}^2} - \frac{\partial^2 W}{\partial \bar{t}^2} \Big|_{data} \right)^2 \quad (33)$$

$$L_{b,total} = \lambda_{PDE} L_{b,PDE} + \lambda_{data} L_{b,data} \quad (34)$$

Here, we set the value of λ_{PDE} and λ_{data} to 100 and 10,000 manually based on experience.

2.3.2 Inverse problem of uniform beam

For inverse problems, our FSI-PINNs aim to identify the unknown parameters in the governing equations. In this study, the beam density is treated as the unknown parameter, and sparse data are sampled from both the fluid field and the beam to infer the unknown density for uniform and non-uniform beams, respectively.

The framework of FSI-PINNs for a uniform beam is largely similar to that of the forward problem, consisting of two main components: forward propagation and backward propagation.

Forward propagation involves multiplying the input coordinates by the network's hyperparameters to compute the outputs, while backward propagation entails calculating gradients and updating the hyperparameters. Compared to the framework shown in Fig. 2, the inverse problem framework for a uniform beam introduces an additional trainable parameter in Network 2, which is updated via gradient descent and gradually converges to the true value during training. Additionally, a third sub-loss function is incorporated into the overall loss to integrate structural data into the model, as illustrated in Fig. 3.

Due to the similarity of frameworks, the model is able to compute the entire beam's displacement while simultaneously inferring its unknown density; however, its accuracy is lower than that of the forward problem. The loss functions for the inverse problem of a uniform beam can be expressed as follows:

$$L_{b,PDE} = \frac{1}{N_{b,PDE}} \sum_i \left(\frac{1}{X_A^4} \frac{\partial^4 \hat{W}}{\partial \bar{x}^4} + \hat{\alpha} \frac{1}{T_A^2} \frac{\partial^2 \hat{W}}{\partial \bar{t}^2} - \frac{f_{data}}{EI} \right)^2 \quad (35)$$

$$L_{b,data1} = \frac{1}{N_{b,data}} \sum_i \left(\frac{\partial^2 \hat{W}}{\partial \bar{t}^2} - \frac{\partial^2 W}{\partial \bar{t}^2} \Big|_{data} \right)^2 \quad (36)$$

$$L_{b,data2} = \frac{1}{N_{b,data}} \sum_i (\hat{W} - W_{data})^2 \quad (37)$$

$$L_{b,total} = \lambda_{PDE} L_{b,PDE} + \lambda_{data1} L_{b,data1} + \lambda_{data2} L_{b,data2} \quad (38)$$

where the values of λ_{PDE} , λ_{data1} , and λ_{data2} are set to 1, 100, and 10,000, respectively. It is worth noting that $\hat{\alpha}$ in Eq. 35 is the trainable parameter we set in Network 2, which is initialized randomly and ultimately converges to the predicted value of the unknown parameter upon completion of training.

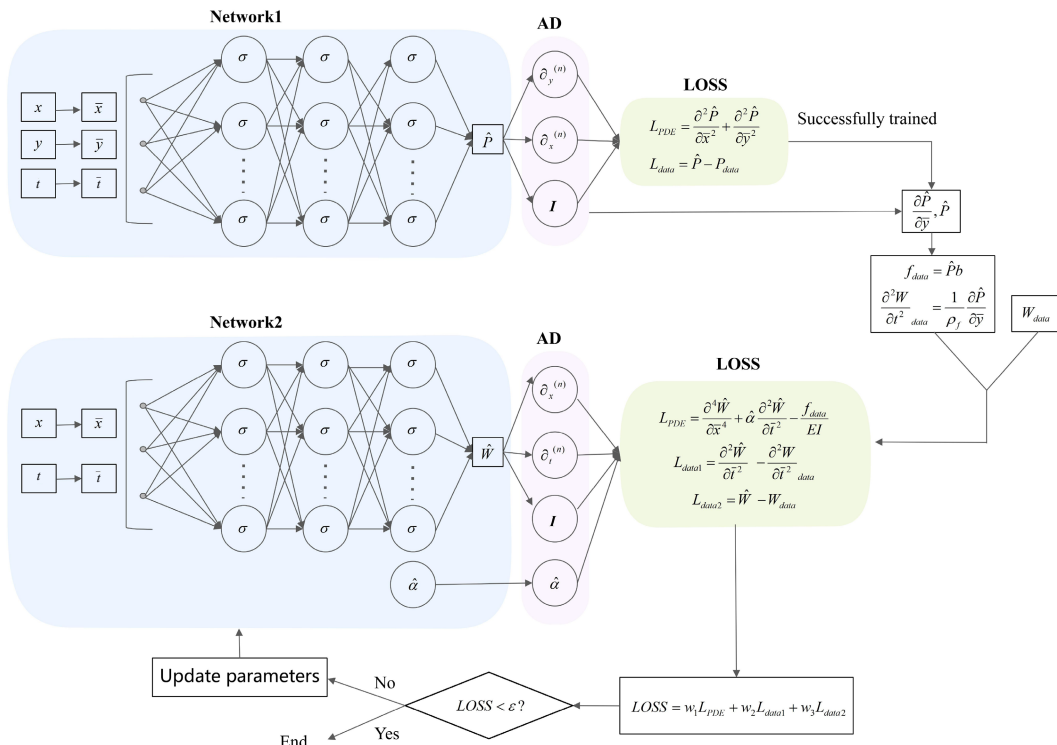


Figure 3 Inverse problem framework of FSI-PINNs for uniform beam. The basic framework of network1 and Network 2 is similar to Fig. 2, but an extra trainable parameter $\hat{\alpha}$ is included in Network 2 to infer the unknown constant density.

2.3.3 Inverse problem of non-uniform beam

For the uniform beam, the unknown parameter, density, is constant, so a single additional trainable parameter can be used to represent it. For the non-uniform beam, however, the density varies along the length of the beam, requiring a vector to represent its values, with each element corresponding to the density at a specific location. To accommodate this, we introduce an additional Network 3 to represent the unknown density, while Network 2 continues to represent the displacement. The input to Network 3 is the spatial coordinate x , since the beam density is a function of position only and does not vary with time. The outputs of Networks 2 and 3, along with their deviations, form the components of the loss functions, and their hyperparameters are updated simultaneously at each iteration. A detailed schematic of the non-uniform beam inverse problem framework is shown in Fig. 4.

The loss functions for the non-uniform beam are the same as those in Eqs. 35–38; however, the weights in Eq. 38 must be adjusted based on training performance, as discussed in a later section.

3 Experiments and results

3.1 Experiments setup

In this section, we apply our FSI-PINNs model to solve both forward and inverse problems. Prior to the experiments, the data

were obtained through numerical simulation, as described in Section 2.1. We assume that the fluid domain and the structure are initially at rest, with an initial velocity v_0 applied to the beam. The parameters of the fluid domain and structure are listed in Table 1. In total, 2×10^7 fluid pressure data points and 2×10^5 beam displacement data points were acquired through numerical simulation based on the finite difference method. In the experiments presented in this section, no more than 1% of the total data are used to train the FSI-PINNs model. Figure 5 illustrates the sampling strategy for the fluid domain and structure. In the 2D fluid domain, 10,000 training points are randomly sampled, as shown in Fig. 5a. Notably, the number of sampling points near the fluid–structure interface is higher than that near the bottom of the fluid domain. This approach is inspired by conventional numerical simulation mesh generation, where denser meshes are generated beside the interface to ensure computational convergence and achieve higher accuracy. Figure 5b shows the distribution of 1,000 sampling points on the structure.

The specific network architecture, including the number of layers and neurons, is chosen based on established practices in prior PINN literature [19,20]. Previous sensitivity analyses [19] have shown that once the network capacity surpasses a certain threshold, increasing the depth or width yields negligible improvements in accuracy. Furthermore, recent studies [20] indicate that training stability in multi-physics PINNs depends more on loss weighting and optimization strategies than on the network architecture itself.

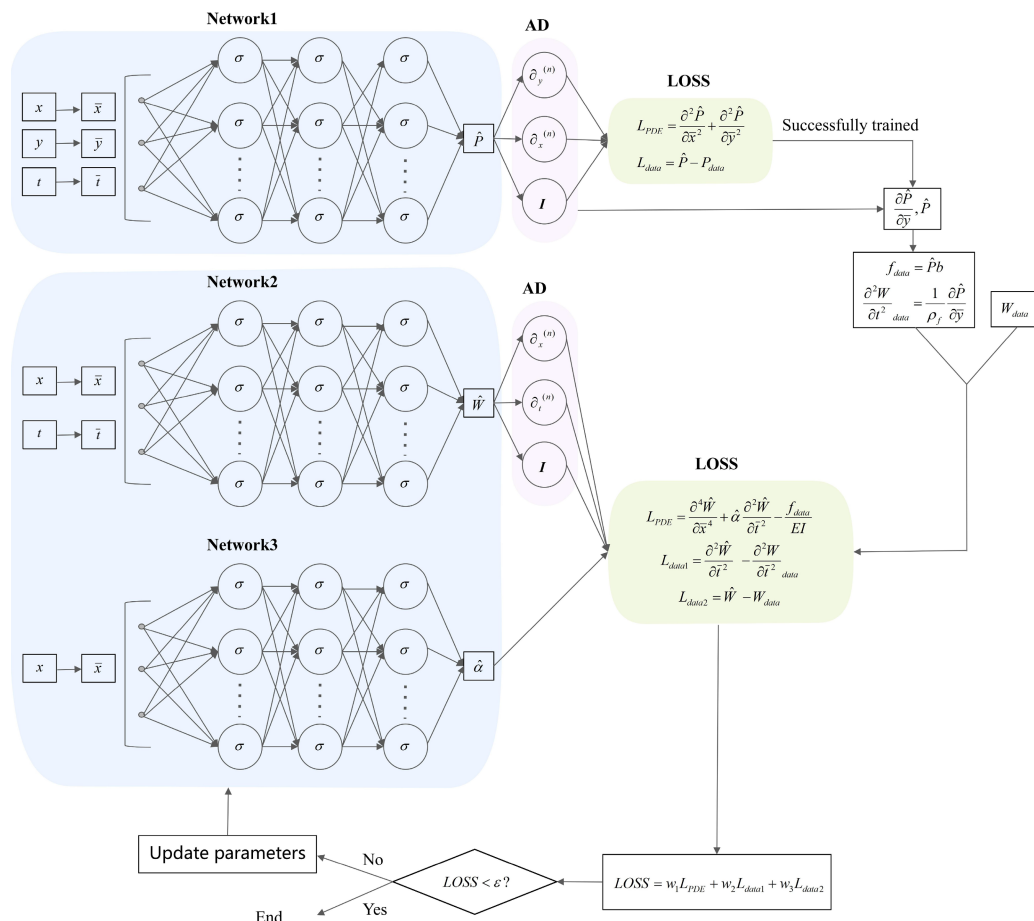


Figure 4 Inverse problem framework of FSI-PINNs for non-uniform beam. Network 1, with inputs (x, y, t) , approximates the fluid pressure $\hat{P}$. Network 2 takes (x, t) to approximate the displacement $\hat{W}$, while an additional Network 3 takes the spatial coordinate x as input to approximate the spatially varying parameter $\hat{a}(x)$.

Accordingly, a standard fully-connected architecture is employed for the scenarios described in Section 2.3, with depth and width ranging from 3 to 4 layers and 128 to 256 neurons per layer, which is sufficient to yield robust and optimal results. Specific parameter settings for Network 2 are provided in Table 2. Instead of optimizing both networks simultaneously, a decoupled sequential training strategy based on domain decomposition is adopted. This approach mitigates the challenges of training multi-physics systems, where fluid and structural dynamics may have vastly different scales and convergence rates. Standard normalization layers (e.g., BatchNorm) were excluded to preserve the smoothness required for accurately computing high-order PDE derivatives via AD. Input standardization and loss weighting were employed to mitigate scale disparities.

The proposed FSI-PINNs framework was implemented using the open-source deep learning library PyTorch. Data generation, pre-processing, and scientific computations were performed using NumPy and SciPy. All visualizations and contour plots presented in this study were created using Matplotlib. The training process was conducted on a workstation equipped with an NVIDIA A100 GPU to accelerate neural network optimization.

3.2 Forward problem

To demonstrate the effectiveness of FIS-PINNs under sparse data for forward problem, we conducted training experiments using different numbers of sampling points in both the beam and the fluid field. Table 3 presents the results of the forward problem for varying sampling densities on the beam and in the fluid domain.

Table 1 Parameters of physical model and numerical simulation.

Parameter	Symbol	Value
Length/height	L/H	1 m
Simulation duration	T	2×10^{-3} s
Young's modulus	E	70×10^9 Pa
Moment of inertia	I	1×10^{-6} m ⁴
Structural density	ρ_s	1×10^3 kg/m ³
Fluid density	ρ_f	876 kg/m ³
Section width	b	0.0154 m
Section height	h	0.0924 m

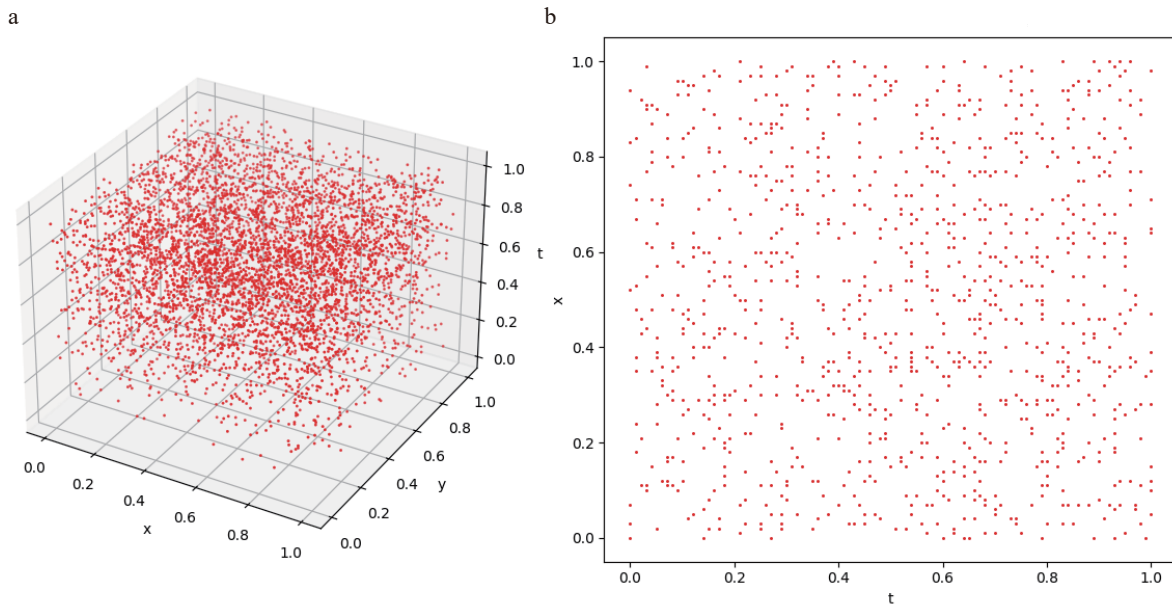


Figure 5 Sampling strategy of the fluid domain (Network 1) and the structure (Network 2): (a) 10000 points in the 2D fluid domain; (b) 1000 points on the structure.

Table 2 Parameters of the networks.

	Architecture	Optimizer	Learning rate	Iterations
Network 1	4×128	Adam	0.001	40,000
		L-BFGS	0.1	10,000
Network 2	4×256	Adam	0.001	60,000
		L-BFGS	0.1	10,000
Network 3	3×128	Adam	0.001	60,000
		L-BFGS	0.1	10,000

The results show that FSI-PINNs can predict beam displacement with high accuracy, achieving an L2 relative error of less than 1%. When the number of sampling points is sufficiently large, its impact on the network's accuracy is relatively minor, regardless of whether the points are in the fluid domain or on the structure. However, when the number of sampling points is limited, the number of fluid domain sampling points has a more significant effect on the final results than the number of beam sampling points. This is because insufficient data in the fluid domain leads to larger reconstruction errors, which in turn compromise the accuracy of beam displacement calculations. Even the incorporation of governing equations cannot fully offset the impact of such large errors.

To provide more details of the results, Fig. 6 demonstrates the predicted value and absolute errors when 1,000 points are sampled on both the beam and in the fluid field. From Fig. 6b, it can be observed that, due to the imposition of the hard constraint, the

absolute errors at the left and right boundaries are zero, indicating that the beam's boundary conditions are perfectly satisfied. In contrast, errors near the initial time are relatively high, as no initial condition was included in the loss function. Figure 7 presents the trends of the loss functions during the training of Network 1 and Network 2. Different loss functions exhibit distinct behaviors and fluctuate significantly during the initial phase of training, indicating the outputs rapidly approach the desired values. After several thousand iterations, the network establishes a consistent balance among the various sub-loss functions to achieve optimal convergence.

In addition to investigating the impact of the number of sampling points, we also conduct noise analysis on our FSI-PINNs. For the original data $\mathbf{D}'$, the noisy data $\mathbf{D}_{\text{noise}}$ can be expressed as

$$\mathbf{D}_{\text{noise}} = \mathbf{D} \cdot \mathbf{n}, \quad \mathbf{n} \sim N(\mu, \sigma^2) \quad (39)$$

Table 3 Results of FIS-PINNs for forward problem.

Number of sampling points in fluid field	Number of sampling points on beam	L2 relative error	
		Network 1	Network 2
1000	50		0.0091
	100		0.0058
	500	0.0012	0.0054
	1000		0.0033
100	1000	0.056	0.37
500		0.0042	0.022
1000		0.0012	0.0033
5000		0.00065	0.0053
10,000		0.00069	0.0027

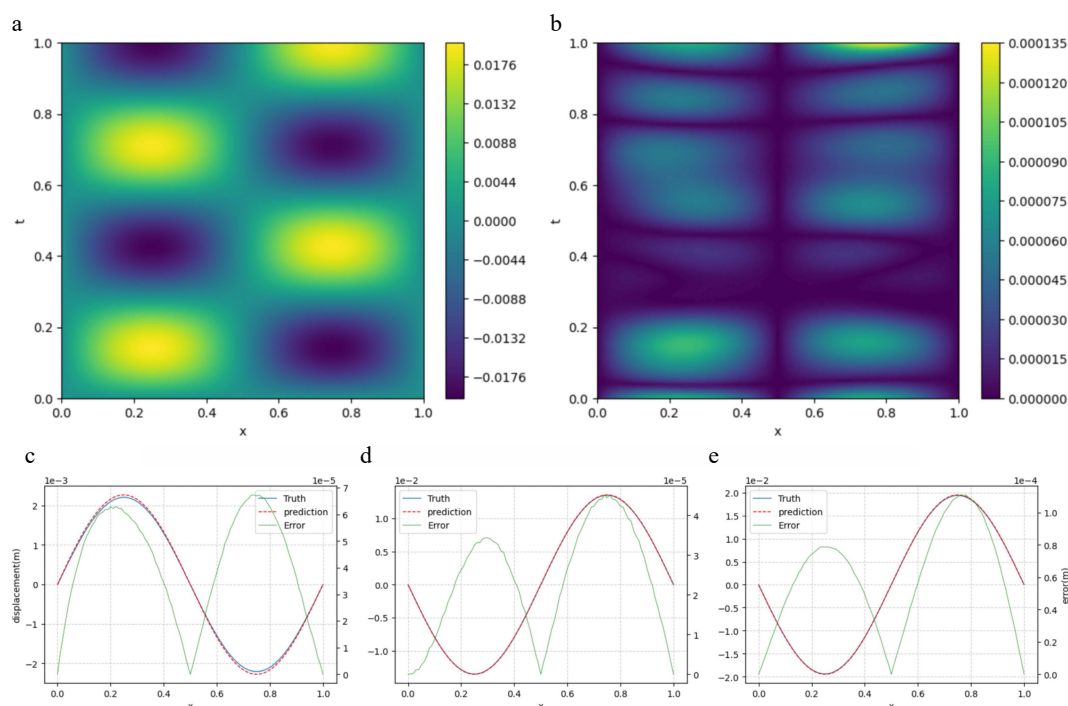


Figure 6 Results of FSI-PINNs for the forward problem with 1000 sampling points, both in the fluid field and on the beam. (a) Contour plot of prediction value. (b) Contour plot of absolute error between prediction and true value. (c)–(e) Comparison between prediction value and true value at $t = 0.01, 0.5, 0.99$, respectively.

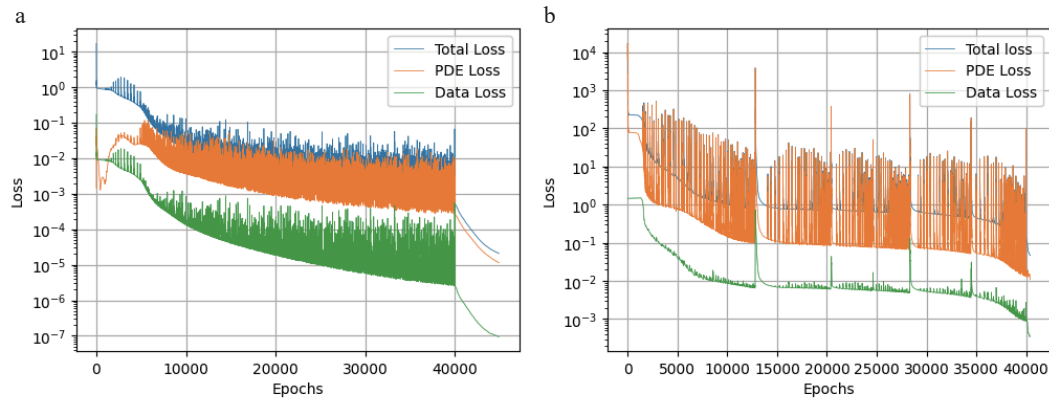


Figure 7 The trends of loss functions: (a) loss functions of Network 1 during training; (b) loss functions of Network 2 during training. The L-BFGS optimizer for Network 2 terminated automatically upon meeting the convergence tolerance. The visible fluctuations in Network 2 loss are attributed to the sensitivity of the fourth-order derivatives in the beam equation.

The data after adding noise, $\mathbf{D}'$, can be expressed as

$$\mathbf{D}' = \mathbf{D} + \mathbf{D}_{\text{noise}} \quad (40)$$

where $\mathbf{D}_{\text{noise}}$ follows a normal distribution with a mean of μ and a standard deviation of σ^2 .

In this section, the mean of the noise is set to zero, and the model's predictive performance is evaluated under different noise standard deviations of 0.01, 0.02, 0.05, 0.1, 0.15, and 0.2, with 1,000 sampling points in both the fluid field and on the beam. Table 4 presents the training results of the network under the influence of noisy data, which are further illustrated by the detailed contours and snapshots in Figs. 8 and 9. The analysis indicates that FSI-PINNs exhibit a certain degree of robustness against noise: when the noise standard deviation is as low as 0.01 or 0.02, the impact on model training is nearly negligible, with predictive relative errors remaining within 1%. As the noise standard deviation increases to about 0.15, the relative L2 error of the final prediction rises to about 0.05. However, when the noise standard deviation reaches 0.2, the relative L2 error sharply increases to 0.62. This indicates that the model has a certain “threshold” for noise tolerance: when the noise intensity is below this threshold, predictions remain accurate and stable; when the noise exceeds the threshold, the model's fitting capability is significantly compromised. Furthermore, it is observed that when the noise level exceeds 0.1, not only does the prediction error increase, but the training time also rises substantially. This suggests that excessive data noise can severely affect network training and disrupt the balance between different loss terms. For high-precision applications, it is therefore essential to minimize noise in the training data.

3.3 Inverse problem

3.3.1 Uniform beam

Regarding the inverse problem, owing to its structural similarity to the forward problem, parameter identification inherently involves the concurrent reconstruction of the physical field. We evaluate the accuracy of inferring unknown parameters while varying the number of sampling points on the beam, keeping the number of fluid domain sampling points fixed at 1,000. Only the results of Network 2 are presented here, as Network 1 is identical to that in the forward problem. As shown in Table 5, increasing the number of sampling points on the beam improves the model's ability to accurately compute the unknown parameter and predict the

Table 4 Results of noise analysis for forward problem.

Mean value of noise	Standard deviation of noise	L2 relative error	
		Network 1	Network 2
0	0.01	0.0021	0.0053
	0.02	0.0032	0.0063
	0.05	0.0075	0.012
	0.1	0.015	0.040
	0.15	0.021	0.066
	0.2	0.025	0.62

displacement of the entire beam.

When 2,000 points are randomly selected from the beam—corresponding to only 1% of the full dataset—the unknown parameter can be roughly inferred with an L2 relative error of about 1%. Remarkably, even when using only 100 beam data points, accounting for just 0.05% of all displacement data, the L2 relative error of the predicted parameter is only around 6.5%, demonstrating the significant potential of FSI-PINNs for more complex applications. However, as shown in Fig. 10, compared to the forward problem, displacement reconstruction in the inverse problem exhibits lower accuracy, with an L2 relative error of approximately 1%. This reduction in accuracy is attributable to the unknown parameter in the governing equation, which affects the training process and disrupts the balance among various sub-loss components.

Figure 11a shows the trends of the trainable parameter for different numbers of sampling points. Notably, the parameter undergoes dramatic changes during the first 5000 iterations, increasing from approximately 0 to around 800 before rapidly approaching the target value. This behavior is consistent with the trends of the loss functions, as illustrated in Fig. 11b. In subsequent iterations, the trainable parameter gradually converges to the true value, exhibiting only minor fluctuations.

3.3.2 Non-uniform beam

In this example, the initial velocity contains two full sine wave periods along the length of the beam, with all other parameters unchanged. To assess the predictive capability of the algorithm, two cases are considered. In Case 1, the beam density increases gradually along the length, from 1200 kg/m³ at $x = 0$ to over 1800 kg/m³ at $x = 1$, while it is opposite in Case 2, with the density declines from 2000 kg/m³ at $x = 0$ to around 1500 kg/m³ at $x = 1$.

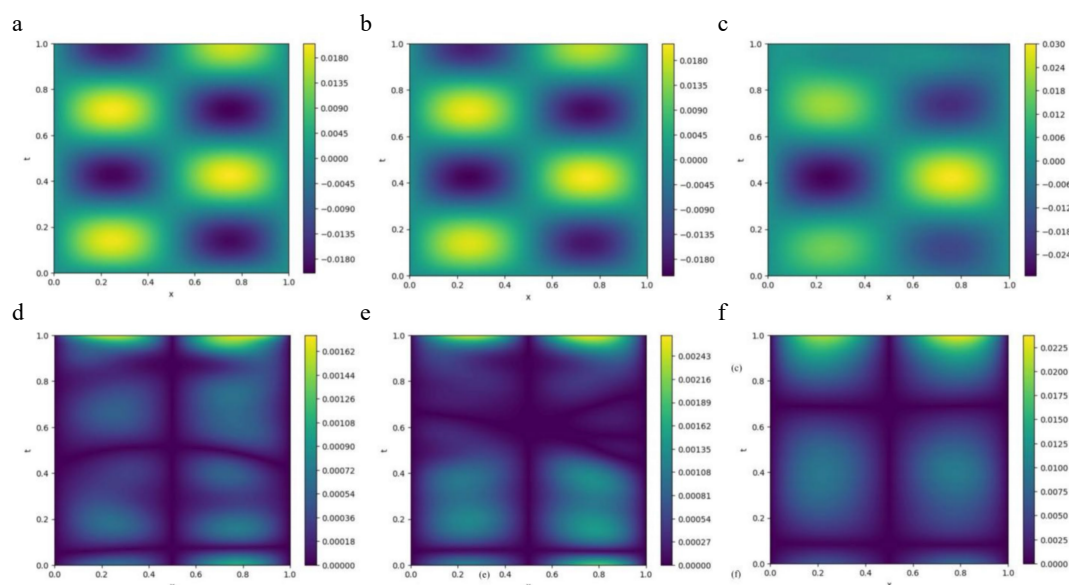


Figure 8 Noise analysis results of FSI-PINNs for forward problem with different standard deviations. (a)–(c) Contour plots of prediction value with noise standard deviations of 0.1, 0.15, and 0.2, respectively; (d)–(f) contour plots of absolute error between prediction value and true value with noise standard deviations of 0.1, 0.15, and 0.2, respectively.

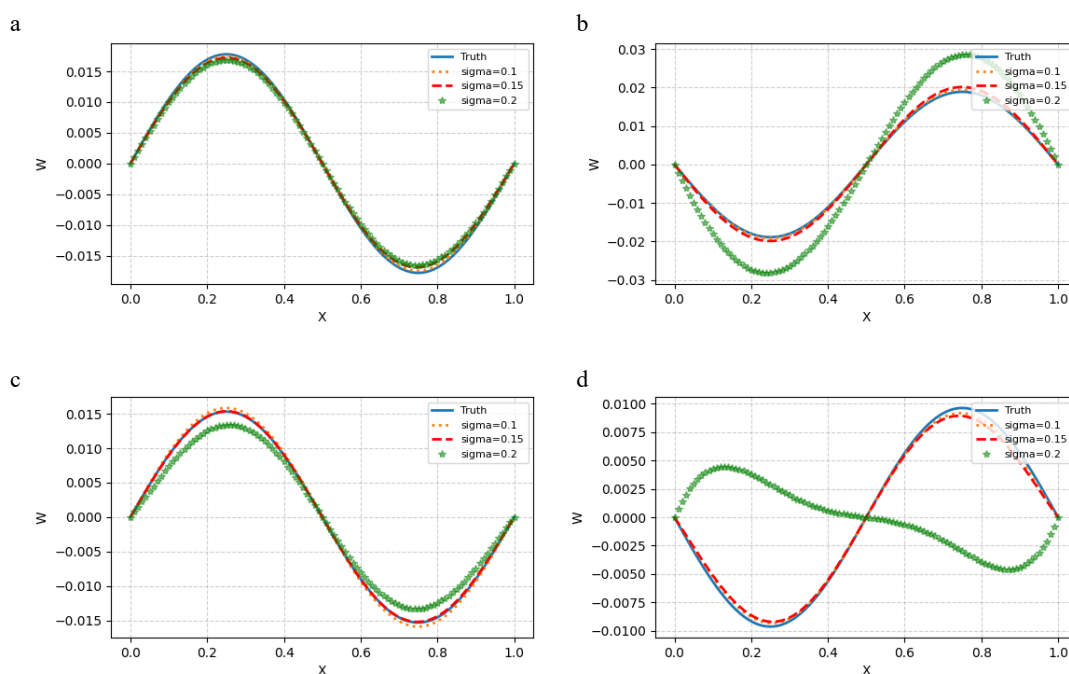


Figure 9 Comparison between prediction value and true value under various standard deviations at specific time. (a) $t = 0.1$; (b) $t = 0.4$; (c) $t = 0.65$; (d) $t = 0.9$.

Table 5 Results of FIS-PINNs for inverse problem on uniform beam.

Number of sampling points on beam	Network 1	L2 relative error	
		Network 2	Parameter
50	0.0012	0.031	0.0065
100		0.019	0.0062
500		0.014	0.0043
1000		0.011	0.0024
2000		0.011	0.0010

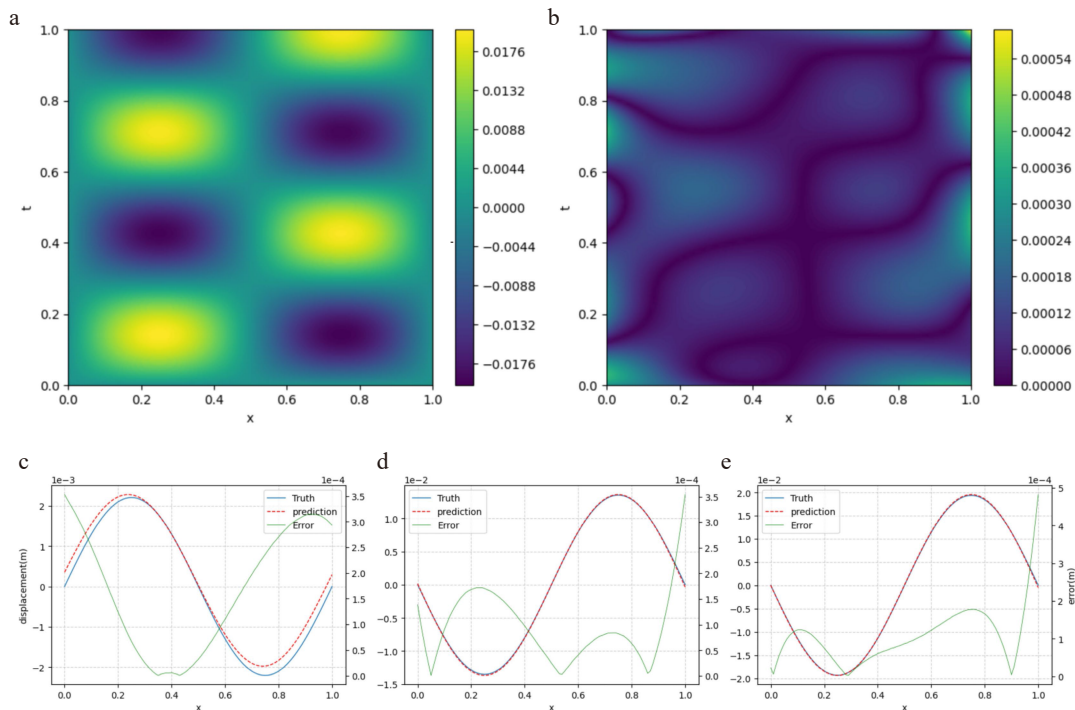


Figure 10 Results of FSI-PINNs for the Inverse problem of uniform beam with 1000 sampling points, both in the fluid field and on the beam. (a) Contour plot of prediction value. (b) Contour plot of absolute error between prediction and true value. (c)–(e) Comparison between prediction value and true value at $t = 0.01, 0.5, 0.99$, respectively.

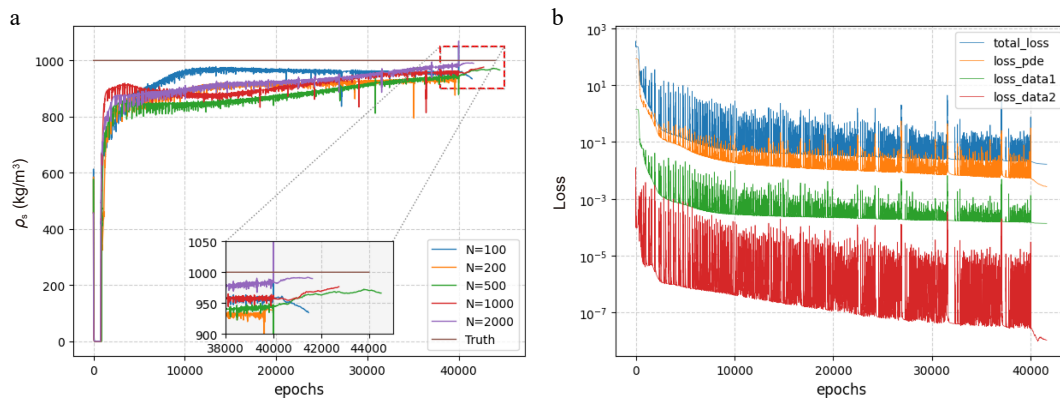


Figure 11 The trends plots of trainable parameter and loss functions in the inverse problem of uniform beam: (a) trends plot of trainable parameter over iterations; (b) loss functions of Network 2 over iterations.

Figures 12–14 present the results of these two cases. In Fig. 12, both panels show that, despite some errors, the model can predict the varying density of the non-uniform beam. In Fig. 12a, the predicted curve aligns closely with the true curve, with an error of only 1.5%. In Fig. 12b, the predicted values in the left half of the beam deviate more noticeably, with errors around 50, which are relatively larger compared to other regions. In the right half, where the density decreases sharply, the predictions match the true values well, with an overall error of 2.5%.

Figure 13 demonstrates the trends of the training losses over the epochs. Compared with the uniform beam (Fig. 11b), the losses of the non-uniform beam demonstrate greater complexity. During a prolonged training period, none of the three sub-losses exhibit a obvious decreasing trend, resulting in the Adam optimizer to fail in reducing the losses to an acceptable range even after 60,000 iterations. However, subsequent optimization using the L-BFGS optimizer results in a clear and sustained improvement over more

than 10,000 steps. Figure 14 presents the displacement results for Case 1 and Case 2, with L2 relative errors of 4.57% and 3.04%, respectively.

Despite the satisfactory results obtained, we also discovered a limitation of FSI-PINNs when applied to inverse problems of non-uniform beams. First, small changes in the physical model can lead to significant variations in the model parameters. For Case 1, the weight values in Eq. 38, λ_{PDE} , λ_{data1} , and λ_{data2} are 1, 1, and 1000, respectively, whereas for Case 2, they are 8, 300, and 100,000. These differences arise solely from altering the beam density between the two cases, highlighting the limited generality of PINNs. Second, adjustments to the weights can drastically affect the final results. For example, in Case 2, increasing or decreasing λ_{data1} or λ_{data2} by 10% can lead to parameter identification and displacement predictions that deviate significantly from the true values, as shown in Table 6. This demonstrates the strong parameter sensitivity inherent in PINNs.

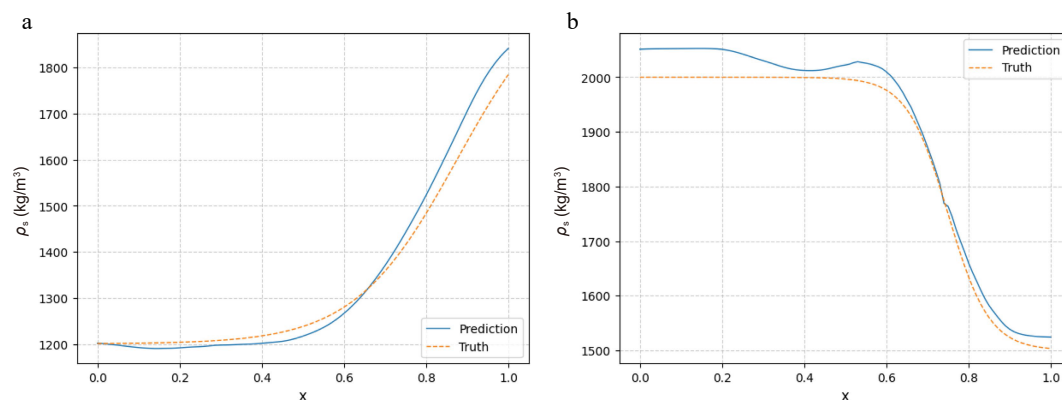


Figure 12 Comparison between predicted density value and true value of non-uniform beam: (a) Case 1; (b) Case 2.

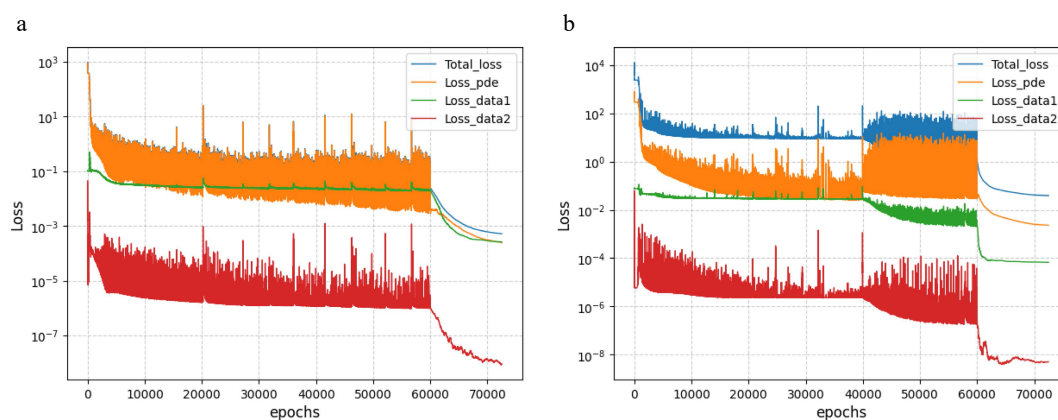


Figure 13 The trends plots of loss functions in inverse problem of non-uniform beam: (a) Case 1; (b) Case 2.

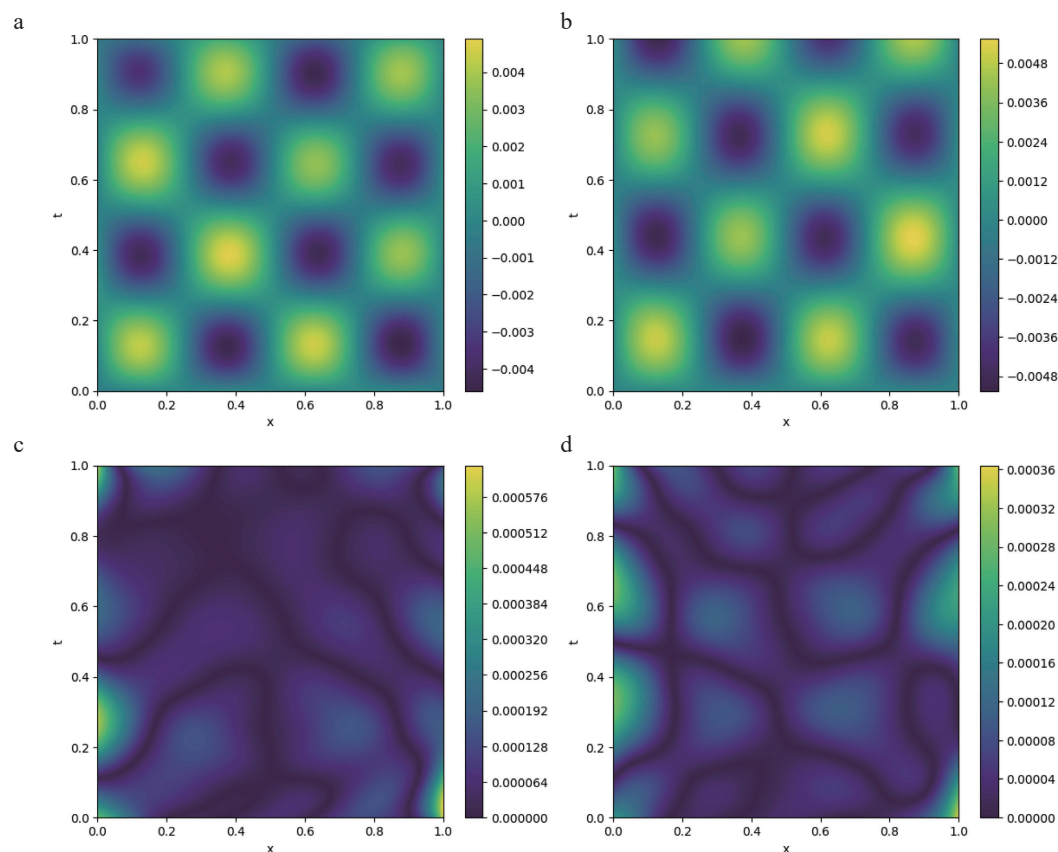


Figure 14 Results of FSI-PINNs for Inverse problem of non-uniform beam. (a) and (b) Contour plot of prediction value of Case 1 and Case 2; (c) and (d) contour plot of absolute error between prediction and true value of Case 1 and Case 2.

Table 6 Parameters of the networks.

λ_{PDE}	λ_{data1}	λ_{data2}	L2 relative error	
			Displacement	Density
8	300	100,000	0.030	0.017
		110,000	0.062	0.094
		120,000	0.027	0.038
		90,000	0.031	0.048
		80,000	0.65	0.91
8	310	100,000	0.031	0.021
	320		0.033	0.032
	290		0.035	0.056
	280		0.16	0.34

4 Summary

We proposed and validated a novel framework of PINNs, FSI-PINNs, for solving fluid–structure interaction problems. This framework can address both forward and inverse problems in two-dimensional fluid–structure interactions by employing two independent neural networks to simulate the fluid field and the structure, respectively, and coupling them through interface conditions. The experiments demonstrate that FSI-PINNs can accurately predict the dynamics of physical fields and infer unknown parameters in the governing equations, even when only very sparse data are available, highlighting its potential as a data-efficient method for numerical simulation.

The main conclusions of our study can be summarized as follows:

(1) Data efficiency and high accuracy: FSI-PINNs can handle fluid–structure interaction problems with less than 1% relative L2 error using sparse data that account for no more than 1% of the whole dataset. This result demonstrates the effectiveness and accuracy of the proposed framework and is particularly significant for practical engineering scenarios where obtaining large amounts of data are challenging.

(2) Framework effectiveness: Two neural networks are employed to decouple the computations of the fluid field and the structure, which are then coupled through Robin interface conditions. This approach has proven to be both effective and robust. Furthermore, the use of a hard constraint ensures that the boundary conditions of the structure are satisfied exactly, enhancing the stability and convergence of the model.

(3) Application prospect: FSI-PINNs perform well for both forward and inverse problems. For the forward problem, the model accurately predicts the displacement of the entire structure using minimal fluid field data and exhibits robustness to noisy data. For the inverse problem, which is challenging for traditional methods, FSI-PINNs can accurately infer the unknown density of the structure by training with data from both the fluid field and the structure. This highlights the framework's strong capability in parameter and system identification.

In summary, this study proposed a novel PINNs framework for addressing forward and inverse problems in fluid–structure interaction and validated its performance through numerical simulations. The results provide a useful reference for practical applications of PINNs in fluid–structure scenarios. It should be noted that the physical model used in this study is simplified and

primarily serves to demonstrate the feasibility of the proposed FSI-PINNs in relatively straightforward cases. For more complex applications, the framework's effectiveness needs further evaluation. Future work will focus on generalizing FSI-PINNs to a wider range of practical scenarios, including three-dimensional fluid–structure interaction models and large-deformation beam problems, as well as integrating advanced deep learning optimization techniques to enhance the model's accuracy and reliability.

Acknowledgements

None.

Author contribution statement

Xuanhan Xia: Conceptualization, Methodology, Software, Data curation, Writing original draft. Xiaofan Li: Supervision, Project administration. Jinfeng Zhang: Validation, Visualization. Shunxiang Cao: Formal analysis. Guangyao Wang: Methodology, Resources, Writing - review & editing, Funding acquisition. All the authors have approved the final manuscript.

Data availability

The data that support the findings of this study are included in the article or supplementary materials and is available from the corresponding author upon reasonable request.

Declaration of competing interest

All the contributing authors report no conflicts of interest in this work.

Funding

This research is financially supported by the National Natural Science Foundation of China (Grant No. 52301336), the Science and Technology Development Fund of Macau S.A.R. (Grant No. 0048/2025/ITP1), the University of Macau (Grant No. SRG2025-00004-FST).

Use of AI statement

None.

References

- [1] Dowell, E. H. (1975). *Aeroelasticity of Plates and Shells*. Springer: Dordrecht.
- [2] Theodorsen, T. (1949). General theory of aerodynamic instability and the mechanism of flutter, *American Institute of Aeronautics and Astronautics*, No. 496
- [3] Paidoussis, M. P., Issid, N. T. (1974). Dynamic stability of pipes conveying fluid. *J. Sound Vib.* 33, 267–294.
- [4] Williamson, C. H. K., Govardhan, R. (2004). Vortex-induced vibrations. *Annu. Rev. Fluid Mech.* 36, 413–455.
- [5] Perktold, K., Rappitsch, G. (1995). Computer simulation of local blood flow and vessel mechanics in a compliant carotid artery bifurcation model. *J. Biomech.* 28, 845–856.
- [6] Oden, J. T. (2006). *Finite Elements of Nonlinear Continua*. Dover Publications: Dover.
- [7] Donea, J., Huerta, A. (2003). *Finite Element Methods for Flow Problems*. John Wiley & Sons: Hoboken.
- [8] Peskin, C. S. (2002). The immersed boundary method. *Acta Numer.* 11, 479–517.
- [9] Jasak, H., Jemcov, A., Tuković, Z. (2007). OpenFOAM: a C++ library for complex physics simulations. In *International Workshop on Coupled Methods in Numerical Dynamics*. Dubrovnik, Croatia.
- [10] Brunton, S. L., Noack, B. R., Koumoutsakos, P. (2020). Machine learning for fluid mechanics. *Ann. Rev. Fluid Mech.* 52, 477–508.
- [11] Fukami, K., Fukagata, K., Taira, K. (2019). Super-resolution reconstruction of turbulent flows with machine learning. *J. Fluid Mech.* 870, 106–120.
- [12] Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P., Koumoutsakos, P. (2018). Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proc. Roy. Soc. A: Math., Phys. Eng. Sci.* 474, 20170844.
- [13] Verma, S., Novati, G., Koumoutsakos, P. (2018). Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proc. Natl. Acad. Sci. USA*. 115, 5849–5854.
- [14] Reddy, G., Celani, A., Sejnowski, T. J., Vergassola, M. (2016). Learning to soar in turbulent environments. *Proc. Natl. Acad. Sci. USA*. 113, E4877–E4884.
- [15] Zhao, T. J., Wang, S., Ouyang, C. J., Chen, M., Liu, C. Y., Zhang, J., Yu, L., Wang, F., Xie, Y., Li, J., et al. (2024). Artificial intelligence for geoscience: progress, challenges, and perspectives. *Innovation* 5, 100691.
- [16] Lagaris, I. E., Likas, A., Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* 9, 987–1000.
- [17] Dissanayake, M. W. M. G., Phan-Thien, N. (1994). Neural-network-based approximations for solving partial differential equations. *Commun. Numer. Methods Eng.* 10, 195–201.
- [18] Meade, A. J. Jr., Fernandez, A. A. (1994). The numerical solution of linear ordinary differential equations by feedforward neural networks. *Math. Comput. Modell.* 19, 1–25.
- [19] Raissi, M., Perdikaris, P., Karniadakis, G. E. (2019). 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.
- [20] Wang, S. F., Sankaran, S., Wang, H. W., Perdikaris, P. (2023). An expert's guide to training physics-informed neural networks. *arXiv preprint*, arXiv: 2308.08468.
- [21] Wang, S. F., Teng, Y. J., Perdikaris, P. (2021). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.* 43, A3055–A3081.
- [22] Cai, S. Z., Mao, Z. P., Wang, Z. C., Yin, M. L., Karniadakis, G. E. (2021). Physics-informed neural networks (PINNs) for fluid mechanics: a review. *Acta Mech. Sin.* 37, 1727–1738.
- [23] Raissi, M., Yazdani, A., Karniadakis, G. E. (2018). Hidden fluid mechanics: a Navier-Stokes informed deep learning framework for assimilating flow visualization data. *arXiv preprint*, arXiv: 1808.04327.
- [24] Samaniego, E., Anitescu, C., Goswami, S., Nguyen-Thanh, V. M., Guo, H., Hamdia, K., Zhuang, X., Rabczuk, T. (2020). An energy approach to the solution of partial differential equations in computational mechanics via machine learning: concepts, implementation and applications. *Comput. Methods Appl. Mech. Eng.* 362, 112790.
- [25] Yin, M. L., Zheng, X. N., Humphrey, J. D., Karniadakis, G. E. (2021). Non-invasive inference of thrombus material properties with physics-informed neural networks. *Comput. Methods Appl. Mech. Eng.* 375, 113603.
- [26] Jin, X. W., Cai, S. Z., Li, H., Karniadakis, G. E. (2021). NSFnets (Navier-Stokes flow nets): physics-informed neural networks for the incompressible Navier-Stokes equations. *J. Comput. Phys.* 426, 109951.
- [27] Auddy, S., Dey, R., Turner, N. J., Basu, S. (2024). GRINN: a physics-informed neural network for solving hydrodynamic systems in the presence of self-gravity. *Mach. Learn. : Sci. Technol.* 5, 025014.
- [28] Kharazmi, E., Zhang, Z. Q., Karniadakis, G. E. M. (2021). hp-VPINNs: variational physics-informed neural networks with domain decomposition. *Comput. Methods Appl. Mech. Eng.* 374, 113547.
- [29] Rao, C. P., Sun, H., Liu, Y. (2020). Physics-informed deep learning for incompressible laminar flows. *Theor. Appl. Mech. Lett.* 10, 207–212.
- [30] Wu, J. L., Xiao, H., Paterson, E. (2018). Physics-informed machine learning approach for augmenting turbulence models: a comprehensive framework. *Phys. Rev. Fluids* 3, 074602.
- [31] Mao, Z. P., Jagtap, A. D., Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. *Comput. Methods Appl. Mech. Eng.* 360, 112789.
- [32] Sun, L. N., Gao, H., Pan, S. W., Wang, J. X. (2020). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Comput. Methods Appl. Mech. Eng.* 361, 112732.
- [33] Zhou, W., Xu, Y. F. (2024). Damage identification for plate structures using physics-informed neural networks. *Mech. Syst. Sign. Process.* 209, 111111.
- [34] Liu, C., Wu, H. A. (2023). A variational formulation of physics-informed neural network for the applications of homogeneous and heterogeneous material properties identification. *J. Appl. Mech.* 15, 2350065.
- [35] Shukla, K., Di Leoni, P. C., Blackshire, J., Sparkman, D., Karniadakis, G. E. (2020). Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks. *J. Nondestruct. Eval.* 39, 61.
- [36] Hou, X. R., Zhou, X. Y., Liu, Y. (2024). Reconstruction of ship propeller wake field based on self-adaptive loss balanced physics-informed neural networks. *Ocean Eng.* 309, 118341.
- [37] Liu, T., Zhao, J. T., Huang, J. H., Li, Z. L., Xu, L. J., Zhao, B. (2024). Research on model predictive control of autonomous underwater vehicle based on physics informed neural network modeling. *Ocean Eng.* 304, 117844.
- [38] Cao, W. B., Song, J. H., Zhang, W. W. (2024). A solver for subsonic flow around airfoils based on physics-informed neural networks and mesh transformation. *Phys. Fluids* 36, 027134.
- [39] Moschou, S. P., Hicks, E., Parekh, R. Y., Mathew, D., Majumdar, S., Vlahakis, N. (2023). Physics-informed neural networks for modeling astrophysical shocks. *Mach. Learn. : Sci. Technol.* 4, 035032.
- [40] Cao, S. X., Wang, G. Y., Wang, K. G. (2021). A spatially varying

- robin interface condition for fluid-structure coupled simulations. *Int. J. Numerical Methods Eng.* 122, 5176–5203.
- [41] Schwing, A. G., Urtasun, R. (2015). Fully connected deep structured networks. *arXiv preprint*, arXiv: 1503.02351.
- [42] Lu, L., Pestourie, R., Yao, W. J., Wang, Z. C., Verdugo, F., Johnson, S. G. (2021). Physics-informed neural networks with hard constraints for inverse design. *SIAM J. Sci. Comput.* 43, B1105–B1132.
- [43] Alkhadhr, S., Almekkawy, M. (2023). Wave equation modeling via physics-informed neural networks: models of soft and hard constraints for initial and boundary conditions. *Sensors* 23, 2792.
- [44] Márquez-Neila, P., Salzmann, M., Fua, P. (2017). Imposing hard constraints on deep networks: promises and limitations. *arXiv preprint*, arXiv: 1706.02025.
- [45] Jagtap, A. D., Karniadakis, G. E. (2020). 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.
- [46] Jagtap, A. D., Kharazmi, E., Karniadakis, G. E. (2020). Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* 365, 113028.
- [47] Moseley, B., Markham, A., Nissen-Meyer, T. (2023). Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations. *Adv. Comput. Math.* 49, 62.



Open Access This article is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits reusers to distribute, remix, adapt, and build upon the material in any medium or format, so long as attribution is given to the original author(s) and the source, a link to the license is provided, and any changes made are indicated. See <https://creativecommons.org/licenses/by/4.0/>

© The author(s) 2026. Published by Tsinghua University Press.



清华大学出版社
Tsinghua University Press

SciOpen