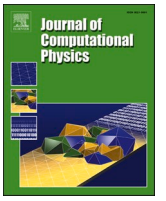
Journal of Computational Physics 515 (2024) 113284

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp



f-PICNN: A physics-informed convolutional neural network for partial differential equations with space-time domain

Biao Yuan [a], He Wang [b], Ana Heitor [a], Xiaohui Chen [a,*]

[a] Geomodelling and Artificial Intelligence Centre, School of Civil Engineering, University of Leeds, Leeds, LS2 9JT, UK
[b] Department of Computer Science, Faculty of Engineering Sciences, University College London, London, WC1E 6BT, UK

ARTICLE INFO

Keywords:
Physics-informed convolutional neural network
Nonlinear convolutional units (NCUs)
Memory mechanism
Partial differential equations
Finite discretization
Auto-regressive model

ABSTRACT

The physics and interdisciplinary problems in science and engineering are mainly described as partial differential equations (PDEs). Recently, a novel method using physics-informed neural networks (PINNs) to solve PDEs by employing deep neural networks with physical constraints as data-driven models has been pioneered for surrogate modelling and inverse problems. However, the original PINNs based on fully connected neural networks pose intrinsic limitations and poor performance for the PDEs with nonlinearity, drastic gradients, multiscale characteristics or high dimensionality in which the complex features are hard to capture. This leads to difficulties in convergence to correct solutions and high computational costs. To address the above problems, in this paper, a novel physics-informed convolutional neural network framework based on finite discretization schemes with a stack of a series of nonlinear convolutional units (NCUs) for solving PDEs in the space-time domain without any labelled data (f-PICNN) is proposed, in which the memory mechanism can considerably speed up the convergence. Specifically, the initial conditions (ICs) are hard-encoded into the network as the first time-step solution and used to extrapolate the next time-step solution. The Dirichlet boundary conditions (BCs) are constrained by soft BC enforcement while the Neumann BCs are hard enforced. Furthermore, the loss function is designed as a set of discretized PDE residuals and optimized to conform to physics laws. Finally, the proposed auto-regressive model has been proven to be effective in a wide range of 1D and 2D nonlinear PDEs in both space and time under different finite discretization schemes (e.g., Euler, Crank Nicolson and fourth-order Runge-Kutta). The numerical results demonstrate that the proposed framework not only shows the ability to learn the PDEs efficiently but also provides an opportunity for greater conceptual simplicity, and potential for extrapolation from learning the PDEs using a limited dataset.

1. Introduction

Partial differential equations (PDEs) are critical for accurately describing a range of physical phenomena in various fields such as heat transfer, wave dynamics, fluid mechanics, electrodynamics and quantum mechanics. Despite the success of traditional approaches in solving PDEs, challenges remain in terms of computational convergence and the solution of inverse problems. Recently, physics-informed neural networks (PINNs) approach has been emerging as a way to address these challenges because it combines the

* Corresponding author.
E-mail address: X.Chen@leeds.ac.uk (X. Chen).

https://doi.org/10.1016/j.jcp.2024.113284
Received 24 January 2024; Received in revised form 19 June 2024; Accepted 13 July 2024
Available online 15 July 2024

strengths of both physical principles and deep learning to model and simulate the PDE systems [6,8,11,20,26,57,78]. Due to the universal approximation capacity of neural networks, PINNs demonstrate strong potential as surrogate models for nonlinear systems, both in forward inference and inverse problems. Generally, the primary challenge for neural network-based solvers is to efficiently solve PDEs for a variety of complex problems. Many researchers have employed diverse methods to conduct related studies focussing on PINNs for PDEs. For instance, the performance of PINNs can be highly improved by adopting adaptive activation function [24], adaptation weights [41,42], self-adaptive loss function or loss function optimization [3,25,53,66,75,79], adaptive sampling methods [68,72] and optimal PINNs initialization [36]. Other studies focus on domain decomposition for enhancing computational efficiency [13,22,27,32,44,52] and parallel computation [43,64]. Furthermore, convergence analysis [63,74] and uncertainty quantification [80,86] have been proposed to measure the performance of PINNs. Nevertheless, it has been found that the learning ability of PINNs is substantially impacted when tackling more complex problems [30] and better algorithms in terms of performance optimization are required in that case. In recent years, various improved models and libraries for PDEs have been introduced. For instance, Lu et al. [39] proposed the Python library DeepXDE for PINNs for solving problems in computational science and engineering. In addition, the operator learning was proven to be efficient for learning various nonlinear operators from data, such as DeepONet [7,29,38,69], Fourier Neural Operator (FNO) [33]. Traditional grid-based methods, such as Finite Difference Method (FDM), Finite Volume Method (FVM), and Finite Element Method (FEM), have limitations in terms of computational dimensionality and complexity [4,65,87]. While operator learning methods are superior to vanilla grid-based methods in higher dimensional problems, they tend to be computationally intensive and require high-quality training data [2,50,60].

Inspired by the development and successful application of PINNs ideas in related fields, many studies showcased examples on how neural network models could be used for solving PDEs more efficiently. For instance, the Recurrent Neural Networks (RNNs) [21,45], the Residual Neural Networks (ResNets) [10,73,81], the Generative Adversarial Networks (GANs) [5,14,56,77], the Graph Neural Networks (GNNs) [25,17], and the Transformers [34,35,46,70], et.al. Among them, convolutional neural networks (CNNs) have been proven to have faster convergence and higher efficiency than fully connected neural networks in solving time-dependent PDEs [1,15, 18,58,60,71,83,85]. CNNs are analogous to finite different stencils in translationally equivariant PDE discretizations and share the same structures as the multigrid method [26]. For example, Özbay et al. [47] proposed a CNNs framework to solve the Poisson equation. The proposed CNNs can handle different boundary conditions, grid sizes, and grid resolutions without re-training, and can also speed up the convergence process. DiscretizationNet was proposed for Navier-Stokes equations using a generative CNN encoder-decoder model based on finite volume discretization to compute PDE residuals and update network weights [59]. Moreover, a method for learning time-dependent PDEs from a few data using a CNN framework with a linear and a nonlinear part was introduced [55]. This method can avoid overfitting problems by hard encoding the boundary conditions and using an auto-regressive framework for time-series data. For spatiotemporal PDEs, Ren et al. [60] proposed PhyCRNet using an encoder–decoder convolutional long short-term memory (ConvLSTM) network for solving PDEs without any labeled data, in which the loss function is defined as the aggregated discretized PDE residuals, while the I/BCs are hard-encoded in the network to ensure forcible satisfaction. In addition, Mavi et al. [40] optimized PhyCRNet by employing a novel activation function and the nonlocal Peridynamic Differential Operator (PDDO) as a convolutional filter to improve the predictive capability of the learning architecture for physics with periodic behavior. For backward stochastic PDEs (B-SPDEs), Dai [12] used a generic CNN to solve and simulate many real-world system dynamics through conditional expectation projection and discrete iteration, while Xiao and Qiu [76] studied the high-dimensional PDE cases. Most similar models are mainly based on regular or simple domains and the PDEs upon nonuniform grids need to be studied. Therefore, based on coordinate transformation, Gao et al. [16] proposed PhyGeoNet aiming to learn solutions of parametric PDEs on irregular domains without the need for any labeled data, which is shown to be effective and efficient for various PDE problems with different types of boundary conditions and parameters. Other models based on CNN for solving PDEs or learning dynamics [19,37,67,82,85] also prove that it is more intuitive to consider CNN on related problems.

In general, research on the use of recurrent auto-regressive models based on convolutional neural networks and using different discretization schemes to improve the approximation of discrete physical systems is still limited. The main contribution of this study is summarized as follows. (1) A novel physics-informed convolutional neural network framework based on finite discretization schemes with a series of nonlinear convolutional units (NCUs) for solving PDEs in the space-time domain without any labeled data (f-PICNN) is proposed. The key component of the framework is NCU, which is designed to be flexible enough to approximate various operators. (2) The utilization of NCU permits the network to possess a memory mechanism, whereby the trained network parameters from the preceding time step are used as initialization parameters for the ensuing time step. In this way the convergence at a given time step is considerably sped up and enables the proposed model to proffer from fast yet precise inferences in uncomplicated chronological issues problems (e.g., time-dependent PDEs). (3) The contribution of the loss function mainly comes from the physical governing equations, which are finitely discretized over the domain. Several finite discretization schemes are adopted in this physical discrete system to improve stability and accuracy. The convolution kernels are used to approximate the finite difference operators and integrate the 4^{th}-order Runge-Kutta method into the network to improve the approximation accuracy based on mathematical principles. (4) Building upon prior research [15,55,60], a boundary processing method is used and incorporates relevant techniques from computer vision to carefully design the network output to satisfy the boundary conditions of PDEs. (5) As a potential alternative for surrogate modeling and inverse problems, numerical experiments are conducted to verify the proposed methods. The numerical outcomes demonstrate the superiority of f-PICNN in terms of its conceptual simplicity, mathematical plausibility, and favorable extrapolation performance compared to vanilla PINN-type solvers.

For enhanced clarity, the paper is organized in several sections. Section 2 outlines the problem of solving PDE systems via neural networks. The framework of the proposed method is described in Section 3. Section 4 introduces the reference solution and baseline vanilla PINN employed for comparison. In Section 5, numerical experiments are conducted to compare the performance of the

proposed models with baseline methods. Lastly, the discussion and conclusion are summarized in Section 6.

2. Problem statement

In this study, the general form of a series of multidimensional and nonlinear PDEs in the space-time domain is considered as follows:

$$u_t + N[u, \theta, \gamma] = 0, \ x \in \Omega, \ t \in [0, T] \tag{1}$$

$$u(x, 0) = u_0(x), \ x \in \Omega, t = 0 \tag{2}$$

and

$$B[u, \nabla_x u, \ldots] = 0, x \in \partial\Omega, \ t \in [0, T] \tag{3}$$

where $u(x,t)$ is the solution of the PDE in the spatial domain $x \in \Omega$ and the temporal domain $t \in [0,T]$; u_t is the time derivative of u with respect to t; $N[\cdot]$ is a nonlinear differential operator parameterized by θ acting on u with respect to x. γ indicates the coefficient of the PDE. The initial condition of the PDE is set to $u_0(x)$ and the boundary condition is considered as $B[u, \nabla_x u, \ldots, x \in \partial\Omega, \ t \in [0,T]] = 0$, and the $\partial\Omega$ indicates the boundary of the spatial domain.

The general goal is to design neural network-based methods to solve PDEs in the space-time domain mainly by analyzing their physical dynamics forward under the given governing equations. These approaches can serve as numerical solvers and surrogate models for time-dependent PDEs through given initial and boundary conditions and have the potential to solve the corresponding inverse problems. Previous studies have reported that convolutional neural networks have more advantages than fully connected neural networks when solving PDEs in terms of convergence time (shorter) and accuracy (higher) [1,15,18,58,60,71,83,85]. In addition, the parameters of convolution calculation are relatively few [60,83]. One-dimensional spatial training data can be regarded as word vectors in natural language processing, while two-dimensional spatial data can be regarded as image-like data in computer vision tasks. The entire neural network training process is unsupervised and does not have any labeled data, only the initial conditions and boundary conditions are given. In this study, the entire domain is discretized while considering mainly the regular physical domains, where both time and space are uniformly discretized and convolution filters can be easily applied. The solution of the PDE in the entire domain is sequentially extrapolated along the time axis from the given initial condition as the exact solution of the first time step.

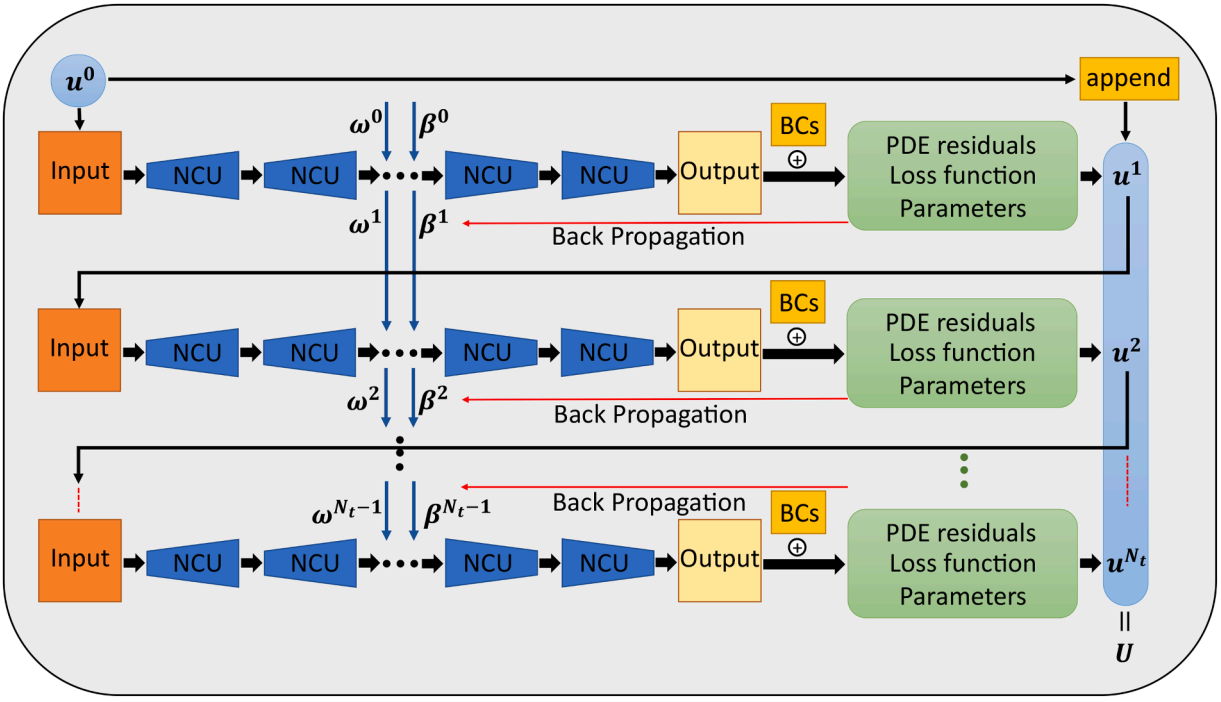


Fig. 1. The neural network architecture of the f-PICNN with Nonlinear Convolutional Units (NCUs) with input, hidden and output units for time-dependent PDEs with spatial and temporal coordinates (x,t). The heights on the left and right sides of the NCU indicate the relative number of input and output channels (number of neurons). The ω and β are the unknown trainable parameters in f-PICNN. u_0 is the given initial condition served as the training data at the first time step t_0 to extrapolate the solutions at following time points. U denotes the entire solution space (tensor). Note: The activation function employed at the hidden units is Tanh or ReLU. Finite differentiation is adopted to compute the partial derivatives in the governing equation that serves as a physical constraint for optimization together.

The loss function of the neural network training process is constrained by physical laws. The main objective is to approximate the solution of the equation by training neural networks to optimize the loss function composed of PDE residuals. A detailed introduction to the method is including in the subsequent section.

3. Methodology

In this section, a physics-informed model based on convolutional neural networks is proposed for learning PDEs in the space-time domain. Prior research has indicated that convolutional neural networks are more convenient and efficient than fully connected neural networks in solving time-dependent PDEs [55]. In this study, the main focus is not to demonstrate that the proposed model surpasses traditional numerical methods or other PINN models, but rather to provide novel perspectives and alternatives with both time efficiency and desirable accuracy for surrogate modelling in simulating the PDE systems. For example, when solving the Navier-Stokes equation, Gao et al. found that PINN could not accurately predict the pressure distribution, while the results of the CNN-based framework were in good agreement with the CFD benchmark with faster convergence and higher accuracy. It makes the CNN-based model a preferred choice [15]. This shows potential not only in terms of faster inference in forward problems but also in data-driven scientific computation such as the robustness in inverse problems against random perturbations caused by white noises, as well as higher-dimensional problems especially those facing computational difficulties with traditional solvers or other PINN solvers. Therefore, a novel f-PICNN structure constructed from nonlinear convolutional units (NCUs) based on convolutional operation is proposed in this paper. The key is to construct a loss function based on the PDE residuals, while the computation of the derivative terms is obtained from the convolutional operation through the finite discretization scheme, that is to say, the basic principle is to employ a series of convolutional kernels to approximate the differential operators. There are different finite difference operators and iterative optimization schemes in different finite discretization schemes. The Euler, Crank-Nicolson and 4th Runge-Kutta schemes are considered in this study to verify the effectiveness of the convolution kernels approximating the finite difference operators. Both soft and hard constraints are considered for the boundary conditions to demonstrate their effects in the proposed model. The network structure, mathematical concepts and implementation details of the proposed model are discussed in this section.

3.1. Neural network architecture: f-PICNN

In this section, the proposed architecture of f-PICNN is outlined, which comprises of a series of nonlinear convolutional units (NCUs), finite differentiation, auto-regressive process (AR), BC encoding, PDE residual calculation and loss function optimization. The schematic diagram is shown in Fig. 1, which demonstrates that the f-PICNN model receives the state at the previous time step as input and produces the output state for the next step.

$$u^{t+1} = f^{cnn}(u^t) \tag{4}$$

Where u^t and u^{t+1} represents the state variable at time step t and $(t+1)$, respectively. f^{cnn} denotes the discrete dynamical system surrogated by f-PICNN.

In f-PICNN, the operations required to solve PDEs in each unit (i.e., nonlinear convolutional unit, NCU) are encapsulated. This is illustrated in Fig. 2. The depth of f-PICNN is $[input, n_1, n_2, \ldots, n_r, output]$, where n_r is the number of output neurons of the r-th NCU, which is also the number of channels for the input of the $(r+1)$-th NCU. The entire network contains $(r+1)$ NCUs, and the output of the last NCU is also the output of the entire network. The number of initial input channels of the entire network is the number of solution variables of the PDE, and only the cases of a single solution variable u are considered in this study. The output of last NCU at former time step serves as input to first NCU at next time step until the final output produces a solution or prediction. The weights and biases of the proposed neural network are learned through a process named backpropagation.

Fig. 2 shows the structure of the $(r+1)$-th NCU in the f-PICNN framework, where $\hat{\tau}(\cdot)$ is the pre-processing or normalization operations on the input data; $\odot$ denotes the convolutional operation. Conv1d is applied for the one-dimensional case. The shape of 1D input data is (N, C, L), where N is the batch size, C is the number of features and L is the length of the one-dimensional input data, while the Conv2d is applied in the two-dimensional case and the shape of 2D input data is (N, C, H, W), where H and W denote the height and width of the 2D image-like data; $\boxed{\Omega}$ represents the boundary processing operation. n_r is the number of output neurons of the r-th NCU

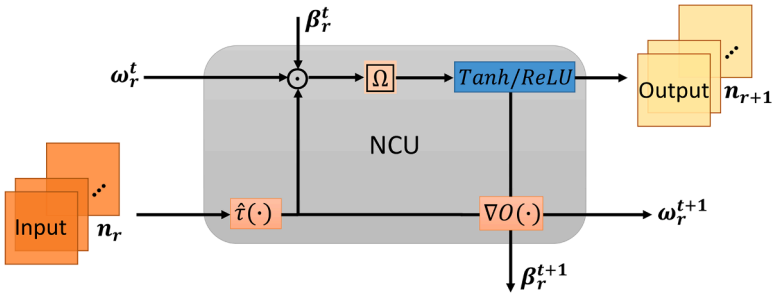


Fig. 2. The single nonlinear convolutional unit (NCU) at time t.

and is also the number of input neurons of the $(r+1)$-th NCU; n_{r+1} represents the number of output neurons of the $(r+1)$-th NCU. Such a NCU design makes it more flexible to approximate differential operators.

Furthermore, ω_r^t and β_r^t are the initialization network parameters of the r-th NCU at time t, and ω_r^{t+1} and β_r^{t+1} are the network parameters stored in the r-th NCU after optimization as the initialization parameters of the r-th NCU at the next time point, namely the memory mechanism (Fig. 3), which is also the novelty of this work. In many PDE systems, the dynamics of two adjacent time steps are evolutionarily similar, thus the dynamics of the previous step can be used to infer the dynamics of the next step. The network parameters trained at the previous time point can be used as the initial parameters of the network at the next time point, which can greatly accelerate the convergence process of the network. The Tanh or ReLU are applied as the activation function for the NCU, and $\nabla O(\cdot)$ indicates the optimization of the NCU, which has the functions of backpropagation, optimization and adjustment of network parameters, and storing optimized parameters as initialization parameters for the next step.

In this study, the model training was performed on a NVIDIA Tesla T4 GPU, PyTorch is a Python-based open-source machine learning library used for training and developing deep learning models [49], which is developed by Facebook's AI research team and provides an easy-to-use platform for establishing neural networks and other machine learning models.

3.2. Discretization schemes

As introduced in the previous sections, the proposed method is based on the finite discretization scheme using a series of convolution kernels in the convolutional neural network to approximate the finite difference operators. The partial derivatives should be computed before the optimization of the loss function. The finite difference method (FDM) is used with different discretization schemes to solve approximate solutions to PDEs and merge the principles into the f-PICNN to obtain desirable results. The idea is to establish a grid for the independent variables and discretize the PDEs, turning a continuous problem into a discrete problem of finite multiple equations. To establish a numerical solution to a differential equation, its discretization is critical. By using the numerical integration formula or Taylor Series Expansion (TSE) method, the general differential equation is considered using the initial conditions as follows:

$$\begin{cases} \dfrac{du}{dt} = f(t, u^t), t_0 \le t \le t_n \\ u^{t_0} = u_0 \end{cases} \tag{5}$$

When calculating numerical integrals using the rectangular formula:

$$u^{t+1} - u^t = \int_t^{t+1} f(t, u^t)dt \approx \Delta t \cdot f(t, u^t) \tag{6}$$

Or using Taylor Series Expansion (TSE):

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \ldots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + O_n(x) \tag{7}$$

Where $O_n(x) = O[(x - x_0)^n]$ is the truncation error from the spatial direction, and the first-order Taylor Series Expansion (TSE) is considered to approximate the solution:

$$u^{t+1} = u^t + \Delta t \cdot \frac{du}{dt} + O(\Delta t^2) \approx u^t + \Delta t \cdot f(t, u^t) \tag{8}$$

Where $O(\Delta t^2)$ is the second-order local truncation error about Δt, as long as the method is stable, the local truncation error can give a good expectation of the overall error. Thus, the iteration equation can be figured out by summarizing the above results:

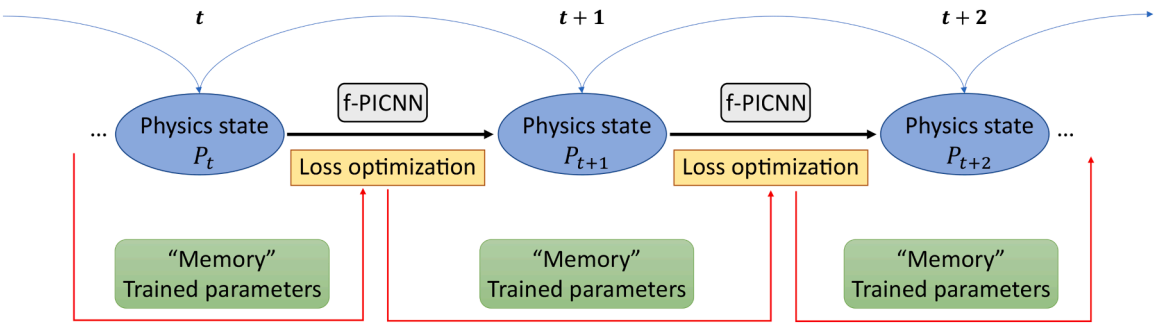


Fig. 3. The illustration of the memory mechanism in f-PICNN.

$$\begin{cases} u^{t_0} = u_0 \\ u^{t+1} \approx u^t + \Delta t \cdot f(t, u^t) \end{cases} \tag{9}$$

Based on the finite discretization scheme above, the following difference operators are considered to approximate the differential operators of PDEs:

$$\frac{\partial u}{\partial t} = \frac{u(x, t+\Delta t) - u(x,t)}{\Delta t} + O(\Delta t) \tag{10}$$

$$\frac{\partial u}{\partial x} = \frac{u(x+\Delta x, t) - u(x,t)}{\Delta x} + O(\Delta x) \tag{11}$$

First, the entire spatial domain is separated into equal-distance grids Δx. Meanwhile, the time is also divided into small segments Δt. The $O(\cdot)$ represents the local truncation error, $O(\Delta t)$ and $O(\Delta x)$ represent the discretized errors from the time direction and the space direction respectively, and both have first-order error accuracy. In addition, the second-order central difference formula concerning the second derivative of x is:

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+\Delta x, t) - 2u(x,t) + u(x-\Delta x, t)}{\Delta x^2} + O\left(\Delta x^2\right) \tag{12}$$

The above equations denote the commonly used Euler method when solving differential equations. In one-dimensional situations, the solution tensor u can be written as:

$$u(x,t) = \left[u_0^t, u_1^t, u_2^t, \ldots, u_{i-1}^t, u_i^t, u_{i+1}^t, \ldots, u_{N_x}^t\right] \tag{13}$$

The convolution kernel size is 3, which corresponds to the three-point difference format of FDM in one-dimensional problems. The implementation area of each finite-difference filter or convolution kernel of size 3 is denominated as the local receptive field, which is recorded as $S_{u_i^t}^1$ in one-dimensional cases at time t. It is mainly related to the values of adjacent points in the spatial domain.

$$S_{u_i^t}^1 = \left[u_{i-1}^t, u_i^t, u_{i+1}^t\right]. \tag{14}$$

While in two-dimensional problems, the solution tensor u can be defined as:

$$u(x,y,t) = \begin{bmatrix} u_{0,0}^t, & u_{0,1}^t, & \cdots & , u_{0,N_x}^t \\ u_{1,0}^t, & u_{1,1}^t, & \cdots & , u_{1,N_x}^t \\ \cdots & \cdots & , u_{j,i}^t, & \cdots \\ u_{N_y,0}^t & , u_{N_y,1}^t & \cdots & , u_{N_y,N_x}^t \end{bmatrix} \tag{15}$$

And the local receptive field $S_{u_i^t}^2$ in two-dimensional cases at time t is:

$$S_{u_i^t}^2 = \begin{bmatrix} u_{j-1,i-1}^t, & u_{j-1,i}^t, & u_{j-1,i+1}^t \\ u_{j,i-1}^t, & u_{j,i}^t, & u_{j,i+1}^t \\ u_{j+1,i-1}^t, & u_{j+1,i}^t, & u_{j+1,i+1}^t \end{bmatrix} \tag{16}$$

In the entire spatial definition domain, the x and y coordinate domains are divided into (N_x+1) and $\left(N_y+1\right)$ discrete points, respectively. Then the partial derivatives can be approximated according to Eq. (10) as follows:

$$\frac{\partial u^t}{\partial t} \approx \frac{u^{t+1} - u^t}{\Delta t} \tag{17}$$

Where u^t and u^{t+1} denote the solution at t and $(t+1)$ time step. This is the time difference scheme of the explicit first-order forward Euler method. Its essence is to use the function value and first-order derivative (slope) of the t-th point to approximately solve the function value of the $(t+1)$-th point. Therefore, while its accuracy is not high, the advantage is that it does not require an iterative solution. Von Neumann stability analysis [9] estimates the amplification or expansion of the error. For a stable method, the step size must be selected so that the error amplification factor is not greater than 1. Because the stability of the explicit difference method depends on the choice of step size, it is called conditionally stable. If the unconditionally stable method is required, we can consider the implicit Euler method of backward time difference, which is stable for any chosen step size according to Von Neumann analysis. Then, the first-order derivative of the spatial variable can be approximated by the following Upwind scheme:

$$\frac{\partial u_i^t}{\partial x} \approx \frac{1}{\Delta x}(0, -1, 1)\left[u_{i-1}^t, u_i^t, u_{i+1}^t\right]^T = \frac{1}{\Delta x} F_{US}^1 * S_{u_i^t}^1 \tag{18}$$

Or Leapfrog scheme:

$$\frac{\partial u_i^t}{\partial x} \approx \frac{1}{\Delta x}\left(-\frac{1}{2}, 0, \frac{1}{2}\right)\left[u_{i-1}^t, u_i^t, u_{i+1}^t\right]^T = \frac{1}{\Delta x} F_{LS}^1 * S_{u_i^t}^1 \tag{19}$$

The second-order central difference:

$$\frac{\partial^2 u_i^t}{\partial x^2} \approx \frac{1}{\Delta x^2}(1, -2, 1)\left[u_{i-1}^t, u_i^t, u_{i+1}^t\right]^T = \frac{1}{\Delta x^2} F_x^2 * S_{u_i^t}^1 \tag{20}$$

We can extend the scheme to high-order situations, the third-order difference and the fourth-order difference can be approximated as:

$$\frac{\partial^3 u_i^t}{\partial x^3} \approx \frac{1}{\Delta x^3}(-1, 3, -3, 1)\left[u_{i-1}^t, u_i^t, u_{i+1}^t, u_{i+2}^t\right]^T \tag{21}$$

$$\frac{\partial^4 u_i^t}{\partial x^4} \approx \frac{1}{\Delta x^4}(1, -4, 6, -4, 1)[\left[u_{i-2}^t, u_{i-1}^t, u_i^t, u_{i+1}^t, u_{i+2}^t\right]^T \tag{22}$$

For two-dimensional cases, we can use similar approaches to get the approximation of the derivatives:

$$\frac{\partial^2 u_{j,i}^t}{\partial x^2} \approx \frac{1}{\Delta x^2}\begin{pmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{pmatrix} * \begin{bmatrix} u_{j-1,i-1}^t, & u_{j-1,i}^t, & u_{j-1,i+1}^t \\ u_{j,i-1}^t, & u_{j,i}^t, & u_{j,i+1}^t \\ u_{j+1,i-1}^t, & u_{j+1,i}^t, & u_{j+1,i+1}^t \end{bmatrix} = \frac{1}{\Delta x^2} F_x^2 * S_{u_i^t}^2 \tag{23}$$

$$\frac{\partial^2 u_{j,i}^t}{\partial y^2} \approx \frac{1}{\Delta y^2}\begin{pmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{pmatrix} * \begin{bmatrix} u_{j-1,i-1}^t, & u_{j-1,i}^t, & u_{j-1,i+1}^t \\ u_{j,i-1}^t, & u_{j,i}^t, & u_{j,i+1}^t \\ u_{j+1,i-1}^t, & u_{j+1,i}^t, & u_{j+1,i+1}^t \end{bmatrix} = \frac{1}{\Delta y^2} F_y^2 * S_{u_i^t}^2 \ . \tag{24}$$

Thus, the Laplace Operator can be approximated as:

$$\nabla^2 u_{j,i}^t = \frac{\partial^2 u_{j,i}^t}{\partial x^2} + \frac{\partial^2 u_{j,i}^t}{\partial y^2} \approx \frac{1}{\Delta h^2}\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} * \begin{bmatrix} u_{j-1,i-1}^t, & u_{j-1,i}^t, & u_{j-1,i+1}^t \\ u_{j,i-1}^t, & u_{j,i}^t, & u_{j,i+1}^t \\ u_{j+1,i-1}^t, & u_{j+1,i}^t, & u_{j+1,i+1}^t \end{bmatrix} = \frac{1}{h^2} F_L * S_{u_i^t}^2 \ . \tag{25}$$

Where F indicates the finite difference filter, h is the uniform step size in the two-dimensional domain, $(\cdot) * [\cdot]$ denotes the inner product between two-dimensional tensors, which is quite similar with the convolution operation in convolutional neural network (CNN), and the finite difference coefficient tensor is highly consistent with the convolutional kernel (filter). The Euler method is the simplest scheme, and in order to better approximate the PDEs, we also consider the Crank-Nicolson method in this study. The second-order derivative of spatial direction is considered, and the Crank-Nicolson Stencil is:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{1}{\Delta x^2}\left\{\left(\frac{1}{2}, -1, \frac{1}{2}\right)\left[u_{i-1}^t, u_i^t, u_{i+1}^t\right]^T + \left(\frac{1}{2}, -1, \frac{1}{2}\right)\left[u_{i-1}^{t+1}, u_i^{t+1}, u_{i+1}^{t+1}\right]^T\right\} \tag{26}$$

According to Von Neumann stability analysis. The Crank-Nicolson method is a combination of explicit and implicit methods. It is unconditionally stable and has error order $O(\Delta h^2 + \Delta t^2)$, which ensures stability while improving accuracy. Finally, the 4^{th}-order Runge-Kutta scheme is also considered in the proposed network. The accuracy of Eq. (9) can be improved by enhancing the order of iteration:

$$u^{t+1} \approx u^t + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{27}$$

Where:

$$\begin{cases} k_1 = f(t, u^t) \\ k_2 = f\left(t + \frac{\Delta t}{2}, u^t + \frac{\Delta t}{2} k_1\right) \\ k_3 = f\left(t + \frac{\Delta t}{2}, u^t + \frac{\Delta t}{2} k_2\right) \\ k_4 = f(t + \Delta t, u^t + \Delta t k_3) \end{cases} \tag{28}$$

It should be noted that the reference solutions for some cases in this study are also generated using the FDM method with the 4^{th}-order Runge-Kutta scheme. With the description of the 4^{th}-order Runge-Kutta scheme, we can be ready to discuss our framework to

learn the PDEs without any labeled data by incorporating the 4th-order Runge-Kutta scheme. According to Eqs. (1), (17), (27), the PDEs can be obtained by the following approximations:

$$\frac{u^{t+1}-u^t}{\Delta t} \approx u_t = -N\left[u, \ldots, \nabla_x^{(n)}u, \gamma\right] \approx -N\left[u, \ \ldots, \sigma^{cnn}\left(\nabla_x^{(n)}u\right), \gamma\right] \approx \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{29}$$

Where $\sigma^{cnn}\left(\nabla_x^{(n)}u\right)$ denotes the approximation of the n^{th}-order derivative to x output by the convolutional neural networks, thus connections between the network and the 4th-order Runge-Kutta scheme can be built. In summary in this study, different discrete schemes are adopted into the proposed novel network and the convolution units are used to approximate the difference operators or iterative operators. Obviously, different schemes have different finite difference filters, and the convolution kernels are closely related to them.

3.3. Boundary conditions encoding

Generally, to solve a PDE that describes a specific physical phenomenon, not only the initial conditions (ICs) need to be given, but also the boundary conditions (BCs) are usually provided. Moreover, it should also be noted that the boundary should be added manually when using the finite difference method to accurately handle points on the boundary as the boundary conditions are often specified [31,51,61,65]. Therefore, we also need to rigorously consider the handling of boundary conditions in the proposed network. The boundary conditions of PDEs can be softly imposed as a penalty term in the loss function, or they can be forced to be added on the domain boundary using the padding operation in convolutional neural networks, respectively, which are known as soft BC enforcement and hard BC enforcement [15,60]. Both of these two BC treatment methods are considered and used in the proposed framework to elucidate their applicability. For demonstration purposes, the Dirichlet BCs are constrained by soft BC enforcement although they can also be hard enforced, while the Neumann BCs are added by hard BC enforcement in this study. For Neumann BCs, the enforced values on the boundary can be inferred from the values of internal points by finite difference approximation. When implementing hard BC enforcement in convolutional neural networks, it is natural to consider using the modified padding operation. Fig. 4 shows the hard boundary condition processing in the one-dimensional cases.

Where the x coordinate domain is divided into $(N_x + 1)$ points, and $x = [x_0, x_1, \ldots, x_{N_x}]$. Each element in the feature map can be regarded as a pixel in 1D natural language processing problems, the values in the input feature map consist of the tensor of the solution u^t at time t. Fig. 4 profoundly presents boundary condition processing and the relationship between input u^t and output u^{t+1} at two time points t and $(t+1)$ under network processing in one-dimensional cases. In addition, for two-dimensional cases, the hard BC enforcement and convolution operation are depicted in Fig. 5.

Where the x and y coordinate domains are divided into $(N_x + 1)$ and $(N_y + 1)$ points, respectively, and $x = [x_0, x_1, \ldots, x_{N_x}]$, $y = \left[y_0, y_1, \ldots, y_{N_y}\right]$. Each element within the feature map may be analogous to a pixel in 2D image identification scenarios. The values contained in the input feature map comprise the tensor of the solution u^t at the given time t. Fig. 5 illustrates in depth the processing of boundary conditions and the correlation between the input u^t and the output u^{t+1} at two time points t and $(t+1)$ during network processing in a two-dimensional context.

Moreover, the size of the spatial domain at each time step is consistent, which can be achieved through elaborate design based on the feature map and receptive field theory of the convolutional neural network. After the input data undergoes the convolution operation of NCU, the feature map size of the output result is:

$$H_2 = \frac{H_1 - F_H + 2P}{S} + 1 \tag{30}$$

$$W_2 = \frac{W_1 - F_W + 2P}{S} + 1 \tag{31}$$

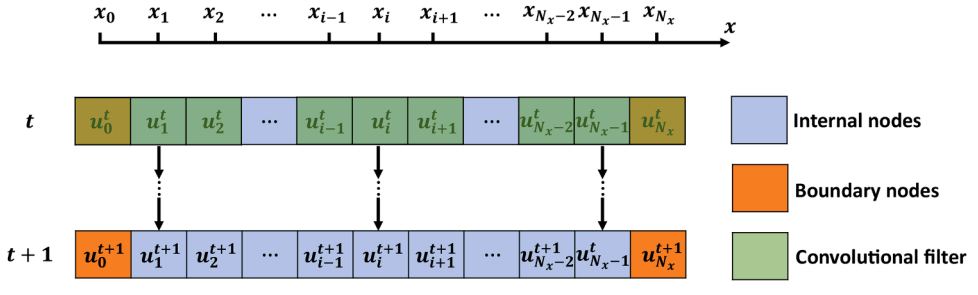


Fig. 4. Boundary condition processing in one-dimensional case. One-dimensional input data can be regarded as 1D word vectors in natural language processing. The blue part is the internal field, the orange part represents the boundary area, and the green square represents the 1D convolution filter.

Where W_1 and H_1 represent the width and length of the input; W_2 and H_2 represent the width and length of the output feature map; F_H and F_W are the length and width of the convolution kernel, respectively; S indicates the step size of the sliding window; and P denotes the number of boundary padding. Specifically, in this study, the convolution kernel size of the network is set to 3, the sliding step size is 1, and the padding number is 1. Therefore, the output size is $W_2 = W_1$ and $H_2 = H_1$ according to the feature map size calculation formula, which means the size of the output feature map will remain consistent with the size of the input feature map.

3.4. Loss function

The construction of the loss function is a significant step in building a physics-informed neural network. Appropriate constraints for the neural network are the key to ensuring that it can accurately approximate the PDEs. The innovation of this work is that the network is unsupervised and does not contain any label data, so the key point is to use the governing equation to construct the loss function of the PDE residuals under certain conditions and parameters. The use of the loss function provides an intuitive and clear measure of how well the model performs. Among the physics-informed neural network models, the Mean Squared Error (MSE) loss function is a commonly used metric to evaluate the difference and error between the predictions and the ground truth. Our goal in establishing such a loss function is to ensure the MSE loss function converges towards zero, which is achieved by adjusting the weights and biases of the model using an optimization algorithm. According to the physical equation, the PDE residuals can be defined as:

$$R(x,t,\theta,\gamma) := \Delta u_t^\theta + N\left[u^\theta, \nabla_x u^\theta,\ \nabla_x^2 u^\theta,\ \cdots, \gamma\right] \tag{32}$$

Where R represents the PDE residuals; γ is the coefficient of the PDE; $(\cdot)^\theta$ means the output of the neural network and the θ is the parameter of the neural network. The right side of the equation is the deformation calculated using the PDE representing the physical process, which is expected to have a value of zero. Therefore, taking the two-dimensional PDE as an example, the loss function of PDE residuals MSE_f based on the governing equation is defined as follows:

$$MSE_f = \frac{1}{N_f}\sum_{m=1}^{N_f} \|R_m\left(x_i, y_j, t_k, \theta, \gamma\right)\|^2 \tag{33}$$

Where N_f is the number of training points used to calculate the PDE residuals in the defined domain; $\|\cdot\|^2$ denotes the squared error operator. In this study, the initial condition of PDE is taken as the training data in the first time step, and the Neumann BC is hard encoded into the network which we do not need to consider in the loss function. Dirichlet BC using soft BC enforcement can also achieve satisfactory results. MSE_u represents the loss from the Dirichlet BC and can be defined as follows:

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} \|\widehat{u}_i - u_i\|^2 \tag{34}$$

Where N_u refers to the number of training points in boundary conditions, $\widehat{u}_i$ is the approximated solutions of the PDEs predicted by

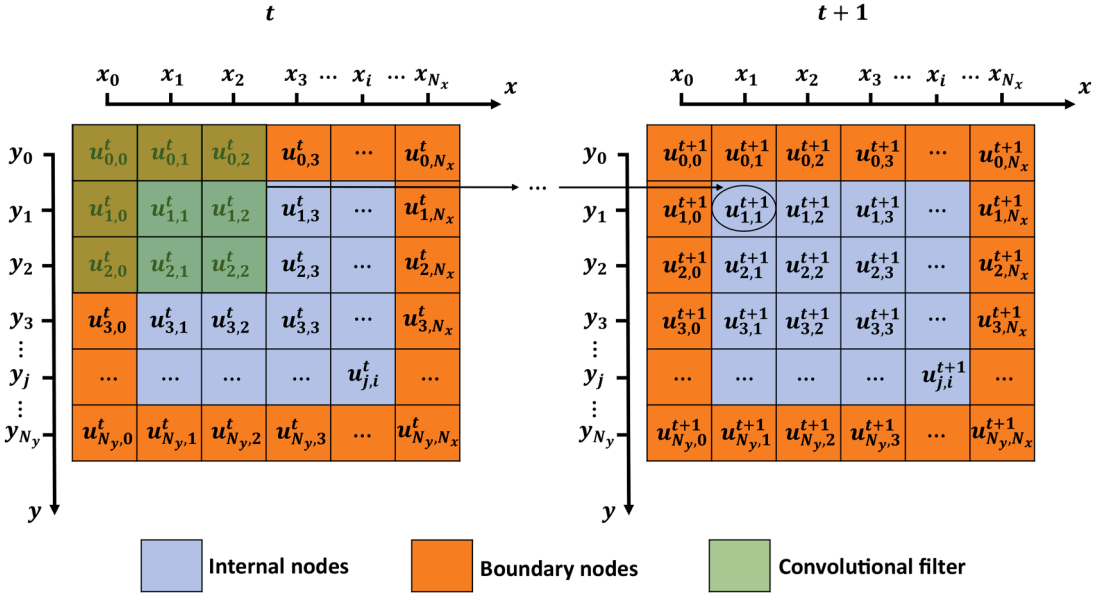


Fig. 5. Boundary condition processing in two-dimensional case. Two-dimensional input data can be regarded as 2D image-like data in computer vision. The blue part is the internal field, the orange part represents the boundary area, and the green square represents the 2D convolution filter.

neural networks, and u_i is the ground truth values of the PDEs. Thus, the total loss function in the neural network can be written as:

$$MSE = \omega_u MSE_u + \omega_f MSE_f \tag{35}$$

Where ω_u and ω_f are the loss weights parameters typically set to $\omega_u = \omega_f = 1$ in this study. Specifically, if the BC is hard encoded into the neural network, the total loss function only contains the loss term of PDE residuals, it is $MSE = MSE_f$ in this case. And if the loss function does not contain physical equation constraints and is only optimized by using the exact solution, then the network in this case is a purely data-driven CNN.

4. Reference solution and baseline method for comparisons

4.1. Ground truth

To evaluate the accuracy and effectiveness of the proposed f-PICNN methods, an error evaluation is required. Here, the mathematical analytical or numerical solutions are used as a reference for PDE case experiments. The following sections provide comprehensive comparisons to quantify the difference between the neural network solution $\widehat{u}_i$ and the reference solution u_i. In this study, some of the PDE cases provided exact analytical solutions as ground truth references. While several numerical methods can be employed when analytical solutions are not available, and the 4^{th}-order Runge-Kutta FDM is selected to generate numerical solutions as references in these cases.

4.2. Vanilla PINN methods

The original baseline of PINN with fully connected neural network is used to make comparisons with the proposed f-PICNN method. Firstly, the loss function in PINN is still constructed from the soft constraints of the initial and boundary conditions, as well as the PDE residuals. The derivatives of the PDEs are computed by automatic differentiation. Therefore, the loss function in PINN is $L = \omega_u L_u + \omega_f L_f$ [57], where L_u is the loss term based on the given initial and boundary conditions and L_f is the loss term based on the physical laws. All the weights of the loss terms are set to $\omega_u = \omega_f = 1$. The collocation points used for PINN training are generated from the whole spatiotemporal domain using Latin Hypercube Sampling (LHS), which is a typically uniform sampling method. The activation function for network training is $Tanh(x)$, and the optimizer uses L-BFGS with its learning rate set to $LR = 1.0$, the maximum number of iterations of the L-BFGS optimizer is set to 5×10^4 until convergence.

In the one-dimensional numerical case experiments, the network structure of PINN contains one input layer, eight hidden layers with 40 neurons each, and an output layer. We select a sufficiently large number of collocation points 5×10^4 and 400 random boundary points as input datasets for the PINN training. In addition, in the 2D numerical case experiments, we set up one input layer, four hidden layers with 40 neurons each, and one output layer in the PINNs. The input training dataset contains 3×10^5 randomly selected collocation points and 2×10^4 boundary points. All experimental results of PINN will also be compared with the ground truth, and PINN serves as a reference model for comparison with f-PICNN in this study.

5. Numerical case experiments

In this section, the performance of the proposed method on several PDEs to demonstrate its advantages is evaluated and compared with the baseline algorithms: the vanilla PINN methods [57] and traditional methods. Generally, the traditional methods considered as reference solutions in this study include analytical solutions and numerical solutions. To quantify the difference between the predicted solution and reference solution, the L_2 norm of the relative error e_1, root mean square error (RMSE) e_2, and mean absolute error (MAE) e_3 are introduced and expressed as follows:

$$\begin{aligned} e_1 &= \frac{\| \widehat{u} - u \|_{L_2}}{\| u \|_{L_2}} \\ e_2 &= \sqrt{\frac{\sum |\widehat{u} - u|^2}{N_f}} \\ e_3 &= \frac{\sum |\widehat{u} - u|}{N_f} \end{aligned} \tag{36}$$

where $\widehat{u}$ represents the predictions and u represents the reference solutions. $\| \cdot \|_{L_2}$ denotes the calculation of the L_2 norm, and N_f is the number of collocation points in the whole domain. In order to control the variables in comparisons and simplify the parameter search of the neural network models, the number of convolutional neurons in each convolutional unit and the depth of the convolutional units are assumed and fixed as $[1, 32, 64, 128, 256, 64, 1]$ in all the numerical experiments in this paper. The Adam algorithm [28] is used as the optimizer to train the neural network with a learning rate (LR) of 1×10^{-4}. All cases are trained with a sufficient number of epochs and iterations, we set 500 epochs for the first time-step training and 100 to 200 epochs for the following training according to the loss function decline. A careful design of hyperparameters can mitigate the overfitting problems.

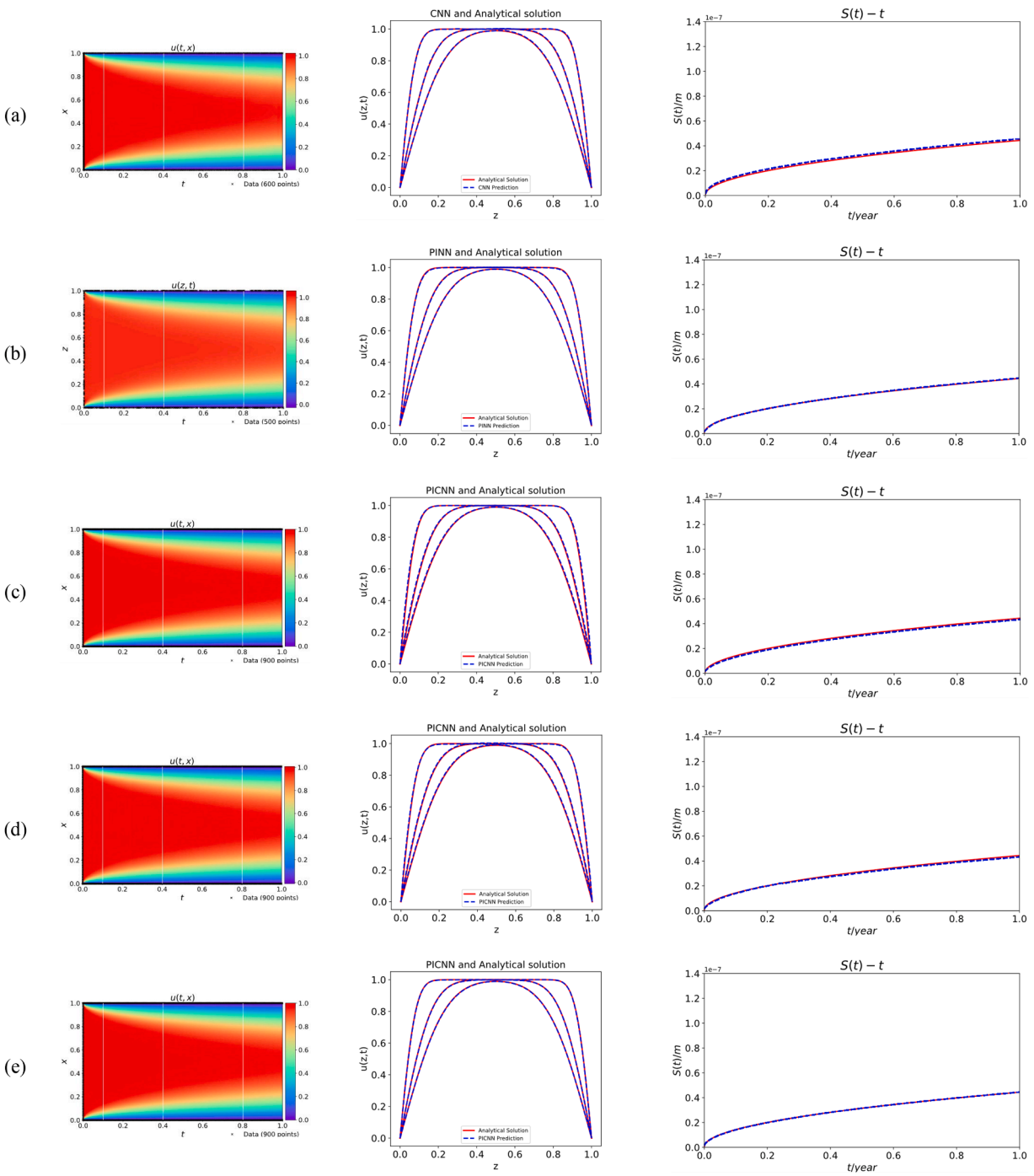


Fig. 6. The heat maps of the results of consolidation equation with Dirichlet BC predicted by different schemes and the comparisons between different scheme predictions and reference solution at three selected time points $\boldsymbol{t}_0 = 0.10$, $\boldsymbol{t}_1 = 0.40$, $\boldsymbol{t}_2 = 0.80$. The excess pore water pressure decreases over time in the whole domain. The first column shows the heat maps, the second column shows the comparison charts and the last column shows the predictions of soil settlement. (a) purely data-driven CNN; (b) PINN; (c) f-PICNN (Euler); (d) f-PICNN (Crank-Nicolson); (e) f-PICNN (4^{th}-order Runge-Kutta).

The numerical experiments cover several 1D and 2D nonlinear PDEs: Consolidation equation; Allen-Cahn equation; Burgers equation; Advection equation; Diffusion equation; 2D Consolidation equation and 2D Burgers equation. Among these PDE systems, Dirichlet BC, Neumann BC and Non-zero BC are considered in specific scenarios. All the numerical implementations in this study are coded in Pytorch and performed on a NVIDIA Tesla T4 GPU card.

5.1. Consolidation equation

The 1D Terzaghi's consolidation PDE with Dirichlet BC and Neumann BC is used as an example to verify that f-PICNN can predict the soil settlement accurately, the equation is given below:

$$\begin{cases} \dfrac{\partial u}{\partial t} = C_v \dfrac{\partial^2 u}{\partial z^2} \\ u(z,0) = u_0(z) \end{cases} \tag{37}$$

Where $\Omega = (z,t) \in [0,1] \times \ (0,1]$, the initial condition is set as $u_0(z) = 1$, and the coefficient of consolidation is set as $C_v = 0.02$. u denotes the excess pore water pressure in the soil layer. The consolidation equation is usually used to figure out the soil consolidation and settlement, as well as a benchmark model to verify the performance of different computational algorithms. The zero Dirichlet BC refers to the drained boundary while the Neumann BC indicates the undrained drained boundary. Finally, two different BC cases are considered in this section:

$$\begin{cases} u(0,t) = u(1,t) = 0; \ \textit{drained top and bottom boundaries} \\ u(0,t) = 0; \ \dfrac{\partial u(1,\ t)}{\partial z} = 0; \ \textit{drained top and undrained bottom boundaries} \end{cases} \tag{38}$$

f-PICNN can capture the PDEs based on finite difference discretization. Several schemes, e.g., Euler, Crank Nicolson and 4$^{\text{th}}$ Runge-Kutta, can be used to discrete the consolidation equation. Therefore, the learnable convolutional kernels are trained to approximate the filters for finite difference methods (FDM). The exact reference solution is calculated using the analytical solution, and the heat map results for drained top and bottom boundaries are shown in Fig. 5. In this study, an interpolation method based on the excess pore water pressure values of the nearest points is adopted and it demonstrates that the neural network models can learn the dynamics with 100 spatial coordinate points and a time-step length $\delta t = 0.0025$. Specifically, the purely data-driven CNN adopts a no-physics loss function while the PINN uses a physics-informed loss function with automatic differentiation and f-PICNN employs a physics-informed loss function with finite differentiation. The total inference time for purely data-driven CNN and f-PICNN, and the total training time for PINN are 75s, 345s and 160s, respectively. The mean square error (MSE) loss value is in the order of 10^{-6} at the final epoch of the model training, which means the difference between the predicted and true values is at a desirable small level. It indicates that f-PICNN has a good performance for solving PDEs.

For better comparisons, three time points ($t_0 = 0.10,\ t_1 = 0.40,\ t_2 = 0.80$) are randomly selected to make intuitive comparisons between different schemes and ground truth reference solution at these time points, the results are also shown in Fig. 6. The soil settlement can be represented by the mathematical theory and the predictions obtained from the neural networks, which can be calculated according to the following equation at time t:

$$S_t = \frac{a_v}{1+e_0}\left(u_0(z) - \frac{1}{h}\int_0^h u dz\right)h \tag{39}$$

Where the coefficient of volume and initial porosity are set as $a_v = 0.00025 kpa^{-1}$ and $e_0 = 0.8$, respectively. u is the excess pore water pressure through the whole soil thickness h, namely the solution of consolidation equation at time t. The results are depicted as $S(t) - t$ relationship in Fig. 6, where the red solid line represents the ground truth reference solution, and the blue dashed line corresponds to the neural network solution. The findings substantiate a strong agreement between the soil settlement predicted by the neural networks and the theoretical analysis and underscore the efficacy of the f-PICNN in accurately capturing the complex process of soil consolidation.

The relative L_2 errors between the neural network solution and the analytical solution are summarized in Table 1. In this section, f-PICNN with 4$^{\text{th}}$ Runge-Kutta scheme gives the most accurate results among all the schemes with a relative L_2 error of $5.856 \ \times 10^{-4}$. The f-PICNN with Euler scheme provides acceptable results, PINN and f-PICNN with Crank Nicolson scheme achieve desirable outputs, with relative L_2 errors of $1.214 \ \times 10^{-2}$, $8.861 \ \times 10^{-3}$ and $7.996 \ \times 10^{-3}$, respectively. Although the purely data-driven method can yield a good result, it only uses a neural network to approximate the reference solution, there are no physical constraints in the network, and the results are difficult to be convinced of. In f-PICNN, the Dirichlet BC is imposed using soft BC enforcement, and it's

Table 1
Relative L_2 Errors between different scheme predictions and reference solution for consolidation equation with Dirichlet BC.

Scheme	Purely data-driven CNN	PINN	f-PICNN (Euler)	f-PICNN (Crank-Nicolson)	f-PICNN (4$^{\text{th}}$ Runge-Kutta)
Error	$4.105 \ \times 10^{-3}$	$8.861 \ \times 10^{-3}$	$1.214 \ \times 10^{-2}$	$7.996 \ \times 10^{-3}$	$5.856 \ \times 10^{-4}$

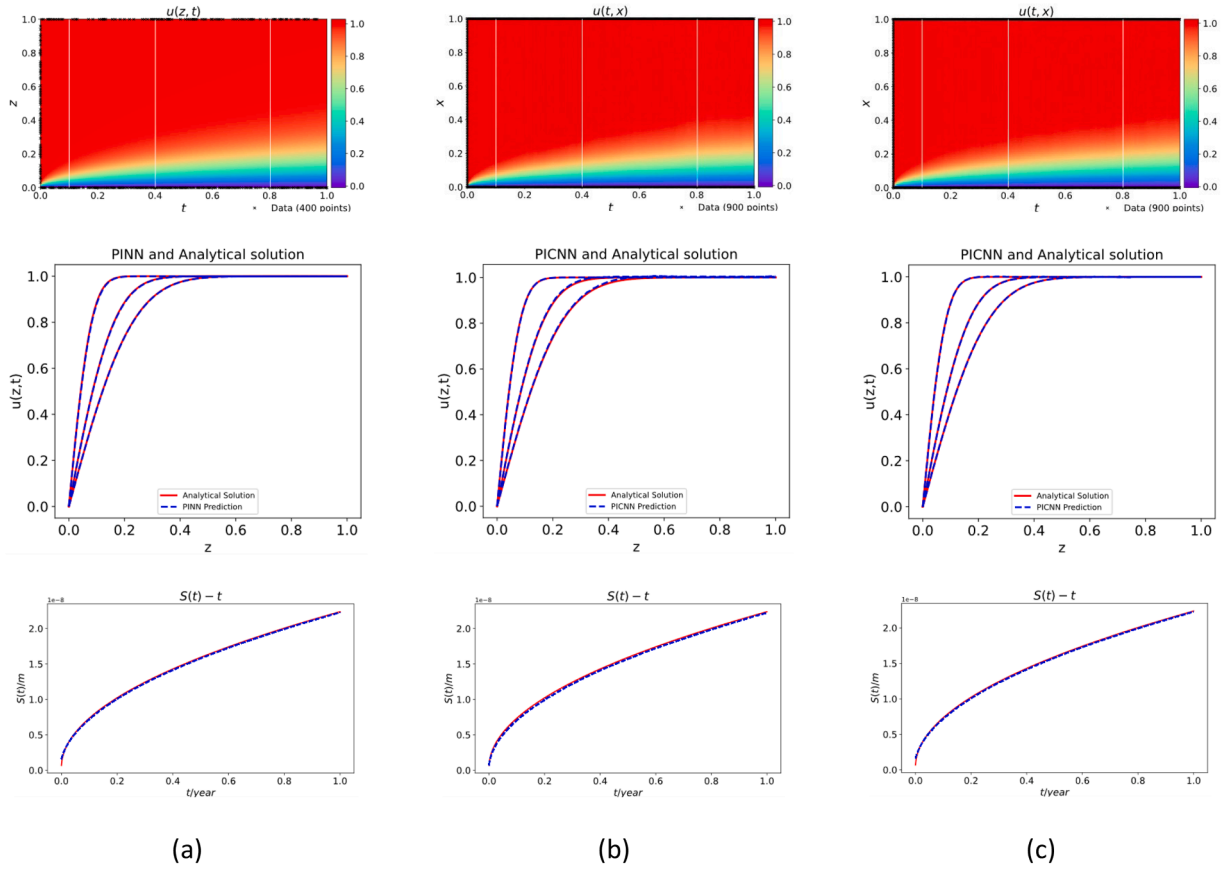


Fig. 7. The comparisons of different scheme predictions and reference solution on consolidation equation with Neumann BC at three selected time points $t_0 = 0.10,\ t_1 = 0.40,\ t_2 = 0.80$. The excess pore water pressure decreases over time in the whole domain. The first row shows the heat maps, the second row shows the comparison charts, and the last row shows the predictions for soil settlement. (a) PINN; (b) f-PICNN (Euler); (c) f-PICNN (4th Runge-Kutta).

Table 2
Relative L_2 Errors between different scheme predictions and reference solution for consolidation equation with Neumann BC.

Scheme	PINN	f-PICNN (Euler)	f-PICNN (Crank Nicolson)	f-PICNN (4^{th} Runge-Kutta)
Error	4.945×10^{-3}	9.443×10^{-3}	6.943×10^{-3}	3.788×10^{-3}

easy to realize successfully. The findings indicate that f-PICNN can solve consolidation equation with Dirichlet BC well and the working concepts of f-PICNN are simply reasonable. Theoretically and empirically, more refined mesh discretization and more careful hyperparameters design can bring better precision.

Similarly, the results for drained top and undrained bottom boundaries are shown in Fig. 7. The Neumann BC is hard encoded into the f-PICNN as it is more efficient for training convergence in this situation, and the hard-coded values can be derived by FDM.

The relative L_2 errors are shown in Table 2. Although the best results are obtained by the 4^{th} Runge-Kutta scheme, it cannot be concluded that this scheme is most effective in improving the accuracy because neural network training is also affected by other factors, and the specific mechanism needs to be further explored in the future. Moreover, PINN can also provide a good solution in this case.

In conclusion, the results indicate that f-PICNN can capture the consolidation process very well and accurately predicts soil settlement and thus implies that it can be used as a surrogate model for solving similar problems.

5.2. Allen-Cahn equation

The Allen-Cahn equation is a reaction-diffusion equation in mathematical physics that elucidates the dynamics of phase separation in multi-component alloy systems, particularly order-disorder transitions. The equation describes the temporal progression of a scalar-valued state variable u within a domain Ω over a specified time interval. The Allen-Cahn equation and conditions are given below:

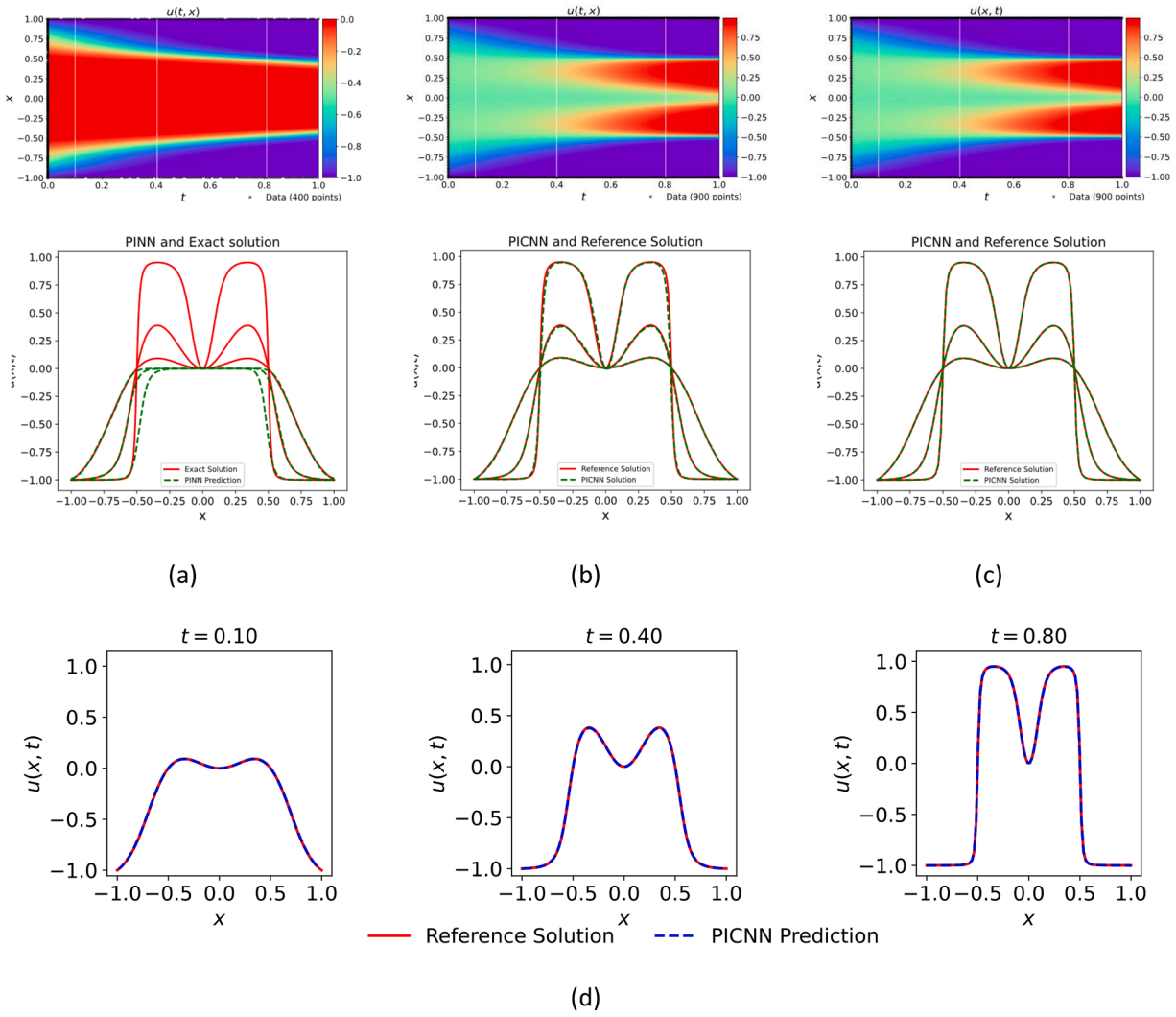


Fig. 8. The comparisons of different scheme predictions and reference solution on Allen-Cahn equation at three selected time points $t_0 = 0.10,\ t_1 = 0.40,\ t_2 = 0.80$. (a) PINN; (b) f-PICNN (Euler); (c) f-PICNN (4th Runge-Kutta); (d) details of (c).

Table 3
Relative L_2 Errors between different scheme predictions and reference solution for Allen-Cahn equation.

Scheme	PINN	f-PICNN (Euler)	f-PICNN (4^{th} Runge-Kutta)
Error	5.533×10^{-1}	5.097×10^{-2}	5.830×10^{-3}

$$\begin{cases} \dfrac{\partial u}{\partial t} = \gamma \dfrac{\partial^2 u}{\partial x^2} + 5u - 5u^3 \\ u(x,0) = u_0(x) = x^2 \cos(\pi x) \\ u(-1,t) = u(1,t) = -1 \end{cases} \tag{40}$$

Where $\Omega = (x,t) \in [-1,1] \times (0,1]$, the coefficient is set as $\gamma = 0.0001$, $u_0(x)$ is the initial condition. The discrete time step size for the neural network training is set to $\delta t = 0.0025$ and there are 100 points on the spatial coordinate axis. To facilitate quantitative comparisons between the neural network solution and the ground truth reference solution generated by 4^{th} Runge-Kutta FDM, three specific time points have been carefully selected ($t_0 = 0.10,\ t_1 = 0.40,\ t_2 = 0.80$). The testing results obtained for each solution are displayed in Fig. 8, allowing for a comprehensive evaluation and analysis of their respective performances.

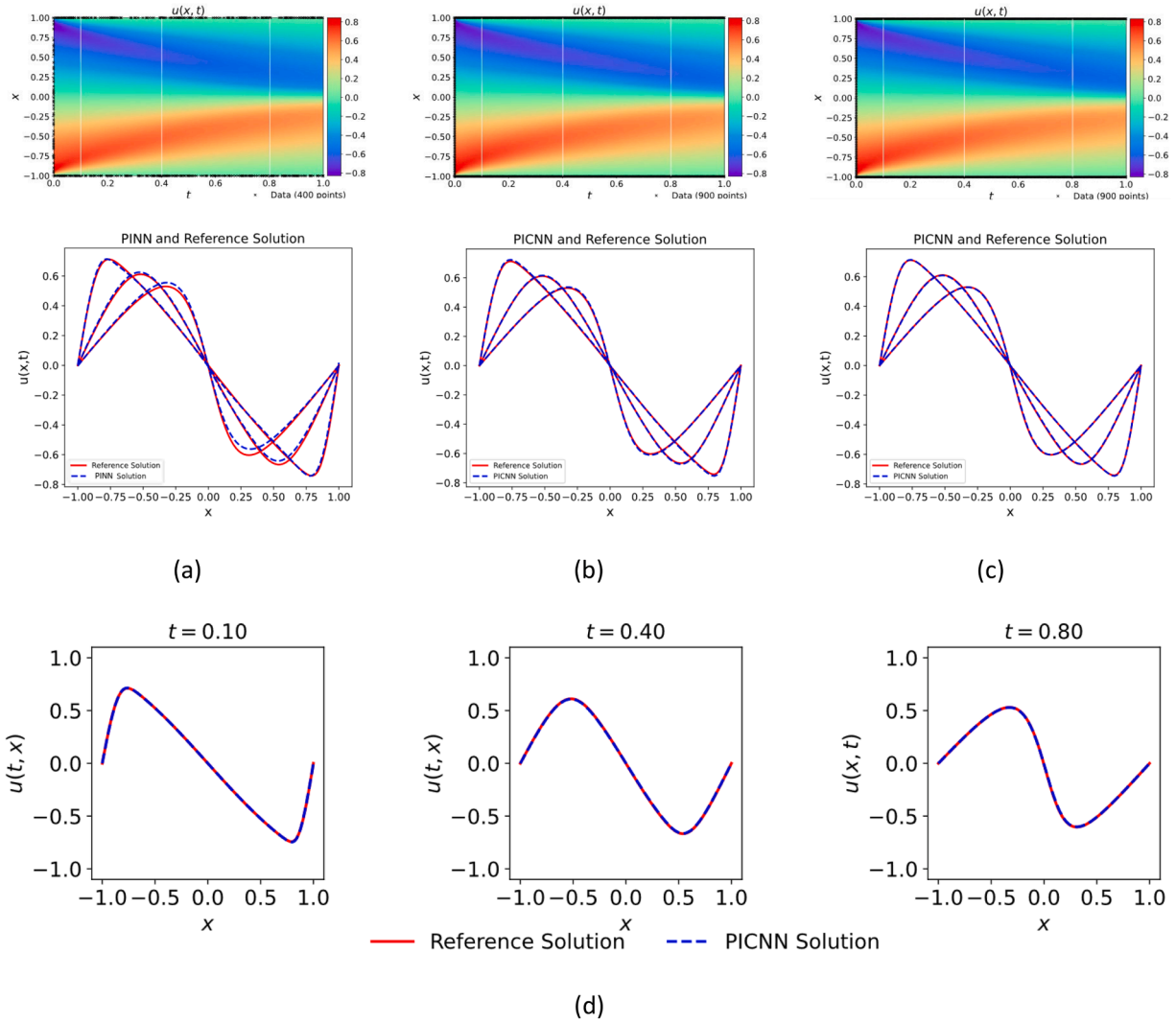


Fig. 9. The comparisons of different scheme predictions and reference solution on Burgers equation at three selected time points $\boldsymbol{t}_0 = 0.10$, $\boldsymbol{t}_1 = 0.40$, $\boldsymbol{t}_2 = 0.80$. The third row is the of f-PICNN with 4^{th} Runge-Kutta scheme. (a) PINN; (b) f-PICNN (Euler); (c) f-PICNN (4th Runge-Kutta); (d) details of (c).

Table 4
Relative $\boldsymbol{L}_2$ Errors between different scheme predictions and reference solution for Burgers equation.

Scheme	PINN	f-PICNN (Euler)	f-PICNN (4th Runge-Kutta)
Error	1.285×10^{-2}	3.143×10^{-3}	1.819×10^{-3}

Fig. 8 indicates that the f-PICNN solution and reference solution match well, and the comparisons are quantified by relative L_2 error, which is shown in Table 3. The relative L_2 error for f-PICNN with 4^{th}-order Runge-Kutta scheme is 5.830×10^{-3}, which indicates that the f-PICNN can capture the Allen-Cahn equation well based on the auto-regressive framework with a time-marching mechanism. Moreover, the vanilla PINN prediction fails to match the ground truth with the worst error.

5.3. Burgers equation

The Burgers equation is a crucial convection-diffusion PDE that appears in several fields of applied mathematics, including gas dynamics, nonlinear acoustics, fluid mechanics, and traffic flow. The equation describes the behavior of a given field $u(x,t)$ and diffusion coefficient (kinematic viscosity) ϑ in one space dimension as a dissipative system. We consider the Burgers equation and conditions:

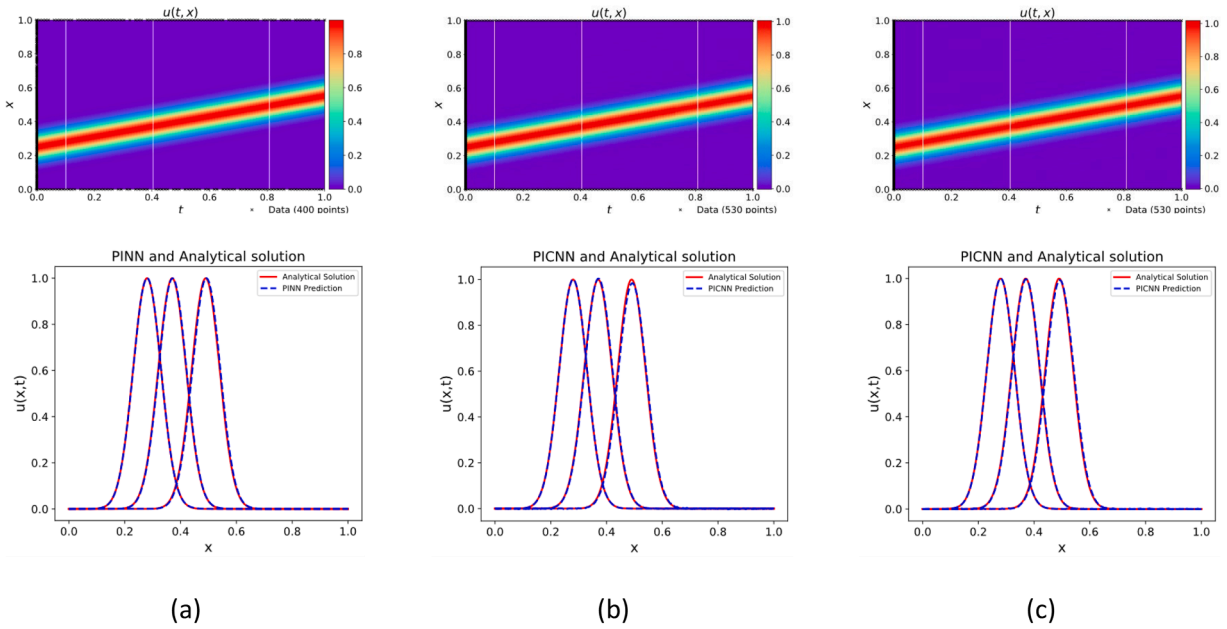


Fig. 10. The comparisons of different scheme predictions and reference solution on Advection equation at three selected time points $t_0 = 0.10$, $t_1 = 0.40$, $t_2 = 0.80$. (a) PINN; (b) f-PICNN (Euler); (c) f-PICNN (4th Runge-Kutta).

Table 5
Relative L_2 Errors between different scheme predictions and reference solution for Advection equation.

Scheme	PINN	f-PICNN (Euler)	f-PICNN (4^{th} Runge-Kutta)
Error	2.451×10^{-3}	6.392×10^{-3}	3.046×10^{-3}

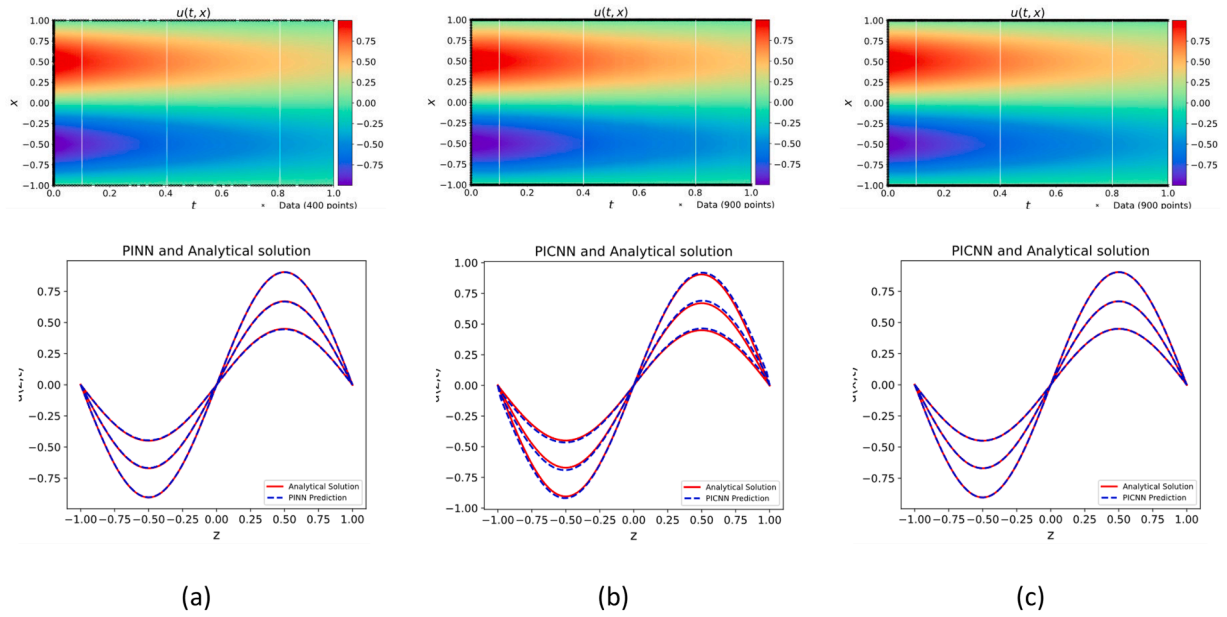


Fig. 11. The comparisons of different scheme predictions and reference solution on Diffusion equation at three selected time points $t_0 = 0.10$, $t_1 = 0.40$, $t_2 = 0.80$. (a) PINN; (b) f-PICNN (Euler); (c) f-PICNN (4th Runge-Kutta).

$$\begin{cases} \dfrac{\partial u}{\partial t} + u \cdot \dfrac{\partial u}{\partial x} = \vartheta \dfrac{\partial^2 u}{\partial x^2} \\ u(x,0) = u_0(x) = -\sin x \\ u(-1,t) = u(1,t) = 0 \end{cases} \tag{41}$$

Where $\Omega = (x,t) \in [-1,1] \times (0,1]$, the coefficient is set as $\vartheta = 0.025$, $u_0(x)$ is the initial condition, which is used as the solution of first time step to extrapolate the solutions at following time points, the reference solution is generated by 4^{th}-order Runge-Kutta FDM. In the numerical experiments, the Dirichlet BCs are considered and soft-enforced into the f-PICNN, and there are 100 spatial coordinate points with a time interval $\delta t = 0.0025$ in the whole domain.

Fig. 9 depicts a comprehensive evaluation and analysis of the respective performances of different schemes, and it indicates that the f-PICNN solution and reference solution are in good agreement. The comparisons are quantified by relative L_2 error, which is shown in Table 4. The experimental results demonstrate that f-PICNN has good extrapolation in solving the Burgers equation, while PINN has poor convergence.

5.4. Advection equation

The advection equation is a fundamental PDE in fluid dynamics and other fields that describes the behavior of a conserved scalar field as it is advected by the velocity field over time. The Advection equation and its conditions considered in this study are given as follows:

$$\begin{cases} \dfrac{\partial u}{\partial t} = -c \dfrac{\partial u}{\partial x} \\ u(x,0) = u_0(x) = e^{-200(x-0.25)^2} \\ u(0,t) = u(1,t) = 0 \end{cases} \tag{42}$$

Where $\Omega = (x,t) \in [0,1] \times (0,1]$, the coefficient is set as $c = 0.3$, $u_0(x)$ is the initial condition, which is used as the solution of first time step to extrapolate the solutions at following time points. The whole domain is discretized with a time step size $\delta t = 0.01$ and 330 points on the spatial coordinate axis for neural network training. In addition, the ground truth of Eq. (42) is from the analytical solution, which is $u(x,t) = e^{-200(x-0.25-ct)^2}$.

Fig. 10 shows the results predicted by the PINN and the proposed model, and the relative L_2 errors are shown in Table 5. Specifically, the loss function at the final iteration step of PINN and f-PICNN decreases to the order of 10^{-6} and 10^{-9}, respectively. However, the inference time of f-PICNN is 716s while the training time is only 16s for PINN. The relative L_2 errors of all schemes are in the order of 10^{-3}, which indicates that both the PINN and f-PICNN can capture the Advection equation very well. The proposed f-PICNN method can still be considered a good surrogate model for this problem.

5.5. Diffusion equation

The diffusion equation is a parabolic PDE that describes the macroscopic behavior of the micro-particles in Brownian motion. This behavior arises from random movements and collisions of the particles. The equation has applications in many fields, including information theory, materials science, and biophysics. The diffusion equation and its conditions in this study are considered as follows:

$$\begin{cases} \dfrac{\partial u}{\partial t} = D \dfrac{\partial^2 u}{\partial x^2} + e^x\big(-\sin(\pi x) + \pi^2 \sin(\pi x)\big) \\ u(x,0) = u_0(x) = \sin(\pi x) \\ u(0,t) = u(1,t) = 0 \end{cases} \tag{43}$$

Where $\Omega = (x,t) \in [-1,1] \times (0,1]$, $u(x,t)$ is the density of the diffusing material at location x and time t and D is the diffusion coefficient which is set to $D = 1.0$. If the diffusion coefficient depends on the density, then the equation is nonlinear, otherwise it is linear. $u_0(x)$ is the initial condition, which is used as the exact solution for the first time step and fed into the neural network for inferring the solution for the following time steps. The entire domain is discretized into $\delta t = 0.0025$ and there are 100 points on the spatial coordinate axis. The ground truth reference solution is generated from the analytical solution which is $u(x,t) = \sin(\pi x) \cdot e^{-t}$. Finally, the solutions predicted by PINN and f-PICNN, compared with the ground truth reference, are shown in Fig. 11.

Table 6
Relative L_2 Errors between different scheme predictions and reference solution for Diffusion equation.

Scheme	PINN	f-PICNN (Euler)	f-PICNN (4^{th} Runge-Kutta)
Error	9.846×10^{-4}	6.392×10^{-3}	6.246×10^{-4}

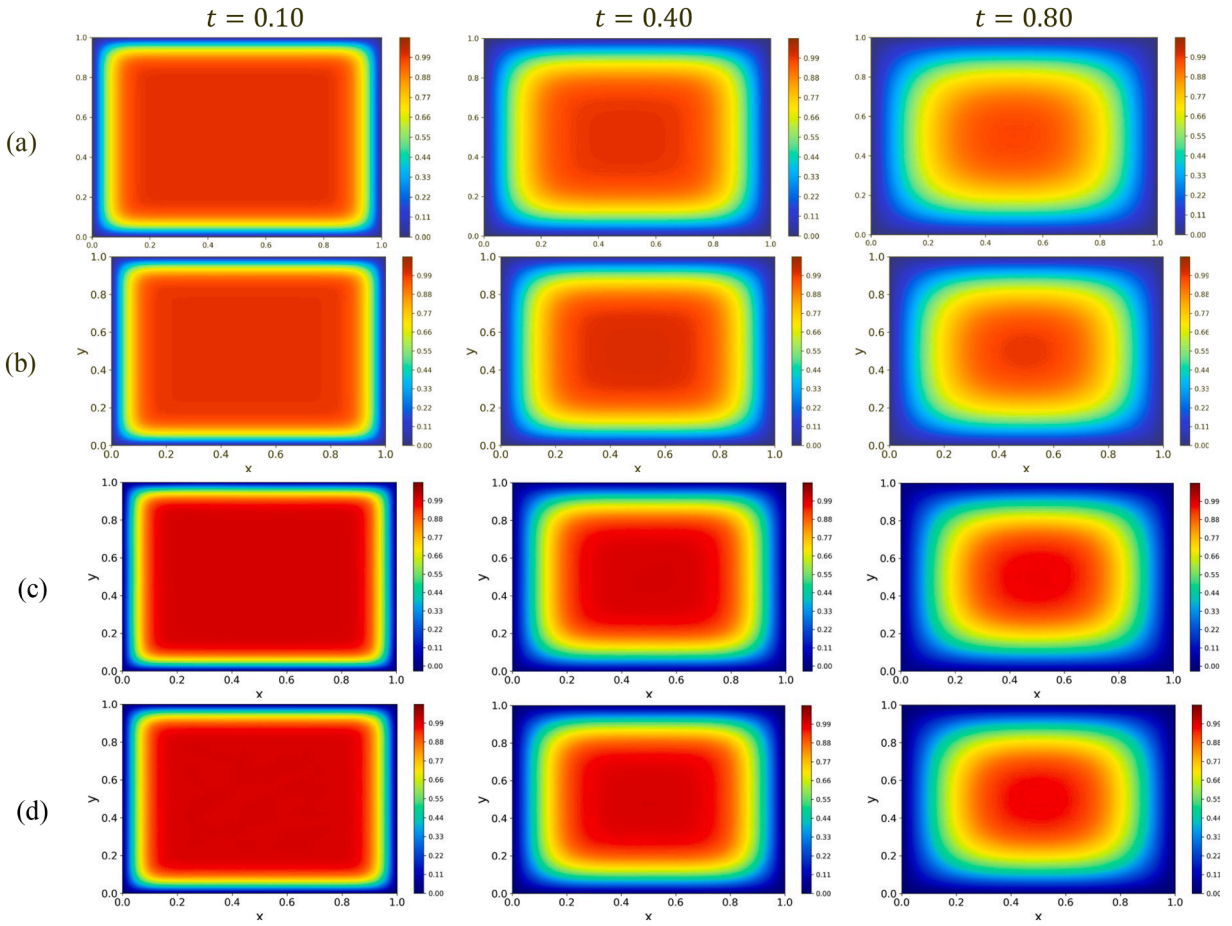


Fig. 12. The heat maps of the PDE solution solved by different schemes for two-dimensional Terzaghi's consolidation equation with Dirichlet BCs. For Dirichlet BC: (a) Reference; (b) f-PICNN; (c) PINN; (d) Purely data-driven CNN.

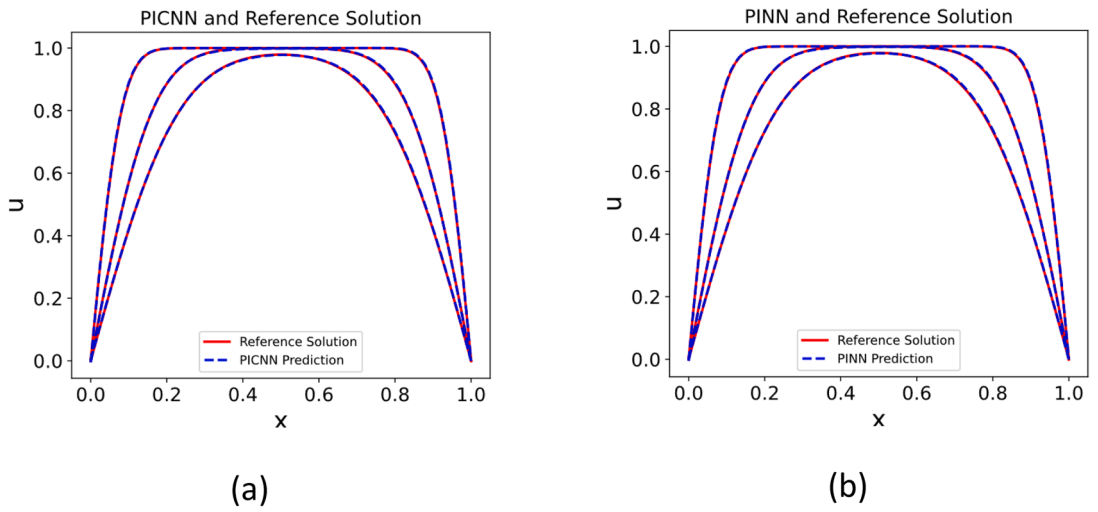


Fig. 13. The comparisons between different scheme predictions and reference solution at three selected time points $t_0 = 0.10$, $t_1 = 0.40$, $t_2 = 0.80$. The excess pore water pressure decreases over time in the whole domain. (a) f-PICNN; (b) PINN.

Table 7
Mean absolute Errors between different scheme predictions and reference solution for 2D consolidation equation.

Scheme	PINN	f-PICNN (Euler)	f-PICNN (4th Runge-Kutta)	BC
Error	2.846×10^{-2}	8.101×10^{-3}	6.290×10^{-3}	Dirichlet

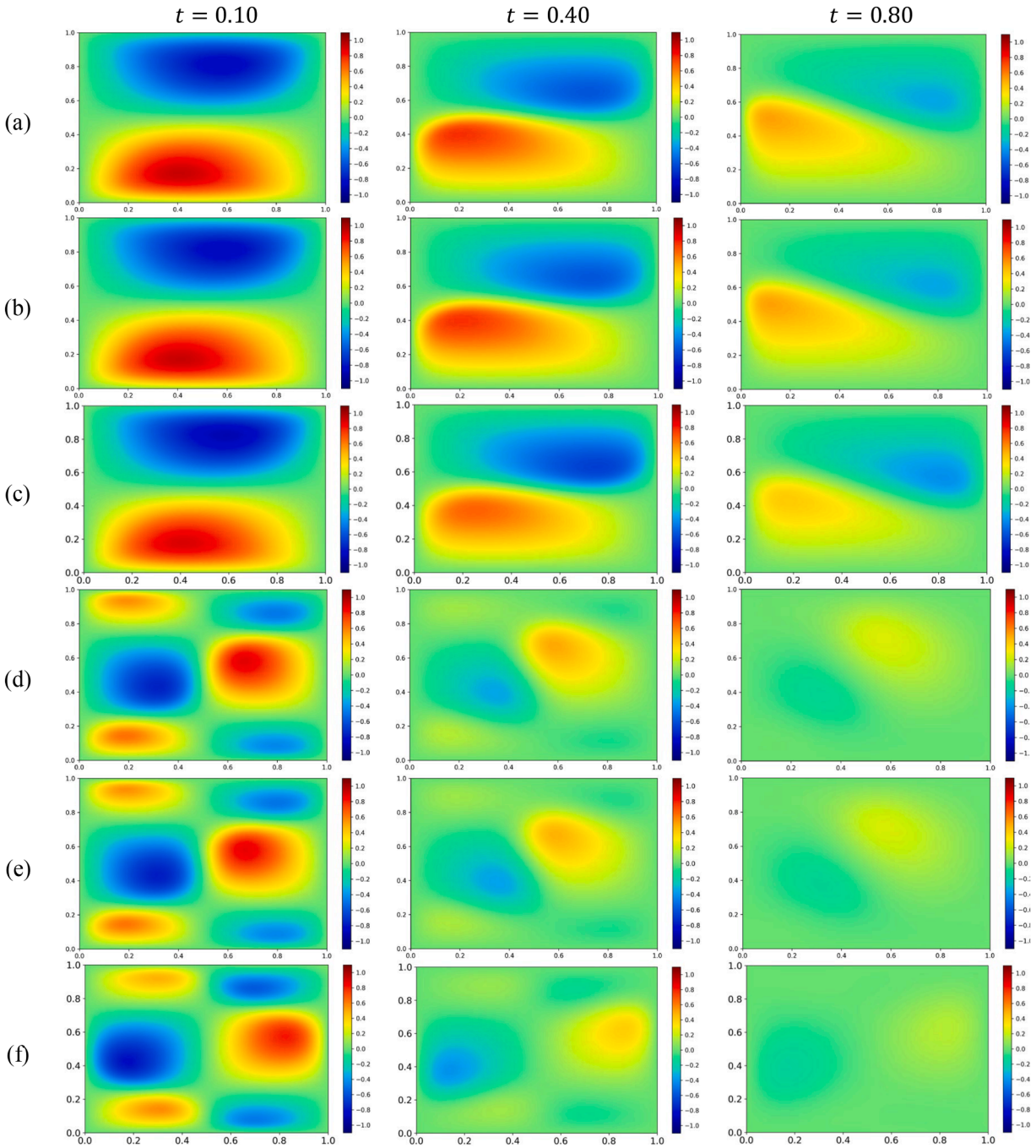


Fig. 14. The heat maps of the PDE solution solved by PINN and f-PICNN for two-dimensional Burgers equation. For Non-zero BC: (a) Reference; (b) f-PICNN; (c) PINN. For Dirichlet BC: (d) Reference; (e) f-PICNN; (f) PINN.

In this section, the PINN and f-PICNN are used to solve the Diffusion equation and Dirichlet BC is considered in the numerical experiments. From the results we can indicate that both PINN and f-PICNN can solve this equation well, we consider this phenomenon coming from the truth that the portraits of this diffusion system are simple, stable and steady. The total training time of PINN is 485s with a loss function value order of 10^{-6} in the final epochs while the total inference time of f-PICNN is 648s with a loss function value order of 10^{-6}. Therefore, the PINN training is slightly faster than that of f-PICNN, the f-PICNN can still be a good surrogate model according to the results comparisons.

The relative L_2 errors are summarized in Table 6. Both PINN and f-PICNN can easily provide accurate predictions for a moderate time duration as they can achieve a small level error with an order of 10^{-4}.

5.6. 2D Consolidation equation

The two-dimensional problems should be considered as benchmarks to further test the effectiveness and accuracy of the proposed network, which is based on a set of convolutional operations and can be extended to high-dimensional scenarios easily. Therefore, the two-dimensional consolidation equation is considered as an example to demonstrate the performance of f-PICNN on two-dimensional problems. The two-dimensional Terzaghi's consolidation equation with initial conditions is shown below:

$$\begin{cases} \dfrac{\partial u}{\partial t} = C_v\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}\right) \\ u(x,y,0) = u_0(x,y) \end{cases} \tag{44}$$

Where $\Omega = (x,y,t) \in [0,1] \times [0,1] \times (0,1]$, the coefficient of consolidation is set to $C_v = 0.02$, and the initial condition is $u_0(x,y) = 1$. The results, namely the solution of the PDE, are shown in Fig. 12. The ground truth reference solution is generated by 4^{th} order Runge-Kutta FDM. For model training, the time step size is set to $\delta t = 0.0025$ and the spatial interval is set to $\delta x = \delta y = 0.01$. The results show that excess pore water pressure in the soil layer changes with time. To compare the neural network solution and the reference solution, three time points are selected ($t_0 = 0.10,\ t_1 = 0.40,\ t_2 = 0.80$), and the results are presented in the form of heat maps (Fig. 12).

Moreover, a quantitative comparison between the neural network solution and the reference solution is shown in Fig. 13:

The mean absolute errors for two-dimensional consolidation problems are shown in Table 7. For Dirichlet BC, we carry out a numerical experiment on purely data-driven CNN, and its error is 1.465×10^{-3}. This case in two dimensions can be regarded as an image recognition task, but the model lacks physical information, which makes it unconvincing.

The training time for PINN is 2953 seconds, while f-PICNN only takes 835 seconds of inference time. This indicates a significant advantage for f-PICNN. Both PINN and f-PICNN can get acceptable solutions, but f-PICNN takes less time and has slightly better extrapolation performance on 2D consolidation equation.

5.7. 2D Burgers equation

We consider a two-dimensional Burgers equation as a testing benchmark,

$$\begin{cases} \dfrac{\partial u}{\partial t} + u \cdot \left(\dfrac{\partial u}{\partial x} + \dfrac{\partial u}{\partial y}\right) = \vartheta\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}\right) \\ u(x,y,0) = u_0(x,y) \end{cases} \tag{45}$$

Where $\Omega = (x,y,t) \in [0,1] \times [0,1] \times (0,1]$. The ground truth reference solution is generated by 4^{th} order Runge Kutta FDM, and the initial condition is set as the first time step solution and used to train the f-PICNN to extrapolate the solution at the following time steps. For Non-zero BC, we set the coefficient as $\vartheta = 0.02$, the initial condition is considered as $u_0 = \sin(\pi x)\cos(\pi x)$, and the boundary conditions are $u(0,y,t) = u(1,y,t) = 0$, $u(x,0,t) = e^{-2\pi^2 t}\sin(\pi x)$, $u(x,1,t) = -e^{-2\pi^2 t}\sin(\pi x)$. For Dirichlet BC, the coefficient is set to $\vartheta = 0.025$, the initial condition is $u_0 = \sin(2\pi x)\cos(2\pi x)$, and the boundary conditions are $u(0,y,t) = u(1,y,t) = u(x,0,t) = u(x,1,t) = 0$. To quantitatively compare the neural network solution and ground truth reference, three time points are selected ($t_0 = 0.10,\ t_1 = 0.40,\ t_2 = 0.80$), and the results are presented in the form of heat maps (Fig. 14).

A time step size of $\delta t = 0.0025$ and a spatial interval $\delta x = \delta y = 0.02$ is selected over the discretized domain considering the time marching in f-PICNN. A quantitative comparison between the neural network solution and the reference solution is shown in Fig. 15.

The mean absolute errors for two-dimensional Burgers equation are shown in Table 8.

The results show that f-PICNN can get relatively better results compared to PINN, the error order of f-PICNN and PINN is in 10^{-3} and 10^{-2}, respectively. Moreover, the total inference time of f-PICNN is 459s and the total training time of PINN is 3584s, which shows that f-PICNN is not only more efficient in time but also more precise in accuracy when solving the 2D Burgers equation.

6. Discussion and conclusions

In this study, a novel physics-informed convolutional neural network framework is proposed based on finite discretization schemes with a stack of a series of nonlinear convolutional units (NCUs) for solving PDEs without any labeled data (f-PICNN). It can make reasonable and accurate predictions of PDEs under given initial conditions without any assumptions and prior knowledge. The soft BC

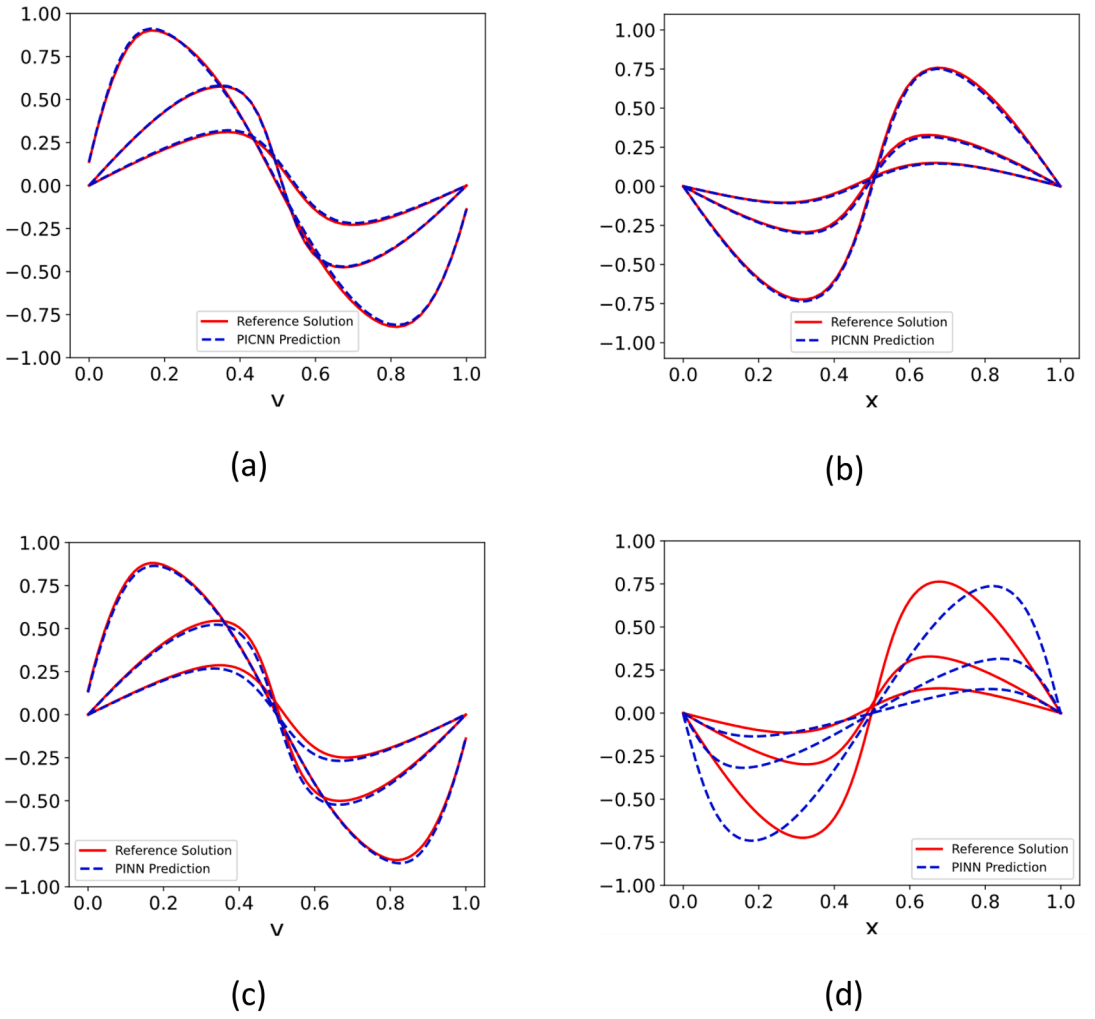


Fig. 15. The comparisons between different scheme predictions and reference solution at three selected time points $t_0 = 0.10$, $t_1 = 0.40$, $t_2 = 0.80$. The excess pore water pressure decreases over time in the whole domain. (a) f-PICNN (Non-zero BC); (b) f-PICNN (Dirichlet BC); (c) PINN (Non-zero BC); (d) PINN (Dirichlet BC).

Table 8
Mean Absolute Errors between different scheme predictions and reference solution for 2D Burgers equation.

Scheme	PINN	f-PICNN (Euler)	f-PICNN (4^{th} Runge-Kutta)	BC
Error	1.553×10^{-2}	9.563×10^{-3}	3.911×10^{-3}	Non-zero
Error	6.037×10^{-2}	9.742×10^{-3}	5.876×10^{-3}	Dirichlet

enforcement is used for Dirichlet BCs while the hard BC enforcement is adopted for Neumann BCs in network training, and both of them can give desirable results. Several numerical experiments are carries out to demonstrate the performance of the proposed method on some 1D and 2D PDEs, including the Consolidation equation; Allen-Cahn equation; Burgers equation; Advection equation; Diffusion equation; 2D Consolidation equation and 2D Burgers equation. The performance of f-PICNN with the baseline algorithms, the vanilla PINN methods [57] are also compared, there was observed that the proposed framework has better performance on Consolidation equation, Allen-Cahn equation and 2D problems, and the training time of f-PICNN is significantly less than that of PINN on 2D problems. The main reason for this difference is that PINN faces convergence difficulties on some problems, e.g., it either has difficulty to converge to correct solution or it costs much more time due to the large training data or heavy parameters in PINN. In contrast, f-PICNN is more efficient on time-dependent PDEs than fully connected neural network due to its conceptual simplicity and faster convolutional operation based on memory mechanism. In addition, through investigation and comparison of the research results in other literature [55,67], f-PICNN takes a relatively shorter time while ensuring high accuracy. The testing results of the numerical experiments in this study are in good agreement with the real dynamics, which show the excellent ability of f-PICNN as a reliable

surrogate model to simulate the physical systems, and the promising potential to solve the inverse problems and higher-dimensional problems.

However, there are some improvements and future research perspectives to be made of current work: (1) The proposed method cannot be used directly on domains with non-uniform grids, but there is existing research that can be referred to, and it will be studied in the future; (2) To achieve high solution accuracy while ensuring high time efficiency, we did not consider combining the proposed network with LSTM or RNN. However, when faced with high-dimensional complex problems, such models can be considered. (3) The discrete methods can be extended to more efficient schemes to ensure both reasonable results and absolute convergence in solving PDEs. Better algorithms can be considered to design filters; (4) Although most of the results show that the 4^{th} Runge-Kutta scheme has better accuracy, whether it improves the performance of the neural network needs further study since neural network training is affected by multiple factors, the Runge-Kutta scheme does not always output good results. (5) A potential limitation of the proposed method is that the number of trainable parameters scales with time. Hence, extrapolation over longer time intervals could be challenging. (6) Adaptive hyperparameter tuning will be considered in future research. (7) Whether the proposed network is suitable for coupled PDEs will be studied in the future.

CRediT authorship contribution statement

Biao Yuan: Writing – original draft, Visualization, Software, Methodology, Formal analysis, Data curation, Conceptualization. **He Wang:** Writing – review & editing, Supervision, Conceptualization. **Ana Heitor:** Writing – review & editing, Supervision, Conceptualization. **Xiaohui Chen:** Writing – review & editing, Supervision, Conceptualization.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgments

The first author would like to acknowledge the financial support from the China Scholarship Council-University of Leeds Scholarships for his study at the University of Leeds. The authors gratefully acknowledge the financial support from HORIZON 2020 through the MSCA Staff Exchanges: GRID programme.

Appendix A. Loss history

The training loss histories for these PDEs are similar. Therefore, we choose the 1D consolidation equation as a representative example to illustrate the convergence histories of PINN and f-PICNN (see Fig. 16). Both methods achieve satisfactory performance in the training stage. The training loss of PINN (at final epoch: 5.97×10^{-4}) is larger than that of f-PICNN (at final epoch: 7.79×10^{-6}) for three reasons: (1) PINN uses more collocation points than the grid size of f-PICNN, which hinders its convergence; (2) The training epoch of PINN is much shorter than that of f-PICNN. Therefore, a longer history of loss reduction is observed in the PINN. (3) In addition, Fig. 16(c) indicates the presence of memory mechanism to accelerate the convergence in f-PICNN.

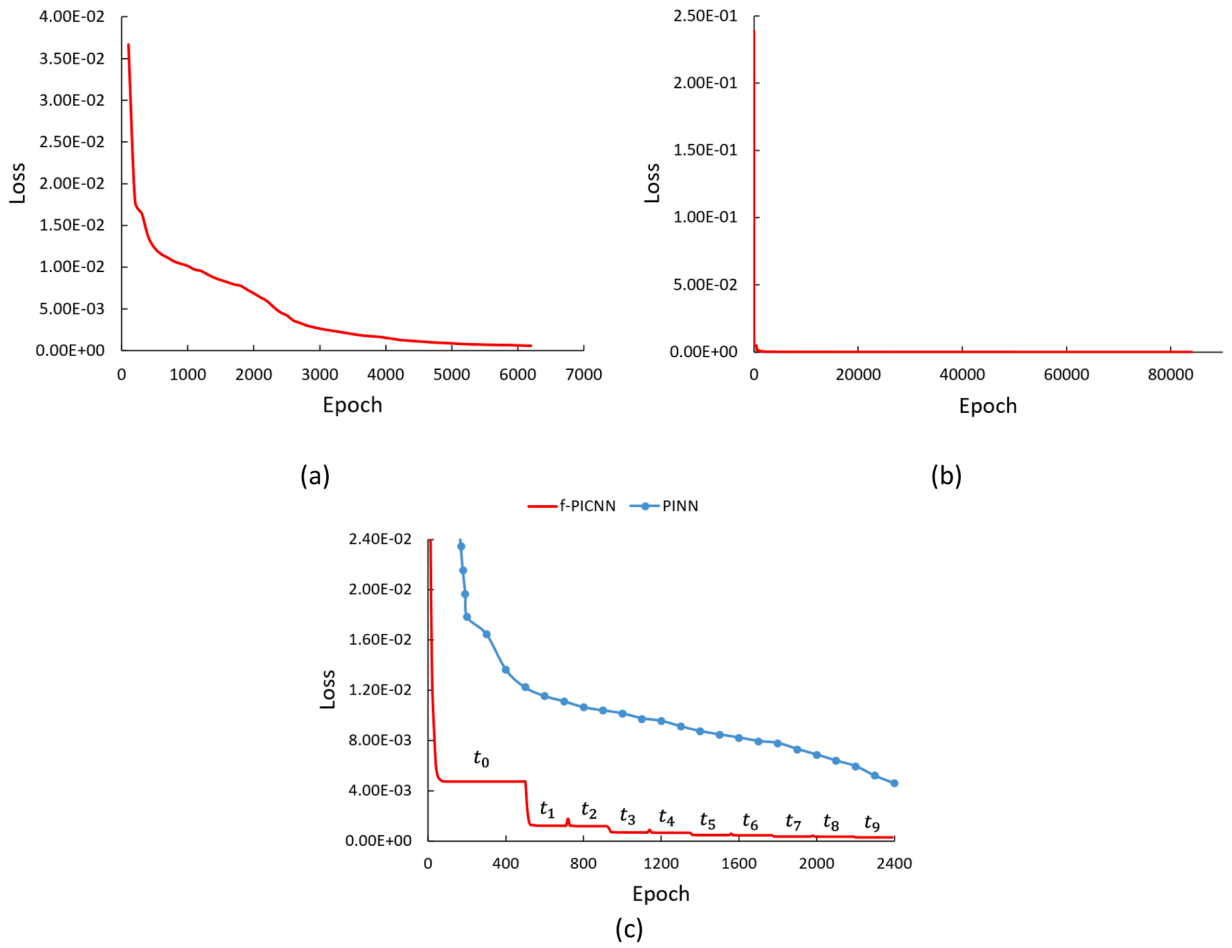


Fig. 16. The comparisons of training loss between PINN and f-PICNN. (a) Training loss of PINN; (b) Training loss of f-PICNN; (c) Training Loss for the first 2400 epochs (first 10 time steps of f-PICNN).

Appendix B. Relative computational cost and efficiency

To further quantify the computational overhead of f-PICNN, the Relative Computational Cost (RCC) of f-PICNN to PINN can be defined as the training time of f-PICNN ($T_{f-PICNN}$) divided by the training time of PINN (T_{PINN}). The 4th-order Runge-Kutta scheme for f-PICNN is considered the benchmark for all cases in this part. Table 9 shows the relative computational cost for different PDEs. f-PICNN has a distinct advantage in two-dimensional problems, with significantly shorter computation times and faster convergence.

$$RCC = \frac{T_{f-PICNN}}{T_{PINN}} \tag{46}$$

Table 9
The relative computational cost of solving different PDEs using PINN and f-PICNN.

Equation	Computation Time (s)		Relative Computational Cost	
	PINN	f-PICNN	PINN	f-PICNN
Consolidation	160	345	1.00	2.16
Allen-Cahn	115	740	1.00	6.43
Burgers	120	941	1.00	7.84
Advection	16	716	1.00	44.75
Diffusion	485	648	1.00	1.37
2D Consolidation	**2953**	**835**	**1.00**	**0.28**
2D Burgers	**3584**	**459**	**1.00**	**0.13**

In addition, to further quantify the computational performance of f-PICNN, the Relative Computational Efficiency (RCE) of f-PICNN to PINN can be defined as the Relative Computational Error Scale Factor (RCESF) divided by the Relative Computational Cost (RCC),

and the RCESF can be defined as the error of PINN ($Error_{PINN}$) divided by the error of f-PICNN ($Error_{f-PICNN}$). Table 10 shows the relative computational efficiency for different PDEs. f-PICNN has a remarkable advantage in the Consolidation equation, Allen-Cahn equation and 2D problems. In the specific case of the 2D Burgers equation, f-PICNN is 79 times more computationally efficient than PINN.

$$RCESF = \frac{Error_{PINN}}{Error_{f-PICNN}} \tag{47}$$

$$RCE = \frac{RCESF}{RCC} \tag{48}$$

Table 10
The relative computational efficiency of solving different PDEs using PINN and f-PICNN.

Equation	Error			Relative Computational Efficiency	
	PINN	f-PICNN	RCESF	PINN	f-PICNN
Consolidation	$\mathbf{8.861 \times 10^{-3}}$	$\mathbf{5.856 \times 10^{-4}}$	**15.132**	**1.00**	**7.005**
Allen-Cahn	$\mathbf{5.533 \times 10^{-1}}$	$\mathbf{5.830 \times 10^{-3}}$	**94.906**	**1.00**	**14.760**
Burgers	1.285×10^{-2}	1.819×10^{-3}	7.064	1.00	0.901
Advection	2.451×10^{-3}	3.046×10^{-3}	0.805	1.00	0.018
Diffusion	$\mathbf{9.846 \times 10^{-4}}$	$\mathbf{6.246 \times 10^{-4}}$	**1.576**	**1.00**	**1.151**
2D Consolidation	$\mathbf{2.846 \times 10^{-2}}$	$\mathbf{6.290 \times 10^{-3}}$	**4.525**	**1.00**	**16.159**
2D Burgers	$\mathbf{6.037 \times 10^{-2}}$	$\mathbf{5.876 \times 10^{-3}}$	**10.274**	**1.00**	**79.031**

For more insightful conclusions and detailed analysis, Fig. 17 displays the computation time and error of PINN and f-PICNN discussed for all PDEs. It indicates that f-PICNN has advantages over PINN in computational efficiency. Furthermore, the errors reported in the literature for SA-PINN and LA-PINN [42,66] in solving the Allen-Cahn equation are 3.34×10^{-2} and 8.22×10^{-3} with the computation time of 1300s and 3600s, respectively. f-PICNN shows better performance than them on this problem (5.83×10^{-3} with 740s).

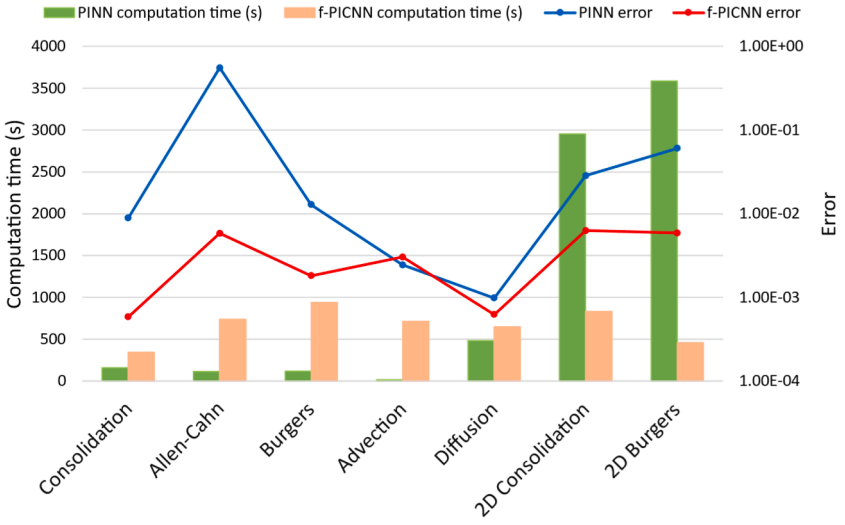


Fig. 17. The computation time and error of PINN and f-PICNN for all discussed PDEs.

Appendix C. Noisy data

The Allen-Cahn equation is used as a benchmark to test the effect of noisy initial condition data on the performance of f-PICNN. The noisy data is created from the exact data according to Gaussian distribution:

$$u_{noise} = u_{exact} + \varepsilon \cdot STD(u_{exact}) \cdot N(0,1) \tag{49}$$

The u_{exact} and u_{noise} are the vector of the exact data and the noisy data, respectively; ε is the magnitude of gaussian noise level (%); $STD(\cdot)$ is the operation of the standard deviation, and $N(0,1)$ is a randomly generated vector conforming to the Gaussian Distribution (GD).

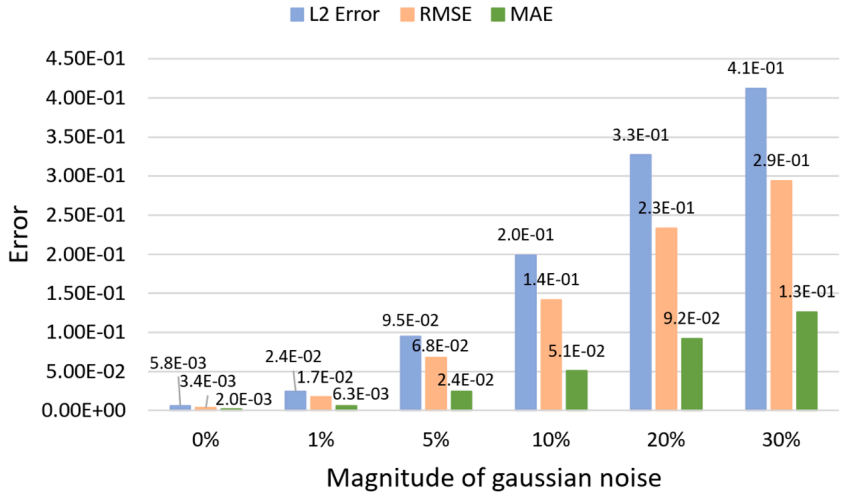


Fig. 18. The influence of noise level of initial condition on the accuracy of f-PICNN on Allen-Cahn equation.

Fig. 18 shows that the accuracy declines as the Gaussian noise level increases: f-PICNN is sensitive to noisy data and clean data helps f-PICNN to get good results. The reason for this is that adding noise to the initial conditions is equivalent to telling f-PICNN a wrong initial condition, and errors in these initial conditions are extrapolated to the entire domain. Meanwhile, a higher noise level in initial conditions means that a larger error will be introduced.

Appendix D. Loss weight

The Allen-Cahn equation is used as a benchmark to study the effect of loss weight on the performance of f-PICNN. Considering adding a weight α to the PDE loss term L_f, therefore, the loss function is:

$$L = L_u + \alpha \cdot L_f \tag{50}$$

Fig. 19 demonstrates that the loss function weight exerts minimal influence on the performance of f-PICNN. This is due to the fact that the boundary conditions represent a relatively minor aspect of the one-dimensional problem during a single time step operation. If the boundary condition is hard-coded constrained, it will not be included in the loss function term and there will be no loss weight.

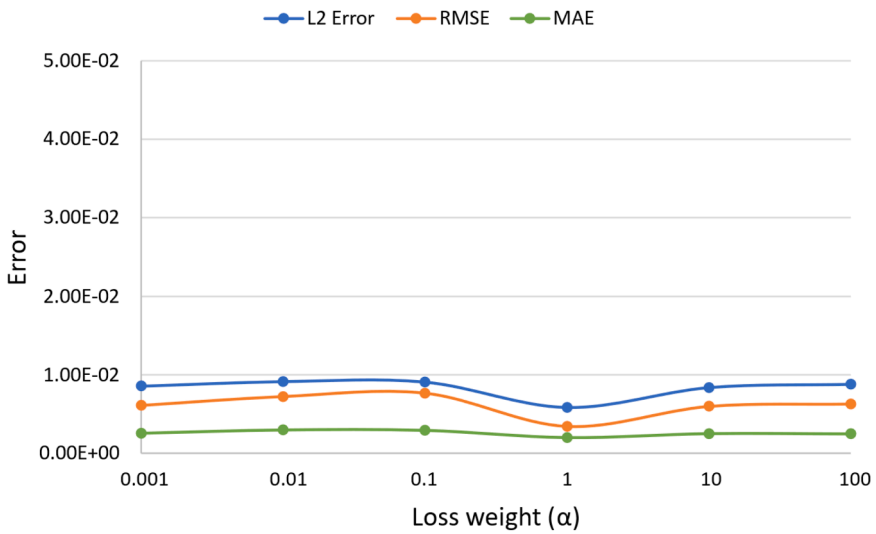


Fig. 19. The influence of loss weight on the accuracy of f-PICNN on Allen-Cahn equation.

Appendix E. L1 and L2 loss weight regularization for f-PICNN

Previous research has investigated the influence of L1 or L2 regularization on PINN performance [23,48,54,62,84]. This study will examine the impact of L1 and L2 on f-PICNN. The Allen-Cahn equation is used as a benchmark to study the effect of weight regularization in the loss function on the performance of f-PICNN. The loss functions L1 and L2 considering L1 weight regularization and L2 weight regularization are shown below.

$$L_1 = L_u + L_f + \lambda \sum_i |\omega_i| \tag{51}$$

$$L_2 = L_u + L_f + \lambda \sum_i \omega_i^2 \tag{52}$$

Fig. 20 demonstrates that the error of the model increases as the strength of the weight regularization penalty increases. And the model with L2 weight regularization works better than that with L1 weight regularization. The findings indicate that f-PICNN does not necessitate the application of weight regularization.

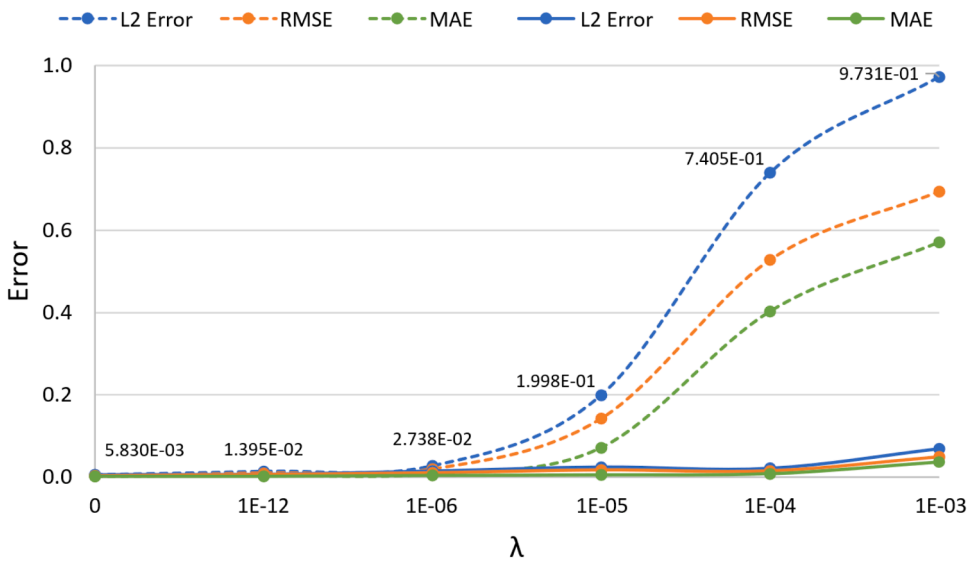


Fig. 20. The influence of L1 and L2 weight regularization on the accuracy of f-PICNN on Allen-Cahn equation. The dashed line represents L1 weight regularization and the solid line indicates L2 weight regularization.

References

[1] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Comput. Mech. 64 (2019) 525–545.
[2] K. Bhattacharya, B. Hosseini, N.B. Kovachki, A.M. Stuart, Model reduction and neural networks for parametric PDEs, SMAI J. Comput. Math. 7 (2021) 121–157.
[3] Bischof, R. & Kraus, M. 2021. Multi-objective loss balancing for physics-informed deep learning. *arXiv preprint* arXiv:2110.09813.
[4] J. Blazek, Computational Fluid Dynamics: Principles and Applications, Butterworth-Heinemann, 2015.
[5] Bullwinkel, B., Randle, D., Protopapas, P. & Sondak, D. 2022. Deqgan: Learning the loss function for pinns with generative adversarial networks. *arXiv preprint* arXiv:2209.07081.
[6] S. Cai, Z. Mao, Z. Wang, M. Yin, G.E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: a review, Acta Mech. Sin. 37 (2021) 1727–1738.
[7] S. Cai, Z. Wang, L. Lu, T.A. Zaki, G.E. Karniadakis, DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks, J. Comput. Phys. (2021) 436.
[8] S. Cai, Z. Wang, S. Wang, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks for heat transfer problems, J. Heat. Transfer. 143 (2021) 060801.
[9] J.G. Charney, R. Fjörtoft, J.V Neumann, Numerical integration of the barotropic vorticity equation, Tellus 2 (1950) 237–254.
[10] M. Chen, R. Niu, W. Zheng, Adaptive multi-scale neural network with resnet blocks for solving partial differential equations, Nonlinear. Dyn. 111 (2023) 6499–6518.
[11] Y. Chen, L. Lu, G.E. Karniadakis, L. Dal Negro, Physics-informed neural networks for inverse problems in nano-optics and metamaterials, Opt. Express. 28 (2020) 11618–11633.
[12] W. Dai, Convolutional neural network based simulation and analysis for backward stochastic partial differential equations, Comput. Math. Appl. 119 (2022) 21–58.
[13] Dolean, V., Heinlein, A., Mishra, S. & Moseley, B. 2022. Finite basis physics-informed neural networks as a Schwarz domain decomposition method. *arXiv preprint* arXiv:2211.05560.
[14] Farimani, A. B., Gomes, J. & Pande, V. S. 2017. Deep learning the physics of transport phenomena. *arXiv preprint* arXiv:1709.02432.
[15] H. Gao, L. Sun, J.-X. Wang, PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, J. Comput. Phys. 428 (2021) 110079.
[16] H. Gao, L. Sun, J.-X. Wang, PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, J. Comput. Phys. 428 (2021) 110079.
[17] H. Gao, M.J. Zahr, J.-X. Wang, Physics-informed graph neural Galerkin networks: a unified framework for solving PDE-governed forward and inverse problems, Comput. Methods Appl. Mech. Eng. 390 (2022) 114502.
[18] N. Geneva, N. Zabaras, Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks, J. Comput. Phys. 403 (2020) 109056.
[19] Goyal, P. & Benner, P. 2021. Learning dynamics from noisy measurements using deep learning with a Runge-Kutta constraint. *arXiv preprint* arXiv:2109.11446.
[20] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Eng. 379 (2021) 113741.
[21] Hu, Y., Zhao, T., Xu, S., Xu, Z. & Lin, L. 2009. Neural-PDE: A rnn based neural network for solving time dependent PDEs (2020). *arXiv preprint arXiv*.
[22] Hu, Z., Jagtap, A. D., Karniadakis, G. E. & Kawaguchi, K. 2022. Augmented Physics-Informed Neural Networks (APINNs): a gating network-based soft domain decomposition methodology. *arXiv preprint* arXiv:2211.08939.
[23] B. Huang, J. Wang, Applications of physics-informed neural networks in power systems-a review, IEEE Trans. Power Syst. 38 (2022) 572–588.
[24] A.D. Jagtap, K. Kawaguchi, G.E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404 (2020) 109136.
[25] Jiang, L., Wang, L., Chu, X., Xiao, Y. & Zhang, H. PhyGNNet: solving spatiotemporal PDEs with physics-informed graph neural network. 2023. 143-147.
[26] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (2021) 422–440.
[27] E. Kharazmi, Z. Zhang, G.E. Karniadakis, hp-VPINNs: Variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Eng. 374 (2021) 113547.
[28] Kingma, D. P. & Ba, J. 2014. Adam: a method for stochastic optimization. *arXiv preprint* arXiv:1412.6980.

[29] S. Koric, D.W. Abueidda, Data-driven and physics-informed deep learning operators for solution of heat conduction equation with parametric heat source, Int. J. Heat. Mass Transf. 203 (2023) 123809.
[30] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M.W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 26548–26560.
[31] R.J. Leveque, Finite difference methods for differential equations, Draft Version For Use A Math 585 (1998) 112.
[32] K. Li, K. Tang, T. Wu, Q. Liao, D3M: a deep domain decomposition method for partial differential equations, IEEe Access. 8 (2020) 5283–5294.
[33] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. & Anandkumar, A. 2020b. Fourier neural operator for parametric partial differential equations. *arXiv preprint* arXiv:2010.08895.
[34] Li, Z., Meidani, K. & Farimani, A. B. 2022. Transformer for partial differential equations' operator learning. *arXiv preprint* arXiv:2205.13671.
[35] Liu, X., Xu, B. & Zhang, L. 2022a. HT-Net: Hierarchical Transformer based Operator Learning Model for Multiscale PDEs. *arXiv preprint* arXiv:2210.10890.
[36] X. Liu, X. Zhang, W. Peng, W. Zhou, W. Yao, A novel meta-learning initialization method for physics-informed neural networks, Neural Comput. Appl. 34 (2022) 14511–14534.
[37] Z. Long, Y. Lu, B. Dong, PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network, J. Comput. Phys. 399 (2019) 108925.
[38] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (2021) 218–229.
[39] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, DeepXDE: a deep learning library for solving differential equations, SIAM Rev. 63 (2021) 208–228.
[40] A. Mavi, A.C. Bekar, E. Haghighat, E. Madenci, An unsupervised latent/output physics-informed convolutional-LSTM network for solving partial differential equations using peridynamic differential operator, Comput. Methods Appl. Mech. Eng. 407 (2023) 115944.
[41] Mcclenny, L. & Braga-Neto, U. 2020. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint* arXiv:2009.04544.
[42] L.D. Mcclenny, U.M. Braga-Neto, Self-adaptive physics-informed neural networks, J. Comput. Phys. 474 (2023) 111722.
[43] X. Meng, Z. Li, D. Zhang, G.E. Karniadakis, PPINN: parareal physics-informed neural network for time-dependent PDEs, Comput. Methods Appl. Mech. Eng. 370 (2020) 113250.
[44] Moseley, B., Markham, A. & Nissen-Meyer, T. 2021. Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations. *arXiv preprint* arXiv:2107.07871.
[45] Nascimento, R. G. & Viana, F. A. 2019. Fleet prognosis with physics-informed recurrent neural networks. *arXiv preprint* arXiv:1901.05512.
[46] Ovadia, O., Kahana, A., Stinis, P., Turkel, E. & Karniadakis, G. E. 2023. ViTO: Vision Transformer-Operator. *arXiv preprint* arXiv:2303.08891.
[47] A.G. Özbay, A. Hamzehloo, S. Laizet, P. Tzirakis, G. Rizos, B. Schuller, Poisson CNN: convolutional neural networks for the solution of the Poisson equation on a Cartesian mesh, Data-Centric Eng. 2 (2021) e6.
[48] G. Pang, L. Lu, G.E. Karniadakis, fPINNs: Fractional physics-informed neural networks, SIAM J. Sci. Comput. 41 (2019) A2603–A2626.
[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, Pytorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019) 8026–8037.
[50] R.G. Patel, N.A. Trask, M.A. Wood, E.C. Cyr, A physics-informed operator regression framework for extracting data-driven continuum models, Comput. Methods Appl. Mech. Eng. 373 (2021) 113500.
[51] Y.T. Peet, M.M. Peet, A new treatment of boundary conditions in PDE solution with Galerkin methods via Partial Integral Equation framework, J. Comput. Appl. Math. 442 (2024) 115673.
[52] Penwarden, M., Jagtap, A. D., Zhe, S., Karniadakis, G. E. & Kirby, R. M. 2023. A unified scalable framework for causal sweeping strategies for Physics-Informed Neural Networks (PINNs) and their temporal decompositions. *arXiv preprint* arXiv:2302.14227.
[53] A.F. Psaros, K. Kawaguchi, G.E. Karniadakis, Meta-learning PINN loss functions, J. Comput. Phys. 458 (2022) 111121.
[54] Pua, J. & Chena, Y. 2021. Data-driven forward-inverse problems and modulational instability for Yajima-Oikawa system using deep learning with parameter regularization. *arXiv preprint* arXiv:2112.04062.
[55] J. Qu, W. Cai, Y. Zhao, Learning time-dependent PDEs with a linear and nonlinear separate convolutional neural network, J. Comput. Phys. 453 (2022) 110928.
[56] Radford, A., Metz, L. & Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint* arXiv:1511.06434.
[57] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[58] R. Ranade, C. Hill, J. Pathak, DiscretizationNet: a machine-learning based solver for Navier–Stokes equations using finite volume discretization, Comput. Methods Appl. Mech. Eng. 378 (2021) 113722.
[59] R. Ranade, C. Hill, J. Pathak, DiscretizationNet: a machine-learning based solver for Navier–Stokes equations using finite volume discretization, Comput. Methods Appl. Mech. Eng. 378 (2021) 113722.
[60] P. Ren, C. Rao, Y. Liu, J.-X. Wang, H. Sun, PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs, Comput. Methods Appl. Mech. Eng. 389 (2022) 114399.
[61] T. Sauer, Numerical Analysis, Addison-Wesley Publishing Company, 2011.
[62] Schäfer, V. 2022. Generalization of physics-informed neural networks for various boundary and initial conditions.
[63] Shin, Y., Darbon, J. & Karniadakis, G. E. 2020. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. *arXiv preprint* arXiv:2004.01806.
[64] K. Shukla, A.D. Jagtap, G.E. Karniadakis, Parallel physics-informed neural networks via domain decomposition, J. Comput. Phys. 447 (2021).
[65] G.D. Smith, Numerical Solution of Partial Differential Equations: Finite Difference Methods, Oxford university press, 1985.
[66] Y. Song, H. Wang, H. Yang, M.L. Taccari, X. Chen, Loss-attentional physics-informed neural networks, J. Comput. Phys. 501 (2024) 112781.
[67] Stevens, B. & Colonius, T. 2020. FiniteNet: A fully convolutional LSTM network architecture for time-dependent partial differential equations. *arXiv preprint* arXiv:2002.03014.
[68] K. Tang, X. Wan, C. Yang, DAS-PINNs: a deep adaptive sampling method for solving high-dimensional partial differential equations, J. Comput. Phys. 476 (2023) 111868.
[69] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepONets, Sci. Adv. 7 (2021) eabi8605.
[70] X. Wang, S. Zhu, Y. Guo, P. Han, Y. Wang, Z. Wei, X. Jin, TransFlowNet: a physics-constrained Transformer framework for spatio-temporal super-resolution of flow simulations, J. Comput. Sci. 65 (2022) 101906.
[71] N. Winovich, K. Ramani, G. Lin, ConvPDE-UQ: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains, J. Comput. Phys. 394 (2019) 263–279.
[72] C. Wu, M. Zhu, Q. Tan, Y. Kartha, L. Lu, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, Comput. Methods Appl. Mech. Eng. 403 (2023) 115671.
[73] K. Wu, D. Xiu, Data-driven deep learning of partial differential equations in modal space, J. Comput. Phys. 408 (2020) 109307.
[74] S. Wu, A. Zhu, B. Lu, Convergence of Physics-Informed Neural Networks Applied to Linear Second-Order Elliptic Interface Problems, Commun. Comput. Phys. 33 (2023) 596–627.
[75] Z. Xiang, W. Peng, X. Liu, W. Yao, Self-adaptive loss balanced Physics-informed neural networks, Neurocomputing. 496 (2022) 11–34.
[76] Xiao, X. & Qiu, W. 2022. Numerical approximation based on deep convolutional neural network for high-dimensional fully nonlinear merged PDEs and 2BSDEs. *arXiv preprint* arXiv:2209.04997.
[77] L. Yang, D. Zhang, G.E. Karniadakis, Physics-informed generative adversarial networks for stochastic differential equations, SIAM J. Sci. Comput. 42 (2020) A292–A317.

[78] J. Yu, L. Lu, X. Meng, G.E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Comput. Methods Appl. Mech. Eng. 393 (2022) 114823.
[79] S. Zeng, Z. Zhang, Q. Zou, Adaptive deep neural networks methods for high-dimensional partial differential equations, J. Comput. Phys. 463 (2022) 111232.
[80] D. Zhang, L. Lu, L. Guo, G.E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, J. Comput. Phys. 397 (2019) 108850.
[81] G. Zhang, H. Yang, G. Pan, Y. Duan, F. Zhu, Y. Chen, Constrained self-adaptive physics-informed neural networks with resnet block-enhanced network architecture, Mathematics 11 (2023) 1109.
[82] Zhang, S., Zhang, C. & Wang, B. 2022. MRF-PINN: a Multi-Receptive-Field convolutional physics-informed neural network for solving partial differential equations. *arXiv preprint* arXiv:2209.03151.
[83] Z. Zhang, A physics-informed deep convolutional neural network for simulating and predicting transient Darcy flows in heterogeneous reservoirs without labeled data, J. Pet. Sci. Eng. 211 (2022) 110179.
[84] H. Zhou, Parallel Physics-Informed Neural Networks Method with Regularization Strategies for the Forward-Inverse Problems of the Variable Coefficient Modified KdV Equation, J. Syst. Sci. Complex 37 (2024) 511–544.
[85] Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. 366 (2018) 415–447.
[86] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, J. Comput. Phys. 394 (2019) 56–81.
[87] O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, The Finite Element Method: Its Basis And Fundamentals, Elsevier, 2005.