Fractional Physics-informed Neural Networks for Time-fractional Phase Field Models

Shupeng Wang
shandong university
Hui Zhang (zhangh@sdu.edu.cn)
Shandong University
Xiaoyun Jiang
shandong university

Research Article

Keywords: Time-fractional phase field models, FPINNs, Fractional forward problem, Fractional inverse problem

Posted Date: November 8th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-993221/v1

Noname manuscript No.
(will be inserted by the editor)

Fractional physics-informed neural networks for time-fractional phase field models

Shupeng Wang · Hui Zhang✉ · Xiaoyun Jiang

Received: date / Accepted: date

Abstract In this paper, a new fractional physics-informed neural networks (fPINNs) is proposed, which combines f-PINNs with spectral collocation method to solve the time-fractional phase field models. Compared to fPINNs, it has large representation capacity due to the property of spectral collocation method, which reduces the number of approximate points of discrete fractional operators, improves the training efficiency and has higher error accuracy. Unlike traditional numerical method, it directly optimizes the spectral collocation coefficient, saves the time of matrix calculation, is easy to deal with the high-dimensional model, and also has higher error accuracy. First, fPINNs based on a spectral collocation method is used to obtain the numerical solutions of the models under consideration. The spectral collocation method is used to discretize the space direction, and the fractional backward difference formula is used to approximate the time-fractional derivative. The error accuracy in different cases is discussed, and it is observed that the point-wise error is 10^{-5} to 10^{-7}. Next, fPINNs is employed to solve several inverse problems in time-fractional phase field models to identify the order of fractional derivative, mobility constant, and other coefficients. The results of numerical experiments are presented to prove the effectiveness of fPINNs in solving time-fractional phase field models and their inverse problems.

Keywords Time-fractional phase field models · FPINNs · Fractional forward problem · Fractional inverse problem

S. Wang · H. Zhang✉ · X. Jiang
School of Mathematics, Shandong University,
Jinan 250100, People's Republic of China
E-mail: zhangh@sdu.edu.cn

1 Introduction

With the recent explosion in the amount of available data and computing resources, advances in neural networks (NNs) and data analysis have produced transformative results in various scientific fields. Studies on applications of NNs can be divided into two main categories. The first category involves data-driven NNs using only information from the observed data. For example, a number of researchers have discussed the applications of deep learning in fluid [1,2] and thermal simulations [3], material property prediction [4], system monitoring [5], and structure analysis [6,7]. These studies use a data-driven approach because they either lack physics-based models or rely on computationally expensive models. However, such an approach is obviously inappropriate for mathematical models with explicit expressions. Thus, a different type of NNs has been developed, explicitly utilizing information from equations by involving the equations directly in the loss function to be optimized. For example, Karpatne et al. [8] conceptualized a theory-driven data science framework and proposed several approaches to integrate domain knowledge into loss functions. Raissi et al. [9] proposed physics-informed neural networks (PINNs) to solve partial differential equations (PDEs) by incorporating the constraints of the PDEs into the loss function. They proved the robustness of PINNs and discussed both the forward and the inverse problems of PDEs. For other examples, see references [10–13].

The fractional derivative, an effective and powerful tool for describing physical systems with long-term memory, has become popular in mathematics and physics [14,15]. Finding stable and accurate methods of approximating fractional PDEs (FPDEs) has been the goal of some researchers. In particular, with the development of NNs for solving classical PDEs, the use of NNs to solve FPDEs has become a research hotspot. For example, fractional PINNs (fPINNs)

[16] can bypass the problem of automatic differentiation not being applicable to fractional operators; thus, it is possible to solve space–time fractional advection–diffusion equations and obtain a convergence accuracy of 10^{-3} to 10^{-4} by discretizing the fractional derivative. Subsequently, Rostami [17] bypassed the difficulty of dealing with the space-fractional derivative by replacing the solution with a series expansion. This enabled them to solve high-order linear FPDEs with greater accuracy than that achieved with traditional techniques.

Phase field models have been widely used to study material [18], physical [19], and biological systems [20,21]. In particular, phase field models have made great contributions to the study of coarsening dynamics [22,23], which is widely observed in material systems. However, there is an increasing need to study more complex mixtures of materials, or materials with elastic memory, which results in significant changes in the scaling dynamics. Classical phase field models are no longer suitable for studying such anomalous coarsening dynamics; therefore, new models are needed. In recent years, fractional phase field models have attracted much research attention, and many studies have focused on numerical analysis of time-fractional phase field models [24]. Based on spectral methods, Antil et al. [25] analyzed the features and numerical solutions of fractional phase field models. An efficient finite difference scheme and a Fourier spectral scheme were developed to solve the nonlocal behavior of time-fractional phase field models in [26]. In [27], the authors employed a Petrov–Galerkin spectral method to discretize the spatial derivative and a stabilized ADI scheme to discretize the time-fractional derivative; they proved that the method had spectral accuracy and could deal with both erroneous boundary effects and sharpening of the interface at no extra resolution.

An important problem related to time-fractional phase field models is their inverse problem. The core of the inverse problem is the estimation of the unknown parameters in the model according to the known model and some observed data. Wu et al. [28] proved the Lipschitz stability and uniqueness of the inverse problem of phase field models with an inertial term. Stability results concerning the inverse problem of determining two time-independent coefficients with the observation data were presented in [29]. For other examples, please see references [30,31]. Many works have focused on the inverse problem of classic phase field models, but few have considered time-fractional phase field models. A NNs approach has been applied to several models. Bandai & Ghezzehei [32] employed PINNs to solve the inverse problem of the Richardson–Richards equation without initial and boundary conditions, and fPINNs to obtain the order of fractional derivative and diffusion coefficient of the space–time fractional advection-diffusion equation from observations was derived in [16].

In this work, we focus on the applications of fPINNs for solving forward and inverse problems of time-fractional phase field models. A systematic study is performed by using a spectral collocation method to discretize the space direction, in order to accelerate training and enhance accuracy. Specifically, the influence of the structure of the spectral collocation method on the numerical results is discussed systematically. Numerical results based on different numbers of spectral collocation points, neurons, and approximation points are also discussed. For inverse problems, we solve one-dimensional (1D) and 2D time-fractional phase field models with fPINNs, present the results, and analyze the causes of the error. The highlights of our paper are as follows.

- By using a shallow-layer NNs, the loss function optimization process can be expressed analytically, which means that simple numerical analyses of these formulas can be performed.
- With the special structure of loss function, we can make full use of the data and model information to achieve the expected results.
- Owing to the combination of fPINNs with a spectral collocation method, the computational efficiency and the convergence accuracy are improved.

The rest of the paper is organized as follows. Section 2 presents the time-fractional phase field models. In Section 3, we develop fPINNs to solve the forward and inverse problems of the time-fractional phase field models. Several examples are presented in Section 4 to verify the effectiveness of fPINNs for both forward and inverse problems. Finally, some conclusions are discussed in Section 5.

2 Mathematical Model

The general form of the time-fractional phase field model can be written as [24]:

$$\begin{aligned} &D_t^{\alpha} u(\mathbf{x},t) = \gamma \mathfrak{g} \mu, \ \ (\mathbf{x},t) \in \Omega \times (0,T], \\ &\mu = -\varepsilon^2 \Delta u(\mathbf{x},t) + f(u(\mathbf{x},t)), \ \ (\mathbf{x},t) \in \Omega \times (0,T], \end{aligned} \tag{1}$$

and the initial boundary conditions are given as

$$\begin{cases} u(\mathbf{x},0) = g(\mathbf{x}), \ \mathbf{x} \in \Omega, \\ u(\mathbf{x},t) = h(\mathbf{x},t), \ (\mathbf{x},t) \in \partial\Omega \times (0,T], \end{cases} \tag{2}$$

where γ is a mobility constant. Eq. (1) denotes the time-fractional Allen–Cahn (AC) equation when $\mathfrak{g} = -1$. When $\mathfrak{g} = \Delta$, Eq. (1) represents the time-fractional Cahn–Hilliard (CH) equation. Here, μ is the chemical potential; and $f(\cdot)$ is a function satisfying $f(u) = F'(u)$, where $F(u)$ is a double-well potential that usually takes the form $F(u) = \frac{1}{4}(1-u^2)^2$.

The left-hand side of the first equation is the Caputo fractional derivative of the order α, defined as follows [24]:

$$D_t^\alpha u(\mathbf{x},t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{u'(\mathbf{x},\tau)}{(t-\tau)^\alpha} d\tau, \ \ (0<\alpha<1), \tag{3}$$

where $\Gamma(\cdot)$ is the gamma function. As $\alpha \to 1$, it reduces to the first derivative. $g(\mathbf{x})$ and $h(\mathbf{x},t)$ are some (sufficiently) known functions, which we set to be the white-box (W-B) function here. It is worth noting that although periodic boundary conditions are used as an example in this paper, the proposed method can also be applied to the Dirichlet and Neumann boundary conditions. In addition, in order to satisfy the discrete method [24], we limit the order of α to $(0,1)$.

3 fPINNs

3.1 Forward problem with fPINNs based on spectral collocation method

In this subsection, a new fPINNs based on spectral collocation method is developed to solve the forward problems of time-fractional phase field models. We consider the forward problem of this form, as well as the corresponding NNs, as shown in Fig. 1:

$$\begin{aligned} &D_t^\alpha u(\mathbf{x},t) + L\{u(\mathbf{x},t)\} = f(\mathbf{x},t), \ \ (\mathbf{x},t) \in \Omega \times (0,T], \\ &u(\mathbf{x},0) = g(\mathbf{x}), \ \ \mathbf{x} \in \Omega, \\ &u(\mathbf{x},t) = h(\mathbf{x},t), \ \ (\mathbf{x},t) \in \partial\Omega \times (0,T], \end{aligned} \tag{4}$$

where $L\{u(\mathbf{x},t)\} = -\varepsilon^2 \Delta u(\mathbf{x},t) - u(\mathbf{x},t) + u^3(\mathbf{x},t)$ is a nonlinear operator. Let $\widetilde{u}$ be the approximated solution of Eq. (4), and let u_{NN} be an output of the NNs. In this paper, the second-order fractional backward differential formula (FBDF) [24] is used to discretize the time-fractional derivative, and the spectral collocation method is used to deal with the space direction.

We first divide the interval $[0,T]$ into K equal subintervals with a time-step of size $\tau = T/K$. Let $t_n = n\tau$, $u^n = u(\mathbf{x},t_n)$, $0 \le n \le K$. Using the FBDF method, we can discretize the fractional derivative $D_t^\alpha u$ as

$$[D_t^\alpha u]_{t=t_n} = D_\tau^\alpha u^n + R^n, \tag{5}$$

in which

$$D_\tau^\alpha u^n = \frac{1}{\tau^\alpha} \sum_{j=0}^{n} w_{n-j}^{(\alpha)} (u^j - u^0). \tag{6}$$

Here, $\{w_i^{(\alpha)}, \ i = 0,...,n\}$ are coefficients that can be generated using the following generating function:

$$w^{(\alpha)}(z) = (\frac{3}{2} - 2z + \frac{1}{2}z^2)^\alpha = \sum_{n=0}^{\infty} w_n^{(\alpha)} z^n. \tag{7}$$

Then, the expansion of $\widetilde{u}(\mathbf{x},t)$ by the Lagrange interpolation polynomials $\{\phi_k(\mathbf{x})\}_{i=0}^N$ at Legendre–Gauss–Lobatto $\{\mathbf{x}_j\}(j = 0,...,N)$ within the given Ω can be written as

$$\widetilde{u}(\mathbf{x},t) = \sum_{k=0}^{N} \phi_k(\mathbf{x}) c(\mathbf{x}_k,t), \quad \phi_k(\mathbf{x}) = \prod_{j=0, j\neq k}^{N} \frac{\mathbf{x}-\mathbf{x}_j}{\mathbf{x}_k-\mathbf{x}_j}, \tag{8}$$

where N is the number of spectral collocation points. And we use the notation $\widetilde{u}(\mathbf{x}_m,t_n) = \sum_{k=0}^N \phi_k(\mathbf{x}_m) c_k^n$.

Based on the spectral collocation method and the FBDF method, we have the following fully discrete scheme for Eq. (4):

$$\begin{aligned} &\frac{1}{\tau^\alpha} \sum_{s=0}^{n} \sum_{k=0}^{N} w_{n-s}^{(\alpha)} (\phi_k(\mathbf{x}_m) c_k^s - \phi_k(\mathbf{x}_m) c_k^0) + L\{u_m^n\} = f(\mathbf{x}_m,t_n), \\ &\sum_{k=0}^{N} \phi_k(\mathbf{x}_m) c_k^0 = g(\mathbf{x}_m), \\ &\sum_{k=0}^{N} \phi_k(\mathbf{x}_m) c_k^n = h(\mathbf{x}_m,t_n), \end{aligned} \tag{9}$$

where

$$\begin{aligned} L\{u_m^n\} = &-\varepsilon^2 \Delta (\sum_{k=0}^{N} \phi_k(\mathbf{x}_m) c_k^n) - \sum_{k=0}^{N} \phi_k(\mathbf{x}_m) c_k^n \\ &+ (\sum_{k=0}^{N} \phi_k(\mathbf{x}_m) c_k^n)^3. \end{aligned} \tag{10}$$

By using the output of the NNs, $u_{NN}(\mathbf{x}_k,t)$, to approximate the coefficients $\{c(\mathbf{x}_k,t), k = 0,..,N\}$, the explicit expression of the approximated solution $\widetilde{u}(\mathbf{x},t)$ is obtained. Based on the orthogonality of the Lagrange interpolation polynomials, we can also obtain

$$\widetilde{u}(\mathbf{x}_i,t_n) = \Sigma_j c_j^n \phi_j(\mathbf{x}_i) = c_i^n, \ \ j = 0,...,N, \tag{11}$$

where c_i^n is the value of solution $\widetilde{u}$ at spectral collocation point $(\mathbf{x}_i,t_n)$. Thus, we can use a single output u_{NN} to represent the values of the coefficients c_j and the approximated solution $\widetilde{u}$. In addition, compared with the fPINNs, we have made some changes to the input structure of the approximated solution and the loss function. Instead of integrating all the initial boundary conditions into the approximated solution, only the boundary condition is added to the approximated solution, and the initial condition is added to the loss function as a penalty term. This construction is intended not only to reduce the complexity of the loss function but also to optimize its direction for corrective purposes. It is advantageous for the network to approximate the distribution of the true value $u(\mathbf{x},t)$ and improve the generalization accuracy.

With the approximated solution

$$\widetilde{u}(\mathbf{x},t) = u_{NN}(\mathbf{x},t) = \sum_{k=0}^{N} u_{NN}(\mathbf{x}_k,t) \phi_k(\mathbf{x}) \rho(\mathbf{x}), \tag{12}$$

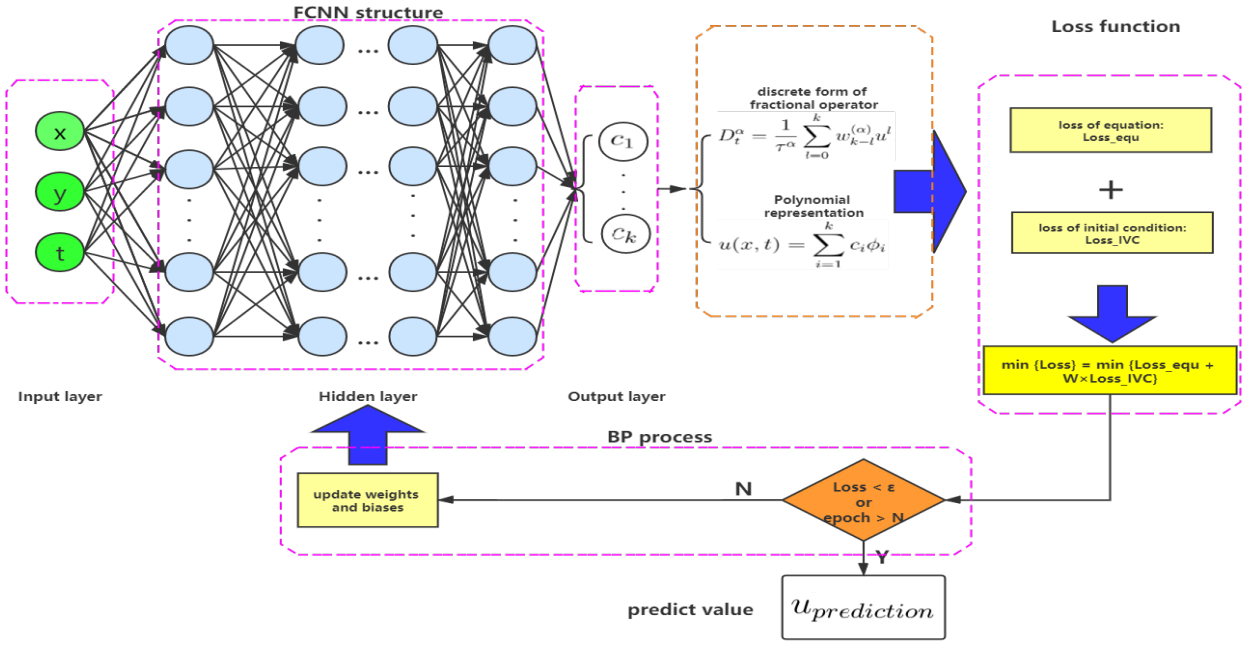

Fig. 1: Schematic of fPINNs based on the spectral collocation method. The structure of the loss function is shown at the top, and the network optimization process and output are shown at the bottom.

where $\rho(\mathbf{x})$ is pre-selected to satisfy $\sum_{k=0}^{N} u_{NN}(\mathbf{x}_k,t)\phi_k(\mathbf{x})$ $\rho(\mathbf{x}) = h(\mathbf{x},t)$ when $\mathbf{x} \in \partial\Omega$, the loss function can be defined as the mean-squared error (MSE) of the equation residual:

$$\mathrm{L}(\mu = \{w,b\}) = MSE_{equ}(\mu) + W_I \times MSE_{IVC}(\mu), \tag{13}$$

with

$$\begin{aligned} MSE_{equ}(\mu) = \frac{1}{N_{equ}} \sum_{m=1}^{N_{equ}} \Big(& \frac{1}{\tau^{\alpha}} \sum_{s=0}^{n} \sum_{k=0}^{N} w_{n-s}^{(\alpha)} (\phi_k(\mathbf{x}_m) u_{NN}(\mathbf{x}_k,t_s) \\ & - \phi_k(\mathbf{x}_m) u_{NN}(\mathbf{x}_k,0)) \rho(\mathbf{x}_m) - \varepsilon^2 \Delta \Big(\sum_{k=0}^{N} \phi_k(\mathbf{x}_m) \\ & u_{NN}(\mathbf{x}_k,t_n)) \rho(\mathbf{x}_m) - \sum_{k=0}^{N} \phi_k(\mathbf{x}_m) u_{NN}(\mathbf{x}_k,t_n) \rho(\mathbf{x}_m) \\ & + \Big(\sum_{k=0}^{N} \phi_k(\mathbf{x}_m) u_{NN}(\mathbf{x}_k,t_n) \rho(\mathbf{x}_m) \Big)^3 - f(\mathbf{x}_k,t_m) \Big)^2, \end{aligned} \tag{14}$$

$$MSE_{IVC}(\mu) = \frac{1}{N_{IVC}} \sum_{m=1}^{N_{IVC}} (u_{NN}(\mathbf{x}_m,0) - g(\mathbf{x}_m))^2. \tag{15}$$

Here, W_I is the initial weight; μ are the hyperparameters of the NNs; and N_{equ} and N_{IVC} denote training points in the spatio-temporal domain and initial condition, respectively. Then, the weights and the biases are updated by back-propagation as follows:

$$\begin{aligned} \frac{\partial L}{\partial w_{i,j}^{(h)}} &= \frac{\partial L}{\partial u_{NN}} \frac{\partial u_{NN}}{\partial w_{i,j}^{(h)}} \\ &= \frac{\partial MSE_{equ}}{\partial u_{NN}} \frac{\partial u_{NN}}{\partial w_{i,j}^{(h)}} + W_I \times \frac{\partial MSE_{IVC}}{\partial u_{NN}} \frac{\partial u_{NN}}{\partial w_{i,j}^{(h)}}, \end{aligned} \tag{16}$$

$$\begin{aligned} \frac{\partial L}{\partial b_{i}^{(h)}} &= \frac{\partial L}{\partial u_{NN}} \frac{\partial u_{NN}}{\partial b_{i}^{(h)}} \\ &= \frac{\partial MSE_{equ}}{\partial u_{NN}} \frac{\partial u_{NN}}{\partial b_{i}^{(h)}} + W_I \times \frac{\partial MSE_{IVC}}{\partial u_{NN}} \frac{\partial u_{NN}}{\partial b_{i}^{(h)}}. \end{aligned} \tag{17}$$

Using the gradient descent method, we can update these weights and biases as follows:

$$w_{i,j}^{(h+1)} = w_{i,j}^{(h)} - \eta_1 \cdot \frac{\partial L}{\partial w_{i,j}^{(h)}}, \tag{18}$$

$$b_{i}^{(h+1)} = b_{i}^{(h)} - \eta_2 \cdot \frac{\partial L}{\partial b_{i}^{(h)}}, \tag{19}$$

where $w_{i,j}^{(h)}$ and $b_i^{(h)}$ are the values of the iterations of $w_{i,j}$ and b_i for h, and we represent their initial values by $w_{i,j}^{(0)}$ and $b_i^{(0)}$, respectively. By iterative training, we obtain the

optimal network parameters and output the required numerical solution. The initialization of weights is among the most important problem in network design. There are many ways to determine the initial values of weights, including Gaussian initialization, uniform initialization, and Xavier initialization. In this work, we use Xavier initialization, which not only ensures that the information flow of each layer satisfies the same variance but also keeps the activation value and gradient variance of each layer consistent, thereby enabling better transmission of the information in the network.

Finally, the forward problem can be simply described as follows. First, the coefficients c_j and the approximated solution $\widetilde{u}$ are expressed by the output u_{NN} at the corresponding points. Then, the loss function is constructed by merging them into the mathematical model, in which MSE_{equ} is constructed by $\widetilde{u}$ in the form of a spectral collocation expression and MSE_{IVC} is constructed by the output u_{NN}. Finally, the loss function of the network is optimized to obtain the optimal network parameters and output $\widetilde{u}(\mathbf{x}_{test},t_{test})$. In this setting, the problem of solving Eq. (4) can be described as follows:

$$\begin{aligned}
&by\ \ [u_{NN}(\mathbf{x}_0,t_n;\mu),...,u_{NN}(\mathbf{x}_N,t_n;\mu)] \to [c(\mathbf{x}_0,t_n),...,c(\mathbf{x}_N,t_n)],\\
&\quad u_{NN}(\mathbf{x},t;\mu) \to \widetilde{u}(\mathbf{x},t),\\
&construct\ \ MSE_{equ}\ by\ \widetilde{u}(\mathbf{x},t)=\sum_{j=0}^{N} c(\mathbf{x}_k,t)\phi_i(\mathbf{x})\rho(\mathbf{x}),\\
&\qquad MSE_{IVC}\ by\ u_{NN}(\mathbf{x},t),\\
&\qquad \mathrm{L}(\mu)=MSE_{equ}+W_I\times MSE_{IVC},\\
&by\ training\ \ \{\mu\}=\underbrace{argmin}_{\mu}(\mathrm{L}(\mu))\ obtain\ the\ optimal\ \mu,\\
&such\ that\ \ ||\widetilde{u}(\mathbf{x}_{test},t_{test})-u(\mathbf{x}_{test},t_{test})||<\varsigma,
\end{aligned} \tag{20}$$

where ς is the desired error accuracy.

3.2 Inverse problem by fPINNs

In this subsection, we regard the inverse problem as an optimization problem and use fPINNs to estimate the order of fractional derivative α, the mobility constant γ, and the coefficient ε.

The unknown parameter vector is denoted by $\xi=(\alpha,\gamma,\varepsilon)\in\Theta$, where $\Theta=[Lb_\alpha,Ub_\alpha]\times[Lb_\gamma,Ub_\gamma]\times[Lb_\varepsilon,Ub_\varepsilon]$. Considering the inverse problem of the form

$$\begin{aligned}
&D_t^\alpha u(\mathbf{x},t)+L_{\gamma,\varepsilon}\{u(\mathbf{x},t)\}=f(\mathbf{x},t),\ \ (\mathbf{x},t)\in\Omega\times(0,T],\\
&u(\mathbf{x},0)=g(\mathbf{x}),\ \ \mathbf{x}\in\Omega,\\
&u(\mathbf{x},t)=h(\mathbf{x},t),\ \ (\mathbf{x},t)\in\partial\Omega\times(0,T],
\end{aligned} \tag{21}$$

where $L_{\gamma,\varepsilon}\{u(\mathbf{x},t)\}=\gamma\Delta(-\varepsilon^2\Delta u(\mathbf{x},t)-u(\mathbf{x},t)+u^3(\mathbf{x},t))$, and the fPDE parameters $\alpha,\gamma,\varepsilon$ and the solution $u(\cdot)$ are to be recovered from the initial boundary conditions. As the inverse problem complicates the loss function by increasing the dimension of the parameter space, we use the same treatment as in [16] to reduce the complexity. Specifically, we consider the following transformations for $\{\alpha,\gamma,\varepsilon\}$:

$$\begin{aligned}
&\alpha=0.5tanh(\alpha_0)+0.5\in(0,1),\\
&\gamma=exp(\gamma_0)\in(0,+\infty),\\
&\varepsilon=exp(\varepsilon_0)\in(0,+\infty),
\end{aligned} \tag{22}$$

where $\{\alpha_0,\gamma_0,\varepsilon_0\}$ are parameters in the transformed parameter space.

Instead of discretize the time-fractional derivative and the space direction at the same time as in the forward problem, we only discretize the time-fractional derivative by the FBDF. This is because in the inverse problem the parameters $\{\alpha_0,\gamma_0,\varepsilon_0\}$ of the model are jointly optimized with the parameters μ of the network, and we do not set the penalty term on the parameter directly in the loss function. The NNs can optimize the loss function to its minimum only when the relationships between the model parameters, the model equation, and the label data set are established. If the space direction also is discretized, as in the forward problem, this will complicate the relationship and may result in erroneous network information; this in turn may lead to the model parameters being optimized in the wrong direction, meaning we would be unable to obtain the optimal result.

Let $\widetilde{u}(x,t)=u_{NN}(\mathbf{x},t)\rho(\mathbf{x})$ be the approximated solution, where $\rho(\mathbf{x})$ is pre-selected to automatically satisfy the boundary condition. Moreover, the loss function is expressed in two parts. The loss function L_{INV} for the inverse problem can be written as

$$\mathrm{L}_{INV}(\mu,\alpha_0,\gamma_0,\varepsilon_0)=MSE_{equ}(\mu,\alpha_0,\gamma_0,\varepsilon_0)+W_I\times MSE_{IVC}(\mu), \tag{23}$$

with

$$\begin{aligned}
&MSE_{equ}(\mu,\alpha_0,\gamma_0,\varepsilon_0)\\
&=\frac{1}{N_{equ}}\sum_{m=1}^{N_{equ}}\Big(\frac{1}{\tau^{0.5tanh(\alpha_0)+0.5}}\sum_{s=0}^{n} w_{n-s}^{(0.5tanh(\alpha_0)+0.5)}\rho(\mathbf{x}_m)\\
&(u_{NN}(\mathbf{x}_m,t_s)-u_{NN}(\mathbf{x}_m,0))+L_{\gamma_0,\varepsilon_0}(u_{NN}(\mathbf{x}_m,t_n)\rho(\mathbf{x}_m))\\
&-f(\mathbf{x}_m,t_n)\Big)^2,
\end{aligned} \tag{24}$$

$$MSE_{IVC}(\mu)=\frac{1}{N_{IVC}}\sum_{m=1}^{N_{IVC}}(u_{NN}(\mathbf{x}_m,0)-g(\mathbf{x}_m))^2. \tag{25}$$

Under this construction of the loss function, we can control the weight W_I to intervene the influence of the model and the

label data set on the parameters estimation to obtain a better result. There are several ways to set the value of the weight W_I; these can generally be categorized as adaptive and artificial methods. Here, we use an artificial method. This is because although an adaptive method could be used to select the appropriate weight value based on the physical information of the NNs, this will increase the complexity of the network and affect the optimization of the parameters.

A difference between the inverse problem and the forward problem is the optimization object of the NNs. Parameters in the model are optimized together with the hyperparameters μ of the network. After constructing the loss function, we can minimize it using an appropriate optimization algorithm, such as the gradient descent method or stochastic gradient descent method (SGD). The SGD method is very popular in the field of machine learning (see [33] for more details). In this work, we use a version of the SGD method as our optimization algorithm, namely, the Adam optimizer. By optimizing the loss function, the approximated values of the fractional order α, the mobility constant γ, the coefficient ε, the hyperparameters μ, and the approximated solution of $\widetilde{u}(x,t)$ are obtained. However, as we do not have any real model parameters by which to judge the network's training parameters, we need to evaluate them indirectly. We compare the difference between the approximated solution $\widetilde{u}$ and the exact solution u at test points $(\mathbf{x}_{test}, t_{test})$ to confirm the accuracy of the estimated parameters when

$$\frac{||\widetilde{u}(\mathbf{x}_{test}, t_{test}) - u(\mathbf{x}_{test}, t_{test})||^2}{||u(\mathbf{x}_{test}, t_{test})||^2} < \varsigma, \tag{26}$$

where ς is our required error accuracy. When [26] is satisfied, we assume that the network obtains the desired model parameters, and stop training. The process of the inverse problem is summarized in Algorithm 1.

Algorithm 1 fPINN algorithm for the inverse problem

1: Build a training set in the inner space and the initial condition
Training data: $\{(\mathbf{x}_f^i, t^i)\}_{i=1}^{N_{equ}}$;
Initial condition points: $\{(\mathbf{x}_{IC}^i, t^i)\}_{i=1}^{N_{IVC}}$;
2: Specify the test points: $\{(\mathbf{x}_{test}^i, t_{test}^i)\}_{i=1}^{N_{test}}$;
3: Set the desired error accuracy ς and the maximum number of iterations N, and transform the parameters;
4: **while** $(\frac{\sum_{i=1}^{N_{test}} ||\widetilde{u}(\mathbf{x}_{test}^i, t_{test}^i) - u(\mathbf{x}_{test}^i, t_{test}^i)||^2}{\sum_{i=1}^{N_{test}} ||u(\mathbf{x}_{test}^i, t_{test}^i)||^2} < \varsigma$ or niter $< N)$ **do**
5: Construct the NN with initialization network parameters μ and model parameters α_0, ξ_0;
6: Specify the loss function L_{INV} as $\mathrm{L}_{INV}(\mu, \alpha_0, \gamma_0, \varepsilon_0) = MSE_{equ}(\mathbf{x}_{equ}, t; \mu, \alpha_0, \gamma_0, \varepsilon_0) + W_I \times MSE_{IVC}(\mathbf{x}_{IVC}, t; \mu)$;
7: Find the best parameters for minimizing the loss function using the Adam optimizer
$\{\mu, \alpha_0, \gamma_0, \varepsilon_0\} = \underbrace{argmin}_{\mu, \alpha_0, \gamma_0, \varepsilon_0, w, b} (\mathrm{L}_{INV})$;
8: Give the approximated solution $\widetilde{u}$ at test points $(\mathbf{x}_{test}^i, t_{test}^i)$.
9: **end while**

4 Results and Discussion

In this section, several numerical examples are considered to verify the effectiveness of fPINNs for solving the forward and inverse problems of time-fractional phase field models.

To evaluate the performance of our method, we consider the point-wise error $\frac{\{|u(x_i,t_i)-\widetilde{u}(x_i,t_i)|^2\}^{\frac{1}{2}}}{\{|u(x_i,t_i)|^2\}^{\frac{1}{2}}}$ and L^2 relative error at each time step as

$$\frac{\{\Sigma_i |u(x_i,t_i) - \widetilde{u}(x_i,t_i)|^2\}^{\frac{1}{2}}}{\{\Sigma_i |u(x_i,t_i)|^2\}^{\frac{1}{2}}}, \tag{27}$$

where $\widetilde{u}$ denotes the approximated solution, u is the exact solution, and (x_i,t_i) denotes the i-th test point. We select 1000 space points from the Sobol sequence for each example, and divide them into training points and test points according to a ratio of $7:3$.

We employe Tensorflow for programming to take advantage of its automatic differentiation capability, and used the Adam and L-BFGS algorithms to optimize the loss function. We also use Xavier initialization and choose the Sobol sequence as the dataset. For the forward problem, the learning rate, the number of hidden layers, the number of neurons, the number of iterations, and the active function are fixed to 1×10^{-3}, 1, 6, 10^4, and $\tanh(x)$, respectively. For the inverse problem, we use the same code as that used for the forward problem but with four hidden layers with 20 neurons.

4.1 Time-fractional AC equation

Example 1. Considering the following 1D time-fractional AC equation

$$\begin{aligned} &D_t^\alpha u(x,t) = \varepsilon^2 \Delta u + u - u^3 + s(x,t), \ \ (x,t) \in (0,1)\times(0,T], \\ &u(x,0) = g(x), \ \ x \in (0,1), \\ &u(0,t) = u(1,t), \ \ t \in (0,T]. \end{aligned} \tag{28}$$

Here, we fix $\alpha = 0.8$ and $\varepsilon = 1$, with $\alpha_0 = 0.7$ and $\varepsilon_0 = 0$.

Case 1: Considering the smooth fabricated solution $(t^2+1)\sin(2\pi x)$ and the forcing term

$$\begin{aligned} s(x,t) = &\frac{\Gamma(3)}{\Gamma(2.2)} t^{1.2}\sin(2\pi x) + 4\pi^2(t^2+1)\sin(2\pi x) \\ &- (t^2+1)\sin(2\pi x) + ((t^2+1)\sin(2\pi x))^3, \end{aligned} \tag{29}$$

the approximated solution can be constructed as

$$\widetilde{u}(x,t) = u_{NN}(x,t) = \sum_{k=0}^{N} u_{NN}(x_k,t)\phi_k(x)\sin(2\pi x). \tag{30}$$

The loss function can be written as

$$L(\mu) = MSE_{equ} + W_I \times MSE_{IVC}, \tag{31}$$

where

$$MSE_{equ} = \frac{1}{N_{equ}} \sum_{k=1}^{N_{equ}} \Big(\frac{1}{\tau^{\alpha}} \sum_{j=0}^{n} w_{n-j}^{(\alpha)} \Big(\sum_{i=0}^{N} u_{NN}(x_i,t_j)\phi_i(x_k) \sin(2\pi x_k) - u_{NN}(x_i,0)\phi_i(x_k)\sin(2\pi x_k) \Big) - \varepsilon^2 \Delta \Big(\sum_{i=0}^{N} u_{NN}(x_i,t_n) \phi_i(x_k)\sin(2\pi x_k) \Big) - \sum_{i=0}^{N} u_{NN}(x_i,t_n)\phi_i(x_k)\sin(2\pi x_k) + \Big(\sum_{i=0}^{N} u_{NN}(x_i,t_n)\phi_i(x_k)\sin(2\pi x_k) \Big)^3 - s(x_k,t_n) \Big)^2, \tag{32}$$

and

$$MSE_{IVC} = \frac{1}{N_{IVC}} \sum_{j=1}^{N_{IVC}} (u_{NN}(x_j,0) - g(x_j))^2. \tag{33}$$

Then, we can find an approximated solution by optimizing

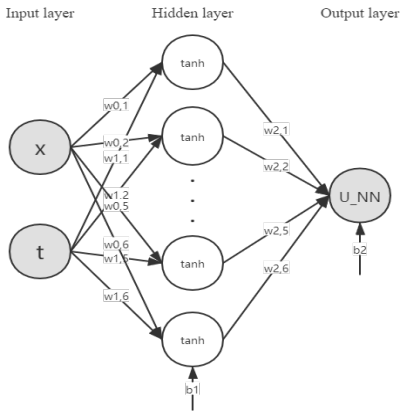

Fig. 2: NN with one hidden layer, six neurons, and *tanh* activation function.

the loss function such that $u(x,t) = \widetilde{u}(x,t)$. Considering a shallow neural network of one hidden layer with six neurons and a tanh activation function. There are 14 parameters, including 12 weights and two biases, which need to be optimized by the NN; see Fig. 2. For this shallow network, we can easily write an expression for c_j:

$$\begin{aligned} c_j &= u_{NN}(x_j,t) \\ &= \sum_{k=1}^{6} w_{2,k} \tanh(w_{0,k}x_j + w_{1,k}t) + b_1) + b_2, \end{aligned} \tag{34}$$

which offers the possibility of numerical analysis. In addition, because different NN structures give different results for different models, we study the effect of the NN approximation error by altering the depth and width of the NN to optimize error accuracy. The results are shown in Figures 3–5. For each experiment, the code runs 10 times.

Figure 3 shows the point-wise error with different learning rates and different numbers of approximate points and basis functions. As shown in Figure 3 (a), for a fixed number of approximate points and basis functions, similar convergence accuracy is achieved with different learning rates. This is because the NN is simple in structure and thus requires few optimization parameters; therefore, it does not need a large number of iterations to achieve the optimal results. When the network optimization reaches its optimal value, further iterations will only cause it to fluctuate around the optimal result and may even lead to over-fitting; they will not produce real change. That is, our NN does not require a lower learning rate or larger number of iterations. Next, we change the number of approximate points and basis functions, and observe the resulting changes in point-wise error. As shown in Figure 3 (b), we first consider the case where the number of basis functions and approximate points are the same. When the number of approximate points and basis functions are changed from 6 to 8, the point-wise error changes from 10^{-4} to 10^{-5}; these results indicate that using fewer basis functions and approximate points would be detrimental to the network's prediction ability, whereas increasing the number can improve prediction ability. However, when the number of basis functions and approximate points reach certain values, further increases will not result in higher accuracy. This limitation is based on the NN approach taken. As shown in Figure 3 (c), the best point-wise error is 10^{-6}, which is obtained by choosing 10 approximations and 12 basis functions. The variation in the loss function is also shown, in Figure 3 (d) and (e).

In the loss function, the penalty operator W_I is added before the initial condition penalty term, and the coefficient before the information penalty term of the equation is 1. In this construction, when W_I is greater than 1, the network will be more inclined to fit the initial boundary conditions; here, the function of the equation term is to correct the initial condition but not the other time point. Therefore, finding the optimal match ratio by adjusting the penalty operator W_I in the loss function is another aspect to consider. We present the results of our investigation in Figure 4. As shown in the figure, W_I acts as a correction; as W_I increases, the error curve becomes wider, indicating that the global error is decreasing and the network optimization tends to fit the initial condition. Although the minimum point-wise error becomes larger, the global error decreases. Moreover, as W_I increases, the error first increase and then decrease. This means that there is an optimal value of W_I under this condition.

Figure 5 shows the influence of different methods of selecting collocation points on the point-wise error, that is, whether uniform points or random points are taken. Taking points uniformly means dividing the intervals between

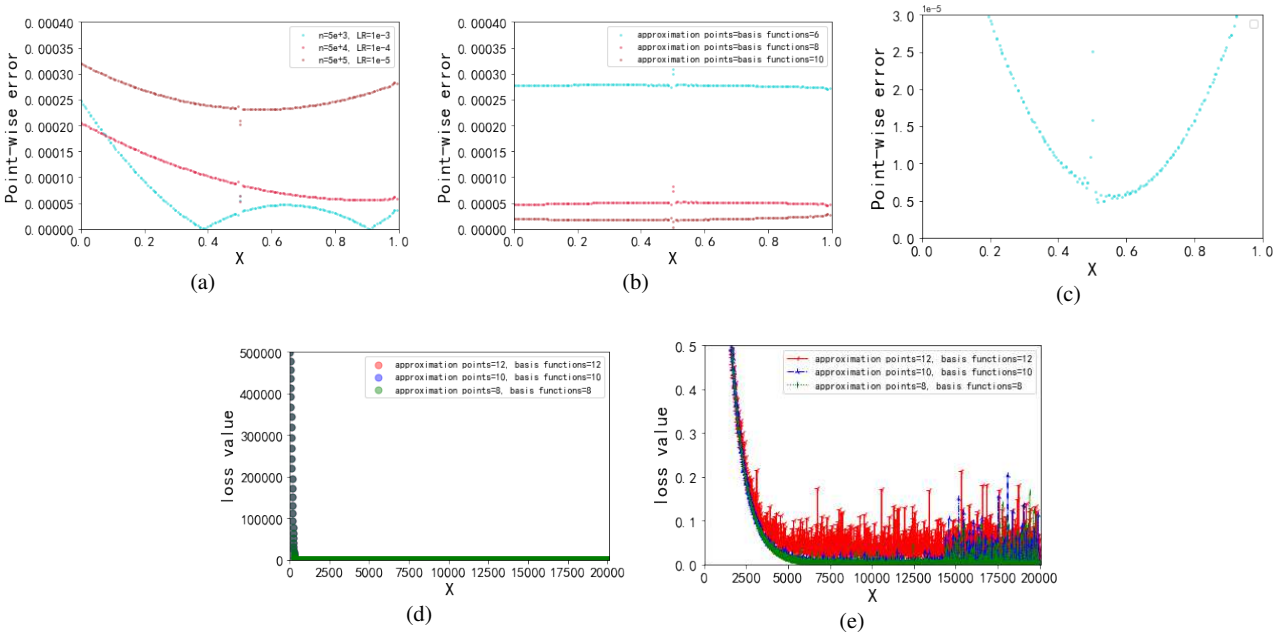

Fig. 3: 1D time-fractional AC equation with the smooth fabricated solution $(1+t^2)\sin(2\pi x)$: effects of numbers of iterations, approximate points, and basis functions on optimization performance. (a) Point-wise error with different numbers of iterations. (b) Point-wise error of varying the same number of approximate points and basis functions. (c) Point-wise error of predicted and exact solutions with 10 approximate points and 12 basis functions. (d) Loss value. (e) Detail of (d).

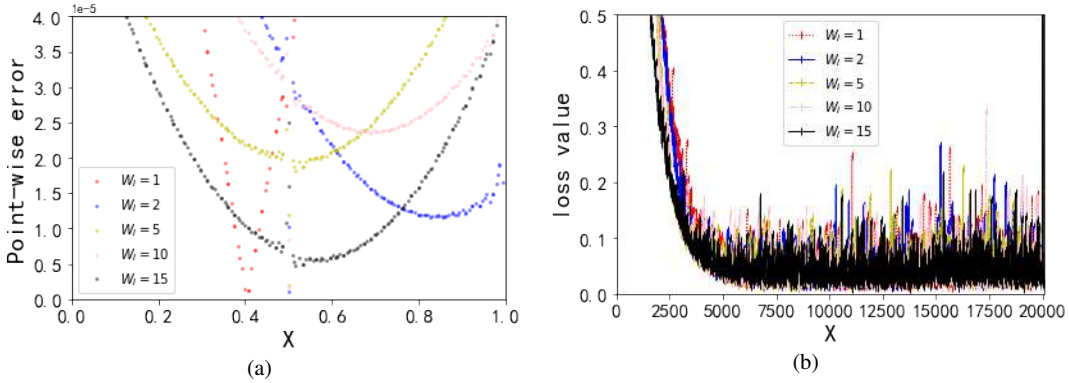

Fig. 4: 1D time-fractional AC equation with smooth fabricated solution: effects of penalty parameter W_I in the loss function. (a) Point-wise error of different penalty parameters W_I. (b) Loss value.

points equally according to the number of points needed. Taking points at random means selecting random values in the interval. We find no great difference in the convergence accuracy between the two cases. This may be because in a relatively small area, random and uniform points are used to cover the whole area do not differ much, resulting in the same convergence accuracy. However, in a larger space, taking random points may be advantageous. Moreover, the selection of collocation points must be performed in the whole space, and the points must cover the whole space; otherwise, the convergence accuracy will be greatly reduced.

Figure 6 shows the relationship of the number of neurons and the number of approximate points with the point-wise error. The relationship of the number of basis functions and the number of approximate points with the point-wise error is presented in Figure 7. We also consider the effect of the structure of the network by changing the depth and width of the NN; Figure 8 displays the L^2 relative error corresponding to different depths and widths. In each case, the code runs 10 times.

As shown in Figure 6 (a), given a fixed number of approximate points and varying number of neurons, when the

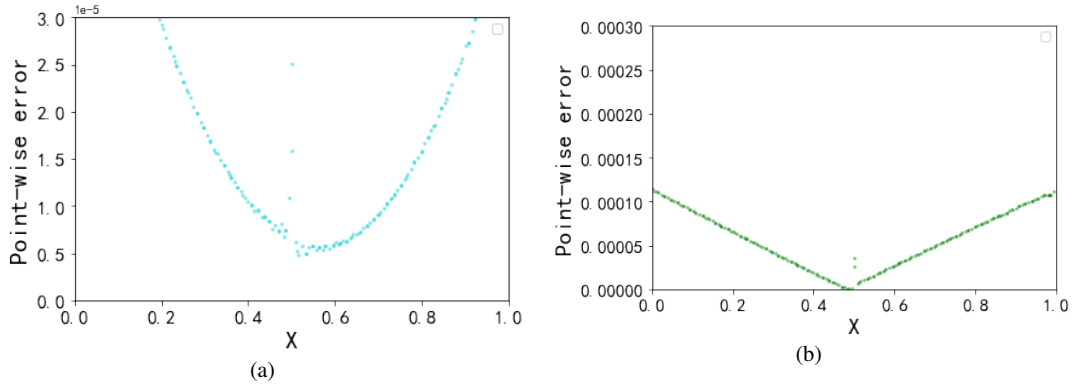

Fig. 5: 1D time-fractional AC equation with smooth solution: effects of collocation points selection method. (a) Point-wise error with uniformly selected points. (b) Point-wise error with randomly selected points.

number of neurons is small, the point-wise error is relatively large. The reason is the small number of neurons means that the generalization ability of the network is weak. Thus, we can not achieve a high convergence accuracy. With the increase of number of neurons, the generalization ability of the network is enhanced; thus, the convergence accuracy improved and optimization is achieved gradually. After the network has been optimized, we continue to increase the number of neurons. Although the generalization ability of the network is still increasing, the number of parameters that need to be optimized increases exponentially, which makes it difficult for the network to reach its global optimum and easy for it to fall into local minima. Therefore, increasing the number of neurons does not always improve the generalization ability of a network; there are different optimal numbers of neurons in different situations. As shown in Figure 6 (c), the number of different approximate points has similar effects on the point-wise error. As the number increases from 5 to 40, the error changes from 10^{-4} to 10^{-5}. This indicates that there is an optimal number of approximate points under this condition.

Figure 7 (a) and (b) show the results for different numbers of basis functions, from 4 to 20, with fixed number of approximate points and neurons. When the number of basis functions is greater than 10, the point-wise error begins to increase, from 10^{-5} to 10^{-2}. This result shows that the number of basis functions has a great influence on the point-wise error. When more basis functions are selected, more parameters need to be optimized, which means the optimization can easily fall into local minima and we cannot obtain a good generalization. Moreover, when the number of basis functions is less than 10, the point-wise error is relatively stable at a value of 10^{-5}. Further research is needed to determine the optimal number of basis functions. Figure 7 (c) describes the effect of the number of approximate points on the point-wise error for a fixed number of basis functions. As the number of approximate points increases, the point-wise error tends to decrease at first and then increase. However, the change in the point-wise error is far smaller than that caused by the increase of the number of basis functions.

From Figure 8 (a), we can find that the depth of the NN has an obvious effect on the convergence accuracy when the width is 6. Moreover, the depth has a greater impact on the L^2 relative error than the width. The difference is not great, but larger depth leads to worse accuracy. In this example, the L^2 relative error is relatively stable, with only a few oscillations. This indicates that there is an optimal pairing relationship between depth and width, which is also closely related to other conditions, including learning rate, W_I, and the number of training points, approximations, basis functions, and iterations.

Case 2: We consider the non-smooth fabricated solution $u(x,t) = (1+t^2)\sin(2\pi x^{1.2})$ and the force term

$$\begin{aligned} &s(x,t) \\ &= \frac{\Gamma(3)}{\Gamma(2.2)} t^{1.2}\sin(2\pi x^{1.2}) + 5.76\pi^2 x^{0.4}(1+t^2)\sin(2\pi x^{1.2}) \\ &- 0.48\pi x^{-0.8}(1+t^2)\cos(2\pi x^{1.2}) - (1+t^2)\sin(2\pi x^{1.2}) \\ &+ ((1+t^2)\sin(2\pi x^{1.2}))^3. \end{aligned} \tag{35}$$

Unlike case 1, we consider another form of the hypothetical solution; this form can also be used in the previous example. When we do not know whether the exact solution is smooth, we need to add an unknown parameter to our hypothetical solution, i.e. $u(x,t) = u(x,t)\sin(2\pi c x^{\beta})$. The smoothness of the solution is determined by the expression of the space vector $\mathbf{x}$ in the solution. The expression of the space vector $\mathbf{x}$ inside the network is determined by the network parameter μ, so no special treatment is required. The space vector $\mathbf{x}$ outside the network should be expressed in a general form to satisfy all possible solutions. Thus, we add a coefficient c and a power β to $\mathbf{x}$. Obviously, this increases the difficulty of

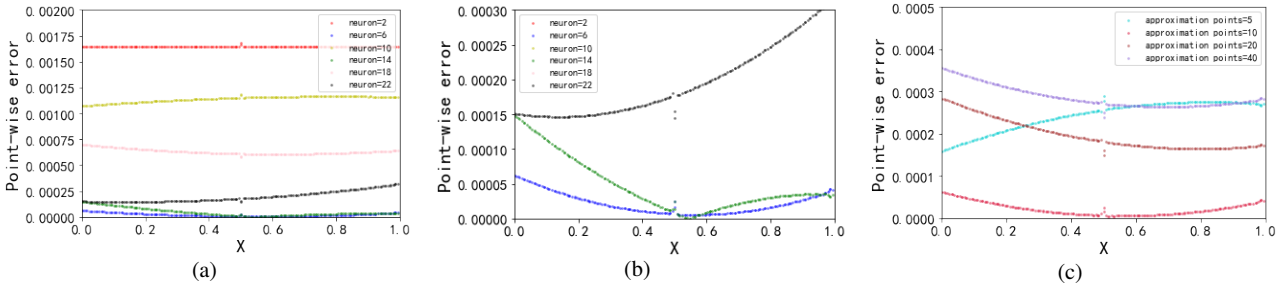

Fig. 6: 1D time-fractional AC equation with smooth fabricated solution: effects of numbers of neurons and approximate points on point-wise error. (a) Point-wise error versus neuron number N for a fixed number of approximate points. (b) Detail of (a). (c) Point-wise error versus approximate points for a fixed number of neurons.

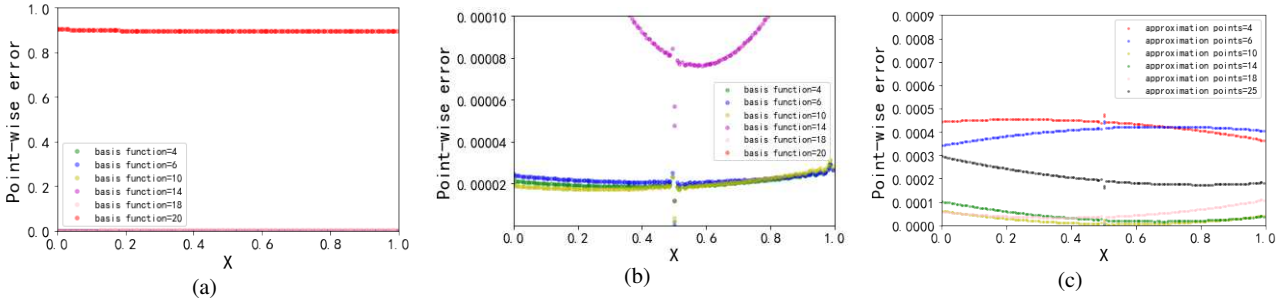

Fig. 7: 1D time-fractional AC equation with smooth fabricated solution: effects of numbers of basis functions and approximate points on point-wise error. (a) Point-wise error versus number of basis function ϕ_i for a fixed number of approximate points. (b) Detail of (a). (c) Point-wise error versus number of approximate points for a fixed number of basis functions ϕ_i.

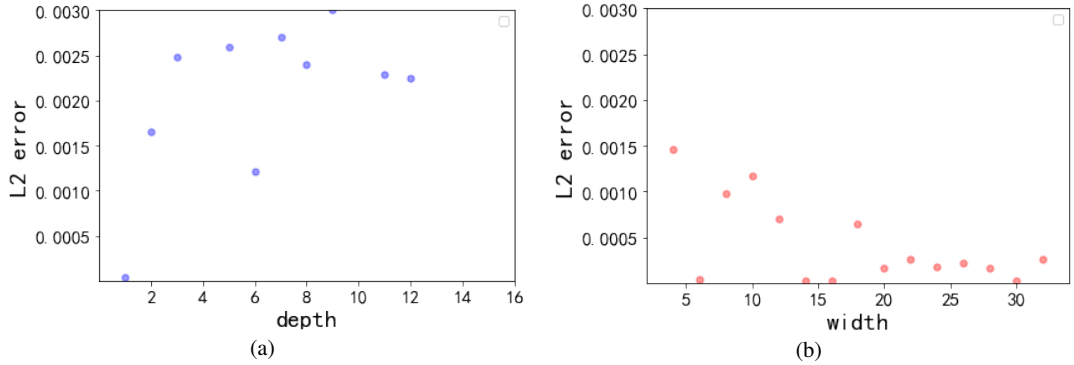

Fig. 8: 1D time-fractional AC equation with smooth fabricated solution: effects of NN structure on the error. (a) Point-wise error versus depth for width 6. (b) Point-wise error versus width for depth 1.

solving such problems by traditional methods, but they are very easily solved using NNs. Figure 9 gives the results of solving case 1 and case 2 problems with the above structure.

The results shown in Figure 9 indicate that the assumed solutions construct in this new way can be used to obtain equations with high convergence accuracy in the case of non-smooth solutions. We show that they can also deal with the case where the fabricated solution is smooth. The corrected results are presented in Figure 9 (c) and (d). However, it is necessary to find the optimal values of certain parameters, including the number of neurons, the number of basis functions, and the number of approximate points. Here, we take the values of these parameters to be 15, 40 and 12, respectively, and obtain a point-wise error of 10^{-4}.

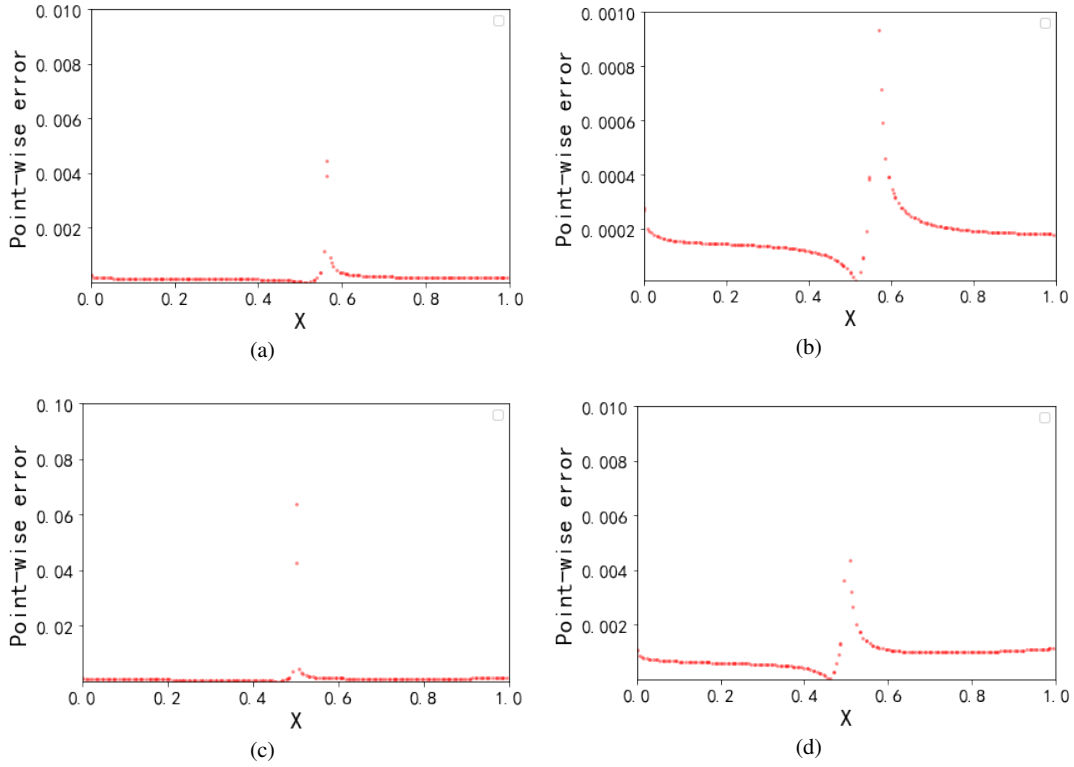

Fig. 9: 1D time-fractional AC equation. (a) Point-wise error of the non-smooth fabricated solution $u(x,t) = (1+t^2)\sin(2\pi x^{1.2})$. (b) Detail of (a). (c) Point-wise error of the smooth fabricated solution $u(x,t) = (1+t^2)\sin(2\pi x)$. (d) Detail of (c).

Example 2. Here, considering the following 2D time-fractional AC equation:

$$\begin{aligned} &D_t^{\alpha} u(x,y,t) = \varepsilon^2 \Delta u(x,y,t) + u(x,y,t) - u^3(x,y,t) + s(x,y,t), \\ &(x,y,t) \in (0,1)^2 \times (0,T], \\ &u(x,y,0) = g(x,y),\ (x,y) \in (0,1)^2. \end{aligned} \tag{36}$$

For consistency, we use a periodic boundary condition and fixed $\alpha = 0.8$, $\varepsilon = 1$, with $\alpha_0 = 0.7$ and $\varepsilon_0 = 0$. With the fabricated solution $(1+t^2)\sin(2\pi x)\sin(2\pi y)$, the force term $s(x,y,t)$ can be obtained:

$$\begin{aligned} &\frac{\Gamma(3)}{\Gamma(2.2)} t^{1.2} \sin(2\pi x)\sin(2\pi y) + 8\pi^2 (1+t^2)\sin(2\pi x)\sin(2\pi y) \\ &- (1+t^2)\sin(2\pi x)\sin(2\pi y) + ((1+t^2)\sin(2\pi x)\sin(2\pi y))^3. \end{aligned} \tag{37}$$

Similar to the 1D case, we construct a NN and define the loss function as $L(\mu) = MSE_{equ} + W_I \times MSE_{IVC}$. To define the same residual of the network, we consider the 2D Lagrange interpolation polynomials, which have same properties as in the 1D form.

With the approximated solution

$$\begin{aligned} \widetilde{u}(x,y,t) &= u_{NN}(x,y,t) \\ &= \sum_{i=0}^{N} u_{NN}(x_i,y_i,t)\phi_i(x,y)\sin(2\pi x)\sin(2\pi y), \end{aligned} \tag{38}$$

we can rewrite the loss function as

$$\mathrm{L}(\mu) = MSE_{equ} + W_I \times MSE_{IVC}, \tag{39}$$

where

$$\begin{aligned} MSE_{equ} &= \frac{1}{N_{equ}} \sum_{k=1}^{N_{equ}} \Big(\frac{1}{\tau^{\alpha}} \sum_{j=0}^{n} w_{n-j}^{(\alpha)} \sum_{i=0}^{N} u(x_i,y_i,t_j)\phi_i(x_k,y_k) \\ &\sin(2\pi x_k)\sin(2\pi y_k) - \varepsilon^2 \Delta \Big(\sum_{i=0}^{N} u(x_i,y_i,t_j)\phi_i(x_k,y_k)\sin(2\pi x_k) \\ &\sin(2\pi y_k)\Big) - \sum_{i=0}^{N} u(x_i,y_i,t_j)\phi_i(x_k,y_k)\sin(2\pi x_k)\sin(2\pi y_k) + \\ &\Big(\sum_{i=0}^{N} u(x_i,y_i,t_j)\phi_i(x_k,y_k)\sin(2\pi x_k)\sin(2\pi y_k)\Big)^3 - s(x_k,y_k,t_n)\Big)^2, \end{aligned} \tag{40}$$

$$MSE_{IVC} = \frac{1}{N_{IVC}} \sum_{j=1}^{N_{IVC}} (u_{NN}(x_j, y_j, 0) - g(x_j, y_j))^2. \tag{41}$$

We can extrapolate in this way to solve 3D problems.

Table 1 shows the structure and the corresponding parameters of our NN. The exact solution, prediction, pointwise error, and the value of loss function are shown in Figure 10. As shown in Figure 10, the 2D time-fractional AC equation can still obtain a good convergence accuracy, indicating that our method can effectively deal with multidimensional time-fractional phase field models.

Table 1: 2D time-fractional AC equation: Neural network, optimizer and the parameter of fPINNs based on spectral collocation method.

Approximation points	10
Basis function	12
τ	15
Hidden layer	1
Neurons	6
Activation function	tanh
Optimizer	Adam + L-BFGS
Learning rate	10^{-3}

We construct the NN by selecting six collocation points in x and y directions, respectively. Our method can accurately capture the exact solution, demonstrating its effectiveness for solving the forward problem of 2D time-fractional AC equations. To solve higher-dimensional problems, the same code could be used, just changing the form of the basis function. In addition, we can analyze our error sources from the following perspectives: sample selection error, basis function selection error, collocation point selection error, discrete approximation error, NN approximation error, and optimization error.

Table 2 shows the performance comparison between fPINNs based on spectral collocation method and a high-efficiency second-order numerical method [24] using different $1/\tau$. In [24], the FBDF is used to approximate the time-fractional derivative and the extended scalar auxiliary variable method is used to deal with the nonlinear terms. As shown in Table 2, we can observe that our method can achieve higher error accuracy under larger time division.

Finally, we show the results of inverse problems for 1D and 2D time-fractional AC equations. Better error accuracy is obtained by setting the appropriate NN structure. Table 3 gives the estimated results for the two parameters in 1D and 2D time-fractional AC equations with different number of iterations. As the number of iterations increases, the estimation becomes more accurate. The initial values of the parameters are $\alpha^0 = 0.6$ and $\varepsilon^0 = 0.1$. The iteration process for the two parameters generated by the fPINNs is given in Fig. 11. Across the whole iteration process, the number of iterations and the convergence of these two parameters are both good; their estimated values are $\widehat{\alpha} = 0.73208$ and $\widehat{\varepsilon} = 0.00058$. These results imply that the fPINNs are effective for solving the inverse problem of time-fractional AC equations.

4.2 Time-fractional CH equation

Example 3. This example considers the ability of our approach to handle the time-fractional CH equation, which has higher-order terms. We start with the 1D time-fractional CH equation with periodic boundary conditions:

$$\begin{aligned} &D_t^{\alpha} u(x,t) - \gamma \partial_x^2 (\varepsilon \partial_x^2 u(x,t) - u^3(x,t) + u(x,t)) \\ &= f(x,t), \ \ (x,t) \in (0,1) \times (0,T], \\ &u(x,0) = g(x), \ \ x \in (0,1), \\ &u(0,t) = u(1,t), \ \ t \in (0,T], \end{aligned} \tag{42}$$

and fix $\alpha = 0.8$, $\gamma = 1$, and $\varepsilon = 1$, with $\alpha_0 = 0.7$, $\gamma_0 = 0$, and $\varepsilon_0 = 0$. This equation has one more higher-order differential term than the time-fractional AC equation. The fabricated solution $u(x,t) = (1+t^2)\sin(2\pi x)$ is considered, which is smooth and has a fourth-order differential in the direction of space.

The corresponding forcing term is

$$\begin{aligned} f(x,t) = &\frac{\Gamma(3)}{\Gamma(3-\alpha)} t^{1.2} \sin(2\pi x) - 24\pi^2((1+t^2)\sin(2\pi x)) \\ &((1+t^2)\cos(2\pi x))^2 - 12\pi^2((1+t^2)\sin(2\pi x))^3 + 16\pi^4(1 \\ &+t^2)\sin(2\pi x)\sin(2\pi x) + 4\pi^2(1+t^2)\sin(2\pi x). \end{aligned}$$

Then, with the approximated solution as in Eq. (30), the loss function can be written as follows:

$$L(\mu) = MSE_{equ} + W_I \times MSE_{IVC}, \tag{43}$$

where

$$\begin{aligned} MSE_{equ} = &\frac{1}{N_{equ}} \sum_{k=1}^{N_{equ}} \Big(\frac{1}{\tau^{\alpha}} \sum_{j=0}^{n} w_{n-j}^{(\alpha)} \Big(\sum_{i=0}^{N} u_{NN}(x_i, t_j) \phi_i(x_k) \sin(2\pi x_k) \\ &- u_{NN}(x_i, 0) \phi_i(x_k) \sin(2\pi x_k) \Big) - \partial_x^4 \Big(\sum_{i=0}^{N} u_{NN}(x_i, t_n) \\ &\phi_i(x_k) \sin(2\pi x_k) \Big) - \partial_x^2 \Big(\sum_{i=0}^{N} u_{NN}(x_i, t_n) \phi_i(x_k) \sin(2\pi x_k) \Big)^3 \\ &+ \partial_x^2 \Big(\sum_{i=0}^{N} u_{NN}(x_i, t_n) \phi_i(x_k) \sin(2\pi x_k) \Big) - f(x_k, t_n) \Big)^2, \end{aligned}$$

and

$$MSE_{IVC} = \frac{1}{N_{IVC}} \sum_{j=1}^{N_{IVC}} (u_{NN}(x_j, 0) - g(x_j))^2. \tag{44}$$

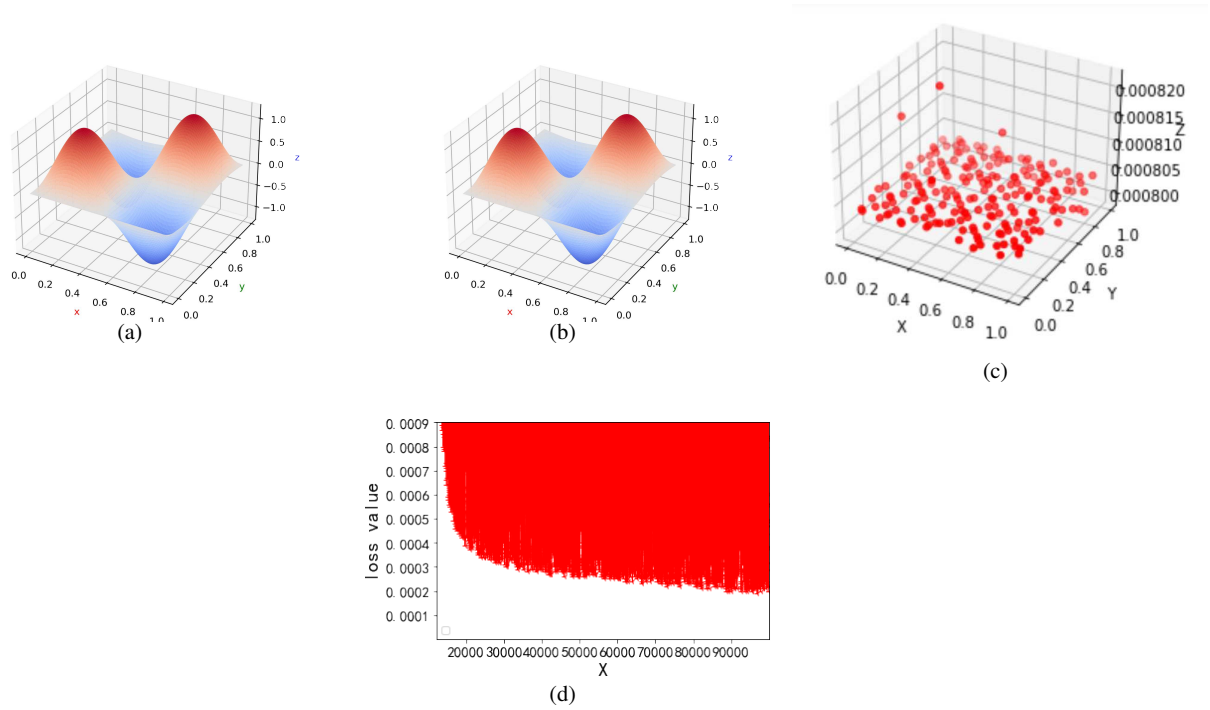

Fig. 10: 2D time-fractional AC equation with smooth solution $(1+t^2)\sin(2\pi x)\sin(2\pi y)$. (a) Exact solution. (b) Approximation of fPINNs based on spectral collocation method. (c) Point-wise error. (d) Loss value.

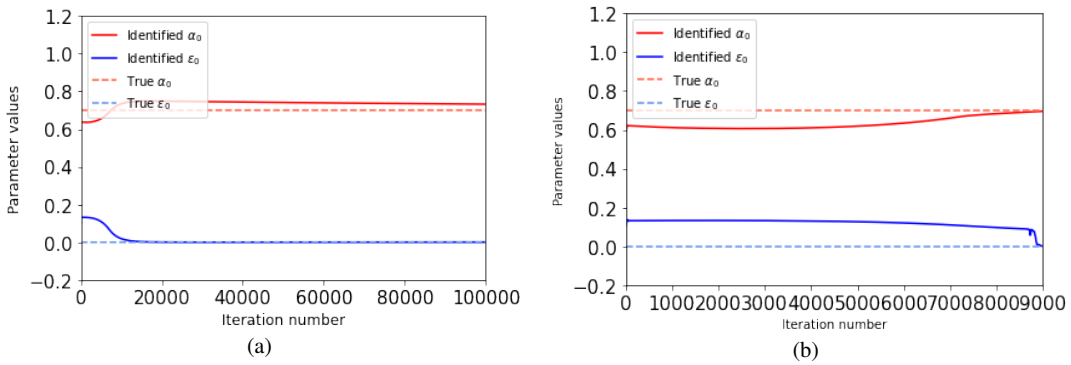

Fig. 11: Inverse problem with smooth fabricated solution. (a) Parameter evolution of 1D time-fractional AC equation. (b) Parameter evolution of 2D time-fractional AC equation.

We use the Xavier initialization, the Adam algorithm and the L-BFGS method for optimization, the results are shown in Figures 11–15.

In Figure 12 (a), the point-wise error is obviously better when the learning rate is low compared with when it is high. This is due to the existence of a high-order differential term in time-fractional CH model, which is relatively complex in structure and can easily interfere with the ability of the network to reach an optimal value. Thus, we need to adjust the learning rate and the number of iterations to achieve the optimal value. Furthermore, for the time-fractional CH model, the convergence accuracy reaches 10^{-6}. Figure 12 (c) shows the effects of changing the same number of approximate points and basis functions on point-wise error; these results indicate that the number of approximate points and basis functions in the time-fractional CH model also has an important effect on the point-wise error.

Next, the effect of the number of neurons and the number of approximate points on the point-wise error are considered. As shown in Figure 13 (a), as the number of neurons increases from 2 to 6, the point-wise error decreases from 10^{-4} to 10^{-6}. With the increase of the number of neuron-

Table 2: Comparison of performance between fPINNs based on spectral collocation method and numerical method in [24] on time-fractional AC equation with periodic boundary condition for $\alpha = 0.8$.

$1/\tau$	fPINNs based on spectral collocation method	numerical method in [24]
	L^2 error	L^2 error
10	8.12058e-04	1.7771e-02
20	7.40948e-04	5.3519e-03
40	7.12075e-04	1.4220e-03

Table 3: Identified parameters and the L^2 relative error of u for inverse problems of time-fractional AC equation. We set the number of approximation points to 400 for all cases.

Model	True parameters	Identified parameters	L^2 relative error
1D time-fractional AC equation	$\alpha = 0.8$, $\varepsilon = 1$	$\widehat{\alpha} = 0.81214$, $\widehat{\varepsilon} = 1.0005$	2.81e-4
2D time-fractional AC equation	$\alpha = 0.8$, $\varepsilon = 1$	$\widehat{\alpha} = 0.80086$, $\widehat{\varepsilon} = 1.0010$	6.93e-4

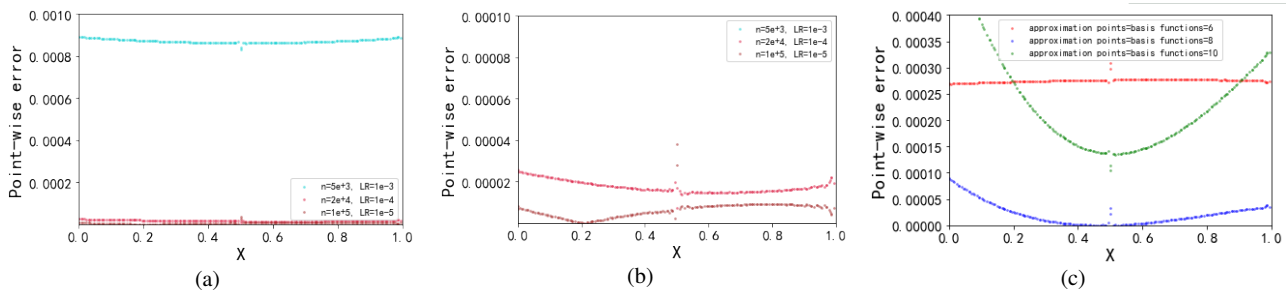

Fig. 12: 1D time-fractional CH equation with smooth solution: effect of iterations and the effect of the same number of approximate points and basis functions on point-wise error. (a) Point-wise error versus learning rate and number of iterations for fixed numbers of approximate points and basis functions. (b) Detail of (a). (c) Different numbers of approximate points and basis functions with fixed number of iterations and learning rate.

s, the convergence accuracy is improved slightly. When we fix the number of neurons and change the number of approximate points, we find that when the number of approximate points is too high, the convergence accuracy decreases. However, regardless of this number, the convergence accuracy is 10^{-6}. An increasing number of approximate points may lead to more parameters to be optimized, whereas the number of neurons does not increase sufficiently to substantially change the convergence accuracy. These results indicate that there is an optimal match between the number of neurons and the number of approximate points. They also show that the convergence accuracy does not depend only on these two factors.

Similar to the case of the time-fractional AC model, the number of basis functions is an important factor affecting the convergence accuracy of the time-fractional CH model. As shown in Figure 14, when the number of basis functions increases from 4 to 20, the convergence accuracy presents obvious wave crest variation, indicating that having too few or, especially, too many basis functions will have a detrimental effect on the convergence accuracy. Moreover, too many basis functions will make the model more complex and increase the number of factors that need to be considered; thus, the network will not be able to accurately describe the solution that satisfies all these factors. This makes generalization less precise. In other words, we need to consider not only whether the basis functions can represent the whole world but also whether the network has the required fitting ability. As shown in Figures 14 (c)–(d), when we fix the number of basis functions and change the number of approximate points, we find that the convergence accuracy changes; however, this is a relatively weak effect compared with that of the number of basis functions. Thus, when the NN structure is fixed, the influence of the number of basis functions is greater than that of the number of approximate points.

We also consider the influence of the NN structure on the L^2 relative error in Figure 15. The convergence accuracy is between 10^{-4} and 10^{-6}, that is, it is relatively stable. Note that the greater the width, the greater the uncertainty, for training 10 times up to a minimum of 10^{-7}. As shown in

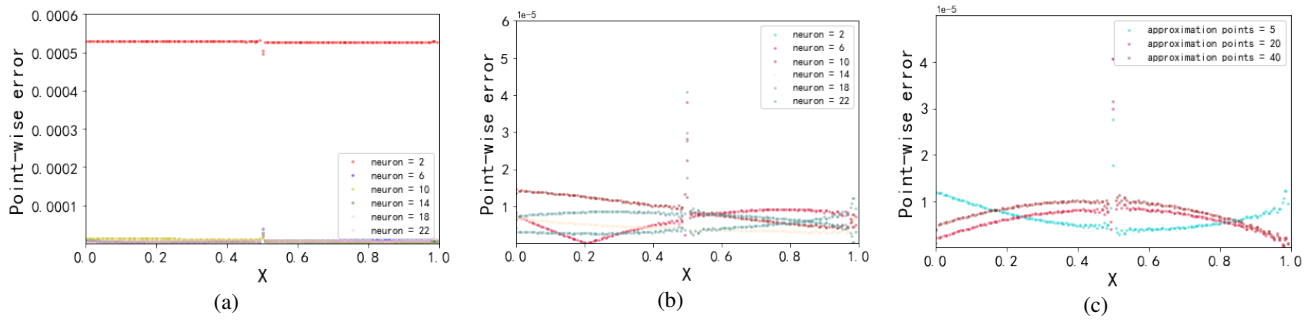

Fig. 13: 1D time-fractional CH equation with smooth solution: effects of numbers of neurons and approximate points on point-wise error. (a) Point-wise error versus neuron number N for a fixed number of approximate points. (b) Detail of (a). (c) Point-wise error versus number of approximate points for a fixed number of neurons.

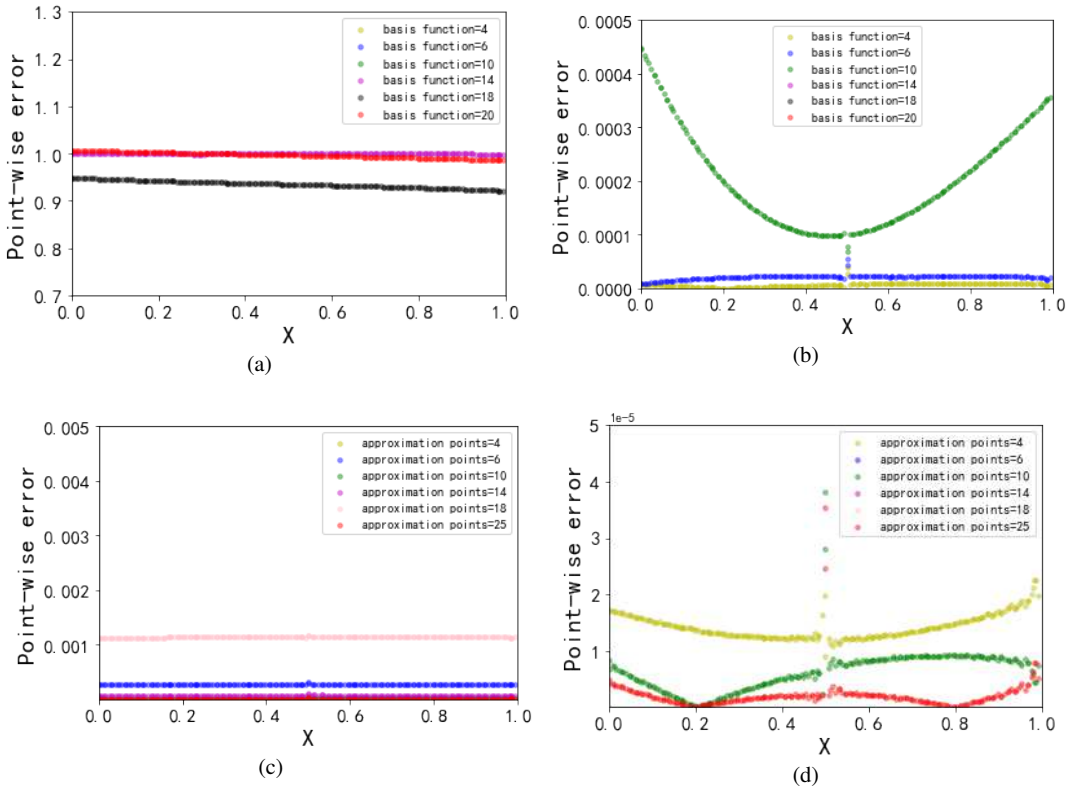

Fig. 14: 1D time-fractional CH equation with smooth solution: effects of numbers of basis function and approximate points. (a) Point-wise error versus number of basis functions ϕ_i for fixed number of approximate points. (b) Detail of (a). (c) Point-wise error versus number of approximate points for a fixed number of basis functions. (d) Detail of (c).

Figure 16 (a), as W_I increases, the error curve flattens out. This is also due to the preference of the optimization center for the initial conditions. These results also show that W_I has an effect on the accuracy of L^2 relative error but is not the main influencing factor.

Example 4. Finally, 2D time-fractional CH equation is considered,

$$\begin{aligned} &D_t^{\alpha} u(x,y,t) - \gamma\Delta(\varepsilon\Delta u(x,y,t) + u^3(x,y,t) - u(x,y,t)) \\ &= f(x,y,t), \\ &(x,y,t) \in (0,1)^2 \times (0,T], \\ &u(x,y,0) = g(x,y),\ (x,y) \in (0,1)^2, \end{aligned} \tag{45}$$

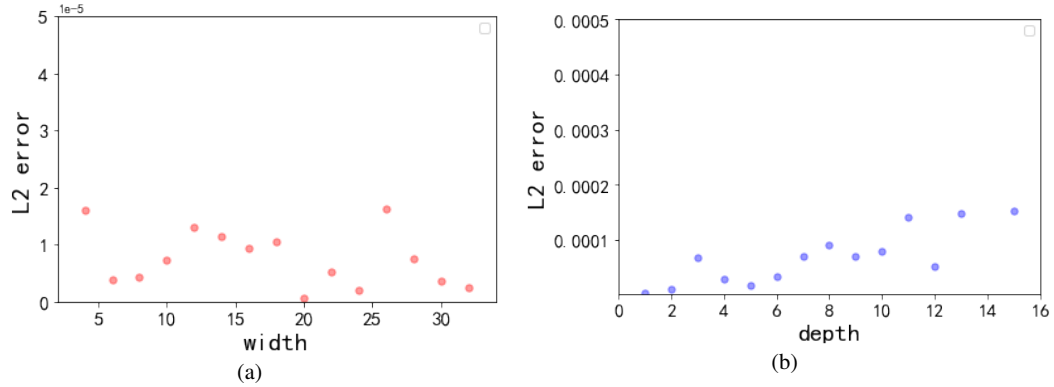

Fig. 15: Effect of NN architecture on the error. (b) Point-wise error versus NN width for depth of 1. (c) Point-wise error versus NN depth for width of 6.

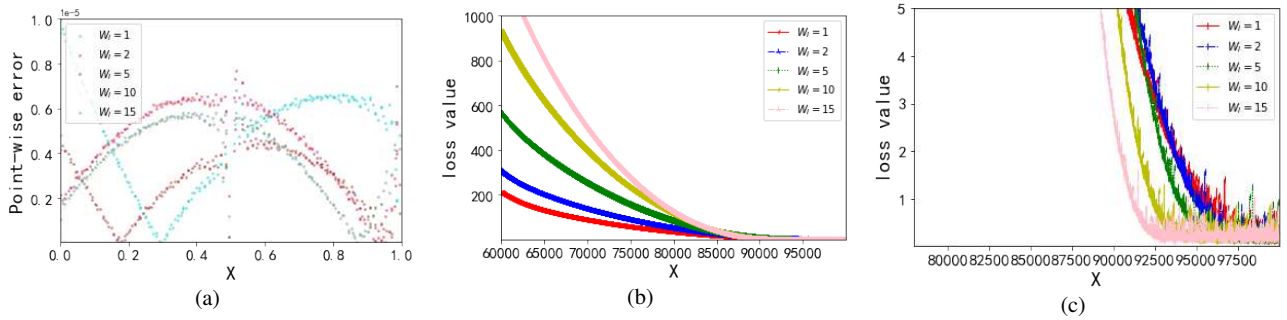

Fig. 16: 1D time-fractional CH equation with smooth fabricated solution: effects of penalty parameter τ. (a) Point-wise error for different τ values. (b) Loss value. (c) Detail of (b).

with periodic condition, and fix $\alpha = 0.8$, $\gamma = 1$, and $\varepsilon = 1$, with $\alpha_0 = 0.7$, $\gamma_0 = 0$, and $\varepsilon_0 = 0$. We consider the smooth fabricated solution $(1+t^2)\sin(2\pi x)\sin(2\pi y)$, where the corresponding force term is:

$$f(x,y,t) = \frac{\Gamma(3)}{\Gamma(2.2)}t^{1.2}\sin(2\pi x)\sin(2\pi y) - 64\pi^4(1+t^2)\sin(2\pi x)\sin(2\pi y) - 24\pi^2((1+t^2)\sin(2\pi x)\sin(2\pi y))(((1+t^2)\cos(2\pi x)\sin(2\pi y))^2 + ((1+t^2)\sin(2\pi x)\cos(2\pi y))^2) - 8\pi^2(1+t^2)\sin(2\pi x)\sin(2\pi y) + 24\pi^2((1+t^2)\sin(2\pi x)\sin(2\pi y))^3. \tag{46}$$

With the same numerical solution as in Eq. (38), the loss function can be written as follows:

$$L(\mu) = MSE_{equ} + W_I \times MSE_{IVC}, \tag{47}$$

where

$$MSE_{equ} = \frac{1}{N_{equ}}\sum_{k=1}^{N_{equ}}\Big(\frac{1}{\tau^\alpha}\sum_{j=0}^{n} w_{n-j}^{(\alpha)}\sum_{i=0}^{N} u(x_i,y_i,t_j)\phi_i(x_k,y_k)\sin(2\pi x_k)\sin(2\pi y_k) - \Delta\Big(\Delta\Big(\sum_{i=0}^{N} u(x_i,y_i,t_j)\phi_i(x_k,y_k)\sin(2\pi x_k)\sin(2\pi y_k)\Big) + \Big(\sum_{i=0}^{N} u(x_i,y_i,t_j)\phi_i(x_k,y_k)\sin(2\pi x_k)\sin(2\pi y_k)\Big)^3 - \sum_{i=0}^{N} u(x_i,y_i,t_j)\phi_i(x_k,y_k)\sin(2\pi x_k)\sin(2\pi y_k)\Big) - f(x_k,y_k,t_n)\Big)^2, \tag{48}$$

$$MSE_{IVC} = \frac{1}{N_{IVC}}\sum_{j=1}^{N_{IVC}}(u_{NN}(x_j,y_j,0) - g(x_j,y_j))^2. \tag{49}$$

Figure 17 shows the exact solutions, predictions, and point-wise error. In Figure 17 (c), fPINNs based on the spectral collocation method achieves a good convergence accuracy of 10^{-5}, indicating that our method is effective for 2D time-fractional CH equations.

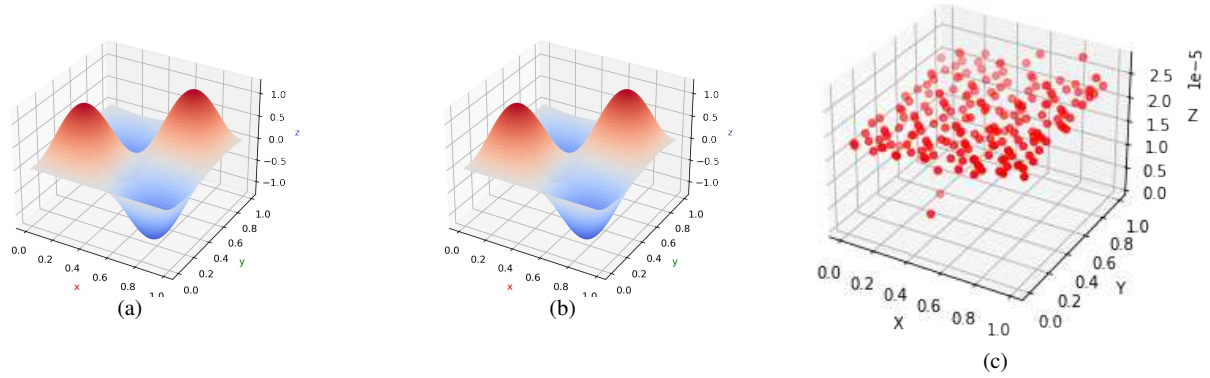

Fig. 17: 2D time-fractional CH equation with smooth solution. (a) Exact solution. (b) Approximation of fPINNs based on spectral collocation method. (c) Point-wise error.

Table 4: Identified parameters and the L^2 relative error of u for inverse problems of time-fractional CH equation. We set the number of approximation points to 400 for all cases.

Model	True parameters	Identified parameters	L^2 relative error
1D time-fractional CH equation	$\alpha = 0.8, \gamma = 1, \varepsilon = 1$	$\widehat{\alpha} = 0.80508, \widehat{\gamma} = 1.03102, \widehat{\varepsilon} = 0.97930$	1.02e-2
2D time-fractional CH equation	$\alpha = 0.8, \gamma = 1, \varepsilon = 1$	$\widehat{\alpha} = 0.78555, \widehat{\gamma} = 1.00146, \widehat{\varepsilon} = 1.16116$	7.53e-2

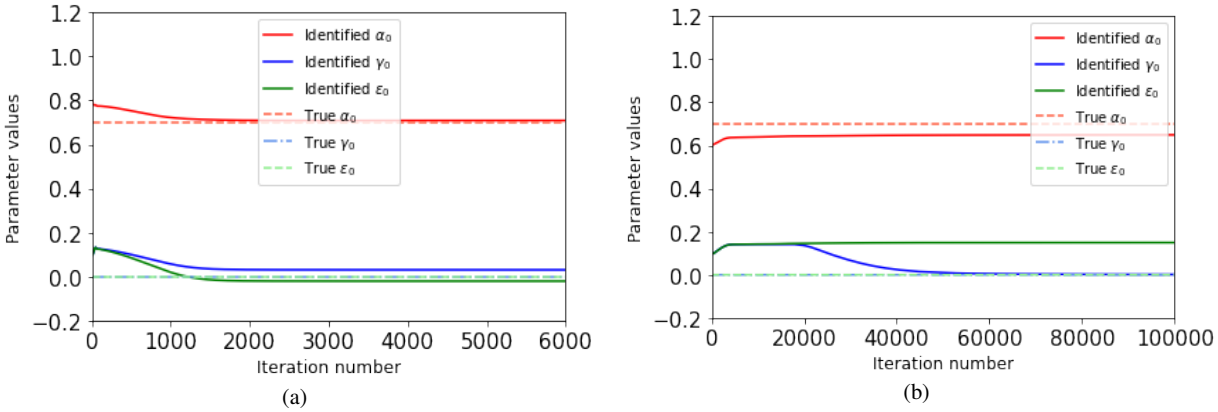

Fig. 18: Inverse problem with smooth fabricated solution. (a) Parameter evolution of 1D time-fractional CH equation. (b) Parameter evolution of 2D time-fractional CH equation.

The results of solving the inverse problems of 1D and 2D time-fractional CH equations are given in Figure 18. Better results are obtained for the inverse problem of the 1D time-fractional CH equation. For the three parameters of value $\alpha_0 = 0.7$, $\gamma_0 = 0$, and $\varepsilon_0 = 0$, we obtain the following estimates: $\widehat{\alpha} = 0.70919$, $\widehat{\gamma} = 0.03055$, and $\widehat{\varepsilon} = -0.020910$. The similar result is also achieved for the 2D time-fractional CH equation with an error accuracy of 0.0939, see Table 4.

These results demonstrate that fPINNs can deal with the inverse problem for time-fractional phase field models and and has a better ability to deal with the time-fractional AC equations. The poor results for the inverse problem of the time-fractional CH equations may have been due to the existence of higher-order derivatives, which means the loss function can not be optimized to the global minimum. In future work, we will further explore this inverse problem and find a way to deal with high-order derivatives. We will use meta-learning [34] to tune the hyperparameters of the NN architecture automatically and consider using different initialization methods to obtain better results.

Finally, in order to prove the superior generalizability of this method, we consider two different fabricated solution-

s. The results are presented in Table 5. Although we need to adjust the number of network layers and neurons to make the loss function reach the global minimum, the results demonstrate that our method can achieve better convergence accuracy with L^2 relative error of 10^{-4}.

Table 5: fabricated solution of L^2 relative error.

Fabricated solution	L^2 error
$(1+t^2)\sin^2(2\pi x)$	$4.248429e^{-4}$
$(1+t^2)x\sin(2\pi x)$	$5.414722e^{-4}$

5 Conclusion

In this work, we employe fPINNs based on a spectral collocation method to solve the forward problem and inverse problem of the time-fractional phase field models. For the forward problems, we uses the FBDF to approximate the time-fractional derivative and a spectral collocation method to discrete the space direction. Combined with the fPINNs, the numerical solution of the model is obtained. The convergence accuracy of this method is discussed in different cases. We show that the point-wise error accuracy of this method is 10^{-5} to 10^{-7}. For the inverse problems, the fPINNs is demonstrated to estimate the order of fractional derivative α, the mobility constant γ, and the coefficient ε. Several numerical examples are presented to demonstrate the effectiveness of our method. In future studies, the given methods can also be used to other time-fractional equations in biology and physics.

6 Summary statement

Conflict of interest The authors declare that they have no conflict of interests regarding the publication of this paper.

Funding This study was funded by the National Natural Science Foundation of China (grant number 12120101001, 12001326), Natural Science Foundation of Shandong Province (grant number ZR2020QA032), and China Postdoctoral Science Foundation (grant number BX20190191, 2020M672038).

Data availability statement The datasets generated during and analysed during the current study are available from the corresponding author on reasonable request.

References

1. S. Brunton, J. Proctor, J. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Nat. Acad. Sci. 113 (15) (2016) 3932 – 3937.
2. S. Brunton, B. Noack, P. Koumoutsakos, Machine learning for fluid mechanics. Annu. Rev. Fluid Mech. 52 (2020) 477 – 508.
3. G. Gao, J. Li, Y. Wen, DeepComfort: energy-efficient thermal comfort control in buildings via reinforcement learning. IEEE Internet Things J. 7 (2020) 8472 - 8484.
4. K. Bouman, B. Xiao, P. Battaglia, W. Freeman, Estimating the material properties of fabric from video. IEEE. (2013) 1984 - 1991.
5. R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, R. Gao, Deep learning and its applications to machine health monitoring. Mech. Syst. Signal Process. 115 (2019) 213 - 237.
6. Q. Wang, G. Zhang, C. Sun, N. Wu, High efficient load paths analysis with U index generated by deep learning. Comput. Methods Appl. Mech. Engrg. 344 (2019) 499 - 511.
7. D. Finol, Y. Lu, V. Mahadevan, A. Srivastava, Deep convolutional neural networks for eigenvalue problems in mechanics. Internat. J. Numer. Methods Engrg. 118 (5) (2019) 258 – 275.
8. A. Karpatne, G. Atluri, J. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, V. Kumar, Theory-guided Data Science: A New Paradigm for Scientific Discovery from Data. IEEE Trans. Knowl. Data Engrg. 29 (10) (2017) 2318 - 2331.
9. M. Raissi, P. Perdikaris, G. 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.
10. E. Kharazmi, Z. Zhang, G. Karniadakis, hp-VPINNs: Variational physics-informed neural networks with domain decomposition. Comput. Methods Appl. Mech. Eng. 374 (C) (2021) 113547.
11. G. Pang, M. D'Elia, M. Parks, G. Karniadakis, nPINNs: Nonlocal physics-informed neural networks for a parametrized nonlocal universal Laplacian operator. Algorithms and applications. J. Comput. Phys. 422 (2020) 109760.
12. A. Jagtap, G. Karniadakis, Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations. Commun. Comput. Phys. 28 (5) (2020) 2002 - 2041.
13. Q. Zheng, L. Zeng, G. Karniadakis, Physics-informed semantic inpainting: Application to geostatistical modeling. J. Comput. Phys. 419 (2020) 109676.
14. A. Giusti, A comment on some new definitions of fractional derivative. Nonlinear Dyn. 93 (2018) 1757 – 1763.
15. G. Wu, D. Baleanu, Discrete fractional logistic map and its chaos. Nonlinear Dyn. 75 (2014) 283 – 287.
16. G. Pang, L. Lu, G. Karniadakis, fPINNs: Fractional physics informed neural networks. SIAM J. Sci. Comput. 41 (2019) 2603 – 2626.
17. F. Rostami, A. Jafarian, A new artificial neural network structure for solving high-order linear fractional differential equations. Int. J. Comput. Math. 95 (3) (2018) 528 – 539.
18. M. Kapustina, D. Tsygankov, J. Zhao, T. Wesller, X. Yang, A. Chen, N. Roach, T. Elston, Q. Wang, K. Jacobson, M. Forest, Modeling the excess cell membrane stored in a complex morphology of bleb-like protrusions. Plos Comput. Biol. 12 (3) (2016) 1004841.
19. D. Shao, W. Pappel, H. Levine, Computational model for cell morphodynamics. Phys. Rev. Lett. 105 (10) (2010) 108104.
20. F. Ziebert, I. Aranson, Effects of adhesion dynamics and substrate compliance on the shape and motility of crawling cells. Plos One, 8 (5) (2013) 64511.
21. J. Zhao, Y. Shen, M. Happasalo, Z. Wang, Q. Wang, A 3D numerical study of antimicrobial persistence in heterogeneous multispecies biofilms. J. Theor. Biol. 392 (2016) 83 - 98.

22. R. Kohn, X. Yan, Upper bound on the coarsening rate for an epitaxial growth model. Comm. Pure Appl. Math. 56 (11) (2003) 1549 - 1564.
23. B. Li, J. Liu, Epitaxial growth without slope selection: energetics, coarsening and dynamic scaling. J. Nonlinear Sci. 14 (5) (2004) 429 - 451.
24. H. Zhang, X. Jiang, A high-efficiency second-order numerical scheme for time-fractional phase field models by using extended SAV method. Nonlinear Dyn. 102 (2020) 589 – 603.
25. H. Antil, S. Baetrls, Spectral Approximation of Fractional PDEs in Image Processing and Phase Field Modeling. J. Comput. Methods Appl. Math. 17 (4) (2017) 661 – 678.
26. H. Liu, A. Cheng, H. Wang, J. Zhao, Time-fractional Allen-Cahn and Cahn-Hilliard phase-field models and their numerical investigation. Comput. Math. Appl. 76 (8) (2018) 1876 – 1892.
27. F. Song, C. Xu, G. Karniadakis, A fractional phase-field model for two-phase flows with tunable sharpness: Algorithms and simulations. Comput. Methods Appl. Mech. Engrg. 305 (2016) 376 – 404.
28. B. Wu, Q. Chen, Z. Wang, Uniqueness and stability of an inverse problem for a phase field model using data from one component. Comput. Math. Appl. 66 (2013) 2126 - 2138.
29. N. Baranibalan, K. Sakthivel, K. Balachandran, J. Kim, Reconstruction of two time independent coefficients in an inverse problem for a phase field system. Nonlinear Analy. 72 (2010) 2841 – 2851.
30. F. Colombo, Direct and inverse problems for a phase-field model with memory. J. Math. Anal. Appl. 260 (2001) 517 - 545.
31. A. Khodadadian, N. Noii, M. Parvizi, M. Abbaszadeh, T. Wick, C. Heitzinger, A Bayesian estimation method for variational phase-field fracture problems. Comput. Mech. 66 (4) (2020) 827 - 849.
32. T. Bandai, T. Ghezzehei, Physics-Informed Neural Networks With Monotonicity Constraints for Richardson-Richards Equation: Estimation of Constitutive Relationships and Soil Water Flux Density From Volumetric Water Content Measurements. Water Resour. Res. 57 (2) (2021) 20.
33. S. Ruder, An overview of gradient descent optimization algorithms. arXiv:1609.04747v2 (2016).
34. M. Huisman, J. van Rijn, A. Plaat, A survey of deep meta-learning. Artif. Intell. Rev. 54 (6) (2021) 4483-4541.