

LT-PINNs: Physics-informed neural networks based on Laplace transform for solving Caputo-type fractional partial differential equations

Ruibo Zhang(张瑞波), Fengjun Li(李风军), and Jianqiang Liu(刘建强)
Citation: Chin. Phys. B, 2026, 35 (**3**): 030201. DOI: 10.1088/1674-1056/adf827
http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Spatial patterns of the Brusselator model with asymmetric Lévy diffusion

Hongwei Yin(尹洪位), Shangtao Yang(杨尚涛), Xiaoqing Wen(文小庆), Haohua Wang(王浩华), and Shufen Yang(杨淑芬)
Chin. Phys. B, 2024, 33 (**11**): 110202. DOI: 10.1088/1674-1056/ad74e5

MetaPINNs: Predicting soliton and rogue wave of nonlinear PDEs via the improved physics-informed neural networks based on meta-learned optimization

Yanan Guo(郭亚楠), Xiaoqun Cao(曹小群), Junqiang Song(宋君强), and Hongze Leng(冷洪泽)
Chin. Phys. B, 2024, 33 (**2**): 020203. DOI: 10.1088/1674-1056/ad0bf4

Meshfree-based physics-informed neural networks for the unsteady Oseen equations

Keyi Peng(彭珂依), Jing Yue(岳靖), Wen Zhang(张文), and Jian Li(李剑)
Chin. Phys. B, 2023, 32 (**4**): 040207. DOI: 10.1088/1674-1056/ac9cb9

Adaptive multi-step piecewise interpolation reproducing kernel method for solving the nonlinear time-fractional partial differential equation arising from financial economics

Ming-Jing Du(杜明婧), Bao-Jun Sun(孙宝军), and Ge Kai(凯歌)
Chin. Phys. B, 2023, 32 (**3**): 030202. DOI: 10.1088/1674-1056/ac9368

Data-driven parity-time-symmetric vector rogue wave solutions of multi-component nonlinear Schrödinger equation

Li-Jun Chang(常莉君), Yi-Fan Mo(莫一凡), Li-Ming Ling(凌黎明), and De-Lu Zeng(曾德炉)
Chin. Phys. B, 2022, 31 (**6**): 060201. DOI: 10.1088/1674-1056/ac4cc5

LT-PINNs: Physics-informed neural networks based on Laplace transform for solving Caputo-type fractional partial differential equations

Ruibo Zhang(张瑞波)[1,3], Fengjun Li(李风军)[2,3,†], and Jianqiang Liu(刘建强)[1,3,‡]

[1] *School of Mathematics and Statistics, Ningxia University, Yinchuan* 750021, *China*
[2] *School of Mathematics and Information Science, North Minzu University, Yinchuan* 750021, *China*
[3] *Ningxia Basic Science Research Center of Mathematics, Ningxia University, Yinchuan* 750021, *China*

(Received 27 April 2025; revised manuscript received 28 July 2025; accepted manuscript online 6 August 2025)

The solution of fractional partial differential equations (PDEs) is an important topic in scientific computing. However, the traditional physics-informed neural networks (PINNs) have problems of memory overflow and low computational efficiency when the derivative is discretized for a long time. Therefore in this paper we innovatively propose a framework of Laplace transform physics-informed neural networks (LT-PINNs), which is dedicated to solving the forward and inverse problems of Caputo-type fractional PDEs. The core of this method is to use the Laplace transform to construct the loss function, which skillfully avoids the dilemma that the fractional derivative operator in traditional PINNs is difficult to operate effectively. By studying the benchmark problem of parameter α in a series of different scenarios we verify that LT-PINNs can predict the solution of Caputo-type fractional PDEs more accurately than fractional PINNs. The excellent performance of LT-PINNs in identifying inverse problems involving fractional order, convection and diffusion coefficients is further explored. At the same time, the effects of network structure, the number of sampling points and noise on the LT-PINNs method are analyzed in detail. The results show that the method can predict the solution of the equation satisfactorily even under severe noise interference. The proposed LT-PINNs framework opens up a new path for efficiently solving fractional PDEs. It shows significant advantages in improving computational efficiency, reducing memory usage and dealing with complex noise environments. It is expected to promote the further development of fractional PDEs in many fields.

Keywords: fractional partial differential equation, Caputo-type derivative, physics-informed neural network, Laplace transform

PACS: 02.30.Jr, 07.05.Mh, 07.05.Tp, 47.11.–j

DOI: 10.1088/1674-1056/adf827 **CSTR:** 32038.14.CPB.adf827

1. Introduction

In traditional physical modeling, the system dynamics described by integer-order differential equations implies the "Markov" assumption that the future state of the system depends only on the current state and is independent of the historical path. However, observations of modern science in complex media and multi-scale processes show that a large number of physical phenomena are strongly non-Markovian, and their evolution has a significant memory effect and path dependence. In such systems, the current state is cumulatively affected by the entire historical evolution trajectory, and the weight of historical contributions often appears in the form of power-law decay. The classical integer-order derivative cannot describe this long-range time correlation because of its locality. The introduction of a Caputo-type fractional derivative provides a unified physical and mathematical framework to solve this problem. Many real-life phenomena have been modeled as fractional partial differential equations (PDEs), such as porous media,[1,2] electrochemistry,[3] acoustic wave propagation[4,5] and viscoelastic constitutive law,[6–8] and have attracted considerable research interest. In particular, in stress relaxation experiments with polymers, biological soft tissues and other materials,[9–12] the stress decay curve shows a power-law form ($\sigma(t) \sim t^{-\alpha}$) rather than exponential decay. This is due to the historical cumulative effect of the multi-scale entanglement and recombination process of molecular segments. The Caputo derivative replaces the Newton viscosity term and establishes a fractional-order constitutive equation (such as $\sigma(t) = E\tau^{\alpha} D_t^{\alpha} \varepsilon(t)$) to accurately describe the integral dependence of stress on the historical strain rate. When particles move in fractal porous media (such as rocks or bones) or intracellular crowded environments,[13,14] the mean squared displacement satisfies $\langle x^2(t) \rangle \sim t^{\alpha} (\alpha < 1)$ (sub-diffusion). Particles "remember" their past positions due to frequent collisions, adsorption or space constraints. In the time fractional diffusion equation $\partial_t^{\alpha} u = K_{\alpha} \nabla^2 u$, the Caputo operator quantifies the probability that the particle is trapped in the historical position through the $(t-\tau)^{-\alpha}$ kernel. The volatility of asset prices shows clustering (high volatility follows high volatil-

†Corresponding author. E-mail: fjli@nxu.edu.cn
‡Corresponding author. E-mail: liujq@amss.ac.cn

© 2026 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

ity), reflecting the persistence of market panic, and its autocorrelation function decays slowly with a power-law form. In the fractional Black–Scholes equation $\partial_t^\alpha V = LV$, the Caputo derivative of $\alpha < 1$ encodes the continuous influence of the historical price path on option pricing.[15–18] However, it is very difficult to obtain the analytical solutions of fractional PDEs. Therefore, more and more researchers are focusing on studying numerical solutions. In a direct discrete fractional-order derivative, the Grünwald–Letnikov formula[19] and the $L1$ formula[20,21] are considered as the two major numerical schemes. In particular, for the fractional-order derivative α $(0 < \alpha < 1)$, the $L1$ formula can improve the accuracy of order $2-\alpha$.[22–24] Gao *et al.*[25] obtained a new numerical formula (called the $L1-2$ formula) by using the quadratic interpolation approximation for fractional-order derivatives of order α $(0 < \alpha < 1)$, which achieved $3-\alpha$-order accuracy. Recently, Yang *et al.*[26] proposed an improved spectral deferred correction method for solving nonlinear fractional PDEs which can be easily constructed with high accuracy from low-order methods. Xu[27] developed a method based on generalized homotopy to give highly accurate solutions of fractional PDEs. For PDEs with a Caputo-type fractional-order derivative, the numerical scheme may require using the values of all previous time steps when computing the time derivative term, which will increase the cost of memory to store the necessary data during long computations, especially for multi-dimensional fractional PDEs.

With the explosive growth of data volume and rapid development of computer hardware, deep neural networks (DNNs) have been widely used in computer vision,[28] feature selection,[29] natural language processing,[30] classification tasks,[31] reinforcement learning,[32] data mining[33] and so on. In recent years, some researchers have also tried to apply DNNs to solve such problems of PDEs as high dimension and nonlinearity which do not require grid generation of the solution region compared with conventional numerical methods.[34–38] However, the main problems with pure data-driven methods are a lack of physical interpretability and flexibility, as well as an inability to ensure the convergence of the algorithm. For this reason, Raissi *et al.*[39–41] developed a new DNN framework called physics-informed neural networks (PINNs), which successfully embeded the underlying physical information into neural networks. In the PINN framework, DNNs are employed as the core of the surrogate model to predict the solution of PDEs, and the automatic differentiation (AD)[42] of neural networks is used to calculate the differential operator in PDEs. Compared with traditional numerical methods, the AD technique does not involve discrete computation, which can effectively reduce the computation cost and avoid the error propagation problem. Due to the significant advantage of PINNs in describing complex relationships between inputs and outputs, they provide a new paradigm for solving forward and inverse problems in nonlinear PDEs. PINNs have been successfully applied to complex problems in computational science and engineering, such as nonlinear optics,[43,44] materials science,[45–47] finance[48] and fluid mechanics.[49–51] However, PINNs cannot be directly applied to fractional PDEs because the AD technique is not valid in fractional calculus. For this reason, Pang *et al.*[52] proposed fractional PINNs (fPINNs), which used numerical discretization to numerically approximate the fractional-order derivative. However, in order to achieve satisfactory accuracy, high-order discretization is usually required. With increase in discrete points and discrete times, the computational cost of each iteration increases significantly, which will reduce the training efficiency of PINNs. Although MC-fPINN[53] solved this problem using the Monte Carlo method, use of the Monte Carlo method to calculate the singular integral of fractional derivatives may lead to significant variance and error, which hinders the convergence of the PINN. In addition, Zhang *et al.*[54] proposed a fractional PINN framework based on spectral methods, which is based on a more efficient global discretization method of Jacobi polynomials, thereby reducing the need for auxiliary points. There are a lot of researches on fractional PDEs, but they all focus on the discretization and sampling of fractional derivatives in the training process rather than the offline process.

In this paper, inspired by the traditional numerical methods of PDEs,[55] we propose a Laplace transform physics-informed neural network (LT-PINN) framework to solve the forward and inverse problems of Caputo-type fractional PDEs. An LT-PINN is a kind of DNN with the same intended target as a PINN, and will be trained as a surrogate model to approximate the exact solutions of Caputo-type fractional PDEs. Here, we would like to emphasize that in the LT-PINN framework, based on the Laplace transform, Caputo-type fractional PDEs can be used for traditional PINNs and then the model can be trained by using the general approximation ability of neural networks[56] and the AD technique. The LT-PINN framework has several appealing features. (i) By using the Laplace transform, LT-PINNs can easily solve Caputo-type fractional PDEs because the fractional order derivative avoids discretization. (ii) LT-PINNs just need to transform fractional PDEs into integer-order PDEs in an offline form and then use traditional PINNs for training. (iii) When we rewrite the code for the inverse problem, LT-PINNs require only the addition of identified parameters to the list of optimized parameters in the forward problem; nothing else needs to change. (iv) LT-PINNs are very noise tolerant and a satisfactory result can be obtained even with very noisy data.

This paper is organized as follows. In Section 2, we describe the general form of the forward and inverse problems of

Caputo-type fractional PDEs. In Section 3, we first briefly introduce standard PINNs, and subsequently derive the Laplace transform of Caputo-type fractional PDEs for the cases $0 < \alpha < 1$ and $1 < \alpha < 2$, and propose LT-PINNs. In Section 4, we demonstrate, with a large number of examples, the effectiveness of the LT-PINN framework for solving the forward and inverse problems of various Caputo-type fractional PDEs. Specifically, we first address the benchmark problem of Caputo-type fractional PDEs in the case of $0 < \alpha < 1$. Furthermore, we study the forward problem of Caputo-type fractional PDEs for the case $1 < \alpha < 2$. In order to verify the ability of LT-PINNs to solve the inverse problem of Caputo-type fractional PDEs, the fractional order, convection and diffusion coefficients are identified. Finally, we discuss the performance of LT-PINNs with different noisy data, and explore the influence of the number of sampling points and the network structure on the accuracy of the solution. The conclusion is given in Section 5.

2. Problem statement

For PDEs with a Caputo-type fractional derivative such as

$$ {}_0^{\mathrm{C}}D_t^{\alpha}u(x,t) = \beta\Delta u(x,t) - \gamma\nabla u(x,t) + h(x,t), \tag{1} $$

where $u(x,t)$ is the solution of the equation, $h(x,t)$ is the source term and β and γ are the diffusion and convection coefficients, respectively. The left-hand side of the above equation is a time-fractional derivative of order α, which is defined in the Caputo sense:[57]

$$ \begin{aligned} {}_0^{\mathrm{C}}D_t^{\alpha}u(x,t) &= \frac{1}{\Gamma(n-\alpha)}\int_0^t \frac{u_t(x,\tau)}{(t-\tau)^{\alpha+1-n}}\,\mathrm{d}\tau, \\ & n-1 \le \alpha < n, \end{aligned} \tag{2} $$

where $\Gamma(\cdot)$ is the gamma function. In order to compute the term ${}_0^{\mathrm{C}}D_t^{\alpha}u(x,t)$, the numerical scheme needs to use the values of all previous time steps. However, this requires a large amount of memory to store the data in long computations, especially when solving multi-dimensional fractional PDEs, which can lead to a memory overflow problem.

In this paper, we consider both forward and inverse problems of Caputo-type fractional PDEs in the cases $0 < \alpha < 1$ and $1 < \alpha < 2$. In the forward problem, given the time-fractional derivative order α, the diffusion coefficient β, the convection coefficient γ, the source term $h(x,t)$, as well as the initial and boundary conditions, we solve Eq. (1). On the other hand, in the inverse problem, the parameters α, β and γ in Eq. (1) are unknown. We will use some measurement data to identify the unknown parameters α, β and γ.

3. Method

In this section, we propose LT-PINNs for solving the forward and inverse problems of Caputo-type fractional PDEs. Firstly, we introduce the original PINNs and the Laplace transform. Then, by using the Laplace transform, integer-order PDEs are introduced into the PINN framework. Finally, the LT-PINN framework is given in detail.

3.1. Physics-informed neural networks

PINNs are a machine learning framework based on DNNs that successfully solve forward and inverse problems of nonlinear PDEs. To introduce the idea behind PINNs we consider the forward problem of a one-dimensional integer-order diffusion equation with Dirichlet boundary conditions, given by

$$ \begin{aligned} \frac{\partial u(x,t)}{\partial t} &= \lambda\frac{\partial^2 u(x,t)}{\partial x^2} + h(x,t), \quad (x,t) \in (0,1)\times(0,1), \\ u(x,0) &= g(x,0), \quad x \in (0,1), \\ u(0,t) &= \varphi(0,t), \quad t \in (0,1), \\ u(1,t) &= \varphi(1,t), \quad t \in (0,1), \end{aligned} \tag{3} $$

where $g(x,0)$ is the initial condition at $t = 0$ and $\varphi(0,t)$ and $\varphi(1,t)$ are the Dirichlet boundary conditions at $x = 0$ and $x = 1$, respectively. A parameter λ is known in the forward problem. In PINNs, the training points (x,t) are the inputs of the DNNs, and (x,t) consists of three parts: the collocation points (x^f,t^f) in the solution region, the sampling points $(x^i,0)$ on the initial condition and the sampling points (x^b,t^b) on the boundary condition. A DNN is used to calculate the predicted solution $u_{\mathrm{pred}}(x,t;\theta)$ corresponding to the input point (x,t), where θ represents the parameters of the DNN. By using the AD technique, we expect to optimize the parameters of the DNN so that the resulting predicted solution will satisfy both the equation and the initial boundary conditions. Now, we can give the loss function *Loss* for solving Eq. (3) using a PINN, as follows:

$$ Loss = Loss_i + Loss_b + Loss_f, \tag{4} $$

where

$$ Loss_i = \frac{1}{N_i}\sum_{i=1}^{N_i}\left|u_{\mathrm{pred}}(x^i,0;\theta) - g(x^i,0)\right|^2, \tag{5} $$

$$ \begin{aligned} Loss_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\Bigg(&\left|u_{\mathrm{pred}}(0,t^b;\theta) - \varphi(0,t^b)\right|^2 \\ &+ \left|u_{\mathrm{pred}}(1,t^b;\theta) - \varphi(1,t^b)\right|^2\Bigg), \end{aligned} \tag{6} $$

$$ \begin{aligned} Loss_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\Bigg|&\frac{\partial u_{\mathrm{pred}}(x^f,t^f;\theta)}{\partial t} \\ &- \lambda\frac{\partial^2 u_{\mathrm{pred}}(x^f,t^f;\theta)}{\partial x^2} - h(x^f,t^f)\Bigg|^2, \end{aligned} \tag{7} $$

in which N_i, N_b and N_f are the numbers of sampling points on the initial condition, boundary condition and solution region, respectively. $\varphi(0,t^b)$ and $\varphi(1,t^b)$ are the exact values of the sampling points $(x^i,0)$, $(0,t^b)$ and $(1,t^b)$, respectively. $Loss_i$, $Loss_b$ and $Loss_f$ are the mean squared errors of the initial condition, boundary condition and equation, respectively.

In solving the inverse problem of a PDE (i.e., discovering the unknown parameter λ in PDEs using measured data), we usually only know the general form of the equation and the measured data. Therefore, the loss function includes the residuals of the equation and measured data. The loss function $Loss$ can be expressed as

$$Loss = Loss_d + Loss_f, \tag{8}$$

where

$$Loss_d = \frac{1}{N_d}\sum_{i=1}^{N_d}\left|u_{\text{pred}}(x^d,t^d;\theta,\lambda)-u_d(x^d,t^d)\right|^2, \tag{9}$$

$$Loss_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\frac{\partial u_{\text{pred}}(x^f,t^f;\theta,\lambda)}{\partial t} - \lambda\frac{\partial^2 u_{\text{pred}}(x^f,t^f;\theta,\lambda)}{\partial x^2} - h(x^f,t^f)\right|^2, \tag{10}$$

in which (x^d,t^d) is the measured data, N_d is the number of measured data, and $u_d(x^d,t^d)$ is the exact value of the measured data (x^d,t^d). $Loss_d$ and $Loss_f$ are the mean squared errors of the measured data and equation, respectively.

3.2. Laplace transform PINNs

In this section, we can derive approximate integer-order PDEs of the original fractional differential equation by using the Laplace transform. The obtained PDEs can be solved by the traditional PINN method, which greatly reduces the computational cost. Consider PDEs with a Caputo-type fractional derivative and the initial boundary conditions as follows:

$$\begin{aligned} {}_0^{\mathrm{C}}D_t^{\alpha}u(x,t) &= \beta\frac{\partial^2 u(x,t)}{\partial x^2} - \gamma\frac{\partial u(x,t)}{\partial x} + h(x,t), \\ &\quad 0<x<L,\ t>0,\ 0<\alpha<2, \\ u(x,0) &= f(x), \quad u_t(x,0)=g(x), \quad 0<x<L, \\ u(0,t) &= \phi(t), \quad u(L,t)=\psi(t), \quad t>0, \end{aligned} \tag{11}$$

where $f(x)$ and $g(x)$ are the initial conditions and $\phi(t)$ and $\psi(t)$ are the boundary conditions. In order to compute ${}_0^{\mathrm{C}}D_t^{\alpha}u(x,t)$ in fPINNs, the fractional order derivative is approximated using a finite difference scheme, which requires storing the values of all previous time steps. To avoid this trouble, we approximate the Caputo-type fractional derivative by using the Laplace transform method. Here, we consider the Laplace transform of ${}_0^{\mathrm{C}}D_t^{\alpha}u(x,t)$ in two different cases, that is, $0<\alpha<1$ and $1<\alpha<2$. In the case of $0<\alpha<1$, we ignore $u_t(x,0)=g(x)$.

Case 1 $0<\alpha<1$

When $0<\alpha<1$, the Caputo-type fractional derivative is approximated using the Laplace transform as follows:

$$\begin{aligned} L\left\{{}_0^{\mathrm{C}}D_t^{\alpha}u(x,t)\right\} &= s^{\alpha}\hat{u}(x,s) - s^{\alpha-1}u(x,0) \\ &= s^{\alpha}[\hat{u}(x,s) - s^{-1}u(x,0)], \end{aligned} \tag{12}$$

where $\hat{u}(x,s)$ is the Laplace transform of $u(x,s)$. Since $0<\alpha<1$, we linearize the term s^{α} as

$$s^{\alpha} \approx \alpha s^1 + (1-\alpha)s^0 = \alpha s + (1-\alpha), \tag{13}$$

and then substitute in into Eq. (12). This gives

$$\begin{aligned} L\left\{{}_0^{\mathrm{C}}D_t^{\alpha}u(x,t)\right\} &\approx [\alpha s+(1-\alpha)][\hat{u}(x,s)-s^{-1}u(x,0)] \\ &= \alpha s[\hat{u}(x,s)-s^{-1}u(x,0)] \\ &\quad +(1-\alpha)[\hat{u}(x,s)-s^{-1}u(x,0)]. \end{aligned} \tag{14}$$

Thus, the inverse Laplace transform gives

$${}_0^{\mathrm{C}}D_t^{\alpha}u(x,t) \approx \alpha\frac{\partial u(x,t)}{\partial t} + (1-\alpha)[u(x,t)-u(x,0)]. \tag{15}$$

Substituting Eq. (15) into Eq. (11), we can obtain the equation of integer order as follows:

$$\begin{aligned} &\alpha\frac{\partial u(x,t)}{\partial t} + (1-\alpha)[u(x,t)-f(x)] \\ &= \beta\frac{\partial^2 u(x,t)}{\partial x^2} - \gamma\frac{\partial u(x,t)}{\partial x} + h(x,t), \\ &0<x<L, \quad t>0, \\ &u(x,0)=f(x), \quad 0<x<L, \\ &u(0,t)=\phi(t), \quad u(L,t)=\psi(t), \quad t>0. \end{aligned} \tag{16}$$

In the following, we prove that the above simplified initial and boundary value problem is reasonable. In fact, by multiplying Eq. (16) by $u(x,t)$ and then integrating the resulting equation with respect to x over the interval $[0,L]$, we can obtain

$$\begin{aligned} &\alpha\int_0^L \frac{\partial u}{\partial t}u\,\mathrm{d}x + (1-\alpha)\int_0^L u^2\,\mathrm{d}x \\ &= \beta\int_0^L \frac{\partial^2 u}{\partial x^2}u\,\mathrm{d}x - \gamma\int_0^L \frac{\partial u}{\partial x}u\,\mathrm{d}x \\ &\quad +(1-\alpha)\int_0^L fu\,\mathrm{d}x + \int_0^L hu\,\mathrm{d}x. \end{aligned} \tag{17}$$

Using the homogeneous Dirichlet boundary condition, integration by parts and the Cauchy–Schwarz inequality, Eq. (17) can be simplified to

$$\begin{aligned} &\frac{\alpha}{2}\frac{\mathrm{d}}{\mathrm{d}t}\int_0^L u^2(x,t)\,\mathrm{d}x + (1-\alpha)\int_0^L u^2(x,t)\,\mathrm{d}x \\ &-\beta\int_0^L\left(\frac{\partial u(x,t)}{\partial x}\right)^2\mathrm{d}x + \gamma\int_0^L\frac{\partial u(x,t)}{\partial x}\,\mathrm{d}x \\ &= (1-\alpha)\int_0^L f(x)u(x,t)\,\mathrm{d}x + \int_0^L h(x,t)u(x,t)\,\mathrm{d}x \\ &\le \frac{1-\alpha}{2}\left[\int_0^L u^2(x,t)\,\mathrm{d}x + \int_0^L f^2(x)\,\mathrm{d}x\right] \end{aligned}$$

$$+\frac{1-\alpha}{2}\int_0^L u^2(x,t)\,\mathrm{d}x+\frac{1}{2(1-\alpha)}\int_0^L h^2(x,t)\,\mathrm{d}x, \quad (18)$$

implying that

$$\begin{aligned}&\frac{\alpha}{2}\frac{\mathrm{d}}{\mathrm{d}t}\int_0^L u^2(x,t)\,\mathrm{d}x-\beta\int_0^L\left(\frac{\partial u(x,t)}{\partial x}\right)^2\mathrm{d}x\\&+\gamma\int_0^L\frac{\partial u(x,t)}{\partial x}\,\mathrm{d}x\\\le&\frac{1-\alpha}{2}\int_0^L f^2(x)\,\mathrm{d}x+\frac{1}{2(1-\alpha)}\int_0^L h^2(x,t)\,\mathrm{d}x. \quad (19)\end{aligned}$$

Thus, integrating both sides of Eq. (19) with respect to t we can obtain

$$\begin{aligned}&\frac{\alpha}{2}\int_0^L u^2(x,t)\,\mathrm{d}x-\beta\int_0^t\int_0^L\left(\frac{\partial u(x,\tau)}{\partial x}\right)^2\mathrm{d}x\mathrm{d}\tau\\&+\gamma\int_0^t\int_0^L\frac{\partial u(x,\tau)}{\partial x}\,\mathrm{d}x\mathrm{d}\tau\\\le&\left(\frac{\alpha}{2}+\frac{1-\alpha}{2}t\right)\int_0^L f^2(x)\,\mathrm{d}x\\&+\frac{1}{2(1-\alpha)}\int_0^t\int_0^L h^2(x,\tau)\,\mathrm{d}x\mathrm{d}\tau. \quad (20)\end{aligned}$$

This indicates that the solution is controlled by the initial condition $f(x)$ and the source term $h(x,t)$. Therefore, the simplified initial and boundary value problem is reasonable. In addition, the simplified problem can be easily solved using a finite different scheme such as the conventional two-level in time Crank–Nicolson scheme as follows:

$$\begin{aligned}&\alpha\frac{u_j^{n+1}-u_j^n}{\Delta t}+(1-\alpha)\left[\frac{u_j^{n+1}+u_j^n}{2}-f_j\right]\\=&\beta\delta_x^2\frac{u_j^{n+1}+u_j^n}{2}-\gamma\delta_x\frac{u_j^{n+1}+u_j^n}{2}+h_j^{n+\frac{1}{2}},\\&1\le j\le M-1,\ \ n\ge 0,\\&u_j^0=f(x_j),\ \ 1\le j\le M-1,\\&u_0^n=\phi(t_n),\ \ u_M^n=\psi(t_n),\quad \text{for any } n, \quad (21)\end{aligned}$$

where a uniform grid is designed on $[0,L]$, the grid points are $x_j=j\Delta x$, $j=0,1,\ldots,M$, $\Delta x=L/M$ is the grid size, $t_n=n\Delta t$ is the time step and n is the time level. Here, δ_x^2 is a second-order finite difference operator such that $\delta_x^2u_j^n=\frac{u_{j-1}^n-2u_j^n+u_{j+1}^n}{(\Delta x)^2}$, and so on. Obviously, the truncation error at the location $(x_{j+1/2},t_{n+1/2})$ is $O[(\Delta t)^2+(\Delta x)^2]$. Furthermore, it can be proved that the finite difference scheme in Eq. (21) is unconditionally stable, and the order of convergence in time and space is 2 in the sense of the maximum norm.

Case 2 $1<\alpha<2$

(1) Linear interpolation

Similarly, we can simplify the Caputo-type fractional derivative at $1<\alpha<2$. Using the Laplace transform, we can obtain

$$\begin{aligned}L\left\{{}_0^{\mathrm{C}}D_t^\alpha u(x,t)\right\}&=s^\alpha\hat{u}(x,s)-s^{\alpha-1}u(x,0)-s^{\alpha-2}u_t(x,0)\\&=s^\alpha[\hat{u}(x,s)-s^{-1}u(x,0)-s^{-2}u_t(x,0)]. \quad (22)\end{aligned}$$

Here, using a linear interpolation of the power function of s^α at interpolation points 1 and 2, we have

$$s^\alpha\approx(\alpha-1)s^2+(2-\alpha)s^1. \quad (23)$$

Then substitution into Eq. (22) gives

$$\begin{aligned}&L\left\{{}_0^{\mathrm{C}}D_t^\alpha u(x,t)\right\}\\\approx&[(\alpha-1)s^2+(2-\alpha)s^1][\hat{u}(x,s)-s^{-1}u(x,0)-s^{-2}u_t(x,0)]\\=&(\alpha-1)s^2[\hat{u}(x,s)-s^{-1}u(x,0)-s^{-2}u_t(x,0)]\\&+(2-\alpha)s[\hat{u}(x,s)-s^{-1}u(x,0)]-(2-\alpha)s^{-1}u_t(x,0). \quad (24)\end{aligned}$$

Thus, the inverse Laplace transform gives

$$\begin{aligned}{}_0^{\mathrm{C}}D_t^\alpha u(x,t)\approx&(\alpha-1)\frac{\partial^2u(x,t)}{\partial t^2}\\&+(2-\alpha)\left[\frac{\partial u(x,t)}{\partial t}-u_t(x,0)\right]. \quad (25)\end{aligned}$$

Therefore, the original PDE with Caputo-type fractional derivative can be transformed into an equation of integer order as follows:

$$\begin{aligned}&(\alpha-1)\frac{\partial^2u(x,t)}{\partial t^2}+(2-\alpha)\left[\frac{\partial u(x,t)}{\partial t}-g(x)\right]\\=&\beta\frac{\partial^2u(x,t)}{\partial x^2}-\gamma\frac{\partial u(x,t)}{\partial x}+h(x,t),\\&0<x<L,\quad t>0,\\&u(x,0)=f(x),\quad u_t(x,0)=g(x),\quad 0<x<L,\\&u(0,t)=\phi(t),\quad u(L,t)=\psi(t),\quad t>0. \quad (26)\end{aligned}$$

(2) Quadratic interpolation

If the quadratic interpolation of s^α at the interpolation points 0, 1 and 2 is used, we can obtain

$$s^\alpha\approx\frac{\alpha(\alpha-1)}{2}s^2+\alpha(2-\alpha)s^1+\frac{(\alpha-1)(\alpha-2)}{2}s^0. \quad (27)$$

Next, Eq. (22) can be expressed as

$$\begin{aligned}&L\left\{{}_0^{\mathrm{C}}D_t^\alpha u(x,t)\right\}\\\approx&\left[\frac{\alpha(\alpha-1)}{2}s^2+\alpha(2-\alpha)s+\frac{(\alpha-1)(\alpha-2)}{2}\right]\\&\times\left[\hat{u}(x,s)-s^{-1}u(x,0)-s^{-2}u_t(x,0)\right]\\=&\frac{\alpha(\alpha-1)}{2}s^2\left[\hat{u}(x,s)-s^{-1}u(x,0)-s^{-2}u_t(x,0)\right]\\&+\alpha(2-\alpha)s\left[\hat{u}(x,s)-s^{-1}u(x,0)-s^{-2}u_t(x,0)\right]\\&+\frac{(\alpha-1)(\alpha-2)}{2}\left[\hat{u}(x,s)-s^{-1}u(x,0)-s^{-2}u_t(x,0)\right]. \quad (28)\end{aligned}$$

Thus, the inverse Laplace transform gives

$$\begin{aligned}&{}_0^{\mathrm{C}}D_t^\alpha u(x,t)\\\approx&\frac{\alpha(\alpha-1)}{2}\frac{\partial^2u(x,t)}{\partial t^2}+\alpha(2-\alpha)\left[\frac{\partial u(x,t)}{\partial t}-u_t(x,0)\right]\\&+\frac{(\alpha-1)(\alpha-2)}{2}\left[u(x,t)-u(x,0)-tu_t(x,0)\right], \quad (29)\end{aligned}$$

hence we obtain another approximation for Caputo-type fractional derivatives as follows:

$$\begin{aligned}
&\frac{\alpha(\alpha-1)}{2}\frac{\partial^2 u(x,t)}{\partial t^2}+\alpha(2-\alpha)\left[\frac{\partial u(x,t)}{\partial t}-g(x)\right]\\
&+\frac{(\alpha-1)(\alpha-2)}{2}[u(x,t)-f(x)-tg(x)]\\
=\ &\beta\frac{\partial^2 u(x,t)}{\partial x^2}-\gamma\frac{\partial u(x,t)}{\partial x}+h(x,t),\quad 0<x<L,\quad t>0,\\
&u(x,0)=f(x),\quad u_t(x,0)=g(x),\quad 0<x<L,\\
&u(0,t)=\phi(t),\quad u(L,t)=\psi(t),\quad t>0.
\end{aligned}\tag{30}$$

Now, Eq. (11) has been transformed into integer-order Eqs. (16), (26) and (30). Taking Eq. (16) as an example, we describe the process of solving LT-PINNs in detail. We define the LT-PINN $N(x,t)$, which can be expressed as

$$\begin{aligned}
N:=\ &\alpha\frac{\partial u(x,t)}{\partial t}+(1-\alpha)[u(x,t)-f(x)]\\
&-\beta\frac{\partial^2 u(x,t)}{\partial x^2}+\gamma\frac{\partial u(x,t)}{\partial x}-h(x,t),
\end{aligned}\tag{31}$$

and proceed by approximating $u(x,t)$ by a DNN $u_{\text{pred}}(x,t;\theta)$. Thus, we can give the loss function for solving Eq. (16) using a LT-PINN specifically as follows:

$$Loss=Loss_i+Loss_b+Loss_f,\tag{32}$$

where

$$Loss_i=\frac{1}{N_i}\sum_{i=1}^{N_i}\left|u_{\text{pred}}(x^i,0;\theta)-f(x^i,0)\right|^2,\tag{33}$$

$$\begin{aligned}
Loss_b=\ &\frac{1}{N_b}\sum_{i=1}^{N_b}\Big(\left|u_{\text{pred}}(0,t^b;\theta)-\phi(0,t^b)\right|^2\\
&+\left|u_{\text{pred}}(L,t^b;\theta)-\psi(L,t^b)\right|^2\Big),
\end{aligned}\tag{34}$$

$$\begin{aligned}
Loss_f=\ &\frac{1}{N_f}\sum_{i=1}^{N_f}\Big|\alpha\frac{\partial u_{\text{pred}}\left(x^f,t^f;\theta\right)}{\partial t}\\
&+(1-\alpha)[u_{\text{pred}}(x^f,t^f;\theta)-f(x^f,0)]\\
&-\beta\frac{\partial^2 u_{\text{pred}}(x^f,t^f;\theta)}{\partial x^2}\\
&+\gamma\frac{\partial u_{\text{pred}}(x^f,t^f;\theta)}{\partial x}-h(x^f,t^f)\Big|^2.
\end{aligned}\tag{35}$$

Unlike the fPINNs that adopt derivative discretization, LT-PINNs do not require grid generation. As pointed out above, the derivative discretization brings about extra calculation costs. Finally, we give the network structure diagram for solving PDEs with a Caputo-type fractional derivative using LT-PINNs (Fig. 1). Specifically, the solution process can be divided into four parts: (i) approximation of a fractional order derivative to an integer-order derivative using the Laplace transform; (ii) input x and t and utilize a DNN to solve u; (iii) using the AD technique, the equation and initial-boundary conditions are embedded into the loss function of the DNNs; (iv) optimize the parameters of the DNN by employing Adam[58] and L-BFGS[59] techniques.

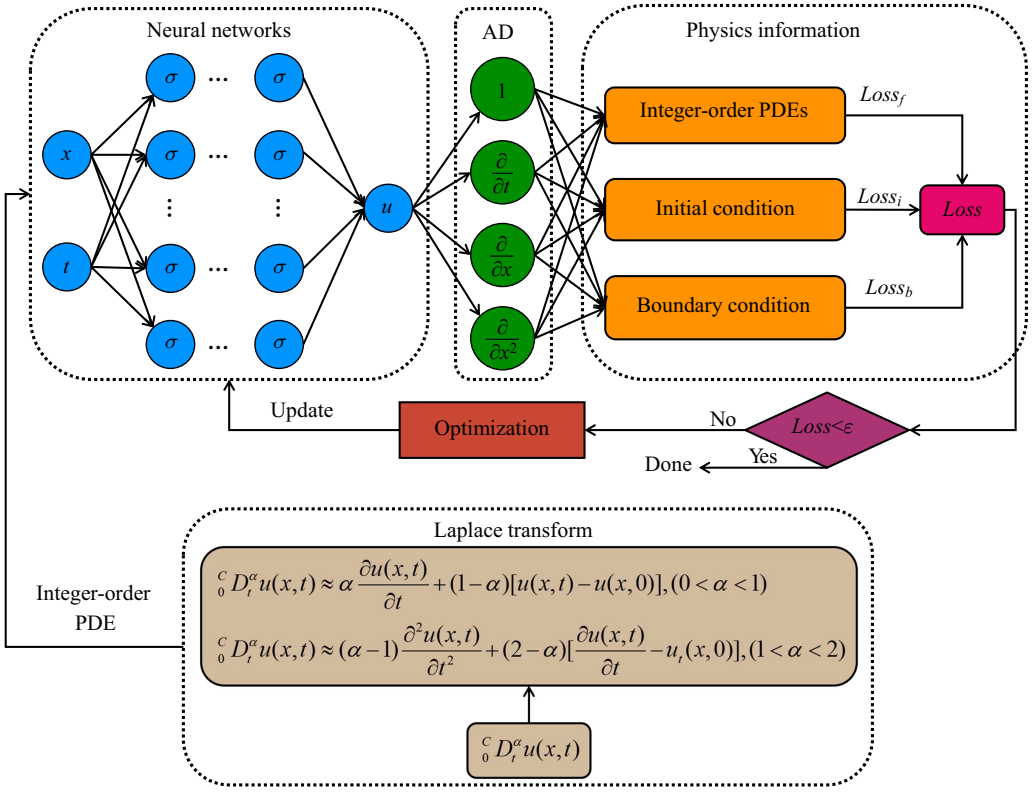


Fig. 1. The framework for solving PDEs with a Caputo-type fractional derivative using LT-PINNs.

4. Numerical experiments

In this section, we demonstrate the performance of LT-PINNs in solving various Caputo-type fractional PDEs. The five forward problems of α in two different cases will be discussed in detail in Section 4.1. In Section 4.2, the solutions to inverse problems with measurement data are given. The effects of network structure, the number of sampling points and noisy data for the LT-PINN method are discussed in Section 4.3. To evaluate the accuracy of the solution, we calculate the relative L_2 error between the solution by LT-PINNs and the exact solution. The relative L_2 error is obtained by

$$L_2 = \frac{\sqrt{\sum_i \left| u^i_{\text{pred}} - u^i_{\text{exact}} \right|^2}}{\sqrt{\sum_i \left| u^i_{\text{exact}} \right|^2}}, \tag{36}$$

where u_{pred} and u_{exact} are approximate and exact solutions, respectively. i denotes the ith test point.

In addition, for all numerical simulations we chose a neural network with four hidden layers and 40 neurons in each layer. The network weights are initialized using the Xavier algorithm. The loss function is simply optimized using the Adam and L-BFGS algorithms. The activation function is a hyperbolic tangent. In the forward and inverse problems, α is set to 0.1 and 1.1 in the cases $0 < \alpha < 1$ and $1 < \alpha < 2$, respectively. All code in this paper is based on Python 3.7, TensorFlow 1.15.0 CPU and all numerical experiments are run on a computer with an Intel(R) Core(TM) i7-14650HX 2.20 GHz processor and 32 GB memory.

4.1. Forward problems

Case 1 $0 < \alpha < 1$

Example 1 Consider

$$\begin{aligned} {}_0^{\text{C}}D_t^{\alpha} u(x,t) &= u_{xx}(x,t) + \left(\frac{6}{\Gamma(6-\alpha)} t^{3-\alpha} + t^3 + 1 \right) \sin x, \\ & 0 < x < \pi, \quad t > 0, \\ u(x,0) &= \sin x, \quad 0 < x < \pi, \\ u(0,t) &= 0, \quad u(\pi,t) = 0, \quad t > 0, \end{aligned} \tag{37}$$

where the analytical solution is $u(x,t) = (t^3+1)\sin(x)$. Replacing the time fractional order derivative ${}_0^{\text{C}}D_t^{\alpha} u(x,t)$ with Eq. (15), Eq. (37) can be re-expressed as

$$\begin{aligned} & \alpha \frac{\partial u(x,t)}{\partial t} + (1-\alpha)[u(x,t) - \sin x] \\ &= u_{xx}(x,t) + \left(\frac{6}{\Gamma(6-\alpha)} t^{3-\alpha} + t^3 + 1 \right) \sin x, \\ & 0 < x < \pi, \quad t > 0, \\ & u(x,0) = \sin x, \quad 0 < x < \pi, \\ & u(0,t) = 0, \quad u(\pi,t) = 0, \quad t > 0. \end{aligned} \tag{38}$$

We obtain the training data from the analytical solution. Specifically, dividing space $(0,\pi)$ into 512 points and time $(0,5)$ into 512 points, $u(x,t)$ is discretized into 512 snapshots accordingly. First, we randomly sample 100 points from the initial and boundary conditions, that is $N_i = 100$ and $N_b = 100$, respectively. Then, the Latin hypercube sampling (LHS)[60] strategy is used to select $N_f = 5000$ points from the spatio-temporal region. We minimize the loss function (32) by using a 20000-step Adam and L-BFGS. After giving the training dataset, $u(x,t)$ is successfully learned by optimizing the parameters of the neural networks.

In Figs. 2(a)–2(c), the three-dimensional motion and density diagrams of $u(x,t)$ are plotted for the exact solution, fPINN and LT-PINN methods, respectively. We find that the LT-PINN method can accurately and quickly predict the solution $u(x,t)$, while the fPINN method cannot. The relative L_2 error of $u(x,t)$ in the LT-PINN method is 4.857×10^{-3}, which is two orders of magnitude lower than the prediction error of the fPINN method. A comparison of the accuracy and speed of predicting $u(x,t)$ for all examples under fPINN and LT-PINN methods is given in Table 1, and the experimental results consistently show the advantages of the LT-PINN method. Figure 2(d) compares the exact solution and the predicted solution obtained using the fPINN and LT-PINN methods at different times $t = 1.96$, 3.91 and 4.89. The results show that the predictions of the LT-PINN method match well with the evolution described by the exact solution.

Table 1. Comparison of the error and speed of the fPINN and LT-PINN methods for predicting the solution $u(x,t)$ for all examples in the forward problem.

Example	Method	Error of $u(x,t)$	Speed (s)
1	fPINNs	7.330×10^{-1}	324.083
	LT-PINNs	4.857×10^{-3}	289.837
2	fPINNs	8.052×10^{-2}	286.342
	LT-PINNs	1.222×10^{-2}	273.442
3	fPINNs	9.016×10^{-2}	318.234
	LT-PINNs	3.375×10^{-3}	292.541
4	fPINNs	5.939×10^{-1}	389.335
	LT-PINNs (linear interpolation)	7.785×10^{-2}	315.801
	LT-PINNs (quadratic interpolation)	1.717×10^{-3}	347.634
5	fPINNs	5.969×10^{-2}	348.892
	LT-PINNs (linear interpolation)	4.755×10^{-3}	263.703
	LT-PINNs (quadratic interpolation)	2.525×10^{-3}	269.505

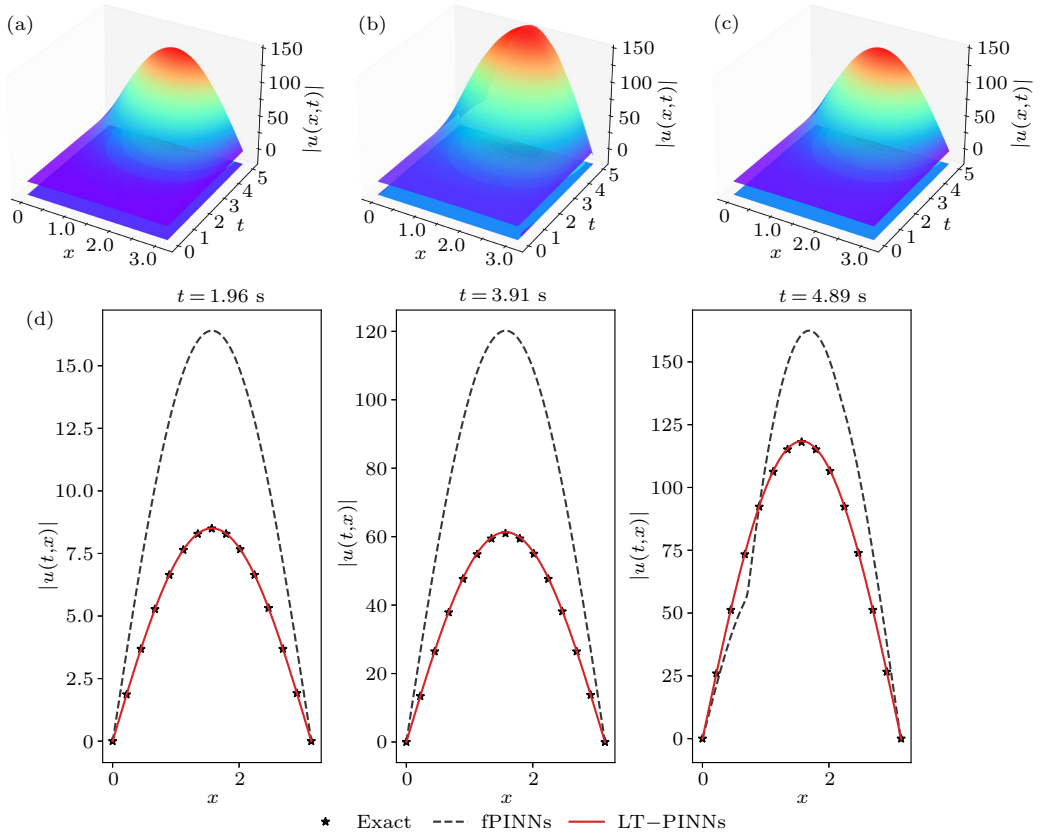


Fig. 2. Comparison between the exact solution, the fPINN approximate solution and the LT-PINN approximate solution of Eq. (37). The top panel shows the exact solution, the fPINN approximate solution and the LT-PINN approximate solution, respectively. The bottom panel shows a comparison of the fPINN and LT-PINN approximate solutions with the exact solution at different times.

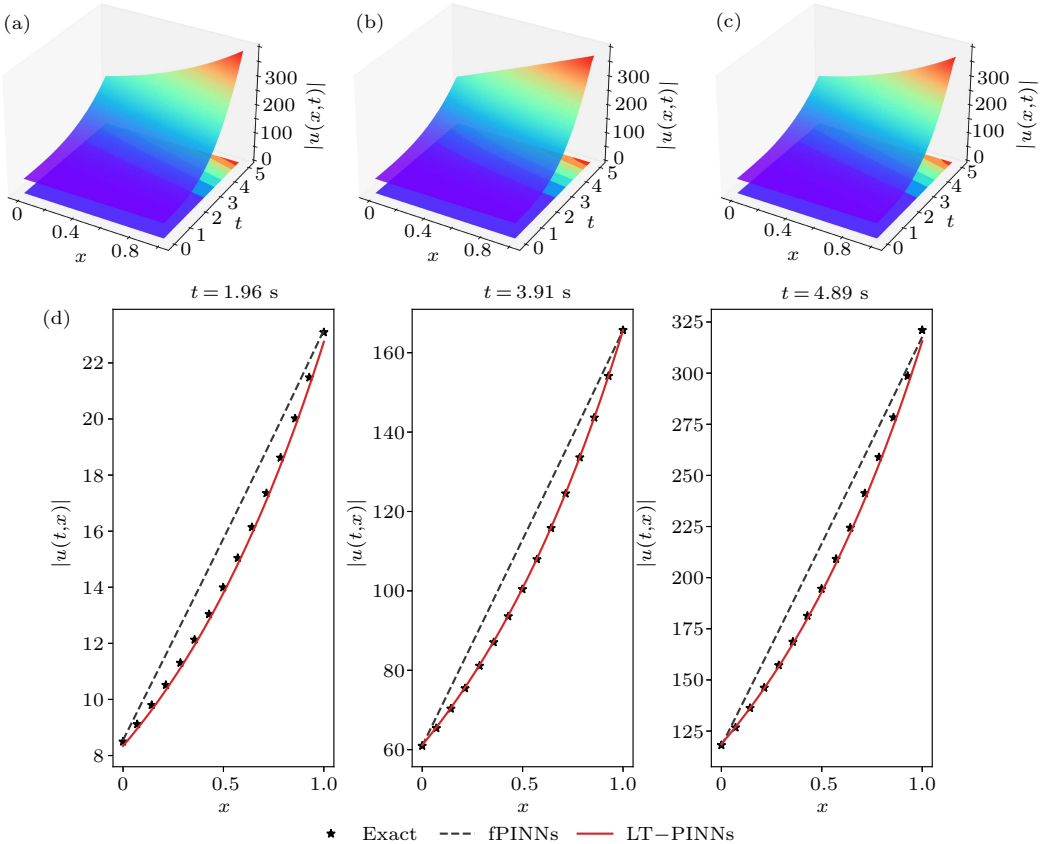


Fig. 3. Comparison between the exact solution, the fPINN approximate solution and the LT-PINN approximate solution of Eq. (39). The top panel shows the exact solution, the fPINN approximate solution and the LT-PINN approximate solution, respectively. The bottom panel compares the fPINN and LT-PINN approximate solutions with the exact solution at different times.

Example 2 Consider

$$\begin{aligned} {}_0^{\mathrm{C}}D_t^{\alpha}u(x,t) &= u_{xx}(x,t)+\left(\frac{6}{\Gamma(4-\alpha)}t^{3-\alpha}-t^3-1\right)\mathrm{e}^x, \\ &\quad 0<x<1,\quad t>0, \\ u(x,0) &= \mathrm{e}^x,\quad 0<x<1, \\ u(0,t) &= t^3+1,\quad u(1,t)=(t^3+1)\mathrm{e},\quad t>0, \end{aligned} \tag{39}$$

where the analytical solution is $u(x,t)=(t^3+1)\mathrm{e}^x$. Using Eq. (15) instead of the time fractional order derivative ${}_0^{\mathrm{C}}D_t^{\alpha}u(x,t)$, Eq. (39) can be re-expressed as

$$\begin{aligned} &\alpha\frac{\partial u(x,t)}{\partial t}+(1-\alpha)[u(x,t)-\mathrm{e}^x] \\ &= u_{xx}(x,t)+\left(\frac{6}{\Gamma(4-\alpha)}t^{3-\alpha}-t^3-1\right)\mathrm{e}^x, \\ &\quad 0<x<1,\quad t>0, \\ &\quad u(x,0)=\mathrm{e}^x,\quad 0<x<1, \\ &\quad u(0,t)=t^3+1,\quad u(1,t)=(t^3+1)\mathrm{e},\quad t>0. \end{aligned} \tag{40}$$

Next, we divide time $(0,5)$ into 512 points and space $(0,1)$ into 512 points. The analytical solution $u(x,t)$ is discretized into 512 snapshots accordingly. The sampling number of the initial boundary conditions and solution region, and the training number, is the same as in Example 1. Based on the above selection, we use the fPINN and LT-PINN methods to numerically simulate the solution of Eq. (39). Figures 3(a)–3(c) show a comparison of the predicted solution obtained using the fPINN and LT-PINN methods with the exact solution, and we find that the LT-PINN method can accurately simulate the evolution of the solution $u(x,t)$ with a relative L_2 error of 1.222×10^{-2} and a training time of 273.442 s. In Fig. 3(d), we show a comparison between the exact and predicted solutions of the solution $u(x,t)$ at different times $t=1.96$, 3.91 and 4.89. Obviously, the predicted solution with the LT-PINN method is very close to the exact solution.

Example 3 Consider

$$\begin{aligned} {}_0^{\mathrm{C}}D_t^{\alpha}u(x,t) &= u_{xx}(x,t)-2u_x(x,t) \\ &\quad +\frac{\sin(\pi x)}{\Gamma(4-\alpha)}\left(\frac{3}{3-\alpha}-\frac{6}{2-\alpha}+\frac{3}{1-\alpha}\right)t^{3-\alpha} \\ &\quad +2\pi(1+t^3)\cos(\pi x)+\pi^2(1+t^3)\sin(\pi x), \\ &\quad 0<x<1,\quad t>0, \\ u(x,0) &= \sin(\pi x),\quad 0<x<1, \\ u(0,t) &= 0,\quad u(1,t)=0,\quad t>0, \end{aligned} \tag{41}$$

where the analytical solution is $u(x,t)=(t^3+1)\sin(\pi x)$. Using Eq. (15) to replace the time fractional order derivative ${}_0^{\mathrm{C}}D_t^{\alpha}u(x,t)$, Eq. (41) can be re-expressed as

$$\begin{aligned} &\alpha\frac{\partial u(x,t)}{\partial t}+(1-\alpha)[u(x,t)-\sin(\pi x)] \\ &= u_{xx}(x,t)-2u_x(x,t) \\ &\quad +\frac{\sin(\pi x)}{\Gamma(4-\alpha)}\left(\frac{3}{3-\alpha}-\frac{6}{2-\alpha}+\frac{3}{1-\alpha}\right)t^{3-\alpha} \\ &\quad +2\pi(1+t^3)\cos(\pi x)+\pi^2(1+t^3)\sin(\pi x), \\ &\quad 0<x<1,\quad 0<t<1, \\ &\quad u(x,0)=\sin(\pi x),\quad 0<x<1, \\ &\quad u(0,t)=0,\quad u(1,t)=0,\quad 0<t<1. \end{aligned} \tag{42}$$

In this example, we have selected 512 points in the time domain $(0,5)$ and space domain $(0,1)$, respectively. Similarly, the solution $u(x,t)$ is discretized into 512 snapshots. N_i, N_b and N_f are the same as in Example 1. We predict the solution $u(x,t)$ using the fPINN and LT-PINN methods. In Figs. 4(a)–4(c) we show the ability to predict the solution $u(x,t)$ using the two methods. The results show that the LT-PINN method can accurately predict the solution of Eq. (41) and the relative L_2 error is 3.375×10^{-3}. In addition, we find that fPINNs have difficulty in predicting the maximum value when the diffusion is long. In Fig. 4(d), we compare the exact and predicted solution for the two methods at different times $t=1.96$, 3.91 and 4.89. It is clear that the predicted solution of the LT-PINN method is very close to the exact solution.

Case 2 $1<\alpha<2$

Example 4 Consider

$$\begin{aligned} {}_0^{\mathrm{C}}D_t^{\alpha}u(x,t) &= u_{xx}(x,t)+\left(\frac{6}{\Gamma(4-\alpha)}t^{3-\alpha}+t^3+1\right)\sin x, \\ &\quad 0<x<\pi,\quad t>0, \\ u(x,0) &= \sin x,\quad u_t(x,0)=0,\quad 0<x<\pi, \\ u(0,t) &= 0,\quad u(\pi,t)=0,\quad t>0, \end{aligned} \tag{43}$$

where the analytical solution is $u(x,t)=(t^3+1)\sin(x)$. We replace the time fractional order derivative ${}_0^{\mathrm{C}}D_t^{\alpha}u(x,t)$ with Eq. (25) or Eq. (29).

Next, we obtain the training data from the analytical solution. Specifically, we have selected 512 points in the spatio-temporal region $(0,\pi)\times(0,5)$. The solution $u(x,t)$ is discretized into 512 snapshots. N_i, N_b and N_f are the same as in Example 1. Given the training dataset, we optimize the parameters of the neural network by three methods: fPINNs, linear interpolation LT-PINNs and quadratic interpolation LT-PINNs, respectively. In Figs. 5(a)–5(d), we show the ability to predict the solution $u(x,t)$ using the three methods. The results show that the quadratic interpolation LT-PINNs method can accurately predict the solution of Eq. (43). In addition, the numerical results show that the quadratic interpolation LT-PINNs method is one and two orders of magnitude lower than the fPINNs and linear interpolation LT-PINNs methods, and the time of the LT-PINN method is overall shorter than that of the fPINN method. However, the quadratic interpolation LT-PINN method is slower than the linear interpolation LT-PINN method due to the fact that the values of the three nodes need to be computed when calculating the quadratic interpolation. Figure 5(e) compares the exact and predicted solutions at different times $t=1.96$, 3.91 and 4.89. It is clear that the predicted solution of the LT-PINN method is very close to the exact solution, especially during the later stages of diffusion, which are more accurately predicted by the quadratic interpolation LT-PINN method.

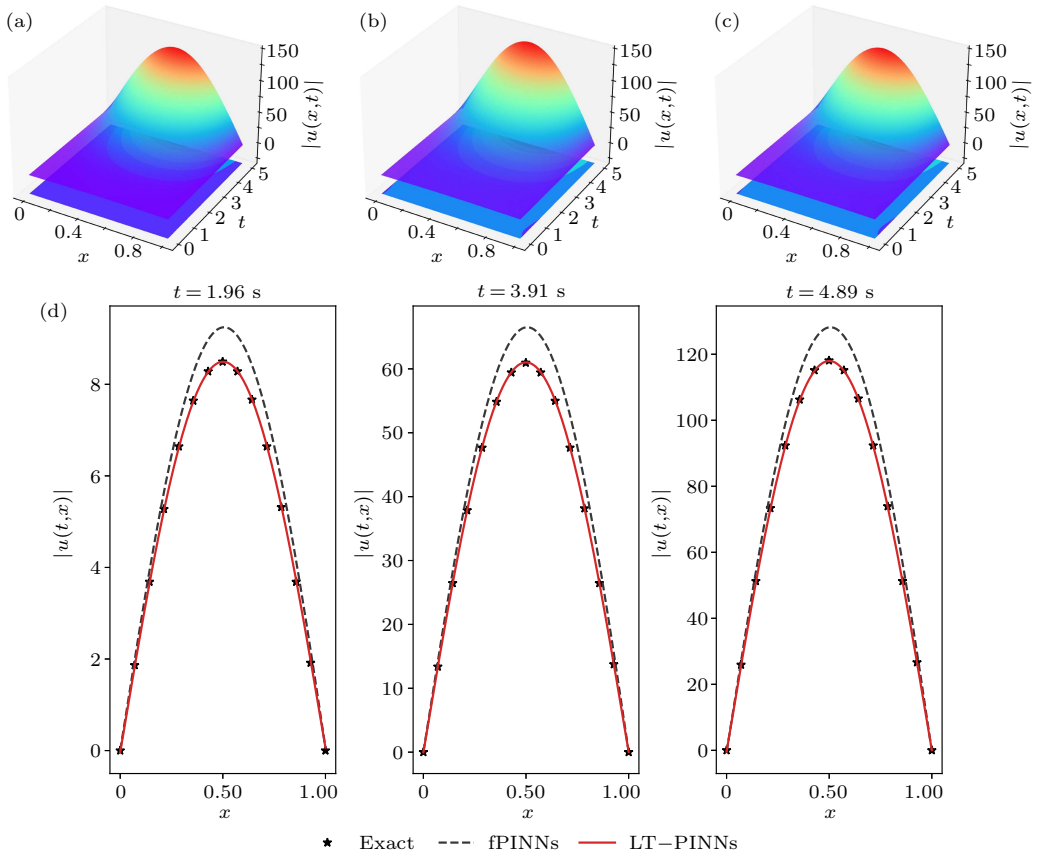


Fig. 4. Comparison between the exact solution, the fPINN approximate solution and the LT-PINN approximate solution of Eq. (41). The top panel shows the exact solution, the fPINN approximate solution and the LT-PINN approximate solution, respectively. The bottom panel compares the fPINN and LT-PINN approximate solutions with the exact solution at different times.

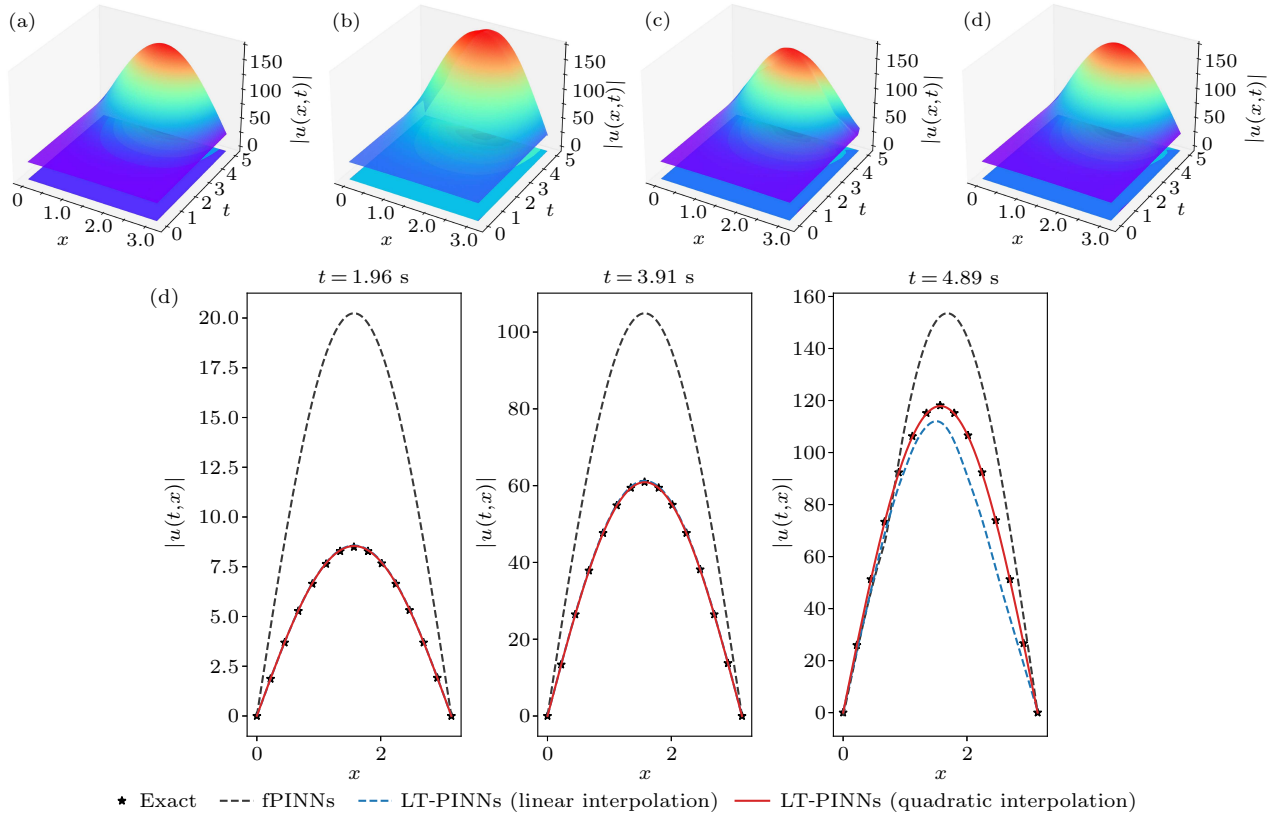


Fig. 5. Comparison between the exact solution, the fPINN approximate solution and the LT-PINN approximate solution of Eq. (43). The top panel shows the exact solution, the fPINN approximate solution, the LT-PINN (linear interpolation) approximate solution and the LT-PINN (quadratic interpolation) approximate solution, respectively. The bottom panel compares the fPINN and LT-PINN approximate solutions with the exact solution at different times.

Example 5 Consider

$$\begin{aligned} {}_0^{\mathrm{C}}D_t^{\alpha}u(x,t) &= u_{xx}(x,t)+\left(\frac{6}{\Gamma(4-\alpha)}t^{3-\alpha}+t^3\right)\cos x-\mathrm{e}^x, \\ & \quad 0<x<1, \quad t>0, \\ u(x,0) &= \mathrm{e}^x, \quad u_t(x,0)=0, \quad 0<x<1, \\ u(0,t) &= t^3+1, \quad u(1,t)=t^3\cos 1+\mathrm{e}, \quad t>0, \end{aligned} \tag{44}$$

where the analytical solution is $u(x,t)=t^3\cos(x)+\mathrm{e}^x$. Using Eq. (25) or Eq. (29) instead of the time fractional order derivative ${}_0^{\mathrm{C}}D_t^{\alpha}u(x,t)$.

In this example, we have selected 512 points in the spatio-temporal region $(0,1)\times(0,5)$. The solution $u(x,t)$ is discretized into 512 snapshots. N_i, N_b and N_f are the same as Example 1. Based on the above selection, we use fPINN and LT-PINN methods to numerically simulate the solution of Eq. (44). Figures 6(a)–6(d) show a comparison between the predicted solution and the exact solution with the help of the fPINN and LT-PINN methods, and we find that the quadratic interpolation LT-PINN method can simulate the evolution of solution $u(x,t)$; the relative L_2 error is 2.525×10^{-3}. Figure 6(e) compares the exact and predicted solutions of the solution $u(x,t)$ at different times $t=1.96$, 3.91 and 4.89; the predicted solution of the quadratic interpolation LT-PINN method is very close to the exact solution.

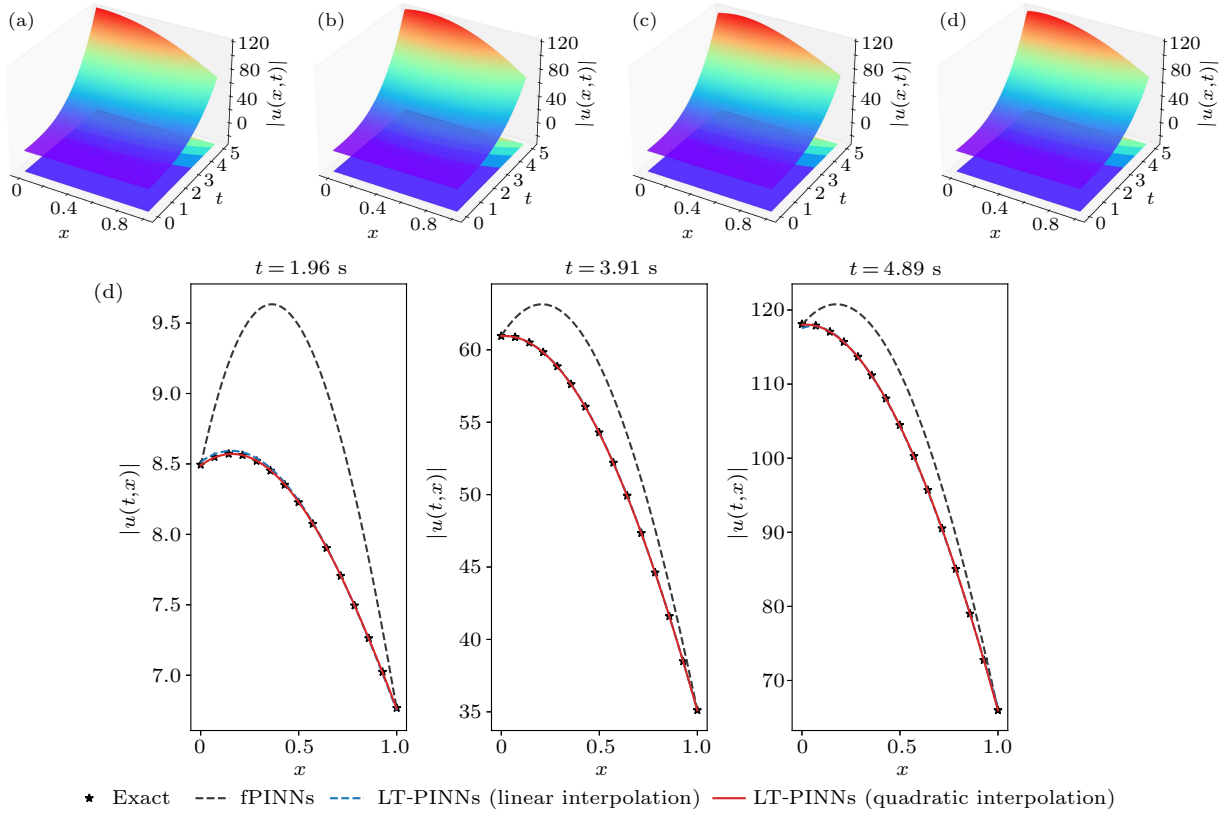


Fig. 6. Comparison between the exact solution, the fPINN approximate solution and the LT-PINN approximate solution of Eq. (44). The top panel shows the exact solution, the fPINN approximate solution, the LT-PINN (linear interpolation) approximate solution and the LT-PINN (quadratic interpolation) approximate solution, respectively. The bottom panel compares the fPINN and LT-PINN approximate solutions with the exact solution at different times.

4.2. Inverse problems

In this section we will show the capability of the LT-PINN framework to pursue the inverse problem of Caputo-type fractional PDEs. In an inverse problem, usually all we know is the control equation and discrete measured data. By using LT-PINNs we can solve the inverse problem with almost the same coding as the forward problem. When writing the code for the inverse problem, only the code for the forward problem needs to be fine-tuned. Specifically, we just need to add the parameters to be identified in the inverse problem to the list of parameters to be optimized in the forward problem without changing anything else. Here, we illustrate with three examples the ability of the LT-PINN method to identify the fractional order, convection and diffusion coefficients of Eq. (1).

To begin with, we use the fPINN and LT-PINN methods to identify the diffusion coefficient of Caputo-type fractional PDEs, writing Example 1 as a parametric equation, as follows:

$$\begin{aligned} {}_0^{\mathrm{C}}D_t^{\alpha}u(x,t) &= \beta u_{xx}(x,t) \\ &\quad +\left(\frac{6}{\Gamma(6-\alpha)}t^{3-\alpha}+t^3+1\right)\sin x, \end{aligned} \tag{45}$$

where β is the diffusion coefficient and is set to 1. The LHS strategy is used to randomly select 2000 data points from the spatio-temporal region $(x,t)\in(0,\pi)\times(0,5)$ for parameter discovery. The rest of the settings are the same as in Example

1. Table 2 shows the details of the identifying diffusion coefficient β in Eq. (45) using the fPINN and LT-PINN methods. The results show that the final identified diffusion coefficient β for the LT-PINN method is 1.0090 and the relative L_2 error for $u(x,t)$ is 1.227×10^{-3}, while the fPINN method clearly gives wrong results. Compared with the fPINN method, the LT-PINN method has a smaller error, showing that it is closer to the exact solution in estimating the parameters and results.

Table 2. Parameter discovery and error comparison of β by fPINN and LT-PINN methods.

Solution	β	Error of β	Error of u
Exact	1	0	0
fPINN	1.9662	9.662×10^{-1}	3.009×10^{-3}
LT-PINN	1.0090	9.041×10^{-3}	1.227×10^{-3}

Next, we consider the identifying diffusion and convection coefficients for the Caputo-type fractional PDEs, and write Example 3 as a parametric equation as follows:

$$\begin{aligned} {}_0^{\mathrm{C}}D_t^{\alpha}u(x,t) &= \beta u_{xx}(x,t)-\gamma u_x(x,t)+\frac{\sin(\pi x)}{\Gamma(4-\alpha)} \\ &\quad\times\left(\frac{3}{3-\alpha}-\frac{6}{2-\alpha}+\frac{3}{1-\alpha}\right)t^{3-\alpha}, \quad (46) \end{aligned}$$

where β is the diffusion coefficient and γ is the convection coefficient. We assume $\beta=1$ and $\gamma=2$. At the same time, the LHS strategy is used to randomly select 2000 data points from the spatio-temporal region $(x,t)\in(0,1)\times(0,5)$ for parameter discovery, and all the experimental settings are the same as in Example 1. Table 3 illustrates the details of the identifying diffusion coefficient β and convection coefficient γ in Eq. (46) using the fPINN and LT-PINN methods. We find that the difference between the two methods is insignificant in identifying the convection coefficient γ, while in identifying the diffusion coefficient β the LT-PINN method is 1.0016 and the relative L_2 error of $u(x,t)$ is 3.561×10^{-3}, which is two orders of magnitude lower than that of the fPINN method.

Table 3. Parameter discovery and error comparison of β and γ by the fPINN and LT-PINN methods.

Method	β	Error of β	γ	Error of γ	Error of u
Exact	1	0	2	0	0
fPINN	1.1000	1.000×10^{-1}	2.0142	7.109×10^{-3}	3.879×10^{-3}
LT-PINN	1.0016	1.659×10^{-3}	1.9903	4.808×10^{-3}	3.561×10^{-3}

Finally, we identify a critical parameter that is of fractional order α. Consider

$$\begin{aligned} {}_0^{\mathrm{C}}D_t^{\alpha}u(x,t) &= u_{xx}(x,t)-u_x(x,t)+\frac{\Gamma(4+\alpha)t^3}{6}\sin(\pi x) \\ &\quad+\pi^2t^{3+\alpha}\sin(\pi x)+\pi t^{3+\alpha}\cos(\pi x), \\ &\quad 0<x<1,\quad t>0, \\ u(x,0) &= 0,\quad 0<x<1, \\ u(0,t) &= 0,\quad u(1,t)=0,\quad t>0, \end{aligned} \quad (47)$$

where the analytical solution is $u(x,t)=t^{3+\alpha}\sin(\pi x)$. At this point, we set the parameter α to take on different values, i.e., 0.1, 0.3, 0.5, 0.7 and 0.9. Next, we use the LHS method to randomly select 2000 data points from the spatio-temporal region defined between 0 and 1 for the x-axis and 0 and 5 for the t-axis for parameter estimation. In addition, all other experimental conditions are consistent with Example 1. We summarize the results of predicting different fractional orders α using the fPINN and LT-PINN methods in Table 4. We can observe that both the fractional order α and the concentration field $u(x,t)$ are well recovered under the LT-PINN method, and are two orders of magnitude lower than the fPINN method.

Table 4. Comparison of the identification ability of the fPINN and LT-PINN methods at different fractional orders α.

Method	α_{true}	$\alpha_{\text{prediction}}$	Error of α	Error of u
fPINN	0.1	0.0356	6.437×10^{-1}	5.781×10^{-3}
LT-PINN	0.1	0.0988	1.127×10^{-2}	2.902×10^{-3}
fPINN	0.3	0.2373	2.089×10^{-1}	8.014×10^{-3}
LT-PINN	0.3	0.2983	5.572×10^{-3}	2.829×10^{-3}
fPINN	0.5	0.4396	1.207×10^{-1}	7.851×10^{-3}
LT-PINN	0.5	0.4978	4.338×10^{-3}	2.959×10^{-3}
fPINN	0.7	0.6414	8.361×10^{-2}	1.082×10^{-2}
LT-PINN	0.7	0.6965	4.908×10^{-3}	9.621×10^{-3}
fPINN	0.9	0.8412	6.529×10^{-2}	1.247×10^{-2}
LT-PINN	0.9	0.8995	4.865×10^{-4}	8.141×10^{-3}

4.3. The stability of LT-PINNs

In this section, taking Eq. (44) as an example we discuss the effects of network structure, the number of sampling points and noise data on the LT-PINN method. To evaluate the performance of the proposed LT-PINN we study its prediction accuracy under different numbers of sampling points and different network structures. We assume that $N_{ib}=200$ and $N_f=2000$ remain constants.

Table 5 shows the effect of the number of network layers and neurons on the performance of the LT-PINN method. The results show that the influences of the number of network layers and neurons on the relative L_2 error are not obvious. Similarly, in Table 6 we give the effects of the number of initial boundary condition sampling points N_{ib} and spatio-temporal region sampling points N_f for the LT-PINN method in the case of a four-layer neural network and 40 neurons. The results show that, overall, when N_f is fixed the larger the number of N_{ib} the smaller the relative L_2 error. When N_{ib} is fixed, the error does not decrease with increase in N_f. To sum up, we can conclude that the relative L_2 error is mainly determined by the initial condition sampling points N_{ib}.

Table 5. The solution of Eq. (44) using the LT-PINN method: relative L_2 error of $u(x,t)$ for different numbers of hidden layers and neurons in each layer.

Layers \ Neurons	10	20	30	40
4	2.930×10^{-3}	2.966×10^{-3}	2.939×10^{-3}	2.947×10^{-3}
5	3.091×10^{-3}	2.934×10^{-3}	2.946×10^{-3}	2.912×10^{-3}
6	2.937×10^{-3}	2.929×10^{-3}	2.928×10^{-3}	2.949×10^{-3}
7	2.964×10^{-3}	2.928×10^{-3}	2.917×10^{-3}	2.927×10^{-3}

Table 6. The solution of Eq. (44) using the LT-PINN method: relative L_2 error of $u(x,t)$ for different numbers of N_{ib} and N_f.

N_{ib} \ N_f	500	1000	1500	2000
50	2.941×10^{-3}	2.955×10^{-3}	2.948×10^{-3}	2.963×10^{-3}
100	2.948×10^{-3}	2.954×10^{-3}	2.994×10^{-3}	2.947×10^{-3}
150	2.940×10^{-3}	2.936×10^{-3}	2.946×10^{-3}	2.930×10^{-3}
200	2.932×10^{-3}	2.927×10^{-3}	2.940×10^{-3}	2.935×10^{-3}

In practical engineering applications, the data obtained by sampling are inevitably noisy, so we add various degrees of noise (5%, 10% Gaussian noise) to the training data to verify the robustness of the LT-PINN method. Based on the different noisy data, we simulate the solution of Eq. (44) using the LT-PINN method. In Figs. 7 and 8, we show a series of experimental results for solving Eq. (44) using the LT-PINN method under various noises. We find that the LT-PINN method can predict the solution accurately. Therefore, the LT-PINN method has better robustness for noisy data.

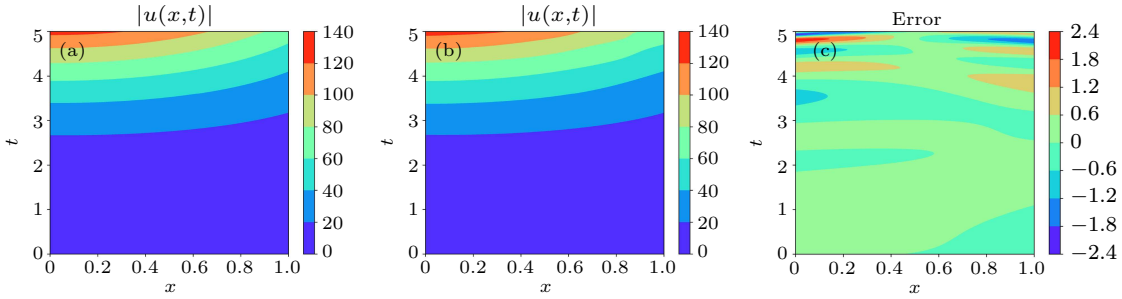


Fig. 7. Density and error diagrams of the LT-PINN method for predicting Eq. (44) when the noise is 5%: (a) exact solution, (b) predicted solution with the LT-PINN method, (c) error of the LT-PINN method.

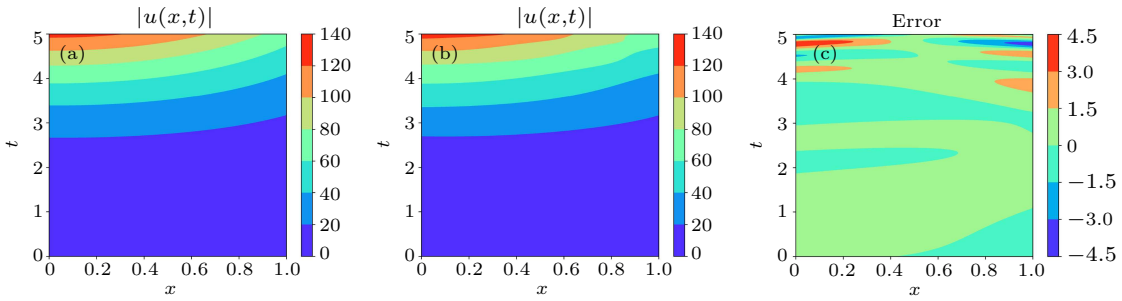


Fig. 8. Density and error diagrams of the LT-PINN method for predicting Eq. (44) when the noise is 10%: (a) exact solution, (b) predicted solution with the LT-PINN method, (c) error of the LT-PINN method.

5. Conclusion

In the field of scientific computing, the problem of solving fractional PDEs has always been challenging, especially the forward and inverse problems of Caputo-type fractional PDEs. Traditional numerical methods face many difficulties in dealing with this problem, such as the complex discretization of fractional derivatives, the strong dependence on grids, and a limited ability to deal with noisy data problems. It is difficult to achieve an ideal balance between calculation accuracy and efficiency. To this end, we innovatively propose the LT-PINN framework, which opens up a new path for the solution of Caputo-type fractional PDEs. This method skillfully uses the Laplace transform to transform the original fractional PDEs into integer-order PDEs so as to make full use of the automatic differentiation advantages of neural networks. In this transformation process, the original complex fractional derivative is skillfully integrated into the transformed loss function so that the neural network can learn and approximate the solution of the equation more efficiently.

Another significant advantage of the LT-PINN framework is its meshless nature. Unlike traditional methods that rely on fixed grids or nodes, LT-PINNs are highly flexible and can directly output solutions at any point in the solution area without the need for interpolation. This not only improves the accuracy of the solution but also greatly expands the application potential of the method in complex geometry and irregular region problems. In addition, LT-PINNs perform well in terms of computational efficiency. Since the conversion process between Caputo-type fractional PDEs and integer-order PDEs is scheduled to be completed in the offline phase, the online solution phase is extremely fast, which is of great value for real-time simulation scenarios that require a timely response.

Through experimental verification of Caputo-type fractional PDEs in two different cases of $0<\alpha<1$ and $1<\alpha<2$, LT-PINNs show excellent performance. The obtained numerical solution is highly consistent with the analytical solution, and is significantly better in comparison with the fPINN method. LT-PINNs also show a strong ability to solve inverse problems, and can accurately identify key parameters such as fractional order, convection coefficient and diffusion coefficient. We also deeply analyze the influence of network struc-

ture and the number of sampling points on the performance of the method, which provides an important reference for optimizing the LT-PINN method.

However, LT-PINNs also have limitations. At present, the framework mainly focuses on Caputo-type fractional PDEs in a specific fractional range, and it still faces challenges in the extended application of more complex fractional derivative types or coupled multi-physics fractional problems. In future research work, we will explore a hybrid method combining LT-PINNs with other numerical methods, which is expected to further improve the efficiency and accuracy of solving complex fractional PDE systems. In addition, we will strive to extend the LT-PINN method to a wider range of fractional PDE types and continuously expand its application boundaries.

Acknowledgements

This study was funded by the National Natural Science Foundation of China (Grant No. 12061055), the Key Projects of the Natural Science Foundation of Ningxia Hui Autonomous Region of China (Grant No. 2022AAC02005), and the Scientific and Technological Innovation Leading Talent Project of Ningxia Hui Autonomous Region of China (Grant No. 2021GKLRLX06).

References

[1] Colbrook M and Ayton L 2022 *J. Comput. Phys.* **454** 110995
[2] Lei D, Sun H, Zhang Y, Blaszczyk T and Yu Z 2023 *J. Hydrol.* **627** 130280
[3] Chen L, Bajdich M, Martirez J, Krauter C, Gauthier J, Carter E, Luntz A, Chan K and Nørskov J 2018 *Nat. Commun.* **9** 3202
[4] Fendzi-Donfack E, Temgoua G, Djoufack Z, Kenfack-Jiotsa A, Nguenang J and Nana L 2022 *Chaos, Solitons and Fractals* **160** 112253
[5] Baleanu D, Shekari P, Torkzadeh L, Ranjbar H, Jajarmi A and Nouri K 2023 *Chaos, Solitons and Fractals* **166** 112990
[6] Ngounou A, Feulefack S, Tabejieu L and Nbendjo B 2022 *Chaos, Solitons and Fractals* **157** 111952
[7] Qing J, Zhou S, Wu J and Shao M 2023 *Commun. Nonlinear Sci. Numer. Simul.* **116** 106810
[8] Hanif H 2021 *Chaos, Solitons and Fractals* **153** 111463
[9] Feng L, Liu F, Turner I and Zhuang P 2017 *Int. J. Heat Mass Tran.* **115** 1309
[10] Feng L, Liu F and Turner I 2019 *Commun. Nonlinear Sci. Numer. Simul.* **70** 354
[11] Feng L, Liu F, Turner I and Zheng L 2018 *Fract. Calc. Appl. Anal.* **21** 1073
[12] Chen X H, Yang W D, Zhang X R and Liu F W 2019 *Appl. Math. Lett.* **95** 143
[13] Sheikh N, Ching D, Khan I and Sakidin H 2022 *Sci. Rep.* **12** 14117
[14] Jani M, Babolian E and Bhatta D 2021 *Math. Method. Appl. Sci.* **44** 2021
[15] Karipova G and Magdziarz M 2017 *Chaos, Solitons and Fractals* **102** 245
[16] Kumar Y and Singh V 2021 *Math. Comput. Simulat.* **190** 531
[17] Nuugulu S, Gideon F and Patidar 2021 *Chaos, Solitons and Fractals* **145** 110753
[18] Wu H and Yang H 2026 *Commun. Nonlinear Sci. Numer. Simul.* **152** 109133
[19] Podlubny I 1998 *Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications* (Amsterdam: Elsevier) Vol. 198
[20] Sun Z and Wu X 2006 *Appl. Numer. Math.* **56** 193
[21] Lin Y and Xu C 2007 *J. Comput. Phys.* **225** 1533
[22] Zhang Y, Sun Z and Wu H 2011 *SIAM J. Numer. Anal.* **49** 2302
[23] Ren J and Sun Z 2013 *J. Sci. Comput.* **56** 381
[24] Yang Q, Turner I, Liu F and Ilić M 2011 *SIAM J. Sci. Comput.* **33** 1159
[25] Gao G, Sun Z and Zhang H 2014 *J. Comput. Phys.* **259** 33
[26] Yang C 2023 *Chaos, Solitons and Fractals* **168** 113204
[27] Xu H 2023 *Chaos, Solitons and Fractals* **166** 112917
[28] Abdelrahim M, Saiga H, Maeda N, Hossain E, Ikeda H and Bhandari P 2022 *Gut* **71** 7
[29] Cronin A, Smit J, Muñoz M, Poirier A, Moran P, Jerem P and Halfwerk W 2022 *Biol. Rev.* **97** 1325
[30] Miller D, Stern A and Burstein D 2022 *Nat. Commun.* **13** 5731
[31] Zhang R, Su J and Feng J 2024 *Evol. Intell.* **17** 1567
[32] Adams S, Cody T and Beling P 2022 *Artif. Intell. Rev.* **55** 4307
[33] Schneider J, Seidel S, Basalla M and vom Brocke J 2023 *Bus. Inform. Syst. Eng.* **65** 65
[34] Brink A, Najera-Flores D and Martinez C 2021 *Neural Comput. Appl.* **33** 5591
[35] Bar-Sinai Y, Hoyer S, Hickey J and Brenner M 2019 *Proc. Natl. Acad. Sci. USA* **116** 15344
[36] Panghal S and Kumar M 2021 *Eng. Comput.* **37** 2989
[37] Georgoulis E, Loulakis M and Tsiourvas A 2023 *Commun. Nonlinear Sci. Numer. Simul.* **117** 106893
[38] Han J, Jentzen A and Weinan E 2018 *Proc. Natl. Acad. Sci. USA* **115** 8505
[39] Raissi M, Perdikaris P and Karniadakis G 2017 arXiv:1711.10561 [cs.AI]
[40] Raissi M, Perdikaris P and Karniadakis G 2017 arXiv:1711.10566 [cs.AI]
[41] Raissi M, Perdikaris P and Karniadakis G 2019 *J. Comput. Phys.* **378** 686
[42] Baydin A, Pearlmutter B, Radul A and Siskind J 2015 arXiv:1502.05767 [cs.SC]
[43] Jiang X, Wang D, Fan Q, Zhang M, Lu C and Lau A 2022 *Laser Photonics Rev.* **16** 2100483
[44] Zhang R, Su J and Feng J 2023 *Nonlinear Dynam.* **111** 13399
[45] Zhu Q, Liu Z and Yan J 2021 *Comput. Mech.* **67** 619
[46] Goswami S, Anitescu C, Chakraborty S and Rabczuk T 2020 *Theor. Appl. Fract. Mec.* **106** 102447
[47] Chen Y, Lu L, Karniadakis G and Dal Negro L 2020 *Opt. Express* **28** 11618
[48] Bai Y, Chaolu T and Bilige S 2022 *Nonlinear Dynam.* **107** 3655
[49] Mao Z, Jagtap A and Karniadakis G 2020 *Comput. Methods Appl. Mech. Eng.* **360** 112789
[50] He Q, Barajas-Solano D, Tartakovsky G and Tartakovsky A 2020 *Adv. Water. Resour.* **141** 103610
[51] Cai S, Mao Z, Wang Z, Yin M and Karniadakis G 2021 *Acta. Mech. Sin.* **37** 1727
[52] Pang G, Lu L and Karniadakis G 2019 *SIAM J. Sci. Comput.* **41** A2603
[53] Guo L, Wu H, Yu X and Zhou T 2022 *Comput. Methods Appl. Mech. Eng.* **400** 115523
[54] Zhang T, Zhang D, Shi S and Guo Z 2025 *Nonlinear Dynam.* **113** 12565
[55] Ren J, Sun Z and Dai W 2016 *Appl. Math. Model.* **40** 2625
[56] Hornik K, Stinchcombe M and White H 1990 *Neural networks* **3** 551
[57] Kilbas A, Srivastava H and Trujillo J *Theory and applications of fractional differential equations* (Amsterdam: Elsevier) Vol. 204
[58] Kingma D and Ba J 2014 arXiv:1412.6980 [cs.ML]
[59] Liu D and Nocedal J 1989 *Math. Program.* **45** 503
[60] McKay M, Beckman R and Conover W 2000 *Technometrics* **42** 55