



Article

Numerical Investigation of Distributed-Order Cattaneo-Christov Model Based on Fractional Physics-Informed Neural Networks

Xuehui Chen ^{1,*}, Weijia Zhao ¹, Jingbo Yang ¹, Weidong Yang ² and Yang Liu ³

¹ School of Mathematics and Physics, University of Science and Technology Beijing, Beijing 100083, China; m202510829@xs.ustb.edu.cn (W.Z.); 15235140375@139.com (J.Y.)

² Natural Science Basic Experimental Center, University of Science and Technology Beijing, Beijing 100083, China; yangwd@ustb.edu.cn

³ School of Mathematical Sciences, Inner Mongolia University, Hohhot 010021, China; mathliuyang@imu.edu.cn

* Correspondence: xuehuichen@ustb.edu.cn

Abstract

A novel distributed-order Cattaneo–Christov model is proposed to effectively characterize non-classical heat conduction processes with memory effect and time–space relaxation behaviors originating from distributed-order fractional derivatives. A fractional physics-informed neural networks (fPINN) algorithm is employed to address both the forward and inverse problems of the distributed-order heat conduction model. For the forward problem, we propose an SfPINN algorithm that incorporates a squared loss term and employs an adaptive updating strategy for the loss-term weights. First, the boundary conditions are embedded into the network output such that they are automatically satisfied. In addition, we design a two-stage training strategy to enhance computational efficiency: in the first stage, the squared loss term associated with the initial condition is incorporated into the loss function; in the second stage, the squared residual term of the governing equation is introduced into the loss function. Numerical results show that the proposed algorithm outperforms the standard fPINN method in both solution accuracy and training iteration speed. For the inverse problem, the numerical results demonstrate that as the iteration number increases, the estimated parameter values progressively converge to their true values and finally stabilize.

Keywords: distributed-order; constitutive relations; Cattaneo-Christov model; neural network

Academic Editors: Kamyar Hosseini, Khadijeh Sadri, Evren Hınçal, Soheil Salahshour and Farzaneh Alizadeh

Received: 17 May 2026

Revised: 19 June 2026

Accepted: 24 June 2026

Published: 29 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Fick’s law [1] serves as the cornerstone for modeling anomalous diffusion phenomena. Despite its widespread utilization across numerous practical applications, this classical framework exhibits certain deficiencies [2,3] that necessitate further refinement. To overcome these limitations, the Cattaneo constitutive model [4,5] introduces a relaxation time coefficient. However, the Cattaneo constitutive relation only involves time derivatives, and the complete process may need to take into account the influence of spatial gradients [6]. Christov [7] modified the time derivative by introducing Oldroyd’s upper

convection derivative and proposed an extended Cattaneo model. Liu et al. [8] then further extended the research on heat conduction by using the fractional time–space Cattaneo–Christov equation. They reformulated the equation into a one-dimensional form by treating velocity as a constant and taking into account fractional derivatives of time and space:

$$q + \xi \left(\tau^{\alpha-1} \frac{\partial^\alpha q}{\partial t^\alpha} d\alpha + u \frac{\partial q}{\partial x} \right) = -k \left(\gamma \frac{\partial^\beta T(x,t)}{\partial x^\beta} - (1-\gamma) \frac{\partial^\beta T(x,t)}{\partial (-x)^\beta} \right). \quad (1)$$

Although fractional Cattaneo–Christov models are more capable of characterizing the memory effect of non-classical heat conduction, compared to integer-order models, the fixed fractional order in such models still has essential limitations. Specifically, a single constant fractional order can only describe the thermal relaxation behavior at a single characteristic scale, while non-classical heat conduction processes in complex media such as porous media, composite materials, and non-Newtonian fluids usually exhibit multiple relaxation characteristics and wide-spectrum memory effects. With such complex processes, a single fractional order encounters difficulties in simultaneously characterizing the thermal transport behaviors across different time and space scales, which will lead to a reduction in the accuracy of the model in practical applications. The distributed-order fractional derivative with weight function was first proposed by Caputo [9]. Duan and Chen [10] investigated the steady oscillatory shear flow between two parallel plates, examining the effects of different forms of order-weight functions. Their research outcomes illustrate the potential of distributed-order derivatives in capturing viscoelastic characteristics, where the order-weight functions act as a powerful instrument for describing this kind of behavior. For the purpose of exploring the dynamics of start-up pipe flow, Chen and his colleagues [11] employed a finite difference method to obtain numerical solutions corresponding to the double time-distributed order Maxwell model.

Yin et al. [12] employed the convolution quadrature method and subsequently extended its application to distributed-order problems. Morgado and Rebelo [13] proposed an implicit numerical scheme to approximate solutions for distributed-order time-fractional reaction-diffusion equations incorporating nonlinear source terms. In the field of heat conduction, distributed-order fractional models have been widely developed to characterize non-Fourier heat transfer with memory effects. A distributed-order time fractional heat conduction model was developed by Liu et al. [14], the validation of which through experimental data fitting confirmed its capability to capture non-classical heat transfer phenomena. Cao et al. [15] proposed a distributed-order fractional heat conduction model for laser-induced thermal therapy that outperformed classical and single fractional-order models in capturing biological tissue heat transfer characteristics. Khan et al. [16] extended the Cattaneo heat flux model with distributed fractional-order derivatives, expanding its application in coupled flow and heat transfer problems.

Recently, the fractional PINN (fPINN) has been proposed as an improvement to the PINN algorithm for exploring solutions to nonlinear fractional partial differential equations (FPDE). The fractional PINN (fPINN) algorithm was first proposed by Pang et al. [17]. Subsequently, Guo et al. [18] introduced the MC-FPINN algorithm, which significantly reduces the computational cost. Wang et al. [19] developed a PINN algorithm based on power series expansion for solving one-dimensional and two-dimensional space-fractional advection–diffusion equations with variable coefficients. In addition, a fast second-order spectral method was proposed based on the fractional backward difference formula (FBDF) to calculate the numerical solutions, and a new fractional physical information neural network was used for parameter estimation [20]. Wang et al. [21] introduced a Hermite neural network solver that couples the fractional-order Richards model with a fractional PINN to solve unsaturated flow problems through an attention mechanism. Pei et al. [22] proposed the Adaptive-fPINN for simultaneously recovering the fractional order

and the space-dependent zeroth-order coefficient in a subdiffusion inverse problem. Xu et al. [23] introduced fractional Physics-Informed Convolutional–Recurrent Neural Networks (fPICRNNs), which integrate an encoder–decoder convolutional–recurrent learning structure with peridynamic differential operators to solve fractional order partial differential equations.

In this study, a time–space distributed-order Cattaneo–Christov model is developed, and, inspired by Liu et al. [24], fPINN with square loss terms (SfPINN) are introduced for the first time to solve the forward and inverse problems associated with distributed-order heat conduction equations. The subsequent sections are arranged in the following manner. Section 2 introduces the distributed-order heat-conduction model. In Section 3, two numerical methods are investigated, including the standard fPINN scheme and the adaptive-weight SfPINN algorithm developed in this work. In Section 4, numerical tests are performed to validate and analyze the forward and inverse problems of the distributed-order heat conduction model. Section 5 presents the conclusion.

2. Mathematical Model

Inspired by the research on distributed-order constitutive models by Caputo [9] and Duan et al. [10], and the fractional Cattaneo–Christov model by Liu et al. [8], in order to effectively characterize non-classical heat conduction processes with memory and relaxation behaviors, a distributed-order Cattaneo–Christov model is constructed as follows:

$$q + \xi \left(\int_0^1 \tau_1^{\alpha-1} \varphi(\alpha) \frac{\partial^\alpha q}{\partial t^\alpha} d\alpha + u \frac{\partial q}{\partial x} \right) = -k \int_0^1 \tau_2^{\beta-1} \phi(\beta) \left(\gamma \frac{\partial^\beta T(x,t)}{\partial x^\beta} - (1-\gamma) \frac{\partial^\beta T(x,t)}{\partial (-x)^\beta} \right) d\beta, \quad (2)$$

where τ_1 and τ_2 are introduced to ensure dimensional consistency in the constitutive equation and ξ acts as a slack parameter associated with the time dimension. The variable u denotes the convection velocity in the x -direction, while γ (where $0 \leq \gamma \leq 1$) serves as a weight coefficient. The functions $\varphi(\alpha)$ and $\phi(\beta)$ are weight functions satisfying the normalization conditions $\int_0^1 \varphi(\alpha) d\alpha = W_\alpha$ and $\int_0^1 \phi(\beta) d\beta = W_\beta$, where W_α and W_β are constants. The symbol $\frac{\partial^\alpha}{\partial t^\alpha}$ is the Caputo fractional derivative [8], defined as

$$\frac{\partial^\alpha T(x,t)}{\partial t^\alpha} \equiv \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{1}{(t-\tau)^\alpha} \frac{\partial T(x,\tau)}{\partial \tau} d\tau, \quad (3)$$

where $\Gamma(\cdot)$ represents the Euler gamma function. The symbols $\frac{\partial^\beta}{\partial x^\beta}$ and $\frac{\partial^\beta}{\partial (-x)^\beta}$ are the left and right Riemann–Liouville fractional derivatives of order β ($0 < \beta \leq 1$), and the corresponding definitions on a finite domain $[a,b]$ are given by

$$\frac{\partial^\beta T(x,t)}{\partial x^\beta} = \frac{1}{\Gamma(1-\beta)} \frac{\partial}{\partial x} \int_a^x (x-\xi)^{-\beta} T(\xi,t) d\xi, \quad (4)$$

and

$$\frac{\partial^\beta T(x,t)}{\partial (-x)^\beta} = \frac{-1}{\Gamma(1-\beta)} \frac{\partial}{\partial x} \int_x^b (\xi-x)^{-\beta} T(\xi,t) d\xi, \quad (5)$$

Subsequently, the mass conservation equation is given, as follows [8]:

$$c\rho \frac{\partial T}{\partial t} + c\rho u \frac{\partial T}{\partial x} + \text{div} q = 0, \quad (6)$$

where c and ρ denote the specific heat capacity and mass density, respectively.

By substituting the distributed-order Cattaneo–Christov model (2) into the mass conservation equation, Equation (6), the distributed-order heat conduction equation is obtained, as

$$\begin{aligned}
& c\rho \frac{\partial T}{\partial t} + c\rho u \frac{\partial T}{\partial x} + \xi c\rho \int_0^1 \tau_1^{\alpha-1} \varphi(\alpha) \frac{\partial^{1+\alpha} T}{\partial t^{1+\alpha}} d\alpha + \xi c\rho u \int_0^1 \tau_1^{\alpha-1} \varphi(\alpha) \frac{\partial^{1+\alpha} T}{\partial x \partial t^\alpha} d\alpha + c\rho u \xi \frac{\partial^2 T}{\partial t \partial x} \\
& + c\rho u^2 \xi \frac{\partial^2 T}{\partial x^2} - k \int_0^1 \tau_2^{\beta-1} \phi(\beta) \left(\gamma \frac{\partial^{\beta+1} T(x,t)}{\partial x^{\beta+1}} + (1-\gamma) \frac{\partial^{\beta+1} T(x,t)}{\partial (-x)^{\beta+1}} \right) d\beta = 0,
\end{aligned} \quad (7)$$

with the initial and boundary conditions

$$T(x,0) = \frac{1}{L^4} x^2 (L-x)^2, \quad \frac{\partial T(x,0)}{\partial t} = 0, \quad (8)$$

$$T(0,t) = T(L,t) = 0. \quad (9)$$

The dimensionless quantities are introduced as shown below:

$$t \rightarrow \tau_1 t^*, \quad \xi \rightarrow \tau_1 \xi^*, \quad u \rightarrow \frac{\tau_2}{\tau_1} u^*, \quad x \rightarrow \tau_2 x^*, \quad T \rightarrow \tau_1 T^*, \quad \gamma \rightarrow \frac{\tau_2^2}{\tau_1 D} \gamma^*, \quad 1-\gamma \rightarrow \frac{\tau_2^2}{\tau_1 D} (1-\gamma^*),$$

where $D = k / c\rho$. Using the defined dimensionless variables, we can transform Equations (7)–(9) into the following non-dimensional forms:

$$\begin{aligned}
& \frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + \xi \int_0^1 \varphi(\alpha) \frac{\partial^{1+\alpha} T}{\partial t^{1+\alpha}} d\alpha + \xi u \int_0^1 \varphi(\alpha) \frac{\partial^{1+\alpha} T}{\partial x \partial t^\alpha} d\alpha + \xi u \frac{\partial^2 T}{\partial t \partial x} + \xi u^2 \frac{\partial^2 T}{\partial x^2} \\
& - \int_0^1 \phi(\beta) \left(\gamma \frac{\partial^{\beta+1} T(x,t)}{\partial x^{\beta+1}} + (1-\gamma) \frac{\partial^{\beta+1} T(x,t)}{\partial (-x)^{\beta+1}} \right) d\beta = 0.
\end{aligned} \quad (10)$$

The initial and boundary conditions are

$$T(x,0) = x^2 (1-x)^2, \quad \frac{\partial T(x,0)}{\partial t} = 0, \quad (11)$$

$$T(0,t) = T(1,t) = 0. \quad (12)$$

When the weight function of the distributed-order derivative is taken as the Dirac delta function $\delta(x)$, the distributed-order model degenerates into a fractional-order model. The integral of the Dirac delta function over the entire real axis equals one, that is, $\int_{-\infty}^{\infty} \delta(x) dx = 1$. By setting $\beta = 1, \xi = 1, \gamma = 1, u = 0$ and $\varphi(\alpha), \phi(\beta)$ as the Dirac delta function $\delta(x)$, Equation (10) reduces to the time fractional Cattaneo model, and when $\beta = 1, \xi = 0, \gamma = 1, u = 0$, Equation (10) reduces to the classical heat conduction model.

3. Numerical Methods

3.1. Standard fPINN

Suppose that the unknown function to be solved is $T(x,t)$, and let $\mathcal{L}(T)$ denote the partial differential equation operator, $f(x,t)$ is the source item. The governing physical equation can be written as

$$\mathcal{L}(T) = f(x,t).$$

Let $I(T)$ denote the initial condition equation operator; the initial condition is given by

$$I(T) = T_0(x).$$

The boundary conditions are defined as

$$T(0,t) = g_1(t), \quad T(L,t) = g_2(t).$$

The structure of a standard PINN can be represented as

$$T_\theta(x) = (C_m \circ \sigma \circ C_{m-1} \circ \sigma \circ \dots \circ \sigma \circ C_1)(x, t),$$

where T_θ denotes the model output used as the predicted solution, θ denotes the set of model parameters, x and t are the input vectors, $\circ$ denotes the function composition

operator, $C_1, C_2, \dots, C_m$ represent the sequentially arranged network layers, and σ denotes the activation function. Generally, the loss function of the PINN method consists of the equation residual loss, initial condition loss, and boundary condition loss, as shown below:

$$Loss(\theta) = \omega_{\text{PDE}} \text{MSE}_{\text{PDE}} + \omega_{\text{IC}} \text{MSE}_{\text{IC}} + \omega_{\text{BC}} \text{MSE}_{\text{BC}}, \quad (13)$$

where ω_{PDE} , ω_{IC} and ω_{BC} are weighting coefficients that balance the respective contributions of the PDE residual loss, initial condition loss, and boundary condition loss to the total loss function. In the subsequent training process, the weighting coefficients we selected are all equal. And

$$\text{MSE}_{\text{PDE}} = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} |\mathcal{L}(T) - f(x, t)|^2, \quad (14)$$

$$\text{MSE}_{\text{IC}} = \frac{1}{N_{\text{IC}}} \sum_{i=1}^{N_{\text{IC}}} |\hat{T}(T(x_i, 0)) - T_0(x_i, 0)|^2 \quad (15)$$

$$\text{MSE}_{\text{BC}} = \frac{1}{N_{\text{BC}}} \sum_{i=1}^{N_{\text{BC}}} |\hat{T}(0, t_i) - g_1(t_i)|^2 + \frac{1}{N_{\text{BC}}} \sum_{i=1}^{N_{\text{BC}}} |\hat{T}(L, t_i) - g_2(t_i)|^2. \quad (16)$$

The training objective is to minimize the total loss function $\min_{\theta} Loss(\theta)$. After training, the output of the well-trained neural network is used to approximate the solution of the original equation:

$$T(x, t) \approx \hat{T}(x, t; \theta^*).$$

PINN performs excellently in solving integer-order differential equations, since integer-order derivatives can be accurately computed via automatic differentiation. However, fractional-order derivatives cannot be directly evaluated using automatic differentiation. The mainstream approach is to discretize fractional-order derivatives with numerical schemes, such as the L1 formula and the Grünwald–Letnikov formula, while keeping other components consistent with the standard PINN framework. This integrated method is known as the fractional physics-informed neural network (fPINN) algorithm [17].

Define $x_i = ih$ ($i = 0, 1, 2, \dots, m$, $mh = 1$) and $t_j = j\tau$ ($j = 0, 1, 2, \dots, n$), where h is the grid size in space and τ is grid size in time, T_i^j represents the values at the mesh point (x_i, t_j) . The Caputo fractional derivative of order $0 < \alpha \leq 1$ with respect to time at $t = t_j$ is approximated by L1-approximation; the discrete scheme is given as follows [8]:

$$\frac{\partial^\alpha T(x, t_j)}{\partial t^\alpha} \approx \frac{1}{\tau^\alpha \Gamma(2-\alpha)} \left[T(x_i, t_j) + \sum_{k=1}^{j-1} (c_{j-k} - c_{j-k-1}) T(x_i, t_k) - c_{j-1} T(x_i, t_0) \right], \quad (17)$$

$$\begin{aligned} \frac{\partial^{1+\alpha} T(x, t_j)}{\partial t^{1+\alpha}} &= \frac{\partial}{\partial t} \left(\frac{\partial^\alpha T(x, t_j)}{\partial t^\alpha} \right) \\ &\approx \frac{1}{\tau^{1+\alpha} \Gamma(2-\alpha)} \left[c_0 (T(x_i, t_j) - T(x_i, t_{j-1})) \right. \\ &\quad \left. - \sum_{l=1}^{j-1} (c_{j-l-1} - c_{j-l}) (T(x_i, t_l) - T(x_i, t_{l-1})) \right]. \end{aligned} \quad (18)$$

where $c_0 = 1$, $c_k = (k+1)^{1-\alpha} - k^{1-\alpha}$, $k = 1, 2, \dots$.

The shifted Grünwald formulae are employed to approximate the space fractional derivative of order $1 < \beta \leq 2$:

$$\frac{\partial^{\beta+1} T(x_i, t_j)}{\partial x^{\beta+1}} \approx \frac{1}{h^{\beta+1}} \sum_{l=0}^{i+1} \omega_l^{\beta+1} T(x_{i-l+1}, t_j), \quad (19)$$

$$\frac{\partial^{\beta+1}T(x_i, t_j)}{\partial(-x)^{\beta+1}} \approx \frac{1}{h^{\beta+1}} \sum_{l=0}^{m-i+1} \omega_l^{\beta+1} T(x_{i+l-1}, t_j), \quad (20)$$

where the symbol $\omega_l^{\beta+1}$ refers to the Grünwald weight coefficient, and

$$\omega_0^{\beta+1} = 1, \quad \omega_l^{\beta+1} = \left(1 - \frac{\beta+2}{l}\right) \omega_{l-1}^{\beta+1}.$$

For the distributed-order derivative, we use the mid-point quadrature rule to approximate it:

$$\int_0^1 \phi(\alpha) \frac{\partial^{1+\alpha}T}{\partial t^{1+\alpha}} d\alpha = h_\alpha \sum_{s=0}^S \phi(\alpha_s) \frac{\partial^{1+\alpha_s}T}{\partial t^{1+\alpha_s}}, \quad (21)$$

$$\int_0^1 \phi(\alpha) \frac{\partial^{1+\alpha}T}{\partial x \partial t^\alpha} d\alpha = h_\alpha \sum_{s=0}^S \phi(\alpha_s) \frac{\partial^{1+\alpha_s}T}{\partial x \partial t^{\alpha_s}}$$

$$\int_0^1 \phi(\beta) \frac{\partial^{\beta+1}T}{\partial x^{\beta+1}} d\beta = h_\beta \sum_{s=0}^S \phi(\beta_s) \frac{\partial^{\beta_s+1}T}{\partial x^{\beta_s+1}}, \quad (22)$$

$$\int_0^1 \phi(\beta) \frac{\partial^{\beta+1}T}{\partial(-x)^{\beta+1}} d\beta = h_\beta \sum_{s=0}^S \phi(\beta_s) \frac{\partial^{\beta_s+1}T}{\partial x^{\beta_s+1}}$$

where $h_\alpha = h_\beta = \frac{1}{S+1}$ is the fractional step.

3.2. fPINN with Square Loss Terms (SfPINN)

Firstly, we incorporate a boundary-enforcing function $\rho(x)$ into the network output T_θ , where $\rho(x)$ is constructed such that $\rho(x) = 0$ for all on the boundary of the domain Ω , $x \in \partial\Omega$. This yields the predicted solution,

$$\hat{T} = T_\theta \rho(x) \quad (23)$$

which inherently satisfies the prescribed boundary conditions.

As a concrete example, consider the boundary conditions $T(0, t) = T(1, t) = 0$ of our problem. By choosing $\rho(x) = x(1-x)$, the network's output automatically enforces these boundary conditions, obviating the need to include a separate boundary loss term in the overall loss function and consequently simplifying its structure.

For the remaining equation residual loss and initial condition loss, we attempt to express the loss in a squared form:

$$\mathcal{L}(T)^2 - f^2 = (\mathcal{L}(T) - f)(\mathcal{L}(T) + f) \quad (24)$$

$$I(T)^2 - T_0^2 = (I(T) - T_0)(I(T) + T_0), \quad (25)$$

Taking the residual loss of the equation as an example, it is not difficult to conclude

$$\begin{aligned} \|\mathcal{L}_\theta(T)^2 - f^2\|_2 &= \sqrt{\sum_{i=1}^N |\mathcal{L}_\theta(T_i)^2 - f_i^2|^2} \\ &= \sqrt{\sum_{i=1}^N |(\mathcal{L}_\theta(T_i) - f_i)(\mathcal{L}_\theta(T_i) + f_i)|^2} \\ &\leq \sqrt{\left(\sum_{i=1}^N |\mathcal{L}_\theta(T_i) - f_i| \|\mathcal{L}_\theta(T_i) + f_i\|\right)^2} \\ &\leq \sqrt{\left(\sum_{i=1}^N |\mathcal{L}_\theta(T_i) - f_i|^2\right) \left(\sum_{i=1}^N |\mathcal{L}_\theta(T_i) + f_i|^2\right)} \\ &= \|\mathcal{L}_\theta(T) - f\|_2 \|\mathcal{L}_\theta(T) + f\|_2, \end{aligned} \quad (26)$$

when $\|\mathcal{L}_\theta(T) - f\|_2 < \varepsilon$, it can be derived that

$$\| \mathcal{L}_\theta(T)^2 - f^2 \|_2 < \frac{\varepsilon}{\| \mathcal{L}_\theta(T) + f \|_2},$$

Therefore, if this squared term is employed as the loss function, the convergence of the first-order term implies the convergence of the squared term. Moreover, during the training process, the squared loss generally exhibits a faster convergence rate, as compared with first-order loss functions, so it achieves a more precise control over loss. The new loss function is formulated as follows:

$$Loss(\theta) = \omega_{\text{PDE}} \text{MSE}_{\text{PDE}} + \omega_{\text{IC}} \text{MSE}_{\text{IC}} + \omega_{\text{SPDE}} \text{MSE}_{\text{SPDE}} + \omega_{\text{SIC}} \text{MSE}_{\text{SIC}}, \quad (27)$$

where

$$\text{MSE}_{\text{SPDE}} = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} |\mathcal{L}(T)^2 - f^2|^2, \quad (28)$$

$$\text{MSE}_{\text{SIC}} = \frac{1}{N_{\text{IC}}} \sum_{i=1}^{N_{\text{IC}}} |\hat{T}(x_i, 0)^2 - T_0^2|^2. \quad (29)$$

In this work, a two-stage training paradigm is adopted to train the neural network; we divide the number of iterations into two equal parts. Specifically, the first training stage leverages the strict initial condition constraint denoted by MSE_{SIC} to rapidly lock the distribution interval of the numerical solution, laying a solid foundation for subsequent optimization. In the second stage, the network is fine-tuned via the strict PDE residual constraint represented by MSE_{SPDE} , which further refines the solution, enabling it to strictly satisfy the underlying physical laws. Notably, the loss functions employed in these two successive stages are distinct:

$$Loss_1(\theta) = \omega_{\text{PDE}} \text{MSE}_{\text{PDE}} + \omega_{\text{IC}} \text{MSE}_{\text{IC}} + \omega_{\text{SIC}} \text{MSE}_{\text{SIC}}, \quad (30)$$

$$Loss_2(\theta) = \omega_{\text{PDE}} \text{MSE}_{\text{PDE}} + \omega_{\text{IC}} \text{MSE}_{\text{IC}} + \omega_{\text{SPDE}} \text{MSE}_{\text{SPDE}}, \quad (31)$$

Figure 1 presents the flowchart of our SfPINN algorithm for the time–space distributed-order heat conduction model. In this framework, x and t are first taken as the inputs to the network. After processing through several hidden layers, the network outputs T_θ . This output is then multiplied by $\rho(x)$ to automatically satisfy the boundary and initial conditions. Two types of loss functions, $Loss_1$ and $Loss_2$, are provided; they can be used either individually or simultaneously in a two-stage training scheme.

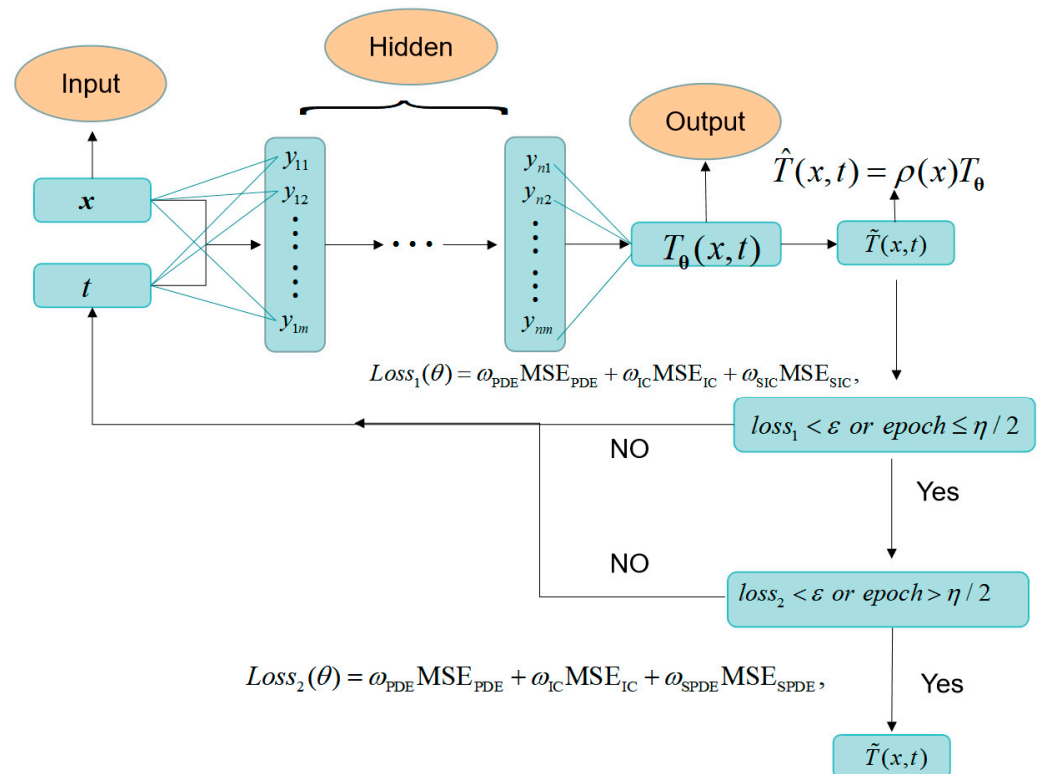


Figure 1. Flowchart of SfPINN for the time and space distributed-order Cattaneo–Christov model.

3.3. Adaptive Loss-Weight Update

In the preceding section, we augmented the conventional loss function by introducing an additional squared loss term, multiplied by a weighting coefficient. In practice, the magnitudes of the first-order loss and the squared loss term usually differ considerably, making it difficult to prescribe a fixed weight heuristically. Moreover, the weights should evolve as training progresses. For instance, in the two-stage training strategy described earlier, the initial condition loss is emphasized in the early stage, while the PDE residual loss is refined later; accordingly, the initial condition loss receives a larger weight at the beginning, and the residual loss is assigned a larger weight in the later stage.

To address this issue, we adopt the Learning Rate Annealing (LRA) adaptive weighting algorithm. This method is grounded in a key observation: if the gradient norm of a certain loss term is large, this indicates that this term is dominating the parameter updates, and its weight should be reduced; conversely, a loss term with a small gradient norm should have its weight increased. By equalizing the effective learning rates across different loss terms, the algorithm achieves automatic balancing.

Let the total loss function be defined as

$$Loss_{total} = \sum_{i=1}^N \lambda_i Loss_i, \quad (32)$$

where $Loss_i$ can be the PDE residual loss, initial condition loss, boundary condition loss, data loss, etc., and λ_i is the corresponding weight.

We adopt an exponential moving average (EMA) update rule for the weights:

$$\lambda_i^{(k)} = (1 - \alpha) \cdot \lambda_i^{(k-1)} + \alpha \cdot \frac{\max_j \{ \|\nabla_{\theta} Loss_j\| \}}{\|\nabla_{\theta} Loss_i\|}, \quad (33)$$

where $\|\nabla_{\theta} Loss_i\|$ is the L_2 norm of the gradient of the i -th loss term with respect to the network parameters θ . $\|\nabla_{\theta} Loss_i\|$ is the exponential moving average (EMA) of this gradient norm, employed to stabilize training:

$$\|\nabla_{\theta} Loss_i\|^{(k)} = \beta \cdot \|\nabla_{\theta} Loss_i\|^{(k-1)} + (1 - \beta) \|\nabla_{\theta} Loss_i^{(k)}\|; \quad (34)$$

and $\alpha \in (0,1)$ is the weight update rate and $\beta \in (0,1)$ is the EMA decay factor. And the initial weights are set as $\lambda_i^{(0)} = 1$.

4. Numerical Tests

4.1. Forward Problem

To verify the correctness of the fPINN and SfPINN algorithms, we construct the following governing equation by incorporating a source term $f(x,t)$:

$$\begin{aligned} \frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + \xi \int_0^1 \varphi(\alpha) \frac{\partial^{1+\alpha} T}{\partial t^{1+\alpha}} d\alpha + \xi u \int_0^1 \varphi(\alpha) \frac{\partial^{1+\alpha} T}{\partial x \partial t^{\alpha}} d\alpha + \xi u \frac{\partial^2 T}{\partial t \partial x} + \xi u^2 \frac{\partial^2 T}{\partial x^2} \\ - \int_0^1 \phi(\beta) \left(\gamma \frac{\partial^{\beta+1} T(x,t)}{\partial x^{\beta+1}} + (1-\gamma) \frac{\partial^{\beta+1} T(x,t)}{\partial (-x)^{\beta+1}} \right) d\beta = f(x,t). \end{aligned} \quad (35)$$

Let $T(x,t) = (t^2 + 1)x^2(1-x)^2$ be the exact solution and let $\varphi(\alpha) = \Gamma(3-\alpha)$ and $\phi(\beta) = \Gamma(4-\beta)$; the source term $f(x,t)$ can be obtained as shown below:

$$\begin{aligned} f(x,t) = & 2tx^2(1-x)^2 + 2u(t^2+1)x(1-x)^2 + 2u(t^2+1)x^2(x-1) + 4\xi utx(1-x)^2 \\ & + 4\xi utx^2(x-1) + 2\xi u^2(t^2+1)(1-x)^2 + 8\xi u^2(t^2+1)x(x-1) \\ & + 2\xi u^2(t^2+1)x^2 + 2\xi x^2(x-1)^2 \frac{\ln t(2t-1)-t+1}{(\ln t)^2} + \frac{4\xi utx(t-1)(2x^2-3x+1)}{\ln t} \\ & - 2\gamma(t^2+1) \frac{2x + (\ln x)^2(12x^3-30x^2+18x-2) + \ln x(6x^2-11x+3)-2}{(\ln x)^3} \\ & + 2(1-\gamma)(t^2+1) \frac{(2x - (\ln(1-x))^2(-12x^3+6x^2-2) + \ln(1-x)(-6x^2+x+2))}{(\ln(1-x))^3}. \end{aligned} \quad (36)$$

When solving traditional integer-order differential equations, automatic differentiation can be used, but this approach cannot compute fractional-order derivatives. For the fractional-derivative terms $\frac{\partial^{\alpha}}{\partial t^{\alpha}}$, $\frac{\partial^{\beta}}{\partial x^{\beta}}$ and $\frac{\partial^{\beta}}{\partial (-x)^{\beta}}$, the L1-approximation and the shifted Grünwald formulae are used for numerical discretization progress [25–27] in Equations (17)–(20), and the distributed-order terms can be discretized to the sum of multiple fractional order terms by using the mid-point quadrature rule [28] in Equations (21) and (22). Finally, the discretization formula of Equation (28) can be obtained:

$$\begin{aligned} \frac{\partial T}{\partial t}(x_i, t_k) + u \frac{\partial T}{\partial x}(x_i, t_k) + \xi h_{\alpha} \sum_{s=0}^S \phi(\alpha_s) \left(\frac{1}{\Delta t^{1+\alpha_s} \Gamma(2-\alpha_s)} \{c_0(T_i^k - T_i^{k-1}) \right. \\ \left. - \sum_{l=1}^{k-1} (c_{k-l-1} - c_{k-l})(T_i^l - T_i^{l-1})\} \right) \\ + \xi u^2 \frac{\partial^2 T}{\partial x^2}(x_i, t_k) + \xi u \frac{\partial^2 T}{\partial x \partial t}(x_i, t_k) \\ - h_{\beta} \sum_{s=0}^S \phi(\beta_s) \left(\gamma \frac{1}{\Delta x^{\beta_s+1}} \sum_{l=0}^{i+1} \omega_l^{\beta_s+1} T_{i-l+1}^k \right. \\ \left. + (1-\gamma) \frac{1}{\Delta x^{\beta_s+1}} \sum_{l=0}^{m-i+1} \omega_l^{\beta_s+1} T_{i+l-1}^k \right) \\ + \xi u h_{\alpha} \sum_{s=0}^S \phi(\alpha_s) \left(\frac{\Delta t^{-\alpha_s}}{\Gamma(2-\alpha_s) \Delta x} \{ (T_i^k - T_{i-1}^k) \right. \\ \left. - \sum_{l=1}^{k-1} (c_{k-l-1} - c_{k-l})(T_i^l - T_{i-1}^l) - c_{k-1}(T_i^0 - T_{i-1}^0) \} \right) = f(x_i, t_k), \end{aligned} \quad (37)$$

where $h_{\alpha} = h_{\beta} = \frac{1}{S+1}$ are fractional steps and $\alpha_s = \beta_s = \frac{sh_{\alpha} + (s+1)h_{\alpha}}{2}$, $s = 0, 1, 2, \dots, S$. In addition, the values of i , k are taken as $i = 0, 1, \dots, m$, $k = 2, 3, \dots, n$, respectively.

As to our experiments, the parameters selected are as shown in Table 1, unless otherwise specified. All numerical experiments were performed on a workstation equipped

with an AMD Ryzen 9 9950X CPU (16 cores, 32 threads), all algorithm codes were implemented based on the PyTorch 2.4.1 deep learning framework, and the Adam optimizer was adopted for network training. Moreover, the only condition for stopping training is that the preset number of epochs has been completed.

Table 1. Hyperparameters and settings used in the numerical experiments.

Training Parameters	Values
Neurons per layer	20
Hidden layer connection method	Fully connected
Number of hidden layers	4
Activation function	tanh function
Parameters in model equation	$\xi = 0.1, u = 0.5, \gamma = 0.5$
Distribution order weighting function	$\varphi(\alpha) = \Gamma(3 - \alpha), \phi(\beta) = \Gamma(4 - \beta)$
Number of discrete nodes for distributed order	$S = 20$
Number of iterations	epochs = 2000
Learning rate of training	$lr = 5 \times 10^{-4}$
Number of sampling points in space and time directions	$(N_x, N_t) = (20, 20)$
Weight of loss term	$\omega_{\text{PDE}} = \omega_{\text{IC}} = \omega_{\text{SPDE}} = \omega_{\text{SIC}} = 1.0$

Figure 2 presents the results of a comparison between the analytical solutions and the approximate solutions obtained based on the fPINN algorithm of Equation (35) with $N = 20$, where x and t are selected as Sobel sequences in the interval. It can be observed that for different values of x and t , both the analytical solutions and the approximate solutions show relatively close fitting, which indicates the correctness of our numerical results.

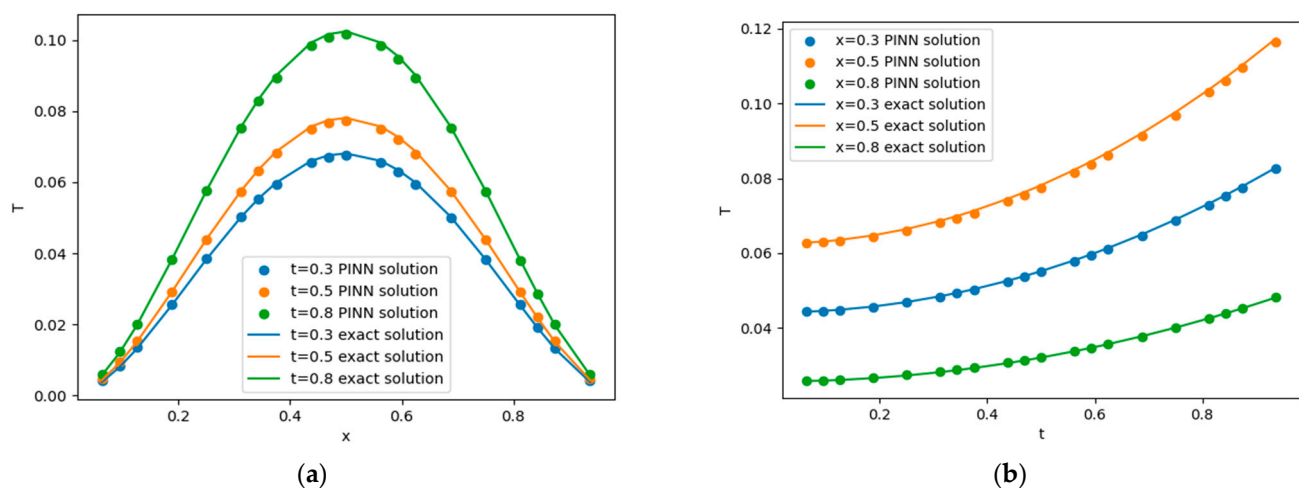


Figure 2. Comparison between the exact solutions and numerical solutions. (a) the relationship between the spatial variable x and temperature T at different times t ; (b) the relationship between time t and temperature T for different spatial variables x .

Figure 3 displays the point-wise error distribution of approximate solutions for the various parameters ξ , u , γ , as well as for the different weight functions $\varphi(\alpha)$ and $\phi(\beta)$. Figure 3a presents a three-dimensional plot of point-wise error for approximate solutions with varying values of ξ , it can be observed that for $\xi = 0.1$ and $\xi = 0.35$, the distribution trends of the point-wise error in both time and space are the same, and the point-wise error distributions at the same locations are very similar. Figure 3b–i show two-dimensional curve distributions of point-by-point errors under different parameters, where darker colors indicate smaller errors in the approximate solutions, while lighter colors

represent larger errors. It can be observed from Figure 3b–i that for different parameters, the approximate solutions obtained by our method generally have relatively small errors. Moreover, the errors of the approximate solutions are smaller near the boundaries and larger in the middle. This is because the constructed approximate solution will automatically satisfy the boundary and initial conditions of the control equation, resulting in relatively small errors in these areas. However, the penalty term for the internal points in the loss function has a very complex formula, resulting in a larger loss value and, consequently, higher errors for the internal points.

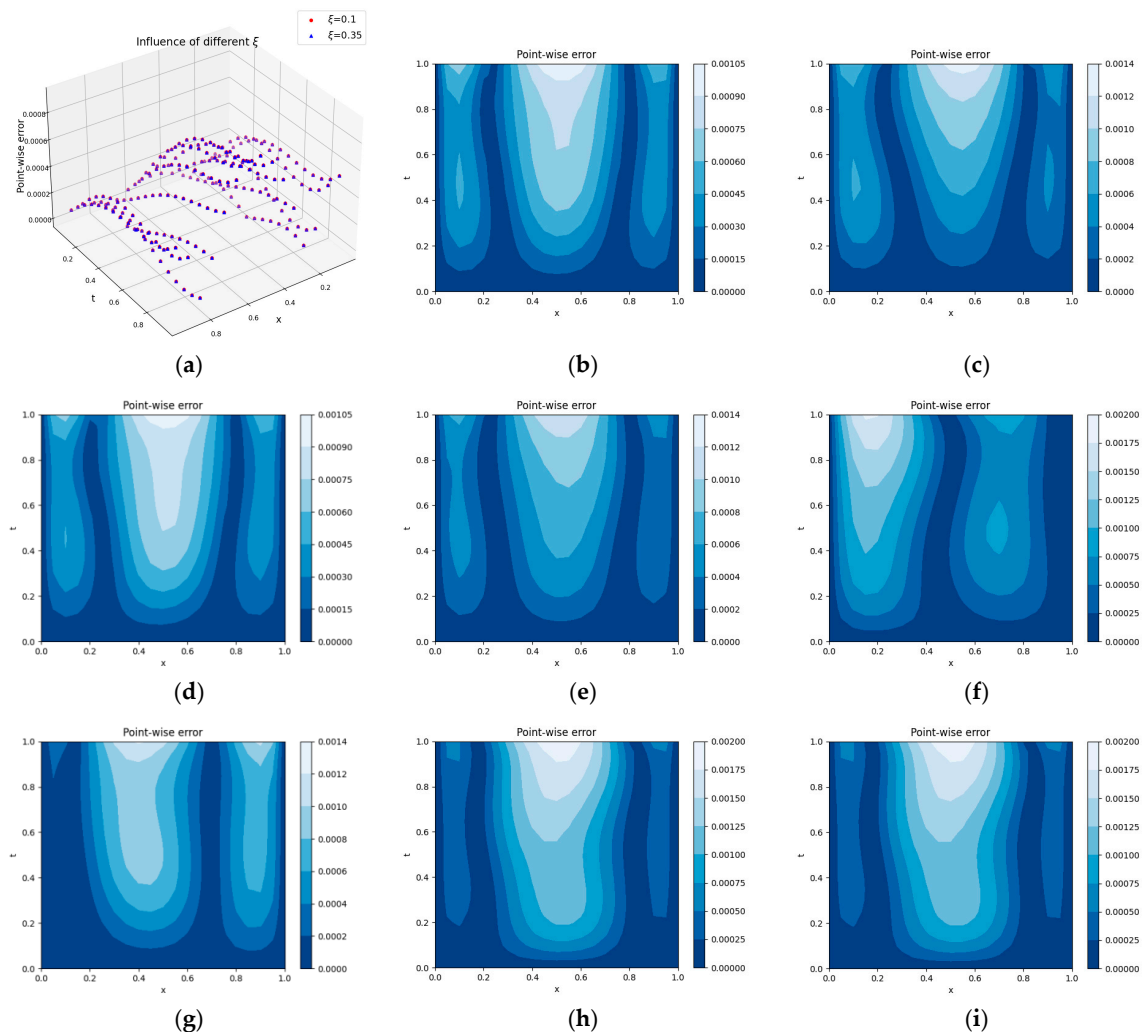


Figure 3. Point-wise error of fractional PINN with different parameters. (a) distribution of point-wise error in time and space for different values of ξ ; (b) $\xi = 0.1$; (c) $\xi = 0.35$; (d) $u = 0.3$; (e) $u = 0.6$; (f) $\gamma = 0.3$; (g) $\gamma = 0.6$; (h) $\phi(\beta) = \Gamma(5 - \beta)$; (i) $\phi(\alpha) = \Gamma(4 - \alpha)$.

It should be noted that for the different weight functions in Figure 3h,i, the source term $f(x, t)$ will also be modified accordingly. Specifically, if $\phi(\beta) = \Gamma(5 - \beta)$, the expression for $f(x, t)$ is given as follows:

$$\begin{aligned}
f(x, t) = & 2tx^2(1-x)^2 + 2u(t^2+1)x(1-x)^2 + 2u(t^2+1)x^2(x-1) + 4\xi utx(1-x)^2 \\
& + 4\xi ux^2(x-1) + 2\xi u^2(t^2+1)(1-x)^2 + 8\xi u^2(t^2+1)x(x-1) + 2\xi u^2(t^2+1)x^2 \\
& + 2\xi x^2(x-1)^2 \cdot \frac{\ln t(2t-1)-t+1}{(\ln t)^2} + \frac{4\xi utx(t-1)(2x^2-3x+1)}{\ln t} \\
& + 2\gamma(t^2+1) \left(\frac{6x+56x(\ln x)^2-60x(\ln x)^3-11(\ln x)^2+6(\ln x)^3}{(\ln x)^4} + \frac{-54x^2(\ln x)^2+108x^2(\ln x)^3}{(\ln x)^4} \right) \\
& + 2\gamma(t^2+1) \left(\frac{12x^3(\ln x)^2-48x^3(\ln x)^3}{(\ln x)^4} + \frac{\ln x(12x^2-30x+12)-6}{(\ln x)^4} \right) \\
& - 2(1-\gamma)(t^2+1) \left(\frac{6x+6\ln(1-x)-3(\ln(1-x))^2}{(\ln(1-x))^4} \right. \\
& + \frac{-6(\ln(1-x))^3+18x^2(\ln(1-x))^2+36x^2(\ln(1-x))^3+12x^3(\ln(1-x))^2}{(\ln(1-x))^4} \\
& \left. + \frac{-48x^3(\ln(1-x))^3-6x\ln(1-x)-16x(\ln(1-x))^2-12x^2\ln(1-x)+12x(\ln(1-x))^3}{(\ln(1-x))^4} \right).
\end{aligned} \tag{38}$$

When $\varphi(\alpha) = \Gamma(4-\alpha)$, the expression of the source term $f(x, t)$ is as follows:

$$\begin{aligned}
f(x, t) = & 2tx^2(1-x)^2 + 2u(t^2+1)x(1-x)^2 + 2u(t^2+1)x^2(x-1) + 4\xi utx(1-x)^2 \\
& + 8\xi u^2(t^2+1)x(x-1) + 2\xi u^2(t^2+1)x^2 \\
& + 2\xi x^2(x-1)^2 \cdot \frac{2t-\ln t(5t-3)+(\ln t)^2(6t-2)-2}{(\ln t)^3} \\
& + 4\xi ux(2x^2-3x+1) \cdot \frac{-[t(t-1)-t\ln t(3t-2)]}{(\ln t)^2} + 4\xi utx^2(x-1) \\
& + 2\xi u^2(t^2+1)(1-x)^2 \\
& - 2\gamma(t^2+1) \cdot \frac{2x+(\ln x)^2(12x^3-30x^2+18x-2)+\ln x(6x^2-11x+3)-2}{(\ln x)^3} \\
& + 2(1-\gamma)(t^2+1) \cdot \frac{2x-(\ln(1-x))^2(-12x^3+6x^2-2)+\ln(1-x)(-6x^2+x+2)}{(\ln(1-x))^3}.
\end{aligned} \tag{39}$$

Figure 4 compares the loss and relative L^2 error across different training strategies, with the number of iterations fixed at 1000 for all three experiments. fPINN denotes the standard fPINN algorithm described in Section 3.1, which is used throughout the entire training process. SIC refers to the method presented in Section 3.2, where the squared term of the initial condition is incorporated into the loss function as given in Equation (26), and this loss is used for the entire training. SIC&SPDE represents the two-stage training method introduced in Section 3.2: the total number of iterations is evenly divided into two halves; in the first half, the loss function $loss_1$ from Equation (28) is adopted, while in the second half, $loss_2$ from Equation (29) is used. And all the loss terms are distributed with equal weight in the total loss. As shown in Figure 4a, the SIC method achieves the fastest convergence in both loss and relative L^2 error in the early stage, even outperforming the SPDE approach. This indicates that, for this problem, prioritizing the optimization of the initial condition loss enables the model's solution space to quickly approach the vicinity of the true solution. In the later stage of training, as the number of iterations increases, the SPDE method exhibits smaller oscillations in the L^2 error, compared to SIC. Overall, both improved methods outperform the original standard fPINN algorithm.

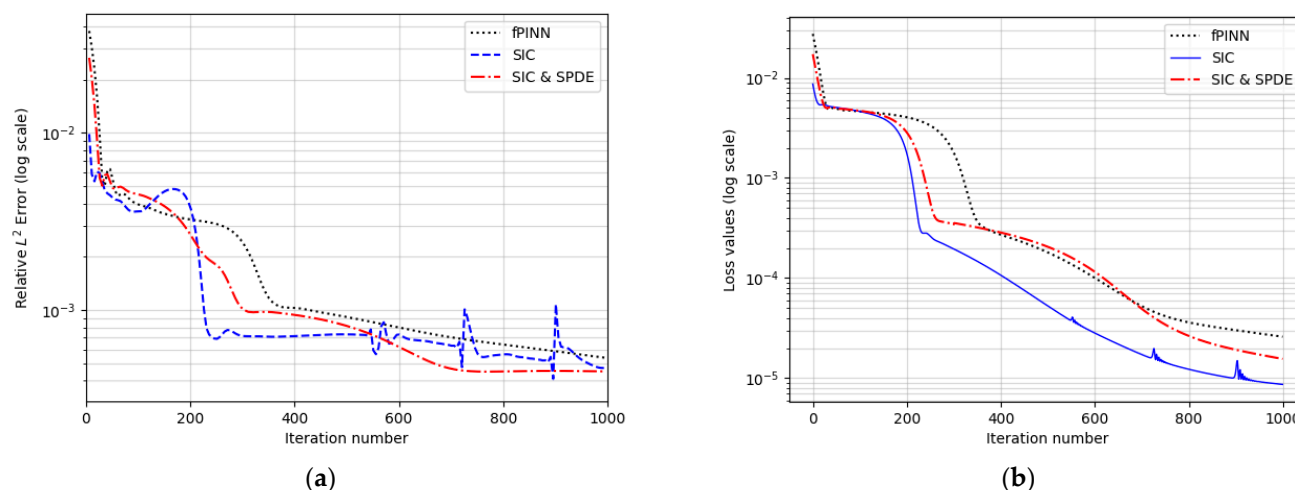


Figure 4. (a) Comparison of relative L^2 error of different training strategies; (b) Comparison of training loss values of different training strategies.

Table 2 illustrates the effects of the adaptive loss-weight updating algorithm elaborated in Section 3.3. All hyperparameters remain consistent with those listed in Table 1, with the only discrepancy lying in the weight updating strategy. Specifically, uniform weight allocation is adopted for all loss terms in fPINN and SIC. By contrast, adaptive-fPINN and adaptive-SIC incorporate the weight updating algorithm proposed in Section 3.3 on the basis of the original fPINN and SIC frameworks, respectively. As indicated by the tabular results, the adaptive loss-weight algorithm yields superior solution accuracy under an identical number of training iterations.

Table 2. The relative L^2 errors of fPINN, adaptive-fPINN, SIC, and adaptive-SIC for different numbers of training iterations, as described in Section 3.3.

Methods	Weight of the Loss Term	Iterations	Relative Errors
fPINN	$\omega_{\text{PDE}} = \omega_{\text{IC}}$	1000	6.922×10^{-4}
adaptive-fPINN	Adaptive	1000	4.126×10^{-4}
SIC	$\omega_{\text{PDE}} = \omega_{\text{IC}} = \omega_{\text{SIC}}$	1000	3.856×10^{-4}
adaptive-SIC	Adaptive	1000	3.220×10^{-4}

Figure 5a plots the evolutionary relationship between training iterations and the weights corresponding to the initial condition loss as well as the PDE residual loss during the training of adaptive-fPINN. Figure 5b depicts the weight evolution of three loss components, namely, the initial condition loss, squared initial condition loss, and PDE residual loss, against the training iterations for adaptive-SIC. It can be concluded from the graphical results that the initial condition loss dominates the total loss for both the fPINN and SIC frameworks. This phenomenon indicates that when neural networks are employed to solve differential equations, the PDE residual can be readily constrained via automatic differentiation and discrete numerical schemes; nevertheless, the initial conditions are more difficult to satisfy, owing to their relatively large approximation errors. The same principle applies to boundary conditions: the prediction accuracy at interior collocation points substantially outperforms that at boundary collocation points. Accordingly, enhancing the prediction precision near boundaries constitutes a critical approach for boosting the overall accuracy of the neural network model.

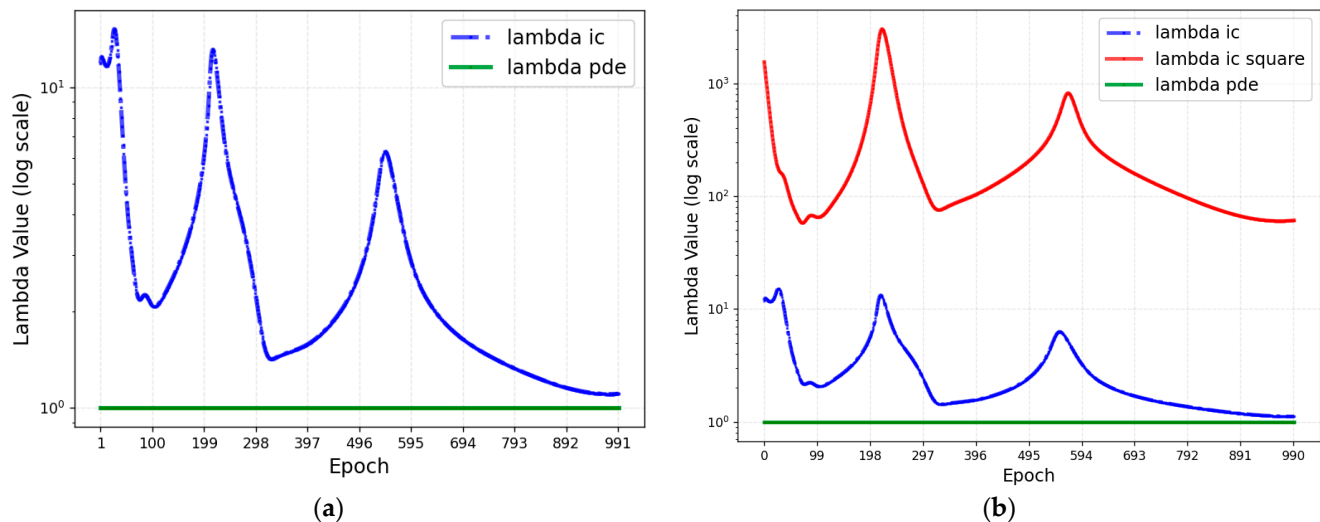


Figure 5. Weights of different loss terms versus training epoch in the adaptive weight update algorithm. (a) adaptive-fPINN algorithm; (b) adaptive-SIC algorithm.

Taking the standard fPINN algorithm as a representative case, Table 3 illustrates the influences of diverse hyperparameters, including network depth, network width, and total number of sampling points in temporal and spatial dimensions, as well as the model learning rate, on the relative L^2 error, training duration, and peak memory consumption. All training configurations remain identical except for the parameters specified in the table, and training terminates exclusively upon the completion of 1000 training epochs. As revealed by the experimental results, elevating network complexity leads to prolonged computational time and higher peak memory usage, yet it does not necessarily reduce the relative L^2 error. A plausible interpretation is that the adopted neural network framework has already approached the lower bound of approximation error; tuning network hyperparameters merely induces fluctuations around this error threshold. To achieve further accuracy improvements, modifications to network architectures or the development of novel algorithms are required. This observation further demonstrates the robustness of the fPINN framework for solving problems relating to this category, whereby a roughly consistent predictive performance can be attained across varied hyperparameter combinations. In contrast, increasing the number of sampling points yields a notable reduction in the model's L^2 error, indicating a positive correlation between sampling quantity and prediction accuracy. It is worth highlighting that all experiments described above consume thousands of seconds of runtime, owing to the sequential input of individual sampling points during the training process. Training efficiency can be drastically optimized by adopting mini-batch processing, wherein multiple sampling points are concatenated into vectorized batches and fed into the model simultaneously to substantially cut down training time.

Table 3. The relative L^2 errors, training time and peak memory use with respect to the fPINN with different hyperparameters.

Parameters				Results		
Network Depth	Network Width	Number of Sampling Points	Learning Rate	Relative L^2 Error	Training Time	Memory Use
3	20	(20×20)	5×10^{-4}	3.581×10^{-4}	4692 s	1403 MB
4	20	(20×20)	5×10^{-4}	4.553×10^{-4}	5250 s	1535 MB
5	20	(20×20)	5×10^{-4}	4.794×10^{-4}	6073 s	1659 MB
3	10	(20×20)	5×10^{-4}	3.913×10^{-4}	5033 s	1060 MB

3	30	(20×20)	5×10^{-4}	4.293×10^{-4}	5112 s	1472 MB
3	40	(20×20)	5×10^{-4}	3.593×10^{-4}	5256 s	1466 MB
3	20	(10×10)	5×10^{-4}	2.198×10^{-3}	1283 s	889 MB
3	20	(25×25)	5×10^{-4}	6.467×10^{-4}	8285 s	1956 MB
3	20	(50×50)	5×10^{-4}	2.450×10^{-4}	42,625 s	5685 MB
3	20	(20×20)	1×10^{-4}	2.270×10^{-3}	4504 s	1410 MB
3	20	(20×20)	1×10^{-3}	3.891×10^{-4}	4585 s	1416 MB
3	20	(20×20)	1.5×10^{-3}	4.128×10^{-4}	4687 s	1408 MB

Table 4 presents the effects of different allocations of training iterations across two training stages on the relative L^2 error under the two-stage training strategy of the SIC&SPDE method. In this method, the first phase uses the squared terms of the initial conditions for training, whereas the second phase adopts the squared residual term of the governing equation. The tabular results reveal that the relative L^2 error of the model declines as the number of training iterations assigned to the squared equation residual term in the second phase increases. This observation verifies the validity of the proposed two-stage training strategy: the model first prioritizes the initial condition loss in the early training stage to rapidly drive the predicted solution close to the exact solution, and then uses the squared term of the equation residual loss for fine-tuning. Such a training schedule effectively boosts the prediction accuracy and convergence rate of the model.

Table 4. The relative L^2 error with different stage-distributed training iterations for the SIC&SPDE method.

First Phase Iterations	Second Phase Iterations	Total Iterations	Relative L^2 Error
666	334	1000	1.013×10^{-3}
600	400	1000	9.248×10^{-4}
500	500	1000	1.100×10^{-3}
400	600	1000	7.970×10^{-4}
334	666	1000	6.397×10^{-4}

When the weight function is taken as a Dirac function, the distributed-order model can be reduced to a fractional-order model. To more accurately investigate the effects of the distributed-order Cattaneo–Christov model, the weight functions are normalized as $\phi^*(\alpha) = \phi(\alpha) / W_\alpha$ and $\psi^*(\beta) = \psi(\beta) / W_\beta$. For simplicity, the symbol “*” is omitted in the following discussion.

Figure 6 illustrates the influences of different fractional-order and distributed-order parameters on the temperature distribution at the center point. We select three weight functions with distinct properties (constant, Gamma, and Dirac delta functions) to systematically investigate the effects of different weight function characteristics on the results of the distributed-order Cattaneo–Christov model; these correspond to the uniform weight distribution, monotonically decreasing weight distribution, and the limiting form of degenerating into a single fractional order, respectively. It can be observed from Figure 6a that the rate of temperature decrease varies with different fractional-order parameters. At earlier times, a smaller α leads to a faster decrease in temperature, whereas at later times, a larger α results in a faster decrease. Consequently, intersections exist among these curves. When α is taken as a distributed-order fractional parameter over the interval $[0.6, 1.0]$, the temperature curve at the center lies between the two curves corresponding to $\alpha = 0.6$ and $\alpha = 1.0$. From Figure 6b, it can be seen that a larger fractional-order parameter β corresponds to a faster rate of temperature decrease. When β is taken as a distributed-order fractional parameter over the interval $[0.6, 1.0]$, the temperature curve at the center lies between the two curves corresponding to $\beta = 0.6$ and $\beta = 1.0$. Thus, the

distributed-order model governed by the weight function offers greater flexibility in describing the memory characteristics of temperature.

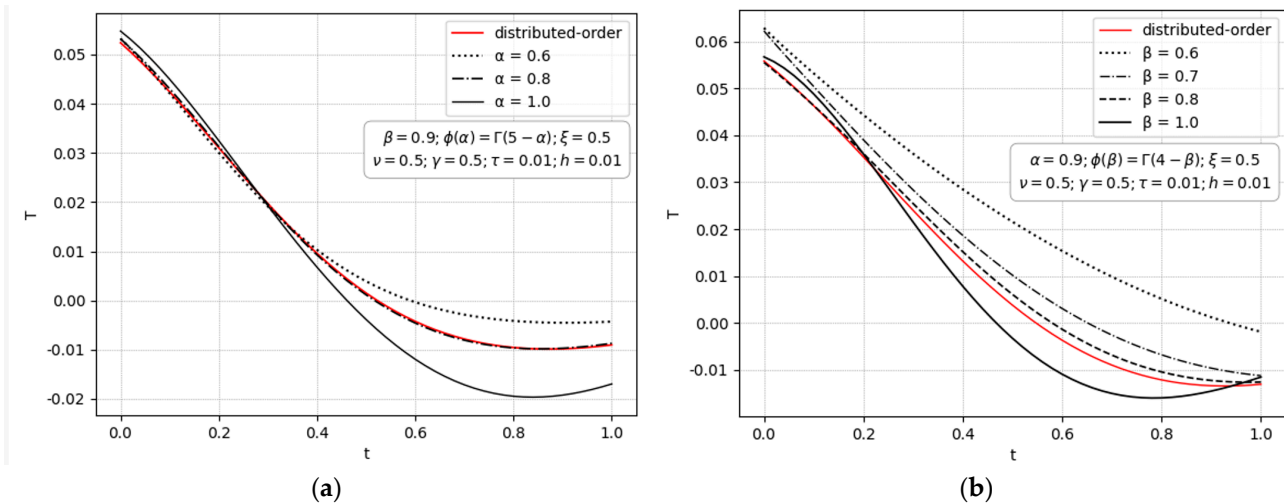


Figure 6. Center temperature distributions with respect to t . (a) change with α ; (b) change with β .

4.2. Inverse Problem

The fPINN method enables effective solution of inverse problems within a coding framework nearly identical to that used for forward problems. The inverse problem needs to be solved by incorporating the unknown parameters ξ , u and γ of Equation (35) into the neural network's trainable parameters. And the increased number of optimization parameters leads to greater training complexity and extends the time required for each iteration. Notably, the selection of initial values significantly impacts training efficacy. In contrast to forward problems, acquiring initial and boundary conditions for inverse problems is considerably challenging. To address this, we can incorporate measured data at $t = 1$ and add the sum of the squared residuals to the loss function.

The loss function for the inverse problem can be expressed as

$$Loss(\boldsymbol{\mu}) = w_1 Loss_f(\boldsymbol{\mu}) + w_2 \frac{1}{|\Xi_m|} \sum_{(x,\tau) \in \Xi_m} (\hat{u}(x,\tau) - u_m(x,\tau))^2, \quad (40)$$

where $|\Xi_m|$ represents the number of training points at $t = 1$. The observation points at the boundary are usually 20, 40, or 80 points ($|\Xi_m| = 20, 40$ or 80). In addition, w_1 and w_2 are the two penalty weights, and it is recommended that the value of w_2 be relatively small.

Figure 7 illustrates the variation of the parameter vector of the distributed-order control equation, Equation (35), as the optimizer iterations progress. Obviously, as the number of iterations increases, the estimated values of the parameters will gradually approach the true values and eventually stabilize. The convergence rate depends critically on the quality of the initial values. If the initial values are poorly chosen, convergence will be slow; conversely, good initial values lead to a significantly faster convergence rate. Table 5 provides a detailed comparison of the effects of different true values and different initial values of parameters on parameter estimation. The results show that the proposed method yields accurate parameter estimation across various true values and the estimation performance improves when the initial parameter values are closer to the true values.

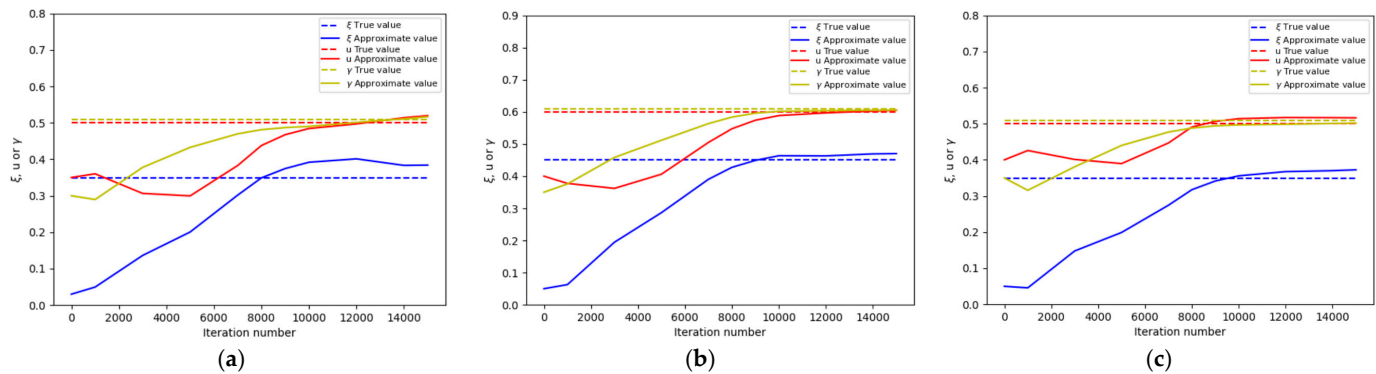


Figure 7. Parameter evolution as the iteration of optimizer progresses with different initial values (ξ, u, γ) . (a) $(0.03, 0.35, 0.30)$; (b) $(0.05, 0.40, 0.35)$; (c) $(0.05, 0.40, 0.35)$.

Table 5. The estimated results of the parameters (ξ, u, γ) .

Initial Values	True Parameters	Identified Parameters	L^2 Relative Error	Iteration Number
$(0.03, 0.35, 0.30)$	$(0.35, 0.50, 0.51)$	$(0.3839, 0.5197, 0.5167)$	1.01×10^{-2}	15,000
$(0.05, 0.40, 0.35)$	$(0.35, 0.50, 0.51)$	$(0.3725, 0.5164, 0.5017)$	8.9×10^{-3}	15,000
$(0.05, 0.40, 0.35)$	$(0.45, 0.60, 0.61)$	$(0.4702, 0.6047, 0.6055)$	9.1×10^{-3}	15,000

Table 6 presents the impacts of noise with different intensities on the parameter identification of the inverse problem, and verifies the robustness of the proposed algorithm against measurement disturbances. Additive Gaussian white noise is adopted to simulate observation errors in practical scenarios. For the exact source term $f(x, t)$ at each observation point, the corresponding noisy observation is given by

$$f_{\text{obs}}(x, t) = f(x, t) + \eta \cdot \sigma_f \cdot \varepsilon \quad (41)$$

where η denotes the preset noise intensity coefficient, σ_f represents the standard deviation of the exact source term over all observation points, and ε is an independent random variable following the standard normal distribution $\mathcal{N}(0, 1)$. In each training iteration, 10% of the noisy observation samples are randomly selected to simulate a situation where random disturbances are present in the measurement data. Unless otherwise specified, the hyperparameter settings for the neural network are consistent with those in Table 1.

Table 6. Identification of parameters (ξ, u, γ) for noises of different intensity.

Initial Values	True Parameters	Identified Parameters	Noise Strength η	L^2 Relative Error	Iteration Number
$(0.25, 0.40, 0.35)$	$(0.7, 0.8, 0.6)$	$(0.685, 0.740, 0.598)$	0%	1.475×10^{-3}	15,000
$(0.25, 0.40, 0.35)$	$(0.7, 0.8, 0.6)$	$(0.682, 0.740, 0.599)$	5%	1.544×10^{-3}	15,000
$(0.25, 0.40, 0.35)$	$(0.7, 0.8, 0.6)$	$(0.685, 0.749, 0.599)$	10%	1.512×10^{-3}	15,000
$(0.25, 0.40, 0.35)$	$(0.7, 0.8, 0.6)$	$(0.649, 0.751, 0.599)$	20%	1.665×10^{-3}	15,000

Table 6 presents the identification results for the parameters ξ , u and γ under different noise intensities, together with the overall L^2 relative error of the model, while Figure 8 shows the variation of each identified parameter value with the number of training iterations. It can be observed that, as the number of training iterations increases, the differences among the identified values of ξ and u under different noise levels tend to widen gradually. In contrast, the identified values of γ exhibit marked stability against varying noise intensities, with the corresponding curves remaining tightly clustered throughout. Moreover, the overall L^2 relative errors of the equation differ only slightly

across different noise intensities, indicating that the model possesses favorable robustness in both parameter identification and equation approximation within the noise range considered in the experiments. Building on these findings, future work can conduct multi-parameter coupled experiments with respect to factors such as noise intensity and parameter initialization strategies, so as to systematically evaluate the comprehensive impact of noise on the identification accuracy and robustness of the model.

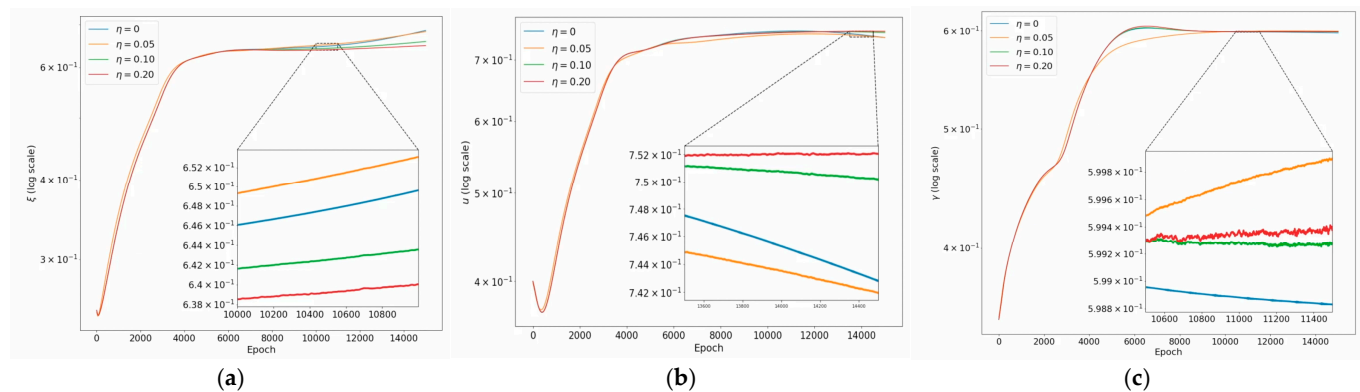


Figure 8. Parameter evolution during training iterations at different noise levels. (a) ξ ; (b) u ; (c) γ .

5. Conclusions

This paper proposes a distributed-order Cattaneo–Christov constitutive model to describe the heat conduction process. A novel numerical method based on fPINN is developed to solve the distributed-order heat conduction model. By embedding the boundary conditions directly into the network output, the boundary conditions are automatically satisfied. Meanwhile, the squared loss terms corresponding to the initial conditions and the residual of the governing equation are added to the original loss function, and a two-stage training strategy is constructed to enhance model accuracy. Furthermore, it adaptively updates the weights of these terms in the total loss function based on the gradients of the individual loss terms. The experimental results demonstrate that our algorithm outperforms the standard fPINN method as to both relative L^2 error and convergence speed. In addition, comparative experiments between the distributed-order model and the fractional-order model are conducted to illustrate how the distributed-order model captures memory effects during the heat conduction process. For the inverse problem, the experimental results show that as the number of iterations increases, the estimated parameter values gradually approach, and eventually stabilize at, the true values. The convergence rate depends on the quality of the initial guess: a poor initial guess leads to slow convergence, whereas a good initial guess accelerates the convergence.

Although the proposed SfPINN framework exhibits good accuracy and robustness for both forward and inverse problems of distributed-order heat conduction, several limitations remain. First, the computational cost of fPINN increases significantly with the number of training points and iterations, especially when high-resolution discretization of the distributed-order terms is required. Second, the convergence behavior is sensitive to the choice of hyperparameters, such as network depth, width, learning rate, and penalty weights in the loss function. Third, this study focuses only on one-dimensional problems with relatively simple geometries; extending the method to higher-dimensional domains and complex geometries requires further investigation.

Author Contributions: Conceptualization, X.C., W.Z., J.Y. and W.Y.; methodology, X.C., W.Z. and Y.L.; software, W.Z. and X.C.; validation, X.C., Y.L. and W.Z.; writing—original draft preparation,

X.C. and W.Z.; writing—review and editing, X.C., W.Z., J.Y. and W.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (12202041), the funding of the Postdoctoral Special Support Fund of China (2025T180849) and the State Key Laboratory for Advanced Metals and Materials of China (2025-S10).

Data Availability Statement: The data that support the findings of this study can be produced by using our methods.

Conflicts of Interest: The authors declare no conflicts of interest.

Nomenclature

c_p	Specific heat capacity
ρ	Mass density
u	Convection velocity in the x -direction
τ	Relaxation time coefficient
α, β	Fractional order parameters of time and space derivatives
$\phi(\alpha), \psi(\beta)$	Weight functions for distributed-order fractional derivatives
γ, ν	Weight coefficients in the distributed-order Cattaneo–Christov model
$\Gamma(\cdot)$	Euler gamma function
${}_0^C D_t^\alpha$	Caputo time fractional derivative of order α
${}_{RL} D_x^\beta, {}_x D_{RL}^\beta$	Left and right Riemann–Liouville space fractional derivatives
$\delta(\cdot)$	Dirac delta function
$\Delta x, \Delta t$	Spatial and temporal grid step size
x_i, t_n	Discrete spatial and temporal mesh points
$T(x, t)$	Temperature field function
T_i^n	Numerical solution at mesh point (x_i, t_n)
$\mathcal{L}$	Partial differential equation operator
θ	Trainable parameter set of neural network
$\sigma(\cdot)$	Activation function of neural network
η	Adaptive weights for initial condition loss and PDE residual loss
$\ \nabla \mathcal{L}_i\ $	Gradient norm of the i -th loss term with respect to network parameters
$\overline{\ \nabla \mathcal{L}_i\ }$	EMA of gradient norm of the i -th loss term

References

1. Crank, J. *The Mathematics of Diffusion*, 2nd ed.; Clarendon Press: Oxford, UK, 1975; pp. 1–414. Available online: <https://people.bath.ac.uk/ensdasr/PAPERS/Crank-The-Mathematics-of-Diffusion.pdf> (accessed on 11 September 2025).
2. Khan, M.; Khan, W.A. Three-dimensional flow and heat transfer to burgers fluid using Cattaneo–Christov heat flux model. *J. Mol. Liq.* **2016**, *221*, 651–657.
3. Atanackovic, T.M.; Pilipovic, S.; Zorica, D. A diffusion wave equation with two fractional derivatives of different order. *J. Phys. A Math. Theor.* **2007**, *40*, 5319–5333.
4. Ghazizadeh, H.R.; Maerefat, M.; Azimi, A. Explicit and implicit finite difference schemes for fractional Cattaneo equation. *J. Comput. Phys.* **2010**, *229*, 7042–7057.
5. Cattaneo, C. Sulla conduzione del calore. *Atti Sem. Mat. Fis. Univ. Modena* **1948**, *3*, 83–101.
6. Liu, L.; Zheng, L.; Zhang, X. Fractional anomalous diffusion with Cattaneo–Christov flux effects in a comb-like structure. *Appl. Math. Model.* **2016**, *40*, 6663–6675.
7. Christov, C.I. On frame indifferent formulation of the Maxwell–Cattaneo model of finite-speed heat conduction. *Mech. Res. Commun.* **2009**, *36*, 481–486.
8. Liu, L.; Zheng, L.C.; Liu, F.W.; Zhang, X.X. Heat conduction with fractional Cattaneo–Christov upper-convective derivative flux model. *Int. J. Therm. Sci.* **2017**, *112*, 421–426.
9. Caputo, M. *Elasticità e Dissipazione*; Zanichelli: Bologna, Italy, 1969.

10. Duan, J.; Chen, L. Oscillatory shear flow between two parallel plates for viscoelastic constitutive model of distributed-order derivative. *Int. J. Numer. Methods Heat Fluid Flow* **2020**, *30*, 1137–1148.
11. Chen, X.; Xie, H.; Yang, W.; Chen, M.; Zheng, L. Start-up flow in a pipe of a double distributed-order Maxwell fluid. *Appl. Math. Lett.* **2022**, *134*, 108302.
12. Yin, B.; Liu, Y.; Li, H.; Zhang, Z. Approximation methods for the distributed order calculus using the convolution quadrature. *Discrete Contin. Dyn. Syst. B* **2021**, *26*, 1447–1468.
13. Morgado, M.L.; Rebelo, M. Numerical approximation of distributed order reaction–diffusion equations. *J. Comput. Appl. Math.* **2015**, *275*, 216–227.
14. Liu, L.; Chen, S.; Feng, L.; Zhu, J.; Zhang, J.; Zheng, L.; Xie, C. A novel distributed order time fractional model for heat conduction, anomalous diffusion, and viscoelastic flow problems. *Comput. Fluids* **2023**, *265*, 105991.
15. Cao, X.; Rong, Z.; Lin, P.; Zheng, L.; Zhang, X. A novel distributed-order time fractional derivative model of laser-induced thermal therapy for deep-lying tumor. *Comput. Math. Appl.* **2025**, *184*, 107–115.
16. Khan, M.; Zhang, Z.; Lu, D. Numerical simulations and modeling of MHD boundary layer flow and heat transfer dynamics in Darcy-forchheimer media with distributed fractional-order derivatives. *Case Stud. Therm. Eng.* **2023**, *49*, 103234.
17. Pang, G.; Lu, L.; Karniadakis, G.E.M. fPINNs: Fractional Physics-Informed Neural Networks. *SIAM J. Sci. Comput.* **2019**, *41*, A2603–A2626.
18. Guo, L.; Wu, H.; Yu, X.; Zhou, T. Monte Carlo fPINNs: Deep learning method for forward and inverse problems involving high dimensional fractional partial differential equations. *Comput. Methods Appl. Mech. Eng.* **2022**, *400*, 115523.
19. Wang, S.; Zhang, H.; Jiang, X. Physics-informed neural network algorithm for solving forward and inverse problems of variable-order space-fractional advection-diffusion equations. *Neurocomputing* **2023**, *535*, 64–82.
20. Wang, S.; Zhang, H.; Jiang, X. Parameter estimation for unsteady MHD oscillatory free convective flow of generalized second grade fluid with Hall effects and thermal radiation effects. *Int. J. Heat Mass Transfer* **2024**, *219*, 125123.
21. Wang, X.; Wang, X.; Xu, H.; Qi, H. Numerical analysis and comparison of fractional physics-informed neural networks in unsaturated flow process. *Commun. Nonlinear Sci. Numer. Simul.* **2025**, *147*, 108833.
22. Pei, M.J.; Yan, X.B.; Wei, T. Adaptive-fPINNs method for recovering simultaneously the fractional order and the space-dependent zeroth-order coefficient in subdiffusion. *Phys. D Nonlinear Phenom.* **2025**, *483*, 134993.
23. Xu, Y.; Liu, Y.; Zhang, Y.; Pang, G.; Sun, H. fPICRNNs: Fractional physics-informed convolutional-recurrent neural networks for solving fractional order partial differential equations. *Neurocomputing* **2026**, *669*, 132433.
24. Liu, W.; Liu, Y.; Li, H.; Zhang, Z. Physics-informed neural network combined with weighted square terms and two-phase training strategy for partial differential equations. *Eng. Anal. Bound. Elem.* **2026**, *183*, 106596.
25. Yang, X.; Zhang, Z. Analysis of a new NFV scheme preserving DMP for two-dimensional sub-diffusion equation on distorted meshes. *J. Sci. Comput.* **2024**, *99*, 80.
26. Yang, X.; Zhang, Z. Superconvergence analysis of a robust orthogonal Gauss collocation method for 2D fourth-order subdiffusion equations. *J. Sci. Comput.* **2024**, *100*, 62.
27. Jain, S.; Abuomar, M. Numerical investigation of the fractional Riccati equation: Analysis, discretization, and applications. *Commun. Nonlinear Sci. Numer. Simul.* **2026**, *154*, 109539.
28. Yang, W.; Chen, X.; Zhang, X.; Zheng, L.; Liu, F. Flow and heat transfer of viscoelastic fluid with a novel space distributed-order constitution relationship. *Comput. Math. Appl.* **2021**, *94*, 94–103.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.