AIMS Mathematics, 11(9): 29696–29726.
DOI: 10.3934/math.20261178
Received: 11 March 2026
Revised: 14 August 2026
Accepted: 28 August 2026
Published: 14 September 2026

https://www.aimspress.com/journal/Math

Research article

Numerical study of fractional-order Ginzburg-Landau equations using fPINNs

Hong Lu[1], **Yujie Zhai**[1] **and Mingji Zhang**[2,*]

[1] School of Mathematics and Statistics, Shandong University, Weihai, Shandong 264209, China; ljwenling@163.com, 202317831@mail.sdu.edu.cn

[2] Department of Mathematics, New Mexico Institute of Mining and Technology, Socorro, NM 87801, USA

* **Correspondence:** Email: mingji.zhang@nmt.edu.

Abstract: In this work, we propose an efficient fractional-order physics-informed neural network (fPINN) approach to solve the complex fractional Ginzburg-Landau equations. To handle the temporal fractional derivatives, the $L2\text{-}1_\sigma$ difference scheme is adopted, outperforming $L1$ and $L1$-2 schemes in accuracy. Spatial fractional derivatives are handled via the shifted Grünwald-Letnikov (G-L) formula, enabling high-precision nonlinear simulation. Supported by detailed numerical error analysis, our method provides a robust and reliable tool for modeling dissipative dynamical systems.

Keywords: fractional Ginzburg-Landau equations; fPINNs; $L2\text{-}1_\sigma$ formula; shifted G-L formula; reverse-value functions; multi-output network
Mathematics Subject Classification: 35R11, 65M99

1. Introduction

Proposed by Ginzburg and Landau in 1950 [1], the Ginzburg-Landau equations (GLEs) are fundamental, highly universal nonlinear evolution equations. Originally formulated to describe the macroscopic phenomenological laws of superconductivity [2], the GLEs have since been extended to model diverse complex systems, including superfluids [3, 4], fluid dynamics [5, 6], Bose-Einstein condensation [7, 8], and nonlinear optics [9, 10]. Despite the maturity of numerical methods for classical GLEs [11–14], these integer-order models struggle to capture spatial non-locality and long-range memory effects. To address this, Milovanov [15] pioneered fractional-order generalizations to describe critical phenomena, a framework further extended by Tarasov [16] to include spatial fractional-order GLEs for dynamics in fractal dispersive media. These advancements provide a robust theoretical toolkit for analyzing nonlinear dynamics in complex systems.

Research into numerical solutions for fractional GLEs has advanced significantly, with algorithmic sophistication steadily increasing. He and Pan [17] established an unconditionally stable linearized implicit finite difference method for 1D GLEs. For fractional Laplace GLEs, Wang [18] introduced a high-efficiency spectral Galerkin method, while Wang and Huang [19] developed a stepwise quasi-compact finite difference scheme for 1D and 2D nonlinear models, proving its stability and convergence. Spatial and temporal discretization techniques have also evolved: Chen et al. [20] coupled the $L1$ scheme with the Galerkin finite element method for 2D nonlinear GLEs, whereas Mahmoud et al. [21] combined the $L2\text{-}1_\sigma$ formula with a Legendre-Galerkin spectral approach to achieve high-accuracy solutions for Caputo-Riesz fractional GLEs. Zhao et al. [22] proposed a dynamical low-rank numerical method for space fractional Ginzburg-Landau equations, and proved its convergence, accuracy, and robustness via numerical examples. Gu et al. [23] investigated the well-posedness of the real fractional Ginzburg-Landau equation in different function spaces and analyzed the long-time asymptotic behavior of its nonnegative global solutions. Further methodological developments are detailed in [24–27].

Deep learning [28] has introduced transformative paradigms for approximating partial differential equation (PDE) solutions, leveraging its exceptional function approximation capabilities and data-driven nature. Unlike traditional mesh-based methods, such as finite element methods (FEMs) [29, 30], finite difference methods (FDMs) [17, 31], and spectral methods (SMs) [32, 33], deep learning approaches offer a mesh-free, flexible framework that enhances computational efficiency and throughput, enabling robust and efficient numerical solutions for partial differential equations. A prominent example are the Physics−Informed Neural Networks (PINNs) [34, 35], which effectively learn from sparse data by embedding governing physical laws directly into the loss function as a regularization term. By simultaneously minimizing data discrepancy and the physical residual, PINNs ensure that predictions remain physically consistent, thereby improving reliability and generalization beyond purely data-driven models. Building on the established versatility of PINNs [36–39], Lu et al. [37] extended the original PINN concept by integrating discrete fractional derivative operators into the core PINN architecture, using a hybrid residual construction that combines automatic differentiation for integer-order terms and numerical discretization for fractional terms, thereby addressing the limitations of standard automatic differentiation in fractional calculus. This study employs the fPINN framework [37] to solve fractional GLEs by integrating discrete fractional derivative operators with the core PINN architecture. Specifically, for time-fractional GLEs, we systematically compare the performance of fPINNs with three difference schemes ($L1$, $L1$-2, $L2\text{-}1_\sigma$) and demonstrate that the $L2\text{-}1_\sigma$ scheme provides superior accuracy. The effectiveness of the proposed framework is also validated for solving space-fractional GLEs.

To clarify the research novelty and positioning of this work, the main contributions are summarized as follows.

- This work extends the fPINNs framework to complex-valued fractional Ginzburg-Landau equations. The neural network directly outputs complex solutions, and the magnitude of complex values is adopted for result evaluation, forming a complete mesh-free solving system for complex fractional Ginzburg-Landau problems that is rarely reported in existing studies.
- Under unified experimental conditions, three commonly used temporal fractional discretization schemes, including $L1$, $L1$-2, and $L2\text{-}1_\sigma$, are systematically compared within the fPINNs framework. The numerical results verify the accuracy superiority of the $L2\text{-}1_\sigma$ scheme,

which provides an optimized temporal discretization strategy for solving time-fractional Ginzburg-Landau equations.

- We systematically explore the influences of discrete schemes, sampling quantities, network architectures, and fractional orders on numerical accuracy. By comparing with traditional FDM and FEM, the advantages and weaknesses of fPINNs for fractional Ginzburg-Landau equations are clarified. Meanwhile, the error characteristics and inherent optimization limitations of the fPINNs method are revealed, which provides a useful reference for the further application of fPINNs in nonlinear fractional dynamical systems.

The remainder of this paper is organized as follows: Section 2 introduces the fractional GLEs model approximated via fPINNs. Section 3 details the fundamental principles of PINNs and fPINNs, alongside their numerical algorithms and implementation. In Section 4, we derive three discrete schemes for the Caputo time-fractional derivative and the G-L formulation for the spatial fractional Laplacian to provide a theoretical basis for the numerical solutions. Section 5 evaluates the performance of the proposed fPINNs through quantitative numerical error analysis and numerical experiments. Finally, Section 6 summarizes the findings and suggests directions for future research.

2. Fractional Ginzburg-Landau equations (fGLEs)

We consider the fractional GLE defined on a bounded domain with Dirichlet boundary conditions, which is expressed as follows:

$$\begin{cases} {}_0^{\mathrm{C}}D_t^{\alpha}u(\mathbf{x},t)+(\nu+\mathrm{i}\eta)(-\Delta)^{\frac{\beta}{2}}u(\mathbf{x},t)+(\kappa+\mathrm{i}\zeta)|u(\mathbf{x},t)|^2u(\mathbf{x},t)-\gamma u(\mathbf{x},t)=h(\mathbf{x},t), & \mathbf{x}\in\Omega, t\in[0,T], \\ u(\mathbf{x},0)=u_0(\mathbf{x}), & \mathbf{x}\in\Omega, \\ u(\mathbf{x},t)=0, & \mathbf{x}\in\partial\Omega, t\in[0,T]. \end{cases} \tag{2.1}$$

Here, $u(\mathbf{x},t)$ denotes the complex-valued unknown function, defined on the interval $\Omega\times[0,T]$, $\Omega\subset\mathbb{R}^d(d=1,2)$, and we assume that the function $u(\mathbf{x},t)$ vanishes identically outside this domain. $\mathbf{x}=(x_1,x_2,\ldots,x_d)$ denotes the spatial coordinate, $i=\sqrt{-1}$ is the imaginary unit, the parameter $\nu>0$, representing the diffusion coefficient of the system, $\kappa>0$, related to the nonlinearity strength of the system, η, ς, and γ are real constants, $h(\mathbf{x},t)$ is a nonlinear function serving as the forcing term of the equation, and $u_0(\mathbf{x})$ is an unknown complex-valued function at $t=0$. The first term in the above equation corresponds to the Caputo time-fractional derivative, whose specific form is given by

$${}_0^{C}D_t^{\alpha}u(\mathbf{x},t)=\frac{1}{\Gamma(1-\alpha)}\int_0^t(t-s)^{-\alpha}\frac{\partial u}{\partial s}(\mathbf{x},s)ds, \quad 0<\alpha\le 1. \tag{2.2}$$

$\Gamma(\cdot)$ denotes the gamma function, and $\frac{\partial u}{\partial s}(\mathbf{x},s)$ is the first-order partial derivative of $u(\mathbf{x},s)$ with respect to the variable s. As $\alpha\to 0$, this Caputo fractional derivative reduces to an integral form; as $\alpha\to 1$, it simplifies to the first-order partial derivative of $u(\mathbf{x},t)$, i.e., $\frac{\partial u}{\partial t}(\mathbf{x},t)$. The second term in the above equation describes the second-order spatial variation of the complex-valued function u, which is characterized by the directional fractional Laplacian $(-\Delta)^{\frac{\beta}{2}}$ and can be written as a directional fractional derivative [40] of the following form:

$$(-\Delta)^{\frac{\beta}{2}}u(\mathbf{x},t)=C_{\beta,d}\int_{\|\mathbf{\Phi}\|_2=1}D_{\mathbf{\Phi}}^{\beta}u(\mathbf{x},t)d\mathbf{\Phi}, \quad \mathbf{\Phi}\in\mathbb{R}^d, 1<\beta\le 2, \tag{2.3}$$

where $C_{\beta,d} = \frac{\Gamma\left(\frac{1-\beta}{2}\right)\Gamma\left(\frac{d+\beta}{2}\right)}{2\pi^{\frac{1+d}{2}}}$ is a constant, $\|\mathbf{\Phi}\|_2$ denotes the L^2 norm, and $\mathbf{\Phi}$ represents the direction vector of differentiation with its specific value dependent on the spatial dimensionality. $D^{\beta}_{\mathbf{\Phi}}u(\mathbf{x},t)$ represents the Riemann-Liouville directional fractional derivative. As the parameter $\beta \to 2$, the fractional Laplacian reduces to the classical Laplacian operator, given by $-\Delta = -\left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \ldots + \frac{\partial^2}{\partial x_d^2}\right)$. Equation (2.3) presents the general high-dimensional definition of the directional fractional Laplacian. In this paper, we only consider the 1D case for spatial-fractional equations. This spherical-integral definition can be reduced to a 1D form, which is adopted for the numerical implementation in Section 4.2.

Equation (2.1) shows the general form of the space-time fractional Ginzburg-Landau equation. In this paper, we focus on two independent cases: The time-fractional and space-fractional cases with the temporal fractional order set to $\alpha = 1$ and the spatial fractional order set to $\beta = 2$, while the space-time fractional equation is not considered. We adopt fPINNs to solve these fractional GLEs efficiently, where governing equations are embedded as physical constraints into a composite loss function together with residuals, and initial and boundary conditions.

Equation (2.1) presents the homogeneous Dirichlet boundary condition for the target problem. It is worth noting that the proposed approach is also applicable to inhomogeneous Dirichlet boundary conditions. For inhomogeneous cases, one only needs to modify the boundary loss term by enforcing the network output to match the given non-zero boundary data, while keeping the main framework unchanged.

3. Principles and implementation of PINNs and fPINNs

In this section, we present a comprehensive overview of PINN and fPINN methodology, covering both theoretical principles and implementation techniques.

3.1. Physics-Informed Neural Networks (PINNs)

In 2019, Raissi et al. [34] introduced Physics−Informed Neural Networks (PINNs), a deep learning paradigm that integrates governing physical laws into the loss function to solve forward and inverse PDE problems. By embedding physical principles as prior knowledge, PINNs significantly enhance the accuracy and generalization of models for complex systems.

To systematically evaluate the performance of PINNs, we solve a non-linear partial differential equation subject to homogeneous boundary conditions, utilizing this as a test case for the approach.

$$\begin{cases} u_t(\mathbf{x},t) + \mathcal{N}[u(\mathbf{x},t):\lambda] = h(\mathbf{x},t), & (\mathbf{x},t) \in \Omega \times [0,T], \\ u(\mathbf{x},0) = u_0(\mathbf{x}), & \mathbf{x} \in \Omega, \\ u(\mathbf{x},t) = 0, & (\mathbf{x},t) \in \partial\Omega \times (0,T], \end{cases} \tag{3.1}$$

where $\Omega \subset \mathbb{R}^d$, $u(\mathbf{x},t)$ denotes the unknown function to be determined, while $\mathcal{N}[\cdot:\lambda]$ represents a nonlinear operator depending on the parameter λ, which describe the nonlinear interaction term in the equation.

We consider using a multilayer feedforward neural network (MLP, i.e., multilayer perceptron) to approximate the exact solution $u(\mathbf{x},t)$ of Eq (3.1). To this end, we construct a deep feedforward neural

network comprising L layers, where the specific computational rule for each neuron in the $l-$th hidden layer is as follows:

$$\mathcal{A}^{[l]}(a^{[l-1]}) = w^{[l]}a^{[l-1]} + b^{[l]},$$

where $w^{[l]}$ and $b^{[l]}$ correspond to the weight vector and bias vector of the $l-$th layer in the network, respectively, and $a^{[l-1]}$ represents the output of the first $l-1$ layers of the neural network. The neural network output $u_\theta(\mathbf{x}, t)$ is generated via layer-wise composition operations involving network parameters (weights $w^{[l]}$ and biases $b^{[l]}$) and the activation function $\sigma^{[l]}$, and has the following composite structure:

$$u_\theta(\mathbf{x}, t) := \mathcal{A}^L \circ \sigma^L \circ \mathcal{A}^{L-1} \circ \sigma^{L-1} \circ \cdots \circ \sigma \circ \mathcal{A}^0(\mathbf{x}, t),$$

with $\theta = \left[w^l, b^l\right]_{1\le l\le L}$. Based on the above construction scheme, the approximate solution to the PDE is parameterized by the neural network output, namely $\tilde{u}(\mathbf{x}, t) = u_\theta(\mathbf{x}, t)$. The core of the PINNs approach resides in embedding the a priori physical information inherent in the PDEs (i.e., the governing equations themselves and relevant physical laws) into the design of the neural network's loss function. Through iterative training to minimize this loss function, the network parameters are optimized to their optimal state, thereby yielding approximate numerical solutions that satisfy the PDE constraints.

We adopt the mean squared error (MSE) to formulate the loss function for the neural network,

$$\text{Loss}_u = \lambda_r \mathcal{L}_{pde} + \lambda_0 \mathcal{L}_{ic} + \lambda_b \mathcal{L}_{bc}.$$

Among

$$\begin{cases} \mathcal{L}_{pde} = \frac{1}{N_r} \sum_{i=1}^{N_r} \left[\tilde{u}_t(\mathbf{x}_i^r, t_i^r) + \mathcal{N}(\tilde{u}(\mathbf{x}_i^r, t_i^r) - h(\mathbf{x}_i^r, t_i^r))\right]^2, \\ \mathcal{L}_{ic} = \frac{1}{N_0} \sum_{i=1}^{N_0} \left[\tilde{u}(\mathbf{x}_i^0, t_i^0) - u_0(\mathbf{x}_i^r)\right]^2, \\ \mathcal{L}_{bc} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left[\tilde{u}(\mathbf{x}_i^b, t_i^b)\right]^2. \end{cases} \tag{3.2}$$

$\mathcal{L}_{pde}$, $\mathcal{L}_{ic}$, and $\mathcal{L}_{bc}$ represent the residual term (reflecting the neural network model's compliance to physical laws), the initial condition error term, and the boundary condition error term (ensuring boundary compatibility of the solution), respectively. $\left\{\left(\mathbf{x}_i^r, t_i^r\right)\right\}_{i=1}^{N_r}$ represents the set of sampling points within the solution domain $\Omega \times [0, T]$, the initial condition sampling points are denoted by $\left\{\left(\mathbf{x}_i^0, t_i^0\right)\right\}_{i=1}^{N_0} \in \Omega \times \{0\}$, and the boundary condition sampling points are defined as $\left\{\left(\mathbf{x}_i^b, t_i^b\right)\right\}_{i=1}^{N_b} \in \partial\Omega \times [0, T]$. Additionally, λ_r, λ_0, and λ_b denote the regularization coefficients, which are used to adjust the gradient contributions of each term in the loss function, thereby mitigating training imbalance arising from magnitude discrepancies.

The training process for approximating PDEs within the physics-informed neural network (PINNs) framework can be summarized as follows. Given the spatio-temporal coordinates $(\mathbf{x}, t)$, with the multilayer perceptron output acting as a candidate numerical solution $u_\theta(\mathbf{x}, t)$ for the PDE, the partial derivatives of the equation are computed via automatic differentiation (AD) and incorporated into the residual term of the loss function $Loss_u$. The network parameters θ are then iteratively updated using an appropriate optimizer. Through multiple training iterations, the loss function is minimized to determine the optimal parameters, ultimately yields the approximate solution $\tilde{u}(\mathbf{x}, t)$ to the equation.

The PINN framework converts PDEs into an optimization problem, where integer-order operators are efficiently computed via automatic differentiation (AD). This approach facilitates mesh-free approximation. Nonetheless, this paradigm encounters a fundamental barrier when extending

to fractional PDE (fPDE) solvers: Fractional differential operators are inherently non-local and memory-dependent. Since standard AD methods lack the capability to compute these global, integral-based operator forms through the chain rule, they cannot be directly embedded into the loss function, hindering the direct application of traditional PINNs to fPDEs.

In the literature [37], Lu et al. proposed fractional physics-informed neural networks (fPINNs) based on the finite difference method, in which the loss function is constructed by discretizing fractional differential operators, thereby overcoming the limitations of traditional PINNs for solving fPDEs.

3.2. Fractional Physics-Informed Neural Networks (fPINNs)

The fPINNs algorithm functions by numerically approximating fractional-order operators using methods like $L1$ difference scheme and shifted G-L formula, embedding these discrete forms directly into the neural network loss function. This approach resolves the limitations of traditional PINNs, which cannot compute fractional derivatives via automatic differentiation. By directly incorporating the nonlocal characteristics of fractional calculus, fPINNs enable the efficient solution of fractional PDEs, extending the applicability of PINNs to fractional-order systems.

To illustrate the fPINN approximation procedure, we consider the following general nonlinear fractional PDE with Dirichlet boundary conditions on a finite domain:

$$\begin{cases} \mathcal{N}\{u(\mathbf{x},t)\} = h(\mathbf{x},t), & (\mathbf{x},t)\in\Omega\times[0,T], \\ u(\mathbf{x},0) = u_0(\mathbf{x}), & \mathbf{x}\in\Omega, \\ u(\mathbf{x},t) = 0, & (\mathbf{x},t)\in\partial\Omega\times(0,T]. \end{cases} \tag{3.3}$$

Here, $\mathcal{N}$ represents a nonlinear differential operator acting on the unknown solution $u(\mathbf{x},t)$, which includes the fractional derivative terms. Depending on whether it is amenable to automatic differentiation, $\mathcal{N}$ can be decomposed into two components, $\mathcal{N}_{AD}$ and $\mathcal{N}_{nonAD}$, such that $\mathcal{N} = \mathcal{N}_{AD} + \mathcal{N}_{nonAD}$. In fPINNs, $\mathcal{N}_{AD}$ (e.g., u_t, $-\Delta$) denotes the integer-order derivative terms that can be automatically differentiated using the chain rule. Since Eq (2.1) is a general form, this symbol is adopted uniformly for all fractional cases. Specifically, it refers to the spatial integer-order operator for the time-fractional GLE, and corresponds to the temporal integer-order operator for the space-fractional GLE. For the operator $\mathcal{N}_{nonAD}$ (e.g., ${}_0^{\mathrm{C}}D_t^{\alpha}u$, $(-\Delta)^{\frac{\beta}{2}}$) that is not amenable to automatic differentiation , we adopt the FDMs for discretization (alternatively, FEMs or SMs can also be used for this purpose), which is then denoted as $\mathcal{N}_{FDM}$. We assume the approximate solution $\tilde{u}(\mathbf{x},t)$ to Eq (3.3) is the output $u_\theta(\mathbf{x},t)$ of the fPINNs, and by combining this with the loss function construction strategy for traditional PINNs, the corresponding loss function for fPINNs can be expressed as

$$\mathrm{Loss}_u = \lambda_r \mathrm{MSE}_{pde} + \lambda_0 \mathrm{MSE}_{ic} + \lambda_b \mathrm{MSE}_{bc}.$$

Among these, the forms of MSE_{ic} and MSE_{bc} take the same forms as the mean square error terms $\mathcal{L}_{ic}$ and $\mathcal{L}_{bc}$ for the initial and boundary conditions in Eq (3.2). In addition, MSE_{pde}, which consists of the

residual terms of the fractional PDE, is given by the following form:

$$\begin{aligned}\mathrm{MSE}_{pde} &= \frac{1}{N_r}\sum_{i=1}^{N_r}\left[\mathcal{N}(\tilde{u}(\mathbf{x}_i^r, t_i^r) - h(\mathbf{x}_i^r, t_i^r))\right]^2 \\ &= \frac{1}{N_r}\sum_{i=1}^{N_r}\left[\mathcal{N}_{AD}(\tilde{u}(\mathbf{x}_i^r, t_i^r) + \mathcal{N}_{nonAD}(\tilde{u}(\mathbf{x}_i^r, t_i^r) - h(\mathbf{x}_i^r, t_i^r))\right]^2 \\ &:= \frac{1}{N_r}\sum_{i=1}^{N_r}\left[\mathcal{N}_{AD}(\tilde{u}(\mathbf{x}_i^r, t_i^r) + \mathcal{N}_{FDM}(\tilde{u}(\mathbf{x}_i^r, t_i^r) - h(\mathbf{x}_i^r, t_i^r))\right]^2.\end{aligned}$$

The training process and flow chart for solving partial differential equations with fPINNs are provided in Algorithm 1 and Figure 1, respectively.

Algorithm 1 fPINN Algorithm for Solving PDEs.

input:

PDE-related: Training point sets, nonlinear term $\mathcal{N}_{AD}$, $\mathcal{N}_{nonAD}$.

Neural Network Structure: Activation function σ, input layer,hidden layers,output layer, initialize network parameters θ_0.

Optimization: Optimizer, learning rate δ, iterations max_{it}.

output:

Optimal Network Parameters.

Given initialization parameters: θ gets $\theta_0 = [w^0, b^0]$.

Obtain the network output solution $u_\theta(\mathbf{x}, t)$ by processing the input variables $\mathbf{x}$ and t through the hidden layers of the network.

Construct the approximate solution: $\tilde{u}(\mathbf{x}, t) = u_\theta(\mathbf{x}, t)$.

for i in range(max_it) **do**

The loss function:

$\mathrm{Loss}_u \leftarrow \lambda_r \mathrm{MSE}_{pde} + \lambda_0 \mathrm{MSE}_{ic} + \lambda_{bc} \mathrm{MSE}_b$,

$\mathrm{MSE}_{pde} := \frac{1}{N_r}\sum_{i=1}^{N_r}\left[\mathcal{N}_{AD}(\tilde{u}(\mathbf{x}_i^r, t_i^r) + \mathcal{N}_{FDM}(\tilde{u}(\mathbf{x}_i^r, t_i^r) - h(\mathbf{x}_i^r, t_i^r))\right]^2$,

$\mathrm{MSE}_{ic} = \frac{1}{N_0}\sum_{i=1}^{N_0}\left[\tilde{u}(\mathbf{x}_i^0, t_i^0) - u_0(\mathbf{x}_i^r)\right]^2$,

$\mathrm{MSE}_{bc} = \frac{1}{N_b}\sum_{i=1}^{N_b}\left[\tilde{u}(\mathbf{x}_i^b, t_i^b)\right]^2$.

N_r, N_0, N_b represent the number of training points for the spatio-temporal interior, initial condition and boundary condition, respectively.

Minimize the loss function using an optimizer to train the neural network.

adjust network parameters: θ gets $\theta_0 - \delta \bigtriangledown \theta$.

end for

return Network Parameters θ^*.

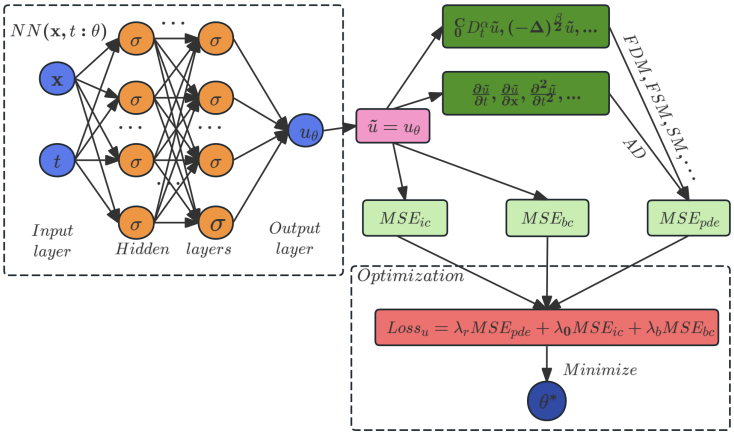


Figure 1. Flowchart of the fPINNs algorithm for solving fPDEs.

In neural-network-based numerical solvers, the construction of the approximate solution significantly influences training dynamics and overall performance. Depending on whether the approximation is constrained to satisfy initial (IC) and boundary conditions (BC), these constructions are classified into four scenarios:

- **Unconstrained (Soft constraints):** The approximation is the direct network output, $\tilde{u}(\mathbf{x}, t) = u_\theta(\mathbf{x}, t)$. The loss function must simultaneously optimize residuals for the governing equations,
- **ICs, and BCs.IC-Satisfying (Hard IC):** The approximation is structured as $\tilde{u}(\mathbf{x}, t) = tu_\theta(\mathbf{x}, t) + u_0(\mathbf{x})$, where $u_0(x)$ is the initial condition. This removes the IC error from the loss function, leaving only governing equation and BC terms.
- **BC-Satisfying (Hard BC):** The approximation takes the form $\tilde{u}(\mathbf{x}, t) = \rho(\mathbf{x})u_\theta(\mathbf{x}, t)$, where $\rho(x)$ is a distance function that vanishes on the boundary $\partial\Omega$. Consequently, the loss function only accounts for governing equation and IC residuals.
- **Fully constrained (Hard IC and BC):** The approximation is formulated as $\tilde{u}(\mathbf{x}, t) = t\rho(\mathbf{x})u_\theta(\mathbf{x}, t) + u_0(\mathbf{x})$, satisfying both ICs and BCs by design. This dual hard constraint approach restricts the loss function solely to the governing equation's residual.

Generally, reducing the number of constraint terms in the loss function, by explicitly embedding initial/boundary conditions into the network architecture, lowers the model's effective degrees of freedom and focuses training on satisfying physical laws. Consequently, this approach tends to yield lower approximation errors. Therefore, we design the network to natively satisfy the governing equations, improving both training accuracy and convergence efficiency.

4. Finite difference methods (FDMs)

In this section, we establish the theoretical foundation for our numerical approach by deriving three discrete Caputo time-fractional schemes alongside a G-L formulation for the spatial fractional Laplacian.

4.1. Discretization of time-fractional derivatives

For the Caputo-type fractional derivatives defined in Eq (2.2), we consider only problems with sufficiently smooth solutions, which satisfy the regularity conditions required by the Caputo-type discretization methods. We employ the $L2\text{-}1_\sigma$ scheme for discretization, and compare this discretization approach with the $L1$ and $L1$-2 difference schemes.

4.1.1. $L1$ formula

Let N_t be a positive integer, and define the time step $\Delta t = \frac{T}{N_t}$ and the time nodes $t_i = i\Delta t$, $i = 0, 1, \ldots, N_t$. We also define the coefficients $a_i^{(\alpha)} = (i+1)^{1-\alpha} - i^{1-\alpha}$, $i \geq 0$. Then, we have [41]

$$ {}_0^C D_t^\alpha u(\mathbf{x}, t_k) = \frac{1}{\Gamma(1-a)} \int_0^{t_k} \frac{u_s'(\mathbf{x}, s)}{(t_k - s)^\alpha} \mathrm{d}s = \frac{1}{\Gamma(1-a)} \sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} \frac{u_s'(\mathbf{x}, s)}{(t_k - s)^\alpha} \mathrm{d}s. \tag{4.1} $$

Linear interpolation of the function $u(\mathbf{x}, t)$ over the interval $[t_{i-1}, t_i]$ yields

$$ L_{1,i}(\mathbf{x}, t) = \frac{t_i - t}{\Delta t} u(\mathbf{x}, t_{i-1}) + \frac{t - t_{i-1}}{\Delta t} u(\mathbf{x}, t_i). \tag{4.2} $$

By using the above linear interpolation formula $L_{1,i}(\mathbf{x}, t)$ to approximate $u(\mathbf{x}, t)$ in the computation of the Caputo fractional derivative, we have

$$ \begin{aligned} {}_0^C D_t^\alpha u(\mathbf{x}, t_k) &\approx \frac{1}{\Gamma(1-\alpha)} \sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} \frac{\partial L_{1,i}(\mathbf{x}, s)}{\partial s} (t_k - s)^{-\alpha} \mathrm{d}s \\ &= \frac{\Delta t^{-a}}{\Gamma(2-\alpha)} \sum_{i=1}^{k} a_{k-i}^{(\alpha)} \left[u(\mathbf{x}, t_i) - u(\mathbf{x}, t_{i-1}) \right] \\ &= \frac{\Delta t^{-\alpha}}{\Gamma(2-\alpha)} \left[a_0^{(\alpha)} u(\mathbf{x}, t_k) - \sum_{i=1}^{k-1} \left(a_{k-i-1}^{(\alpha)} - a_{k-i}^{(\alpha)} \right) u(\mathbf{x}, t_i) - a_{k-1}^{(\alpha)} u(\mathbf{x}, t_0) \right]. \end{aligned} \tag{4.3} $$

4.1.2. $L1$-2 formula

Using the same temporal mesh partitioning as for the $L1$ scheme, we adopt a uniform temporal grid with $\Delta t = \frac{T}{N_t}$, and grid nodes $t_i = i\Delta t$, for $i = 0, 1, \ldots, N_t$. Before introducing the $L1$-2 difference scheme [42], we first define the following difference quotient operators for $k \geq 1$

$$ \begin{aligned} &u(\mathbf{x}, t_{i-\frac{1}{2}}) = \frac{u(\mathbf{x}, t_i) + u(\mathbf{x}, t_{i-1})}{2}, \quad \delta_t u(\mathbf{x}, t_{i-\frac{1}{2}}) = \frac{u(\mathbf{x}, t_i) - u(\mathbf{x}, t_{i-1})}{\Delta t}, \\ &\delta_t^2 u(\mathbf{x}, t_i) = \frac{\delta_t u(\mathbf{x}, t_{i+\frac{1}{2}}) - \delta_t u(\mathbf{x}, t_{i-\frac{1}{2}})}{\Delta t}. \end{aligned} \tag{4.4} $$

Within the interval $[t_0, t_1]$, we approximate $u(\mathbf{x}, t)$ by the linear interpolation function $L_{1,i}(\mathbf{x}, t)$ given in Eq (4.2). The partial derivative of $L_{1,i}(\mathbf{x}, t)$ with respect to t is given by

$$ \frac{d}{dt} \left(L_{1,1}(\mathbf{x}, t) \right) = \frac{u(\mathbf{x}, t_1) - u(\mathbf{x}, t_0)}{\Delta t} = \delta_t u(\mathbf{x}, t_{\frac{1}{2}}). $$

For the interval $[t_{i-1}, t_i]$ with $i \geq 2$, we construct the quadratic interpolation function $L_{2,i}(\mathbf{x}, t)$ by using the function values of $u(\mathbf{x}, t)$ at the nodes $(\mathbf{x}, t_{i-2})$, $(\mathbf{x}, t_{i-1})$, and $(\mathbf{x}, t_i)$, and restrict this function to the interval $[t_{i-1}, t_i]$.

$$\begin{aligned} L_{2,i}u(\mathbf{x}, t) &= u(\mathbf{x}, t_{i-2})\frac{(t-t_{i-1})(t-t_i)}{2\Delta t^2} + u(\mathbf{x}, t_{i-1})\frac{(t-t_{i-2})(t_i-t)}{\Delta t^2} + u(\mathbf{x}, t_i)\frac{(t-t_{i-1})(t-t_{i-2})}{2\Delta t^2} \\ &= L_{1,i}u(\mathbf{x}, t) + \frac{1}{2}\left(\delta_t^2 u(\mathbf{x}, t_{i-1})\right)(t-t_{i-1})(t-t_i). \end{aligned} \tag{4.5}$$

The first-order partial derivative of $L_{2,i}u(\mathbf{x}, t)$ with respect to t is

$$\frac{d}{dt}\left(L_{2,i}u(\mathbf{x}, t)\right) = \delta_t u(\mathbf{x}, t_{i-\frac{1}{2}}) + \delta_t^2 u(\mathbf{x}, t_{i-1})(t - t_{i-\frac{1}{2}}). \tag{4.6}$$

For the *Caputo* fractional derivative at the time node t_k, we approximate $u(\mathbf{x}, t_{i-\frac{1}{2}})$ with $L_{1,i}u(\mathbf{x}, t)$ over the interval $[t_0, t_1]$ and with $L_{2,i}u(\mathbf{x}, t)$ over the interval $[t_{i-1}, t_i]$ for $i \geq 2$, which yields the $L1$-2 difference scheme in the following form:

$$\begin{aligned} {}_0^C D_t^\alpha u(\mathbf{x}, t_k) &= \frac{1}{\Gamma(1-\alpha)}\sum_{i=1}^{k}\int_{t_{i-1}}^{t_i}\frac{u_t'(\mathbf{x}, s)}{(t_k-s)^\alpha}\,dss \\ &\approx \frac{1}{\Gamma(1-\alpha)}\left[\int_{t_0}^{t_1}\frac{\partial(L_{1,i}(\mathbf{x}, s))}{\partial s}(t_k-s)^{-\alpha}\,ds + \sum_{i=2}^{k}\int_{t_{i-1}}^{t_i}\frac{\partial(L_{2,i}(\mathbf{x}, s))}{\partial s}(t_k-s)^{-\alpha}\,ds\right] \\ &= \frac{1}{\Gamma(1-\alpha)}\left\{\delta_t u(\mathbf{x}, t_{i-\frac{1}{2}})\int_{t_0}^{t_1}(t_k-s)^{-\alpha}\,ds\right. \\ &\quad \left. + \sum_{i=2}^{k}\int_{t_{i-1}}^{t_i}(t_k-s)^{-\alpha}\left[\delta_t u(\mathbf{x}, t_{i-\frac{1}{2}}) + \delta_t^2 u(\mathbf{x}, t_{i-1})(s-t_{i-\frac{1}{2}})\right]ds\right\} \\ &= \frac{1}{\Gamma(1-\alpha)}\left[\sum_{i=1}^{k}\delta_t u(\mathbf{x}, t_{i-\frac{1}{2}})\int_{t_{i-1}}^{t_i}(t_k-s)^{-\alpha}\,ds + \sum_{i=2}^{k}\delta_t^2 u(\mathbf{x}, t_{i-1})\int_{t_{i-1}}^{t_i}(s-t_{i-\frac{1}{2}})(t_k-s)^{-\alpha}\,ds\right] \\ &= D_t^\alpha u(\mathbf{x}, t_k) + \frac{\Delta t^{2-\alpha}}{\Gamma(2-\alpha)}\sum_{i=2}^{k} b_{k-i}^{(\alpha)}\delta_t^2 u(\mathbf{x}, t_{i-1}) \\ &= D_t^\alpha u(\mathbf{x}, t_k) + \frac{\Delta t^{-\alpha}}{\Gamma(2-\alpha)}\sum_{i=2}^{k} b_{k-i}^{(\alpha)}(u(\mathbf{x}, t_i) - 2u(\mathbf{x}, t_{i-1}) + u(\mathbf{x}, t_{i-2})), \end{aligned}$$

of which $D_t^\alpha(u(\mathbf{x}, t_k)$ denotes the classical $L1$ formula given in Eq (4.3), i.e.,

$$D_t^\alpha u(\mathbf{x}, t_k) = \frac{\Delta t^{-a}}{\Gamma(2-\alpha)}\sum_{i=1}^{k} a_{k-i}^{(\alpha)}\left[u(\mathbf{x}, t_i) - u(\mathbf{x}, t_{i-1})\right] = \frac{\Delta t^{1-a}}{\Gamma(2-\alpha)}\sum_{i=1}^{k} a_{k-i}^{(\alpha)}\delta_t u(\mathbf{x}, t_{i-\frac{1}{2}}).$$

For any fractional order α with $\alpha(0 < \alpha \leq 1)$,

$$\begin{aligned} a_i^{(\alpha)} &= (i+1)^{1-\alpha} - i^{1-\alpha}, \quad 0 \leq i \leq k-1. \\ b_i^{(\alpha)} &= \frac{1}{2-\alpha}\left[(i+1)^{2-\alpha} - i^{2-\alpha}\right] - \frac{1}{2}\left[(i+1)^{1-\alpha} + i^{1-\alpha}\right], \quad i \geq 0. \end{aligned}$$

From the above expressions, we can rewrite the $L1$-2 difference formula as

$$\begin{aligned}
{}_0^C D_t^\alpha u(\mathbf{x}, t_k) &= D_t^\alpha u(\mathbf{x}, t_k) + \frac{\Delta t^{2-\alpha}}{\Gamma(2-\alpha)} \sum_{i=2}^{k} b_{k-i}^{(\alpha)} \delta_t^2 u(\mathbf{x}, t_{i-1}) \\
&= \frac{\Delta t^{1-a}}{\Gamma(2-\alpha)} \left[\sum_{i=1}^{k} a_{k-i}^{(\alpha)} \delta_t u(\mathbf{x}, t_{i-\frac{1}{2}}) + \sum_{i=2}^{k} b_{k-i}^{(\alpha)} (\delta_t u(\mathbf{x}, t_{i-\frac{1}{2}}) - \delta_t u(\mathbf{x}, t_{i-\frac{3}{2}})) \right] \\
&= \frac{\Delta t^{-\alpha}}{\Gamma(2-\alpha)} \left[c_0^{(\alpha)} u(\mathbf{x}, t_k) - \sum_{i=1}^{k-1} \left(c_{k-i-1}^{(\alpha)} - c_{k-i}^{(\alpha)} \right) u(\mathbf{x}, t_i) - c_{k-1}^{(\alpha)} u(\mathbf{x}, t_0) \right],
\end{aligned} \tag{4.7}$$

where the coefficient $c_i^{(\alpha)}$ satisfies $c_0^{(\alpha)} = a_0^{(\alpha)} = 1$ for $k = 1$, and for $k \geq 2$,

$$c_i^{(\alpha)} = \begin{cases} a_0^{(\alpha)} + b_0^{(\alpha)}, & i = 0. \\ a_i^{(\alpha)} + b_i^{(\alpha)} - b_{i-1}^{(\alpha)}, & 1 \leq i \leq k-2. \\ a_i^{(\alpha)} - b_{i-1}^{(\alpha)}, & i = k-1. \end{cases}$$

4.1.3. $L2$-1_σ formula

We consider an equidistant temporal grid with $\Delta t = \frac{T}{N_t}$, and $t_i = i\Delta t, i = 0, 1, \ldots, N_t$. Furthermore, let $\sigma = 1 - \alpha/2$, and denote $t_{i+\sigma} = (i+\sigma)\Delta t, i = 0, 1, \ldots, N_t - 1$. The grid function $u(\mathbf{x}, t_i)$ for $0 \leq i \leq N_t$ satisfies Eq (4.4). We then have [43]

$$\begin{aligned}
{}_0^C D_t^\alpha u(\mathbf{x}, t_{k+\sigma}) &= \frac{1}{\Gamma(1-\alpha)} \int_0^{t_{k+\sigma}} \frac{u_s'(\mathbf{x}, s)}{(t_{k+\sigma} - s)^\alpha} \mathrm{d}s \\
&= \frac{1}{\Gamma(1-\alpha)} \left[\sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} \frac{u_s'(\mathbf{x}, s)}{(t_{k+\sigma} - s)^\alpha} \mathrm{d}s + \int_{t_k}^{t_{k+\sigma}} \frac{u_s'(\mathbf{x}, s)}{(t_{k+\sigma} - s)^\alpha} \mathrm{d}s \right].
\end{aligned}$$

For each interval $[t_{i-1}, t_i]$ with $1 \leq i \leq k$, we approximate $u(\mathbf{x}, t)$ by the quadratic interpolation formula Eq (4.5) given in the previous subsection. We then perform linear interpolation $u(\mathbf{x}, t)$ via the linear interpolation formula Eq (4.2) using the two grid nodes $(\mathbf{x}, t_k)$ and $(\mathbf{x}, t_{k+1})$ over the interval $[t_k, t_{k+\sigma}]$. Hence, we obtain the $L2$-1_σ formula in the following form:

$$\begin{aligned}
{}_0^C D_t^\alpha u(\mathbf{x}, t_{k+\sigma}) &= \frac{1}{\Gamma(1-\alpha)} \left[\sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} \frac{u_s'(\mathbf{x}, s)}{(t_{k+\sigma} - s)^\alpha} \mathrm{d}s + \int_{t_k}^{t_{k+\sigma}} \frac{u_s'(\mathbf{x}, s)}{(t_{k+\sigma} - s)^\alpha} \mathrm{d}s \right] \\
&= \frac{1}{\Gamma(1-\alpha)} \left[\sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} \frac{\partial L_{2,i}(\mathbf{x}, s)}{\partial s} (t_{k+\sigma} - s)^{-\alpha} \, \mathrm{d}s \right] \\
&\quad + \frac{1}{\Gamma(1-\alpha)} \left[\int_{t_k}^{t_{k+\sigma}} \frac{\partial L_{1,k+1}(\mathbf{x}, s)}{\partial s} (t_{k+\sigma} - s)^{-\alpha} \, \mathrm{d}s \right] \\
&= \frac{1}{\Gamma(1-\alpha)} \left[\sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} (\delta_t u(\mathbf{x}, s_{i-\frac{1}{2}}) + \delta_t^2 u(\mathbf{x}, s_{i-1})(s - t_{i-\frac{1}{2}})) (t_{k+\sigma} - s)^{-\alpha} \, \mathrm{d}s \right]
\end{aligned}$$

$$
\begin{aligned}
&+\frac{1}{\Gamma(1-\alpha)}\left[\int_{t_k}^{t_{k+\sigma}}\delta_t u(\mathbf{x}, s_{k+\frac{1}{2}})\,(t_{k+\sigma}-s)^{-\alpha}\,\mathrm{d}s\right]\\
&=\frac{\Delta t^{1-\alpha}}{\Gamma(1-\alpha)}\sum_{i=1}^{k+1}c_{k-i+1}^{(k,\alpha,\sigma)}\delta_t u(\mathbf{x}, t_{i-\frac{1}{2}})\\
&=\frac{\Delta t^{1-\alpha}}{\Gamma(1-\alpha)}\sum_{i=0}^{k}c_{i}^{(k,\alpha,\sigma)}\delta_t u(\mathbf{x}, t_{k-i+\frac{1}{2}})\\
&=\frac{\Delta t^{\alpha}}{\Gamma(1-\alpha)}\sum_{i=0}^{k}c_{i}^{(k,\alpha,\sigma)}(u(\mathbf{x}, t_{k+1-i})-u(\mathbf{x}, t_{k-i})).
\end{aligned}
\tag{4.8}
$$

Let

$$
\begin{aligned}
&a_0^{(\alpha,\sigma)}=\sigma^{1-\alpha},\quad a_i^{\alpha,\sigma}=(i+\sigma)^{1-\alpha}-(i-1+\sigma)^{1-\alpha},\quad i\geq 1,\\
&b_i^{(\alpha,\sigma)}=\frac{(i+\sigma)^{2-\alpha}-(i-1+\sigma)^{2-\alpha}}{2-\alpha}-\frac{(i+\sigma)^{1-\alpha}+(i-1+\sigma)^{1-\alpha}}{2},\quad i\geq 1.
\end{aligned}
$$

Therefore, the constant term coefficient $c_i^{(k,\alpha,\sigma)}$ in Eq (4.8) is given by the following form: $c_0^{(0,\alpha,\sigma)}=a_0^{(\alpha,\sigma)}$, for $k=0$, and for $k\geq 1$,

$$
c_i^{(k,\alpha,\sigma)}=\begin{cases}a_0^{(\alpha,\sigma)}+b_1^{(\alpha,\sigma)}, & i=0,\\ a_i^{(\alpha,\sigma)}+b_{i+1}^{(\alpha,\sigma)}-b_i^{(\alpha,\sigma)}, & 1\leq i\leq k-1,\\ a_k^{(\alpha,\sigma)}-b_k^{(\alpha,\sigma)}, & i=k.\end{cases}
\tag{4.9}
$$

A common pattern emerges across the three difference schemes: Given a spatial coordinate $\mathbf{x}$, when discretizing the Caputo derivative at any time t, we observe that the time-fractional derivative of the approximate solution $\tilde{u}(\mathbf{x}, t)$ at t depends on the numerical solutions at all historical time instants $(0, \Delta t, 2\Delta t, \ldots, t)$. Within neural network construction, we define the target time step t as the training point, while the preceding instants $(0, \Delta t, \ldots)$ are treated as auxiliary points corresponding to t.

To clarify the implementation:

- **Precompute auxiliary values:** In each iteration, we run a single forward pass of network output solution $u_\theta(\mathbf{x}, t)$ over all time points $0, \Delta t, \ldots, T$ to store predictions at all auxiliary points.
- **Compute residuals:** At each target t, we use stored values $u_\theta(\mathbf{x}, 0), u_\theta(\mathbf{x}, \Delta t), \ldots, u_\theta(\mathbf{x}, t)$ to approximate the Caputo time fractional derivative via finite differences.
- **Gradient flow:** Backpropagation uses stored values at all auxiliary points as constants, avoiding gradient flow through the entire historical chain.

This design explicitly organizes auxiliary points within the neural network framework, thereby ensuring the reproducibility of our implementation.

4.2. Discretization of spatial fractional derivatives

Regarding the discretization of the space fractional Laplace operator Eq (2.3), it is first necessary to perform an interpolation approximation for the Riemann-Liouvile directional fractional derivative $D_{\Phi}^{\beta}u(\mathbf{x}, t)$. Its definition is given as follows:

$$
D_{\mathbf{\Phi}}^{\beta}u(\mathbf{x}, t)=(\mathbf{\Phi}\cdot\nabla)^2\left(\frac{1}{\Gamma(2-\beta)}\int_0^{+\infty}\xi^{1-\beta}u(\mathbf{x}-\xi\mathbf{\Phi}, t)\mathrm{d}\xi\right),\quad \mathbf{x}, \xi\in\mathbb{R}^d.
\tag{4.10}
$$

Here, the spatial variable $\mathbf{x}$ and the directional variable ξ lie in the d-dimensional Euclidean space $\mathbb{R}^d$. $\mathbf{\Phi}$ denotes the differential direction, whose value varies with the spatial dimension is defined as $\mathbf{\Phi} = \cos\phi$ where $\phi = 0, \pi$ for $d = 1$, $\mathbf{\Phi} = [\cos\phi, \sin\phi]$ where $\phi \in [0, 2\pi)$ for $d = 2$. ∇ denotes the first-order gradient operator, and $(\mathbf{\Phi}\cdot\nabla)$ denotes the dot product of two vectors, which is defined as $(\mathbf{\Phi}\cdot\nabla) = \sum_{i=1}^{d}\phi_i\frac{\partial}{\partial x_i}$. For example, when $d=1$, $(\mathbf{\Phi}\cdot\nabla) = \cos\phi\frac{\partial}{\partial x}$; when $d=2$, $(\mathbf{\Phi}\cdot\nabla) = \cos\phi\frac{\partial}{\partial x_1} + \sin\phi\frac{\partial}{\partial x_2}$. Furthermore,

$$(\mathbf{\Phi}\cdot\nabla)^2 = \sum_{i=1}^{d}\phi_i\frac{\partial}{\partial x_i}\sum_{j=1}^{d}\phi_j\frac{\partial}{\partial x_j} = \sum_{j=1}^{d}\sum_{i=1}^{d}\phi_j\phi_i\frac{\partial^2}{\partial x_j\partial x_i}.$$

Since $u(\mathbf{x}, t)$ vanishes outside the computational domain, the aforementioned fractional derivative Eq (4.10) can be rewritten as

$$D^{\alpha}_{\mathbf{\Phi}}u(\mathbf{x}, t) = (\mathbf{\Phi}\cdot\nabla)^2\left(\frac{1}{\Gamma(2-\beta)}\int_0^{d(\mathbf{x},\mathbf{\Phi},\Omega)}\xi^{1-\beta}u(\mathbf{x}-\xi\mathbf{x}, t)\mathrm{d}\xi\right), \quad \mathbf{x}\in\Omega\subset\mathbb{R}^d. \tag{4.11}$$

$d(\mathbf{x}, \mathbf{\Phi}, \Omega)$ denotes the distance from the direction $-\mathbf{\Phi}$ to the boundary $\partial\Omega$ along the spatial coordinate $\mathbf{x}$, which is referred to as the backward distance. Let the spatial step size be Δx. Then, we have $\mathbf{x}_i = \mathbf{x}_0 + i\Delta x$ for $i = 0, 1, \ldots, N_x$, where x_0 denotes a boundary point of the computational domain. Discretizing the fractional derivative Eq (4.11) via the shifted Grünwald-Letnikov (G-L) formula [40] yields

$$D^{\beta}_{\mathbf{\Phi}}u(\mathbf{x}_k, t) = \frac{1}{(\Delta x)^{\beta}}\sum_{i=1}^{\left\lceil\frac{d(\mathbf{x}_k,\mathbf{\Phi},\Omega)}{\Delta x}\right\rceil}c_i^{(\beta)}u(\mathbf{x}_k-(i-1)\Delta x\mathbf{\Phi}, t) + O(\Delta x). \tag{4.12}$$

Here, $\left\{c_i^{(\beta)}\right\}$ denotes the coefficients in the binomial power series expansion of $(1-z)^{\beta}$, with $c_i^{(\beta)}u = (-1)^i\binom{\beta}{i}$, and $\binom{\beta}{i} = \frac{\beta(\beta-1)\cdots(\beta-i+1)}{i!}$. These coefficients satisfy the following recurrence relation:

$$c_0^{(\beta)} = 1,\ c_i^{(\beta)} = \left(1-\frac{\beta+1}{i}\right)c_{i-1}^{(\beta)}.$$

Similar to the auxiliary points in the time-fractional discretization systems, at a fixed time, for each training point $(\mathbf{x}, t)$, there exist corresponding auxiliary points $(\mathbf{x} - i\Delta x\mathbf{\Phi}, t)$ along the direction of spatial discretization. Substituting Eq (4.12) into the expression of the spatial fractional Laplace operator Eq (2.3) yields

$$\begin{aligned}(-\Delta)^{\frac{\beta}{2}}u(\mathbf{x}_k, t) &= C_{\beta,d}\int_{\|\mathbf{\Phi}\|_2=1}D^{\beta}_{\mathbf{\Phi}}u(\mathbf{x}, t)d\mathbf{\Phi}\\ &= C_{\beta,d}\int_{\|\mathbf{\Phi}\|_2=1}\left\{\frac{1}{(\Delta x)^{\beta}}\sum_{i=1}^{\left\lceil\frac{d(\mathbf{x}_k,\mathbf{\Phi},\Omega)}{\Delta x}\right\rceil}c_i^{(\beta)}u(\mathbf{x}_k-(i-1)\Delta x\mathbf{\Phi}, t)\right\}d\mathbf{\Phi}.\end{aligned} \tag{4.13}$$

Consider the one-dimensional case for simplicity, with $\Omega = [a, b]$, and $x_i = a + i\Delta x, i = 0, 1, \ldots, N_x$, and $N_x = b$. The reflection formula for the $\gamma(\cdot)$ function is given by $\Gamma(s)\Gamma(1-s) = \frac{\pi}{\sin(\pi s)}$, thus

$$C_{\beta,1} = \frac{\Gamma\left(\frac{1-\beta}{2}\right)\Gamma\left(\frac{1+\beta}{2}\right)}{2\pi} = \frac{\Gamma\left(\frac{1-\beta}{2}\right)\Gamma\left(1-\frac{1-\beta}{2}\right)}{2\pi} = \frac{\frac{\pi}{\sin(\pi\frac{1-\beta}{2})}}{2\pi} = \frac{1}{2\sin(\frac{\pi}{2}-\frac{\pi\beta}{2})} = \frac{1}{2\cos(\frac{\pi\beta}{2})}.$$

At this point, the directional fractional Laplacian operator takes the form

$$(-\Delta)^{\frac{\beta}{2}}u(\mathbf{x},t)=\frac{1}{2\cos(\frac{\pi\beta}{2})}(D^{\beta}_{\mathbf{\Phi}=1}u(\mathbf{x},t)+D^{\beta}_{\mathbf{\Phi}=-1}u(\mathbf{x},t)).$$

In this case, the backward distances are given by $d(x,1,[a,b])=x-a$ and $d(x,-1,[a,b])=b-x$. Therefore, the left- and right-Riemann-Liouvile directional fractional derivatives $D^{\beta}_{\mathbf{\Phi}=1}$ and $D^{\beta}_{\mathbf{\Phi}=-1}$ are expressed as follows:

$$D^{\beta}_{\mathbf{\Phi}=1}u(\mathbf{x},t)=\frac{\partial^2}{\partial x^2}\left[\frac{1}{\Gamma(2-\beta)}\int_0^{d(x,1,[a,b])}\xi^{1-\beta}u(\mathbf{x}-\xi,t)\mathrm{d}\xi\right]=\frac{1}{\Gamma(2-\beta)}\frac{\partial^2}{\partial x^2}\int_0^{x-a}\xi^{1-\beta}u(\mathbf{x}-\xi,t)\mathrm{d}\xi.$$
$$D^{\beta}_{\mathbf{\Phi}=-1}u(\mathbf{x},t)=\frac{\partial^2}{\partial x^2}\left[\frac{1}{\Gamma(2-\beta)}\int_0^{d(x,-1,[a,b])}\xi^{1-\beta}u(\mathbf{x}-\xi,t)\mathrm{d}\xi\right]=\frac{1}{\Gamma(2-\beta)}\frac{\partial^2}{\partial x^2}\int_0^{b-x}\xi^{1-\beta}u(\mathbf{x}-\xi,t)\mathrm{d}\xi.$$

For any x_i, we have $d(x_i,1,[a,b])=x_i-a=i\Delta x$ and $d(x_i,-1,[a,b])=b-x_i=N_x-i$. Thus, from Eq (4.13), the discrete form of the directional fractional Laplace operator in one-dimensional space can be derived as

$$\begin{aligned}
&(-\Delta)^{\frac{\beta}{2}}u(\mathbf{x}_k,t)\\
&=\frac{1}{2\cos(\frac{\pi\beta}{2})(\Delta x)^{\beta}}\left[\sum_{i=1}^{\left\lceil\frac{d(\mathbf{x}_k,1,[a,b])}{\Delta x}\right\rceil}c_i^{(\beta)}u(\mathbf{x}_k-(i-1)\Delta x,t)+\sum_{i=1}^{\left\lceil\frac{d(\mathbf{x}_k,-1,[a,b])}{\Delta x}\right\rceil}c_i^{(\beta)}u(\mathbf{x}_k+(i-1)\Delta x,t)\right]\\
&=\frac{1}{2\cos(\frac{\pi\beta}{2})(\Delta x)^{\beta}}\left[\sum_{i=1}^{\left\lceil\frac{x_k-a}{\Delta x}\right\rceil}c_i^{(\beta)}u(\mathbf{x}_k-(i-1)\Delta x,t)+\sum_{i=1}^{\left\lceil\frac{b-x_k}{\Delta x}\right\rceil}c_i^{(\beta)}u(\mathbf{x}_k+(i-1)\Delta x,t)\right]\\
&=\frac{1}{2\cos(\frac{\pi\beta}{2})(\Delta x)^{\beta}}\left[\sum_{i=0}^{k}c_i^{(\beta)}u(\mathbf{x}_{k-i+1},t)+\sum_{i=0}^{N_x-k}c_i^{(\beta)}u(\mathbf{x}_{k+i-1},t)\right].
\end{aligned}\tag{4.14}$$

Notably, the numerical experiments for the temporal fractional and spatial fractional GLEs are conducted separately. For the temporal fractional GLEs, only the time derivative is fractional, while the spatial derivatives are treated as standard integer-order terms. For the spatial fractional GLEs, only the spatial derivative is fractional, while the time derivative remains integer-order. Consequently, the convergence accuracies in time and space do not affect each other.

5. Numerical experiments

This section validates the effectiveness of fPINNs through detailed numerical simulations of 1D spatial and 2D temporal fractional GLEs, respectively. We adopt finite differences to handle fractional derivative terms of N_{nonAD}. Benefiting from the mesh-free property of neural networks, this method relieves the curse of dimensionality. Batch forward propagation is used in the calculation of L^2 relative error to avoid video memory overflow. A comparative analysis of the temporal discretization methods (Section 4.1) confirms that the $L2\text{-}1_{\sigma}$ scheme provides superior accuracy for the fPINNs model. We adopt the L^2 relative error, $\mathcal{E}=\frac{\|u-\tilde{u}\|_2}{\|u\|_2}$ (where u and $\tilde{u}$ are the exact and numerical solutions, respectively),

as the standard metric for quantitatively evaluating the approximation precision and numerical stability of the proposed approach.

Before presenting numerical results, note that Eq (2.1) involves the imaginary unit i, making its solution u a complex-valued function. Following the method in [34], we decompose u into its real part f and imaginary part g. Our complex-valued neural network adopts this decomposition by splitting the network output u_θ into two components, $u_\theta = [f_\theta, g_\theta]$. The approximate solution to the governing equation is then defined as $\tilde{u} = \left[\tilde{f}, \tilde{g}\right]$.

Furthermore, denote the forced term $h(\mathbf{x}, t)$ in Eq (2.1) as $\left[h_f, h_g\right]$, where h_f and h_g represent the real and imaginary parts of the source term $h(\mathbf{x}, t)$, respectively. Similarly, the initial function $u_0(\mathbf{x})$ is expressed as $[f_0(\mathbf{x}), g_0(\mathbf{x})]$, where f_0 and g_0 denote the real and imaginary parts of $u_0(\mathbf{x})$, respectively.

Since the solution to the equation is complex-valued, we can construct the loss function as the sum of the real-part MSE and the imaginary-part MSE. This is equivalent to the mean square modulus of the complex prediction error, expressed as follows:

$$\mathrm{MSE}_{GLE} = \lambda_r \mathrm{MSE}_r + \lambda_0 \mathrm{MSE}_0 + \lambda_b \mathrm{MSE}_b,$$

where

$$\begin{aligned}
\mathrm{MSE}_r &= \sqrt{\left(\frac{1}{N_r}\sum_{i=1}^{N_r}\left[gle_f(\mathbf{x}_i^r, t_i^r)\right]^2\right)^2 + \left(\frac{1}{N_r}\sum_{i=1}^{N_r}\left[gle_g(\mathbf{x}_i^r, t_i^r)\right]^2\right)^2}, \\
\mathrm{MSE}_0 &= \sqrt{\left(\frac{1}{N_0}\sum_{i=1}^{N_0}\left[\tilde{f}(\mathbf{x}_i^0, t_i^0) - f_0(\mathbf{x}_i^0)\right]^2\right)^2 + \left(\frac{1}{N_0}\sum_{i=1}^{N_0}\left[\tilde{g}(\mathbf{x}_i^0, t_i^0) - g_0(\mathbf{x}_i^0)\right]^2\right)^2}, \\
\mathrm{MSE}_b &= \sqrt{\left(\frac{1}{N_b}\sum_{i=1}^{N_b}\left[\tilde{f}(\mathbf{x}_i^b, t_i^b)\right]^2\right)^2 + \left(\frac{1}{N_b}\sum_{i=1}^{N_b}\left[\tilde{g}(\mathbf{x}_i^b, t_i^b)\right]^2\right)^2}.
\end{aligned} \tag{5.1}$$

Among them, gle_f and gle_g denote the real and imaginary parts of the governing equations, respectively. Here, λ_r, λ_0, and λ_b represent the weights for the residual loss MSE_r, initial loss MSE_0, and boundary loss MSE_b, respectively. N_r, N_0, and N_b denote the number of internal sampling points, initial sampling points, and boundary sampling points, respectively.

For complex-valued solutions, the above-mentioned L^2 relative error is implemented by combining the contributions from real and imaginary components. Let the network-predicted approximate solution be $\tilde{u} = [\tilde{f}, \tilde{g}]$ and the corresponding exact solution be $u = [f, g]$. The L^2 relative error for complex-valued cases is given as

$$\mathcal{E} = \frac{\sqrt{\left\|\tilde{f} - f\right\|_2^2 + \left\|\tilde{g} - g\right\|_2^2}}{\sqrt{\left\|f\right\|_2^2 + \left\|g\right\|_2^2}},$$

which is the modulus-based implementation of the aforementioned general L^2 relative error for complex-valued problems.

5.1. 2D time fractional order GLE

We consider the following 2D time fractional order GLE:

$$\begin{cases} {}_0^{\mathrm{C}}D_t^{\alpha}u(x,y,t)-(1+\mathrm{i})\Delta u(x,y,t)+(1+\mathrm{i})|u(x,y,t)|^2u(x,y,t)-u(x,y,t)=h(x,y,t) \\ u(x,y,0)=0, \\ u(0,y,t)=u(1,y,t)=u(x,0,t)=u(x,1,t)=0, \end{cases} \tag{5.2}$$

where $0<\alpha<1$, the domain of the equation is $x\times y\in[0,1]\times[0,1]$ and $t\in[0,1]$, with the exact solution to the equation given by $u(x,y,t)=t^2\left[(x-x^2)^2(y-y^2)^2+\mathrm{i}\sin(\pi x)\sin(\pi y)\right]$. We separate u into its real part f and imaginary part g, and split the corresponding forced term $h(x,y,t)$ into real part $h_f(x,y,t)$ and imaginary part $h_g(x,y,t)$, given by

$$\begin{aligned} h_f=&\frac{\Gamma(3)}{\Gamma(3-\alpha)}t^{2-\alpha}(x-x^2)(y-y^2)+t^2\left[2(x-x^2)+2(y-y^2)-2\pi^2\sin(\pi x)\sin(\pi y)\right] \\ &+t^6\left[((x-x^2)(y-y^2))^2+(\sin(\pi x)\sin(\pi y))^2\right]\left[(x-x^2)(y-y^2)-\sin(\pi x)\sin(\pi y)\right] \\ &-t^2(x-x^2)(y-y^2), \\ h_g=&\frac{\Gamma(3)}{\Gamma(3-\alpha)}t^{2-\alpha}\sin(\pi x)\sin(\pi y)+\left[2(x-x^2)+2(y-y^2)+2\pi^2\sin(\pi x)\sin(\pi y)\right] \\ &+t^6\left[((x-x^2)(y-y^2))^2+(\sin(\pi x)\sin(\pi y))^2\right]\left[(x-x^2)(y-y^2)+\sin(\pi x)\sin(\pi y)\right] \\ &-t^2\sin(\pi x)\sin(\pi y). \end{aligned}$$

The approximate solution $\tilde{u}(x,y,t)$ of Eq (5.2) is constructed to automatically satisfy the initial conditions, which is achieved by setting $\tilde{u}(x,y,t)=tu_\theta(x,y,t)$, where u_θ denotes the network's output. With $\tilde{u}(x,y,t)=\left[\tilde{f}(x,y,t),\tilde{g}(x,y,t)\right]$, the loss function is defined as $\mathrm{MSE}_{GLE}=\lambda_r\mathrm{MSE}_r+\lambda_b\mathrm{MSE}_b$. Considering that the $L2\text{-}1_\sigma$ scheme Eq (4.8) is remarkably different from the $L1$ and $L1$-2 schemes, the corresponding fPINNs loss function under the $L2\text{-}1_\sigma$ scheme is given as follows:

$$\mathrm{MSE}_{GLE}=\lambda_r\sqrt{(\mathrm{MSE}_{gle_f})^2+(\mathrm{MSE}_{gle_g})^2}+\lambda_b\sqrt{(\mathrm{MSE}_{b_f})^2+(\mathrm{MSE}_{b_g})^2},$$

where

$$\begin{aligned} \mathrm{MSE}_{gle_f}=&\frac{1}{N_r}\sum_{j=1}^{N_y}\sum_{i=1}^{N_x}\sum_{k=1}^{N_t}\Big[{}_0^{\mathrm{C}}D_t^{\alpha}\tilde{f}(x_i^r,y_j^r,t_k^r)-\Delta\tilde{f}(x_i^r,y_j^r,t_k^r)+\Delta\tilde{g}(x_i^r,y_j^r,t_k^r) \\ &+\left(\tilde{f}(x_i^r,y_j^r,t_k^r)^2+\tilde{g}(x_i^r,y_j^r,t_k^r)^2\right)\left(\tilde{f}(x_i^r,y_j^r,t_k^r)-\tilde{g}(x_i^r,y_j^r,t_k^r)\right)-\tilde{f}(x_i^r,y_j^r,t_k^r)-h_f(x_i^r,y_j^r,t_k^r)\Big]^2 \\ =&\frac{1}{N_r}\sum_{j=1}^{N_y}\sum_{i=1}^{N_x}\sum_{k=1}^{N_t}\Big[\frac{\Delta t^{\alpha}}{\Gamma(1-\alpha)}\sum_{l=0}^{k}c_l^{(k,\alpha,\sigma)}\left(u(x_i^r,y_j^r,t_{k+1-l}^r)-u(x_i^r,y_j^r,t_{k-l}^r)\right)-\Delta\tilde{f}(x_i^r,y_j^r,t_k^r) \\ &+\Delta\tilde{g}(x_i^r,y_j^r,t_k^r)+\left(\tilde{f}(x_i^r,y_j^r,t_k^r)^2+\tilde{g}(x_i^r,y_j^r,t_k^r)^2\right)\left(\tilde{f}(x_i^r,y_j^r,t_k^r)-\tilde{g}(x_i^r,y_j^r,t_k^r)\right) \\ &-\tilde{f}(x_i^r,y_j^r,t_k^r)-h_f(x_i^r,y_j^r,t_k^r)\Big]^2, \end{aligned}$$

$$\mathrm{MSE}_{b_f} = \frac{1}{N_t \times N_y}\Big[\sum_{j=1}^{N_y}\sum_{k=1}^{N_t}\Big(\tilde{f}(0, y_j^b, t_k^b) - 0\Big)^2 + \sum_{j=1}^{N_b}\sum_{k=1}^{N_b}\Big(\tilde{f}(1, y_j^b, t_k^b) - 0\Big)^2\Big]$$
$$+ \frac{1}{N_t \times N_x}\Big[\sum_{j=1}^{N_y}\sum_{k=1}^{N_t}\Big(\tilde{f}(x_i^b, 0, t_k^b) - 0\Big)^2 + \sum_{j=1}^{N_y}\sum_{k=1}^{N_t}\Big(\tilde{f}(x_i^b, 1, t_k^b) - 0\Big)^2\Big].$$

Δt denotes the time step size of the $L2\text{-}1_\sigma$ difference scheme, and the constant coefficients $c_l^{(k,\alpha,\sigma)}$ are defined in Eq (4.9). Here, we only present the internal loss function MSE_{gle_f} and boundary loss function MSE_{b_f} corresponding to the real part. The construction of MSE_{gle_g} and MSE_{b_g} for the imaginary part follows the same rule.

We employ a uniform sampling strategy, aligning the number of auxiliary points with the time-domain sampling points to yield a time step of $\Delta t = 1/N_t = 0.1$. Unless otherwise specified, the loss weights are set to $\lambda_r = 1$ and $\lambda_b = 10$. Finally, the L_2 relative error is calculated over a uniform $100\times100\times100$ test grid.

In this example, two network initialization schemes are investigated: PyTorch Xavier uniform initialization and He normal initialization. The optimization workflow employs the L-BFGS optimizer. Five independent runs are performed for each initialization scheme. The mean L^2 relative error, standard deviation, and mean computational time over the five repeated runs are summarized in Table 1.

Table 1. Statistics over five independent runs for different initializations: 2D time-fractional GLE.

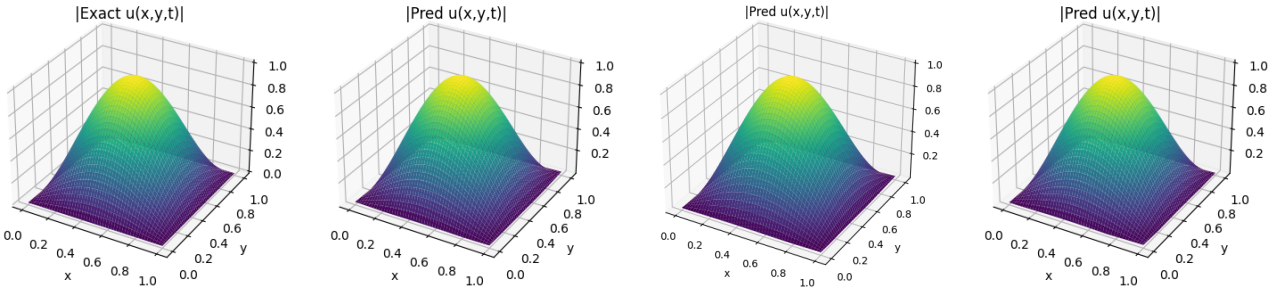
Init.	Mean L^2–err	Std of L^2–err	Mean time (s)
He normal	5.27×10^{-3}	1.46×10^{-3}	724.98
Xavier uniform	7.01×10^{-3}	1.57×10^{-3}	728.54

From Table 1, we observe that both Xavier uniform and He normal initialization can produce valid numerical results for the 2D time-fractional GLE. Compared with Xavier uniform, He normal initialization yields a lower mean L^2 relative error and smaller standard deviation, which indicates more accurate predictions and better stability across repeated runs. The average computational time is comparable for the two initialization strategies. Therefore, He normal is adopted as the default initialization. All subsequent numerical results and analyses presented in this subsection are obtained under the He normal initialization setting. Table 2 summarizes the problem settings and neural network configurations for Eq (5.2).

Figure 2 and Figure 3 present 3D comparisons and heatmaps of the exact and fPINN-modeled solutions across different difference schemes, demonstrating the model's excellent predictive performance. Figure 4 illustrates the error distribution contours between the predicted and exact solutions, confirming the high-fidelity fitting capability of the fPINNs, with the $L2\text{-}1_\sigma$ scheme yielding the highest accuracy. Furthermore, Figure 5 shows the evolution of the loss functions for both the governing equations and boundary conditions, confirming convergence to a stable state after training. These results validate the efficacy and stability of the proposed fPINNs for fractional PDE simulation.

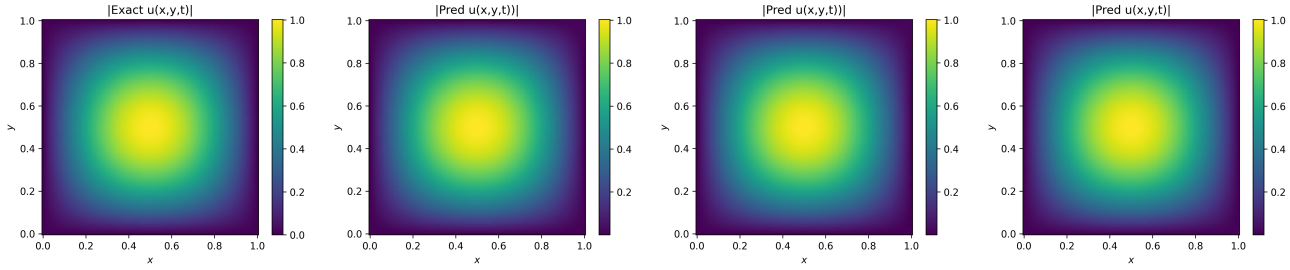
Table 2. Problem settings and neural network configurations for temporal fractional-order GLE.

Parameter	Value
Time Fractional GLEs	$gle_f := {}_0^{\mathrm{C}}D_t^{\alpha}\tilde{f} - \Delta\tilde{f} + \Delta\tilde{g} + (\tilde{f}^2+\tilde{g}^2)(\tilde{f}-\tilde{g}) - \tilde{f} - h_f,$ $gle_g := {}_0^{\mathrm{C}}D_t^{\alpha}\tilde{g} - \Delta\tilde{f} - \Delta\tilde{g} + (\tilde{f}^2+\tilde{g}^2)(\tilde{f}+\tilde{g}) - \tilde{g} - h_g.$
Initial conditions	$\tilde{f}(x,y,0) = \tilde{g}(x,y,0) = 0.$
Boundary conditions	$\tilde{f}(0,y,t) = \tilde{f}(1,y,t) = \tilde{f}(x,0,t) = \tilde{f}(x,1,t) = 0,$ $\tilde{g}(0,y,t) = \tilde{g}(1,y,t) = \tilde{g}(x,0,t) = \tilde{g}(x,1,t) = 0.$
Internal sampling points	$N_r = N_x \times N_y \times N_t = 10 \times 10 \times 10$
Boundary sampling points	20
Layers of net	$[3]+3\times[30]+[2]$
Activation function	Tanh
Optimizer	L-BFGs
Learning rate	0.1
Iteration	10000



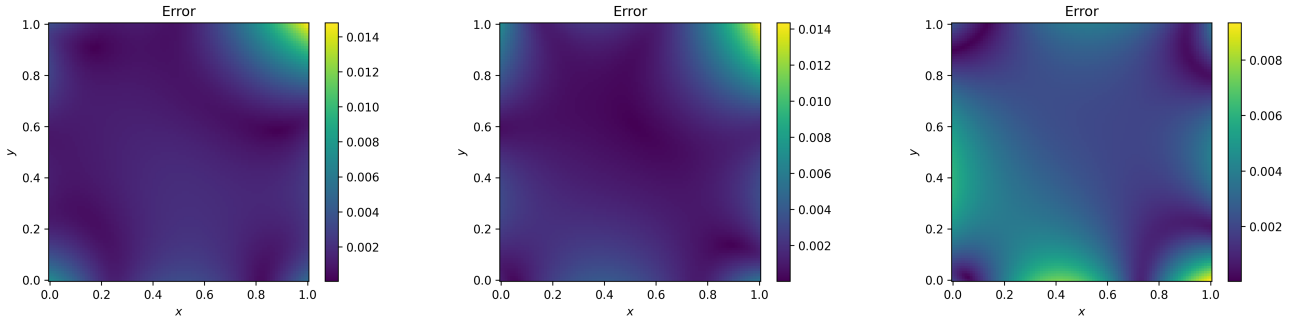
(a) Exact **(b)** $L1$-fPINN **(c)** $L1$-2-fPINN **(d)** $L2$-1_σ-fPINN

Figure 2. Three-dimensional comparison plots of exact and approximate solutions via fPINNs with different difference schemes.



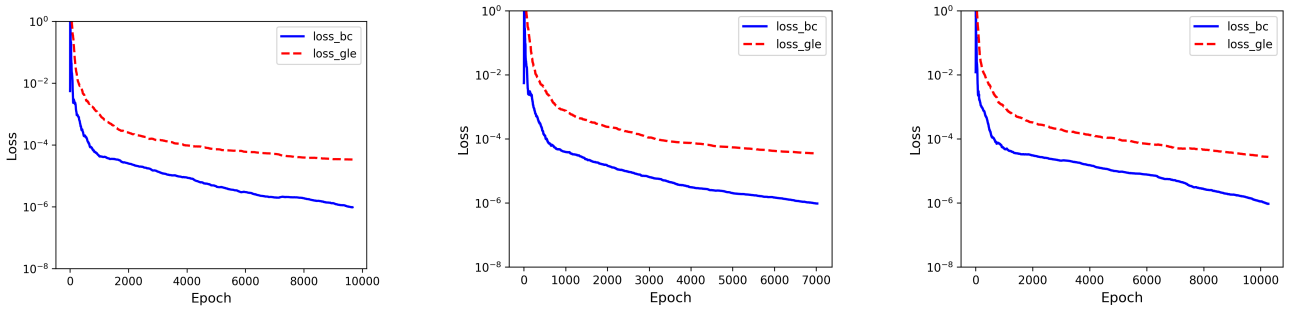
(a) Exact **(b)** $L1$-fPINN **(c)** $L1$-2-fPINN **(d)** $L2$-1_σ-fPINN

Figure 3. Heatmaps of exact and approximate solutions via fPINNs with different difference schemes.



(a) $L1$-fPINN **(b)** $L1$-2-fPINN **(c)** $L2$-1_σ-fPINN

Figure 4. Error heatmaps of exact and approximate solutions via fPINNs with different difference schemes.



(a) $L1$-fPINN **(b)** $L1$-2-fPINN **(c)** $L2$-1_σ-fPINN

Figure 5. Loss function evolution curves for different difference schemes.

The experiment quantitatively evaluated the approximation accuracy of fPINNs across various difference schemes. Using the L^2 relative error as a metric, the $L1$ and $L1$-2 formulas yielded errors of 6.06×10^{-3} and 5.60×10^{-3}, respectively, while the $L2$-1_σ formula achieved a superior error of 3.93×10^{-3}. These results demonstrate that while all three schemes provide satisfactory performance, the $L2$-1_σ scheme offers the most significant error reduction, confirming its superior predictive accuracy. Furthermore, sensitivity analysis, varying the fractional order α while maintaining the parameters in Table 1, shows that the $L2$-1_σ model maintains consistent stability, with errors ranging strictly between 3×10^{-3} and 6×10^{-3}.

To further verify the performance of our optimal $L2$-1_σ-based fPINNs, we conduct comparative tests with the L^2-error results obtained by the finite element method (FEM) for Example 1 in [20], which solves the two-dimensional time-fractional GLE. The L^2 relative error of the conventional FEM and our proposed approach are summarized in Table 3.

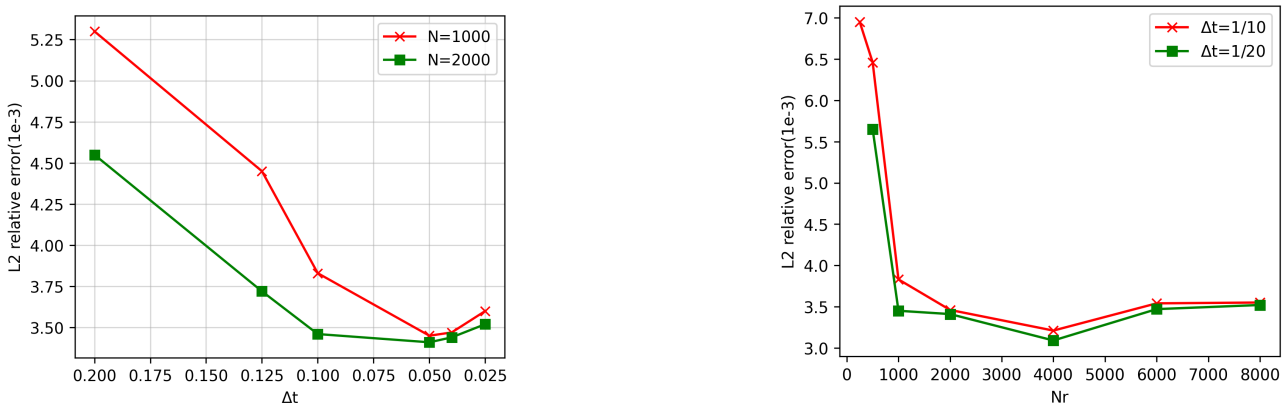
It can be observed that our fPINNs achieve competitive accuracy under comparable parameter settings. Note that there exists a slight difference in the form of nonlinear terms between our numerical setup and Ref. [20], while the fractional-order parameters and boundary conditions remain consistent. Therefore, this comparison serves as semi-quantitative cross-validation rather than a rigorous benchmark test under completely identical problem configurations.

Table 3. Comparison of L^2 relative errors for 2D time-fractional GLE.

	$\alpha = 0.50(N_t = 7, 11)$		$\alpha = 0.75(N_t = 10, 28)$	
$N_x \times N_y$	Chen et al. [20]	$L2\text{-}1_\sigma$-fPINNs	Chen et al. [20]	$L2\text{-}1_\sigma$-fPINNs
4×4	8.72×10^{-2}	5.72×10^{-3}	9.45×10^{-2}	5.22×10^{-3}
8×8	2.48×10^{-2}	4.90×10^{-3}	2.62×10^{-2}	2.86×10^{-3}

Analysis of the fPINN-based approach for solving fractional GLEs reveals that approximation errors stem from four sources: discretization, sampling, network structure, and optimization. Dissecting these components elucidates how neural networks influence numerical solutions in fractional differential equations.

First, we examine how discretization errors-arising from the $L2\text{-}1_\sigma$ scheme applied to the temporal fractional operator impact the numerical results. By analyzing the construction of this scheme Eq (4.8), we quantitatively investigate these errors by adjusting the time step Δt. The neural network architecture, learning rate, iteration count, and boundary sampling points remained consistent with Table 2. For each Δt, we evaluated the model on a $100 \times 100 \times 100$ uniform grid. Figure 6a displays the L^2 relative error versus time steps for boundary point counts of 1000 and 2000. While the results generally show decreasing error as Δt shrinks, the error begins to rise when Δt is reduced to 0.05. This behavior is likely due to optimization limitations involving the learning rate and loss weight coefficients. Reducing Δt introduces more temporal sampling points, which increases the complexity of the loss function. The neural network tends to converge to local minima instead of the desired optimum, leading to deteriorated accuracy.



(a) **(b)**

Figure 6. (a) L^2 Relative error vs. time step length plot. (b) L^2 relative error vs. number of training points plot.

Sampling error arises from fluctuations in the number of training points, specifically influenced by both interior and boundary point densities. Given that boundary points are fewer than interior ones, this study focuses on the relationship between interior sampling density and numerical accuracy. To isolate this, we fix the neural network architecture, learning rate, and time step, analyzing six cases of interior points ($N_r = 500, 1000, 2000, 4000, 6000, 8000$) using $100 \times 100 \times 100$ test points, with results detailed

in Table 2 and Figure 6b. As shown in Figure 6b, the L^2 relative error drops significantly as N_r increases under both time steps ($\Delta t = 0.10, 0.05$). The error reaches a minimum at $N_r = 4000$ (3.21×10^{-3} and 3.09×10^{-3}), after which it increases, indicating that while increasing training points reduces sampling error, an excessive number renders the model susceptible to higher optimization error.

Neural network approximation error is significantly influenced by network depth (number of layers) and width (neurons per layer). To isolate the impact of architecture on numerical computation results, we kept the training points, time step, and learning rate constant, as detailed in Table 2. Figure 7 illustrates the L^2 relative error across a $100 \times 100 \times 100$ test set for various configurations. Results indicate that increasing network complexity does not yield a monotonic decrease in error. While expansion reduces error for smaller models, over-parameterization eventually leads to higher L^2 errors, driven by optimization difficulties rather than approximation limitations. Consequently, a depth of 4 and width of 40 was selected, yielding a minimum L^2 relative error of 2.40×10^{-3}.

As model complexity increases, optimization errors become the primary determinant of experimental results. Our analysis indicates that when dealing with excessive depth/width, large datasets, or extreme time-step discretization, optimization errors overshadow sampling and approximation errors.

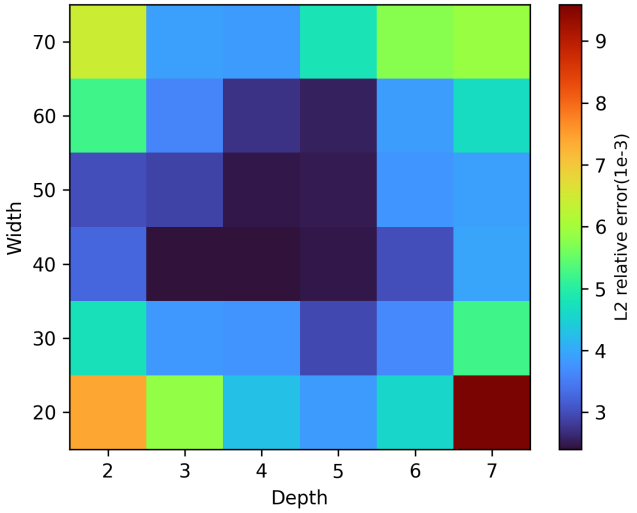


Figure 7. Heatmap of L^2 relative error as a function of neural network depth and width.

5.2. 1D space fractional order GLE

In this section, we consider the following 1D spatial fractional−order GLE with homogeneous Dirichlet boundary conditions

$$\begin{cases} \partial_t u(x,t) + (-\Delta)^{\frac{\beta}{2}} u(x,t) + (1+\mathrm{i})|u(x,t)|^2 u(x,t) - u(x,t) = h(x,t), & x \times t \in [0,1] \times [0,1], \\ u(x,0) = 0, & x \in [0,1], \\ u(0,t) = u(1,t) = 0, & t \in (0,1]. \end{cases} \tag{5.3}$$

Here, $1 < \beta < 2$, the exact solution is given by $u(x,t) = x(1-x)\,[t(1-t) + \mathrm{i}\sin(\pi t)]$, and the corresponding forced term $h(x,t)$ is split into real part $h_f(x,t)$ and imaginary part $h_g(x,t)$, which correspond to the following equations:

$$
\begin{aligned}
h_f = & \frac{1}{2\cos(\frac{\beta\pi}{2})}\left[\frac{\Gamma(2)}{\Gamma(2-\beta)}(x^{1-\beta}+(1-x)^{1-\beta})-\frac{\Gamma(3)}{\Gamma(3-\beta)}(x^{2-\beta}+(1-x)^{2-\beta})\right]t(1-t) \\
& +x(1-x)(1-2t)+(x(1-x))^3\left[(t(1-t))^2+(\sin(\pi t))^2\right](t(1-t)-\sin(\pi t)) \\
& -x(1-x)t(1-t), \\
h_g = & \frac{1}{2\cos(\frac{\beta\pi}{2})}\left[\frac{\Gamma(2)}{\Gamma(2-\beta)}(x^{1-\beta}+(1-x)^{1-\beta})-\frac{\Gamma(3)}{\Gamma(3-\beta)}(x^{2-\beta}+(1-x)^{2-\beta})\right]\sin(\pi t) \\
& +x(1-x)\pi\cos(\pi t)+(x(1-x))^3\left[(t(1-t))^2+(\sin(\pi t))^2\right](t(1-t)+\sin(\pi t)) \\
& -x(1-x)\sin(\pi t).
\end{aligned}
$$

Separating $u(x,t)$ into its real part $f(x,t)$ and imaginary part $g(x,t)$, we construct the approximate solution $\tilde{u}(x,t)$ to automatically satisfy the initial and boundary conditions, specifically, by setting it as $\tilde{u}(x,t) = tx(1-x)u_\theta(x,t)$, where u_θ is the network's output. We further denote $\tilde{u}(x,t) = \left[\tilde{f}(x,t), \tilde{g}(x,,t)\right]$. The loss function is constructed to incorporate only the residual term of the equation, i.e., $\mathrm{MSE}_{GLE} = \mathrm{MSE}_r$, thus eliminating the necessity of balancing multiple terms via loss weights (e.g., λ_r).

The test set is built with equidistant sampling, where spatial auxiliary points are equal to spatial sampling points, yielding a spatial stride of $\Delta x = 1/N_x = 0.1$. It consists of a 100×100 uniform grid.

Consistent with the aforementioned 2D time-fractional experiment, this 1D spatial-fractional case is also tested under two network initialization schemes (Xavier uniform and He normal) with five independent repeated runs. The statistical results are listed in Table 4. Similar to the 2D case, He normal initialization provides higher accuracy and smaller error fluctuation with comparable computational cost. Accordingly, all numerical results, figures, and analyses in this subsection are obtained under the He normal initialization strategy. Table 5 details the problem configurations and neural network hyperparameters for Eq (5.3).

Figure 8 presents 3D visualizations and heatmaps comparing the exact and fPINN solutions for the spatial fractional-order equation ($\beta = 1.8$, $\Delta x = 0.1$), demonstrating the fPINN model's ability to accurately capture complex spatial patterns and its robust numerical performance.

The loss function curve in Figure 9 demonstrates that the network stabilizes within 1750 iterations, converging to a low value of 5×10^{-3} and confirming the efficacy of the proposed method. As shown in Figure 10, cross-sectional comparisons reveal that the fPINNs approximation closely matches the exact solution at three distinct time points.

Table 4. Statistics over five independent runs for different initializations: 1D spatial-fractional GLE.

Init.	Mean L^2–err	Std of L^2–err	Mean time (s)
He normal	1.72×10^{-3}	4.35×10^{-5}	8.56
Xavier uniform	1.75×10^{-3}	6.54×10^{-5}	9.17

Table 5. Problem settings and neural network configurations for the spatial fractional-order GLE.

Parameter	Value
Space Fractional GLEs	$gle_r = \partial_t \tilde{f} + (-\Delta)^{\frac{\beta}{2}} \tilde{f} + (\tilde{f}^2 + \tilde{g}^2)(\tilde{f} - \tilde{g}) - \tilde{f} - h_f,$ $gle_i = \partial_t \tilde{g} + (-\Delta)^{\frac{\beta}{2}} \tilde{g} + (\tilde{f}^2 + \tilde{g}^2)(\tilde{f} + \tilde{g}) - \tilde{g} - h_g.$
Initial conditions	$\tilde{f}(x,0) = \tilde{g}(x,0) \quad = 0.$
Boundary conditions	$\tilde{f}(0,t) = \tilde{f}(1,t) \quad = 0,$ $\tilde{g}(0,t) = \tilde{g}(1,t) \quad = 0.$
Internal sampling points	$N_r = N_x \times N_t = 10 \times 10$
Layers of net	$[2] + 3 \times [30] + [2]$
Activation function	Tanh
Optimizer	L-BFGs
Learning rate	1
Iteration	1750

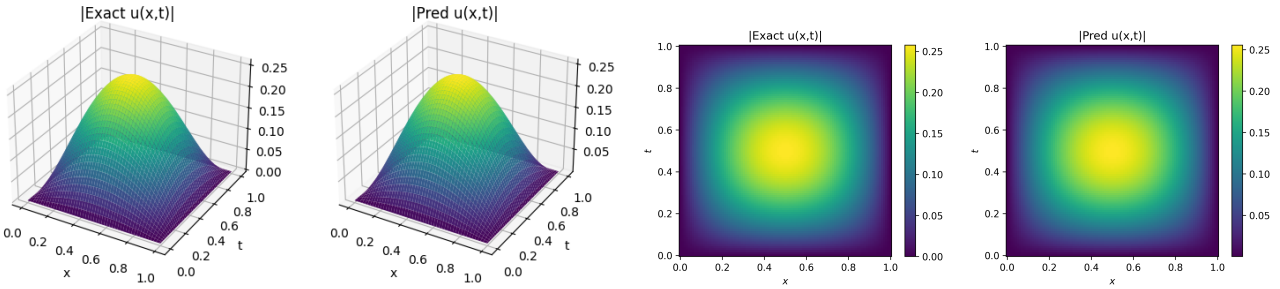


Figure 8. Graphs (a) and (b) show comparison of 3D images for the exact solution and approximate solution of fPINNs, while graphs (c) and (d) provide comparison of heatmaps for the exact solution and approximate solution of fPINNs.

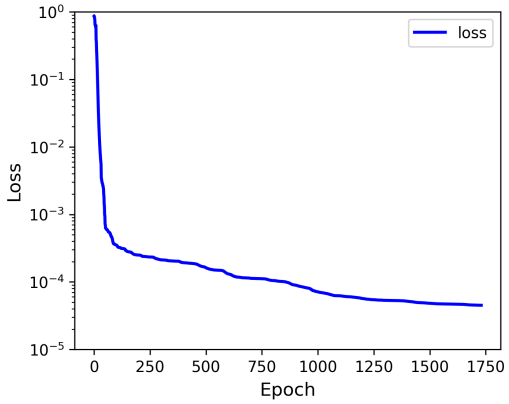
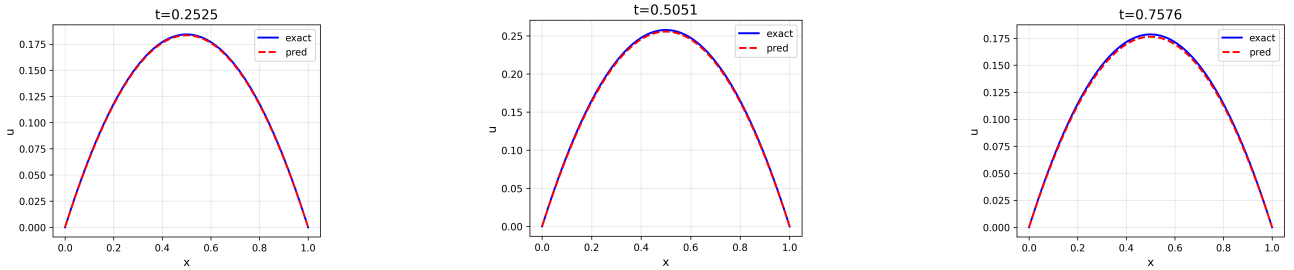


Figure 9. Loss function variation plot.



(a) (b) (c)

Figure 10. Cross-sectional views at different time points.

A numerical evaluation of the model is performed, and for the fractional order $\beta = 1.8$, the L^2 relative error of the results obtained by fPINNs is 1.87×10^{-3}. By varying the spatial fractional order parameter, the L^2 relative errors under the parameter settings of $\beta = 1.2$ and $\beta = 1.5$ are further calculated, with the corresponding results being 1.41×10^{-2} and 7.58×10^{-3}, respectively. These results fully validate that the fPINNs method based on the shifted G-L formula possesses high computational accuracy and excellent fitting performance in solving the GLEs with spatial Laplace terms.

To further verify the performance of the proposed fPINNs for spatial fractional GLEs, we perform comparative analysis using numerical data of Example 5.2.1 from [44] for 1D spatial fractional GLE. Only the fractional-order value and terminal simulation time are kept consistent to mitigate comparison bias. There exist structural discrepancies between the reference model and our governing equation, including different nonlinear-term forms, equation coefficients, and computational interval boundaries. The L^2 relative errors of the traditional numerical method and the proposed fPINNs are summarized in Table 6. It can be observed that our fPINNs achieve competitive accuracy under comparable physical backgrounds. Given the inconsistencies in nonlinear terms, equation coefficients, and interval boundaries, this comparison serves as semi-quantitative cross-validation to examine the feasibility of our solver, rather than a rigorous benchmark test under fully identical problem configurations.

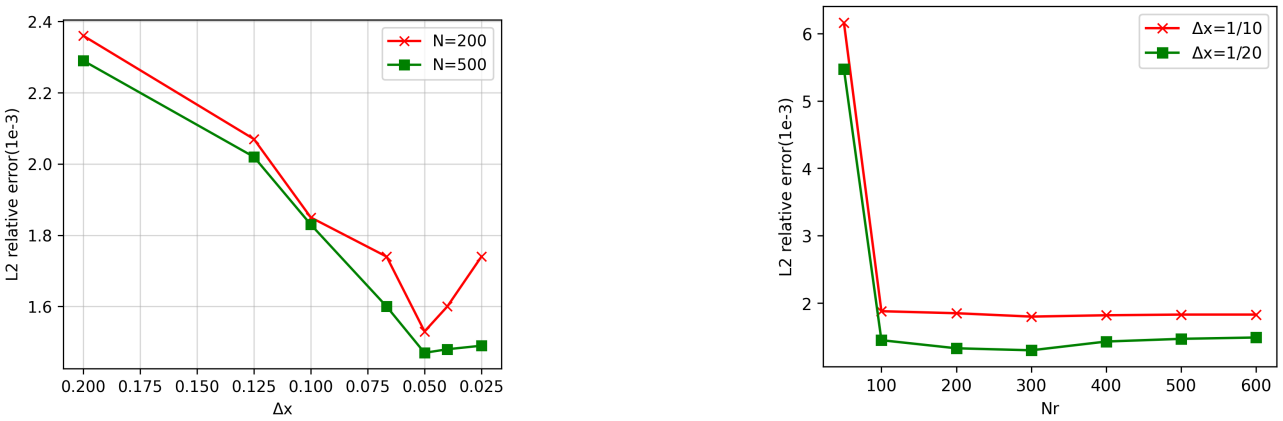
Table 6. Comparison of L^2 relative errors for 1D space-fractional GLE ($N_t = 50, N_x = 5$).

β	Wang and Huang [44]	fPINNs
1.6	3.31×10^{-2}	2.34×10^{-2}
1.9	2.12×10^{-2}	2.39×10^{-3}
2.0	1.74×10^{-2}	0.74×10^{-3}

The error sources of spatial-fractional GLEs are identical to those of time-fractional GLEs.

This study first evaluates the discretization error introduced by the Grünwald-Letnikov (G-L) formula for the fractional Laplace operator. Based on Eq (4.14), we analyze the impact of discretization on numerical accuracy by varying the spatial step size, Δx, using 10000 uniformly sampled test points. Figure 11a illustrates the L^2 relative error trends for $N_r = \{200, 500\}$. While the L^2 error initially decreases with smaller step sizes, it begins to rise when Δx reaches 0.05. This phenomenon is attributed to the increased complexity of the loss function, where optimization error eventually supersedes discretization error as the primary error source.

To analyze the impact of sampling errors on the approximate solutions of spatial fractional-order generalized Langevin equations (GLEs), we varied the number of domain training points, $N_r \in \{50, 100, 200, 300, 400, 500, 600\}$, while maintaining the parameters listed in Table 5. Figure 11b presents the test performance for step sizes $\Delta x = 0.10$ and $\Delta x = 0.05$. For both cases, the L^2 relative error decreases significantly, reaching minima of 1.80×10^{-3} and 1.30×10^{-3} at $N_r = 300$, before increasing with further sampling. This suggests that while increasing N_r initially mitigates sampling error, an excessive number of points leads to optimization challenges that dominate the total error.



(a) **(b)**

Figure 11. (a) L^2 relative error vs. time step length plot. (b) L^2 relative error vs. number of training points plot.

Finally, we investigated network approximation error by adjusting the depth and width of the neural network. Figure 12 plots the L^2 relative errors across the test points as a function of the architecture. The analysis identifies the optimal configuration as a hidden layer network with four layers and 50 neurons per layer, which yields a minimal L^2 relative error of 1.66×10^{-3}.

Optimization errors frequently occur in scenarios involving complex neural network structures and high computational costs, such as redundant network architectures, excessive training data points, or unduly small spatial step sizes.

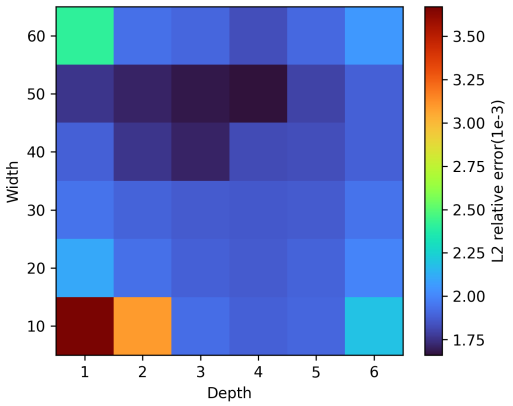


Figure 12. Heatmap of L2 relative error variation with neural network depth and width.

It is worth noting that fPINNs do not intrinsically require equally-spaced or structured grids. Random or non-uniform spatial collocation points are admissible without finite-difference stencil constraints, and temporal auxiliary points can also be non-uniform. From the above one-dimensional spatial-fractional and two-dimensional time-fractional numerical tests, traditional FDMs and FEMs can achieve high accuracy, while fPINNs obtains more precise numerical solutions. In addition, fPINNs yield a continuous global solution field, which allows direct evaluation at arbitrary interior positions without additional interpolation procedures, and natively supports non-uniform sampling strategies. In contrast, FDMs strictly rely on structured grids and only output discrete nodal values. Such inherent advantages make fPINNs particularly prominent when solving problems with complex computational geometries.

6. Conclusions

This study advances the numerical modeling of fractional GLEs using fPINNs. By addressing the computational challenges of fractional derivatives through diversified discretization methods, our approach yields a highly accurate and efficient solution framework for both time- and space-fractional problems.

We implement fPINNs for time-fractional GLEs by constructing a residual-based MSE loss, incorporating initial/boundary conditions, and evaluating $L1$, $L1$-2, and $L2$-1_σ finite difference schemes. The network parameters are optimized using L-BFGs to minimize this loss, with automatic differentiation employed for gradient calculation. Comparative analysis identifies the $L2$-1_σ scheme as providing the highest accuracy. For spatial fractional GLEs, we use the shifted G-L formula to discretize the fractional Laplacian. Final model performance is shown to be a combined function of discretization, sampling, network architecture, and optimization errors.

fPINNs offer a data-driven approach for solving fractional-order equations, leveraging known data to train networks and providing reliable approximations when analytical solutions are unavailable. By optimizing the loss function across the entire domain, these models achieve high-precision solutions that satisfy both governing equations and boundary conditions efficiently. Furthermore, fPINNs are highly adaptable, capable of handling diverse fractional-order problems by adjusting the network architecture and loss formulation.

The primary strength of fPINNs lies in their capacity to handle fractional derivatives directly, making them uniquely suited for solving fPDEs. Applying fPINNs to temporal fractional GLEs allows for a more precise description of complex systems characterized by memory effects and non-locality, offering solutions that are simultaneously efficient and reliable. Consequently, fPINNs hold immense potential across disciplines such as nonlinear optics, biophysics, and quantitative finance.

Despite these advantages, fPINNs face inherent challenges when addressing high-complexity problems. Specifically, achieving high-order accuracy remains difficult, and the models exhibit significant sensitivity to hyperparameter tuning. Furthermore, the computational overhead associated with fPINNs is often substantial. Future advancements, such as refined numerical discretization, adaptive training strategies, and multi-layer modeling, are essential to overcoming these bottlenecks. We also explore integrating fast algorithms for symmetric Toeplitz matrices to accelerate the computation of spatial fractional derivatives [45, 46]. Besides, fast evaluation formulas for temporal fractional derivatives [47, 48] will be considered to mitigate the full-history-dependent computational

overhead in future studies. Additionally, we will conduct accuracy comparisons with the finite difference method to verify the reliability of the framework. We will optimize the network structure and algorithm efficiency to enhance numerical accuracy and calculation speed, and further extend the method to high-dimensional problems to explore the strengths of this mesh-free neural network method. In addition, we will conduct systematic numerical experiments under diverse initial condition settings to further verify the stability and applicability of the three fractional discretization schemes. We anticipate that these enhancements will extend the utility of fPINNs to higherspatio-temporal dimensional and spatio-temporal fractional GLEs, further broadening their analytical scope.

Author contributions

Hong Lu: Conceptualization, Funding acquisition, Project administration, Methodology, Supervision, Writing–review & editing. **Yujie Zhai:** Formal analysis,Investigation, Methodology, Software, Writing–original draft. **Mingji Zhang:** Methodology, Project administration, Supervision, Validation, Writing–review & editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors used Doubao (ByteDance) solely to assist with language refinement and structural editing throughout the manuscript. All scientific content, methodology, data, numerical results, figures, tables, and conclusions were independently reviewed and verified by the authors, who take full responsibility for the final content.

Acknowledgments

The authors thank the referees very much for their careful reviews and comments that helped improve the manuscript. This work is partially supported by the NSF of Shandong Province (No. ZR 2021MA055).

Conflict of interest

Prof. Mingji Zhang is a guest editor for AIMS Mathematics and was not involved in the editorial review and/or the decision to publish this article.

The authors declare that there is no conflict of interest.

References

1. V. L. Ginzburg, L. D. Landau, *On the theory of superconductivity*, Berlin, Heidelberg: Springer, 2009, 113–137. https://doi.org/10.1007/978-3-540-68008-6_4
2. R. Du, Y. Wang, Z. Hao, High-dimensional nonlinear Ginzburg-Landau equation with fractional Laplacian: discretization and simulations, *Commun. Nonlinear Sci. Numer. Simul.*, **102** (2021), 105920. http://doi.org/10.1016/j.cnsns.2021.105920

3. K. Otsuka, Self-induced phase turbulence, spot dancing, and chaotic itinerancy in evanescent-field coupled waveguide lasers, *Nonlinear Dynamics in Optical Systems*, 1990. http://doi.org/10.1364/NLDOS.1990.SDSLAD101

4. A. Tozar, New analytical solutions of fractional complex Ginzburg-Landau equation, *Univ. J. Math. Appl.*, **3** (2020), 129–132. http://doi.org/10.32323/ujma.760899

5. Y. Rameshwar, V. Anuradha, G. Srinivas, L. M. Perez, D. Laroze, H. Pleiner, Nonlinear convection of binary liquids in a porous medium, *Chaos*, **28** (2018), 075512. http://doi.org/10.1063/1.5027468

6. K. A. Bekmaganbetov, G. A. Chechkin, A. A. Tolemis, Attractors of Ginzburg-Landau equations with oscillating terms in porous media: homogenization procedure, *Appl. Anal.*, **103** (2024), 29–44. http://doi.org/10.1080/00036811.2023.2173182

7. H. Mohebalizadeh, H. Adibi, M. Dehghan, Well-posedness of space fractional Ginzburg-Landau equations involving the fractional Laplacian arising in a Bose-Einstein condensation and its kernel based approximation, *Commun. Nonlinear Sci. Numer. Simul.*, **126** (2023), 107469. http://doi.org/10.1016/j.cnsns.2023.107469

8. T. Horikiri, M. Yamaguchi, K. Kamide, Y. Matsuo, T. Byrnes, N. Ishida, et al., High-energy side-peak emission of exciton-polariton condensates in high density regime, *Sci. Rep.*, **6** (2016), 25655. http://doi.org/10.1038/srep25655

9. W. Liu, W. Yu, C. Yang, M. Liu, Y. Zhang, M. Lei, Analytic solutions for the generalized complex Ginzburg-Landau equation in fiber lasers, *Nonlinear Dyn.*, **89** (2017), 2933–2939. https://doi.org/10.1007/s11071-017-3636-5

10. Z. Yan, Y. Yan, M. Liu, W. Liu, Propagation properties of bright solitons generated by the complex Ginzburg-Landau equation with high-order dispersion and nonlinear gradient terms, *Appl. Math. Lett.*, **157** (2024), 109164. http://doi.org/10.1016/j.aml.2024.109164

11. X. Li, S. Li, A linearized element-free Galerkin method for the complex Ginzburg-Landau equation, *Comput. Math. Appl.*, **90** (2021), 135–147. http://doi.org/10.1016/j.camwa.2021.03.027

12. Y. Shen, J. Zweck, S. Wang, C. R. Menyuk, Spectra of short pulse solutions of the cubic-quintic complex Ginzburg-Landau equation near zero dispersion, *Stud. Appl. Math.*, **137** (2016), 238–255. http://doi.org/10.1111/sapm.12136

13. M. Caliari, F. Cassini, Efficient simulation of complex Ginzburg-Landau equations using high-order exponential-type methods, *Appl. Numer. Math.*, **206** (2024), 340–357. http://doi.org/10.1016/j.apnum.2024.08.009

14. S. Chen, B. Guo, Classical solutions of general Ginzburg-Landau equations, *Acta Math. Sci.*, **36** (2016), 717–732. http://doi.org/10.1016/S0252-9602(16)30034-0

15. A. V. Milovanov, J. J. Rasmussen, Fractional generalization of the Ginzburg-Landau equation: an unconventional approach to critical phenomena in complex media, *Phys. Lett. A*, **337** (2005), 75–80. http://doi.org/10.1016/j.physleta.2005.01.047

16. V. E. Tarasov, G. M. Zaslavsky, Fractional Ginzburg-Landau equation for fractal media, *Phys. A*, **354** (2005), 249–261. http://doi.org/10.1016/j.physa.2005.02.047

17. D. He, K. Pan, An unconditionally stable linearized difference scheme for the fractional Ginzburg-Landau equation, *Numer. Algorithms*, **79** (2018), 899–925. http://doi.org/10.1007/s11075-017-0466-y

18. P. Wang, Fast exponential time differencing/spectral-Galerkin method for the nonlinear fractional Ginzburg-Landau equation with fractional Laplacian in unbounded domain, *Appl. Math. Lett.*, **112** (2021), 106710. http://doi.org/10.1016/j.aml.2020.106710

19. N. Wang, C. Huang, An efficient split-step quasi-compact finite difference method for the nonlinear fractional Ginzburg-Landau equations, *Comput. Math. Appl.*, **75** (2018), 2223–2242. http://doi.org/10.1016/j.camwa.2017.12.005

20. F. Chen, M. Li, Y. Zhao, Y. Tang, Convergence and superconvergence analysis of finite element methods for nonlinear Ginzburg-Landau equation with Caputo derivative, *Comput. Appl. Math.*, **42** (2023), 271. http://doi.org/10.1007/s40314-023-02409-4

21. M. A. Zaky, A. S. Hendy, J. E. Macías-Díaz, High-order finite difference/spectral-Galerkin approximations for the nonlinear time-space fractional Ginzburg-Landau equation, *Numer. Methods Partial Differ. Equ.*, **39** (2023), 4549–4574. http://doi.org/10.1002/num.22630

22. Y. Zhao, A. Ostermann, X. Gu, A low-rank Lie-Trotter splitting approach for nonlinear fractional complex Ginzburg-Landau equations, *J. Comput. Phys.*, **446** (2021), 110652. http://doi.org/10.1016/j.jcp.2021.110652

23. X. M. Gu, L. Shi, T. Liu, Well-posedness of the fractional Ginzburg-Landau equation, *Appl. Anal.*, **98** (2019), 2545–2558.http://doi.org/10.1080/00036811.2018.1466281

24. K. Pan, X. Jin, D. He, Pointwise error estimates of a linearized difference scheme for strongly coupled fractional Ginzburg-Landau equations, *Math. Methods Appl. Sci.*, **43** (2020), 512–535. http://doi.org/10.1002/mma.5897

25. M. Zhang, G. F. Zhang, L. D. Liao, Fast iterative solvers and simulation for the space fractional Ginzburg-Landau equations, *Comput. Math. Appl.*, **78** (2019), 1793–1800. http://doi.org/10.1016/j.camwa.2019.01.026

26. M. H. Heydari, M. Hosseininia, A. Atangana, Z. Avazzadeh, A meshless approach for solving nonlinear variable-order time fractional 2D Ginzburg-Landau equation, *Eng. Anal. Bound. Elem.*, **120** (2020), 166–179. http://doi.org/10.1016/j.enganabound.2020.08.015

27. M. H. Heydari, M. Razzaghi, Piecewise fractional Chebyshev cardinal functions: application for time fractional Ginzburg-Landau equation with a non-smooth solution, *Chaos Soliton. Fract.*, **171** (2023), 113445. http://doi.org/10.1016/j.chaos.2023.113445

28. G. E. Hinton, S. Osindero, Y. W. Teh, A fast learning algorithm for deep belief nets, *Neural Comput.*, **18** (2006), 1527–1554. http://doi.org/10.1162/neco.2006.18.7.1527

29. M. Li, C. Huang, N. Wang, Galerkin finite element method for the nonlinear fractional Ginzburg-Landau equation, *Appl. Numer. Math.*, **118** (2017), 131–149. http://doi.org/10.1016/j.apnum.2017.03.003

30. J. Zhou, X. Yao, W. Wang, Two-grid finite element methods for nonlinear time-fractional parabolic equations, *Numer. Algorithms*, **90** (2022), 709–730. http://doi.org/10.1007/s11075-021-01205-7

31. S. Santra, J. Mohapatra, A novel finite difference technique with error estimate for time fractional partial integro-differential equation of Volterra type, *J. Comput. Appl. Math.*, **400** (2022), 113746. http://doi.org/10.1016/j.cam.2021.113746
32. M. Fei, C. Huang, N. Wang, G. Zhang, Galerkin-Legendre spectral method for the nonlinear Ginzburg-Landau equation with the Riesz fractional derivative, *Math. Methods Appl. Sci.*, **44** (2021), 2711–2730. http://doi.org/10.1002/mma.5852
33. S. Kumar, Numerical solution of fuzzy fractional diffusion equation by Chebyshev spectral method, *Numer. Methods Partial Differ. Equ.*, **38** (2022), 490–508. http://doi.org/10.1002/num.22650
34. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, **378** (2019), 686–707. http://doi.org/10.1016/j.jcp.2018.10.045
35. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (Part I): data-driven solutions of nonlinear partial differential equations, *arXiv Preprint*, 2017. http://doi.org/10.48550/arXiv.1711.10561
36. Z. Ma, J. Hou, W. Zhu, Y. Peng, Y. Li, PMNN: Physical model-driven neural network for solving time-fractional differential equations, *Chaos Soliton. Fract.*, **177** (2023), 114238. http://doi.org/10.1016/j.chaos.2023.114238
37. G. Pang, L. Lu, G. E. Karniadakis, fPINNs: Fractional physics-informed neural networks, *SIAM J. Sci. Comput.*, **41** (2019), A2603–A2626. http://doi.org/10.1137/18M1229845
38. E. Kharazmi, Z. Zhang, G. E. Karniadakis, Variational physics-informed neural networks for solving partial differential equations, *arXiv Preprint*, 2019. http://doi.org/10.48550/arXiv.1912.00873
39. D. Zhang, L. Lu, L. Guo, G. E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, *J. Comput. Phys.*, **397** (2019), 108850. http://doi.org/10.1016/j.jcp.2019.07.048
40. A. Lischke, G. Pang, M. Gulian, F. Song, C. Glusa, X. Zheng, et al., What is the fractional Laplacian? A comparative review with new results, *J. Comput. Phys.*, **404** (2020), 109009. http://doi.org/10.1016/j.jcp.2019.109009
41. Z. Z. Sun, X. Wu, A fully discrete difference scheme for a diffusion-wave system, *Appl. Numer. Math.*, **56** (2006), 193–209. http://doi.org/10.1016/j.apnum.2005.03.003
42. G. H. Gao, Z. Z. Sun, H. W. Zhang, A new fractional numerical differentiation formula to approximate the Caputo fractional derivative and its applications, *J. Comput. Phys.*, **259** (2014), 33–50. http://doi.org/10.1016/j.jcp.2013.11.017
43. A. A. Alikhanov, A new difference scheme for the time fractional diffusion equation, *J. Comput. Phys.*, **280** (2015), 424–438. http://doi.org/10.1016/j.jcp.2014.09.031
44. P. Wang, C. Huang, An implicit midpoint difference scheme for the fractional Ginzburg-Landau equation, *J. Comput. Phys.*, **312** (2016), 31–49. http://doi.org/10.1016/j.jcp.2016.02.018
45. H. Y. Jian, T. Z. Huang, X. M. Gu, Y. L. Zhao, Fast compact implicit integration factor method with non-uniform meshes for the two-dimensional nonlinear Riesz space-fractional reaction-diffusion equation, *Appl. Numer. Math.*, **156** (2020), 346–363. http://doi.org/10.1016/j.apnum.2020.05.005

46. H. Y. Jian, T. Z. Huang, X. M. Gu, X. L. Zhao, Y. L. Zhao, Fast implicit integration factor method for nonlinear space Riesz fractional reaction-diffusion equations, *J. Comput. Appl. Math.*, **378** (2020), 112935. http://doi.org/10.1016/j.cam.2020.112935

47. L. Qing, X. Li, Analysis of a meshless generalized finite difference method for the time-fractional diffusion-wave equation, *Comput. Math. Appl.*, **172** (2024), 134–151. http://doi.org/10.1016/j.camwa.2024.08.008

48. J. Shen, C. Li, Z. Z. Sun, An H2N2 interpolation for Caputo derivative with order in (1, 2) and its application to time-fractional wave equations in more than one space dimension, *J. Sci. Comput.*, **83** (2020), 38. http://doi.org/10.1007/s10915-020-01219-8

AIMS Press