Bergische Universität Wuppertal

Fakultät für Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM)

Preprint BUW-IMACM 26/06

Himanshu Kumar Dwivedi, Matthias Ehrhardt and Rajeev

Alikhanov–XfPINNs: Adaptive Physics-Informed Learning for Nonlinear Fractional PDEs on Nonuniform Meshes

May 1, 2026

http://www.imacm.uni-wuppertal.de

ALIKHANOV–XFPINNS: ADAPTIVE PHYSICS-INFORMED LEARNING FOR NONLINEAR FRACTIONAL PDES ON NONUNIFORM MESHES

HIMANSHU KUMAR DWIVEDI*, MATTHIAS EHRHARDT†, AND RAJEEV*

Abstract. To address the initial singularity inherent in solutions to fractional partial differential equations (fPDEs), we propose an accelerated Alikhanov discretization formulation implemented on nonuniform time grids. Based on the *physics-informed neural networks (PINNs)* framework, we introduce an Alikhanov-extended fractional PINNs (XfPINNs) architecture that combines high-order temporal discretization and deep learning. The nonlocal memory term in fPDEs leads to high computational cost, while the weak singularity near $t \to 0^+$ can deteriorate accuracy on uniform meshes. To separate temporal discretization effects from optimization and sampling errors, we further develop an auxiliary time-marching configuration that enables auditable temporal-convergence studies under controlled training tolerances. This architecture can solve general nonlinear fPDEs. The XfPINNs approach is designed for forward and inverse problems, allowing for data-driven solution reconstruction and parameter estimation. First, the neural network approximates the solution of nonlinear fPDEs; then, an adaptive activation function accelerates convergence and enhances training efficiency. The optimization framework embeds a variational loss function constructed from the Alikhanov scheme, where the initial and boundary conditions are imposed using a combination of hard and soft constraints. Numerical experiments, including cases with known and unknown exact solutions which demonstrate the robustness, computational efficiency, and significant CPU time savings of the Alikhanov–XfPINNs method.

Key words. Nonuniform Alikhanov formula, deep learning, physics-informed neural networks, data-driven scientific computing.

MSC codes. 68M06, 65M12, 65M22

1. Introduction.

The integration of classical scientific computing and modern machine learning has given rise to the emerging field of *scientific machine learning*, offering a paradigm shift for approximating partial differential equations (PDEs). Leveraging abundant data and advances in computational power, recent machine learning developments have driven remarkable progress across disciplines, from image recognition [1] to cognitive science [2]. At the forefront of this advancement is the *physics-informed neural network* (PINN) framework [3, 4], which harnesses the universal approximation capabilities of deep neural networks to solve fractional PDEs by embedding the governing physical laws directly into the learning architecture.

PINNs are appealing because they are versatile and easy to implement, making them highly effective for solving diverse classes of PDEs. Their applications span computational physics, fluid dynamics [5, 6], meta-material design [7], and biomedical modeling [8]. Due to their mesh-free nature and reliance on randomly sampled collocation points, PINNs are well-suited for high-dimensional problems [9] and can handle complex geometries and irregular domains with ease [10, 11]. In particular, *fractional PINNs* (fPINNs) have emerged as a promising approach for modeling anomalous diffusion and memory-driven phenomena prevalent in fractional-order systems. By incorporating discretizations of fractional-order derivatives, such as the Riemann–Liouville or Caputo operators, into the loss function, fPINNs can accurately solve and identify parameters in nonlocal and weakly singular PDEs. Recent advance-

*Department of Mathematical Sciences, IIT (BHU), Varanasi 221005, India (himanshukrdwivedi.rs.mat21@itbhu.ac.in, rajeev.apm@iitbhu.ac.in).

†Corresponding author: Applied and Computational Mathematics, University of Wuppertal, Gaußstraße 20, 42119 Wuppertal, Germany (ehrhardt@uni-wuppertal.de).

ments include combining fPINNs with graded temporal schemes and fast convolution approximations to overcome the challenges posed by long-memory effects and singular behavior near the initial time.

In this work, we consider a nonlinear, parametrized time-fractional PDE of the general form

$$ {}_0^{\mathcal{C}}\partial_t^{\alpha} v + \mathcal{N}[v;\lambda] = 0, \quad t \in [0,T],\ \mathbf{x} \in \Omega, \tag{1.1} $$

where $\partial_t^{\alpha} = {}_0^{C}\mathcal{D}_t^{\alpha}$ denotes the *Caputo time-fractional derivative of order* α

$$ {}_0^{\mathcal{C}}\partial_t^{\alpha} u(t) := \big((\mathcal{RL})_t^{(1-\alpha)}(u)'\big)(t) = \int_0^t \omega_{1-\alpha}(t-\xi)\,(u(\xi))'\,d\xi, \quad 0<\alpha<1, \tag{1.2} $$

where $(\mathcal{RL})_t^{\beta}$ is defined below and referred as Riemann-Liouville fractional integral of order $\beta > 0$,

$$ (\mathcal{RL})_t^{\beta} := \int_0^t \omega_{\beta}(t-\xi)u(\xi)\,d\xi, \quad \omega_{\beta}(s) = \frac{s^{\beta-1}}{\Gamma(\beta)}. \tag{1.3} $$

The function $v(\cdot,t)$ represents the latent solution, and $\mathcal{N}[\cdot;\lambda]$ denotes a nonlinear differential operator parameterized by the vector λ. This formulation encompasses a wide range of mathematical models in physics, including conservation laws, diffusion processes, advection-reaction-diffusion systems, and kinetic equations. For instance, the 1D time-dependent viscous Burgers equation [12] corresponds to the case $\mathcal{N}[v;\lambda] = \lambda_1 v v_x - \lambda_2 v_{xx}$, where $\lambda = (\lambda_1, \lambda_2)$. Classical numerical approaches such as finite-difference, finite-element, and spectral methods [13] – have long been employed to address forward and inverse problems for *nonlinear fractional partial differential equations* (NfPDEs).

In this work, we develop a second-order Alikhanov scheme on nonuniform temporal grids, guided by the following key considerations

- Time-fractional PDEs with Caputo derivatives exhibit a weak singularity near $t = 0$, cf. [14, 15, 16]. Thus, graded meshes are recommended for accurately resolving the initial layer [15, 17].
- Classical schemes for time-fractional PDEs typically require $\mathcal{O}(MK_t)$ memory and $\mathcal{O}(MK_t^2)$ operations, where M and K_t denote the spatial and temporal grid sizes, respectively. To reduce this burden, we develop an accelerated Alikhanov scheme inspired by [18, 19, 20, 21, 22, 23].

The goal of inverse problems in fPDEs is to identify unknown functions or parameters using limited observational data, thereby revealing key mechanisms in physical, chemical, or biological systems. Traditional methods, such as regularization, eigenfunction expansions, and Laplace transforms [24, 25], are widely used, but they have limitations in high-dimensional settings and are sensitive to complex initial and boundary conditions. Additionally, they lack algorithmic flexibility. Inverse problems are commonly classified into two types: function reconstruction [26] and parameter estimation [27]. The latter is the focus of this work in the context of NfPDEs.

PINNs [3] provide a powerful alternative by embedding the governing physical laws directly into the neural network loss function during training. This allows PINNs to solve both forward and inverse fPDEs [28, 29] with enhanced generalizability, high-dimensional scalability [30], and robustness to noisy or sparse data [31]. However, a key challenge lies in the nonlocal nature of fractional derivatives, such as Caputo and Riemann–Liouville derivatives, which are not directly amenable to automatic differentiation. Several fPINN variants have been proposed to overcome this challenge by

incorporating finite difference [32], spectral, and power series [33] discretizations into the loss function. These models enable accurate solution recovery and parameter inference. Further developments include hybrid interpolation schemes [34], dual-network structures [35], and applications to complex fPDEs with oscillatory or singular dynamics.

Recent progress in fPINNs has focused on incorporating numerical schemes for Caputo derivatives into neural networks. However, challenges remain, particularly regarding the high cost due to memory effects, initial singularities, and suboptimal training efficiency in large-scale problems. To address these issues, we present an accelerated Alikhanov-fPINNs approach on nonuniform time grids for high-dimensional time-fractional PDEs. The key contributions are:

- To improve training efficiency, we impose initial and boundary conditions through suitable hard and soft constraint formulations, covering both homogeneous and nonhomogeneous cases. We further incorporate an adaptive activation function with a trainable parameter (a) and a scaling factor (n).
- The proposed method is validated across diverse benchmark tests, including high-dimensional forward and inverse problems, showcasing its robustness and computational effectiveness.

The manuscript is organised as follows: Section 2 introduces the neural network framework for nonlinear fractional PDEs. Section 2.2 derives the Alikhanov approximation and its accelerated version on nonuniform time grids. Section 2.3 presents the implementation of Alikhanov-based fPINNs/XfPINNs with adaptive activation functions. Section 3 reports numerical results for forward and inverse problems in multi-dimensional settings. The final conclusions are given in Section 4.

2. Neural Network Framework for Fractional PDEs. First, we revisit the core PINN methodology as applied to NfPDEs. Then, in Section 2.2, we present our Alikhanov–XfPINN framework for nonuniform time meshes of time-fractional NfPDEs. We provide a detailed global consistency error analysis for the accelerated sum-of-exponentials (SOE) approximation on nonuniform meshes, explicitly accounting for the initial singularity.

2.1. Physics-Informed Neural Networks. PINNs provide an effective framework for approximating complex PDEs, including models with nonlocal or fractional dynamics. Their main strength lies in embedding the governing physical laws directly into the training objective, thereby producing neural approximations that remain consistent with the underlying equations rather than relying solely on data. In practice, PINNs can be implemented efficiently using standard deep learning libraries such as TensorFlow and PyTorch.

First, consider a general nonlinear fractional PDE (NfPDE) defined over a spatial domain $\Omega \subset \mathbb{R}^d$, a time interval $t \in (0, T]$, and a fractional order $\alpha \in (0, 1)$. The problem is formulated through the governing operator ($\mathcal{G}$), boundary operator ($\mathcal{B}$), and initial operator ($\mathcal{I}$) as

$$\begin{aligned} \mathcal{G}[\mathbf{x}, t, v(\mathbf{x}, t; \lambda)] &= 0, \quad (\mathbf{x}, t) \in \Omega \times (0, T], \\ \mathcal{B}[\mathbf{x}, t, v(\mathbf{x}, t; \lambda)] &= 0, \quad \mathbf{x} \in \partial\Omega, ; t \in (0, T], \\ \mathcal{I}[\mathbf{x}, v(\mathbf{x}, 0; \lambda)] &= 0, \quad \mathbf{x} \in \Omega. \end{aligned}$$

Here, $v(\mathbf{x}, t)$ denotes the solution, and the operator $\mathcal{G}$ typically involves a Caputo-type fractional derivative $\partial_t^\alpha v$ along with nonlinear spatial components, as $\nabla v, \nabla^2 v, \ldots$.

Mathematically, a deep neural network can be viewed as a nested composition of

multivariate functions. A PINN seeks to approximate the mapping $v_{NN}(\mathbf{x},t)\colon \Omega \times [0,T] \to \mathbb{R}$, which associates each input pair $(\mathbf{x},t) \in \Omega \times [0,T]$ with the scalar output $v_{NN}(\mathbf{x},t) \in \mathbb{R}$ through a sequence of linear operations and nonlinear activations. The resulting representation for $1 \le l \le L-1$ is given by

$$\begin{aligned} \text{input layer}: &\quad NN^0 = (\mathbf{x},t), \\ \text{hidden layers}: &\quad NN^l = \sigma\big(w^l \cdot NN^{l-1} + b^l\big), \\ \text{output layer}: &\quad NN^L = w^L \cdot NN^{L-1} + b^L. \end{aligned} \tag{2.1}$$

Here, NN^l denotes the output of the l-th layer. w and b are the trainable weight matrices and bias vectors, respectively, and σ is the activation function. For each hidden layer ($l \ge 1$), the previous layer's output is first mapped by a learnable affine transformation and then passed through σ, and finally forwarded to the next layer. The final layer only applies the affine mapping to produce the network's output. This architecture is purely feed-forward, with no recurrent or feedback connections.

In PINNs, the governing PDE is incorporated explicitly into the loss functional, ensuring that the learned solution satisfies the underlying physics. To illustrate this framework, we consider a general form of a NfPDE.

$$\begin{aligned} {}_0^{\mathcal{C}}\partial_t^{\alpha} v + \mathcal{N}[v;\lambda] &= g(\mathbf{x},t), \quad t \in [0,T],\, x \in \Omega, \\ v(\mathbf{x},0) &= \varphi(\mathbf{x}), \qquad \mathbf{x} \in \Omega, \\ v(\mathbf{x},t) &= \phi(\mathbf{x},t), \quad (\mathbf{x},t) \in \partial\Omega \times [0,T], \end{aligned} \tag{2.2}$$

where $v(\mathbf{x}a,t)$ denotes the exact analytical solution, $\varphi(\mathbf{x})$ specifies the initial conditions, $\phi(\mathbf{x},t)$ enforces the boundary constraints and $\partial\Omega$ marks the spatial boundary of Ω. The PINNs embeds the NfPDE (2.2) by minimizing the residual between the PDE's left- and right-hand sides. All required derivatives are obtained via automatic differentiation to ensure a consistent application of the chain rule. Let the training data be denoted by

$$\mathcal{I} = \mathcal{I}_f, \mathcal{I}_{ic}, \mathcal{I}_{bc},$$

where $(\mathcal{I}_f)$, $(\mathcal{I}_{ic})$, and $(\mathcal{I}_{bc})$ represent the interior collocation, initial-condition, and boundary-condition datasets, respectively:

$$\begin{aligned} \mathcal{I}_f &= \big\{(\mathbf{x}_f^i, t_f^i, g(\mathbf{x}_f^i, t_f^i))\big\}_{i=1}^{N_f}, \\ \mathcal{I}_{ic} &= \big\{(\mathbf{x}_{ic}^i, 0, \varphi(\mathbf{x}_{ic}^i))\big\}_{i=1}^{N_{ic}}, \\ \mathcal{I}_{bc} &= \big\{(\mathbf{x}_{bc}^i, t_{bc}^i, \phi(\mathbf{x}_{bc}^i, t_{bc}^i))\big\}_{i=1}^{N_{bc}}. \end{aligned}$$

Accordingly, the total loss associated with PINN is defined as

$$\mathcal{L}(\Theta) = \mathrm{MSE}_f + \mathrm{MSE}_{ic} + \mathrm{MSE}_{bc}\,. \tag{2.3}$$

where the mean-squared error terms corresponding to the PDE residual, initial con-

dition, and boundary constraints are defined as follows:

$$\mathrm{MSE}_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left(\frac{\partial v_{NN}(\mathbf{x}_f^i, t_f^i)}{\partial t} + \mathcal{N}[v_{NN}(\mathbf{x}_f^i, t_f^i); \lambda] - g(\mathbf{x}_f^i, t_f^i)\right)^2,$$

$$\mathrm{MSE}_{ic} = \frac{1}{N_{ic}}\sum_{i=1}^{N_{ic}}\big(v_{NN}(\mathbf{x}_{ic}^i, 0) - \varphi(\mathbf{x}_{ic}^i)\big)^2,$$

$$\mathrm{MSE}_{bc} = \frac{1}{N_{bc}}\sum_{i=1}^{N_{bc}}\big(v_{NN}(\mathbf{x}_{bc}^i, t_{bc}^i) - \phi(\mathbf{x}_{bc}^i, t_{bc}^i)\big)^2.$$

Here, Θ denotes the complete set of trainable network parameters, including weights and biases. The objective functional $\mathcal{L}(\Theta)$ quantifies how well the neural approximation v_{NN} satisfies the NfPDE together with the prescribed initial and boundary data. Although v_{NN} is generally not identical to the exact solution $(v(\mathbf{x}, t))$, the mismatch can be reduced by optimizing Θ. Thus, the training process seeks

$$\Theta^* = \arg\min_{\Theta} \mathcal{L}(\Theta),$$

hence, v_{NN} produces a sufficiently small residual in (2.2) and satisfies the imposed constraints with high accuracy. This optimization procedure constitutes the training stage, during which the network learns an effective approximation to the NfPDE solution.

2.2. Extension to Alikhanov–XfPINNs on Nonuniform Time Meshes. Motivated by the strengths of PINNs, we propose Alikhanov–XfPINNs, a generalized framework designed to solve nonlinear fractional PDEs on nonuniform time grids. A central challenge in solving such problems is the nonlocal nature of the Caputo derivative, which does not permit the direct application of the chain rule. To overcome this challenge, we use a second-order Alikhanov scheme to discretize the temporal fractional derivative on a nonuniform mesh and integrate it directly into the loss function.

To accelerate convergence and enhance representation power, we also incorporate adaptive activation functions. Although this study focuses on the Caputo derivative, the Alikhanov–XfPINNs framework can be adapted to other types of fractional operators (e.g., the Riemann–Liouville derivative) by modifying the corresponding discretization technique. This flexibility makes the proposed methodology a versatile tool for solving a broad class of fractional PDEs.

2.2.1. ***Time Discretization on Nonuniform Grids.*** Throughout this work, we employ nonuniform temporal discretizations. Let

$$0 = t_0 < t_1 < \cdots < t_{k-1} < t_k < \cdots < t_{K_t} = T$$

be a partition of the time interval $[0, T]$, with local time step

$$\tau_k := t_k - t_{k-1}, \qquad 1 \le k \le K_t,$$

and maximum step size $\tau := \max_{1\le k\le K_t} \tau_k$. For $\theta = \alpha/2 \in (0, 1)$, the off-set time level is defined as

$$t_{k-\theta} := \theta t_{k-1} + (1-\theta) t_k.$$

The adjacent step ratio and its global bound are denoted by

$$\rho_k := \frac{\tau_k}{\tau_{k+1}}, \qquad \rho := \max_{k\geq 1} \rho_k.$$

We impose the following mesh conditions.

M1: The maximum step ratio satisfies

$$\rho \leq \frac{3}{2}.$$

Assumption **M1** permits moderate variation between consecutive time steps. In particular, it allows time steps to decrease by at most a factor of 2/3, while no additional restriction is imposed on their growth. Such a condition is standard in variable-step fractional discretizations, although an upper bound on τ may still be required in the stability analysis.

Nonuniform meshes are especially useful for fractional evolution problems, since their solutions often show weak singular behavior around $t = 0$, for example

$$u_t = \mathcal{O}(t^{\alpha-1}) \qquad \text{as } t \to 0,$$

even for smooth data. Graded meshes of the form

$$t_n = T\Big(\frac{n}{K_t}\Big)^{\gamma}$$

are therefore commonly used to cluster time levels near $t = 0$ and improve accuracy; see, for example, [15]. Additional mesh-regularity assumptions are needed to ensure convergence on general nonuniform grids [36].

M2: There exist constants $C_{1\gamma}, C_{2\gamma} > 0$, independent of the mesh size and a grading parameter $\gamma \geq 1$ such that

$$\tau_n \leq \tau \min\{1, C_{1\gamma} t_n^{1-1/\gamma}\}, \qquad 1 \leq n \leq K_t,$$

and

$$t_n \leq C_{2\gamma} t_{n-1}, \qquad 2 \leq n \leq K_t.$$

The qunatity γ determines the degree of refinement near the origin. The choice $\gamma = 1$ corresponds to a quasi-uniform mesh satisfying **M2**, while larger values of γ produce stronger clustering near $t = 0$ and are better suited for resolving initial singularities.

2.2.2. ***Alikhanov Formula for a Nonuniform Time Mesh.*** We consider a grid function $\{v^k\}_{k=0}^K$ on a nonuniform time mesh. Next, we define the weighted operator $v^{k-\theta} = \theta v^{k-1} + (1-\theta)v^k$ and the standard difference operator $\nabla_\tau v^k = v^k - v^{k-1}$. To streamline the presentation and clarify the notation, we introduce:

$$\bar{\omega}_k = \omega_{2-\alpha}(t - t_{k-\theta}), \quad \text{and} \quad \bar{\omega}'_k = \omega_{1-\alpha}(t_{k-\theta} - t), \quad t \in [0, t_{k-\theta}].$$

Let $(\mathcal{L}_{1,k}v)$ denote the linear interpolant of v at the nodes (t_{k-1}) and (t_k), while $(\mathcal{Q}_{2,k}v)$ represents the quadratic interpolant constructed from $(t_{k-1}), (t_k)$, and (t_{k+1}).

We define

$$(\mathcal{L}_{1,k}v)'(t) = \frac{\nabla_\tau v^k}{\tau_k}, \quad (\mathcal{Q}_{2,k}v)'(t) = \frac{\nabla_\tau v^k}{\tau_k} + \frac{2(t - t_{k-1/2})}{\tau_k(\tau_k + \tau_{k+1})}(\rho_k \nabla_\tau v^{k+1} - \nabla_\tau v^k). \tag{2.4}$$

We now present the *nonuniform Alikhanov formula*, which reads:

$$
\begin{aligned}
({}_0^{\mathcal{C}}\partial_t^{\alpha} v)^{k-\theta} &= \int_{t_{k-1}}^{t_{k-\theta}} \omega_{1-\alpha}(t_{k-\theta}-s)\,(\mathcal{L}_{1,k}v)'(s)\,ds \\
&\quad + \sum_{n=1}^{k-1}\int_{t_{n-1}}^{t_n} \omega_{1-\alpha}(t_{k-\theta}-s)\,(\mathcal{Q}_{2,n}v)'(s)\,ds \\
&= \mathbf{a}^{(0,k)}\nabla_\tau v^k + \sum_{n=1}^{k-1}\big(\mathbf{a}^{(k-n,k)}\nabla_\tau v^n + \mathbf{b}^{(k-n,k)}(\rho_n\nabla_\tau v^{n+1} - \nabla_\tau v^n)\big).
\end{aligned} \tag{2.5}
$$

We define

$$
\mathbf{a}^{(k-n,k)} = \frac{1}{\tau_n}\int_{t_n}^{\min\{t_n,t_{k-\theta}\}} \bar{\omega}_k'(s)\,ds, \quad 1\le n\le k, \tag{2.6}
$$

$$
\mathbf{b}^{(k-n,k)} = 2\int_{t_{n-1}}^{t_n} \frac{(s-t_{n-1/2})}{\tau_n(\tau_n+\tau_{n+1})}\bar{\omega}_k'(s)\,ds, \quad 1\le n\le k-1. \tag{2.7}
$$

Rearranging the terms in (2.5), we obtain the compact form

$$
({}_0^{\mathcal{C}}\partial_t^{\alpha} v)^{k-\theta} = \sum_{n=1}^{k}\mathcal{D}^{(k-n,k)}\nabla_\tau v^n, \tag{2.8}
$$

where the discrete convolution kernels are given by $\mathcal{D}^{(0,1)} = a^{(0,1)}$ if $k=1$, and

$$
\mathcal{D}^{(k-n,k)} = \begin{cases} \mathbf{a}^{(k-1,k)} - \mathbf{b}^{(k-1,k)}, & n=1, \\ \mathbf{a}^{(k-n,k)} + \rho_{n-1}\mathbf{b}^{(k-n+1,k)} - \mathbf{b}^{(k-n,k)}, & 2\le n\le k-1, \\ \mathbf{a}^{(0,k)} + \rho_{k-1}\mathbf{b}^{(1,k)}, & n=k. \end{cases} \tag{2.9}
$$

The nonuniform formula (2.5) generalizes the Alikhanov scheme on uniform meshes [37] and the nonuniform counterpart was first introduced in [23] to address initial singularities via graded meshes. The following recent developments concerning the properties of the coefficients of accelerated approximation can be found in [38].

LEMMA 2.1 ([38]). *Assume that the time-step condition* **M1** *is satisfied. We then examine the discrete convolution kernels* $\mathcal{D}^{(k-n,k)}$ *defined in* (2.9)*. The following properties hold:*

(a) *For* $1\le n\le k$*, the kernels* $\mathcal{D}^{(k-n,k)}$ *satisfy*

$$
\begin{aligned}
&\frac{24}{11\tau_k}\int_{t_{k-1}}^{t_k}\omega_{1-\alpha}(t_k-s)\,ds \ge \mathcal{D}^{(0,k)}, \\
&\mathcal{D}^{(k-n-1,k)} - \mathcal{D}^{(k-n,k)} \ge \frac{4}{11\tau_k}\int_{t_{k-1}}^{t_k}\omega_{1-\alpha}(t_n-s)\,ds.
\end{aligned} \tag{2.10}
$$

(b) *When* $1\le n\le k-1$*, the kernels* $\mathcal{D}^{(k-n,k)}$ *are monotone*

$$
\begin{aligned}
0 \le (1+\rho_k)\mathbf{b}^{(k-n,k)} &- \frac{1}{5\tau_n}\int_{t_{n-1}}^{t_n}(t_n-s)\omega_{1-\alpha}(t_{k-\theta}-s)\,ds \\
&< \mathcal{D}^{(k-n-1,k)} - \mathcal{D}^{(k-n,k)},
\end{aligned} \tag{2.11}
$$

(c) *For $k \geq 2$, the leading kernel $\mathcal{D}^{(0,k)}$ dominates the subsequent kernel, namely,*

$$\mathcal{D}^{(1,k)} < \frac{(1-2\theta)}{(1-\theta)}\,\mathcal{D}^{(0,k)}. \tag{2.12}$$

2.2.3. ***Accelerated Alikhanov formula for nonuniform time mesh.*** The approximation (2.9) becomes expensive in long-time simulations because the fractional derivative retains the full solution history. To reduce this memory and computational burden, we use the sum-of-exponentials approximation [18] for the kernel, yielding an accelerated approximation. The main idea is as follows.

LEMMA 2.2 ([18]). *When $0 < \alpha < 1$, the absolute tolerance error $\epsilon \ll 1$, a cut-off time size $\Delta t > 0$, and a final time T are given, then there exists a positive integer $\mathcal{N}_q$, a set of positive quadrature nodes s^l, and their respective positive weights $\nu^l (1 \leq l \leq \mathcal{N}_q)$, with*

$$\left|\omega_{1-\alpha}(t) - \sum_{l=1}^{\mathcal{N}_q} \nu^l \,\mathrm{e}^{-s^l t}\right| \leq \epsilon, \quad \forall t \in [\Delta t, T], \tag{2.13}$$

and the number $\mathcal{N}_q$ of required quadrature nodes is of order

$$\mathcal{N}_q = O\left(\log\frac{1}{\epsilon}\left(\log\log\frac{1}{\epsilon} + \log\frac{T}{\Delta t}\right) + \log\frac{1}{\Delta t}\left(\log\log\frac{1}{\epsilon} + \log\frac{1}{\Delta t}\right)\right). \tag{2.14}$$

As shown in Lemma 2.2, we decompose the Caputo derivative (1.1) into a local term over $[t_{k-1}, t_{k-\theta}]$ and a history term over $[0, t_{k-1}]$. The local term is approximated using linear interpolation of $v'(t)$. The history term is efficiently evaluated via the sum-of-exponentials approximation of $\bar{\omega}_k'(s)$, yielding:

$$\begin{aligned}
({}_0^{\mathcal{C}}\partial_t^{\alpha} v)^{k-\theta} &\approx \int_{t_{k-1}}^{t_{k-\theta}} \bar{\omega}_k'(s)(\mathcal{L}_{1,k} v)'(s)\,ds + \int_0^{t_{k-1}} \sum_{l=1}^{\mathcal{N}_q} \nu^l \,\mathrm{e}^{-s^l(t_{k-\theta}-s)} v'(s)\,ds, \\
&= \boldsymbol{a}^{(0,k)} \nabla_\tau v^k + \sum_{l=1}^{\mathcal{N}_q} \nu^l \mathcal{V}_{his}^l(t_{k-1}), \quad k \geq 1,
\end{aligned} \tag{2.15}$$

where $\mathcal{V}_{his}^l(t_0) = 0$ and $\mathcal{V}_{his}^l(t_k) = \int_0^{t_k} \mathrm{e}^{-s^l(t_{k-\theta}-\xi)} v'(\xi)\,d\xi$. To evaluate $(\mathcal{V}_{\mathrm{his}}^l(t_k))$, we employ a recursive update obtained by approximating $v'(t)$ with a quadratic interpolant on each subinterval $[t_{l-1}, t_l]$, $1 \leq l \leq k-1$, as follows:

$$\begin{aligned}
\mathcal{V}_{his}^l(t_n) &\approx \int_0^{t_{n-1}} \mathrm{e}^{-s^l(t_{n+1-\theta}-s)} v'(s)\,ds + \int_{t_{n-1}}^{t_n} \mathrm{e}^{-s^l(t_{n+1-\theta}-s)} (\mathcal{Q}_{2,n} v)'(s)\,ds, \\
&= \mathrm{e}^{-s^l(\theta\tau_n + (1-\theta)\tau_{n-1})} \mathcal{V}_{his}^l(t_{n-1}) + \boldsymbol{c}^{(n,l)} \nabla_\tau v^n + \boldsymbol{d}^{(n,l)} (\rho_n \nabla_\tau v^{n+1} - \nabla_\tau v^n),
\end{aligned} \tag{2.16}$$

where the discrete coefficients are defined by

$$\boldsymbol{c}^{(n,l)} = \frac{1}{\tau_n} \int_{t_{n-1}}^{t_n} \mathrm{e}^{-s^l(t_{n+1-\theta}-\xi)}\,d\xi, \tag{2.17}$$

$$\boldsymbol{d}^{(n,l)} = \int_{t_{n-1}}^{t_n} \mathrm{e}^{-s^l(t_{n+1-\theta}-\xi)} \frac{2(\xi - t_{n-1/2})}{\tau_n(\tau_n + \tau_{n+1})}\,d\xi. \tag{2.18}$$

From (2.15)–(2.16), we obtain the following form of the approximation

$$({}_0^{\mathcal{FC}}\partial_t^{\alpha} v)^{k-\theta} = \boldsymbol{a}^{(0,k)} \nabla_\tau v^k + \sum_{l=1}^{\mathcal{N}_q} \nu^l \mathcal{V}_{his}^l(t_{k-1}), \quad k \geq 1. \tag{2.19}$$

Here, $\mathcal{V}_{\text{his}}^l(t_k)$ is evaluated through the following recurrence relation.

$$\mathcal{V}_{his}^l(t_n) = \mathrm{e}^{-s^l(\theta\tau_n+(1-\theta)\tau_{n+1})}\mathcal{V}_{his}^l(t_{n-1}) + \boldsymbol{c}^{(n,l)} \nabla_\tau v^n + \boldsymbol{d}^{(n,l)}(\rho_n \nabla_\tau v^{n+1} - \nabla_\tau v^n). \tag{2.20}$$

By Lemma 2.2, the number of quadrature nodes satisfies $\mathcal{N}_q = \mathcal{O}(\log N)$ for $T \geq 1$ and $\mathcal{N}_q = \mathcal{O}(\log^2 N)$ for $T \approx 1$, leading to substantial reductions in computational cost and memory consumption.

We examine the scheme by defining the consistency error associated with the accelerated approximation (2.15):

$$\Upsilon^j[u] := ({}_0^{\mathcal{C}}\partial_t^{\alpha} u)(t_{j-\theta}) - ({}_0^{\mathcal{FC}}\partial_t^{\alpha} u)^{j-\theta}, \qquad j \geq 1.$$

Following the framework of [38], Lemma 2.3 establishes a discrete convolution-type bound for $\Upsilon^j[u]$ on general nonuniform time meshes. Furthermore, the fractional Grönwall lemma shows that the dominant contribution to the error in solution come from the convolution part $\sum_{j=1}^n \mathsf{C}^{(n-j,n)}|\Upsilon^j[u]|$, here complementary kernels $\mathsf{C}^{(k-n,k)}$ defined as

$$\begin{aligned} \mathsf{C}^{(0,k)} &:= \frac{1}{\mathcal{D}^{(0,k)}}, \\ \mathsf{C}^{(k-n,k)} &:= \frac{1}{\mathcal{D}^{(0,n)}} \sum_{i=n+1}^{k} (\mathcal{D}^{(i-n-1,i)} - \mathcal{D}^{(i-n,i)})\mathsf{C}^{(k-i,k)}, \quad 1 \leq n \leq k-1. \end{aligned} \tag{2.21}$$

Lemma 2.3 ([23]). *Suppose that the condition* **M1** *on the step ratio is satisfied, and that* $v \in \mathcal{C}^3(0,T]$ *with* $\int_0^T \xi^2 |v'''(\xi)|\, d\xi < \infty$*, moreover a constant* $\mathcal{C}_v > 0$ *such that*

$$|v'''(t)| \leq \mathcal{C}_v\big(1 + t^{\upsilon-2}\big), \qquad 0 < t \leq T,$$

where $\upsilon \in (0,1)$ *characterizes the temporal regularity. For the fast Alikhanov formula on nonuniform grids* (2.15) *with the discrete convolution kernels, denoted as* $\mathcal{D}^{(k-n,k)}$*, along with the local consistency error* $\Upsilon^{j-\theta}$*, a convolutional framework is exhibited, as illustrated below*

$$\begin{aligned} &|\Upsilon^{k-\theta}[v]| \\ &\leq \mathcal{D}^{(0,k)}\mathcal{G}_{loc}^k + \sum_{i=1}^{k-1}\Big(\mathcal{D}^{(k-i-1,k)} - \mathcal{D}^{(k-i,k)}\Big)\mathcal{G}_{his}^i + \frac{\mathcal{C}_v}{\upsilon}\hat{t}_{k-1}^2\epsilon, \quad 1 \leq k \leq K_t, \end{aligned} \tag{2.22}$$

where the two quantities $\mathcal{G}_{loc}$ *and* $\mathcal{G}_{his}$ *are*

$$\begin{aligned} \mathcal{G}_{loc}^k &= \frac{3}{2}\bigg(\int_{t_{k-1}}^{t_{k-1/2}} (s-t_{k-1})^2 |v'''(s)|\, ds + \tau_k \int_{t_{k-1/2}}^{t_{k-1}} (t_k - s)|v'''(s)|\, ds\bigg), \\ \mathcal{G}_{his}^k &= \frac{5}{2}\bigg(\int_{t_{k-1}}^{t_k} (s-t_{k-1})^2 |v'''(s)|\, ds + \int_{t_k}^{t_{k+1}} (t_{k+1} - s)^2 |v'''(s)|\, ds\bigg), \end{aligned}$$

and $\hat{t}_k = \max_{1\le k\le K_t}\{1, t_k\}$. *Then, the global convolution error is*

(2.23)
$$\sum_{i=1}^{k} \mathsf{C}^{(k-i,k)}|\Upsilon^i[v]| \le \sum_{i=1}^{k} \mathsf{C}^{(k-i,k)}\mathcal{D}^{(0,k)}\mathcal{G}^k_{loc} + \sum_{i=1}^{k} \mathsf{C}^{(k-i,k)}\mathcal{D}^{(0,k)}\mathcal{G}^k_{his} + \frac{\mathcal{C}_v}{\upsilon} t_k^{\alpha}\hat{t}^2_{k-1}\epsilon.$$

Lemma 2.3 directly yields the following theorem, which establishes a global error bound for the accelerated approximation.

THEOREM 2.4 ([38]). *Suppose that* $v \in \mathcal{C}^3(0,T]$ *and a constant* $\mathcal{C}_v > 0$ *such that*

$$|v'''(t)| \le \mathcal{C}_v\big(1 + t^{\upsilon-2}\big), \quad 0 < t \le T,$$

where temporal regularity $\upsilon \in (0,1)$. *Under the assumptions stated in* **M1**, *the global consistency error of the accelerated approximation* (2.19) *satisfies the following estimate:*

(2.24)
$$\sum_{i=1}^{k} \mathsf{C}^{(k-i,k)}|\Upsilon^i[v]| \le \mathcal{C}_v\left(\frac{\tau_1^{\upsilon}}{\upsilon} + \frac{1}{(1-\alpha)}\max_{2\le n\le k} t_n^{\alpha} t_{n-1}^{\upsilon-3}\tau_n^3\tau_{n-1}^{-\alpha} + \frac{\epsilon}{\upsilon}t_k^{\alpha}\hat{t}^2_{k-1}\right), \quad 1 \le n \le K_t.$$

Particularly, under a temporal discretization satisfying the graded-type condition **M2**, *it follows that*

(2.25)
$$\sum_{i=1}^{k} \mathsf{C}^{(k-i,k)}|\Upsilon^i[v]| \le \frac{\mathcal{C}_v}{\upsilon(1-\alpha)}\left(\tau^{\min(\gamma\upsilon,2)} + \epsilon\right).$$

The following theorem shows that the error in the temporal direction of the term $v^{k-\theta}$ is limited by the Alikhanov approximation error.

THEOREM 2.5 ([38]). *Let* $v \in C^2((0,T])$ *and a positive constant* $\mathcal{C}_v$ *such that*

$$|v''(t)| \le \mathcal{C}_v\big(1 + t^{\upsilon-2}\big), \qquad 0 < t \le T,$$

where $\upsilon \in (0,1)$ *denotes the temporal regularity parameter. The local truncation error associated with* $v^{k-\theta}$ *is defined by*

$$\mathcal{R}^k[v] := v(t_{k-\theta}) - v^{k-\theta}, \qquad 1 \le k \le K_t.$$

Under the graded-type mesh condition **M2**, *the corresponding global consistency error is bounded for* $1 \le k \le K_t$ *as follows:*

(2.26)
$$\sum_{i=1}^{k} \mathsf{C}^{(k-i,k)}|\mathcal{R}^i_{\tau}[v]| \le \mathcal{C}_v\left(\frac{\tau_1^{\upsilon+\alpha}}{\upsilon} + t_k^{\alpha}\max_{2\le n\le k} t_{n-1}^{\upsilon-2}\tau_n^2\right)$$

is taken into account.

2.3. Adaptive Activation Function. To optimize the Alikhanov–XfPINN on graded time grids more quickly, each standard activation is replaced by its adaptive counterpart. his counterpart is parameterized by a trainable slope a and a prescribed

scaling factor n, as proposed by Jagtap, Kawaguchi, and Karniadakis [39]. The six adaptive activation functions considered in this work are listed in Table 1. The parameter a adjusts the local steepness of the activation and thus modifies the gradient flow during optimization, as illustrated in Figure 1. The additional scaling factor n amplifies this adaptive effect and helps mitigate slow convergence caused by small learning rates.

For the Alikhanov–XfPINN framework on a nonuniform temporal mesh, the neural approximation is written as

$$v_{NN}(\mathbf{x},t;\Theta) = \big(\phi_h \circ \sigma \circ na\,\phi_{h-1} \circ \cdots \circ \sigma \circ na\,\phi_1\big)(\mathbf{x},t), \tag{2.27}$$

where $\circ$ denotes function composition and

$$\phi_i = W_i\phi_{i-1} + b_i, \qquad i = 1,\dots,h.$$

Here, the spatio-temporal input is $\phi_0 = (\mathbf{x},t)$, and the resulting network output is $v_{NN}(\mathbf{x},t;\Theta)$. In the forward $\mathcal{FL}_{21}\sigma$-XfPINN setting, the trainable parameter set is

$$\Theta = \{W_i, b_i, a\}_{i=0}^{h},$$

whereas, in the inverse Alikhanov–XfPINN formulation, the unknown physical parameter ζ is also learned, giving

$$\Theta = \{W_i, b_i, a, \zeta\}_{i=0}^{h}.$$

Table 1: Performance comparison: standard vs. adaptive variants of activation functions.

Activation	**Fixed**	**Adaptive**
Sigmoid	$\dfrac{1}{1+\mathrm{e}^{-\mathbf{x}}}$	$\dfrac{1}{1+\mathrm{e}^{-na\,\mathbf{x}}}$
Swish	$\dfrac{\mathbf{x}}{1+\mathrm{e}^{-\mathbf{x}}}$	$\dfrac{na\,\mathbf{x}}{1+\mathrm{e}^{-na\,\mathbf{x}}}$
SeLU	$\begin{cases}\lambda\,\mathbf{x}, & \mathbf{x}>0,\\ \lambda\,\alpha\,(\mathrm{e}^{\mathbf{x}}-1), & \mathbf{x}\le 0\end{cases}$	$\begin{cases}\lambda\,na\,\mathbf{x}, & \mathbf{x}>0,\\ \lambda\,\alpha\,(\mathrm{e}^{na\,\mathbf{x}}-1), & \mathbf{x}\le 0\end{cases}$
ReLU	$\max(0,\mathbf{x})$	$\max(0,\ na\,\mathbf{x})$
tanh	$\tanh(\mathbf{x})$	$\tanh(na\,\mathbf{x})$
$\mathbf{x}\tanh$	$\mathbf{x}\,\tanh(\mathbf{x})$	$na\,\mathbf{x}\,\tanh(na\,\mathbf{x})$

2.4. Loss Function. We construct the proposed framework by embedding the prescribed constraints directly into the trial solution. Unlike penalty-based approaches

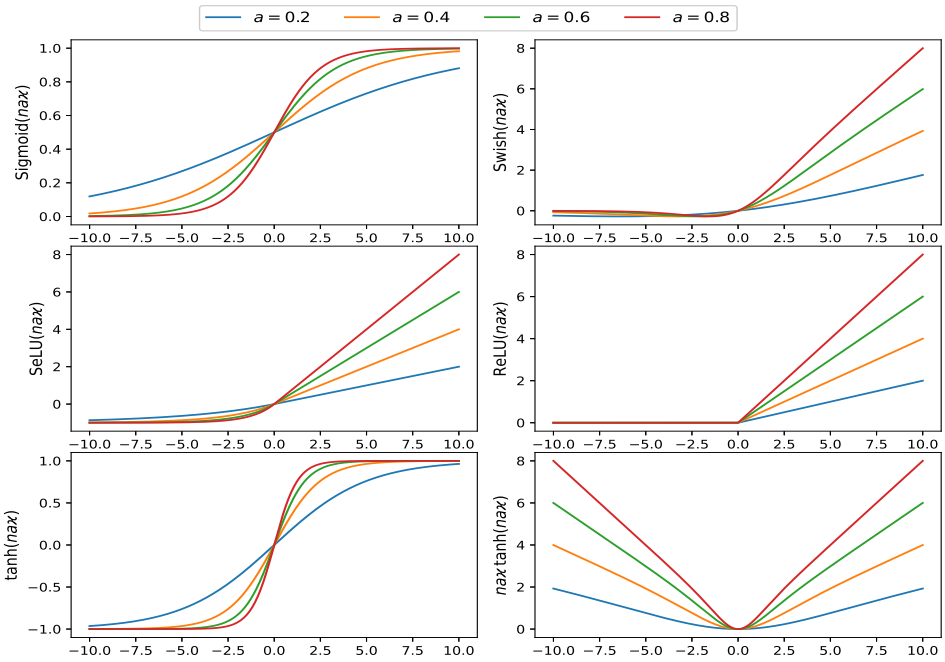


Fig. 1: Standard activation functions modified adaptively by varying the hyperparameter a, with the scaling factor fixed at $n = 1$.

to initial and boundary conditions, this strategy incorporates the prescribed constraints directly into the neural ansatz [40]. Consequently, separate penalty terms for these conditions are not required in the loss functional, and fewer training samples are needed for constraint enforcement.

Specifically, we define the constrained neural approximation by

$$\tilde{v}(\mathbf{x}, t) = t\,\rho(\mathbf{x})\,v_{NN}(\mathbf{x}, t; \Theta), \tag{2.28}$$

where $v_{NN}(\mathbf{x}, t; \Theta)$ denotes the neural-network output and $\rho(\mathbf{x})$ is a given boundary mask chosen to satisfy

$$\rho(\mathbf{x}) = 0, \qquad \mathbf{x} \in \partial\Omega.$$

The factor t in (2.28) enforces the initial condition, while $\rho(\mathbf{x})$ imposes the homogeneous boundary constraint. Therefore, the approximation $\tilde{v}(\mathbf{x}, t)$ satisfies the prescribed initial and boundary conditions (IBCs) by construction. Figure 2 depicts the proposed Alikhanov–XfPINN structure under a graded time discretization, where the IBCs are enforced through the hard-constraint ansatz.

Since the IBCs are already embedded in the ansatz, the training set only contains interior residual points

$$\mathcal{I} = \{\mathcal{I}_f\}, \qquad \mathcal{I}_f = \{(\mathbf{x}_f^i, t_f^i, g(\mathbf{x}_f^i, t_f^i))\}_{i=1}^{N_f}.$$

Here, N_f represents the total number of residual sampling points. Therefore, the

forward-training objective only contains the NfPDE residual term:

$$\mathcal{L}(\Theta) = \mathrm{MSE}_f, \tag{2.29}$$

with

$$\mathrm{MSE}_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\Big[{}_0^{\mathcal{C}}\partial_t^{\alpha}\tilde{v}(\mathbf{x}_f^i, t_f^i) + \mathcal{N}(\tilde{v}(\mathbf{x}_f^i, t_f^i); \lambda) - g(\mathbf{x}_f^i, t_f^i)\Big]^2.$$

In the inverse Alikhanov–XfPINN formulation, we augment the forward-loss with a terminal-time term so that the NfPDEs parameters and network weights are learned simultaneously. We define the terminal dataset $\mathcal{I}_T = \big\{(\mathbf{x}_T^i, t_T^i, \psi(\mathbf{x}_T^i))\big\}_{i=1}^{N_T}$, where N_T is its size. With the combined dataset $\mathcal{I} = \{\mathcal{I}_f, \mathcal{I}_T\}$, the hard-constraint inverse loss is

$$\mathcal{L}_{\mathrm{inv}}(\Theta) = w_f\,\mathrm{MSE}_f + w_T\,\mathrm{MSE}_T, \tag{2.30}$$

where

$$\mathrm{MSE}_T = \frac{1}{N_T}\sum_{i=1}^{N_T}\Big[\tilde{v}(\mathbf{x}_T^i, T) - \psi(\mathbf{x}_T^i)\Big]^2.$$

The learnable parameters are $\Theta = \{W_i, b_i, a, \zeta\}_{i=0}^{h}$, and w_f, w_T are the respective weights for MSE_f and MSE_T.

2.5. Adaptive Training Process. Pang, Lu, and Karniadakis [32] developed fractional PINNs for space-time fractional advection-diffusion models. They treated fractional derivatives using finite-difference discretizations and computed integer-order differential terms via automatic differentiation. For models involving Caputo time-fractional derivatives, they employed the standard $\mathcal{L}1$ discretization and determined the trainable parameters Θ^* by optimizing the physics-informed loss.

To reduce the cost associated with the nonlocal memory of the Caputo operator and better resolve the weak singularity near $t = 0$, Shi, Liu, and Yang [41] developed the FL1–fPINN method on graded meshes. This approach combines an accelerated nonuniform $\mathcal{L}1$ approximation with the fPINN framework to efficiently simulate multi-dimensional time-fractional PDEs.

In this work, we focus on models governed by time-fractional dynamics. While fPINNs and FL1–fPINNs can both be applied to the time-fractional Allen–Cahn (TFAC) problem, our numerical results demonstrate that the Alikhanov–XfPINN framework offers higher convergence rates and significantly greater accuracy across the benchmark tests considered.

First, we replace the accelerated $\mathcal{L}1$ formula with an accelerated Alikhanov approximation on a graded mesh to improve the convergence rate. As Theorems 2.4 and 2.5 establish, this approximation achieves order 2 for smooth solutions. For solutions with an initial singularity, the convergence rate is $\min\{\gamma\alpha, 2\}$, where γ denotes the grading parameter. Depending on the value of γ, the convergence order can be as high as 2. Thus, by minimizing the loss function, we can significantly improve performance.

Second, we introduce a residual-based progressive training strategy that improves computational efficiency and stabilizes the optimization process. At stage j, the

residual training set is restricted to the current temporal window and is defined as

$$\mathcal{I}_f^j = \big\{(\mathbf{x}_f^i, t_f^i, g(\mathbf{x}_f^i, t_f^i)) \in \mathcal{I}_f \colon t_f^i \le t_{j-\theta}\big\}, \qquad j = 1, \dots, K-1.$$

The training process begins with the initial short-time interval $[t_{1-\theta}, t_{2-\theta}]$, in which the network uses the dataset I_f^1 to learn the local approximation $\tilde{v}^1$. For the subsequent stages ($j = 3, \dots, K$), the temporal domain is gradually expanded to the interval $[t_{1-\theta}, t_{j-\theta}]$, and the network is retrained with an updated dataset $\mathcal{I}_f^j$. In the final stage, the network uses the dataset $\mathcal{I}_f^K$ to approximate the function $\tilde{v}$ over the entire computational interval, $[0, t_{K-\theta}]$. Each stage is stopped once the prescribed tolerance is reached or the maximum number of iterations is attained. At stage j, the trainable parameters are obtained by solving

$$\mathcal{L}^j(\Theta) = \frac{1}{N_f^j} \sum_{(\mathbf{x}_f, t_f, g) \in \mathcal{I}_f^j} \Big[{}_0^{\mathcal{C}}\partial_t^\alpha \tilde{v}(\mathbf{x}_f, t_f) + \mathcal{N}(\tilde{v}(\mathbf{x}_f, t_f); \lambda) - g \Big]^2, \tag{2.31}$$

yielding $\Theta_j^* = \arg\min \mathcal{L}^j(\Theta)$, where $N_f^j = |\mathcal{I}_f^j|$. We follow the exact same stage-wise schedule as in the inverse Alikhanov–XfPINN formulation, but now we minimize the minimize the inverse loss with a hard-constraint at stage j:

$$\begin{aligned} \mathcal{L}_{\text{inv}}^j(\Theta) = w_f \, \frac{1}{N_f^j} \sum_{(\mathbf{x}_f, t_f, g) \in \mathcal{I}_f^j} \Big[{}_0^{\mathcal{C}}\partial_t^\alpha \tilde{v}(\mathbf{x}_f, t_f) + \mathcal{N}(\tilde{v}(\mathbf{x}_f, t_f); \lambda) - g \Big]^2 \\ + w_T \, \frac{1}{N_T} \sum_{(\mathbf{x}_T^i, t_T^i, \psi) \in \mathcal{I}_T} \Big[\tilde{v}(\mathbf{x}_T^i, T) - \psi(\mathbf{x}_T^i) \Big]^2. \end{aligned} \tag{2.32}$$

Because $\mathcal{I}_f^j$ only contains points with $t_f \le t_{j-\theta}$, the second (terminal) term vanishes for all $j < K$ and only "turns on" at the final stage $j = K$. All subsequent steps, including the implementation of residual-based adaptive refinement on $\mathcal{I}_f^j$ are performed identically to the forward Alikhanov–XfPINN formulation.

The proposed framework is implemented in TensorFlow [42] to efficiently evaluate the residual terms and optimize the trainable parameters. TensorFlow provides automatic differentiation to compute spatial derivatives and gradient information, optimized CPU/GPU kernels to efficiently train models, direct GPU support, and a flexible Python interface to rapidly implement and test models. The broader TensorFlow ecosystem, which includes visualization and monitoring tools such as TensorBoard, facilitates systematic model development.

We employ the Adam algorithm [43] for optimization, with suitable hyperparameter adjustments for the graded-mesh Alikhanov–XfPINN setting. These adjustments are made to improve convergence speed while maintaining numerical accuracy. The complete training strategy is summarized below. Algorithm 2.1 summarizes the computational workflow of the Alikhanov–XfPINN for NfPDEs. First, we introduce the temporal nodes t_k, where $k = 0, \dots, K$. Then, we construct the initial dataset $\mathcal{I}$ by randomly selecting spatio-temporal samples in Ω_T. In the stage-wise Alikhanov–XfPINN procedure, the active training set is updated at each stage according to the current temporal window (see line 9 of Algorithm 2.1. Once the collocation samples are fixed, we use the accelerated approximation to compute the fractional derivative and evaluate the total loss. Then, we update the trainable parameters using gradient-based optimization.

In Algorithm 2.1, the trainable variables are only initialized during the first stage ($r = 1$). For all subsequent stages, training begins with the optimal parameters obtained in the previous stage. This implementation uses TensorFlow's automatic differentiation in Python and adopts Xavier initialization for the network weights to ensure stable signal propagation. The neural output is embedded into the hard-constrained form (2.28), producing the complete approximation $v_\Theta(x, y, t)$. At each collocation point corresponding to t_k, the network values at the preceding time levels $\{t_j\}_{j=0}^{k}$ are evaluated simultaneously in vectorized form. This allows the fractional history term to be computed efficiently.

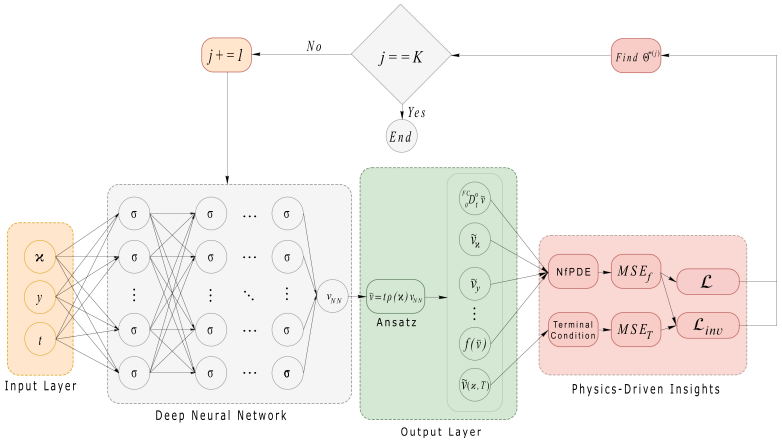


Fig. 2: Schematic diagram of the XfPINN architecture.

2.6. Time-Marching Configuration: Alikhanov-fPINN-M. We now present the time-marching variant of the Alikhanov-fPINN framework, which we denote as Alikhanov-fPINN-M. This configuration sequentially enforces the SOE-accelerated second-order time-discrete residual along the graded temporal mesh. At the shifted time level $t_{k-\theta}$, the residual is defined as

$$\begin{aligned}\mathcal{R}_\Theta^{k-\theta}(\mathbf{x}) := \mathbf{a}^{(0,k)} \nabla_\tau \tilde{v}_\Theta^k(\mathbf{x}) + \sum_{l=1}^{N_{\exp}} \nu^l \, \mathrm{e}^{-s^l \theta \tau_k} \, \mathcal{V}_\Theta^l(t_k; \mathbf{x}) \\ + \mathcal{N}\big(\tilde{v}_\Theta(\mathbf{x}, t_{k-\theta}); \lambda\big) - g(\mathbf{x}, t_{k-\theta}), \qquad 1 \le k \le K.\end{aligned} \tag{2.33}$$

Here,

$$\nabla_\tau \tilde{v}_\Theta^k(\mathbf{x}) := \tilde{v}_\Theta^k(\mathbf{x}) - \tilde{v}^{k-1}(\mathbf{x}),$$

and the SOE history variables $\mathcal{V}_\Theta^l(t_k; \mathbf{x})$ are updated recursively according to (2.20). The Alikhanov-fPINN-M method imposes the discretization-consistent residual (2.33) one time level at a time, so that the contribution of the temporal discretization can

Algorithm 2.1 Alikhanov–XfPINNs on nonuniform meshes for simulation and estimation in NfPDEs

Require: A set of training dataset $\mathcal{I}$, domain bounds $\mathbf{lb}, \mathbf{ub}$, network layer sizes $\{d_0, d_1, \ldots, d_L\}$, fractional order α, subdivisions K, shift parameter θ, λ_1, λ_2, stage-wise max. iterations m_{stage} and threshold ε.

Ensure: Trained parameters Θ^* and approximate solution $\tilde{v}(\mathbf{x}, t)$.

Initialize weights and biases:
for $l = 0$ to $L - 1$ **do**
  Sample weight matrix $W^{(l)} \sim \mathcal{N}(0, \sigma_l^2)$ with $\sigma_l = \sqrt{2/(d_l + d_{l+1})}$
  Set bias vector $b^{(l)} \leftarrow \mathbf{0}$
end for
$\Theta^{*(0)} \leftarrow$ initial parameters
for $j = 2$ to K **do**
  $t_{j-\theta} \leftarrow t_{j-1} + (1 - \theta)\,\tau_j$
  $\mathcal{I}_f^j \leftarrow \{(\mathbf{x}^i, t_f^i, g(\mathbf{x}^i, t^i)) \in \mathcal{I} : t_{1-\theta} \le t_f^i \le t_{j-\theta}\}$
  $N_f^j \leftarrow |\mathcal{I}_f^j|$
  $\Theta^{(j)} \leftarrow \Theta^{*(j-1)}$
  for $k = 1$ to m_{stage} **do**
    Compute $\tilde{v}(\mathbf{x}, t) = t\,\rho(\mathbf{x})\,v_{NN}(\mathbf{x}, t; \Theta^{(j)})$ and ${}_0^{\mathcal{C}}\partial_t^\alpha \tilde{v}(\mathbf{x}, t)$ via Alikhanov formula
    Compute stage loss for the forward and inverse problems:

$$\mathcal{L}^j(\Theta^{(j)}) = \frac{1}{N_f^j} \sum_{(\mathbf{x}_f, t_f, g) \in \mathcal{I}_f^j} \sum_{i=1}^{N_f} \left[\partial_t^\alpha \tilde{v}(\mathbf{x}_f, t_f) + \mathcal{N}(\tilde{v}(\mathbf{x}_f, t_f); \lambda) - g(\mathbf{x}_f,\, t_f) \right]^2$$

$$\mathcal{L}_{\text{inv}}^j(\Theta) = w_f\, \frac{1}{N_f^j} \sum_{(\mathbf{x}_f, t_f, g) \in \mathcal{I}_f^j} \left[{}_0^{\mathcal{C}}\partial_t^\alpha \tilde{v}(\mathbf{x}_f, t_f) + \mathcal{N}(\tilde{v}(\mathbf{x}_f, t_f); \lambda) - g(\mathbf{x}_f,\, t_f) \right]^2 + w_T\, \frac{1}{N_T} \sum_{(\mathbf{x}_T^i, t_T^i, \psi) \in \mathcal{I}_T} \left[\tilde{v}(\mathbf{x}_T^i, T) - \psi(\mathbf{x}_T^i) \right]^2.$$

    Update:

$$\Theta^{(j)} \leftarrow \Theta^{(j)} - \eta\, \nabla_{\Theta^{(j)}}\, \mathcal{L}^{(j)}(\Theta^{(j)}) \quad \text{(via tf.keras.optimizers.Adam)}$$

    if $\mathcal{L}^{(j)}(\Theta^{(j)}) < \varepsilon$ **then**
      break
    end if
  end for
  Store $\Theta^{*(j)} \leftarrow \Theta^{(j)}$
end for
return $\Theta^* = \Theta^{*(K-1)}$, $\tilde{v}(\mathbf{x}, t) = t\,\rho(\mathbf{x})\,v_{NN}(\mathbf{x}, t; \Theta^*)$

be examined transparently and reproducibly. In this setting, previously computed solution snapshots are kept fixed and used as stored history when advancing to the next shifted time level.

Let $0 = t_0 < t_1 < \cdots < t_K = T$ be the graded temporal mesh introduced in Section 2.2.1. For each fixed $\mathbf{x} \in \Omega$, let $\tilde{v}^k(\mathbf{x})$ denote the frozen approximation at time t_k produced by the marching procedure. For a candidate parameter vector Θ, we write

$$\tilde{v}_\Theta^k(\mathbf{x}) := \tilde{v}_\Theta(\mathbf{x}, t_k), \tag{2.34}$$

where $\tilde{v}_\Theta$ is the hard-constrained trial function (2.28). The computation starts from the initial snapshot

$$\tilde{v}^0(\mathbf{x}) := 0, \qquad \mathbf{x} \in \Omega, \tag{2.35}$$

Algorithm 2.2 Alikhanov-fPINN-M on nonuniform meshes for time-marching enforcement

Require: Collocation points $\{\boldsymbol{x}_f^i\}_{i=1}^{N_x} \subset \Omega$, final time T, number of time levels K, grading parameter γ_g, offset parameter θ, model data λ_1, λ_2, λ_3, g, physical parameters $(\alpha, \varepsilon, \gamma)$, SOE nodes and weights $\{(s^l, \nu^l)\}_{l=1}^{N_{\exp}}$, discretization coefficients $\{\mathbf{a}^{(0,k)}, a^{(n,l)}, b^{(n,l)}\}$, maximum inner iterations m_{step}, stopping tolerance ε_{tol}

Ensure: Frozen snapshots $\{\tilde{v}^k\}_{k=1}^K$ and stored history states $\{\mathcal{V}^l(t_k;\cdot)\}_{k=1}^K$

1: Build the graded temporal mesh

$$t_k = T\Big(\frac{k}{K}\Big)^{\gamma_g}, \qquad \tau_k = t_k - t_{k-1}, \qquad t_{k-\theta} = \theta t_{k-1} + (1-\theta)t_k.$$

2: Initialize $\tilde{v}^0(\boldsymbol{x})$ from the initial condition and set $\mathcal{V}^l(t_0;\boldsymbol{x}) \leftarrow 0$ for $1 \le l \le N_{\exp}$.
3: Initialize the network parameters Θ^0.
4: **for** $k = 1$ to K **do**
5: Warm start the trainable parameters by setting $\Theta \leftarrow \Theta^{k-1}$.
6: **for** $m = 1$ to m_{step} **do**
7: Evaluate the current candidate

$$\tilde{v}_\Theta^k(\boldsymbol{x}) = \tilde{v}_\Theta(\boldsymbol{x}, t_k), \qquad \tilde{v}_\Theta^{k-\theta} = \theta\tilde{v}^{k-1} + (1-\theta)\tilde{v}_\Theta^k.$$

8: Compute

$$v_\Theta^{k-\theta} = \mathcal{N}\big[\tilde{v}_\Theta^{k-\theta};\lambda\big] - g\big(\tilde{v}_\Theta^{k-\theta}\big), \qquad \nabla_\tau \tilde{v}_\Theta^k = \tilde{v}_\Theta^k - \tilde{v}^{k-1}.$$

9: Update the candidate history variables $\mathcal{V}_\Theta^l(t_k;\cdot)$ using (2.20), the frozen snapshots $\{\tilde{v}^j\}_{j=0}^{k-1}$, and the current increment $\nabla_\tau \tilde{v}_\Theta^k$.
10: Compute the time-marching residual loss

$$\mathcal{L}_M^k(\Theta) = \frac{1}{N_x}\sum_{i=1}^{N_x}\big|\mathcal{R}_\Theta^{k-\theta}(\boldsymbol{x}_f^i)\big|^2,$$

where $\mathcal{R}_\Theta^{k-\theta}$ is evaluated from (2.33).
11: Update Θ by Adam using $\mathcal{L}_M^k(\Theta)$.
12: **if** $\mathcal{L}_M^k(\Theta) < \varepsilon_{\text{tol}}$ **then**
13: **break**
14: **end if**
15: **end for**
16: Freeze the trained state:

$$\Theta^k \leftarrow \Theta, \qquad \tilde{v}^k \leftarrow \tilde{v}_{\Theta^k}(\cdot, t_k), \qquad \mathcal{V}^l(t_k;\cdot) \leftarrow \mathcal{V}_{\Theta^k}^l(t_k;\cdot), \quad 1 \le l \le N_{\exp}.$$

17: **end for**
18: **return** $\{\tilde{v}^k\}_{k=1}^K$ and $\{\mathcal{V}^l(t_k;\cdot)\}_{k=1}^K$.

which the ansatz (2.28) satisfies exactly. The history variables are initialized by

$$\mathcal{V}^l(t_0;\mathbf{x}) = 0, \qquad 1 \le l \le N_q.$$

Assume that the snapshots $\{\tilde{v}^n(\mathbf{x})\}_{n=0}^{k-1}$ and the corresponding history states have already been computed and stored. At the next time level, the unknown snapshot is represented by the neural ansatz

$$\tilde{v}_\Theta^k(\mathbf{x}) = \tilde{v}_\Theta(\mathbf{x}, t_k) = t_k\rho(\mathbf{x})v_{NN}(\mathbf{x}, t_k;\Theta), \qquad \mathbf{x} \in \Omega,$$

while all earlier snapshots remain fixed. The shifted state used in the nonlinear and spatial terms is defined by

$$\tilde{v}_\Theta^{k-\theta}(\mathbf{x}) := \theta\,\tilde{v}^{k-1}(\mathbf{x}) + (1-\theta)\tilde{v}_\Theta^k(\mathbf{x}), \qquad \mathbf{x} \in \Omega.$$

The corresponding time increment is

$$\nabla_\tau \tilde{v}^k_\Theta(\mathbf{x}) := \tilde{v}^k_\Theta(\mathbf{x}) - \tilde{v}^{k-1}(\mathbf{x}).$$

The nonlocal memory contribution in (2.33) is evaluated through the SOE variables, which are updated recursively. For $k \geq 1$ and $1 \leq l \leq N_q$, we set

$$\mathcal{V}^l_\Theta(t_k;\mathbf{x}) = e^{-s^l\tau_k}\mathcal{V}^l(t_{k-1};\mathbf{x}) + a^{(k,l)}\nabla_\tau \tilde{v}^k_\Theta(\mathbf{x}) + b^{(k,l)}\big(\rho_k \nabla_\tau \tilde{v}^{k+1}_\Theta(\mathbf{x}) - \nabla_\tau \tilde{v}^k_\Theta(\mathbf{x})\big),$$

with the initial value $\mathcal{V}^l(t_0;\mathbf{x}) = 0$. Using these quantities, the residual at the shifted level $t_{k-\theta}$ is evaluated through (2.33).

Let the set of interior spatial collocation points $\{\mathbf{x}^i_f\}_{i=1}^{N_x} \subset \Omega$ be the fixed points used at every marching step. These points are generated once before training and remain unchanged throughout the sequential procedure. At step k, only the temporal evaluation level changes from one shifted time to the next. The stepwise loss functional is defined by

$$\mathcal{L}^k_M(\Theta) := \frac{1}{N_x}\sum_{i=1}^{N_x}\big|\mathcal{R}^{k-\theta}_\Theta(\mathbf{x}^i_f)\big|^2, \qquad 1 \leq k \leq K, \tag{2.36}$$

and the trainable parameters at the k-th step are obtained from

$$\Theta^k := \arg\min_\Theta \mathcal{L}^k_M(\Theta). \tag{2.37}$$

After the minimization is completed, the newly computed snapshot is frozen as

$$\tilde{v}^k(\mathbf{x}) := \tilde{v}^k_{\Theta^k}(\mathbf{x}), \qquad \mathbf{x} \in \Omega,$$

and the memory variables are stored at the optimized parameters:

$$\mathcal{V}^l(t_k;\mathbf{x}) := \mathcal{V}^l_{\Theta^k}(t_k;\mathbf{x}), \qquad 1 \leq l \leq N_q, \quad \mathbf{x} \in \Omega.$$

These stored quantities are then used as the memory state for the subsequent marching step. In practice, the optimization at each new step is warm-started from the parameters obtained at the preceding step, which improves training stability on graded meshes and reduces fluctuations in the optimization error.

The Alikhanov-fPINN-M procedure sequentially advances from $k = 1$ to K, producing the discrete-time surrogate $\{\tilde{v}^k\}_{k=1}^K$. Since the residual is imposed at each shifted level and the SOE memory variables are updated recursively, this configuration remains consistent with the accelerated second-order temporal discretization. In this work, the Alikhanov-fPINN-M method is used primarily as an auxiliary configuration to verify temporal convergence with respect to the maximum time step τ. By isolating the residual enforcement at each time level and treating the history terms with stored recursive variables, the method provides a controlled environment for evaluating the accuracy of the proposed time discretization.

3. Numerical Results. This section presents the essential implementation details of the proposed numerical procedure. We apply our proposed Alikhanov-PINNs scheme to different test examples. To evaluate the efficiency and accuracy of the proposed numerical algorithm for various NfPDEs, we conduct forward and inverse problem tests, including cases involving the nonlinear subdiffusion equation, the generalized viscous Burgers equation, and cases without known exact solutions. Through

analysis of network architecture, training data size, and adaptive variants of activation functions, we identify optimal Alikhanov-PINNs parameters. All computations are performed in Python on a workstation equipped with an Intel® Core™ i7-10870H CPU at 2.21 GHz and an NVIDIA GeForce GTX 1660 Ti GPU. Our implementation is based on TensorFlow 2.18, but can easily be adapted to other deep learning frameworks such as PyTorch. We compute the gradient with respect to the network parameters and inputs using the TensorFlow's built-in auto-differentiation module, and use the Adam optimizer to minimize the loss function.

To quantify the accuracy and temporal convergence of the Alikhanov–PINN on nonuniform meshes, we use the maximum-norm error $E_\infty(K) = \max_k \|U^k - u^k\|_\infty$ and L_2-error $E_2(K) = \frac{\|\tilde{v}(\cdot,t)-v(\cdot,t)\|_2}{\|v(\cdot,t)\|_2}$. The temporal convergence rate is then computed by the following formula:

$$\text{rate}_j \approx \log_2\Big(\frac{E_j(K)}{E_j(2K)}\Big), \tag{3.1}$$

where $j = 2$ or $j = \infty$.

3.1. Forward Problem:.

EXAMPLE 3.1. *Nonlinear time-fractional subdiffusion equation (NTFSDE) with smooth exact solution*
In this example, we consider

$$\begin{aligned} {}_0^C\partial_t^\alpha v + \mathcal{N}[v;\lambda] &= g(\mathbf{x},t), \\ v(\mathbf{x},0) &= 0, \quad \mathbf{x}\in\Omega, \\ v(\mathbf{x},t) &= 0, \quad \mathbf{x}\in\partial\Omega, \end{aligned}$$

where $\mathcal{N}[v;\lambda] = \lambda_1(v^3 - v) - \lambda_2(v_{xx}) + g(\cdot,t)$ *with* (λ_1,λ_2) *being unknown parameters which is the case of 1D problem setup. To extend this problem to 2D setup and 3D setup, we assume*

$$\begin{aligned} \mathcal{N}[v;\lambda] &= \lambda_1(v^3 - v) - \lambda_2(v_{xx} + v_{yy}) - g(x,y,t), \\ \mathcal{N}[v;\lambda] &= \lambda_1(v^3 - v) - \lambda_2(v_{xx} + v_{yy} + v_{zz}) - g(x,y,z,t) \end{aligned}$$

with the smooth exact solutions being

$$\begin{aligned} v(\cdot,t) &= t^{2+\alpha}\sin(\pi x), \quad \Omega = [0,1], \\ v(\cdot,t) &= t^{2+\alpha}\sin(\pi x)\sin(\pi y), \quad \Omega = [0,1]^2, \\ v(\cdot,t) &= t^{2+\alpha}\sin(\pi x)\sin(\pi y)\sin(\pi z), \quad \Omega = [0,1]^3, \end{aligned}$$

and the corresponding body force g *is calculated based on these exact solutions.*

The performance of the Alikhanov-PINNs is evaluated using the 1D NTFSDE problem to systematically examine the effects of activation functions (adaptive variant), neural network architecture and the size of the training data sample. The solution is represented as $\tilde{v} = t\mathbf{x}(1-\mathbf{x})v_{NN}(\mathbf{x},t,\Theta)$, $\tilde{v}$ is the Alikhanov-PINNs solution and IBC are imposed via hard constraints. The training data are sampled from a nonuniform mesh ($\gamma = 1$).

To determine the most effective activation function, we compare six adaptive variants in Table 2 for $\alpha = 0.5$. The Swish($n\mathbf{a}\mathbf{x}$) function yields the lowest $\mathcal{E}_\infty$-error and relative $\mathcal{E}_2$-error outperforming its fixed counterpart Swish($\mathbf{x}$) by capturing the

nonlinear dynamics more accurately. Figure 3 confirms that Swish($na\mathbf{x}$) leads to a faster loss decay and better agreement with the analytical solution, highlighting the importance of adaptive activations in improving the training efficiency.

Figures 4a and 4b analyze the influence of the scale factor n on training dynamics. A higher n value accelerates the convergence due to more aggressive parameter updates; however, overly large values may cause instability. Convergence of the product na during training indicates stabilization of the adaptive training rate. Based on these results, Swish(nax)($n = 3$) is selected as the optimal activation function.". Table 3 summarizes the final network configuration and associated training times for the 1D-3D problems. Table 4 reports the $\mathcal{E}_\infty(K)$-errors and the corresponding temporal convergence rates obtained by the Alikhanov-fPINN-M for different values of α. The results show that, when the grading parameter is chosen as $\gamma = 2/\alpha$, the Alikhanov-fPINN-M recovers the expected second-order accuracy in the temporal direction. Table 5 reports the adaptive-training accuracy for the NTFSDE with smooth exact solution in Example 3.1. The results show that Alikhanov-XfPINNs consistently produce smaller $\mathcal{E}_\infty(K)$ and $\mathcal{E}_2(K)$ errors than Alikhanov-PINNs for all tested fractional orders, which confirms that the adaptive XfPINN formulation improves the approximation accuracy under the same training setting. Figure 5 displays the exact solution, the corresponding numerical approximation, and the absolute error obtained from the proposed scheme.

Table 2: Maximum-norm error $\mathcal{E}_\infty(K)$ and relative L^2-error $\mathcal{E}_2(K)$ of the Alikhanov–XfPINNs with adaptive variant ($\alpha = 0.5$, $K = 16$).

Adaptive Activation Function	$\mathcal{E}_2(K)$	$\mathcal{E}_\infty(K)$
Sigmoid($\mathbf{x}$)	$3.975e-04$	$3.298e-04$
Swish($\mathbf{x}$)	$2.685e-04$	$1.678e-04$
ReLU($\mathbf{x}$)	$9.479e-04$	$9.893e-04$
tanh($\mathbf{x}$)	$3.478e-04$	$2.523e-04$
SeLU($\mathbf{x}$)	$8.577e-04$	$1.178e-03$
$\mathbf{x}$ tanh($\mathbf{x}$)	$3.012e-04$	$2.223e-04$

Table 3: Overview of Alikhanov–XfPINNs architecture details and the quantity of training samples utilized.

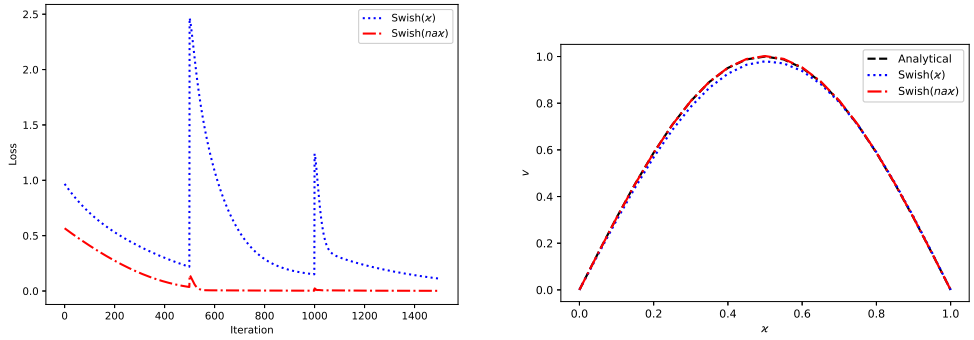
K	K_{total}	L	N_i	P
8	288	2	14	281
16	576	3	15	556
32	1152	4	18	1117

EXAMPLE 3.2. *Generalized Burgers' equation with nonsmooth exact solution*
To verify the effectiveness of our nonuniform scheme, we consider the time-fractional generalized viscous Burgers' problem, for which the exact solution exhibits an initial singularity at $t = 0$. We consider

$$ {}_0^C\partial_t^\alpha v + \mathcal{N}[v;\lambda] = g(x,t), \tag{3.2}$$

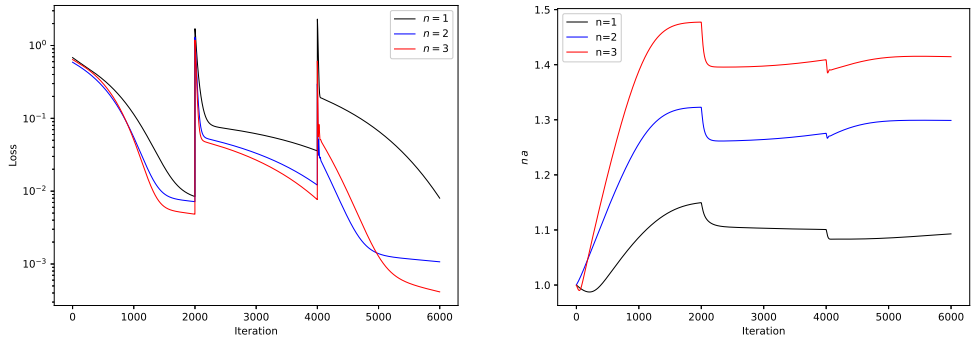
where $\mathcal{N}[v;\lambda] = \lambda_1 v^p v_x + \lambda_2 v_{xx}$, $p > 0$ and the exact solution is given by $v = (1 + \omega_{1+\alpha}(t))\sin(\pi x)\sin(\pi y)$, $(\mathbf{x}, t) \in [0,1]\times[0,1]$, for the initial condition $v_0(x) = \sin(\pi x)$.

Due to the lack of first order time continuity, the exact solution exhibit singular behavior as $t \to 0$. To address this we utilize the same fully connected neural



(a) (b)

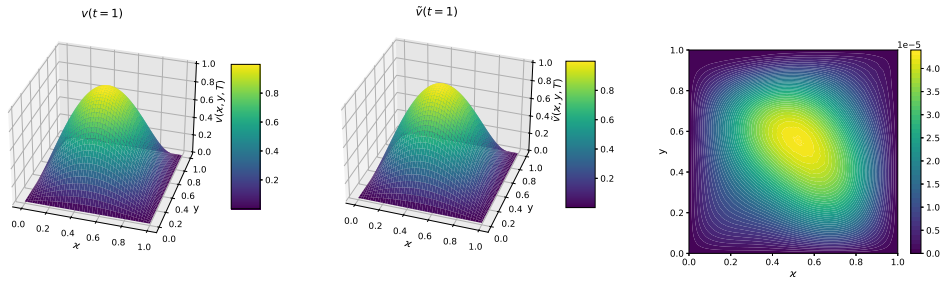
Fig. 3: Comparison of loss evolution and Alikhanov–XfPINNs approximation results for Swish(x) and Swish(nax) ($\alpha = 0.5$): (a) training loss progression; (b) exact and predicted field profiles.



(a) (b)

Fig. 4: Evolution of the training loss and na across iterations with $\alpha = 0.5$ for adaptive activation functions with various n : (a) loss trajectory; (b) na.

network(FCNN) structure as in Example 3.1, embedding the initial and boundary conditions as hard constraint. The network approximation is expressed as $\tilde{v}(x,t) = tx(1-x)v_{\mathrm{NN}}(x,t,\Theta)$. To resolve the initial singularity, we examine the effect of graded temporal meshes with refinement parameters $\gamma = 1$(uniform), $\gamma = \frac{2}{\alpha}, \frac{3-\alpha}{\alpha}$. Figure 6 presents the absolute error profiles when $\alpha = 0.9$. Tables 6, 7 and 8 present the $\mathcal{E}_\infty$-error obtained from the Alikhanov-PINNs and Alikhanov–XfPINNs trained on a different grading parameter. As can be seen from the tables, the optimal grading parameter that offers the best convergence rate is clearly $\gamma = \frac{2}{\alpha}$. Table 9 reports the adaptive-training accuracy of Alikhanov-PINNs and Alikhanov–XfPINNs for the generalized viscous Burgers equation with nonsmooth exact solution. The results show that the adaptive Alikhanov-XfPINNs consistently reduce both $\mathcal{E}_\infty(K)$ and $\mathcal{E}_2(K)$ for all tested fractional orders, confirming the effectiveness of the adaptive XfPINN



(a) (b) (c)

Fig. 5: Comparison of 2D-NTFSDe with smooth exact solution at $\alpha = 0.9$ and $K = 8$: (a) exact solution v, (b) Alikhanov–XfPINNs approximation $\tilde{v}$, and (c) absolute error $|v - \tilde{v}|$.

Table 4: $\mathcal{E}_\infty(K)$-errors and temporal order of convergence (TOC) rates of the Alikhanov–fPINN-M time-marching formulation for the 2D-NTFSDE with smooth exact solution in Example 3.1.

α	K	Alikhanov-fPINN-M	
		$\mathcal{E}_\infty(K)$	$rate_\tau$
0.30	2^3	$2.582e-04$	$--$
	2^4	$6.730e-05$	1.939
	2^5	$1.715e-05$	1.972
TOC			2.0
0.60	2^3	$4.651e-04$	$--$
	2^4	$1.228e-04$	1.921
	2^5	$3.147e-05$	1.964
TOC			2.0
0.90	2^3	$6.521e-04$	$--$
	2^4	$1.730e-04$	1.914
	2^5	$4.399e-05$	1.975
TOC			2.0

formulation for this weakly singular benchmark.

EXAMPLE 3.3. *To examine a more realistic setting, we consider a problem without a closed-form exact solution, prescribed only by the initial condition and with no source term*

$$ {}_0^C\partial_t^\alpha v(\cdot,t) + \mathcal{N}[v;\lambda] = 0, \tag{3.3}$$

where $\mathcal{N}[v;\lambda] = \lambda \cdot \nabla v + v(v-1)(\rho - v)$ *with* $\lambda_1 = \lambda_2 = 1, \rho > 1$ *and* $\mathbf{x} \in \Omega, t \in [0,T]$.

The exact solution to (3.3) *remains unknown. However, for the special case* $\alpha = 1$, *Table 10 provides analytical expressions corresponding to the classical integer-order Fisher–Nagumo (FN) equation in one and two dimensions. These expressions are used to approximate the initial and boundary conditions (IBCs).*

To solve the time-fractional Fisher–Nagumo (TFFN) problem with $\rho = 1$, *and* $\phi = \pi/2$, *we use the Alikhanov–XfPINNs framework incorporating soft-constrained IBCs. The predicted solution is denoted by* $\tilde{v}(\mathbf{x},t) = v_{NN}(\mathbf{x},t;\Theta)$, *with training data sampled from random boundary points, initial conditions, and a uniform spatial grid. Table 11*

Table 5: Maximum-norm error $\mathcal{E}_\infty(K)$ and relative L^2 error $\mathcal{E}_2(K)$ obtained by the adaptive-training Alikhanov-PINNs and Alikhanov-XfPINNs for the NTFSDE with smooth exact solution in Example 3.1.

α	Alikhanov-PINNs		Alikhanov-XfPINNs	
	$\mathcal{E}_\infty(K)$	$\mathcal{E}_2(K)$	$\mathcal{E}_\infty(K)$	$\mathcal{E}_2(K)$
0.30	$3.435e-03$	$4.536e-03$	$1.396e-03$	$3.589e-03$
0.60	$2.185e-03$	$3.874e-03$	$1.057e-03$	$1.247e-03$
0.90	$1.864e-03$	$3.318e-03$	$9.735e-04$	$1.196e-03$

Table 6: $\mathcal{E}_\infty$–errors and temporal convergence rates of Alikhanov-fPINN-M for the generalized viscous Burgers equation with nonsmooth exact solution ($\alpha = 0.3$).

α	γ	K	$p=2$		$p=4$	
			$\mathcal{E}_\infty(K)$	$rate_\tau$	$\mathcal{E}_\infty(K)$	$rate_\tau$
0.30	1	2^3	$8.122e-03$	–	$8.095e-03$	–
		2^4	$7.472e-03$	0.120	$6.103e-03$	0.409
		2^5	$6.670e-03$	0.164	$4.350e-03$	0.489
TOC				–		–
	$2/\alpha$	2^3	$2.812e-03$	–	$2.700e-03$	–
		2^4	$8.315e-04$	1.758	$7.514e-04$	1.845
		2^5	$2.335e-04$	1.832	$2.030e-04$	1.888
TOC				2.0		2.0
	$(3-\alpha)/\alpha$	2^3	$1.845e-03$	–	$2.861e-03$	–
		2^4	$5.094e-04$	1.857	$8.544e-04$	1.744
		2^5	$1.362e-04$	1.903	$2.315e-04$	1.884
TOC				2.0		2.0

summarizes the optimal network architecture and its associated training epoch.

Given the absence of an analytical solution for the TFFN model, we train the Alikhanov–XfPINN at fractional orders $\alpha \in \{0.30, 0.60, 0.90\}$ and assess its ability to recover the integer-order FN solution ($\alpha = 1$). As illustrated in Figure 7, the neural approximation increasingly aligns with the known solution as $\alpha \to 1$, validating the effectiveness of Alikhanov–XfPINN for inverse problems where the exact solution is unknown.

3.2. Inverse Problem.

EXAMPLE 3.4. *Here, we consider the inverse problem of NfPDEs with constant coefficients,*

$$ {}_0^C\partial_t^\alpha v + \mathcal{N}[v;\lambda] = g(\mathbf{x},t), \tag{3.4}$$

with $\mathcal{N}[v;\lambda] = \lambda_1(v_{xx} + v_{yy}) + \lambda_2 f(v)$ *and*

$$v(\mathbf{x},0) = v_1(\mathbf{x}), \mathbf{x} \in \Omega,\ v(\mathbf{x},t) = v_2(\mathbf{x}), \mathbf{x}, \mathbf{t} \in \partial\Omega, t \in (0,T), v(\mathbf{x},T) = v_3(\mathbf{x},t), \mathbf{x} \in \Omega,$$

where $\alpha, \lambda_1, \lambda_2$ being the unknown parameters. Given the reaction term $f(v)$, source term $g(\mathbf{x},t)$, initial–boundary conditions, and partial solution data $v_3(\mathbf{x},t)$ at the final time – along with complete solution fields $v_1(\mathbf{x},t)$ and $v_2(\mathbf{x},t)$, the Alikhanov–PINN framework is trained to recover the fractional order α, diffusion coefficient λ_1, reaction coefficient λ_2, and the full solution of the NfPDE.

For the inverse problem, the training dataset is extended by incorporating 30 randomly sampled terminal time solutions into the original forward dataset. Table 12

Table 7: $\mathcal{E}_\infty$–errors and temporal convergence rates of Alikhanov-fPINN-M for the generalized viscous Burgers equation with nonsmooth exact solution ($\alpha = 0.6$).

α	γ	K	$p=2$		$p=4$	
			$\mathcal{E}_\infty(K)$	$rate_\tau$	$\mathcal{E}_\infty(K)$	$rate_\tau$
0.60	1	2^3	$4.439e-03$	–	$4.558e-03$	–
		2^4	$3.779e-03$	0.232	$3.635e-03$	0.326
		2^5	$2.825e-03$	0.419	$2.593e-03$	0.487
TOC				–		–
	$2/\alpha$	2^3	$1.295e-03$	–	$1.437e-03$	–
		2^4	$3.811e-04$	1.765	$5.037e-04$	1.512
		2^5	$1.042e-04$	1.870	$1.357e-04$	1.892
TOC				2.0		2.0
	$(3-\alpha)/\alpha$	2^3	$1.047e-03$	–	$1.226e-03$	–
		2^4	$2.970e-04$	1.818	$3.463e-04$	1.824
		2^5	$7.801e-05$	1.929	$9.405e-05$	1.881
TOC				2.0		2.0

Table 8: $\mathcal{E}_\infty$–errors and temporal convergence rates of Alikhanov-fPINN-M for the generalized Burgers equation with nonsmooth exact solution ($\alpha = 0.9$).

α	γ	K	$p=2$		$p=4$	
			$\mathcal{E}_\infty(K)$	$rate_\tau$	$\mathcal{E}_\infty(K)$	$rate_\tau$
0.90	1	2^3	$2.069e-03$	–	$1.895e-03$	–
		2^4	$1.573e-03$	0.395	$1.475e-03$	0.361
		2^5	$1.116e-03$	0.495	$9.390e-04$	0.652
TOC				–		–
	$2/\alpha$	2^3	$5.773e-04$	–	$6.631e-04$	–
		2^4	$1.516e-04$	1.929	$2.302e-04$	1.526
		2^5	$3.898e-05$	1.959	$6.019e-05$	1.935
TOC				2.0		2.0
	$(3-\alpha)/\alpha$	2^3	$5.210e-04$	–	$5.934e-04$	–
		2^4	$1.345e-04$	1.954	$1.822e-04$	1.703
		2^5	$3.418e-05$	1.976	$4.863e-05$	1.906
TOC				2.0		2.0

Table 9: Maximum-norm error $\mathcal{E}_\infty(K)$ and relative L^2 error $\mathcal{E}_2(K)$ obtained by the adaptive-training Alikhanov-PINNs and Alikhanov-XfPINNs for the generalized viscous Burgers equation with nonsmooth exact solution.

α	Alikhanov-PINNs		Alikhanov-XfPINNs	
	$\mathcal{E}_\infty(K)$	$\mathcal{E}_2(K)$	$\mathcal{E}_\infty(K)$	$\mathcal{E}_2(K)$
0.30	$8.024e-03$	$8.474e-03$	$7.575e-03$	$7.834e-03$
0.60	$6.825e-03$	$7.284e-03$	$5.927e-03$	$6.135e-03$
0.90	$6.015e-03$	$6.275e-03$	$5.285e-04$	$5.593e-03$

Table 10: Integer-order FN ($\alpha = 1$): exact solutions $v(\mathbf{x}, t)$ in $1D$ and $2D$.

Domain	$v(x,t)$
$\Omega = [0,1]$	$1/2 + 1/2\ \tanh\Big(\frac{1}{\sqrt{8}}[x - t\frac{2\rho-1}{\sqrt{2}}]\Big)$
$\Omega = [0,1]^2$	$1/2 + 1/2\ \tanh\Big(\frac{1}{\sqrt{8}}[x\sin\varphi + y\cos\varphi] - t\frac{2\rho-1}{\sqrt{2}}\Big)$

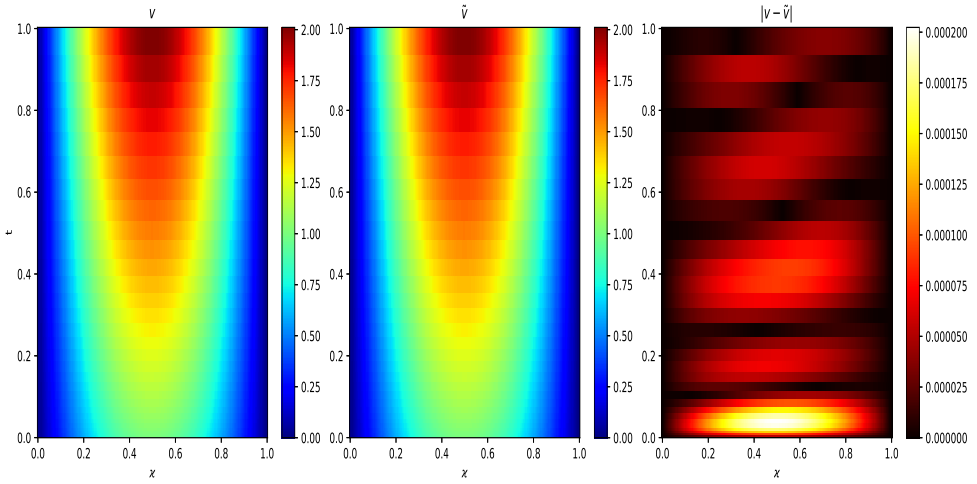


Fig. 6: Comparison of exact solution v, Alikhanov–XfPINNs prediction $\tilde{v}$ and absolute error $|v - \tilde{v}|$ for time fractional generalized Burgers' problem.

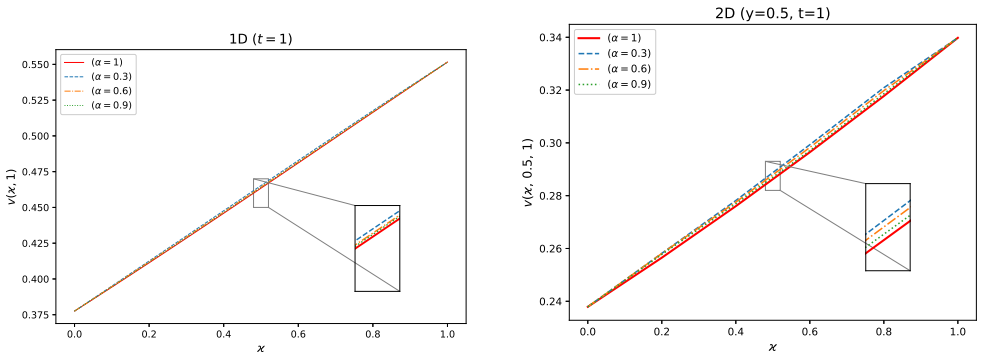
Table 11: Network architectures and training parameters for 1D-TFFN and 2D-TFFN models.

Equation	Depth	Width	Optimizer	Learning rate	Epoch
1D-TFFN	1	30	Adam	$1e-04$	1500
2D-TFFN	3	14	Adam	$1e-04$	1000

lists the optimal neural network configurations such as depth, width, learning rate, IBC enforcement and unknown parameters along with the training durations for all test cases. The stopping criteria are defined using the maximum epoch count m_{stage}*, and a tolerance threshold on the relative* $\mathcal{E}_2$*-error* ϵ*.*

The influence of the mesh grading parameter γ*, is explored for the time-fractional reaction-diffusion equation (TFRD) inverse problem. Figure 8 shows the evolution of the parameters* $(\alpha, \lambda_1, \lambda_2)$ *on different graded meshes* $(\gamma = 1, \frac{2}{\alpha}, \frac{3-\alpha}{\alpha})$*. It is evident that nonuniform meshes* $(\gamma = 2/\alpha, (3-\alpha)/\alpha)$ *yield parameter estimates that converge more precisely to the true values than the uniform mesh* $(\gamma = 1)$*, highlighting the superior accuracy of nonuniform discretization in inverse TFRD problems. Lastly, Table 13 compares the ground truth and inferred parameters for 2D TFRD problems. The results confirm that all parameters are accurately estimated and that both error metrics diminish when using nonuniform meshes in place of uniform mesh.*

Example 3.5. *Here, we investigate the 2D time-fractional Allen–Cahn (TFAC) equation for simultaneous parameter inference and solution reconstruction. The proposed methodology is applied on TFAC instances with a weak initial singularity to recover the unknown coefficients. Accuracy is quantified by the maximum-norm error* $E_\infty(K)$ *and the relative* L^2*-error* $E_2(K)$*. Throughout,* L *denotes the network depth (number of layers),* N_i *is the width of layer* i*.*



(a) (b)

Fig. 7: Alikhanov–XfPINN predictions for $\alpha = 0.3, 0.6, 0.9$ vs. exact integer-order FN solutions: (a) $1D$-TFFN; (b) $2D$-TFFN.

Table 12: Optimal network hyperparameters for the inverse 2D-TFRD problem.

Parameter	Value
L	1
N_i	30
Optimizer	Adam
IBCs constraints	Hard
Learning rate (Adam)	$1e-04$
Learning rate (unknown parameters)	$\alpha = 1e-04$
	$\lambda_1 = 1e-04$
	$\lambda_2 = 1e-04$
m_{stage}	8000

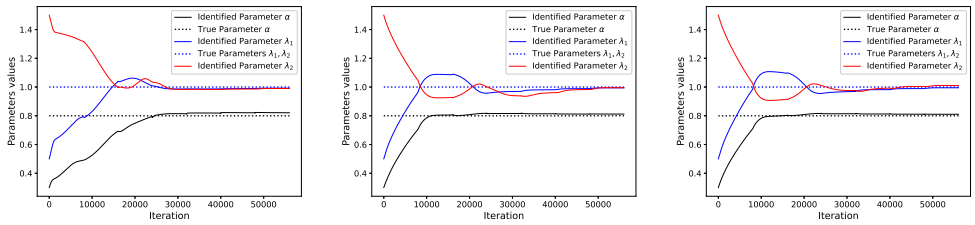
Consider the inverse problem of TFAC equation on $\Omega^2 = [0, L] \times [0, L]$

$$
\begin{aligned}
{}_{0}^{C}\partial_t^{\alpha} v(x,y,t) &= \Psi(\epsilon^2 \Delta v(x,t) - f(v)) + g(x,y,t), && (x,y,t) \in \Omega^2 \times [0,T], \\
v(x,y,0) &= \varphi(x,y), && (x,y) \in \Omega^2, \\
v(x,y,t) &= \phi(x,y,t), && (x,y,t) \in \partial\Omega^2 \times [0,T], \\
v(x,y,T) &= \psi(x,y), && (x,y) \in \Omega^2,
\end{aligned}
\tag{3.5}
$$

where, the parameters α, ϵ, *and* Γ *are treated as unknowns. Given* $f(v)$ *reaction term,* $g(x, y, t)$ *source term,* $\varphi(x, y)$ *ICs,* $\phi(x, y, t)$ *BCs, and a sparse set of terminal-time observations* $\psi(x, y)$ *at* $t = T$*, Alikhanov–XfPINNs framework is employed to identify the fractional order* α*, the interfacial width* ϵ*, and the mobility parameter* Ψ*.*

Table 14 summarizes the network configurations used for the inverse TFAC problems. The listed parameters include the layer count L*, the number of neurons* N_i *per layer, the IBC enforcement type, the optimizer step size, and the target parameter set. The termination criteria are also defined by the stage-wise maximum iteration count* m_{stage} *and the convergence threshold* ε*.*

We now evaluate the effect of time-mesh grading on the recovery of inverse parameters for the TFAC problem. For the two-dimensional TFAC equation with a weak initial singularity, Figure 9 plots the Alikhanov–XfPINNs estimates of α*,* ϵ*, and* Γ *ob-*



(a) $\gamma = 1$. (b) $\gamma = 2/\alpha$. (c) $\gamma = (3-\alpha)/\alpha$.

Fig. 8: Evolution of the estimated parameters α, λ_1, and λ_2 during training of the Alikhanov–XfPINNs for the 2D-TFRD equation on nonuniform meshes with varying grading parameter γ.

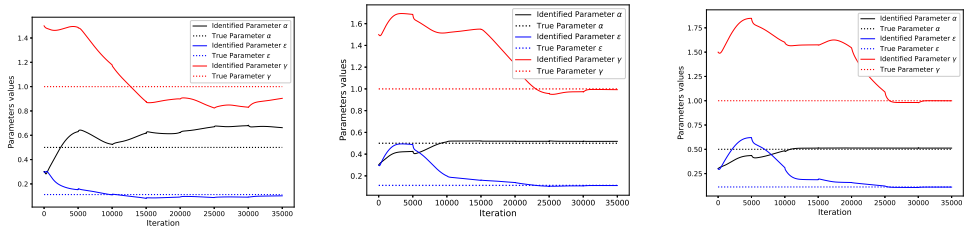
Table 13: Comparison of true and estimated parameters, relative L^2-error, and maximum error for different γ-values.

γ	True Parameters	Estimated Parameters	$E_2(K)$	$E_\infty(K)$
1	$\alpha = 0.8$	$\alpha = 0.821$	$8.299e-03$	$6.951e-03$
	$\lambda_1 = 1$	$\lambda_1 = 0.992$		
	$\lambda_2 = 1$	$\lambda_2 = 0.990$		
$2/\alpha$	$\alpha = 0.8$	$\alpha = 0.812$	$5.182e-03$	$3.083e-03$
	$\lambda_1 = 1$	$\lambda_1 = 0.994$		
	$\lambda_2 = 1$	$\lambda_2 = 0.995$		
$(3-\alpha)/\alpha$	$\alpha = 0.8$	$\alpha = 0.810$	$4.599e-03$	$2.875e-03$
	$\lambda_1 = 1$	$\lambda_1 = 0.995$		
	$\lambda_2 = 1$	$\lambda_2 = 1.010$		

tained with grading parameters $\gamma = 1$, $\gamma = 2/\alpha$, and $\gamma = (3-\alpha)/\alpha$. Compared to the uniform mesh ($\gamma = 1$), the graded settings $\gamma = 2/\alpha$ and $\gamma = (3-\alpha)/\alpha$ bring the estimates closer to the ground truth, underscoring the advantage of nonuniform time discretizations for the inverse TFAC problem. Table 15 presents and compares the exact and recovered parameters, reporting the maximum-norm error E_∞ and the relative L^2-error $E_2(K)$. In line with the visual trends, both error measures are reduced on graded meshes, and all parameters are identified with high accuracy.

Table 14: Optimal network hyperparameters for the inverse 2D-TFAC problem.

Parameter	Value
L	1
N_i	30
IBCs constraints	Hard
Optimizer	Adam
Learning rate (Adam)	$1e-04$
Learning rate (unknown parameters)	$\alpha = 1e-04$
	$\varepsilon = 1e-04$
	$\Psi = 1e-04$
m_{stage}	5000



(a) $\gamma = 1$. (b) $\gamma = 2/\alpha$. (c) $\gamma = (3-\alpha)/\alpha$.

Fig. 9: Evolution of the estimated parameters α, ε, and Γ during training of the Alikhanov–XfPINNs for the 2D-TFAC equation on nonuniform meshes with varying grading parameter γ.

Table 15: Comparison of true and estimated parameters, relative L^2-error, and maximum error for different γ-values.

γ	True Parameters	Estimated Parameters	$E_2(K)$	$E_\infty(K)$
1	$\alpha = 0.5$	$\alpha = 0.663$	9.572e-03	1.258e-02
	$\varepsilon = \sqrt{2}/4\pi$	$\varepsilon = 0.102$		
	$\Psi = 1$	$\Psi = 0.903$		
$2/\alpha$	$\alpha = 0.5$	$\alpha = 0.518$	3.280e-03	2.553e-03
	$\varepsilon = \sqrt{2}/4\pi$	$\varepsilon = 0.111$		
	$\Psi = 1$	$\Psi = 0.993$		
$(3-\alpha)/\alpha$	$\alpha = 0.5$	$\alpha = 0.513$	2.974e-03	2.109e-03
	$\varepsilon = \sqrt{2}/4\pi$	$\varepsilon = 0.111$		
	$\Psi = 1$	$\Psi = 0.999$		

4. Conclusion. This work introduces a data-driven approach to solving nonlinear fractional partial differential equations (NfPDEs) that integrates an accelerated Alikhanov discretization on nonuniform time meshes with physics-informed neural networks (PINNs). This new method is called an *Alikhanov-PINN*. This method builds efficient surrogate models that accurately approximate forward and inverse solutions of high-dimensional NfPDEs, including those with initial singularities and unknown analytical solutions. Unlike traditional finite difference-based fPINNs, our approach avoids full-domain training and dense system solvers. It employs a time-marching adaptive strategy that enables localized and efficient approximation on each temporal interval. Additionally, the method requires no spatial meshing, making it effective for irregular and curved domains in high dimensions. Including an adaptive activation function significantly accelerates convergence, and the flexible incorporation of hard and soft constraints enables the method to handle homogeneous and nonhomogeneous initial-boundary conditions.

We systematically examine the influence of network depth, adaptive activation, and collocation strategies on predictive accuracy for forward problems. Extensive experiments demonstrate the robustness of the method across 1D, 2D, and geometrically complex domains and validate its scalability for long-time simulations. In the inverse problem setting, Alikhanov-PINNs reliably identify unknown parameters and accu-

rately recover latent dynamics. The auxiliary time-marching configuration, termed Alikhanov-fPINN–M, further enforces the discrete residual sequentially in time and provides an auditable protocol for examining empirical temporal convergence with respect to the maximal time step on graded meshes. Under controlled training tolerances, this configuration provides clear numerical evidence of the expected temporal convergence behavior of the underlying Alikhanov discretization. The observed convergence behavior aligns with theoretical expectations, affirming the method's numerical reliability. Looking ahead, this work establishes a foundation for several promising directions. These include extending the framework to accommodate other types of fractional operators, applying the method to broader classes of PDEs in physics and biology, and developing theoretical analyses of its approximation, generalization, and convergence properties. Furthermore, refining the methodology for hyperparameter selection and adaptive sampling strategies is essential for future optimization.

In future work, we will apply the proposed framework to real observational data. Estimating the fractional order α directly from measurements would provide important data-driven justification for using time-fractional models. Allowing the order to vary over time, i.e., $\alpha = \alpha(t)$, could capture evolving memory effects and anomalous transport regimes that a constant fractional order may not adequately represent.

Author Contributions. Himanshu Kumar Dwivedi: Conceptualization, Software, Methodology, Validation, Formal Analysis, Investigation, Writing-Original Draft, Writing-Review and Editing, Visualization.
Matthias Ehrhardt: Conceptualization, Supervision, Methodology, Formal Analysis, Investigation, Writing-Review and Editing.
Rajeev: Supervision, Funding acquisition, Writing-Review and Editing, Validation.

Data availability. The datasets supporting this study are available from the corresponding author upon reasonable request.

Declarations.

Conflict of interest. The authors confirm they have no conflicts of interest to report.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, Adv. Neural Process. Syst. (2012) 1097-1105.
[2] B. M. Lake, R. Salakhutdinov, J. B. Tenenbaum, Human-level concept learning through probabilistic program induction, Science 350 (2015) 1332-1338.
[3] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problem involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686-707.
[4] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3(6) (2021) 422-440.
[5] X. Jin, S. Cai, H. Li, G. E. Karniadakis, NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, J. Comput. Phys. 426 (2021) 109951.
[6] M. Raissi, Z. Wang, M. S. Triantafyllou, G. E. Karniadakis, Deep learning of vortex-induced vibrations, J. Fluid Mech. 861 (2019) 119-137.
[7] D. Liu, Y. Wang, Multi-fidelity physics-constrained neural network and its application in material modeling, J. Mech. Des. 141(12) (2019) 121403.
[8] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks, Comput. Methods Appl. Mech. Eng. 358 (2020) 112623.

[9] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339-1364.
[10] L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, DeepXDE: a deep learning library for solving differential equations, SIAM Rev. 63(1) (2021) 208-228.
[11] S. Wang, S. Sankaran, P. Perdikaris, Respecting causality for training physics-informed neural networks, Comput. Methods Appl. Mech. Eng Phys. 421 (2024) 116813.
[12] C. Basdevant, M. Deville, P. Haldenwang, J. Lacorix, J. Ouazzani, R. Peyret, P. Orlandi, A. Patera, Spectral and finite difference solutions of the Burgers' equation, Comput. Fluids. 14 (1986) 23-41.
[13] Z. Sun, G. Gao, Fractional differential equations: finite difference methods, De Gruyter, Berlin (2020).
[14] B. Jin, R. Lazarov, Z. Zhou, An analysis of the L1 scheme for the subdiffusion equation with nonsmooth data, IMA J. Numer. Anal. 36 (2016) 197–221.
[15] M. Stynes, E. O´Riordan, J. Gracia, Error analysis of a finite difference method on graded meshes for a time-fractional diffusion equation, SIAM J. Numer. Anal. 55(2) (2017) 1057–1079.
[16] W. McLean, Regularity of solutions to a time-fractional diffusion equation, ANZIAM J. 52 (2010) 123–138.
[17] H. Chen, M. Stynes, Error analysis of a second-order method on fitted meshes for a time-fractional diffusion problem, J. Sci. Comput. 79(1) (2019) 624–647.
[18] S. Jiang, J. Zhang, Q. Zhang, Z. Zhang, Fast evaluation of the Caputo fractional derivative and its applications to fractional differential equations, Commun. Comput. Phys. 21(3) (2017). 650-678.
[19] H. K. Dwivedi, Rajeev, A novel fast second order approach with high-order compact difference scheme and its analysis for the tempered fractional Burgers equation, Math. Comput. Simul. 227 (2025) 168-188.
[20] H. K. Dwivedi, Rajeev, A novel fast tempered algorithm with high-accuracy scheme for 2D tempered fractional reaction-advection-subdiffusion equation, Comput. Math. Appl. 176 (2024) 371-397.
[21] J. Cao, A. Xiao, W. Bu, Finite difference/finite element method for the tempered time fractional advection-dispersion equation with fast evaluation of Caputo derivatives, J. Sci. Comput. (2020) 83:48.
[22] Y. Yan, Z. Z. Sun, J. Zhang, Fast evaluation of the Caputo fractional derivative and its applications to fractional diffusion equations: a second order scheme, Commun. Comput. Phys. 22(4) (2017) 1028-1048.
[23] X. Li, H. L. Liao, L. Zhang, A second-order fast compact scheme with unequal time-steps for subdiffusion problems, Numer. Algor. 86 (2021) 1011–1039.
[24] Y. Luchko, W. Rundell, M. Yamamoto, L. Zuo, Uniqueness and reconstruction of an unknown semilinear term in a time-fractional reaction diffusion equation, Inverse Probl. 29(6) (2013) 065019.
[25] M. Kern, Numerical methods for inverse problems, Wiley, Hoboken (2016).
[26] D. Lukyanenko, R. Argun, A. Borzunov, A. Gorbachev, V. Shinkarev, M. Shislenin, A. Yagola, On the features of numerical solution of coefficient inverse problems for nonlinear equations of the reaction-diffusion-advection type with data of various types, Differ. Equ. 59(12) (2023) 1734-1757
[27] X. An, Q. Wang, F. Liu, V. V. Anh, I. W. Turner, Parameter estimation for time-fractional Black-Scholes equation with S&P 500 index option, Numer. Algor. 95(1) (2024) 1-30.
[28] M. Srati, A. Oulmelk, L. Afraites, A. Hadri, M. A. Zaky, A. Aldraiweesh, A. S. Hendy, An inverse problem of determining the parameters in diffusion equations by using fractional physics-informed neural networks, Appl. Numer. Math. 208 (2025) 189-213.
[29] L. Ma, R. Li, F. Zeng, L. Guo, G. E. Karniadakis, Bi-orthogonal fPINN: a physics-informed neural network method for solving time-dependent stochastic fractional PDEs, Commun. Comput. Phys. 34(4) (2023) 1133-1176.
[30] X. Fang, L. Qiao, F. Zhang, F. Sun, Explore deep network for a class of fractional partial differential equations, Chaos Solitons Fractals 172 (2022) 113528.
[31] J. Shi, X. Liu, X, Yang, Data-driven solutions and parameter estimation of the high-dimensional time-fractional reaction-diffusion equations using an improved fPINN method, Nonlinear Dyn. 113 (2025) 9577-9604.
[32] G. Pang, L. Lu, G. E. Karniadakis, fPINNS: fractional physics-informed neural networks, SIAM J. Sci. Comput. 41(4) (2019) 2603-2626 .
[33] S. Wang, H. Zhang, X. Jiang, Fractional physics-informed neural networks for time-fractional phase field models, Nonlin. Dyn. 110(3) (2022) 2715-2739.

[34] Z. Ma, J. Hou, W. Zhu, Y. Peng, Y. Li, PMNN: physical model driven neural network for solving time-fractional differential equations, Chaos Solitons Fractals 177 (2023) 114238.
[35] H. Ren, X. Meng, R. Liu, J. Hou, Y. Yu, A class of improved fractional physics-informed neural networks, Neurocomputing 562 (2023) 126890.
[36] H. L. Liao, , D. Li, J. Zhang, Sharp error estimate of nonuniform L1 formula for time-fractional reaction-subdiffusion equations, SIAM J. Numer. Anal. 56 (2018) 1112–1133.
[37] A. A. Alikhanov, A new difference scheme for the time fractional diffusion equation, J. Comput. Phys. 280, (2015) 424-438.
[38] H. L. Liao, W. Mclean, J. Zhang, A second-order scheme with nonuniform time steps for a linear reaction-subdiffusion problem, Commun. Comput. Phys. 30(2), (2021) 567-601.
[39] A. D. Jagtap, K. Kawaguchi, G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404, (2020) 109136.
[40] H. Gao, L. Sun, J. Wang, PhyGeoNet: physics-informed geometry-adaptive convolutional neural networks for solving parametrized steady-state PDE on irregular domain, J. Comput. Phys. 428, (2021) 110079.
[41] J. Shi, X. Liu, X. Yang, Data-driven solutions and parameter estimation of the high-dimensional time-fractional reaction-diffusion equations using an improved fPINN method, Nonlinear Dyn. 113 (2025)9577-9604.
[42] M. Abadi, A. Agarwal, P. Barham, E. Brevdeo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv:1603.04467 (2016).
[43] A. Barkat, P. Bianchi, Convergence and dynamical behavior of the ADAM algorithm for nonconvex stochastic optimization, SIAM J. Optimiz. 31(1) (2021) 244-274.