Journal of Intelligent Computing and Networking
https://www.ffspub.com/index.php/jicn/index
ISSN 3079-9228 (print)
ISSN 3107-2798 (online)
E-mail: jicn.office@ffspub.com

Article

Refining Graph Learning Dynamics with the Riemann–Liouville Fractional Derivative

Yufeng Peng[1], Qiyu Kang[1†], Kai Zhao[2], Xuhao Li[3] and Qinxu Ding[4]
[1]*School of Information Science and Technology, University of Science and Technology of China, Hefei 230026, China*
[2]*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore*
[3]*School of Mathematical Sciences, Anhui University, Hefei 230601, China*
[4]*School of Business, Singapore University of Social Sciences, Singapore 599494, Singapore*
[†]*E-mail: qiyukang@ustc.edu.cn*

Received: March 30, 2026 / Revised: May 12, 2026 / Accepted: May 26, 2026 / Published online: June 1, 2026

Abstract: We introduce the Riemann–Liouville Graph neural Fractional-order Differential Equation (RL-GFDE), a novel continuous Graph Neural Network (GNN) framework that incorporates the Riemann–Liouville (RL) fractional derivative. This framework effectively captures feature updating dynamics with pronounced memory effects, offering an advantage over standard graph neural ordinary differential equation (ODE) models that rely solely on integer-order derivatives. Our contributions are twofold: First, we establish a generalized continuous GNN framework grounded in fractional calculus. We also provide a comprehensive convergence analysis for the solvers applied to RL-GFDE, ensuring precision and efficiency in our method. Second, through theoretical analysis and extensive empirical experiments, we demonstrate the enhanced performance of RL-GFDE by comparing fractional adaptations of various well-known contemporary graph neural ODE models against their original integer-order counterparts. Our results validate the efficacy of RL-GFDE, showcasing its versatility as an extension to boost the performance of graph neural ODE models in various applications.

Keywords: Graph Neural Networks; Fractional Calculus; Riemann–Liouville Derivative
https://doi.org/10.64509/jicn.22.93

1 Introduction

Graph Neural Networks (GNNs) [1–3] have emerged as a dominant architecture for learning on graph-structured data, boasting exceptional performance in a wide array of applications such as object detection [4], materials science [5], and social recommender systems [6], among others. Recent developments in GNN research have seen a growing emphasis on utilizing ordinary or partial differential equations (ODE/PDE) for modeling neighbor aggregation and feature updates, leading to the development of continuous GNN models [7]. Prominent contributions in this area include GDE [8], CGNN [9], GRAND [10], GRAND++[11], GraphCON [12], GraphBel [13], GREAD [14] and CDE [15], among others. Leveraging the properties of continuous dynamical systems, the proposed graph neural ODE models demonstrate notable advantages such as enhanced robustness [13, 16], improved handling of heterophilic data [14, 15], and mitigation of oversmoothing [12].

In recent years, the exploration of Fractional-order Differential Equations (FDEs) has gained significant momentum in various fields such as scientific modeling and engineering. Distinct from traditional ODEs, which are limited to integer orders, FDEs significantly broaden the scope by allowing the order α in the differential operator $\frac{d^\alpha}{dt^\alpha}$ to be any real number. This flexibility allows FDEs to encompass a wider range of phenomena, making them a more versatile and comprehensive mathematical tool. The ability of FDEs to include nonlocal behaviors and memory effects in functions, as highlighted by [17], has been pivotal in enhancing dynamic models creation. These models more accurately capture the lasting impacts of historical states, leading to a finer representation of complex behaviors that exhibit significant memory-dependence. Such characteristics are crucial in various natural and engineered systems. The application range of FDEs is extensive and diverse, covering fields such as anomalous diffusion [18], image processing [19], and fractal media

† Corresponding author: Qiyu Kang
*Academic Editor: Xueyang Fu

[20]. This paper aims to extend the reach of FDEs by incorporating them into the continuous GNN field, further exploring their potential in advanced computational models.

A plethora of fractional derivative definitions for $\frac{\mathrm{d}^\alpha}{\mathrm{d}t^\alpha}$ exist in the literature, with notable examples attributed to Riemann, Liouville, Chapman, and Caputo [21]. The Riemann-Liouville (RL) and Caputo derivatives are the two primary types discussed. The RL derivative is often favored over the Caputo derivative due to its broader mathematical scope and suitability for theoretical analysis. It is particularly useful in scenarios demanding a detailed account of memory effects. When contrasting the RL fractional derivative with its integer-order counterpart, consider a scalar function $f(t)$. The first-order derivative of $f(t)$ with respect to time t, assuming a well-defined limit, is defined as:

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} := \lim_{\Delta t \to 0} \frac{f(t+\Delta t) - f(t)}{\Delta t}. \tag{1}$$

This standard derivative quantifies the instantaneous rate of change of the function at a specific point, reflecting the local behavior around that point. In contrast, the RL derivative, offering a broader perspective, is defined as:

$${}^{\mathrm{RL}}D^\alpha f(t) := \frac{1}{\Gamma(n-\alpha)} \frac{\mathrm{d}^n}{\mathrm{d}t^n} \int_0^t (t-\tau)^{n-\alpha-1} f(\tau)\mathrm{d}\tau. \tag{2}$$

Here, $\alpha > 0$ and $n = \lceil \alpha \rceil$ (the smallest integer greater than or equal to α). Unlike the immediate locality captured by the first-order derivative, the RL derivative ${}^{\mathrm{RL}}D^\alpha f(t)$ is influenced by the *historical trajectory* of f over the entire interval $[0,t]$. This represents a significant deviation from the localized influence captured in Equation 1. More precisely, the RL fractional derivative heavily weights influences from the distant past. These are characterized as integrals over past values of f, leveraging a power-law kernel that assigns slowly decreasing significance to historical events. This characteristic enables the RL derivative to capture long-range dependencies more effectively, making them especially useful in situations where past states have a lasting impact on future dynamics.

In this study, we merge graph neural ODE models with the RL derivative to introduce the Riemann-Liouville Graph neural FDE (RL-GFDE) framework. We demonstrate its adaptability by empirically evaluating its performance on a range of benchmark graph neural ODE models, including GRAND, GraphCON, CDE, and GREAD. Across the reported benchmarks, the fractional versions improve over their integer-order counterparts in most evaluated settings, while the magnitude of improvement depends on the dataset and backbone model. A common issue in GNNs is oversmoothing [22], where node representations become increasingly similar as the number of graph layers (or depth) increases. Our findings reveal that the RL-GFDE framework can effectively mitigate this oversmoothing in graph ODE models, even in those with depths up to 256 layers. This improvement follows from the inherent memory effect in our proposed graph dynamical system, where historical node states have a direct influence on their present versions, ensuring a richer and more diverse representation at each layer.

We also clarify the relation between RL-GFDE and adjacent lines of work. Neural FDE methods usually study fractional dynamics in general neural dynamical systems, fractional GNNs often introduce fractional operators or gradients directly into graph architectures, and memory-based graph learning methods typically add explicit recurrent, residual, or attention-based memory modules. In contrast, RL-GFDE keeps the graph neural ODE backbone and its right-hand-side graph operator intact, but replaces the left-hand-side integer-order time derivative with the RL fractional derivative. This makes the contribution a plug-in continuous-dynamics extension for graph neural ODE backbones, together with graph-oriented numerical solvers and convergence analysis.

Main Contributions. The key contributions of our study are summarized as follows:

- We propose a generalized continuous graph learning framework that extends the traditional graph neural ODE models, incorporating the more general RL fractional derivative in place of integer-order derivatives. As $\alpha \to 1$, the non-local RL fractional derivative operator, detailed in Equation 2, converges to the conventional local first-order derivative $\frac{\mathrm{d}}{\mathrm{d}t}$ in Equation 1 [23]. Our approach therefore recognizes these traditional models as specific cases within a broader spectrum, where the RL fractional derivative enhances their ability to capture more complex feature updating dynamics with memory effects. Furthermore, we also provide a comprehensive convergence analysis for the solvers applied to RL-GFDE, ensuring that our method is not only precise but also efficient.
- Our empirical evaluations of RL-GFDE adapted graph neural ODE models demonstrate performance gains in most of the evaluated settings compared to their integer-order counterparts. Additionally, the RL-GFDE framework exhibits unique advantages in long-range graph benchmarks, attributable to its inherent long memory effect.
- The RL-GFDE framework is both theoretically and empirically validated as an effective solution for addressing the oversmoothing issue, a common challenge in deep GNNs. Our method maintains distinct and informative node representations, even in models with depths of up to 256 layers after discretization.

The rest of this paper is organized as follows. Section 2 provides foundational information on graph neural ODE models and the RL integral and differential operators. Section 3 outlines the RL-GFDE framework, including details on numerical solvers and their convergence. Section 4 presents experimental results pertaining to various implementations of RL-GFDE. We also analyze the unique benefits offered by the RL-GFDE approach in section A. In sections B and D we provide some technique details and the proof of our proposed theorems.

2 Preliminaries

2.1 Graph Neural ODE Models

The seminal work of [24] introduces neural ODEs with open-source solvers to model continuous residual layers, which has subsequently been applied to the field of continuous GNNs.

The models discussed below are notable examples of this application. They will be further extended to their RL fractional derivative versions in Section 3. A concise overview of their dynamics is provided herein.

Notations: Consider a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V} = 1, \ldots, N$ represents the set of N nodes, and $\mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}$ denotes the set of edges. The features of the vertices at time t are denoted by $\mathbf{X}(t) \in \mathbb{R}^{|\mathscr{V}| \times d}$, where d is the dimension of the node features. In graph neural ODE models, the initial value of the ODE, $\mathbf{X}(0)$, is derived from a linear layer or a Multilayer Perceptron (MLP) applied to the raw input features of the graph. Given a time T, once we obtain $\mathbf{X}(T)$ through numerical solvers, it can be employed for graph learning tasks, such as node classification or link prediction.

GRAND: Inspired by heat diffusion in graph manifolds, the paper [10] introduces the GRAND model, which employs continuous ODEs to represent feature updates in GNNs. In simpler terms, we can analogize the integral time to the layers in GNNs. The ODE in the GRAND model is formulated as follows:

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = (\mathbf{A}(\mathbf{X}(t)) - \mathbf{I})\mathbf{X}(t). \tag{3}$$

Here, $\mathbf{A}(\mathbf{X}(t))$ represents the attention matrix, and $\mathbf{I}$ is the identity matrix. The GRAND model is divided into two versions based on the nature of the attention matrix $\mathbf{A}(\mathbf{X}(t))$: the linear version (GRAND-l), where the attention matrix is time-independent and remains fixed over the integration, and the non-linear version (GRAND-nl), where the attention matrix is time-dependent, allowing weights to change in accordance with $\mathbf{X}(t)$.

GRAND++: Building on the GRAND model, the paper [11] introduces the GRAND++ model. This model enhances the original GRAND by including a source term, which aims to tackle the challenge of training with limited labeled data. The differential equation employed in GRAND++ is given by:

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = -\mathbf{L}(\mathbf{X}(t)) + \mathbf{C}(0) \tag{4}$$

where $\mathbf{L}(\mathbf{X}(t))$ represents the graph Laplacian matrix, $\mathbf{C}(0)$ is a subset of $\mathbf{X}(0)$, comprised exclusively of nodes deemed "trustworthy".

GraphCON: In the paper [12], the authors propose another extension to the GRAND model by reinterpreting GNNs as coupled oscillator networks, utilizing second-order ODEs. This approach introduces stability and addresses the over-smoothing problem inherent in GNNs. For computational feasibility, the second-order ODE is divided into two first-order ODEs. This division incorporates the concept of velocity $\mathbf{Y}(t)$, as illustrated in the following equations:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{Y}(t)}{\mathrm{d}t} &= \sigma(\mathbf{F}_\theta(\mathbf{X}(t), t)) - \gamma\mathbf{X}(t) - \alpha\mathbf{Y}(t), \\ \frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} &= \mathbf{Y}(t), \end{aligned} \tag{5}$$

where $\sigma(\cdot)$ is the activation function, $\mathbf{F}_\theta(\mathbf{X}(t), t)$ is the neural network function parameterized by θ, γ and α are learnable scalar coefficients,$\mathbf{Y}(t)$ is introduced as the velocity term in the transformation of the second-order ODE into two first-order ODEs. Similar to the GRAND model, the GraphCON model also comes in both linear (GraphCON-l) and non-linear (GraphCON-nl) versions with respect to time. The distinction between these two versions lies in whether or not the function $\mathbf{F}_\theta$ updates based on time t.

GREAD: To address the challenges associated with heterophilic graphs, the paper [14] introduces the GREAD model. This model enhances the GRAND framework by incorporating a reaction term, thereby formulating a diffusion-reaction equation for GNNs. The equation for this model is depicted as follows:

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = -\alpha\mathbf{L}(\mathbf{X}(t)) + \alpha r(\mathbf{X}(t)), \tag{6}$$

where $r(\mathbf{X}(t))$ is a reaction term, and α is a trainable parameter used to emphasize each term.

CDE: In a similar vein, focusing on heterophilic graphs, the concept of convection-diffusion equations (CDE) is introduced into GNNs, culminating in the neural CDE model [15]. This model integrates a convection term and adopts the concept of velocity for each node, aiming to preserve node feature diversity in heterophilic graph contexts. The CDE model is mathematically expressed as:

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = (\mathbf{A}(\mathbf{X}(t)) - \mathbf{I})\mathbf{X}(t) + \mathrm{div}(\mathbf{V}(t) \circ \mathbf{X}(t)), \tag{7}$$

where $\mathbf{V}(t)$ is the velocity field of the graph at time t, $\mathrm{div}(\cdot)$ is the divergence operator defined in the paper [13], $\circ$ denotes the element-wise (Hadamard) product.

We observe that existing graph neural ODE models typically propose dynamic functions to address various problems, with a focus primarily on the right-hand side of ODEs. In contrast, our paper introduces the RL-GFDE, an innovative framework that incorporates fractional-order derivatives on the left-hand side, effectively transforming ODEs into FDEs. The RL-GFDE construction applies to all models presented in Equations (3–7) and to graph neural ODE models with analogous continuous-time dynamics.

2.2 Riemann–Liouville Integral and Differential Operators

In fractional calculus, the Riemann–Liouville integral and differential operators are of paramount importance, extending traditional integration and differentiation to encompass non-integer orders. We discuss its generalization over traditional integer-order derivatives. These operators are crucial for modeling continuous processes that are not localized in time but depend on their entire history.

Definition 1 Let $\alpha \in \mathbb{R}^+$. The operator J^α, for $0 \le t \le b$ and defined on $L_1[0,b]$ by

$$J^\alpha f(t) := \frac{1}{\Gamma(\alpha)} \int_a^t (t-\tau)^{\alpha-1} f(\tau)\, d\tau,$$

is called the RL fractional integral operator of order α.

Here, $\Gamma(\alpha)$ is the gamma function. Analogous to a standard integral that aggregates a function's values over a specific interval, the RL integral assigns weights to these values following a power law. This weighting reflects memory or hereditary effects in physical systems. Like standard integration, RL integral operators form a commutative semigroup with respect to concatenation [23, Theorem 2.4].

Definition 2 Let $\alpha \in \mathbb{R}^+$ and let $n = \lceil \alpha \rceil$. The operator ${}^{\mathrm{RL}}D^{\alpha}$, defined by

$$ {}^{\mathrm{RL}}D^{\alpha} f := \frac{\mathrm{d}^n}{\mathrm{d}t^n} J^{n-\alpha} f, $$

is called the RL fractional differential operator of order α.

In this context, $\frac{\mathrm{d}^n}{\mathrm{d}t^n}$ represents the standard integer differential operator. The RL fractional derivative extends the concept of a standard derivative to include non-integer orders. It is not simply a local rate of change as defined in Equation 1, but rather an aggregate rate that incorporates the function's historical behavior, weighted by the fractional order. When α is an integer, the fractional derivative reverts to the standard derivative $\frac{\mathrm{d}^{\alpha}}{\mathrm{d}t^{\alpha}}$. For suitable functions $f(t)$ that $f^{(1)}(t) \in C[0,t]$ and $\alpha \in (0,1)$, the RL fractional derivative of order α can also be written as a pure integral operator [25, Theorem 2.1]:

$$ {}^{\mathrm{RL}}D^{\alpha} f(t) = \frac{1}{\Gamma(-\alpha)} \int_0^t (t-\tau)^{-\alpha-1} f(\tau) d\tau. \tag{8} $$

This integral operator representation emphasizes the connection between fractional derivatives and traditional integrals, highlighting the non-local nature of fractional differentiation. Unlike standard differentiation, which focuses on a function's behavior at a single point, fractional differentiation encompasses the entire history of the function, capturing long-memory dependencies.

We next define the Mittag-Leffler function, a concept of significant importance in the field of fractional calculus.

Definition 3 (Mittag-Leffler function) The two-parameter Mittag-Leffler function is defined by the series expansion:

$$ E_{\alpha,\beta}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k + \beta)}. \quad (\alpha > 0,\ \beta > 0) $$

This function is entire, meaning that the series converges for all values of the argument z. Considering the following linear homogeneous RL FDE:

$$ {}^{\mathrm{RL}}D^{\alpha} f(t) = -\lambda f(t), \tag{9} $$

where λ is a positive constant, $0 < \alpha \leq 1$ and the initial condition is $\lim_{t \to 0^+} J^{1-\alpha} f(t) = 1$. It follows that the function

$$ f(t) = t^{\alpha-1} E_{\alpha,\alpha}(-\lambda t^{\alpha}) \tag{10} $$

is a solution of the above linear FDE Equation 9. Furthermore, the solution $f(t)$ exhibits a slow algebraic convergence rate $\Theta(t^{-\kappa})$ towards 0 with $\kappa > 0$ for $0 < \alpha < 1$ [1]. A detailed proof of this assertion, along with its extension to a multidimensional vector form, will be provided later in Section A.2. This appendix section will also explore the connection between the slow algebraic convergence phenomenon and the oversmoothing mitigation within our RL-GFDE framework.

3 Methods

3.1 Framework

As detailed in Section 2.1 on graph neural ODE models, we extend this concept to define neural RL-GFDE. This integration involves incorporating RL derivatives into the neural ODE framework. Our proposed RL-GFDE framework can be represented by the following equation:

$$ {}^{\mathrm{RL}}D^{\alpha} \mathbf{X}(t) = \mathscr{F}(\mathscr{E}, \mathbf{X}(t)), \quad 0 < \alpha \leq 1, \tag{11} $$

where α is the fractional order, and $\mathscr{F}(\mathscr{E}, \mathbf{X}(t))$ is a dynamic operator on the graph. In our framework, α acts as a hyperparameter, introducing enhanced flexibility and adaptability to the model. When α is in the range $0 < \alpha \leq 1$, the RL-GFDE models interpolate between purely memoryless systems (at $\alpha = 1$, corresponding to standard ODEs) and systems with maximal memory (approaching $\alpha = 0$). This allows for a smooth transition between different dynamic behaviors. The function $\mathscr{F}$ is specifically tailored for graph dynamics and can be set to the functions detailed in Equations (3–7). Using this approach, we derive several RL-GFDE models, including RL-GRAND, RL-GRAND++, RL-GraphCON, RL-GREAD, and RL-CDE. Below, we detail the formulations of these models:

- **RL-GRAND & RL-GRAND++:**

$$ {}^{\mathrm{RL}}D^{\alpha} \mathbf{X}(t) = (\mathbf{A}(\mathbf{X}(t)) - \mathbf{I})\mathbf{X}(t) \tag{12} $$

$$ {}^{\mathrm{RL}}D^{\alpha} \mathbf{X}(t) = -\mathbf{L}(\mathbf{X}(t)) + \mathbf{C}(0) \tag{13} $$

- **RL-GraphCON:**

$$ {}^{\mathrm{RL}}D^{\alpha} \mathbf{Y}(t) = \sigma(\mathbf{F}_{\theta}(\mathbf{X}(t), t)) - \gamma \mathbf{X}(t) - \alpha \mathbf{Y}(t) $$

$$ {}^{\mathrm{RL}}D^{\alpha} \mathbf{X}(t) = \mathbf{Y}(t) \tag{14} $$

- **RL-GREAD:**

$$ {}^{\mathrm{RL}}D^{\alpha} \mathbf{X}(t) = -\alpha \mathbf{L}(\mathbf{X}(t)) + \alpha r(\mathbf{X}(t)) \tag{15} $$

- **RL-CDE:**

$$ {}^{\mathrm{RL}}D^{\alpha} \mathbf{X}(t) = (\mathbf{A}(\mathbf{X}(t)) - \mathbf{I})\mathbf{X}(t) + \mathrm{div}(\mathbf{V}(t) \circ \mathbf{X}(t)) \tag{16} $$

3.2 Solving RL-GFDE

In the established graph neural ODE paradigm, as explored in Section 2.1, the continuous time variable t serves a role analogous to that of layers in GNNs [10]. The discretization of time becomes a critical process in many neural ODE solvers. For instance, with the explicit Euler scheme, neural ODEs are reduced to residual networks with skip connections [24].

Our framework includes FDEs, inherently more complex than their ordinary differential counterparts. Drawing on [17], we apply the *Grünwald-Letnikov* and *Product Trapezoidal methods* in this study. We introduce initial numerical solvers called *Grünwald-Letnikov predictor* and *Product Trapezoidal predictor*, employing time discretization $t_k = kh$, where the discretization parameter h is a small positive value representing the step size.

3.2.1 Grünwald-Letnikov predictor

The numerical solution for an RL-GFDE defined in Equation 11 can be formulated as:

$$ \mathbf{X}(t_k) = \mathscr{F}(\mathscr{E}, \mathbf{X}(t_{k-1}))h^{\alpha} - \sum_{j=1}^{k} C_j^{\alpha} \mathbf{X}(t_{k-j}), \tag{17} $$

where C_j^{α} are the binomial coefficients which are calculated recursively with $C_0^{\alpha} = 1$ and $C_j^{\alpha} = \left(1 - \frac{1+\alpha}{j}\right) C_{j-1}^{\alpha}, j = 1, 2, \ldots, k$. The coefficients C_j^{α} provide a means of approximating the fractional derivative of the function at different points

[1] We adopt the asymptotic order notations from [26].

in time, giving an accurate numerical solution. By adapting this method to the graph context, we can accurately and robustly estimate the states of graph nodes over time, reflecting the complex dynamics and dependencies that characterize graph-structured data.

3.2.2 Product Trapezoidal predictor

Building upon the approximation of the RL integral operator as defined in Equation 2.3.1 of the paper [17], we introduce the product trapezoidal numerical solver for RL-GFDE as follows:

$$\mathbf{X}(t_k) = \Gamma(2-\alpha)\mathscr{F}(\mathscr{E}, \mathbf{X}(t_{k-1}))h^{\alpha} - \Gamma(2-\alpha)\sum_{j=0}^{k-1} A_{j,k}\mathbf{X}(t_j), \tag{18}$$

where $A_{j,k}$ is a coefficient depending on both j and k. The formula for $A_{j,k}$ can be derived from the properties of the product trapezoidal rule. This method aims to approximate the fractional integral by means of a product trapezoidal rule, which is a commonly used numerical integration method for regular integrals. In this case, the coefficients $A_{j,k}$ capture the contribution of the function values at different time steps, weighted according to the trapezoidal rule. The complete formula for $A_{j,k}$ is

$$A_{j,k}\begin{cases} \frac{(k-1)^{1-\alpha}-(k+\alpha-1)^{-\alpha}}{\Gamma(2-\alpha)}, & \text{if} \quad j=0, \\ \frac{(k-j+1)^{1-\alpha}+(k-j-1)^{1-\alpha}-2(k-j)^{1-\alpha}}{\Gamma(2-\alpha)}, & \text{if} \quad 1\le j\le k-1, \\ \frac{1}{\Gamma(2-\alpha)}, & \text{if} \quad j=k. \end{cases} \tag{19}$$

3.3 Convergence Analysis

We establish convergence for Equation 17 and Equation 18 respectively in this section. In what follows, we present consistency error of *Grünwald-Letnikov* and *Product Trapezoidal* approximation in the appendix, and analyze the convergence of Equation 17 and Equation 18 in below. The oracle solution of the system is denoted as $\bar{\mathbf{X}}(t)$. The assumptions used below follow the standard analysis of fractional initial value problems. The condition $\bar{\mathbf{X}} \in C^2[0,T]$ means that the exact node-feature trajectory is sufficiently smooth on the finite integration interval. Lipschitz continuity of $\mathscr{F}$ means that small perturbations in node features lead to proportionally small changes in the graph dynamics, which is the stability condition needed to propagate local discretization errors through the solver. For the RL derivative, the initial condition is understood in the fractional sense through $\lim_{t\to 0^+} J^{1-\alpha}\mathbf{X}(t)$ rather than only through the point value $\mathbf{X}(0)$; in the graph learning implementation, this fractional initial state is determined by the encoded input node features.

Theorem 1 *(Convergence of Equation 17) The* Grünwald-Letnikov predictor *Equation 17, produces an approximate solution for linear or nonlinear initial value problem Equation 11 that satisfies*

$$|\mathbf{X}(t_j) - \bar{\mathbf{X}}(t_j)| = O(h)$$

uniformly for all j if the solution $\bar{\mathbf{X}} \in C^2[0,T]$ and $\mathscr{F}$ is Lipschitz continuous.

Theorem 2 *(Convergence of Equation 18) The* Product Trapezoidal predictor *Equation 18, produces an approximate solution for linear or nonlinear initial value problem Equation 11 that satisfies*

$$|\mathbf{X}(t_j) - \bar{\mathbf{X}}(t_j)| = O(h)$$

uniformly for all j if the solution $\bar{\mathbf{X}} \in C^2[0,T]$ and $\mathscr{F}$ is Lipschitz continuous.

In our implementation, with the step size h carefully chosen to be smaller than or equal to 1, the method aligns with the convergence properties outlined in the provided Theorem 1 and Theorem 2. This results in a notably strong convergence rate, primarily because the error term is directly proportional to h. Such a choice of step size, in light of the theorem, guarantees an impressive rate of convergence, ensuring that our method is not only accurate but also efficient in approaching the solution as h decreases. For the Product Trapezoidal predictor, the appendix gives a sharper consistency estimate of order $O(h^{2-\alpha})$ for the fractional quadrature approximation. The theorem in the main text reports the conservative global convergence guarantee $O(h)$ for the explicit predictor applied to the full linear or nonlinear RL-GFDE under the Lipschitz error-propagation argument. Thus, the $O(h)$ statement should be read as a uniform global bound rather than as a claim that the product trapezoidal quadrature consistency error is tight.

The process for inference using the RL-GFDE framework is outlined in Algorithm 1.

Algorithm 1 Inference using RL-GFDE

Require: Graph structure $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, integration time T, dynamic operator $\mathscr{F}$, fractional order α.
Ensure: Results for graph inference tasks.

1: **Step 1**: Compress sparse raw input node features using a single layer, such as a fully-connected (FC) layer, to establish initial conditions $\mathbf{X}(0)$.
2: **Step 2**: Solve the RL-GFDE (defined in Section 3.1) using the explicit solvers in Section 3.2.
3: **Step 3**: Obtain $\mathbf{X}(T)$, which are the solutions of Equation 11 at time T.
4: **Step 4**: Proceed with further graph tasks, such as node classification.

4 Results

To demonstrate the integration of our RL-GFDE framework with existing graph neural ODE models, we conducted experiments using popular graph neural ODE models, including GRAND [10], GRAND++ [11], GraphCON [12], CDE [15] and GREAD[14]. By forming RL-GRAND, RL-GRAND++, RL-GraphCON, RL-CDE and RL-GREAD, we aim to show that our RL-GFDE framework works harmoniously with graph neural ODE models and enhances their performance in most evaluated settings. We ensured a fair comparison by replicating the experiments detailed in the original papers for both our RL-GFDE adaptations and the original graph neural ODE models. It is important to note that our framework is not a standalone new model aiming for state-of-the-art results. Instead, it is a plug-in method designed for neural ODE backbones, with performance gains reported according to the dataset, backbone, and chosen fractional order.

Table 1: Dataset statistics of homophilic graphs.

Dataset	Type	Classes	Features	Nodes	Edges
Cora	citation	7	1433	2485	5069
Citeseer	citation	6	3703	2120	3679
PubMed	citation	3	500	19717	44324
Coauthor CS	co-author	15	6805	18333	81894
Computers	co-purchasing	10	767	13381	245778
Photos	co-purchasing	8	745	7487	119043
CoauthorPhy	co-author	5	8415	34493	247962
Airport	tree-like	4	4	3188	3188
Disease	tree-like	2	1000	1044	1043
Ogb-Products	co-purchasing	47	100	2449029	61859140

4.1 Datasets Overview

The statistics of the datasets used in our experiments are summarized in Tables 1,2. We have selected datasets from various domains and scales, encompassing both homophily and heterophily graphs.

Table 2: Dataset statistics of heterophilic graphs

Dataset	Nodes	Edges	Classes	Node Features
Roman-empire	22662	32927	18	300
Wiki-cooc	10000	2243042	5	100
Minesweeper	10000	39402	2	7
Questions	48921	153540	2	301
Workers	11758	519000	2	10
Amazon-ratings	24492	93050	5	300

4.2 Node Classification on Homophilic Dataset

In our study, we follow the experimental setup outlined in the GRAND [10] and GraphCON [12] papers, conducting experiments on homophilic datasets to assess the effectiveness of the RL-GRAND and RL-GraphCON models. We adopt the same dataset splitting method as in [10], using the Largest Connected Component (LCC) and performing random splits. Additionally, our study extends to tree-structured datasets, such as Disease and Airport [27], benchmarks in the hyperbolic GNN domain, to highlight the distinct benefits of integrating RL-GFDE with the GRAND and GraphCON models. For these tree-structured datasets, we follow the splitting method used in the hyperbolic GNN literature [27], dividing the data into 60% for training, 20% for validation, and 20% for testing.

From Table 3, we observe that our RL-GFDE-based models, RL-GRAND and RL-GraphCON, improve the performance of GRAND and GraphCON in most reported homophilic benchmarks, both in linear and non-linear versions. Interestingly, although GRAND and GraphCON exhibit limitations in handling tree-structured datasets such as Airport and Disease, our RL-GFDE models show remarkable adaptability to these types of data. Specifically, we achieve the best performance on the Airport data and show competitive results on the Disease dataset, whose self-similar and scale-invariant structures align well with fractional dynamics [21, 28, 29]. This suggests that RL-GFDE can operate across scales and capture these structures more comprehensively.

4.3 Graph Classification and Regression

As demonstrated in section A.1, the RL-GFDE framework possesses a unique intrinsic property: its ability to capture long-range dependencies in graph data. To empirically substantiate this capability, we conduct experiments using the Long-Range Graph Benchmark (LRGB) [31]. Our focus is on the Peptides molecular graphs dataset, where we perform graph classification on the Peptides-func dataset and graph regression based on the 3D structure of the peptides in the Peptides-struct dataset. The performance metrics employed are Average Precision (AP) for classification and Mean Absolute Error (MAE) for regression tasks. For baseline comparisons, we reference the results reported in [31].

As shown in Table 4, our RL-GRAND model exhibits strong performance in the Peptides-func graph classification task. Notably, it achieves higher reported AP than the fully connected Transformer-based methods with position encoding (PE), such as LapPE [32] and RWSE [33]. It's important to highlight that our RL-GRAND models do not employ any form of PE, making this achievement particularly significant. In the graph regression task on the Peptides-Struct dataset, RL-GRAND continues to enhance the results compared to the standard GRAND model, demonstrating a performance that is comparable with the state-of-the-art (SOTA) methods.

4.4 Node Classification on Heterophilic Dataset

Following the experimental setup in GREAD [14], we evaluate the performance of our RL-GREAD model on heterophilic graph datasets, namely Chameleon, Squirrel, and Film, as described in the paper [34]. Table 5 reveals that our RL-GREAD model improves the accuracy of GREAD on these three datasets, showcasing the advantages of our RL-GFDE framework. Furthermore, we extend our experimentation to large-scale heterophilic datasets referenced in the CDE paper [15].

Table 3: Node classification results(%). The best outcomes for the GRAND series and the GraphCON series are separately emphasized in bold.

Method	Cora	Citeseer	Pubmed	CoauthorCS	Computer	Photo	CoauthorPhy	Airport	Disease
GCN [1]	81.5±1.3	71.9±1.9	77.8±2.9	91.1±0.5	82.6±2.4	91.2±1.2	92.8±1.0	81.6±0.6	69.8±0.5
GAT [2]	81.8±1.3	71.4±1.9	78.7±2.3	90.5±0.6	78.0±19.0	85.7±20.3	92.5±0.9	81.6±0.4	70.4±0.5
HGCN [27]	78.7±1.0	65.8±2.0	76.4±0.8	90.6±0.3	80.6±1.8	88.2±1.4	90.8±1.5	85.4±0.7	89.9±1.1
GIL [30]	82.1±1.1	71.1±1.2	77.8±0.6	89.4±1.5	–	89.6±1.3	–	91.5±1.7	90.8±0.5
GRAND-l	83.6±1.0	73.4±0.5	78.8±1.7	92.9±0.4	83.7±1.2	92.3±0.9	93.5±0.9	80.5±9.6	74.5±3.4
GRAND-nl	82.3±1.6	70.9±1.0	77.5±1.8	92.4±0.3	82.4±2.1	92.4±0.8	91.4±1.3	90.9±1.6	81.0±6.7
RL-GRAND-l	**85.2±1.2**	**74.6±1.0**	**80.1±1.2**	**92.8±0.3**	**87.4±1.1**	**93.3±0.7**	**94.1±0.3**	**96.2±0.2**	**90.7±1.3**
RL-GRAND-nl	83.2±0.9	74.1±1.5	79.0±1.2	93.0±0.3	85.9±1.7	92.8±1.0	93.5±0.3	93.3±0.7	87.5±3.0
RL-GRAND-l-PT	**85.4±0.9**	**74.8±1.1**	79.1±1.4	92.8±0.5	87.1±1.0	**93.4±0.6**	94.0±0.5	**96.9±0.4**	88.1±2.1
GraphCON-l	81.9±1.7	72.9±2.1	78.8±2.6	92.3±0.3	84.9±0.5	90.8±1.8	93.9±0.4	68.6±2.1	87.5±4.1
GraphCON-nl	84.2±1.3	74.2±1.7	79.4±1.3	88.7±0.9	79.2±1.1	85.5±2.3	93.1±0.3	74.1±2.7	65.7±5.9
RL-GraphCON-l	**85.2±1.0**	**74.2±1.1**	79.6±1.3	**92.8±0.4**	**86.2±0.8**	93.3±1.0	**94.1±0.5**	**96.4±0.6**	**92.1±2.8**
RL-GraphCON-nl	84.2±0.9	73.5±1.2	**80.3±1.7**	90.4±0.6	83.6±2.2	**94.1±0.7**	93.0±0.6	96.3±0.9	86.9±4.0

Table 4: Numerical results for various methods on LRGB tests. The best and the second-best results in each metric are highlighted in **bold** and underlined, respectively.

Method	Peptides-func Test AP ↑	Peptides-Struct Test MAE ↓
GCN [1]	0.5930±0.0023	0.3496±0.0013
GCNII [35]	0.5543±0.0078	0.3471±0.0010
GINE [36]	0.5498±0.0079	0.3447±0.0045
GatedGCN [37]	0.5864±0.0077	0.3420±0.0013
Transformer +LapPE [38]	0.6326±0.0126	**0.2529±0.0016**
SAN+LapPE [39]	0.6384±0.0121	0.2683±0.0043
SAN+RWSE [33]	0.6439±0.0075	0.2545±0.0012
GCN+DRew [40]	0.6996±0.0076	0.2781±0.0028
PathNN [41]	0.6816±0.0026	0.2545±0.0032
GRAND-l	0.6962±0.0015	0.2867±0.0009
RL-GRAND-l	**0.7457±0.0038**	0.2535±0.0010

Table 5: Node classification results(%). We follow the same experimental setting as in the GREAD paper.

Model	Chameleon	Squirrel	Film
GREAD-BS	71.38±1.31	59.22±1.44	37.90±1.17
RL-GREAD-BS	**71.56±1.47**	**60.92±1.21**	**38.29±0.85**

As shown in Table 6, our RL-CDE model achieves higher accuracy than the baseline on 5 out of 6 datasets and achieves comparable results on the remaining dataset with much smaller variance. These results further demonstrate the compatibility of RL-GFDE methods and their scalability to larger heterophilic datasets. It is worth noting that the impact of our RL-GFDE framework is not limited to a specific type of dataset or model. The results demonstrate broad applicability, while also showing that the gains are dataset-dependent rather than uniform across all metrics.

4.5 Solvers

We have introduced two types of numerical solvers for RL-GFDE models. To prove that the choice of numerical solver does not significantly impact the classification accuracy. We present the results of node classification using these two different solvers in Table 3 (RL-GRAND-l and RL-GRAND-l-PT), where we can observe that the results are comparable.

4.6 Oversmoothing Mitigation

One significant advantage that RL-GFDE brings to graph neural ODE models is the ability to mitigate oversmoothing as analysed in section A.2. Although GRAND or GraphCON demonstrate their capability to maintain stable performance even with a large integral time T, here we show that RL-GFDE-based models possess an even stronger ability to mitigate oversmoothing.

In our implementation, we utilized the fixed-step Euler solver with a step size of one from the paper [24] to implement the GRAND model. In this context, the integral time T can be interpreted as corresponding to the concept of layers in GNN models such as GCN or GAT. We present the results for Cora, Citeseer, and Pubmed in Figure 1. In these experiments, we employ the original data split from [1], which does not use the LCC as in Table 3. From Figure 1, we observe that our RL-GRAND maintains stronger accuracy than GRAND even when the integral time is increased to 256. In contrast, the GRAND model succumbs to oversmoothing and exhibits poor performance under these conditions. This supports our findings in section A.2, which indicate that the RL-GFDE exhibits a solution with slower convergence rate. This in turn reduces the convergence of node embeddings in deeper layers.

To illustrate the impact of RL-GFDE, we have included results for GCN and GAT models augmented with self-connections (SC) in Table 7. As indicated by the results, both GCN-SC and GAT-SC are unable to effectively counteract the oversmoothing issue. This underscores the unique capability of the RL-GFDE approach in mitigating oversmoothing problems in GNNs.

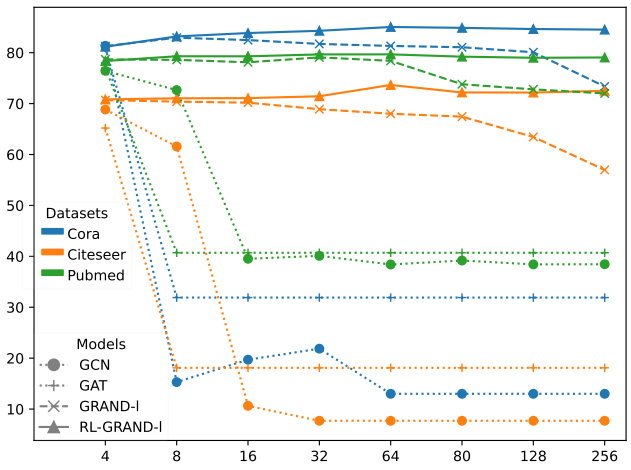

Figure 1: Test accuracy vs. model depth

4.7 Training with limited labeled data

We introduce the RL-GRAND++ model, an extension of GRAND++ [11] within the RL-GFDE framework, designed to address the challenges posed by training with limited labeled data. The datasets utilized in our experiments are summarized in Section 4.1.

In our experiments, we follow the approach in the GRAND++ paper [11], where the authors evaluated their model's performance under limited training label rates of 1/2/5/10/20 samples per class. These tests were conducted on several homophilic datasets.

We directly report the results of GRAND++ from their paper. As presented in Table 8, our RL-GRAND++ model improves over the original GRAND++ in most test scenarios, achieving performance enhancements of up to 6%.

Crucially, as previously highlighted, our RL-GFDE framework defaults to a conventional neural ODE model when α is set to 1, ensuring that our model achieves comparable, if not superior, results on various datasets using a traditional neural ODE solver. In instances where $\alpha = 1$ is determined to be optimal, we report the results from GRAND++.

These experimental results show that incorporating the RL-GFDE framework into the GRAND++ model enhances its performance across a variety of datasets, even when training samples are scarce. This highlights the RL-GFDE framework's capacity to improve the performance of graph neural ODE models in a range of tasks and datasets.

4.8 Model complexity

It should be emphasized that our RL-GFDE framework does not introduce any additional training parameters to the backbone graph neural ODE models. Instead, we simply modify the integration method from standard integration to fractional integration. The time complexity of our framework is $\mathcal{O}(CE\log(E))$, where C represents the time complexity of evaluating $\mathscr{F}(\mathscr{E},\mathbf{X})$ in Equation 11, and $E = T/h$ denotes the number of discretization steps. When $\mathscr{F}$ is given by Equations 12,14,15,16, $C = O(|\mathscr{E}|d)$, with $|\mathscr{E}|$ being the size

Table 6: Node classification results(%). We follow the same experimental setting as in the CDE paper.

Model	Roman-empire	Wiki-cooc	Minesweeper	Questions	Workers	Amazon-ratings
CDE	91.64±0.28	98.04±0.35	**95.50±5.32**	75.17±0.99	80.70±1.04	47.63±0.43
RL-CDE	**92.94±1.13**	**98.87±0.30**	94.68±0.61	**76.24±0.43**	**81.70±1.15**	**48.61±0.54**

Table 7: Self-connections (SC) baselines

Dataset	Model	4	8	16	32	64	80	128	256
Cora	GCN-SC	72.48±1.0	57.39±0.03	15.10±1.41	17.48±2.84	16.43±4.8	15.82±2.95	14.34±2.88	13.0±0.0
	GAT-SC	78.99±0.93	72.75±2.78	11.26±3.51	15.10±8.45	13.00±0.0	13.00±0.0	13.00±0.0	13.00±0.0

Table 8: Node classification results(%) on different training sizes, per class refers to the number of training samples per class. We follow the same experimental setting as in the GRAND++ paper.

Model	per class	Cora	Citeseer	Pubmed	CoauthorCS	Computer	Photo
GRAND++	1	54.94±16.09	58.95±9.59	65.94±4.87	60.30±1.50	67.65±0.37	83.12±0.78
RL-GRAND++	1	**57.27±8.81**	**59.11±6.73**	**65.98±2.72**	**66.38±2.33**	67.65±0.37	83.12±0.78
GRAND++	2	66.92±10.04	**64.98±8.31**	69.31±4.87	76.53±1.85	74.47±1.48	83.71±0.90
RL-GRAND++	2	**70.76±8.00**	**64.11±4.67**	**69.37±5.96**	**82.86±1.48**	**78.56±0.65**	83.71±0.90
GRAND++	5	77.80±4.46	70.03±3.63	71.99±1.91	84.83±0.84	82.64±0.56	88.33±1.21
RL-GRAND++	5	**80.02±1.97**	**70.47±2.04**	**74.22±3.40**	**89.69±0.35**	82.64±0.56	**88.54±0.79**
GRAND++	10	80.86±2.99	72.34±2.42	75.13±3.88	86.94±0.46	82.99±0.81	90.65±1.19
RL-GRAND++	10	**82.52±1.05**	**73.29±2.18**	**77.30±2.43**	**90.88±0.28**	**83.18±0.69**	**91.10±0.57**
GRAND++	20	82.95±1.37	73.53±3.31	79.16±1.37	90.80±0.34	85.73±0.50	93.55±0.38
RL-GRAND++	20	**84.71±1.0**	**73.77±1.95**	**79.93±1.29**	**92.24±0.14**	85.73±0.50	93.55±0.38

of the edge set and d denoting the dimensionality of the features. From Table 9, we observe that the inference time of the RL-GFDE model exhibits only a slight increase compared to that of the neural ODE models, while the GPU memory usage remains comparable. These results demonstrate the scalability of the RL-GFDE framework. The memory cost of the fractional solvers deserves separate discussion. Because Equations 17 and 18 aggregate historical states, a direct implementation stores the previous hidden states $\mathbf{X}(t_0),\ldots,\mathbf{X}(t_{k-1})$. This introduces an activation-state storage cost that scales as $O(E|\mathcal{V}|d)$, where $E = T/h$ is the number of discretization steps. For long integrations, such as the oversmoothing study with $T = 256$ and $h = 1$, this historical-state buffer is a meaningful memory trade-off even though no new trainable parameters are introduced. Practical memory reductions include limiting the history window, checkpointing intermediate states, using lower-precision state buffers, and adopting more efficient convolutional or recursive implementations of the fractional history term.

4.9 Large Scale Ogb-Products Dataset

As mentioned in Section 4.8, our RL-GFDE models do not introduce any new training parameters into the backbone graph neural models. To demonstrate the scalability of RL-GFDE to large-scale datasets, we extended our evaluation to the Ogbn-products dataset, adhering to the experimental settings outlined in [42]. For efficient handling of this large dataset, we employed a mini-batch training approach, which involves sampling nodes and constructing subgraphs, as proposed by GraphSAINT [43]. From the results presented in Tbale 10, it is evident that our RL-GRAND-l model performs slightly better than the GRAND-l model on the Ogb-Products dataset. This demonstrates that our RL-GFDE approach can still enhance graph neural ODE models even in large-scale datasets.

4.10 Impact of α

In our RL-GFDE model, the fractional order α serves as a critical hyperparameter that can be fine-tuned for each specific dataset. As demonstrated in Figure 2, the optimal value of α varies across different datasets for the RL-GRAND-l model. Notably, for all datasets, the optimal α values are found to be less than 1. For practical tuning, we treat $\alpha = 1$ as the integer-order ODE baseline and search $\alpha \in (0,1]$ on the validation set. A coarse grid, for example values spaced by 0.1, identifies a useful range; when the validation curve is sensitive, a narrower local search then refines the best coarse value. For cost-sensitive settings, a small candidate set such as $\{0.5, 0.7, 0.9, 1.0\}$ provides a low-cost validation protocol for selecting the fractional order.

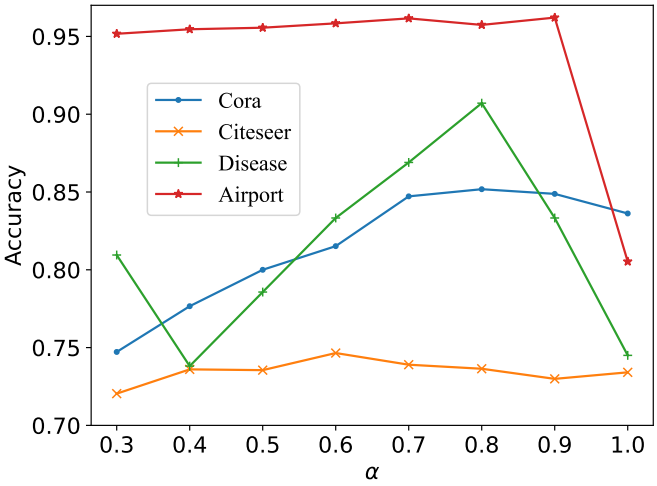

Figure 2: Test accuracy vs. α

4.11 Time Complexity

We present additional numerical results to compare the time consumption between RL-GFDE and its ODE counterparts. Table 11 details the comparison of training time costs, while Table 12 provides a comparison of inference times across various datasets. These tables reveal that the RL-GFDE framework offers time costs comparable to those of the base models, with almost identical GPU memory consumption. This underscores the efficiency of the RL-GFDE framework.

5 Discussion

Our work introduces the RL-GFDE framework. The primary significance of this approach lies in its ability to model non-local dynamics and long-range memory effects. By capturing the historical trajectory of node features, RL-GFDE mitigates the oversmoothing problem prevalent in deep GNNs, offering empirical gains in the reported long-range dependency tasks. However, this benefit comes with limitations. Current implementations rely on numerical approximations, which incur higher computational complexity compared to standard ODE solvers as the depth increases. They also require memory to store or reconstruct historical hidden states. Future research should focus on developing more efficient solvers to reduce computational overhead.

Conclusions: We propose the continuous RL-GFDE framework for graph learning, a novel extension of Graph Neural ODEs. Our work effectively captures feature updating dynamics with pronounced memory effects. We prove

Table 9: Inference time of models on the Cora dataset: integral time $T = 10$ and step size of 1.

Model	RL-GRAND-l	GRAND-l	RL-GraphCON-l	GraphCON-l
Inf. Time(s)	2.23	1.89	3.51	3.20
GPU Memory(MB)	1120	1110	1128	1136

Table 10: Node classification accuracy(%) on Ogb-products dataset

Model	MLP	Node2vec	Full-batch GCN	GraphSAGE	GRAND-l	RL-GRAND-l
Acc	61.06±0.08	72.49±0.10	75.64±0.21	78.29±0.16	75.56±0.67	76.61±0.78

Table 11: Training time of models per epoch on the Cora.

Model	RL-GraphCON-l	GraphCON-l	RL-GRAND-l	GRAND-l
Train. Time(ms)	44.33	43.05	19.72	13.53
GPU Memory(MB)	1340	1158	1192	1102

Table 12: Time and memory consumption on more datasets.

Dataset	Model	RL-GRAND-l	GRAND-l	RL-GraphCON-l	GraphCON-l
Computers	Inf. Time	14.76	13.54	17.79	14.85
	GPU Memory	1854	1474	2218	1598
Airport	Inf. Time	3.04	1.91	3.77	2.55
	GPU Memory	1164	1092	1242	1110

the framework's capability to mitigate oversmoothing. We also validate the integration of RL-GFDE into various graph neural ODE models.

Funding

This work is supported by the National Natural Science Foundation of China under Grants 62576326 and 12301491.

Author Contributions

Conceptualization, Yufeng Peng and Qiyu Kang; Methodology, Yufeng Peng and Qiyu Kang; Software, Yufeng Peng and Kai Zhao; Validation, Yufeng Peng, Qiyu Kang, and Xuhao Li; Formal Analysis, Qiyu Kang and Kai Zhao; Investigation, Yufeng Peng and Qiyu Kang; Resources, Qinxu Ding; Data Curation, Kai Zhao and Xuhao Li; Writing—Original Draft Preparation, Yufeng Peng and Qiyu Kang; Writing—Review and Editing, All authors; Visualization, Yufeng Peng and Kai Zhao; Supervision, Qinxu Ding and Qiyu Kang; Project Administration, Qinxu Ding; Funding Acquisition, Qiyu Kang.

Conflict of Interest

All the authors declare that they have no conflict of interest.

Data Available

All datasets used in this study are public citation network datasets, openly available via standard graph learning libraries [44].

References

[1] Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. In 5th International Conference on Learning Representations, pp. 1–14 (2017)

[2] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. In 6th International Conference on Learning Representations, pp. 1–12 (2018)

[3] Hamilton, W., Ying, Z., Leskovec, J.: Inductive Representation Learning on Large Graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 1025–1035 (2017)

[4] Shi, W., Rajkumar, R.: Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1711–1719 (2020). https://doi.org/10.1109/CVPR42600.2020.00178

[5] Reiser, P., Neubert, M., Eberhard, A., Torresi, L., Zhou, C., Shao, C., Metni, H., Hoesel, C., Schopmans, H., Sommer, T., *et al.*: Graph neural networks for materials science and chemistry. Communications Materials **3**(1), 93 (2022). https://doi.org/10.1038/s43246-022-00315-6

[6] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D.: Graph neural networks for social recommendation. In The World Wide Web Conference, pp. 417–426 (2019). https://doi.org/10.1145/3308558.3313488

[7] Han, A., Shi, D., Lin, L., Gao, J.: From Continuous Dynamics to Graph Neural Networks: Neural Diffusion and Beyond. arXiv preprint arXiv:2310.10121 (2023). https://doi.org/10.48550/arXiv.2310.10121

[8] Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., Park, J.: Graph neural ordinary differential equations. arXiv preprint arXiv:1911.07532 (2019). https://doi.org/10.48550/arXiv.1911.07532

[9] Xhonneux, L.-P., Qu, M., Tang, J.: Continuous graph neural networks. In Proceedings of the 37th International Conference on Machine Learning, pp. 10432–10441 (2020)

[10] Chamberlain, B.P., Rowbottom, J., Goronova, M., Webb, S., Rossi, E., Bronstein, M.M.: GRAND: Graph Neural Diffusion. In Proceedings of the 38th International Conference on Machine Learning, pp. 1401–1418 (2021)

[11] Thorpe, M., Xia, H., Nguyen, T., Strohmer, T., Bertozzi, A., Osher, S., Wang, B.: GRAND++: Graph neural diffusion with a source term. In The Tenth International Conference on Learning Representations, pp. 1–21 (2022)

[12] Rusch, T.K., Chamberlain, B., Rowbottom, J., Mishra,

S., Bronstein, M.: Graph-coupled oscillator networks. In Proceedings of the 39th International Conference on Machine Learning, pp. 18888–18909 (2022)

[13] Song, Y., Kang, Q., Wang, S., Zhao, K., Tay, W.P.: On the Robustness of Graph Neural Diffusion to Topology Perturbations. In Proceedings of the 36th International Conference on Neural Information Processing Systems, pp. 6384–6396 (2022)

[14] Choi, J., Hong, S., Park, N., Cho, S.-B.: GREAD: Graph Neural Reaction-Diffusion Equations. In Proceedings of the 40th International Conference on Machine Learning, pp. 5722–5747 (2023)

[15] Zhao, K., Kang, Q., Song, Y., She, R., Wang, S., Tay, W.P.: Graph Neural Convection-Diffusion with Heterophily. In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, pp. 4656–4664 (2023). https://doi.org/10.24963/ijcai.2023/518

[16] Zhao, K., Kang, Q., Song, Y., She, R., Wang, S., Tay, W.P.: Adversarial Robustness in Graph Neural Networks: A Hamiltonian Approach. In 37th Conference on Neural Information Processing Systems (NeurIPS 2023), pp. 3338–3361 (2023)

[17] Baleanu, D., Diethelm, K., Scalas, E., Trujillo, J.J.: Fractional Calculus: Models and Numerical Methods in Series on Complexity, Nonlinearity and Chaos. World Scientific, Singapore (2012). https://doi.org/10.1142/8180

[18] Metzler, R., Klafter, J.: The random walk's guide to anomalous diffusion: a fractional dynamics approach. Physics reports **339**(1), 1–77 (2000). https://doi.org/10.1016/S0370-1573(00)00070-3

[19] Yang, Q., Chen, D., Zhao, T., Chen, Y.: Fractional calculus in image processing: a review. Fractional Calculus and Applied Analysis **19**(5), 1222–1249 (2016). https://doi.org/10.1515/fca-2016-0063

[20] Tarasov, V.E., Zaslavsky, G.M.: Fractional Ginzburg–Landau equation for fractal media. Physica A: Statistical Mechanics and its Applications **354**, 249–261 (2005). https://doi.org/10.1016/j.physa.2005.02.047

[21] Tarasov, V.E.: Fractional dynamics: applications of fractional calculus to dynamics of particles, fields and media. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-14003-7

[22] Cai, C., Wang, Y.: A Note on Over-Smoothing for Graph Neural Networks. arXiv preprint arXiv 2006.13318 (2020). https://doi.org/10.48550/arXiv.2006.13318

[23] Diethelm, K.: The analysis of fractional differential equations: an application-oriented exposition using differential operators of Caputo type. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14574-2

[24] Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 6572–6583 (2018)

[25] Elliott, D.: An asymptotic analysis of two algorithms for certain Hadamard finite-part integrals. IMA journal of numerical analysis **13**(3), 445–462 (1993). https://doi.org/10.1093/imanum/13.3.445

[26] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. MIT Press, Cambridge, Massachusetts, USA (2022)

[27] Chami, I., Ying, Z., Ré, C., Leskovec, J.: Hyperbolic graph convolutional neural networks. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pp. 4868–4879 (2019)

[28] Kim, J., Goh, K.-I., Salvi, G., Oh, E., Kahng, B., Kim, D.: Fractality in complex networks: Critical and supercritical skeletons. Physical Review E **75**(1), 016110 (2007). https://doi.org/10.1103/PhysRevE.75.016110

[29] Nigmatullin, R.: Fractional integral and its physical interpretation. Theoretical and Mathematical Physics **90**(3), 242–251 (1992). https://doi.org/10.1007/BF01036529

[30] Zhu, S., Pan, S., Zhou, C., Wu, J., Cao, Y., Wang, B.: Graph Geometry Interaction Learning. In Proceedings of the 34th International Conference on Neural Information Processing Systems, pp. 7548–7558 (2020)

[31] Dwivedi, V.P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A.T., Beaini, D.: Long Range Graph Benchmark. In Proceedings of the 36th International Conference on Neural Information Processing System, pp. 22326–22340 (2023)

[32] Dwivedi, V.P., Joshi, C.K., Luu, A.T., Laurent, T., Bengio, Y., Bresson, X.: Benchmarking Graph Neural Networks. The Journal of Machine Learning Research **24**(1), 1730–1777 (2022)

[33] Dwivedi, V.P., Luu, A.T., Laurent, T., Bengio, Y., Bresson, X.: Graph Neural Networks with Learnable Structural and Positional Representations. In The Tenth International Conference on Learning Representations (ICLR 2022), pp. 1–25 (2022)

[34] Pei, H., Wei, B., Chang, K.C.-C., Lei, Y., Yang, B.: Geom-GCN: Geometric Graph Convolutional Networks. In International Conference on Learning Representations (ICLR 2020), pp. 1–12 (2019)

[35] Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y.: Simple

and deep graph convolutional networks. In Proceedings of the 37th International Conference on Machine Learning, pp. 1725–1735 (2020)

[36] Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., Leskovec, J.: Strategies for Pre-training Graph Neural Networks. In International Conference on Learning Representations(ICLR 2020), pp. 1–22 (2020)

[37] Bresson, X., Laurent, T.: Residual Gated Graph ConvNets. In International Conference on Learning Representations(ICLR 2018), pp. 1–11 (2018)

[38] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 6000–6010 (2017)

[39] Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., Tossou, P.: Rethinking graph transformers with spectral attention. In Proceedings of the 35th International Conference on Neural Information Processing Systems, pp. 21618–21629 (2021)

[40] Gutteridge, B., Dong, X., Bronstein, M.M., Di Giovanni, F.: Drew: Dynamically rewired message passing with delay. In Proceedings of the 40th International Conference on Machine Learning, pp. 12252–12267 (2023)

[41] Michel, G., Nikolentzos, G., Lutzeyer, J.F., Vazirgiannis, M.: Path neural networks: Expressive and accurate graph neural networks. In Proceedings of the 40th International Conference on Machine Learning, pp. 24737–24755 (2023)

[42] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs (2020)

[43] Zeng, H., Zhou, H., Srivastava, A., Kannan, R., Prasanna, V.: GraphSAINT: Graph Sampling Based Inductive Learning Method. In 8th International Conference on Learning Representations (ICLR 2020), pp. 1–19 (2020)

[44] Fey, M., Sunil, J., Nitta, A., Puri, R., Shah, M., Stojanović, B., Bendias, R., Barghi, A., Kocijan, V., Zhang, Z., He, X., Lenssen, J.E., Leskovec, J.: PyG 2.0: Scalable Learning on Real World Graphs. In Temporal Graph Learning Workshop @ KDD 2025, pp. 1–9 (2025)

[45] Feng, J., Chen, Y., Li, F., Sarkar, A., Zhang, M.: How Powerful are K-hop Message Passing Graph Neural Networks. In Thirty-Sixth Conference on Neural Information Processing Systems (NeurIPS 2022), pp. 1–15 (2022)

[46] Oono, K., Suzuki, T.: Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In 8th International Conference on Learning Representations (ICLR 2020), pp. 1–37 (2020)

[47] Chung, F.R.: Spectral graph theory. American Mathematical Society, Providence, Rhode Island, USA (1997)

[48] Li, G., Muller, M., Thabet, A., Ghanem, B.: DeepGCNs: Can GCNs Go As Deep As CNNs? In 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9267–9276 (2019). https://doi.org/10.1109/ICCV.2019.00936

[49] Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In Proceedings of the 35 Th International Conference on Machine Learning, pp. 5453–5462 (2018)

[50] Li, G., Xiong, C., Thabet, A., Ghanem, B.: DeeperGCN: All You Need to Train Deeper GCNs. In The Tenth International Conference on Learning Representations (ICLR 2022), pp. 1–18 (2022)

[51] Jin, B., Li, B., Zhou, Z.: Numerical analysis of nonlinear subdiffusion equations. SIAM Journal on Numerical Analysis **56**(1), 1–23 (2018). https://doi.org/10.1137/16M1089320

[52] Diethelm, K.: An algorithm for the numerical solution of differential equations of fractional order. Electronic Transactions on Numerical Analysis **5**, 1–6 (1997)

[53] Podlubny, I.: Fractional Differential Equations. Academic Press, San Diego, CA, USA (1999)

[54] Horn, R.A., Johnson, C.R.: Matrix analysis. Cambridge university press, New York, USA (2012)

Appendix A Why RL-GFDE Improves Graph Neural ODE Models

A.1 Incorporation of Memory and History

To analyze how graph neural FDE models enhance the performance of traditional graph neural ODE models by extending integer orders to fractional orders, we examine a specific case study. We delve into the capabilities of both RL-GRAND and GRAND, comparing and contrasting their capabilities. In the implementation of both RL-GRAND and GRAND models, leveraging discrete numerical solvers is essential to transition from continuous to discrete models. We begin by presenting the discrete version of the linear GRAND model, employing the widely used explicit Euler method.

$$\mathbf{X}(t_k) = (\mathbf{A} - \mathbf{I})(\mathbf{X}(t_{k-1})) + \tau\mathbf{X}(t_{k-1}) \tag{A.1}$$

where τ is the step size. The linear GRAND model essentially functions as a 1-hop message passing framework, akin to the one formulated in [45].

In the context of RL-GRAND, particularly for the linear case, we can reformulate the Grünwald-Letnikov solver presented in Equation 17 to adapt it specifically for the RL-GRAND model as follows:

$$\mathbf{X}(t_k) = w_1(\mathbf{A}-\mathbf{I})\mathbf{X}(t_{k-1}) + \sum_{j=1}^{k} w_j \mathbf{X}(t_{k-j}) \tag{A.2}$$

where w_1 and w_j are predetermined coefficients representing the weights of the current and past states, respectively. The adapted solver allows the model to take into account not just the immediate past state of the nodes but a weighted sum of several past states, providing a richer context and capturing more complex dependencies within the graph. This ability to integrate a broader temporal context is particularly beneficial in complex graphs where the node's label is dependent on long-range interactions or temporal patterns.

A.2 Mitigation of Oversmoothing

The effectiveness of GNNs in node classification significantly decreases due to inherent limitations in their design and capabilities. The seminal research [46][Corollary 3. and Remark 1] has highlighted that, when considering a GNN as a layered dynamical system, oversmoothing is a broad expression of the *exponential convergence* to stationary states that only retain information about graph connected components and node degrees. The memoryless graph neural ODE model, GRAND-l, is demonstrated to approach asymptotic stationary states as shown in [11, 16], with a known *exponentially rapid convergence rate* [47]. In contrast, the fractional memory-dependent dynamic model, RL-GRAND-l, as described in Equation 12 with a specific linear setting, converges to stationary states at a *slow algebraic rate*, thereby helping to mitigate oversmoothing. The findings are detailed in Theorem 3. In practical applications with finite time horizons, this reduced rate of convergence is preferable when deep models must extract distinct features rather than converge rapidly to stationary states.

Theorem 3 *Assuming the graph is strongly connected and aperiodic, the solution* $\mathbf{X}(t)$ *to Equation 12 with a specific linear setting converges to a stationary state at a slow algebraic rate* $\Theta(t^{-\kappa})$*, for some* $\kappa > 0$*, provided that the initial condition* $\mathbf{X}(0)$ *differs from the stationary state and* $0 < \alpha < 1$.

Remark 1 The above theorem demonstrates how RL-GFDE analytically mitigates oversmoothing, particularly through example Equation 12. In model discretization Equations 17 and 18, it is apparent that all previous layers $(\mathbf{X}(0),\ldots,\mathbf{X}(t_{k-1}))$ exert a direct influence on $\mathbf{X}(t_k)$. This shows that initial features receive direct historical contributions, unlike a formulation that uses only a skip connection to the preceding layer $\mathbf{X}(t_{n-1})$. Therefore, dense layer connections yield a slower asymptotic convergence rate than the typical skip layer connections in graph ODE model discretization. In GNN literature, both skipping and dense connections enhance model performance at increased layer depths. The primary cause of performance degradation—oversmoothing or vanishing gradients—remains debatable. Using a continuous analog approach, we conclude that oversmoothing, especially with only skip connections, adversely affects performance. Theoretically, dense connections between layers mitigate the shift from exponential to slower algebraic convergence, which reduces oversmoothing, as supported by empirical studies like [35, 48–50].

Appendix B Review of Riemann–Liouville Fractional Derivative

In this section, we provide a succinct introduction to the Riemann–Liouville (RL) fractional derivative and discuss its generalization over traditional integer-order derivatives.

Recalling the conventional integer-order derivative and integration:

- first-order derivative: By D, we denote the operator that maps a differentiable function onto its derivative,
$$Df(x) := f'(x)$$
- first-order integration: By J_a, we denote the operator that maps a function f, assumed to be (Riemann) integrable on the compact interval $[a,b]$, onto its primitive centered at a,
$$J_a f(x) := \int_a^x f(t)\mathrm{d}t \quad \text{for } a \le x \le b.$$
- high integer order derivative/integration: For $n \in \mathbb{N}$ we use the symbols D^n and J_a^n to denote the n-fold iterates of D and J_a, respectively, i.e. we set $D^1 := D, J_a^1 := J_a$, and
$$D^n := DD^{n-1} \text{ and } J_a^n := J_a J_a^{n-1} \text{ for } n \ge 2$$

From [23, Lemma 1.1.], we have that: let f be Riemann integrable on $[a,b]$, then for $a \le x \le b$ and $n \in \mathbb{N}$, we have

$$J_a^n f(x) = \frac{1}{(n-1)!}\int_a^x (x-t)^{n-1} f(t)\mathrm{d}t, \quad n \in \mathbb{N} \tag{B.1}$$

The Riemann–Liouville (RL) fractional integral operator aims to generalize the integer-order integration by extending $n \in \mathbb{N}$ in Equation B.1 to encompass positive real numbers. Let $n \in \mathbb{R}_+$, representing the set of positive real numbers. The operator ${}^{\mathrm{RL}}J_a^n$, defined on $L_1[a,b]$ by

$${}^{\mathrm{RL}}J_a^n f(x) := \frac{1}{\Gamma(n)}\int_a^x (x-t)^{n-1} f(t)\mathrm{d}t, \quad n \in \mathbb{R}_+ \tag{B.2}$$

for $a \le x \le b$, is called the RL fractional integral operator of order n. For $n = 0$, we set $J_a^0 := I$, the identity operator. We clearly see that Equation B.2 generalizes Equation B.1.

Recall that for integer order derivatives and integrations, we have the following relation: [23, Lemma 1.2.] Let $m, n \in \mathbb{N}$ such that $m > n$, and let f be a function having a continuous n-th derivative on the interval $[a,b]$. Then,

$$D^n f = D^m J_a^{m-n} f, \quad n \in \mathbb{N} \tag{B.3}$$

Building upon Equation B.3, we generalize to the RL derivative for $n \in \mathbb{R}_+$. Specifically, let $n \in \mathbb{R}_+$and define $m = \lceil n \rceil$, which is the smallest integer greater than or equal to n. The operator ${}^{\mathrm{RL}}D_a^n$ is then defined as:

$${}^{\mathrm{RL}}D_a^n f := D^m J_a^{m-n} f, \quad n \in \mathbb{R}_+ \tag{B.4}$$

is called the RL fractional differential operator of order n. For $n = 0$, we set $D_a^0 := I$, the identity operator.

For a clearer understanding of the RL fractional derivative, we present the following examples:

Let $f(x) = (x-a)^\beta$ for some $\beta > -1$ and $n > 0$, from definition Equation B.4, we have that

$$ {}^{\mathrm{RL}}D_a^n f(x) = D^{\lceil n\rceil} J_a^{\lceil n\rceil - n} f(x) = \frac{\Gamma(\beta+1)}{\Gamma(\lceil n\rceil - n + \beta + 1)} D^{\lceil n\rceil}\left[(\cdot - a)^{\lceil n\rceil - n + \beta}\right](x). $$

1. if $n - \beta \in \mathbb{N}$, the right-hand side is the $\lceil n\rceil$-th derivative of a classical polynomial of degree $\lceil n\rceil - (n-\beta) \in \{0,1,\ldots,\lceil n\rceil - 1\}$, and so the expression vanishes, i.e.

   $${}^{\mathrm{RL}}D_a^n\left[(\cdot - a)^{n-m}\right](x) = 0 \text{ for all } n > 0, m \in \{1,2,\ldots,\lceil n\rceil\}.$$

2. if $n - \beta \notin \mathbb{N}$, we find

$$ {}^{\mathrm{RL}}D_a^n\left[(\cdot - a)^\beta\right](x) = \frac{\Gamma(\beta+1)}{\Gamma(\beta+1-n)}(x-a)^{\beta-n}. \quad \text{(B.5)} $$

Both these relations are straightforward generalizations of what we know for integer-order derivatives.

Appendix C Implementation Details

C.1 RL-GRAND, RL-GraphCON, and RL-GRAND++

We evaluate both linear (l) and non-linear (nl) variants of RL-GRAND and RL-GraphCON, along with RL-GRAND++.

RL-GRAND and Variants: The general formulation follows:

$$ {}^{\mathrm{RL}}D^\alpha \mathbf{X}(t) = (\mathbf{A}(\mathbf{X}(t)) - \mathbf{I})\mathbf{X}(t). \quad \text{(C.1)} $$

For the non-linear variant (RL-GRAND-nl), we adopt the Transformer-style attention mechanism [38] where $\mathbf{A}(\mathbf{X}(t))$ is dynamic. The linear variant (RL-GRAND-l) utilizes a static normalized Laplacian $\mathbf{L}$ derived from the initial features:

$$ {}^{\mathrm{RL}}D^\alpha \mathbf{X}(t) = -\mathbf{L}\mathbf{X}(t). \quad \text{(C.2)} $$

RL-GraphCON: We extend GraphCON to its RL-fractional form:

$$ \begin{aligned} {}^{\mathrm{RL}}D^\alpha \mathbf{Y}(t) &= \sigma(\mathbf{F}_\theta(\mathbf{X}(t),t)) - \gamma\mathbf{X}(t) - \alpha\mathbf{Y}(t), \\ {}^{\mathrm{RL}}D^\alpha \mathbf{X}(t) &= \mathbf{Y}(t). \end{aligned} \quad \text{(C.3)} $$

Here, $\mathbf{F}_\theta$ employs either the static adjacency (linear) or the attention-based dynamic matrix (non-linear) as defined in RL-GRAND. Consistent with [12], no activation is applied in the linear implementation.

RL-GRAND++: Extending the source term framework [11], RL-GRAND++ is formulated as:

$$ {}^{\mathrm{RL}}D^\alpha \mathbf{X}(t) = -\mathbf{L}(\mathbf{X}(t)) + \mathbf{C}(0), \quad \text{(C.4)} $$

integrating the RL-GFDE framework to handle limited labeled data scenarios.

C.2 RL-CDE

In our implementation of CDE, we refer to CDE as CDE-GRAND-GAT [15]. The formulation for RL-CDE is as follows:

$$ {}^{\mathrm{RL}}D^\alpha \mathbf{X}(t) = (\mathbf{A}(\mathbf{X}(t)) - \mathbf{I})\mathbf{X}(t) + \mathrm{div}(\mathbf{V}(t) \circ \mathbf{X}(t)) \quad \text{(C.5)} $$

where

$$ (\mathrm{div}(\mathbf{V}(t) \circ \mathbf{X}(t)))_i := \sum_{j:(i,j)\in\mathscr{E}} \mathbf{V}_{ij}(t) \odot \mathbf{x}_j(t) \quad \text{(C.6)} $$

for each node $i \in \mathscr{V}$.

$$ \mathbf{V}_{ij} = \sigma\left(W(\mathbf{x}_j(t) - \mathbf{x}_i(t))\right), \quad \text{(C.7)} $$

$$ a(\mathbf{x}_i, \mathbf{x}_j) = \frac{\exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_j]\right)\right)}{\sum_{k\in\mathscr{N}i} \exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{W}\mathbf{x}i \| W\mathbf{x}_k]\right)\right)}, \quad \text{(C.8)} $$

where $\mathbf{W}$ and $\mathbf{a}$ are learned and $\|$ is the concatenation operator.

C.3 RL-GREAD

In line with the experimental methodology detailed in GREAD study [14], our experiments were specifically tailored to heterophilic datasets. We implemented the Blurring-Sharpening (BS) reaction term in both GREAD-BS and our fractional adaptation, RL-GREAD-BS, inspired by the good performance of GREAD-BS highlighted in Table 4 of the GREAD study [14].

Appendix D Theorem Proofs

D.1 Convergence Analysis

In this section, we concentrate on the 1-dimensional case, which suffices for our proof, using the non-bold notation X. The oracle solution of the system is denoted as $\bar{X}(t)$. To improve clarity in this section, we adopt the notation X_j with an iterative subscript, a departure from $X(t_j)$ used before, to distinctly present the solutions obtained from the solvers.

D.1.1 Convergence of GL predictor

We first give consistency error of *Grünwald-Letnikov* approximation:

Theorem 4 *(Consistency error of GL predictor) Let $f \in C^2[0,T]$. Then*

$$ \left|{}^{\mathrm{RL}}D^\alpha[f](x_k) - {}^{\mathrm{RL}}_h D^\alpha_{GL}[f](x_k)\right| = O(h) $$

uniformly for all k.

Then we can analyze convergence of Equation 17.

Proof We discuss nonlinear case only. Result for linear case can be similarly derived.

Firstly, using Theorem 4, it is readily seen that X satisfies

$$ {}^{\mathrm{RL}}_h D^\alpha_{\mathrm{GL}} \bar{X}(t_k) = \mathscr{F}(\bar{X}(t_k)) + O(h). $$

Noticing that Equation 17 is equivalent to

$$ {}^{\mathrm{RL}}_h D^\alpha_{\mathrm{GL}} X_k = \mathscr{F}(X_{k-1}). $$

this combining with above result gives

$$ \begin{aligned} {}^{\mathrm{RL}}_h D^\alpha_{\mathrm{GL}} e_k \le\ & |\mathscr{F}(\bar{X}(t_k)) - \mathscr{F}(\bar{X}(t_{k-1}))| \\ & + |\mathscr{F}(\bar{X}(t_{k-1})) - \mathscr{F}(X_{k-1})| + O(h), \end{aligned} $$

where $e_k = |\bar{X}(t_k) - X_k|$ and we have used the properties of the coefficients C_j^α.

Since $\mathscr{F}$ is Lipschitz continuous and $X \in C^2[0,T]$, we get

$$ |\mathscr{F}(\bar{X}(t_k)) - \mathscr{F}(\bar{X}(t_{k-1}))| \le L \max_{t\in[0,T]} |X'(t)| h, $$

and

$$|\mathscr{F}(\bar{X}(t_{k-1})) - \mathscr{F}(X_{k-1})| \leq Le_{k-1},$$

where L is Lipschitz constant.

From above formulas, we finally obtain

$${}^{\mathrm{RL}}_{h}D^{\alpha}_{\mathrm{GL}}e_k \leq Le_{k-1} + O(h).$$

By necessary calculations, it is found that conditions of [51, Theorem 2.8] are satisfied. Hence, we have

$$e_j = |\bar{X}(t_j) - X_j| = O(h).$$

□

D.1.2 Convergence of PT predictor

Same as before, We give consistency error of *Product Trapezoidal* approximation:

Theorem 5 *(Consistency error of PT predictor) Let $f \in C^2[0,T]$. Then*

$$\left|{}^{\mathrm{RL}}D^{\alpha}[f](x_k) - {}^{\mathrm{RL}}_{h}D^{\alpha}_{Tr}[f](x_k)\right| = O(h^{2-\alpha})$$

uniformly for all k.

Then we can analyze convergence of Equation 18.

Proof As in the proof of Theorem 5, we consider nonlinear case only.

Let $e_k = |\bar{X}(t_k) - X_k|$. Using Theorem 3 and following similar procedure as in the proof of Theorem 2, we shall get

$${}^{\mathrm{RL}}_{h}D^{\alpha}_{Tr}e_k \leq Le_{k-1} + O(h) + O(h^{2-\alpha}),$$

where L is Lipschitz constant.

Similar to the proof of Theorem 1.1 in [52], let $d_0 = 1$ and

$$d_k = \Gamma(2-\alpha)h^{\alpha}Ld_{k-1} - \Gamma(2-\alpha)\sum_{j=1}^{k-1}A_{jk}d_j, k \geq 1$$

then it is found that

$$e_j \leq d_k O(h^{1+\alpha} + h^2),$$

where

$$d_j \leq (1+Lh)^j \leq exp(LT)$$

by induction. □

D.2 Mitigation of Oversmoothing

Before presenting the formal proof, we intend to offer additional insights and intuition concerning the algebraic convergence in the general high-dimensional case, building upon the discussion of the 1-dimensional case. The notably slow algebraic rate mainly arises from the slow convergence of the Mittag-Leffler function to zero.

To clarify this point, consider the following linear homogeneous RL FDE (Equation 9). It can be demonstrated that the function

$$f(t) = t^{\alpha-1}E_{\alpha,\alpha}(-\lambda t^{\alpha}) \tag{D.1}$$

is a solution of the above linear FDE Equation 9. To see this, we first recall from [23][Example 2.4.] (cf. Equation B.5) that RL fractional derivative of t^n is given by:

$${}^{\mathrm{RL}}D^{\alpha}t^n = \frac{\Gamma(n+1)}{\Gamma(n+1-\alpha)}t^{n-\alpha}.$$

Recall that $E_{\alpha,\beta}(z) = \sum_{k=0}^{\infty}\frac{z^k}{\Gamma(\alpha k+\beta)}$, we can express $f(t)$ as the following power series:

$$f(t) = t^{\alpha-1}\sum_{k=0}^{\infty}\frac{(-\lambda t^{\alpha})^k}{\Gamma(\alpha k+\alpha)} = \sum_{k=0}^{\infty}\frac{(-\lambda)^k t^{\alpha k+\alpha-1}}{\Gamma(\alpha k+\alpha)}. \tag{D.2}$$

Differentiating term-by-term, the RL derivative of $f(t)$ is:

$$\begin{aligned}
{}^{\mathrm{RL}}D^{\alpha}f(t) &= \sum_{k=0}^{\infty}\frac{(-\lambda)^k}{\Gamma(\alpha k+\alpha)}{}^{\mathrm{RL}}D^{\alpha}t^{\alpha k+\alpha-1} \\
&= \sum_{k=0}^{\infty}\frac{(-\lambda)^k}{\Gamma(\alpha k+\alpha)}\frac{\Gamma(\alpha k+\alpha)}{\Gamma(\alpha k)}t^{\alpha k-1} \\
&= \sum_{k=0}^{\infty}\frac{(-\lambda)^k}{\Gamma(\alpha k)}t^{\alpha k-1} \\
&= -\lambda\sum_{k=1}^{\infty}\frac{(-\lambda)^{k-1}}{\Gamma(\alpha k)}t^{\alpha k-1} \\
&= -\lambda\sum_{k=0}^{\infty}\frac{(-\lambda)^k}{\Gamma(\alpha k+\alpha))}t^{\alpha k+\alpha-1} \\
&= -\lambda f(t),
\end{aligned}$$

where the relation $\Gamma(0) = \infty$ is utilized in the fourth equality.

Furthermore, the solution $f(t)$ exhibits a slow algebraic convergence rate $\Theta(t^{-\kappa})$ towards 0 with $\kappa > 0$. We next show that this is true and $\kappa = \alpha + 1$ From [53, Theorem 1.4], as $t \to \infty$,

$$\begin{aligned}
&E_{\alpha,\alpha}(-\lambda t^{\alpha}) \\
=&\left(\frac{(-\lambda t^{\alpha})^{-1}}{\Gamma(\alpha-\alpha)} + \frac{(-\lambda t^{\alpha})^{-2}}{\Gamma(\alpha-2\alpha)}\right)(1+o(1)) \\
=&\Theta(t^{-2\alpha}).
\end{aligned} \tag{D.3}$$

It therefore follows that

$$f(t) = t^{\alpha-1}E_{\alpha,\alpha}(-\lambda t^{\alpha}) = \Theta(t^{-(\alpha+1)}). \tag{D.4}$$

Theorem 3 extends the scalar discussion to a more general high-dimensional case by replacing the scalar λ with the Laplacian matrix $\mathbf{L}$. In this context, the eigenvalues of $\mathbf{L}$ assume a crucial role, analogous to λ in the scalar situation. In cases where the Laplacian matrix $\mathbf{L}$ is diagonalizable, the proof essentially mirrors the scalar case as previously outlined. However, when $\mathbf{L}$ is non-diagonalizable and assumes a general Jordan normal form, it becomes imperative to utilize the Laplace transform technique. This approach is necessary to demonstrate that the solution maintains its algebraic rate of convergence as in the scalar case.

Proof It is evident that for the matrix $\mathbf{WD}^{-1}$ (or $\mathbf{D}^{-1}\mathbf{W}$), given that it is column (or row) stochastic and the graph is strongly connected and aperiodic, the Perron-Frobenius theorem [54, Lemma 8.4.3., Theorem 8.4.4] confirms that the value 1 is the unique eigenvalue of this matrix that equals its spectral radius, which is also 1. Consequently, it follows that the matrix $\mathbf{L} = \mathbf{I} - \mathbf{WD}^{-1}$ (or $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$) has an eigenvalue of 0, with all other eigenvalues possessing positive real parts. Considering the Jordan canonical form of $\mathbf{L}$, denoted as $\mathbf{L} = \mathbf{SJS}^{-1}$, it is observed that $\mathbf{J}$ contains a block that consists solely of a single 0, while the other blocks are characterized by eigenvalues λ_k possessing positive real parts. WLOG, we assume the feature dimension is one and we rewrite Equation C.2 as

$${}^{\mathrm{RL}}D^{\alpha}\mathbf{Y}(t) = -\mathbf{JY}(t) \tag{D.5}$$

where $\mathbf{S}^{-1}\mathbf{X}(t) = \mathbf{Y}(t) \in \mathbb{R}^N$ representing a transformation of the feature space and the transformed initial condition is defined as $\boldsymbol{Y} := \lim_{t\to 0^+} J^{1-\alpha}\mathbf{Y}(t) = \mathbf{S}^{-1}\lim_{t\to 0^+} J^{1-\alpha}\mathbf{X}(t)$.

If the matrix $\mathbf{L}$ is diagonalizable, then its Jordan canonical form $\mathbf{J}$ becomes a diagonal matrix, with the diagonal elements representing the eigenvalues of $\mathbf{L}$. In this scenario, the differential equation can be decoupled into a set of independent equations, each described by

$$^{\mathrm{RL}}D^{\alpha}\mathbf{Y}_k(t) = -\lambda_k \mathbf{Y}_k(t). \tag{D.6}$$

Here, $\mathbf{Y}_k$ signifies the k-th component of the vector $\mathbf{Y}$. According to our discussion for the scalar case Equation 9 above, the solution to each differential equation in the given context is represented as:

$$\mathbf{Y}_k(t) = \boldsymbol{Y}_k t^{\alpha-1} E_{\alpha,\alpha}\left(-\lambda_k t^{\alpha}\right). \tag{D.7}$$

This formulation leads to two important observations:

1. For the index j such that the eigenvalue $\lambda_j = 0$, the solution simplifies to $\mathbf{Y}_j(t) = \boldsymbol{Y}_j t^{\alpha-1} = \Theta(t^{\alpha-1})$.
2. As shown in Equation D.4, for indices $k \neq j$, since λ_k has positive real part, the convergence to zero is characterized by the following order:
$$\mathbf{Y}_k(t) = \Theta(t^{-(\alpha+1)}).$$

Asymptotically, this indicates that all components $\mathbf{Y}_k(t)$ will converge to zero at an algebraic rate.

If the matrix $\mathbf{J}$ is not diagonal, the entries of $\mathbf{Y}(t)$ corresponding to distinct Jordan blocks in $\mathbf{J}$ remain uncoupled. Therefore, it suffices to consider a single Jordan block corresponding to eigenvalue λ_k. For the block $\lambda_k = 0$, we have $\mathbf{Y}_j(t) = \Theta(t^{\alpha-1})$ as discussed in the above Item 1. We next consider the case $\lambda_k \neq 0$. In this case, employing the Laplace transform technique becomes useful for demonstrating that the algebraic rate of convergence remains valid. We assume the Jordan block $\mathbf{J}(\lambda_k)$, associated with λ_k, is of size m. It follows that for this Jordan block we have

$$\begin{aligned}
^{\mathrm{RL}}D^{\alpha}\mathbf{Y}_1(t) &= -\lambda_k \mathbf{Y}_1(t) - \mathbf{Y}_2(t),\\
&\vdots \quad \vdots\\
^{\mathrm{RL}}D^{\alpha}\mathbf{Y}_{m-1}(t) &= -\lambda_k \mathbf{Y}_{m-1}(t) - \mathbf{Y}_m(t),\\
^{\mathrm{RL}}D^{\alpha}\mathbf{Y}_m(t) &= -\lambda_k \mathbf{Y}_m(t),
\end{aligned}$$

which can be solved from the bottom up. Beginning with the last equation, we obtain:

$$\mathbf{Y}_m(t) = \boldsymbol{Y}_m t^{\alpha-1} E_{\alpha,\alpha}\left(-\lambda_k t^{\alpha}\right) = \Theta(t^{-(\alpha+1)}).$$

Further, the differential equation for $\mathbf{Y}_{m-1}(t)$ is given by:

$$^{\mathrm{RL}}D^{\alpha}\mathbf{Y}_{m-1}(t) = -\lambda_k \mathbf{Y}_{m-1}(t) - \mathbf{Y}_m(t)$$

Applying the Laplace transform and referring to [23, Theorem 7.1.], we obtain:

$$\mathscr{L}\left\{^{\mathrm{RL}}D^{\alpha}\mathbf{Y}_{m-1}(t)\right\} = s^{\alpha} Y_{m-1}(s) - \lim_{t\to 0^+} J^{1-\alpha}\mathbf{Y}_{m-1}(t)$$

where $Y_{m-1}(s)$ is the Laplace transform of $\mathbf{Y}_{m-1}(t)$. For the right-hand side of the differential equation, we have $\mathscr{L}\{\lambda_k \mathbf{Y}_{m-1}(t)\} = \lambda_k Y_{m-1}(s)$. Additionally, the Laplace transform of $t^{\alpha-1}E_{\alpha,\alpha}\left(-\lambda_k t^{\alpha}\right)$ known to be $\frac{1}{s^{\alpha}+\lambda_k}$ [53, Equation 1.80]. Consequently, the equation in the Laplace domain is represented as:

$$\begin{aligned}
&s^{\alpha} Y_{m-1}(s) - \lim_{t\to 0^+} J^{1-\alpha}\mathbf{Y}_{m-1}(t)\\
=& -\lambda_k Y_{m-1}(s) - \frac{\boldsymbol{Y}_m}{s^{\alpha}+\lambda_k}
\end{aligned}$$

Rearranging this equation to isolate $Y_{m-1}(s)$ yields:

$$\begin{aligned}
Y_{m-1}(s) &= \frac{\lim_{t\to 0^+} J^{1-\alpha}\mathbf{Y}_{m-1}(t) - \frac{\boldsymbol{Y}_m}{s^{\alpha}+\lambda_k}}{s^{\alpha}+\lambda_k}\\
&= c_{m-1}\frac{1}{s^{\alpha}+\lambda_k} + \boldsymbol{Y}_m \frac{1}{(s^{\alpha}+\lambda_k)^2}
\end{aligned}$$

where $c_{m-1} := \lim_{t\to 0^+} J^{1-\alpha}\mathbf{Y}_{m-1}(t)$ is used to simplify the notation. Since we already know that $t^{\alpha-1}E_{\alpha,\alpha}\left(-\lambda t^{\alpha}\right) = \Theta(t^{-(\alpha+1)})$ and its Laplace transform is $\frac{1}{s^{\alpha}+\lambda_k}$, we have that the $\mathbf{Y}(t) = \Theta(t^{-(\alpha+1)})$. Applying the same process recursively, we can conclude that for all indices $i = 1, \ldots, m$, the following relationship holds:

$$\mathbf{Y}_i(t) = \Theta(t^{-(\alpha+1)}). \tag{D.8}$$

Consequently, we can deduce that, akin to the scenarios involving diagonalizable matrices, $\mathbf{Y}(t)$ converge to 0 at an algebraic rate. The proof now is complete. □