

Singularity-Aware Cubic-Hermite Neural Collocation for Semilinear Time-Fractional Reaction–Diffusion Equations

Alvaro H. Salas

ahsalass@unal.edu.co

Universidad Nacional de Colombia

Research Article

Keywords: Caputo derivative, cubic Hermite interpolation, physics-informed neural network, fractional reaction–diffusion equation, product integration, inverse parameter identification

Posted Date: October 26th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11261815/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Singularity-Aware Cubic-Hermite Neural Collocation for Semilinear Time-Fractional Reaction–Diffusion Equations

Alvaro H. Salas

Department of Mathematics and Statistics, Universidad Nacional de Colombia, Sede Manizales, Colombia

FIZMAKO Research Group

Corresponding author: ahsalass@unal.edu.co; ORCID: 0000-0001-9343-6062

Abstract

We develop a cubic-Hermite neural collocation method for semilinear reaction–diffusion equations with a Caputo time derivative. The central device is a product-integration representation of the memory term in which the neural state and its automatically differentiated time slope are used as the endpoint data of a cubic Hermite interpolant on every time panel. The weak singularity of the Caputo kernel is handled when the panel weights are precomputed, so training requires only matrix-like history accumulation. Initial and homogeneous Dirichlet data are imposed exactly by the neural ansatz, avoiding penalty parameters for these constraints. A consistency estimate predicts an $O(h^{4-\alpha})$ memory defect for sufficiently smooth temporal traces. In a scalar benchmark $v(t) = t^{4.2}$, the Hermite memory approximation is three to six orders of magnitude more accurate than the classical L1 formula on the tested meshes. For a manufactured semilinear fractional reaction–diffusion problem with $\alpha = 0.7$, a small CPU-trained network attains a relative L^2 error of 1.98×10^{-5} and maximum error 3.80×10^{-5} . A joint identification experiment also recovers the fractional order and diffusivity from several remote initial guesses. The complete source code and the numerical data used in the paper are supplied for reproducibility.

Keywords: Caputo derivative; cubic Hermite interpolation; physics-informed neural network; fractional reaction–diffusion equation; product integration; inverse parameter identification.

1 Introduction

Time-fractional diffusion and reaction–diffusion equations model hereditary transport and subdiffusive relaxation by replacing the classical first-order time derivative with a nonlocal operator. The Caputo derivative is especially convenient because it is compatible with classical initial data, but its history dependence makes each residual evaluation substantially more expensive than in an integer-order PDE; see, for example, [6, 7]. Classical discretizations include convolution quadrature [8], L1-type formulas [5, 9], and the $L2-1_\sigma$ construction [4].

Physics-informed neural networks (PINNs) provide a mesh-light representation in which differential equations enter the training objective through residuals [2, 10]. Fractional PINNs (fPINNs) extended this concept to nonlocal operators by combining automatic differentiation with numerical fractional differentiation [3]. Nevertheless, the memory discretization remains a decisive accuracy bottleneck: automatic differentiation does not remove the history integral. Training pathologies and unbalanced losses can further limit accuracy [11, 12].

A recent Hermite neural solver for the time-fractional Allen–Cahn equation showed that endpoint derivative information can be exploited to construct higher-order approximations of the Caputo

term [1]. The present work develops a distinct formulation around three practical objectives: (i) precompute singularity-aware cubic-Hermite product-integration weights once and reuse them throughout training; (ii) impose both initial and homogeneous Dirichlet data exactly, so the training loss contains only the PDE residual; and (iii) expose the same memory operator for parameter identification, making the fractional order itself an estimable quantity.

The contributions are summarized as follows.

1. We derive a panelwise cubic-Hermite representation of the Caputo derivative using values and automatic-differentiation slopes at time nodes.
2. We give a consistency estimate of order $h^{4-\alpha}$ for smooth temporal traces and compare the memory approximation directly with the L1 formula.
3. We use a hard-constrained neural ansatz that satisfies the initial and boundary conditions identically and therefore eliminates boundary-penalty tuning.
4. We provide a fully reproducible forward benchmark and a joint fractional-order/diffusivity identification test. No numerical values in the tables or figures are hand entered; they are generated by the accompanying script.

2 Model problem

Let $\Omega = (0, 1)$ and $0 < T \leq 1$. We consider

$${}_0^C D_t^\alpha u(x, t) - \kappa u_{xx}(x, t) + \lambda u(x, t)^3 = f(x, t), \quad (x, t) \in \Omega \times (0, T], \quad (1)$$

with

$$u(x, 0) = g(x), \quad u(0, t) = u(1, t) = 0, \quad (2)$$

where $0 < \alpha < 1$, $\kappa > 0$, and $\lambda \geq 0$. The Caputo derivative is

$${}_0^C D_t^\alpha v(t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{v'(s)}{(t-s)^\alpha} ds. \quad (3)$$

The one-dimensional setting is chosen so that the temporal memory approximation can be isolated cleanly. The construction extends directly to higher-dimensional spatial networks because only the time trace enters (3).

3 Cubic-Hermite product integration

Let $0 = t_0 < t_1 < \dots < t_N = T$ with $h_j = t_{j+1} - t_j$. On $I_j = [t_j, t_{j+1}]$, write $z = (s - t_j)/h_j$. For a temporal trace $v(s)$, define the cubic Hermite interpolant

$$H_j(s) = h_{00}(z)v_j + h_{10}(z)h_j\dot{v}_j + h_{01}(z)v_{j+1} + h_{11}(z)h_j\dot{v}_{j+1}, \quad (4)$$

where

$$h_{00} = 2z^3 - 3z^2 + 1, \quad h_{10} = z^3 - 2z^2 + z, \quad h_{01} = -2z^3 + 3z^2, \quad h_{11} = z^3 - z^2.$$

At t_n we approximate (3) by

$$\mathcal{H}_h^\alpha v(t_n) = \frac{1}{\Gamma(1-\alpha)} \sum_{j=0}^{n-1} \int_{t_j}^{t_{j+1}} \frac{H_j'(s)}{(t_n - s)^\alpha} ds. \quad (5)$$

Because H'_j is linear in $v_j, v_{j+1}, \dot{v}_j, \dot{v}_{j+1}$, every panel integral can be written as four scalar weights. These weights depend only on the temporal grid, n, j , and α ; they are precomputed using quadrature that directly integrates the integrable endpoint singularity. Training then uses only weighted sums of neural values and automatic-differentiation slopes.

Proposition 1 (Smooth-trace consistency). *Assume $v \in C^4[0, T]$ and a uniform grid of width h . For each fixed $t_n > 0$,*

$$\left| {}^C_0 D_t^\alpha v(t_n) - \mathcal{H}_h^\alpha v(t_n) \right| \leq C_{\alpha, T} h^{4-\alpha} \max_{0 \leq s \leq t_n} |v^{(4)}(s)|,$$

where $C_{\alpha, T}$ is independent of h .

Proof sketch. The cubic Hermite remainder on I_j is $R_j(s) = v^{(4)}(\xi_s)(s - t_j)^2(s - t_{j+1})^2/4!$, hence $|R_j(s)| \leq Ch^4 \|v^{(4)}\|_\infty$. Subtracting (5) from (3), integrating R'_j against the weakly singular kernel by parts panelwise, and using $R_j(t_j) = R_j(t_{j+1}) = 0$ gives integrals of $R_j(s)(t_n - s)^{-\alpha-1}$. The final panel contributes $O(h^{4-\alpha})$; summing the preceding panels gives the same order. The factor $\Gamma(1 - \alpha)^{-1}$ is absorbed into $C_{\alpha, T}$. $\square$

Remark 1. *Fractional diffusion solutions can have an initial weak singularity even for smooth data [5]. Proposition 1 therefore describes the smooth-trace regime. A graded temporal mesh can be inserted without changing the neural formulation; the present experiments use a uniform grid to isolate the effect of the Hermite memory operator.*

4 Hard-constrained neural collocation

Let $N_\theta(x, t)$ be a fully connected network. For homogeneous Dirichlet data and a compatible initial profile $g(0) = g(1) = 0$, we use

$$u_\theta(x, t) = g(x) + tx(1 - x)N_\theta(x, t). \quad (6)$$

Then $u_\theta(x, 0) = g(x)$ and $u_\theta(0, t) = u_\theta(1, t) = 0$ identically. Consequently, no penalty weights are needed for initial or boundary losses.

At interior spatial points x_i and time nodes t_n , $n \geq 1$, the residual is

$$R_{i,n}(\theta) = \mathcal{H}_h^\alpha u_\theta(x_i, t_n) - \kappa \partial_{xx} u_\theta(x_i, t_n) + \lambda u_\theta(x_i, t_n)^3 - f(x_i, t_n), \quad (7)$$

and training minimizes

$$\mathcal{L}(\theta) = \frac{1}{N_x N_t} \sum_{i=1}^{N_x} \sum_{n=1}^{N_t} |R_{i,n}(\theta)|^2. \quad (8)$$

Both $\partial_t u_\theta$ needed by the Hermite memory term and $\partial_{xx} u_\theta$ are computed by automatic differentiation.

5 Numerical experiments

All reported values are generated by `code/reproduce.py`. Computations use double precision, PyTorch automatic differentiation, seed 1234, and CPU execution. The figures in this manuscript are also generated by that script.

Algorithm 1 Cubic-Hermite neural collocation

- 1: Choose $\{x_i\}$, $\{t_n\}$ and network N_θ .
 - 2: Precompute all four Hermite-Caputo weights for each pair (n, j) .
 - 3: Form the hard-constrained state u_θ from (6).
 - 4: **for** each optimization step **do**
 - 5: Evaluate u_θ , $\partial_t u_\theta$, and $\partial_{xx} u_\theta$ on the collocation grid.
 - 6: Assemble $\mathcal{H}_h^\alpha u_\theta$ from the precomputed history weights.
 - 7: Minimize the residual loss (8).
 - 8: **end for**
-

5.1 Accuracy of the memory operator

We first remove the neural approximation entirely and test $v(t) = t^\beta$ with $\beta = 4.2$. The exact Caputo derivative at $t = 1$ is

$${}^C D_t^\alpha t^\beta|_{t=1} = \frac{\Gamma(\beta + 1)}{\Gamma(\beta + 1 - \alpha)}.$$

Table 1 compares absolute errors of the L1 formula and the cubic-Hermite product integration. The Hermite errors decrease at a rate consistent with $4 - \alpha$, while L1 remains much less accurate on the same grids.

Table 1: Absolute error in the Caputo derivative of $t^{4.2}$ at $t = 1$.

α	N	L1 error	Cubic Hermite error
0.4	8	6.64×10^{-2}	3.86×10^{-5}
0.4	16	2.43×10^{-2}	3.37×10^{-6}
0.4	32	8.60×10^{-3}	2.90×10^{-7}
0.4	64	2.98×10^{-3}	2.47×10^{-8}
0.7	8	2.52×10^{-1}	8.41×10^{-5}
0.7	16	1.09×10^{-1}	8.78×10^{-6}
0.7	32	4.61×10^{-2}	9.06×10^{-7}
0.7	64	1.91×10^{-2}	9.27×10^{-8}
0.9	8	5.37×10^{-1}	6.74×10^{-5}
0.9	16	2.64×10^{-1}	7.99×10^{-6}
0.9	32	1.27×10^{-1}	9.40×10^{-7}
0.9	64	5.99×10^{-2}	1.11×10^{-7}

5.2 Forward semilinear reaction–diffusion problem

We choose $T = 1$, $\alpha = 0.7$, $\kappa = 0.05$, $\lambda = 1$, and manufacture

$$u(x, t) = \sin(\pi x) + t^2 x(1 - x). \tag{9}$$

The forcing f is obtained by substituting (9) into (1); in particular,

$${}^C D_t^\alpha [t^2 x(1 - x)] = \frac{2}{\Gamma(3 - \alpha)} t^{2-\alpha} x(1 - x).$$

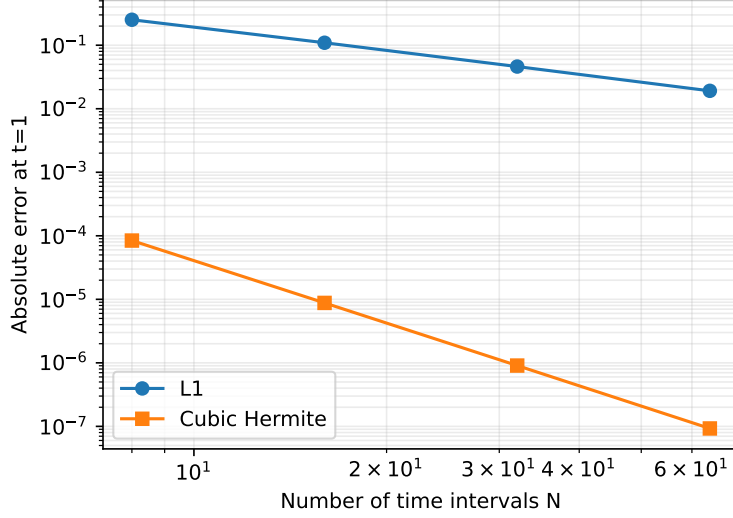


Figure 1: Memory-operator convergence for $v(t) = t^{4.2}$ and $\alpha = 0.7$.

We use $N_x = 25$ interior spatial points and $N_t = 16$ time intervals. The network has three hidden layers with 32 tanh units each. Adam is run for 1500 iterations with learning rate 2×10^{-3} , followed by L-BFGS with at most 200 internal iterations.

On a dense 201×101 test grid, the relative discrete L^2 error is

$$\frac{\|u_\theta - u\|_2}{\|u\|_2} = 1.98 \times 10^{-5},$$

and the maximum pointwise error is 3.80×10^{-5} . The final collocation residual MSE is 2.01×10^{-8} and the maximum absolute residual on the training grid is 4.44×10^{-4} . The complete CPU run took approximately 20.2 s in the reference execution.

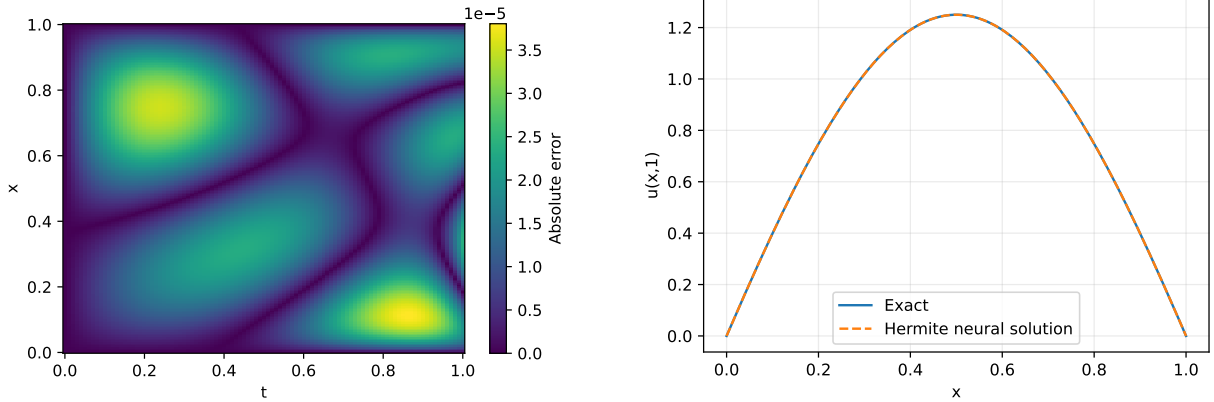


Figure 2: Forward benchmark. Left: absolute error over the dense test grid. Right: exact and computed final-time profiles.

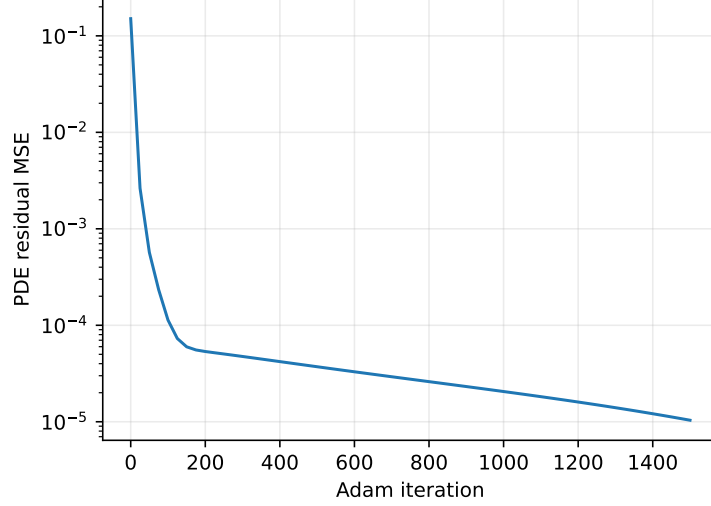


Figure 3: PDE residual MSE during the Adam phase. The subsequent L-BFGS stage produces the final reported residual.

5.3 Joint recovery of fractional order and diffusivity

To test identifiability of the memory operator independently of network approximation error, we use the manufactured state (9) as a synthetic observed field and treat (α, κ) as unknown while keeping $\lambda = 1$. The objective is the RMS discrete PDE residual assembled with the same cubic-Hermite history operator. Table 2 shows recovery from four deliberately separated initial guesses. All runs converge to the true pair $(0.7, 0.05)$.

Table 2: Joint recovery of α and κ from the synthetic state field.

α_0	κ_0	$\hat{\alpha}$	$\hat{\kappa}$	function evaluations
0.30	0.12	0.7000000000	0.0500000000	10
0.55	0.08	0.7000000000	0.0500000000	8
0.85	0.02	0.7000000001	0.0500000000	7
0.90	0.15	0.7000000000	0.0500000000	10

6 Discussion

The numerical evidence separates two error sources that are often entangled in fractional PINNs. The scalar benchmark shows that the memory discretization itself can be made substantially more accurate than L1 on smooth traces. Once these weights are precomputed, the training cost is dominated by network differentiation and the history accumulation; no singular quadrature is executed inside the optimizer. The hard-constrained ansatz also removes a user-chosen boundary penalty, which is useful because multi-term PINN objectives can be poorly balanced.

The present study is intentionally compact. First, the smooth benchmark does not address the generic initial singularity of subdiffusion solutions; graded meshes should be incorporated for that regime [5]. Second, the inverse experiment uses a clean synthetic state in order to isolate parameter

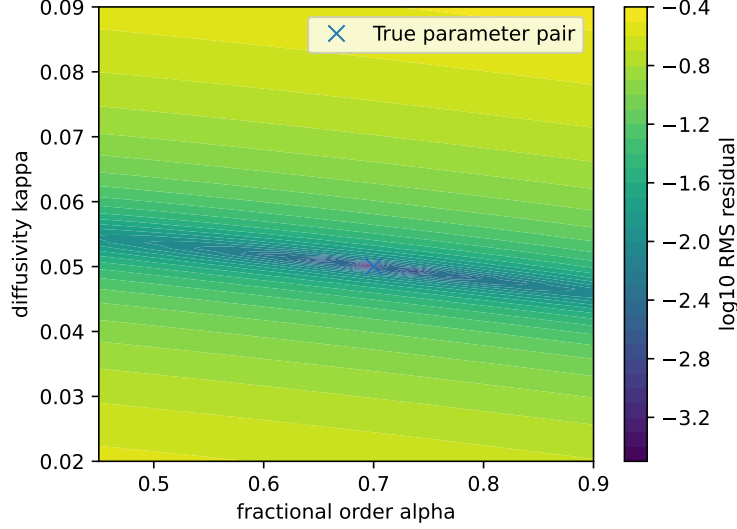


Figure 4: Logarithm of the RMS PDE residual in the (α, κ) plane for the inverse benchmark. The minimum is sharply localized at the true parameter pair.

identifiability. No claim is made here about robustness to substantial observation noise. Third, large N_t gives the usual $O(N_t^2)$ history assembly unless fast convolution or compression is added. These are natural directions for extension.

7 Conclusions

A singularity-aware cubic-Hermite product-integration operator has been combined with a hard-constrained neural representation for semilinear time-fractional reaction–diffusion equations. The scheme uses neural time slopes as Hermite data, precomputes all weakly singular history weights, and trains with a single PDE residual loss. The operator benchmark exhibits the expected high-order behavior, while the forward test reaches a relative L^2 error below 2×10^{-5} on a modest CPU-only configuration. The same memory operator also yields a well-localized inverse map for joint estimation of the fractional order and diffusivity. The supplied code reproduces every numerical table and figure and provides a base for graded meshes, multidimensional domains, and noisy inverse problems.

Declarations

Funding. No external funding was used for the computations reported in this manuscript.

Conflict of interest. The author declares no conflict of interest.

Data and code availability. The complete Python script, trained state dictionary, numerical results, and figure-generation code are included with the submission package.

Author contributions. A.H.S. conceived the study, developed the numerical formulation, implemented the computations, analyzed the results, and wrote the manuscript.

References

- [1] X. Wang, X. Wang, H. Qi, H. Xu, Numerical simulation of time fractional Allen–Cahn equation based on Hermite neural solver, *Applied Mathematics and Computation* 491 (2025) 129234. <https://doi.org/10.1016/j.amc.2024.129234>.
- [2] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics* 378 (2019) 686–707. <https://doi.org/10.1016/j.jcp.2018.10.045>.
- [3] G. Pang, L. Lu, G. E. Karniadakis, fPINNs: Fractional Physics-Informed Neural Networks, *SIAM Journal on Scientific Computing* 41(4) (2019) A2603–A2626. <https://doi.org/10.1137/18M1229845>.
- [4] A. A. Alikhanov, A new difference scheme for the time fractional diffusion equation, *Journal of Computational Physics* 280 (2015) 424–438. <https://doi.org/10.1016/j.jcp.2014.09.031>.
- [5] M. Stynes, E. O’Riordan, J. L. Gracia, Error analysis of a finite difference method on graded meshes for a time-fractional diffusion equation, *SIAM Journal on Numerical Analysis* 55(2) (2017) 1057–1079. <https://doi.org/10.1137/16M1082329>.
- [6] K. Diethelm, *The Analysis of Fractional Differential Equations*, Springer, Berlin, 2010.
- [7] I. Podlubny, *Fractional Differential Equations*, Academic Press, San Diego, 1999.
- [8] E. Cuesta, C. Lubich, C. Palencia, Convolution quadrature time discretization of fractional diffusion-wave equations, *Mathematics of Computation* 75 (2006) 673–696.
- [9] B. Jin, R. Lazarov, Z. Zhou, An analysis of the L1 scheme for the subdiffusion equation with nonsmooth data, *IMA Journal of Numerical Analysis* 36(1) (2016) 197–221.
- [10] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, *Nature Reviews Physics* 3 (2021) 422–440.
- [11] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, *SIAM Journal on Scientific Computing* 43(5) (2021) A3055–A3081.
- [12] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, *Advances in Neural Information Processing Systems* 34 (2021) 26548–26560.
- [13] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, *Journal of Computational Physics* 375 (2018) 1339–1364.
- [14] L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, *SIAM Review* 63(1) (2021) 208–228.
- [15] S. M. Allen, J. W. Cahn, A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening, *Acta Metallurgica* 27(6) (1979) 1085–1095.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [IJACMSingularityAwareHermiteNeuralFULL.zip](#)