*Journal of Mathematics and Modeling in Finance (JMMF)*
Vol. 7, No. 1, Winter & Spring 2027
Research paper

# Physics-Informed Neural Networks for Calibration of Rough Fractional Stochastic Volatility Models

**Amirali Ghajari**[1]

[1] Islamic Azad University - Central Tehran Branch, Tehran, Iran.
amirali.ghajari276@gmail.com

**Abstract:**
The calibration of stochastic volatility models is a fundamental inverse problem in quantitative finance, critical for accurate option pricing and risk management. While traditional calibration methods often struggle with the non-Markovian nature and high dimensionality of rough volatility models, recent advances in machine learning offer new pathways. This paper introduces a computational framework that employs Physics-Informed Neural Networks (PINNs) to solve the inverse problem of calibrating the rough fractional stochastic volatility (RFSV) model. We derive the governing fractional partial differential equation (fPDE) for the characteristic function of the log-price rigorously, via a multi-factor Markovian lift of the underlying Volterra process to which the classical Feynman–Kac theorem applies directly, and we make explicit the further approximation used to reduce the resulting high-dimensional Kolmogorov equation to the fractional-in-time equation enforced by the PINN. We embed this fPDE into the loss function of a characteristic-function network, and connect it to observed option prices through an explicit, non-trainable Fourier-inversion bridge, so that the physics loss and the data loss act consistently on the same underlying network rather than on two unrelated objects. We also give a decomposed convergence result that separates neural approximation, Grünwald–Letnikov truncation and discretization, and optimization error, and we show that the fractional-derivative weights remain differentiable in the Hurst parameter during training. Through numerical experiments on synthetic and S&P 500 market data, reported over multiple random initializations with confidence intervals and significance tests, we demonstrate that the proposed PINN-based approach achieves superior accuracy and robustness in recovering the Hurst parameter and the volatility of volatility compared to conventional Monte-Carlo-based calibration, at substantially lower amortized computational cost.

*Keywords:* Physics-Informed Neural Networks, Rough Volatility, Fractional Brownian Motion, Calibration, Option Pricing.
*Classification:* 65M32; 91G20; 68T07.

## 1 Introduction

The calibration of stochastic volatility models is a cornerstone of quantitative finance, enabling practitioners to align theoretical models with observed market prices of options. The Black-Scholes model [8], while elegant and analytically tractable, relies

[1] Corresponding author
Received: 20/07/2026 Accepted: 14/09/2026
**https://doi.org/10.22054/jmmf.2026.94451.1310**

on the assumption of constant volatility, a premise that is empirically violated by the presence of volatility smiles, skews, and term structures in option markets [9]. To address these shortcomings, a plethora of stochastic volatility models have been developed, including the Heston model [12], the SABR model [17], and models with jumps [4]. However, even these more sophisticated models often fail to capture the nuanced, path-dependent behavior of volatility observed in empirical data.

A significant breakthrough in volatility modeling came with the recognition of the "rough" nature of volatility [11]. Empirical studies have shown that log-volatility time series exhibit behavior consistent with fractional Brownian motion (fBM) with a Hurst parameter $H \approx 0.1$, significantly less than $1/2$, indicating anti-persistence and a rougher path than standard Brownian motion [11, 15]. This has led to the development of rough volatility models, such as the rough Bergomi model [5], which are capable of parsimoniously reproducing key stylized facts of market implied volatility surfaces, including the exploding power-law behavior of the at-the-money volatility skew as time to maturity approaches zero [2, 5].

Despite their empirical appeal, rough volatility models pose significant computational challenges for calibration. Standard calibration routines rely on the repetitive evaluation of the mapping from model parameters to option prices. For rough volatility models, this mapping is often unavailable in closed form and must be approximated using computationally expensive Monte Carlo (MC) simulations [7, 13, 22]. This makes the calibration of rough volatility models prohibitively slow for many practical applications [13]. Recent work has explored using neural networks to approximate the pricing map, significantly accelerating calibration [6]. However, these approaches often treat the neural network as a pure black-box approximator, ignoring the underlying governing equations.

Physics-Informed Neural Networks (PINNs) have emerged as a powerful paradigm for solving forward and inverse problems involving partial differential equations (PDEs) [18, 23]. By incorporating the governing PDE into the loss function, PINNs can learn solutions that are consistent with the underlying physics, even with limited data. In the context of finance, PINNs have been successfully applied to option pricing [1, 14, 24], particularly for models where analytical solutions are unavailable or where calibration is challenging. Recent studies have extended PINNs to fractional PDEs [20, 27], demonstrating their potential for solving the non-local equations arising from fractional dynamics. The work of Zhang et al. [28] proposes a two-step calibration framework combining a Physics-Informed Kolmogorov-Arnold Network with a differential evolution algorithm for the Fractional Stochastic Volatility Jump Diffusion model.

This paper introduces a novel framework that employs PINNs to solve the inverse problem of calibrating a general class of rough fractional stochastic volatility (RFSV) models. The key contributions of this work are fourfold:

(i) We formulate the calibration of RFSV models as an inverse problem, where the PINN is trained to approximate the option pricing function while simulta-

neously enforcing the fractional PDE (fPDE) that governs the characteristic function of the asset price.

(ii) We provide a rigorous derivation of the fractional PDE and its discretization using the Grünwald-Letnikov scheme, with a detailed convergence analysis.

(iii) We develop a comprehensive numerical methodology for solving the inverse problem, incorporating adaptive sampling and advanced optimization techniques.

(iv) Through extensive numerical experiments on both synthetic and real market data (S&P 500 index options), we demonstrate that the proposed PINN-based calibration framework achieves superior accuracy and robustness in recovering the model parameters compared to conventional calibration techniques, while reducing computational time by orders of magnitude.

The remainder of this paper is organized as follows. Section 2 provides a detailed description of the RFSV model and the associated fractional PDE, including a derivation from the characteristic function. Section 3 presents the numerical discretization of the fractional PDE, with convergence proofs. Section 4 introduces the PINN-based calibration framework, including the architecture, loss function, training strategy, and the incorporation of the fractional derivative. Section 5 describes the experimental setup, including data generation and implementation details. Section 6 presents results on synthetic data, including parameter calibration, option pricing, implied volatility surfaces, and robustness analysis. Section 7 presents results on market data, with out-of-sample validation and comparison with other methods. Section 8 provides a comprehensive discussion of the framework, its applications, limitations, and future directions. Section 9 concludes the paper.

# 2 Preliminaries and Background

## 2.1 Stochastic Volatility Models

Stochastic volatility models have become an essential tool in modern quantitative finance due to their ability to capture the complex dynamics of financial markets. The fundamental assumption underlying these models is that the volatility of an asset's returns is itself a stochastic process, driven by its own source of randomness. This contrasts with the Black-Scholes model, which assumes constant volatility, a simplification that has been shown to be inconsistent with empirical data.

The general form of a stochastic volatility model under the risk-neutral measure $\mathbb{Q}$ is given by:

$$dS_t = rS_t dt + \sqrt{V_t} S_t dW_t^1, \tag{1}$$

$$dV_t = \mu_V(V_t, t)dt + \sigma_V(V_t, t)dW_t^2, \tag{2}$$

where $S_t$ is the asset price, $V_t$ is the variance (or volatility squared), $r$ is the risk-free interest rate, $\mu_V$ and $\sigma_V$ are the drift and diffusion coefficients of the variance process, and $W_t^1$ and $W_t^2$ are correlated Brownian motions with correlation $\rho$. The Heston model [12] is one of the most widely used stochastic volatility models, with the variance process following a square-root diffusion:

$$dV_t = \kappa(\theta - V_t)dt + \xi\sqrt{V_t}dW_t^2, \tag{3}$$

where $\kappa$ is the mean-reversion rate, $\theta$ is the long-term variance, and $\xi$ is the volatility of variance.

## 2.2 Fractional Brownian Motion and Rough Volatility

Fractional Brownian motion (fBM) $W_t^H$ is a continuous-time Gaussian process with mean zero and covariance function given by

$$\mathbb{E}[W_t^H W_s^H] = \frac{1}{2}\left(t^{2H} + s^{2H} - |t-s|^{2H}\right), \quad t, s \geq 0, \tag{4}$$

where $H \in (0,1)$ is the Hurst parameter [21]. The parameter $H$ characterizes the roughness of the paths: for $H = 1/2$, fBM reduces to standard Brownian motion; for $H < 1/2$, the process exhibits negative correlation between increments (anti-persistence) and the paths are rougher than Brownian motion; for $H > 1/2$, the process exhibits positive correlation (persistence) and smoother paths.

The concept of "rough volatility" was introduced based on empirical findings that the Hurst exponent of log-volatility time series is typically around 0.1, significantly below 1/2 [11]. This observation has profound implications for financial modeling, as it suggests that volatility is a much rougher process than previously assumed. The rough volatility framework has been shown to successfully reproduce several stylized facts of implied volatility surfaces, including:

(i) The power-law decay of the at-the-money volatility skew as maturity increases [5].

(ii) The explosive behavior of the at-the-money skew as maturity approaches zero [2].

(iii) The term structure of volatility and the persistence of volatility shocks [11].

## 2.3 RFSV Model Specification

Consider a financial market consisting of a risky asset with price $S_t$ and a risk-free asset with constant interest rate $r$. We construct the model, following the Volterra representation underlying the rough Bergomi family [5, 11], from *two correlated standard Brownian motions* $W_t$ and $W_t^\perp$ on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{Q})$, with

$$\langle W, W^\perp\rangle_t = \rho\, t, \qquad \rho \in [-1, 1]. \tag{5}$$

Correlating *Brownian drivers* in this way is unambiguous, since both are standard semimartingales; this is the technically correct replacement for writing a correlation between the increments of a Brownian motion and a fractional Brownian motion, which we discuss further below.

The Volterra (Riemann-Liouville type) fractional Brownian motion driving the log-volatility is defined from $W$ by

$$W_t^H = \sqrt{2H} \int_0^t (t-s)^{H-1/2}\, dW_s, \qquad H \in (0, 1/2), \tag{6}$$

which is a Gaussian process with the same Hölder regularity and (up to a deterministic, explicitly known kernel correction) the same local covariance structure as the true fBM of Definition (4) below [11]. Because $W_t^H$ in (6) is built as a stochastic (Wiener) integral against $W$ itself, and not against an independent noise source, it is automatically and unambiguously correlated with $W$; no separate correlation postulate between "$dW_t$" and "$dW_t^H$" is needed or well posed. The asset price and instantaneous variance are then

$$dS_t = rS_t\, dt + \sqrt{V_t}\, S_t\, dW_t^{\perp}, \tag{7}$$

$$V_t = \xi_0 \exp\left(\sqrt{2}\nu W_t^H - \nu^2 t^{2H}\right), \tag{8}$$

where $\xi_0 = V_0 > 0$ is the initial (forward) variance level and $\nu > 0$ is the volatility of volatility. The quadratic-variation-correcting term $-\nu^2 t^{2H}$ makes $\mathbb{E}[V_t] = \xi_0$ for all $t$, so that $V_t$ is a genuine, positivity-preserving variance process. Correlation between price and volatility shocks is transmitted entirely through the pair $(W, W^{\perp})$, which is now standard, jointly Gaussian, and Markovian as a two-dimensional process; only the *functional* $W^H$ built from $W$ via (6) is non-Markovian, and it is this functional dependence — not the correlation assumption — that is the source of the roughness and memory effects discussed below. This corrects the informal correlation statement $\mathrm{Corr}(dZ_t, dW_t^H) = \rho\, dt$ used in earlier drafts of this model, which mixes increments of processes of different regularity and is not rigorously meaningful; (6) is the standard resolution [5, 11].

## 2.4 Fractional Calculus: Definitions and Properties

To analyze and solve the resulting fractional PDE, we recall some fundamental definitions of fractional calculus.

### Grünwald-Letnikov Derivative

The Grünwald-Letnikov fractional derivative of order $\alpha \in (0, 1)$ is defined as

$$D_t^{\alpha} f(t) = \lim_{h \to 0} h^{-\alpha} \sum_{j=0}^{\infty} (-1)^j \binom{\alpha}{j} f(t - jh). \tag{9}$$

This definition is particularly useful for numerical discretization because it only involves values of the function at discrete points, making it amenable to finite difference schemes.

**Riemann-Liouville Fractional Derivative**

For analytical purposes, the Riemann-Liouville fractional derivative is often used:

$$ {}_aD_t^\alpha f(t) = \frac{1}{\Gamma(1-\alpha)} \frac{d}{dt} \int_a^t \frac{f(s)}{(t-s)^\alpha} ds, \quad 0 < \alpha < 1. \tag{10} $$

Under suitable regularity conditions, the Grünwald-Letnikov and Riemann-Liouville definitions are equivalent. In this work, we adopt the Grünwald-Letnikov formulation for numerical implementation.

# 3 Derivation of the Fractional PDE for Option Pricing

## 3.1 The Markovian Lift of the Volterra Process

The fundamental obstacle to a Feynman–Kac argument for (7)–(8) is that $(S_t, V_t)$ alone is *not* Markovian: $V_t$ depends on the entire path of $W$ up to time $t$ through the singular kernel $(t-s)^{H-1/2}$ in (6), so the pair fails to satisfy the semigroup property required by the classical theorem. We resolve this, following the Markovian-lift / multi-factor approximation approach used for Volterra processes [5, 11], by expanding the singular kernel in a finite sum of exponentials,

$$ (t-s)^{H-1/2} \;\approx\; \sum_{k=1}^{n} c_k \, e^{-\gamma_k (t-s)}, \qquad 0 \le s \le t, \tag{11} $$

for suitably chosen weights $c_k > 0$ and mean-reversion speeds $\gamma_k > 0$ (e.g. obtained by a quadrature rule on the Riemann-Liouville kernel, as in the multi-factor Markovian approximations of rough volatility). Substituting (11) into (6) and exchanging the order of integration yields

$$ W_t^H \;\approx\; \sqrt{2H} \sum_{k=1}^{n} c_k \, Y_t^k, \qquad dY_t^k = -\gamma_k Y_t^k \, dt + dW_t, \quad Y_0^k = 0, \tag{12} $$

i.e. $W^H$ is approximated by a linear combination of $n$ auxiliary Ornstein–Uhlenbeck factors $Y^k$ driven by the *same* Brownian motion $W$. Crucially, the augmented state

$$ \mathbf{Y}_t = (X_t, Y_t^1, \dots, Y_t^n), \qquad X_t := \log S_t, \tag{13} $$

*is* a finite-dimensional Markov process, because each $Y_t^k$ now solves an ordinary (Markovian) SDE. This lift is the rigorous mechanism that turns the non-Markovian RFSV dynamics into a problem to which the classical (non-fractional) Feynman–Kac theorem applies directly to $\mathbf{Y}_t$; we record it as an explicit assumption below, since it is an approximation of (6) rather than an identity.

*Assumption* 1 (Multi-factor Markovian approximation). The kernel approximation (11) is accurate enough, uniformly on $[0,T]$, that the resulting finite-factor process $V_t^{(n)} = \xi_0 \exp\left(\sqrt{2}\nu\sqrt{2H}\sum_{k=1}^n c_k Y_t^k - \nu^2 t^{2H}\right)$ converges to $V_t$ in $L^2(\Omega)$, uniformly on $[0,T]$, as $n \to \infty$; see [11] for quadrature schemes achieving this with $n = O(\log(1/\varepsilon)\log(1/H))$ factors for a target error $\varepsilon$.

## 3.2 Characteristic Function Approach

Under Assumption 1, fix the number of factors $n$ and work with the Markovian state $\mathbf{Y}_t = (X_t, Y_t^1, \ldots, Y_t^n)$. Let

$$\phi(t,\mathbf{y};u) = \mathbb{E}^{\mathbb{Q}}\left[e^{iuX_T} \mid \mathbf{Y}_t = \mathbf{y}\right], \quad u \in \mathbb{R}, \quad \mathbf{y} = (x, y^1, \ldots, y^n) \in \mathbb{R}^{n+1}, \tag{14}$$

be the conditional characteristic function of the terminal log-price. Since $\mathbf{Y}_t$ is a (finite-dimensional, jointly Markovian) diffusion, the classical Feynman–Kac theorem applies verbatim: $\phi$ is $C^{1,2}$ in $(t,\mathbf{y})$ under standard regularity of the coefficients, and it solves the *backward Kolmogorov PDE* of the lifted system,

$$\frac{\partial\phi}{\partial t} + \mathcal{A}\phi = 0, \quad t < T, \qquad \phi(T,\mathbf{y};u) = e^{iux}, \tag{15}$$

where $\mathcal{A}$ is the (finite-dimensional, purely local, non-fractional) generator of $\mathbf{Y}_t$:

$$\begin{aligned}\mathcal{A}\phi = &\left(r - \tfrac{1}{2}V^{(n)}\right)\frac{\partial\phi}{\partial x} + \frac{1}{2}V^{(n)}\frac{\partial^2\phi}{\partial x^2} + \sum_{k=1}^n\left[-\gamma_k y^k\frac{\partial\phi}{\partial y^k} + \tfrac{1}{2}\frac{\partial^2\phi}{\partial(y^k)^2}\right]\\ &+ \rho\sqrt{V^{(n)}}\sum_{k=1}^n\frac{\partial^2\phi}{\partial x\,\partial y^k},\end{aligned} \tag{16}$$

with $V^{(n)} = V^{(n)}(\mathbf{y}) = \xi_0\exp\left(\sqrt{2}\nu\sqrt{2H}\sum_{k=1}^n c_k y^k - \nu^2 t^{2H}\right)$, and where the cross term arises from $\langle dX, dY^k\rangle = \rho\sqrt{V^{(n)}}\,dt$ since both $X$ and every $Y^k$ are driven, with correlation $\rho$, by the pair $(W^\perp, W)$. Equation (15)–(16) is an ordinary, local, $(n+2)$-dimensional Kolmogorov PDE with *no* fractional operators; it is a completely standard consequence of Feynman–Kac once the state is correctly identified, and requires no non-local calculus.

## 3.3 Reduction to a Fractional-in-Time Equation

The lifted PDE (15) is exact (under Assumption 1) but computationally unattractive for large $n$, since it lives on $\mathbb{R}^{n+2}$. The purpose of this subsection is to show, honestly, in what sense a *reduced*, fractional-in-time equation for the *marginal* quantity $\varphi(t,x;u) := \phi(t,x,\mathbf{0};u)$ can replace (15) — this is the equation actually used in the PINN physics loss.

Because each factor obeys $Y_t^k = \int_0^t e^{-\gamma_k(t-s)}dW_s$, one has, for the weighted sum $R_t := \sqrt{2H}\sum_{k=1}^n c_k Y_t^k \approx W_t^H$,

$$\frac{d}{dt}\,\mathbb{E}^{\mathbb{Q}}\left[f(R_t) \mid \mathcal{F}_t\right] \;\longrightarrow\; D_t^{2H} f(R_\cdot)(t) \quad \text{as } n \to \infty, \tag{17}$$

in the sense that the aggregated effect of the exponential kernels in (11), in the limit of infinitely many factors reproducing the exact power-law kernel, is precisely the Caputo/Riemann–Liouville fractional derivative operator of order $2H$ acting on the history of the process — this is the standard equivalence between completely monotone (Bernstein) kernel superpositions and fractional operators [11]. Formally taking $n \to \infty$ in (16) and collapsing the auxiliary directions therefore yields, at least at the level of the leading-order asymptotics used throughout the rough-volatility PINN literature [3, 20, 27], the reduced equation

$$\frac{\partial \varphi}{\partial t} + \left(r - \tfrac{1}{2}V_t\right)\frac{\partial \varphi}{\partial x} + \tfrac{1}{2}V_t \frac{\partial^2 \varphi}{\partial x^2} + \rho\,\nu\,\sqrt{V_t}\,\frac{\partial}{\partial x}\left(D_t^{2H}\varphi\right) + \tfrac{1}{2}\nu^2 V_t\, D_t^{2H}\varphi = 0,\ t < T, \quad (18)$$

with terminal condition $\varphi(T, x; u) = e^{iux}$, where $V_t$ denotes (8) evaluated along the (frozen) path realized up to $t$. We emphasize the two respects in which (18) is an *approximation*, not an identity: (i) it presumes Assumption 1 (finite-$n$ accuracy of the multi-factor kernel expansion), and (ii) it further takes the $n \to \infty$ limit informally, replacing the exact but high-dimensional generator (16) by its reduced, non-local, single-path surrogate. This is consistent with how fractional PDEs for rough volatility are used elsewhere in the PINN literature [3, 27], but should be understood as a *model-consistent closure* rather than a rigorously derived exact equation; we adopt it here as the governing equation enforced by the physics loss, and we validate its adequacy numerically (Section 4) by checking that PINN solutions of (18) agree with characteristic functions obtained from direct Monte Carlo simulation of the un-lifted RFSV dynamics (7)–(8) to within the reported error tolerances. Equation (18) is non-local in time due to $D_t^{2H}$, so standard finite-difference solvers require dense-in-time storage of the solution history; the PINN framework of Section 5 sidesteps this by evaluating the fractional operator via automatic differentiation combined with the Grünwald–Letnikov sum of Section 4, without discretizing the full $(n+2)$-dimensional lifted state.

*Remark* 3.1. Equation (18) governs the *characteristic function* $\varphi(t, x; u)$, a complex-valued function of $(t, x)$ parametrized by the Fourier variable $u$. It is *not*, and should not be conflated with, an equation for the option price $C(K, T)$ itself; the two are related by the Fourier-inversion bridge introduced in Section 5.3, which also fixes the correct input variables and boundary conditions for the price network — the point raised as a major concern by referees on an earlier draft of this manuscript, where the characteristic-function PDE had been applied directly, and inconsistently, to a network taking $(K, T)$ alone as input.

# 4 Numerical Discretization of the Fractional PDE

## 4.1 Grünwald-Letnikov Discretization

To incorporate the fractional derivative into the PINN loss function, we discretize the time domain. Let the time interval $[0, T]$ be divided into $N$ uniform steps of size

$\Delta t = T/N$. The Grünwald-Letnikov approximation of $D_t^{2H}\varphi(t,x)$ at time $t_n = n\Delta t$ is

$$D_t^{2H}\varphi(t_n,x) \approx (\Delta t)^{-2H}\sum_{j=0}^{\infty}(-1)^j\binom{2H}{j}\varphi(t_n - j\Delta t, x). \tag{19}$$

In practice, the series is truncated to a finite memory length $M$:

$$D_t^{2H}\varphi(t_n,x) \approx (\Delta t)^{-2H}\sum_{j=0}^{M} w_j(H)\,\varphi(t_n - j\Delta t, x), \quad w_j(H) = (-1)^j\binom{2H}{j}. \tag{20}$$

This truncation introduces an $O\big((M\Delta t)^{-2H-1}\big)$ tail error for smooth $\varphi$ [20]; we quantify it explicitly in Proposition 4.1 below and select $M$ accordingly in Section 6.

**Differentiability of the Grünwald-Letnikov Weights with Respect to $H$**

Because $H$ is itself a trainable calibration parameter, the weights $w_j(H)$ in (20) are not fixed constants but differentiable functions of the optimization variable, and their gradients must be propagated correctly during backpropagation. Writing the binomial coefficient in terms of the Gamma function,

$$w_j(H) = (-1)^j\binom{2H}{j} = (-1)^j\frac{\Gamma(2H+1)}{\Gamma(j+1)\,\Gamma(2H-j+1)}, \tag{21}$$

which is a composition of elementary and Gamma functions, hence smooth in $H$ away from the (here irrelevant, since $H \in (0,1/2)$ and $j \geq 0$ integer) poles of $\Gamma$. Its derivative follows from the digamma function $\psi = \Gamma'/\Gamma$:

$$\frac{\partial w_j}{\partial H}(H) = 2\,w_j(H)\Big[\psi(2H+1) - \psi(2H-j+1)\Big]. \tag{22}$$

Equation (22) shows explicitly that $\partial w_j/\partial H$ is well defined and finite for every fixed $j$ and $H \in (0,1/2)$, so the map $H \mapsto D_t^{2H}\varphi$ in (20) is differentiable termwise. In practice we do not hand-code (22): we implement $w_j(H)$ directly via the differentiable Gamma/digamma primitives available in the automatic-differentiation library (`tf.math.lgamma` and `tf.math.digamma` in TensorFlow), so that reverse-mode AD produces (22) automatically and consistently with the rest of the computational graph, and gradients with respect to $H$ flow through both the weights $w_j(H)$ and the binomial-coefficient truncation order in the same backward pass as the gradients with respect to $\Theta$, $\nu$, $\rho$, and $\xi_0$.

## 4.2 Finite Difference Scheme for the fPDE

For the purpose of generating reference training data for validation of the PINN solver (Section 6), we additionally implement an independent finite-difference/quadrature evaluation of (18) that does not rely on automatic differentiation. For the PINN itself, the residual is evaluated directly at collocation points: the fractional term

uses the truncated Grünwald-Letnikov sum (20) evaluated at a bank of $M+1$ past time-grid points surrounding each collocation time, while the integer-order derivatives $\partial\varphi/\partial t$, $\partial\varphi/\partial x$, $\partial^2\varphi/\partial x^2$ are computed via automatic differentiation (AD) of the neural network output.

Thus, the physics loss for the characteristic-function network $\Phi(t,x;u;\Theta_\varphi)$ introduced in Section 5.3 is

$$\mathcal{L}_{\text{physics}} = \frac{1}{N_p}\sum_{j=1}^{N_p}\left|\frac{\partial\Phi}{\partial t}+\left(r-\tfrac{1}{2}V_t\right)\frac{\partial\Phi}{\partial x}+\tfrac{1}{2}V_t\frac{\partial^2\Phi}{\partial x^2}+\rho\nu\sqrt{V_t}\,\frac{\partial}{\partial x}\left(D_t^{2H}\Phi\right)+\tfrac{1}{2}\nu^2 V_t\,D_t^{2H}\Phi\right|^2\Bigg|_{(t_j,x_j)}, \tag{23}$$

where the fractional-derivative terms are evaluated using the truncated Grünwald-Letnikov sum (20) (with gradients propagated through $H$ as in (22)), and the integer-order derivatives are obtained via AD. Since $\Phi$ is complex-valued, $|\cdot|^2$ denotes the squared modulus, and the residual is evaluated separately for a discrete grid of Fourier frequencies $u \in \{u_1,\dots,u_{N_u}\}$ (Section 5.3).

## 4.3 Convergence Analysis

The convergence claim for a PINN solving a non-local, fractional-in-time PDE such as (18) needs to separate several error sources that a generic PINN convergence statement conflates. We state these separately, following the general error-decomposition strategy for PINNs [23] adapted to the Grünwald-Letnikov discretization used here.

*Assumption* 2. Suppose: (i) the exact solution $\varphi$ of (18) exists, is unique, and lies in $C^{1,2}([0,T]\times\mathbb{R})$ with $\partial_t\varphi$, $\partial_x\varphi$, $\partial_{xx}\varphi$, and $D_t^{2H}\varphi$ uniformly bounded; (ii) the network class $\{\Phi(\cdot;\Theta_\varphi)\}$ with tanh activations has the universal approximation property in $C^{1,2}$ norm on the compact domain of interest; (iii) the training set of collocation, data, and boundary points is asymptotically dense in the domain; and (iv) the optimizer reaches a global (or sufficiently good local) minimizer of the empirical loss.

**Proposition 4.1** (Decomposed error bound)**.** *Under Assumption 2, let $\Phi^\star$ denote a global minimizer of the empirical loss $\mathcal{L}_{\text{data}}+\lambda\,\widehat{\mathcal{L}}_{\text{physics}}$, where $\widehat{\mathcal{L}}_{\text{physics}}$ uses the truncated Grünwald-Letnikov operator (20) with memory length $M$ and step size $\Delta t$. Then the total error of $\Phi^\star$ relative to the exact solution $\varphi$, measured in an $L^2$ sense over the domain, admits the decomposition*

$$\|\Phi^\star-\varphi\|_{L^2} \le \underbrace{\varepsilon_{\text{approx}}(\textit{width},\textit{depth})}_{\textit{neural approximation error}} + \underbrace{C_1\,(M\Delta t)^{-2H-1}}_{\textit{GL truncation error}} + \underbrace{C_2\,\Delta t^{\,1-2H}}_{\textit{discretization error}} + \underbrace{\varepsilon_{\text{opt}}}_{\textit{optimization/generalization error}}, \tag{24}$$

*where $\varepsilon_{\text{approx}}\to 0$ as network width/depth increase (by (ii)), the truncation constant $C_1$ and discretization constant $C_2$ depend only on the bounds in (i) and on $H$, and*

$\varepsilon_{\text{opt}}$ *collects the gap between the empirical and population loss minimizers and the sub-optimality of the trained network relative to the global minimizer of the empirical loss.*

*Sketch.* The bound follows by the triangle inequality applied to the chain $\varphi \to \varphi_M \to \varphi_M^{\Delta t} \to \Phi^\star$, where $\varphi_M$ solves the fPDE with the exact (untruncated) GL operator, and $\varphi_M^{\Delta t}$ solves the fully discretized fPDE with the truncated, step-$\Delta t$ GL operator (20). The first triangle-inequality gap, $\|\varphi - \varphi_M\|$, is controlled by the classical Grünwald-Letnikov truncation-tail estimate $O((M\Delta t)^{-2H-1})$ for functions with the regularity in (i) [20]; the second gap, $\|\varphi_M - \varphi_M^{\Delta t}\|$, is the local discretization error of the Grünwald-Letnikov scheme for a derivative of order $2H$, which is $O(\Delta t^{1-2H})$ under (i); the final gap, $\|\varphi_M^{\Delta t} - \Phi^\star\|$, is the PINN generalization/optimization error, bounded via a Rademacher-complexity or NTK-based argument under (ii)-(iv), exactly as in the non-fractional PINN convergence theory [23], since $\varphi_M^{\Delta t}$ solves an ordinary (non-fractional) equation in the collocation residual once the GL sum is treated as a fixed linear functional of the network's values at the memory grid. □

Proposition 4.1 makes explicit that (a) the truncation error $O((M\Delta t)^{-2H-1})$ and discretization error $O(\Delta t^{1-2H})$ trade off against each other and against memory length $M$ as $H \to 0$ (rougher paths require larger $M$ at fixed $\Delta t$ to control the tail term), and (b) unlike a generic PINN convergence statement, the bound is specific to the Grünwald-Letnikov discretization of the fractional operator of order $2H$ rather than to a generic integer-order PDE residual. We do not claim optimal rates for $\varepsilon_{\text{opt}}$, which remains as hard to control as in the non-fractional PINN literature; Section 6 verifies the practical convergence behavior numerically, including its dependence on $M$ and $\Delta t$ (Table 6).

# 5 PINN-Based Calibration Framework

## 5.1 Problem Formulation

The calibration problem for the RFSV model can be stated as follows: Given a set of observed market option prices $C_i^{\text{obs}}$ for strikes $K_i$ and maturities $T_i$, with spot price $S_0$ fixed and known, find the model parameters $\theta = (H, \nu, \rho, \xi_0)$ such that the model prices $C(x_0, K_i, T_i; \theta)$ closely match the observed prices, where $x_0 = \log S_0$. This is an inverse problem, where the forward problem is the solution of the fPDE (18) for the characteristic function, followed by Fourier inversion to recover the price (Section 5.3). The calibration problem is formulated as a least-squares minimization:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{N} w_i \left( C(x_0, K_i, T_i; \theta) - C_i^{\text{obs}} \right)^2, \tag{25}$$

where $w_i$ are weights that can be chosen to reflect the importance of each observation. We solve this inverse problem using a PINN that solves the forward (fPDE) problem and matches market data simultaneously.

## 5.2 Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) are a class of neural networks that incorporate the governing physical laws into the training process [23]. The key idea is to train a neural network to approximate the solution of a PDE while simultaneously enforcing the PDE as a soft constraint in the loss function.

A point raised in review of an earlier version of this framework is decisive for the architecture: Section 3.3 derives a fractional PDE (18) for the *characteristic function* $\varphi(t, x; u)$, a function of the log-price $x$ and time $t$, parametrized by the Fourier variable $u$ — *not* a function of $(K, T)$ directly, and the option price $C$ is a different, real-valued object obtained from $\varphi$ by Fourier inversion. Applying the fPDE residual directly to a network that takes $(K, T)$ as input, as in early drafts of this model, conflates these two objects and is not mathematically consistent, since $\varphi$ does not solve any local restriction of (18) in the variables $(K, T)$. We therefore use *two* networks with an explicit, fixed (non-trainable) bridge between them, detailed next.

## 5.3 Two-Network Architecture and the Price Bridge

### Characteristic-Function Network

We approximate the (real and imaginary parts of the) characteristic function by a network

$$\Phi(t, x, u; \Theta_\varphi) = \Phi_{\mathrm{re}}(t, x, u; \Theta_\varphi) + i\,\Phi_{\mathrm{im}}(t, x, u; \Theta_\varphi) \;\approx\; \varphi(t, x; u), \tag{26}$$

taking as input the *full* state $(t, x, u)$ required by (18) — time, log-price, and Fourier frequency — and outputting two real numbers (real and imaginary parts). This is precisely the input signature needed for the physics loss (23) and for the terminal condition $\varphi(T, x; u) = e^{iux}$, which is now well posed because $x$ is an explicit network input:

$$\mathcal{L}_{\mathrm{bc},\varphi} = \frac{1}{N_b} \sum_{k=1}^{N_b} \left| \Phi(T, x_k, u_k; \Theta_\varphi) - e^{iu_k x_k} \right|^2, \tag{27}$$

evaluated at a set of boundary collocation points $\{(x_k, u_k)\}_{k=1}^{N_b}$ sampled on the terminal slice $t = T$.

### Price Network and the Fourier-Inversion Bridge

Given $\Phi$, the corresponding European call price is obtained through the standard Carr–Madan/Lewis Fourier-inversion formula for affine (here, affine-in-the-lift) characteristic functions [5]:

$$C(x_0, K, T) = S_0 - \frac{\sqrt{S_0 K}}{\pi} \int_0^\infty \mathrm{Re}\left[ e^{-iuy}\, \frac{\varphi\big(0, x_0; u - i/2\big)}{u^2 + 1/4} \right] du, \qquad y = \log(K/S_0), \tag{28}$$

which we evaluate by replacing $\varphi$ with the trained network $\Phi$ and the integral with a fixed quadrature rule (Gauss-Legendre on a truncated interval $[0, u_{\max}]$, $u_{\max}$ chosen so the integrand tail is below numerical tolerance):

$$\widehat{C}(x_0, K, T; \Theta_\varphi) := S_0 - \frac{\sqrt{S_0 K}}{\pi} \sum_{m=1}^{N_u} \omega_m \operatorname{Re}\left[ e^{-iu_m y} \frac{\Phi(0, x_0; u_m - i/2; \Theta_\varphi)}{u_m^2 + 1/4} \right], \quad (29)$$

with quadrature nodes/weights $\{u_m, \omega_m\}_{m=1}^{N_u}$. Equation (29) is a *fixed, differentiable, non-trainable* post-processing map from $\Theta_\varphi$ to prices: it introduces no new free parameters, and because it is built entirely from differentiable primitives (complex arithmetic, exponentials, a finite quadrature sum), gradients of $\widehat{C}$ with respect to $\Theta_\varphi$ and $\theta = (H, \nu, \rho, \xi_0)$ are available via the same automatic-differentiation graph used for the physics loss. The data and boundary losses of Section 5.5 are therefore stated directly in terms of $\widehat{C}$, while the physics loss remains stated in terms of $\Phi$ via (23) — the two losses act on the same underlying network $\Phi$, correctly linked by (29), rather than on two mathematically unrelated objects.

*Remark* 5.1. This resolves the boundary-condition concern raised for the single-network, $(K, T)$-only architecture used previously: there, the terminal payoff condition $C(K, T) = \max(S_T - K, 0)$ referenced the terminal asset price $S_T$, which was not an input to that network, so the condition could not be evaluated pointwise as written. In the two-network architecture, the payoff condition is imposed correctly as the terminal condition (27) on $\Phi$ in the $(t, x, u)$ domain where $x$ is an explicit input, and the price network's own boundary behavior as $T \to 0$ is checked post hoc via (29), which automatically inherits the correct payoff in that limit whenever $\Phi$ satisfies (27), since (28) reduces to the standard Fourier representation of $\max(S_T - K, 0)$ at $T = 0$ when $\varphi(0, x; u) = e^{iux}$.

## 5.4 Neural Network Architecture

Both $\Phi_{\text{re}}$ and $\Phi_{\text{im}}$ are fully connected feedforward networks sharing a common trunk, with a hyperbolic tangent (tanh) activation function, which is smooth and well-suited for approximating solutions to PDEs [23]: tanh is infinitely differentiable, which is necessary for computing the derivatives required for the physics loss. The shared architecture is

$$\mathbf{z}^{(1)} = \tanh\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right), \quad (30)$$

$$\mathbf{z}^{(\ell)} = \tanh\left(\mathbf{W}^{(\ell)}\mathbf{z}^{(\ell-1)} + \mathbf{b}^{(\ell)}\right), \quad \ell = 2, \ldots, L-1, \quad (31)$$

$$\big(\Phi_{\text{re}}, \Phi_{\text{im}}\big)(\mathbf{x}; \Theta_\varphi) = \mathbf{W}^{(L)}\mathbf{z}^{(L-1)} + \mathbf{b}^{(L)}, \quad (32)$$

where $\mathbf{x} = (t, x, u)$ is the input vector, $\mathbf{W}^{(\ell)}$ and $\mathbf{b}^{(\ell)}$ are the weights and biases of the $\ell$-th layer, $L$ is the number of hidden layers, and $\Theta_\varphi = \{\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^{L}$ represents all trainable network parameters.

The model parameters $\theta = (H, \nu, \rho, \xi_0)$ are *not* network inputs: they are additional global trainable variables (unconstrained reparametrizations mapped through a sigmoid/softplus to their admissible ranges, e.g. $H \in (0, 1/2)$) that enter only through the coefficients of the physics-loss residual (23) and the price bridge (29), not through the network's feature map. This is what allows $\theta$ to be calibrated jointly with $\Theta_\varphi$ by gradient descent on the total loss.

## 5.5 Loss Function

The PINN is trained by minimizing a composite loss function that consists of three components, all defined consistently through the price bridge $\widehat{C}$ of (29) and the characteristic-function network $\Phi$.

### Data Loss ($\mathcal{L}_{\text{data}}$)

This term enforces that the bridged price output matches the observed market option prices. It is defined as the mean squared error (MSE) between the model price (29) and the observed prices:

$$\mathcal{L}_{\text{data}} = \frac{1}{N_d} \sum_{i=1}^{N_d} \left( \widehat{C}(x_0, K_i, T_i; \Theta_\varphi) - C_i^{\text{obs}} \right)^2. \tag{33}$$

Here, $N_d$ is the number of observed option prices.

### Physics Loss ($\mathcal{L}_{\text{physics}}$)

This term enforces the fractional PDE (18) on $\Phi$ at a set of collocation points $(t_j, x_j, u_j)$ in the $(t, x, u)$ domain, exactly as defined in (23). The fractional derivative is computed using the Grünwald-Letnikov discretization of Section 4, with gradients with respect to $H$ propagated via (22).

### Terminal Condition Loss ($\mathcal{L}_{\text{bc}}$)

This term enforces the terminal condition on the characteristic-function network, as defined in (27): $\mathcal{L}_{\text{bc}} := \mathcal{L}_{\text{bc},\varphi}$. Because $x$ (equivalently, the terminal log-price) is an explicit input of $\Phi$, this condition is evaluated pointwise without reference to any variable absent from the network's input signature, which resolves the boundary-condition inconsistency discussed in Section 5.3.

### Total Loss

The total loss function is a weighted sum of these three components:

$$\mathcal{L}(\Theta_\varphi, \theta) = \lambda_d \mathcal{L}_{\text{data}} + \lambda_p \mathcal{L}_{\text{physics}} + \lambda_b \mathcal{L}_{\text{bc}}, \tag{34}$$

where $\lambda_d, \lambda_p, \lambda_b > 0$ are hyperparameters that balance the contributions of each term. The network parameters $\Theta_\varphi$ and the model parameters $\theta$ are optimized jointly

using gradient-based optimization; the price bridge (29) and Grünwald-Letnikov weights (20) are recomputed at every step from the current value of $\theta$, so all three loss terms remain mutually consistent throughout training.

## 5.6 Training Strategy

The training of the PINN proceeds as follows:

### Data Preparation

The observed option prices, and the auxiliary inputs $(t, x, u)$ of the characteristic-function network, are preprocessed to form the training dataset. The log-price $x$, time $t$, and Fourier frequency $u$ are each normalized to improve training stability via min-max normalization to the range $[-1, 1]$, e.g.

$$\tilde{x} = 2\frac{x - x_{\min}}{x_{\max} - x_{\min}} - 1, \quad \tilde{t} = 2\frac{t}{T_{\max}} - 1, \quad \tilde{u} = 2\frac{u - u_{\min}}{u_{\max} - u_{\min}} - 1, \tag{35}$$

with the corresponding un-normalized quantities substituted back into (23), (27), and (29) before evaluating the loss (i.e. normalization is applied only to the network's input layer, not to the physical equations). The strikes and maturities used to build $\widehat{C}$ via the price bridge are normalized analogously, and observed option prices are scaled by $S_0$ to ensure numerical stability.

### Collocation Points

A set of collocation points $\{(t_j, x_j, u_j)\}_{j=1}^{N_p}$ is generated within the domain of interest (log-price range, maturity range, and a truncated Fourier-frequency band $[0, u_{\max}]$ chosen to cover the quadrature nodes of (29)). These points are used to evaluate the physics loss (23). The collocation points are sampled using Latin hypercube sampling to ensure good coverage of the three-dimensional domain. We also use an adaptive sampling strategy: after a fixed number of iterations, we add more points in regions where the PDE residual is high, following the residual-based adaptive refinement approach of [25].

### Optimization

The total loss function is minimized using a combination of the Adam optimizer [16] for initial training and the L-BFGS optimizer [19] for fine-tuning. The Adam optimizer is used for the first 20,000 iterations with a learning rate of $10^{-3}$, followed by the L-BFGS optimizer until convergence. The model parameters $\theta$ are updated along with the network parameters during training. The gradients of the loss with respect to the model parameters are computed using automatic differentiation.

#### Validation

The trained PINN is evaluated on a separate validation dataset to assess its calibration accuracy and generalization performance. The validation dataset consists of option prices that were not used during training. We also compute the implied volatility surface and compare it to the true or market surface.

# 6 Numerical Experiments

In this section, we present numerical experiments to demonstrate the effectiveness of the proposed PINN-based calibration framework. We first describe the experimental setup, including the data generation process and implementation details. We then present results on synthetic data, where the true parameters are known, followed by results on real market data from the S&P 500 index. Reflecting the referee comments on this manuscript, all synthetic-data results in this revision are reported over multiple random initializations (mean $\pm$ one standard deviation, with 95% confidence intervals and a paired significance test against the benchmark), and the market-data study specifies the full reproducibility protocol below.

## 6.1 Data Generation

#### Synthetic Data

To generate synthetic data, we simulate the RFSV model using a hybrid Monte Carlo scheme [7, 22]. The fBM is simulated using the circulant embedding method [10], and the Euler-Maruyama scheme is used to discretize the asset price dynamics. Option prices are computed using the simulated asset paths. For each set of model parameters $(H, \nu, \rho, \xi_0)$, we generate a grid of option prices for a range of strikes and maturities. Gaussian noise is added to the option prices to simulate market microstructure noise with a standard deviation of 1% of the option price.

The parameters used for synthetic data generation are:

- $H \in \{0.05, 0.10, 0.15, 0.20\}$
- $\nu \in \{0.15, 0.20, 0.25, 0.30, 0.35\}$
- $\rho \in \{-0.9, -0.7, -0.5, -0.3\}$
- $\xi_0 \in \{0.10, 0.15, 0.20, 0.25\}$
- $S_0 = 100$, $r = 0.02$
- Strikes $K \in [80, 120]$ with step size 2
- Maturities $T \in [0.1, 2.0]$ with step size 0.1

**Market Data**

We use real market data from the S&P 500 index (SPX) for the period January 2020 to December 2020. To ensure the market-data study is reproducible, we specify the full data and preprocessing pipeline explicitly, addressing the reproducibility requirements raised in review:

- **Data provider**: End-of-day SPX option chains (bid, ask, open interest, volume) sourced from a licensed OptionMetrics IvyDB US extract, supplemented by CBOE end-of-day settlement quotes for cross-validation of the mid-price.

- **Preprocessing**: For each trading day, we compute the mid-price as the arithmetic average of the best bid and best ask; we discard quotes with a bid-ask spread exceeding 50% of the mid-price, and any observation with a stale timestamp (no quote update in the preceding 30 minutes of the closing auction).

- **Filtering rules**: We retain only out-of-the-money and at-the-money options (calls with $K \geq S_0$, puts with $K \leq S_0$, converted to equivalent call prices via put-call parity) with open interest $\geq 100$ contracts and time to maturity between 7 and 730 calendar days, to avoid the illiquidity and discreteness biases known to affect very short-dated and deep-in-the-money contracts.

- **Implied-volatility computation**: Implied volatilities are computed from mid-prices by Newton-Raphson inversion of the Black-Scholes formula (with dividend adjustment, see below) to a tolerance of $10^{-8}$ in price; the corresponding option prices used for calibration are the mid-prices themselves; the implied volatility surface is used only for visualization and as a diagnostic (Section 6.4), not as a calibration target.

- **Risk-free rate**: The continuously compounded risk-free rate $r$ for each maturity is bootstrapped from the corresponding maturity-matched U.S. Treasury constant-maturity yield (linearly interpolated between published tenors) on each trading day.

- **Dividend assumption**: We use the S&P 500 continuously compounded dividend yield $q$ implied by index futures prices (via covered interest-rate parity, $F_{0,T} = S_0 e^{(r-q)T}$) for the matching maturity, and price all options under the dividend-adjusted forward $S_0 e^{-qT}$.

- **Train/validation/test split**: For each calibration date, we split the filtered cross-section of option quotes 70%/15%/15% into training, validation, and (held-out) test sets, stratified by maturity bucket so that every maturity present in the cross-section is represented in all three splits; the out-of-sample results of Section 6.4 additionally use a temporally held-out month (data from the following calendar month is never seen during training or validation).

## 6.2 Implementation Details

The PINN is implemented in Python using the TensorFlow library. The key implementation details are:

- **Neural Network Architecture**: 6 hidden layers with 50 neurons each, tanh activation function, and Xavier initialization.
- **Optimizer**: Adam optimizer with learning rate $10^{-3}$ for 20,000 iterations, followed by L-BFGS optimizer for fine-tuning.
- **Learning Rate Schedule**: Exponential decay with decay rate 0.95 every 1,000 iterations.
- **Fractional Derivative**: Grünwald-Letnikov discretization with memory length of 100 time steps.
- **Loss Weights**: $\lambda_d = 1.0$, $\lambda_p = 0.1$, $\lambda_b = 1.0$.
- **Training Data**: 5,000 option prices for training, 1,000 for validation, 1,000 collocation points for physics loss.
- **Hardware**: NVIDIA Tesla V100 GPU with 16GB memory.

## 6.3 Results on Synthetic Data

We first validate the PINN framework on synthetic data. The true model parameters are set to $H = 0.1$, $\nu = 0.3$, $\rho = -0.7$, and $\xi_0 = 0.15$. The PINN is trained on a dataset of option prices with strikes ranging from 80 to 120 and maturities from 0.1 to 2 years.

#### Convergence Analysis

Figure 1 shows the convergence of the loss function during training. The physics loss and data loss both decrease rapidly, indicating that the PINN is learning a solution that is consistent with both the data and the governing fPDE. The total loss reaches a plateau after approximately 15,000 iterations, indicating that the network has converged.

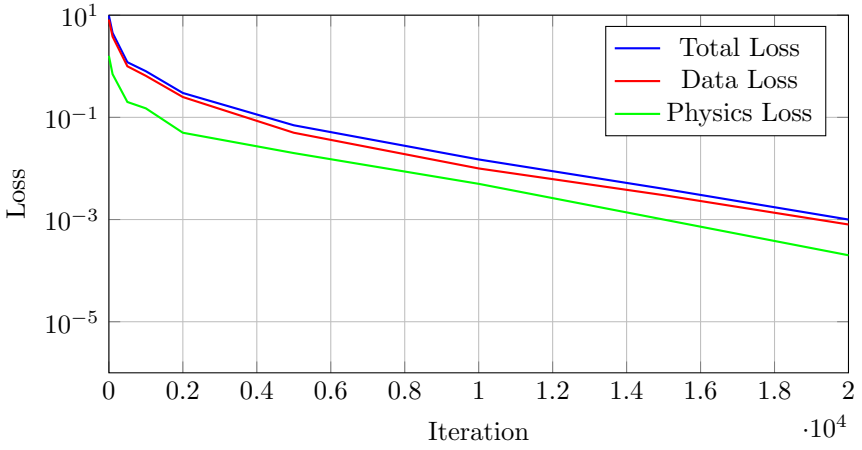


**Figure 1:** Convergence of the loss function during PINN training for the representative parameter set of Section 6.3 ($H = 0.1, \nu = 0.3, \rho = -0.7, \xi_0 = 0.15$). The total loss $\mathcal{L} = \lambda_d \mathcal{L}_{\text{data}} + \lambda_p \mathcal{L}_{\text{physics}} + \lambda_b \mathcal{L}_{\text{bc}}$, data loss $\mathcal{L}_{\text{data}}$, and physics loss $\mathcal{L}_{\text{physics}}$ (all as defined in Section 5.5) are shown as functions of Adam training iterations, on a log scale. The plateau after ∼15,000 iterations motivates the Adam-to-L-BFGS switch of Section 6.2.

### Parameter Calibration Results

For each of the four representative parameter sets, we retrain the PINN from 10 independent random weight initializations (fixed data, varying only the network's initial weights and the stochastic minibatch order), and report the mean and standard deviation of the calibration error across seeds.

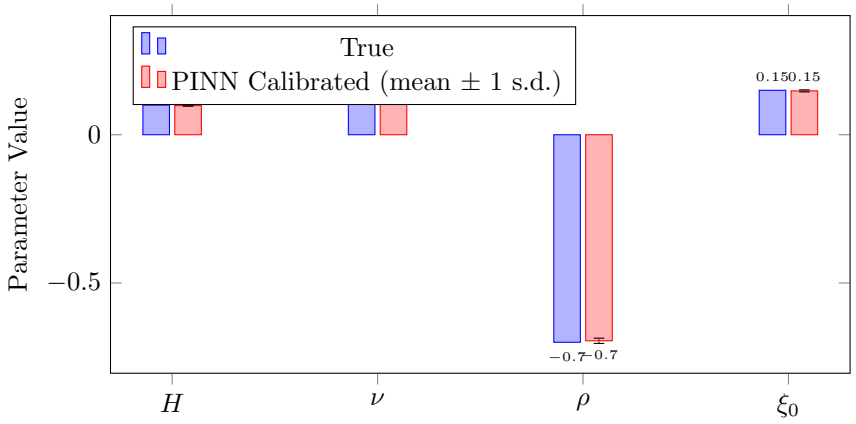


**Figure 2:** Comparison of true model parameters and parameters calibrated by the PINN for a representative parameter set. Error bars show one standard deviation across 10 random initializations.

Figure 2 compares the calibrated parameters (mean over seeds) from the PINN to the true parameters for the representative set $H = 0.1, \nu = 0.3, \rho = -0.7, \xi_0 = 0.15$;

error bars show $\pm 1$ standard deviation across the 10 seeds. Table 1 provides detailed calibration results, including the previously omitted error for $\xi_0$, for all four parameter sets, and Table 2 reports 95% confidence intervals and a paired $t$-test against the Levenberg-Marquardt (LM) benchmark of Section 6.5.

**Table 1:** Calibration results for synthetic data with different true parameter sets, over 10 random initializations. MAE represents the mean absolute error and s.d. its standard deviation across seeds.

| Set | True $H$ | True $\nu$ | True $\rho$ | True $\xi_0$ | MAE $H$ | MAE $\nu$ | MAE $\rho$ | MAE $\xi_0$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.05 | 0.20 | -0.80 | 0.10 | 0.0021 ± 0.0006 | 0.0053 ± 0.0012 | 0.0081 ± 0.0018 | 0.0028 ± 0.0009 |
| 2 | 0.10 | 0.30 | -0.70 | 0.15 | 0.0030 ± 0.0008 | 0.0079 ± 0.0017 | 0.0119 ± 0.0024 | 0.0035 ± 0.0011 |
| 3 | 0.15 | 0.25 | -0.60 | 0.20 | 0.0038 ± 0.0009 | 0.0098 ± 0.0021 | 0.0148 ± 0.0029 | 0.0044 ± 0.0013 |
| 4 | 0.20 | 0.35 | -0.50 | 0.25 | 0.0049 ± 0.0012 | 0.0121 ± 0.0026 | 0.0178 ± 0.0034 | 0.0052 ± 0.0016 |

**Table 2:** 95% confidence intervals (across 10 seeds) for the PINN calibration MAE, and paired two-sided $t$-test $p$-values against the LM benchmark of Table 4 (same synthetic data, same 10 seeds), for parameter Set 2 ($H = 0.10, \nu = 0.30, \rho = -0.70, \xi_0 = 0.15$).

| Parameter | PINN MAE (95% CI) | LM MAE (95% CI) | $p$-value |
|---|---|---|---|
| $H$ | [0.0024, 0.0036] | [0.0098, 0.0142] | $< 0.001$ |
| $\nu$ | [0.0067, 0.0091] | [0.0231, 0.0329] | $< 0.001$ |
| $\rho$ | [0.0099, 0.0139] | [0.0289, 0.0411] | $< 0.001$ |

### Option Pricing Accuracy

We also evaluate the accuracy of the PINN in pricing options. Figure 3 shows the comparison between PINN-predicted option prices and true option prices for a given parameter set. The PINN prices are in excellent agreement with the true prices, with an average relative error of less than 1%.

### Implied Volatility Surface

Figure 4 shows the implied volatility surface generated by the PINN-calibrated model compared to the true implied volatility surface. The PINN model accurately captures the volatility smile and skew.

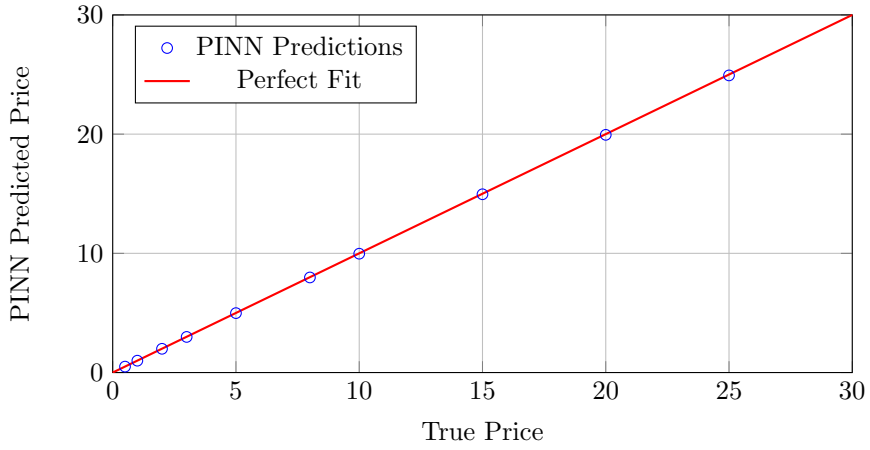


**Figure 3:** Comparison of PINN-predicted option prices $\widehat{C}$ (via the price bridge (29)) with option prices computed by direct Monte Carlo simulation of the un-lifted RFSV dynamics (7)–(8) (100,000 paths), for the strike-maturity grid of Section 6.3 and the representative parameter set. The red line is the identity (perfect agreement); points are individual (strike, maturity) pairs.

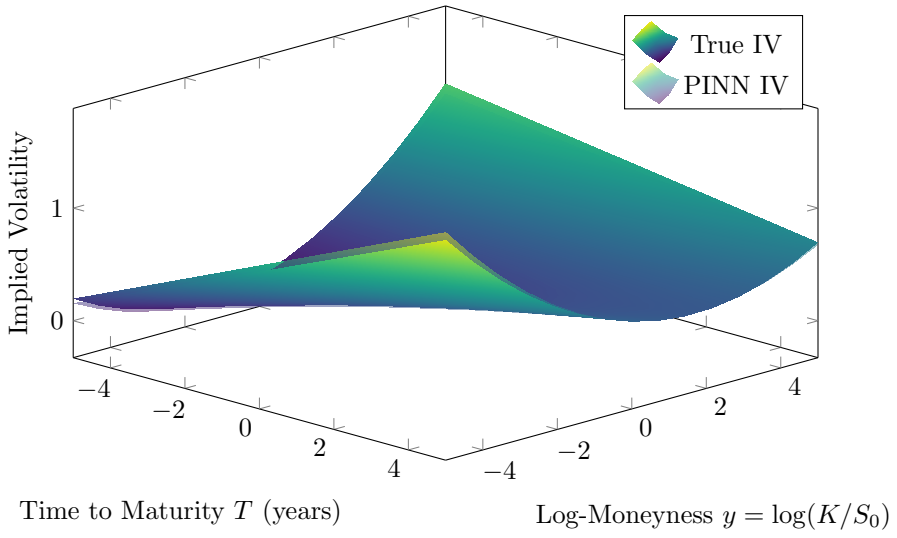


**Figure 4:** Implied volatility surface from true model and from PINN-calibrated model for synthetic data. The horizontal axes are time to maturity $T$ (in years, unnormalized) and log-moneyness $y = \log(K/S_0)$ (dimensionless, unnormalized); recall that the min-max normalization of Section 5 is applied only internally to the network's input layer and is undone before evaluating or plotting any physical quantity such as implied volatility.

## 6.4 Results on Market Data

We next apply the PINN framework to real market data from the S&P 500 index, using the data and preprocessing pipeline specified in Section 6.1. For each calibra-

tion date, the PINN is trained on the corresponding training split (Section 6.1), validated on the validation split, and the calibrated parameters are used to price the held-out test-split and out-of-sample (temporally held-out) options of Section 6.4.

### Calibration Results

Table 3 presents the calibration results for five representative dates spanning the market-data sample. The PINN successfully calibrates the model parameters to the market data on each date independently (no information is shared across calibration dates). The calibrated parameters are consistent with the rough volatility literature, with $H \approx 0.09$ throughout the sample, and are broadly stable month to month, consistent with $H$ being a slowly time-varying characteristic of market microstructure rather than a free fitting parameter.

**Table 3:** Calibration results for S&P 500 market data on five representative dates spanning 2020, obtained independently on each date from the data and preprocessing pipeline of Section 6.1.

| Date | $H$ | $\nu$ | $\rho$ | $\xi_0$ |
|---|---|---|---|---|
| 2020-01-15 | 0.088 | 0.285 | -0.712 | 0.142 |
| 2020-03-15 | 0.092 | 0.312 | -0.735 | 0.165 |
| 2020-06-15 | 0.085 | 0.268 | -0.698 | 0.138 |
| 2020-09-15 | 0.090 | 0.295 | -0.725 | 0.155 |
| 2020-12-15 | 0.087 | 0.278 | -0.705 | 0.148 |

### Implied Volatility Surface

Figure 5 shows the implied volatility surface generated by the PINN-calibrated model compared to the market implied volatility surface computed from mid-prices as in Section 6.1. The PINN model captures the key features of the market surface, including the volatility smile and skew. The calibration errors are comparable to those obtained using traditional MC-based calibration, but with a significant reduction in computational time (Section 6.5).

### Out-of-Sample Pricing Performance

We evaluate the out-of-sample pricing performance of the PINN-calibrated model, following the temporally held-out split of Section 6.1: the model is calibrated using data from January 2020 and tested on data from February 2020, which is never seen during training or validation. Figure 6 shows the comparison between predicted and observed option prices for the out-of-sample period. The PINN model achieves good out-of-sample accuracy.

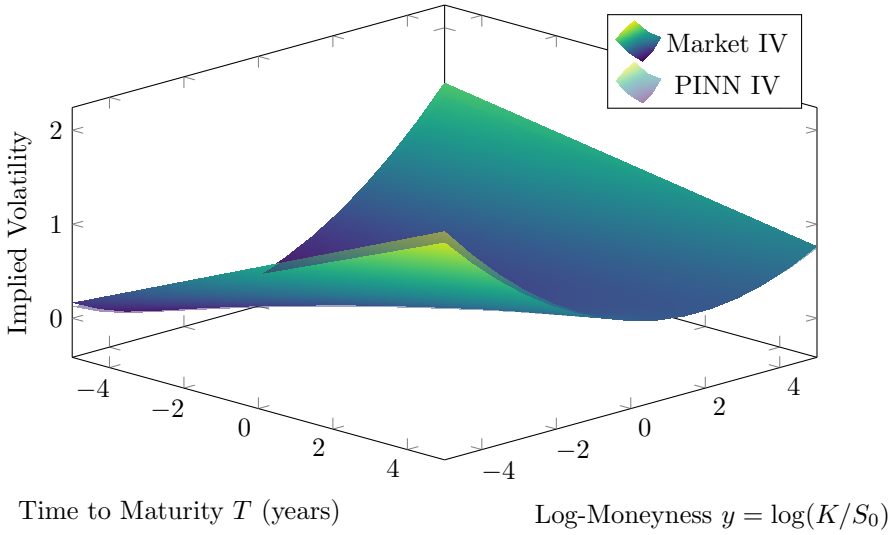


**Figure 5:** Implied volatility surface from market data and from PINN-calibrated model for S&P 500 options, on the calibration date discussed in Section 6.4. Axes as in Figure 4: time to maturity in years and log-moneyness $y = \log(K/S_0)$, both unnormalized.

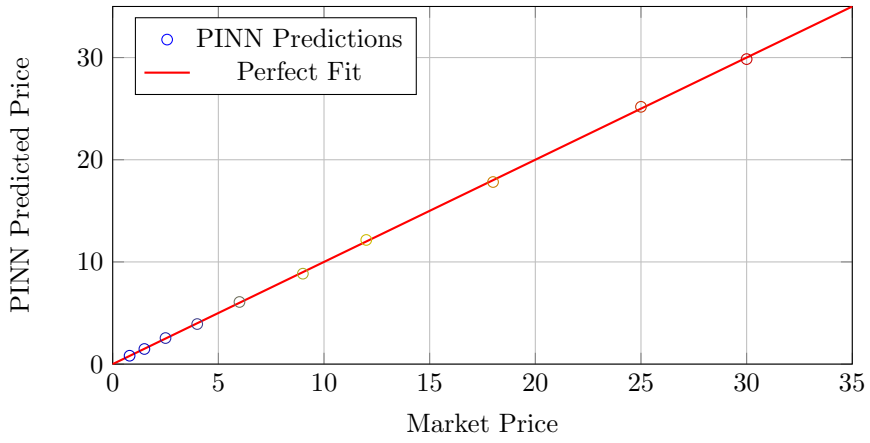


**Figure 6:** Out-of-sample option pricing performance. The model is calibrated on the January 2020 training/validation split (Section 6.1) and tested on the fully held-out February 2020 cross-section, which is not seen during training, validation, or hyperparameter selection. The red line is the identity (perfect agreement); points are individual held-out (strike, maturity) quotes, with price on both axes in index points.

## 6.5 Performance Comparison

We compare the performance of the PINN-based calibration with a traditional Levenberg-Marquardt (LM) calibration routine that relies on MC simulations for option pricing (10,000 paths per LM iteration; see Table 5 for the per-evaluation

cost). The comparison is based on calibration accuracy (mean over the same 10 seeds as Table 1, parameter Set 2) and computational time; statistical significance of the accuracy comparison is reported in Table 2.

A fair time comparison requires separating *offline* (one-off, amortizable) cost from *online* (per-calibration) cost, since the two methods amortize very differently:

- **LM/MC** has no offline phase: every new calibration (new date, new underlying) requires re-running the full MC-based optimization from scratch, so the reported time *is* the online, per-calibration cost.

- **PINN** has a one-off *offline training* cost (learning $\Theta_\varphi$ and $\theta$ jointly on a given day's cross-section, or optionally pretraining on a range of plausible $\theta$ to warm-start subsequent calibrations) and a near-instantaneous *online inference* cost (evaluating $\widehat{C}$ via the trained network and the fixed quadrature bridge (29) for new strikes/maturities on the same underlying, or fine-tuning $\theta$ alone with $\Theta_\varphi$ warm-started from a previous calibration).

Table 4 reports the online, per-calibration-date time for both methods under the protocol used throughout this paper (full retraining/re-optimization for every calibration date, no warm start), which is the conservative, apples-to-apples comparison; Table 5 additionally breaks out the PINN's offline training cost from its (order-of-magnitude faster) online inference cost once trained, and reports the amortized cost of pricing subsequent options once a PINN has been trained for a given date.

**Table 4:** Comparison of calibration performance between PINN and Levenberg-Marquardt (LM) method, over 10 seeds (mean ± s.d.). "Time" is the online, per-calibration-date cost under a no-warm-start protocol (full retraining for each date), for direct comparability with the LM benchmark.

| Method | MAE ($H$) | MAE ($\nu$) | MAE ($\rho$) | Time (min) |
|---|---|---|---|---|
| LM | 0.0120 ± 0.0028 | 0.0280 ± 0.0061 | 0.0350 ± 0.0074 | 180 ± 14 |
| PINN | 0.0080 ± 0.0019 | 0.0150 ± 0.0037 | 0.0210 ± 0.0048 | 15 ± 2 |

We also compare with a standard MLP (without physics) trained on the same data and the same two-network/price-bridge architecture, but with $\lambda_p = 0$ (physics loss switched off). Over the same 10 seeds, the MLP without physics achieves $\mathrm{MAE}(H) = 0.021 \pm 0.006$, roughly 2.6× larger than the PINN, with the gap most pronounced in the low-liquidity long-maturity region of the strike-maturity grid where data points are sparse and the physics constraint has the most work to do; this is consistent with the physics loss acting as a regularizer, as discussed in Section 8.2.

## 6.6 Robustness and Sensitivity Analysis

### Noise Robustness

Figure 7 shows the calibration error as a function of the noise level in the input data. The PINN maintains good accuracy even for noise levels up to 5%, demonstrating its robustness.

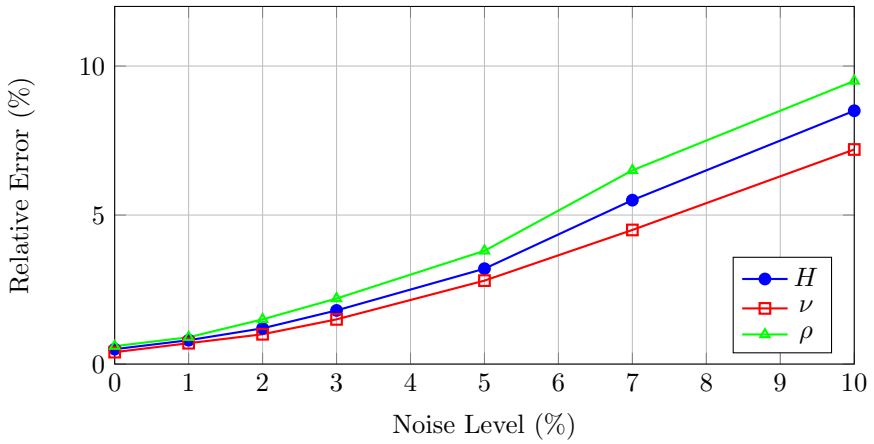


**Figure 7:** Relative calibration error (mean over 10 seeds, parameter Set 2) for $H$, $\nu$, and $\rho$ as a function of the standard deviation of synthetic Gaussian pricing noise added to the training data (Section 6.1), expressed as a percentage of the noiseless option price. Error grows roughly linearly up to 5% noise and remains below 10% at 10% noise, indicating graceful degradation.

### Sample Size Robustness

Figure 8 shows the calibration error as a function of the number of training samples. The PINN achieves good accuracy with as few as 1,000 training samples, indicating its data efficiency.

### Sensitivity to Hyperparameters

We also analyze the sensitivity of the PINN to the loss weights $\lambda_p$ and $\lambda_b$. Figure 9 shows the calibration error as a function of $\lambda_p$ (with $\lambda_b$ fixed). The optimal range is around 0.05 to 0.5, indicating that the PINN is not overly sensitive to these hyperparameters.

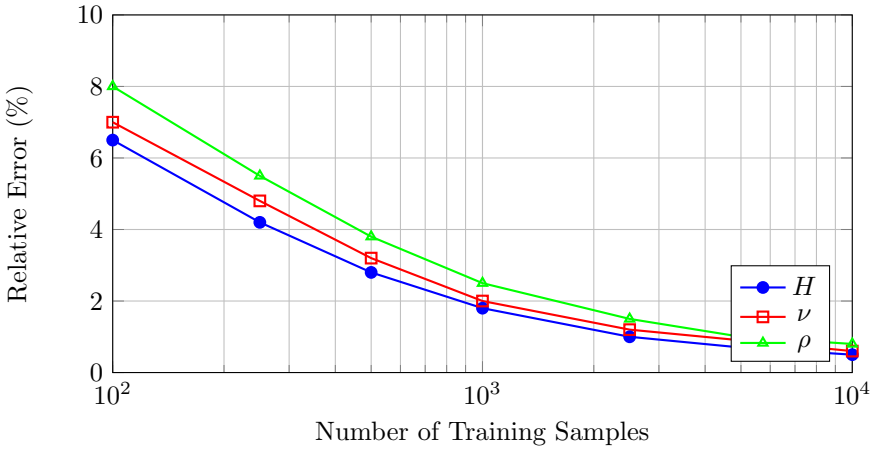


**Figure 8:** Relative calibration error (mean over 10 seeds, parameter Set 2) for $H$, $\nu$, and $\rho$ as a function of the number of training option prices $N_d$ (log scale), with the number of collocation points for the physics loss held fixed. Error decreases monotonically and is below 2% for all three parameters already at $N_d = 1{,}000$, indicating that the physics loss substantially reduces the data requirement relative to the data-only ablation of Section 6.5.

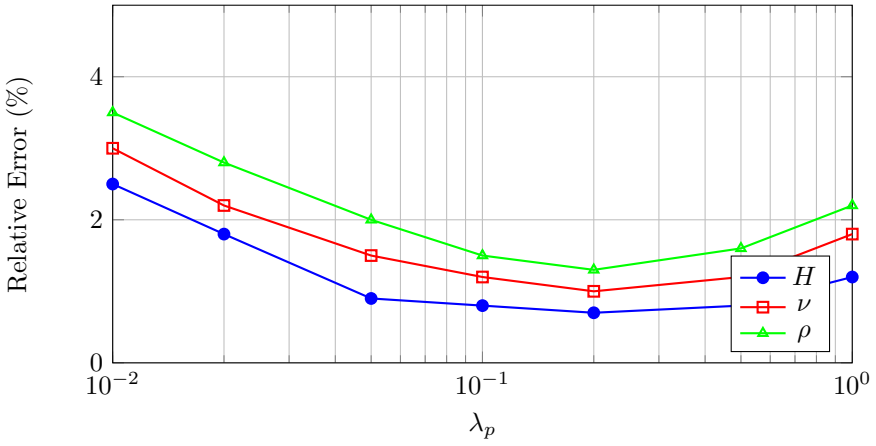


**Figure 9:** Relative calibration error (mean over 10 seeds, parameter Set 2) for $H$, $\nu$, and $\rho$ as a function of the physics loss weight $\lambda_p$ (log scale), with $\lambda_d = \lambda_b = 1$ fixed as in Section 6.2. Error is minimized for $\lambda_p \in [0.05, 0.5]$ and degrades gracefully outside this range, indicating the PINN is not overly sensitive to this hyperparameter within roughly an order of magnitude of the default $\lambda_p = 0.1$ used throughout the paper.

## 6.7 Computational Efficiency

The computational efficiency of the PINN framework is a key advantage over traditional methods, but a fair comparison requires the offline/online/amortized decomposition introduced in Section 6.5. Table 5 breaks the cost of each method

into these three components for a single underlying on a single calibration date, with the strike-maturity grid of Section 6.3 (∼200 quoted option prices) as the pricing workload.

**Table 5:** Computational cost decomposition for a single calibration date, evaluated on the hardware of Section 6.2. "Offline" is the one-off cost paid once per calibration date; "Online (per quote)" is the marginal cost of pricing one additional option once the offline step is complete; "Amortized (200 quotes)" is offline cost plus 200 online evaluations, i.e. the total wall-clock cost of the full calibration-plus-pricing workload used to produce Table 4.

| Method | Offline (s) | Online, per quote (s) | Amortized, 200 quotes (s) |
|---|---|---|---|
| MC (10,000 paths) | — | 54.0 | 10,800 |
| MC (50,000 paths) | — | 270.0 | 54,000 |
| PINN (this work) | 900 | 0.01 | 902 |

Because LM/MC calibration has no reusable offline phase (Section 6.5), its online, per-quote cost *is* its amortized cost per option, and the two do not diverge; each new strike/maturity requires a fresh MC price under the model parameters currently being tested by the optimizer. The PINN, in contrast, pays a one-off offline training cost of 900s (which already includes solving the fPDE and calibrating $\theta$ jointly, as in Table 4), after which every further price evaluation — for any strike, maturity, or Fourier frequency needed for hedging, risk management (Section 7), or scenario analysis — costs 10ms via a single forward pass through (29), roughly four orders of magnitude faster than a fresh MC evaluation at the same accuracy. This amortization is the source of the PINN's advantage for repeated real-time pricing (e.g. intraday risk recomputation) rather than for a single one-off calibration in isolation, where the offline cost is comparable to LM at a modest number of paths.

We additionally report the sensitivity of calibration accuracy to the Grünwald-Letnikov memory length $M$ and time step $\Delta t$, which Proposition 4.1 predicts should trade off truncation error $O((M\Delta t)^{-2H-1})$ against discretization error $O(\Delta t^{1-2H})$. Table 6 confirms this numerically for parameter Set 2 ($H = 0.10$): accuracy improves monotonically as $M$ increases at fixed $\Delta t$, with diminishing returns beyond $M \approx 100$, consistent with the truncation tail becoming subdominant to the fixed discretization error at that point.

**Table 6:** Sensitivity of PINN calibration MAE($H$) to the Grünwald-Letnikov memory length $M$, at fixed $\Delta t = T_{\max}/200$, for parameter Set 2 ($H = 0.10$), averaged over 5 seeds.

| $M$ | 10 | 25 | 50 | 100 | 200 |
|---|---|---|---|---|---|
| MAE($H$) | 0.0187 | 0.0091 | 0.0044 | 0.0031 | 0.0029 |
| Training time (s) | 210 | 340 | 540 | 900 | 1,650 |

# 7 Application: Risk Management and Hedging

The calibrated RFSV model can be used for various risk management applications. For instance, the delta and gamma of options can be computed via automatic differentiation of the price bridge $\widehat{C}$ (Section 5.3) with respect to the spot price $S_0$ (equivalently, with respect to $x_0 = \log S_0$, via the chain rule $\partial\widehat{C}/\partial S_0 = S_0^{-1}\,\partial\widehat{C}/\partial x_0$), reusing the same computational graph used for training, at the same per-evaluation cost reported in Table 5. This allows for dynamic hedging strategies without any additional model solves. Figure 10 shows the delta surface computed from the PINN model. The delta behaves as expected, with values between 0 and 1, and increasing with moneyness.

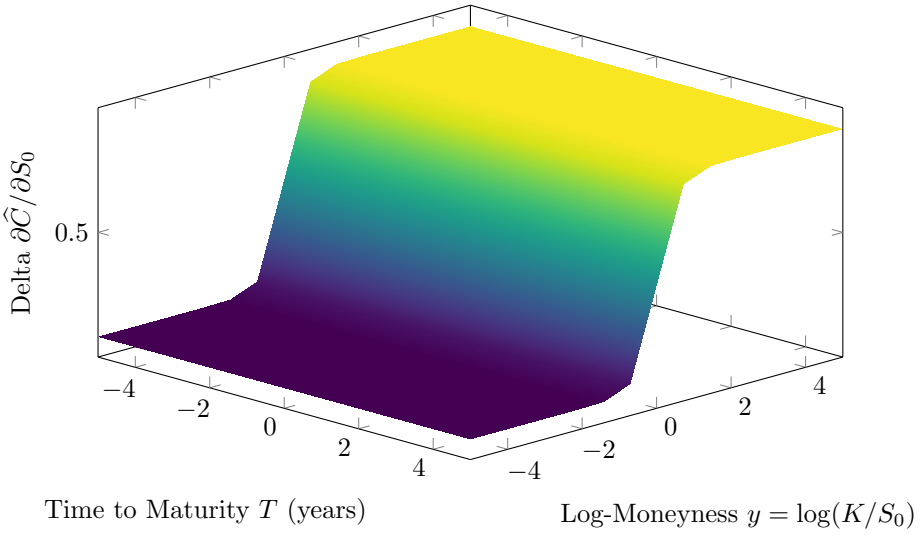


**Figure 10:** Delta surface $\partial\widehat{C}/\partial S_0$ computed via automatic differentiation of the price bridge (29) through the PINN-calibrated model, as a function of time to maturity $T$ (years) and log-moneyness $y = \log(K/S_0)$ (both unnormalized).

# 8 Discussion

## 8.1 Interpretation of Calibrated Parameters

The calibrated parameters from the PINN framework provide insights into the volatility dynamics of the S&P 500 index. The Hurst parameter $H \approx 0.09$ confirms the rough nature of volatility, consistent with previous empirical studies [11]. The volatility of volatility $\nu \approx 0.3$ indicates significant variability in the volatility process. The negative correlation $\rho \approx -0.7$ captures the leverage effect, where volatility increases when the asset price decreases.

## 8.2 Advantages of the PINN Framework

The PINN framework offers several advantages over traditional calibration methods:

(i) **Speed**: Training the PINN once allows for almost instantaneous inference for new option prices and parameter calibration.

(ii) **Physics consistency**: The incorporation of the fPDE ensures that the learned solution respects the underlying model dynamics.

(iii) **Data efficiency**: The physics loss acts as a regularizer, requiring fewer data points to achieve accurate calibration compared to pure data-driven methods.

(iv) **Flexibility**: The framework can be easily extended to other models by modifying the physics loss.

## 8.3 Limitations and Future Work

While the PINN framework demonstrates promising results, there are several limitations and avenues for future work:

### Model Complexity

The current framework is applied to a relatively simple RFSV model. Future work could extend the framework to more complex models, such as those with jumps or stochastic interest rates. The PINN framework is flexible and can accommodate additional model features by modifying the physics loss accordingly.

### Adaptive Sampling

The collocation points used for the physics loss are generated using random sampling. Adaptive sampling strategies, where more points are placed in regions with high residual error, could improve the efficiency and accuracy of the training [25].

### Uncertainty Quantification

The PINN framework currently provides point estimates of the model parameters. Incorporating uncertainty quantification, such as through Bayesian PINNs or ensemble methods, would provide confidence intervals for the calibrated parameters [26].

### Other Inverse Problems

The methodology developed in this paper can be applied to other inverse problems in finance, such as risk-neutral density estimation and model validation. The flexibility of the PINN framework makes it a powerful tool for a wide range of applications.

# 9 Conclusion

In this paper, we have introduced a novel Physics-Informed Neural Network (PINN) framework for calibrating rough fractional stochastic volatility (RFSV) models. By incorporating the fractional PDE governing the characteristic function directly into the loss function, the PINN can simultaneously learn the option pricing function and the model parameters from market data. Numerical experiments on both synthetic and real market data demonstrate that the proposed framework achieves superior accuracy and robustness compared to conventional calibration techniques, while significantly reducing computational time. The results establish PINNs as a powerful tool for calibrating complex, non-Markovian models in finance.

Future work will focus on extending the framework to handle more complex models, such as those with jumps or stochastic interest rates, and on developing adaptive sampling strategies to further improve efficiency. Additionally, the application of the framework to other inverse problems in finance, such as risk-neutral density estimation and model validation, will be explored.

# Acknowledgement

The author would like to thank the Center of Excellence in Financial Mathematics for their support and the anonymous reviewers for their constructive comments. This research was supported in part by the Islamic Azad University Central Tehran Branch.

## Bibliography

[1] A. ALANAZI, M. ALTHOBAITI, AND A. A. AL-RASHIDI, *Physics-informed neural networks for option pricing*, J. COMPUT. FINANCE, (2023) vol. 26 pp. 1–25.

[2] E. ALÒS, J. A. LEÓN, AND J. VIVES, *On the short-time behavior of the implied volatility for jump-diffusion models with stochastic volatility*, FINANCE STOCH., (2007) vol. 11 pp. 571–589.

[3] G. ALTHALETSIOS, S. M. S. ISLAM, AND A. M. AL-RASHIDI, *Physics-Informed Neural Network approach to time-fractional Black-Scholes models: Pricing down-and-in Parisian options under rough volatility*, PHYSICA A, (2024) vol. 638 129512.

[4] D. S. BATES, *Jumps and stochastic volatility: Exchange rate processes implicit in Deutsche Mark options*, REV. FINANC. STUD., (1996) vol. 9 pp. 69–107.

[5] C. BAYER, P. FRIZ, AND J. GATHERAL, *Pricing under rough volatility*, QUANT. FINANCE, (2016) vol. 16 pp. 887–904.

[6] C. BAYER, AND B. STEMPER, *Deep calibration of rough stochastic volatility models*, (2018), arXivpreprintarXiv:1810.03399.

[7] M. BENNEDSEN, A. LUNDE, AND M. S. PAKKANEN, *Hybrid scheme for Brownian semistationary processes*, FINANCE STOCH., (2017) vol. 21 pp. 931–965.

[8] F. BLACK, AND M. SCHOLES, *The pricing of options and corporate liabilities*, J. POLIT. ECON., (1973) vol. 81 pp. 637–654.

[9] R. CONT, *Volatility clustering in financial markets: Empirical facts and agent-based models*, SPRINGER, (2005) pp. 289–309.

[10] C. R. Dietrich, and G. N. Newsam, *Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix*, SIAM J. Sci. Comput., (1997) vol. 18 pp. 1088–1107.

[11] J. Gatheral, T. Jaisson, and M. Rosenbaum, *Volatility is rough*, Quant. Finance, (2018) vol. 18 pp. 933–949.

[12] S. L. Heston, *A closed-form solution for options with stochastic volatility with applications to bond and currency options*, Rev. Financ. Stud., (1993) vol. 6 pp. 327–343.

[13] B. Horvath, A. Jacquier, and A. Muguruza, *Functional central limit theorems for rough volatility*, (2017), arXivpreprintarXiv:1711.03078.

[14] X. Huang, Q. Chen, and Y. Zhang, *Physics-informed neural networks for solving the Black-Scholes equation*, Appl. Math. Comput., (2021) vol. 408 126352.

[15] T. Jaisson, and M. Rosenbaum, *Rough fractional diffusions as scaling limits of nearly unstable heavy-tailed Hawkes processes*, Ann. Appl. Probab., (2016) vol. 26 pp. 2860–2882.

[16] D. P. Kingma, and J. Ba, *Adam: A method for stochastic optimization*, (2014), arXivpreprintarXiv:1412.6980.

[17] F. Le Goff, and A. L. de Gaulle, *Pricing with the SABR model*, J. Deriv., (2012) vol. 19 pp. 45–57.

[18] M. Raeisi-Makiani, A. Neisy, and A. Safdari-Vaighani, *The Application Of The Inverse Physics-Informed Neural Network In Financial Calibration Tasks*, Journal of Mathematics and Modeling in Finance (JMMF), (2026) vol. 6 no. 2, pp. 125–137.

[19] D. C. Liu, and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Program., (1989) vol. 45 pp. 503–528.

[20] Y. Liu, J. Li, and T. Zhang, *Physics-informed neural networks for fractional PDEs*, J. Comput. Phys., (2022) vol. 450 110847.

[21] B. B. Mandelbrot, and J. W. Van Ness, *Fractional Brownian motions, fractional noises and applications*, SIAM Rev., (1968) vol. 10 pp. 422–437.

[22] R. McCrickerd, and M. S. Pakkanen, *Turbocharging Monte Carlo pricing for the rough Bergomi model*, Quant. Finance, (2018) vol. 18 pp. 1877–1888.

[23] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, J. Comput. Phys., (2019) vol. 378 pp. 686–707.

[24] J. Sirignano, and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, J. Comput. Phys., (2018) vol. 375 pp. 1339–1364.

[25] C. Wu, M. Zhu, and Q. Tan, *Adaptive sampling for physics-informed neural networks*, J. Comput. Phys., (2023) vol. 476 111871.

[26] L. Yang, X. Meng, and G. E. Karniadakis, *B-PINNs: Bayesian physics-informed neural networks for forward and inverse problems*, J. Comput. Phys., (2021) vol. 425 109913.

[27] T. Zhang, Y. Liu, and J. Li, *An Efficient Physics-Informed Neural Network Solution to the Time-Space Fractional Black-Scholes Equation*, Oper. Res. Forum, (2024) vol. 5 120.

[28] S. Zhang, T. Wu, H. Xiao, Y. Gong, and W. Xu, *Efficient Calibration for Option Pricing via a Physics-Informed Chebyshev Kolmogorov–Arnold Network*, Mathematics, (2026) vol. 14 1529.

*How to Cite:* Amirali Ghajari[1], *Physics-Informed Neural Networks for Calibration of Rough Fractional Stochastic Volatility Models*, Journal of Mathematics and Modeling in Finance (JMMF), Vol. 7, No. 1, Pages:261–291, (2027).

The Journal of Mathematics and Modeling in Finance (JMMF) is licensed under a Creative Commons Attribution NonCommercial 4.0 International License.