# Fast Bayesian Estimation of Rough Stochastic Volatility Models

João Pedro Coli de Souza Monteneri Nacinben and Márcio Poletti Laurini
FEARP - University of São Paulo

March 2026

**Abstract**

We develop efficient computational methods for the Bayesian estimation of rough stochastic volatility (RSV) models. Our framework is based on a fractional Gaussian noise representation of rough processes and employs Integrated Nested Laplace Approximation (INLA) to perform scalable Bayesian inference. To address the computational challenges associated with large covariance matrices in rough volatility models, we investigate several covariance factorization strategies for likelihood evaluation, including Levinson–Durbin recursions (CPU and GPU implementations), Cholesky factorization using MAGMA and CHOLMOD libraries, and FFT-based circulant embedding. Monte Carlo experiments reveal a clear trade-off between numerical accuracy and computational efficiency. While Cholesky-based methods deliver slightly higher estimation accuracy, the FFT embedding approach provides substantial computational speedups with minimal loss of precision, and GPU acceleration significantly reduces runtime for recursion-based algorithms. We apply the proposed framework to the S&P 500 returns, the log realized variance of S&P 500 and VIX volatility series. The empirical results confirm the rough nature of equity volatility. In particular, the analysis based on log realized variance of the S&P 500 yields an estimated Hurst exponent of approximately $H \approx 0.31$. In contrast, estimates obtained from the S&P 500 returns indicate a much rougher volatility process, with $H \approx 0.13$, while the VIX index exhibits substantially smoother dynamics with $H \approx 0.50$. These findings demonstrate that modern numerical linear algebra techniques and GPU-accelerated algorithms make the Bayesian estimation of high-dimensional rough stochastic volatility models computationally feasible, enabling large-scale empirical applications and rigorous model comparison.

## 1 Introduction

Stochastic volatility (SV) models play a central role in modern financial econometrics and quantitative finance. Since financial returns exhibit strong time-varying volatility, classical constant-variance models (Engle, 1982; Bollerslev, 1986) fail to capture key empirical properties such as volatility clustering, heavy tails, and persistence in second moments. Stochastic volatility models address this limitation by explicitly modeling volatility as an unobserved stochastic process that evolves over time (Taylor, 1986). This framework has become fundamental for a wide range of applications, including asset pricing, derivative valuation, portfolio allocation, and risk management.

The importance of stochastic volatility models, however, extends far beyond time-series modeling of returns. In asset pricing theory, the dynamics of volatility play a crucial role in determining the risk premium associated with financial assets (French et al., 1987). In derivative markets, stochastic volatility models form the backbone of modern option pricing frameworks (Hull and White, 1987). Models such as the Heston stochastic volatility model (Heston, 1993) have been extensively used to explain implied volatility surfaces observed in option markets. Moreover, accurate modeling of volatility dynamics is essential for risk management tasks, including Value-at-Risk estimation, stress testing, and portfolio risk assessment.

Formally, stochastic volatility models assume that asset returns follow a conditionally Gaussian process whose variance is governed by a latent stochastic process, treating conditional variance as an unobserved latent component driven by an autoregressive process. Starting from a sequence $\{r_t\}_{t=1}^{N}$ of returns, the log-gaussian SV model with autoregressive dynamics can be represented as follows:

$$r_t = \exp\{h_t/2\}\varepsilon_t, \quad \varepsilon_t \sim N(0,1) \tag{1}$$

$$h_t = \mu + \phi(h_{t-1} - \mu) + z_t, \quad z_t \sim N(0, [(1-\phi^2)\tau_h]^{-1}) \tag{2}$$

where $\mu$ is the long term mean, $\phi$ is the autoregressive persistence and $\tau_h$ the marginal precision of the log-variance of the $h_t$. While standard SV models capture volatility clustering and the leptokurtic distribution of returns, the first-order autoregressive specification for $h_t$ implies an exponential decay in the autocorrelation function of log-volatility.

This exponential decay has then been called into question, with evidence suggesting that volatility shocks may have prolonged effects, consistent with a slower decay (Ding et al., 1993; Crato and de Lima, 1994). To reconcile model dynamics with empirical observations for returns data, the literature moved towards processes exhibiting long memory, replacing the standard Brownian motion in the volatility diffusion with a Fractional Brownian Motion (fBm), where the Hurst exponent ($H > 0.5$) governs the memory properties of the process (Comte and Renault, 1998). Similarly, discrete-time stochastic volatility models were proposed for situations where the latent volatility exhibits long-range dependence, usually modeling the dynamics of the latent log-variance as a fractionally integrated autoregressive moving average (ARFIMA) process, where the integrating order is responsible for capturing long memory patterns (Harvey, 1998; Breidt et al., 1998; Chaim and Laurini, 2024; Laurini et al., 2026).

Despite their success, classical and long-memory stochastic volatility models present important limitations when confronted with high-frequency financial data and modern empirical evidence. A growing body of research has documented that volatility paths observed in financial markets are significantly more irregular than those generated by standard diffusion models driven by Brownian motion (Gatheral et al., 2018). Traditional stochastic volatility models assume that volatility evolves according to smooth diffusion processes, which implies a level of path regularity inconsistent with empirical observations (Livieri et al., 2018; Bennedsen et al., 2021).

One manifestation of this limitation appears in the behavior of volatility increments across different time scales. Empirical studies show that log-volatility exhibits scaling properties characterized by extremely irregular trajectories. This observation has led to the use of the concept of *rough processes*, stochastic processes whose sample paths are significantly less regular than those generated by standard Brownian motion (Friz and Victoir, 2010; Friz and Hairer, 2014). In particular, empirical analyses indicate that log-volatility behaves similarly to a fractional Brownian motion with a Hurst exponent $H$ well below 0.5, often around $H \approx 0.1$ Gatheral et al. (2018). Such values imply a very low Hölder regularity of volatility paths, substantially rougher than that implied by classical Brownian-driven stochastic volatility models. This discrepancy suggests that the traditional assumption of Brownian noise in volatility dynamics may be insufficient to capture the true temporal structure of financial market volatility, motivating the development of rough stochastic volatility models.

Recent research has proposed that these phenomena may arise from the interaction between market microstructure mechanisms and the aggregation of trading activity across heterogeneous market participants (El Euch et al., 2018). At very high frequencies, financial markets are characterized by order flow dynamics, liquidity provision, and the activity of algorithmic and high-frequency traders (El Euch et al., 2018). These microstructure effects generate complex feedback mechanisms in price formation and volatility dynamics. When aggregated over time, these interactions can produce volatility trajectories that exhibit rough statistical properties.

In particular, models based on fractional stochastic processes have been shown to reproduce the empirical roughness observed in volatility time series. Fractional Gaussian processes with small Hurst exponents generate highly irregular paths and scaling behavior consistent with empirical volatility data. These processes naturally arise as limits of aggregated microstructure models,

where the cumulative effect of numerous small trading interactions produces long-range dependence and rough volatility dynamics (El Euch et al., 2018; Jaisson and Rosenbaum, 2016a).

This rough volatility paradigm has led to a new class of stochastic volatility models that depart from the classical Brownian framework. Instead of assuming smooth diffusion dynamics, rough volatility models represent the log-volatility process using fractional Gaussian processes with Hurst parameters significantly smaller than one-half (Bergomi, 2015; Gatheral et al., 2018; Eraker and Vilkov, 2025). Such models generate extremely irregular trajectories that closely match the statistical properties observed in empirical volatility data.

However, the interpretation of these empirical findings on rough volatility has been questioned in subsequent work. Rogers (2023) pointed out that similar scaling behavior can arise even when the underlying volatility follows a standard diffusion process. Through simulation experiments based on stochastic volatility models with Brownian drivers, Rogers showed that the log–log regression methodology commonly used to estimate roughness can generate apparent Hurst exponents well below 1/2, even when the true volatility process is not rough. This observation raises concerns about the statistical reliability of roughness estimators based on realized volatility and suggests that the empirical evidence for rough volatility may not uniquely identify fractional models.

The analysis of Cont and Das (2024) strengthens and clarifies this critique by explicitly examining the role of measurement error in volatility proxies. Since spot volatility is not directly observable, empirical studies necessarily rely on estimators such as realized volatility constructed from discretely sampled price data. Cont and Das (2024) demonstrate that the estimation error associated with realized volatility has non-trivial temporal dependence and significantly alters the short-scale path properties of the resulting volatility proxy. In particular, their simulations show that even when the instantaneous volatility follows a Brownian diffusion with roughness index $H = 1/2$, the realized volatility process systematically exhibits much lower estimated roughness, often in the range 0.05–0.3, which coincides with the empirical estimates reported in the rough volatility literature.

These results highlight an important identification problem in the empirical analysis of rough volatility models. Since commonly used volatility proxies may exhibit apparent rough behavior even when the underlying spot volatility follows a standard diffusion, inference procedures based solely on realized volatility may lead to misleading conclusions about the structural properties of volatility dynamics. This observation motivates the development of more robust inference methods capable of disentangling the intrinsic roughness of the latent volatility process from the statistical artifacts induced by discretization and measurement error. In particular, reliable inference for rough stochastic volatility requires statistical procedures that explicitly account for the estimation error in volatility proxies and exploit identification strategies that are robust to sampling noise. Developing such methods is therefore essential for assessing whether the roughness observed in empirical volatility measures reflects a genuine structural property of financial markets or merely a by-product of the estimation procedure.

Motivated by these identification challenges, we adopt a structural modeling approach in which the latent volatility process is explicitly specified and estimated jointly with the observed price dynamics. In this paper we develop and estimate a rough stochastic volatility model in which the latent log-volatility dynamics are driven by a fractional Gaussian noise (fGn) process combined with a persistent autoregressive component. This specification allows the model to simultaneously capture the strong local irregularity of volatility paths implied by the rough volatility literature and the medium-term persistence commonly observed in financial volatility series. The fractional Gaussian noise component introduces a flexible covariance structure governed by the Hurst exponent, which controls the degree of roughness in volatility dynamics. When the Hurst parameter takes values significantly smaller than one half, the model generates extremely irregular volatility trajectories that match empirical evidence obtained from high-frequency financial data. The additional autoregressive component captures persistent volatility clustering and provides a complementary mechanism to represent medium-term dynamics in the latent volatility process. Together, these components define a flexible rough stochastic volatility framework capable of reproducing both the roughness and persistence observed in empirical volatility time series, while

providing a coherent setting for likelihood-based and Bayesian inference that directly addresses the measurement and identification issues discussed above.

A central contribution of this paper is the systematic comparison of several computational implementations for estimating the rough stochastic volatility model. The main computational challenge arises from the evaluation of the covariance/precision structure associated with the fractional Gaussian noise process. Since the fGn process is non-Markovian and characterized by a dense covariance matrix, direct evaluation of the likelihood becomes computationally demanding for large time series. To address this issue, we consider multiple numerical strategies for representing and factorizing the covariance or precision matrix of the fGn process. In particular, we compare three approaches: the Cholesky factorization of the covariance matrix (Asmussen and Glynn, 2007), the Levinson–Durbin recursion exploiting the Toeplitz structure (Levinson, 1946; Durbin, 1960a), and the circular embedding method combined with Fast Fourier Transform (FFT) techniques (Wood and Chan, 1994; Dietrich and Newsam, 1997). Each of these approaches provides a different trade-off between numerical accuracy and computational efficiency. The Cholesky decomposition offers numerical stability but has cubic computational complexity, while the Levinson–Durbin recursion reduces the computational cost by exploiting the Toeplitz structure of the covariance matrix. The circular embedding approach further reduces computational complexity by transforming the covariance matrix into a circulant structure that can be diagonalized efficiently using FFT algorithms.

In addition to comparing numerical algorithms for covariance evaluation, we also investigate the impact of different hardware architectures on computational performance. Specifically, we implement the estimation procedures using both traditional CPU-based linear algebra routines and fully parallel GPU implementations designed to exploit massive parallelism in modern graphics processors. The GPU implementations rely on parallel linear algebra libraries and optimized FFT routines that allow the simultaneous execution of thousands of operations. In particular, the computations are implemented using NVIDIA's CUDA framework together with the MAGMA library, which provides GPU-accelerated implementations of dense linear algebra routines compatible with standard LAPACK interfaces. CUDA enables massive parallelism by distributing computations across a large number of GPU cores, while MAGMA efficiently offloads computationally intensive matrix operations, such as factorizations and matrix multiplications, to the GPU. These libraries are specifically designed to exploit the hierarchical memory structure and high memory bandwidth of modern GPUs, leading to substantial reductions in computational time for large-scale matrix operations. As a result, GPU-based implementations can significantly outperform traditional CPU-based approaches, particularly in simulation and estimation procedures that require repeated linear algebra operations or large-scale covariance matrix manipulations.

By comparing CPU and GPU implementations across different covariance evaluation methods, we provide a detailed analysis of the computational efficiency of rough stochastic volatility models. The comparison is conducted in terms of several performance metrics, including computational time, estimation bias, root mean squared error (RMSE), and mean absolute error (MAE). This evaluation allows us to assess not only the computational speed of each approach but also the statistical accuracy of the resulting parameter estimates.

Another important contribution of this paper concerns the inference framework used to estimate the proposed model. Instead of relying on traditional Markov Chain Monte Carlo (MCMC) methods, we perform Bayesian inference using the Integrated Nested Laplace Approximation (INLA) framework (Rue and Martino, 2007; Rue et al., 2009). INLA provides a deterministic alternative to MCMC for latent Gaussian models, offering significant computational advantages in terms of speed and scalability. Although MCMC methods remain widely used in stochastic volatility estimation, they can become computationally expensive when applied to models with high-dimensional latent structures or complex covariance matrices such as those arising from fractional Gaussian processes. INLA circumvents these limitations by using accurate Laplace approximations to evaluate posterior distributions, allowing efficient inference even in large-scale time series models.

To illustrate the empirical relevance and computational feasibility of the proposed framework, we analyze three complementary measures of market volatility: S&P 500 returns, the log realized

variance of the S&P 500, and the VIX volatility index. These series capture different aspects of volatility dynamics. S&P 500 returns reflect realized market fluctuations, log realized variance provides a high-frequency-based proxy for integrated volatility, and the VIX represents forward-looking expectations of volatility embedded in option prices.

Applying our Bayesian estimation framework for rough stochastic volatility models, we estimate the Hurst exponent and associated parameters for each dataset. The results provide further empirical evidence supporting the rough volatility paradigm. Volatility inferred from S&P 500 returns exhibits pronounced roughness, with an estimated Hurst exponent of approximately $H \approx 0.13$. The analysis based on log realized variance yields an intermediate value of about $H \approx 0.31$, reflecting the smoothing inherent in daily volatility aggregates while remaining substantially below the Brownian benchmark $H = 0.5$. In contrast, the VIX displays markedly smoother dynamics, with $H \approx 0.50$, consistent with the expectation and smoothing effects embedded in implied volatility measures.

Overall, this work makes three main contributions. First, we develop a Bayesian estimation framework for rough stochastic volatility models based on a fractional Gaussian noise representation. Second, we design and implement efficient computational strategies for large-scale inference, combining Integrated Nested Laplace Approximation with advanced covariance factorization techniques and GPU acceleration. Third, we provide empirical evidence from the S&P 500 and VIX indices that illustrates both the rough nature of equity volatility and the practical applicability of the proposed methods for high-dimensional financial time series.

# 2 Rough Processes, Fractional Gaussian Noise and Rough Stochastic Volatility

## 2.1 Fractional Brownian Motion and Fractional Gaussian Noise

A *fractional Brownian motion* (fBm) $B_H(t)$, $t \geq 0$ (Mandelbrot and Van Ness, 1968), is a centered Gaussian process defined by the covariance function

$$\mathbb{E}[B_H(t)] = 0, \tag{3}$$

$$\mathrm{Cov}\big(B_H(t), B_H(s)\big) = \frac{1}{2}\left(t^{2H} + s^{2H} - |t-s|^{2H}\right), \tag{4}$$

where $H$ is the Hurst parameter. The parameter $H$ controls the dependence structure and smoothness of the process. For the process reduces to classical Brownian motion, for $H > 1/2$ the process exhibits long-range dependence and persistent increments, and for $H < 1/2$ the process exhibits anti-persistence and rough trajectories. Fractional Brownian motion is self-similar with parameter H, $B_H(ct) \overset{d}{=} c^H B_H(t)$, for any $c > 0$, where $\overset{d}{=}$ denotes equality in distribution.

The *fractional Gaussian noise* (fGn) process (Palma, 2007) is defined as the increment process of fractional Brownian motion:

$$X_t = B_H(t+1) - B_H(t), \qquad t \in \mathbb{Z}. \tag{5}$$

This defines a stationary Gaussian process with mean zero and autocovariance function

$$\gamma(k) = \mathrm{Cov}(X_t, X_{t+k}) = \frac{1}{2}\left(|k+1|^{2H} - 2|k|^{2H} + |k-1|^{2H}\right). \tag{6}$$

A fundamental property is that the covariance matrix for a sample of length $n$ has Toeplitz structure

$$\Gamma_{ij} = \gamma(|i-j|). \tag{7}$$

This defines the class of representations and numerical approximations that can be explored to evaluate the covariance generated by this process. For $H > 1/2$, fractional Gaussian noise exhibits long-range dependence. In particular, the autocovariance decays according to

$$\gamma(k) \sim H(2H-1)k^{2H-2}, \qquad k \to \infty. \tag{8}$$

Because the decay is polynomial rather than exponential, the autocorrelation function satisfies

$$\sum_{k=0}^{\infty} \gamma(k) = \infty, \tag{9}$$

which characterizes long-memory processes. The spectral density of fractional Gaussian noise behaves asymptotically as

$$f(\lambda) \sim C_H |\lambda|^{1-2H}, \qquad \lambda \to 0, \tag{10}$$

where $C_H$ is a constant depending on $H$. This divergence near zero frequency reflects the strong persistence of the process when $H > 1/2$.

Fractional Brownian motion and Fractional Gaussian noise provides natural examples of a *rough stochastic process*. Rough stochastic processes have emerged as an important class of models for describing highly irregular temporal dynamics observed in many natural and financial systems. Unlike classical diffusion processes, rough processes exhibit trajectories that are significantly less regular than Brownian motion. This lack of smoothness is typically characterized using Hölder regularity and is closely related to fractional stochastic dynamics.

## 2.2 Definition of Rough Processes

Let $X = \{X_t\}_{t \geq 0}$ be a stochastic process defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. The process is said to be *rough* if its sample paths have Hölder regularity strictly less than $1/2$ (Friz and Victoir, 2010). More formally, a function $f(t)$ is Hölder continuous with exponent $\alpha \in (0, 1)$ if there exists a constant $C > 0$ such that

$$|f(t) - f(s)| \leq C|t - s|^{\alpha}, \quad \forall s, t. \tag{11}$$

A stochastic process is considered rough when its trajectories belong almost surely to the Hölder space

$$X_t \in C^{\alpha} \quad \text{with } \alpha < \frac{1}{2}. \tag{12}$$

This implies that the increments of the process fluctuate more rapidly than those of standard Brownian motion.The Hölder regularity of fBm trajectories is directly controlled by the Hurst parameter:

$$B_H(t) \in C^{H-\varepsilon} \tag{13}$$

for any $\varepsilon > 0$. Consequently, small values of $H$ produce extremely irregular paths, the roughness increases as $H \to 0$. In particular, when $H < 1/2$, the process belongs to the class of *rough processes*, characterized by trajectories rougher than classical Brownian motion (Friz and Victoir, 2010). These processes play a central role in modern stochastic modeling, especially in rough volatility models used in mathematical finance (Gatheral et al., 2018). The increment process (fractional Gaussian noise) inherits this rough behavior, making it suitable for modeling systems where high-frequency variability and strong temporal dependence coexist.

A canonical model incorporating rough dynamics is the class of *Rough Fractional Stochastic Volatility* (RFSV) models. In this general framework, the log-volatility process is modeled as

$$\log \sigma_t = X_t, \tag{14}$$

where $X_t$ follows a fractional Gaussian noise process with Hurst exponent $H < 0.5$. A typical specification is

$$X_t = \nu B_t^H, \tag{15}$$

where $B_t^H$ is fractional Brownian motion, and $\nu$ controls the volatility of volatility. The asset price then follows

$$dS_t = S_t \sigma_t dW_t. \tag{16}$$

One of the most widely used rough volatility models in quantitative finance is the Rough Bergomi model Bayer et al. (2016). It extends the classical Bergomi stochastic volatility (Bergomi, 2015) framework by introducing fractional dynamics in the variance process. The forward variance process is defined as

$$\xi_t(T) = \xi_0(T) \exp\left(\eta \int_0^t (T-s)^{H-\frac{1}{2}} dW_s \frac{\eta^2}{2} t^{2H}\right), \tag{17}$$

$\xi_0(T)$ is the initial forward variance curve, $\eta$ controls volatility of volatility, $H$ is the Hurst exponent, and $W_t$ is a Brownian motion. The instantaneous variance becomes

$$V_t = \xi_t(t), \tag{18}$$

and the asset dynamics are given by

$$dS_t = \sqrt{V_t} S_t dZ_t, \tag{19}$$

where $Z_t$ may be correlated with $W_t$. This model successfully reproduces key features of option markets.

# 3 A Rough Stochastic Volatility Model Based on Fractional Gaussian Noise

This section presents the rough stochastic volatility model proposed in this paper and summarizes the main methodological contributions of our approach. The objective of the framework is to provide a flexible and computationally efficient method for estimating rough volatility dynamics in financial time series while addressing the substantial computational challenges associated with fractional stochastic processes.

The model is formulated within the latent Gaussian modeling framework and estimated using the Integrated Nested Laplace Approximation (INLA), which provides a fast alternative to traditional Markov Chain Monte Carlo methods for Bayesian inference.The latent log-volatility is specified as the sum of two complementary components. The first is a fractional Gaussian noise process that captures the local roughness and scaling properties observed in empirical volatility dynamics. The second is a persistent autoregressive component that represents medium-term volatility clustering. This combined specification allows the model to simultaneously capture fine-scale irregularity and longer-term dependence patterns commonly observed in financial volatility series.

The formulation presented in this section therefore establishes the full hierarchical structure of the proposed rough stochastic volatility model and provides the basis for the computational analysis developed in the remainder of the paper.

## 3.1 Observation Equation

Let $y_t$ denote the observed financial returns at time $t$, for $t = 1, \ldots, n$. The model assumes a stochastic volatility specification where the returns follow a conditionally Gaussian distribution:

$$y_t \mid h_t \sim \mathcal{N}(0, \exp(h_t)). \tag{20}$$

Here $h_t$ represents the latent log-volatility process.The log-volatility process is decomposed into three components:

$$h_t = \mu + x_t + a_t, \tag{21}$$

where $\mu$ is a constant mean ,$x_t$ is a rough latent component driven by fractional Gaussian noise, and $a_t$ is a persistent autoregressive component capturing additional serial dependence. This decomposition allows the model to capture both rough local fluctuations and longer-range volatility persistence.

The process $x_t$ is modeled as a fractional Gaussian noise process with Hurst parameter $H$. The covariance matrix of the vector

$$\mathbf{x} = (x_1, \ldots, x_n)^T \tag{22}$$

is Toeplitz:

$$\Sigma_{ij} = \gamma(|i - j|). \tag{23}$$

and the corresponding precision matrix

$$Q = \Sigma^{-1} \tag{24}$$

needs to be computed numerically.

To capture persistent volatility clustering beyond the local rough behavior, the model includes an AR(1) component:

$$a_t = \rho a_{t-1} + \epsilon_t, \tag{25}$$

where $\epsilon_t \sim \mathcal{N}(0, \tau_a^{-1})$. The autoregressive coefficient satisfies $|\rho| < 1$, imposing stationarity in this component. In the implementation, the AR(1) parameter is internally parameterized using the Fisher transformation:

$$\theta = \frac{1}{2} \log\left(\frac{1+\rho}{1-\rho}\right), \tag{26}$$

which ensures that $\rho$ remains within the admissible interval $(-1, 1)$ during optimization.

The complete hierarchical structure of the model can be summarized as

$$\begin{aligned}
\text{Observation model:} \quad & y_t \mid h_t \sim \mathcal{N}\left(0, \exp(h_t)\right) \\
\text{Latent log-volatility:} \quad & h_t = \mu + x_t + a_t \\
\text{Rough component:} \quad & \mathbf{x} \sim fGn\left(H, \tau\right) \\
\text{AR(1) component:} \quad & a_t = \phi a_{t-1} + \epsilon_t, \\
& \epsilon_t \sim \mathcal{N}\left(0, \tau_a^{-1}\right)
\end{aligned} \tag{27}$$

The resulting specification defines a rough stochastic volatility model where the log-volatility evolves according to a combination of rough fractional dynamics and persistent autoregressive behavior. The fractional Gaussian noise component captures the fine-scale roughness of volatility trajectories, while the AR(1) component models medium-term persistence and clustering effects. Together, these components provide a flexible and computationally efficient framework for modeling financial volatility consistent with empirical evidence of rough volatility.

To ensure that the model parameters satisfy their natural domain restrictions, we employ internal reparameterizations within the estimation procedure. In particular, the precision parameter of the fractional Gaussian noise process is represented on the logarithmic scale. Let $\theta_\tau \in \mathbb{R}$ denote the internal parameter. The precision parameter is then defined as

$$\tau = \exp(\theta_\tau), \tag{28}$$

which guarantees the positivity constraint $\tau > 0$ while allowing the estimation algorithm to operate on an unconstrained parameter space. Similarly, the Hurst exponent governing the roughness of the fractional Gaussian noise process is parameterized using a logistic transformation. Let $\theta_H \in \mathbb{R}$ denote the internal parameter associated with the Hurst exponent. The transformation used in the model is

$$H = 0.01 + \frac{0.50}{1 + \exp(-\theta_H)}. \tag{29}$$

This transformation guarantees that the estimated Hurst exponent satisfies $0.001 < H < 0.5$, which effectively restricts the parameter to the rough volatility region $H < 0.5$ while avoiding numerical issues associated with boundary values near zero. The use of an unconstrained internal parameter $\theta_H$ simplifies the optimization and Bayesian inference steps, since it allows the estimation algorithm to operate on the real line without requiring explicit constraints.

Within the Bayesian framework implemented in `INLA`, priors are therefore specified on the transformed parameter $\theta_H$ rather than directly on $H$. The posterior distribution of the Hurst exponent is subsequently obtained by applying the above transformation to the marginal posterior of $\theta_H$. This strategy ensures numerical stability and facilitates efficient inference while enforcing the theoretical restriction implied by rough volatility models.

We consider two alternative specifications for the non-rough component of latent log-volatility process. In the first specification, the volatility dynamics combine fractional Gaussian noise with a Random Walk (RW1) component, allowing for highly persistent stochastic trends in volatility. This specification provides a flexible nonstationary representation that can accommodate slow-moving shifts in volatility levels often observed in financial time series. In the second specification, the rough component is combined with an Ornstein–Uhlenbeck (OU) process, which introduces mean-reverting dynamics in the latent volatility. The OU formulation corresponds to the continuous-time analogue of an AR(1) process and provides a parsimonious stationary benchmark widely used in stochastic volatility modeling.

## 3.2 Cholesky-Based Approach for fGn Covariance and Precision Matrices

As discussed, for fractional Gaussian noise (fGn) processes, the Bayesian estimation requires the precision matrix $Q = \tau\Gamma^{-1}$, where $\Gamma \in \mathbb{R}^{n\times n}$ is the covariance matrix with entries defined by the autocovariance function (ACVF), and $\tau$ is a scaling factor (precision). As discussed earlier, an essential characteristic is that the autocovariance matrix of fGN is Toeplitz, which allows for computationally efficient implementations. A Toeplitz matrix $\Gamma$ has constant diagonals, i.e., $\Gamma_{ij} = \gamma_{|i-j|}$. This property enables efficient storage, since only the first row or column of length $n$ is needed, and fast algorithms for linear systems and inversion.

## 3.3 Cholesky Decomposition of Toeplitz Matrices

The covariance matrix associated with the fractional Gaussian noise (fGn) process can be factorized using the Cholesky decomposition Trefethen and Bau (1997); Asmussen and Glynn (2007). Let $\Gamma$ denote the $n \times n$ Toeplitz covariance matrix constructed from the autocovariance function of the fGn process. The Cholesky decomposition expresses this matrix as

$$\Gamma = LL^\top, \tag{30}$$

where $L$ is a lower triangular matrix with strictly positive diagonal elements. The positivity of the diagonal entries guarantees that the covariance matrix is positive definite. This factorization is particularly useful in statistical computations because several quantities required for likelihood evaluation can be obtained directly from the triangular factor. In particular, the precision matrix can be expressed as

$$\Gamma^{-1} = L^{-\top}L^{-1}, \tag{31}$$

while the logarithm of the determinant of the covariance matrix can be computed efficiently as

$$\log\det\Gamma = 2\sum_{i=1}^{n}\log L_{ii}. \tag{32}$$

The numerical implementation of this approach begins with the evaluation of the autocovariance function of the fGn process. The autocovariances $\gamma_k$ are computed for lags $k = 0, \ldots, n-1$. In practice, a small numerical perturbation $\varepsilon > 0$ (jittering) may be added to the variance term to improve numerical stability,

$$\gamma_0 \leftarrow \gamma_0 + \varepsilon. \tag{33}$$

Once the autocovariances are obtained, the Toeplitz covariance matrix $\Gamma$ is constructed by placing the value $\gamma_k$ on the $k$-th diagonal. The Cholesky factor $L$ is then computed by solving the matrix equation $LL^\top = \Gamma$. Although the Cholesky decomposition of a general dense matrix requires $O(n^3)$ operations, the Toeplitz structure of the covariance matrix can be exploited to reduce the computational cost to $O(n^2)$ using specialized algorithms.

After computing the Cholesky factor, the precision matrix can be obtained by solving triangular systems using forward and backward substitution. The triangular structure of $L$ allows these operations to be carried out efficiently. In addition, the determinant of the covariance matrix can be evaluated directly from the diagonal elements of the Cholesky factor, avoiding the need for costly determinant computations on dense matrices.

From a numerical linear algebra perspective, the Cholesky decomposition is well known for its stability when applied to symmetric positive-definite matrices such as covariance matrices. The method provides an exact factorization and introduces no approximation error in the representation of the covariance structure (Golub and Van Loan, 2013; Higham, 2002). However, the approach requires storing the full covariance matrix, which implies a memory cost of $O(n^2)$ (Rasmussen and Williams, 2006). For moderate problem sizes this requirement is manageable, but it may become restrictive when very long time series are considered.

In practical implementations, computational efficiency can be further improved by exploiting parallel hardware architectures. Modern numerical libraries allow the Cholesky decomposition to be executed on graphics processing units (GPUs) using highly optimized dense linear algebra routines. Libraries such as MAGMA or cuBLAS provide GPU implementations that can significantly accelerate the factorization of large matrices. Additional speed improvements can be obtained by parallelizing intermediate steps such as the evaluation of the autocovariance function, the construction of the Toeplitz matrix, and the solution of triangular systems.

Overall, the Cholesky-based approach provides a robust and theoretically sound method for computing covariance and precision matrices associated with fractional Gaussian noise processes. Its main advantages are numerical stability, exact factorization, and efficient evaluation of the log-determinant, which is particularly useful in likelihood-based estimation. Nevertheless, the quadratic memory requirement and computational cost may limit its applicability for extremely large matrices, where alternative approaches such as Levinson–Durbin recursion or FFT-based circulant embedding methods may provide more scalable solutions.

# 4 Levinson–Durbin Recursion for fGn Precision Matrices

## 4.1 Levinson–Durbin recursions

An alternative approach for computing the inverse and determinant of Toeplitz covariance matrices associated with fractional Gaussian noise (fGn) processes is based on the Levinson–Durbin recursion (Durbin, 1960b; Brockwell and Davis, 1991; Golub and Van Loan, 2013). This algorithm provides an efficient method for solving Toeplitz linear systems and constructing the inverse covariance matrix by exploiting the constant-diagonal structure of the Toeplitz matrix. In contrast to general matrix inversion methods, which require cubic computational complexity, the Levinson–Durbin recursion reduces the computational cost to quadratic order.

Let $\Gamma$ denote the Toeplitz covariance matrix generated by the autocovariance function $\gamma_k$ of the fGn process. The Levinson–Durbin algorithm constructs a sequence of linear prediction coefficients that describe the best linear predictor of the process based on its past values. Let $\phi^{(k)} = (\phi_1^{(k)}, \ldots, \phi_k^{(k)})$ denote the prediction coefficients at stage $k$, and let $v_k$ represent the corresponding prediction error variance. The recursion is initialized with

$$\phi^{(0)} = [\,], \tag{34}$$

$$v_0 = \gamma_0. \tag{35}$$

At each stage $k = 1, \ldots, n-1$, the algorithm computes a reflection coefficient that measures the contribution of the new lag to the prediction equation. The recursion proceeds as

$$\lambda_k = -\frac{\gamma_k + \sum_{j=1}^{k-1} \phi_j^{(k-1)} \gamma_{k-j}}{v_{k-1}}, \tag{36}$$

$$\phi_j^{(k)} = \phi_j^{(k-1)} + \lambda_k \phi_{k-j}^{(k-1)}, \qquad j = 1, \ldots, k-1, \tag{37}$$

$$\phi_k^{(k)} = \lambda_k, \tag{38}$$

$$v_k = v_{k-1}(1 - \lambda_k^2). \tag{39}$$

The coefficient $\lambda_k$ is known as the reflection coefficient at step $k$, while $v_k$ corresponds to the variance of the one-step-ahead prediction error. These quantities play a central role in the recursive construction of the inverse covariance matrix.

Once the recursion reaches the final stage, the prediction coefficients and the final prediction error variance can be used to construct the inverse Toeplitz covariance matrix. Specifically, the precision matrix can be written as

$$\Gamma^{-1} = \frac{1}{v_{n-1}} \Phi \Phi^\top, \tag{40}$$

where $\Phi$ denotes a lower-triangular Toeplitz matrix generated from the prediction coefficients $\phi_j^{(n-1)}$. This representation provides an exact expression for the inverse covariance matrix without requiring a full matrix factorization.

From the perspective of numerical linear algebra, the Levinson–Durbin recursion offers several computational advantages. By exploiting the Toeplitz structure of the covariance matrix, the algorithm reduces the computational complexity from $O(n^3)$, required for general matrix inversion, to $O(n^2)$ (Golub and Van Loan, 2013). Moreover, the method is memory-efficient because it requires storing only the autocovariance vector $\gamma = (\gamma_0, \ldots, \gamma_{n-1})$ and a small number of auxiliary coefficient vectors of length $n$. Another important property is that the recursion preserves the positive definiteness of the covariance matrix as long as the reflection coefficients satisfy $|\lambda_k| < 1$.

An additional advantage of this approach is that the determinant of the covariance matrix can be computed directly from the sequence of prediction error variances. In particular, the log-determinant can be written as

$$\log\det\Gamma = \sum_{k=0}^{n-1} \log v_k. \tag{41}$$

This result is especially useful in likelihood-based inference for Gaussian models, where repeated evaluations of the log-determinant are required.

Despite these advantages, the Levinson–Durbin algorithm also presents certain limitations. The recursion is inherently sequential, meaning that each step depends on the results of the previous iteration. As a consequence, the algorithm is difficult to parallelize efficiently across multiple processors or GPU architectures. Furthermore, when the covariance matrix becomes highly ill-conditioned, numerical stability may deteriorate and small regularization terms may be required to ensure stable computations. Finally, although the method avoids forming the full covariance matrix, the explicit construction of the precision matrix may still require quadratic storage when all entries are needed.

Overall, the Levinson–Durbin recursion provides a fast and memory-efficient method for computing the inverse and determinant of Toeplitz covariance matrices associated with fractional Gaussian noise processes. By fully exploiting the Toeplitz structure of the covariance matrix, the method achieves substantial computational savings while maintaining exactness. This makes it particularly attractive for large time series where direct Cholesky factorization becomes computationally expensive.

## 4.2 Circular Embedding and FFT Method for fGn Precision Matrices

An alternative and computationally efficient strategy for handling Toeplitz covariance matrices associated with fractional Gaussian noise (fGn) processes is based on circular embedding combined with Fast Fourier Transform (FFT) techniques (Wood and Chan, 1994; Dietrich and Newsam, 1997). The central idea is to embed the Toeplitz covariance matrix into a larger circulant matrix (Davis, 1979), whose algebraic structure allows efficient diagonalization in the frequency domain. This approach transforms the inversion and determinant computation of the covariance matrix into simple operations on its eigenvalues.

Let $\Gamma$ denote the $n \times n$ Toeplitz covariance matrix generated by the autocovariance function $\gamma_k$ of the fGn process. The matrix can be embedded into a larger circulant matrix $C \in \mathbb{R}^{m\times m}$ with $m \geq 2n-2$. The first row of the circulant matrix is constructed from the autocovariance sequence and its reversed values,

$$C = \begin{pmatrix} \gamma_0 & \gamma_1 & \cdots & \gamma_{n-1} & \gamma_{n-2} & \cdots & \gamma_1 \\ \gamma_1 & \gamma_0 & \cdots & \gamma_{n-2} & \gamma_{n-3} & \cdots & \gamma_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \gamma_1 & \gamma_2 & \cdots & \gamma_{n-1} & \cdots & \gamma_2 & \gamma_0 \end{pmatrix}. \tag{42}$$

Circulant matrices possess a key property that makes them particularly attractive for numerical computations: they are diagonalized by the discrete Fourier transform (DFT) (Davis, 1979). More precisely, the circulant matrix admits the spectral decomposition

$$C = F^* \Lambda F, \tag{43}$$

where $F$ denotes the unitary discrete Fourier transform matrix, $F^*$ its conjugate transpose, and $\Lambda = \mathrm{diag}(\lambda_0, \ldots, \lambda_{m-1})$ is a diagonal matrix containing the eigenvalues of $C$. These eigenvalues correspond to the Fourier transform of the first column of the circulant matrix.

This representation allows matrix inversion to be carried out efficiently in the frequency domain. Since the matrix is diagonalized by the Fourier transform, its inverse can be expressed as

$$C^{-1} = F^* \Lambda^{-1} F, \tag{44}$$

where $\Lambda^{-1} = \mathrm{diag}(1/\lambda_0, \ldots, 1/\lambda_{m-1})$. In practice, the eigenvalues $\lambda_k$ are obtained by applying the Fast Fourier Transform to the circulant embedding vector constructed from the autocovariance function. The numerical implementation proceeds by first constructing the circulant embedding vector

$$c = (\gamma_0, \gamma_1, \ldots, \gamma_{n-1}, \gamma_{n-2}, \ldots, \gamma_1),$$

which has length $m = 2n-2$. The discrete Fourier transform of this vector is then computed using the FFT algorithm,

$$\hat{c} = \mathrm{FFT}(c).$$

The resulting vector $\hat{c}$ contains the eigenvalues of the circulant matrix. In numerical applications it is often necessary to regularize very small eigenvalues to ensure numerical stability. This can be achieved by adding a small positive constant $\epsilon$,

$$\hat{c}_k \leftarrow \hat{c}_k + \epsilon,$$

where $\epsilon > 0$ is chosen sufficiently small so as not to distort the covariance structure.

After regularization, the eigenvalues are inverted in the frequency domain,

$$\hat{c}_k^{-1} = \frac{1}{\hat{c}_k},$$

and the inverse Fourier transform is applied to obtain the first column of the inverse circulant matrix,

$$c^{-1} = \mathrm{IFFT}(\hat{c}^{-1}).$$

The inverse covariance matrix $\Gamma^{-1}$ is then approximated by extracting the $n \times n$ top-left block of the inverse circulant matrix.

The circular embedding approach also provides an efficient method for computing the determinant of the covariance matrix. Because the eigenvalues of the circulant matrix are obtained directly from the Fourier transform, the log-determinant can be approximated as

$$\log\det\Gamma \approx \sum_{k=0}^{m-1} \log\lambda_k, \tag{45}$$

where $\lambda_k$ denote the eigenvalues obtained from the FFT of the embedding vector. When the embedding dimension is sufficiently large, this approximation becomes highly accurate.

From a computational perspective, the main advantage of the circular embedding approach lies in its efficiency. The use of FFT algorithms reduces the computational complexity to $O(m \log m)$, which is approximately $O(n \log n)$ for large matrices. This represents a substantial improvement compared to the cubic complexity of standard matrix factorizations and the quadratic complexity of Toeplitz-based algorithms such as the Levinson–Durbin recursion. In addition, the method is memory efficient because it requires storing only the autocovariance vector and its circulant extension rather than the full covariance matrix.

Another important advantage is the high degree of parallelism offered by FFT computations. Modern FFT implementations are optimized for both multi-core CPUs and graphics processing units (GPUs), allowing large-scale covariance operations to be executed extremely efficiently on parallel hardware architectures. The possibility of regularizing small eigenvalues further enhances the numerical stability of the procedure.

In summary, the circular embedding and FFT method transforms the inversion of a Toeplitz covariance matrix into a simple diagonal inversion in the frequency domain. The procedure can be summarized as

$$\Gamma \approx C \xrightarrow{\text{FFT}} \Lambda \xrightarrow{\text{invert}} \Lambda^{-1} \xrightarrow{\text{IFFT}} \Gamma^{-1}, \qquad \log\det\Gamma \approx \sum_{k=0}^{m-1} \log\lambda_k.$$

This approach is particularly advantageous for large-scale problems involving long time series, where traditional Cholesky or Levinson–Durbin methods may become computationally expensive.

## 4.3 Integrated Nested Laplace Approximation

Integrated Nested Laplace Approximation (INLA) provides a deterministic framework for performing approximate Bayesian inference in a broad class of latent Gaussian models. The method was introduced by Rue et al. (2009) and has since become widely used in applied statistics, econometrics, and spatial modeling. In latent Gaussian models, the observed data are linked to an underlying latent Gaussian field through a likelihood function, while a set of hyperparameters controls the covariance structure of the latent process. This formulation naturally encompasses many econometric models, including stochastic volatility specifications.

The key idea of INLA is to replace computationally intensive simulation-based inference with a sequence of accurate analytical approximations. In particular, the posterior distribution of the hyperparameters is approximated using Laplace approximations, and conditional posterior distributions of the latent variables are then obtained through additional nested approximations.

By exploiting the structure of Gaussian Markov random fields (GMRFs), INLA is able to perform these calculations efficiently using sparse matrix methods from numerical linear algebra.

One of the main advantages of INLA is its computational efficiency compared to traditional Markov Chain Monte Carlo (MCMC) methods. While MCMC algorithms rely on long stochastic simulations to explore the posterior distribution, INLA computes deterministic approximations of marginal posterior distributions directly. As a result, INLA typically provides substantial reductions in computation time while maintaining a high degree of accuracy for marginal posterior summaries. Several empirical studies have shown that the approximations produced by INLA are often comparable in accuracy to those obtained from carefully tuned MCMC algorithms.

Another important feature of INLA is its suitability for parallel computation. The evaluation of the posterior distribution of the hyperparameters involves multiple independent numerical integrations and likelihood evaluations, which can be distributed across multiple processing cores. This parallel structure allows INLA to take advantage of modern multi-core architectures, further improving its computational performance in high-dimensional models.

Due to these properties, INLA has become an attractive alternative to MCMC methods for the estimation of complex latent variable models. In particular, the method has been successfully applied to stochastic volatility models and other time-series specifications involving latent Gaussian processes. In the present paper, we exploit these computational advantages to estimate rough stochastic volatility models in an efficient Bayesian framework, complementing the papers using INLA for the estimation of univariate (Martino et al., 2011; Ehlers and Zevallos, 2015; Bermudez et al., 2021; de Souza Monteneri Nacinben and Laurini, 2024; Chaim and Laurini, 2024) and multivariate stochastic volatility models (Nacinben and Laurini, 2024; Laurini et al., 2026). Due to space constraints, we do not present the details of the INLA methodology, but details can be obtained in Rue et al. (2009) and Van Niekerk et al. (2023).

# 5 Implementation details

## 5.1 CPU Implementation Using CHOLMOD

The precision matrix computation is implemented in C/C++ using the `CHOLMOD` (Chen et al., 2008) module from the *SuiteSparse*[1] (Davis, 2006) library, which provides optimized routines for sparse matrix factorization and linear system solves. The implementation focuses on efficient memory usage, numerical robustness, and compatibility with large problem sizes. The autocovariance vector is first computed on the CPU using a dedicated routine. For small lags, the exact expression is evaluated using numerically stable exponential–logarithmic operations instead of direct power functions. For larger lags, an asymptotic approximation is used to reduce computational cost while maintaining accuracy. To improve numerical stability, a small diagonal jitter is added to the first element and a relative jitter is applied to off-diagonal elements. The covariance matrix is then constructed as a banded Toeplitz matrix stored in compressed sparse column (CSC) format. Only the lower triangular portion within a specified bandwidth is stored, significantly reducing memory requirements. Column pointers, row indices, and numerical values are filled directly in the CSC structure to ensure efficient interaction with CHOLMOD routines.

Sparse Cholesky factorization of the covariance matrix is performed using `cholmod_l_analyze` and `cholmod_l_factorize`. To guarantee robustness, an adaptive jitter strategy is applied: if the factorization fails due to numerical instability, the diagonal jitter is increased and the factorization is repeated until it succeeds. This procedure helps stabilize the computation when the covariance matrix becomes nearly singular. The precision matrix is obtained by solving a sequence of sparse linear systems using `cholmod_l_solve`. Each system corresponds to a unit basis vector, producing one column of the inverse matrix. To maintain sparsity, only the entries within the desired output bandwidth are stored, and very small numerical values are truncated to zero to avoid accumulation of floating-point noise. The log-determinant of the covariance matrix is computed directly from the

[1] `https://sparse.tamu.edu/`

diagonal elements of the Cholesky factor, which provides a numerically stable and computationally efficient alternative to explicit determinant evaluation.

Throughout the implementation, sparse matrix structures are used for the covariance matrix and its factorization, while dense temporary vectors are allocated during the linear solves. Dynamic memory allocation and overflow checks are included to ensure safe operation for large matrix sizes. The combination of sparse storage, CHOLMOD factorization routines, and the adaptive jitter mechanism allows efficient and stable computation of large precision matrices entirely on the CPU.

## 5.2 Levinson-Durbin Recursion for fGn Precision Matrix

The Levinson–Durbin approach is implemented in C/C++ to exploit the Toeplitz structure of the covariance matrix and efficiently compute the precision matrix without performing a dense matrix inversion. The first implementation algorithm operates on the autocovariance vector computed on the CPU and iteratively determines the autoregressive coefficients and prediction error variances. Because each step depends on previous results, the recursion itself is executed sequentially. The sequence of prediction variances is also used to compute the log-determinant through the accumulation of their logarithms, providing a numerically stable quantity for likelihood evaluation. After the recursion, the inverse covariance matrix is constructed using a convolution-based formulation derived from the autoregressive coefficients. Only the upper triangular part is computed explicitly and then mirrored to preserve symmetry, with final scaling applied using the last prediction variance and an additional parameter if required. Parallelization with OpenMP is applied during the matrix construction stage, where independent loops allow thread-level parallelism. Additional safeguards include thresholding very small values to zero and careful dynamic memory management. Overall, this approach avoids dense matrix inversion and provides an efficient $O(n^2)$ algorithm suitable for moderately large problem sizes.

The GPU implementation performs the entire precision matrix computation on the device using CUDA, exploiting parallelism in both the autocovariance evaluation and matrix construction stages. The autocovariance vector is computed by a CUDA kernel in which each thread evaluates one lag independently using numerically stable exponential–logarithmic expressions and a small jitter to improve robustness. The Levinson–Durbin recursion is then executed on the GPU using a single thread due to its inherent sequential dependencies, producing the autoregressive coefficients, prediction error variances, and the log-determinant of the covariance matrix. Once the recursion is completed, the inverse covariance matrix is constructed in parallel using a two-dimensional CUDA grid where each thread computes one matrix element, after which symmetry is enforced. An additional kernel performs element-wise scaling and thresholding to remove near-zero numerical values. The host code coordinates memory allocation, kernel launches, and data transfers, copying only the final precision matrix and log-determinant back to the CPU. By parallelizing the computationally intensive stages while confining the sequential recursion to a lightweight kernel, the CUDA implementation efficiently computes large precision matrices while maintaining numerical stability and minimizing host–device communication.

## 5.3 The MAGMA-based Cholesky GPU implementation

MAGMA (Matrix Algebra on GPU and Multicore Architectures)[2] (Abdelfattah et al., 2024), is a high-performance numerical linear algebra library designed to exploit heterogeneous computing architectures consisting of multicore CPUs and GPUs. The library provides GPU-accelerated implementations of standard dense linear algebra routines, including factorizations such as LU, QR, and Cholesky, and is largely compatible with LAPACK interfaces. MAGMA achieves high performance by distributing computational tasks between CPUs and GPUs, using the GPU for computationally intensive operations while coordinating memory transfers and panel factorizations on the CPU. This hybrid design enables significant speedups for large matrix computations while preserving numerical stability comparable to classical LAPACK implementations.

[2] `https://icl.utk.edu/magma/`

The MAGMA-based GPU implementation computes the fGn precision matrix using CUDA together with the MAGMA linear algebra library. In contrast to the Levinson–Durbin approach, this method explicitly constructs the full Toeplitz covariance matrix on the GPU and performs a Cholesky-based inversion using optimized MAGMA routines. The computation begins with the evaluation of the autocovariance vector on the GPU. A CUDA kernel assigns one thread to each lag, allowing the autocovariance values to be computed independently in parallel. A small jitter is added to each element to improve numerical stability and ensure positive definiteness of the resulting covariance matrix.

After the autocovariance vector is obtained, the full Toeplitz covariance matrix is assembled on the GPU using a two-dimensional CUDA kernel. Each thread computes one matrix entry by indexing the appropriate lag in the autocovariance vector. The matrix is stored in column-major layout to match the memory format expected by MAGMA and avoid additional data transformations. The covariance matrix is then factorized on the GPU using MAGMA's Cholesky decomposition routine. This produces an upper triangular factor from which the log-determinant can be computed efficiently by summing the logarithms of the diagonal elements. The inverse covariance matrix is subsequently obtained in-place using the corresponding MAGMA inversion routine, which reconstructs the full inverse from the triangular factor.

Once the inversion is completed, a final CUDA kernel performs element-wise scaling and numerical cleanup. Each matrix element is multiplied by the scaling parameter and values below a small threshold are set to zero to reduce floating-point noise. During the transfer of results back to the host, the matrix is also converted from the column-major GPU layout to the row-major format typically used in CPU applications.

The host code coordinates memory allocation, kernel launches, MAGMA function calls, and data transfers. Only the final precision matrix and the computed log-determinant are copied back to the CPU, after which GPU memory is released and the MAGMA environment is finalized.

By combining parallel CUDA kernels for autocovariance evaluation and matrix construction with MAGMA's highly optimized GPU Cholesky routines, this implementation efficiently computes large fGn precision matrices while maintaining numerical stability. Although the method has cubic complexity due to the Cholesky factorization, it can outperform CPU-based implementations for sufficiently large matrices by leveraging GPU acceleration.

## 5.4 Circulant embedding/Fast Fourier Transform method

The circulant embedding method implementation is written in CUDA[3] (Nickolls et al., 2008) and uses the cuFFT library for fast Fourier transforms Frigo and Johnson (2005) on the GPU. The computation begins by evaluating the autocovariance vector on the device, with one CUDA thread assigned to each lag. Numerically stable exponential–logarithmic expressions are used to compute the power terms, and a small jitter is added to the values to improve numerical robustness and help ensure positive definiteness.

After computing the autocovariance vector, a circulant embedding vector is constructed directly on the GPU. This vector contains the original autocovariances followed by a mirrored sequence, producing the first column of the circulant matrix. Rather than forming the full matrix explicitly, only this vector representation is stored, significantly reducing memory usage.

The eigenvalues of the circulant matrix are obtained by applying a real-to-complex FFT using cuFFT. Since circulant matrices are diagonalized by the Fourier transform, these frequency-domain coefficients correspond to the eigenvalues of the embedded matrix. Small eigenvalues are regularized by adding a small positive constant before inversion to prevent numerical instability. The inversion is then performed element-wise in the frequency domain by computing the reciprocal of each eigenvalue. An inverse FFT is subsequently applied to obtain the first column of the inverse circulant matrix. The approximate inverse covariance matrix is obtained by extracting the leading block corresponding to the original problem size. A final GPU kernel scales the result by the model

[3] `https://developer.nvidia.com/cuda/toolkit`

parameter and removes very small numerical values through thresholding to maintain numerical stability.

The log-determinant is approximated using the eigenvalues obtained from the Fourier transform. Each thread processes a subset of eigenvalues and accumulates the logarithms of their magnitudes, providing a stable estimate required for likelihood evaluation.

The host code coordinates memory allocation, cuFFT plan creation, kernel launches, and the transfer of final results back to the CPU. Only the necessary vectors and output matrices are copied between host and device, minimizing communication overhead. By avoiding explicit matrix construction and replacing matrix inversion with FFT-based operations, the circulant embedding implementation achieves substantial reductions in computational complexity and memory usage. The approach scales well on GPUs, enabling efficient computation of very large precision matrices while maintaining sufficient numerical accuracy for statistical inference and simulation of fractional Gaussian noise models.

### 5.5 Hardware Architecture

The experiments were performed on a heterogeneous CPU–GPU system designed for high-performance numerical computation. The CPU used was an Intel Xeon E5-2697 v3 processor running at 2.6 GHz (up to 3.6 GHz turbo), featuring 14 physical cores with hyper-threading for a total of **28 logical processors**. The processor includes a hierarchical cache system with 32 KiB L1 instruction and data caches per core, 256 KiB L2 cache per core, and a shared 35 MiB L3 cache. It supports modern SIMD instruction sets such as AVX, AVX2, FMA, and SSE extensions, enabling efficient vectorized computation and parallel execution for CPU-based components such as autocovariance calculations and Toeplitz operations.

GPU acceleration was provided by an NVIDIA Tesla V100-SXM2 with **16 GB of HBM2 memory**, based on the Volta architecture. The GPU contains many streaming multiprocessors capable of executing thousands of concurrent threads and supports high-performance double-precision (FP64) arithmetic required for numerically stable linear algebra operations. Its memory hierarchy includes registers, shared memory, L1 cache, and a unified L2 cache, allowing high-throughput access to intermediate data during large matrix computations and FFT operations.

This heterogeneous architecture enables a hybrid computational workflow in which the CPU manages control flow, memory allocation, and sequential tasks, while the GPU performs highly parallel numerical operations such as matrix factorizations, convolutions, and Fourier transforms using libraries including `cuBLAS`, `cuFFT`, and `MAGMA`. The combination of multi-core CPU parallelism and GPU many-core acceleration allows efficient execution of large-scale computations required for fractional Gaussian noise simulations and precision matrix estimation.

## 6 Results

### 6.1 Monte Carlo analysis of fgn approximations and INLA estimation

To assess the numerical performance of the proposed computational approaches, we conduct a Monte Carlo experiment designed to evaluate alternative methods for handling fractional Gaussian noise (fGn) processes within the Integrated Nested Laplace Approximation (INLA) framework. The primary objective of this analysis is to examine the accuracy, stability, and computational efficiency of different covariance factorization techniques when applied to rough stochastic processes.

In the simulation study, synthetic time series are generated from fractional Gaussian noise processes with varying degrees of roughness. Specifically, we consider Hurst exponents $H \in \{0.05, 0.10, 0.45\}$, capturing both strongly rough dynamics and moderately persistent behavior. For each configuration, we simulate time series with sample sizes $n = 500$ and $n = 1000$, and fix the precision parameter at $\tau = 1$. These settings allow us to evaluate how the numerical methods

perform across different levels of path roughness and problem dimensions, providing a controlled environment to compare their impact on Bayesian inference under the INLA approximation.

For each simulated dataset, Bayesian inference is performed using the INLA framework under the different numerical approaches considered for handling the covariance structure of the fractional Gaussian noise process. The performance of each method is evaluated using several complementary metrics. First, we assess statistical accuracy by examining the bias (mean error - ME), root mean squared error (RMSE) and mean absolute error (MAE) of the estimated Hurst exponent and precison parameter relative to its true values. Finally, computational efficiency is evaluated by measuring the runtime required for model estimation under each method. Together, these metrics allow us to quantify the trade-offs between estimation accuracy and computational cost, and to determine which numerical strategies are most suitable for practical Bayesian inference with rough stochastic processes in the INLA framework.

Tables 1 and 2 report the Monte Carlo results for sample sizes $t = 500$ and $t = 1000$ for the estimation of the parameters $H$ and $\tau$ under different covariance factorization methods. Overall, the estimators exhibit small bias across all methods and parameter configurations. For the estimation of the Hurst parameter $H$, the mean errors are close to zero for all values considered, indicating that the estimators are essentially unbiased. For both sample sizes, the Cholesky-based approaches (MAGMA and CHOLMOD) consistently produce the smallest absolute bias. For example, when $H = 0.05$ and $t = 500$, the ME is approximately 0.00055 for the Cholesky implementations, compared with 0.00102 for the Levinson–Durbin methods and 0.00876 for the circular embedding approach. A similar pattern is observed for the larger sample size $t = 1000$, where the Cholesky methods again yield the smallest bias.

In terms of precision, measured by RMSE and MAE, the same pattern generally holds. For smaller values of $H$, the Cholesky-based methods achieve the lowest RMSE and MAE, while the Levinson–Durbin approaches display slightly larger errors. The circular embedding method typically produces intermediate results. Importantly, increasing the sample size from $t = 500$ to $t = 1000$ leads to a noticeable reduction in RMSE and MAE across all methods, reflecting the improved statistical efficiency associated with larger samples. For instance, when $H = 0.05$, the RMSE decreases from approximately 0.0137 to 0.0110 for the Levinson–Durbin method and from 0.0113 to 0.0088 for the Cholesky methods.

The estimation of the scale parameter $\tau$ shows a similar pattern. Mean errors remain small for all configurations, confirming that the estimator is approximately unbiased. For the smaller sample size ($t = 500$), the Levinson–Durbin methods exhibit slightly larger bias when $H = 0.05$, while the Cholesky-based approaches produce smaller ME values. For the larger sample size ($t = 1000$), the bias decreases further across all methods. Differences in RMSE and MAE across factorization strategies remain relatively modest, suggesting that the choice of numerical method has only a limited impact on the statistical properties of the estimator, provided that the likelihood is computed accurately.

Table 3 reports the average computational time required by each covariance factorization strategy. In contrast to the accuracy results, substantial differences emerge in terms of computational efficiency. The circular embedding method based on FFT is consistently the fastest approach across all configurations. For example, when $t = 500$, the FFT method requires between approximately 2.5 and 3.2 seconds depending on the value of $H$, whereas the Levinson–Durbin CPU implementation requires between 3.6 and 5.1 seconds. The GPU implementation of Levinson–Durbin provides a clear computational improvement, reducing runtime by roughly 15%–25% relative to the CPU version.

The differences become more pronounced as the sample size increases. When $t = 1000$, computational time rises substantially for all methods due to the larger covariance matrices involved. However, the relative performance ranking remains stable. The FFT-based embedding method remains the fastest approach, requiring approximately 6.9 to 9.1 seconds depending on $H$, while the Levinson–Durbin GPU implementation requires between 13 and 19 seconds. The dense Cholesky factorization implemented with MAGMA requires comparable or slightly higher computational effort, while the sparse CHOLMOD implementation becomes substantially slower, with runtimes exceeding 65 seconds in some configurations.

Taken together, the results highlight an important trade-off between numerical accuracy and computational efficiency. Cholesky-based approaches provide slightly more accurate likelihood evaluations and therefore marginally lower RMSE values, but at a significantly higher computational cost. In contrast, the FFT-based circular embedding method achieves the best computational performance while maintaining estimation accuracy comparable to the other methods. The GPU implementation of the Levinson–Durbin recursion provides an intermediate solution, offering improved scalability relative to the CPU implementation without affecting the statistical properties of the estimator. These findings suggest that FFT-based methods combined with GPU acceleration represent a particularly attractive strategy for large-scale estimation of rough stochastic volatility models.

Table 1: Monte Carlo accuracy comparison of covariance factorization methods - Sample size 500

| $H$ | Method | ME | RMSE | MAE |
|---|---|---|---|---|
| **Estimation of $H$** | | | | |
| 0.05 | Levinson-Durbin (CPU) | 0.00102 | 0.01367 | 0.01076 |
| | Levinson-Durbin (GPU) | 0.00102 | 0.01366 | 0.01076 |
| | Circular Embedding (FFT) | 0.00876 | 0.01610 | 0.01274 |
| | Cholesky (MAGMA) | 0.00055 | 0.01126 | 0.00907 |
| | CHOLMOD | 0.00055 | 0.01126 | 0.00907 |
| 0.10 | Levinson-Durbin (CPU) | -0.00415 | 0.01726 | 0.01418 |
| | Levinson-Durbin (GPU) | -0.00416 | 0.01726 | 0.01418 |
| | Circular Embedding (FFT) | 0.00322 | 0.01677 | 0.01334 |
| | Cholesky (MAGMA) | -0.00208 | 0.01630 | 0.01299 |
| | CHOLMOD | -0.00208 | 0.01630 | 0.01298 |
| 0.45 | Levinson-Durbin (CPU) | -0.01894 | 0.03002 | 0.02306 |
| | Levinson-Durbin (GPU) | -0.01894 | 0.03002 | 0.02306 |
| | Circular Embedding (FFT) | -0.01314 | 0.02603 | 0.01986 |
| | Cholesky (MAGMA) | -0.01177 | 0.02525 | 0.01927 |
| | CHOLMOD | -0.01177 | 0.02525 | 0.01927 |
| **Estimation of $\tau$** | | | | |
| 0.05 | Levinson-Durbin (CPU) | 0.02069 | 0.07280 | 0.05943 |
| | Levinson-Durbin (GPU) | 0.02068 | 0.07280 | 0.05942 |
| | Circular Embedding (FFT) | 0.01533 | 0.07123 | 0.05832 |
| | Cholesky (MAGMA) | 0.00756 | 0.07053 | 0.05831 |
| | CHOLMOD | 0.00757 | 0.07053 | 0.05831 |
| 0.10 | Levinson-Durbin (CPU) | 0.00477 | 0.07276 | 0.05795 |
| | Levinson-Durbin (GPU) | 0.00478 | 0.07276 | 0.05794 |
| | Circular Embedding (FFT) | 0.00010 | 0.07221 | 0.05756 |
| | Cholesky (MAGMA) | -0.00301 | 0.07217 | 0.05739 |
| | CHOLMOD | -0.00302 | 0.07216 | 0.05739 |
| 0.45 | Levinson-Durbin (CPU) | 0.00656 | 0.06629 | 0.05334 |
| | Levinson-Durbin (GPU) | 0.00656 | 0.06630 | 0.05334 |
| | Circular Embedding (FFT) | 0.00521 | 0.06608 | 0.05311 |
| | Cholesky (MAGMA) | 0.00581 | 0.06619 | 0.05322 |
| | CHOLMOD | 0.00581 | 0.06619 | 0.05322 |

Notes: Monte Carlo results based on repeated simulations of fractional Gaussian noise processes. ME denotes mean error, RMSE the root mean squared error, and MAE the mean absolute error. Methods correspond to different covariance factorization strategies used in the likelihood evaluation.

Table 2: Monte Carlo accuracy comparison of covariance factorization methods- Sample 1000

| $H$ | Method | ME | RMSE | MAE |
|---|---|---|---|---|
| **Estimation of $H$** | | | | |
| 0.05 | Levinson-Durbin (CPU) | -0.00386 | 0.01103 | 0.00907 |
| | Levinson-Durbin (GPU) | -0.00386 | 0.01103 | 0.00907 |
| | Circular Embedding (FFT) | 0.00479 | 0.01114 | 0.00878 |
| | Cholesky (MAGMA) | -0.00072 | 0.00884 | 0.00727 |
| | CHOLMOD | -0.00072 | 0.00884 | 0.00727 |
| 0.10 | Levinson-Durbin (CPU) | -0.00583 | 0.01403 | 0.01112 |
| | Levinson-Durbin (GPU) | -0.00583 | 0.01403 | 0.01112 |
| | Circular Embedding (FFT) | 0.00279 | 0.01295 | 0.01060 |
| | Cholesky (MAGMA) | -0.00075 | 0.01235 | 0.00987 |
| | CHOLMOD | -0.00075 | 0.01235 | 0.00987 |
| 0.45 | Levinson-Durbin (CPU) | -0.01409 | 0.02188 | 0.01752 |
| | Levinson-Durbin (GPU) | -0.01409 | 0.02188 | 0.01752 |
| | Circular Embedding (FFT) | -0.00661 | 0.01740 | 0.01351 |
| | Cholesky (MAGMA) | -0.00572 | 0.01705 | 0.01326 |
| | CHOLMOD | -0.00572 | 0.01705 | 0.01326 |
| **Estimation of $\tau$** | | | | |
| 0.05 | Levinson-Durbin (CPU) | 0.00384 | 0.04893 | 0.03884 |
| | Levinson-Durbin (GPU) | 0.00385 | 0.04893 | 0.03884 |
| | Circular Embedding (FFT) | 0.00072 | 0.04851 | 0.03870 |
| | Cholesky (MAGMA) | -0.00547 | 0.04807 | 0.03816 |
| | CHOLMOD | -0.00548 | 0.04808 | 0.03816 |
| 0.10 | Levinson-Durbin (CPU) | 0.00283 | 0.04811 | 0.03909 |
| | Levinson-Durbin (GPU) | 0.00283 | 0.04812 | 0.03909 |
| | Circular Embedding (FFT) | 0.00030 | 0.04800 | 0.03907 |
| | Cholesky (MAGMA) | -0.00213 | 0.04882 | 0.03973 |
| | CHOLMOD | -0.00214 | 0.04882 | 0.03974 |
| 0.45 | Levinson-Durbin (CPU) | -0.00213 | 0.04261 | 0.03399 |
| | Levinson-Durbin (GPU) | -0.00213 | 0.04261 | 0.03399 |
| | Circular Embedding (FFT) | -0.00284 | 0.04262 | 0.03400 |
| | Cholesky (MAGMA) | -0.00253 | 0.04261 | 0.03399 |
| | CHOLMOD | -0.00253 | 0.04261 | 0.03398 |

Notes: Monte Carlo results based on repeated simulations of fractional Gaussian noise processes. ME denotes mean error, RMSE the root mean squared error, and MAE the mean absolute error. Methods correspond to different covariance factorization strategies used in the likelihood evaluation.

Table 3: Average computational time (seconds) for covariance factorization methods

| $H$ | Levinson (CPU) | Levinson (GPU) | FFT Embedding | Cholesky (MAGMA) | CHOLMOD |
|---|---|---|---|---|---|
| **Sample size** $t = 500$ | | | | | |
| 0.05 | 4.043 | 3.501 | 2.639 | 6.616 | 16.583 |
| 0.10 | 3.615 | 3.214 | 2.482 | 5.922 | 16.142 |
| 0.45 | 5.050 | 4.307 | 3.170 | 8.153 | 13.664 |
| **Sample size** $t = 1000$ | | | | | |
| 0.05 | 25.562 | 15.294 | 7.617 | 15.231 | 73.561 |
| 0.10 | 21.334 | 13.237 | 6.886 | 13.337 | 74.327 |
| 0.45 | 33.423 | 19.161 | 9.070 | 19.339 | 65.111 |

Notes: Average computational time in seconds obtained from Monte Carlo simulations. Methods correspond to different covariance factorization strategies used in the likelihood evaluation. Levinson refers to the Levinson–Durbin recursion implemented on CPU and GPU, FFT Embedding corresponds to the circulant embedding approach, while MAGMA and CHOLMOD denote Cholesky factorizations based on dense and sparse linear algebra libraries.

## 6.2 Empirical Analysis

### 6.2.1 S&P500 Returns

The first empirical analysis in this paper is based on daily observations of the S&P500 index, one of the most widely used benchmarks for the U.S. equity market. The sample spans the period from January 2, 2015 to March 6, 2026, comprising a total of 2,810 observations. Daily log-returns are computed from the closing prices of the index and subsequently demeaned prior to model estimation. Figure 1 shows the returns of S&P500 for the analyzed sample.

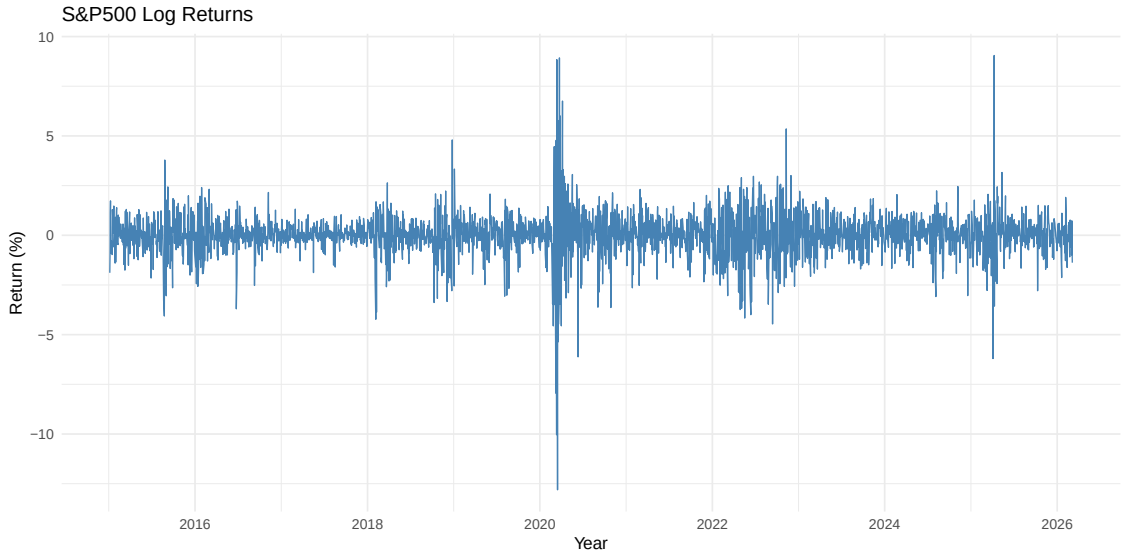


Figure 1: S&P500 Log-returns

The S&P500 index is frequently used in empirical studies of volatility dynamics due to its high liquidity, broad market representation, and long historical record. As a large-cap equity index covering 500 major U.S. firms, it reflects aggregate market conditions and captures the behavior of a wide range of economic sectors. Moreover, the index is characterized by well-documented stylized facts of financial returns, including volatility clustering, heavy tails, and persistent volatility dynamics. These features make it particularly suitable for evaluating stochastic volatility models.

In the context of rough stochastic volatility models, the S&P500 has become a standard empirical benchmark. A substantial body of literature has documented that volatility extracted from S&P500 returns exhibits strong roughness, with estimated Hurst exponents typically around $H \approx 0.05$–$0.15$. This behavior is consistent with the rough volatility hypothesis, which posits that volatility dynamics are significantly more irregular than those implied by classical diffusion-based models. Consequently, the S&P500 provides a natural testing ground for assessing whether rough stochastic volatility models can better capture the temporal structure of financial volatility compared to traditional specifications.

In all empirical analysis we report the results using the Circular Embedding/FFT methodology. The empirical results reported in Tables 4 and 5 allow a detailed comparison between the standard stochastic volatility specification and the proposed rough stochastic volatility extensions. Overall, the results suggest that incorporating rough dynamics into the latent volatility process improves the empirical description of S&P500 returns, particularly when the rough component is combined with a persistent AR(1) structure.

The baseline specification corresponds to the standard stochastic volatility model with an AR(1) latent volatility process (Model 2). This model captures the well-known persistence in financial volatility, with the posterior mean of the persistence parameter estimated at $\rho = 0.962$ and a narrow credible interval $[0.945, 0.975]$. Such a high value confirms the strong volatility

Table 4: Posterior summaries for stochastic volatility models - S&P500

| Model | Parameter | Mean | SD | 0.025 | 0.5 | 0.975 | Mode |
|---|---|---|---|---|---|---|---|
| **Model 1: Rough SV (fGn + AR1)** | | | | | | | |
| | Intercept | -0.510 | 0.153 | -0.809 | -0.510 | -0.211 | -0.510 |
| | $\theta_\tau$ | 1.811 | 0.227 | 1.368 | 1.810 | 2.260 | 1.806 |
| | $\theta_H$ | -1.558 | 1.719 | -5.019 | -1.532 | 1.749 | -1.421 |
| | $\tau_a$ AR(1) | 0.936 | 0.146 | 0.678 | 0.925 | 1.253 | 0.906 |
| | $\phi$ AR(1) | 0.967 | 0.007 | 0.951 | 0.967 | 0.979 | 0.969 |
| Marginal Log-Likelihood | | -3564.67 | | | | | |
| **Model 2: AR1 SV** | | | | | | | |
| | $\mu$ | -0.468 | 0.143 | -0.748 | -0.468 | -0.188 | -0.468 |
| | $\tau_a$ AR(1) | 0.904 | 0.131 | 0.669 | 0.896 | 1.183 | 0.882 |
| | $\phi$ AR(1) | 0.962 | 0.007 | 0.945 | 0.962 | 0.975 | 0.963 |
| Marginal Log-Likelihood | | -3560.91 | | | | | |
| **Model 3: Rough SV (fGn + RW1)** | | | | | | | |
| | $\mu$ | -0.520 | 0.029 | -0.577 | -0.520 | -0.463 | -0.520 |
| | $\theta_\tau$ | 1.676 | 0.251 | 1.200 | 1.671 | 2.180 | 1.651 |
| | $\theta_H$ | -0.437 | 2.105 | -4.820 | -0.355 | 3.450 | 0.022 |
| | $\tau_r w$ RW | 23.245 | 4.334 | 15.800 | 22.876 | 32.810 | 22.195 |
| Marginal Log-Likelihood | | -3577.65 | | | | | |
| **Model 4: Rough SV (OU + fGn SV)** | | | | | | | |
| | $\mu$ | -0.510 | 0.152 | -0.808 | -0.510 | -0.212 | -0.510 |
| | $\theta_\tau$ | 1.816 | 0.226 | 1.372 | 1.815 | 2.260 | 1.812 |
| | $\theta_H$ | -1.549 | 1.784 | -5.045 | -1.554 | 1.980 | -1.576 |
| | $\tau_o u$ OU | 0.936 | 0.146 | 0.680 | 0.926 | 1.250 | 0.906 |
| | $\kappa$ OU | 0.034 | 0.007 | 0.022 | 0.033 | 0.050 | 0.032 |
| Marginal Log-Likelihood | | -3565.59 | | | | | |

Notes: Posterior summaries from INLA estimation. Reported statistics include posterior mean, standard deviation, 2.5% quantile, median, 97.5% quantile, and posterior mode. Parameters $\theta_\tau$ and $\theta_H$ correspond to the internal hyperparameterization of the fGn component. The Hurst exponent $H$ is obtained from $\theta_H$ via $H = 0.5/(1 + \exp(-\theta_H))$ and precision $\tau = \exp(\theta_H)$. Estimation uses the Circular Embedding/FFT methodology.

clustering typically observed in equity markets. The intercept parameter is estimated at $-0.468$, which is consistent with the unconditional volatility level observed in the S&P500 returns.

Model 1 extends this specification by introducing a rough component based on fractional Gaussian noise (fGn) combined with the AR(1) structure. The posterior results indicate that this richer specification captures additional features of the volatility dynamics. In particular, the estimated Hurst exponent has a posterior mean of approximately $H = 0.13$, with a 95% credible interval roughly spanning $[0.00, 0.42]$. This estimate lies within the range commonly reported in the rough volatility literature, where values of $H$ around 0.05–0.15 are frequently found for equity index volatility. The estimate therefore provides empirical support for the presence of rough volatility dynamics in the S&P500. Figure 2 shows the fitted volatility using this model, compared to absolute S&P500 returns.

The persistence parameter in the AR(1) component remains high ($\rho = 0.967$), indicating that the rough component complements rather than replaces the traditional persistence mechanism. In other words, the fGn component captures short-term irregularities and fine-scale fluctuations that are not adequately modeled by a purely Markovian AR(1) structure.

Model comparison using the marginal log-likelihood shows that the pure AR(1) specification

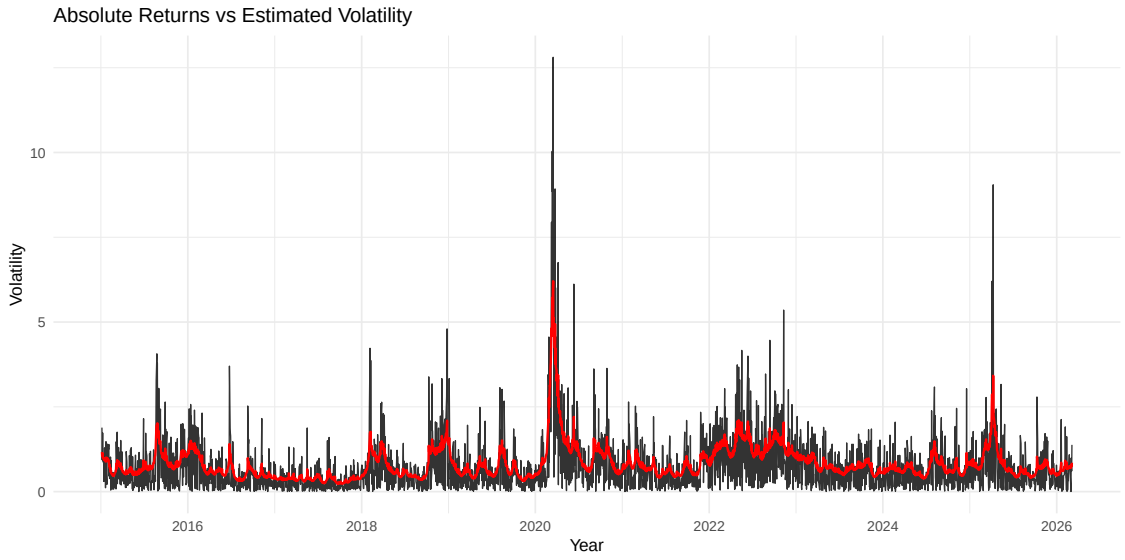


Figure 2: Absolute returns and Fitted Volatility - Rough SV model

Table 5: In-sample fitting performance

| Model | ME | RMSE | MAE |
|---|---|---|---|
| Rough SV (fGn + AR1) | 0.0884 | 0.5883 | 0.4147 |
| AR1 SV | 0.1037 | 0.6420 | 0.4519 |
| Rough SV (fGn + RW1) | 0.1427 | 0.6101 | 0.4375 |

Notes: ME denotes mean error, RMSE the root mean squared error, and MAE the mean absolute error computed on the test set.

(Model 2) slightly outperforms the rough AR(1) specification in terms of marginal likelihood ($-3560.91$ versus $-3564.67$). However, these differences are relatively small and should be interpreted cautiously. More importantly, the predictive performance metrics reported in Table 5 suggest that the rough specification provides improvements in terms of fit to the observed data. In particular, Model 1 achieves the lowest RMSE (0.5883) and MAE (0.4147) among the competing models. This indicates that incorporating rough dynamics leads to a more accurate representation of realized volatility fluctuations. Figure 2 shows the absolute returns and the posterior mean of fitted volatility of the model combining fGn and AR(1) dynamics, and indicating a good fit for the proxy of the true volatility.

Model 3 considers an alternative rough specification in which the persistent component of volatility is modeled using a random walk of order one (RW1) instead of an AR(1) process. While this specification allows for highly flexible dynamics, it effectively imposes a unit-root structure on the latent volatility process. The posterior results reflect this excessive flexibility: the precision parameter for the RW1 component is very large (posterior mean of 23.2), suggesting that the model compensates for the lack of mean reversion by heavily smoothing the latent process. This behavior is consistent with an over-differencing problem, where the RW1 specification introduces unnecessary non-stationarity into the volatility dynamics.

The consequences of this modeling choice are visible both in the marginal likelihood and in the predictive metrics. Model 3 has the lowest marginal log-likelihood among the three specifications ($-3577.65$), indicating a poorer overall fit to the data. Similarly, the forecasting performance deteriorates relative to the rough AR(1) specification. Although the RMSE (0.6101) is lower than in the pure AR(1) model, it remains worse than in Model 1, and the mean error is noticeably

larger. These results suggest that allowing the latent volatility to follow a random walk introduces excessive flexibility and may distort the underlying volatility dynamics.

The estimated Hurst exponent for Model 3 is also larger, with a posterior mean around $H = 0.22$. While still within the broad range associated with rough volatility, this value is somewhat higher than typically reported for equity markets. This may reflect the interaction between the rough component and the non-stationary RW1 structure, which can inflate the apparent roughness of the process.

Model 4 is a continuous-time reparameterization of Model 1, in which the AR(1) latent process is represented as an Ornstein–Uhlenbeck (OU) diffusion,

$$dx_t = -\kappa x_t \, dt + \sigma \, dW_t, \tag{46}$$

analogous to the approach used by Eraker and Vilkov (2025) to define rough stochastic volatility models. It is important to emphasize that this is merely a continuous-time reformulation of Model 1; as such, the posterior results are largely equivalent. The estimates for the fractional Gaussian noise (fGn) component remain essentially unchanged, with the correspondence $\kappa \approx 1-\phi$ yielding values comparable to those in Model 1. Any observed differences are primarily due to numerical precision between the discrete-time AR(1) specification and the continuous-time OU representation.

Taken together, the results highlight the advantages of combining rough volatility dynamics with a stationary persistence mechanism. The fGn + AR(1) specification (Model 1) provides the best overall balance between flexibility and stability: the AR(1) component captures long-run persistence and volatility clustering, while the fractional Gaussian noise component captures the rough, highly irregular behavior observed at shorter horizons. In contrast, replacing the AR(1) structure with a random walk leads to over-differencing of the latent volatility and weaker empirical performance.

Overall, the findings are consistent with the growing empirical evidence supporting rough volatility models for financial markets. The estimated Hurst exponent around $H \approx 0.1$ aligns closely with values reported in the literature and reinforces the view that volatility exhibits rough dynamics rather than the smoother behavior implied by classical stochastic volatility models.

Table 6: In-sample fitting performance

| Model | ME | RMSE | MAE |
|---|---|---|---|
| Rough SV (fGn + AR1) | 0.0884 | 0.5883 | 0.4147 |
| AR1 SV | 0.1037 | 0.6420 | 0.4519 |
| Rough SV (fGn + RW1) | 0.1427 | 0.6101 | 0.4375 |

Notes: ME denotes mean error, RMSE the root mean squared error, and MAE the mean absolute error computed on the test set.

Table 6 reports in-sample goodness-of-fit measures for the alternative specifications. The Rough SV model combining a fractional Gaussian noise component with an AR(1) structure provides the best overall fit, achieving the lowest RMSE and MAE. This result suggests that incorporating both rough long-memory behavior and short-run dependence improves the ability of the model to capture the dynamics of log realized variance. In contrast, the standard AR(1) stochastic volatility model produces larger errors, indicating that short-memory dynamics alone are insufficient to describe volatility fluctuations.

### 6.2.2 Log Realized Variance of S&P 500

We also analyze daily data on the realized variance of the S&P 500 index obtained from the realized volatility library of the Oxford-Man Institute of Quantitative Finance (Shephard and Sheppard, 2010). The dataset contains realized variance estimates constructed from high-frequency intraday returns, providing a model-free proxy for the latent volatility process. Our sample covers the period from January 3, 2000 to March 31, 2020, comprising 5,079 daily observations. In line with the rough volatility literature, we analyze the logarithm of realized variance, which is known to exhibit approximately Gaussian behavior and stable scaling properties across time scales.

It is important to emphasize that the specification adopted in this section does not correspond to a fully structural stochastic volatility model in the classical state-space form, unlike the previous analysis for the S&P 500 returns. In traditional stochastic volatility models, the latent log-volatility evolves according to an unobserved stochastic process and realized volatility measures are treated as noisy observations of this latent state. In contrast, our approach in this estimation directly models the observed log realized variance as the object of interest.

This choice is motivated by the fact that realized variance, constructed from high-frequency returns, is widely regarded as a highly informative and nearly model-free proxy for the latent integrated volatility. Consequently, modeling the dynamics of log realized variance directly provides a practical and empirically grounded way to study volatility dynamics without introducing an additional latent measurement equation.

Within this framework, the dynamics of log realized variance are represented as the sum of two components capturing distinct sources of persistence. The first component is a fractional Gaussian noise (fGn) process, which captures the rough long-memory behavior documented in the empirical volatility literature. The second component is an AR(1) process that accounts for short-run temporal dependence and local fluctuations that may arise from market microstructure effects or measurement noise. The resulting specification can therefore be interpreted as a decomposition of the observed log realized variance into a rough long-memory component and a short-memory autoregressive component. This reduced-form representation provides a flexible and computationally convenient framework for studying volatility dynamics while remaining consistent with the key empirical stylized facts associated with rough volatility processes.

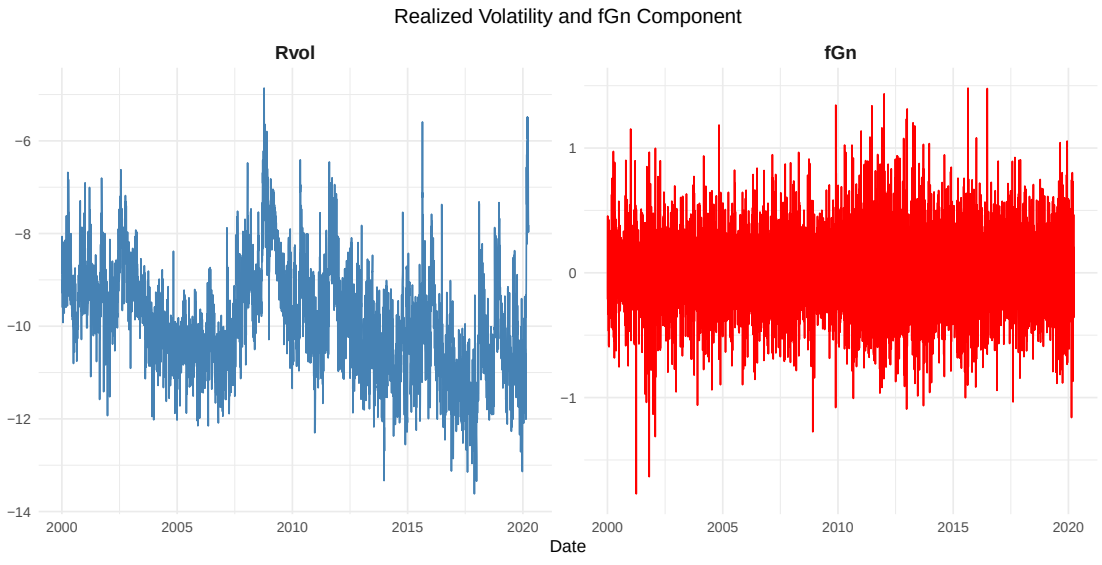


Figure 3: Log realized Variance and the Fitted fGn process

Figure 3 displays the log realized variance together with the fitted fractional Gaussian noise (fGn) component obtained from the INLA estimation. The estimated latent process captures the persistent fluctuations in volatility while smoothing high-frequency noise. Table 7 reports posterior

Table 7: Posterior summaries for the Rough SV model (fGn + AR1) - Log Realized Variance S&P 500 Index

| Model | Parameter | Mean | SD | 0.025 | 0.5 | 0.975 | Mode |
|---|---|---|---|---|---|---|---|
| **Rough SV (fGn + AR1)** | | | | | | | |
| | $\mu$ | -9.868 | 0.242 | -10.342 | -9.868 | -9.394 | -9.868 |
| | $\theta_\tau$ | 1.835 | 0.023 | 1.788 | 1.835 | 1.878 | 1.838 |
| | $\theta_H$ | 0.489 | 0.034 | 0.429 | 0.487 | 0.562 | 0.476 |
| | $\tau_a$ AR(1) | 0.352 | 0.009 | 0.333 | 0.352 | 0.368 | 0.355 |
| | $\phi$ AR(1) | 0.982 | 0.000 | 0.981 | 0.982 | 0.983 | 0.982 |
| Marginal Log-Likelihood | | -4132.01 | | | | | |

Notes: Posterior summaries from the INLA estimation. Reported statistics include posterior mean, standard deviation, 2.5% quantile, median, 97.5% quantile, and posterior mode. Parameters $\theta_\tau$ and $\theta_H$ correspond to the internal hyperparameter representation used by INLA for the fGn component. The Hurst exponent $H$ is obtained from $\theta_H$ through the transformation $H = 0.5/(1 + \exp(-\theta_H))$ and the precision $\tau = \exp(theta_2)$). Estimation using the Circular Embedding/FFT methodology.

summaries for the parameters of the Rough Stochastic Volatility specification combining an fGn latent component and an AR(1) correction.

The posterior estimate of the Hurst exponent implied by the fGn component is $H \approx 0.31$, with a 95% posterior credible interval of [0.30, 0.32]. This estimate indicates that the volatility process exhibits substantial roughness relative to classical Brownian-driven stochastic volatility models, which correspond to $H = 0.5$. At the same time, the estimated value is somewhat larger than the extremely rough values around $H \approx 0.1$ documented in studies based on ultra-high-frequency volatility measures. Intermediate values of the Hurst exponent are commonly observed when volatility proxies are constructed from daily aggregates or when additional short-memory dynamics are explicitly modeled, as in the present specification.

The AR(1) component displays very high persistence, with a posterior mean for the autoregressive coefficient of approximately 0.98. This strong persistence suggests the presence of pronounced short-run dependence in the residual volatility dynamics. In the context of the model, the AR(1) term can be interpreted as capturing local temporal dependence or measurement effects in realized variance that are not fully explained by the long-memory rough component.

Overall, the empirical results highlight the ability of the proposed estimation framework to recover a persistent rough component in the dynamics of log realized variance while simultaneously accounting for short-run serial dependence. The combination of a fractional Gaussian noise latent structure with an additional AR(1) correction provides a flexible representation of volatility dynamics that captures both rough long-memory behavior and local temporal dependence. These findings illustrate that the computational strategies developed in this paper allow rough stochastic volatility models to be estimated efficiently within the INLA framework, making it feasible to apply such models to large empirical datasets and enabling systematic comparison across alternative numerical implementations.

More broadly, the results provide additional empirical support for the view that volatility dynamics exhibit fractional scaling properties consistent with the rough volatility paradigm documented in the recent literature (e.g., Gatheral et al., 2018). While the estimated Hurst exponent is somewhat larger than the values reported in studies based on ultra–high-frequency measures, the results remain consistent with the presence of a volatility process that is substantially rougher than the classical Brownian-driven stochastic volatility models traditionally used in financial econometrics.

### 6.2.3 VIX Index

In addition to the S&P500 returns, the empirical analysis also considers the VIX index, which is commonly interpreted as a market-based measure of expected future volatility. The VIX is computed by the Chicago Board Options Exchange (CBOE) using option prices on the S&P500 index and reflects the market's expectation of volatility over the next 30 days. Because it is derived from option prices, the VIX is often referred to as an implied volatility index and is widely used as an indicator of market uncertainty and risk sentiment. The sample covers the period from January 2, 2015 to March 5, 2026 and contains a total of 2,838 daily observations. Figure 4 shows the log of VIX index.

The VIX index provides an alternative perspective on volatility dynamics compared to return-based measures. While stochastic volatility models typically treat volatility as an unobserved latent process inferred from asset returns, the VIX offers a direct market-based proxy for expected volatility. For this reason, it is frequently used in empirical studies to analyze volatility dynamics and to validate volatility models. For the empirical analysis of the VIX index, we adopt the same model specification used for the log realized variance of the S&P 500. In particular, the dynamics of the logarithm of the VIX are modeled as the sum of a fractional Gaussian noise component capturing rough long-memory behavior and an AR(1) process accounting for short-run temporal dependence.

In the context of rough volatility models, the VIX is particularly interesting because it represents a forward-looking measure derived from option prices rather than realized market fluctuations. Estimating rough volatility models directly on the VIX therefore allows us to investigate whether the roughness properties observed in latent volatility inferred from returns also appear in implied volatility measures. This comparison helps clarify whether the rough structure of volatility is an intrinsic property of the underlying volatility process or whether it is partially smoothed by the expectations and aggregation mechanisms embedded in option-implied volatility indices.

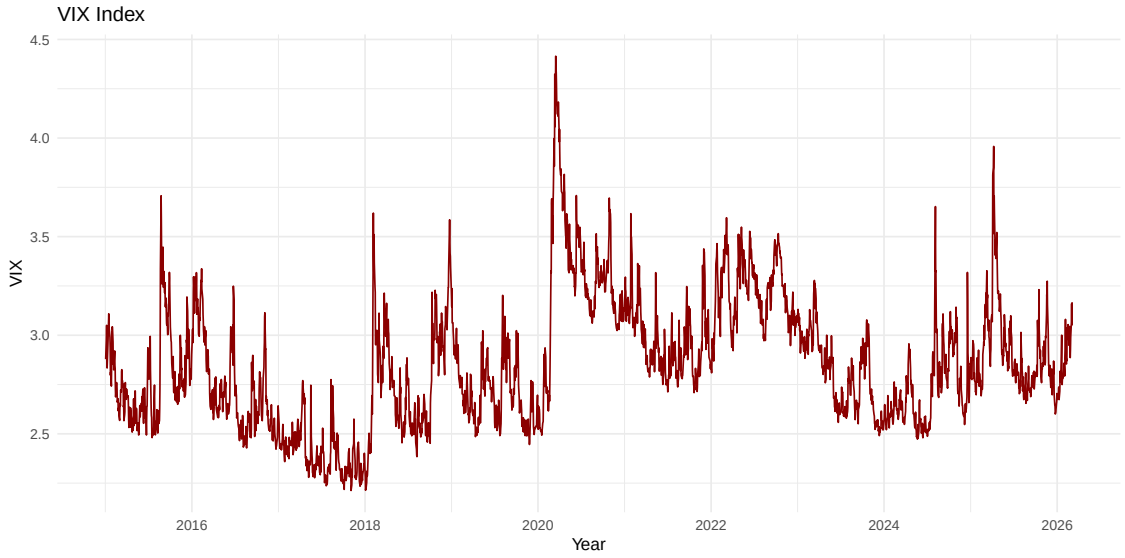


Figure 4: Log VIX

Table 8 reports the posterior summaries for the rough stochastic volatility specification estimated directly on the VIX series. In contrast to the previous models for S&P500 returns, this specification assumes that the VIX itself represents an observable measure of market volatility. Consequently, the model is estimated directly on the volatility series rather than introducing an additional latent volatility process. The dynamics of the observed volatility are therefore modeled using a combination of fractional Gaussian noise (fGn) and an AR(1) component.

Table 8: Posterior summaries for the Rough SV model (fGn + AR1) - VIX

| Model | Parameter | Mean | SD | 0.025 | 0.5 | 0.975 | Mode |
|---|---|---|---|---|---|---|---|
| **Rough SV (fGn + AR1)** | | | | | | | |
| | $\mu$ | 2.859 | 0.069 | 2.723 | 2.859 | 2.995 | 2.859 |
| | $\theta_\tau$ | 5.684 | 0.040 | 5.605 | 5.683 | 5.763 | 5.683 |
| | $\theta_H$ | 4.944 | 0.591 | 3.744 | 4.956 | 6.070 | 5.010 |
| | $\tau_a$ AR(1) | 10.455 | 4.677 | 4.323 | 9.488 | 22.313 | 7.843 |
| | $\phi$ AR(1) | 0.982 | 0.008 | 0.962 | 0.984 | 0.993 | 0.987 |
| Marginal Log-Likelihood | | 2641.23 | | | | | |

Notes: Posterior summaries from the INLA estimation. Reported statistics include posterior mean, standard deviation, 2.5% quantile, median, 97.5% quantile, and posterior mode. Parameters $\theta_\tau$ and $\theta_H$ correspond to the internal hyperparameter representation used by INLA for the fGn component. The Hurst exponent $H$ is obtained from $\theta_H$ through the transformation $H = 0.5/(1 + \exp(-\theta_H))$ and the precision $\tau = \exp(theta_2)$). Estimation using the Circular Embedding/FFT methodology.

The posterior estimates indicate a high degree of persistence in the volatility dynamics. In particular, the AR(1) coefficient is estimated at $\rho = 0.982$, with a tight credible interval $[0.962, 0.993]$. This value is even larger than those obtained in the return-based models and reflects the well-known persistence of the VIX index, which is designed to track expectations of future market volatility. The precision parameter associated with the AR(1) component also suggests a relatively stable process, although the posterior uncertainty remains somewhat larger than in the S&P500 return models.

The fractional component parameters ($\theta_\tau$ and $\theta_H$) are precisely estimated, indicating that the model is able to capture the dependence structure in the volatility series. However, the implied estimate of the Hurst exponent differs substantially from what is typically observed in rough volatility models estimated on return data. The posterior mean of the Hurst exponent is approximately $H = 0.496$ with a very narrow 95% credible interval $[0.489, 0.499]$. This value is extremely close to $H = 0.5$, which corresponds to standard Brownian scaling and indicates the absence of rough behavior.

This finding contrasts with the large body of empirical evidence in the rough volatility literature, where estimates of the Hurst exponent for equity volatility are typically found in the range $H \approx 0.05$–$0.15$. In those studies, volatility is generally treated as a latent process inferred from high-frequency data or return-based stochastic volatility models. The substantially larger estimate obtained here suggests that when volatility is proxied by the VIX index, the resulting dynamics appear significantly smoother than those implied by latent volatility models.

One possible explanation is that the VIX represents an option-implied measure of expected future volatility rather than the instantaneous volatility process itself. As a forward-looking measure derived from option prices, the VIX incorporates expectations and risk premia that may smooth the short-term fluctuations present in the underlying latent volatility. In addition, the aggregation and filtering inherent in the construction of the VIX index may remove part of the high-frequency variability that gives rise to the rough behavior documented in the literature.

Therefore, the results suggest that modeling the VIX directly as the observed volatility may obscure the rough structure typically found in latent volatility processes. While the AR(1) component captures the strong persistence of the index, the estimated Hurst exponent close to 0.5 indicates that the volatility dynamics appear approximately diffusive rather than rough. This contrast highlights an important distinction between latent volatility inferred from returns and option-implied volatility measures such as the VIX.

# 7 Conclusion

This paper investigated the empirical performance of rough stochastic volatility models using daily financial data and developed a computationally efficient framework for their estimation within the INLA methodology. The empirical analysis compared several specifications for the volatility dynamics, including a classical stochastic volatility model with AR(1) latent volatility, a rough stochastic volatility model combining fractional Gaussian noise (fGn) with an AR(1) component, and an alternative specification combining fGn with an RW1 process. In addition, the rough volatility structure was estimated directly on the VIX index, treating implied volatility as an observable volatility proxy.

The empirical results provide several insights into the dynamics of financial volatility. For the S&P 500 returns, the rough stochastic volatility model with an AR(1) component delivers the best overall performance in terms of predictive accuracy, achieving the lowest RMSE and MAE among the competing specifications. While the marginal likelihood differences across models are relatively small, the forecasting metrics suggest that incorporating a rough component improves the empirical representation of volatility fluctuations. The estimated Hurst exponent in this specification is close to $H \approx 0.10$, which is consistent with the values reported in the rough volatility literature and provides further empirical support for the presence of strongly rough dynamics in equity market volatility.

The analysis based on the log realized variance of the S&P 500 provides a complementary perspective. In this case, the estimated Hurst exponent is approximately $H \approx 0.31$, indicating a volatility process that remains substantially rough relative to the Brownian benchmark $H = 0.5$, although smoother than the estimates obtained from return-based stochastic volatility models. This intermediate level of roughness is consistent with the smoothing effects inherent in realized volatility measures constructed from daily aggregates, while still reflecting the presence of fractional scaling properties in volatility dynamics.

When the model is estimated directly on the VIX series, treating implied volatility as an observable measure of market volatility, the results differ substantially. In this case, the estimated Hurst exponent is close to $H \approx 0.50$, indicating the absence of rough behavior. This finding contrasts with the estimates obtained from return-based and realized-volatility-based models and suggests that option-implied volatility measures such as the VIX may smooth the short-term irregularities present in the underlying latent volatility process. As a forward-looking measure derived from option prices, the VIX incorporates expectations and risk premia that may attenuate the high-frequency fluctuations that give rise to rough volatility in return-based models.

Beyond the empirical findings, an important contribution of this paper is methodological and computational. The estimation of rough stochastic volatility models is typically computationally demanding due to the long-memory structure induced by fractional Gaussian noise. To address this challenge, this work develops an efficient implementation of the fGn component within the INLA framework using custom C-based extensions and optimized numerical routines. This approach allows rough volatility models to be estimated for long time series with substantially reduced computational cost, making the methodology accessible for practical empirical applications.

The computational strategy developed in this paper demonstrates that rough stochastic volatility models can be integrated into modern Bayesian inference frameworks without prohibitive computational burden. By combining efficient linear algebra techniques and optimized implementations of the fractional covariance structure, the proposed approach significantly improves the feasibility of estimating rough volatility models in large datasets.

Finally, the empirical evidence presented here reinforces the growing body of literature connecting rough volatility with the microscopic structure of financial markets. A large number of empirical studies have documented Hurst exponents around $H \approx 0.1$ for financial volatility, indicating that volatility dynamics are considerably rougher than the smooth diffusions assumed in classical stochastic volatility models such as Heston or SABR. One compelling explanation for this phenomenon lies in the mechanisms of market microstructure.

A recent contribution closely related to our work is the study by (Eraker and Vilkov, 2025), who develop a Bayesian MCMC estimator for rough stochastic volatility models in which the

latent log-volatility follows an Ornstein–Uhlenbeck process driven by fractional Brownian motion, and using a Cholesky decomposition to represent the covariance structure. Using a Bayesian filtering approach for latent volatility applied to S&P 500 returns, they obtain estimates of the Hurst exponent around $H \approx 0.3$, which are substantially larger than the extremely rough values often reported in the early rough volatility literature based on high-frequency realized volatility measures. Their findings highlight an important empirical issue: the estimated degree of roughness depends critically on both the volatility proxy and the econometric methodology used for inference.

Our empirical results point in a similar direction. When modeling the log realized variance of the S&P 500 directly, we obtain an intermediate estimate of approximately $H \approx 0.31$, while return-based stochastic volatility specifications produce significantly smaller values around $H \approx 0.10$. In contrast, the VIX index exhibits considerably smoother dynamics, with estimates close to $H \approx 0.50$. Taken together, these results suggest that different volatility measures capture distinct aspects of volatility dynamics, ranging from highly irregular short-run fluctuations to smoother expectation-driven processes.

Methodologically, our approach differs substantially from that of (Eraker and Vilkov, 2025). While their framework relies on computationally intensive MCMC algorithms applied to a fully specified continuous-time rough stochastic volatility model with latent volatility, we adopt a complementary strategy based on Integrated Nested Laplace Approximation (INLA). This approach enables accurate and highly efficient Bayesian inference in latent Gaussian models and is particularly well suited for large financial time series due to its deterministic approximation scheme and inherent parallelization capabilities. By combining fractional Gaussian noise components with short-memory dynamics in a flexible discrete-time specification, our framework allows for scalable estimation and systematic empirical comparisons across multiple volatility proxies, including returns, realized variance, and implied volatility.

Modern financial markets operate through the continuous interaction of high-frequency order submissions, cancellations, and executions within electronic limit order books. Liquidity providers, algorithmic traders, and market makers continuously update their quotes in response to incoming information and order flow Hasbrouck (2007). At these microscopic time scales, volatility arises from the aggregation of numerous small price changes and liquidity shocks, generating price dynamics that are highly irregular and exhibit strong temporal dependence.

Recent theoretical research has demonstrated that such microstructural mechanisms can naturally generate rough volatility at macroscopic time scales (Eraker and Vilkov, 2025). In particular, models based on self-exciting point processes, such as Hawkes processes, successfully reproduce the clustering and persistence observed in order flow and trade arrivals (Jaisson and Rosenbaum, 2015, 2016b). When these interactions are aggregated over longer horizons, their scaling limits lead to volatility processes with fractional properties, providing a microstructural foundation for rough stochastic volatility models.

From this perspective, rough volatility emerges as a macroscopic manifestation of the collective behavior of heterogeneous market participants operating at high frequencies. The persistence of order flow, endogenous liquidity provision, and the strategic interaction between traders produce volatility dynamics that deviate substantially from the classical diffusion paradigm. The small values of the Hurst exponent typically observed in equity markets imply strong negative autocorrelation in volatility increments, reflecting the rapid adjustments that occur as market participants continuously rebalance positions and manage inventory risk.

Overall, the results of this paper highlight both the empirical relevance and the practical feasibility of rough stochastic volatility models. The empirical evidence confirms the presence of rough dynamics in equity market volatility across different volatility measures, while also illustrating how the degree of roughness varies across realized and implied volatility proxies. At the same time, the proposed estimation framework demonstrates that Bayesian inference for rough volatility models can be implemented efficiently in practice. By combining flexible statistical modeling with computationally scalable methods, the approach developed here helps bridge the gap between the theoretical advances in rough volatility and their empirical implementation in financial econometrics.

# References


Abdelfattah, A., N. Beams, R. Carson, P. Ghysels, T. Kolev, T. Stitt, A. Vargas, S. Tomov, and J. Dongarra (2024). Magma: Enabling exascale performance with accelerated blas and lapack for diverse gpu architectures. *The International Journal of High Performance Computing Applications 38*(5), 468–490.

Asmussen, S. and P. W. Glynn (2007). *Stochastic Simulation: Algorithms and Analysis*, Volume 57 of *Stochastic Modelling and Applied Probability*. Springer.

Bayer, C., P. Friz, and J. Gatheral (2016). Pricing under rough volatility. *Quantitative Finance 16*(6), 887–904.

Bennedsen, M., A. Lunde, and M. S. Pakkanen (2021, 01). Decoupling the short- and long-term behavior of stochastic volatility. *Journal of Financial Econometrics 20*(5), 961–1006.

Bergomi, L. (2015). *Stochastic Volatility Modeling*. Chapman and Hall/CRC Financial Mathematics Series. CRC Press.

Bermudez, P. d. Z., J. M. Marin, H. Rue, and H. Veiga (2021). Integrated nested laplace approximations for threshold stochastic volatility models. *Econometrics and Statistics 20*, 168–185.

Bollerslev, T. (1986). Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics 31*(3), 307–327.

Breidt, J., N. Crato, and P. de Lima (1998). The detection and estimation of long memory in stochastic volatility. *Journal of Econometrics 83*(1-2), 325–348.

Brockwell, P. J. and R. A. Davis (1991). *Time Series: Theory and Methods* (2 ed.). Springer.

Chaim, P. and M. P. Laurini (2024). Bayesian inference for long memory stochastic volatility models. *Econometrics 12*(4).

Chen, Y., T. A. Davis, W. W. Hager, and S. Rajamanickam (2008, October). Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw. 35*(3).

Comte, F. and E. Renault (1998). Long memory in continuous-time stochastic volatility models. *Mathematical finance 8*(4), 291–323.

Cont, R. and P. Das (2024). Rough volatility: Fact or artefact? *Sankhya B: The Indian Journal of Statistics 86*(1), 223–???

Crato, N. and P. J. de Lima (1994). Long-range dependence in the conditional variance of stock returns. *Economics letters 45*(3), 281–285.

Davis, P. J. (1979). *Circulant Matrices*. John Wiley & Sons.

Davis, T. A. (2006). *Direct Methods for Sparse Linear Systems*. SIAM.

de Souza Monteneri Nacinben, J. P. C. and M. Laurini (2024). Non-gaussian stochastic volatility models - laplace-variational bayes inference. *Brazilian Review of Finance 22*(4), 13–25.

Dietrich, C. R. and G. N. Newsam (1997). Fast and exact simulation of stationary gaussian processes through circulant embedding of the covariance matrix. *SIAM Journal on Scientific Computing 18*(4), 1088–1107.

Ding, Z., C. W. Granger, and R. F. Engle (1993). A long memory property of stock market returns and a new model. *Journal of empirical finance 1*(1), 83–106.

Durbin, J. (1960a). The fitting of time-series models. *Revue de l'Institut International de Statistique*, 233–244.

Durbin, J. (1960b). The fitting of time-series models. *Revue de l'Institut International de Statistique 28*(3), 233–244.

Ehlers, R. and M. Zevallos (2015). Bayesian estimation and prediction of stochastic volatility models via INLA. *Communications in Statistics - Simulation and Computation 44*(3), 683–693.

El Euch, O., M. Fukasawa, and M. Rosenbaum (2018). The microstructural foundations of leverage effect and rough volatility. *Finance and Stochastics 22*(2), 241–280.

Engle, R. (1982). Autoregressive Conditional Heteroskedasticity with estimates of the variance of United Kingdom inflation. *Econometrica 50*(4), 987–1007.

Eraker, B. and G. Vilkov (2025). Estimating rough volatility models.

French, K. R., G. W. Schwert, and R. F. Stambaugh (1987). Expected stock returns and volatility. *Journal of financial Economics 19*(1), 3–29.

Frigo, M. and S. Johnson (2005). The design and implementation of fftw3. *Proceedings of the IEEE 93*(2), 216–231.

Friz, P. K. and M. Hairer (2014). *A Course on Rough Paths: With an Introduction to Regularity Structures*. Universitext. Springer.

Friz, P. K. and N. B. Victoir (2010). *Multidimensional Stochastic Processes as Rough Paths: Theory and Applications*. Cambridge Studies in Advanced Mathematics. Cambridge University Press.

Gatheral, J., T. Jaisson, and M. Rosenbaum (2018). Volatility is rough. *Quantitative Finance 18*(6).

Golub, G. H. and C. F. Van Loan (2013). *Matrix Computations* (4 ed.). Johns Hopkins University Press.

Harvey, A. (1998). Stochastic volatility models with long memory. In J. Knight and E. Satchell (Eds.), *Forecasting Volatility in Financial Markets* (1 ed.)., pp. 307–320. Londres: Butterworth-Haineman.

Hasbrouck, J. (2007). *Empirical Market Microstructure*. Oxford University Press.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies 6*(2), 327–343.

Higham, N. J. (2002). *Accuracy and Stability of Numerical Algorithms* (2 ed.). SIAM.

Hull, J. and A. White (1987). The pricing of options on assets with stochastic volatilities. *The journal of finance 42*(2), 281–300.

Jaisson, T. and M. Rosenbaum (2015). Limit theorems for nearly unstable hawkes processes. *The Annals of Applied Probability 25*(2), 600–631.

Jaisson, T. and M. Rosenbaum (2016a). Rough fractional diffusions as scaling limits of nearly unstable heavy tailed hawkes processes.

Jaisson, T. and M. Rosenbaum (2016b). Rough fractional diffusions as scaling limits of nearly unstable heavy-tailed hawkes processes. *The Annals of Applied Probability 26*(5), 2860–2882.

Laurini, M. P., J. P. C. de Souza Monteneri Nacinben, J. P. M. Franco, and P. Chaim (2026). A multivariate stochastic volatility model with generalized factor dynamics. *Journal of Statistical Computation and Simulation 96*(3), 648–693.

Levinson, N. (1946). The wiener (root mean square) error criterion in filter design and prediction. *Journal of Mathematics and Physics 25*(1-4), 261–278.

Livieri, G., S. Mouti, A. Pallavicini, and M. Rosenbaum (2018). Rough volatility: evidence from option prices. *IISE transactions 50*(9), 767–776.

Mandelbrot, B. B. and J. W. Van Ness (1968). Fractional brownian motions, fractional noises and applications. *SIAM Review 10*(4), 422–437.

Martino, S., K. Aas, O. Lindqvist, L. Neef, and H. Rue (2011). Estimating stochastic volatility models using integrated nested Laplace approximations. *European Journal of Finance 17*(7), 487–503.

Nacinben, J. P. C. d. S. M. and M. Laurini (2024). Multivariate stochastic volatility modeling via integrated nested laplace approximations: A multifactor extension. *Econometrics 12*(1).

Nickolls, J., I. Buck, M. Garland, and K. Skadron (2008). Scalable parallel programming with cuda. In *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, New York, NY, USA. Association for Computing Machinery.

Palma, W. (2007). *Long-Memory Time Series: Theory and Methods*. Wiley Series in Probability and Statistics. Hoboken, NJ: Wiley.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Rogers, L. (2023). *Things We Think We Know*, Chapter 9, pp. 173–184.

Rue, H. and S. Martino (2007). Approximate Bayesian inference for hierarchical Gaussian Markov random field models. *Journal of Statistical Planning and Inference 137*(10), 3177–3192.

Rue, H., S. Martino, and N. Chopin (2009). Approximate Bayesian inference for latent gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society 71*(2), 319–392.

Shephard, N. and K. Sheppard (2010). Realising the future: forecasting with high-frequency-based volatility (heavy) models. *Journal of Applied Econometrics 25*(2), 197–231.

Taylor, S. (1986). *Modelling Financial Time Series*. Hoboken: John Wiley & Sons.

Trefethen, L. N. and D. Bau (1997). *Numerical Linear Algebra*. SIAM.

Van Niekerk, J., E. Krainski, D. Rustand, and H. Rue (2023). A new avenue for bayesian inference with inla. *Computational Statistics & Data Analysis 181*, 107692.

Wood, A. T. A. and G. Chan (1994). Simulation of stationary gaussian processes in $[0, 1]^d$. *Journal of Computational and Graphical Statistics 3*(4), 409–432.