Neural Option Pricing for Rough Bergomi Model

Changqing Teng * and Guanglian Li†

February 6, 2024

Abstract

The rough Bergomi (rBergomi) model can accurately describe the historical and implied volatilities, and has gained much attention in the past few years. However, there are many hidden unknown parameters or even functions in the model. In this work we investigate the potential of learning the forward variance curve in the rBergomi model using a neural SDE. To construct an efficient solver for the neural SDE, we propose a novel numerical scheme for simulating the volatility process using the modified summation of exponentials. Using the Wasserstein 1-distance to define the loss function, we show that the learned forward variance curve is capable of calibrating the price process of the underlying asset and the price of the European style options simultaneously. Several numerical tests are provided to demonstrate its performance.

Keywords: rBergomi model, neural SDE, initial forward variance, Wasserstein 1-distance, summation of exponentials

1 Introduction

Empirical studies of a wild range of assets volatility time-series show that the log-volatility in practice behaves similarly to fractional Brownian motion (FBM)[1] with a Hurst index $H \approx 0.1$ at any reasonable time scale [8]. Motivated by this empirical observation, several rough stochastic volatility models have been proposed, all of which are essentially based on fBM and involve the fractional kernel. The rough Bergomi (rBergomi) model [3] is one recent rough volatility model that has a remarkable capability of fitting both historical and implied volatilities.

The rBergomi model can be formulated as follows. Let S_t be the price of underlying asset with the time horizon $[0, T]$ on a given filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{Q})$,

*Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong. Email: u3553440@connect.hku.hk.

†Corresponding author. Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong. Email: lotusli@maths.hku.hk.

[1]FBM with a Hurst index H is the unique centered Gaussian process $W^H(t)$ with autocorrelation being $\mathbb{E}[W_t^H W_s^H] = \frac{1}{2}(|t|^{2H} + |s|^{2H} - |t-s|^{2H})$.

with $\mathbb{Q}$ being a risk-neutral martingale measure:

$$S_t = S_0 \exp\left(-\frac{1}{2}\int_0^t V_s \mathrm{d}s + \int_0^t \sqrt{V_s}\mathrm{d}Z_s\right), \quad t \in [0,T]. \tag{1}$$

Here, V_t is the spot variance process satisfying

$$V_t = \xi_0(t) \exp\left(\eta\sqrt{2H}\int_0^t (t-s)^{H-\frac{1}{2}}\mathrm{d}W_s - \frac{\eta^2}{2}t^{2H}\right), \quad t \in [0,T]. \tag{2}$$

$S_0 > 0$ denotes the initial value of the underlying asset, and the parameter η is defined by

$$\eta := 2\nu\sqrt{\frac{\Gamma(3/2-H)}{\Gamma(H+1/2)\Gamma(2-2H)}},$$

with ν being the ratio between the increment of $\log V_t$ and the FBM over $(t, t+\Delta t)$ [3, Equation (2.1)].

The Hurst index $H \in (0, 1/2)$ reflects the regularity of the volatility V_t. $\xi_0(\cdot)$ is the so-called initial forward variance curve, defined by $\xi_0(t) = \mathbb{E}^{\mathbb{Q}}[V_t|\mathcal{F}_0] = \mathbb{E}[V_t]$ [3]. Z_t is a standard Brownian motion:

$$Z_t := \rho W_t + \sqrt{1-\rho^2}W_t^{\perp}. \tag{3}$$

Here, $\rho \in (-1, 0)$ is the correlation parameter and $W_t^{\perp}$ is a standard Brownian motion independent of W_t. The price of a European option with payoff function $h(\cdot)$ and expiry T is given by

$$P_0 = \mathbb{E}[\mathrm{e}^{-rT}h(S_T)]. \tag{4}$$

Despite its significance from a modeling perspective, using a singular kernel in the rBergomi model leads to the loss of Markovian and semimartingale structure. In practice, simulating and pricing options using this model involves several challenges. The primary difficulty arises from the singularity of the fractional kernel

$$G(t) := t^{H-\frac{1}{2}}, \tag{5}$$

at $t = 0$, which impacts the simulation of the Volterra process given by:

$$I_t := \sqrt{2H}\int_0^t G(t-s)\mathrm{d}W_s. \tag{6}$$

Note that this Volterra process is a Riemann-Liouville FBM or Lévy's definition of FBM up to a multiplicative constant [18]. The deterministic nature of the kernel $G(\cdot)$ implies that I_t is a centered, locally $(H-\epsilon)$-Hölder continuous Gaussian process with $\mathbb{E}[I_t^2] = t^{2H}$. Furthermore, a straightforward calculation shows

$$\mathbb{E}[I_{t_1}I_{t_2}] = t_1^{2H}C\left(\frac{t_2}{t_1}\right), \quad \text{for } t_2 > t_1,$$

where for $x > 1$, the univariate function $C(\cdot)$ is given by

$$C(x) := 2H \int_0^1 (1-s)^{H-1/2}(x-s)^{H-1/2}\mathrm{d}s,$$

indicating that I_t is non-stationary.

Even though the rBergomi model enjoys remarkable calibration capability with the market data, there are many hidden unknown parameters even functions. For example, the Hurst exponent (in a general SDE) is regarded as an unknown function $H := H(t)\colon \mathbb{R}_+ \to (0,1)$ parameterized by a neural network in [21], which is learned using neural SDEs. According to [3], $\xi_0(t)$ can be any given initial forward variance swap curve consistent with the market price. In view of this fact and the high expressivity of neural SDEs [9, 10, 14, 16, 17, 22, 23], in this work, we propose to parameterize the initial forward variance curve $\xi_0(t)$ using a feedforward neural network:

$$\xi_0(t) = \xi_0(t;\theta), \tag{7}$$

where θ represents the weight parameter vector of the neural network. The training data are generated via a suitable numerical scheme for a given target initial forward variance curve $\xi_0(t)$.

Motivated by the implementation in [7], we adopt the Wasserstein metric as the loss function to train the weights θ. Since the Wasserstein loss is convex and has a unique global minima, any SDE maybe learnt in the infinite data limit [13]. Remarkably, the attained Wasserstein-1 distance during the training is a natural upper bound for the discrepancy between the exact option price P_0 and the learned option price $P_0(\theta^*)$ given by the neural SDE. This in particular implies that if the approximation of the underlying dynamics of stock price S_t is accurate to some level, then the European option price P_0 with this stock as the underlying asset will be accurate to the same level. In this manner, the loss can realize two optimization goals simultaneously.

To generate the training data to learn θ, we need to jointly simulate the Volterra process I_t and the underlying Brownian motion Z_t for the stock price. The non-stationarity feature and the joint simulation requirement make Cholesky factorization [3] the only available exact method. However, it has an $\mathcal{O}(n^3)$ complexity for the Cholesky factorization, $\mathcal{O}(n^2)$ complexity and $\mathcal{O}(n^2)$ storage with n being the total number of time steps. Clearly, the method is infeasible for large n. The most well known method to reduce the computational complexity and storage cost is the hybrid scheme [4]. The hybrid scheme approximates the kernel $G(\cdot)$ by a power function near zero and by a step function elsewhere, which yields an approximation combining the Wiener integrals of the power function with a Riemann sum. Since the fractional kernel $G(\cdot)$ is a power function, it is exact near zero. Its computational complexity is $\mathcal{O}(n \log n)$ and storage cost is $\mathcal{O}(n)$.

In this work, we propose an efficient modified summation of exponentials (mSOE) based method (16) to simulate (1)-(2) (to facilitate the training of the neural SDE). Similar approximations have already been considered, e.g. by Bayer and Breneis [2], and Abi Jaber and El Euch [1]. We further enhance the numerical performance by keeping the kernel exact near the singularity, which achieves high accuracy with the fewest number of summation terms N. Numerical experiments show that the mSOE scheme considerably

improve the prevalent hybrid scheme [4] in terms of accuracy, while having less computational complexity $\mathcal{O}(Nn)$ and storage costs $\mathcal{O}(N)$. Besides the rBergomi model, the proposed approach is applicable to a wide class of stochastic volatility models, especially the rough volatility models with completely monotone kernels.

In sum, the contributions of this work are three-fold. First, we derive an efficient modified sum-of-exponentials (mSOE) based method (16) to solve (1)-(2) even with very small Hurst parameter $H \in (0, 1/2)$. Second, we propose to learn the forward variance curve by the neural SDE using the loss function based on the Wasserstein 1-distance, which can learn the underlying dynamics of the stock price as well as the European option price. Third and last, the mSOE scheme is further utilized to obtain the training data to train our proposed neural SDE model and serves as the solver for the neural SDE, which improves the efficiency significantly.

The remaining of the paper is organized as follows. In Section 2, we introduce an approximation of the singular kernel $G(\cdot)$ by the mSOE, and then describe two approaches for obtaining them. Based on the mSOE, we propose a numerical scheme for simulating the rBergomi model (1)-(2). We introduce in Section 3 the neural SDE model and describe the training of the model. We illustrate in Section 4 the numerical performance of the mSOE scheme. Moreover, several numerical experiments on the performance of our neural SDE model are shown for different target initial forward variance curves. Finally, we summarize the main findings in Section 5.

2 Simulation of the rBergomi model

In this section, we develop an mSOE scheme for efficiently simulating the rBergomi model (1)-(2). Throughout, we consider an equidistant temporal grid $0 = t_0 < t_1 < \cdots < t_n = T$ with a time stepping size $\tau := T/n$ and $t_i := i\tau$.

2.1 Modified SOE based numerical scheme

The non-Markovian nature of the Gaussian process I_t in (6) poses multiple theoretical and numerical challenges. A tractable and flexible Markovian approximation is highly desirable. By the well-known Bernstein's theorem [25], a completely monotone function (i.e., functions that satisfy $(-1)^k g^{(k)}(x) \geq 0$ for all $x > 0$ and $k = 0, 1, 2, \cdots$) can be represented as the Laplace transform of a nonnegative measure. We apply this result to the fractional kernel $G(\cdot)$ and obtain

$$G(t) = \frac{1}{\Gamma(\frac{1}{2} - H)} \int_0^\infty \mathrm{e}^{-xt} x^{-H-\frac{1}{2}} \mathrm{d}x =: \int_0^\infty \mathrm{e}^{-xt} \mu(\mathrm{d}x) \tag{8}$$

where $\mu(\mathrm{d}x) = w(x)\mathrm{d}x$, with $w(x) = \frac{1}{\Gamma(\frac{1}{2}-H)} x^{-H-\frac{1}{2}}$. $\Gamma(\cdot)$ denotes Euler's gamma function, defined by $\Gamma(z) = \int_0^\infty s^{z-1}\mathrm{e}^{-s}\mathrm{d}s$ for $\Re(z) > 0$. That is, $G(t)$ is an infinite mixture of exponentials.

The stochastic Fubini theorem implies

$$I_t = \sqrt{2H}\int_0^t G(t-s)\mathrm{d}W_s = \sqrt{2H}\int_0^\infty \int_0^t \mathrm{e}^{-x(t-s)}\mathrm{d}W_s\mu(\mathrm{d}x)$$
$$=: \sqrt{2H}\int_0^\infty Y_t^x \mu(\mathrm{d}x). \tag{9}$$

Note that for any fixed $x \geq 0$, $(Y_t^x; t \geq 0)$ is an Ornstein-Uhlenbeck process with parameter x, solving the SDE $\mathrm{d}Y_t^x = -xY_t^x\mathrm{d}t + \mathrm{d}W_t$ starting from the origin. Therefore, $I(t)$ is a linear functional of the infinite-dimensional process $\mathcal{Y}_t := (Y_t^x, x \geq 0)$ [6]. We propose to simulate I_t exactly near t and apply numerical quadrature to (8) elsewhere to enhance the computational efficiency, and refer the resulting method to as the modified SOE (mSOE) scheme.

Summation of exponentials can be utilized to approximate the completely monotone functions $g(x)$. This assumption covers most non-negative, non-increasing and smooth functions and is less restrictive than the requirement on the hybrid scheme [20].

Remark 1. For the truncated Brownian semistationary process Y_t of the form $Y_t = \int_0^t g(t-s)\sigma_s\mathrm{d}W_s$, the hybrid scheme requires that the kernel $g(\cdot)$ to satisfy the following two conditions: (a) There exists an $L_g(x) \in C^1((0,1])$ satisfying $\lim_{x\to 0}\frac{L_g(tx)}{L_g(x)} = 1$ for any $t > 0$ and the derivative L_g' satisfying $|L_g'(x)| \leq C(1+x^{-1})$ for $x \in (0,1]$ such that

$$g(x) = x^\alpha L_g(x), \quad x \in (0,1],\ \alpha \in (-0.5, 0.5)\backslash\{0\}.$$

(b) g is differentiable on $(0,\infty)$.

Condition (a) implies that $g(\cdot)$ does not allow for strong singularity near the origin such that it can be well approximated by a certain power function. However, the rough fractional kernel cx^α with $\alpha \in [-1, -0.5]$ fails to satisfy this assumption.

Common examples of completely monotone functions are the exponential kernel $c\mathrm{e}^{-\lambda x}$ with nonnegative parameters c and λ, the rough fractional kernel cx^α with $c \geq 0$ and $\alpha < 0$ and the shifted power law-kernel $(1+t)^\beta$ with $\beta \leq 0$. More flexible kernels can be constructed from these building blocks as complete monotonicity is preserved by summation and products [19, Theorem 1]. The approximation property of the summation of exponentials is guaranteed by the following theorem.

Theorem 2.1 ([5, Theorem 3.4]). *Let $g(\cdot)$ be completely monotone and analytic for $\Re(x) > 0$, and let $0 < a < b$. Then there exists a uniform approximation $\hat{g}(x) := \sum_{j=1}^n \omega_j \mathrm{e}^{-\lambda_j x}$ on the interval $[a,b]$ such that*

$$\lim_{n\to\infty} \|g - \hat{g}\|_\infty^{1/n} \leq \sigma^{-2}.$$

Here, $\sigma = \exp(\frac{\pi\mathcal{K}(k)}{\mathcal{K}'(k)})$, $\mathcal{K}(k) = \mathcal{K}'(k')$ and $\mathcal{K}'(k) = \int_0^\infty \frac{\mathrm{d}t}{\sqrt{(k^2+t^2)(1^2+t^2)}}$, with $k^2 + (k')^2 = 1$.

The proposed mSOE scheme relies on replacing the kernel $G(\cdot)$ by $\hat{G}(\cdot)$, which is de-

fined by

$$\hat{G}(t) := \begin{cases} t^{H-\frac{1}{2}}, & t \in [t_0, t_1), \\ \sum_{j=1}^{N} \omega_j \mathrm{e}^{-\lambda_j t}, & t \in [t_1, t_n], \end{cases} \tag{10}$$

for certain non-negative paired sequence $\{(\omega_j, \lambda_j)\}_{j=1}^N$, where λ_j's are the interpolation points (nodes) and ω_j's are the corresponding weights. This scheme is also referred to as the mSOE-N scheme below. By replacing $G(\cdot)$ with $\hat{G}$ in (5), we derive the associated approximation $\hat{I}(t_i)$ for I_{t_i},

$$\hat{I}(t_i) := \sqrt{2H} \int_{t_{i-1}}^{t_i} (t_i - s)^{H-\frac{1}{2}} \mathrm{d}W_s + \sqrt{2H} \sum_{j=1}^{N} \omega_j \int_0^{t_{i-1}} \mathrm{e}^{-\lambda_j (t_i - s)} \mathrm{d}W_s. \tag{11}$$

Then the Itô isometry implies

$$\begin{aligned} \mathbb{E}\left[|I(t_i) - \hat{I}(t_i)|^2 \right] &= 2H \int_0^{t_{i-1}} |G(t_i - s) - \hat{G}(t_i - s)|^2 \mathrm{d}s \\ &= 2H \int_{t_1}^{t_i} |G(s) - \hat{G}(s)|^2 \mathrm{d}s. \end{aligned} \tag{12}$$

This indicates that we only need to choose non-negative paired sequence $\{(\omega_j, \lambda_j)\}_{j=1}^N$ which minimize $\int_{t_1}^{t_i} |G(s) - \hat{G}(s)|^2 \mathrm{d}s$ in order to obtain a good approximation of $I(t_i)$ in the sense of pointwise mean square error. Next, we describe two known approaches to obtain the non-negative paired sequence $\{(\omega_j, \lambda_j)\}_{j=1}^N$ in (10). Both are essentially based on Gauss quadrature.

Summation of exponentials: approach A

Approach A in [2] applies m-point Gauss-Jacobi quadrature with weight function $x^{-H-\frac{1}{2}}$ to n geometrically spaced intervals $[\zeta_i, \zeta_{i+1}]_{i=0,\cdots,n-1}$ and uses a Riemann-type approximation on the interval $[0, \zeta_0]$. The resulting approximator $\hat{G}_A(t)$ with $N+1$ number of summations is given by

$$\hat{G}_A(t) := \sum_{j=0}^{N} \omega_j \mathrm{e}^{-\lambda_j t} \quad t \in [t_1, t_n],$$

where $\lambda_0 = 0$ and $\omega_0 = \frac{1}{\Gamma(\frac{1}{2}-H)} \int_0^{\zeta_0} x^{-H-\frac{1}{2}} \mathrm{d}x$. Let $N := nm$ be the prespecified total number of nodes. The parameters m, n and ζ_i are computed based on a set of parameters $(\alpha, \beta, a, b) \in (0, \infty)$ as follows.

$$m = \left\lceil \frac{\beta}{A} \sqrt{N} \right\rceil, \quad n = \left\lfloor \frac{A}{\beta} \sqrt{N} \right\rfloor \, (mn \approx N),$$

$$
\begin{aligned}
&A = \left(\frac{1}{H} + \frac{1}{3/2 - H}\right)^{1/2},\\
&\zeta_0 = a \exp\left(-\frac{\alpha}{(3/2-H)A}\sqrt{N}\right), \quad \zeta_n = b \exp\left(\frac{\alpha}{HA}\sqrt{N}\right),\\
&\zeta_i = \zeta_0 \left(\frac{\zeta_n}{\zeta_0}\right)^{i/n}, \quad i = 0, \cdots, n.
\end{aligned}
$$

The following lemma gives the approximation error of the above Gaussian quadrature, which is a modification of [2, lemmas 2.8 and 2.9].

Lemma 2.1. *Let $(\omega_j)_{j=1}^{nm}$ be the weights and $(\lambda_j)_{j=1}^{nm}$ be the nodes of Gaussian quadrature on the intervals $[\zeta_i, \zeta_{i+1}]_{i=0,\cdots,n-1}$ computed on the set of parameters $(\alpha, \beta, 1, 1)$. Then the following error estimates hold*

$$
\begin{aligned}
\left|\int_0^{\zeta_0} \mathrm{e}^{-xt}\mu(\mathrm{d}x) - \omega_0\right| &\le \frac{t}{\Gamma(\frac{1}{2}-H)(\frac{3}{2}-H)} \exp\left(-\frac{\alpha}{A}\sqrt{N}\right),\\
\left|\int_{\zeta_0}^{\zeta_n} \mathrm{e}^{-xt}\mu(\mathrm{d}x) - \sum_{j=1}^{nm} \omega_j \mathrm{e}^{-\lambda_j t}\right| &\le \sqrt{\frac{5\pi^3}{18}} \frac{n\left(\mathrm{e}^{\alpha\beta}-1\right) t^{H-\frac{1}{2}}}{\Gamma(\frac{1}{2}-H) 2^{2m+1} m^H} \exp\left(\frac{2\beta}{A}\sqrt{N}\log\left(\mathrm{e}^{\alpha\beta}-1\right)\right),\\
\left|\int_{\zeta_n}^{\infty} \mathrm{e}^{-xt}\mu(\mathrm{d}x)\right| &\le \frac{1}{t\Gamma(\frac{1}{2}-H){\zeta_n}^{H+\frac{1}{2}}} \exp(-\zeta_n t).
\end{aligned}
$$

Summation of exponentials: approach B

Approach B in [11] approximates the kernel $G(t)$ efficiently on the interval $[t_1, t_n]$ with the desired precision $\epsilon > 0$. It applies n_0-point Gauss-Jacobi quadrature on the interval $[0, 2^{-M}]$ with the weight function $x^{-H-\frac{1}{2}}$ where $n_0 = \mathcal{O}(\log\frac{1}{\epsilon})$, $M = \mathcal{O}(\log T)$; n_s-point Gauss-Legendre quadrature on M small intervals $[2^j, 2^{j+1}]$, $j = -M, \cdots, -1$, where $n_s = \mathcal{O}(\log\frac{1}{\epsilon})$, and n_l-point Gauss-Legendre quadrature on $N+1$ large intervals $[2^j, 2^{j+1}]$, $j = 0, \cdots, N$, where $n_l = \mathcal{O}(\log\frac{1}{\epsilon} + \log\frac{1}{\tau})$, $N = \mathcal{O}(\log\log\frac{1}{\epsilon} + \log\frac{1}{\tau})$ [11, Theorem 2.1]. The resulting approximation $\hat{G}_B(t)$ reads,

$$
\hat{G}_B(t) = \sum_{k=1}^{n_0} \omega_{0,k}\mathrm{e}^{-\lambda_{0,k}t} + \sum_{j=-M}^{-1}\sum_{k=1}^{n_s} \omega_{j,k}\mathrm{e}^{-\lambda_{j,k}t}\lambda_{j,k}^{-H-\frac{1}{2}} + \sum_{j=0}^{N}\sum_{k=1}^{n_l} \omega_{j,k}\mathrm{e}^{-\lambda_{j,k}t}\lambda_{j,k}^{-H-\frac{1}{2}} \quad t \in [t_1, t_n], \tag{13}
$$

where the $\lambda_{j,k}^{-H-\frac{1}{2}}$ terms could be absorbed into the corresponding $\omega_{j,k}$s so that the approximation is in the form of (10). There holds $|G(t) - \hat{G}_B(t)| \le \epsilon$. For further optimization, a modified Prony's method is applied on the interval $(0, 1)$ and standard model reduction method is applied on $[1, 2^{N+1}]$ to reduce the number of exponentials needed.

The approximation error is given below, which is a slight modification of [11, Lemmas 2.2, 2.3 and 2.4].

Table 1: Parameters for mSOE scheme based on approach B with $H = 0.07$, $\epsilon = 0.0008$ and $N = 20$.

j	ω_j	λ_j	j	ω_j	λ_j
1	0.26118	0.47726	11	1.28121	1.92749
2	0.19002	0.22777	12	1.76098	40.38675
3	0.13840	0.108690	13	2.42043	84.62266
4	0.10717	5.11098×10^{-2}	14	3.32681	177.31051
5	0.11366	1.93668×10^{-2}	15	4.57262	371.52005
6	0.14757	2.04153×10^{-3}	16	6.28495	778.44877
7	0.35898	1	17	8.63850	1631.08960
8	0.49341	2.09531	18	11.87339	3417.63439
9	0.67818	4.390310	19	16.31967	7160.99521
10	0.93214	9.19906	20	22.43096	15004.4875

Lemma 2.2. *Let $a = 2^{-M}$, $p = 2^{N+1}$ and follow the settings in* (13)*. Then the following error estimates hold*

$$\left| \int_0^a \mathrm{e}^{-xt} \mu(\mathrm{d}x) - \sum_{k=1}^{n_0} w_{0,k} \mathrm{e}^{-\lambda_{0,k} t} \right| \leq \frac{4\sqrt{\pi}}{\Gamma(\frac{1}{2} - H)} a^{\frac{1}{2} - H} n_0^{\frac{3}{2}} \left(\frac{\mathrm{e} a T}{8(n_0 - 1)} \right)^{2n_0},$$

$$\begin{aligned} &\left| \int_a^p \mathrm{e}^{-xt} \mu(\mathrm{d}x) - \sum_{j=-M}^{-1} \sum_{k=1}^{n_s} \omega_{j,k} \mathrm{e}^{-\lambda_{j,k} t} \lambda_{j,k}^{-H-\frac{1}{2}} - \sum_{j=0}^{N} \sum_{k=1}^{n_l} \omega_{j,k} \mathrm{e}^{-\lambda_{j,k} t} \lambda_{j,k}^{-H-\frac{1}{2}} \right| \\ &\quad \leq \frac{2^{\frac{3}{2}} \pi}{\Gamma(\frac{1}{2} - H)} \left(\frac{\mathrm{e}^{\frac{1}{\mathrm{e}}}}{4} \right)^{\max(n_s, n_l)} \frac{2^{(\frac{1}{2} - H)(N+1)} - 2^{-(\frac{1}{2} - H)M}}{2^{\frac{1}{2} - H} - 1}, \end{aligned}$$

$$\left| \int_p^\infty \mathrm{e}^{-xt} \mu(\mathrm{d}x) \right| \leq \frac{1}{\tau \Gamma(\frac{1}{2} - H) p^{H + \frac{1}{2}}} \mathrm{e}^{-\tau p}.$$

We shall compare Approaches A and B in Section 4, the result shows that Approach B outperforms Approach A. We present in Table 1 the parameter pairs for $H = 0.07$ and $N = 20$. See [12, Section 3.4] for further discussions about SOE approximations for the fractional kernel $G(t)$.

2.2 Numerical method based on mSOE scheme

Next we describe a numerical method to simulate (S_{t_i}, V_{t_i}) for $i = 1, \cdots, n$ based on the mSOE scheme (11). Recall that the integral (6) can be written into a summation of the local part and the history part

$$\begin{aligned} I_{t_{i+1}} &= \sqrt{2H} \int_0^{t_i} (t_{i+1} - s)^{H - \frac{1}{2}} \mathrm{d}W_s + \sqrt{2H} \int_{t_i}^{t_{i+1}} (t_{i+1} - s)^{H - \frac{1}{2}} \mathrm{d}W_s \\ &=: I_{\mathcal{F}}(t_{i+1}) + I_{\mathcal{N}}(t_{i+1}). \end{aligned} \tag{14}$$

The local part $I_{\mathcal{N}}(t_{i+1}) \sim \mathcal{N}(0, \tau^{2H})$ can be simulated exactly. The history part $I_{\mathcal{F}}(t_{i+1})$ is approximated by replacing the kernel $G(\cdot)$ by the mSOE $\widehat{G}(t)$ (by approach B):

$$\bar{I}_{\mathcal{F}}(t_{i+1}) = \sqrt{2H}\sum_{j=1}^{N}\omega_j \int_0^{t_i} \mathrm{e}^{-\lambda_j(t_{i+1}-s)}\mathrm{d}W_s =: \sqrt{2H}\sum_{j=1}^{N}\omega_j \bar{I}^j_{\mathcal{F}}(t_{i+1}).$$

Then direct computation leads to

$$\begin{aligned}
\bar{I}^j_{\mathcal{F}}(t_{i+1}) &= \mathrm{e}^{-\lambda_j\tau}\int_0^{t_i}\mathrm{e}^{-\lambda_j(t_i-s)}\mathrm{d}W_s \\
&= \mathrm{e}^{-\lambda_j\tau}\left(\int_0^{t_{i-1}}\mathrm{e}^{-\lambda_j(t_i-s)}\mathrm{d}W_s + \int_{t_{i-1}}^{t_i}\mathrm{e}^{-\lambda_j(t_i-s)}\mathrm{d}W_s\right) \\
&= \mathrm{e}^{-\lambda_j\tau}\left(\bar{I}^j_{\mathcal{F}}(t_i) + \int_{t_{i-1}}^{t_i}\mathrm{e}^{-\lambda_j(t_i-s)}\mathrm{d}W_s\right).
\end{aligned}$$

Consequently, we obtain the recurrent formula for each history component

$$\bar{I}^j_{\mathcal{F}}(t_i) = \begin{cases} 0 & i = 1, \\ \mathrm{e}^{-\lambda_j\tau}\left(\bar{I}^j_{\mathcal{F}}(t_{i-1}) + \int_{t_{i-2}}^{t_{i-1}}\mathrm{e}^{-\lambda_j(t_{i-1}-s)}\mathrm{d}W_s\right) & i > 1. \end{cases} \tag{15}$$

This, together with (14), implies that we need to simulate a centered $(N+2)$-dimensional Gaussian random vector at t_i,

$$\Theta_i := \left(\Delta W_{t_i}, \int_{t_{i-1}}^{t_i}\mathrm{e}^{-\lambda_1(t_i-s)}\mathrm{d}W_s, \cdots, \int_{t_{i-1}}^{t_i}\mathrm{e}^{-\lambda_N(t_i-s)}\mathrm{d}W_s, I_{\mathcal{N}}(t_i)\right) \quad \text{for } i = 1, \cdots, n.$$

Here, $\Delta W_{t_i} =: W_{t_i} - W_{t_{i-1}}$ denotes the increment. Note that Θ_i is determined by its covariance matrix Σ, which is defined

$$\begin{aligned}
&\Sigma_{1,1} = \tau, \quad \Sigma_{1,l} = \Sigma_{l,1} = \frac{1}{\lambda_{l-1}}\left(1-\mathrm{e}^{-\lambda_{l-1}\tau}\right), \quad \Sigma_{k,l} = \frac{1}{\lambda_{k-1}+\lambda_{l-1}}\left(1-\mathrm{e}^{-(\lambda_{k-1}+\lambda_{l-1})\tau}\right) \\
&\Sigma_{N+2,1} = \Sigma_{1,N+2} = \frac{\sqrt{2H}}{H+1/2}\tau^{H+\frac{1}{2}} \\
&\Sigma_{N+2,l} = \Sigma_{l,N+2} = \frac{\sqrt{2H}}{\lambda_{l-1}^{H+1/2}}\gamma(H+\tfrac{1}{2}, \lambda_{l-1}\tau) \\
&\Sigma_{N+2,N+2} = \tau^{2H}
\end{aligned}$$

for $k, l = 2, \cdots, N+1$, where $\gamma(\cdot,\cdot)$ refers to the lower incomplete gamma function. We only need to inplement the Cholesky decomposition once since Σ is independent of i.

Finally, we present the two-step numerical scheme for $(S_{t_{i+1}}, V_{t_{i+1}})$ for $i = 0, \cdots, n-1$,

$$\begin{aligned}
\bar{S}_{t_{i+1}} &= \bar{S}_{t_i} \exp\left(\sqrt{\bar{V}_{t_i}}\left(\rho \Delta W_{t_{i+1}} + \sqrt{1-\rho^2}\Delta W^{\perp}_{t_{i+1}}\right) - \frac{\tau}{2}\bar{V}_{t_i}\right), \\
\bar{V}_{t_{i+1}} &= \xi_0(t_{i+1}) \exp\left(\eta\left(\bar{I}_{\mathcal{F}}(t_{i+1}) + I_{\mathcal{N}}(t_{i+1})\right) - \frac{\eta^2}{2} t_{i+1}^2\right), \\
\bar{I}_{\mathcal{F}}(t_{i+1}) &= \sqrt{2H}\sum_{j=1}^{N} \omega_j \bar{I}^j_{\mathcal{F}}(t_{i+1}), \\
\bar{I}^j_{\mathcal{F}}(t_{i+1}) &= \mathrm{e}^{-\lambda_j \tau}\left(\bar{I}^j_{\mathcal{F}}(t_i) + \int_{t_{i-1}}^{t_i} \mathrm{e}^{-\lambda_j(t_i - s)} \mathrm{d}W_s\right).
\end{aligned} \tag{16}$$

To simulate $\{\bar{I}_{\mathcal{F}}(t_{i+1})\}_{i=1}^{n-1}$, this mSOE-N based simulation scheme (16) only requires $\mathcal{O}(N^3)$ offline computation complexity which accounts for the Cholesky decomposition of the covariance matrix Σ, $\mathcal{O}(Nn)$ computation complexity and $\mathcal{O}(N)$ storage.

3 Learning the forward variance curve

This section is concerned with learning the forward variance curve $\xi_0(t;\theta)$ with θ being the weight parameters from the feedforward neural network (7) by Neural SDE. Without loss of generality, assuming $S_0 = 1$, we can rewrite (1)-(2) as

$$\begin{cases}
S_t = 1 + \int_0^t S_s \exp(X_s) \mathrm{d}Z_s, \\
X_t = \frac{1}{2}\log(V_t).
\end{cases} \tag{17}$$

Upon parameterizing the forward variance curve by (7), the dynamics of the price process and variance process are also parameterized accordingly, which is given by the following neural SDE

$$\begin{cases}
S_t(\theta) = 1 + \int_0^t S_s(\theta) \exp(X_s(\theta)) \mathrm{d}Z_s, \\
X_t(\theta) = \frac{1}{2}\log(V_t(\theta)), \\
V_t(\theta) = \xi_0(t;\theta) \exp\left(\eta I(t) - \frac{\eta^2}{2} t^{2H}\right).
\end{cases} \tag{18}$$

Then we propose to use the Wasserstein-1 distance as the loss function to train this neural SDE. That is, the learned weight parameters θ^* are given by

$$\theta^* := \arg\min_{\theta} W_1\left(S_T, S_T(\theta)\right). \tag{19}$$

Next, we recall the Wasserstein distance. Let (M, d) be a Radon space. The Wasserstein-p distance for $p \in [1, \infty)$ between two probability measures μ and ν on M with finite

p-moments is defined by

$$W_p(\mu,\nu) = \left(\inf_{\gamma\in\Gamma(\mu,\nu)} \mathbb{E}_{(x,y)\sim\gamma} d(x,y)^p\right)^{1/p},$$

where $\Gamma(\mu,\nu)$ is the set of all couplings of μ and ν. A coupling γ is a joint probability measure on $M\times M$ whose marginals are μ and ν on the first and second factors, respectively. If μ and ν are real-valued, then their Wasserstein distance can be simply computed by utilising their cumulative distribution functions [24]:

$$W_p(\mu,\nu) = \left(\int_0^1 |F^{-1}(z) - G^{-1}(z)|^p \mathrm{d}z\right)^{1/p}.$$

In particular, when $p=1$, according to the Kantorovich-Robenstein duality [24], Wasserstein-1 distance can be represented by

$$W_1(\mu,\nu) = \sup_{\mathrm{Lip}(f)\le 1} \mathbb{E}_{x\sim\mu}[f(x)] - \mathbb{E}_{y\sim\nu}[f(y)], \tag{20}$$

where $\mathrm{Lip}(f)$ denotes the Lipschitz constant of f.

In the experiment, we work with the empirical distributions on $\mathbb{R}$. If ξ and η are two empirical measures with m samples $X_1,\cdots,X_m$ and $Y_1,\cdots,Y_m$, then the Wasserstein-p distance is given by

$$W_p(\xi,\eta) = \left(\frac{1}{m}\sum_{i=1}^m |X_{(i)} - Y_{(i)}|^p\right)^{1/p}, \tag{21}$$

where $X_{(i)}$ is the ith smallest value among the m samples. With $S_t(\theta^*)$ being the price process of the underlying asset after training, the price $P_0(\theta^*)$ of a European option with payoff function $h(x)$ and expiry T is given by

$$P_0(\theta^*) = \mathbb{E}[\mathrm{e}^{-rT} h(S_T(\theta^*))]. \tag{22}$$

Since the payoff function $h(\cdot)$ is clearly Lipschitz-1 continuous, according to (20), Wasserstein-1 distance is a natural upper bound for the pricing error of the rBergomi model. Thus, the choice of the training loss is highly desirable.

Proposition 3.1. *Set the interest rate $r=0$, the Wasserstein-1 distance is a natural upper bound for the pricing error of the rBergomi model:*

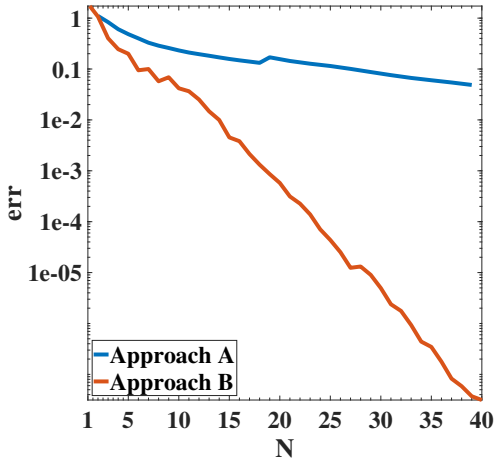
$$|P_0 - P_0^*| = |\mathbb{E}\left[h(S_T)\right] - \mathbb{E}\left[h(S_T(\theta^*))\right]| \le W_1(S_T, S_T(\theta^*)).$$

4 Numerical tests

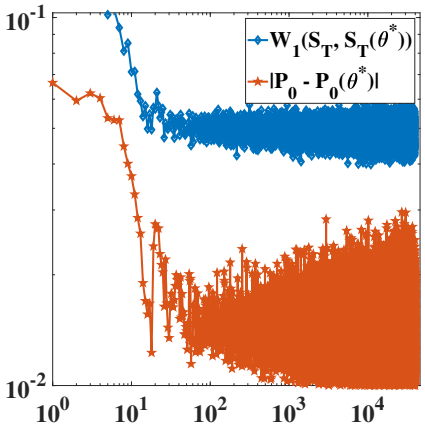
In this section, we illustrate the performance of the proposed mSOE scheme (16) for simulating (1)-(2), and demonstrate the learning of the forward variance curve (7) using neural SDE (18).

4.1 Numerical test for mSOE scheme (16)

First we compare approaches A and B. They only differ in the choice of nodes λ_j and weights ω_j. To compare their performance in the simulation of the stochastic Volterra process $I(t)$, one only needs to estimate the discrepancy between G and $\hat{G}$ in $L^2(t_1, T)$-norm (12). Thus, as the criterion for comparison, we use err $:= (\int_{t_1}^{T}(G(t) - \hat{G}(t))^2 \mathrm{d}t)^{\frac{1}{2}}$. We show in Figure 1(a) the err versus the number of nodes N for approaches A and B with $H = 0.07$. We observe that approach B has a better error decay. This behavior is also observed for other $H \in (0, 1/2)$. Thus, we implement approach B in (10), which involves $\mathcal{O}(N^3)$ computational cost to obtain the non-negative tuples $\{(\omega_j, \lambda_j)\}_{j=1}^{N}$.



(a) mSOE-N approximation error



(b) training dynamics

Figure 1: (a) The performance of the mSOE-N schemes based on approaches A and B with $\tau = 0.0005$, $T = 1$ and $H = 0.07$, and (b) the training dynamics.

Then we calculate the implied volatility curves using the mSOE-N based numerical schemes with $N = 2, 4, 8, 10, 32$ using the parameters listed in Table 2. To ensure that the temporal discretization error and the Monte Carlo simulation error are negligible, we take the number of temporal steps $n = 2000$ and the number of samples $M = 10^6$. The resulting implied volatility curves are shown in Figure 2, which are solved via the Newton-Raphson method. We compare the proposed mSOE scheme with the exact method, i.e., Cholesky decomposition, and the hybrid scheme. We observe that mSOE-4 scheme can already accurately approximate the implied volatility curves given by the exact method. This clearly shows the high efficiency of the proposed mSOE based scheme for simulating the implied volatility curves.

Table 2: Parameter values used in the rBergomi model.

S_0	ξ_0	η	H	ρ
1	0.235^2	1.9	0.07	-0.9

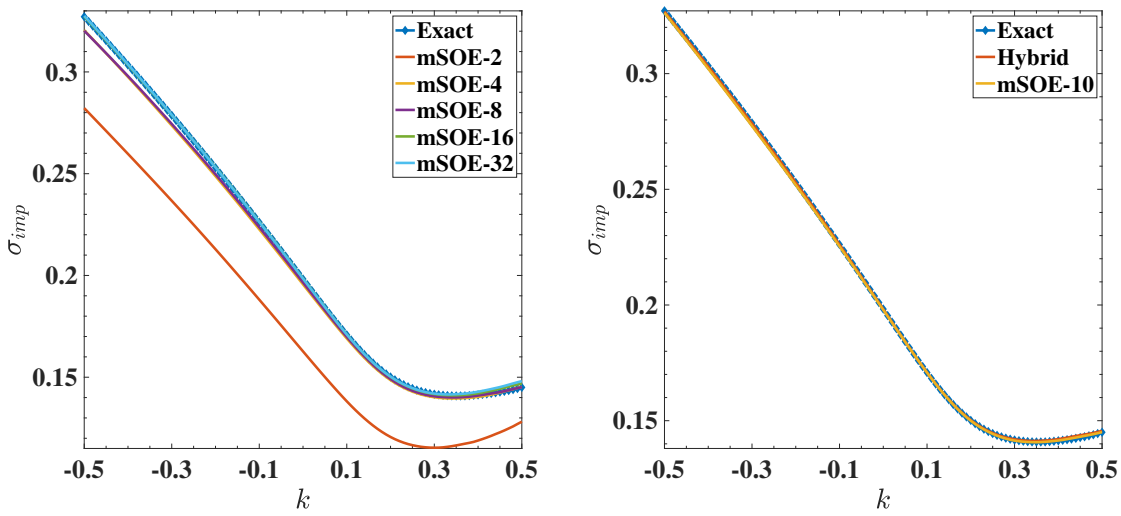


Figure 2: The implied volatility curves σ_{imp} computed by the mSOE-N based numerical scheme (16) with different number of summation terms N. In the plot, $k = \log(K/S_0)$ is the log-moneyness and K is the strike price.

Next, we depict in Figure 3 the root mean squared error (RMSE) of the first moment and second moment of the following Gaussian random variable using the same set of parameters,

$$\mathcal{G}(t) := \eta I_t - \frac{\eta^2}{2} t^{2H} \sim \mathcal{N}\left(-\frac{\eta^2}{2} t^{2H}, \eta^2 t^{2H}\right).$$

The RMSEs are defined by

$$\text{RMSE 1st moment} = \left(\sum_{i=1}^{n} \left(\mathbb{E}\left[\hat{\mathcal{G}}(t_i)\right] + \frac{\eta^2}{2} t_i^{2H}\right)^2\right)^{1/2},$$

$$\text{RMSE 2nd moment} = \left(\sum_{i=1}^{n} \left(\mathbb{E}\left[\hat{\mathcal{G}}(t_i)^2\right] - \left(\frac{\eta^4 t_i^{4H}}{4} + \eta^2 t_i^{2H}\right)\right)^2\right)^{1/2},$$

where $\hat{\mathcal{G}}(t)$ is the approximation of $\mathcal{G}(t)$ under the mSOE-N scheme or the hybrid scheme. These quantities characterize the fundamental statistical feature of the distribution. We observe that the mSOE-N based numerical scheme achieves high accuracy for the number of terms $N > 15$, again clearly showing the high efficiency of the proposed scheme.

4.2 Numerical performance for learning the forward variance curve

Now we validate (3.1) using three examples of ground truth forward variance curves:

$$\xi_0(t) \equiv 0.235^2, \tag{23a}$$

$$\xi_0(t) = 2|W_t|, \tag{23b}$$

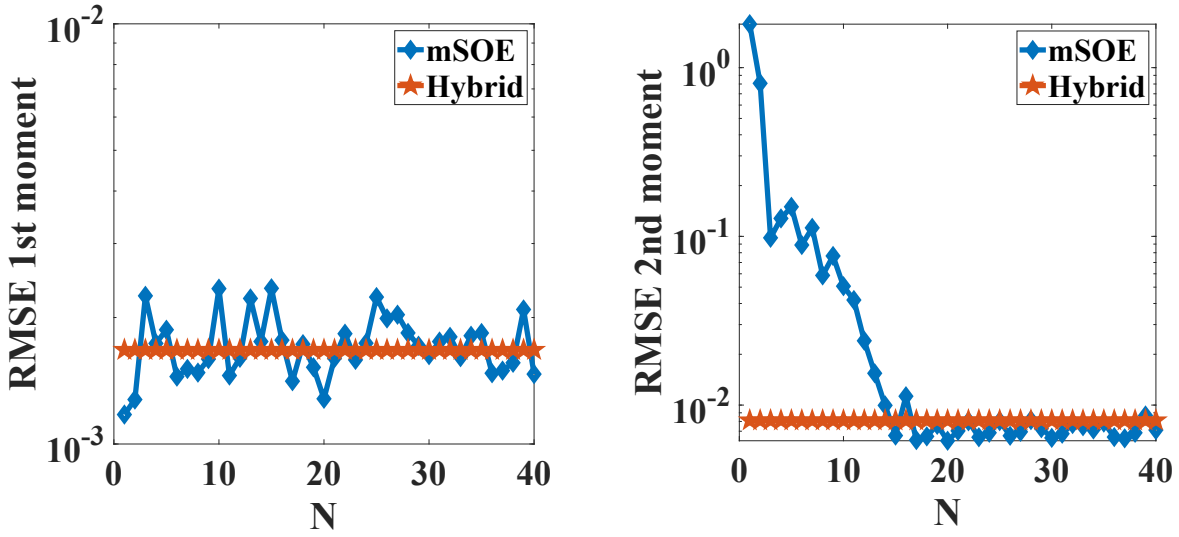


Figure 3: The RMSEs for the first and second moments generated by the mSOE-N based numerical scheme (16) and the hybrid scheme.

$$\xi_0(t) = 0.1|W_t^H| \text{ with } H = 0.07. \tag{23c}$$

The first example (23a) corresponds to a constant forward variance curve. The second example (23b) takes a scaled sample path of Brownian motion as the forward variance curve, and the third example (23c) utilises a scaled path of fractional Brownian motion as the forward variance curve.

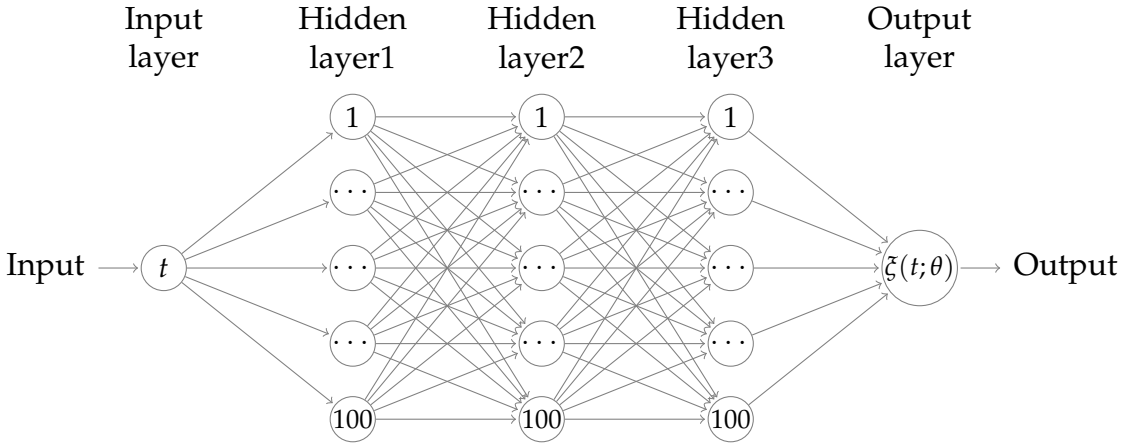


Figure 4: Architecture: 3-layer feedforward neural network.

The initial forward variance curve ξ_0 is parameterized by a feed forward neural network, cf. (7), which has 3 hidden layers, width 100, and leaky ReLU activations; see Fig. 4 for a schematic illustration. The weights of each model were carefully initialized in order to prevent gradient vanishing.

To generate the training data, we use the stock price samples generated by the scheme (16). The total number of samples used in the experiments is 10^5. Training of the neural

network was performed on the first 81.92% of the dataset and the model's performance was evaluated on the remaining 18.08% of the dataset. Each sample is of length 2000, which is the number of discretized time intervals. The batch size was 4096, which was picked as large as possible that the GPU memory allowed for. Each model was trained for 100 epochs. We consider Wasserstein-1 distance as loss metric by applying (21) to the empirical distributions on batches of data at the final time grid of the samples.

The neural SDEs were trained using Adam [15]. The learning rate was taken to be 10^{-4}, which was then gradually reduced until a good performance was achieved. The training was performed on a Linux server and a NVIDIA RTX6000 GPU and takes a few hours for each experiment. $V_t(\theta)$ in (18) was solved by the mSOE scheme to reduce the computational complexity, then $S_t(\theta)$ was solved using the Euler-Maruyama method. The example code for dataset sampling and network training will be made available at the Github repository `https://github.com/evergreen1002/Neural-option-pricing-for-rBergomi-model`.

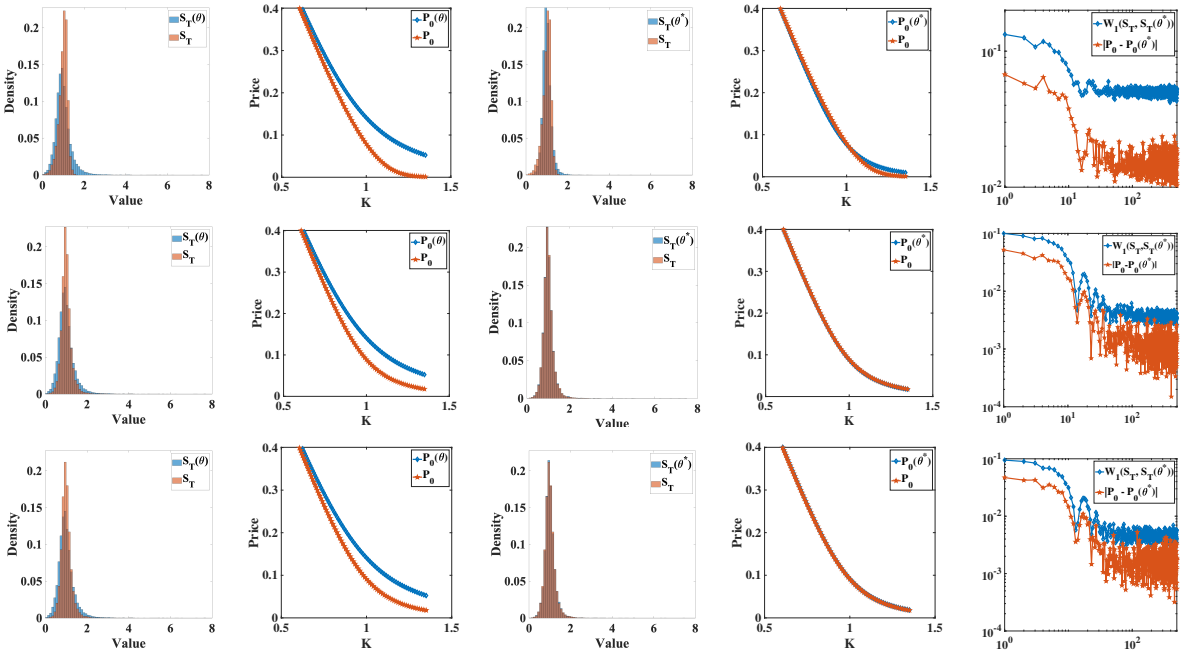


Figure 5: Left to right: Empirical distribution of terminal stock price before the training, option price before the training, empirical distribution of terminal stock price after the training, option price after training, learning curve.

Now we present the numerical results for the learned forward variance curves in Figure 5. These three rows correspond to three different initial forward variance curves defined in (23a), (23b) and (23c) used for neural network training, respectively. The first two columns display the empirical distributions $S_T(\theta)$ and the corresponding European call option prices $P_0(\theta)$ for the test set against a number of strikes before training has begun. The next two columns display $S_T(\theta^*)$ and $P_0(\theta^*)$ after training. It is observed that $S_T(\theta^*)$ and $P_0(\theta^*)$ closely match the exact one on the test set, thereby showing the high accuracy of learning the rBergomi model. The last column shows the Wasserstein-1 distance at time T compared to the maximum error in the option price over each batch during training. These two quantities are precisely the subjects in (3.1). Only the first 500 iterations

were plotted. In all cases, the training loss can be reduced to an acceptable level, so is the error of pricing, clearly indicating the feasibility of learning within the rBergomi model. Finally, we investigate whether the training loss can be further reduced after more iterations for the first example (23a). Figure 1(b) shows the training loss against the number of iterations. Gradient descent is applied with learning rate 10^{-5}. We observe that the training loss is oscillating near 0.05 even after 100,000 iterations. Indeed, the oscillation of the loss values aggravates as the iteration further proceeds, indicating the necessity for early stopping of the training process.

5 Conclusion

In this work, we have proposed a novel neural network based SDE model to learn the forward variance curve for the rough Bergomi model. We propose a new modified summation of exponentials (mSOE) scheme to improve the efficiency of generating training data and facilitating the training process. We have utilized the Wasserstein 1-distance as the loss function to calibrate the dynamics of the underlying assets and the price of the European options simultaneously. Furthermore, several numerical experiments are provided to demonstrate their performances, which clearly show the feasibility of neural networks for calibrating these models. Future work includes learning all the unknown functions in the rBergomi model (as well as other rough volatility models) by the proposed approach, using the market data instead of the simulated data as in the present work.

Acknowledgements

GL acknowledges the support from GRF (project number: 17317122) and Early Career Scheme (Project number: 27301921), RGC, Hong Kong.

References

[1] E. Abi Jaber and O. El Euch. Multifactor approximation of rough volatility models. *SIAM Journal on Financial Mathematics*, 10(2):309–349, 2019.

[2] C. Bayer and S. Breneis. Markovian approximations of stochastic Volterra equations with the fractional kernel. *Quantitative Finance*, 23(1):53–70, 2023.

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

[4] M. Bennedsen, A. Lunde, and M. S. Pakkanen. Hybrid scheme for Brownian semistationary processes. *Finance and Stochastics*, 21:931–965, 2017.

[5] D. Braess. *Nonlinear approximation theory*. Springer Science & Business Media, 2012.

[6] L. Coutin and P. Carmona. Fractional Brownian motion and the Markov property. *Electronic Communications in Probability*, 3:12, 1998.

[7] T. DeLise. Neural options pricing. Preprint, arXiv:2105.13320, 2021.

[8] J. Gatheral, T. Jaisson, and M. Rosenbaum. Volatility is rough. In *Commodities*, pages 659–690. Chapman and Hall/CRC, 2022.

[9] L. Hodgkinson, C. van der Heide, F. Roosta, and M. W. Mahoney. Stochastic normalizing flows. *arXiv preprint arXiv:2002.09547*, 2020.

[10] J. Jia and A. R. Benson. Neural jump stochastic differential equations. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[11] S. Jiang, J. Zhang, Q. Zhang, and Z. Zhang. Fast evaluation of the Caputo fractional derivative and its applications to fractional diffusion equations. *Communications in Computational Physics*, 21(3):650–678, 2017.

[12] B. Jin and Z. Zhou. *Numerical Treatment and Analysis of Time-Fractional Evolution Equations*, volume 214 of *Applied Mathematical Sciences*. Springer, Cham, 2023.

[13] P. Kidger, J. Foster, X. Li, and T. J. Lyons. Neural SDEs as infinite-dimensional GANs. In *International conference on machine learning*, pages 5453–5463. PMLR, 2021.

[14] P. Kidger, J. Foster, X. C. Li, and T. Lyons. Efficient and accurate gradients for neural SDEs. In *Advances in Neural Information Processing Systems*, volume 34, pages 18747–18761, 2021.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*, San Diego, 2015.

[16] X. Li, T.-K. L. Wong, R. T. Chen, and D. Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pages 3870–3882. PMLR, 2020.

[17] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, and C.-J. Hsieh. Neural SDE: Stabilizing neural ODE networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

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

[19] K. S. Miller and S. G. Samko. Completely monotonic functions. *Integral Transforms and Special Functions*, 12(4):389–402, 2001.

[20] S. E. Rømer. Hybrid multifactor scheme for stochastic volterra equations with completely monotone kernels. *Available at SSRN 3706253*, 2022.

[21] A. Tong, T. Nguyen-Tang, T. Tran, and J. Choi. Learning fractional white noises in neural stochastic differential equations. In *Advances in Neural Information Processing Systems*, volume 35, pages 37660–37675, 2022.

[22] B. Tzen and M. Raginsky. Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. arXiv preprint arXiv:1905.09883, 2019.

[23] B. Tzen and M. Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Conference on Learning Theory*, pages 3084–3114. PMLR, 2019.

[24] C. Villani. *Topics in optimal transportation*. American Mathematical Soc., Providence, 2021.

[25] D. V. Widder. *The Laplace Transform*, volume vol. 6 of *Princeton Mathematical Series*. Princeton University Press, Princeton, NJ, 1941.